



OCIENT HYPERSCALE DATA WAREHOUSE

User Documentation
Version 25.2

Contents

1	Get Started with the Ocient Hyperscale Data Warehouse	4
1.1	Key Concepts	5
2	Connect to Ocient	9
2.1	Ocient Integrations	9
2.2	Connect Using JDBC	14
2.3	Connect Using pyocient	15
3	Load Data	18
3.1	Load by Data Source	22
3.2	Data Types for Data Pipelines	32
3.3	Data Formats for Data Pipelines	35
3.4	Transform Data in Data Pipelines	55
3.5	Load Geospatial Data in Data Pipelines	92
3.6	Load Metadata Values in Data Pipelines	94
3.7	Deduplication in Data Pipelines	97
3.8	Error Tolerance in Data Pipelines	99
3.9	Monitor Data Pipelines	100
3.10	Manage Errors in Data Pipelines	106
3.11	Frequently Asked Questions for Data Pipelines	110
4	Query Ocient	116
4.1	Quick Start Querying	116
4.2	General SQL Syntax	117
4.3	Ocient HTTP Query API	153
4.4	Understanding Data Types	161
4.5	Functions Overview	166
5	Analysis in Ocient	168
5.1	Text Analysis in Database Queries	168
5.2	Machine Learning in Ocient	168
6	Database Administration	210
6.1	Ocient Security Guide	211
6.2	Manage Users, Groups, and Roles	217
6.3	Authentication Methods	231
6.4	Schema Design	239
6.5	Manage Distributed Tasks	259
6.6	Remove Records from an Ocient System	261
7	System Administration	263
7.1	Install an Ocient System	264
7.2	Amazon Web Services Ocient Installation	289
7.3	Configure Storage Spaces	295
7.4	Ocient Software Upgrade	298
7.5	Maintenance Overview	299
7.6	Monitoring	329
7.7	System Configuration	355
7.8	Secure Connections Using TLS	361
7.9	Multiple Storage Clusters for Loading Data	362
7.10	Loading Administration	362
7.11	Air-Gapped Environment with an Ocient System	369
8	Ocient Simulator	374

9	Ocient Architecture	384
9.1	Core Elements of an Ocient System	384
9.2	Compute-Adjacent Input and Output on Large Working Sets	388
9.3	Query Performance Optimizations	392
9.4	Plan Optimization	396
9.5	Exabyte Scalability Design	397
9.6	Loading Characteristics and Concepts	400
10	SQL Reference	403
10.1	Alphabetical SQL Functions List	404
10.2	Categorized SQL Functions List	426
10.3	Aggregate Functions	449
10.4	Array Functions and Operators	450
10.5	Character and Binary Functions	459
10.6	Data Types	482
10.7	Date and Time Functions	486
10.8	Formatting Functions	503
10.9	HyperLogLog Functions	507
10.10	Geospatial Functions	513
10.11	Machine Learning Model Functions	590
10.12	Math Functions and Operators	617
10.13	Matrix Functions and Operators	633
10.14	Network Type Functions	637
10.15	Scalar Data Conversion Functions	639
10.16	Sorted Aggregate Functions	647
10.17	Time Zone Functions	648
10.18	Tuple Functions and Operators	654
10.19	Window Aggregate Functions	655
10.20	Other Functions and Expressions	674
10.21	Data Definition Language (DDL) Statement Reference	681
10.22	Data Control Language (DCL) Statement Reference	822
10.23	Errors and Warnings	831
10.24	Reserved Words	844
10.25	System Catalog Reference	845
11	Connection Driver Reference	946
11.1	JDBC Manual	946
11.2	Ocient Python Module: pyocient	966
12	Loading and Transformation	977
12.1	Ingest Data with Legacy LAT Reference	977
12.2	LAT Overview	1011
12.3	LAT Data Types in Loading	1012
12.4	LAT Pipeline Configuration	1021
12.5	LAT Data Transformation with JMESPath	1058
12.6	LAT Service Configuration	1078
12.7	LAT Metrics	1082
12.8	LAT Log4J Configuration	1087
12.9	LAT Packaging and Installation	1088
12.10	LAT Advanced Topics	1093
12.11	LAT Client Command Line Interface	1096
13	Version Compatibility	1105
14	Release Notes	1109
14.1	Ocient Hyperscale Data Warehouse Release Notes	1109

1 Get Started with the Ocient Hyperscale Data Warehouse

The Ocient Hyperscale Data Warehouse™ is a data warehouse for running analytics against very large and complex data sets. It is designed to be cost-effective and flexible in its deployment options. Built using standard ANSI SQL, Ocient® does not require users to spend time and money learning unfamiliar programming languages, and with industry-standard interfaces like SQL and JDBC, Ocient makes it easy for organizations to interact with data. Using commodity hardware, Ocient can be deployed on premises in a data center, in the cloud, or as a fully-managed service in the OcientCloud®. In addition to being flexible and cost-effective, Ocient is designed to be extremely fast and scalable with large data sets.

Traditional relational databases, even those designed for analytics, do not scale from a cost or performance perspective to accommodate extremely large data sets. Big data solutions can scale to use large data sets, but are often expensive, complex, inflexible, and take too long to execute queries. Ocient is designed to give users the interactive analysis capabilities of traditional relational databases at the scale of big data solutions, and with a significantly smaller footprint. This means that queries that used to take hours, or not run at all, now execute in seconds, and systems that used to fill up multiple data center racks, now require up to 90% less space and energy.

Ocient also supports real-time analytics, traditional OLAP workloads, geospatial analysis, and in-database machine learning and AI. By enabling users to perform all of these tasks on a single data platform, Ocient eliminates additional data movement and enables organizations to operate at a scale that was previously impossible or impractical.

Key Concepts

Core Ocient concepts and terms needed for a quick start

Install an Ocient System

Installation, Upgrade, System Configuration, Maintenance, and Monitoring of an Ocient System

Connect to Ocient

Connection to Ocient with JDBC and other drivers

Database Administration

Overview of key database administration capabilities like workload management, result set caching, and managing users, groups, and roles

Load Data

Loading and transformation capabilities, managing pipelines, transformation functions, and error handling

Query Ocient

Overview of Ocient data types and the components of a query

Core Elements of an Ocient System

Overview of Ocient System design and architecture

These sections outline the capabilities of Ocient for data ingestion, transformation, and advanced analytics. Ocient SQL supports integrated libraries for geospatial analysis and machine learning tools.

SQL Reference

SQL command, function, operator, and keyword reference including DDL, DCL, and query reference

Connection Driver Reference

Different SQL client connectors

Machine Learning in Ocient

Walkthroughs to deploy various ML models integrated into Ocient

Geospatial Functions

Functions for analyzing geospatial data sets

For information about integrations and connections to third-party applications and tools, see [Ocient Integrations](#).

For a detailed explanation of the Compute Adjacent Storage Architecture® in Ocient, design principles, storage capabilities, and query engine, see [Ocient Architecture](#).

For information on ingesting data using the legacy Loading and Transformation (LAT) functionality, see [Ingest Data with Legacy LAT Reference](#).

1.1 Key Concepts

The Ocient Hyperscale Data Warehouse™ is a real-time OLAP datastore designed for running analytics against large time series data sets. The Ocient® System operates in compliance with the ANSI standard for SQL.

The Ocient System includes:

- **Ultra-fast query speeds** for analytics on hyperscale data sets.
- **Scalability** for cost and performance on a system built with commodity hardware.
- **Flexible deployment** for running on premises, in the cloud, or as an Ocient-managed service.
- **Robust functionality** for DDL, geospatial (OcientGeo™), and machine learning (OcientML™).
- **Data movement** at scale for ETL and ELT ingestion and operations.

The Ocient System

The Ocient System is based on Compute Adjacent Storage Architecture® (CASA), which collocates NVMe drive storage with the system compute resources to optimize performance. This design keeps records near and accessible for computation without a separate storage layer, which avoids many common bottlenecks for database engines, such as limits on network capacity or processing throughput. The design achieves superior query performance when it operates on trillions of rows of data.

To learn more about the design principles behind the Ocient System, see [Ocient Architecture](#).

Deployment

Ocient is designed for flexible deployment for unique use scenarios that include:

- **OcientCloud®** — A cloud deployment managed by Ocient Support. This option uses performance-tested hardware and network infrastructure.
- **Public Cloud** — Installation on third-party cloud providers, including Amazon® Web ServicesSM (AWSSM) and Google CloudTM.
- **On premises** — A self-managed option for hosting Ocient in your own data center.

System Nodes

Ocient is a distributed system consisting of interconnected nodes within clusters, which together form the data warehouse. The Ocient System designates nodes for these system roles:

- **SQL Nodes** — Parse incoming SQL statements and administer commands throughout the system. These nodes serve as the interface for system and database administration.
- **Loader Nodes** — Manage ETL ingestion from batch or streaming sources and index data for optimized performance.
- **Foundation Nodes** — Store data and perform the bulk of query processing. Foundation Nodes perform as much query processing as possible on their stored data before interacting with other nodes.

Understanding nodes and their roles in an Ocient System is most useful for system setup, administration, and maintenance. For details about nodes, see [Ocient Architecture](#). However, the Ocient System makes it easy to load and query data without knowing these details.

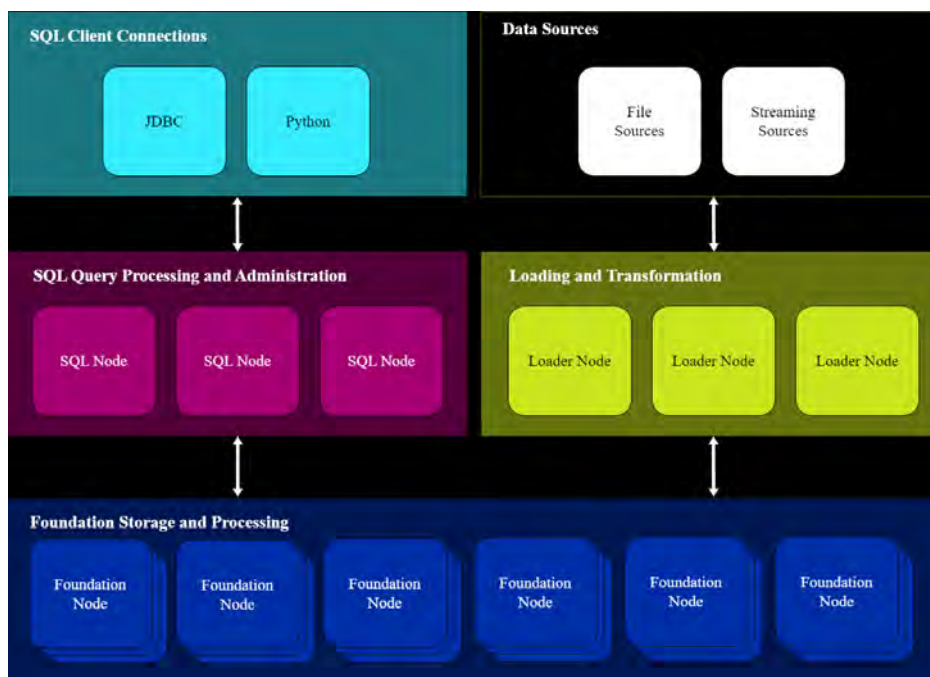


Figure 1: Ocient architecture diagram that shows the relationship between data sources, loading and transformation of the data, and data storage

Data Types

The Ocient Hyperscale Data Warehouse, the functions, and the indexes use only native SQL data types to optimize storage and performance. Supported data types include common SQL scalar types and complex types such as IP, arrays, tuples, and geospatial. For details about supported data types, see [Understanding Data Types](#).

Loading

The Ocient System can load data for an end-to-end flow from file or streaming sources, including common sources such as:

- Apache® Kafka®
- AWS S3
- Apache® Hadoop® Distributed File System (HDFS)

For all the source and format options, Ocient uses pipelines to control ingestion. You can control pipelines using DDL commands, API on Loader Nodes, or a command-line interface. You can transform data during loading using SQL functions.

Loading ingestion throughput can scale as needed based on the Ocient architecture.

For details, see [Load Data](#).

Query Processing

The design of the Ocient architecture minimizes the amount of on-disk data that must be read and processed to execute a query. To do this, the system compiles a custom I/O pipeline for each data segment relevant to a query. These custom pipelines make use of any keys or indexes that can help facilitate faster throughput.

To learn more about query processing, see [Query Performance Optimizations](#).

Ocient Indexing

An Ocient datastore can use multiple layers of indexing to facilitate query performance. Using these indexes is pivotal to optimizing query performance for large data sets.

You can apply indexes using basic DDL statements that do not require deep knowledge of the internal system or data set. You also have some ability to customize indexes for their specific use cases.

Indexes are divided into two main categories.

Segment Keys

These indexes require no additional storage and usually should be deployed on every table. Segment keys operate by accessing data from the partitioned data segments to quickly filter rows without I/O processing.

Segment keys include:

- **Time Key** — A segment key that partitions data based on a time-series data column.
- **Clustering Key** — A series of columns that are frequently queried together. The system subdivides these segments on disk for faster reference.

For more information on segment keys, see [TimeKeys and Clustering Keys](#).

Secondary Indexes

When deployed precisely, Ocient secondary indexes can dramatically reduce the time for queries to run on large data sets.

Ocient supports secondary indexes for these data categories:

- **Numeric**
- **String**
- **Partial string**
- **Geospatial**

For more information, see [Secondary Indexes](#).

Connecting to Ocient

Ocient provides SQL access to its database using industry-standard drivers:

- JDBC
- pyocient, a Python®-based driver for Ocient.

End users can query and analyze data in Ocient without understanding the organization, nodes, networking, or other parts of the system architecture.

For more information, see [Connect to Ocient](#).

Integrations

Ocient supports integration with various third-party tools for database administration, business intelligence, and system monitoring.

For details about supported third-party tools, see [Ocient Integrations](#).

Resilience to Hardware Failure

The Ocient System uses erasure coding to organize and compute parity blocks so the system is fault-tolerant and can rebuild missing data. This failsafe does not require redundant copies of data, meaning that storage requirements are minimal.

Security

As a unified platform, Ocient helps keep data secure by consolidating security capability and reactions in one place with a suite of auditing and monitoring tools. Ocient supports these security standards:

- **Data Encryption:** Optional through TLS/SSL protocols.
- **Security Compliance:** Audited and certified SOC 2 Type 2.
- **Monitoring:** Log-level monitoring and alerts on key system information.
- **Access Controls:** SSO, role-based, and system-level access controls are available.

Related Links

- [Core Elements of an Ocient System](#)
- [Ocient Architecture](#)
- [Ocient Simulator](#)
- [Connect to Ocient](#)

Related Videos

- At the Whiteboard with Ocient: Compute Adjacent Storage Architecture™
- At the Whiteboard with Ocient: SQL at Scale
- At the Whiteboard with Ocient: Indexing at Hyperscale

2 Connect to Ocient

You can connect to any of the SQL Nodes in an Ocient® System to run queries or commands. The SQL Node is available by default on port 4050 at the IP address or hostname assigned to the SQL Nodes. When more than one SQL Node is present in the Ocient System, Ocient automatically loads balance connections between the nodes to even out the load across connections.

Ocient provides a JDBC driver and a Python® driver named `pyocient`. You can use any of these drivers to connect to an online Ocient System.

To connect from a client computer that can access the SQL Node using the network, follow any of the following examples. Note that you must have the IP address of the SQL Node and a valid username and password to connect.

- [Connect Using JDBC](#)
- [Connect Using pyocient](#)

If you are unable to connect, check the logs on the SQL Nodes for issues or check the [Errors and Warnings](#) for more detail on error codes.

For supported third-party tools, see [Ocient Integrations](#).

2.1 Ocient Integrations

Ocient® databases can integrate with various industry tools and applications to expand their functionality.

The third-party tools listed in this section are certified to work with Ocient hyperscale systems. The included walkthroughs are intended to explain any requirements, setup instructions and additional specifications needed for the applications to work fully with Ocient Systems.

Database tools not included in this section can still function with Ocient Systems.

Supported Ocient Integrations

- [Superset Integration](#)
- [DBeaver Integration](#)
- [Metabase Integration](#)
- [SQLAlchemy Integration](#)
- [Tableau Integration](#)

Ocient Systems assume that all date and time data is in UTC format. Therefore, third-party tools can require time zone configuration to get accurate results on date and time functionality. To configure the time zone settings for integrated applications, refer to the documentation for the respective tools. For more information, see the [Time Zone Functions](#) section.

SQLAlchemy is a trademark of Michael Bayer. mike(ℹ)zzzcomputing.com All rights reserved.

2.1.1 DBeaver Integration

DBeaver is a free and open-source application for database administration and SQL querying.

Prerequisites

To use DBeaver with the Ocient® System, you must have these software prerequisites:

- Ocient System — Use latest version.
- JDBC Driver — Use latest version. See the Ocient Maven repository for a list of all versions.

To install, download the latest version of DBeaver from the DBeaver Community for your respective operating system.

Follow the installation steps and open DBeaver.

Connect to Ociant

You can connect to your database by following these steps:

- In the main DBeaver window, click **New Database Connection** in the top left of the window.
- In the New Database Connection window, find the Ociant driver. Select it, and click **Next**.
- Enter your database parameters with this information.

Field	Description
Host	The hostname or IP address of the SQL Node of your Ociant System. For example: <code>databasehost-sql0</code>
Port	The port number for your connection. The default Ociant port is 4050.
Database/Schema	The name of the database for connection. For example: <code>my_db</code>
User	The username associated with your database.
Password	The password associated with your username.

- Click **Finish**.

Update Ociant JDBC

By default, DBeaver downloads the latest JDBC version available in the Maven repository. If you need to update the JDBC driver or load a specific version, follow these steps.

- In the DBeaver window, select **Database > Driver Manager**.
- Search among the list of driver options to find the Ociant driver, and select it.
- With the Ociant driver highlighted, click **Edit**.
- In the Edit window, select the **Libraries** tab.

Update the Driver to the Latest Version

To update the driver to the latest version, click **Download/Update**.

Load a Specific JDBC Driver Version

- Double-click on the `com.ociant:ociant-jdbc4:RELEASE` in the Libraries window. The Edit Maven™ Artifact window opens.
- Enter the JDBC version in the **Version** field.

For accurate time values, you might need to configure the DBeaver time zone settings.

- For macOS® systems, configure this setting in **Settings > User Interface**.
- For Windows® systems, configure this setting in **Preferences > User Interface**.

Related Links

[Connect to Ociant](#)

[Ociant Integrations](#)

2.1.2 Metabase Integration

MetabaseSM is an open-source business intelligence tool designed to create charts and dashboards from various data sources.

Prerequisites

To use Metabase with your Ociant® System, you must have these software prerequisites:

- Ociant System — Use version 22.0 or later.
- JDBC driver — Use the latest version.

Install Metabase

1. Go to the Installing Metabase and follow the directions for installation, including all prerequisites.
2. Download the latest ociant-metabase-driver.jar file and copy it to your Metabase plugins folder. This JAR file enables your Metabase installation to connect to Ociant.

Connect to Ociant

To add a database connection, select the icon at the top right of the Metabase homepage. Select **Admin settings > Databases > Add database** in the menu.

This connection page appears. The displayed connection fields change after you select “Ociant” as the database type. Enter these fields in your connection settings.

Field	Description
Database type	The type of database. Select “Ociant”.
Display name	The display name for your database in Metabase.
Host	The host name or IP address of the SQL Node of your database. For example: <code>databasehost-sql0</code>
Port	The port number for your connection. Unless you have to alter this, the default Ociant port is 4050.
Database name	The name of the database for connection. For example: <code>my_db</code>
Schemas	The identifiers for any schemas you want to use or exclude. To use all available schemas, select <code>All</code> .
Authentication method	Choose between <code>Single Sign-On (Token)</code> for single sign-on (SSO) or <code>Password</code> for password authentication.
User	The username associated with your database.
Password	The password associated with your username.
Type of SSO token	Use <code>access_token</code> .
SSO token	Specify your SSO access token string.

To use SSO authentication, you must have SSO integration enabled for the specified database and group. For more information, see [Database, Tables, Views, and Indexes](#).

When you successfully connect, you see your database added to Metabase. Your database appears as one of the active connections in Metabase.

Advanced Connection Options

You can add more connection settings as part of the JDBC connection string in Metabase. To add these, go to the **Add database** page and select **Show advanced options**.

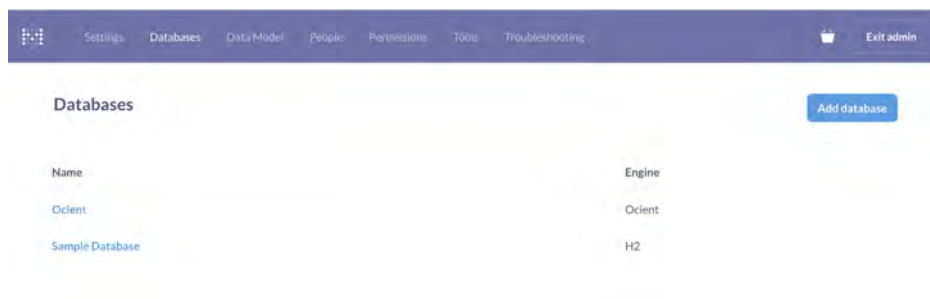


Figure 2: List of Metabase Databases

You can specify options in the field **Additional JDBC connection string options**. In this image, the specified options include a maximum of 10,000,000 rows and a timeout of 0 milliseconds, using the `maxRows` and `timeoutMillis` parameters respectively.

For a list of supported JDBC options, see the [JDBC Manual](#).

Related Links

[Connect to Ocient](#)

[Ocient Integrations](#)

2.1.3 SQLAlchemy Integration

SQLAlchemy is an open-source SQL toolkit and object-relational mapper for Python®.

Prerequisites

To use SQLAlchemy, you must have these software packages:

- Ocient® System — Use the latest version.
- Python — Use version 3.5 or later.
- pyocient driver — Use version 1.0.15. For details, see [pyocient Release Notes](#).

Installation

For information about installing SQLAlchemy, please see the [sqlalchemy-ocient PyPISM](#) page.

Related Links

[Connect to Ocient](#)

[Ocient Integrations](#)

SQLAlchemy is a trademark of Michael Bayer. mike(📧)zzzcomputing.com All rights reserved.

2.1.4 Superset Integration

Apache Superset is an open-source business intelligence web application that can quickly generate visualizations, dashboard, and geospatial maps.

Prerequisites

To use Superset with your Ocient® System, you must have these software prerequisites:

- Ocient System — Use the latest version.

- Python® — Use version 3.8 or later.
- SQLAlchemy — Install the latest version by using the sqlalchemy-ocient PyPISM page.
- pyocient driver — Use version 1.0.15. For details, see [pyocient Release Notes](#).

Installation

To install, follow the Installing Superset Locally Using Docker Compose guide for your respective system.

Connect to Ociant

For specific steps on connecting Superset to your Ociant System, see the Ociant DB page.

Related Links

[Connect to Ociant](#)

[Ociant Integrations](#)

SQLAlchemy is a trademark of Michael Bayer. mike(✉)zzzcomputing.com All rights reserved.

2.1.5 Tableau Integration

Tableau® is an interactive data visualization application designed to create charts and dashboards from various data sources.

Prerequisites

To use Tableau with the Ociant® System, you must have these software prerequisites:

- Ociant System — Use the latest version.
- JDBC driver — Download the latest version with dependencies from the Maven repository.

Installation

1. Download the latest version of Tableau Desktop for your respective operating system from the Download Tableau Desktop website.
2. Register and sign in to Tableau Desktop. Then, follow steps to complete the installation.

Add Necessary Packages

To connect to Ociant, you must add these files to your Tableau installation:

- Ociant JDBC driver (.jar) — Use the latest version with dependencies from the Maven repository.
- Ociant Tableau connector file (.taco) — Download the latest version from Tableau Exchange.

The .taco connector file supports Tableau Desktop and Tableau Server. Contact Tableau for support regarding Tableau Cloud.

Add the Ociant JDBC Driver JAR File

Move the downloaded JDBC driver JAR file to the corresponding folder based on your operating system:

- Windows® — C:\Users[Windows User]\Documents\My Tableau Repository\Connectors
- macOS® — /Users/[user]/Documents/My Tableau Repository/Connectors

Add the Tableau Connector File (.taco)

Move the downloaded .taco connector file to the corresponding folder based on your operating system:

- Windows — C:\Program Files\Tableau\Drivers

- macOS — /Users/[user]/Library/Tableau/Drivers

Connect to Ocient

Connect to your database by following these steps:

- Open Tableau and find the **Connect to a Server** menu. Select **Ocient JDBC** from the list of installed connectors. If **Ocient JDBC** does not appear as an option in Tableau, ensure that the Ocient Tableau connector file (.taco) is located in the correct folder.
- In the connection window, enter your database parameters with this information.

Field	Description
Host	The hostname or IP address of the SQL Node of your Ocient System. For example:databasehost-sql0
Port	The port number for your connection. The default Ocient port is4050.
Database	The name of the database for connection. For example:my_db
User	The username associated with your database.
Password	The password associated with your username.

- Select **Sign In** after you enter all the information.

Related Links

[Connect to Ocient](#)

[Ocient Integrations](#)

2.2 Connect Using JDBC

The JDBC driver can be executed in a command-line interface (CLI) mode or used directly in a program with standard JDBC methods. This example describes how to connect using the CLI mode. For details on using the JDBC driver in code, please see Establishing a Connection.

Step 1: Download the JDBC Driver

First, confirm the recommended version of the JDBC driver for your version of Ocient® by finding it in [Version Compatibility](#). Download the correct JDBC driver from the Apache® Maven™ repository by clicking on your preferred version, then selecting “view all” and finally downloading the JAR file that is labeled with the pattern ocient-jdbc4-<version number>-jar-with-dependencies.jar.

Save this file to a convenient location on your client machine.

Step 2: Run the JDBC CLI Program

To run the JDBC CLI Program, you must have Java® installed on your machine. For the version, see [Version Compatibility](#). Next, run the following command to start the JDBC CLI Program. Assuming your JAR stored at ~/ocient-jdbc4-jar-with-dependencies.jar.

```
java -classpath ~/ocient-jdbc4-jar-with-dependencies.jar com.ocient.cli.CLI
```

The system prompts you to enter the username and password.

```
Username: test@system
Password: fakepassword
```

Example Response:

```
Ocient> _
```

The Ocient CLI prompt appears.

Step 3: Connect

To connect, enter the connection string to your SQL Nodes. SQL Nodes install by default with a self-signed TLS certificate for encryption by default, and a custom certificate can be present. Depending on the TLS configuration, you can use `tls=unverified` (default) or `tls=on` for the required setting.

Assuming the standard port 4050, a self-signed certificate, and the SQL Node IP Address of 10.10.1.1, you can connect to the system database with the following connecting string:

```
Ocient> connect to jdbc:ociant://10.10.1.1:4050/system;
```

Example Response:

```
Connected to jdbc:ociant://10.10.1.1:4050/system
Ocient> _
```

You are now connected and can execute queries based on your user's permissions. Try running: `select * from sys.tables;` to see what tables are defined.

For more details on using the JDBC driver and the CLI program, see the [JDBC Manual](#).

The system cancels any queries made with the JDBC driver if you perform any of these actions:

- Fetch the entire result set.
- Close the connection.
- Execute a new query on the same connection.

Related Links

[Connect to Ocient](#)

[Ocient Integrations](#)

2.3 Connect Using pyocient

The Ocient® Python® module `pyocient` supports direct connections to Ocient Systems in a Command-Line Interface (CLI) as well as in Python programs that follow the Python Database API Specification 2.0.

After you install `pyocient`, you can establish Ocient connections to run queries and commands.

Installation Tutorial

To install and run `pyocient`, follow these steps.

Install pyocient

Install `pyocient` using `pip`.

```
pip3 install pyocient
```

After running the `pip` install, `pyocient` is installed in the `bin` directory.

Use `venv` to install `pyocient` using a virtual environment. For more information, see the Python `venv` documentation.

Start pyocient

pyocient is started from the command line with a connection string. It includes many options to configure the connection and client similar to the JDBC driver. Learn more about supported options in the [Ocient Python Module: pyocient](#).

After installation, you can launch from a command line by typing the **pyocient** command.

```
pyocient ocient://example_user:example_password@10.10.1.1/system
```

To connect to the system database, replace the example username (**example_user**) and password (**example_password**) with your credentials, set your IP address,

The connection example assume a SQL Node exists at 10.10.1.1. The example uses the default port 4050, and is set up with default self-signed certificates for TLS.

Example Response

Ocient Database

System Version: <Ocient System Version>, Client Version <pyocient Version>

```
> _
```

Use the command `quit`; to exit the **pyocient** CLI interface.

Run a Query

In the **pyocient** CLI, you can run any SQL query supported by your database. In this example, a query is selecting rows from the system tables:

```
SELECT * FROM sys.tables;
```

You can also run queries from the command line with the connection string:

```
pyocient ocient://example_user:example_password@10.10.1.1/system 'SELECT * from sys.tables'
```

Example Response

```
[
  {
    "id": "359e21b1-770a-4860-aec9-f158d307f725",
    "name": "data_type_coverage",
    "schema": "loading",
    "database_id": "f827ff7c-dd94-43bf-8386-93d048344a32",
    "storage_space_id": "2ad08e7c-15d1-4861-b38b-256c4df4a191",
    "maximum_segment_size_gib": 4,
    "description": null,
    "streamloader_property_string": null,
    "created_at": "2023-08-08T21:22:49.622213"
  }
]
```

The system cancels any queries made with the **pyocient** driver if you perform any of these actions:

- Fetch the entire result set.
- Close the connection.
- Execute a new query using the same connection.

For more information and API reference material for **pyocient**, see the [Ocient Python Module: pyocient](#) page.

Related Links

[Connect to Ocident](#)

[Ocident Integrations](#)

3 Load Data

Loading data into the Ocient Hyperscale Data Warehouse™ using the data pipeline functionality is as simple as writing a SQL statement to query data from different supported sources. The loading operation provides real-time transformations and maximizes performance using all available processing resources.

With data pipelines, you manage data loading activities using Data Definition Language (DDL) statements similar to the ones used for managing database tables. You can access all information about pipelines by querying the system catalog tables. You can execute operations on pipelines with a command-line statement using standard interfaces like JDBC and ODBC.

Data Pipelines Overview

A data pipeline is the primary way that the Ocient® System loads data. Each data pipeline is a database object that defines the end-to-end processing of data for the extraction, transformation, and load into Ocient tables. Data pipelines execute across your Ocient System to coordinate parallel loading tasks across many Loader Nodes and pipelines.

The Pipeline Object

Creating a pipeline is complex as there are many options to consider. Use the `PREVIEW PIPELINE` SQL statement to preview the pipeline creation and load of data to ensure that the pipeline definition returns the results you expect. After you are satisfied with the results, you can execute the `CREATE PIPELINE` statement to create the pipeline at scale.

When you create a pipeline, you assign it a name and the pipeline exists as an object in the Ocient System connected to this name. Then, you can control the object using your chosen name with SQL statements like `START PIPELINE`, `STOP PIPELINE`, `CREATE OR REPLACE PIPELINE`, and `DROP PIPELINE`. You can also define your own function by using the `CREATE PIPELINE FUNCTION` SQL statement. Execute these statements using a SQL connection.

A pipeline maintains its own position during a load and enforces deduplication logic to ensure that the Ocient System only loads data once. The lifecycle of a pipeline defines both the deduplication logic and load position.

- Pipeline Events: During the life of a pipeline, you can start, stop, modify, and resume the pipeline without duplicating source data or losing the position in a load.
- Pipeline Updates: You can modify pipelines using the `CREATE OR REPLACE` SQL statement to update the transforms in a pipeline, but maintain the current position in the load.
- Load Position: If target tables are truncated or if a target table is dropped and then recreated, the pipeline maintains its own position in the load and will continue from its last position.

Pipelines also gracefully handles many error cases and system failure modes. The pipeline stores any errors that occur in the system catalog in association with the pipeline.

For more details on Deduplication and Error Tolerance settings, see:

- [Deduplication in Data Pipelines](#)
- [Error Tolerance in Data Pipelines](#)
- [Data Pipelines](#)

Parts of a Pipeline

Pipelines can operate in either a `BATCH` or `CONTINUOUS` mode based on how you plan to load your data.

The DDL for Data Pipelines has three sections.

Section	Syntax
Data Source	<code>SOURCE ...</code>
Data Format	<code>EXTRACT FORMAT ...</code>

Section	Syntax
Transformations	INTO table SELECT ...

Thus, for example, the full SQL syntax has this format for the `orders_pipeline` data pipeline. This syntax has an S3 data source and loads data into the `orders` table in the `public` schema. The `SELECT` SQL statement in the `CREATE PIPELINE` statement selects the `id`, `user_id`, `product_id`, and other columns from a table.

```
CREATE PIPELINE orders_pipeline
SOURCE
  S3
    ENDPOINT '...'
    BUCKET '...'
    FILTER '...'
EXTRACT
  FORMAT delimited
  ...
INTO "public"."orders"
SELECT
  $1 as id,
  $2 as user_id,
  $3 as product_id,
  ...
```

Data Source

The `SOURCE` part in a pipeline defines the source of the data to load. This DDL part identifies the type of the source (e.g., Amazon® Web ServicesSM (AWSSM) S3, Apache® Kafka®) as well as the details for the data that is relevant to the source (e.g., S3 bucket and filters, Kafka topic and consumer configuration).

- [Data Pipelines](#)
- [Data Pipelines](#)
- [Data Pipelines](#)
- [Data Pipelines](#)

Data Format

The `EXTRACT` part in a pipeline defines the format of the data that the Ociant System extracts from the data source. This part includes the data format (e.g., delimited, binary, JSON) and the details about the records (e.g., record delimiter, how to treat empty data). In addition, the Ociant System extracts metadata that you can load into tables.

- [Data Formats for Data Pipelines](#)
- [Data Formats for Data Pipelines](#)
- [Data Formats for Data Pipelines](#)
- [Data Formats for Data Pipelines](#)
- [Data Formats for Data Pipelines](#)
- [Data Pipelines](#)

Transformations

The `INTO table SELECT ...` SQL statement defines the target tables where the data loads and uses a `SELECT` SQL statement to extract fields from the source data (e.g., CSV field index `$5`, JSON selector `$order.user_name`) and map them to target columns (e.g., `... as my_column`).

The `SELECT` statement in a pipeline can utilize a subset of SQL functions to convert the extracted data into the required format during loading (e.g., `TO_TIMESTAMP($my_date, 'YYYY-MM-DD')`).

The selectors you use to extract data from the source records can differ based on the data format (e.g., JSON or DELIMITED).

- [Transform Data in Data Pipelines](#)
- [Transform Data in Data Pipelines](#)
- [Data Formats for Data Pipelines](#)

For loading workflows, see these examples:

- [Data Pipeline Load of CSV Data from S3](#)
- [Data Pipeline Load of JSON Data from Kafka](#)
- [Data Pipeline Load of Parquet Data from S3](#)

Pipeline Lifecycle

This state machine shows the state transitions for a data pipeline. You can start and stop data pipelines using DDL statements. The status of the pipeline transitions automatically as the pipeline completes all assigned work or reaches a failed state.

While executing, the pipeline lists files or connects to streaming sources and creates tasks to execute work. These states reflect the overall progress of a pipeline and capture details of the underlying tasks that execute the work of the pipeline.

State	Description
Created	The pipeline is created, but no user has started it. Zero tasks in the <code>CREATED</code> state. Zero files listed or no Kafka Consumer group created.
Running	The pipeline is running after a user starts the pipeline. At least one task is in the <code>QUEUED</code> , <code>RUNNING</code> , or <code>CANCELLING</code> states. At least one file listed or Kafka Consumer group created.
Stopped	The pipeline is not processing data after a user stops the pipeline. All tasks are in the <code>QUEUED</code> , <code>COMPLETE</code> , <code>FAILED</code> , or <code>CANCELLED</code> states.
Completed	The pipeline finished all assigned work according to the error limits defined for the pipeline. All tasks are in the <code>COMPLETE</code> state.
Failed	The pipeline failed to complete and no longer runs due to error limits defined for the pipeline. At least one task in the <code>FAILED</code> state. All tasks are in the <code>QUEUED</code> , <code>COMPLETE</code> , <code>FAILED</code> , or <code>CANCELLED</code> states.

In this state diagram, the transition arrows display the user actions that can trigger a state change. State transitions without a label are system-initiated transitions.

Observe Pipelines

During pipeline operation, all of the information you need to observe progress, pipeline status, key events, success or failure, and errors is available in system catalog tables in the Ocient System. In addition, performance counters are available for many key loading metrics that you can add to observability systems.

- [Monitor Data Pipelines](#)
- System Catalog

Troubleshooting Pipelines

Data pipelines support robust error-handling capabilities crucial for developing new pipelines, identifying issues with ongoing operations, and correcting bad data. Pipelines can include a bad data target where the Ocint System saves bad data for troubleshooting. In addition, the system captures errors that you encounter during loading in system catalog tables such as `sys.pipeline_events` and `sys.pipeline_errors`. Exploring the data in these tables enables you to detect issues and identify the root cause of errors during loading.

- [Manage Errors in Data Pipelines](#)
- [Frequently Asked Questions for Data Pipelines](#)
- [Data Pipeline Loading Errors](#)
- [Data Pipelines](#)
- [Data Pipelines](#)

Loading Architecture

Pipelines require Loader Nodes to operate. Loader Nodes are Ocint® Nodes that have the `streamloader` role assignment. The loading system can use all active nodes with this role to process pipelines.

When you execute a pipeline, Ocint spreads the work among the available Loader Nodes. The system executes these parallel processes as tasks on each of the Loader Nodes. Then, the system partitions the files or Kafka partitions to load across the tasks, and loading proceeds in parallel. When all tasks are completed, the pipeline is completed. Many pipelines can run in parallel on the Loader Nodes, and these Nodes share resources.

The scale-out architecture of the Ocint System allows you to add more Loader Nodes as needed to expand the volume and complexity of pipelines that you can operate at the throughput required by your application.

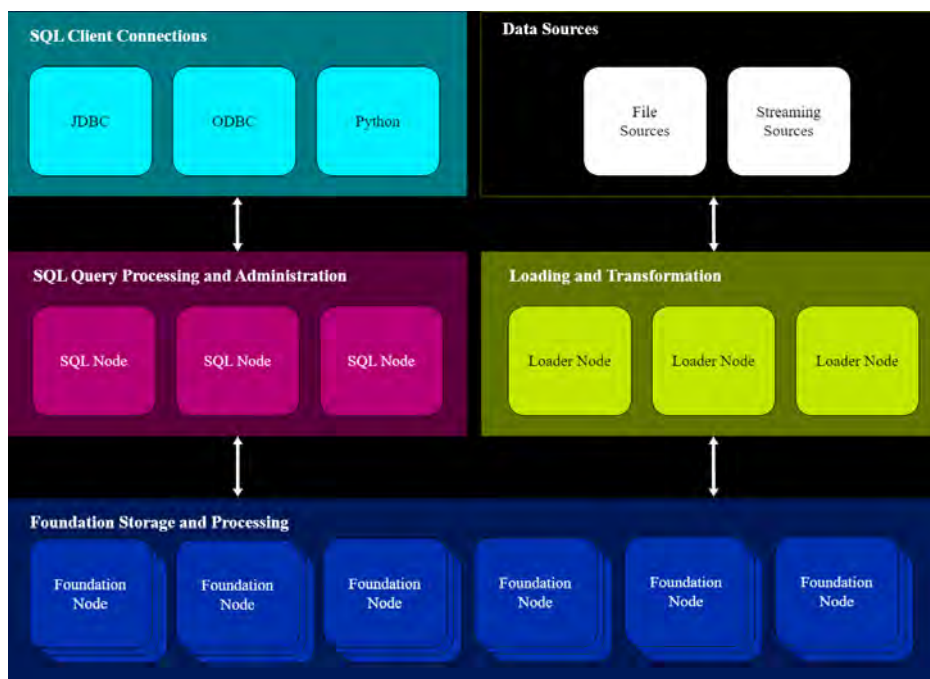


Figure 3: Ocint loading architecture that consists of SQL client connections that interface with SQL Nodes, data sources that interface with Loader Nodes, and SQL and Loader Nodes interface with Foundation Nodes for storage and processing.

Internal Operations of a Pipeline

As pipelines process records from a data source, the Ocint System builds these rows into a row-based storage structure called a page. The system optimizes pages for rapid ingestion of data. The system stores pages on Foundation Nodes to ensure data integrity.

As pages accumulate, Loader Nodes convert pages into a columnar storage format named a Segment. Segments are highly compressed data structures that include data, indexes, and other statistical metadata to optimize query performance. After the Ocient System creates Segments, the system forms them into groups and transfers them to Foundation Nodes for storage.

You do not typically need to be concerned with or control these internal operations. However, it is important to understand that pipeline throughput can be affected by the parallel processing of many pipelines on shared resources, the number of indexes, and the type of compression used on your tables.

Related Links

[Data Control Language \(DCL\) Statement Reference](#)

[Data Pipelines](#)

[Data Pipelines](#)

[Data Pipelines](#)

[Data Pipelines](#)

[Data Pipelines](#)

[Data Pipelines](#)

[Data Pipelines](#)

[Data Pipelines](#)

[Data Pipelines](#)

[Frequently Asked Questions for Data Pipelines](#)

3.1 Load by Data Source

3.1.1 Data Pipeline Load of CSV Data from S3

A common setup for loading files in a batch into Ocient® is to load from a bucket on Amazon® Web ServicesSM (AWSSM) S3 with time-partitioned data. Often, you must perform a batch load repeatedly to load new files.

Ocient uses data pipelines to transform each document into rows in one or more different tables. The loading and transformation capabilities use a simple SQL-like syntax for transforming data. This tutorial guides you through a simple example load using a small data set in CSV format. The data in this example comes from a test set for the MetabaseSM Business Intelligence tool.

Prerequisites

This tutorial assumes that:

1. The Ocient System has network access to S3 from the Loader Nodes.
2. An Ocient System is installed and configured with an active Storage Cluster (See the [Ocient Application Configuration](#) guide).

Step 1: Create a New Database

To begin, load two example tables in a database. First, connect to a SQL Node using the [Commands Supported by the Ocient JDBC CLI Program](#). Then, execute the CREATE DATABASE SQL statement for a database named `metabase`.

```
CREATE DATABASE metabase;
```

Step 2: Create Table

Create the `orders` table in the new database. First, connect to that database (e.g., `connect to jdbc:ocient://sql-node:4050/m`) and then execute this `CREATE TABLE` SQL statement that specifies to create a table with these columns and a clustering index based on the `user_id` and `product_id` columns:

- `created_at` as a TimeKey® timestamp that is not nullable.
- `id`, `user_id`, and `product_id` as integers that are not nullable.
- `subtotal`, `tax`, `total`, and `discount` as floating point numbers.
- `quantity` as an integer.

```
CREATE TABLE public.orders (
  created_at TIMESTAMP TIME KEY BUCKET(30, DAY) NOT NULL,
  id INT NOT NULL,
  user_id INT NOT NULL,
  product_id INT NOT NULL,
  subtotal DOUBLE,
  tax DOUBLE,
  total DOUBLE,
  discount DOUBLE,
  quantity INT,
  CLUSTERING INDEX idx01 (user_id, product_id)
);
```

The database creates the `orders` table, and you can begin loading data.

Step 3: Create a Data Pipeline

Create data pipelines using the `CREATE PIPELINE` SQL statement. To load data, you first create a pipeline with the definition of the source, data format, and transformation rules using a SQL-like declarative syntax. Then, you execute the `START PIPELINE` command to start the load. You can observe progress and status using system tables and views.

Each Ocient pipeline defines a single data source and the target table or tables into which data loads. A data source includes the location of the source and filters on the source to define the specific data set to load. This example loads data from two data sources, where each source is located in a directory on the same S3 bucket.

First, inspect the data that you plan to load. Each document has a format similar to this example CSV file named `orders.csv`.

```
id,user_id,product_id,subtotal,tax,total,discount,created_at,quantity
1,1,14,37.65,2.07,39.72,null,2019-02-11T21:40:27.892Z,2
2,1,123,110.93,6.1,117.03,null,2018-05-15T08:04:04.580Z,3
3,1,105,52.72,2.9,49.2,6.42,2019-12-06T22:22:48.544Z,2
...
```

In this case, Ocient automatically transforms the data to the target columns using some sensible conventions. In other cases, loads require some transformation. Most transformations are identical to functions that already exist in the SQL dialect of the Ocient System.

Create a pipeline named `orders_pipeline` for the orders data set from your database connection prompt. Use the S3 data source with endpoint `https://s3.us-east-1.amazonaws.com`, bucket `ocient-examples`, and filter `metabase_samples/csv/orders.csv`. Specify the CSV format with one header line. Load the data into the `public.orders` table. The `SELECT` part of the SQL statement maps the fields in the CSV file to the target columns in the created table.

SQL

```
CREATE PIPELINE orders_pipeline
SOURCE
  S3
```

```

        ENDPOINT 'https://s3.us-east-1.amazonaws.com'
        BUCKET 'ocient-examples'
        FILTER 'metabase_samples/csv/orders.csv'
EXTRACT
    FORMAT csv
    NUM_HEADER_LINES 1
INTO public.orders
SELECT
    $1 as id,
    $2 as user_id,
    $3 as product_id,
    $4 as subtotal,
    $5 as tax,
    $6 as total,
    $7 as discount,
    $8 as created_at,
    $9 as quantity;

```

The pipeline has three main sections:

- **SOURCE**— In this case, load data from S3. Specify the endpoint and bucket. The **FILTER** parameter identifies the file or files to load. The load uses a single CSV file. Options exist to add wildcards or regular expressions to isolate different file sets.
- **EXTRACT**— Set the format to CSV files and note that there is one header line in the file. This specification skips that row when the Ociant System processes the file. Many other options exist for delimited data such as a record delimiter and field delimiter.
- **INTO ... SELECT** — Choose the target table **public.orders** and select the fields from the CSV file. The numeric index identifies each file field. Importantly, similar to other SQL syntax, the first field in the file is **\$1**, not **\$0**. Each field maps to a target column using the **as** syntax.

After you successfully create the **orders_pipeline** pipeline, execute the **START PIPELINE** command.

SQL

```
START PIPELINE orders_pipeline;
```

Step 4: Observe the Load Progress

With your pipeline running, data immediately begins to load from the S3 files that you defined. If there are many files in each file group, the load process first sorts the files into batches, partitions them for parallel processing, and assigns them to Loader Nodes.

You can check the pipeline status and progress by querying **information_schema.pipeline_status** or by executing **SHOW PIPELINE_STATUS**

SQL

```
SHOW PIPELINE_STATUS;
```

Output

SQL

database_name	pipeline_name	table_names	status	status_message
metabase	orders_pipeline	["public.orders"]	RUNNING	Started processing pipeline ord

After the status of the pipeline changes to **COMPLETED**, all data is available in the target table.

After a few seconds, the data is available for query in the **public.orders** table.

```
SELECT COUNT(*) FROM public.orders;
```

Output

SQL

```
count(*)
```

```
-----  
18760
```

You can drop the pipeline with the `DROP PIPELINE orders_pipeline;` SQL statement. Execution of this statement leaves the data in your target table, but removes metadata about the pipeline execution from the system.

Related Links

[Data Pipelines](#)

[Data Formats for Data Pipelines](#)

[Data Types for Data Pipelines](#)

[Transform Data in Data Pipelines](#)

[Manage Errors in Data Pipelines](#)

3.1.2 Data Pipeline Load of JSON Data from Kafka

A common setup for streaming data into Ocident® is to load JSON data from an Apache® Kafka® topic.

Ocident uses data pipelines to transform each document into rows in one or more different tables. The loading and transformation capabilities use a simple SQL-like syntax for transforming data. This tutorial guides you through a simple example load using a small data set in JSON format. The data in this example comes from a test set for the MetabaseSM Business Intelligence tool.

Prerequisites

This tutorial assumes that:

1. The Ocident System has network access to a Kafka broker from the SQL and Loader Nodes.
2. An Ocident System is installed and configured with an active Storage Cluster (See the [Ocident Application Configuration](#) guide).

Step 1: Create a New Database

To begin, load two example tables in a database. First, connect to a SQL Node using the [Commands Supported by the Ocident JDBC CLI Program](#). Then, execute the `CREATE DATABASE` SQL statement to create the `metabase` database.

```
CREATE DATABASE metabase;
```

Step 2: Create Table

Create the `orders` table in the new database. First, connect to that database (e.g., `connect to jdbc:ocident://sql-node:4050/m`) and then execute this `CREATE TABLE` SQL statement that specifies to create a table with these columns and a clustering index based on the `user_id` and `product_id` columns:

- `created_at` as a TimeKey® timestamp that is not nullable.
- `id`, `user_id`, and `product_id` as integers that are not nullable.
- `subtotal`, `tax`, `total`, and `discount` as floating point numbers.
- `quantity` as an integer.


```
CREATE TABLE public.orders (
  created_at TIMESTAMP TIME KEY BUCKET(30, DAY) NOT NULL,
  id INT NOT NULL,
  user_id INT NOT NULL,
  product_id INT NOT NULL,
  subtotal DOUBLE,
  tax DOUBLE,
  total DOUBLE,
  discount DOUBLE,
  quantity INT,
  CLUSTERING INDEX idx01 (user_id, product_id)
);
```

The database creates the `orders` table, and you can begin loading data.

Step 3: Create a Data Pipeline

Create data pipelines using the `CREATE PIPELINE` SQL statement. To load data, you first create a pipeline with the definition of the source, data format, and transformation rules using a SQL-like declarative syntax. Then, you execute the `START PIPELINE` command to start the load. You can observe progress and status using system tables and views.

Each Ocident pipeline defines a single data source and the target table or tables into which data loads. A data source includes the location of the source and filters on the source to define the specific data set to load. This example loads data from two data sources, where each source is located in a directory on the same S3 bucket.

First, inspect the data that you plan to load. Each document has a format similar to this example.

```
{"id": 1, "user_id": 1, "product_id": 14, "subtotal": 37.65, "tax": 2.07, "total": 39.72, "discount": null}
{"id": 2, "user_id": 1, "product_id": 123, "subtotal": 110.93, "tax": 6.1, "total": 117.03, "discount": null}
```

In this case, Ocident automatically transforms the data to the target columns using some sensible conventions. In other cases, loads require some transformation. Most transformations are identical to functions that already exist in the SQL dialect of the Ocident System.

Create a pipeline named `orders_pipeline` for the orders data set from your database connection prompt. Specify the Kafka source with broker address `192.168.0.1:9092` (replace this example Kafka broker address with your address) and topic `orders`. Load the data into the `public.orders` table. The `SELECT` part of the SQL statement maps the JSON fields to the target columns in the created table.

SQL

```
CREATE PIPELINE orders_pipeline
SOURCE
  KAFKA
    BOOTSTRAP_SERVERS '192.168.0.1:9092'
    TOPIC 'orders'
EXTRACT
  FORMAT json
INTO public.orders
SELECT
  $id as id,
  $user_id as user_id,
  $product_id as product_id,
  $subtotal as subtotal,
  $tax as tax,
  $total as total,
  $discount as discount,
```

```
$created_at as created_at,
$quantity as quantity;
```

The pipeline has three main sections:

- **SOURCE** — Loads data from Kafka. Specify the address of the bootstrap servers and topic. Options exist to set Kafka consumer configurations.
- **EXTRACT** — Sets the format to JSON.
- **INTO ... SELECT** — Targets the `public.orders` table and selects the chosen fields from the JSON records. In this case, all data is available at the top level of the JSON records, so the example references the fields by the attribute name (e.g., `$id`, `$user_id`, etc.). For nested data, reference the nested fields using dot notation (e.g., `$order.user.first_name`). Each field maps to a target column using the `as` syntax.

After you successfully create this pipeline, execute the `START PIPELINE` SQL statement.

SQL

```
START PIPELINE orders_pipeline;
```

Step 4: Confirm that Loading is Operating Correctly

With the pipeline in place and running, data immediately begins loading off of the Kafka topics that are configured in the pipeline. If you do not have data in the Kafka topics yet, now is a good time to start producing data into the topics.

Produce Test Data into Kafka

For test purposes, `kafka-cat` is a helpful utility that makes it easy to produce records into a topic. For example, if you have a file of sample data `orders.jsonl` in a JSONL format (newline-delimited JSON records), you can run this command to send those records into your Kafka broker. Specify your broker IP address instead of `<broker_ip_address>` and your topic name `<topic_name>`.

```
kafka-cat -b <broker_ip_address>:9092 -t <topic_name> -T -P -l orders.jsonl
```

Save an example document to your file system to use for this test. For this example, you can download an example file from https://ocient-examples.s3.amazonaws.com/metabase_samples/jsonl/orders.jsonl and save it to `~/orders.jsonl`.

Assuming the broker is running at the IP address `10.0.0.3`, send data into the `orders` topic defined in the pipeline definition by executing this command.

```
kafka-cat -b 10.0.0.3:9092 -t orders -T -P -l orders.jsonl
```

This command pushes the entire JSONL file of messages into Kafka with one record per line. As these records are produced into Kafka, the running pipeline begins to load them into Ocint.

Step 5: Observe the Load Progress

With your pipeline running, data immediately begins to load from the Kafka topic. The pipeline creates parallel Kafka consumers for each partition. If there are more partitions than processing cores available, the pipeline automatically handles spreading consumers across available processing cores and Loader Nodes.

You can check the pipeline status and progress by querying `information_schema.pipeline_status` or executing the `SHOW PIPELINE_STATUS` SQL statement.

SQL

```
SHOW PIPELINE_STATUS;
```

Output

SQL

database_name	pipeline_name	table_names	status	status_message
metabase	orders_pipeline	["public.orders"]	RUNNING	Started processing pipeline ord

After a few seconds, the data is available for query in `public.orders`.

SQL

```
SELECT COUNT(*) FROM public.orders;
```

Output

SQL

```
count(*)
```

```
-----
18760
```

Unlike a batch file load, Kafka pipelines run continuously, so they never change to a status of `COMPLETED`. To examine progress, you can use the information schema and system catalog tables. Key details are in `information_schema.pipelines`, `information_schema.pipeline_status`, and `sys.pipeline_partitions`.

For example, this statement shows the status and key metrics for duration, loaded records, and failed records.

SQL

```
SELECT
    pipeline_name,
    status,
    status_message,
    duration_seconds,
    records_loaded,
    records_failed
FROM
    information_schema.pipeline_status;
```

Output

SQL

pipeline_name	status	status_message	duration_seconds	records_loaded
orders_pipeline	RUNNING	Started processing pipeline orders_pipeline	10.089936	18760

In this example, there are no errors. The `sys.pipeline_errors` system catalog table captures any errors that occur during the pipeline process.

You can drop the pipeline with the `DROP PIPELINE orders_pipeline;` SQL statement. Execution of this statement leaves the data in your target table, but removes metadata about the pipeline execution from the system.

Related Links

[Data Pipelines](#)

[Data Formats for Data Pipelines](#)

[Data Types for Data Pipelines](#)

[Transform Data in Data Pipelines](#)

[Manage Errors in Data Pipelines](#)

3.1.3 Data Pipeline Load of Parquet Data from S3

A common setup for loading files in a batch into the Ocient® System is to load from a bucket on Amazon® Web ServicesSM (AWSSM) S3 with time-partitioned data. Often, you must perform a batch load repeatedly to load new files.

Ocient uses data pipelines to transform each document into rows in one or more different tables. The loading and transformation capabilities use a simple SQL-like syntax for transforming data. This tutorial guides you through a simple example load using a small data set in Apache® ParquetTM format. The data in this example comes from a test set for the MetabaseSM Business Intelligence tool.

Parquet Loading Recommendations

Follow this set of recommendations for an optimal loading experience of Parquet files.

Prerequisites

This tutorial assumes that:

1. The Ocient System has network access to S3 from the Loader Nodes.
2. An Ocient System is installed and configured with an active Storage Cluster. For details, see [Ocient Application Configuration](#).

Step 1: Create a New Database.

To begin, load two example tables in a database. First, connect to a SQL Node using the [Commands Supported by the Ocient JDBC CLI Program](#). Then, execute the `CREATE DATABASE` SQL statement to create the `metabase` database.

```
CREATE DATABASE metabase;
```

Step 2: Create a New Table in the Database.

Create the `orders` table in the new database. First, connect to that database (e.g., `connect to jdbc:ocient://sql-node:4050/m`) and then execute this `CREATE TABLE` SQL statement that specifies to create a table with these columns and a clustering index based on the `user_id` and `product_id` columns:

- `created_at` as a TimeKey® timestamp that is not nullable.
- `id`, `user_id`, and `product_id` as integers that are not nullable.
- `subtotal`, `tax`, `total`, and `discount` as floating point numbers.
- `quantity` as an integer.

```
CREATE TABLE public.orders (
  created_at TIMESTAMP TIME KEY BUCKET(30, DAY) NOT NULL,
  id INT NOT NULL,
  user_id INT NOT NULL,
  product_id INT NOT NULL,
  subtotal DOUBLE,
  tax DOUBLE,
  total DOUBLE,
  discount DOUBLE,
  quantity INT,
  CLUSTERING INDEX idx01 (user_id, product_id)
);
```

The database creates the `orders` table, and you can begin loading data.

Step 3: Preview and Create a Data Pipeline.

Create data pipelines using the `CREATE PIPELINE SQL` statement. To load data, you first create a pipeline with the definition of the source, data format, and transformation rules using a SQL-like declarative syntax. Then, you execute the `START PIPELINE SQL` statement to start the load. You can observe progress and status using system catalog tables and views.

Each Ocient pipeline defines a single data source and the target table or tables into which data loads. A data source includes the location of the source and filters on the source to define the specific data set to load. This tutorial loads data from two data sources, where each source is located in a directory on the same S3 bucket.

First, inspect the data that you plan to load. Each document has a JSON format similar to this example.

```
{ "id": 1, "user_id": 1, "product_id": 14, "subtotal": 37.65, "tax": 2.07, "total": 39.72, "discount": null }
{ "id": 2, "user_id": 1, "product_id": 123, "subtotal": 110.93, "tax": 6.1, "total": 117.03, "discount": null }
```

Inspecting the file with Pandas, you can see this schema with details.

Parquet Schema Description

#	Column	Non-Null Count	Dtype
0	id	18760 non-null	int64
1	user_id	18760 non-null	int64
2	product_id	18760 non-null	int64
3	subtotal	18760 non-null	float64
4	tax	18760 non-null	float64
5	total	18760 non-null	float64
6	discount	1915 non-null	float64
7	created_at	18760 non-null	datetime64[ns, UTC]
8	quantity	18760 non-null	int64

In this case, Ocient automatically transforms the data to the target columns using some sensible conventions. In other cases, loads require some transformation. Most transformations are identical to functions that already exist in the SQL syntax of the Ocient System.

Prior to creating a pipeline, you can use the `PREVIEW PIPELINE SQL` statement to create your pipeline iteratively. This statement returns a result set that shows the final values that would be loaded but does not load the data into the target table.

Preview the `orders_pipeline` pipeline for the `orders` data set from your database connection prompt. Use the S3 data source with endpoint `https://s3.us-east-1.amazonaws.com`, bucket `ocient-examples`, and filter `metabase_samples/parquet/orders.parquet`. Specify the `parquet` format. Load the data into the `public.orders` table. The `SELECT` part of the SQL statement maps the fields in the Parquet file to the target columns in the created table. In this example, limit the result set to the first five records in the data source.

```
PREVIEW PIPELINE orders_pipeline
SOURCE
  S3
    ENDPOINT 'https://s3.us-east-1.amazonaws.com'
    BUCKET 'ocient-examples'
    FILTER 'metabase_samples/parquet/orders.parquet'
  LIMIT 5
EXTRACT
  FORMAT parquet
INTO public.orders
SELECT
  $id as id,
  $user_id as user_id,
  $product_id as product_id,
  $subtotal as subtotal,
  $tax as tax,
```

```
$total as total,
$discount as discount,
$created_at as created_at,
$quantity as quantity;
```

Output

SQL

id	user_id	product_id	subtotal	tax	total	discount
1	1	14	37.65	2.07	39.72	NULL
8	1	123	110.93	6.1	117.03	NULL
8	1	105	52.72	2.9	49.2	6.42
8	1	94	109.22	6.01	115.23	NULL
8	1	132	127.88	7.03	134.91	NULL

Fetches 5 rows

With this preview, you can confirm that the results of the pipeline match your requirements. If there is an issue, you can update the statement and run the PREVIEW PIPELINE statement again until it meets your needs.

Next, create the pipeline named `orders_pipeline` for the `orders` data set from your database connection prompt. The pipeline has three main sections:

- **SOURCE** — Loads data from S3. Set the S3 endpoint, bucket name, and filter for the Parquet files.
- **EXTRACT** — Sets the format to Parquet.
- **INTO ... SELECT** — Targets the `public.orders` table and selects the chosen fields from the Parquet records. In this case, all data is available at the top level of the Parquet records, so the example references the fields by the attribute name (e.g., `$id`, `$user_id`, etc.). For nested data, reference the nested fields using dot notation (e.g., `$order.user.first_name`). Each field maps to a target column using the `as` syntax.

SQL

```
CREATE PIPELINE orders_pipeline
SOURCE
  S3
  ENDPOINT 'https://s3.us-east-1.amazonaws.com'
  BUCKET 'ocient-examples'
  FILTER 'metabase_samples/parquet/orders.parquet'
EXTRACT
  FORMAT parquet
INTO public.orders
SELECT
  $id as id,
  $user_id as user_id,
  $product_id as product_id,
  $subtotal as subtotal,
  $tax as tax,
  $total as total,
  $discount as discount,
  $created_at as created_at,
  $quantity as quantity;
```

After you successfully create this pipeline, execute the `START PIPELINE` SQL statement to start the load.

SQL

```
START PIPELINE orders_pipeline;
```

Step 4: Observe the Load Progress.

With your pipeline running, data begins to load immediately from the S3 files that you defined. If there are many files in each file group, the load process first sorts the files into batches, partitions them for parallel processing, and assigns them to Loader Nodes.

You can check the pipeline status and progress by querying the `information_schema.pipeline_status` system catalog table or executing the `SHOW PIPELINE_STATUS` SQL statement.

SQL

```
SHOW PIPELINE_STATUS;
```

Output

SQL

database_name	pipeline_name	table_names	status	status_message
metabase	orders_pipeline	["public.orders"]	RUNNING	Started processing pipeline ord

After the status of the pipeline changes to `COMPLETED`, all data is available in the target table.

After a few seconds, the data is available for query in the `public.orders` table.

SQL

```
SELECT COUNT(*) FROM public.orders;
```

Output

SQL

```
count(*)
-----
18760
```

You can drop the pipeline with the `DROP PIPELINE orders_pipeline;` SQL statement. Execution of this statement leaves the data in your target table, but removes metadata about the pipeline execution from the system.

Related Links

[Data Pipelines](#)

[Data Formats for Data Pipelines](#)

[Data Types for Data Pipelines](#)

[Transform Data in Data Pipelines](#)

[Manage Errors in Data Pipelines](#)

3.2 Data Types for Data Pipelines**Source Fields**

When you load data using a data pipeline, you reference extracted data using a special syntax named a source field reference. These source field references can behave differently depending on the data format. It is important to understand how to reference source field data and also how Ociant® assigns the type of data you reference.

Source Field References

You reference source fields in a pipeline using a named reference such as `$my_field` or a numeric field reference such as `$1`. For some data formats like JSON and Apache® Parquet™, Ociant supports nested data access such as `$my_field.sub_field`.

Learn more about what source field references are supported for each data format in [Data Formats for Data Pipelines](#).

Source Field Data Types

The Ociant System treats references to source fields (e.g., `$my_field`) differently depending on the data format `FORMAT` specified in the data pipeline.

Text-Based Data Formats

For text-based formats such as `DELIMITED` and `JSON`, the Ociant System treats source fields as `VARCHAR`. For `JSON` data, the system treats source fields strictly as `VARCHAR` data even if the field value is represented logically in `JSON` as Boolean, integer, `NULL`, or double. Learn more about text-based formats in [Data Formats for Data Pipelines](#) and [Data Formats for Data Pipelines](#).

Binary-Based Data Format

For the binary-based format `BINARY`, Ociant treats source fields as the `BINARY` data type. Learn more about binary formats in [Data Formats for Data Pipelines](#).

Parquet-Based Data Format

For the Parquet-based format `PARQUET`, Ociant encodes information about the data type of the fields during the load.

Supported Data Types

The data pipeline functionality supports all column data types except:

- Tuples of Tuples
- Tuples of Arrays

`ST_POINT` is not supported in a 3-coordinate format.

When you load data into a column type, in many cases, Ociant automatically casts the source data to the target column. However, in some cases, you might need to cast data to the target column type explicitly. Learn more about supported casting functions in [Transform Data in Data Pipelines](#).

Automatic Conversion and Casting

Loading pipelines apply automatic casting to the final values of your transformations and source fields where the data type does not match the target column type. The casting function that the Ociant System applies varies based on the SQL type of the data that results from your transformation and the SQL type of the target column.

The system applies casting automatically in cases to resolve conflicts between data types:

- Promotion of a type to expand to a wider data type (e.g., `SMALLINT` to `BIGINT`)
- Implicit casting from one family of data type to another (e.g., `VARCHAR` to `TIMESTAMP`)
 - Ociant supports implicit casting of all simple data types to `VARCHAR`.

Auto-casting in Parquet does not support the automatic conversion to `VARCHAR` columns. You must explicitly cast data to the `CHAR` data type when you convert Parquet data that is not string data to a `VARCHAR` column or `VARCHAR` function argument.

Automatic casting does not apply in some situations where data loss might occur:

- Implicit downcasting to a narrower data type (e.g., `BIGINT` to `SMALLINT`)

Where automatic casting is not supported, you can explicitly cast data or use built-in transformation functions according to your requirements. See [Transform Data in Data Pipelines](#) for more details.

Implicit Casting Support List

VARCHAR Source Data

The Ocient System treats the source field reference from JSON and DELIMITED formatted data as VARCHAR regardless of the logical type in the source.

Ocient supports the implicit casting of VARCHAR to all simple data types. These types are not supported:

- Geospatial types
- Arrays
- Tuples
- Matrixes

BINARY Source Data

The system treats the source field reference from BINARY formatted data as raw bytes with automatic casting functions tailored to the BINARY data type.

Ocient supports implicit casting of BINARY to:

- TINYINT, SMALLINT, INT, BIGINT
- FLOAT, DOUBLE
- TIMESTAMP, DATE, TIME (as VARCHAR data)

For BINARY or VARCHAR to BOOL:

- You can represent true as: 't', 'true', 'yes', '1', 'y', 'on'
- You can represent false as: 'f', 'false', 'no', '0', 'n', 'off'

For BINARY or VARCHAR to TIMESTAMP, DATE, and TIME, the automatic casting assumes that the character data has the format:

- TIMESTAMP — YYYY-MM-DD HH:mm:ss[.SSSSSSSSS]
- DATE — YYYY-MM-DD
- TIME — HH:mm:ss

Other Implicit Casts or Downcasts

The system also supports some other implicit casts that are not type promotions:

- TIMESTAMP data implicitly downcasts to DATE and TIME columns, truncating the unused portion.
 - ≤ This option is available in version 25 or later.
- All simple types implicitly cast to VARCHAR columns except:
 - HASH or BINARY
 - DECIMAL
 - IP or IPV4
 - ARRAY, TUPLE, MATRIX

For geospatial data types, see [Load Geospatial Data in Data Pipelines](#).

Data Type Promotion List

The data type promotion list describes how the Ocient System can automatically promote the specified type to a target column type. The system achieves the promotion using standard casting functions (e.g., BIGINT(SMALLINT)). See [Data Types for Data Pipelines](#) for implicit casting, for example casting to VARCHAR.

Source Data Type	Automatic Casting Precedence for Target Column Type
BYTE or TINYINT	BYTE` `TINYINT` `SMALLINT` `INT` `BIGINT` `FLOAT` `DOUBLE
SMALLINT	SMALLINT` `INT` `BIGINT` `FLOAT` `DOUBLE

Source Data Type	Automatic Casting Precedence for Target Column Type
INT	INT``BIGINT``FLOAT``DOUBLE
BIGINT	BIGINT``DOUBLE``TIMESTAMP``TIME
FLOAT	FLOAT``DOUBLE
DOUBLE	DOUBLE
DECIMAL	DECIMAL
BOOL	BOOLEAN
UUID	UUID
DATE	DATE``TIMESTAMP
TIME	TIME
TIMESTAMP	TIMESTAMPSeeData Types for Data Pipelinesfor supported downcasting.
IPV4	IPV4
IP	IP
BINARY	BINARYSeeData Types for Data Pipelinesfor supported cross-casting.
ARRAY	ARRAY
TUPLE	TUPLE
VARCHAR	VARCHARSeeData Types for Data Pipelinesfor supported cross-casting.
ST_POINT	ST_POINT,ST_LINestring,ST_POLYGON
ST_LINestring	ST_LINestring,ST_POLYGON
ST_POLYGON	ST_POLYGON

- BIGINT to TIME automatic casting assumes a time value in milliseconds.
- BIGINT to TIMESTAMP automatic casting assumes milliseconds after the epoch. (January 1, 1970 at midnight UTC)
- For complex types, the system applies the type precedence for automatic casting to the elements in the type.
- Automatic casting of integral types to DOUBLE and FLOAT are potentially lossy due to limits in precision of floating point representations.
 - $\leq$ This option is available in version 25 or later.

The system treats BYTE and TINYINT aliases identically.

Related Links

[Load Data](#)

[Data Formats for Data Pipelines](#)

[Data Pipelines](#)

[Load Metadata Values in Data Pipelines](#)

3.3 Data Formats for Data Pipelines

Loading in Ocident® differs in subtle ways that depend on the data format of the source. Loading runs with strict interpretation of source data to allow pipelines to achieve maximum performance. For text-based formats like JSON and DELIMITED, the Ocident System performs no preemptive casting on the data when using a source field selector.

For example, with the JSON string { "my_field": 1234 }, the selector \$my_field returns the string "1234" not the integer 1234.

When you use transformation functions, keep in mind that the Ocient System treats all data in the JSON and DELIMITED formats as text data.

While the Ocient System sends data you select to a final target column, the system automatically casts the data in the final step to ensure that the data is compatible with the target column type. See [Data Types for Data Pipelines](#) for supported automatic conversion rules.

Format-specific differences also appear in the pipelines.

Load Delimited and CSV Data

When you load data from delimited or CSV files, the Ocient System tokenizes the data during loading. The system detects records and fields in the input data during pipeline execution. You can reference fields and use them in combination with transformation functions before the system stores values in the column of a target table.

Referencing fields of the source data for the formats happens by using a field index. The index is a number that follows the dollar sign \$. To maintain consistency with SQL array semantics, the field indexes start at 1.

Reference the first field of tokenized source records for the DELIMITED and CSV formats as \$1.

For the BINARY format, \$0 represents the entire record. In this case, you must specify \$0 in combination with the SUBSTRING function to extract specific bytes from the source data.

For a complete list of supported options for DELIMITED and CSV data format, see [Data Pipelines](#).

Delimited Loading Example

Use this example delimited data.

```
iphone|60607|viewed|502|293.99|[shopping,news]
```

For this example row, this table shows the field references for each value in the row.

Field Reference	Value
\$1	iphone
\$2	60607
\$3	viewed
\$4	502
\$5	293.99
\$6[]	['shopping','news']

To load this data in a pipeline with the DELIMITED data format, this CREATE PIPELINE statement specifies the | character for the field delimiter. This statement loads data into Amazon® Web ServicesSM (AWSSM) S3. The SELECT statement uses fields 1, 2, 3, 5, and 6 of the source data. The statement specifies that the system should not load field 4 to the target table. Field 6 is an array of data matching the default array settings for delimited data. You can indicate this with the array brackets like \$6[] to load into a CHAR[] typed column.

The outer casting functions in this example are optional and shown for completeness. If they are omitted, the pipeline automatically casts the source fields to the target column type.

```
CREATE PIPELINE
SOURCE s3
...
FORMAT delimited
  FIELD_DELIMITERS ['|']
...
SELECT
  CHAR($1) as device_model,
```

```
INT($2) as zip,
INT($3) as amount,
DOUBLE($5) as price,
CHAR[]($6[]) as categories
```

Load JSON Data

The data pipeline syntax enables the load of JSON data, including nested scalars, arrays, and points `ST_POINT`.

Strict Loading and Transformations

When you use transformation functions, remember that the Ocint System treats all data in JSON and DELIMITED format as text data, not the logical data type.

For example, if you specify the JSON string `{ "my_timestamp": 1709208000000 }`, the selector `$my_timestamp` returns the string `"1709208000000"` and not the integer `1709208000000`.

As a result, if you cast this data into a timestamp column, such as `TIMESTAMP($my_timestamp) as created_at`, the Ocint System returns an error. The conversion fails because the cast function assumes you are specifying `TIMESTAMP(VARCHAR)`, which assumes a format like `YYYY-MM-DD HHss[.SSSSSSSS]`.

To correct this issue, cast the value explicitly to make use of the `TIMESTAMP(BIGINT)` function that treats the argument as milliseconds after the epoch as in `TIMESTAMP(BIGINT($my_timestamp)) as created_at`.

Supported JSON Selectors

JSON selectors consist of `$` followed by a dot-separated list of JSON keys. If a key refers to an array, it is followed by a set of brackets `[]` to correspond to its dimensionality. If the square brackets contain an index, like `[1]`, then the selector refers to an array element.

The Ocint System treats JSON selectors as lowercase. To use case-sensitive selectors, you must enclose the selector in double quotation marks. For example, `"testSelector"`. With case-sensitive selectors having multiple JSON keys, each key needs double quotation marks. For example, `"testData"."Responses"."SuccessResponse"`.

For special characters (any identifier that starts with any character other than a letter or contains any character that is not a letter, number, or an underscore) or reserved SQL keywords (such as `SELECT`), you must enclose such selectors in double quotation marks. For example, if you have a JSON document `{ "test-field": 123 }`, then the selector for the query should be `"test-field"`.

The Ocint System does not support identifiers with a backslash as the last character in the key name.

This table shows the selector and provides its description. The cells in the last column of the table show an example for each selector. First, the cell shows example data in JSON format. Then, the cell shows the use of the selector and its output after the arrow.

Selector	Description	Examples
<code>\$a</code>	Scalar — Refers to key <code>a</code> at the top of a JSON document, where the value of <code>a</code> is a scalar.	<code>> { "first_name": "Chris" }\$first_name -> "Chris"</code>
<code>\$a[]</code>	Array — Refers to key <code>a</code> at the top of a JSON document, where the value of <code>a</code> is a one-dimensional array.	<code>> { "names": ["Chris", "Joe"] }\$names[] -> ["Chris", "Joe"]</code>

Selector	Description	Examples
<code>\$a[1]</code>	Array Element Selection — Selects a single element from the array a at the top of a JSON document, where the value of a is a one-dimensional array. Also works on multi-dimensional arrays. Indexes start at 1.	<pre>> { "names": ["Chris", "Joe"] }\$names[2] -> "Joe"</pre>
<code>\$a[] []</code>	Multi-dimensional Array — Refers to key a at the top of a JSON document, where the value of a is a two-dimensional array.	<pre>> { "coordinates": [[87.31, 18.25],[65.22, 19.41]] }\$coordinates[] [] -> [[87.31,18.25],[65.22,19.41]]</pre>
<code>\$a.b.c</code>	Nested Selector — Drills into the value at key a , then value at key b , and refers to the value at key c , where none of the values are arrays.	<pre>> { "user": { "name": "Chris" } }\$user.name -> "Chris"</pre>
<code>\$a[] .b</code>	Array Projection — A projection applies a JSON selector on all elements in an array, returning an array. Array brackets in the notation indicate which items are the arrays. The a[] is an array. The b selector is applied to each element in the array a .	<pre>> { "users": [{ "name": "Chris", "orders": [{ "order_id": 1, "subtotal": 19.29 }, { "order_id": 2, "subtotal": 16.71 }] }, { "name": "Joe", "orders": [{ "order_id": 3, "subtotal": 17.29 }, { "order_id": 4, "subtotal": 22.49 }] }] }\$users[] .name -> ["Chris", "Joe"]</pre>
<code>\$a.b[] .c.d[]</code>	Multi-level Array Projection — Applies selectors to each element in the specified arrays and allows multiple levels of objects and arrays for the selection of an N-dimensional array of values. The values of b and d are arrays, resulting in two-dimensional arrays. The selector c.d[] is projected into the elements in the array b .	<pre>> { "users": [{ "name": "Chris", "orders": [{ "order_id": 1, "subtotal": 19.29 }, { "order_id": 2, "subtotal": 16.71 }] }, { "name": "Joe", "orders": [{ "order_id": 3, "subtotal": 17.29 }, { "order_id": 4, "subtotal": 22.49 }] }] }\$users[] .orders[] .order_id -> [[1,2],[3,4]]</pre>
<code>\$a.b[1].c.d[]</code>	Array Slice — Selects the first element from the array b and drills into c and then the array d . JSON array selectors have an index that starts at 0.	<pre>> { "users": [{ "name": "Chris", "orders": [{ "order_id": 1, "subtotal": 19.29 }, { "order_id": 2, "subtotal": 16.71 }] }, { "name": "Joe", "orders": [{ "order_id": 3, "subtotal": 17.29 }, { "order_id": 4, "subtotal": 22.49 }] }] }\$users[1].orders[] .subtotal -> [17.29,22.49]</pre>
<code>\$a.{b,c}</code>	Tuple Selector — Selects the values from an object as a tuple. You must cast the values to an appropriate TUPLE data type.	<pre>> { "user": { "first_name": "John", "last_name": "Doe" }}\$user.{first_name,last_name} -> <<"John","Doe">></pre>
<code>\$a[] .{b,c}</code>	Array of Tuples Selector — Selects the values from an array of objects as an array of tuples. You must cast the values to an appropriate TUPLE[] data type.	<pre>> { "users": [{ "first_name": "John", "last_name": "Doe" }, { "first_name": "Steve", "last_name": "Smith" }]}\$users[] .{first_name,last_name} -> [<<"John","Doe">>,<<"Steve","Smith">>]</pre>

Selector	Description	Examples
<code>\$a[_]</code>	Flatten Array Selector — Selects the values of an array and flattens the dimension indicated by the underscore by one level.	<code>> { a: { b: [{c: [1,2,3] }, {c: [4,5,6] }] } } \$a.b[_].c -> [1,2,3,4,5,6]</code>
<code>\$a[!]</code>	Compact Array Selector — Selects the values of an array and removes any NULL values from the array at this level in the array.	<code>> { a: { b: { c: [1,null,2,null,3] } } } \$a.b.c[!] -> [1,2,3]</code>

For more examples of using JSON selectors in data pipelines, see [JSON Selectors Examples in Data Pipelines](#).

NULL and Empty Handling for JSON Scalars

The Ocint System handles all JSON NULL, empty, and missing values in the same way. The system loads these values as NULL. These values fail to load into non-nullable columns.

Provide an explicit default in the pipeline using `IF_NULL` or `COALESCE` or use the `COLUMN_DEFAULT_IF_NULL` option to accept the configured column default instead of attempting to load NULL values.

NULL and Empty Handling for JSON Arrays

The Ocint System handles NULL, empty, and missing values the same way for arrays as for scalars. The system converts a value that is NULL, empty, or missing to NULL and loads it as NULL.

Provide an explicit default in the pipeline or use the `COLUMN_DEFAULT_IF_NULL` option to accept the configured column default instead of attempting to load NULL.

NULL and Empty Handling for JSON Tuples

All the rules for handling NULL, empty, and missing elements that apply to scalars and arrays also apply to tuples. If any part of the selector is NULL, empty, or missing, data pipeline loading converts that value to NULL.

Additionally, because you can apply functions to tuple elements (and not array elements), you can use the `NULL_IF` function to convert a tuple element to NULL. For example, `tuple<<char,varchar>>($a.name, NULL_IF($a.hometown, 'N/A'))` indicates to the pipeline that the string 'N/A' signifies NULL for the `hometown` element but not for the `name` element.

Load Binary Data

The Ocint System loads the binary data format using a fixed record length to split a binary stream into chunks that represent records. Each record is available in the `SELECT` portion of a pipeline definition using a special binary extract syntax `$"[5,8]"`. This operates similarly to a substring function, beginning at byte 5 and taking 8 bytes from that location. The starting index is a 1-based offset, consistent with other SQL arrays and offsets. You can use this syntax to select specific bytes within a record to parse together as a unit.

Binary Selector

Selector	Description	Examples
<code>\$"<start_index>,<length_in_bytes>"</code>	Extract Bytes — Extracts the bytes beginning at the <code><start_index></code> and returning <code><length_in_bytes></code> total bytes to extract from the start index. <code><start_index></code> is 1-based.	<code>> abc123.45xyz\$"[4,6]" -> "123.45"</code> This example illustrates the operation using ascii characters, but the Binary Selector returns BINARY data, byte arrays, that may be used in special ways by fixed width processing functions described in Transform Data in Data Pipelines .

Example

The Binary Selector takes 8 bytes starting at offset 11 on the fixed-width binary record. Consistent with SQL functions in the Ocint System, the first argument value 11 is the 1-based offset into the byte array.

SQL

```
$"[11, 8]"
```

The Binary Selector returns BINARY data, not VARCHAR.

Special BINARY transformation functions can operate on this BINARY data. However, if you cast data to the VARCHAR type by using CHAR(), then functions like INT operate on this data as VARCHAR data, not binary data.

When you load binary data into VARCHAR columns, the Ocint System automatically converts from binary to character data using the configured CHARSET_NAME before final loading.

The Ocint System supports special transformation functions that operate uniquely on binary data. With these functions, you can convert binary representations from mainframe systems such as packed decimals, zoned decimals, big and little endian integers (signed and unsigned), and floating point values. For more details, see [Transform Data in Data Pipelines](#).

For a complete list of supported options for DELIMITED and CSV data formats, see [Data Pipelines](#).

Binary Loading Example

If each record in your fixed-width binary schema includes these fields, you can use the SUBSTRING function and the transforms shown in this example.

Column	Start Index	Record Length	Source Data Type	Ocint Target Data Type
first_name	1	20	Character	VARCHAR
last_name	21	20	Character	VARCHAR
age	41	4	Big Endian signed 4-byte integer	INT
total_spent	45	10	Packed Decimal	DECIMAL(10,2)
user_id	55	8	Little Endian unsigned 8-byte long	BIGINT

Each record includes 62 bytes, so the record length RECORD_LENGTH is 62. The encoding of this file is CP500 instead of the default IBM1047 code page. The CREATE PIPELINE SQL statement specifies this encoding.

```
CREATE PIPELINE binary_users_pipeline
SOURCE
  S3 ...
FORMAT
  BINARY
```



```

RECORD_LENGTH 62
CHARSET_NAME 'cp500'
INTO public.users
SELECT
  $"[1, 20]" as first_name,
  $"[21, 20]" as last_name,
  INT($"[41, 4]") as age,
  DECIMAL($"[45, 10]", 'packed', 2) as total_spent,
  BIGINT($"[55, 8]", 'unsigned', 'little') as user_id;

```

This SQL statement:

- Uses the **BINARY SELECTOR** to extract names and load them into the respective columns. The Ocient System automatically decodes the values using **cp500** and loads them into a **VARCHAR** column. An explicit cast such as **CHAR(\$"[1, 20]") as first_name** works equivalently.
- Indicates the extraction of four bytes that represent age from bytes 41-44. The statement instructs the casting of these bytes as an integer **INT**. This function uses the default endianness (big) and treats the bytes as signed. Unsigned values can overflow target columns because integral types are all signed.
- Extracts the 10 bytes for **total_spent** using the **BINARY SELECTOR**, and converts the values using the **packed** decimal option for the **DECIMAL** cast. The casting requires specifying the number of decimal points in the source data. In this case, there are 2 decimal points, which matches the target column.
- Extracts the 8 bytes that represent **user_id** using the **BINARY SELECTOR** and casts these bytes to a **BIGINT** while interpreting the bytes as unsigned with the little endian representation.

Load Parquet Data

The data pipeline functionality enables loading Apache® Parquet™ files with this configuration.

Use selectors as you do when loading JSON data to specify data to load.

For details, see [Parquet Selectors Examples in Data Pipelines](#).

When you use the **FORMAT PARQUET** option with an AWS S3 source, the **ENDPOINT** option is required in the **CREATE PIPELINE SQL** statement.

Auto-casting in Parquet does not support the automatic conversion to **VARCHAR** columns. You must explicitly cast data to the **CHAR** data type when you convert Parquet data that is not string data to a **VARCHAR** column or **VARCHAR** function argument.

Schema Evolution

The Ocient System supports schema evolution when you load a set of Parquet files. Specifically, if the pipeline selects a set of Parquet files where an individual file might have more or fewer columns than another, the system attempts to merge those schemas together to support loading without requiring you to create the pipeline again.

For example, the **test_table** table has three columns.

```

CREATE TABLE test_table (
  col_a INT NULL,
  col_b VARCHAR NULL,
  col_c VARCHAR NULL
);

```

You have two Parquet files with these schemas:

```

message file1_schema {
  OPTIONAL INT32 col_a;

```

```

    OPTIONAL BYTE_ARRAY col_b (UTF8);
}

message file2_schema {
    OPTIONAL BYTE_ARRAY col_b (UTF8);
    OPTIONAL BYTE_ARRAY col_c (UTF8);
}

```

However, you must specify how you want to handle the schema evolution within the `EXTRACT SQL` statement. You can choose to sample the first file only for its schema or sample the entire data set to merge the schemas together. This DDL statement samples on one file.

```

EXTRACT
  FORMAT parquet
  SCHEMA (infer_from sample_file)

```

The disadvantage is that sampling multiple files can potentially take a long time (scaling with the number of files in the data set) when you execute the `CREATE PIPELINE` and `START PIPELINE SQL` statements. If you know that all of the Parquet files have the same schema, use this syntax.

The Ociant System does not support the case where a column within the schema changes type. For example, if `col_a` is an `INT` type in one file and a `VARCHAR` type in another.

The default behavior of schema evolution is inferring the schema from all files. Use this syntax to infer from all files.

```

EXTRACT
  FORMAT parquet
  SCHEMA (infer_from all_files)

```

Supported Data Types in Parquet

Parquet data types are separated into primitive and logical types. The Ociant System converts these types to Ociant SQL types. See these tables for the respective conversions.

Parquet Primitive Type	Ociant SQL Type
BOOLEAN	BOOLEAN
INT32	INT
INT64	BIGINT
INT96	TIMESTAMP
FLOAT	FLOAT
DOUBLE	DOUBLE
BYTE_ARRAY	VARCHAR
FIXED_LEN_BYTE_ARRAY	VARCHAR

Parquet Logical Types	Ociant SQL Type
STRING	VARCHAR
UTF8	VARCHAR
ENUM	VARCHAR
UUID	UUID
INT8	TINYINT
INT16	SMALLINT
INT32	INT
INT64	BIGINT
UINT8	SMALLINT
UINT16	INT

Parquet Logical Types	Ocient SQL Type
UINT32	BIGINT
UINT64	BIGINT
DECIMAL	DECIMAL
DATE	DATE
TIME	TIME
TIME_MILLIS	TIME
TIME_MICROS	TIME
TIMESTAMP	TIMESTAMP
TIMESTAMP_MILLIS	TIMESTAMP
TIMESTAMP_MICROS	TIMESTAMP
DURATION	BIGINT
JSON	VARCHAR
BSON	VARCHAR

- The INTERVAL data type is not supported.
- The UINT64 data type can overflow the BIGINT conversion.
- The DURATION data type conversion to BIGINT preserves the underlying units. For example, the number of microseconds stays as microseconds in the BIGINT data type.

Further, Parquet contains nested types that the Ociant System also converts to SQL types, as shown in this table.

Parquet Nested Types	Ocient SQL Type
LIST	TYPE[]
TUPLE	TUPLE

Parquet Partition Keys

With Parquet, you can load partition keys from a partitioned Parquet file using the file path structure. Use the filename metadata selector along with transformation functions.

Assume files with these file paths.

```
s3://aaa.bbb.ccc/bigdata/eee/abc/def/dt=2021-11-24/file.parquet
.../dt=2021-11-25/file.parquet
.../dt=2021-11-26/file.parquet
...
```

Construct the selector using this code to create a DATE field.

```
TO_DATE(SUBSTRING(METADATA('filename'), 49, 10)) AS partition_key_date_column
```

For details about transformation functions, see [Transform Data in Data Pipelines](#).

Parquet Example

Create a data pipeline that loads Parquet files using an AWS S3 bucket. Specify the bucket, endpoint, access key identifier, secret access key, and filter options to find all Parquet files in the specified file path. Use the `parquet_base_table` table to store the loaded data. Retrieve integer, text, floating point, double, integer, JSON, and BSON fields.

```
CREATE PIPELINE testpipeline
SOURCE S3
  BUCKET 'testbucket'
  ENDPOINT 'https://endpoint'
```

```

ACCESS_KEY_ID ''
SECRET_ACCESS_KEY ''
FILTER_GLOB '/data/*/2024/*.parquet'
EXTRACT
  FORMAT PARQUET
INTO parquet_base_table
SELECT $int32_field AS int32_field,
       $utf8_field AS utf8_field,
       $float_field AS float_field,
       $double_field AS double_field,
       $int64_field AS int64_field,
       $json_field AS json_field,
       $bson_field AS bson_field;

```

Related Links

[JSON Selectors Examples in Data Pipelines](#)

[Data Pipelines](#)

[Load Metadata Values in Data Pipelines](#)

3.3.1 JSON Selectors Examples in Data Pipelines

The data pipeline functionality enables you to load data in the structured JSON data format. Use these examples to see how you can access the structure in different ways. For an overview of loading JSON data, see:

- [Data Formats for Data Pipelines](#)
- [Data Pipeline Load of JSON Data from Kafka](#)

Scalar Extraction from JSON

These examples load a scalar into the `VARCHAR` column named `customer_name`. Use the `CREATE TABLE SQL` statement to create the table with this column.

```
CREATE TABLE my_schema.my_table (customer_name VARCHAR NOT NULL);
```

For a JSON file with the value at the top level, the selector can refer to the key directly.

Load a file with this data.

```
{"name": "John"}
```

Use the selector `$name`.

```

CREATE PIPELINE my_pipeline
  SOURCE filesystem
  FILTER 'data.json'
EXTRACT
  FORMAT json
INTO my_schema.my_table
SELECT
  $name AS customer_name;

```

List all top-level JSON keys used in the `SELECT` statement in `JSON_FIELDS` of the `CREATE PIPELINE SQL` statement.

For a JSON file that has a nested value, you need a complex selector.

```
{"a": {"b": {"name": "John"}}}
```

In this case, the same `CREATE PIPELINE` statement for this file has `$a.b.name` instead of `$name`.

```
CREATE PIPELINE my_pipeline
  SOURCE filesystem
  FILTER 'data.json'
EXTRACT
  FORMAT json
INTO my_schema.my_table
SELECT
  $a.b.name AS customer_name;
```

NULL and Empty Handling for JSON Scalars

The Ociant® System handles all JSON NULL, empty, and missing values in the same way. The system loads these values as NULL. These values fail to load into non-nullable columns. Provide an explicit default in the pipeline or use the COLUMN_DEFAULT_IF_NULL option to accept the configured column default instead of attempting to load NULL values.

The JSON data contains this information.

```
{"a": {"b": {"name": "John"}}}
{"blah": {}}
```

In this case, the output table has two rows because there are two records in the JSON source file.

```
CREATE PIPELINE my_pipeline
  SOURCE filesystem
  FILTER 'data.json'
EXTRACT
  FORMAT json
INTO my_schema.my_table
SELECT
  $a.b.name AS customer_name;
```

Output

```
Ocient> select * from my_schema.my_table;
name
-----
John
NULL
Fetched 2 rows
```

When you load all these JSON files with the same pipeline definition, the Ociant System creates a table with the same two rows.

The middle value is empty.

```
{"a": {"b": {"name": "John"}}}
{"a": {"b": {}}}
```

The middle value is NULL.

```
{"a": {"b": {"name": "John"}}}
{"a": {"b": null}}
```

The leaf value is NULL.

```
{"a": {"b": {"name": "John"}}}
{"a": {"b": {"name": null}}}
```

Array Extraction from JSON

Supported array extraction scenarios are:

- One-dimensional arrays.
- Multi-dimensional arrays.
- Array projection is the nested application of JSON selectors over the elements of arrays for both one-dimensional and multidimensional arrays.
- Arbitrary nesting of arrays within JSON objects, including splitting array dimensions across the JSON path.
- Loading individual array elements, which can also be arrays.

Mapping a transformation function over array elements is not supported.

One-Dimensional Arrays

The next few examples load the same data into the `my_table` table both directly and using array projection.

```
CREATE TABLE my_schema.my_table (customer_names VARCHAR[] NOT NULL);
```

Load data directly from a JSON array of strings. The JSON data contains this information.

```
{
  "a": {
    "b": {
      "names": ["John", "Bob", "Rajiv"]
    }
  }
}
```

```
CREATE PIPELINE my_pipeline
  SOURCE filesystem
  FILTER 'data.json'
  EXTRACT
    FORMAT json
  INTO my_schema.my_table
  SELECT
    $a.b.names[] AS customer_names;
```

Output

```
Ocient> select * from my_schema.my_table;
customer_names
-----
[John, Bob, Rajiv]
Fetched 1 row
```

Array Projection

Array projection is the application of the specified selector over all elements in an array, similar to a for loop.

Load data using array projection into the same table using this JSON data.

```
{ "a": [ {"name": "John", "hometown": "Chicago"},
         {"name": "Bob", "hometown": "Cucamonga"},
         {"name": "Rajiv", "hometown": "Glendale Heights"} ] }
```

```
CREATE PIPELINE my_pipeline
  SOURCE filesystem
  FILTER 'data.json'
  EXTRACT
    FORMAT json
```

```

INTO my_schema.my_table
SELECT
    $a[].name AS customer_names;

```

Output

```

Ocient> select * from my_schema.my_table;
customer_names
-----
[John, Bob, Rajiv]
Fetched 1 row

```

The output of both loading operations is the same.

Repeat this example with data in an array that contains nested objects. The JSON file contains this information.

```

{ "a": [ { "b": { "name": "John", "hometown": "Chicago" } },
          { "b": { "name": "Bob", "hometown": "Cucamonga" } },
          { "b": { "name": "Rajiv", "hometown": "Glendale Heights" } } ] }

```

```

CREATE PIPELINE my_pipeline
SOURCE filesystem
  FILTER 'data.json'
EXTRACT
  FORMAT json
INTO my_schema.my_table
SELECT
    $a[].b.name AS customer_names;

```

Output

```

Ocient> select * from my_schema.my_table;
customer_names
-----
[John, Bob, Rajiv]
Fetched 1 row

```

The output is still the same.

Multi-Dimensional Arrays

Multi-dimensional arrays work similarly to one-dimensional arrays.

These examples load the same data into the `my_table` table both directly, and then using array projection.

```

CREATE TABLE my_schema.my_table (customer_names VARCHAR[] [] [] NOT NULL);

```

Load data directly from a JSON array of strings. The JSON file contains this information.

```

{ "a": {
  "b": {
    "names": [
      ["John", "Bob", "Rajiv"], ["Anna", "Hanna", "Vanna", "Rosanna"]],
      ["Masha", "Natasha", "Sasha"], ["Chad", "Thad", "Brad"]]]
  }
}

```

```

CREATE PIPELINE my_pipeline
SOURCE filesystem
  FILTER 'data.json'
EXTRACT
  FORMAT json

```

```

INTO my_schema.my_table
SELECT
    $a.b.names[] [] AS customer_names;

```

Load data using array projection. The JSON file contains this information.

```

{
  "a": [
    [
      {
        "b": {
          "name": "John",
          "hometown": "Chicago"
        }
      },
      {
        "b": {
          "name": "Bob",
          "hometown": "Cucamonga"
        }
      },
      {
        "b": {
          "name": "Rajiv",
          "hometown": "Glendale Heights"
        }
      }
    ],
    [
      {
        "b": {
          "name": "Anna",
          "hometown": "Chicago"
        }
      },
      {
        "b": {
          "name": "Hanna",
          "hometown": "Cucamonga"
        }
      },
      {
        "b": {
          "name": "Vanna",
          "hometown": "Glendale Heights"
        }
      },
      {
        "b": {
          "name": "Rosanna",
          "hometown": "Mahwah"
        }
      }
    ]
  ],
  [
    [
      {
        "b": {
          "name": "Masha",
          "hometown": "Boston"
        }
      },
      {
        "b": {
          "name": "Natasha",
          "hometown": "Cambridge"
        }
      },
      {
        "b": {
          "name": "Sasha",
          "hometown": "Summerville"
        }
      }
    ],
    [
      {
        "b": {
          "name": "Chad",
          "hometown": "Champaign"
        }
      },
      {
        "b": {
          "name": "Thad",
          "hometown": "Urbana"
        }
      },
      {
        "b": {
          "name": "Brad",
          "hometown": "Mahomet"
        }
      }
    ]
  ]
]
}

```

`$a[] [] .b.name` accesses each object such as `{"b": {"name": "John", "hometown": "Chicago"}}` and uses `b.name` to apply this nested attribute selector on all of the elements in the inner arrays.

```

CREATE PIPELINE my_pipeline
  SOURCE filesystem
  FILTER 'data.json'
  EXTRACT
    FORMAT json
  INTO my_schema.my_table
  SELECT
    $a[] [] .b.name AS customer_names;

```

Output

```

Ocient> select * from my_schema.my_table;
customer_names
-----
[[[John, Bob, Rajiv], [Anna, Hanna, Vanna, Rosanna]], [[Masha, Natasha, Sasha],
[Chad, Thad, Brad]]]
Fetched 1 row

```

The output of both loading operations is the same.

Multi-Dimensional Arrays With Dimensions Split Across the JSON Path

To load a two-dimensional array, for example, use a one-dimensional array that contains JSON objects, each of which contains another one-dimensional array. Load the relevant parts of this JSON object into a two-dimensional array.

In this example, the column has `VARCHAR[] []` type and the `SELECT` expression is `$x.a[] .b.c[] .name`. The loaded data is a two-dimensional array of names.

The JSON file contains this information.

```
{
  "x": {
    "a": [
      {
        "b": {
          "c": [
            {"name": "John", "hometown": "Chicago"},
            {"name": "Bob", "hometown": "Cucamonga"},
            {"name": "Rajiv", "hometown": "Glendale Heights"},
          ]
        }
      },
      {
        "b": {
          "c": [
            {"name": "Anna", "hometown": "Chicago"},
            {"name": "Hanna", "hometown": "Cucamonga"},
            {"name": "Vanna", "hometown": "Glendale Heights"},
            {"name": "Rosanna", "hometown": "Mahwah"},
          ]
        }
      }
    ]
  }
}
```

```
CREATE TABLE my_schema.my_table (customer_names VARCHAR[] []);
```

```
CREATE PIPELINE my_pipeline
SOURCE filesystem
  FILTER 'data.json'
EXTRACT
  FORMAT json
INTO my_schema.my_table
SELECT
  $x.a[] .b.c[] .name AS customer_names;
```

Output

```
Ocient> select * from my_schema.my_table;
customer_names
-----
[[John, Bob, Rajiv], [Anna, Hanna, Vanna, Rosanna]]
Fetched 1 row
```

The output is a two-dimensional array of names.

Extract Data from Individual Array Elements

Extract data from an element of a JSON array. That element can also be an array.

In this example, extract the first element of **a**, which is an array. JSON array indexes start at 1 so that all array indexing in Pipelines are consistent with the SQL standard.

The JSON file contains this information.

```
{
  "a": [
    {"b": [{"name": "John", "hometown": "Chicago"}]},
    {
      "b": [
        {"name": "Bob", "hometown": "Cucamonga"},
        {"name": "Rajiv", "hometown": "Glendale Heights"},
      ]
    },
    {"b": [{"name": "Rajiv", "hometown": "Glendale Heights"}]}
  ]
}
```

```
CREATE TABLE my_schema.my_table (customer_names VARCHAR[]);
```

```
CREATE PIPELINE my_pipeline
  SOURCE filesystem
  FILTER 'data.json'
  EXTRACT
    FORMAT json
  INTO my_schema.my_table
  SELECT
    $a[2].b[].name AS customer_names;
```

Output

```
Ocient> select * from my_table.my_schema;
customer_names
-----
[Bob, Rajiv]
Fetched 1 row
```

NULL and Empty Handling for JSON Arrays

The Ociant System handles NULL, empty, and missing values the same way for arrays as for scalars. The system converts a value that is NULL, empty, or missing to NULL and loads it as NULL.

Provide an explicit default in the pipeline or use the `COLUMN_DEFAULT_IF_NULL` option to accept the configured column default instead of attempting to load NULL.

Tuples of Scalars

Data pipeline loading supports tuples of scalars in the JSON source.

Tuples of arrays and tuples of tuples are not supported.

The specification of tuple elements using curly braces is only supported for the basic case (i.e. `$a.{b,c}`). More complex selectors are not supported.

The recommended way of specifying tuples is to use an individual JSON selector for each tuple element, such as `$a.b` or `$a.c`.

You can apply functions to tuple elements.

Basic Tuple Construction

Load two strings into a tuple.

```

{"users_hometowns": {"a": {"name": "Bob", "hometown": "Cucamonga"}}}

CREATE TABLE my_schema.my_table (users_hometowns TUPLE<<VARCHAR,VARCHAR>>);

CREATE PIPELINE my_pipeline
...
SELECT
    tuple<<char,char>>($users_hometowns.a.name, $users_hometowns.a.hometown)
AS users_hometowns;

```

Output

```

Ocient> select * from my_schema.my_table;
users_hometowns
-----
<<Bob, Cucamonga>>
Fetched 1 row

```

Applying Functions (Transformations) To Tuple Elements

In this example, load a tuple where elements have different types, VARCHAR and TIMESTAMP, and where the second element has to be converted from a JSON string into an Ocient timestamp. The JSON file contains this information.

```

{"event_timestamp": {"event": "my_event", "timestamp": "1980-02-03 15:16:17.12345"}}

CREATE TABLE my_schema.my_table (event_timestamp TUPLE<<VARCHAR,TIMESTAMP>> NOT NULL);

CREATE PIPELINE my_pipeline
...
SELECT
    tuple<<char,timestamp>>($event_timestamp.event,
    TO_TIMESTAMP(char($event_timestamp.timestamp), 'yyyy-MM-dd HH:mm:ss.SSSSSS'))
AS event_timestamp;

```

Output

```

Ocient> select * from my_schema.my_table;
event_timestamp
-----
<<my_event, 1980-02-03 15:16:17.123450000>>
Fetched 1 row

```

The output table has one row of tuples.

NULL and Empty Handling for JSON Tuples

All the rules for handling NULL, empty, and missing elements that apply to scalars and arrays also apply to tuples. If any part of the selector is NULL, empty, or missing, data pipeline loading converts that value to NULL.

Additionally, because you can apply functions to tuple elements (and not array elements), you can use the NULL_IF function to convert a tuple element to NULL. For example, `tuple<<char,vvarchar>>($a.name, NULL_IF($a.hometown, 'N/A'))` indicates to the pipeline that the string 'N/A' signifies NULL for the hometown element but not for the name element.

Special Characters in JSON Keys

This example shows loading data where the JSON key names contain a special character.

Create a table for the load using the `customer_name` column.

```
CREATE TABLE my_schema.my_table (customer_name VARCHAR NOT NULL);
```

Load a file with this data. The JSON key name contains the `-` special character.

```
{"first-name": "John"}
```

Use double quotes for the selector `$first-name` in the CREATE PIPELINE SQL statement.

```
CREATE PIPELINE my_pipeline
  SOURCE filesystem
    FILTER 'data.json'
  EXTRACT
    FORMAT json
  INTO my_schema.my_table
  SELECT
    $"first-name" AS customer_name;
```

Array Extraction Operations

Other operations exist in the source field selector syntax to flatten or compact arrays while extracting the data.

Flatten Arrays

Flattening means that the extracted array should not increase the rank, and instead, sub-arrays should be concatenated. To flatten an array, use the underscore character in the array selector (e.g., `$data.array[_]`).

Example

This JSON data contains a nested array with two sub-arrays having data `[1,2,3]` and `[4,5,6]`.

```
{
  a: {
    b: [
      {c: [1,2,3] },
      {c: [4,5,6] }
    ]
  }
}
```

Flatten the nested array.

```
a.b[_].c[]
```

Output: `[1,2,3,4,5,6]`

This selected array has rank 1, and not 2. Contrast this with the non-flattened selector `a.b[] .c[]`, which would have the output `[[1,2,3],[4,5,6]]`.

You can also flatten arrays with the `FLATTEN` function. For details, see [Transform Data in Data Pipelines](#). The functionality of the operator and function is equivalent.

Compact Arrays

Compaction eliminates NULL values from the output data. The exact bit pattern of NULL in the source data is source-type dependent. For JSON, `null` is a literal keyword that is unambiguous. For CSV or other less-defined types, the configuration determines which exact bits equate to NULL. To compact an array, use the exclamation mark character in the array selector (e.g., `$data.array[!]`).

Example

This JSON data contains a nested array with four NULL values.

```
{
  a: {
    b: {
      c: [ 1,null,2,null,3,null,null]
    }
  }
}
```

Compact the nested array.

```
a.b.c[!]
```

Output: [1,2,3]

You can also compact arrays with the `ARRAY_COMPACT` function. For details, see [Transform Data in Data Pipelines](#). The functionality of the operator and function is equivalent.

Related Links

[Data Pipeline Load of Parquet Data from S3](#)

[Data Pipeline Load of Parquet Data from S3](#)

[Data Types for Data Pipelines](#)

[Data Formats for Data Pipelines](#)

[Monitor Data Pipelines](#)

3.3.2 Parquet Selectors Examples in Data Pipelines

The data pipeline functionality enables you to load data in the Apache® Parquet™ data format. Use these examples to see how you can access the structure in different ways. For an overview of loading Parquet data, see [Data Pipeline Load of Parquet Data from S3](#).

Work with Parquet Selectors

The `SELECT` statement for a Parquet file uses the same syntax as the existing JSON selectors. This selection applies to lists, structures, and maps.

The Ocient® System retrieves its list of requested columns from the `SELECT SQL` statement. The columns correspond to the field names of the schema for the Parquet files for the load. The load retrieves required columns from the Parquet source files and leaves the other columns alone.

This example snippet shows a list of ["id", "name", "array[]", "a[].project", "my_map.key_name"] as the fields to select from the data. The statement only retrieves the columns for ["id", "name", "array", "a", "my_map"] in the row groups.

```
SELECT
  $id                as id,
  $name              as name,
  $array[]           as array,
  $a[].project       as projection,
  $my_map.key_name   as key_col
...
```

Work with Complex Parquet Selectors

These examples show how to load a Parquet list and convert it to an array and how to load a map and convert it to a tuple.

Load List as an Array Example

Create the `tips` table to store the tip inquiries a city receives. The columns are:

- `id` — Identifier as an integer
- `categories` — Categories as a list of strings
- `mayor` — Mayor as a string
- `mentions` — Mentions as a list of strings
- `service_name` — Service name as a non-nullable string
- `tags` — Tags as a list of strings
- `url` — URL as a string
- `username` — Username as a string
- `text` — Text description as a non-nullable string
- `venue_name` — Venue name as a non-nullable string

```
CREATE TABLE tips(
  id INT NOT NULL,
  categories VARCHAR(255) [],
  mayor VARCHAR(255),
  mentions VARCHAR(255) [],
  service_name VARCHAR(255) NOT NULL,
  tags VARCHAR(255) [],
  url VARCHAR(255),
  username VARCHAR(255),
  text VARCHAR(255) NOT NULL,
  venue_name VARCHAR(255) NOT NULL);
```

Create the `tips_pipeline` data pipeline for loading the tip data. Specify the Amazon® Web ServicesSM (AWSSM) S3 source with bucket `ocient-examples`, object key `metabase_samples/parquet/tips.parquet`, and endpoint `https://s3.us-east-1.amazonaws.com`. The source format is Parquet.

```
CREATE PIPELINE tips_pipeline
  SOURCE S3
    BUCKET 'ocient-examples'
    OBJECT_KEY 'metabase_samples/parquet/tips.parquet'
    ENDPOINT 'https://s3.us-east-1.amazonaws.com'
  EXTRACT
    FORMAT parquet
INTO tips
SELECT
  $id AS id,
  $source.categories[] AS categories,
  $source.mayor AS mayor,
  $source.mentions[] AS mentions,
  $source."service" AS service_name,
  $source.tags[] AS tags,
  $source.url AS url,
  $source.username AS username,
  $text AS text,
  $venue.name AS venue_name;
```

Start the data pipeline. The data pipeline loads the `categories`, `mentions`, and `tags` columns as arrays.

```
START PIPELINE tips_pipeline;
```

Load Map as a Tuple Example

This example uses the file `/tmp/to-load/locations.parquet` that contains a `MAP<STRING, DOUBLE>` field named `coordinates`. Each value of the map field contains `x` and `y` entries for the coordinates.

Create the `locations` table to store the map values. The `cartesian_coords` column is a tuple with two elements for the `x` and `y` coordinates.

```
CREATE TABLE locations(
  cartesian_coords TUPLE<<DOUBLE, DOUBLE>> NOT NULL);
```

Create the `locations_pipeline` data pipeline for loading the maps data. Specify the filter for the Parquet file `/tmp/to-load/locations.parquet`. The source format is Parquet. Select the `x` and `y` coordinates from the map.

```
CREATE PIPELINE locations_pipeline
  SOURCE filesystem
  FILTER '/tmp/to-load/locations.parquet'
  EXTRACT
    FORMAT parquet
  INTO locations
  SELECT
    TUPLE($coordinates.x, $coordinates.y) AS cartesian_coords;
```

Start the data pipeline. The data pipeline loads the `cartesian_coords` column as a tuple.

```
START PIPELINE locations_pipeline;
```

Related Links

[Data Types for Data Pipelines](#)

[Data Formats for Data Pipelines](#)

[Data Pipeline Load of JSON Data from Kafka](#)

[JSON Selectors Examples in Data Pipelines](#)

[Data Pipeline Load of Parquet Data from S3](#)

[Monitor Data Pipelines](#)

3.4 Transform Data in Data Pipelines

Data pipeline functionality supports data transformation during loading to efficiently convert and clean source data for your chosen table schema. You can use two broad classes of data transformation: source field references or SQL transformation functions.

Source Field References

When you load data using a data pipeline, you reference extracted data using a special syntax named a source field reference. These expressions are either a named reference such as `$my_field` or a numeric field reference such as `$1`.

In structured data formats like JSON, more complex source field references allow you to transform the structured data into the required form. For example, you can extract an array of values from an array of nested objects with an expression such as `$data.items[].price`.

To learn more about transforming data with source field references, see:

- [Data Formats for Data Pipelines](#)
- [Data Types for Data Pipelines](#)

Supported Transformation Functions in Data Pipelines

Data pipeline loading supports numerous SQL functions for use in transforming data during loading. You can use these transformation functions in the `INTO ... SELECT` portion of a pipeline. Transformation functions generally match the syntax and behavior of the function you use when you query in Ocident®. The behavior might differ from querying behavior in two ways:

- Some additional functions exist that provide behavior that does not exist in queries.
- Some existing functions have extended behavior during loading that provides important capabilities that are not in queries.

The types of transformation functions supported in loading include:

- Scalar data conversion and casting of data types
- Special loading transformation functions
- Date and time transformations
- Character data transformations
- Binary data transformations
- Array data transformations
- Logical operations transformations
- Network type functions
- Other data transformations
- Custom user-defined [Data Pipelines](#) using Apache® Groovy™

Scalar Transformation Functions and Casting

Scalar transformation functions include constructors, scalar data conversions, and data-casting functions.

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
T[] (X)	Casts a string representation of an array of values or the result of a JSON array extraction expression to the corresponding array with data type T where T is a SQL data type. Multi-dimensional arrays are supported. To indicate an N-dimensional array cast, use N sets of square brackets in the casting function. All transformation functions can only apply to an array as a whole. To apply functions on an element-wise basis, see TRANSFORM in Special Data Pipeline Transformation Functions .	CHAR	ARRAY[]	CSV:DECIMAL(8,4) [] [] (\$21) as decimal_2d_array_colJSON:CHA as namesDECIMAL(8,2) [] [] (\$a[] .b as pricesCHAR[] [] [] (\$names[] [] [as names
BIGINT(X)	Converts the specified type to a 64-bit signed integer.	BIGINTCHAR	BIGINT	CSV:BIGINT(\$22) as int64_colJSON:BIGINT(\$a.b.c) as int64_col

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
BINARY(X)	Translates a string representation of bytes and converts them to real bytes. The string representation expects hexadecimal representations that convert to the matching bytes with this function. Hexadecimal data is not case-sensitive. Example source data: a891, F1D890	CHAR	BINARY or VARBINARY	CSV: BINARY(\$30) as varbinary_col JSON: BINARY(\$a. as varbinary_col
BOOLEAN(X)	Handles textual representation of Booleans and converts them to its Boolean equivalent. You can represent true as any of these values: "t", "true", "yes", "1", "y", "on" false representation as any of these values: "f", "false", "no", "0", "n", "off"	CHAR	BOOL	CSV: BOOLEAN(\$23) as bool_col JSON: BOOLEAN(\$a.b.c) as bool_col
BYTE(X)	Supports characters and number or byte array representations. Interprets character data as the ASCII numeric representation of a byte. If you specify a byte array, the Ociant System uses the first element of the array. When the system converts text data to a byte, the function throws an error if the value is > 127 or < -128. Example: '101' -> 101 '-23' -> -23	CHAR	BYTE	CSV: BYTE(\$24) as byte_col JSON: BYTE(\$a.b.c) as byte_col

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
CHAR(X)	Converts the specified input type to its string representation.	CHARBINARYFLOATDOUBLEINTBYTESMEDIUMTEXTBLOBJSON	VARCHAR	CSV:CHAR(\$26) as char_col JSON:CHAR(\$a.b.c) as varchar_col CHAR(\$a.b.c) as char_col CHAR(\$array_field[]) as char_col
DECIMAL(X,P,S)	X is input data to convert to a decimal.P is the precision of the decimal (number of significant digits) and S is the scale (number of digits after the decimal point).You can omit P and S or only S. In that case, the Ocint System defaults P to 31 and S to 16.Example: DECIMAL(30.0,4,2) -> 30.00	CHAR	DECIMAL	CSV:DECIMAL(\$27, 8, 4) as decimal_col JSON:DECIMAL(\$a.b 8, 4) as decimal_col
DOUBLE(X)	Converts the specified type to its double representation.	BINARYCHAR	DOUBLE	CSV:DOUBLE(\$28) as double_col JSON:DOUBLE(\$a.b.c as double_col
FLOAT(X)	Converts the specified type to its float representation.	BINARYCHAR	FLOAT	CSV:FLOAT(\$29) as float_col JSON:FLOAT(\$a.b.c) as float_col
HASH(X,N)	Translates a string representation of bytes and converts them to bytes. The string representation expects hexadecimal representations that convert to the matching bytes with this function. Hexadecimal data is not case-sensitive. X is the source data, such as:a891,F1D890N is the fixed number of bytes in the binary array.	CHAR	HASH	CSV:HASH(\$23, 9) as fixed_hash_binary_col JSON:HA 9) as fixed_hash_binary_col

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
INT(X)	Converts the specified type to a 32-bit signed integer.	DECIMALCHARSMALLINT		CSV:INT(\$31) as int32_colJSON:INT(\$a.b.c) as int32_col
IP(X)	Converts the specified string to an IPv6 address equivalent.X is the source data, such as:2001:0db8:1234:5678:abcd:ef01:2345:67892001:db8:0:0:0:0:2192.0.2.1	CHAR	IP	CSV:IP(\$32) as ip_colJSON:IP(\$a.b.c) as ip_col
IPV4(X)	Converts the specified string to an IPv4 address equivalent.X is the source data, such as:192.0.2.1	CHAR	IPV4	CSV:IPV4(\$32) as ipv4_colJSON:IPV4(\$a.b.c) as ipv4_col
SMALLINT(X)	Converts the specified value to its 16-bit signed integer representation.	DECIMALCHARBYTESMALLINT		CSV:SMALLINT(\$33) as int16_colJSON:SMALLINT(\$a.b.c) as int16_col
ST_POINT(Long, Lat)	Accepts a longitude and latitude coordinate for a geospatial point. Arguments must be of type <code>double</code> or string values that can convert to <code>double</code> .For WKT representation, see Point Constructors .	Both DOUBLEBoth CHAR	ST_POINT	CSV:ST_POINT(\$1, \$2) as st_point_colJSON:ST_POINT(\$a.b.c, \$d.e.f) as st_point_col
ST_POINTFROMTEXT(X)	Accepts the WKT text representation for ST_POINT with X and Y coordinates only.Example WKT representation: 'POINT(-40.5318 11.8532)'	CHAR	ST_POINT	st_pointfromtext(\$1)st_point
TINYINT(X)	Converts the specified value to an 8-bit signed integer.	CHAR	TINYINT	CSV:TINYINT(\$34) as int8_colJSON:TINYINT(\$a.b.c) as int8_col

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
TUPLE <<T, [T]*>>(X, [Y]*)	Creates a tuple from the specified types.TUPLE supports arrays of tuples, but does not support tuples within tuples.The arguments to the TUPLE casting function can be either a single source value or must match the number of types defined in the TUPLE. If you specify a single value, the Ocint System infers individual elements of the tuple using the TUPLE format (value1,value2,value3).Example source data:(123,12.75) is valid text data for a TUPLE<<INT,DECIMAL(7,2)>>.	CHAR	TUPLE of type	CSV:TUPLE<<INT,DECIMAL(7,2), as int_decimal_timestamp_binary as int_decimal_timestamp_binary \$b, \$c)) as int_decimal_timestamp_binary
UUID(X)	Creates a unique identifier from the specified value.Example source data: '237e9877-e79b-12d4-a765-321741963000'	CHAR	UUID	CSV:UUID(\$34) as uuid_colJSON:UUID(\$a.b.c) as uuid_col

Special Data Pipeline Transformation Functions

Special data pipeline transformation functions include functions that only exist in loading, such as the METADATA function to load special data during pipeline execution.

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
<code>EXPLODE_OUTER(array)</code>	Expands the contents of the specified one-dimensional array. By default, the array can contain a maximum of 8,191 elements. For supported options, see Special Data Pipeline Transformation Functions .	ARRAY	Elements of the array in their own row.	<code>EXPLODE_OUTER(\$"data.responses" AS own</code>
<code>FILTER(T[], (T) -> Boolean)</code>	Filters elements in an array based on the logic in a lambda expression. The first argument is an expression that evaluates to an array that contains the elements to filter. The second argument is a lambda expression (or function reference) that returns <code>true</code> if an element should be retained. The second argument must be of the form <code>(x T) -> BOOL</code> . This returns a boolean value for a specified element <code>x</code> . The type <code>T</code> of the element <code>x</code> is specified in the lambda function. Elements in the first argument array are included in the resulting array when the lambda function returns <code>true</code> and excluded otherwise. For supported options and examples, see Special Data Pipeline Transformation Functions .	ARRAY Lambda Function <code>(x T) -> BOOL</code> or Function Reference	The elements of the array that matched the filter criteria function.	<code>FILTER(\$data.names, (name CHAR) -> length(name) > 5)</code>

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
METADATA(key)	Extracts the metadata value for the specified key from available metadata for the pipeline. For supported key values, see Load Metadata Values in Data Pipelines .	CHAR	Varies based on the metadata key.	METADATA('filename') → '/path/to/file/orders_183037'

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
<code>PARSE_DELIMITED_ARRAY</code> (<code>text</code> , <code>rank</code> , <code>array_element_delimiter</code> , <code>bracket_spec</code>)	Converts a string of text data representing an array into a <code>CHAR[]</code>. For example, with data in the form <code>[1;2;3]</code>, executing <code>PARSE_DELIMITED_ARRAY('1;2;3', 1, ';', '[]')</code> returns a <code>CHAR[]</code> with the value <code>['1','2','3']</code>.	<code>text</code> is <code>CHAR</code> data that represents the text to convert to an array. <code>rank</code> is an optional argument that represents the numeric rank of the array. The default value is 1. You can only specify <code>rank</code> as a literal value because you cannot retrieve it dynamically from a source field selector. <code>array_element_delimiter</code> is an optional argument that contains the control character for delimiting array elements. For example, <code>' , '</code> indicates that a comma separates each element in the <code>text</code> character data. The default value definition comes from the <code>ARRAY_ELEMENT_DELIMITER</code> value of the data pipeline. <code>bracket_spec</code> is an optional argument that is the pair of control characters for the open and close array brackets. For example, <code>' [] '</code> indicates that each rank of the array is enclosed in square brackets. The default definition comes from the <code>OPEN_ARRAY</code> and <code>CLOSE_ARRAY</code> characters of the data pipeline. Valid options are: <code>' '</code>, <code>' [] '</code>, <code>' {} '</code>, <code>' () '</code>, and <code>' < > '</code>.	<code>CHAR[]</code>	<pre> PARSE_DELIMITED_ARRAY(\$text_ 1, ';', ' [] ') PARSE_DELIMITED_ARRAY(\$ 2, ',', ' [] ') </pre>

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
<code>TRANSFORM(T[], (T) -> U)</code>	Transforms an array based on the logic in a lambda expression. The first argument is an expression that evaluates to an array that contains the elements to transform. The second argument is a lambda expression (or function reference) that defines how to transform each element. <code>TRANSFORM</code> returns an array of elements of the same length as the array argument of the type returned by the lambda function. The second argument must be of the form <code>(x T) -> U</code> . This returns a value of type <code>U</code> for a specified element <code>x</code> . The type <code>T</code> of the element <code>x</code> is specified in the lambda function. The second argument can also be a function reference for a single argument function. For supported options and examples, see Special Data Pipeline Transformation Functions .	ARRAYLambda Function (x) -> U or Function Reference	Elements of the array transformed to the return value and type of the anonymous function.	<code>TRANSFORM(\$data.names[], (e1 CHAR) -> LOWER(e1)) as lower_case_name_array</code> <code>TRANSFORM(\$data.names[], LOWER) as lower_case_name_array</code>

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
WIDTH_BUCKET(<i>X</i> , <i>min</i> , <i>max</i> , <i>N</i>)	Creates <i>N</i> equal-width buckets in the range [<i>min</i> , <i>max</i>) as a histogram. If <i>X</i> is less than <i>min</i> , then the function returns 0. If <i>X</i> is greater than or equal to the value of <i>max</i> , the function returns <i>N</i> +1 buckets. If <i>X</i> is less than the value of <i>max</i> and greater than or equal to the value of <i>min</i> , the function returns the number of the bucket corresponding to where the input value belongs in the histogram. The values are inclusive on the lower bound and exclusive on the upper bound, so all buckets are the same width. Requesting 100 buckets retrieves 102 possible buckets, bucket 0 through 101, where 0 captures all values below the minimum and 101 captures all values at the maximum and beyond.	<i>X</i> is the value for bucket determination. <i>min</i> is the minimum value for the histogram. <i>max</i> is the maximum value for the histogram. <i>N</i> is the number of buckets in the histogram.	Integer	WIDTH_BUCKET('\$2', 1, 5, 2) AS output

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
ZIP_WITH(array1, ..., array2, func)	Merges two or more input arrays, element-wise, into a single array using the trailing combining function. If one array is shorter than the others, the system appends NULL values at the end to match the length of the longest array before applying. Returns an array with elements that result from applying the function func to each pair of elements from array1 and array2 . If the arrays are not the same length, the system pads the shorter array with NULL values. func must be of the form (x T, y U) -> V, where x is the element from array1 and y is the element from array2 . Types T and U are specified in the lambda function. For supported options and examples, see Special Data Pipeline Transformation Functions .	ARRAYARRAYLambda Function (x T, y U) -> V or Function Reference	An array of length max(length(array1), length(array2)) containing elements of type V as returned by the lambda function.	ZIP_WITH(\$line_items[].product, \$line_items[].sku, (name CHAR, sku CHAR) -> concat(name, '[' , sku, ' '))

Date and Time Transformation Functions

Date and time transformation functions provide utilities to parse, convert, or transform DATE, TIME, and TIMESTAMP values in pipelines.

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
DATE(X)	Converts the specified value to a DATE value. This function always creates dates from a timestamp in UTC time. If the argument is a CHAR type, the function assumes that the value has a textual representation in the YYYY-MM-DD format. Supports any date separator character. For example, YYYY:MM:DD is also valid.	CHARTIMESTAMP	DATE	CSV:DATE(\$24) as date_col DATE(TIMESTAMP(\$24)) as date_col JSON:DATE(\$a.b.c) as date_col DATE(TIMESTAMP(\$a.b.c) as date_col
INT_TO_DATE(arg)	Converts the specified value to a DATE. This function creates a DATE type from an INT type representing the number of days after the Unix® epoch (January 1, 1970).	INT	DATE	INT_TO_DATE(\$1) AS col_date_1
MICROS_TO_TIME(arg)	Converts the specified value to the TIME type, treating the argument with microsecond precision.	BIGINT	TIME	MICROS_TO_TIME(\$1) AS col_time
MICROS_TO_TIMESTAMP(X)	Converts the specified value to a TIMESTAMP. This function creates a TIMESTAMP from a BIGINT that represents the number of microseconds after the Unix epoch (January 1, 1970).	BIGINT	TIMESTAMP	MICROS_TO_TIMESTAMP(\$35) as timestamp_col
MILLIS_TO_TIME(arg)	Converts the specified value to the TIME type, treating the argument with millisecond precision. Alias for TIME(X) where X is the BIGINT type.	BIGINT	TIME	MILLIS_TO_TIME(\$1) AS col_time

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
MILLIS_TO_TIMESTAMP(X)	Converts the specified value to a TIMESTAMP. This function creates a TIMESTAMP from a BIGINT that represents the number of milliseconds after the Unix epoch (January 1, 1970). Alias for TIMESTAMP(X) where X is the BIGINT type.	BIGINT	TIMESTAMP	MILLIS_TO_TIMESTAMP(\$35) as timestamp_col
NANOS_TO_TIME(X)	Converts the specified value to a TIME value. If the argument has the BIGINT type, the function treats the number with nanosecond precision.	BIGINT	TIME	CSV:NANOS_TO_TIME(\$34) as time_col NANOS_TO_TIME(BIGINT) as time_col JSON:NANOS_TO_TIME(\$34) as time_col NANOS_TO_TIME(BIGINT) as time_col
NANOS_TO_TIMESTAMP(X)	Converts the specified value to a TIMESTAMP. This function creates a TIMESTAMP from a BIGINT that represents the number of nanoseconds after the Unix epoch (January 1, 1970).	BIGINT	TIMESTAMP	CSV:NANOS_TO_TIMESTAMP(\$35) as timestamp_col JSON:NANOS_TO_TIMESTAMP(\$35) as timestamp_col NANOS_TO_TIMESTAMP(\$35) -> 2022-05-04T16:29:48.12345678
NOW()	Returns the current TIMESTAMP.	none	TIMESTAMP	now() as col_load_timestamp
SECONDS_TO_TIME(arg)	Converts the specified value to the TIME type, treating the argument with second precision.	BIGINT	TIME	SECONDS_TO_TIME(\$1) AS col_time
SECONDS_TO_TIMESTAMP(X)	Converts the specified value to a TIMESTAMP. This function creates a TIMESTAMP from a BIGINT that represents the number of nanoseconds after the Unix epoch (January 1, 1970).	BIGINT	TIMESTAMP	CSV:SECONDS_TO_TIMESTAMP(\$35) as timestamp_col JSON:SECONDS_TO_TIMESTAMP(\$35) as timestamp_col NANOS_TO_TIMESTAMP(\$35) -> 2022-05-04T16:29:48

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
TIME(X)	Converts the specified value to a TIME value.If the argument has the BIGINT type, the function treats the number with millisecond precision.If the argument is a CHAR type, the function assumes that the value has a textual representation in the HH:mm:ss format.Supports any separation character. For example, HH/mm/ss is also valid.	CHARBIGINTTIMESTAMP	TIME	CSV:TIME(\$34) as time_colTIME(BIGINT(\$34)) as time_colJSON:TIME(\$a.b.c) as time_colTIME(BIGINT(\$a.b.c)) as time_col
TIMESTAMP(X)	Converts the specified value to a TIMESTAMP.This function creates a TIMESTAMP from a BIGINT that represents the number of milliseconds after the Unix epoch (January 1, 1970). When you convert from the DATE type, the function defaults the TIMESTAMP value to the start of the day in UTC time.If the argument is a CHAR type, the function assumes that the value has a textual representation in the YYYY-MM-DD HH:mm:ss[.SSSSSSSS] format. In this case, the nanoseconds are optional. The delimiters in the date-time value cannot be in a different format than the default value.	CHARBIGINTDATE	TIMESTAMP	CSV:TIMESTAMP(\$35) as timestamp_colJSON:TIMESTAMP(\$a.b.c) as timestamp_colTIMESTAMP('2023-03-01T18:22:58.826') -> 2023-03-01T18:22:58.826TIMESTAMP('2022-05-04T16:29:48.123')

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
TO_DATE(X,F)	Converts the specified value to a DATE. X — The value to convert.F — (optional) The format string for the interpretation of the textual representation of X. The default format is YYYY-MM-DD.For details, see Formatting Functions .	CHAR	DATE	to_date(\$3, 'YYYY/MM/DD')to_date(\$a.b.c, 'DD.MM.YYYY')
TO_TIME(X,F)	Converts the specified value to a TIME value. X — The value to convert.F — (optional) The format string for the interpretation of the textual representation of X. The default format is HH:MI:SS.	CHAR	TIME	to_time(\$n, 'HH:MI:SS')to_time(\$n, 'HHMISS')

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
<code>TO_TIMESTAMP(X,F,FT)</code>	Converts the specified value to a <code>TIMESTAMP</code>. <code>X</code> — The value to convert. <code>F</code> — (optional) The format string for the interpretation of the textual representation of <code>X</code>. The default format is <code>YYYY-MM-DD HH:MI:SS.FT</code> — (optional) The format type. There are two format types: <code>"ocient"</code> and <code>"java"</code>. If you use the <code>"ocient"</code> type, the function uses the <code>Ocient</code>-based formatting function to parse the timestamp. If you use the <code>"java"</code> type, the function uses the <code>DateTimeFormatter</code> in Java® to parse the timestamp. For details, see DateTimeFormatter (Java Platform SE 8). The default is the <code>"ocient"</code> type. If you specify <code>"java"</code>, you must specify a valid <code>DateTimeFormatter</code> format string, as there is no default. For details, see Formatting Functions.	<code>CHAR</code>	<code>TIMESTAMP</code>	<pre>to_timestamp(\$n, 'YYYY-MM-DD HH:MI:SS.US')to_timestamp(\$n 'DD.MM.YYYY HH:MI:SS.NS')to_timestamp(\$n 'uuuu-mm-dd HH:mm:ss.SSSSSS', 'java')</pre>

For the `TO_TIMESTAMP` function, the `Ocient` formatter currently does not support variable-length sections of timestamps (i.e., the data might have a different number of fractions of seconds across records).

For example, assume this source data.

```
2024-01-01 12:00:00.123456
```

```
2024-01-01 12:00:00.1234
```

Then, you must specify the `format_type` as `java` and enter the `DateTimeFormatter` format string to parse the timestamps.

Character Data Transformation Functions

Character data transformation functions operate on **VARCHAR** data to transform a string of data.

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
CONCAT(arg1, arg2[, ...])	Returns a string that is the result of concatenating the arguments provided. Accepts two or more arguments to concatenate. If one of the arguments to CONCAT is NULL , then CONCAT returns NULL . CONCAT does not support HASH or BINARY arguments in loading.	arg1: CHAR arg2: CHAR arg3... CHAR argN (optional) CHAR	CHAR	CONCAT (\$1, \$2) CONCAT (trim(\$first_name), trim(\$last_name))
SUBSTRING(X, P, L)	Extract a substring from X beginning at the position offset P with a length L . X is the source data in CHAR or BINARY data type. P is the numeric byte position in the record that starts at index 1, which indicates the position to begin reading. L is the length of data to return starting from position P . For VARCHAR data, by default, the Ocient System treats each byte in a source field reference (e.g., \$your_field) as an ASCII-encoded character, not multi-byte character data. To handle multi-byte character data such as UTF-8 in a source field, you must cast X as CHAR(\$your_field) or use another function that returns CHAR data. P and L represent the character offset and the number of characters to retrieve from the source data.	X: CHAR or BINARY P: INT L: INT	Return type matches the type of X : CHAR BINARY	substring (\$order.position, 5) as col_postcode
TRIM(arg1) BT arg2]	Trims the leading and trailing whitespace characters (or optionally, the characters specified in the arg2 argument) from the arg1 argument.	arg1: CHAR arg2 (optional) must be a CHAR of length 1.	CHAR	TRIM ('__testTrimFunction') -> 'testTrimFunction' BT testBtrimFunction '') -> 'testBtrimFunction'
REPLACE(arg1, arg2, arg3)	The arg1 argument is the input string for the replace operation. The arg2 argument is the pattern to replace or the target string. The arg3 argument is the pattern for the replacement or the replacement string. This function replaces occurrences of arg2 in arg1 with arg3 .	arg1: CHAR arg2: CHAR arg3: CHAR	CHAR	REPLACE ('testReplaceFunction', 'X', 'Z') -> 'testReplaceFunction'
LENGTH(arg1)	The CHAR_LENGTH(arg1) and CHARACTER_LENGTH(arg1) functions return the character length of the input string. arg1 is the input string. When you use these functions with the SUBSTRING , RIGHT , or LEFT function, you might experience unintended results for multi-byte characters, such as Arabic in UTF-8 if you do not cast the field reference to CHAR .	CHAR	INT	LENGTH ('hello world') -> 11
UPPER(arg1)	Converts all characters in the input string to all uppercase characters.	CHAR	CHAR	UPPER ('Hello there') -> 'HELLO THERE'

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
LOWER(<i>arg1</i>)	Converts all characters in the input string to all lowercase characters.	CHAR	CHAR	LOWER('Obi Wan Kenobi') -> 'obi wan kenobi'
LOCATE(<i>arg1</i> , <i>arg2</i> [, <i>arg3</i>])	Returns the index position of the first occurrence of the character value <i>arg1</i> in character value <i>arg2</i> . Optionally, you can also include <i>arg3</i> to POSITION(<i>arg1</i> , <i>arg2</i> [, <i>arg3</i>])	<i>arg1</i> : CHAR <i>arg2</i> : CHAR <i>arg3</i> : LONG	INT	LOCATE('F', 'testLocateFunction') -> 11 POSITION('on', 'testPositionFunction') 12) -> 19
LEFT(<i>arg1</i> , <i>arg2</i>)	Returns the number of characters in the <i>arg1</i> string equal to the <i>arg2</i> integer value. If the integer is negative, the function returns all characters except the last number of characters as specified by the <i>arg2</i> value. This function treats source field references (e.g., <i>\$your_field</i>) as single-byte ASCII characters by default, similar to the SUBSTRING function.	<i>arg1</i> : CHAR <i>arg2</i> : LONG	CHAR	LEFT('testLeftFunction', 4) -> 'test' LEFT('testLeftFunction', -4) -> 'testLeftFunc'
RIGHT(<i>arg1</i> , <i>arg2</i>)	Returns the number of trailing characters in the <i>arg1</i> string equal to the <i>arg2</i> integer value. If the integer is negative, the function returns all characters except the first number of characters as specified by the <i>arg2</i> value. This function treats source field references (e.g., <i>\$your_field</i>) as single-byte ASCII characters by default, similar to the SUBSTRING function.	<i>arg1</i> : CHAR <i>arg2</i> : LONG	CHAR	RIGHT('green tea', 3) -> 'tea' RIGHT('coffee beans', -7) -> 'beans'

When you use the SUBSTRING, RIGHT, and LEFT functions with Delimited or JSON formatted data sources, the Ocint System does not treat a source field reference (e.g., *\$your_field*) as multi-byte character data. Each byte is treated as a single-byte ASCII character. To handle multi-byte character data like UTF-8 in a source field, you must cast the source field using **CHAR(*\$your_field*)** or use another function that returns **CHAR** data.

When you use the LENGTH, CHAR_LENGTH, and CHARACTER_LENGTH functions with the SUBSTRING, RIGHT, and LEFT functions, you might experience unintended results for multi-byte characters, such as Arabic in UTF-8 if you do not cast the source field reference to **CHAR**.

Binary Data Transformation Functions

Binary data transformation functions operate on bytes provided when you use the **BINARY** format type in a pipeline. These functions are typically used in processing mainframe data outputs.

Name	Description	Supported Input Type	Example SE- LECT Ex- press- ion Return Type
FLOAT([, array in the endianness])	bytes is a 4-byte array extracted from BINARY data. format, parse bytes in standard IEEE 754 binary32 (single-precision) format.	bytes is a 4-byte array extracted from BINARY data. endianness (optional) must be a literal string. Valid values include 'big' and 'little'. The default value is the ENDIANNESS option in the pipeline definition.	BINARY:FLOAT("\$[4]") as float_col1FLOAT("\$[4]", 'little') as float_col2
DOUBLE([, array in the endianness])	bytes is an 8-byte array extracted from BINARY data. format, parse bytes in standard IEEE 754 binary64 (double-precision) format.	bytes is an 8-byte array extracted from BINARY data. endianness (optional) must be a literal string. Valid values include 'big' and 'little'. The default value is the ENDIANNESS option in the pipeline definition.	BINARY:DOUBLE("\$[8]") as double_col1DOUBLE("\$[8]", 'big') as double_col2
TINYINT([, 1-byte array signed] [, endianness])	bytes is a single-byte array extracted from BINARY data. signed is a string literal that indicates if the bytes represent a signed or unsigned BINARY format. Valid values include 'signed' and 'unsigned'. The default value is 'signed'. endianness is a string literal that indicates whether the bytes are in the big or little endian format. Valid values include 'big' and 'little'. The default value is the ENDIANNESS option in the pipeline definition.	bytes is a single-byte array extracted from BINARY data. signed is a string literal that indicates if the bytes represent a signed or unsigned BINARY format. Valid values include 'signed' and 'unsigned'. The default value is 'signed'. endianness is a string literal that indicates whether the bytes are in the big or little endian format. Valid values include 'big' and 'little'. The default value is the ENDIANNESS option in the pipeline definition.	BINARY:TINYINT("\$[1]") as tiny_1TINYINT("\$[1]", 'unsigned') as tiny_2TINYINT("\$[1]", 'signed', 'little') as tiny_3

Name	Description	Supported Input Type	Example SE- LECT Ex- press- Type
SMALLINT(<i>bytes</i> [<i>signed</i>] <i>endianness</i>)	<i>bytes</i> is a 2-byte array extracted from BINARY data. <i>signed</i> is a string literal that indicates if the bytes represent a signed or unsigned BINARY format. Valid values include 'signed' and 'unsigned'. The default value is 'signed'. <i>endianness</i> is a string literal that indicates whether the bytes are big or little in endian format. Valid values include 'big' and 'little'. The default value is the ENDIANNESS option in the pipeline definition.		BINARY:SMALLINT(2])") as small_1SMALLINT(\$ 2]), 'unsigned') as small_2SMALLINT(\$ 2]), 'signed', 'little') as small_3
INT(<i>bytes</i> [<i>signed</i>] <i>endianness</i>)	<i>bytes</i> is a 4-byte array extracted from BINARY data. <i>signed</i> is a string literal that indicates if the bytes represent a signed or unsigned BINARY format. Valid values include 'signed' and 'unsigned'. The default value is 'signed'. <i>endianness</i> is a string literal that indicates whether the bytes are big or little endian format. Valid values include 'big' and 'little'. The default value is the ENDIANNESS option in the pipeline definition.		BINARY:INT(\$"[9, 4])") as int_1INT(\$"[9, 4]), 'unsigned') as int_2INT(\$"[9, 4]), 'signed', 'little') as int_3

Name	Description	Supported Input Type	Example SELECT Ex- press- Returns- Type
BIGINT(bytes , [, 8-byte array signed], endianness)	The bytes argument is an 8-byte array extracted from BINARY data. The signed argument is a string literal that indicates if the bytes represent a signed or unsigned binary format. Valid values include 'signed' and 'unsigned'. The default value is 'signed'. The endianness argument is a string literal that indicates whether the bytes are big or little endian format. Valid values include 'big' and 'little'. The default value is the ENDIANNESS option in the pipeline definition.	The bytes argument is an 8-byte array extracted from BINARY data. The signed argument is a string literal that indicates if the bytes represent a signed or unsigned binary format. Valid values include 'signed' and 'unsigned'. The default value is 'signed'. The endianness argument is a string literal that indicates whether the bytes are big or little endian format. Valid values include 'big' and 'little'. The default value is the ENDIANNESS option in the pipeline definition.	BINARY:BIGINT(\$ 8]") as bigint_1BIGINT(\$ 8]", 'unsigned') as bigint_2BIGINT(\$ 8]", 'signed', 'little') as bigint_3
DECIMAL(bytes , format , precision , scale , [, packed sign_format])	The bytes argument is an array of bytes extracted from BINARY data. The format argument is a string literal that indicates the data format. Valid values include 'packed' (meaning the array contains decimal data in each nibble for two digits per byte) and 'zoned' (meaning the array contains decimal data in each nibble for one digit per byte). The precision argument is the total number of significant digits. The scale argument is the number of digits after the decimal point. The sign_format argument is a string literal that indicates how the transformation should extract sign information from the bytes. Valid values include 'leading' (means that the system stores the sign in the upper nibble of the first byte for both 'zoned' and 'packed' data formats), 'leading_separate' (means the system stores the sign as a separate new first byte from the digit data), 'trailing' (means that the system stores the sign in the upper nibble of the last byte when the data format is set to 'zoned' and in the lower nibble of the last byte when set to 'packed'), or 'trailing_separate' (means the system stores the sign as a separate new last byte from the digit data). For 'leading' and 'trailing' sign information, accepted sign values are 0x7, 0xB, and 0xD for negative signs and 0xA, 0xC, 0xE, and 0xF for positive. The accepted sign values for 'leading_separate' and 'trailing_separate' sign information are 0x2D for - negative signs and 0x2B for + positive. The default value for sign_format is 'trailing' for both format data formats.	The bytes argument is an array of bytes extracted from BINARY data. The format argument is a string literal that indicates the data format. Valid values include 'packed' (meaning the array contains decimal data in each nibble for two digits per byte) and 'zoned' (meaning the array contains decimal data in each nibble for one digit per byte). The precision argument is the total number of significant digits. The scale argument is the number of digits after the decimal point. The sign_format argument is a string literal that indicates how the transformation should extract sign information from the bytes. Valid values include 'leading' (means that the system stores the sign in the upper nibble of the first byte for both 'zoned' and 'packed' data formats), 'leading_separate' (means the system stores the sign as a separate new first byte from the digit data), 'trailing' (means that the system stores the sign in the upper nibble of the last byte when the data format is set to 'zoned' and in the lower nibble of the last byte when set to 'packed'), or 'trailing_separate' (means the system stores the sign as a separate new last byte from the digit data). For 'leading' and 'trailing' sign information, accepted sign values are 0x7, 0xB, and 0xD for negative signs and 0xA, 0xC, 0xE, and 0xF for positive. The accepted sign values for 'leading_separate' and 'trailing_separate' sign information are 0x2D for - negative signs and 0x2B for + positive. The default value for sign_format is 'trailing' for both format data formats.	BINARY:DECIMAL(\$ 10]", 'packed', 4, 2) as revenueDECIMAL(\$ 10]", 'zoned', 4, 2, 'leading') as revenue

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
TO_BINARY(F, S)	Converts a string representation of bytes and converts them to real bytes.	X is the string representation of bytes in BINARY or HASH format. F is either 'hex' for hexadecimal representations or the character set name (such as 'utf-8') for character representations. S (optional) is the number of bytes that should be present in the translated value. The default 0 means that the number of bytes can vary.	BINARYHASH	Example SELECT Expression TO_BINARY(\$1, 'hex', 3) AS col_binary_1, TO_BINARY(\$2, 'utf-8') AS col_binary_2

Array Data Transformation Functions

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
ARRAY_CAP(A, N)	Converts the length of an array to a maximum number of elements.	A is an array of any type. N is the maximum number of elements for the length of the specified array.	Array of any type	Example SELECT Expression ARRAY_CAP(A, 1000)
ARRAY_COMPACT(array)	Removes NULL values from the array.	array is an array that contains any type of elements. The array can be single or multi-dimensional.	Array of any type	Example SELECT Expression ARRAY_COMPACT(\$ints[])
ARRAY_CONTAINS(array, value)	Verifies if the array contains the specified value. If the specified value is NULL, then this transform function returns NULL.	array is an array that contains any type of elements. The array can be single or multi-dimensional. value is an integer.	Boolean or NULL	Example SELECT Expression ARRAY_CONTAINS(\$ints[], 1)
ARRAY_DISTINCT(array)	Removes duplicate values in an array.	array is an array that contains any type of elements. The array can be single or multi-dimensional.	Array of any type	Example SELECT Expression ARRAY_DISTINCT(\$ints[])

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
ARRAY_SORT	ARRAY_SORT(array, function) returns the input array based on the natural ordering of its elements. If one of the array elements is NULL, then this transform function sorts the NULL values to the end of the array. ARRAY_SORT(array, function) sorts and returns the input array based on the results of the specified Lambda function. The function should have two arguments representing two elements of the array. This function should return a negative integer, 0, or a positive integer if the first element is less than, equal to, or greater than the second element, respectively.	From the first syntax, array is an array of elements with these types: BIGINT, BOOLEAN, BYTE, DATE, DOUBLE, FLOAT, INT, SMALLINT, UUID, or VARCHAR. For the second syntax, array is an array that contains any type of elements. The array can be single or multi-dimensional. The function argument is a Lambda function with the format (x T, y T) -> INT or another SQL reference function.	Array of any type	ARRAY_SORT(\$"1d" [], (x int, y int) -> CASE WHEN x = y THEN 0 WHEN INT(COALESCE(x, 1000)) < INT(COALESCE(y, 1000)) THEN -1 ELSE 1 END) sorts the array in 1d column using the Lambda function with a CASE statement. The CASE statement returns 0 when x and y are equal, returns -1 when the integer cast of the non-NULL value x is less than the corresponding value for y, and returns 1 otherwise.
ELEMENT_AT	ELEMENT_AT(array, I) returns the element value of the array or tuple at the specified index. You can specify a negative index value as long as the absolute value of the index does not exceed the length of the array or tuple. If the index is out of bounds, the function returns NULL.	array is an array or tuple. I is a 1-based index, specified as an integer.	Any type.	\$a accesses the array [1,2,3]. ELEMENT_AT(\$a,1) returns 1. ELEMENT_AT(\$a,-1) returns 3. ELEMENT_AT(\$a,-2) returns 2. ELEMENT_AT(\$a,-4) returns NULL.
FLATTEN	FLATTEN(array) is an N-dimensional array into an N-1-dimensional array. If one of the inner array elements is NULL, this transform function returns NULL.	array is a multi-dimensional array, for example T[][], that contains any type of elements.	Array of any type that can be single or multi-dimensional.	FLATTEN(\$ints[] [])
REDUCE	REDUCE(array, start_value, merge_function, finish_function) applies a merge function to a start value and all elements in the array, and then reduces the resulting single value. Optionally, specify a finish function for the returned single value. The function applies the merge function to each element of the array with the start value to return a final value.	array is an array of any type and can be a single or multi-dimensional array. start_value is the starting value of any type. merge_function is a Lambda function or any reference function. This Lambda function has the form (x U, y T) -> U. finish_function is a Lambda function or any reference function. This Lambda function has the form (x U) -> V.	This function returns the value of type V, which can be any type, from the finish function. If you do not specify the finish function, this function returns the type U, which can be any type, from the merge function.	Assuming ints[] is INT[] of [1, 2, 3], REDUCE(\$ints[], 0, (acc int, x int) -> acc + x) returns the value 6. Assuming ints[] is INT[] of [1, 2, 3], REDUCE(\$ints[], 0, (acc int, x int) -> acc + x, BOOLEAN) returns the true value. In this case, the finish function is the BOOLEAN reference function with the input value being the resulting value from the merge function: BOOLEAN(6) .

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
TO_ARRAY_LENGTH(V, N)	Returns the number of copies of any JSON object in an array.	V is any JSON object to copy into the result array. N is the number of copies to make in the result array.	Array of any type	TO_ARRAY_LENGTH(V, 3)

Logical Operations Transformation Functions

Name	Description	Supported In-put Type	Return Type	Example SELECT Expression
IF(X,T,F)	Returns T if the expression X evaluates to true , or the function returns F if X evaluates to false . IF and IFF are function aliases.	Any type	Any type	IF('\$2', '1', '2') AS output
CASE WHEN expression1 THEN result1 WHEN expression2 THEN result2 ... ELSE result END	Returns the result value based on whether an expression is true . If none of the expressions evaluate to true , then the function returns the result value after the ELSE keyword.	Any type	Any type	CASE WHEN \$x THEN 'True was input in' ELSE 'False was input in' END AS toutput
NULL_IF(X, Y)	Returns the NULL value if the two input arguments are equal. If the arguments are not equal, the function returns X.	Any type	Any type	INT(NULL_IF(\$32, '')) INT(NULL_IF(\$a.b.c, ''))

Network Type Functions

Network type functions are also supported in loading. For example, `SUBNET(IP($9),13) AS ipv6_subnet_form1` in the `CREATE PIPELINE SELECT SQL` statement converts the input data in the column to an IPv6-mapped IPv4 address and then computes the prefix of size 13 for that address. For the supported network type functions, see [Network Type Functions](#).

Other Data Transformation Functions

Other data transformation functions include the extraction of values from the JSON format and miscellaneous functions such as `COALESCE`, which are commonly used when working with data pipelines.

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
COALESCE(arg1, arg2)	Returns arg1 if it is non-NULL; otherwise, the function returns arg2. arg1 and arg2 can be any data type, but their types must match.	Any type	Matches arg1 and arg2 type	COALESCE(\$1, \$2) COALESCE(int(\$1), int(\$2)) COALESCE(bigint(\$1), 13) COALESCE(substring(char(\$1), 1, 2), trim(\$2))
JSON_EXTRACT_PATH_TEXT(X, D)	Returns the value of the path D from a JSON string.	X is the JSON string for extracting a value in CHAR type. D is the key of the value to extract in CHAR type.	CHAR	JSON_EXTRACT_PATH_TEXT(JSON_UNESCAPE('b')) AS col_char

Name	Description	Supported Input Type	Return Type	Example SELECT Expression
MAP_KEYS(<i>x</i>)	Returns the keys in the specified JSON string.	<i>x</i> is the JSON string.	ARRAY OF STRING	Input JSON string a: '{"name":"John", "age":30, "car":null}' MAP_KEYS(<i>a</i>) Output:["age", "car", "name"]
MAP_VALUES(<i>x</i>)	Returns the values in the specified JSON string.	<i>x</i> is the JSON string.	ARRAY OF STRING	Input JSON string a: '{"name":"John", "age":30, "car":null}' MAP_VALUES(<i>a</i>) Output:["John", "30", null]

DEFAULT Keyword

You can use the DEFAULT keyword to include the configured default value of a column in transformation functions.

Example

```
COALESCE($my_field, DEFAULT) as my_column
```

Lambda Expressions

Expression	Description
Lambda Expression $(T_1, \dots, T_n) \rightarrow U$	A lambda function is a temporary function that takes 0 or more named arguments as a parenthesis-enclosed list and returns an expression. The expression can reference objects defined within the closure of the expression, which includes locally named arguments that are locally defined within an enclosing scope, field references, literals, etc. For examples, see Special Data Pipeline Transformation Functions .
Reference Function $UPCASE = UPCASE(x) \rightarrow$ $CHARtuple = tuple(T_1, \dots, T_n) \rightarrow tuple\langle T_1, \dots, T_n \rangle$	A reference function points to an existing function in the Ocint function list. Specifying the function name allows you to use that function in contexts where you would otherwise need to create a full lambda function. For examples, see Special Data Pipeline Transformation Functions .

User-Defined Data Pipeline Functions

You can create a user-defined data pipeline function using the CREATE PIPELINE FUNCTION SQL statement and Groovy. After you create data pipeline functions, you can execute them in data pipelines like any other function to transform records.

For more details, see [Data Pipelines](#) and [Data Types for User-Defined Data Pipeline Functions](#).

Related Links

[Formatting Functions](#)

[Data Types for Data Pipelines](#)

[Data Formats for Data Pipelines](#)

[Load Metadata Values in Data Pipelines](#)

[Data Pipeline Load of JSON Data from Kafka](#)

[Data Pipeline Load of Parquet Data from S3](#)

[Data Pipeline Load of CSV Data from S3](#)

3.4.1 Special Data Pipeline Transformation Functions

When you load data in the Ocient® System, you can use special loading transformation functions in the **SELECT** SQL statement to transform the data on load.

EXPLODE_OUTER

EXPLODE_OUTER expands a one-dimensional or multidimensional array into its elements with one element per row of output from the system. The array can contain a maximum of 8,192 elements, which is the limit for the default settings.

The function takes an array of rank N and expands the elements, returning elements of rank N-1. For example, when you apply **EXPLODE_OUTER** to an array of integers, the system returns each integer in its own row. When you apply **EXPLODE_OUTER** to an array of arrays of integers, the system returns each array of integers in its own row. You can add further transforms to these elements by nesting the statement, e.g., **NANOS_TO_TIMESTAMP(EXPLODE_OUTER(\$int_array_field[]))**.

You can use this function with multiple columns in the same table. The system holds constant columns that are not expanded. Also, the system combines expanded columns together in array order. If multiple expanded columns have different array lengths, the system expands the record into N records where N is the maximum length. Columns with array lengths less than N have NULL values after the system expands each value.

Limitations

- When you use this function with multi-dimensional arrays, you cannot apply it multiple times to match the number of dimensions, for example.
- Each record must have an associated 64-bit record number, which uniquely identifies a row within its partition. Configure the format of this number using the **RECORD_NUMBER_FORMAT** advanced pipeline option to modify the maximum number of elements that the array to expand can contain. For details, see [Data Pipelines](#).
 - The system uses record numbers for deduplication, so changing the **RECORD_NUMBER_FORMAT** option of a pipeline can change its deduplication behavior. Always create a new pipeline when you change this option to make rows deduplicate properly.
 - To make rows deduplicate properly when you do not set this option, always create a new pipeline when adding an **EXPLODE_OUTER** statement to your pipeline or removing all such statements from your pipeline.

Syntax

```
EXPLODE_OUTER(array)
```

Argument	Data Type	Description
array	Array	An array to expand.

Example

Create the **test_table** table to load data with three columns where the first two columns are integers, and the third is a string.

```
CREATE TABLE test_table (
  "i" INT NULL,
  "j" INT NULL,
  "v" VARCHAR(536870912) COMPRESSION DYNAMIC NULL );
```

Preview the load using this inline source data in JSON format: `{"id": 1, "int_array": [1, 2, 3], "string_array": ["a", "b"]}`. Use the `EXPLODE_OUTER` function to expand the elements of the arrays in the second and third JSON data values.

```
PREVIEW PIPELINE p1
SOURCE INLINE
  '{"id": 1, "int_array": [1, 2, 3], "string_array": ["a", "b"]}'
EXTRACT FORMAT json
INTO test_table
SELECT $"id" as i,
       EXPLODE_OUTER($"int_array"[]) as j,
       EXPLODE_OUTER($"string_array"[]) as v;
```

Output

```
i j v
-----
1 1 a
1 2 b
1 3 NULL
Fetched 3 rows
```

The output shows that the inline array values are expanded across the individual rows.

Special Functions for Array Transformations in Pipelines

The data pipeline functionality supports special transformation functions for arrays that use lambda functions or function expressions to apply SQL operations to individual elements in an array. These functions are not supported in standard Ocident SQL queries.

The supported special functions are:

- **TRANSFORM** — Transform individual elements in an array. Apply a transformation expression to every element in the array.
- **FILTER** — Filter the elements in an array. Apply a filter function to each element in an array and return an array of elements matching the filter function.
- **ZIP_WITH** — Combines multiple arrays using a zipping function. Apply a zipping function to N-wise pairs of elements from N arrays and return an array with the results from the zipping function.

Each of these functions takes one or more arrays as input arguments and a lambda or other reference function for applying to the elements in the array. A lambda function is an anonymous function that must have a specific signature and return type for the array transformation function where you use the lambda function.

TRANSFORM

Transforms an array based on the logic in a lambda expression. The first argument is an expression that evaluates to an array that contains the elements to transform. The second argument is a lambda expression or other reference function that defines how to transform each element.

The second argument must be of the form $(x \ T) \rightarrow U$. This specified form returns a value of type U for a specified element x . The lambda function specifies the type T of the element x .

The second argument can also be another reference function for a single argument function.

TRANSFORM returns an array of elements with the same length as the array argument. The returned array has the type returned by the lambda function. For multi-dimensional arrays, the **TRANSFORM** function applies to the outermost dimension. You can nest **TRANSFORM** operations to apply them over inner dimensions.

Syntax

```
TRANSFORM(array, function)
```

Argument	DataType	Description
array	T[]	The array of typeTelements can be single or multi-dimensional. TheTRANSFORMfunction only iterates over the first dimension.
function	Lambda function(x T) -> UorReference function	A lambda function (e.g.,(elem CHAR) -> LEFT(elem, 2)) that takes an element of typeTand returns a value of typeU, which can be the same type asTor a different type, or a reference function with the correct argument list (e.g.,UPPER).

Examples

Extract the First Initial from Each Element in an Array Using a Lambda Function

With the specified data in the pipeline source record { "data": { "names": ["Jane", "Charles", "Kim"] }}, create a data pipeline using the TRANSFORM function in the SELECT SQL statement with the lambda function (el CHAR) -> LEFT(el,1). This function returns the first character of each element in the input array \$data.names[]. The example focuses on the SELECT statement and does not show the rest of the data pipeline definition.

```
CREATE PIPELINE
...
SELECT
  TRANSFORM($data.names[], (el CHAR) -> LEFT(el,1)) as first_initial_array
```

Output: ["J", "C", "K"]

Convert All Elements in an Array to Lowercase Using a Reference Function

With the specified data in the pipeline source record { "data": { "names": ["Jane", "Joe", "Jill"] }}, create a data pipeline using the TRANSFORM function in the SELECT SQL statement with the LOWER reference function. This function converts the input strings in \$data.names[] to lowercase. The example focuses on the SELECT statement and does not show the rest of the data pipeline definition.

```
CREATE PIPELINE
...
SELECT
  TRANSFORM($data.names[], LOWER) as lower_case_name_array
```

Output: ["jane", "joe", "jill"]

FILTER

Filters elements in an array based on the logic in a lambda expression. The first argument is an expression that evaluates to an array that contains the elements to filter. The second argument is a lambda expression or reference function that returns true if the system should retain the element in the returned array.

The second argument must be of the form (x T) -> BOOL. This form returns a specified value for the specified element x. The lambda function specifies the type T of the element x.

The FILTER function includes elements in the first argument array in the resulting array when the lambda function returns true. Otherwise, the function excludes the elements.

Syntax

```
FILTER(array, function)
```

Argument	DataType	Description
array	T[]	The array of typeTelements can be single or multi-dimensional. TheFILTERfunction only iterates over the first dimension.
function	Reference functionorLambda function(x T) -> BOOL	A function that takes an element of typeTand returns aBOOL.The function can be a lambda function (e.g.,(elem CHAR) -> length(elem) > 10) or a function name that has the correct argument list (e.g.,UPPER).

Example

With the specified data in the pipeline source record { "data": { "names": ["Jane", "Charles", "Kim"] }}, filter the names in the input array for elements that are greater than five characters long using the Lambda function (name CHAR) -> length(name) > 5. The example focuses on the SELECT statement and does not show the rest of the data pipeline definition.

```
CREATE PIPELINE
...
SELECT
  FILTER($data.names[], (name CHAR) -> length(name) > 5) as long_names
```

Output: ["Charles"]

ZIP_WITH

Merges two or more input arrays, element-wise, into a single array using the trailing combining function. If one array is shorter than the other, the function appends NULLs at the end to match the length of the longest array at the start.

Returns an **array** with elements that result from applying the function to each pair of elements from **array1** and **array2**. If the arrays are not the same length, the function pads the shorter array with NULL values. The function returns an array of length **max(length(array1), length(array2))** containing elements of type **V** as returned by the lambda function.

function must be of the form (x T, y U) -> V where x is the element from **array1** and y is the element from **array2**. The lambda function specifies the types T and U. When zipping more arrays together, use the format (x T1, y T2, z T3) -> V where the arguments to the lambda function align with the order of the input arrays of the ZIP_WITH function.

Syntax

```
ZIP_WITH(array1, array2 [, ...], function)
```

Argument	DataType	Description
array1	T1[]	The array of typeT1elements can be single or multi-dimensional. TheZIP_WITHfunction only iterates over the first dimension.
array2	T2[]	The array of typeT2elements can be single or multi-dimensional. TheZIP_WITHfunction only iterates over the first dimension.

Argument	DataType	Description
function	Lambda function(x T) -> BOOL or Reference function	A function that takes an element of type T and returns a BOOL. The function can be a lambda function (e.g., (elem CHAR) -> length(elem) > 10) or a function name that has the correct argument list (e.g., UPPER).

Example

With the specified data in the pipeline source record { "line_items": [{ "product_name": "suede shoes", "sku": "FT-1023-BLU" }, { "product_name": "cheap sunglasses", "sku": "ZZ-7821-K" }] }, concatenate the product name \$line_items[].product_name strings with the SKU \$line_items[].sku strings as the format "name(sku)". The example focuses on the SELECT statement and does not show the rest of the data pipeline definition.

```
CREATE PIPELINE
...
SELECT ZIP_WITH(
  $line_items[].product_name, $line_items[].sku,
  (name CHAR, sku CHAR) -> concat(name, '(', sku, ')')
)
```

Output: ["suede shoes (FT-1023-BLU)", "cheap sunglasses (ZZ-7821-K)"]

Lambda Functions

A lambda function is a temporary function that takes zero or more named arguments as a parenthesis-enclosed list and returns an expression. The expression can reference objects defined within the closure of the expression, which includes locally named arguments that are locally defined within an enclosing scope, field references, literals, and so on.

Syntax

Define lambda functions using this syntax where T1 represents an element of type T1 and U represents the return type of the function.

```
(T1, ..., Tn) -> U
```

When you use this function in a SQL statement, the syntax requires an argument name and the explicit designation of the type of each argument, but the return type U can be inferred. For example, define a lambda function that accepts two arguments of type CHAR and INT named x and y, respectively.

```
(x CHAR, y INT) -> LEFT(x, y)
```

Examples

String Manipulation Using a Lambda Function

Convert x to uppercase and return the first character.

```
(x CHAR) -> LEFT(UPCASE(x), 1)
```

Concatenate Using a Lambda Function

Concatenate two strings (e.g., first and last name) with a space between them.

```
(x CHAR, y CHAR) -> CONCAT(x, ' ', y)
```

Reference Functions

A reference function points to an existing function in the Ocient function list. Specifying the function name allows you to use that function in contexts where you would otherwise need to create a full lambda function.

The signature of the function must match the argument structure and return value requirements.

Examples

The reference function `UPCASE` is equivalent to the lambda function `(x CHAR) -> UPCASE(x)` because the syntax of `UPCASE` accepts one argument of type `CHAR` and returns a value of type `CHAR`.

Use the `UPCASE` function with the `TRANSFORM` function to capitalize all names in an array. These two SQL statements are equivalent.

```
TRANSFORM($names[] , UPCASE)
```

```
TRANSFORM($names[] , (n CHAR) -> UPCASE(n))
```

Related Links

[Data Pipelines](#)

[Database, Tables, Views, and Indexes](#)

[Data Pipelines](#)

3.4.2 Data Types for User-Defined Data Pipeline Functions

This description provides the Ocient® packages that support the functionality for user-defined data pipeline functions.

3.4.2.0.1 Classes

`com.ocient.streaming.schema.OcientColumnType`

The class that enumerates all defined column types in the Ocient System.

Enum Value
TYPE_NONE
TYPE_INFERRED
TYPE_DELETED
TYPE_INT
TYPE_BIGINT
TYPE_FLOAT
TYPE_DOUBLE
TYPE_VARCHAR
TYPE_IPV4
TYPE_TIMESTAMP
TYPE_DATE
TYPE_BOOLEAN
TYPE_BINARY
TYPE_SMALLINT
TYPE_BYTE
TYPE_UUID
TYPE_HASH
TYPE_IP
TYPE_ST_POINT
TYPE_TIME
TYPE_DECIMAL
TYPE_ARRAY
TYPE_TUPLE

Enum Value

TYPE_ST_LINestring
TYPE_ST_POLYGON

`com.ocient.streaming.schema.OcientColumnTypeArguments`

The class that describes arguments of a column type. Not all column types require arguments.

Modifier And Type	Field and Description
<code>public static final OcientColumnTypeArguments</code>	NONE — A default instance indicating no arguments are necessary for a column type.

`com.ocient.streaming.schema.OcientColumnTypeArguments.Decimal`

A subclass of `OcientColumnTypeArguments` that includes information for `TYPE_DECIMAL`.

Modifier And Type	Field and Description
<code>public int</code>	<code>precision</code> — The number of digits in a number.
<code>public int</code>	<code>scale</code> — The number of digits to the right of the decimal point in a number.
<code>public</code>	<code>fullFormat</code> — True if the serialized format includes one byte for precision and one byte for scale.
<code>boolean</code>	You should always set this to <code>true</code> .

Constructor

`public Decimal(int precision, int scale, boolean fullFormat)` Constructs a newly allocated `OcientColumnTypeArguments` for `OcientColumnType.TYPE_DECIMAL`.

`com.ocient.streaming.schema.OcientColumnTypeArguments.Hash`

A subclass of `OcientColumnTypeArguments` that includes information for `TYPE_HASH`.

Modifier And Type	Field and Description
<code>public int</code>	<code>length</code> — The number of bytes in the hash value.

Constructor

`public Hash(int length)` — Constructs a newly allocated `OcientColumnTypeArguments` for `OcientColumnType.TYPE_HASH`.

`com.ocient.streaming.schema.OcientColumnTypeArguments.Array`

A subclass of `OcientColumnTypeArguments` that includes information for `TYPE_ARRAY`.

Modifier And Type	Field and Description
<code>public OcientColumnType</code>	<code>elementType</code> — The array element type.
<code>public OcientColumnTypeArguments</code>	<code>elementTypeArguments</code> — Type arguments for the array element type.

Constructor

`public Array(OcientColumnType elementType)` — Constructs a newly allocated `OcientColumnTypeArguments` for an array of `elementType`. The `elementTypeArguments` field initializes to `OcientColumnTypeArguments.NONE`.

`public Array(OcientColumnType elementType, OcientColumnTypeArguments elementTypeArguments, int offsetSizeOverride)` — Constructs a newly allocated `OcientColumnTypeArguments` for an array of `elementType`. You should set `offsetOverride` to 0.

`com.ocient.streaming.schema.OcientColumnTypeArguments.TopLevelTuple`

A subclass of tuple that includes information for top-level tuples.

Modifier And Type	Field and Description
<code>private List<OcientColumn></code>	<code>elementOcientColumns</code> — List of tuple element types.

Modifier And Type	Method and Description
<code>public List<OcientColumnTypeArguments></code>	<code>getTupleTypeArguments()</code> — Returns the list of tuple element type arguments.
<code>public List<OcientColumnType></code>	<code>getTupleTypes()</code> — Returns the list of tuple element types.
<code>public List<OcientColumn></code>	<code>getElementOcientColumns()</code> — Returns the list of tuple element types paired with their arguments.

Constructor

`public TopLevelTuple(List<OcientColumn> elementOcientColumns)` — Constructs a newly allocated `OcientColumnTypeArguments` for a `OcientColumnType.TYPE_TUPLE`.

`com.ocient.streaming.schema.OcientColumn`

Represents an Ocient column with information parsed from metadata of the Ocient System.

Modifier And Type	Method and Description
<code>public String</code>	<code>name()</code> — The name of the column.
<code>public OcientColumnType</code>	<code>type()</code> — The data type of the column.
<code>public OcientColumnTypeArguments</code>	<code>typeArguments()</code> — Some column types, for example <code>Decimal</code> , have arguments that are required to understand their format. A relevant subclass of <code>OcientColumnTypeArguments</code> returns, as appropriate, or <code>OcientColumnTypeArguments.NONE</code> .
<code>public boolean</code>	<code>required()</code> — Returns <code>true</code> if the user is required to provide this column. This is generally the case for non-defaulted, non-nullable columns.
<code>public boolean</code>	<code>nullable()</code> — Returns <code>true</code> if this column is configured as nullable, false otherwise.

`com.ocient.streaming.data.types.gis.Coordinate`

Represents a two-dimensional cartesian coordinate.

Modifier And Type	Field and Description
public double	x — The x coordinate of the point.
public double	y — The y coordinate of the point.

Modifier And Type	Method and Description
public double	x() — Returns the x coordinate of this point.
public double	y() — Returns the y coordinate of this point.

Constructor

public Coordinate(double x, double y) — Constructs a newly allocated Coordinate.

com.ocient.streaming.data.types.gis.STPoint

Represents a two-dimensional cartesian coordinate.

Modifier And Type	Field and Description
public Coordinate	coordinate — The ordered, x and y coordinate pair.

Modifier And Type	Method and Description
public static STPoint	fromWKT(String text) — Parses an STPoint from the WKT text.
public static STPoint	fromEWKT(String text) — Parses an STPoint from the EWKT text.
public double	getX() — Returns the x coordinate of this point.
public double	getY() — Returns the y coordinate of this point.
public STLinestring	toLinestring() — Promotes the STPoint to an STLinestring container. Useful when representing a geometry collection consisting of STLinestring and STPoint elements.
public STPolygon	toPolygon() — Promotes the STPoint to an STPolygon container. Useful when representing a geometry collection consisting of any geospatial element type.

Constructor

public STPoint(Coordinate coordinate) — Constructs a newly allocated STPoint.

com.ocient.streaming.data.types.gis.STLinestring

Geometric type that is composed of a sequence of N points.

Modifier And Type	Field and Description
public STPoint[]	points — The ordered sequence of points in the linestring.

Modifier And Type	Method and Description
public static STLinestring	fromWKT(String text) — Parses an STLinestring from the WKT text.
public static STLinestring	fromEWKT(String text) — Parses an STLinestring from the EWKT text.
public STPoint[]	points() — Returns the points of this linestring.
public STLinestring	toLinestring() — Returns this.
public STPolygon	toPolygon() — Promotes the STLinestring to an STPolygon container. Useful when representing a geometry collection consisting of any geospatial element type.

Constructor

public STLinestring(STPoint[] points) — Constructs a newly allocated STLinestring.

com.ocient.streaming.data.types.gis.STPolygon

Geometric type composed of N closed linestrings.

Modifier And Type	Field and Description
public STLinestring	exteriorRing — The exterior ring of the polygon.
public STLinestring[]	interiorRings — The interior rings of the polygon.

Modifier And Type	Method and Description
public static STLinestring	fromWKT(String text) — Parses an STPolygon from the WKT text.
public static STLinestring	fromEWKT(String text) — Parses an STPolygon from the EWKT text.
public STLinestring	exteriorRing() — Returns the exterior linestring.
public STLinestring[]	interiorRings() — Returns the interior linestrings.
public STPolygon	toPolygon() — Returns this.

Constructor

public STPolygon(STLinestring exteriorRing, STLinestring[] interiorRings) — Constructs a newly allocated STPolygon.

com.ocient.streaming.data.types.Timestamp

Timestamp represented as nanoseconds from the Unix® epoch.

Modifier And Type	Method and Description
public long	value() — Returns the number of nanoseconds after the Unix epoch.

Constructor

public Timestamp(long nanosFromEpoch) — Constructs a newly allocated Timestamp.

`com.ocient.streaming.data.types.Time`

Time that is represented as nanoseconds from midnight (00:00:00).

Modifier And Type	Method and Description
<code>public long</code>	<code>value()</code> — Returns the number of nanoseconds after midnight.

Constructor

`public Time(long nanosFromMidnight)` — Constructs a newly allocated `Time`.

`com.ocient.streaming.data.types.Decimal`

A class representing a decimal number with binary-coded decimal (BCD) storage, supporting specified precision and scale.

Modifier And Type	Field and Description
<code>public static long</code>	<code>MAX_PRECISION</code> — The maximum supported precision in the database.

Modifier

And

Type Method and Description

`public fromString(String s, byte precision, byte scale)` Creates a `Decimal` object from a string representation with the specified precision and scale. `s` — The string representation of the decimal (e.g., `Decimal "123.45", "-0.1"`). `precision` — The precision (total number of digits) of the decimal. `scale` — The scale (number of digits after the decimal point) of the decimal.

`public fromBigDecimal(java.math.BigDecimal d, byte precision, byte scale)` Creates a `Decimal` object from a `BigDecimal`, ensuring compatibility with the specified precision and scale. `d` — The `BigDecimal` to convert. `precision` — The precision (total number of digits) of the decimal. `scale` — The scale (number of digits after the decimal point) of the decimal.

`public fromDouble(double d, byte precision, byte scale)` Creates a `Decimal` object from a double value with the specified precision and scale. `d` — The double to convert. `precision` — The precision (total number of digits) of the decimal. `scale` — The scale (number of digits after the decimal point) of the decimal.

`public toString()` — Returns the `Decimal` string representation of this `Decimal`.

`String`

`public asLong()` — Converts this `Decimal` object to a long, truncating any fractional part.

`long`

`public asDouble()` — Converts this `Decimal` object to a double, preserving the full value including the fractional part.

`double`

`public bcd()` — Returns the BCD representation of this `Decimal` object.

`byte[]`

`public getFullFormat()` — Returns the full format representation, including precision and scale bytes followed by the BCD data padded to `MAX_PRECISION`.

`byte[]`

`public precision()` — Returns the precision of this `Decimal` object.

`int`

`public scale()` — Returns the scale of this `Decimal` object.

`int`

`com.ocient.streaming.data.types.OcientTuple`

A class representing a tuple with a defined schema of column types and associated values. Supports fixed-length and variable-length elements.

Modifier And Type	Method and Description
public int	numValues() — Returns the number of values currently stored in this tuple.
public void	setValue(int index, Object value) Sets or updates the value at the specified index, ensuring type compatibility. index — The index of the value to set (0-based). value — The value to set must match the column type at the index.
public void	set(List<Object> values) Sets or resets all values in the tuple from the provided list. values — The list of values to set must match the schema size and types.
public String	toString() — Returns a string representation of this tuple, including its schema and values (e.g., "TUPLE<INT,DOUBLE>(1,2.0)").
public List<Object>	get() — Returns the list of values in this tuple, in order.
public List<OcientColumnType>	getTypes() — Returns the list of column types defining the schema of this tuple, in order.
public List<OcientColumnTypeArguments>	getTypeArgs() — Returns the list of type arguments corresponding to the column types of this tuple.
Constructor	
public OcientTuple	OcientTuple(List<OcientColumnType> types, List<OcientColumnTypeArguments> typeArgs) Constructs an empty tuple with the specified schema of column types and type arguments. types — The list of column types defining the tuple schema. typeArgs — The list of type arguments corresponding to each column type.

3.5 Load Geospatial Data in Data Pipelines

Data pipeline loading supports the load of geospatial data for use with OcientGeo™ functionality in the Ocient Hyperscale Data Warehouse™. Use geospatial functions to manipulate this data. For details, see [Geospatial Functions](#).

Supported Geospatial Functions

These geospatial functions are supported for data pipeline loading:

- [Polygon Constructors](#)
- [Point Constructors](#) or [Point Constructors](#)
- [Point Constructors](#)
- [Point Constructors](#)
- [Linestring Constructors](#)
- [Linestring Constructors](#)
- [Linestring Constructors](#)
- [Linestring Constructors](#)
- [Polygon Constructors](#)
- [Polygon Constructors](#)
- [Polygon Constructors](#) or [Polygon Constructors](#)
- [Polygon Constructors](#)
- [Polygon Constructors](#)

Data Pipeline Loading of Geospatial Data Considerations

Auto-Casting Behavior of Source Data Types

The Ocient® System automatically casts geospatial data from the CHAR type to POINT, LINESTRING, or POLYGON types using the [Point Constructors](#), [Linestring Constructors](#), or [Polygon Constructors](#) functions, respectively. The data must be in WKT format for the casting to work. If your data is not in the WKT format, use one of the supported geospatial functions to transform your data to the required type.

Point Data Normalization During Data Pipeline Loading

During loading, the Ociant System automatically performs normalization of point data into a regular format used within Ociant. The Ociant System performs these operations on point data during the load:

- Constrain longitude to [-180, 180] and latitude to [-90, 90].
- Wrap around invalid coordinates using correct geographical handling.
- Snap points near the pole to the pole.
- Set the longitude of points on the pole to 0.
- Remove signed zeros from coordinates, so -0 becomes 0.

Polygon Rotation Convention

The Ociant System has a standardized polygon orientation convention. The standard convention has the polygon with a counterclockwise rotation of the outer polygon ring and a clockwise rotation of the inner polygon ring. Use the [Polygon Constructors](#) function to convert polygons to the standardized convention from systems that interpret polygon rotation differently.

Size Limit for Geospatial Data Types

The maximum allowed size of geospatial data types is 512 MiB.

Examples of Loading Geospatial Data

Use these examples to understand how to create a data pipeline to load geospatial data for different formats (point and WKT data).

Load Geospatial Data Using Point Data

Create the target table `cities` using the `CREATE TABLE` SQL statement with these columns:

- `id` — Integer
- `name` — Variable-length string
- `location` — Numeric point

```
CREATE TABLE cities (
  id INT,
  name VARCHAR,
  location ST_POINT
);
```

Create the `cities` data pipeline using the `CREATE PIPELINE` SQL statement for an Amazon® Web ServicesSM (AWSSM) S3 source with bucket `ocient-examples`, filter for the folder `metabase_samples/jsonl/cities.jsonl`, and region `us-east-1`. Use the JSON format to load a JSON file into the `cities` table. Specify a numeric point using the [Point Constructors](#) function with the longitude and latitude coordinates.

```
CREATE PIPELINE cities
  SOURCE S3
    BUCKET 'ocient-examples'
    FILTER 'metabase_samples/jsonl/cities.jsonl'
    REGION 'us-east-1'
  EXTRACT
    FORMAT json
  INTO cities
  SELECT
    $id AS id,
    $name AS name,
    ST_POINT($longitude, $latitude) AS location;
```

Load Geospatial Data in WKT Format

Create the target table `geospatial_table` using the `CREATE TABLE` SQL statement with these non-nullable columns:

- `ts` — TimeKey® with daily time bucket as a timestamp
- `row_number` — Variable-length string
- `point` — POINT object
- `linestring` — LINESTRING object
- `polygon` — POLYGON object

```
CREATE TABLE geospatial_table(
  ts TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  row_number INT NOT NULL,
  point ST_POINT NOT NULL,
  linestring ST_LINESTRING NOT NULL,
  polygon ST_POLYGON NOT NULL
);
```

Create the `geospatial_pipeline` data pipeline using the `CREATE PIPELINE` SQL statement for an S3 source with bucket `ocient-examples`, filter for the folder `gis_small/gis_types/*.csv`, and region `us-east-1`. Use the CSV format to load a CSV file into the `geospatial_table` table. The file contains geospatial data in WKT format in the sixth, ninth, and twelfth columns with POLYGON, LINESTRING, and POINT data, respectively. Specify the polygon data in WKT format by using the [Polygon Constructors](#) function and then reorient the resulting polygon counter-clockwise for the exterior and clockwise for the interior by using the [Polygon Constructors](#) function.

During the load, the Ocient System automatically casts the WKT-formatted data in the three columns to the corresponding geospatial data types.

```
CREATE PIPELINE geospatial_pipeline
  SOURCE s3
    BUCKET 'ocient-examples'
    FILTER 'gis_small/gis_types/*.csv'
    REGION 'us-east-1'
  EXTRACT
    FORMAT csv
INTO geospatial_table
SELECT
  $1 AS ts,
  $2 AS row_number,
  $12 AS point,
  $9 AS linestring,
  ST_FORCECCW(ST_POLYGONFROMTEXT($6)) AS polygon;
```

Related Links

[Geospatial Functions](#)

[Load Data](#)

3.6 Load Metadata Values in Data Pipelines

The data pipeline loading infrastructure enables the loading of certain metadata values of a source record into the Ocient® System. Metadata represents information about a record that is not represented in the record data itself, e.g., the filename of the source record or the timestamp when a record was created in Apache® Kafka®.

Syntax

The `METADATA` function can load a variety of metadata along with each record of the data pipeline, or it can use key-value pairs to return the value in the filename metadata.

To load a metadata value into a column, use the **METADATA** function in the **SELECT** SQL statement and replace **key** with the key you want to load. You can use transformation functions to achieve the final data type or allow the pipeline to apply automatic casting.

To retrieve a value from a key-value pair in the filename metadata, specify the first argument as **'hive_partition'** and provide a search string for the key. In this case, the filename must follow Apache® Hive™ naming standards. The standard embeds field names and key-value pairs in path segments, such as **/year=2019/month=2/data.parquet**. If there are duplicate key-value pairs in the filename, the function uses the last pair. The function returns a string with the value of the associated key, and returns **NULL** if the file does not contain the search string or the filename does not follow Hive naming standards.

METADATA(**key**)

METADATA('hive_partition',search_string)

Parameter	Data Type	Description
key	string	The name of the specified metadata key. Example: 'topic'
search_string	string	The search string to use as a key for searching the filename metadata to return the value from the key-value pair as a string.

Metadata fields cannot be combined in a transformation function with other source data.

Supported Values for the Metadata Key

The metadata values you can load differ based on the source type. This table lists the available values for the metadata keys specified using the **key** argument by source type and their returned data type. Metadata key values are not case-sensitive.

Metadata Key	Source Type	Returned Value Data Type	Description
row_id	File Load and Kafka Load	UUID	A unique row identifier that is generated by the Ocident System during loading. The Ocident System guarantees this identifier is unique for each loaded record because the generation is based on internal mechanisms that enforce record deduplication.
filename	File Load	VARCHAR	The filename and full file path indicate the source of the record.
file_modified_timestamp	File Load	TIMESTAMP	The timestamp that indicates when the file was last modified on the data source.
offset	Kafka Load	BIGINT	The Kafka offset of the record.

Metadata Key	Source Type	Returned Value Data Type	Description
partition	Kafka Load	BIGINT	The name of the Kafka partition that indicates the source of the record.
record_timestamp	Kafka Load	TIMESTAMP	The timestamp that indicates when the record was created in the Kafka Broker.
source_record_id	File and Kafka Load	UUID	A unique source row identifier that is generated by the Ocient System during loading. The Ocient System guarantees this unique identifier for each source record with a specific SOURCE and EXTRACT configuration because the generation is based on source names, source record numbers, and source record offsets.
topic	Kafka Load	VARCHAR	The name of the Kafka topic that indicates the source of the record.

Examples

Load Filename Metadata

This example snippet uses the `CREATE PIPELINE` SQL statement to load data and the source filename with each record in a pipeline. For details about this pipeline definition and setup, see [Data Pipeline Load of CSV Data from S3](#).

```
CREATE PIPELINE orders_pipeline
SOURCE
  S3
    ENDPOINT 'https://s3.us-east-1.amazonaws.com'
    BUCKET 'ocient-examples'
    FILTER 'metabase_samples/csv/orders.csv'
EXTRACT
  FORMAT csv
  NUM_HEADER_LINES 1
INTO public.orders
SELECT
  $1 as id,
  $2 as user_id,
  $3 as product_id,
  $4 as subtotal,
  $5 as tax,
  $6 as total,
  $7 as discount,
  $8 as created_at,
  $9 as quantity,
  METADATA('filename') AS filename_column;
```


Load Specified Metadata from Filename

This example uses the `CREATE PIPELINE` SQL statement to load data and the value of the search string 'year' of the filename with each record in a pipeline. For details about this pipeline definition and setup, see [Data Pipeline Load of CSV Data from S3](#). The example assumes the filename follows Hive-style naming standards.

```
CREATE PIPELINE orders_pipeline
SOURCE
  S3
    ENDPOINT 'https://s3.us-east-1.amazonaws.com'
    BUCKET 'ocient-examples'
    FILTER 'metabase_samples/csv/orders.csv'
EXTRACT
  FORMAT csv
  NUM_HEADER_LINES 1
INTO public.orders
SELECT
  $1 as id,
  $2 as user_id,
  $3 as product_id,
  $4 as subtotal,
  $5 as tax,
  $6 as total,
  $7 as discount,
  $8 as created_at,
  $9 as quantity,
  METADATA('hive_partition','year') AS hive;
```

Related Links

[Load Data](#)

[Data Pipelines](#)

[Transform Data in Data Pipelines](#)

3.7 Deduplication in Data Pipelines

Exactly Once Loading in Pipelines

The job of a pipeline is to load the data assigned to it exactly once. A pipeline maintains its own position during a load and enforces deduplication logic. The lifecycle of a pipeline object in the database defines both the deduplication logic and load position.

Because the pipeline, not the target tables, defines the load position and deduplication scope, if the system truncates target tables or if the system drops and recreates a target table, the pipeline continues from its last position. It does not reset to the beginning position of the load if you attempt to load data into a new table.

Key benefits are:

- Pipeline Events — During the life of a pipeline, you can stop and resume the pipeline without duplicating source data or changing the position in a load.
- Pipeline Updates — You can modify pipelines using the `CREATE OR REPLACE` SQL statement to update the transforms in a pipeline but maintain the current position in the load.
- Exactly Once Loading — In the event of transient system failures, the pipeline ensures exactly-once loading into the target table through replay and deduplication guarantees.

Two concepts are critical to understand how Ociant® controls which data is loaded in a pipeline and how Ociant pipelines ensure exactly-once loading.

1. Load Position
2. Deduplication Scope

Load Position

You can control the pipeline by starting, stopping, modifying, and resuming the pipeline. While running, each pipeline maintains its position in the overall load. When it has reached the end of all data, the pipeline completes and does not attempt to reload the data. To load the data again, you must create a new pipeline or drop and recreate the pipeline.

- For an Apache® Kafka®-based load, the position consists of using consumer group offsets appropriately to create checkpoints.
- For File-based loads, file details are stored in the `sys.pipeline_files` system catalog table and the system updates the status of each file as the load progresses.

In both cases, the load position defines where the pipeline resumes loading if stopped and resumed.

Deduplication Scope

The deduplication scope defines the conditions under which a pipeline does not cause the same row to appear twice in a target table. This deduplication is how a pipeline ensures that it is safe to replay data during loading. This situation is common in failover situations or when you stop and restart a pipeline.

The deduplication scope is the unique combination of:

1. A pipeline object
2. A target table

A pipeline guarantees that if it sends the same row to the same target table twice, the system only loads it into the table once. If you drop and recreate either the pipeline or the table, then the situation is a new deduplication scope.

If you begin sending data to a new target table, this situation is a new deduplication scope. A row for the same record can appear in both the old and new tables.

Restarts and Deduplication

You can restart a pipeline without creating duplicate data in the target tables. However, there are some limitations and key assumptions for each data source.

For example, in a file load, if you modify the contents of a file after the pipeline has started, then you can experience duplication of data or missed rows.

For more details and key considerations, see [Data Pipelines](#) and [Data Pipelines](#).

Deduplication and DROP TABLE

You can drop a table using the `DROP TABLE` SQL statement and then recreate it with the same name. In this case, an existing pipeline loads into the new table. When you drop a table and recreate it, these actions do not update the load position of the pipeline. When you restart the pipeline, it resumes from the load position of the pipeline, and all records from that position forward are loaded with a new deduplication scope.

Deduplication and TRUNCATE TABLE

You can truncate a table (using the `TRUNCATE TABLE` SQL statement) that is the target of a pipeline whether the pipeline is running or stopped. These actions do not update the load position of the pipeline or change the deduplication scope. The system deduplicates any data that was loaded before the truncate and replays after the truncate for any reason (e.g., restart or transient failure). Also, the system loads all new data.

When you truncate the target tables and restart the pipeline, the Ocint System does not reload the data.

Load Duplicate Data

Sometimes, you might want to load the same data multiple times, but the pipeline load position and deduplication scope prevent this.

If you want to load a second copy of the source data, you can follow one of these approaches:

File-Based Pipelines

- Drop and recreate the pipeline to reset the `sys.pipeline_files` system catalog table.
- Create a second pipeline with a new name and the same configuration.

Kafka-Based Pipelines

- Drop the pipeline and recreate it with the same name, which defaults to the same consumer `group.id` but has a new deduplication scope. Then, reset the consumer group offsets using Kafka tools to the chosen starting point.
- Create a new pipeline with a different name and load from the beginning of the topic using `auto.offset.reset`.

Related Links

[Data Pipelines](#)

[Data Pipelines](#)

[Data Pipelines](#)

[Data Pipelines](#)

3.8 Error Tolerance in Data Pipelines

Error Limits

The error limit for SQL statements such as `START PIPELINE my_pipeline ERROR LIMIT 100` defines the number of record-level errors that a batch of files tolerates before failing the entire pipeline.

The error process is:

- Periodically, the Ocint® System assigns a batch of pending files to a Loader Node.
- The system enforces the maximum error limit for each batch.
- Files that have at least one record-level error reach the terminal status `LOADED_WITH_ERRORS` after all processing is complete.
- If the file batch reaches the error limit, the current file reaches the status `FAILED`, and then the pipeline status becomes `FAILED`. Processing stops on the pipeline.

By default, `BATCH` pipelines run with `ERROR LIMIT 0`. This permits zero errors on a pipeline. This setting stops the pipeline on the first error and stores the error in `sys.pipeline_errors`. When you restart the pipeline, it retries the failed file and failed record. The system does not duplicate any previously loaded data. To complete the load, fix the issue with the data or the pipeline definition or increase the error limit when you restart the pipeline.

Unrecoverable File Errors

You can manage unrecoverable file errors such as Gzip decompression errors, tokenization errors, or missing files by starting the pipeline with the `FILE_ERROR` option.

The strictest setting is `FILE_ERROR FAIL`. When you use the `FAIL` setting, any file-level error causes the pipeline to fail. If any file is missing or cannot be processed, the pipeline marks the file `FAILED`, sets the pipeline status to `FAILED`, and processing stops on the pipeline.

The most tolerant setting is `FILE_ERROR TOLERATE`. Use this setting in a statement such as `START PIPELINE my_pipeline ERROR FILE_ERROR TOLERATE`. When you use the `TOLERATE` setting, missing files reach the terminal status `SKIPPED`. Files that encounter other file-level errors reach the terminal status `LOADED_WITH_ERRORS`. When the Ociant System encounters a file-level error, processing stops on the file and continues with the next file in the partition. When you use the `TOLERATE` setting, the pipeline automatically executes with an unlimited error limit.

Recoverability

If a pipeline fails when it tolerates no errors, manually fix the row or file where the error occurs and restart the pipeline so that the load proceeds with all records correctly deduplicated. Restarting the pipeline without fixing the error but with tolerance of record or file-level errors also allows the load to proceed with all records correctly deduplicated. However, restarts do not guarantee correct deduplication if you apply manual fixes after this point.

When a pipeline fails, the number of loaded rows is nondeterministic. As a result, the Ociant System does not guarantee the reflection of any manual modification of files with the statuses `SKIPPED`, `LOADED_WITH_ERRORS`, or `FAILED` in a restart operation. However, if the pipeline fails due to a low error limit, you can raise the error limit on a restart operation to allow the pipeline to make further progress.

You can use the `BAD_DATA_TARGET` option to capture failing records for troubleshooting and reloading.

Restarts and Deduplication

Ociant pipelines enable the restart of a pipeline without creating duplicate data in the target tables. However, there are some limitations for each type of data source. For details, see [Data Pipelines](#)

Related Links

[Data Pipelines](#)

[Data Pipelines](#)

[Data Pipeline Load of JSON Data from Kafka](#)

3.9 Monitor Data Pipelines

You can monitor data pipelines through a combination of:

1. System catalog tables
2. Information schema views
3. Operational metrics endpoints for use with operations monitoring tools

This tutorial describes how to use these system catalog objects to monitor your pipelines:

- `information_schema.pipelines`
- `information_schema.pipeline_status`
- `sys.pipelines`
- `sys.pipeline_events`
- `sys.pipeline_errors`
- `sys.pipeline_files`
- `sys.pipeline_partitions`
- `sys.pipeline_metrics`
- `sys.pipeline_metrics_info`

A complete reference of system catalog tables and views is available in the System Catalog;

Pipelines Information Schema

After you create a pipeline, two convenient views are most useful for observing the pipeline. At a glance, you can view the configuration of your pipeline in `information_schema.pipelines` and the status information in `information_schema.pipeline_status`.

Access these views conveniently with the **SHOW** command.

SQL

```
> SHOW PIPELINES;
```

database_name	pipeline_name	loading_mode	source_type	data_format	created_at
test	my_pipeline	BATCH	FILESYSTEM	CSV	2024-02-07 18:01:22

Fetches 1 rows

```
> SHOW PIPELINE_STATUS;
```

database_name	pipeline_name	table_names	status	status_message
test_database	test_pipeline	['test_schema.test_table']	COMPLETED	Completed processing pipeline

Fetches 1 rows

For a quick look at the progress of a pipeline, the **SHOW PIPELINE_STATUS** command provides:

- Name
- Target tables
- Current status
- Last status message that was received in processing
- Percentage completion (from 0.0 - 1.0)
- Duration of pipeline execution
- Number of files and records processed
- Error counts

To join these two views to other catalog tables, use the **pipeline_name** to join to the **sys.pipelines** table and then join tables using the **pipeline_id**.

Pipeline Catalog Tables

Beyond the information schema views, many catalog tables offer details about your pipelines.

Table	Description	Importance
sys.pipelines	List of pipelines that exist and the configuration information.	High
sys.pipeline_events	Event log generated through the lifecycle of a pipeline.	High
sys.pipeline_errors	Log of errors created by pipelines during execution for troubleshooting record-level errors or system errors.	High
sys.pipeline_files	List of files that each file-based pipeline is loading or has loaded.	High
sys.pipeline_partitions	List of {{Kafka}} source partitions on each Kafka pipeline with relevant metadata for offsets.	Medium
sys.pipeline_metrics	Metrics that are regularly collected for each pipeline to capture performance and progress.	Medium

Table	Description	Importance
<code>sys.pipeline_metrics_info</code>	Descriptive details about the available pipeline metrics. Provides information about how to interpret each metric type.	Low

Monitor Activity and Events

While your pipeline is running, the pipeline generates events in the `sys.pipeline_events` system catalog table to mark significant checkpoints when something has occurred. You can use this table to observe progress and look for detailed messages about changes in the pipeline.

For example, the `sys.pipeline_events` table captures events when a pipeline is created, started, stopped, fails, and completed. This table also captures relevant system events such as rebalancing Kafka partitions or retrying a process due to a transient failure.

Pipeline events include messages from many different tasks. These tasks are the background processes that execute across different Loader Nodes during pipeline operation.

Monitor Files or Partitions

For file-based loads, the `sys.pipeline_files` system catalog table contains one row for each file, which contains the status of the file and other file metadata such as filename, creation or modified timestamps, and the file size. The pipeline updates this list of files when it is started and maintains status information as it processes files. You can use this list to understand which files are loading, which succeeded, and which failed.

For Kafka partition-based loads, the `sys.pipeline_partitions` system catalog table contains one row for each partition, which contains the current offsets and record counts. You can use this information to understand how partitions are distributed across Loader Nodes and also to observe lag on each topic and partition.

Monitor Performance

Pipeline performance metrics are captured in the `sys.pipeline_metrics` system catalog table. This table contains samples of the metrics over time. The Ocient® System collects samples regularly. You can query the samples using standard SQL queries to inspect the behavior of a single pipeline, a single task on a Loader Node, or across all pipelines. The `sys.pipeline_metrics_info` system catalog table provides metadata that explains the individual metric types.

See [Discover Insights From System Catalog Tables](#) for a detailed description of how to use `sys.pipeline_metrics` and `sys.pipeline_metric_info` to monitor the performance of your pipelines. For the system catalog table definitions, see System Catalog.

Metrics Endpoints

In the same way that the Ocient System monitors system performance using [Statistics Monitoring](#) endpoints with operational monitoring tools, data pipelines expose an API that allows operators to capture performance metrics over time.

Access Metrics Endpoints

You query the metrics using an HTTP endpoint on each Loader Node located at `<loader_ip_address_or_hostname>:8090/metrics`.

For example, you can retrieve metrics from a Loader Node by executing this command from a shell on the node or by running a similar request from a monitoring agent on the node.

```
curl http://localhost:8090/metrics/v1
```

Each metric is represented as a JSON object that contains:

- Metric name
- Metric value
- Scope information
- Query time
- Value units
- Whether or not the metric is incremental

Metrics can be incremental or instantaneous. Incremental metrics accumulate value over time, so they never decrease in value. Instantaneous metrics capture the current value, so they can decrease.

Metrics Scope

Similar to the `sys.pipeline_metrics` system catalog table, the Ocient System scopes metrics to a Loader Node, a pipeline task, or a partition. You can determine the scope of a metric based on the presence of the `pipeline_name`, `pipeline_name_external`, and `partition` keys or reference the details in [Monitor Data Pipelines](#).

Scope	Description
Loader	Metrics scoped to a loader do not have information about pipelines or partitions. Because metrics are collected on each node, this scope contains all activity on the node where you access the endpoint.
Pipeline	Metrics scoped to a pipeline have a key named <code>pipeline_name</code> with a value that corresponds to the identifier of the extractor task with hyphens removed. The metrics also have a key named <code>pipeline_name_external</code> with a value that corresponds to the identifier of the pipeline.
Partition	Metrics scoped to a partition have the same keys as metrics scoped to a pipeline. The metrics also have a key named <code>partition</code> with a value that is the stream source identifier of the partition.

Examples

This JSON structure contains a metric scoped to a Loader Node.

```
{
  "name": "usage.os.heap",
  "time": 1713993689613976,
  "timestamp": "2024-04-24T21:21:29.613976119Z",
  "value": 317855288,
  "units": "bytes",
  "incremental": false
}
```

This JSON structure contains a metric scoped to a pipeline.

```
{
  "name": "count.record.sent",
  "time": 1713994532578624,
  "timestamp": "2024-04-24T21:35:32.578624253Z",
  "value": 100000,
  "units": "unitless",
  "incremental": true,
  "pipeline_name_external": "test-rest-external-create",
  "pipeline_name": "test"
}
```

This JSON structure contains a metric scoped to a partition.

```
{
  "name": "duration.record.processed",
  "time": 1714509826724477,
  "timestamp": "2024-04-30T20:43:46.724477142Z",
  "value": 322432,
  "units": "milliseconds",
  "incremental": true,
  "pipeline_name_external": "test-rest-external-create",
  "pipeline_name": "58ff07cf732846f9830b728b3a8e4a7a",
  "partition": "226979",
  "sink_index": "0"
}
```

Pipeline Metrics Details

Name	Scope	Description	Importance
count.error	pipeline	The count of failures when processing records during pipeline execution for a pipeline.	High
count.file.fail	pipeline	The number of files that have failures.	High
count.file.processed	pipeline	The number of files that have been extracted from the source for processing. The Ocient System tracks this number before tokenization, transformation, or sending to the sink. This number does not indicate the durability of the files or if files load with errors.	Medium
count.file.total	pipeline	The total number of files in the static file list.	High
count.partition.durable.offset.committed	partition	The last committed offset of the Kafka partition for the consumer group.	High
count.partition.offset.lag	partition	count.partition.offset.max - count.partition.durable.offset.committed for a specified Kafka partition.	Medium
count.partition.offset.max	partition	The maximum offset of the Kafka partition.	High
count.partition.records.failed	partition	The count of failures when processing records during pipeline execution for a specified Kafka partition.	High
count.partition.records.processed	partition	The number of records that have been extracted for processing on a specified Kafka partition.	Medium

Name	Scope	Description	Importance
<code>count.record.bytes.maximum</code>	partition	The largest record processed in bytes.	Medium
<code>count.record.bytes.minimum</code>	partition	The smallest record processed in bytes.	Medium
<code>count.record.bytes.transformed</code>	pipeline	The number of bytes that have been transformed, excluding delimiters.	Medium
<code>count.record.sent</code>	pipeline	The number of records that have been fully transformed and sent to the sink.	High
<code>count.sink.bytes.queued</code>	pipeline	The number of transformed records queued to send to the sink.	Medium
<code>count.stream.paused</code>	partition	The number of times the processing of the record stream has been paused due to reaching a high watermark.	Medium
<code>count.stream.unpaused</code>	partition	The number of times the processing of the record stream has been unpaused due to reaching a low watermark.	Medium
<code>duration.file.listed</code>	pipeline	The number of milliseconds spent listing files from the data source.	High
<code>duration.record.durable</code>	partition	The duration in milliseconds for making records durable. $\leq$ The database takes time to make records durable, so this duration might be longer than the record processing duration.	Medium
<code>duration.record.processed</code>	partition	The duration in milliseconds for reading the records from the source, transforming them, and buffering them for sending to the sink.	Medium
<code>uptime</code>	Loader Noce	The number of seconds after the last restart of the extraction process.	High
<code>usage.os.heap</code>	Loader Noce	The number of bytes used in the heap.	High
<code>usage.os.percentage.cpu</code>	Loader Noce	The percentage of CPU used by the <code>{{JVM}}</code> .	High

Related Links

System Catalog

3.10 Manage Errors in Data Pipelines

When you load data into Ocident®, you can manage the way that the system responds to errors. This guide outlines different mechanisms for managing errors through examples.

The guide uses the same setup as [Data Pipeline Load of JSON Data from Kafka](#) but it modifies that pipeline to illustrate how to manage errors that you might encounter in data pipeline functionality.

You learn about:

- Bad data and error handling
 - Inspecting record-level processing errors
 - Configuring a bad data target
- Controlling pipeline behavior when you encounter errors
 - File-level error behavior
 - Record-level error behavior
- Understanding pipeline statuses in file loading

Step 1: Complete the Load JSON Data from Kafka Tutorial

See [Data Pipeline Load of JSON Data from Kafka](#) and complete the tutorial, then return to this guide for Step 2.

Step 2: Modify the Pipeline with a Bad Data Target

Bad data targets are the way to capture record-level errors for later troubleshooting and correction. Data pipelines support an Apache® Kafka® bad data target that allows you to send any failing records to a designated Kafka topic. This topic supports easy replay of those records in a separate recovery pipeline or inspection of the failing records.

For details about bad data targets, see the [Data Pipelines](#).

Stop the Data Pipeline

Stop the `orders_pipeline` data pipeline using the `STOP PIPELINE` SQL statement.

```
STOP PIPELINE orders_pipeline;
```

Inspect the pipeline status using this `SELECT` SQL statement and the `pipeline_status` information schema view.

```
SELECT
  status
FROM information_schema.pipeline_status
WHERE
  pipeline_name = 'orders_pipeline';
```

While the pipeline is stopping, the status might appear as `STOPPING` while the pipeline wraps up work in process and commits offsets back to Kafka. After the status changes to `STOPPED`, you can proceed to the next step. A pipeline might not be in a running state when you modify it.

Modify the Data Pipeline

In this example, use the `CREATE PIPELINE OR REPLACE` SQL statement to update the data pipeline. This statement maintains the same offsets with the Kafka topic and ensures that data is deduplicated correctly when the pipeline resumes.

Add the `BAD_DATA_TARGET` option and specify the Kafka target and details. Use the same Kafka broker as the source and bad data target in this example.

```
CREATE PIPELINE OR REPLACE orders_pipeline
BAD_DATA_TARGET
  KAFKA
    BOOTSTRAP_SERVERS '192.0.2.1:9092'
    TOPIC 'orders_errors'
```

```

SOURCE
  KAFKA
    BOOTSTRAP_SERVERS '192.0.2.1:9092'
    TOPIC 'orders'
EXTRACT
  FORMAT json
INTO public.orders
SELECT
  $id as id,
  $user_id as user_id,
  $product_id as product_id,
  $subtotal as subtotal,
  $tax as tax,
  $total as total,
  $discount as discount,
  $created_at as created_at,
  $quantity as quantity;

```

Step 3: Start the Data Pipeline with an Error Limit

Start the `orders_pipeline` data pipeline. In this case, set a maximum error limit so that the pipeline fails if 10 errors occur. An `ERROR LIMIT` might not be chosen for a continuously streaming pipeline if it is critical to keep up with newly arriving data, because the pipeline can stop suddenly and cause data freshness to fall behind. However, this example shows how error limits behave and how error states appear in pipelines.

```

START PIPELINE orders_pipeline
ERROR LIMIT 10;

```

The pipeline status changes to `STARTED` to indicate that it has begun processing.

For file-based loads, the `START PIPELINE` SQL statement includes options for `FILE_ERRORS` that are useful for controlling error handling in different file failure modes. Options include failing on error, skipping on file error, and tolerating file-level errors. See the [Data Pipelines](#) for details.

Step 4: Produce Data with Errors into Kafka

With the data pipeline in place and running, data immediately begins to load off of the Kafka topics that are configured in the pipeline from the last committed offsets after the pipeline stopped.

For test purposes, `kafkacat` is a helpful utility that makes it easy to produce records in a topic. For example, if you have a file of sample data `orders_with_errors.jsonl` in a JSONL format (newline-delimited JSON records), you can run this command to send those records into your Kafka broker. `<broker_ip_address>` is the IP address of the Kafka broker and `<topic_name>` is the name of the Kafka topic.

```
kafkacat -b <broker_ip_address>:9092 -t <topic_name> -T -P -l orders_with_errors.jsonl
```

Save an example document to your file system to use for this test. For this example, you can download an example file from https://ocient-examples.s3.amazonaws.com/metabase_samples/jsonl/orders_with_errors.jsonl and save it to `~/orders_with_errors.jsonl`.

Assume the broker is running at `192.0.2.1`, to send data into the `orders` topic defined in the pipeline definition, run this command.

```
kafkacat -b 192.0.2.1:9092 -t orders -T -P -l orders_with_errors.jsonl
```

This command pushes the entire JSONL file of messages into Kafka with one record per line. As these records are produced into Kafka, the running pipeline begins to load them into Ocint.

Step 5: Observe Loading Progress

With your pipeline running, data immediately begins to load from the Kafka topic. The pipeline creates parallel Kafka Consumers for each partition. If there are more partitions than processing cores available, the pipeline automatically handles spreading consumers across available processing cores and Loader Nodes.

You can check the pipeline status and progress by querying the `information_schema.pipeline_status` view or running the command `SHOW PIPELINE_STATUS`. Use the view to query the name of the pipeline, status, number of failed records, and the status message.

SQL

```
SELECT
    pipeline_name, status, records_failed, status_message
FROM
    information_schema.pipeline_status;
```

Output

SQL

pipeline_name	status	records_failed	status_message
orders_pipeline	FAILED	11	Started processing pipeline orders_pipeline

Inspect Errors

Examine the pipeline errors directly using the `sys.pipeline_errors` system catalog table.

SQL

```
SELECT count(*) FROM sys.pipeline_errors;
```

Output

SQL

```
count(*)
-----
11
```

Using this table, you can see detailed error messages, the source of the error, the column that encountered the error, and other helpful details to help troubleshoot why the record did not load.

```
SELECT * from sys.pipeline_errors;
```

Output

pipeline_id	extractor_task_id	error_index	error_type
f3ca5a1c-1bf4-40a4-8d01-b8c028fdffb58	1ecc5c4a-613f-48a5-8234-2257317a4a85	1	TRANSFORMATION_ERROR
f3ca5a1c-1bf4-40a4-8d01-b8c028fdffb58	1ecc5c4a-613f-48a5-8234-2257317a4a85	2	TRANSFORMATION_ERROR
f3ca5a1c-1bf4-40a4-8d01-b8c028fdffb58	1ecc5c4a-613f-48a5-8234-2257317a4a85	3	TRANSFORMATION_ERROR
f3ca5a1c-1bf4-40a4-8d01-b8c028fdffb58	1ecc5c4a-613f-48a5-8234-2257317a4a85	4	TRANSFORMATION_ERROR
f3ca5a1c-1bf4-40a4-8d01-b8c028fdffb58	1ecc5c4a-613f-48a5-8234-2257317a4a85	5	TRANSFORMATION_ERROR
f3ca5a1c-1bf4-40a4-8d01-b8c028fdffb58	1ecc5c4a-613f-48a5-8234-2257317a4a85	6	TRANSFORMATION_ERROR
f3ca5a1c-1bf4-40a4-8d01-b8c028fdffb58	1ecc5c4a-613f-48a5-8234-2257317a4a85	7	TRANSFORMATION_ERROR
f3ca5a1c-1bf4-40a4-8d01-b8c028fdffb58	1ecc5c4a-613f-48a5-8234-2257317a4a85	8	TRANSFORMATION_ERROR
f3ca5a1c-1bf4-40a4-8d01-b8c028fdffb58	1ecc5c4a-613f-48a5-8234-2257317a4a85	9	TRANSFORMATION_ERROR
f3ca5a1c-1bf4-40a4-8d01-b8c028fdffb58	1ecc5c4a-613f-48a5-8234-2257317a4a85	10	TRANSFORMATION_ERROR
f3ca5a1c-1bf4-40a4-8d01-b8c028fdffb58	1ecc5c4a-613f-48a5-8234-2257317a4a85	11	TRANSFORMATION_ERROR

For a more abbreviated view, query a few columns in the table. This example queries the error type, error code, error message, and name of the column.

```
SELECT
  error_type, error_code, error_message, column_name
FROM sys.pipeline_errors;
```

Output

error_type	error_code	error_message
TRANSFORMATION	OLT01	Failed to transform value 'not_int' using AsciiIntTransform. Details: In
TRANSFORMATION	OLT01	Failed to transform value '2019-08-22T:30:42.392Z' using ByteBufferAsc
TRANSFORMATION	OLT01	Failed to transform value 'not_decimal' using StringDoubleTransform. D
TRANSFORMATION	OLT01	Failed to transform value '55.11' using AsciiIntTransform. Details: In
TRANSFORMATION	OLT01	Failed to transform value 'not_int' using AsciiIntTransform. Details: In
TRANSFORMATION	OLS00	Failed to load transformed value 'null' to sink after ZCNullIfEmptyTra
TRANSFORMATION	OLS00	Failed to load transformed value 'null' to sink after ZCNullIfEmptyTra
TRANSFORMATION	OLS00	Failed to load transformed value 'null' to sink after ZCNullIfEmptyTra
TRANSFORMATION	OLS00	Failed to load transformed value 'null' to sink after ZCNullIfEmptyTra
TRANSFORMATION	OLT03	Failed to transform value '2018-06-26' using ByteBufferAsciiTimestampW
TRANSFORMATION	OLS00	Failed to load transformed value 'null' to sink after ZCNullIfEmptyTra

From this result set, you can quickly inspect error messages such as the first error message.

```
Failed to transform value 'not_int' using AsciiIntTransform. Details: Invalid ASCII digit: 'n'. Expected ASCII
digit in [0, 9].
```

This message occurs for the `user_id` column. From this information, you can see that you have some bad data in the data set.

In other errors, you can see the error code OLS00 and a message that indicates some issues with NULL handling. The table has a NOT NULL setting on a few columns, but the data is missing on the source records. Fix this by updating the pipeline to supply a default, or you can modify the table schema if you expect NULLs to occur and you do not have a reasonable default. For a continuous pipeline, you might expect to see an error like this if something changes in the data source.

Inspect Bad Data

To observe the original data, you can go to the original source file from this example, or you can load the specific message from the bad data topic that was configured to inspect it.

Using the `kafkacat` utility, you can consume the messages on the topic. The following command consumes one message on the `orders_errors` topic:

```
kafkacat -C -b 192.0.2.1:9092 -t orders_errors -o -1 -c 1
```

Output

```
{"id": 1011, "user_id": 3, "subtotal": 76.83, "tax": 5.28, "total": 82.11, "discount": null, "created_at":
```

A quick inspection of the source data reveals that this record is missing the `product_id` field, but this is a required column in our table.

If you run this command again, but with the `-J` flag, you can inspect the headers that are associated with the pipeline. The headers in the bad data target capture important metadata from the data pipeline execution that you can use to troubleshoot the failing data including the `error_message`.

```
kafkacat -C -b 192.0.2.1:9092 -t orders_errors -o -1 -c 1 -J
```

Output

```
{"topic":"orders_errors","partition":0,"offset":89,"tstype":"create","ts":1713566346480,"broker":1,"header":1}
```

Next Steps

Now that you have loaded data into tables, you can explore a few advanced topics including handling pipeline errors, loading from other data sources, loading different data formats, and transforming data.

Related Links

[Data Formats for Data Pipelines](#)

[Data Formats for Data Pipelines](#)

[Data Types for Data Pipelines](#)

[Transform Data in Data Pipelines](#)

[Data Pipelines](#)

[Data Pipeline Loading Errors](#)

3.11 Frequently Asked Questions for Data Pipelines

The questions featured in this topic cover information that you need most often when you start creating and running data pipelines.

Running Pipelines

Who can run a data pipeline?

Any database user with EXECUTE pipeline privileges can run a data pipeline on their database. This privilege allows database administrators to distinguish the users who can create pipelines from those who can start them.

To learn more about access control for pipelines, see [Data Control Language \(DCL\) Statement Reference](#).

Do pipelines run automatically after creation?

No, pipelines have a user-controlled lifecycle that starts with creation. After you create a data pipeline, you determine when to start the pipeline. A pipeline either completes successfully or moves to a failed status if the Ocient® System reaches an error limit.

For a basic batch file load, you would first use the CREATE PIPELINE SQL statement and then the START PIPELINE statement. After all data is successfully loaded, the pipeline status changes to completed.

How do you run multiple pipelines at the same time?

Multiple data pipelines can run concurrently in Ocient. To run more than one simultaneously, any user with privileges to start a pipeline can use the START PIPELINE SQL statement. Execution of this statement begins the execution of the selected pipeline. The Ocient System uses Loader Nodes in parallel to process all of the pipelines that are running.

If you need to control the relative priority of one pipeline over another when running concurrently, use advanced pipeline settings to limit the number of cores allocated to a specified pipeline or isolate the pipeline to a particular set of Loader Nodes.

What happens if I start a pipeline that has been stopped?

When you execute a `START PIPELINE SQL` statement on a pipeline that has been stopped due to either hitting an `ERROR LIMIT` or due to user intervention, the load resumes from the last point in the pipeline. Hence, you can effectively pause a pipeline and resume from the last checkpoint in the pipeline. This feature works for continuous pipelines such as Apache® Kafka® pipelines as well as batch pipelines.

Common uses of stopping and resuming a pipeline are:

- Stopping to update the transformations in a continuous pipeline.
- Stopping to update the table schema and refresh a pipeline.
- Stopping to allow a higher priority workload to proceed.
- Stopping for maintenance.

It is important to exercise caution when resuming a pipeline to ensure that the files in the underlying data source have not changed. Modifying data in the source for an in-process pipeline can result in unexpected results such as duplicate data.

File Loading**When is the file list determined for a batch file load?**

When you load a batch of files in a data pipeline, the Ociant System determines the batch when the pipeline starts using the bucket, the prefix, and the filter parameters you specify in your pipeline. This batch of files is frozen until the pipeline completes. Even if the batch pipeline stops and restarts, the Ociant System keeps the list of files constant.

Can I update the file list after starting a pipeline?

You cannot manually modify the file list for a batch pipeline. When a pipeline restarts, it finishes the files that were determined when the batch was first established.

To add files that arrive on the file system after a batch pipeline was first started, you can create a new batch pipeline that uses filters to load all files created after a specified timestamp.

What is the behavior if I start a pipeline that has been stopped?

When you start a file-based pipeline that has been stopped, the pipeline adds any new files to the pipeline as files with the `PENDING` status. The pipeline then loads these files as execution proceeds. The Ociant System does not reload any files that the pipeline has already loaded.

It is important to exercise caution when resuming a pipeline to ensure that the files in the underlying data source have not changed. When you modify data in the source for an in-process pipeline, the Ociant System can have unexpected results, such as duplicate data.

What options exist to maximize performance on S3-type file loads?

The most common performance enhancement for Amazon® Web ServicesSM (AWSSM) S3 file loads is the `PREFIX` parameter in your pipeline. This setting restricts the files in the bucket that your pipeline considers, speeding up listing operations.

You can also use other advanced options such as `MAX_CONCURRENCY`, `REQUEST_DEPTH`, and `REQUEST_RETRIES` described in [Data Pipelines](#).

Kafka Loading

How does a continuous Kafka pipeline work?

Continuous pipelines for Kafka stream records from Kafka partitions and ensure exactly-once semantics are delivered. Ocient assigns consumers to different Loader Nodes and commits offsets back on the assigned partitions as the source data loads into Ocient. This operation critically only occurs after data is durable in Ocient, which means that the system replicates data in such a way that the loss of a configurable number of nodes would not lead to data loss.

All supported data formats are compatible with Kafka pipelines.

When you run a continuous Kafka pipeline, the pipeline is never complete so it remains in a running state. You can observe progress using the standard system catalog tables and the Kafka-specific table `sys.pipeline_partitions` that captures individual partition assignments and lag.

When you stop and restart a Kafka pipeline, it resumes from the last commit checkpoint and continues processing records. The Ocient System deduplicates any record that has already been sent to Ocient but not committed to Kafka in the Loader Nodes.

How do I update the transforms or the table schema in a continuous Kafka pipeline?

To update transforms in a Kafka pipeline, first, stop the pipeline. After the pipeline stops, you can execute a `CREATE OR REPLACE PIPELINE` SQL statement. This action allows you to update the transformations in the pipeline and maintains all of the checkpoint information about progress on the pipeline. When you execute the `START PIPELINE` SQL statement, the pipeline resumes from the last Kafka committed offsets. Then, the Loader Nodes deduplicate any record that has already been sent to the Loader Nodes but is not yet committed to Kafka.

Where does a Kafka pipeline resume loading after a stop and then start of the pipeline?

When you execute the `START PIPELINE` SQL statement, the pipeline resumes from the last Kafka committed offsets. The Loader Nodes deduplicate any record that has already been sent to the Loader Nodes but is not yet committed to Kafka.

If you have never started a Kafka pipeline, the pipeline starts from the configured `auto.offset.reset` value. You can set this value in the `CONFIG` option of a Kafka pipeline. The Ocient System uses this value when a pipeline first starts to establish the consumer group for the pipeline.

For more details about Kafka configuration, see [Data Pipelines](#).

Data Duplication and Deduplication**What records are deduplicated during loading?**

Ocient Loader Nodes deduplicate data that is loaded on the same pipeline. You can think of the deduplication scope of a pipeline as being tied to the life of the pipeline database object.

If you drop a pipeline and create a new one with the same name, this data is *not* deduplicated.

For a specified pipeline, the Ocient System deduplicates data automatically when you stop and resume a pipeline. The system ensures that no duplicate data is loaded for any reprocessed data due to managing watermarks with data sources such as Kafka or S3.

It is important to note that deduplication in Ocient is based on a shared contract between the data source and the Ocient System. Deduplication is not based on a primary key or record identifier that is part of the data itself. The [Frequently Asked Questions for Data Pipelines](#) question explains the implications more clearly.

How can I load the same data set twice into the same table?

Sometimes, it is useful to load data multiple times for testing, but deduplication can get in the way.

To load the same data set twice in the same table, there are two simple approaches:

1. Drop the pipeline from the first load, create the same pipeline, and run it to completion.

2. Create a new pipeline with the same SQL statement as the original, but with a new name, and run it to completion.

In both cases, the Ocient System does not deduplicate the data and you can load a second copy of your data.

What mistakes can accidentally lead to duplicating data?

The Ocient System can have duplicate data when the deduplication contract between the source and the pipeline is not maintained.

Ocient uses a proprietary approach to deduplication that helps it deliver high throughput when loading while still ensuring exactly-once semantics. If the data in the source changes during the execution of a pipeline, the changes can lead to a mismatch in the way the Ocient System determines the deduplication identifier for a record.

Things to avoid include:

- Modifying a file in the source file system after it has been included in a pipeline, but the pipeline has not been completed.
- Deleting a file in the source that has been included in a pipeline, but the pipeline has not been completed.

You can add new files safely to a pipeline before it completes, but you should avoid modifying files already registered by the pipeline and appear in the `sys.pipeline_files` system catalog table.

Troubleshooting and Errors

How do I detect when an entire file is failed in a pipeline?

If an entire file has failed, you can see this in the `sys.pipeline_files` system catalog table. Any file with the `FAILED` status indicates that the file failed to process. This failure can occur due to a file-level error such as corruption.

In addition, if a file has the `SKIPPED` status, this status indicates that the file was skipped when processing the pipeline. You can control this behavior using the `FILE_ERROR` setting in the `START PIPELINE` SQL statement.

If a specified file experiences record-level errors, but the Ocient System processes the file, the file has the `LOADED_WITH_ERRORS` status.

How do I find the individual records that failed to load?

You can find the individual errors in the `sys.pipeline_errors` system catalog table.

You can determine if a pipeline encounters errors by viewing the `information_schema.pipeline_status` view that includes counts of records with errors.

Transformations

How can I convert a string representation of an array of data into an array data type?

The Ocient System automatically converts a string representation of an array into an array using the settings defined on your pipeline. The system uses the `CLOSE_ARRAY`, `OPEN_ARRAY`, and `ARRAY_ELEMENT_DELIMITER` settings to convert a string to a target array column type. These types default to a PostgreSQL®-style array (e.g., `{1,2,3}`).

For details, see the supported options in [Data Pipelines](#).

How do I apply a transformation function to an array of data?

Data pipelines do not yet support the mapping of arbitrary transformation functions over an array. However, the Ocient System supports casting and type conversion. You can apply an array cast to an array and the system applies the type of each element as a conversion to the elements of the array.

For details, see [Transform Data in Data Pipelines](#).

My data is in a non UTF-8 encoding. How do I get the load to decode my character set properly?

You can change the character set of your pipeline using the `CHARSET_NAME` option in a pipeline. For details, see the supported options in [Data Pipelines](#).

Can a data pipeline extract nested data in JSON?

Ocient supports a JSON selector syntax similar to JSON Path. The Ocient System can extract individual keys as well as nested data. A simple example is `$order.user.first_name`, which extracts the first name `first_name` from the `user` object on the `order` object in JSON.

For details, see [Data Formats for Data Pipelines](#).

Can a data pipeline extract attributes of objects in an array in JSON?

Ocient supports a JSON selector syntax similar to JSON Path. The Ocient System can extract individual keys, nested objects, arrays, and arrays of objects. The system can also project a selector into the arrays. A simple example is `$data.line_items[].price`, which extracts a one-dimensional array of prices `price` from the `line_items` array inside the `data` object.

For details, see [Data Formats for Data Pipelines](#).

How do I apply a user-defined transformation to the data?

Data Pipelines support the creation of user-defined functions that you can use to apply more advanced transformations. For details, see the [Data Pipelines SQL](#) statement.

What automatic conversions does Ocient perform from my source data to the target columns?

The Ocient System supports many automatic conversions to make it easy to load source data into your target tables. `VARCHAR` data automatically casts into most target column types. The system casts many data types automatically to a `VARCHAR` target column. Some other common automatic conversions, such as `BIGINT` to `TIMESTAMP`, are supported, especially where the conversion is lossless.

For details, see [Data Types for Data Pipelines](#).

How do I cast data to an array column?

To cast to an array column, use a casting function. This function converts a string of data in the configured array format to the array type you specify, casting the elements to the inner type. For example, if `$my_column` represents the data `'{123,456,789}'` and your pipeline is using the default `CLOSE_ARRAY`, `OPEN_ARRAY`, and `ARRAY_ELEMENT_DELIMITER` settings, this expression snippet casts the elements to an array of `INT` values. Ocient also automatically casts `$my_column` in this example to an `INT[]` when that is the type of the target column.

```
CREATE PIPELINE
...
SELECT
  INT[]($my_column) as column_array_of_ints;
```

How do I cast data to a tuple column?

Similar to casting to an array column, you can cast to a tuple column using tuple casting syntax. For example, if `$my_column` represents the data `'(123,test)'`, and your pipeline is using the default `CLOSE_TUPLE`, `OPEN_TUPLE`, and `TUPLE_ELEMENT_DELIMITER` settings, this expression snippet casts the elements as a tuple of an integer and a string `TUPLE<<int,varchar>>`.

```
CREATE PIPELINE
...
SELECT
  TUPLE<<int,varchar>>($my_column) as column_tuple;
```

Related Links

[Load Data](#)

[Data Pipelines](#)

4 Query Ociant

The Ociant Hyperscale Data Warehouse™ contains one or more relational databases based on SQL. Ociant® follows the PostgreSQL® dialect closely and most functions operate the same in Ociant as in PostgreSQL.

- [Quick Start Querying](#)
- [General SQL Syntax](#)
 - [SQL Syntax Conventions](#)
 - [Identifiers](#)
 - [Generate Tables Using sys.dummy](#)
- [Understanding Data Types](#)
 - [Array, Tuple, and Matrix Overview](#)
- [Functions Overview](#)
- [Text Analysis in Database Queries](#)

4.1 Quick Start Querying

This example demonstrates a quick start to begin querying an Ociant® System.

Prerequisites

To get started querying in Ociant, the following are required:

1. A valid username and password for your Ociant System — These credentials can be obtained from your Database Administrator or System Administrator.
2. Network access to the SQL Nodes and the IP Address of the Hostname of these nodes in the Ociant System.
3. The port used for SQL Nodes. By default, this is 4050.
4. A SQL Client or SQL Driver — This example uses the Ociant JDBC Driver. See [Connect Using JDBC](#).

Run a Query

With these prerequisites, you can construct a connection string for your SQL Client of choice, connect, and begin querying Ociant in a few minutes. This example assumes the use of the JDBC driver and CLI. If using a different connection method, see [Connect to Ociant](#) to complete Step 2.

Step 1: Create a Connection String

A connection string follows a standard format for Ociant:

```
jdbc:ocient://<HOSTNAME_OR_IP_ADDRESS>:<PORT>/<DATABASE>
```

In this example, you are going to replace each of the placeholders with the following:

- PORT: 4050
- HOSTNAME_OR_IP_ADDRESS: 10.10.1.1
- DATABASE: system

Some systems use a hostname like `sql1.example.com` instead of an IP Address. Additionally, if you have a database configured already, use that instead of `system` to access your tables.

The final connection string is:

```
jdbc:ocient://10.10.1.1:4050/system
```

Step 2: Connect

Start the JDBC CLI according to the instructions in [Connect Using JDBC](#). You need to specify a username and password.

The Fully Qualified Username is specified as `<username>@<database>`, so in your case, for a username of `example_user`, enter `example_user@system`.

Next, you should see an Ocient CLI prompt:

```
Ocient> _
```

Enter the connect statement using your connection string:

```
connect to jdbc:ociant://10.10.1.1:4050/system;
```

You receive a response similar to `Connected to jdbc:ociant://10.10.1.1:4050/system` indicating that you are now connected.

Step 3: Run a Query

You can now query any user tables or system catalog tables to which you have access in the database. System Catalog tables provide a wealth of information about the underlying system and configuration. Because you are connected to the system database, you can begin with a few catalog queries.

At the Ocient prompt, enter this SQL statement that displays information for all tables in the system.

```
Ocient> SELECT * FROM information_schema.tables;
```

To filter this information, use the LIKE predicate.

```
Ocient> SELECT * FROM information_schema.tables WHERE table_name LIKE '%data%';
```

Now that you are connected, you can run any SQL queries or perform Database Administrator operations like [Data Definition Language \(DDL\) Statement Reference](#) or [Data Control Language \(DCL\) Statement Reference](#) where you have permission.

Related Links

[General SQL Syntax](#)

[SQL Reference](#)

4.2 General SQL Syntax

The Ocient Hyperscale Data Warehouse™ supports querying using the SQL syntax that follows ANSI SQL standards.

Ocient SQL Syntax

This code block shows the general syntax to perform SQL querying in Ocient® and the order in which commands should go.

For specific descriptions and syntax for the commands, see the respective SQL statement sections on this page.

Syntax

```
[ WITH ... ]
SELECT ...
  [ EXCEPT(...) ]
  [ FROM ...
    [ JOIN ... ] ]

  [ WHERE ... ]
  [ GROUP BY ...
    [ HAVING ... ] ]
  [ ORDER BY ... ]
  [ LIMIT ... ]
  [ OFFSET ... ]
  [ INTERSECT ... ]
  [ EXCEPT ... ]
```

```
[ UNION ... ] ]
[ USING ... ]
[ TRACE ... ]
```

Default Schema

Ocient identifies every table using a database and schema. For example, the fully qualified path to the `movies` table is `cinema.adventure.movies`, where `cinema` is the database and `adventure` is the schema.

When you do not fully qualify a table name, the Ocient System uses a default schema. When you first log into the system, the default schema is your fully qualified username. You can change the default schema using the [Commands Supported by the Ocient JDBC CLI Program](#) command.

Querying Command Reference

Ocient supports the following SQL statements.

WITH

Assigns a name to a common table expression, allowing an auxiliary query to be used in the main query. This is helpful for breaking down complex queries into smaller parts.

Syntax

```
WITH
  <cte_name> [ ( <cte_column> [ ,... ] ) ]
  AS ( <sub_query> )
<select_query>
```

Parameters

Parameter	Description
<cte_name>	A name for the common table expression populated with the result set from the <code>sub_query</code> statement.
<cte_column>	Optional. This is a list of column names for common table expressions populated with the <code>sub_query</code> result set. If no column names are provided, the column names are the same as the table referenced in the <code>sub_query</code> statement.
<sub_query>	An auxiliary query that collects data for a common table expression. <code>sub_query</code> can use regular query commands, including ORDER BY, LIMIT, OFFSET, UNION, INTERSECT, EXCEPT, WHERE, GROUP BY, and HAVING. See the respective SQL statement reference sections for usage rules.
<select_query>	The main SELECT query. Note that the <code>FROM</code> statement in the main SELECT query must also include the <code>cte_name</code> if the query uses its values.

Example

In this example, the subquery calculates the average budget for all rows in the `movies` table. The main query uses that average to find all movies that spent more.

```

WITH avg_budget_table (average_budget) AS (
    SELECT AVG(budget)
    FROM movies
)
SELECT title,
       budget,
       revenue
FROM movies,
     avg_budget_table
WHERE budget > avg_budget_table.average_budget;

```

Output

title	budget	revenue
CGI Why	237000000	2787965087
Titania	200000000	1845034188
Merchandise Vehicle 5	200000000	1066969703
The Tentpole	220000000	1519557910
Pirates of Palm Springs	140000000	655011224
Spyman 16	200000000	1108561013
Frigid	150000000	1274219009
Fury 7	190000000	1506249360
Superhero 23	250000000	1084939099
Triassic World	150000000	1513528810
Iron Chef 3	200000000	1215439994
The Last Airman	150000000	318502923

SELECT

Initiates a query statement or a subquery clause within other statements.

For information on using SELECT as a subquery for filtering or ordering results, see the [General SQL Syntax \(cloned\)](#) and [General SQL Syntax \(cloned\)](#) sections.

For information on using SELECT as a subquery for a common table expression, see the [General SQL Syntax](#) section.

Syntax

```

SELECT [ ALL | DISTINCT ]
      [ * [ EXCEPT ( column_name [ , ... ] ) ]
      | <select_list_entry> [ , ... ] ]
      [ <from_clause> ]

```

Parameters

Parameter	Description
ALL	SELECT and SELECT ALL are both the same. Returns all valid data rows from the database that meets the criteria of your query.
DISTINCT	SELECT DISTINCT returns only unique rows that do not match other rows based on the criteria of your query.
*	Returns all columns from the specified tables in the result set for the query.
EXCEPT	When used with *, EXCEPT allows specific columns to be excluded from the query results.

Parameter	Description
column_name	The name of one or more columns from the specified table that you want to exclude (using EXCEPT) from your query results.

<select_list_entry>

The **select_list_entry** defines a column or expression to include in your query result set.

Syntax

```
<select_list_entry> ::=
    column_name | expression [ AS new_name ]
```

Parameters

Parameter	Description
column_name	The name of one or more columns from the specified table that you want to include in your query results.
expression	One or more expressions that you want to include in your query results. Expressions can be any combination of literal values, column names, arithmetic expressions, parentheses, and function calls.
new_name	An alias, treated as an identifier, for an alternative name for the column or the results of the expression.

<from_clause>

For information on the **<from_clause>**, see the [General SQL Syntax](#) documentation.

Examples

Using **SELECT ***

This example uses **SELECT *** to return all the columns in the **movies** table.

```
SELECT * FROM movies LIMIT 5;
```

Output

movie_id	title	budget	popularity	release_date	revenue	runtime	movie_status	vote_average	vote_count
211672	Minnows	74000000	875.581305	2015-06-17	1156730962	91	Released	6.40	4571
24	Billy the Killy	30000000	79.754966	2003-10-10	180949000	111	Released	7.70	4949
19995	CGI Why	237000000	150.437577	2009-12-10	2787965087	162	Released	7.20	11800
37724	Spyman 16	200000000	93.004993	2012-10-25	1108561013	143	Released	6.90	7604
24428	The Tentpole	220000000	144.448633	2012-04-25	1519557910	143	Released	7.40	11776

Using **SELECT * EXCEPT**

This example uses **SELECT * EXCEPT** to exclude certain columns from the result set.


```
SELECT *
EXCEPT (movie_id, runtime, vote_average, vote_count)
FROM movies
LIMIT 5;
```

Output

title	budget	release_date	revenue
Swords & Scabbards	94000000	2003-12-01	1118888979
Billy the Killy	30000000	2003-10-10	180949000
Merchandise Vehicle 5	200000000	2010-06-16	1066969703
CGI Why	237000000	2009-12-10	2787965087
Space Odyssey 6000	10500000	1968-04-10	68700000

Using Lateral Column Aliases

Ocient SQL queries support lateral column aliases, meaning you can immediately reuse aliases for calculations in the same query as new inputs. Hence, you can simplify queries that normally require subqueries and common table expressions.

These examples use the `products` table with these columns:

- `product_id` — Product identifier as an integer
- `product_name` — Product name as a string
- `price` — Price as a floating point number

Create this table using the `CREATE TABLE` SQL statement.

```
CREATE TABLE products (
  product_id INT,
  product_name VARCHAR(100),
  price DECIMAL(10, 2)
);
```

Insert four records into the `products` table.

```
INSERT INTO products (product_id, product_name, price) VALUES
(1, 'Laptop', 1000.00),
(2, 'Tablet', 500.00),
(3, 'Smartphone', 800.00),
(4, 'Monitor', 300.00);
```

Create a query to determine prices after discounts and taxes by using a common table expression subquery `DiscountedPrices`. Use the `WITH` keyword to create the subquery.

```
WITH DiscountedPrices AS (
  SELECT
    product_id,
    product_name,
    price,
    price * 0.90 AS discounted_price,
    price * 0.90 * 1.05 AS total_price_after_tax
  FROM
    products
)
SELECT
  product_id,
  product_name,
```

```

    price,
    discounted_price,
    total_price_after_tax
FROM
    DiscountedPrices;

```

Lateral aliases allow the same calculations to be packaged in a single query.

This simpler query is essentially the same as the longer common table expression example, but the logic is condensed because you can reference the `discounted_price` alias immediately to calculate the `total_price_after_tax` value in the same query.

```

SELECT
    product_id,
    product_name,
    price,
    price * 0.90 AS discounted_price,
    discounted_price * 1.05 AS total_price_after_tax
FROM
    products;

```

FROM

Specifies the table or view to use in a `SELECT` statement.

Syntax

```

<select_clause> FROM
{ table_name | ( <sub_query> ) } [ ,... ]
[ <join_clause> [ ,... ] ]

```

Parameters

Parameter	Description
<select_clause>	A query statement using the <code>SELECT</code> SQL statement. For details, see the General SQL Syntax section.
table_name	The names of one or more tables or views that you want to query. If you specify multiple tables in the <code>FROM</code> list, separated by commas, the effect is the same as if you explicitly perform a <code>General SQL Syntax</code> on all of those sources.
<sub_query>	One or more subqueries using the <code>SELECT</code> SQL statement. A subquery generates a table from which the <code>select_clause</code> query references data. Each subquery must be enclosed in parentheses with an optional correlation name. For details, see the General SQL Syntax section.
<join_clause>	A <code>JOIN</code> clause used to retrieve data from two or more tables for your query. For details, see the General SQL Syntax (cloned) section.

JOIN

Combines rows from multiple tables so they can be accessed by a query.

Syntax

```

<select_query>
  FROM | **Parameter** | **Description** |
  | ---/---/
  | `<select_query>` | A valid`SELECT`query.For details, seeGeneral SQL Syntax (cloned). |
  | `<table_reference1>` | Either a table name or a full`SELECT`in parentheses`table_reference1`takes preced
  | `<table_reference2>` | Either a table name or a full`SELECT`in parentheses function.`table_reference2`ta
  | `table1_column` | A column from either`table_reference`The`JOIN`operation uses this column to match rows
  | `<boolean_operator>` | Join conditions can use any Boolean expression, including`=`,`!=`,`<`,`>`,`=>`,`a
  | `table2_column` | A column from the`table_reference`not used for`table1_column`The`JOIN`operation uses t

##### Types of Join Operations ( `<join_operation>` ) {.unnumbered}

```

Ocient supports the following types of `JOIN` operations.

****Syntax****

```

```sql
<join_operation> ::=
{ INNER JOIN
| LEFT [OUTER] JOIN
| RIGHT [OUTER] JOIN
| FULL [OUTER] JOIN
| CROSS JOIN
| SEMI JOIN
| ANTI JOIN }

```

## JOIN Type Descriptions

| JOIN Type        | Description                                                                                                                                                                                                                                      |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| INNER JOIN       | Returns all rows that exist in both tables. This join type is the default type of join.                                                                                                                                                          |
| LEFT OUTER JOIN  | Returns all rows from the first table regardless of whether there are matching rows in the second table.LEFT JOINperforms the same operation asLEFT OUTER JOIN.                                                                                  |
| RIGHT OUTER JOIN | Returns all rows from the second table regardless of whether there are matching rows in the first table.RIGHT JOINperforms the same operation asRIGHT OUTER JOIN.                                                                                |
| FULL OUTER JOIN  | Returns all matched and unmatched rows.FULL JOINperforms the same operation asFULL OUTER JOIN.                                                                                                                                                   |
| CROSS JOIN       | Returns the Cartesian product of the joined tables. Each row in the first table is joined with all rows in the second table. The result set contains the number of rows in the first table multiplied by the number of rows in the second table. |
| SEMI JOIN        | Returns the rows from the first table that have at least one match in the second table. No columns from the second table are available in the query.                                                                                             |
| ANTI JOIN        | Returns the rows from the first table that have no matches in the second table. No columns from the second table are available in the query.                                                                                                     |

By default, **JOIN** statements that involve subqueries can operate laterally if necessary. This behavior allows subqueries to reference joined columns from the preceding items included in the **FROM** clause. For example, both parts of this join operation use subqueries that reference table **x**. The **LATERAL** keyword is optional; the join operation acts the same regardless of whether it is included.

```
SELECT * FROM (SELECT 1 AS cx1, 2 AS cx2) AS x INNER JOIN LATERAL (SELECT x.cx1 AS
cy1, x.cx2 AS cy2) AS y ON x.cx1 = y.cy1;
```

Lateral joins are primarily useful when a cross-referenced column is necessary for computing the rows to join. A common application is providing an argument value for a set-returning function.

## Examples

These examples join two tables:

- **games** — A table of video game titles and their genre\_ids. Note that some games have a **NULL** value assigned to their genre\_id.

| game_name              | genre_id |
|------------------------|----------|
| Star Battle 6000       | 9        |
| Dwarf Simulator        | NULL     |
| 2002 Futbol            | 11       |
| Space Attack!          | 9        |
| Fantasy Fool           | 8        |
| Blasto the Squirrel    | 5        |
| Trucker Quest          | 7        |
| Block, Stack, and Cry! | 6        |
| Basketball Day         | 11       |
| Pizza Mutant           | 1        |
| Italian Plumber        | 5        |
| Zombies!               | 1        |
| Space Ninjas           | 12       |
| Moon Mayor             | 10       |
| Spreadsheet Hero       | NULL     |
| Zoom Tycoon            | 12       |
| Porch Poacher          | 6        |

- **genre** — A table of video game genres, which are identified by IDs.

| genre_name   | id |
|--------------|----|
| Racing       | 7  |
| Puzzle       | 6  |
| Adventure    | 2  |
| Simulation   | 10 |
| Shooter      | 9  |
| Misc         | 4  |
| Action       | 1  |
| Sports       | 11 |
| Role-Playing | 8  |
| Platform     | 5  |
| Strategy     | 12 |
| Fighting     | 3  |

## INNER JOIN

This example uses an `INNER JOIN` operation to capture only the rows that exist in both tables. Games with `NULL` values for their `genre_id` are eliminated from the result set.

```
SELECT game_name,
 genre_name
FROM video_games.game
 INNER JOIN video_games.genre ON game.genre_id = genre.id;
```

*Output*

| game_name              | genre_name   |
|------------------------|--------------|
| Fantasy Fool           | Role-Playing |
| Blasto the Squirrel    | Platform     |
| Italian Plumber        | Platform     |
| Porch Poacher          | Puzzle       |
| Space Ninjas           | Strategy     |
| Basketball Day         | Sports       |
| 2002 Futbol            | Sports       |
| Moon Mayor             | Simulation   |
| Block, Stack, and Cry! | Puzzle       |
| Zombies!               | Action       |
| Pizza Mutant           | Action       |
| Space Attack!          | Shooter      |
| Trucker Quest          | Racing       |
| Zoom Tycoon            | Strategy     |

## LEFT OUTER JOIN

This example uses a `LEFT OUTER JOIN` operation to capture all rows from the left table (game), even if they have no matching row in the right table (genre).

```
SELECT game_name,
 genre_name
FROM video_games.game
 LEFT OUTER JOIN video_games.genre ON game.genre_id = genre.id;
```

*Output*

| game_name              | genre_name   |
|------------------------|--------------|
| Fantasy Fool           | Role-Playing |
| Space Ninjas           | Strategy     |
| Zoom Tycoon            | Strategy     |
| Zombies!               | Action       |
| Pizza Mutant           | Action       |
| Trucker Quest          | Racing       |
| Blasto the Squirrel    | Platform     |
| 2002 Futbol            | Sports       |
| Basketball Day         | Sports       |
| Italian Plumber        | Platform     |
| Block, Stack, and Cry! | Puzzle       |
| Star Battle 6000       | Shooter      |
| Dwarf Simulator        | NULL         |
| Moon Mayor             | Simulation   |
| Porch Poacher          | Puzzle       |
| Spreadsheet Hero       | NULL         |

| game_name     | genre_name |
|---------------|------------|
| Space Attack! | Shooter    |

## RIGHT OUTER JOIN

This example uses a `RIGHT OUTER JOIN` operation to capture all rows from the right table (genre), even if they have no matching row in the left table (game).

```
SELECT game_name,
 genre_name
FROM video_games.game
 RIGHT OUTER JOIN video_games.genre ON game.genre_id = genre.id;
```

*Output*

| game_name              | genre_name   |
|------------------------|--------------|
| NULL                   | Adventure    |
| Trucker Quest          | Racing       |
| Pizza Mutant           | Action       |
| Moon Mayor             | Simulation   |
| Fantasy Fool           | Role-Playing |
| Space Attack!          | Shooter      |
| NULL                   | Misc         |
| Star Battle 6000       | Shooter      |
| Space Ninjas           | Strategy     |
| Porch Poacher          | Puzzle       |
| Block, Stack, and Cry! | Puzzle       |
| Blasto the Squirrel    | Platform     |
| Italian Plumber        | Platform     |
| Zoom Tycoon            | Strategy     |
| NULL                   | Fighting     |
| Zombies!               | Action       |
| 2002 Futbol            | Sports       |
| Basketball Day         | Sports       |

## FULL OUTER JOIN

This example uses a `FULL OUTER JOIN` operation to capture all rows from both tables, even if they do not match.

```
SELECT game_name,
 genre_name
FROM video_games.game
 FULL OUTER JOIN video_games.genre ON game.genre_id = genre.id;
```

*Output*

| game_name           | genre_name |
|---------------------|------------|
| Basketball Day      | Sports     |
| 2002 Futbol         | Sports     |
| Zombies!            | Action     |
| Blasto the Squirrel | Platform   |
| Moon Mayor          | Simulation |
| Trucker Quest       | Racing     |
| NULL                | Misc       |

| game_name              | genre_name   |
|------------------------|--------------|
| Star Battle 6000       | Shooter      |
| Space Attack!          | Shooter      |
| Spreadsheet Hero       | NULL         |
| NULL                   | Adventure    |
| NULL                   | Fighting     |
| Italian Plumber        | Platform     |
| Pizza Mutant           | Action       |
| Dwarf Simulator        | NULL         |
| Fantasy Fool           | Role-Playing |
| Block, Stack, and Cry! | Puzzle       |
| Space Ninjas           | Strategy     |
| Zoom Tycoon            | Strategy     |
| Porch Poacher          | Puzzle       |

## CROSS JOIN

This example uses a **CROSS JOIN** operation to capture every possible combination of rows from both tables, regardless of whether they match. Note that unlike other **JOIN** operations, **CROSS JOIN** does not require an **ON** statement.

```
SELECT game_name,
 genre_name
FROM video_games.genre
 CROSS JOIN video_games.game;
```

*Output*

As the result set for this **CROSS JOIN** example is more than 200 rows, the results are abbreviated.

| genre_name       | genre_id     |
|------------------|--------------|
| Spreadsheet Hero | Role-Playing |
| Spreadsheet Hero | Misc         |
| Spreadsheet Hero | Sports       |
| Spreadsheet Hero | Platform     |
| Spreadsheet Hero | Strategy     |
| Spreadsheet Hero | Shooter      |
| Spreadsheet Hero | Fighting     |
| Spreadsheet Hero | Action       |
| Spreadsheet Hero | Simulation   |
| Spreadsheet Hero | Racing       |
| Spreadsheet Hero | Puzzle       |
| Spreadsheet Hero | Adventure    |
| Space Attack!    | Role-Playing |
| Space Attack!    | Misc         |
| Space Attack!    | Sports       |
| Space Attack!    | Platform     |
| Space Attack!    | Strategy     |
| Space Attack!    | Shooter      |
| Space Attack!    | Fighting     |
| Space Attack!    | Action       |
| Space Attack!    | Simulation   |
| Space Attack!    | Racing       |
| Space Attack!    | Puzzle       |

| genre_name    | genre_id  |
|---------------|-----------|
| Space Attack! | Adventure |
| ...           | ...       |

## SEMI JOIN

This example uses a **SEMI JOIN** operation to capture genre names that have at least one match in the games table. Rows are only included once, even if there are multiple matches.

```
SELECT genre_name,
 id
FROM video_games.genre
SEMI JOIN video_games.game ON genre.id = game.genre_id;
```

*Output*

| genre_name   | genre_id |
|--------------|----------|
| Simulation   | 10       |
| Shooter      | 9        |
| Racing       | 7        |
| Strategy     | 12       |
| Puzzle       | 6        |
| Platform     | 5        |
| Action       | 1        |
| Sports       | 11       |
| Role-Playing | 8        |

## ANTI JOIN

This example uses an **ANTI JOIN** operation to capture any rows in the game table that do not match any rows in the genre table.

```
SELECT game_name
FROM video_games.game
ANTI JOIN video_games.genre ON game.genre_id = genre.id;
```

*Output*

| game_name        |
|------------------|
| Spreadsheet Hero |
| Dwarf Simulator  |

## WHERE

Filters rows based on a specified condition.

### Syntax

```
<select_query>
WHERE { column_name <filter_condition> filter_value |
[NOT] EXISTS (<sub_query>) }
```

### Parameters



| Parameter                         | Description                                                                                                                      |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| <code>&lt;select_query&gt;</code> | A valid <code>SELECT</code> query.                                                                                               |
| <code>column_name</code>          | A column used for grouping that is evaluated by the <code>filter_condition</code> .                                              |
| <code>filter_value</code>         | A value used to evaluate the specified <code>column_name</code> . For details, see the General SQL Syntax section.               |
| <code>&lt;sub_query&gt;</code>    | A subquery with a filter condition that follows the <code>EXISTS</code> clause. For details, see the General SQL Syntax section. |

### `<filter_condition>`

A logical combination of predicates used to evaluate the referenced `column_name` based on the `filter_value`.

#### Syntax

```

<filter_condition> ::=
{ =
 | ==
 | <>
 | !=
 | [NOT] EQUALS
 | <
 | <=
 | >
 | >=
 | [NOT] IN
 | [NOT] LIKE
 | [NOT] SIMILAR TO
 | BETWEEN
 | FOR SOME
 | FOR ALL
 | IS [NOT] NULL
 | IS [NOT] DISTINCT FROM
}

```

#### Definitions

| Operator                 | Description           | Example                                  |
|--------------------------|-----------------------|------------------------------------------|
| <code>=,==,EQUALS</code> | Equal to.             | <code>Author = 'Alcott'</code>           |
| <code>&lt;&gt;,!=</code> | Not equal to          | <code>Dept &lt;&gt; 'Sales'</code>       |
| <code>&gt;</code>        | Greater than          | <code>Hire_Date &gt; '2012-01-31'</code> |
| <code>&lt;</code>        | Less than             | <code>Bonus &lt; 50000.00</code>         |
| <code>&gt;=</code>       | Greater than or equal | <code>Dependents &gt;= 2</code>          |
| <code>&lt;=</code>       | Less than or equal    | <code>Rate &lt;= 0.05</code>             |

| Operator                  | Description                                                                                                                                                                                                                                                                                                                                                               | Example                                                           |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|
| [ NOT ] LIKE              | Begins with a character pattern to match. The pattern is enclosed in parentheses and can include the wildcard characters %, representing zero, one or multiple characters, and _ representing a single character. To have a literal % or _ that does not act as a wildcard on the right-hand side of a LIKE or NOT LIKE expression, put a backslash before the character. | Full_Name LIKE 'Will%'                                            |
| [ NOT ] SIMILAR TO        | For usage information, see the General SQL Syntax section.                                                                                                                                                                                                                                                                                                                | 'abc' SIMILAR TO 'abc'                                            |
| BETWEEN value1 AND value2 | Requires two values. The filter evaluates to TRUE if the specified column is within the range between the two values. BETWEEN bounds are inclusive. Values can be times, numbers, or strings.                                                                                                                                                                             | Sales BETWEEN 1000 AND 5000                                       |
| FOR_SOME()                | Evaluates an array to determine if at least one value meets the filter criteria. In the example, the FOR_SOME() operator would return all rows of the column col_int_array where at least one value is greater than 10. For details, see General SQL Syntax.                                                                                                              | FOR_SOME(col_int_array) > 10<br>10 ``FOR_SOME(col_int_array) > 10 |
| FOR_ALL()                 | Evaluates all values of an array to determine if all meet the filter criteria. In the example, the FOR_ALL() operator would return any rows of the column col_int_array where all values are greater than 10. For details, see General SQL Syntax.                                                                                                                        | FOR_ALL(col_int_array) > 10<br>10 ``FOR_ALL(col_int_array) = 1    |
| [ NOT ] IN                | Equal to one of multiple possible values                                                                                                                                                                                                                                                                                                                                  | DeptCode IN (101, 103, 209)                                       |
| IS [ NOT ] NULL           | Compare to NULL (missing data)                                                                                                                                                                                                                                                                                                                                            | Address IS NOT NULL                                               |

| Operator                 | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Example                               |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|
| IS [ NOT ] DISTINCT FROM | Compares the equality of two expressions for a Boolean result, including comparisons that involve <code>NULL</code> values. This operator ensures reliable comparisons in scenarios where <code>NULL</code> values need to be treated as significant data points. Normally, <code>NULL = NULL</code> evaluates to <code>false</code> because <code>NULL</code> represents an unknown value. However, the statement <code>a IS NOT DISTINCT FROM b</code> treats <code>NULL</code> as a comparable value and returns <code>true</code> when both values are <code>NULL</code> or when <code>a = b</code> . Inversely, <code>a IS DISTINCT FROM b</code> returns <code>true</code> if the values are different or if one value is <code>NULL</code> while the other is not <code>NULL</code> . | Debt IS NOT DISTINCT FROM Receivables |

## EXISTS

An `EXISTS` clause used in a `WHERE` statement evaluates if a subquery returns any rows.

For each row that the database computes in the outer query, if the subquery returns at least one row, the `EXISTS` clause evaluates to `true`. In this outcome, the outer query returns its row.

If the subquery returns zero rows, the `EXISTS` clause evaluates to `false`, and the outer query excludes those rows.

The `NOT EXISTS` clause performs the opposite Boolean logic. In this case, the outer query returns rows only when the subquery has no matches.

### Examples

These examples use two tables, one for company departments and the other for employees who are assigned to those departments.

Create the `departments` table for department data.

```
CREATE TABLE departments (
 id INT,
 name VARCHAR(50)
);
```

Create the `employees` table for employee data.

```
CREATE TABLE employees (
 id INT,
 name VARCHAR(50),
 department_id INT
);
```

Insert department data for three departments, `HR`, `IT`, and `Marketing`, into the `departments` table.

```
INSERT INTO departments (id, name) VALUES
(1, 'HR'),
(2, 'IT'),
(3, 'Marketing');
```

Insert employee data for four employees, `Alice`, `Bob`, `Charlie`, and `David`, into the `employees` table.

```
INSERT INTO employees (id, name, department_id) VALUES
 (1, 'Alice', 1),
 (2, 'Bob', 2),
 (3, 'Charlie', 2),
 (4, 'David', 4);
```

### Find Employees Belonging to an Existing Department

This example uses an `EXISTS` clause to find any names from the `employees` table who are assigned to a department listed in the `departments` table.

```
SELECT name
FROM employees AS e
WHERE EXISTS (
 SELECT *
 FROM departments AS d
 WHERE e.department_id = d.id
);
```

The query returns all employees except David, who is assigned to a department that does not exist in the `departments` table.

*Output*

```
Bob
Charlie
Alice
```

### Find Departments That Have Employees

This query returns departments that have at least one employee. The subquery uses `SELECT 1` to check whether any matching rows exist in the `employees` table. The output is the same if the query uses `SELECT *` instead.

```
SELECT name
FROM departments AS d
WHERE EXISTS (
 SELECT 1
 FROM employees AS e
 WHERE e.department_id = d.id
);
```

The query results exclude the `Marketing` department because no employees belong to it.

*Output*

```
HR
IT
```

### Filter Departments Based on an Uncorrelated Table

In this example, the outer query filters the `departments` table where `id != 3` (excluding the `Marketing` department). The subquery checks if there is at least one employee with a department identifier less than 4.

The query is uncorrelated because the subquery does not reference the `departments` table.

```
SELECT *
FROM departments d
WHERE d.id != 3
AND EXISTS (
 SELECT *
 FROM employees e
```

```
WHERE e.department_id < 4
);
```

*Output*

```
HR
IT
```

### Find Employees Without a Valid Department

This example uses a `NOT EXISTS` clause. The example returns only employees who are assigned to a department identifier `department_id` not listed in the `departments` table.

```
SELECT name
FROM employees AS e
WHERE NOT EXISTS (
 SELECT *
 FROM departments AS d
 WHERE e.department_id = d.id
);
```

*Output:* David

### SIMILAR TO Operator

`SIMILAR TO` is a keyword that extends the `LIKE` operator, adding more features for match filtering including many metacharacters used in regular expressions.

`%` and `_` both act as wildcard operators, but other supported metacharacters match traditional regular expressions.

#### Syntax

```
WHERE string1 SIMILAR TO string2
```

This table describes the metacharacters supported by the `SIMILAR TO` keyword.

| Metacharacter      | Description                                                                                                                             |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <code>%</code>     | Repeated wildcard, matches any characters zero or more times. Equivalent usage to <code>LIKE</code> .                                   |
| <code>-</code>     | Wildcard, matches exactly one character. Equivalent usage to <code>LIKE</code> .                                                        |
| <code> </code>     | Alteration, meaning either of two alternatives ( <code>a b</code> represents <code>a</code> or <code>b</code> ).                        |
| <code>*</code>     | Repetition of zero or more times.                                                                                                       |
| <code>+</code>     | Repetition of one or more times.                                                                                                        |
| <code>?</code>     | Repetition of zero or one times.                                                                                                        |
| <code>{m}</code>   | Repetition of exactly <code>m</code> times.                                                                                             |
| <code>{m,n}</code> | Repetition between <code>m</code> and <code>n</code> times (inclusive).                                                                 |
| <code>{m,}</code>  | Repetition of at least <code>m</code> times.                                                                                            |
| <code>()</code>    | Logical grouping.                                                                                                                       |
| <code>[]</code>    | A character class, equivalent to character classes in regular expressions. Example: <code>[a-z]</code> is any lowercase English letter. |
| <code>^</code>     | Beginning of the line anchor and the negation in character classes.                                                                     |
| <code>\$</code>    | End of the line anchor.                                                                                                                 |

`SIMILAR TO` also supports escaping these metacharacters with `\` and certain escape sequences supported by C family languages.

| Metacharacter | Description                                                                                                                                                                                                          |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \a            | The alert/bell character.                                                                                                                                                                                            |
| \b            | The backspace character.                                                                                                                                                                                             |
| \B            | A single\character. Equivalent to\\.                                                                                                                                                                                 |
| \cX           | The character with the same five lower order bits asXwith all zeros as higher order bits.                                                                                                                            |
| \e            | The character with the numeric value 27, Esc.                                                                                                                                                                        |
| \f            | Form feed.                                                                                                                                                                                                           |
| \n            | Newline.                                                                                                                                                                                                             |
| \r            | Carriage return.                                                                                                                                                                                                     |
| \t            | Horizontal tab.                                                                                                                                                                                                      |
| \uvwxyz       | wxyzrepresents four hexadecimal digits, including two characters: one with numeric valuewx, and one with numeric valueyz.                                                                                            |
| \Uabcdefgh    | abcdefghrepresents eight hexadecimal digits, including four characters: numeric valueab, numeric valuecd, numeric valueef, and numeric valuegh.                                                                      |
| \v            | Vertical tab.                                                                                                                                                                                                        |
| \xd....       | dis any number of hexadecimal digits. If the number of digits is odd, a 0 is added to the left of the sequence. Numeric characters with values represented by every group of two symbols fromd.                      |
| \0            | NULL character 0.                                                                                                                                                                                                    |
| \xy           | xyare octal digits that represent the character with the numeric valuexy.                                                                                                                                            |
| \xyz          | xyzare octal digits that represent the character with the numeric valuexyz. Ifxis greater than or equal to4, the database returns two characters: the numeric value1and the numeric value of the lower 8 bits ofxyz. |
| \d            | Matches any digit.                                                                                                                                                                                                   |
| \s            | Matches any whitespace.                                                                                                                                                                                              |
| \w            | Matches any alphanumeric character or_.                                                                                                                                                                              |
| \D            | Matches any non-digit.                                                                                                                                                                                               |
| \S            | Matches any non-whitespace.                                                                                                                                                                                          |
| \W            | Matches any non-alphanumeric that is also not_.                                                                                                                                                                      |
| \A            | The beginning of a line. Equivalent to^.                                                                                                                                                                             |
| \m,\M, or\y   | The boundary of a word. Equivalent to\b in normal regular expressions.                                                                                                                                               |
| \Y            | Any position that is not the boundary of a word.                                                                                                                                                                     |
| \Z            | The end of a line. Equivalent to\$in normal regular expressions.                                                                                                                                                     |
| \*            | The star character.                                                                                                                                                                                                  |
| \+            | The plus character.                                                                                                                                                                                                  |
| \?            | The question mark character.                                                                                                                                                                                         |

- The database ignores any \ metacharacter that does not precede a metacharacter (including \) or an escape sequence in the metacharacter table.
- Invalid expressions (unpreceded quantifiers, unclosed parentheses, etc.) result in query errors.
- The database does not recognize the . character as a metacharacter, unlike the behavior in traditional regular expressions.
- The database applies escape metacharacters that apply to string literals. To match a string containing a single '\', you might need to use '\\\\'.

## Examples

```
'abc' SIMILAR TO 'abc' true
'abc' SIMILAR TO 'a' false
'abc' SIMILAR TO '%(b|d)%' true
'abc' SIMILAR TO '(b|c)%' false
'-abc-' SIMILAR TO '%\mabc\M%' true
'xabcy' SIMILAR TO '%\mabc\M%' false
```

## Array Filters

A filter expression can use the functions `FOR_SOME()` and `FOR_ALL()` to apply a predicate against all values of an input array.

Array filter functions have the following rules:

- They can only evaluate array types.
- They can go on either the left or right side of a boolean comparison expression, but not both sides at the same time.
- They must directly use a boolean comparison operator.
- They cannot use the `SOME` and `ALL` SQL keywords.

## Filter Behavior with Empty Arrays

Array filter functions have unique behavior when evaluating empty arrays.

- `FOR_ALL()` evaluates an empty array as `TRUE`.
- `FOR_SOME()` evaluates an empty array as `FALSE`.

If empty arrays must be evaluated for a different result, you can use the `ARRAY_LENGTH` function to specify a minimum array length. For example, this statement would evaluate an array as `TRUE` only if it is not empty and all values matched `%ocient%`:

```
ARRAY_LENGTH(array_col) > 0 AND FOR_ALL(array_col) LIKE '%ocient%
```

For details about array functions, see the [Array Functions and Operators](#) page.

## Filter Behavior with NULL rows

Filtering with array functions can have different outcomes depending on whether the array row is `NULL` or if the values contained in the array are `NULL`.

If `FOR_SOME()` or `FOR_ALL()` evaluate a `NULL` row (the row itself is `NULL`, not that the array contains `NULL` values), then the result is always `FALSE`.

To check an array for the presence of `NULL` values, you must use the `IS NULL` operator. For example:

```
FOR_SOME(array_col) IS NULL
```

Array comparison operators, such as `@>`, `<@`, and `&&` ignore `NULL` values. For details, see the [Array Functions and Operators](#) page.

## GROUP BY

Groups rows with the same values into summary rows, based on a specified aggregate function.

### Syntax

```
SELECT <select_list_clause> FROM table_name
 GROUP BY { column_name | expression | integer } [, ...]
 [HAVING <filter_condition>]
```

### Parameters

| Parameter            | Description                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <select_list_clause> | A list of one or more columns for the <code>SELECT</code> query. To use <code>GROUP BY</code> to summarize the query result set, at least one of the columns in <code>column_list</code> must include an aggregate function, such as <code>SUM()</code> , <code>COUNT()</code> , or <code>MAX()</code> . For a list of supported functions, see the <a href="#">Aggregate Functions</a> page. |
| table_name           | The table to use for the query.                                                                                                                                                                                                                                                                                                                                                               |
| column_name          | The name of one or more columns to use to group the result set. If you specify multiple columns, the result set is grouped by each unique combination of values from the columns.                                                                                                                                                                                                             |
| expression           | Any combination of literal values, column names, arithmetic expressions, parentheses, and function calls.                                                                                                                                                                                                                                                                                     |
| integer              | An integer representing the position of the columns referenced in the <code>column_list</code> . The first position starts at 1.                                                                                                                                                                                                                                                              |
| <filter_condition>   | A <code>HAVING</code> clause that filters the aggregated groups based on a specified condition. For details, see the <a href="#">General SQL Syntax</a> section.                                                                                                                                                                                                                              |

### Example

This example uses a movie database to calculate the total amount spent on movie production per year.

```
SELECT YEAR(release_date),
 SUM(budget)
FROM movies
GROUP BY 1
ORDER BY 1 ASC;
```

*Output*

| year(release_date) | sum(budget) |
|--------------------|-------------|
| 1968               | 10500000    |
| 1979               | 31500000    |
| 1981               | 18000000    |
| 1982               | 28000000    |
| 1992               | 14000000    |
| 1997               | 200000000   |
| 2003               | 264000000   |
| 2009               | 237000000   |
| 2010               | 350000000   |
| 2012               | 670000000   |
| 2013               | 350000000   |
| 2015               | 414000000   |

### HAVING

Filters aggregated rows based on a specified condition.

`HAVING` operates in a `GROUP BY` statement by setting a filter for the rows to be aggregated and grouped.

For information on using `GROUP BY` in a query statement, see [General SQL Syntax](#).

### Syntax



```
GROUP BY column_name HAVING <filter_condition>
```

### Parameters

| Parameter          | Description                                                                                                           |
|--------------------|-----------------------------------------------------------------------------------------------------------------------|
| column_name        | A column used for grouping that is evaluated by the filter_condition.                                                 |
| <filter_condition> | A logical combination of boolean predicates. For information on supported logical predicates, see General SQL Syntax. |

### Example

This example calculates the total amount spent on movie production per year. The **HAVING** clause filters out any rows that do not have a sum of at least \$100 million.

```
SELECT YEAR(release_date),
 SUM(budget)
FROM movies
GROUP BY 1
HAVING SUM(budget) > 100000000
ORDER BY 1 ASC;
```

### Output

| year(release_date) | sum(budget) |
|--------------------|-------------|
| 1997               | 200000000   |
| 2003               | 264000000   |
| 2009               | 237000000   |
| 2010               | 350000000   |
| 2012               | 670000000   |
| 2013               | 350000000   |
| 2015               | 414000000   |

## ORDER BY

Sorts the result set in ascending or descending order based on one or more specified columns.

If you specify multiple columns, they are sorted hierarchically from left to right.

### Syntax

```
ORDER BY { column_position | column_name } [ASC | DESC]
 [NULLS FIRST | NULLS LAST] [, ...]
```

### Parameters

| Parameter       | Data Type | Description                                                                                       |
|-----------------|-----------|---------------------------------------------------------------------------------------------------|
| column_position | Integer   | The position of a column to be used for the <b>ORDER BY</b> sorting. Column positions start at 1. |
| column_name     | String    | The name of a column to be used for the <b>ORDER BY</b> sorting.                                  |

| Parameter                | Data Type | Description                                                                                                                                                                 |
|--------------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ASC   DESC               | String    | Optional.Specifies whether to sort the column in ascending (ASC) or descending (DESC) order.If unspecified, defaults toASC.                                                 |
| NULLS FIRST   NULLS LAST | String    | Optional.NULLS FIRSTmeans null values go at the start of the result set.NULLS LASTmeans null values go at the end of the result set.If unspecified, defaults toNULLS FIRST. |

### Example

In this example, the `ORDER BY` statement orders the movies from newest to oldest.

```
SELECT release_date, title FROM movies
ORDER BY release_date DESC;
```

*Output*

| release_date | title                        |
|--------------|------------------------------|
| 2015-06-17   | Minnows                      |
| 2015-06-09   | Triassic World               |
| 2015-04-01   | Fury 7                       |
| 2013-11-27   | Frigid                       |
| 2013-04-18   | Iron Chef 3                  |
| 2012-10-25   | Spyman 16                    |
| 2012-07-16   | Superhero 23                 |
| 2012-04-25   | The Tentpole                 |
| 2010-06-30   | The Last Airman              |
| 2010-06-16   | Merchandise Vehicle 5        |
| 2009-12-10   | CGI Why                      |
| 2003-12-01   | Swords & Scabbards           |
| 2003-10-10   | Billy the Killy              |
| 2003-07-09   | Pirates of Palm Springs      |
| 1997-11-18   | Titania                      |
| 1992-08-07   | Bang Bang Western            |
| 1982-06-25   | Blade Walker                 |
| 1981-06-12   | Raiders of the Jungle Temple |
| 1979-08-15   | Apocalypse None              |
| 1968-04-10   | Space Odyssey 6000           |

### LIMIT

Limits the number of returned rows from a query to a specified amount.

The returned rows are nondeterministic unless you use an [General SQL Syntax](#) clause.

#### Syntax

```
<select_query> LIMIT limit_number
```

#### Parameters

| Parameter      | Description                                                                                                                                          |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <select_query> | A valid <code>SELECT</code> query.                                                                                                                   |
| limit_number   | The number of rows to return from the query, specified as a positive integer or any constant scalar expression that evaluates to a positive integer. |

## Examples

### Limit Returned Rows Using a Number

This example uses `LIMIT` to restrict the number of returned movie titles to only three.

```
SELECT title FROM movies LIMIT 3;
```

*Output*

```
Minnows
CGI Why
Billy the Killy
```

### Limit Returned Rows Using an Expression

The `LIMIT` SQL statement can also accept expressions. This query uses the `1+2` expression with the `sys.dummy` table to create a table of three incrementing integers. For details, see [Generate Tables Using sys.dummy](#).

```
SELECT c1 FROM sys.dummy10 LIMIT 1+2;
```

*Output*

```
1
2
3
```

## OFFSET

Skips a specified number of rows from the result set.

The returned rows are nondeterministic unless you use an [General SQL Syntax](#) clause.

### Syntax

```
<select_query> OFFSET offset_number
```

### Parameters

| Parameter      | Description                                                                                                                                                               |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <select_query> | A valid <code>SELECT</code> query.                                                                                                                                        |
| offset_number  | The number of rows to skip when returning results from the query, specified as a positive integer or any constant scalar expression that evaluates to a positive integer. |

## Examples

### Skip Rows From the Result Set Using a Number

In this example, the `ORDER BY` statement orders the results chronologically. As a result, the `OFFSET` clause removes the oldest six movies from the result set.

```
SELECT release_date,
 title
FROM movies
ORDER BY release_date OFFSET 6;
```

*Output*

| release_date | title                   |
|--------------|-------------------------|
| 2003-07-09   | Pirates of Palm Springs |
| 2003-10-10   | Billy the Killy         |
| 2003-12-01   | Swords & Scabbards      |
| 2009-12-10   | CGI Why                 |
| 2010-06-16   | Merchandise Vehicle 5   |
| 2010-06-30   | The Last Airman         |
| 2012-04-25   | The Tentpole            |
| 2012-07-16   | Superhero 23            |
| 2012-10-25   | Spyman 16               |
| 2013-04-18   | Iron Chef 3             |
| 2013-11-27   | Frigid                  |
| 2015-04-01   | Fury 7                  |
| 2015-06-09   | Triassic World          |
| 2015-06-17   | Minnows                 |

### Skip Rows From the Result Set Using an Expression

The OFFSET SQL statement can also accept expressions. This query uses the expression `3+4` with the `sys.dummy` table to create a column of incrementing integers by skipping the first seven out of 10 rows. For details, see [Generate Tables Using sys.dummy](#).

```
SELECT c1 FROM sys.dummy10 OFFSET 3+4;
```

*Output*

```
8
9
10
```

## INTERSECT

Returns any rows that match between two separate **SELECT** queries. To use **INTERSECT**, the two queries must be compatible, meaning they must return the same number of columns and have similar data types.

By default, the SQL statement eliminates duplicate rows unless the query includes the optional **ALL** keyword. Using the **DISTINCT** keyword is the same as this default behavior.

### Syntax

```
<select_query1> INTERSECT [ALL | DISTINCT] <select_query2>
```

### Parameters

| Parameter       | Description           |
|-----------------|-----------------------|
| <select_query1> | A valid SELECT query. |
| <select_query2> | A valid SELECT query. |

### Example

This example has two **SELECT** queries with an **INTERSECT** statement. The full query retrieves movies that had a budget of at least \$200 million, but also earned more than \$1 billion in revenues.

```
SELECT *
FROM movies
WHERE revenue > 1000000000
INTERSECT
SELECT *
FROM movies
WHERE budget > 200000000;
```

*Output*

| movie_id | title              | budget    | popularity | release_date | revenue    | runtime | movie_status | note_average | note_count |
|----------|--------------------|-----------|------------|--------------|------------|---------|--------------|--------------|------------|
| 597      | Titania            | 200000000 | 100.025899 | 1997-11-18   | 1845034188 | 194     | Released     | 7.5          | 7562       |
| 19995    | CGI                | 237000000 | 150.437577 | 2009-12-10   | 2787965087 | 162     | Released     | 7.2          | 11800      |
| 49026    | Why Superhero      | 250000000 | 112.31295  | 2012-07-23   | 1084939099 | 165     | Released     | 7.6          | 9106       |
| 10193    | Merchandise        | 200000000 | 59.995418  | 2010-06-16   | 1066969703 | 103     | Released     | 7.6          | 4597       |
| 211672   | Vehicle 5 Minnows  | 74000000  | 875.581305 | 2015-06-17   | 1156730962 | 91      | Released     | 6.4          | 4571       |
| 122      | Swords & Scabbards | 94000000  | 123.630332 | 2003-12-01   | 1118888979 | 201     | Released     | 8.1          | 8064       |
| 37724    | Spyman 16          | 200000000 | 93.004993  | 2012-10-25   | 1108561013 | 143     | Released     | 6.9          | 7604       |
| 109445   | Frigid             | 150000000 | 165.125366 | 2013-11-27   | 1274219009 | 102     | Released     | 7.3          | 5295       |
| 168259   | Furious 7          | 190000000 | 102.322217 | 2015-04-01   | 1506249360 | 137     | Released     | 7.3          | 4176       |
| 68721    | Iron Chef 3        | 200000000 | 77.68208   | 2013-04-18   | 1215439994 | 130     | Released     | 6.8          | 8806       |
| 135397   | Triassic World     | 150000000 | 418.708552 | 2015-06-09   | 1513528810 | 124     | Released     | 6.5          | 8662       |
| 24428    | The Tentpole       | 220000000 | 144.448633 | 2012-04-25   | 1519557910 | 143     | Released     | 7.4          | 11776      |

## EXCEPT

The **EXCEPT** keyword returns the result set of a first **SELECT** query minus any matching rows from a second **SELECT** query.

**EXCEPT** requires the two queries to be compatible, meaning they must return the same number of columns and have similar data types. By default, the result set eliminates duplicate rows unless the query includes the optional **ALL** keyword. Using the **DISTINCT** keyword is the same as this default behavior.

The **EXCEPT** keyword can also be used to exclude specific columns from a **SELECT \*** query. For information on that alternate usage, see the [General SQL Syntax](#) syntax and example.

## Syntax

```
<select_query1> EXCEPT [ALL | DISTINCT] <select_query2>
```

## Parameters

| Parameter       | Description                                                                                                                                                      |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <select_query1> | A valid SELECT query that is compared toselect_query2. The results of an EXCEPT statement are the rows inselect_query1that do not match any row inselect_query2. |
| <select_query2> | A valid SELECT query that is compared toselect_query1.                                                                                                           |

### Example

In this example, the first **SELECT** statement queries all rows in the **movies** table. The **EXCEPT** clause and the second **SELECT** statement eliminate from the results any rows with movies that had a budget greater than 20000000.

```
SELECT *
FROM movies
EXCEPT
SELECT *
FROM movies
WHERE budget > 20000000;
```

### Output

| title                              | budget   | popularity | release_date | revenue   | runtime | movie_status | vote_average | vote_count |
|------------------------------------|----------|------------|--------------|-----------|---------|--------------|--------------|------------|
| Bang<br>Bang<br>Western            | 14000000 | 37.380435  | 1992-08-07   | 159157447 | 131     | Released     | 7.7          | 1113       |
| Space<br>Odyssey<br>6000           | 10500000 | 86.201184  | 1968-04-10   | 68700000  | 149     | Released     | 7.9          | 2998       |
| Raiders of<br>the Jungle<br>Temple | 18000000 | 68.159596  | 1981-06-12   | 389925971 | 115     | Released     | 7.7          | 3854       |

## UNION

Returns the combined result set of two or more **SELECT** queries.

By default, **UNION** eliminates duplicate rows from the result set unless you specify the **ALL** keyword. Using the **DISTINCT** keyword is the same as this default behavior.

### Syntax

```
<select_query1> UNION [ALL | DISTINCT] <select_query2> [...]
```

### Parameters

| Parameter     | Data Type | Description                                            |
|---------------|-----------|--------------------------------------------------------|
| select_query1 | String    | A validSELECTquery that is combined withselect_query2. |
| select_query2 | String    | A validSELECTquery that is combined withselect_query1. |

### Example

This example uses **UNION** to merge two separate queries for identical columns into the same result set.

```

SELECT title,
 popularity,
 vote_average
FROM movie
WHERE popularity > 800
UNION
SELECT title,
 popularity,
 vote_average
FROM movies.movie
WHERE vote_average > 7.5;

```

*Output*

| title                        | popularity | vote_average |
|------------------------------|------------|--------------|
| Raiders of the Jungle Temple | 68.159596  | 7.7          |
| Minnows                      | 875.581305 | 6.4          |
| Apocalypse None              | 49.973462  | 8            |
| Superhero 23                 | 112.31295  | 7.6          |
| Billy the Killy              | 79.754966  | 7.7          |
| Bang Bang Western            | 37.380435  | 7.7          |
| Blade Walker                 | 94.056131  | 7.9          |
| Swords & Scabbards           | 123.630332 | 8.1          |
| Merchandise Vehicle 5        | 59.995418  | 7.6          |
| Space Odyssey 6000           | 86.201184  | 7.9          |

## USING

Overrides various system configurations for processing a specified query.

The **USING** keyword is required for only the first query override, not for subsequent ones.

For descriptions of the supported query configurations, see the parameter table below.

### Syntax

```

<select_query>
 USING
 [SCHEDULING_PRIORITY = priority_value]
 [MAX_ROWS_RETURNED = max_rows]
 [MAX_ELAPSED_TIME = max_elapsed_time]
 [MAX_TEMP_DISK_USAGE = max_temp_disk_usage]
 [CACHE_MAX_TIME = cache_max_time]
 [CACHE_MAX_BYTES = cache_max_bytes]
 [SERVICE CLASS service_class]

```

### Parameters

| Parameter      | Description                  |
|----------------|------------------------------|
| <select_query> | A valid <b>SELECT</b> query. |

| Parameter                        | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>priority_value</code>      | An optional clause that specifies the priority at which the query should run. The priority value is a floating point number, which indicates the priority compared to the other queries. The priority value does not specify any percentage of resources that the query takes or any specific degree of difference between queries. If unspecified, defaults to <code>SCHEDULING_PRIORITY = 1.0</code> .                                                                                                                                                                                                                                                                                                                     |
| <code>max_rows</code>            | An optional clause that limits the number of rows returned by a query. If a query exceeds this number of rows, the system kills it. For example, <code>max_rows_returned = 10</code> kills any queries returning more than ten rows.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>max_elapsed_time</code>    | An optional clause that limits how long a query can run. If a query exceeds this time limit in seconds, the system kills it. For example, <code>max_elapsed_time = 10</code> kills any queries taking longer than 10 seconds.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>max_temp_disk_usage</code> | An optional clause that limits the percentage of temporary disk used by a query. If a query exceeds this temporary disk limit, the system kills it. For example, <code>max_temp_disk_usage = 10</code> kills any queries taking more than 10% of temporary disk space.                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>cache_max_time</code>      | An optional clause that affects whether a specified query uses results from the cache rather than executing the query. If there is a cached result with the same query text, executed less than <code>cache_max_time</code> seconds ago, the database uses that cached result. The default value is 0, and thus by default, the database does not return any cached results. The cache considers all SQL Nodes, and if a potential cached result is only available on a different SQL Node, the database redirects the query to that node. If you specify the <code>force</code> attribute on the connection, which disables load balancing and redirection, the database considers only cached results on the current node. |
| <code>cache_max_bytes</code>     | An optional clause that controls whether the database stores the results of any specified query in the cache. The system stores the results of all queries executed using a service class specifying this attribute in the cache if the result size is smaller than this value. You can determine the size of a result set in bytes by querying the <code>bytes_returned</code> field of the <code>completed_queries</code> virtual table. The default value is 0, and thus by default, the database does not cache any results. The database caches results in memory on the SQL Nodes. These results are not cached if there is insufficient memory available.                                                             |
| <code>service_class</code>       | An optional clause at the end of a query that sets a specific service class to run that query. For details, see <a href="#">Users, Groups, and Service Classes</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

### Example



```
SELECT * from sys.dummy10
 USING
 SCHEDULING_PRIORITY = 1.0
 MAX_ROWS_RETURNED = 10
 MAX_ELAPSED_TIME = 10
 MAX_TEMP_DISK_USAGE = 10;
```

#### Output

```
c1

1
2
3
4
5
6
7
8
9
10
```

## TRACE

An optional clause that executes the query, but discards the original result set. Instead, **TRACE** returns a result set of tracing data that describe the execution of the query.

Append the **TRACE** SQL statement to the end of a query. The statement can include optional parameters to control its frequency and level of detail.

To understand the results of this statement, see [General SQL Syntax](#).

Trace queries have the same effect to the workload as a query run without the clause.

Contact Ocient Support for guidance in using the **TRACE** SQL statement.

## Syntax

```
<select_query> TRACE
 [FREQUENCY frequency_int]
 [RESOLUTION resolution_int]
```

## Parameters

| Parameter      | Description                                                                                                                                                                                                          |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <select_query> | A valid <b>SELECT</b> query.                                                                                                                                                                                         |
| frequency_int  | Controls how often the database samples trace events during query execution in milliseconds. If unspecified, defaults to 500 milliseconds.                                                                           |
| resolution_int | Determines the level of detail captured for a trace or event. If the resolution of a trace is greater than the resolution of an event, the database records the event. If unspecified, defaults to 100 milliseconds. |

## TRACE Results

Each row of the trace output provides some information about what an operator instance has done in the time between the previous trace sample and the most current sample.

| Column         | Description                                                                                                                                                  |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| plan_parent_id | The UUID of the parent of the operator instance in the query plan. If an operator has multiple parents, this field specifies the UUID of one of the parents. |
| plan_op_id     | The UUID of the operator in the query plan.                                                                                                                  |
| operator_type  | The type of operator that the database uses to sample trace events.                                                                                          |
| node_id        | The UUID of the node where this operator executes.                                                                                                           |
| code_id        | The identifier of the VM core where this operator instance executes. Core identifiers are unique for each node.                                              |
| op_id          | The identifier of the operator instance. Identifiers are unique for each node.                                                                               |
| time           | The time in milliseconds when the database collects the information for the trace event relative to the start of the query.                                  |
| group          | Group related trace events together.                                                                                                                         |
| sample         | The sample of trace event data. For possible values, see the following table.                                                                                |
| value          | The value of trace event data.                                                                                                                               |

## Sample and Value Columns

The `sample` and `value` columns describe what occurred in a given trace period.

| Sample              | Description                                                                     |
|---------------------|---------------------------------------------------------------------------------|
| ROWS_IN             | The number of rows an operator instance has read from its children operators.   |
| ROWS_OUT            | The number of rows the operator instance returns.                               |
| BLOOM_FILTERED_ROWS | The number of rows that this operator instance discards by using Bloom filters. |
| SCHEDULE_CYCLE      | The number of cycles in the schedule for this operator instance.                |
| OOM_CYCLE           | The number of out-of-memory cycles in the schedule for this operator.           |
| INITIALIZE          | The time when the database initializes the operator.                            |
| FINALIZE            | The time when the database finalizes the operator.                              |

## Example

```
SELECT * FROM sys.dummy10 TRACE;
```

*Output*

| plan_id                              | parent_id                                 | op_id           | operator_type | code_id      | core_id | op_id | time | group                       | sample | value               |
|--------------------------------------|-------------------------------------------|-----------------|---------------|--------------|---------|-------|------|-----------------------------|--------|---------------------|
| 77041d2b-f7ca-49d5-a3f5-4b75b0782ed1 | 498690cf-4a9b-4c1f-affa-a3f5-417f31fddc2d | RENAME_OPERATOR | OPERATOR      | 2174ed048d33 |         | 8441  | 0    | xg::db::vm::block::operator | ROWS   | OperatorTraceEvents |
| 77041d2b-f7ca-49d5-a3f5-4b75b0782ed1 | 498690cf-4a9b-4c1f-affa-a3f5-417f31fddc2d | RENAME_OPERATOR | OPERATOR      | 2174ed048d33 |         | 8441  | 0    | xg::db::vm::block::operator | ROWS   | OperatorTraceEvents |
| 77041d2b-f7ca-49d5-a3f5-4b75b0782ed1 | 498690cf-4a9b-4c1f-affa-a3f5-417f31fddc2d | RENAME_OPERATOR | OPERATOR      | 2174ed048d33 |         | 8441  | 0    | xg::db::vm::block::operator | ROWS   | OperatorTraceEvents |

## String Literals and Escape Sequences

All string literals in SQL statements must be enclosed in single quotes. To use a single quote within a string, you can use another single quote as an escape, `''`.

For other escape sequences, include an `e` character before the string literal, i.e., before the opening single quote. This directs the system to recognize escape sequences in the string literal. This means:

- All single `\` characters in the string now escape themselves.
- Any subsequent character after the `\` is also escaped if it matches an escape sequence (see the table).

If you use escape sequences, be prepared that you might need to alter strings that include `\`, such as directory paths.

## Supported Escape Sequences

| Escape Sequence | Description                           |
|-----------------|---------------------------------------|
| <code>''</code> | Single quotation mark                 |
| <code>\"</code> | Double quotation mark                 |
| <code>\n</code> | Newline character                     |
| <code>\r</code> | Carriage return character             |
| <code>\f</code> | Form feed character (i.e. page break) |
| <code>\b</code> | Backspace                             |
| <code>\\</code> | Backslash                             |
| <code>\t</code> | Tab                                   |

## Examples

These examples show how the system interprets strings with and without escape sequences.

### String Without an Escape Sequence

This example selects the simple string literal `'\my\directory\path'` that does not use escape sequences.

```
SELECT '\my\directory\path';
```

*Output:* `\my\directory\path`

### String With an Escape Sequence

This example selects the simple string literal `'\my\directory\path'` and uses an escape sequence. The result omits the backslashes.

```
SELECT e'\my\directory\path';
```

Output: mydirectorypath

### String With an Escape Sequence to Retain the Backslashes

This example uses the '\\my\\directory\\path' string with an escape sequence to include backslashes in the result.

```
SELECT e'\\my\\directory\\path';
```

Output: \my\directory\path

### Escape Sequences with Regular Expressions

Be careful using escape sequences with strings that also use regular expressions. Escape sequences in SQL syntax override the escape sequences of regular expressions.

In this example, the [Character and Binary Functions](#) function uses the regular expression \w+ to match any word characters.

```
SELECT REGEXP_SUBSTR(
 'abcdefghijklmnopqrstuvwxy',
 '\w+'
);
```

Output: abcdefghijklmnopqrstuvwxy

The function behaves differently if the regular expression is an escape sequence because it overrides the \ character. The regular expression searches only for the character w.

```
SELECT REGEXP_SUBSTR(
 'abcdefghijklmnopqrstuvwxy',
 e'\w+'
);
```

Output: w

### Related Links

[SQL Reference](#)

[Data Control Language \(DCL\) Statement Reference](#)

[Data Definition Language \(DDL\) Statement Reference](#)

#### 4.2.1 SQL Syntax Conventions

##### Syntax Conventions

The Ocient® documentation uses the following conventions for defining SQL statements in syntax blocks, unless otherwise noted.

| Syntax symbol | Definition                                                                                                                                      | Example                                                                                                                   |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| KEYWORD       | SQL statement names, SQL statement option names, stored procedure names, utility names, utility flags, and other SQL keywords are in uppercase. | SELECT * FROM <my_table> LIMIT 100; SELECT, FROM, and LIMIT are all SQL keywords. For that reason, they are in uppercase. |

| Syntax symbol | Definition                                                                                                                                                                                                                                                | Example                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| { }           | Curly braces indicate that you must choose at least one of the enclosed options. Do not include the curly braces in your SQL statement.                                                                                                                   | INSERT INTO table_name [( col1 [, col2 ] [, ...] ) ] { <select_statement>   VALUES ( [ val1 [, val2] [, ...] ) } This syntax example has two options to define what data to add using INSERT INTO: either provide specific values, or use an embedded SELECT query. These options are enclosed in curly braces because the user must use one of these options to successfully run an INSERT INTO SQL statement. |
| [ ]           | Brackets mean the enclosed functionality is optional, not necessary to run the specified SQL statement. Do not include the brackets in your SQL query. When used with a list of options, the brackets mean that you can choose one of the listed options. | CREATE CLUSTER [ IF NOT EXISTS ] cluster_name ( <create_definition> [, ...] ) This syntax example encloses IF NOT EXISTS in brackets because it is optional for running the CREATE CLUSTER SQL statement.                                                                                                                                                                                                       |
|               | The vertical pipe acts as a delimiter for option lists, meaning you can select only one of the options.                                                                                                                                                   | EXPLAIN [ PROTO   JSON   DEBUG ] <query> This example shows three options that the EXPLAIN SQL statement can use. The vertical pipe indicates that each option is exclusive, meaning only one can be used.                                                                                                                                                                                                      |
| [ , ... ]     | The ellipsis means you can choose to add as many of the indicated options or parameters as you like, separating your choices with commas to be typed as part of the SQL statement.                                                                        | INSERT INTO <table_name> [ ( <column_name> [, ...] ) ] <query> The INSERT INTO statement uses the ellipsis [, ...] to indicate that an infinite number of additional <column_name> values can be specified.                                                                                                                                                                                                     |

| Syntax symbol       | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Example                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <syntax_subsection> | For complex syntax blocks, portions are set apart with angle brackets<>to denote there is a separate subsection for a particular portion of the syntax. Syntax blocks with these subsections usually include complicated options that might distract from the core functionality of the syntax of the SQL statement. The names of syntax subsection are written in lowercase with underscores between the words. In the syntax block, the ::= symbol signifies the subsection. | <pre> CREATE TABLE [ IF NOT EXISTS   OR REPLACE ] table_name [ (&lt;create_definition&gt; [, ...] ) ] [ [ WITH ] &lt;create_option&gt; [, ...] ] [ AS (query) ] ``&lt;create_definition&gt; ::= &lt;column_definition&gt;   &lt;index_definition&gt; ``&lt;create_option&gt; ::= STORAGE SPACE storage_space_name   SEGMENT SIZE segment_value   REDUNDANCY segment_part (redundancy_scheme)   STREAM LOADER PROPERTIES streamloader_json   CLUSTERING [KEY   INDEX] clustering_index (clustering_column [, ...])   INDEX index_name (index_column [, ...]) [ USING index_type ] ``&lt;column_definition&gt; ::= column_name [ data_type ] [ &lt;column_constraint&gt; [, ...] ] ``&lt;index_definition&gt; ::= [ CLUSTERING ] KEY index_name (index_column [, ...] ) ``&lt;column_constraint&gt; ::= TIME KEY BUCKET(bucket_value [, DAY, HOUR, MINUTE, SECOND ])   [ NOT ] NULL   DEFAULT literal   COMMENT comment   COMPRESSION GDC [(compression_value), EXISTING schema_name]   MAX SIZE (size_value)   [ NO ] COMPRESSION [ compression_scheme ] </pre> <p>In this example, CREATE TABLE uses subsections for complex options, such as its column definition, options, and column constraints. Including all of these in one block would be difficult for a user to parse, especially if their goal was only to create a basic table. Note that each subsection is defined with the name of the subsection in angle brackets followed by ::=.</p> |

## Rules for Column and Table Names

Names for tables and columns in Ocient have the following rules:

- Names must begin with an alphabet letter.
- Names can contain letters, numbers, and underscores.
- Column names enclosed in double quotes can contain any characters except for newlines and carriage returns. To use double quotation in a name, put two double quotation marks back to back.
- Unless names are enclosed in double quotes, they are treated as case insensitive.

## Related Links

[SQL Reference](#)

[Data Control Language \(DCL\) Statement Reference](#)

[Data Definition Language \(DDL\) Statement Reference](#)

### 4.2.2 Identifiers

In the Ocient® System, identifiers are lexical tokens that name specific entities in the database. For example, a table name is an identifier.

In most cases, you do not have to quote an identifier. However, the identifier must meet these two conditions:

- Start with a letter followed by zero or more letters, digits, or underscores.
- Must not be a reserved word. For the full list of reserved words, see [Reserved Words](#).

If the identifier does not meet either of these conditions, then enclose the identifier in double quotes "" to create a delimited identifier. SQL is case-insensitive, and the database normalizes identifiers to lowercase. For example, you must add double quotes to the `1start_with_digit`, `hy-phen`, and `mixedCase` identifiers so that the database can parse them correctly: `"1start_with_digit"`, `"hy-phen"`, and `"mixedCase"`.

The system uses identifiers to identify these objects:

- Database name
- Schema name
- Function name
- Table name
- Table alias name
- Column name
- Column alias name
- Index name
- View name
- Task name
- Node names
- Machine learning model name
- Connectivity pool name
- Connectivity pool participant name
- Storage space name
- Cluster name
- Service class name
- Single sign-on (SSO) protocol
- Tag name
- JSON key
- Username
- Group name
- Common table expression
- Subexpression name
- Data pipeline name
- Data pipeline type

## Related Links

[SQL Syntax Conventions](#)

[Database, Tables, Views, and Indexes](#)

[SQL Reference](#)

[Machine Learning Model Functions](#)

[Cluster and Node Management](#)

[Users, Groups, and Service Classes](#)

### 4.2.3 Generate Tables Using `sys.dummy`

`sys.dummy` is a virtual table that the Ocident Hyperscale Data Warehouse™ generates with a specific number of rows. This table is a handy tool to quickly create tables for testing without scripting extensive DDL statements.

When you execute `sys.dummy`, the database creates a table with a single column, which is automatically populated with incremental integers.

`sys.dummy` follows similar rules to a table reference. The SQL statement must specify a non-negative integer that represents the number of rows for the SQL statement to generate in the table.

In this syntax block, the `{N}` parameter represents the number of rows for `sys.dummy` to generate.

#### Syntax

```
sys.dummy{N}
```

| Parameter        | Description                                                                                                                                                                                                                                                                                                                                                                |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>{N}</code> | The number of rows for <code>sys.dummy</code> to generate. The minimum numeric value for <code>{N}</code> is 0, and the maximum is 9223372036854775807. This parameter must not have any leading or trailing characters. If <code>{N}</code> does not adhere to these limits, the database returns an error. Do not specify curly brackets with this parameter in queries. |

#### Example

Create a table with 10 rows.

```
SELECT * FROM sys.dummy10;
```

*Output:*

```
c1

1
2
3
4
5
6
7
8
9
10
```



## Related Links

[General SQL Syntax](#)

[SQL Reference](#)

## 4.3 Ocient HTTP Query API

The Ocient Hyperscale Data Warehouse™ has the HTTP Query API that enables you to execute SQL statements. This page uses command-line :Commentexamples using `curl`.

All the examples assume that the `openapi_port` is the standard SSL port 443. If your connectivity pool `openapi_port` has a different port setting, you must specify it in the request URL. For example, this URL specifies using port 8443: `https://sql_node_address:8443/v1/execute/`.

### API Documentation

The API follows OpenAPI Specification (Swagger®) guidelines, which you can access in these formats from any SQL Node address `{sql_node_address}`:

- YAML format: `/openapi.yaml` endpoint
- JSON format: `/openapi.json` endpoint

### YAML Download Example

```
curl https://{sql_node_address}/openapi.yaml
```

### JSON Download Example

```
curl https://{sql_node_address}/openapi.json
```

You can use these specifications with various OpenAPI tools for code generation, testing, and documentation.

### Connection Setup

Before using the API, you must establish an open port for a SQL Node. To do this, you must assign a SQL Node as a participant to a connectivity pool.

The connectivity pool defines connection settings for nodes, including their open port and address. For details, see [Cluster and Node Management](#).

### Enable the OpenAPI Port

To open a port, execute this [Cluster and Node Management](#) SQL statement on a connectivity pool in your system. Change the connectivity pool and node values in the example according to your system.

```
ALTER CONNECTIVITY_POOL benchmark_sql_connectivity_pool
 ALTER PARTICIPANT "benchmark-sql0" SET openapi_port 443;
```

### Check Port Configuration

Verify your port configuration by executing this query.

```
SELECT c.name, n.name, openapi_port
 FROM sys.connectivity_pool_participants AS cp
 LEFT JOIN sys.connectivity_pools AS c ON cp.id = c.id
 LEFT JOIN sys.nodes AS n ON cp.node_id = n.id;
```

The output displays all configured connectivity pools with their associated nodes and OpenAPI ports.

## Authentication Methods

The Ociant® HTTP Query API accepts user credentials in various formats designed for different use cases.

### Simple API Requests

For simple queries, you can include user credentials in the request.

In this example, the request includes credentials after the `-u` flag.

```
curl -u 'admin@system:admin' https://sql_node/v1/execute/system \
 -d '{"statement": "SELECT * FROM sys.dummy2;", "format": "collection"}'
```

### Bearer Token Authentication

For continuous access, you can obtain an authentication token by making a request to the `login` endpoint. Replace the text in this example with the address of your SQL Node, your username, your password, and the name of your database.

```
curl -X POST https://sql_node/v1/login \
 -H "Content-Type: application/json" \
 -d '{
 "username": "admin@system",
 "password": "admin",
 "database": "system"
 }'
```

This request returns an `:Commentauthentication` token that you can add to subsequent requests.

```
curl -X POST https://sql_node/v1/execute \
 -H "Authorization: Bearer TEST_ACCESS_TOKEN" \
 -H "Content-Type: application/json" \
 -d '{
 "statement": "SELECT * FROM sys.dummy2;",
 "database": "system",
 "format": "collection"
 }'
```

### OpenID Connect (SSO)

For SSO authentication, you can use OpenID Connect by targeting the `sso_authentication` endpoint with a callback path. Replace the text in this example with the name of your instance, the name of your database, and your callback path.

```
curl -X POST https://ocient-instance/v1/sso_authentication \
 -H "Content-Type: application/json" \
 -d '{
 "database": "system",
 "sso_callback_path": "/app_callback"
 }'
```

After the redirection and authentication are complete, you can include your token in subsequent requests.

```
curl -X POST https://ocient-instance/v1/sso_token \
 -H "Content-Type: application/json" \
 -d '{
 "id_token": "TEST_ID_TOKEN",
 "database": "system"
 }'
```

## API Endpoint Documentation

The Ociant HTTP Query API supports these endpoints.

For API requests, JSON data must use double quotes (") for all keys and string values. Single quotes should enclose the JSON payload.

Single or double quotes are valid for parameters not nested in JSON objects.

POST /v1/execute/{database}

#### Description

Executes a SQL statement and returns results. Supports standard and streaming responses using HTTP chunked transfer encoding.

#### URL

https://{sql\_node}/v1/execute/{database}

#### Method

POST

Path Parameters

| Name     | Type   | Required | Description                                            |
|----------|--------|----------|--------------------------------------------------------|
| sql_node | string | Yes      | IP/domain of a SQL Node.                               |
| database | string | No       | Overrides the default database from token or settings. |

Query Parameters

| Name   | Type   | Required | Description                                           |
|--------|--------|----------|-------------------------------------------------------|
| format | string | No       | array or collection. Applies with <b>text/plain</b> . |
| schema | string | No       | Target schema (applies with <b>text/plain</b> ).      |

Header Parameters

| Name                        | Type    | Required | Description                                                                                                  |
|-----------------------------|---------|----------|--------------------------------------------------------------------------------------------------------------|
| authorization               | string  | Yes      | Token (Basic or Bearer).                                                                                     |
| content-type                | string  | Yes      | Format of request body (e.g., <b>application/json</b> ).                                                     |
| accept-encoding             | string  | No       | Response compression ( <b>gzip</b> , <b>deflate</b> , <b>br</b> ).                                           |
| accept                      | string  | No       | Media type: <b>application/json</b> (default), <b>application/stream+json</b> , <b>application/parquet</b> . |
| preferred-encoding          | string  | No       | Preferred encoding when multiple values present.                                                             |
| preferred-compression-level | integer | No       | Compression level: 1 (fastest) to 9 (max).                                                                   |

## Body Parameters

| Name              | Type    | Required | Description                         |
|-------------------|---------|----------|-------------------------------------|
| <b>statement</b>  | string  | Yes      | SQL query.                          |
| <b>database</b>   | string  | No       | Overrides path or token database.   |
| <b>format</b>     | string  | No       | <b>array</b> or <b>collection</b> . |
| <b>params</b>     | object  | No       | Key-value for substitution.         |
| <b>fetch_size</b> | integer | No       | Rows per chunk (0 = auto).          |
| <b>max_rows</b>   | integer | No       | Max rows to return.                 |

## Example Request

```
curl -u 'admin@system:admin' https://my_sql_node.com/v1/execute/system \
-d '{"statement": "SELECT * FROM sys.dummy2", "format": "collection"}'
```

## GET /v1/execute

**Description**

Executes a SQL statement using URL query parameters. Use only for simple, read-only queries.

**URL**

https://{sql\_node}/v1/execute

**Method**

GET

## Query Parameters

| Name              | Type   | Required | Description                         |
|-------------------|--------|----------|-------------------------------------|
| <b>statement</b>  | string | Yes      | URL-encoded SQL.                    |
| <b>database</b>   | string | No       | Target DB.                          |
| <b>format</b>     | string | No       | <b>array</b> or <b>collection</b> . |
| <b>fetch_size</b> | string | No       | Rows per chunk (streaming).         |

## Header Parameters

Same as POST /v1/execute.

## Example Request

```
curl -u 'admin@system:admin' \
'https://my_sql_node.com/v1/execute?statement=SELECT%20*%20FROM%20sys.dummy2&database=system&format=coll'
```

## GET /v1/info

**Description**

Returns version and system information for the Ocint System and HTTP Query API server.

**URL**

https://{sql\_node}/v1/info

**Method**

GET

## Example Request

```
curl -u 'admin@system:admin' 'https://my_sql_node.com/v1/info'
```

POST /v1/login

### Description

Authenticates a user and returns a bearer token and session cookie.

### URL

https://{sql\_node}/v1/login

### Method

POST

Body Parameters

| Name     | Type   | Required | Description                 |
|----------|--------|----------|-----------------------------|
| username | string | Yes      | Username.                   |
| password | string | Yes      | Password.                   |
| database | string | No       | Defaults to <b>system</b> . |

Example Request

```
curl -X POST https://my_sql_node.com/v1/login \
 -H "Content-Type: application/json" \
 -d '{
 "username": "admin",
 "password": "admin",
 "database": "system"
 }'
```

POST /v1/logout

### Description

Terminates a cookie-based session. Bearer tokens remain valid until expiry.

### URL

https://{sql\_node}/v1/logout

### Method

POST

Body Parameters

| Name     | Type   | Required | Description                 |
|----------|--------|----------|-----------------------------|
| database | string | No       | Defaults to <b>system</b> . |

Example Request

```
curl -X POST https://my_sql_node.com/v1/logout \
 -d '{
 "database": "retail_analytics"
 }'
```

POST /v1/sso\_authentication

#### Description

Initiates OpenID Connect auth by redirecting the user to the identity provider.

#### URL

https://{sql\_node}/v1/sso\_authentication

#### Method

POST

Body Parameters

| Name              | Type    | Required | Description                                                         |
|-------------------|---------|----------|---------------------------------------------------------------------|
| database          | string  | No       | DB to connect to.                                                   |
| sso_callback_path | string  | No       | Path for redirect after auth.                                       |
| do_redirect       | boolean | No       | <b>true</b> = 302 redirect.<br><b>false</b> = returns redirect URL. |

GET /v1/callback

#### Description

Completes the OpenID authentication flow. Exchanges **code** for a token.

#### URL

https://{sql\_node}/v1/callback

#### Method

GET

Query Parameters

| Name         | Type   | Required | Description                  |
|--------------|--------|----------|------------------------------|
| <b>code</b>  | string | Yes      | Auth code from OpenID.       |
| <b>state</b> | string | Yes      | Must match original request. |

POST /v1/sso\_token

#### Description

Exchange an OpenID **id\_token** or **access\_token** for an Ociant token.

#### URL

https://{sql\_node}/v1/sso\_token

#### Method

POST

Body Parameters

| Name         | Type   | Required | Description                      |
|--------------|--------|----------|----------------------------------|
| database     | string | Yes      | Target DB.                       |
| id_token     | string | Optional | Token from OpenID.               |
| access_token | string | Optional | Alternative to <b>id_token</b> . |

---

POST /v1/sso\_device\_grant

**Description**

Generates a user code and verification URL for devices with no browser.

**URL**

`https://{sql_node}/v1/sso_device_grant`

**Method**

POST

Body Parameters

| Name            | Type   | Required | Description                            |
|-----------------|--------|----------|----------------------------------------|
| <b>database</b> | string | No       | Target DB. Defaults to <b>system</b> . |

---

POST /v1/sso\_device\_grant\_verify

**Description**

Verifies the device code authorization and returns an access token.

**URL**

`https://{sql_node}/v1/sso_device_grant_verify`

**Method**

POST

Body Parameters

| Name             | Type    | Required | Description                                   |
|------------------|---------|----------|-----------------------------------------------|
| <b>user_code</b> | string  | Yes      | Code from previous grant request.             |
| <b>database</b>  | string  | No       | DB to connect to.                             |
| <b>timeout</b>   | integer | No       | Wait time in seconds (default: 30, max: 600). |

**HTTP API Query Examples**

These examples demonstrate how you can sequence API endpoints for common use cases.

**Simple Queries**

This example query format is appropriate for simple read-only queries. The example provides user credentials `admin@system:admin` in the request. The `https://{sql_node}/v1/execute/system` API call specifies the **execute** endpoint with the SQL node address `sql_node` and the **system** database. In this example, the SQL statement is `SELECT * FROM sys.dummy2;`. The `collection` format option returns the result in JSON format.

```
curl -u 'admin@system:admin' https://{sql_node}/v1/execute/system \
 -d '{"statement":"SELECT * FROM sys.dummy2;", "format": "collection"}'
```

**Large Result Set Queries**

Specify the content type as JSON data with the **Content-Type** header option. When working with large amounts of data, enable streaming using the `application/stream+json` value with the **Accept** header option. The request also compresses the response using `gzip` compression with the **Accept-Encoding** header option. In this example, the

SQL statement is `SELECT * FROM large_table;`. The example specifies the `fetch_size` value to set the number of rows each chunk of streaming data contains to 1000.

```
curl -u 'admin@system:admin' https://sql_node/v1/execute/system \
 -H "Content-Type: application/json" \
 -H "Accept: application/stream+json" \
 -H "Accept-Encoding: gzip" \
 -d '{
 "statement": "SELECT * FROM large_table;",
 "fetch_size": 1000
 }'
```

### Token Authentication Script

This example uses a bash script to retrieve a token for the subsequent `curl` query. Use the `login` endpoint to retrieve the token. Specify the username, password, and database name. Then, use the `execute` endpoint to execute the `SELECT * FROM large_table;` SQL statement. Specify the token with the `Authorization` header option. The `gunzip` command decompresses the gzip-compressed API response so it is readable in JSON format.

```
Retrieve the token
TOKEN=$(curl -s -X POST https://sql_node/v1/login \
 -H "Content-Type: application/json" \
 -d '{
 "username": "admin@system",
 "password": "admin",
 "database": "system"
 }' | jq -r '.access_token')

Then use the token for a query with streaming and compression
curl -X POST https://sql_node/v1/execute \
 -H "Authorization: Bearer $TOKEN" \
 -H "Content-Type: application/json" \
 -H "Accept: application/stream+json" \
 -H "Accept-Encoding: gzip" \
 -d '{
 "statement": "SELECT * FROM large_table;",
 "database": "system",
 "fetch_size": 1000
 }' | gunzip
```

### Save Results to a File

This example saves the query results in a `tables.json` file. The example uses the `SELECT * FROM sys.tables;` SQL statement.

```
curl -u 'admin@system:admin' https://sql_node/v1/execute/system \
 -H "Content-Type: application/json" \
 -d '{
 "statement": "SELECT * FROM sys.tables;",
 "format": "collection"
 }' | jq '.data' > tables.json
```

### Use a Wrapper Script

This example specifies the connection parameters in the `ociant-query.sh` file. To run the file at the command line, you must also include an argument for executing the SQL statement (for example: `./ociant-query.sh "SELECT * FROM my_table;"`).



```
#!/bin/bash
ocient-query.sh

HOST="sql-node.com"
USER="admin"
PASSWORD="admin"
DATABASE="system"

QUERY="$1"

curl -s -u "${USER}@${DATABASE}:${PASSWORD}" "https://${HOST}/v1/execute/${DATABASE}" \
 -H "Content-Type: application/json" \
 -d "{
 \"statement\": \"${QUERY}\",
 \"format\": \"collection\"
 }" | jq '.data'
```

## Related Links

Swagger

[General SQL Syntax](#)

## 4.4 Understanding Data Types

Ocient® supports many common data types such as INT, SHORT, DATE, and TIMESTAMP as well as complex types like ARRAY and TUPLE and specialized data types like POINT and LINESTRING used for geospatial analysis. For detailed information about supported data types, refer to the [Data Types](#) page.

All of the following data types can be used as types for columns in tables and for queries. When used in queries, [Scalar Data Conversion Functions](#) can be used to manipulate data types and various functions and aggregates can be run depending on the data type. For example, numeric types can be summed, averaged or be processed with other statistical functions, and character types can be trimmed, manipulated, or aggregated into a single string.

Basic data types such as INT, BOOLEAN, and VARCHAR behave similarly to other databases. Complex types such as ARRAYS and TUPLES can be queried and inspected like any other column and also have special operators used to interrogate the sets of data.

### Numeric and Boolean Data Types

Numeric and Boolean types are used to represent data with different precision and scale. Integer values are supported from a 1-byte signed int up to an 8-byte signed int. All integers are signed in Ocient. Floating point values are possible as either single or double precision, and exact decimal precision is available with the Decimal type. The Boolean type is used to represent true or false in the database.

All numeric types are fixed-length data types.

- BIGINT — 8-byte signed integer
- INT — 4-byte signed integer
- SMALLINT — 2-byte signed integer
- TINYINT — 1-byte signed integer (alias: BYTE)
- FLOAT — 4-byte single precision floating-point number (alias: REAL or SINGLE PRECISION)
- DOUBLE — 8-byte double precision floating-point number (alias: DOUBLE PRECISION)
- DECIMAL(P,S) — Exact numerical decimal value with precision P and Scale S
- BOOLEAN — 1-byte logical Boolean representing true or false

Learn more about working with numeric data types in the following sections:

- [Math Functions and Operators](#)

- [Scalar Data Conversion Functions](#)
- [Aggregate Functions](#)
- [Formatting Functions](#)

## Character and Binary Types

Character Types represent text and character data. Both fixed-length and variable-length options are available. Fixed-length provides some benefits to the database storage and query engine when applicable, but indexes are available that target variable-length character columns. Binary types represent byte array data.

- CHAR(N) — Variable-length character string with length N (alias: CHARACTER(N))
- VARCHAR(N) — Variable-length character string with a maximum length of N
- BINARY(N) — Fixed-length binary array with length N (alias: HASH(N))
- VARBINARY(N) — Variable-length binary array with maximum length N (alias: HASH(N))

Learn more about working with character and binary data types in the following sections:

- [Character and Binary Functions](#)
- [Scalar Data Conversion Functions](#)
- [Sorted Aggregate Functions](#)

## Date and Time Types

Date and Time types are used to represent time in Ocint. Timestamps take on special capabilities when defined as a TimeKey® on a table. This is described in more detail in [TimeKeys and Clustering Keys](#). Ocint provides many functions to operate on time series data.

- DATE — 4-byte Calendar date representing year, month, day
- TIMESTAMP — 8-byte date and time with nanosecond precision in UTC time zone
- TIME — 8-byte time of day in nanoseconds

Learn more about working with date and time data types in the following sections:

- [Date and Time Functions](#)
- [Window Aggregate Functions](#)
- [Formatting Functions](#)

## Networking and Unique Identifier Types

Ocint provides data types for processing of IP addresses and UUIDs that allow schemas to leverage improved searching and analysis of both types as well as easier loading of structured data into these types. The IP type is designed for IP V6 and can handle IP V4 addresses as well.

- IPV4 — 4-byte Internet Protocol V4 address
- IP — 16-byte Internet Protocol V6 address
- UUID — 16-byte universally unique identifier (e.g., *01234567-89ab-cdef-1357-0123456789ab*)

## Geospatial Data Types

Geospatial analysis is a rich capability in Ocint enabled by special data types and functions. The foundation rests on POINT, LINESTRING, and POLYGON types that represent respective points and shapes on the Earth. From these and composites of these types, many proximity, intersection, and computational geospatial analytics can be performed at an incredible scale.

- POINT — Geometric point in 2 dimensions.
- LINESTRING — Geometric type composed of N POINTS.
- POLYGON — Geometric type composed of N closed LINESTRINGS.

Learn more about working with geospatial types in the [Geospatial Functions](#) section.

## Complex Types

Arrays and Tuples are SQL container types used to store other SQL types. These allow for complex schemas to be represented as a column in a table instead of separate, normalized tables. In some situations, using Arrays, Tuples, or Arrays of Tuples can lead to significant performance benefits. Use the Matrix type to store one or two-dimensional arrays of fixed-size data types for matrix computations.

- **ARRAY** — An array of any supported type other than ARRAY.
- **MATRIX** — A one or two-dimensional mathematical matrix used for matrix calculations.
- **TUPLE** — A tuple of elements of different types. Tuple is similar to a structure in other programming languages. You can include Tuples in arrays.

Learn more about complex data types in the following sections:

- [Array Functions and Operators](#)
- [Tuple Functions and Operators](#)
- [Matrix Functions and Operators](#)
- [Sorted Aggregate Functions](#)

## Related Links

[General SQL Syntax](#)

[SQL Reference](#)

### 4.4.1 Array, Tuple, and Matrix Overview

Arrays, tuples, and matrix data types are complex data types in the Ocient® System.

#### Arrays

The array data type is a one-dimensional list of values with the same data type. This type enables the support of a one-to-many relationship without creating another table. You create secondary indexes on arrays, which index the elements of the array. For details about secondary indexes, see [Secondary Indexes](#).

Key features:

- Variable size.
- Can contain many elements.
- Indexing into the array starts at 1.
- Invalid access of the array returns NULL.

For semi-structured data in JSON format, you can express the JSON multi-value column in an array data type. Traditionally in the normalized relational data model, the database represents such data types as a separate table. Each row stores the value of the primary key of the record in the parent table. Therefore, the entity relationship between the tables is always one-to-many, which can result in billions of rows. The array data type eliminates the child tables, which helps reduce the storage footprint and avoids the additional JOIN operations.

#### Examples

Create a one-dimensional integer array [1, 2, 3] using the `sys.dummy` virtual table.

```
SELECT INT[] (1,2,3)
FROM sys.dummy1;
```

*Output*

```
array_int(int((1)), int((2)), int((3)))
```

```

```

```
[1,2,3]
```

Fetches 1 row

Create a two-dimensional integer array `[[1, 2, 3], [4, 5, 6]]`.

```
SELECT INT[] [] (INT[] (1,2,3),INT[] (4,5,6))
FROM sys.dummy1;
```

*Output*

```
array_array(int)(array_int(int((1)), int((2)), int((3))), array_int(int((4)), int((5)), int((6))))

[[1,2,3],[4,5,6]]

Fetched 1 row
```

You can get the same result by casting the array and integers explicitly.

```
SELECT INT[] [] (ARRAY[INT(1),INT(2),INT(3)],ARRAY[INT(4),INT(5),INT(6)])
FROM sys.dummy1;
```

For more examples, see [Array Functions and Operators](#).

## Tuples

The tuple data type is a row. This type is a fixed-sized collection of heterogeneous values. Tuples support more complex recursive computational functions. The memory structure of the tuple is identical to the memory structure of the array type. The only difference is that each entry in a tuple can be any type, so the Ocient System can simultaneously store both fixed and variable-sized values in a tuple.

### Examples

Create a tuple with fixed-size types. In this case, use three integers.

```
SELECT TUPLE(1,2,3)
FROM sys.dummy1;
```

*Output*

```
tuple((1), (2), (3))

<<1, 2, 3>>

Fetched 1 row
```

You can get the same result by using the tuple type constructor.

```
SELECT TUPLE<<INT,INT,INT>>(1,2,3)
FROM sys.dummy1;
```

Create a tuple with variable-size types. In this case, use a one-dimensional and two-dimensional array.

```
SELECT TUPLE(ARRAY[INT(1),INT(2),INT(3)],
 INT[] [] (ARRAY[INT(1),INT(2),INT(3)],ARRAY[INT(4),INT(5),INT(6)]))
FROM sys.dummy1;
```

*Output*

```
tuple(array_int(int((1)), int((2)), int((3))), array_array(int)(array_int(int((1)), int((2)), int((3))), a

<<[1,2,3], [[1,2,3],[4,5,6]]>>

Fetched 1 row
```

Create a tuple with different types. In this case, an integer, string, and array of integers.

```
SELECT TUPLE<<INT,VARCHAR,INT[]>>(1, 'test_string', INT[] (2,4,6))
FROM sys.dummy1;
```

*Output*

```
tuple(int((1)), ('test_string'), array_int(int((2)), int((4)), int((6))))
```

```

<<1, test_string, [2,4,6]>>
```

Fetches 1 row

For more examples, see [Tuple Functions and Operators](#).

## Matrixes

A matrix is a fixed-size two-dimensional array of `DOUBLE` values. A matrix can be a row vector (1xN matrix) or column vector (Nx1 matrix).

Matrixes support machine learning model calculations. For details, see [Machine Learning in Ocint](#).

## Examples

Create a 1x4 matrix with numbers 1 through 4 using the `make_matrix` function that expects values in row-major order. The first two arguments indicate the dimensions of the matrix, and the remaining arguments specify the values for the matrix.

```
SELECT make_matrix_1x4(1,4,1,2,3,4)
FROM sys.dummy1;
```

*Output*

```
make_matrix_1x4((1), (4), (1), (2), (3), (4))
```

```

[[1.0,2.0,3.0,4.0]]
```

Fetches 1 row

Create a 2x3 matrix with numbers 1 through 6.

```
SELECT make_matrix_2x3(2,3,1,2,3,4,5,6)
FROM sys.dummy1;
```

*Output*

```
make_matrix_2x3((2), (3), (1), (2), (3), (4), (5), (6))
```

```

[[1.0,2.0,3.0],[4.0,5.0,6.0]]
```

Fetches 1 row

Create a row vector with four doubles.

```
SELECT _R{1,2,3,4}
FROM sys.dummy1;
```

*Output*

```
_r{1,2,3,4}
```

```

[[1.0,2.0,3.0,4.0]]
```

Fetches 1 row

For more examples, see [Matrix Functions and Operators](#).

### Related Links

[Array Functions and Operators](#)

[Tuple Functions and Operators](#)

[Matrix Functions and Operators](#)

[Generate Tables Using sys.dummy](#)

## 4.5 Functions Overview

Ocient® includes many functions for statistics, mathematics, string manipulation, and working with complex types like Arrays and geospatial data. The following types of functions are supported:

### SQL Functions

#### Math and Arithmetic Functions

Perform mathematical operations on data in Ocient including basic operators on data types and advanced statistical functions.

- [Math Functions and Operators](#)
- [Aggregate Functions](#)

#### Character and Binary Functions

Manipulate character strings and binary data types and use character operators to query character data.

- [Character and Binary Functions](#)

#### Conversion and Formatting Functions

Convert data types with scalar conversions and format numbers and times in character outputs.

- [Scalar Data Conversion Functions](#)
- [Formatting Functions](#)

#### Date and Time Functions

Perform date operations in Ocient including manipulation and comparison of date and time types. Date and Time functions allow users to add, subtract, and truncate date time values and intervals.

- [Date and Time Functions](#)
- [Time Zone Functions](#)

#### Aggregates

Aggregates include basic aggregates and sorted aggregates that combine many rows into a single result as well as window aggregates that combine information from multiple rows within the window into each row in the result set.

- [Aggregate Functions](#)
- [Sorted Aggregate Functions](#)
- [Window Aggregate Functions](#)
- [HyperLogLog Functions](#)

**Array, Tuple, and Matrix Functions**

Special functions and operators to work with container types. These include operators to check set logic, slice, compare, and manipulate arrays, tuples, and matrixes.

- [Array Functions and Operators](#)
- [Tuple Functions and Operators](#)
- [Matrix Functions and Operators](#)

**Geospatial Functions**

Functions and Operators to manipulate and compare points, linestrings, and polygons.

- [Geospatial Functions](#)

**Network Functions**

Functions for using network data types, including IP and IPV4.

- [Network Type Functions](#)

**Miscellaneous Functions**

A set of functions such as `CASE`, `CURRENT_USER`, and `NULL_IF` that provide additional capabilities in Ocint.

- [Other Functions and Expressions](#)

**Related Links**

[SQL Reference](#)

[General SQL Syntax](#)

---

## 5 Analysis in Ocient

### 5.1 Text Analysis in Database Queries

The Ocient Hyperscale Data Warehouse™ enables search and analysis of textual data. You can run queries efficiently using indexes in the Ocient Hyperscale Data Warehouse to perform text analysis. The database supports different types of indexes. The N-gram index is a type of secondary index that enables text analysis.

#### Cluster Key Indexes

Define cluster key (CK) indexes when you create the table.

| Index Type                       | Number of Columns         | Column Types                                            | Filters              |
|----------------------------------|---------------------------|---------------------------------------------------------|----------------------|
| Primary CK index (always exists) | Multiple                  | Fixed-length or GDC columns, fixed-length tuple columns | Equality-like, range |
| Additional CK indexes            | Subset of CK in any order |                                                         |                      |

#### Secondary Indexes

You can create or drop secondary indexes at any time.

| Index Type       | Number of Columns | Column Types                                          | Filters              |
|------------------|-------------------|-------------------------------------------------------|----------------------|
| Inverted indexes | Single            | Fixed-length or GDC column, array, or tuple component | Equality-like, range |
| Hash indexes     |                   | Variable-length column, array, or tuple component     | Equality-like        |
| N-gram indexes   |                   | VARCHAR column, array, or tuple component             | LIKE, equality-like  |

N-gram indexes support text analysis by providing an efficient search of textual data. This type of index works by tokenizing the pattern in the string, and then pruning and transforming the tokens. You can create N-gram indexes at any time. For details about creating an N-gram index, see [Database, Tables, Views, and Indexes](#).

There are other ways to perform text analysis. For details, see the syntax for [General SQL Syntax](#) and [General SQL Syntax](#).

#### Related Links

[General SQL Syntax](#)

[Secondary Indexes](#)

### 5.2 Machine Learning in Ocient

The Ocient Hyperscale Data Warehouse™ has OcientML™ functionality that enables machine learning training and model execution within the database. Whether you are training a model or using machine learning functionality to do scoring or prediction, OcientML enables machine learning in the SQL syntax directly.

You can use the `CREATE MLMODEL` SQL statement to create models. This step is for model training and is referred to as model creation. Similarly, you can use the model on new input data to generate new predictions by executing the scalar function that has the same name as the model. This action is typically referred to as executing the model.

Machine learning in Ocient® depends upon linear algebra functionality that is built into the Ocient System. OcientML provides machine learning capabilities that you can invoke in SQL statements with some application logic.



## Linear Algebra in Ocient

Matrixes are a first-class data type in the Ocient System. Create a matrix using a simple SQL SELECT statement.

```
SELECT {{1, 2}, {3, 4}};
{{1,2},{3,4}}
```

---

```
[[1.0, 2.0], [3.0, 4.0]]
```

Fetches 1 row

```
SELECT {{c1*1, c1*2}, {c1*3, c1*4}} FROM sys.dummy2;
make_matrix_2x2((2), (2), ((1))*(c1), ((2))*(c1), ((3))*(c1), ((4))*(c1))
```

---

```
[[1.0, 2.0], [3.0, 4.0]]
[[2.0, 4.0], [6.0, 8.0]]
```

Fetches 2 rows

You can also use shorthand notations to create row `_r` or column vectors `_c`. For example, `_r{1,2,3}` creates a row vector with values 1.0, 2.0, and 3.0.

```
SELECT _r{1,2,3};
_r{1,2,3}
```

---

```
[[1.0, 2.0, 3.0]]
```

Fetches 1 row

```
SELECT _c{1,2,3};
_c{1,2,3}
```

---

```
[[1.0], [2.0], [3.0]]
```

Fetches 1 row

You can execute functions using the values in the vectors. This query does some matrix arithmetic, finds the inverse matrix, and then returns the two eigenvalues and eigenvectors of the inverse.

```
SELECT EIGEN(INVERSE(2 * {{1,2},{3,4}} + {{5,6},{7,8}} / 2));
EIGEN(INVERSE(((2))*({{1,2},{3,4}})) + ((({5,6},{7,8}})/(2))))
```

---

```
[<<-1.3780529228406495, [[0.8013353799887076, -0.5982153531783962]]>>, <<0.05805292284064968, [[0.48196559
```

Fetches 1 row

## Machine Learning Models

Ocient supports regression, classification, and clustering models. Also, Ocient supports models for dimensionality reduction. For guides on using the different OcientML™ models, see:

- [Regression Analysis](#)
- [Classification Analysis](#)
- [Clustering Analysis and Dimensionality Reduction](#)

## Related Links

[Machine Learning Model Functions](#)

[Regression Models](#)

### 5.2.1 Regression Analysis

OciantML™ enables regression analysis, which is the process of finding a best-fit function for a data set, in the Ociant® System.

#### Simple Linear Regression

The simplest case is simple linear regression. Simple linear regression finds a best-fit linear relationship between one variable and another.

Suppose you have this data with a bit of noise added. Use the `CREATE TABLE AS SELECT` SQL statement to create the `mldemo.slr` table. Generate 100 rows of random data.

```
CREATE TABLE mldemo.slr AS
(SELECT c1 AS x, c1 * 3.14 - 1.41 + (RAND() - 0.5) * 0.01 AS y
FROM sys.dummy100);
Modified 100 rows
```

Create a simple linear regression model using this data.

```
CREATE MLMODEL my_first_model TYPE SIMPLE LINEAR REGRESSION
ON (SELECT x, y FROM mldemo.slr);
```

Explore the created model by viewing the data in the `machine_learning_models` and `simple_linear_regression_models` system catalog tables.

```
SELECT name, on_select, slope, y_intercept
FROM sys.machine_learning_models a, sys.simple_linear_regression_models b
WHERE a.id = b.machine_learning_model_id AND name = 'my_first_model';
```

| name           | on_select                   | slope            |
|----------------|-----------------------------|------------------|
| my_first_model | select x, y from mldemo.slr | 3.13999545011463 |

Fetches 1 row

You can see the calculated slope and the y-intercept from the model, and see that these values are very close to the values used to generate the data. While the input to the model creation, in this case, is very simple, the SQL statement in the `ON` clause of the model creation can be arbitrarily complex as long as it meets the requirements of the input for the specific model type being created.

Execute the created model by executing the SQL scalar function with the same name as the model you created. The input arguments of the function depend on the specific model type. For regression, the values are the independent variables, and then the function returns the dependent variables.

```
SELECT x, y AS actual, my_first_model(x) AS predicted FROM mldemo.slr LIMIT 10;
```

| x  | actual             | predicted          |
|----|--------------------|--------------------|
| 51 | 158.730137908257   | 158.72952690917217 |
| 52 | 161.87052128169267 | 161.8695223592868  |
| 53 | 165.0119331158625  | 165.00951780940144 |
| 54 | 168.1478263920115  | 168.14951325951608 |
| 55 | 171.28758882396247 | 171.28950870963072 |
| 56 | 174.43476312079238 | 174.42950415974536 |
| 57 | 177.57385971080322 | 177.56949960986    |
| 58 | 180.70933136639127 | 180.70949505997464 |
| 59 | 183.84856961481378 | 183.84949051008925 |
| 60 | 186.99044288540566 | 186.9894859602039  |

Fetches 10 rows

The model makes predictions that are very close to the actual training data. Then, you can use this data to generate predictions for data that is unknown.

### Machine Learning Model Options

You can provide options for most model types. Provide options by pairing the string for an option to its corresponding value. For example, simple linear regression models have a `metrics` option that you can use to ask the database to calculate some quality of fit metrics. Use the same model and enable metrics collection, which is off by default.

```
DROP MLMODEL my_first_model;
Modified 0 rows
```

```
CREATE MLMODEL my_first_model TYPE SIMPLE LINEAR REGRESSION
ON (SELECT x, y FROM mldemo.slr) options('metrics' -> 'true');
Modified 0 rows
```

Examine the `machine_learning_models` and `simple_linear_regression_models` system catalog tables for more fields.

```
SELECT name, on_select, slope, y_intercept, coefficient_of_determination,
adjusted_r2, rmse FROM sys.machine_learning_models a, sys.simple_linear_regression_models b
WHERE a.id = b.machine_learning_model_id AND name = 'my_first_model';
```

| name           | on_select                   | slope            |
|----------------|-----------------------------|------------------|
| my_first_model | select x, y from mldemo.slr | 3.13999545011463 |

Fetches 1 row

The database also calculates the coefficient of determination (frequently called  $r^2$ ), the adjusted  $r^2$  value, and the RMSE (root mean squared error). All of these values indicate that this model has an extremely good fit.

In addition, you can force a specific y-intercept such that the value of  $f(0)$  is fixed. Use the `yIntercept` option. Create a model on the same data but force the y-intercept to be zero. The metrics show that the model is not as good a fit as when the intercept is free to vary.

```
CREATE MLMODEL force_zero TYPE SIMPLE LINEAR REGRESSION
ON (SELECT x, y FROM mldemo.slr) options('metrics' -> 'true', 'yIntercept' -> '0');
Modified 0 rows
```

```
SELECT name, on_select, slope, y_intercept, coefficient_of_determination,
adjusted_r2, rmse FROM sys.machine_learning_models a, sys.simple_linear_regression_models b
WHERE a.id = b.machine_learning_model_id AND name = 'force_zero';
```

| name       | on_select                   | slope            |
|------------|-----------------------------|------------------|
| force_zero | select x, y from mldemo.slr | 3.11894707628367 |

Fetches 1 row

### Multiple Linear Regression

In multiple linear regression, the model is still predicting an output value. But instead of handling one input variable, the model can handle an arbitrary number of input variables. Multiple linear regression finds the best-fit model of the form:

$$y = a_1x_1 + a_2x_2 + a_3x_3 + \dots + a_0$$

Simple linear regression and multiple linear regression (as well as polynomial regression and linear combination regression) all use a least-squares loss function. Some of the more advanced models allow you to define best fit yourself if the default definition is not satisfactory. Least-squares best fit means that the model minimizes the sum (over all training data rows) of the square of the model error.

Create a training data set with some noise and show that the model can reconstruct the data. Multiple linear regression models also support a metrics option (the metrics are the same as simple linear regression).

```
CREATE TABLE mldemo.mlr AS (SELECT a.c1 AS x1, b.c1 AS x2, 1 + 2*a.c1 + 3*b.c1 AS y
FROM sys.dummy10 a, sys.dummy10 b);
Modified 100 rows

CREATE MLMODEL mlr_model TYPE MULTIPLE LINEAR REGRESSION ON (SELECT * FROM mldemo.mlr)
options('metrics' -> 'true');
Modified 0 rows

SELECT x1, x2, y AS actual, mlr_model(x1, x2) AS predicted FROM mldemo.mlr LIMIT 10;
x1 x2 actual predicted

6 1 16 15.99999999999975
6 2 19 18.99999999999975
6 3 22 21.99999999999975
6 4 25 24.99999999999975
6 5 28 27.99999999999975
6 6 31 30.99999999999975
6 7 34 33.9999999999997
6 8 37 36.9999999999997
6 9 40 39.9999999999997
6 10 43 42.9999999999997

Fetched 10 rows
```

## Polynomial Regression

With polynomial regression, the database finds a least-squares best-fit polynomial of whatever degree you specify. Same as multiple linear regression, you determine the number of independent variables.

The training data has a  $1/(x1 * x2)$  term that the model does not match, but that term quickly tends towards zero, so the model should provide a good fit with a quadratic polynomial of  $x1$  and  $x2$ . Some of the coefficients match the data but some do not because the model is compensating for not being able to fit the form of the data exactly. However, the metrics indicate that this model is still a great fit.

```
CREATE TABLE mldemo.pr AS
(SELECT x1, x2, 1.0 / (x1 * x2) - 2*x1 - 0.5*x1*x2 + 2.3*x1^2 + 1.1*x2^2 AS y
FROM (SELECT a.c1 AS x1, b.c1 AS x2 FROM sys.dummy10 a, sys.dummy10 b));
Modified 100 rows

CREATE MLMODEL pr1 TYPE POLYNOMIAL REGRESSION
ON (SELECT * FROM mldemo.pr) options('order' -> '2', 'metrics' -> 'true');
Modified 0 rows

SELECT name, on_select, num_arguments, coefficient_of_determination,
adjusted_r2, rmse, y_intercept, coefficients
FROM sys.machine_learning_models a, sys.polynomial_regression_models b
WHERE a.id = b.machine_learning_model_id AND name = 'pr1';
name on_select num_arguments coefficient_of_determination

pr1 select * from mldemo.pr 2 0.9999999999999999

Fetched 1 row
```

Polynomial regression models have many available options, including everything available for multiple linear regression. The `negativePowers` option enables you to fit Laurent polynomials. If you set this option to `true`, the model

includes independent variables raised to negative powers. Sums of such terms are named Laurent polynomials. The model generates all possible terms such that the sum of the absolute value of the power of each term in each product is less than or equal to the order.

For example, with two independent variables and the order set to 2, the model is:

$$y = a_1x_1^2 + a_2x_1^{-2} + a_3x_2^2 + a_4x_2^{-2} + a_5x_1x_2 + a_6x_1^{-1}x_2 + a_7x_1x_2^{-1} + a_8x_1^{-1}x_2^{-1} + a_9x_1 + a_{10}x_1^{-1} + a_{11}x_2 + a_{12}x_2^{-1} + b$$

```
CREATE TABLE mldemo.pr_negativepowers AS
(SELECT x1, x2, 1.0 + 12*x2^-1 - 11*x2 + 10*x1^-1 + 9*x1 + 8*(x1 - 1)*(x2 - 1) +
7*x1*x2^-1 + 6*x1^-1*x2 + 5*x1*x2 + 4*x2^-2 + 3*x2^2 + 2*x1^-2 + x1^2 AS y
FROM (SELECT a.c1 AS x1, b.c1 AS x2 FROM sys.dummy10 a, sys.dummy10 b));
Modified 100 rows
```

```
CREATE MLMODEL pr2 TYPE POLYNOMIAL REGRESSION
ON (SELECT * FROM mldemo.pr_negativepowers)
options('order' -> '2', 'metrics' -> 'true', 'negativePowers' -> 'true');
Modified 0 rows
```

```
SELECT name, on_select, num_arguments, coefficient_of_determination,
adjusted_r2, rmse, y_intercept, coefficients
FROM sys.machine_learning_models a, sys.polynomial_regression_models b
WHERE a.id = b.machine_learning_model_id AND name = 'pr2';
```

| name | on_select               | num_arguments | coefficient_of_determination |
|------|-------------------------|---------------|------------------------------|
| pr2  | select * from mldemo.pr | 2             | 1.0                          |

You can see that now the model almost perfectly recovers the coefficients, with the rest being nearly zero.

Execute the pr2 model.

```
SELECT x1, x2, y AS actual, pr2(x1, x2) AS predicted FROM mldemo.pr LIMIT 10;
```

| x1  | x2   | actual             | predicted           |
|-----|------|--------------------|---------------------|
| 6.0 | 1.0  | 69.06666666666666  | 69.06666666676095   |
| 6.0 | 2.0  | 69.28333333333333  | 69.283333333346139  |
| 6.0 | 3.0  | 71.75555555555556  | 71.755555555569368  |
| 6.0 | 4.0  | 76.44166666666666  | 76.441666666681195  |
| 6.0 | 5.0  | 83.33333333333333  | 83.333333333348473  |
| 6.0 | 6.0  | 92.42777777777778  | 92.427777777793454  |
| 6.0 | 7.0  | 103.72380952380954 | 103.72380952397093  |
| 6.0 | 8.0  | 117.22083333333333 | 117.220833333349865 |
| 6.0 | 9.0  | 132.9185185185185  | 132.91851851868702  |
| 6.0 | 10.0 | 150.81666666666666 | 150.816666666683765 |

Fetches 10 rows

## Linear Combination Regression

The most complex type of linear regression that Ocint supports is linear combination regression. This regression is a generalization of polynomial regression. Polynomial regression tries to find the best linear combination of polynomial terms ( $x_1$ ,  $x_2$ ,  $x_1^2$ ,  $x_2^2$ ,  $x_1x_2$ , and so on). Linear combination regression finds the best fit linear combination where you can specify all the terms.

Assume this training data. Use the CREATE TABLE AS SELECT SQL statement to create the mldemo.lcr table. Generate 1000 rows of data.

```
CREATE TABLE mldemo.lcr AS
(SELECT x1, x2, x3, 10.0 * sin(x1) + 2.0 * cos(x2) - 3.0 * MOD(INT(x1)*INT(x2)*INT(x3), 4)
AS y FROM (SELECT a.c1 AS x1, b.c1 AS x2, c.c1 AS x3
FROM sys.dummy10 a, sys.dummy10 b, sys.dummy10 c));
Modified 1000 rows
```

Fit this data using a fifth-degree polynomial model.

```
CREATE MLMODEL pr_attempting_lcr TYPE POLYNOMIAL REGRESSION
ON (SELECT * FROM mldemo.lcr) options('order' -> '5', 'metrics' -> 'true');
Modified 0 rows
```

```
SELECT name, on_select, num_arguments, coefficient_of_determination,
adjusted_r2, rmse, y_intercept, coefficients
FROM sys.machine_learning_models a, sys.polynomial_regression_models b
WHERE a.id = b.machine_learning_model_id AND name = 'pr_attempting_lcr';
```

| name              | on_select                | num_arguments | coefficient_of_determination |
|-------------------|--------------------------|---------------|------------------------------|
| pr_attempting_lcr | select * from mldemo.lcr | 3             | 0.0                          |

Fetches 1 row

The coefficient of determination shows that the model is not a bad fit, but the fit is not great due to the structure of the data. However, given that the model is still a linear combination of functions, linear combination regression can recover the fit if you specify the functions to use.

```
CREATE MLMODEL lcr1 TYPE LINEAR COMBINATION REGRESSION
ON (SELECT x1, x2, x3, y FROM mldemo.lcr) options('function1' -> 'sin(x1)',
'function2' -> 'cos(x2)', 'function3' -> 'mod(int(x1)*int(x2)*int(x3),4)',
'metrics' -> 'true');
Modified 0 rows
```

```
SELECT name, on_select, num_arguments, coefficient_of_determination,
adjusted_r2, rmse, y_intercept, coefficients
FROM sys.machine_learning_models a, sys.linear_combination_regression_models b
WHERE a.id = b.machine_learning_model_id AND name = 'lcr1';
```

| name | on_select                            | num_arguments | coefficient_of_determination |
|------|--------------------------------------|---------------|------------------------------|
| lcr1 | select x1, x2, x3, y from mldemo.lcr | 3             | 1.0                          |

Fetches 1 row

You can see that the model finds the coefficients perfectly and has a coefficient of determination of 1.0.

The coefficient of determination represents the amount of variance in the data explained by the model. A coefficient of determination of 1.0 indicates that all the variance is explained and that the model is a perfect fit. A coefficient of determination of 0.0 indicates that none of the variance is explained. A negative coefficient of determination is possible, but you do not find it in linear regression models because these models always find a fit that is at least as good as the constant function (the average of all the training data). That model has a coefficient of determination of 0.0.

## Non-Linear Regression

The non-linear regression model is similar to linear combination regression in that you define the form of the model. However, non-linear regression can be more than a linear combination of terms.

Assume this training data. Use the CREATE TABLE AS SELECT SQL statement to create the mldemo.nlo\_input1 table. Generate 100 rows of data. Then, create a non-linear regression model using five parameters.

```
CREATE TABLE mldemo.nlo_input1 AS (
 SELECT x1,
 x2,
 2.0 * sin(3.0 * x1 - 1.0) - 4.0 + 5.0 * x2 as y
 FROM (
 SELECT a.c1 / 4.0 as x1,
 b.c1 / 4.0 as x2
 FROM sys.dummy10 a,
 sys.dummy10 b
)
);
Modified 100 rows

CREATE mlmodel nlo1 TYPE NONLINEAR REGRESSION ON (
 SELECT x1,
 x2,
 y
 FROM mldemo.nlo_input1
) options(
 'numParameters'-'>'5',
 'function'-'>'a1 * SIN(a2 * x1 + a3) + a4 + a5 * x2',
 'metrics'-'>'true'
);
Modified 0 rows
```

In this case, specify only one function that covers the whole model.

Use the `machine_learning_models` and `nonlinear_regression_models` system catalog tables to see the metrics and the values that the model chose.

```
SELECT name,
 on_select
FROM sys.machine_learning_models a,
 sys.nonlinear_regression_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'nlo1';
```

| name | on_select                               | coefficients     |
|------|-----------------------------------------|------------------|
| nlo1 | select x1, x2, y from mldemo.nlo_input1 | [-2.085086077121 |

Fetches 1 row

If you look at the coefficients, you can see the amplitude (2.0) is negative in the model, but that is offset by subtracting pi from the phase. The model still almost exactly matches how the data is generated, but it just found a slightly different, but equivalent, way of doing it.

Non-linear regression models (and all of the other models that use the same infrastructure) have lots of options that control how the search works. These options affect the length of time it takes for training the model, but they also affect the likelihood of getting stuck in a global minimum. The model chooses default values such that they should be sufficient for a wide range of use cases.

With non-linear regression models, you can specify a loss function other than least-squares. In this example, specify the `abs(f-y)` loss function.

```
CREATE MLMODEL nlo2 type NONLINEAR REGRESSION ON (
 SELECT x1,
 x2,
```

```

 y
 FROM mldemo.nlo_input1
) options(
 'numParameters'-'>'5',
 'function'-'>'a1 * sin(a2 * x1 + a3) + a4 + a5 * x2',
 'lossFunction'-'>'abs(f-y)'
);
Modified 0 rows

```

In this case, find the best fit that minimizes the sum of the absolute value of the model errors instead of the sum of the square of the model errors. The variable `f` represents the value of the model, and the variable `y` represents the actual value in the training data.

The catalog indicates the model is nearly the same as the previous model. In other words, changing the loss function here did not substantially change anything. But there are cases where other loss functions are appropriate and they could substantially change the model.

### Non-Linear Regression with Neural Networks

In all of the non-linear cases, you have to make a good guess for the form of the target function. However, with neural networks, you can leave it up to the network to try to figure out that form. The downside is that aspects of the model are a black box.

This example uses the same example as non-linear regression, but instead of specifying the form of the function, it relies on what the neural net can do with little information. The size of the network layers and the number of layers greatly influence how good of a fit a neural net model has. This example uses a simple network.

```

CREATE MLMODEL nn1 TYPE FEEDFORWARD NETWORK ON (
 SELECT x1,
 x2,
 y
 FROM mldemo.nlo_input1
) options(
 'metrics'-'>'true',
 'hiddenLayers'-'>'2',
 'hiddenLayerSize'-'>'2',
 'outputs'-'>'1',
 'lossFunction'-'>'squared_error'
);
Modified 0 rows

```

In the case of neural network models, the metric used is an average loss because RMSE and the coefficient of determination are generally only appropriate for least squares models, and neural networks might be using other loss functions.

You can see the average loss by querying the `machine_learning_models` and `feedforward_network_models` system catalog tables.

```

SELECT name,
 on_select,
 num_arguments,
 average_loss
FROM sys.machine_learning_models a,
 sys.feedforward_network_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'nn1';

```

| name  | on_select | num_arguments | average_loss |
|-------|-----------|---------------|--------------|
| ----- |           |               |              |



```
nn1 select x1, x2, y from mldemo.nlo_input1 2
```

Fetches 1 row

Compute the coefficient of determination manually.

```
SELECT AVG(y)
FROM mldemo.nlo_input1;

(sum(y))/(null_if(double(count(y)), (0)))

2.8698217335180987
```

Fetches 1 row

```
SELECT 1.0 - SUM((nn1(x1, x2) - y) ^ 2) / SUM((y - 2.8698217335180987) ^ 2) AS r2
FROM mldemo.nlo_input1;

r2

0.9981701280250409
```

Fetches 1 row

This model fits the data very well.

### Vector-Valued Regression

Vector-valued regression is the case where the dependent variable has multiple components. You can use any type of model to learn each component separately, or you can use a feedforward network to learn all components in one model.

The example creates 100 rows of sample data.

```
CREATE TABLE mldemo.vector_valued AS (
 SELECT x1,
 { { 2 * x1 + 3,
 3 * x1 - 2 } } as y
 FROM (
 SELECT c1 AS x1
 FROM sys.dummy100
)
);
Modified 100 rows
```

There is a dependent variable that depends on `x1`, which is vector-valued. Each component can fit a linear model, which can then predict vector-valued results.

```
CREATE MLMODEL vv1 TYPE SIMPLE LINEAR REGRESSION ON (
 SELECT x1,
 y [1,1]
 FROM mldemo.vector_valued
);
Modified 0 rows
```

```
CREATE MLMODEL vv2 TYPE SIMPLE LINEAR REGRESSION ON (
 SELECT x1,
 y [1,2]
```

```
FROM mldemo.vector_valued
);
Modified 0 rows
```

The query verifies that it produces reasonable results.

```
SELECT x1,
 y AS actual,
 { { vv1(x1),
 vv2(x1) } } AS predicted
FROM mldemo.vector_valued
LIMIT 10;
```

| x1 | actual           | predicted        |
|----|------------------|------------------|
| 51 | [[105.0, 151.0]] | [[105.0, 151.0]] |
| 52 | [[107.0, 154.0]] | [[107.0, 154.0]] |
| 53 | [[109.0, 157.0]] | [[109.0, 157.0]] |
| 54 | [[111.0, 160.0]] | [[111.0, 160.0]] |
| 55 | [[113.0, 163.0]] | [[113.0, 163.0]] |
| 56 | [[115.0, 166.0]] | [[115.0, 166.0]] |
| 57 | [[117.0, 169.0]] | [[117.0, 169.0]] |
| 58 | [[119.0, 172.0]] | [[119.0, 172.0]] |
| 59 | [[121.0, 175.0]] | [[121.0, 175.0]] |
| 60 | [[123.0, 178.0]] | [[123.0, 178.0]] |

Fetches 10 rows

This example shows a single model can perform this with the feedforward network model type.

```
CREATE MLMODEL vv3 TYPE FEEDFORWARD NETWORK ON (
 SELECT x1,
 y
 FROM mldemo.vector_valued
) options(
 'hiddenLayers'-'>'2',
 'hiddenLayerSize'-'>'2',
 'outputs'-'>'2',
 'lossFunction'-'>'vector_squared_error'
);
Modified 0 rows
```

outputs, which is set to the size of the vector, and the lossFunction option is set to use vector\_squared\_error.

This example shows how well the model can predict the output.

```
SELECT x1,
 y AS actual,
 vv3(x1) AS predicted
FROM mldemo.vector_valued
LIMIT 10;
```

| x1 | actual           | predicted       |
|----|------------------|-----------------|
| 51 | [[105.0, 151.0]] | [[104.999392266 |
| 52 | [[107.0, 154.0]] | [[106.999402723 |
| 53 | [[109.0, 157.0]] | [[108.999413180 |
| 54 | [[111.0, 160.0]] | [[110.999423637 |

```

55 [[113.0, 163.0]]
56 [[115.0, 166.0]]
57 [[117.0, 169.0]]
58 [[119.0, 172.0]]
59 [[121.0, 175.0]]
60 [[123.0, 178.0]]

```

Fetches 10 rows

## Autoregression

Autoregression is linear regression where the independent variables are lags of the dependent variable. In other words, it is a series where the next value depends linearly on the previous N values.

These examples do not use a new model type. Instead, they use multiple linear regression models.

One of the important things about autoregression is that some variables indicate the order of the data. This variable can be a time variable, but it does not need to be. This variable is necessary to correctly set up the model, but it is not directly used by the model. Instead, the variable helps build the correct lags.

This example builds a model to predict the Fibonacci sequence. This statement creates a table with the first several Fibonacci terms.

```

CREATE TABLE mldemo.fib AS (
 SELECT 0 AS t,
 0 AS f
 UNION ALL
 SELECT 1,
 1
 UNION ALL
 SELECT 2,
 1
 UNION ALL
 SELECT 3,
 2
 UNION ALL
 SELECT 4,
 3
 UNION ALL
 SELECT 5,
 5
 UNION ALL
 SELECT 6,
 8
 UNION ALL
 SELECT 7,
 13
 UNION ALL
 SELECT 8,
 21
 UNION ALL
 SELECT 9,
 34
);

```

Modified 10 rows

This example builds a model where the dependent variable depends on two lags.

```
CREATE MLMODEL fib TYPE MULTIPLE LINEAR REGRESSION ON (
 SELECT LAG(f, 1) OVER(
 ORDER BY t
),
 LAG(f, 2) OVER(
 ORDER BY t
),
 f
FROM mldemo.fib
);
Modified 0 rows
```

When you execute the model, which provides the model with two lags, the model can predict the next value of the sequence that is not in the data, 55.

```
SELECT fib(21, 34);

((((1.0000000000001137))*((21)))+((1.4210854715202004e-14)))+(((1))*((34)))

55.00000000000024

Fetched 1 row
```

### Vector Autoregression

Vector autoregression is just a more complex version of autoregression. This model uses multiple time series, where the next value for each time series depends linearly on the lags of all of the time series.

This statement sets up some data where the next value for two time series depends on the previous two values of both time series.

```
CREATE TABLE mldemo.var_input AS (
 SELECT 3.1415 AS s1,
 1.4142 AS s2,
 0 AS t
 UNION ALL
 SELECT 2.7182 AS s1,
 1.618 AS s2,
 1 AS t
);
INSERT INTO mldemo.var_input
SELECT 0.9 * s1l1 - 0.8 * s2l1 + 0.7 * s1l2 - 0.6 * s2l2 AS s1,
 0.5 * s1l1 - 0.4 * s2l1 + 0.3 * s1l2 - 0.2 * s2l2 AS s2,
 2 AS t
FROM (
 SELECT s1 AS s1l1,
 s2 AS s2l1
 FROM mldemo.var_input
 WHERE t = 1
),
(
 SELECT s1 AS s1l2,
 s2 AS s2l2
 FROM mldemo.var_input
 WHERE t = 0
);
INSERT INTO mldemo.var_input
```

```

SELECT 0.9 * s111 - 0.8 * s211 + 0.7 * s112 - 0.6 * s212 AS s1,
 0.5 * s111 - 0.4 * s211 + 0.3 * s112 - 0.2 * s212 AS s2,
 3 AS t
FROM (
 SELECT s1 AS s111,
 s2 AS s211
 FROM mldemo.var_input
 WHERE t = 2
),
(
 SELECT s1 AS s112,
 s2 AS s212
 FROM mldemo.var_input
 WHERE t = 1
);
INSERT INTO mldemo.var_input
SELECT 0.9 * s111 - 0.8 * s211 + 0.7 * s112 - 0.6 * s212 AS s1,
 0.5 * s111 - 0.4 * s211 + 0.3 * s112 - 0.2 * s212 AS s2,
 4 AS t
FROM (
 SELECT s1 AS s111,
 s2 AS s211
 FROM mldemo.var_input
 WHERE t = 3
),
(
 SELECT s1 AS s112,
 s2 AS s212
 FROM mldemo.var_input
 WHERE t = 2
);
INSERT INTO mldemo.var_input
SELECT 0.9 * s111 - 0.8 * s211 + 0.7 * s112 - 0.6 * s212 AS s1,
 0.5 * s111 - 0.4 * s211 + 0.3 * s112 - 0.2 * s212 AS s2,
 5 AS t
FROM (
 SELECT s1 AS s111,
 s2 AS s211
 FROM mldemo.var_input
 WHERE t = 4
),
(
 SELECT s1 AS s112,
 s2 AS s212
 FROM mldemo.var_input
 WHERE t = 3
);
INSERT INTO mldemo.var_input
SELECT 0.9 * s111 - 0.8 * s211 + 0.7 * s112 - 0.6 * s212 AS s1,
 0.5 * s111 - 0.4 * s211 + 0.3 * s112 - 0.2 * s212 AS s2,
 6 AS t
FROM (
 SELECT s1 AS s111,
 s2 AS s211
 FROM mldemo.var_input

```

```

 WHERE t = 5
),
 (
 SELECT s1 AS s1l2,
 s2 AS s2l2
 FROM mldemo.var_input
 WHERE t = 4
);
INSERT INTO mldemo.var_input
SELECT 0.9 * s1l1 - 0.8 * s2l1 + 0.7 * s1l2 - 0.6 * s2l2 AS s1,
 0.5 * s1l1 - 0.4 * s2l1 + 0.3 * s1l2 - 0.2 * s2l2 AS s2,
 7 AS t
FROM (
 SELECT s1 AS s1l1,
 s2 AS s2l1
 FROM mldemo.var_input
 WHERE t = 6
),
(
 SELECT s1 AS s1l2,
 s2 AS s2l2
 FROM mldemo.var_input
 WHERE t = 5
);
INSERT INTO mldemo.var_input
SELECT 0.9 * s1l1 - 0.8 * s2l1 + 0.7 * s1l2 - 0.6 * s2l2 AS s1,
 0.5 * s1l1 - 0.4 * s2l1 + 0.3 * s1l2 - 0.2 * s2l2 AS s2,
 8 AS t
FROM (
 SELECT s1 AS s1l1,
 s2 AS s2l1
 FROM mldemo.var_input
 WHERE t = 7
),
(
 SELECT s1 AS s1l2,
 s2 AS s2l2
 FROM mldemo.var_input
 WHERE t = 6
);

```

The query inspects the data.

```

SELECT *
FROM mldemo.var_input
ORDER BY t;

```

| s1                 | s2                 | t |
|--------------------|--------------------|---|
| 3.1415             | 1.4142             | 0 |
| 2.7182             | 1.618              | 1 |
| 2.50251            | 1.37151            | 2 |
| 2.086991           | 1.1945109999999999 | 3 |
| 1.8515340999999994 | 1.0421421          | 4 |
| 1.5768541099999998 | 0.8961053099999997 | 5 |
| 1.3730730609999995 | 0.7770167409999997 | 6 |

```
1.1802870530999996 0.6695650050999997 7
1.0215474418099997 0.5788360946099997 8
```

Fetches 9 rows

This example fits the vector autoregression model to the data.

```
CREATE MLMODEL var1 type VECTOR AUTOREGRESSION ON (
 SELECT { { s1,
 s2 } },
 { { s1_lag1,
 s2_lag1 } },
 { { s1_lag2,
 s2_lag2 } }
 FROM (
 SELECT s1,
 s2,
 LAG(s1, 1) OVER(
 ORDER BY t
) as s1_lag1,
 LAG(s1, 2) OVER(
 ORDER BY t
) as s1_lag2,
 LAG(s2, 1) OVER(
 ORDER BY t
) as s2_lag1,
 LAG(s2, 2) OVER(
 ORDER BY t
) AS s2_lag2
 FROM mldemo.var_input
)
) options('numVariables'-'>'2', 'numLags'-'>'2');
Modified 0 rows
```

See the details of the model in the `machine_learning_models` and `vector_autoregression_models` system catalog tables.

```
SELECT NAME,
 coefficients,
 coefficient_of_determination
FROM sys.machine_learning_models a,
 sys.vector_autoregression_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'var1';
```

| name | coefficients                                                                |
|------|-----------------------------------------------------------------------------|
| var1 | <code>[-2.2737367544323206E-12, 0.8999999999432475, -0.799999999901]</code> |

Fetches 1 row

The model recovered the expected coefficients. The coefficient of determination is NULL here because the model did not enable metrics collection in the options.

This example checks that the model matches the data reasonably well.

```
SELECT var1({ { s1_lag1, s2_lag1 } }, { { s1_lag2, s2_lag2 } }) AS predicted,
 { { s1,
```

```

s2 } } AS actual
FROM (
 SELECT s1,
 s2,
 s1_lag1,
 s2_lag1,
 s1_lag2,
 s2_lag2
 FROM (
 SELECT s1,
 s2,
 LAG(s1, 1) OVER(
 ORDER BY t
) AS s1_lag1,
 LAG(s1, 2) OVER(
 ORDER BY t
) AS s1_lag2,
 LAG(s2, 1) OVER(
 ORDER BY t
) AS s2_lag1,
 LAG(s2, 2) OVER(
 ORDER BY t
) AS s2_lag2
 FROM mldemo.var_input
)
);

```

predicted

actual

|                                            |                                            |
|--------------------------------------------|--------------------------------------------|
| NULL                                       | [[3.1415, 1.4142]]                         |
| NULL                                       | [[2.7182, 1.618]]                          |
| [[2.5025099999296385, 1.3715099999786722]] | [[2.50251, 1.37151]]                       |
| [[2.086990999938066, 1.1945109999842818]]  | [[2.086991, 1.1945109999999999]]           |
| [[1.8515340999451868, 1.0421420999862723]] | [[1.8515340999999999, 1.0421420999999999]] |
| [[1.5768541099535214, 0.8961053099881894]] | [[1.5768541099999999, 0.8961053099999999]] |
| [[1.3730730609589523, 0.7770167409897997]] | [[1.3730730609999999, 0.7770167409999999]] |
| [[1.1802870530645944, 0.6695650050911028]] | [[1.1802870530999999, 0.6695650050911028]] |
| [[1.0215474417789623, 0.5788360946022622]] | [[1.0215474418099997, 0.5788360946022622]] |

Fetches 9 rows

The first two values have no predictions because the lags necessary to drive the predictions do not exist.

For details, see [Regression Models](#).

## Related Links

[Machine Learning Model Functions](#)

[Regression Models](#)

### 5.2.2 Classification Analysis

Classification machine learning algorithms are available in OcidentML™. There are many different popular classification algorithms, which work in slightly different ways with different strengths and weaknesses. These walkthroughs and



examples cover some basic rules for using classification algorithms and how you can take advantage of these models in Ociant®.

It can be very challenging to anticipate which classification model is best suited for a data set. Often, it is best to experiment with multiple different models to see what happens.

## Logistic Regression

Despite its name, logistic regression is a classification algorithm, not a regression algorithm. Logistic regression is generally a good first step for trying out a classification model before moving on to more advanced options.

Logistic regression shares many restrictions with the support vector machines model. Both models require that all features are numeric. However, you can still use non-numeric data if you convert it using one-hot encoding. The class labels can be any data type.

Use one-hot encoding to convert non-numeric data into numeric form. For example, if a feature is a color (such as red, green, or blue), then one-hot encoding can make three columns named `is_red`, `is_green`, and `is_blue`. For each row, set one color to the value 1 and the other colors to 0.

This example uses the `lr_input` table, which represents the academic performance of students.

```
CREATE TABLE mldemo.lr_input AS (
 SELECT hours_studied,
 grade_in_prev_course,
 CASE
 WHEN RAND() > (1.0 - effective_score) THEN true
 ELSE false
 END AS got_top_marks
 FROM
 (SELECT hours_studied,
 grade_in_prev_course,
 (2 * grade_in_prev_course + hours_studied) / 10.0 AS effective_score
 FROM
 (SELECT mod(a.c1, 10) AS hours_studied,
 mod(b.c1, 5) AS grade_in_prev_course
 FROM sys.dummy100 a,
 sys.dummy100 b
)
)
);
```

There are two features: the number of hours spent studying `hours_studied` and the grade in the previous course `grade_in_prev_course`. Both input features are numeric. There are also two classes: `true` and `false`. Class values do not need to be Boolean, and they can be any distinct values.

Students who have better grades in the previous course and spend more hours studying are more likely to have top marks in the course. However, this table also includes a randomness factor, which means that rows with the same features can have different classifications. This is normal in real-world data.

This example creates the logistic regression model on the table, including the `metrics` option. This option causes the model to calculate the percentage of samples that are correctly classified. The Ociant System puts this information into the system catalog tables.

```
CREATE MLMODEL lr1 TYPE LOGISTIC REGRESSION ON (
 SELECT *
 FROM mldemo.lr_input
) options('metrics'-'>'true');
```

This query selects the `machine_learning_models` and `logistic_regression_models` system catalog tables.

```
SELECT name,
 on_select,
 num_arguments,
 zero_case,
 one_case,
 correctly_classified
FROM sys.machine_learning_models a,
 sys.logistic_regression_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'lr1';
```

| name | on_select                     | num_arguments | zero_case | one_case | correctly_classified |
|------|-------------------------------|---------------|-----------|----------|----------------------|
| lr1  | select * from mldemo.lr_input | 2             |           |          |                      |

Fetches 1 row

The model captures the two values for the two target classes. The `correctly_classified` column shows the model has almost 84 percent accuracy from working on the training data. This level of accuracy is pretty good, especially for a model using training data with significant randomness.

This model can predict which classification is most likely if you provide the number of hours studied and grades in the previous course.

First, review the average hours spent studying for each classification.

```
SELECT avg(hours_studied)
FROM mldemo.lr_input
WHERE got_top_marks;
```

```
(sum(hours_studied)) / (null_if(double(count(hours_studied)), (0)))
```

```
5.231264244536801
```

Fetches 1 row

```
SELECT avg(hours_studied)
FROM mldemo.lr_input
WHERE NOT got_top_marks;
```

```
(sum(hours_studied)) / (null_if(double(count(hours_studied)), (0)))
```

```
2.3534041715859897
```

Fetches 1 row

As expected, the rows where `got_top_marks` is true have a higher average value for `hours_studied`. The same should be true with `grade_in_prev_course`.

```
SELECT AVG(grade_in_prev_course)
FROM mldemo.lr_input
WHERE got_top_marks;
```

```
(SUM(grade_in_prev_course)) / (NULL_IF(DOUBLE(COUNT(grade_in_prev_course)), (0)))
```

```
2.3535326451266925
```

Fetches 1 row

```
SELECT avg(grade_in_prev_course)
FROM mldemo.lr_input
WHERE NOT got_top_marks;

(sum(grade_in_prev_course)) / (NULL_IF(DOUBLE(COUNT(grade_in_prev_course)), (0)))

0.9622195985832349
```

Fetches 1 row

You can also query to find the rows that are anomalies. In most cases, students achieved top marks in their class despite having minimal study hours and low grades in their previous classes.

```
SELECT *
FROM mldemo.lr_input
WHERE got_top_marks != lr1(hours_studied, grade_in_prev_course)
LIMIT 10;
```

| hours_studied | grade_in_prev_course | got_top_marks |
|---------------|----------------------|---------------|
| 1             | 1                    | true          |
| 1             | 1                    | true          |
| 1             | 4                    | false         |
| 1             | 1                    | true          |
| 1             | 2                    | true          |
| 1             | 0                    | true          |
| 1             | 2                    | true          |
| 1             | 1                    | true          |
| 1             | 2                    | true          |
| 1             | 1                    | true          |

Fetches 10 rows

For details about using this model type, see [Classification Models](#).

## Support Vector Machine

Support vector machine (SVM) tries to find a hyperplane to divide data into two classes with these objectives:

- The hyperplane has the maximum margin, meaning the most distance between the two classes.
- The model correctly classifies the highest percentage of data points.

By default, SVM models in Ocint try to balance these two objectives. You can increase the priority of either objective by using the `regularizationCoefficient` option.

For data sets with two features, the SVM hyperplane represents a straight line on a Cartesian graph. For three features, the hyperplane represents a plane.

This example uses the SVM model on the same classification data set of student performance as used in the logistic regression section.

```
CREATE MLMODEL svm1 TYPE SUPPORT VECTOR MACHINE ON (
 SELECT *
 FROM mldemo.lr_input
) options('metrics'-'>'true');
```

Modified 0 rows

The `machine_learning_models` and `support_vector_machine_models` system catalog tables for SVM contain almost the same information as logistic regression.

```
SELECT name,
 on_select,
 num_arguments,
 negative_case,
 positive_case,
 correctly_classified
FROM sys.machine_learning_models a,
 sys.support_vector_machine_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'svm1';
```

| name | on_select                                  | num_arguments | negative_case | positive_case | correctly_classified |
|------|--------------------------------------------|---------------|---------------|---------------|----------------------|
| svm1 | <code>select * from mldemo.lr_input</code> | 2             |               |               |                      |

Fetches 1 row

The system catalog tables indicate the same percentage of correctly classified rows as the logistic regression model (about 84 percent).

The SVM model can increase the accuracy further by using a kernel. A kernel is a transformation to make data linearly separable. SVM in Ocient allows you to specify any kernel of your choice.

For example, suppose that a circle can separate two classes of data, dividing data points between those inside and outside the circle. The SVM model cannot find a line to divide the two classes very well, or this problem is not linearly separable. However, you can transform the data into three dimensions (three features) to make it linearly separable. This is the idea of a kernel.

You specify the kernel using the options `function1`, `function2`, `function3`, etc. Each function defines the formula for the kernel to create new features. Functions can use features from the original data set, represented by `x1`, `x2`, `x3`, etc.

To explain this, revisit the circle example. Suppose that the circle centered at the point (2, 2) is the class inside. Otherwise, the circle is the class outside. The data that is separable by a circle in two dimensions is linearly separable in three dimensions with the features `x1`, `x2`, `x1^2 + x2^2`.

This statement creates the data and the model.

```
CREATE TABLE mldemo.circle AS (
 SELECT x,
 y,
 CASE
 WHEN SQRT((x - 2) ^ 2 + (y - 2) ^ 2) <= 1 THEN 'INSIDE'
 ELSE 'OUTSIDE'
 END AS label
 FROM (
 SELECT RAND() * 10 AS x,
 RAND() * 10 AS y
 FROM sys.dummy10000
)
);
```

Modified 10000 rows

```
CREATE MLMODEL svm2 TYPE SUPPORT VECTOR MACHINE ON (
```

```

SELECT x,
 y,
 label
FROM mldemo.circle
) options(
 'metrics' -> 'true',
 'function1' -> 'x1',
 'function2' -> 'x2',
 'function3' -> 'x1^2 + x2^2'
);

```

Modified 0 rows

The example includes three functions, meaning that the kernel increases from two to three dimensions. The kernel functions refer to the original feature values `x1` and `x2`. If you examine the `correctly_classified` value in the `machine_learning_models` and `support_vector_machine_models` system catalog tables, you can see a model that classifies nearly everything correctly.

```

SELECT NAME,
 negative_case,
 positive_case,
 correctly_classified
FROM sys.machine_learning_models a,
 sys.support_vector_machine_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'svm2';

```

| name | negative_case           | positive_case          |
|------|-------------------------|------------------------|
| svm2 | cast('OUTSIDE' as CHAR) | cast('INSIDE' as CHAR) |

Fetches 1 row

One advantage of SVM kernels is that they are automatic. The model saves all the kernel information and transforms the data as needed. Hence, you can execute this model to predict a new point without redefining the kernel. You can pass in the two features the model was built over, and a prediction comes out.

The `sys.machine_learning_model_options` table is the location where the SVM model saves kernel information. This table saves all the options that you specify. Some models also add additional keys to this dictionary to save certain information needed for when the model is executed.

```

SELECT svm2(2, 2);
COALESCE(('INSIDE'), (NULL));

```

INSIDE

Fetches 1 row

For details about using this model type, see [Classification Models](#).

## Binary Classification with Neural Networks

A neural network using the `FEEDFORWARD_NETWORK` model type can perform binary classification. To use this model for classification, you must either convert labels to either 0 and 1 (using the `log_loss` option) or convert labels to -1 and 1 (using the `hinge_loss` option). Both options can achieve the same result.

This model can handle more complex cases, but using a different model for linearly separable data might be easier. In other words, a neural network model is useful when data is difficult to separate linearly using a kernel.

This example shows a feedforward network model using the `log_loss` option. Note the `CASE` statement to convert labels to 0 or 1.

```
CREATE MLMODEL logloss TYPE FEEDFORWARD NETWORK ON (
 SELECT x,
 y,
 CASE
 WHEN label = 'INSIDE' THEN 0
 ELSE 1
 END AS target
 FROM mldemo.circle
) options(
 'metrics'->'true',
 'hiddenLayers'->'2',
 'hiddenLayerSize'->'2',
 'outputs'->'1',
 'lossFunction'->'log_loss'
);
```

Modified 0 rows

This query finds that the classification accuracy is about 97 percent.

```
SELECT COUNT(*) / 10000.0
FROM (
 SELECT x,
 y,
 label,
 CASE
 WHEN logloss(x, y) = 0 then 'INSIDE'
 ELSE 'OUTSIDE'
 END AS predicted
 FROM mldemo.circle
)
WHERE label = predicted;

(_count(*)_0)/((10000.0))

0.9698
```

Fetches 1 row

To use the `hinge_loss` option, use -1 and 1 in the `CASE` statement.

```
CREATE MLMODEL hingeloss TYPE FEEDFORWARD NETWORK ON (
 SELECT x,
 y,
 CASE
 WHEN label = 'INSIDE' THEN -1
 ELSE 1
 END AS target
 FROM mldemo.circle
) options(
 'metrics'->'true',
 'hiddenLayers'->'2',
 'hiddenLayerSize'->'2',
 'outputs'->'1',
 'lossFunction'->'hinge_loss'
```

```
);
Modified 0 rows
```

The accuracy of the `hinge_loss` version is exactly the same as the `log_loss` version.

```
SELECT count(*) / 10000.0
FROM (
 SELECT x,
 y,
 label,
 CASE
 WHEN hingeloss(x, y) = -1 THEN 'INSIDE'
 ELSE 'OUTSIDE'
 END AS predicted
 FROM mldemo.circle
)
WHERE label = predicted;

(_count(*)_0)/((10000.0))

0.9698

Fetched 1 row
```

For details about using this model type, see [Other Models](#).

## K-Nearest Neighbors

K-nearest neighbors (KNN) can only use numeric features, but KNN handles multi-classification in the Ocient System. This algorithm can handle an arbitrary number of target classes.

One drawback with KNN is that it does not work at training time other than to take a snapshot of the training data. The algorithm operates on the data when you execute the model, so it can be a slow model for large data sets.

To demonstrate KNN functionality, this example uses a similar circle of size (2, 2) with some modifications to use more classes:

- Data within a unit circle of (2, 2) is classified as INSIDE.
- Data within a radius of 2 is classified as MIDDLE.
- Any other data is classified as OUTSIDE.

```
CREATE TABLE mldemo.circles as (
 SELECT x, y,
 CASE
 WHEN SQRT((x -2) ^ 2 + (y -2) ^ 2) <= 1 THEN 'INSIDE'
 WHEN SQRT((x -2) ^ 2 + (y -2) ^ 2) <= 2 THEN 'MIDDLE'
 ELSE 'OUTSIDE'
 END AS label
 FROM (
 SELECT RAND() * 10 as x,
 RAND() * 10 as y
 FROM sys.dummy10000
)
);

Modified 10000 rows
```

KNN works by finding the `k` nearest points to the one the model is trying to predict a classification for. `k` is an integer value that you must specify during model creation.

The model calculates distances using normal Euclidian distances. You can override the definition of distance by specifying a formula using the **distance** option.

After the model identifies the points, it computes a score for each class and returns the class with the highest score. The **weight** option can influence each score for the specified point. By default, the **weight** value is based on the inverse of the distance ( $1.0/(d+0.1)$ ). In other words, a point closer to the point you are trying to predict has a larger influence on the output result than a point further away. Any point outside of the nearest **k** points has no influence.

This example uses three classes.

```
CREATE MLMODEL knn1 TYPE K NEAREST NEIGHBORS ON (
 SELECT x,
 y,
 label
 FROM mldemo.circles
) options('k'-'>'3');
```

Modified 0 rows

These queries test some values to see if the classifications are correct.

```
SELECT knn1(2, 2);
```

-----

INSIDE

Fetches 1 row

```
SELECT knn1(0.5, 2);
```

-----

MIDDLE

Fetches 1 row

```
SELECT knn1(5,5);
```

-----

OUTSIDE

Fetches 1 row

KNN models perform work when you execute the model, and the amount of work increases as the input data increases.

One solution to this is data reduction, which means finding a smaller subset of rows to build the KNN model over, enough to ensure that the accuracy of the model is still high. This method front-loads the work when you create the model so that it is much faster to execute on new data.

To use the data reduction feature, set the **dataReduction** option to **true**. By default, the execution stops when the database finds a model that is 90 percent accurate over the training data or when the reduced model increases to 1,000 rows of data. You can override these defaults by setting the **maxReducedRows** or **targetAccuracy** options.

Additionally, if you are using data reduction, you can ask the database to collect metrics on the percent of rows correctly classified by the model because this is much faster to do on the reduced data set.

This example builds another KNN model over the same data and uses data reduction to find a model that is at least 95 percent accurate.



```
CREATE MLMODEL knn2 TYPE K NEAREST NEIGHBORS ON (
 SELECT x,
 y,
 label
 FROM mldemo.circles
) options(
 'k'-'>'3',
 'dataReduction'-'>'true',
 'metrics'-'>'true',
 'targetAccuracy'-'>'0.95'
);
```

Modified 0 rows

Query the `machine_learning_models` and `k_nearest_neighbor_models` system catalog tables to see the snapshot table used for the model and its accuracy.

```
SELECT table_name,
 correctly_classified
FROM sys.machine_learning_models a,
 sys.k_nearest_neighbor_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'knn2';
```

| table_name                | correctly_classified |
|---------------------------|----------------------|
| temp.knn86947930347148676 | 0.9541               |

Fetches 1 row

The model with data reduction correctly classifies about 95 percent of the training data. To see the size of the KNN model snapshot, you can query it by its table name.

```
SELECT COUNT(*) FROM temp.knn86947930347148676;
```

```
count(*)
```

```

300
```

Fetches 1 row

The snapshot is only 300 rows. Without data reduction, the row count would be much larger by orders of magnitude.

For details about using this model type, see [Classification Models](#).

### Multi-Class Classification with Neural Networks

The `FEEDFORWARD NETWORK` model type can also perform multi-class classification when the feature classes are numeric. Multi-class classification works by using vectors, where each position in the vector indicates the score for that class. The argmax value of the score of each vector determines its class.

Much of this example is similar to the previous [Machine Learning: Classification Models](#) tutorial with these modifications to the model options:

- Set the `lossFunction` option to `cross_entropy`.
- Set the `outputs` option to specify the number of classes for the model to use (in this example: 3).
- Set the `useSoftMax` option to `true`, which applies a softmax function to the final vector.

When the loss function is `cross_entropy_loss`, the option `useSoftMax` defaults to `true`.

```
CREATE MLMODEL crossentropy TYPE FEEDFORWARD NETWORK ON (
 SELECT x,
 y,
 CASE
 WHEN label = 'INSIDE' THEN { { 1.0,
 0.0,
 0.0 } }
 WHEN label = 'MIDDLE' THEN { { 0.0,
 1.0,
 0.0 } }
 ELSE { { 0.0,
 0.0,
 1.0 } }
 END AS TARGET
 FROM mldemo.circles
) options(
 'metrics'->'true',
 'hiddenLayers'->'2',
 'hiddenLayerSize'->'2',
 'outputs'->'3',
 'lossFunction'->'cross_entropy_loss',
 'useSoftMax'->'true'
);
```

Modified 0 rows

Execute this model to return a vector.

```
SELECT crossentropy(5,5) AS predicted;
```

predicted

```

[[1.3436428911280516E-7, 1.811726873154417E-7, 0.9999996844630236]]
```

Fetches 1 row

To get the classification, use the `VECTOR_ARGMAX` function to retrieve the index of the largest value.

Matrixes index the first row and first column with 1. However, the `VECTOR_ARGMAX` function starts the count at 0.

```
SELECT VECTOR_ARGMAX(crossentropy(5,5)) AS predicted;
```

predicted

```

2
```

Fetches 1 row

This query finds the overall accuracy of the model.

```
SELECT count(*) / 10000.0
FROM mldemo.circles
WHERE (
```

```

 label = 'INSIDE'
 AND VECTOR_ARGMAX(crossentropy(x, y)) = 0
)
 OR (
 label = 'MIDDLE'
 AND VECTOR_ARGMAX(crossentropy(x, y)) = 1
)
 OR (
 label = 'OUTSIDE'
 AND VECTOR_ARGMAX(crossentropy(x, y)) = 2
);

(_count(*)_0)/((10000.0))

0.9298

Fetched 1 row

```

The model is about 93 percent effective. You can improve the accuracy by modifying the [Other Models](#), including:

- `hiddenLayers`
- `hiddenLayerSize`
- `lossFuncNumSamples`

Increasing the resources of these options can improve accuracy, but could increase the time needed to train the model.

For details about using this model type, see [Other Models](#).

### Multi-Class Classification with Support Vector Machines and Logistic Regression

Both logistic regression and SVM are binary classifiers, but you can use either to build a multi-class classifier to handle any number of classes. OcintML can perform this automatically with either model.

This example shows the `CREATE` statement for the SVM model, which uses the same data as the [Machine Learning: Classification Models](#) and [Machine Learning: Classification Models](#) examples.

```

CREATE MLMODEL circ1 TYPE SUPPORT VECTOR MACHINE ON (
 SELECT x,
 y,
 label
 FROM mldemo.circles
) options(
 'metrics' -> 'true',
 'function1' -> 'x1',
 'function2' -> 'x2',
 'function3' -> 'x1^2 + x2^2'
);

Modified 0 rows

```

This statement uses the same kernel as the example for performing binary classification on [Machine Learning: Classification Models](#).

This query checks the metrics in the `machine_learning_models` and `support_vector_machine_models` system catalog tables to see the accuracy.

```

SELECT correctly_classified
FROM sys.machine_learning_models a,
 sys.support_vector_machine_models b

```

```
WHERE a.id = b.machine_learning_model_id
 AND name = 'circ1';
```

```
correctly_classified
```

```

0.9579
```

```
Fetches 1 row
```

This model is slightly more accurate than the neural network classifier.

This example tries logistic regression.

```
CREATE mlmodel circ2 TYPE LOGISTIC REGRESSION ON (
 SELECT x,
 y,
 label
 FROM mldemo.circles
) options('metrics'=>'true');
```

```
Modified 0 rows
```

```
SELECT correctly_classified
FROM sys.machine_learning_models a,
 sys.logistic_regression_models b
WHERE a.id = b.machine_learning_model_id
 AND NAME = 'circ2';
```

```
correctly_classified
```

```

0.9028
```

```
Fetches 1 row
```

Logistic regression shows lower accuracy than the neural network model.

These classification models can handle non-numeric data.

### Naive Bayes

The first of these models is naive Bayes. This model can handle binary or multi-class classification and features of any data type.

By default, naive Bayes treats each feature as a discrete value. For example, if you assign features with values of 1, 2, 3, and 4, the model does not give any numeric significance to them. In this case, this model cannot predict the class for the sample with a feature value of 3.5 because that value is not explicitly assigned as a feature.

However, the naive Bayes model can define certain input features as continuous features. These should be numeric values. In this case, the naive Bayes model finds the best fit normal distribution for that feature to do the probability calculations.

Naive Bayes models essentially use the probability rules of Bayes' Theorem to compute the most likely class based on the training data. The rules are only correct if the features are independent (uncorrelated). However, in practice, the models tend to work well with real-world data.

This example uses this simple table to demonstrate naive Bayes models.

```
SELECT * FROM mldemo.cars;
```

| color | type | origin |
|-------|------|--------|
|-------|------|--------|

|        |        |          |
|--------|--------|----------|
| yellow | sports | domestic |
| yellow | sports | imported |
| yellow | suv    | imported |
| red    | sports | domestic |
| yellow | suv    | domestic |
| red    | suv    | imported |
| red    | sports | domestic |
| yellow | suv    | imported |
| red    | sports | imported |
| red    | sports | domestic |

Fetches 10 rows

color, type, and origin are the input features, which are all strings. true and false are the class labels. In this case, all of the input features are discrete.

This example builds the naive Bayes model.

```
CREATE MLMODEL nb1 TYPE NAIVE BAYES ON (
 SELECT color,
 type,
 origin,
 stolen
 FROM mldemo.cars
) options('metrics'-'>'true');
```

Modified 0 rows

This query examines the information in the machine\_learning\_models and naive\_bayes\_models system catalog tables for naive Bayes models.

```
SELECT NAME,
 on_select,
 num_arguments,
 result_probability_table,
 feature_result_matrix_table,
 correctly_classified
FROM sys.machine_learning_models a,
 sys.naive_bayes_models b
WHERE a.id = b.machine_learning_model_id
 AND NAME = 'nb1';
```

| name | on_select                                                      | num_arguments | result_probability_table | feature_result_matrix_table | correctly_classified |
|------|----------------------------------------------------------------|---------------|--------------------------|-----------------------------|----------------------|
| nb1  | select color, "type", origin, stolen from user.nb_input 3 temp | 3             |                          |                             | 80                   |

Fetches 1 row

In the model, the correctly\_classified value indicates the accuracy is 80 percent, meaning just two rows are classified incorrectly. The two table names in the temp schema hold all the necessary data for executing the model as a function.

This query calls the model to classify new data.

```
SELECT nb1('red', 'suv', 'domestic') AS prediction;
```

prediction

```
false
```

```
Fetches 1 row
```

Here is another example with a different data set to show how continuous features work with naive Bayes models.

```
SELECT * FROM mldemo.golf;
```

| humidity | play  |
|----------|-------|
| 70       | false |
| 75       | true  |
| 86       | true  |
| 95       | false |
| 80       | true  |
| 85       | false |
| 96       | true  |
| 90       | true  |
| 91       | false |
| 90       | false |
| 70       | true  |
| 70       | true  |
| 80       | true  |
| 65       | true  |

```
Fetches 14 rows
```

In this case, the only input feature is a continuous feature, but you can have multiple continuous features, and you can have a mix of continuous and discrete features.

To build a naive Bayes model over this data, you must set which input features are continuous in the model.

```
CREATE MLMODEL nb2 TYPE NAIVE BAYES ON (
 SELECT *
 FROM mldemo.golf
) options('continuousFeatures'-'>'1', 'metrics'-'>'true');
```

```
Modified 0 rows
```

The `continuousFeatures` option is a comma-separated list (no spaces) of input column indexes that are continuous. Indexes start at 1.

This query checks the `correctly_classified` value in the `machine_learning_models` and `naive_bayes_models` system catalog tables.

```
SELECT correctly_classified
FROM sys.machine_learning_models a,
 sys.naive_bayes_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'nb2';
```

| correctly_classified |
|----------------------|
| 0.7142857142857143   |

```
Fetches 1 row
```

Naive Bayes is a powerful classification algorithm, but it operates as a black box, making it hard to assess how it makes classification decisions.

For details, see [Classification Models](#).

## Decision Trees

Unlike naive Bayes, the decision tree model allows you to see how the model works and makes its decisions.

A decision tree branches based on the value of a single feature, while the leaves of the tree are the output classes. For SQL syntax, this model uses multiple nested **CASE** statements. You can examine the whole nested **CASE** statement that defines the model, but it might be very long.

The decision tree model in Ocint has limited support for continuous features. The model finds the average value of the feature and allows a two-way branch based on whether the value is lower or higher than the average.

Creating a decision tree model is exactly like a naive Bayes model, except with a different type. Decision trees also use the **continuousFeatures** option, which functions the same.

These decision tree examples use the same data sets as the naive Bayes examples.

```
CREATE MLMODEL dt1 TYPE DECISION TREE ON (
 SELECT color,
 "type",
 origin,
 stolen
 FROM mldemo.cars
) options('metrics'-'>'true');
```

Modified 0 rows

This query examines the `machine_learning_models` and `decision_tree_models` system catalog tables.

```
SELECT correctly_classified
FROM sys.machine_learning_models a,
 sys.decision_tree_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'dt1';
```

correctly\_classified

-----  
0.8

Fetches 1 row

This model has the same 80 percent accuracy as the naive Bayes model.

This query examines the **CASE** statement that defines the model.

```
SELECT case_statement
FROM sys.machine_learning_models a,
 sys.decision_tree_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'dt1';
```

case\_statement

-----

```
CASE
 WHEN x3 = cast('domestic' as CHAR) THEN CASE
 WHEN x1 = cast('red' as CHAR) THEN CASE
 WHEN x2 = cast('sports' as CHAR) THEN cast('true' as BOOLEAN)
 ELSE cast('false' as BOOLEAN)
 END
 WHEN x1 = cast('yellow' as CHAR) THEN cast('false' as BOOLEAN)
```

```

 ELSE cast('false' as BOOLEAN)
 END
 WHEN x3 = cast('imported' as CHAR) THEN CASE
 WHEN x2 = cast('suv' as CHAR) THEN CASE
 WHEN x1 = cast('red' as CHAR) THEN cast('false' as BOOLEAN)
 WHEN x1 = cast('yellow' as CHAR) THEN cast('true' as BOOLEAN)
 ELSE cast('false' as BOOLEAN)
 END
 WHEN x2 = cast('sports' as CHAR) THEN cast('true' as BOOLEAN)
 ELSE cast('false' as BOOLEAN)
 END
 ELSE cast('false' as BOOLEAN)
END

```

Fetches 1 row

Like with the naive Bayes model, the decision tree model can predict new data.

```
SELECT dt1('red', 'suv', 'domestic') AS prediction;
```

```

prediction

false

```

Fetches 1 row

This example builds a model over the golfing data.

```

CREATE mlmodel dt2 TYPE DECISION TREE ON (
 SELECT *
 FROM mldemo.golf
) options('continuousFeatures'-'>'1', 'metrics'-'>'true');

```

Modified 0 rows

Review the `machine_learning_models` and `decision_tree_models` system catalog tables to see the accuracy of this model.

```

SELECT correctly_classified
FROM sys.machine_learning_models a,
 sys.decision_tree_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'dt2';

```

```

correctly_classified

0.7142857142857143

```

Fetches 1 row

Again, the decision tree shows similar accuracy to the naive Bayes model.

## Related Links

[Machine Learning Model Functions](#)

[Classification Models](#)



### 5.2.3 Clustering Analysis and Dimensionality Reduction

This tutorial uses examples to explain the clustering and dimensionality reduction capabilities in OcintML™.

#### Clustering Models

Clustering models bear similarities to classification, but they use unsupervised learning, which means they do not use any class labels. Instead, the algorithm tries to identify groupings on its own by finding clusters of data that seem closer to each other and farther away from other clusters.

These models generate an integer label, but its ordering is arbitrary. The Ocint clustering models require you to specify upfront the number of clusters.

#### K-Means Clustering

K-means is by far the most well-known clustering algorithm because it is simple and fast. K-means performs particularly well if you can scale your features so clusters are roughly circular and equal in size.

These examples demonstrate the k-means model. The examples do not use a data set that is primed for optimal performance of the model such that they show the shortcomings of the k-means model and how it can perform relatively well even with sub-optimal data.

The examples use a data set of three-dimensional points.

```
SELECT x, y, z FROM mldemo.clusters_3d LIMIT 10;
```

| x                   | y                    | z                   |
|---------------------|----------------------|---------------------|
| -2.332818550827427  | 2.1009728159912044   | -0.7199553405333495 |
| 0.27417054580238553 | -0.29545777905624226 | 0.6797340230215586  |
| -2.8608875606526794 | 2.8532647291218884   | -0.7296054608643955 |
| -4.9539478834915345 | -4.987358010702081   | -0.818608017693332  |
| 0.7043958454569171  | -0.7209720558364705  | -0.7879961167668906 |
| 4.949298606341526   | 4.956911476243746    | 0.6015700102502752  |
| 2.245935331812275   | -2.158347482865157   | -1.0786578947921444 |
| -1.3725012757407247 | 1.5652243270484643   | -2.5951429274728794 |
| 4.516051945094492   | 4.560705560578962    | -0.1569348000058415 |
| 5.140105808358035   | -4.9583101530224     | -1.0064835641873153 |

Fetches 10 rows

This example shows a k-means model created over this data. You can specify multiple options, but the only one that is required is the `k` value, which represents the number of clusters.

```
CREATE MLMODEL kmeans1 TYPE KMEANS ON (
 SELECT x,
 y,
 z
 FROM mldemo.clusters_3d
) options('k'=>'3');
```

Modified 0 rows

This data set has labels, but they are hidden from the model to allow for unsupervised learning.

These example queries compare how well the unsupervised k-means clustering performs against the actual correct classes and labels.

Before you can do that, you must figure out which cluster numbers correspond with which labels.

```
SELECT COUNT(*),
 cluster_num
FROM (
```

```

 SELECT kmeans1(x, y, z) AS cluster_num
 FROM mldemo.clusters_3d
 WHERE label = 'LEFT'
)
GROUP BY cluster_num;

```

| count(*) | cluster_num |
|----------|-------------|
| 3165     | 0           |
| 146329   | 1           |

Fetches 2 rows

```

SELECT COUNT(*),
 cluster_num
FROM (
 SELECT kmeans1(x, y, z) AS cluster_num
 FROM mldemo.clusters_3d
 WHERE label = 'RIGHT'
)
GROUP BY cluster_num;

```

| count(*) | cluster_num |
|----------|-------------|
| 3877     | 0           |
| 145761   | 2           |

Fetches 2 rows

```

SELECT COUNT(*),
 cluster_num
FROM (
 SELECT kmeans1(x, y, z) AS cluster_num
 FROM mldemo.clusters_3d
 WHERE label = 'CENTER'
)
GROUP BY cluster_num;

```

| count(*) | cluster_num |
|----------|-------------|
| 700868   | 0           |

Fetches 1 row

The results indicate that cluster 0 is the CENTER class, cluster 1 is LEFT, and cluster 2 is RIGHT.

With that information, you can compute the overall accuracy.

```

SELECT count(*) / 1000000.0
FROM mldemo.clusters_3d
WHERE (
 kmeans1(x, y, z) = 0
 AND label = 'CENTER'
)
OR (
 kmeans1(x, y, z) = 1

```

```

 AND label = 'LEFT'
)
 OR (
 kmeans1(x, y, z) = 2
 AND label = 'RIGHT'
);

(_count(*)_0)/((1000000.0))

0.992958

Fetched 1 row

```

The accuracy is very good. This calculation indicates that the model never miscategorized CENTER, but there were some wrong classifications for the LEFT and RIGHT labels, although they were rare.

For details, see [Clustering and Dimension Reduction Models](#).

### Gaussian Mixture Models

Gaussian mixture models (GMMs) also perform clustering, however they use a significantly more complex algorithm. GMMs can handle several things that k-means cannot, such as:

- GMMs can handle clusters that are not circular, i.e., they have different variances in different directions.
- GMMs can handle clusters that have an arbitrary rotation, i.e., they can have covariances.
- GMMs can handle the fact that all clusters might not be equally as likely, i.e. if a point is located right between two clusters, then it is more likely to be the one more common in the training data.
- GMMs can show the probability of a new point belonging to each cluster rather than outputting a single cluster value.

GMMs operate by finding a weighted mixture of k multi-variate Gaussians that is most likely to represent the population from which the data was sampled. Each Gaussian in the mix has a mean vector, which represents the center of each cluster.

After you create the model, it is simple to determine the cluster that a point most likely belongs to. The examples demonstrate how to find the highest probability cluster.

This example makes a model over the same data set. This model requires the `numDistributions` option, which represents the number of clusters.

```

CREATE MLMODEL gmm TYPE GAUSSIAN MIXTURE MODEL ON (
 SELECT x,
 y,
 z
 FROM mldemo.clusters_3d
) options('metrics'-'>'true', 'numDistributions'-'>'3');

Modified 0 rows

```

This query executes the GMM model.

```

SELECT gmm(0,0,0) AS class_probabilities;

class_probabilities

[[2.0445296213545875E-4, 0.999582295498657, 2.1325153920763012E-4]]

Fetched 1 row

```

The results are very different than executing a k-means model because these are probabilities. In this case, the output means the probability of (0,0,0) being the second class is very high (greater than 99.9 percent), while the

probability of it being the other classes is essentially zero.

To find the most likely class, use the `vector_argmax()` function.

```
SELECT VECTOR_ARGMAX(gmm(0,0,0)) AS class_probabilities;
```

```
class_probabilities
```

```

```

```
1
```

Fetches 1 row

You can write a query to see how accurate this model is. But first, you must figure out the association between classes and labels.

```
SELECT COUNT(*),
 cluster_num
FROM (
 SELECT vector_argmax(gmm(x, y, z)) AS cluster_num
 FROM mldemo.clusters_3d
 WHERE label = 'LEFT'
)
GROUP BY cluster_num;
```

```
count(*) cluster_num
```

```

```

```
149483 2
```

```
2 0
```

```
9 1
```

Fetches 3 rows

```
SELECT COUNT(*),
 cluster_num
FROM (
 SELECT vector_argmax(gmm(x, y, z)) AS cluster_num
 FROM mldemo.clusters_3d
 WHERE label = 'CENTER'
)
GROUP BY cluster_num;
```

```
count(*) cluster_num
```

```

```

```
700868 1
```

Fetches 1 row

```
SELECT COUNT(*),
 cluster_num
FROM (
 SELECT vector_argmax(gmm(x, y, z)) AS cluster_num
 FROM mldemo.clusters_3d
 WHERE label = 'RIGHT'
)
GROUP BY cluster_num;
```

```
count(*) cluster_num
```

```

6 1
3 2
149629 0

```

Fetches 3 rows

This model misclassified 20 rows, which is far better than the 7,042 rows the k-means model misclassified. This is a direct result of the additional complexity of the GMM.

GMM models can handle much more complex situations than k-means models, but this comes at the cost of more time training and executing the model. The time a GMM model takes compared to a k-means model depends on the number of clusters.

For details, see [Clustering and Dimension Reduction Models](#).

## Dimensionality Reduction

Dimensionality reduction algorithms reduce the number of input features while still keeping as much of the meaningful properties of the data as possible. Models are quicker to build, and often higher quality after a dimensionality reduction model reduces the number of input features.

A common first step in an analysis is to use dimensionality reduction to simplify the data. See these examples that demonstrate how to use dimensionality reduction as a preprocessing step before using other model types.

### Principal Component Analysis

Principal component analysis (PCA) is an unsupervised algorithm that only operates on the inputs, and does not understand what the data is being used for. It is also a linear dimensionality reduction algorithm, meaning that the new features it generates are linear combinations of existing features.

PCA generates as many new features as there are input features. So by itself, it is not reducing the number of dimensions. However, PCA creates new features that try to maximize variance and sorts the new features in terms of the amount of variance they contain. The system catalog tables provide information on how much variance is contained by the new features. You can use this information to find how many trailing new features to drop.

This PCA tutorial uses a new data set. Start with a regression problem that is trying to find the best-fit polynomial for  $f(x_1, x_2, x_3) = y$ . The walkthrough starts with a [Regression Models](#) model to see how well it does using three input features, and then it uses PCA to reduce the number of variables without losing accuracy.

This example shows the Polynomial Regression model.

```

CREATE MLMODEL poly TYPE POLYNOMIAL REGRESSION ON (
 SELECT x1,
 x2,
 x3,
 y
 FROM mldemo.pca_poly
) options('order'-'>'2', 'metrics'-'>'true');

```

Modified 0 rows

The system catalog table shows that the model fits the data perfectly.

```

SELECT coefficient_of_determination
FROM sys.machine_learning_models a,
 sys.polynomial_regression_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'poly';

```

```
coefficient_of_determination
```

```

1.0
```

```

Fetched 1 row
```

In this instance, PCA can reduce the three independent variables to two.

The first step is to build a PCA model over the input features.

```
CREATE MLMODEL pca1 TYPE PRINCIPAL COMPONENT ANALYSIS ON (
 SELECT x1,
 x2,
 x3
 FROM mldemo.pca_poly
);
```

```

Modified 0 rows
```

The model does not include y because PCA operates over only the input features.

Examine the `machine_learning_models` and `principal_component_analysis_models` system catalog tables.

```
SELECT importance
FROM sys.machine_learning_models a,
 sys.principal_component_analysis_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'pca1';
```

```
importance
```

```

[0.6022962127628679, 0.33286202821081917, 0.06484175902631285]
```

```

Fetched 1 row
```

The `importance` value indicates that over 93 percent of the signal is in the first two PCA output features, which means it is possible to have a robust model even if you remove the third feature.

This example uses the Polynomial Regression model to access the PCA output features. The example executes the `pca1` PCA function with the PCA input features followed by the number of features. This number starts at 1, so this example uses PCA features 1 and 2, but not the last feature.

```
CREATE MLMODEL poly2 TYPE POLYNOMIAL REGRESSION ON (
 SELECT pca1(x1, x2, x3, 1),
 pca1(x1, x2, x3, 2),
 y
 FROM mldemo.pca_poly
) options('order'-'>'2', 'metrics'-'>'true');
```

```

Modified 0 rows
```

Notice that the example still references all three input features, but this model now has only two independent variables. As a result, the model trains 15 percent faster than the version with three independent variables. It is also a much simpler model. The model has six terms in the polynomial regression instead of 10, when there were three independent variables.

This query examines how well the model fits the data.

```
SELECT coefficient_of_determination
FROM sys.machine_learning_models a,
 sys.polynomial_regression_models b
WHERE a.id = b.machine_learning_model_id
```

```

 AND name = 'poly2';

coefficient_of_determination

0.9952839146818417

Fetched 1 row

```

The model is a great fit with more than 99 percent accuracy.

In contrast, here is an example where PCA is not a good idea. This example goes back to the three-dimensional clusters data set used in the [Machine Learning: Clustering and Dimension Models](#) examples.

This example creates the PCA model and checks the `importance` value of the PCA output features in the `machine_learning_models` and `principal_component_analysis_models` system catalog tables.

```

CREATE MLMODEL pca2 TYPE PRINCIPAL COMPONENT ANALYSIS ON (
 SELECT x,
 y,
 z
 FROM mldemo.clusters_3d
);

Modified 0 rows

SELECT importance
FROM sys.machine_learning_models a,
 sys.principal_component_analysis_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'pca2';

importance

[0.3688781607553818, 0.3333390832624229, 0.2977827559821952]

Fetched 1 row

```

In this case, all three PCA output features are fairly evenly weighted. By removing the last feature, the model covers only about 70% of the signal. This strongly indicates that proceeding with a two-feature model would be problematic and inaccurate.

### Linear Discriminant Analysis

While PCA is unsupervised, linear discriminant analysis (LDA) is a supervised dimensionality reduction algorithm. LDA works only with numeric classification algorithms, but the classification can be binary or multi-class. Hence, LDA understands how the data should be used, making it operate differently.

Creating an LDA model is mostly similar to PCA. One difference is that LDA also requires a class or label.

This tutorial uses the three-dimensional cluster data that the PCA model struggled to work with. The example uses LDA to reduce the features of the cluster data by using the label already contained in the data set.

```

CREATE MLMODEL lda1 TYPE LINEAR DISCRIMINANT ANALYSIS ON (
 SELECT x,
 y,
 z,
 label
 FROM mldemo.clusters_3d
);

```

Modified 0 rows

As with PCA, there is an `importance` column in the system catalog tables.

```
SELECT importance
FROM sys.machine_learning_models a,
 sys.linear_discriminant_analysis_models b
WHERE a.id = b.machine_learning_model_id
 AND name = 'lda1';
```

importance

```

[1.0000000000000133, -1.4051695910079908E-20, -1.3342566035320652E-14]
```

Fetches 1 row

This output strongly indicates that only the first LDA output feature matters, which means that it is possible to make a single feature classification model.

This example takes the LDA-reduced model as input and uses it to create a neural network model, using the FEEDFORWARD NETWORK model. Note that this model uses the `cross_entropy_loss` option to enable multi-class classification.

```
CREATE MLMODEL three_d_clusters_with_1_feature TYPE FEEDFORWARD NETWORK ON (
 SELECT lda1(x, y, z, 1),
 CASE
 WHEN label = 'LEFT' THEN { { 1.0,
 0.0,
 0.0 } }
 WHEN label = 'RIGHT' THEN { { 0.0,
 1.0,
 0.0 } }
 ELSE { { 0.0,
 0.0,
 1.0 } }
 END AS target
 FROM mldemo.clusters_3d
) options(
 'metrics' -> 'true',
 'hiddenLayers' -> '2',
 'hiddenLayerSize' -> '4',
 'outputs' -> '3',
 'lossFunction' -> 'cross_entropy_loss',
 'useSoftMax' -> 'true'
);
```

Modified 0 rows

Execute the model for a new point to see if it returns a reasonable value.

```
SELECT three_d_clusters_with_1_feature(lda1(0, 0, 0, 1)) AS predicted;
```

predicted

```

[[0.007386760165583984, 0.0022085315302674473, 0.9904047083041486]]
```

Fetches 1 row



In this case, the query asks for a prediction for the input 0, 0, 0. This input first passes through the LDA model because that is what the neural network model was built on.

The query asks the LDA model for its first component, and the query executes the neural network model with that value. The highest probability class it returns is the third class, defined as CENTER in the query.

```
SELECT count(*) / 1000000.0
FROM mldemo.clusters_3d
WHERE (
 VECTOR_ARGMAX(
 three_d_clusters_with_1_feature(lda1(x, y, z, 1))
) = 0
 AND label = 'LEFT'
)
OR (
 VECTOR_ARGMAX(
 three_d_clusters_with_1_feature(lda 1(x, y, z, 1))
) = 1
 AND label = 'RIGHT'
)
OR (
 VECTOR_ARGMAX(
 three_d_clusters_with_1_feature(lda1(x, y, z, 1))
) = 2
 AND label = 'CENTER'
);

(_count(*)_0)/((1000000.0))

0.818108

Fetched 1 row
```

This model is nearly 82 percent accurate despite being simplified from three features to one. While the accuracy is worse, the time required to train this model is greatly reduced.

## Related Links

[Machine Learning Model Functions](#)

[Clustering and Dimension Reduction Models](#)

---

## 6 Database Administration

Database Administrators are responsible for ensuring the smooth operation of an Ocient® System and learning how to maximize performance to meet the goals of their organization.

See these pages for key concepts that are useful for understanding Ocient System dynamics and managing users.

### Tutorials

### Security

These pages explain system authentication, SSO, network security, and encryption options.

- [Ocient Security Guide](#)
- [Set Up Data Encryption](#)
- [Authentication Methods](#)
- [Database Password Security Settings](#)

### User Control

These pages explain Ocient user management, including controlling security and priority for user groups:

- [Manage Users, Groups, and Roles](#)
- [Workload Management and Service Classes](#)
- [Workload Management Walkthrough](#)
- [Object-Type Level Privileges Management](#)

### Query Performance and Tuning

Configure keys and indexes before loading large amounts of data into a table. These pages provide detailed information on how to optimize query performance by setting up Ocient keys and indexing:

- [TimeKeys and Clustering Keys](#)
- [Secondary Indexes](#)
- [CREATE TABLE SQL Statement Examples](#)
- [Manage Distributed Tasks](#)

### Storage

These pages explain compression and caching options in the Ocient System:

- [Table Compression Options](#)
- [Global Dictionary Compression](#)
- [Result Set Caching](#)
- [Remove Records from an Ocient System](#)

### SQL Reference Pages

For detailed documentation on supported DDL or DCL commands and syntax, see:

- [General SQL Syntax](#)
- [Data Definition Language \(DDL\) Statement Reference](#)
- [Data Control Language \(DCL\) Statement Reference](#)

### Related Links

[Connect to Ocient](#)

[Ingest Data with Legacy LAT Reference](#)

## 6.1 Ociant Security Guide

This security guide is intended to assist organizations in hardening Software as a Service (SaaS) or customer-deployed Ociant® Systems. OciantCloud® and Ociant Management Services include Ociant support personnel configuring these settings based on organizational requirements and the applicable service agreement. Implement these security features and measures to protect an Ociant System and ensure data privacy.

This guide is not exhaustive and represents a security baseline for an Ociant System. Depending on your environment architecture and services, not all security features and measures might apply.

### Authentication and Access Control;

This section applies to the database application of the Ociant System. You can install the database application on a standard Linux® operating system (OS) distribution and apply authentication and access controls to the application host.

### Identity Provider or Single Sign-On (SSO)

You can configure the Ociant Hyperscale Data Warehouse™ (OHDW) to use any OpenID Connect (OIDC) identity provider (such as OKTA®) for SSO. For details, see [Authentication Methods](#).

Due to the potential sensitive data access provided through the OHDW, use multi-factor authentication (MFA) for all accounts that access the OHDW.

Ociant recommends implementing an OIDC identity provider on the OHDW and utilizing MFA for authentication access. Any local accounts should be limited to emergency access to recover SSO issues.

### Role-Based Access Control (RBAC) Groups

The OHDW includes default System and Database Roles. The system also allows database administrators to create access groups with specific tables or views. You can assign these access groups to database users to restrict access to data based on the data type or classification.

Ociant recommends assigning System and Database Roles based on the principles of least privilege, separation of duties, and need-to-know. Access groups with specific tables, rows, or views can further restrict access based on data security requirements.

For details, see [Manage Users, Groups, and Roles](#).

### Network Security

This section applies to the network segment that hosts the Ociant System.

### Secure Bucket Access and Data Streaming

The OHDW supports data loading using the Apache® Kafka® stream, cloud bucket (e.g., Amazon® Web Services<sup>SM</sup> (AWS<sup>SM</sup>) S3), or local file source. For details, see [Data Pipelines](#).

### Secure Bucket Access

If loading data from a cloud bucket like AWS S3, Google Cloud™ GCP storage bucket, and so on, use network access controls, IP whitelisting, or data-in-transit encryption between the OHDW Loader Nodes and the cloud bucket. Only Loader Nodes need access to a data source.

### Secure Data Streaming

The OHDW supports Kafka Source data streaming to Loader Nodes: `source-configuration`

Ociant recommends utilizing data-in-transit encryption between the Kafka data source and Loader Nodes.

## Network Exposure and Firewall

OHDW hosts should remain in the same network segment. However, you must apply host and network-based firewalls.

Ocident recommends using deny-all allow-by-exception firewall policies on hosts and network infrastructure to secure OHDW environments.

OHDW hosts should not be publicly accessible but within organizational network boundaries and security (or communication service provider (CSP) network boundaries and security).

The OHDW uses these ports, protocols, and services externally and internally.

### Communications External to the OHDW

| Port | Protocol | Service  | Source               | Destination      |
|------|----------|----------|----------------------|------------------|
| 4050 | TCP      | JDBC     | Ocident SQL Hosts    | JDBC Connectors  |
| 8090 | TCP      | telegraf | Ocident Loader Hosts | Data Source      |
| 9090 | TCP      | REST     | Ocident Loader Hosts | Metrics Platform |

### Communications Internal to the OHDW

| Port  | Protocol | Service   | Source                                              | Destination                                         |
|-------|----------|-----------|-----------------------------------------------------|-----------------------------------------------------|
| 5050  | TCP      | rolehostd | Ocident LAT Hosts                                   | Ocident Loader Hosts                                |
| 8443  | TCP      | nginx     | Ocident Loader Hosts                                | Ocident LAT Hosts                                   |
| 17600 | TCP      | rolehostd | Ocident SQL and Metadata Hosts                      | Ocident SQL and Metadata Hosts                      |
| 17700 | TCP      | rolehostd | Ocident SQL, Loader, and Foundation Hosts           | Ocident SQL, Loader, and Foundation Hosts           |
| 17800 | TCP      | rolehostd | Ocident Loader and Foundation Hosts                 | Ocident Loader and Foundation Hosts                 |
| 17900 | TCP      | rolehostd | Ocident SQL, Metadata, Loader, and Foundation Hosts | Ocident SQL, Metadata, Loader, and Foundation Hosts |

## Data Security and Encryption

This section covers the data security and encryption of the database application. Apply the data security and encryption controls to the Linux OS host.

### Data-at-Rest Encryption

The OHDW supports self-encryption drives (SED) that meet the Trusted Computing Group (TCG) Opal standard or better.

Ocident recommends utilizing NVMe drives that support Opal or SED and implementing data-at-rest encryption. Encryption strength should be based on data and system sensitivity. For details, see [Ocident System Requirements](#) and [Ocident System Requirements](#).

## Data-in-Transit Encryption

You can encrypt external connections using JDBC through transport layer security (TLS) or secure sockets layer (SSL). For details, see [Secure Connections Using TLS](#).

Ociant recommends using at least TLS 1.2 for external connections to the OHDW.

## System and Application Security

This section applies to both the Linux OS host and the database application.

### Operating System Hardening

For operating systems supported by the OHDW, see [Ociant System Requirements](#).

Ociant recommends the use of an industry-standard OS hardening baseline like the CIS Benchmarks™ or the Defense Information Systems Agency (DISA) Security Technical Implementation Guides (STIGs).

### Operating System Patching

You must frequently and consistently update hosts running the OHDW to remediate any existing vulnerabilities within the OS.

Ociant recommends performing periodic vulnerability assessments or scans and updating OS packages based on severity and timeframe.

### Application Patching

Ociant tracks, monitors, and remediates vulnerabilities within the OHDW application and publishes updated software releases periodically.

Ociant recommends keeping your OHDW application updated to the most current version.

For details, see [Ociant Software Upgrade](#).

## Application and System Maintenance

Ociant recommends performing regular and periodic application and system maintenance to preserve system performance. For details about the OHDW maintenance activities, see [Maintenance Overview](#).

## Secrets and Key Management

This section applies to secrets and keys within the Ociant System and database application. The section does not include Linux OS host secrets and keys but should have appropriate authentication and access controls applied. For details, see [Set Up Data Encryption](#).

### SED Key

Ociant recommends using a Key Management Interoperability Protocol (KMIP) compliant key management platform to handle SED lock and unlock operations.

By default, during system installation, if the install detects Opal-compliant hardware, the install enables full disk encryption with the encryption key stored on the OS drive. For details, see [Ociant System Requirements](#).

## Auditing and Log Management

This section applies to auditing within the database application. The section does not include Linux OS host and infrastructure logs but should have appropriate auditing controls applied.

## Audit Management

The Ocint System stores security audit logs for the database application in the `/var/log/` directory on the Linux OS host. You can see these files in that directory.

| File                       | Description                                                                                                                                                                                               |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>rolehostd.log</code> | Contains all of the error, informational, and debug logs from the running database system.                                                                                                                |
| <code>query.json</code>    | Contains information on all queries and statements executed on the system. This file includes information such as the user who ran the query (or command), the time and duration, and many other details. |
| <code>security.json</code> | Contains information on both successful and unsuccessful connections to the system.                                                                                                                       |

For details, see [Log Monitoring](#).

## SIEM or System Log Forwarding

Ocint recommends using a security information and event manager (SIEM) or system log (syslog) forwarder to ingest the `rolehostd.log`, `query.json`, and `security.json` audit log files located on the Linux OS host.

## Privacy

This section covers privacy controls of the database application.

### Database View Restrictions

Ocint recommends implementing database table views based on the sensitivity of the data loaded into the database application, organizational policies, and legal requirements. You can grant access to specific columns or rows to specific users and groups by using database table views. For details, see [Database, Tables, Views, and Indexes](#).

### Data Subject Requests

Ocint recommends using these guides to respond to data subject requests related to the General Data Protection Regulation (GDPR) or California Consumer Privacy Act (CCPA).

## Delete

To remove records from an Ocint System, see [Remove Records from an Ocint System](#).

## Update

To update or rectify records in the Ocint System, see [Database, Tables, Views, and Indexes](#).

## Related Links

[Authentication Methods](#)

[Install an Ocint System](#)

[Load Data](#)

### 6.1.1 Set Up Data Encryption

The Ocint® System enables you to set up data encryption on data drives that support The Computing Group (TCG) Opal Specification and store the encryption secrets in an external Key Management System (KMS) utilizing the Key Management Interoperability Protocol (KMIP). This series of workflows demonstrates generating and storing secrets using the Thales≤≤ CipherTrust Manager KMS.

## Prerequisites

A KMIP server, such as Thales CipherTrust Manager, must be available to store secrets.

- The Ocident node must be a registered client of the KMIP server.
- The client certificate and key file must be on the client, e.g., `/var/opt/ocident/kmip/client.crt` and `/var/opt/ocident/kmip/client.key`.
- The CA certificate must be present on the client, e.g., `/var/opt/ocident/ca.pem`.
- The client must be able to communicate with the KMIP server.

The SEDutil CLI utility must be installed to manage the SED configuration of the drive. The `sedutil-cli` command must be available on the system path variable `PATH`.

Ensure all NVMe drives intended for use with Ocident are:

- Bound to the kernel NVMe driver.
- Have no partition table.
- Be in the Opal factory state.
  - Use `--revertTPer` or `--revertNoErase` SEDutil commands.
- Have these initial settings for the drives and `SIDPassword` set to `admin`:
  - `Locked = N`
  - `LockingEnabled = N`
  - `LockingSupported = Y`
  - `MBRDone = N`
  - `MBREnabled = N`
  - `MBRAbsent = N`
  - `MediaEncrypt = Y`

## Set Up Data Encryption

Use a configuration file and script to generate the secret or key.

1. Generate a configuration file with the path `/var/opt/ocident/ockmip.conf`. The configuration file uses the PyKMIP configuration file format such as this text.

```
[client]
host=thales01.example.com
port=5696
certfile=/var/opt/ocident/kmip/client.crt
keyfile=/var/opt/ocident/kmip/client.key
ca_certs=/var/opt/ocident/kmip/ca.pem
```

2. Run this command.

```
sudo /opt/ocident/utils/ockmip init
```

The script performs these operations for each drive that is detected as unused by the OS (no partition table present or with an Ocident partition):

- Generate a password or key for locking the drive.
- Store the secret password or key on the KMIP server.
- Enable Opal locking on the drive and set the `admin1` password to the generated secret.
- Set the `SIDpassword` to `admin` to enable reverting the drive to the factory state in case the `admin1` password is lost.

To enable the script to run on start before the drives are bound to the `uio` or `vfio` driver by the system `bind-uio-driver.service` installed by the `ocident` package, run this command.

```
sudo /opt/ocident/utils/ockmip install
```

On reboot, the `ockmip` service unlocks the drives before they are bound to the `uio` or `vfio` driver and available for use by the `rolehostd` process.

The full setup sequence is as follows:

1. Configure the `ockmip.conf` configuration file with all of the appropriate settings.
2. Stop the `rolehostd` process if it is running.
3. Unbind drives from the `uio` or `vfio` driver back to the NVMe driver.
4. If Ocident has already been bootstrapped on the node, use the `import-drives` command. Otherwise, use the `init-drives` command.

```
Ocident has been bootstrapped.
ockmip import-drives
```

```
Ocident has not been bootstrapped.
ockmip init-drives
```

5. Set the node role configuration to `useOpal: false` within the `rolehostd.conf` file.
6. Back up any existing keys in `/var/opt/ocient/localKeyStore` in case you encounter any issues.
7. Install the `systemd` service to unlock drives on the reboot.

```
ockmip install
```

8. Optionally, perform a power off and power on to ensure the installation was successful and that the `ockmip.service` successfully unlocks drives on boot.

Migration process output might be similar to this output.

```
[root@ppolllercs00006 ocident]# ockmip -l DEBUG import-drives [2024-09-12 12:45:32,853][INFO] Using conf file
/var/opt/ocient/ockmip.conf [2024-09-12 12:45:33,427][INFO] importing drive(s) [2024-09-12 12:45:33,427][INFO]
Importing secret for /dev/nvme15 BTAX240306A015PFGN ...
```

After reboot, the `ockmip` service output might be similar to this output.

```
[root@dpolllercs0002e log]# systemctl status ockmip ockmip.service - Oneshot service to unlock NVMe drives
via ockmip script using secrets stored in KMIP service. Loaded: loaded (/etc/systemd/system/ockmip.service;
enabled; preset: disabled) Active: inactive (dead) since Wed 2024-09-11 19:11:55 UTC; 31min ago Process: 3217
ExecStart=/usr/local/bin/ockmip unlock (code=exited, status=0/SUCCESS) Main PID: 3217 (code=exited,
status=0/SUCCESS) CPU: 3.097s
Sep 11 19:11:52 dpolllercs0002e ockmip[3217]: [2024-09-11 19:11:52,429][INFO] Retrieving secret for /dev/nvme2
S64GNN0WC04870 ...
```

The `rolehostd` process should start up without the process managing encryption because the `ockmip` service handles it.

## Troubleshooting

For more details on the possible commands, see the help using this command.

```
sudo /opt/ocient/utils/ockmip --help
```

To diagnose issues, examine the log located at `/var/opt/ocient/log/ockmip.log`

To perform actions on a particular drive or set of drives, you can manually specify with the `--drives` parameter.

```
sudo ockmip unlock --drives /dev/nvme1n1 --drives /dev/nvme2n1
```

## NCERRInsufficientPermissions Error

When attempting to register the drives, you see this error.



```
[2024-08-26 17:27:51,350][INFO] Using conf file /var/opt/ocient/ockmip.conf [2024-08-26 17:27:51,917][INFO]
Initializing drive(s) [2024-08-26 17:27:51,917][INFO] Generating and storing secret for /dev/nvme15 [2024-
08-26 17:27:52,089][ERROR] Error registering secret with KMIP server /dev/nvme15 Traceback (most
recent call last): File "/opt/ocient/utlils/test/ocutil.sh.runfiles/xgsrc/python/ocutil/ockmip.py", line 232, in
initDrives self.client.register(data) File "/opt/ocient/utlils/test/ocutil.sh.runfiles/pip_packages_pykmp/site-
packages/kmip/pie/client.py", line 41, in wrapper return function(self, *args, **kwargs)
~~~~~ File "/opt/ocient/utlils/test/ocutil.sh.runfiles/pip_packages_pykmp/site-
packages/kmip/pie/client.py", line 573, in register raise exceptions.KmipOperationFailure(status, reason,
message) kmip.pie.exceptions.KmipOperationFailure: OPERATION_FAILED: GENERAL_FAILURE -
[NCERRInsufficientPermissions]:
```

To fix this, regenerate the certificate on the endpoint of the Thales device (as the root CA has no trust) in the Thales Administrator.

## Disk Replacement

After you swap a disk, initialize it using the `ockmip` utility to add the encryption secret to Thales.

## rockmip.service Status

Check the status after the boot of the system to ensure that all drives have been correctly unlocked.

```
[root@ppollllercs00007 ~]# journalctl -u ockmip -S today Sep 12 12:38:09 ppollllercs00007 systemd[1]: Starting Oneshot service to unlock NVMe drives via ockmip script using secrets stored in KMIP service. ... Sep 12 12:38:10 ppollllercs00007 ockmip[2681]: [2024-09-12 12:38:10,623][INFO] Using conf file /var/opt/ocient/ockmip.conf Sep 12 12:38:11 ppollllercs00007 ockmip[2681]: [2024-09-12 12:38:11,221][INFO] Unlocking drive(s) ...
```

## Timeouts When Contacting Thales

If there is a network outage, or Thales maintenance happens, you might see this output when you attempt to unlock the drives.

```

Sep 12 17:05:59 ppollrcs0000d systemd[1]: Starting Oneshot service to unlock NVMe drives via ockmip
script using secrets stored in KMIP service.... Sep 12 17:06:01 ppollrcs0000d ockmip[3346]: [2024-09-12
17:06:01,040][INFO] Using conf file /var/opt/ocient/ockmip.conf Sep 12 17:06:31 ppollrcs0000d ockmip[3346]:
[2024-09-12 17:06:31,345][ERROR] An error occurred while connecting to appliance cmttnprd5a.t-mobile.com:
timed out Sep 12 17:06:31 ppollrcs0000d ockmip[3346]: [2024-09-12 17:06:31,346][ERROR] could not open client
connection: timed out Sep 12 17:06:31 ppollrcs0000d ockmip[3346]: [2024-09-12 17:06:31,346][ERROR] Unable
to open connection to KMIP server ...

```

## Related Links

# Ocient Security Guide

## Authentication Methods

## 6.2 Manage Users, Groups, and Roles

The Ocient Hyperscale Data Warehouse™ can have one or more databases, each of which can run multiple applications. Administrators have the ability to define and manage users in Ocient® to meet the needs of a multi-tenant environment. Through user, group, and role management capabilities in Ocient, administrators can make sure that users only have privileges to the objects and databases they need. They can also set security settings for passwords. For details, see [Database Password Security Settings](#).

## Users

Users in Ocident are represented by a username and password. They can have privileges to the system or one or more databases within the system. Privileges of a user can be granted directly to the user or inherited through groups or

roles. Users can also belong to zero or many groups and can have zero or many pre-defined roles in the system. A user identifier might be used by someone logging directly into Ocint through the Ocint command-line interface (CLI) or by an application connecting to a database.

### Fully Qualified User Names

A Fully Qualified User Name (FQUN) is the combination of a username and the database on which the user was created. For example: `jsmith@metrics`. It is not mandatory for users to use an FQUN to connect to their own database.

An FQUN uniquely identifies a user and follows the pattern "`<user_name>@<database_name>`", where `<user_name>` is the username and `<database_name>` is the name of the database. Use an FQUN to reference a user associated with a database other than the one you are connected to.

If you do not specify an FQUN when you reference a user object, Ocint assumes the user is associated with the database of the active connection.

This example shows the intended use of the FQUN and how the database is implicitly selected when you connect to the database.

Given an Ocint System with two databases, `operations` and `metrics`, create two administrator users and give administrator rights to each on their respective database.

| FQUN                          | Database                | Password               |
|-------------------------------|-------------------------|------------------------|
| <code>admin@operations</code> | <code>operations</code> | <code>password1</code> |
| <code>admin@metrics</code>    | <code>metrics</code>    | <code>password2</code> |

These are two separate users. Those users can set up their database for the remainder of their users, including:

- Create users with various roles and privileges.
- Set up SSO controls for each database.
- Create tables.
- Create pipelines to load data.
- Create service classes.

It is not mandatory for users to use an FQUN to connect to their database. The first user can connect to the `operations` database as the user `admin` with `password1`. The system treats this login as the user `admin@operations`.

Similarly, the second user can connect to the `metrics` database as the user `admin` with `password2`.

In this example, neither user has to use their FQUN, because they only work with the database described by their FQUN.

When users have permission to connect to more than one database, they are required to use their FQUN for any database that does not match their FQUN. For example, if the user `jsmith@metrics` also has access to the `auctions` database, then the user can connect with their FQUN `jsmith@metrics`.

### Groups

A group is a collection of users that has a set of privileges assigned to it. Administrators can create groups that represent an application or a specific job role for users within the database. For example, a group named `US_Analysts` might only have privileges that allow access to data from the US. Groups belong to individual databases. In other words, you cannot define a group with privileges that span multiple databases.

A service class defines a set of limits on groups. The database applies service classes on a per-group basis. By default, all groups are in the `DEFAULT` service class that has no limits. For details, see [Workload Management and Service Classes](#).

## Roles

Similar to groups, users can inherit privileges by being granted one of the pre-defined roles in Ocint. The system pre-defines the names of and privileges assigned to roles in Ocint. Roles can apply to the Ocint System or one of the user-defined databases.

### System Roles

- **Security Administrator** — Can read, create, and modify all users or groups.
- **System Administrator** — Can read, create, and modify system objects such as tables, clusters, databases, etc. Can also kill running queries system-wide.
- **System Analyst** — Read-only access to the entire system.

You can grant system roles to individual users whether they are members of a specific User Database or defined on `system`.

### Database Roles

- **DB Public** — Can grant and distribute privileges using this `PUBLIC` role. Every user created in the database has this role.
- **Database Administrator** — Can read, create, and modify database objects such as tables, clusters, databases, etc. The database administrator can also create users for the database and kill running queries.
- **Database Analyst** — Read-only rights to database objects and data within the database.

These roles are named with the convention "`<database_name> <role_name>`", where `<database_name>` is the name of the database and `<role_name>` is the name of the role. The `sys.roles` system catalog table displays this information.

To learn about the syntax for granting privileges, see [Data Control Language \(DCL\) Statement Reference](#).

### Use DCL to Assign User Privileges

To assign or revoke privileges to users, administrators can use Data Control Language (DCL) statements. You can execute DCL statements like any other SQL statements against the Ocint System and databases. These statements allow for quick and easy modification of user access. For details, see the [Data Control Language \(DCL\) Statement Reference](#).

### System Database

Every Ocint System includes an administration interface named `system`.

The Ocint System lists `system` as a database, but it is not actually an active database. `system` serves as a connection point for administration, particularly during the initial system setup to configure other databases, storage, user access, and other configurations.

When connected to `system`, users can see metadata for all User Databases to which they have rights. Only users who have System Administrator or System Analyst roles can see metadata for Nodes, Node Service Roles, storage spaces, and storage clusters.

A user can log into `system` if any of these conditions are true:

- You create the user on that database — e.g., `johndoe@system`.
- You assign the user the role of either System Administrator or System Analyst.
- The non-system user (e.g., `myuser@some_database`) has been granted `SYSTEM` privileges through DCL, e.g., `GRANT USE on SYSTEM to USER some_user@some_database;`

### User Databases

User Databases are defined by users. These databases can house tables, views, and data. You can create and scope users to a database to allow multi-tenancy on an Ocint System or limit the ability of users to see the data or schemas defined in other User Databases.

When a user is connected to a user database, they can query the user data in this database and see the metadata for this database in system tables.

### Visibility of Catalog Components Based on User Privileges

All system catalog tables (e.g., `sys.tables`, `sys.users`, `sys.databases`) are queryable to all users of Ocint regardless of their connection. The contents of these tables can change based on the rights that the individual user has. For example, if a user is blocked from a specific database, then a query on the system catalog by that user would not show that database or any of its objects.

### Related Links

[Data Control Language \(DCL\) Statement Reference](#)

[Users, Groups, and Service Classes](#)

[Database Password Security Settings](#)

[SQL Reference](#)

#### 6.2.1 Workload Management Walkthrough

Workload Management (WLM) in the Ocint Hyperscale Data Warehouse™ allows a Database Administrator to set restrictions and priorities for groups of users. The main use case for this is to effectively control multiple workflows across the system, prioritizing or limiting resources to groups when needed. Query restrictions are set and assigned using a service class that can be used for multiple user groups. For a full list of restrictions and their definitions, see the documentation.

This example shows how to set up three service classes with differing priorities, explaining the tradeoffs and design decisions one should make when setting up these restrictions.

#### Create Service Classes

Before creating service classes, the System Administrator or Database Administrator first determines the logical grouping of users on their system and their relative priority. In addition, the response time and concurrency requirements of different user groups are understood. Using these inputs, the administrator can design the service classes that allow their users to receive the proper resource allocations that deliver query requirements.

In this example, define the priority at the service level by creating three service classes to service High, Medium, and Low priority users. Different settings are configured to provide an illustration of the types of configuration that you can use to control features such as concurrency, result sizing, or scheduling priority.

#### Prerequisites

1. An Ocint® System has been set up with an operational SQL Node.
2. A user with the system administrator role has access to the system

#### Step 1: Login as a System Administrator User

In order to create service classes, the user signed in must be assigned the **System Administrator** role. For a full list of user permissions and their definitions, see the [Data Control Language \(DCL\) Statement Reference](#).

First, connect to the system database with the command:

```
connect to jdbc:ocint://my_SQL_ip_address:4050/system;user=admin@system;password=securePassword;
```

#### Step 2: Set up the High Priority User Class

User groups that need fast performance and need access to the most system resources should fall under this service class. Their queries should see minimal impact from outside workloads.

When defining a service class, any service class field that is omitted will be set to the maximum/least restrictive value.

### Example Service Class

```
CREATE SERVICE CLASS high_priority_sc
  scheduling_priority = 100.0,
  max_concurrent_queries = 10,
  max_elapsed_time = 3600;
```

### Example System Catalog View

```
SELECT * FROM sys.service_classes;
```

| id                                   | database_id                          | name             |
|--------------------------------------|--------------------------------------|------------------|
| 42649e8e-ae7f-4de5-bb2b-846d158fcb91 | 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 | Default          |
| 06b92c4c-b0b1-4a14-880c-7d257af3b2b5 | 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 | high_priority_sc |

**scheduling\_priority** — A numeric value that defines the query's priority in the database. A default query being run with no service class has a priority set to 1.0. This priority number is used as an ordered list for scheduling with higher values receiving priority. It is not a weighted value, so only the ordering, not the relative sizes of the priority levels matter. When setting the priority for a high-priority service class, set this value to be greater than 1.0.

**max\_concurrent\_queries** — The number of concurrent queries that can be run at a given time for the entire service class. If the max concurrency is hit, queries will enter a queue. A value of -1 indicates unlimited queries, however it is a best practice to set this value so no individual workloads can monopolize the system.

**max\_elapsed\_time** — The maximum amount of time, in seconds, a query can run. This restriction helps reduce the length of queries holding concurrency slots and allows concurrent workloads using the same service class to all make progress. It is recommended to set **max\_elapsed\_time** if **max\_concurrent\_queries** is set to prevent extremely long running queries from blocking other concurrent queries. If a service class needs to run sets of queries with very different expected elapsed times, it might be an indication that a separate service class is warranted.

### Step 3: Set up the Medium Priority User Class

User groups that do not necessarily need the fastest performance fall under the medium priority service class. Higher-priority queries impact the performance of queries in this service class, but these queries receive priority over lower-priority ad hoc workloads.

### Example Service Class

```
CREATE SERVICE CLASS medium_priority_sc scheduling_priority = 10.0, max_concurrent_queries = 2, max_temp_d
```

### Example System Catalog View

```
SELECT * FROM sys.service_classes;
```

| id                                   | database_id                          | name               |
|--------------------------------------|--------------------------------------|--------------------|
| 42649e8e-ae7f-4de5-bb2b-846d158fcb91 | 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 | Default            |
| 06b92c4c-b0b1-4a14-880c-7d257af3b2b5 | 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 | high_priority_sc   |
| bf3a939d-d773-4c7c-9c30-4bc14e7b9422 | 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 | medium_priority_sc |

**scheduling\_priority** — For a “medium” priority, this value should be set to a number in-between the highest priority service class and the default service class.

**max\_temp\_disk\_usage** — From 0-100, this field represents the maximum percentage of unused disk space that a query can see before failing. When setting up the cluster, it is important to keep some disk space unused on a Foundation Node so queries can spill to disk. Note that this percentage is tracked on a system-wide basis per node, not within a service class. As one service class uses its allocation of temporary disk space to fulfill a query,

it also reduces the total temporary disk available to all other service classes. Even if those service classes have a `max_temp_disk_usage` set, if the total available temp disk space is below the amount needed to run a query, it can fail. For example, if a higher-priority service class is using over 50% of the unused disk space during a medium-priority query that needs to spill to disk, the medium-priority query fails during its first attempt to spill without using any disk space. Spill to disk scheduling uses the same priority used for normal execution.

#### Step 4: Set up the Low Priority Service Class

User groups that are less performance sensitive fall under the low-priority service class. The design goal is for these queries to have minimal effect on other workloads. Examples of users in this service class are analysts who want to run ad hoc queries or processes that run time-insensitive background reports.

#### Example Service Class

```
CREATE SERVICE CLASS low_priority_sc
  scheduling_priority = 1.0,
  max_concurrent_queries = 1,
  max_elapsed_time = 60,
  max_rows_returned = 100,
  max_temp_disk_usage = 0,
  cache_max_bytes = 0,
  cache_max_time = 0,
  max_elapsed_time_for_caching = 0;
```

#### Example System Catalog View

```
SELECT * FROM sys.service_classes;
```

| id                                   | database_id                          | name               |
|--------------------------------------|--------------------------------------|--------------------|
| 42649e8e-ae7f-4de5-bb2b-846d158fcb91 | 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 | Default            |
| 06b92c4c-b0b1-4a14-880c-7d257af3b2b5 | 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 | high_priority_sc   |
| bf3a939d-d773-4c7c-9c30-4bc14e7b9422 | 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 | medium_priority_sc |
| dd1f9f03-edff-4c7e-acd7-32997b6fcea  | 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 | low_priority_sc    |

`scheduling_priority` — For a low priority, this value should be set to a low number like 1.0. If a service class wants to be lower priority than default queries, it can schedule a value lower than 1.0.

`max_temp_disk_usage` — Setting this limitation to 0 means that the system does not allow any queries in the low-priority service class to spill to disk. In this example, the goal of the designer is to prevent the low-priority service class from impacting more important workloads that might need the entire temp disk space. In other cases, it could be suitable to allow low priority service class queries to use temp disk.

`cache_max_bytes`, `cache_max_time`, `max_elapsed_time_for_caching` — Result set caching is a feature that attempts to cache result sets for commonly used queries. This result set cache is shared across service classes. For a lower-priority user, it makes sense to limit the time allowed in the cache or cap the maximum number of bytes used in the cache so a higher-priority service class can take advantage of it. In this example, the result set cache is explicitly disabled for low priority users by setting these values to zero. For details, see the [Result Set Caching](#).

#### Step 5: Assign Service Classes to User Groups

After setting up the service classes, you need to assign them to user groups. When a service class is assigned to user groups, users in each user group can run queries under the restrictions set for the given service class. This process can be repeated for all service classes and users.

#### Examples

```
/* Create a user group. */
CREATE GROUP low_priority_group;

/* Assign the service class to the user group. */
```

```
ALTER GROUP low_priority_group SET service class = low_priority_sc;

/* Add a user to a user group. */
ALTER GROUP low_priority_group ADD USER analyst1;
```

### Dynamic Workload Management Priority Adjustment

When a query executes, you can set the priority of a query at the service class level, database session level, or the query level. The system chooses the maximum value from these three priority levels.

While the engine executes a query, the engine can dynamically adjust the priority using a specified adjustment interval. You can change the percentage amount to dynamically adjust query priority using the `priority_adjustment_factor` (or `ADJUSTFACTOR`) and the time period that indicates the frequency for the adjustment of the query priority during query execution using the `priority_adjustment_time` (or `ADJUSTTIME`). You can override this limit at the query level or the session level.

Define the priority at the service level by creating service classes. You can override the priority at the service level by using the priority at the session or query level. For details, see [Users, Groups, and Service Classes](#).

Define the priority at the session level using the command line. This code instructs the engine to dynamically adjust the priority every 15 seconds and decrease the query priority by 25%.

```
SET PRIORITY "5.0";
SET ADJUSTTIME 15;
SET ADJUSTFACTOR "0.75";
```

To reset the priority adjustment factor, enter this SQL statement.

```
SET ADJUSTFACTOR RESET;
```

Or, define the priority at the query level in the SQL statement, such as this statement.

```
SELECT * FROM table1 USING
  SCHEDULING_PRIORITY = 10.0,
  PRIORITY_ADJUSTMENT_FACTOR = 1.75,
  PRIORITY_ADJUSTMENT_TIME = 120;
```

Alternatively, you can also specify a service class to use at the query level. This example creates the service class `lower_priority` and assigns it to the query.

```
CREATE SERVICE CLASS lower_priority
  SCHEDULING_PRIORITY = 10.0,
  PRIORITY_ADJUSTMENT_FACTOR = 0.75,
  PRIORITY_ADJUSTMENT_TIME = 120;

SELECT * FROM table1 USING SERVICE CLASS lower_priority;
```

If you do not identify a specific service class for a query with the `USING SERVICE CLASS` clause and the query does not match any service classes that use the `STATEMENT_TEXT` parameter, the query uses a composite of least restrictive parameters allowed by the service classes eligible for the query.

Using a service class at the query level requires a user to be assigned to a user group that is included in that specific service class.

### Low-Latency Settings



Low-latency service class settings are intended to reduce runtime overhead for classes of subsecond queries with minimal processing work. Classes of queries that are good candidates utilize indexes on their filters, have low cardinality, and perform minimal work such as simple aggregations or scalar functions. Low-latency service class settings might hinder query performance when queries have high cardinality or perform computationally intense work. Workload management is key to isolating these simple, low-latency queries to specific service classes where you can apply these optimizations in isolation.

Queries for latency-sensitive applications, such as populating live dashboards and key-value retrieval might benefit from service-class level settings that help the Ocint System to make runtime choices to reduce latency. These choices often help queries with low cardinality and minimal computational work, but they can be detrimental to queries that violate these assumptions. Database administrators should use service classes to isolate low-latency components of a system from computationally intensive components.

A system administrator can turn on one simple or two advanced settings.

| Setting          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Value                                                                                   |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| LOW_LATENCY      | This optional service class setting enables you to control the <code>HALF_PARALLELISM</code> and <code>LOAD_BALANCE_SHUFFLE</code> settings. When you set this setting to <code>true</code> , it is equivalent to setting <code>HALF_PARALLELISM = true</code> and <code>LOAD_BALANCE_SHUFFLE = false</code> . When you do not set this setting or set it to <code>false</code> , the Ocint System clears the <code>HALF_PARALLELISM</code> and <code>LOAD_BALANCE_SHUFFLE</code> settings.                                                                                                                                                            | Optionally set to <code>true</code> or <code>false</code> . The default value is unset. |
| HALF_PARALLELISM | This optional service class setting controls execution parallelism overrides. When you do not set this setting, the system uses cost-based heuristics to determine runtime parallelism. When you set this setting to <code>true</code> , the system always uses half parallelism for queries that match this service class. When you set this setting to <code>false</code> , the system always uses full parallelism for queries that match this service class. Half parallelism can help latency by reducing the amount of administrative work the system needs to do to run a query, but it reduces the computational power allocated to the query. | Optionally set to <code>true</code> or <code>false</code> . The default value is unset. |



| Setting              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Value                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| LOAD_BALANCE_SHUFFLE | This optional service class setting controls load balancing shuffles above I/O operators. When you do not set this setting, the Ocient System uses statistics-based heuristics to determine whether to use load balance shuffles. When you set this setting to <b>true</b> , the system always performs a load balance shuffle for queries that match this service class. When you set this setting to <b>false</b> , the system never performs a load balance shuffle for queries that match this service class. Load balance shuffles are useful for distributing uneven data distributions on disk, which can occur during live-loading of tables or small cardinality tables with compute-heavy computations. However, this load balancing incurs some system-coordination overhead, which hampers low-latency queries. | Optionally set to <b>true</b> or <b>false</b> . The default value is unset. |

There is further convenience for organizing these parameters in a service class by enclosing **HALF\_PARALLELISM** or **LOAD\_BALANCE\_SHUFFLE** settings in a **LOW\_LATENCY** key-value setting map.

```
CREATE SERVICE CLASS ll_service_class
  LOW_LATENCY = {HALF_PARALLELISM = true,
                 LOAD_BALANCE_SHUFFLE = true};

SELECT * FROM table1 USING SERVICE CLASS ll_service_class;
```

### Other Service Class Considerations

Service classes are a powerful tool to properly allocate resources in the database to the users that need them most. When deciding on the restrictions for your individual use case, consider the total number of service classes that the system creates. Some settings like **max\_concurrent\_queries** and result set caching values like **cache\_max\_bytes** and **cache\_max\_time** can have a cumulative effect on resource availability as the number of service classes increases. For details about result set caching, see [Result Set Caching](#).

For example, concurrency limitations are set on an individual service class level. While the intention might be to limit low priority service classes to only a couple of concurrent queries, the total effect of many service classes with a couple of concurrent queries each can start to affect high priority workloads due to the number of concurrent queries running across all lower priority service classes.

### Related Links

[Workload Management and Service Classes](#)

[Data Definition Language \(DDL\) Statement Reference](#)

[SQL Reference](#)

### 6.2.2 Workload Management and Service Classes

Service classes define limits on groups in the Ocient® System. You can design service classes using these limits.

#### Service Class Setting Applicability

When a user is present in multiple groups with different service class limits, the system applies the least restrictive limit for each parameter. For example, if User A is in groups G1 and G2, which have service class time limits of 10 seconds and 20 seconds, when User A runs a query, the query has a time limit of 20 seconds. This behavior is slightly different for concurrency. If the service class concurrency limits for the groups the user belongs to are 1 and 2, a query can use a concurrency slot from either service class. A user with multiple service classes can experience a query executing with a slot from one service class but using a time limit or other limits from a different service class. Additionally, the system does not apply service class concurrency limits when a query affects only system tables.

#### Choose Parameter Values

If you set a service class at the session or query level, the system uses the value set by that service class, with the query-level service class taking precedence.

Otherwise, if you set a value for this parameter at the session or query level, the system uses the most restrictive of the two.

If the Ocient System chooses the value from multiple service classes, the system takes the least restrictive value from all service classes. Then, the system takes the most restrictive of that value and the query or session level value, if it exists.

This selection does not hold true for parameters related to result set caching: `cache_max_time` and `cache_max_bytes`. For details about the behavior, see [Result Set Caching](#).

If you choose a service class using statement text matching, the system compares its value to the query or session level value, and the system does not review any other service classes.

#### Next Steps

A tutorial for designing a set of service classes for workload management is available in the [Workload Management Walkthrough](#).

See [Users, Groups, and Service Classes](#) for service-class-related DDL statement references, such as creating, altering, and applying a service class.

#### Related Links

[Data Definition Language \(DDL\) Statement Reference](#)

### 6.2.3 Result Set Caching

The Ocient Hyperscale Data Warehouse™ allows query result sets to be cached on the server and returned quickly on subsequent queries. The service class under which the query executes manages result set caching. As with other service class attributes, you can override this by using keywords on the query.

#### When to Use Result Set Caching

In general, result set caching is most advantageous for queries that require high processing to produce a small result. For example, a query that uses aggregate functions, such as finding the average over many rows, would be a good candidate for result caching.

To help evaluate whether to use result set caching, you can examine your query metadata, such as storage cost and processing time. To view these metrics and to see which are stored in cache, query the `sys.completed_queries` table.

## Query Matching Behavior

Result set caching uses the normalized text of the query (i.e., with comments removed, whitespace reduced, and case normalized) to determine whether two queries match. If they match, the system uses the cached result of a previous query.

Query matching requires strict text matching. Queries cannot use cached results if they have different values for arguments such as time ranges or other filters.

## Configuring Query Caching

You can configure caching behavior for the service class (see [Users, Groups, and Service Classes](#)) or for an individual query (see [General SQL Syntax](#)).

You can specify configurations for result set caching with these attributes:

|                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>cache_max_bytes</code> | Controls whether the database stores the results of any specified query in the cache. All queries executed using a service class specifying this attribute have their results stored in the cache if the size of the result is smaller than this value. You can determine the size of a result set in bytes by querying the <code>bytes_returned</code> field of the <code>completed_queries</code> virtual table. The default value is 0, and thus by default, the database does not cache any results. The database caches results in memory on the SQL Nodes. These results are not cached if there is insufficient memory available.                                                             |
| <code>cache_max_time</code>  | Affects whether a specified query uses results from the cache rather than executing the query. If there is a cached result with the same query text, executed less than <code>cache_max_time</code> seconds ago, the database uses that cached result. The default value is 0, and thus by default, the database does not return any cached results. The cache considers all SQL Nodes, and if a potential cached result is only available on a different SQL Node, the database redirects the query to that node. If you specify the <code>force</code> attribute on the connection, which disables load balancing and redirection, the database considers only cached results on the current node. |

When a specified query uses multiple service classes, the system uses the most restrictive configuration values. For example, if a query uses `service_class_a` with `cache_max_bytes` set to 512 and `service_class_b` with `cache_max_bytes` set to 0, the system does not cache the results. This is because 0 is the most restrictive value, and caching is disabled regardless of the result set size.

You can override the values for `cache_max_bytes` and `cache_max_time` at query time by specifying them in the query itself.

### Query Override Example

```
SELECT COUNT(*) FROM TRADES USING CACHE_MAX_BYTES=32, CACHE_MAX_TIME=60;
```

This example states that the query result is cached if it is less than or equal to 32 bytes, and the cache is used if the value stored in the cache is less than 60 seconds old. It is very important to note that attributes specified on the query cannot increase the values from the service classes, only reduce them.

Both `cache_max_bytes` and `cache_max_time` can be explicitly not specified in a service class by using the value `-1`. `-1` is effectively treated as `0` if not overridden. `unspecified`, or `-1` values can always be overridden by other service classes or on the query. A typical use case is to:

- Specify `cache_max_bytes` as `0` for service classes used by ad hoc queries, where results should never be cached and consume memory.
- Specify `cache_max_bytes` and `cache_max_time` as `-1` on service classes used by repeated production queries. As mentioned, the system treats these as `0` if not overridden.
- Explicitly override `cache_max_bytes` on repeated production queries whose results should be cached.
- Explicitly override `cache_max_time` on repeated production queries whose results can be retrieved from the cache.

Note that because the default schema used for a query can affect the results of the query, the system treats two queries with identical text but making use of different schemas differently.

Finally, the contents of the cache can be examined using the `result_cache` virtual table.

### Max Elapsed Time For Caching

The system cancels queries when they exceed the `max_elapsed_time` service class value. An additional service class value, however, allows queries to continue beyond `max_elapsed_time` so that their results can be stored in the cache. The service class (or SQL) can specify `max_elapsed_time_for_caching`, which should be a value larger than `max_elapsed_time`. This value indicates the total time the query can continue executing such that its results are stored in the cache.

`max_elapsed_time_for_caching` and `max_elapsed_time` are both measured in seconds. When a query with `max_elapsed_time_for_caching` exceeds `max_elapsed_time`, a warning indicating `Query exceeded time limit`, but will continue to run and be cached (Limited to 200 second(s) for caching from query) is returned to the JDBC driver, where `max_elapsed_time_for_caching` was set to 200 seconds in this example.

`max_elapsed_time_for_caching` is only supported on queries executed using the JDBC driver.

### Manage the Result Set Cache

You can store and retrieve query results from the cache depending on the settings of the `cache_max_bytes` and `cache_max_time` values, either using the SQL statement or the service class specified for the query execution.

To view all cached queries and their service classes, query the `sys.result_cache` system catalog table.

You can invalidate the result set cache using these DDL statements. Use these statements when you perform table updates, such as an `INSERT INTO SQL` statement operation or the execution of one of the loading and transformation functions.

#### Syntax

Remove the cached result sets for all queries that reference the specified table.

You can invalidate the cached result sets for only tables where you have the `INSERT` privilege. If a cached query references multiple tables, you only need the `INSERT` privilege on any one of those tables.

```
ALTER TABLE table_name INVALIDATE CACHE [SPREAD_OVER_TIME seconds]
```

| Parameter               | Data Type | Description                                                                                             |
|-------------------------|-----------|---------------------------------------------------------------------------------------------------------|
| <code>table_name</code> | string    | Name of the database table.                                                                             |
| <code>seconds</code>    | numeric   | Number of seconds that specifies the interval at which to invalidate a section of the result set cache. |

#### Example

This example immediately invalidates the cache and removes the cached result sets for all queries that reference the `employees` table.

```
ALTER TABLE employees INVALIDATE CACHE;
```

This example invalidates the cache entries for queries that reference the `employees` table for a fraction of the cache every second. Specify the fraction to be 1/10 by using the `spread_over_time` parameter. The database removes 1/10 of the cached result set immediately, then the next 1/10 of the cached result set the subsequent second, and so on. Invalidation causes the re-execution of queries to delay. Spreading the invalidation over time prevents spikes of expensive query executions.

```
ALTER TABLE employees INVALIDATE CACHE SPREAD_OVER_TIME 10;
```

### Syntax

Remove the cached result sets in the specified database.

```
ALTER DATABASE database_name INVALIDATE CACHE SPREAD_OVER_TIME seconds]
```

| Parameter                  | Data Type | Description                                                                                             |
|----------------------------|-----------|---------------------------------------------------------------------------------------------------------|
| <code>database_name</code> | string    | Name of the database.                                                                                   |
| <code>seconds</code>       | numeric   | Number of seconds that specifies the interval at which to invalidate a section of the result set cache. |

### Example

This example immediately invalidates the cache and removes the cached result sets in the `test_database` database.

```
ALTER DATABASE test_database INVALIDATE CACHE;
```

This example invalidates the cache entries in the `test_database` database for a fraction of the cache every second. Specify the fraction to be 1/10 by using the `spread_over_time` parameter. The database removes 1/10 of the cached result set immediately, then the next 1/10 of the cached result set the subsequent second, and so on. Invalidation causes the re-execution of queries to be delayed. Spreading the invalidation over time prevents spikes of expensive query executions.

```
ALTER DATABASE test_database INVALIDATE CACHE SPREAD_OVER_TIME 10;
```

### Related Links

[Data Definition Language \(DDL\) Statement Reference](#)

[Query Ocint](#)

#### 6.2.4 Object-Type Level Privileges Management

You can assign privileges at different levels of granularity on system and database objects to control access for users in the Ocint® System. The levels are:

- All objects of the specified object type in the system
- All objects of the specified object type in the database
- Specific object

This series of workflows shows how you can manage access to the `VIEW` privilege for all views (where a view is the object type) in a specified database.

### Grant Privileges

This workflow creates a group and grants the `VIEW` privilege to all views and the ability to read the views in the database to the group.

Create the `view_analyst` group. You can see the created group using the `sys.groups` system catalog table.

```
CREATE GROUP "view_analyst";
```

Grant the VIEW privilege for views on the test database to the view\_analyst group.

```
GRANT VIEW VIEW ON DATABASE "test" TO GROUP "view_analyst";
```

Grant the SELECT privilege on the database to the group.

```
GRANT SELECT ON DATABASE "test" TO GROUP "view_analyst";
```

View the granted privileges using the sys.privileges system catalog table. The Ocint System grants the USE privilege on the database implicitly.

```
SELECT privilege, privilege_target, object_type,
       grantee, grantor
FROM sys.privileges
WHERE grantee = 'view_analyst' and grantor != 'view_analyst';
```

Output

| privilege | privilege_target | object_type |
|-----------|------------------|-------------|
| VIEW      | VIEW             | DATABASE    |
| USE       | NULL             | DATABASE    |
| SELECT    | NULL             | DATABASE    |

Fetches 3 rows

### Verify Privileges

This workflow creates a user, assigns this user to the created group, and creates a view with generated data. The workflow shows the information that the user has access to in the database based on the granted privileges.

Create the test\_user user.

```
CREATE USER test_user PASSWORD='**';
```

Add the user to the view\_analyst group.

```
ALTER GROUP "view_analyst" ADD USER test_user;
```

Create the view view that selects five generated rows.

```
CREATE VIEW "test"."view" AS (SELECT * FROM sys.dummy5);
```

Connect to the database at the 10.10.1.1 IP address with port number 4050 as the test\_user user.

```
connect to jdbc:ocint://10.10.1.1:4050/test;user=test_user;password=**;
```

View the privileges of the user using the sys.privileges system catalog table. This user has no privileges of their own.

```
SELECT privilege, privilege_target, object_type, grantee, grantor
FROM sys.privileges
WHERE grantee = 'test_user' AND grantor != 'test_user';
```

Output

| privilege              | privilege_target | object_type |
|------------------------|------------------|-------------|
| Fetches 0 rows         |                  |             |
| Warning: No data found |                  |             |

See the metadata for views that the user has access to within this database. The user has access to the `test` view only.

```
SHOW VIEWS;
```

*Output*

| table_catalog | table_schema | table_name |
|---------------|--------------|------------|
| test          | test         | view       |

Fetches 1 row

Verify that the user can select the data in the `view` view.

```
SELECT * FROM "test"."view";
```

*Output*

```
c1
-----
1
2
3
4
5
Fetches 5 rows
```

## Related Links

[Data Control Language \(DCL\) Statement Reference](#)

[System Catalog](#)

[Generate Tables Using sys.dummy](#)

## 6.3 Authentication Methods

Ocient® offers two authentication methods to access the database: Password authentication and Single Sign On (SSO) authentication. You can use these independently or as an organization according to your security and access control needs. This section explains each authentication method and how it can be configured on Ocient.

### Password Based Authentication

Users created using DCL are required to set a password that the system uses for authentication when connecting to the database.

Administrators can use [Database Password Security Settings](#) to manage settings such as minimum password length and password expiration policies.

### Fully Qualified User Name (FQUN)

To connect to the database, the user must provide a FQUN. For database users, the FQUN has this form.

```
<user_name>@<database>
```

For example, the FQUN of the user `alice`, a member of the database `example_database`.

```
alice@example_database
```

## How To Connect using Password Authentication

The FQUN should be set in the connection string along with the password of the user. For example, this code is the command `alice` would use to connect to `example_database` using the Ocint JDBC driver.

```
connect to jdbc:ocient://<host>:<port>/example_database;user=alice@example_database;password=****;
```

The following Ocint driver properties must be set when connecting using password authentication:

- user — `alice@example_database`
- password — `****`

## Single Sign On (SSO)

The Ocint System allows administrators to add a Single Sign On integration, allowing users to authenticate using an external Identity Provider (idP). A database, including the “system” database, can have 0 or 1 SSO integrations.

The existence of an SSO integration has no effect on users who authenticate with the Password Authentication flow.

## Supported Protocols:

The Ocint System supports the following SSO protocols:

| Protocol       | Description                                                                                                                                                                                                                                                                                                                                                                                |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OpenID Connect | OpenID Connect (OIDC) is a simple JSON-/REST-based identity protocol built on top of the OAuth 2.0 protocol. It enables client applications to rely on the authentication performed by an OpenID Connect Provider to verify the identity of a user. Clients can also obtain basic profile information about a user in an interoperable and REST-like manner from OpenID Connect Providers. |

## OpenID Connect

There are two flows that allow users to connect to the database using an external OpenID Connect Provider.

### Authorization Code Flow

This is the most commonly used authentication method and makes use of the [OAuth 2.0 Authorization Code Grant](#).

When connecting to the database, users are directed (using Ocint driver) to the OpenID Provider’s authorization endpoint where they are able to login with their SSO credentials. After the user has been authenticated, the Provider redirects the user back to the database with a temporary code, which the database exchanges for an ID token, Access token, and optionally, a Refresh token.

See this image for a sequence diagram of the flow.

### Connect using the Authorization Code Flow

The client handshake method must be set to SSO, and the connection username and password set to the empty string.

For example, this code is the command any user would use to connect to the `example_database` database using the Ocint JDBC driver.

```
connect to jdbc:ocient://<host>:<port>/example_database;handshake=SSO;user=;password=;
```

These Ocint driver properties must be set when connecting using the Authorization Code flow:



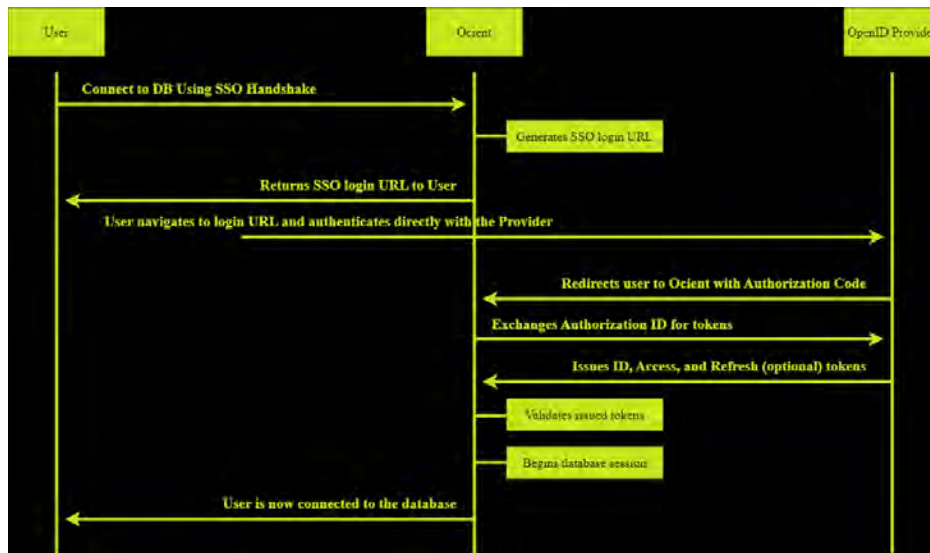


Figure 4: Sequence of authentication flow

- handshake — SSO
- user — `<empty_string>`
- password — `<empty_string>`

The `user` and `password` properties are required and must be set to the empty string.

### SSO Token Flow

Users can provide either an [ID Token](#) or an [Access Token](#) issued by the OpenID Provider to connect to the database.

The provided ID or Access token must contain these claims:

- iss — The issuer must match the issuer contained in the discovery document of the OpenID Provider.
- aud — The token audience must contain the `client_id` used when configuring the OpenID Provider in the database.

Ocint recommends including the `email` scope in any token provided to Ocint. The system uses the value of the email claim to identify the user in the database audit trails.

Ocint supports these signing algorithms for ID Tokens:

- RSA 256
- RSA 384
- RSA 512

### Connect using the SSO Token Flow

These Ocint driver properties must be set when connecting using an OpenID token:

- handshake — SSO
- user — `id_token` or `access_token`
- password — `<token_data>`

The handshake must be set to SSO, with the username set to the token type and the password set to the token payload.

For example, this format is the JDBC command to connect to the `example_database` database using an ID Token.

```
connect to jdbc:ociant://<host>:<port>/example_database;handshake=SSO;user=id_token;password=<id_token_data>
```

This format is the JDBC command to connect to `example_database` database using an Access Token.

```
connect to jdbc:ociant://<host>:<port>/example_database;handshake=SSO;user=access_token;password=<access_token_data>
```

## Configure the OpenID Provider

Before users can authenticate using SSO, you must register the database with the OpenID Provider. This process varies depending on the provider, but typically, the steps for doing so are:

1. Create an application for the Ocient System in the Provider. Select the “Native Application” application type.
2. Enable the Authorization Code, Refresh Token, and Device Authorization grant types.
3. Grant permission to the appropriate users to use the newly created application.
4. Enter this Redirect URI.

| Application   | Redirect URI                                                 |
|---------------|--------------------------------------------------------------|
| JDBC Driver   | <code>http://localhost:7050/ociant/oauth2/v1/callback</code> |
| OpenAPI or UI | <code>https://&lt;advertised_ip&gt;/v1/callback</code>       |

For the OpenAPI or UI application, the redirect URI must use `https` format.

The connectivity pool defines the OpenAPI or WebUI `advertized_ip` setting. For more information, see [Cluster and Node Management](#).

Record the token `issuer` and `client_id` of the newly created application. You need these values to configure the Ocient System.

## SSO Parameters

These SSO parameters are configurable by using DDL statements.

After all parameters have been retrieved, a database administrator can execute a DDL SQL statement using this syntax.

- The user must be a database administrator (or have the “Security Administrator” role for the “system” database).
- The user must be connected to the database.
- Any String key or value can be placed between double quotations (e.g., `default_group` OR `"default_group"`).
- String values that contain characters other than (`[a-zA-z]` | `[0-9]` | `'_'`) must be placed between double quotations (e.g., `this.is.a.complex-$string ≤ "this.is.a.complex-$string"`).
- The NULL value can be used to clear existing configurations of LIST or MAP properties (e.g., `user_claim_ids = NULL`).

To remove the connection, execute this SQL statement.

```
ALTER DATABASE example_database REMOVE SSO INTEGRATION;
```

The OpenID Connect protocol requires TLS for communication between the end user, the database, and the OpenID Provider. A valid TLS key and certificate pair must be configured on the SQL Nodes, which behave as a server during the Authorization Code Flow.

## Set Up Machine-To-Machine SSO Integration

You can use DDL SQL statements to add client credentials for SSO connection to individual databases or to a connectivity pool that operates across SQL Nodes.

These steps set up SSO client credentials on a database but can also apply to connectivity pools.

### Create an SSO protocol.

Use the [Cluster and Node Management](#) SQL statement to make an SSO integration protocol with your preferred configuration. This configuration must include these properties:

- The `enable_id_token_authentication` SSO property must be set to `true`.
- The `additional_audiences` SSO property must use the URL from the OIDC provider, similar to the `issuer` value.

```
CREATE SSO INTEGRATION sso_test PROTOCOL oidc
  issuer = "https://accounts.google.com",
  client_id = example_database_app_id,
  default_group = example_database_group,
  enable_id_token_authentication = true,
  additional_audiences = ["https://accounts.google.com"];
```

### Assign the SSO protocol.

Use the [Database, Tables, Views, and Indexes](#) SQL statement to integrate the SSO protocol into a database. For connectivity pools, use the [Cluster and Node Management](#) SQL statement.

```
ALTER DATABASE example_database
  SET SSO INTEGRATION sso_test;
```

### Connect to your system.

Use a JDBC connection string that includes the access token value `<access_token>`. This string connects to the local host at port number 4050, uses the SSO protocol, and with the username `id_token`.

```
CONNECT TO jdbc:ocient://localhost:4050/system;handshake=SSO;user=id_token;password=<access_token>
```

## User Access Control and Workload Management with SSO

The database maintains an internal User Access Control model consisting of groups, roles, and service classes. The database relies on group or role membership to determine the privileges of a user or service class. The following integration properties allow administrators to grant SSO users membership to a Database group/role:

- `groups_claim_ids` and `roles_claim_ids` define the token claim(s) that specify the user's Provider-defined groups/roles.
- `groups_claim_mappings` and `roles_claim_mappings` define mappings between a Provider and Database groups/roles.
- `additional_scopes` should include any request scopes needed for the Provider to include the group/role claims in tokens it issues.

Consider an example where the Provider lists all groups the user belongs to in the token claim `"user_groups"` when the application requests the `"example_scope"` scope. In this scenario, the administrator wants to associate the Provider Group `"Example Provider Group"` with the Database Role `"example_database Analyst"`.

```
ALTER DATABASE example_database
  ALTER SSO INTEGRATION oidc

  /* Tell the Database to request the "profile", "email" and "offline_access" scopes */
  additional_scopes = [ profile, email, offline_access ],

  /* Tell the Database to look for the "user_groups" token claim to establish role membership */
  roles_claim_ids = [ user_groups ],
```

```
/* Map provider group "Example Provider Group" to database role "example_database Analyst" */
roles_claim_mappings = { "Example Provider Group" = "example_database Analyst" };
```

By default, the database grants all users membership to the “Public Role” of the database for the current connection. Unlike other default roles, you can grant additional privileges to the “Public Role”.

Additionally, database administrators can revoke access to users with membership to any of the specified Provider groups or roles. To revoke access, execute this statement.

```
ALTER DATABASE example_database
  ALTER SSO INTEGRATION oidc
    blocked_groups = [ <provider_group_name> [, ...] ],
    blocked_roles  = [ <provider_role_name>  [, ...] ];
```

The user must have the ALTER privilege on all Ocident-defined groups or roles specified by groups\_claim\_mappings, roles\_claim\_mappings, blocked\_groups, and blocked\_roles.

### User Claim IDs

By default, audit trails, including but not limited to system-level log messages, identify users connecting using SSO by the email claim of the ID Token. If the email claim is not present in the token, the identifier is iss and sub ID Token claims separated by ::, for example: "https://ocient.okta.com::00u5rs1ndgXm9Ey7y5d7".

To change the FQUN, alter the user\_claim\_ids SSO integration property.

For example, to change the FQUN to a combination of the first\_name and last\_name claim values, execute this SQL statement.

```
ALTER DATABASE example_database
  ALTER SSO INTEGRATION oidc
    user_claim_ids = [first_name, last_name, email];
```

After executing this statement, users are identified with the <first\_name>::<last\_name>::<email> format. (For example: John::Smith::jsmith@example.com)

Updating the user\_claim\_ids property has no effect on users that are already connected to the database.

### System Catalog Tables

Properties that do not contain sensitive data can be viewed using the sys.oidc\_integrations virtual table. In addition to the configurable properties, a database\_id column is included in the schema.

To view the configuration for a specific database (if one exists), execute this SQL statement.

```
SELECT *
  FROM sys.databases
 JOIN sys.oidc_integrations
 ON databases.id = oidc_integrations.database_id
 WHERE databases.name = 'example_database';
```

The user must be a database administrator to view the OpenID Connect properties.

## Related Links

[Data Definition Language \(DDL\) Statement Reference](#)

[SQL Reference](#)

[System Catalog](#)

### 6.3.1 Database Password Security Settings

You can manage password security in the Ociant Hyperscale Data Warehouse™ by using a variety of settings and managing user states.

#### Password Security Settings

Ociant® includes password security settings to meet specific security requirements or preferences. Five settings are available to configure your password security.

| Security Setting                            | Description                                                                                                                                                                                                                                                                                                                          |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>password_minimum_length</code>        | The minimum length of passwords. The maximum password length supported by the Ociant System is 32.                                                                                                                                                                                                                                   |
| <code>password_complexity_level</code>      | An integer value representing the password complexity algorithm. Supported values are: 1—Level 1. The password must contain at least one upper case character, lower case character, and number. 2—Level 2. In addition to the requirements specified in level 1, the password must contain at least one non-alphanumeric character. |
| <code>password_no_repeat_count</code>       | The number of unique passwords that a user must use before they can reuse a password. Even if this setting is set to 0, when the system determines that the password lifetime has been reached, you must change the password to a unique value.                                                                                      |
| <code>password_lifetime_days</code>         | The password must be changed after this number of days. After the password is older than this period, the user changes to the <code>PASSWORD_EXPIRED</code> state on their next login.                                                                                                                                               |
| <code>password_invalid_attempt_limit</code> | The number of login attempts with an invalid password before a user changes to the <code>DISABLED</code> state.                                                                                                                                                                                                                      |

Except for the `password_no_repeat_count` setting, a value of 0 for any of these settings means that the system ignores that setting.

#### Password Security Setting Hierarchy and Precedence

You can set all these settings at the system, database, or group levels. The Ociant System uses the most restrictive value. For example, if `password_minimum_length` is 8 at the system level, 10 at the database level, and 12 at the group level, the system applies the value 12 to the user.

You can only add a user to a group after you create the user in the system. Upon user creation, `password_minimum_length`, `password_complexity_level`, and `password_no_repeat_count` settings are based only on system- and database-level settings.

## System Catalog Table for Security Settings

The `sys.security_settings` system catalog table shows current security settings. The table contains settings for only databases and groups if any settings are non-zero.

After setting the `password_invalid_attempt_limit` value, you can inspect the value using the `sys.security_settings` system catalog table.

```
SELECT password_invalid_attempt_limit FROM sys.security_settings;
```

The `sys.users` system catalog table contains information about users, their current state, and details about their security status, such as the last time the password was updated or the number of failed login attempts.

## User States

Ocient local users, not SSO-based users, are in one of these states:

- ENABLED
- DISABLED
- PASSWORD\_EXPIRED

### ENABLED State

Enabled users have normal access privileges to the system. They can connect and execute SQL statements. This state is the default state for all users.

### DISABLED State

A disabled user cannot connect to the system, and if they are currently connected, they cannot execute any SQL statements. Users become disabled automatically if they exceed the `password_invalid` number of failed password attempts.

An administrator must set the state of the user to `ENABLED` for the user to resume access.

### PASSWORD\_EXPIRED State

This state enables a user to log on. However, the only SQL statement they can execute is `ALTER USER <example_user> SET PASSWORD='yyyy'`. The user receives a warning that their password is expired when they authenticate.

Whenever the `password_lifetime_days` number of days has elapsed after the last time the password changed, the system automatically transitions a user to the `PASSWORD_EXPIRED` state on their next login.

You can inspect the last password change timestamp in the `password_updated_at` column in the `sys.users` system catalog table.

### Change User State

To change the state of the user to these different states, use the [Users, Groups, and Service Classes](#) SQL statement.

## Related Links

[Authentication Methods](#)

[Data Control Language \(DCL\) Statement Reference](#)

[Users, Groups, and Service Classes](#)

[Cluster and Node Management](#)

[Database, Tables, Views, and Indexes](#)

[Users, Groups, and Service Classes](#)

## 6.4 Schema Design

### 6.4.1 TimeKeys and Clustering Keys

Segment keys define how an Ocident® System partitions data on disk based on particular columns in a database. These keys allow the system to quickly filter rows and skip irrelevant records for queries involving those columns.

For this reason, segment keys are integral tools for optimizing query performance for pivotal columns. Although segment keys are optional, deploying them on every database table in your system is a good practice because they require practically no additional processing or storage resources.

Ocident supports two types of segment keys:

- **Segment Keys** — A segment key that partitions data based on a single time-series data column.
- **Segment Keys** — A series of columns that are frequently queried together. The system subdivides these segments on disk for faster reference.

Together, these segment keys form the basis for the Ocident time-cluster storage engine, which subdivides data and metadata for optimal performance.

#### Attributes of Segment Keys

##### Advantages

Ocident segment keys share these important advantages:

- Improved performance for query filtering and record retrieval.
- Minimal resource demand for storage or I/O.
- Easy to use. Without sophisticated system knowledge, you can create segment keys with basic data definition language (DDL) statements.

##### Limitations

There are no significant downsides to using segment keys. However, they have some limitations for deployment:

- You can implement segment keys only upon table creation as part of the initial [Database, Tables, Views, and Indexes](#) statement. You cannot add them later to a table.
- Each database table can support only one key of each type. For faster performance on additional columns, you can also add clustering indexes and other secondary indexes. For details, see [TimeKeys and Clustering Keys](#) and [Secondary Indexes](#).
- Segment keys are restricted to specific column data types.

#### TimeKeys

TimeKeys partition data based on a single time-series column in a database table. This column should normally be either a DATE or TIMESTAMP data type.

For optimal performance, you can configure a TimeKey column with two arguments that comprise a time bucket. The time bucket is a unit of time combined with a number value (e.g., `HOURL, 2` or `DAY, 1`). This value sets the granularity for the partitioning.

In general, this value should match the level of filtering that you expect your queries to request. For example, a database table you expect to query for transactions by minute should probably have a small TimeKey bucket value, such as an hour. If you expect your queries to filter by broader time values, your table can have a TimeKey bucket of one day or larger.

The Ocident System uses the TimeKey and the bucket granularity to partition rows into segments that incorporate a minimum and maximum time value for all rows. For queries that involve time filtering, the TimeKey allows an Ocident System to immediately skip data segments that are outside the query bounds without any I/O processing.

This syntax example creates a single TimeKey column and must be included in the column definition of a `CREATE TABLE` SQL statement. For the full syntax, see [Database, Tables, Views, and Indexes](#).

For DDL examples of tables that use TimeKey columns, see [CREATE TABLE SQL Statement Examples](#).

## Clustering Key

Every table in an Ocident System can support a Clustering Key to speed up query operations on a series of adjacent columns.

The Clustering Key determines how data is ordered into segments on disk. This ordering acts like an index, enabling an Ocident System to quickly locate any relevant rows of the Clustering Key columns.

Every database table should define one Clustering Key on a sequence of frequently queried columns for best results.

Clustering Key columns can be of any fixed-length data type except for `DECIMAL`. Variable-length columns can also support Clustering Keys if they are configured with global dictionary compression. For details, see [Table Compression Options](#).

In addition to the Clustering Key, you can also define clustering indexes. Unlike the Clustering Key, clustering indexes do not involve ordering data into segments. They create separate indexes for each specified column set, which boosts performance but requires some additional storage.

This syntax example creates a Clustering Key across multiple columns and must be included in a `CREATE TABLE` SQL statement. For the full syntax, see [Database, Tables, Views, and Indexes](#).

### Select Clustering Key Columns

You can define a Clustering Key and any additional clustering indexes as arguments of a [Database, Tables, Views, and Indexes](#) SQL statement.

The column ordering of a clustering key is significant. You must specify columns in rank order.

### Table Example

This `CREATE TABLE` SQL statement example shows the definition of a clustering key on columns (CK1, CK2, CK3). The data types of these columns are irrelevant to this example.

```
CREATE TABLE example_table (
    ...,
    CK1 INT,
    CK2 INT,
    CK3 INT,
    ...,
    CLUSTERING KEY ck_example (CK1, CK2, CK3),
);
```

The Clustering Key primes the records for search and filter operations on its columns in descending order of importance. This means the first column CK1 is the most important of the subset while the third CK3 is the least important.

To boost performance with a Clustering Key, a query must make references to the columns following this key order.

For a Clustering Key in the (CK1, CK2, CK3) pattern, these sequences of query column references are effective:

- CK1
- (CK1,CK2)
- (CK1,CK2,CK3)

### Query Example

This example `SELECT` query follows the key order for its filter operations.

```
SELECT * FROM example_table (
    WHERE CK1 > 1
    AND CK2 = 0
    AND CK3 < 0;
```



For a Clustering Key in the (CK1, CK2, CK3) pattern, these sequences of query column references are ineffective:

- (CK1, CK3)
- (CK2, CK3)
- (CK3, CK2)

To improve performance on queries that reference columns in these sequences, you can use clustering indexes. For details, see [TimeKeys and Clustering Keys](#).

To check the order of column references in a query, you can test a query using an [Query Analysis SQL](#) statement. This statement shows the I/O pipeline and how the system references indexes.

For DDL examples of tables that use Clustering Key columns, see [CREATE TABLE SQL Statement Examples](#).

### Add Clustering Indexes

A table can only have one Clustering Key, but it can also have any number of clustering indexes to boost performance. Clustering indexes act similarly to the Clustering Key, but they do not order data into segments. Instead, each clustering index creates a separate index for the specified column set, which can impact storage and loading overhead.

Clustering indexes can provide flexibility to your tables, allowing your queries to reference the same columns as your Clustering Key, but in different sequences.

You must define clustering keys and clustering indexes when you first create a table. You cannot add them at a later time.

### Example

Define a Clustering Key across the first four columns (CK1, CK2, CK3, CK4) in the `example_table` table in this `CREATE TABLE SQL` statement.

The statement also includes two clustering indexes using the `INDEX` definition. The first creates an index for a subset of just two columns (CK2, CK3) from the Clustering Key. The second index uses all of the key columns (CK2, CK4, CK1, CK3), but in a different sequence.

```
CREATE TABLE example_table (
  CK1 INT NOT NULL,
  CK2 INT NOT NULL,
  CK3 INT NOT NULL,
  CK4 INT NOT NULL,
  COL5 INT NOT NULL,
  CLUSTERING KEY ck_example (CK1, CK2, CK3, CK4),
  INDEX idx_01 (CK2, CK3),
  INDEX idx_02 (CK2, CK4, CK1, CK3)
);
```

In the example, `COL5` is not part of the Clustering Key, meaning this column cannot be included in any clustering indexes.

### Permissions

To create TimeKeys and Clustering Keys, you must have the `CREATE` privilege for the current database.

### Observability

You can query these system catalog tables to get information on the effectiveness of your configurations for segment keys and indexes:

- `sys.indexes` — Includes information on the index identifier, type, and other information.
- `sys.index_columns` — Contains information on indexes by their column position.
- `sys.segment_parts` — Contains data on the segment parts associated with each segment key.

For examples of querying segment and index metadata tables, see [Discover Insights From System Catalog Tables](#).

## Related Links

[Ingest Data with Legacy LAT Reference](#)

[Query Ocident](#)

[Database, Tables, Views, and Indexes](#)

[Secondary Indexes](#)

## Related Videos

At the Whiteboard with Ocident: Indexing at Hyperscale

### 6.4.2 Secondary Indexes

Secondary indexes are extremely important for optimizing database queries. When used effectively, indexes allow the database to skip irrelevant records when reading data from storage. On large datasets, this can have a big impact, turning a query that would take minutes to run into one that finishes in seconds or less.

The tradeoff for using secondary indexes is they do require additional disk space and loading resources. It is generally a good idea to be strategic about which columns you index. A column is usually a good candidate for an index if you expect it to be frequently referenced in queries, especially for searching and filtering (e.g., in a `WHERE` or `ON` clause of a SQL statement).

Creating an index does not trigger the re-indexing of existing segments. Only segments that are generated after the execution of the `CREATE INDEX` statement contain the new index. For that reason, create indexes early on before you load large amounts of data into a table.

You can create indexes on columns containing various different data types as long as you meet the requirements in this table. Depending on the data type, the Ocident® System can assign different index types by default if you decline to specify this.

### Index Type Requirements and Defaults

This table describes index types that are assigned by default to columns based on the data type. Container data types have special requirements for using indexes.

Unless you explicitly specify an index type, the data type of a column determines a default index type for the system to create.

| Data type                                                                                                 | Requirement                                                                                                      | Default Index Type (if not specified) |
|-----------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|---------------------------------------|
| INT, BIGINT, SMALLINT, FLOAT, DOUBLE, BINARY, DECIMAL, HASH, CHAR, IP / IPV4, TIME, TIMESTAMP, DATE, UUID | None                                                                                                             | INVERTED                              |
| VARCHAR                                                                                                   | None                                                                                                             | HASH                                  |
| TUPLE                                                                                                     | Inner data types are fixed length or VARCHAR                                                                     | HASH                                  |
| ARRAY                                                                                                     | Inner data types are fixed length or VARCHAR.orInner datatype is a tuple containing only fixed-length data types | HASH                                  |

| Data type                  | Requirement | Default Index Type (if not specified) |
|----------------------------|-------------|---------------------------------------|
| POINT, LINESTRING, POLYGON | None        | SPATIAL                               |

You can apply indexes regardless of whether they have GDC compression.

CREATE INDEX on tuples currently supports indexing only on individual tuple component columns and not entire tuples.

### Column Requirements for Index Optimization

All index types can speed up queries that involve filter expressions on their designated columns.

An index takes effect when its specified column is queried with a filter expression of either `<column_name> <op> <literal>` or `<literal> <op> <column_name>`.

The column and operator must meet one of these requirements.

| Operator                                                                                                                                | Requirements                                    |
|-----------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| <code>=, !=, &gt;, &lt;, &lt;=, &gt;=</code> , or <code>BETWEEN</code> .                                                                | The column must have a non-container data type. |
| <code>IS NULL</code> or <code>IS NOT NULL</code>                                                                                        | Column must be nullable.                        |
| <code>LIKE</code> , <code>NOT LIKE</code> , <code>=</code> , <code>!=</code> , <code>SIMILAR TO</code> , or <code>NOT SIMILAR TO</code> | Index type is NGRAM.                            |

### Array operator requirements

Additionally, columns with arrays can use indexes if any of these criteria are met.

| Operator or Function                                                                                                                                                                                                           | Syntax Example                                                                                                                                                        |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| The operator is <i>contains</i> or <i>not contains</i> , i.e., <code>@&gt;</code> .                                                                                                                                            | <code>&lt;filter_expression&gt; = [NOT] (&lt;column_name&gt; @&gt; &lt;scalar_literal&gt;   &lt;array_literal&gt;)</code>                                             |
| The operator is <i>is contained</i> or <i>not is contained by</i> , i.e., <code>&lt;@</code> .                                                                                                                                 | <code>&lt;filter_expression&gt; = [NOT] (&lt;scalar_literal&gt;   &lt;array_literal&gt; &lt;@ &lt;column_name&gt;)</code>                                             |
| The operator is <i>overlap</i> or <i>not overlap</i> , i.e., <code>&amp;&amp;</code> .                                                                                                                                         | <code>&lt;filter_expression&gt; = [NOT] (&lt;column_name&gt; &amp;&amp; &lt;array_literal&gt;)   (&lt;array_literal&gt; &amp;&amp; &lt;column_name&gt;)</code>        |
| The filter expression uses the <code>for_some</code> or <code>for_all</code> function. The operator is <code>=</code> , <code>!=</code> , <code>&gt;</code> , <code>&lt;</code> , <code>&lt;=</code> , or <code>&gt;=</code> . | <code>&lt;filter_expression&gt; = [NOT] (for_all(&lt;column_name&gt;) &lt;op&gt; &lt;literal&gt;)   (for_some(&lt;column_name&gt;) &lt;op&gt; &lt;literal&gt;)</code> |

`BETWEEN` is not supported for `for_some` or `for_all` operators with filter expressions. If the index type is NGRAM, the operator can also be `LIKE`, `NOT LIKE`, `SIMILAR TO`, or `NOT SIMILAR TO`.

For syntax information for secondary indexes, see the [Database, Tables, Views, and Indexes](#) reference section.

### Secondary Index Types

Ocient supports four index types alongside the clustering index: `INVERTED`, `HASH`, `NGRAM`, and `SPATIAL`. An index notionally stores a mapping of the column value to the rows that contain that value, and the index type differentiates the format in which the column values are stored and accessed.

For container data types (e.g., arrays and tuples), the index stores the internal elements of the container and is used on predicates that target the internal values. However, a mapping of NULL column values is generally stored for both scalar and container data types, so the index can always be used for `column IS NULL` predicates.

#### INVERTED Index Type

An **INVERTED** index supports fixed-length column types. The value stored internally in the index is the whole column value, and its storage size is the same as the width of the data type.

This index allows for exact lookups from a column value to rows and supports strict equality comparisons and range comparisons (e.g., `<`, `>`, `>=`, `\<=`, `BETWEEN`). The index supports **EQUALS** predicates that perform exact matches on array values (e.g., `CONTAINS`, `OVERLAPS`, and `for_all()` and `for_some()` functions that use **EQUALS** or **IS NULL**), and range comparisons on array values using `for_all()` and `for_some()`.

The system does not use the inverted index for all range comparisons on an index column. Instead, the system determines in advance when it is more performant to use the inverted index for range comparisons. For example, ranges that cover a smaller number of values are more likely to utilize the index.

| Data Type                                                                                                                                                                                                                                                                                                                         | Function                                                                                                             | Supported Operators                                                                                                                       | Array Operators and Supported Functions          |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|
| Fixed-length columns <code>INT</code> , <code>BIGINT</code> , <code>SMALLINT</code> , <code>FLOAT</code> , <code>DOUBLE</code> , <code>BINARY</code> , <code>DECIMAL</code> , <code>HASH</code> , <code>CHAR</code> , <code>IP / IPV4</code> , <code>TIME</code> , <code>TIMESTAMP</code> , <code>DATE</code> , <code>UUID</code> | Supports range and exact matches. The system stores the whole column value (storage is the same as data-type width). | <code>=</code> , <code>!=</code> , <code>&lt;</code> , <code>&gt;</code> , <code>&gt;=</code> , <code>&lt;=</code> , <code>BETWEEN</code> | <code>for_all()</code> , <code>for_some()</code> |

#### HASH Index Type

The system supports a **HASH** index on variable-length column types, which stores a hash of the indexed column value rather than the full value. You can use **HASH** indexes for exact comparisons, and they support equality and `CONTAINS` or `OVERLAPS` filters. The system supports range operators, such as `<` and `>`, and uses sorting logic to determine how `VARCHAR` types compare to each other.

The internal implementation of a **HASH** index requires additional record filtering, making the **HASH** index more computationally intensive than the **INVERTED** index. The additional filtering is attributable to the **HASH** index being a probabilistic index type; because hash collisions are possible, values returned by the index might include false positives that the database must filter internally during query execution to remove invalid results.

Do not use the **HASH** index for variable-length geospatial types, including `LINESTRING` and `POLYGON`. For better results, use the **SPATIAL** index instead.

In most cases, a **HASH** index on a `VARCHAR` column requires less storage than an **NGRAM** index.

| Data Type                                                             | Function                                                                          | Supported Operators                                                                                                                       | Array Operators and Supported Functions                                                                              |
|-----------------------------------------------------------------------|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| Variable-length columns <code>VARCHAR</code> , <code>VARBINARY</code> | Supports range and exact matches. The system stores a hash of the indexed column. | <code>=</code> , <code>!=</code> , <code>&lt;</code> , <code>&gt;</code> , <code>&gt;=</code> , <code>&lt;=</code> , <code>BETWEEN</code> | <code>@&gt;</code> , <code>&lt;@</code> , <code>&amp;&amp;</code> , <code>for_all()</code> , <code>for_some()</code> |

#### NGRAM Index Type

The Ocint System supports an **NGRAM** index on variable-length column types. An **NGRAM** index stores substrings of size `n` of the column values. For example, using the default `n_value` size of 3, you can divide the column value

Ocient into the 3-grams { “Oci”, “cie”, “ien”, “ent” }. The system stores these 3-grams separately in the index and can find these individually.

The NGRAM index type helps optimize queries that involve either exact or partial matching filters. When the system uses an N-gram index with a filter string, the database divides the filter value into N-grams in the same manner as the column values, and the system finds each filter N-gram in the index.

NGRAM index results are probabilistic because the rows matched by the index can include strings that contain the right N-grams but do not satisfy the filter predicate. As a result, the Ocient query engine must filter column values returned from the index lookup for valid matches. The database uses the index to speed up string similarity where possible, but the index does not support all filters. At a minimum, a filter literal must contain at least one sequence of characters of the `n_value` length that must be present in the resulting column value. For instance, LIKE and SIMILAR TO expressions that are composed entirely of wildcards cannot use the NGRAM index.

For an example of partial matching, see this query on a table that has an NGRAM(3) index on the `example_column` column.

```
SELECT example_column FROM example_table WHERE example_column LIKE 'Ocie%';
```

The filter string 'Ocie%' results in the filter 3-grams { “Oci”, “cie” }. The database finds these 3-grams in the index and returns any rows that contain those substrings, for example, “Ocient”, “The Ocient database”, and “cieOci”. Of these strings, only “Ocient” satisfies the LIKE filter. The database filters out false positives and returns the valid rows.

For an exact matching example, see this query on the same table with the NGRAM(3) index.

```
SELECT example_column FROM example_table WHERE example_column = 'Ocie%';
```

In this case, the database uses similar logic by finding the substrings { “Oci”, “cie”, “ien” } in the index. The database returns only strings that exactly match the input string. While NGRAM can significantly increase performance on equality matches, it is generally slower than a HASH index.

| Data Type                                        | Function                                                                                                           | Supported Operators                        | Array Operators and Supported Functions      |
|--------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|--------------------------------------------|----------------------------------------------|
| Variable-length columns<br>VARCHAR,<br>VARBINARY | Supports exact and partial matches. System stores substrings of <code>then_value</code> size of the column values. | =,!=,<,>,>=,\<=, BETWEEN, LIKE, SIMILAR TO | ESCAPE, FOR, FOR_ALL, FOR_SOME, FOR_SOME_ALL |

### SPATIAL Index Type

The SPATIAL index type helps expedite queries on geographic data types, including POINT, LINESTRING, and POLYGON values. In a table with a column of these objects, a SPATIAL index finds rows quickly using a geographic literal filter.

This index groups geographic objects together by using a packed Hilbert R-tree algorithm. The index allows queries to quickly search through rows by traversing only a subsection of the full data set.

For array types, the SPATIAL index is limited to a maximum of 1 million items per row.



```

totalpackets
FROM ip_data
WHERE starttime BETWEEN '<start_time>' AND '<end_time>'
      AND ipaddress = '<ip_address>'
LIMIT 10;

```

The crucial column is the `ipaddress`, as it is the crux for the exact matching used in this query.

If data storage is not a problem, this query is ideal for an `INVERTED` index. This index supports exact matching for the IPv4 data type of the `ipaddress` column.

Create an inverted index on this column using this SQL statement.

```
CREATE INDEX destination_ipaddress ON ip_data (ipaddress) USING INVERTED;
```

#### Optimize a Range Query Using an `INVERTED` Index

Similar to the exact match query, this query also includes a range filter for the numeric value in the `totalpackets` column.

```

SELECT starttime,
      CHAR(destinationipaddress),
      sourceport,
      destinationport,
      totalpackets
FROM ip_data
WHERE flowstarttime between '<start_time>' AND '<end_time>'
      AND ipaddress = '<ip_address>'
      AND totalpackets between 200 and 400
LIMIT 10;

```

An `INVERTED` index is the best choice for this use case. This index supports range filtering for all fixed-length types, including the `BIGINT` values in the `totalpackets` column.

Create an `INVERTED` index on this column using this SQL statement.

```
CREATE INDEX totalpackets ON ip_data (totalpackets) USING INVERTED;
```

#### Optimize an Exact String Match Using a `HASH` Index

In this case, filter all records where customers accessed the exact web address `ocient.com`.

```

SELECT starttime,
      CHAR(sourceipaddress),
      CHAR(ipaddress),
      sourceport,
      destinationport,
      httprequesturl
FROM ip_data
WHERE starttime between '<start_time>' AND '<end_time>'
      AND httprequesturl = 'ocient.com'
LIMIT 10;

```

If you expect to filter the `httprequesturl` column only with exact matches, the `HASH` index is usually your best choice. The `HASH` index can greatly speed up exact matching on variable-length data types, and it requires less storage than an `NGRAM` index except in rare edge cases.

```
CREATE INDEX httprequesturl ON ip_data (httprequesturl) USING HASH;
```

#### Optimize a Partial String Match Using `NGRAM` Index

For this use case, filter all records where customers accessed a partial or exact web address. The LIKE clause in this query template filters all records for those customers who accessed the Facebook URL.

```
SELECT starttime,
       CHAR(sourceipaddress),
       CHAR(ipaddress),
       sourceport,
       destinationport,
       httprequesturl
FROM   ip_data
WHERE  starttime BETWEEN '<start_time>' AND '<end_time>'
      AND httprequesturl LIKE '%facebook.com%'
LIMIT 10;
```

An NGRAM index is the best choice for this use case. This index supports partial matching for variable-length data types, including the VARCHAR data type of the `httprequesturl` column.

Create an NGRAM index on this column using this SQL statement.

```
CREATE INDEX httprequesturl ON ip_data (httprequesturl) USING NGRAM(3);
```

In this example, the NGRAM index divides the specified column into substrings of three characters, which is the default. This means partial strings such as “Fac”, “ace”, “boo” and other variants are captured in the filter for this query.

You can specify a broader index by setting a higher substring value, such as NGRAM(4) or NGRAM(5). Using one of these values would reduce the data storage requirements.

#### Optimize a Match Query Using a SPATIAL Index

For this use case, filter all records where customers are located in a specific area at a specific time.

This query represents the target area by the bounding box POLYGON((1 2,1 3,1 4,1 2)).

```
SELECT starttime,
       CHAR(sourceipaddress),
       CHAR(ipaddress),
       sourceport,
       destinationport,
       httprequesturl,
       location
FROM   ip_data
WHERE  starttime BETWEEN '<start_time>' AND '<end_time>'
      AND ST_CONTAINS(location, POLYGON((1 2,1 3,1 4,1 2)))
LIMIT 10;
```

A SPATIAL index is the best choice for this use case because it supports a variety of functions for filtering geospatial data for bounding box areas.

Create a SPATIAL index on the `location` column by using this SQL statement.

```
CREATE INDEX location ON ip_data (location) USING SPATIAL;
```

#### Related Links

[Ingest Data with Legacy LAT Reference](#)

[Query Ocint](#)

[Database, Tables, Views, and Indexes](#)

[Database, Tables, Views, and Indexes](#)



## Related Videos

At the Whiteboard with Ociant: Indexing at Hyperscale

### 6.4.3 Table Compression Options

The Ociant® System provides a range of options for compressing tables to reduce storage requirements for your data. Compression can be a major component of query optimization when used effectively by reducing I/O requirements and potentially allowing more system space for indexes.

#### Ociant Compression Key Concepts

Table columns in Ociant can support these compression options:

- [Table Compression Options](#)
- [Table Compression Options](#)
- [Table Compression Options](#)

Ociant System compression adheres to these key concepts:

- By default, the Ociant dynamic compression scheme can determine whether to apply compression on each column if it is statistically advantageous. Sometimes, you might want to use another compression scheme like ZSTD or completely disable compression.
- You can specify custom compression schemes on individual columns through data definition language (DDL) commands, either using a `CREATE TABLE` or `ALTER TABLE` SQL statement.
- GDC can stack onto other compression schemes.

#### General Compression Matrix

This table shows the best compression scheme for different column data types. The suggested best compression schemes consider the column data type and cardinality, but other factors can influence which data compression makes sense.

The default compression in the table shows the compression scheme that the database applies if you do not specify an option. This default compression balances performance and compression.

| Data Type                                           | Default Compression | Best Compression                   |
|-----------------------------------------------------|---------------------|------------------------------------|
| Fixed-length                                        | COMPRESSION DYNAMIC | COMPRESSION ZSTD                   |
| Variable-length and low cardinality                 | COMPRESSION NONE    | COMPRESSION GDCandCOMPRESSION ZSTD |
| Variable-length and high cardinality                | COMPRESSION NONE    | COMPRESSION ZSTD                   |
| Arrays                                              | COMPRESSION NONE    | COMPRESSION ZSTD                   |
| Arrays of variable-length types and low cardinality | COMPRESSION NONE    | COMPRESSION GDCandCOMPRESSION ZSTD |

| Data Type | Default Compression                                | Best Compression                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------|----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tuples    | Compression depends on each tuple index data type. | The database treats each tuple index position as its own column value and can support its own compression scheme. Fixed-length values default to <code>COMPRESSION DYNAMIC</code> , while variable-length values default to <code>COMPRESSION NONE</code> . For best performance, apply rules for each tuple index based on the data-type rules outlined in this table. For example: <code>my_tuple TUPLE&lt;&lt;INT, VARCHAR(255) COMPRESSION GDC(1) COMPRESSION ZSTD, VARCHAR(255)&gt;&gt;</code> This example defines a tuple column with GDC and ZSTD compression only for the first inner <code>VARCHAR</code> value. |

For details about the advantages and disadvantages of each compression scheme, see the `compression_scheme` description in [Database, Tables, Views, and Indexes](#).

For details about the compression syntax, see [Database, Tables, Views, and Indexes](#).

### Global Dictionary Compression

Column parameter: `COMPRESSION GDC`

GDC uses a dictionary encoder to map variable-length column data. Instead of storing the variable length data directly on disk, GDC substitutes each unique string with a corresponding unique integer.

This compression scheme can significantly reduce overall storage requirements for low-cardinality columns with a few distinct values while boosting performance for various operations.

GDC supports an optional `compression_value` argument that specifies how many unique values to use.

#### Advantages

- Reduced disk usage — Variable-length data such as strings or container types can experience massive storage reduction.
- Improved query performance — By mapping each row to an integer, variable-length rows no longer require a full traversal for selective I/O operations, which improves performance in some queries.
- Faster Joins — Join comparisons have improved performance from using GDC integers instead of the variable-length data.
- Variable-Length Cluster Keys — Variable-length columns can use a clustering key index for improved performance.
- Stackable — Columns can use GDC in tandem with other compression schemes.

#### Disadvantages

- Increased complexity — You must know the rough cardinality of your data to size the GDC configuration properly.
- Load complexity — GDC compression adds some overhead to loading data into the Ocident System.
- Not suitable for high-cardinality data — GDC performance decreases as the data cardinality increases.
- Cannot apply retroactively — Unlike other compression options, you must set GDC as part of the initial `CREATE TABLE SQL` statement. You cannot apply it later to the table.

For details, see [Global Dictionary Compression](#).

```
column_name [data_type] COMPRESSION GDC [ (compression_value) ]
```

For details about adding GDC compression to a table schema, see [Database, Tables, Views, and Indexes](#).

### Zstandard Compression

Column parameter: `COMPRESSION ZSTD`

ZSTD uses the ZSTD library to apply a fast and lossless compression algorithm to any column type, including fixed and variable-length scalars, arrays, and tuples on a per-component basis. In addition, ZSTD works on columns where the database first compresses variable-length data by GDC and then compresses the resulting integral type on disk.

This compression scheme can affect performance by requiring additional processing. Therefore, this compression scheme is usually good for less-frequently accessed columns or columns paired with an index.

ZSTD supports these optional values:

- `COMPRESSION_LEVEL` — This value signifies how much compression the data receives. The default value is 0. The full range of values is from -7 (low compression) to 15 (high compression). The database requires less memory when this value is lower, whereas it requires more memory and processing resources when the value is larger.
- `DICTIONARY_SIZE` — This value is the dictionary size that you specify as a positive integer. The value signifies the size of the shared compression dictionary in bytes. The default value is 32768 (32 KB). The full range of values ranges from 4096 (4 KB) to 1048576 (1 MB). This value denotes the amount of memory consumed during segment generation. In general, larger values provide better compression but use more memory.

The default values for `COMPRESSION_LEVEL` and `DICTIONARY_SIZE` are sufficient for most cases.

### Advantages

- Reduced disk storage — Dramatically decreases disk space usage.
- Flexible — You can apply this compression scheme after table creation using an [Database, Tables, Views, and Indexes](#) SQL statement.

### Disadvantages

- Performance loss — ZSTD does require some extra CPU resources, which means that it can impact query performance.
- Memory usage — ZSTD can demand significant memory usage when many columns use it for parallel query workloads.

```
column_name [data_type] COMPRESSION ZSTD
[COMPRESSION_LEVEL = level_value, DICTIONARY_SIZE = size_value]
```

For details about adding ZSTD compression to a table schema, see [Database, Tables, Views, and Indexes](#).

### Dynamic Compression

Column parameter: `COMPRESSION DYNAMIC`

By default, Ocint tables use dynamic compression to reduce storage space requirements automatically. This method employs a variety of compression schemes on columns based on their data type and a statistical determination of what kinds of compression would be advantageous.

- For fixed-length data, block-size units (typically 4 KB) can receive various compression schemes, including run-length encoding and delta-delta encoding.
- For variable-length data (VARCHAR and VARBINARY), the database compresses data row-by-row, with values individually compressed using LZ4 compression. This method produces a good compression ratio for large values. However, small or medium-sized values have negligible compression rates.

## Advantages

- **Ease of use** — The Ociant System determines the compression.
- **Flexibility** — You can apply this compression scheme after table creation using an [Database, Tables, Views, and Indexes](#) SQL statement.

## Disadvantages

- **Secondary option** — Dynamic compression is designed for balance, but it is generally not as effective at compression as ZSTD.
- **Performance loss** — This compression scheme has a small performance impact, but is not as costly as ZSTD.

```
column_name [data_type] COMPRESSION DYNAMIC
```

For details about adding dynamic compression to a table definition, see [Database, Tables, Views, and Indexes](#).

For examples of tables that use compression, see [CREATE TABLE SQL Statement Examples](#).

## Related Links

[Global Dictionary Compression](#)

[Secondary Indexes](#)

### 6.4.3.1 Global Dictionary Compression

Global Dictionary Compression (GDC) is a feature of the database that compresses variable length column data using a dictionary encoder. Instead of storing the variable length data directly on disk, GDC seamlessly substitutes an integer that corresponds to each unique string.

#### Advantages of GDC

1. **Reduced disk usage** — A string like "GDC is really cool!" takes up 19 raw bytes. With GDC, it takes at most 4 bytes. This benefit is multiplied by the number of rows in the table and, when applied to arrays, can lead to a dramatic reduction in required storage.
2. **Fixed-width** — Variable length column data can be more difficult for performing selective I/O. It is much faster for Ociant® to read the 100th element in a list of integers by offsetting 100 times the fixed width of the integer. With variable-length types, a full traversal is required, counting how many elements are passed until the 100th is found. This can improve performance in some queries.
3. **Faster Joins** — When joining on a GDC column, equality comparisons can be performed on the GDC integers instead of the variable-length data. This can improve performance in some queries.
4. **Allows Variable-Length Cluster Keys** — As of version 19.0 of the Ociant System, GDC is the only method by which a variable-length column can be used as a cluster key index.

#### Disadvantages of GDC

1. **Increased complexity** — GDC does add some configuration complexity. It requires a user to know the rough cardinality of their data to size the GDC number of bytes and adds system configuration options.
2. **Load complexity** — GDC compression adds some overhead to loading data into the Ociant System.
3. **Not suitable for high-cardinality data** — If the variable length column has higher cardinality than this, it should not be stored using GDC.

#### Using GDC

To create a GDC column, add the `COMPRESSION GDC(int)` parameter after the column type specifier when creating a column, where `int` is one of 1, 2, or 4. The integer specifies the number of bytes to be used for the integer keys stored on the disk. This also corresponds to the maximum number of unique keys to be stored in a particular column.

| GDC(int) | Maximum Number of Unique Values |
|----------|---------------------------------|
| GDC(1)   | 256                             |
| GDC(2)   | 65,536                          |

| GDC(int) | Maximum Number of Unique Values |
|----------|---------------------------------|
| GDC(4)   | 4,294,967,296                   |

There is a *soft* limit of 1,000,000 keys even on 4-byte integer GDC columns. Contact Ocient Support to evaluate your criteria for changing the limit and to understand the impact of raising limits. See [Global Dictionary Compression](#) for details.

### Applying GDC to a Column

To enable GDC on a column, the keyword is applied in the CREATE TABLE or ALTER TABLE ADD COLUMN statement. An example of GDC on a VARCHAR column:

```
{column_name} VARCHAR(255) COMPRESSION GDC(2)
```

### Reusing an Existing GDC Map

A user can also specify that a column should share GDC space with another column—possibly even a column of a different table. This would be useful if the same data is used in multiple columns and it is often joined in queries. Use COMPRESSION GDC EXISTING `schema.table.column_name` as shown here:

```
{column_name} VARCHAR(255) GDC EXISTING {schema.table.column_name}
```

### GDC on Array Columns

For arrays of variable-length data, GDC operates on the individual elements of the array. Specify compression after the overall array type:

```
{column_name} VARCHAR(255) [] COMPRESSION GDC(2)
```

### GDC on Tuple Columns

Elements of tuple columns can be compressed with GDC. Specify compression on the specific type to be compressed:

```
{column_name} tuple<int, VARCHAR(255) COMPRESSION GDC(2)>
```

### GDC in the System Catalog

You can use the system catalog to inspect the number of keys used by different columns and the maximum count on each.

#### Example Query:

```
SELECT
  t.id as table_id,
  t.name as table_name,
  c.id as column_id,
  c.name as column_name,
  g.compressed_size,
  g.current_count,
  g.max_count
FROM sys.tables t
INNER JOIN sys.global_map_table_info g
  ON t.id = g.table_id
INNER JOIN sys.columns c
  ON c.id = g.column_id;
```

### Truncation

When a table with GDC is partially truncated, *GDC key mappings for removed rows are not removed*. This can result in stale mappings. The only way to remove unwanted mappings is to drop the column and recreate it.

### GDC Column Representation in the System Catalog

To the end user, a table with GDC columns looks like any other table. However, When creating a table with GDC columns, the table configuration is different in system catalog tables. Instead of a table named `schema.tablename`, GDC tables leverage a built-in view.

This view and several pieces of metadata will be created in the system catalog:

1. A view named `schema.tablename` is added in `sys.views`. This view is the representation of the user of the GDC table, and it automatically converts the GDC keys to the loaded variable-length data. This is the table that the user interacts with for querying data, making alterations, and granting or revoking access.
2. For each GDC column in the table, a table named `syslookup.schema.tablename_columnname` is added in `sys.tables`. These tables store the mappings from strings to integers for each column. These tables should be interacted with only rarely.
3. A table called `sysgdc.schema.tablename` is added in `sys.tables`. This is the table that is stored to disk, including all the non-GDC columns and the integers for each GDC column. This table should be interacted with only rarely.

#### GDC Columns and Loading into the Database

GDC stores key mappings in a Raft consensus log that is maintained by all nodes operating with the Admin role. These mappings are created as the data is loaded, and used in queries to fetch the correct variable length data for the stored integers. Mappings can also be created by certain queries if the key does not already exist.

Because the Admin roles are responsible for maintaining the GDC key mappings, there must be a consensus of Admin roles available on the system in order to load new data that includes GDC columns.

Related Links

System Catalog

#### 6.4.4 CREATE TABLE SQL Statement Examples

Tables in the Ocident Hyperscale Data Warehouse™ contain various configuration options that can impact performance, reduce storage requirements, or provide extra data safeguards.

As some table configuration options cannot be added after data is already loaded, it is important that you take the time to design `CREATE TABLE` SQL statements with forethought for the queries that you expect to execute frequently. This topic provides examples of `CREATE TABLE` SQL statements designed for different use cases and describes the configuration choices.

For syntax and parameter information, see [Database, Tables, Views, and Indexes](#).

#### Create a Table with All Data Types

This example creates a table with all supported data types in Ocident®. The intent of this example is to show the required formatting for different data type values, which is shown as the `DEFAULT` value for each column.

```
CREATE TABLE data_type_example (
  col_bigint BIGINT NOT NULL DEFAULT 9876543210,
  col_binary BINARY(3) NOT NULL DEFAULT '0xabcdef',
  col_boolean BOOL NOT NULL DEFAULT TRUE,
  col_char CHAR(4) NOT NULL DEFAULT 'val',
  col_date DATE NOT NULL DEFAULT '2000-01-01',
  col_decimal DECIMAL(18,4) NOT NULL DEFAULT 123.45,
  col_double DOUBLE NOT NULL DEFAULT 3.141592,
  col_float FLOAT NOT NULL DEFAULT 2.718,
  col_int INT NOT NULL DEFAULT 123456789,
  col_int_array INT[] NOT NULL DEFAULT 'INT[0,1,2,3]',
  col_ipv4 IPV4 NOT NULL DEFAULT '127.0.0.1',
  col_ip IP NOT NULL DEFAULT '0123:4567:89ab:cdef:0123:4567:89ab:cdef',
  col_smallint SMALLINT NOT NULL DEFAULT 32767,
  col_matrix MATRIX[2][3] NOT NULL DEFAULT '{ {0, 0, 0}, {0, 0, 0} }',
```

```

col_point POINT NOT NULL DEFAULT 'POINT(0 0)',
col_linestring LINESTRING NOT NULL DEFAULT 'LINESTRING(0 0, 1 1)',
col_polygon POLYGON NOT NULL DEFAULT 'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))',
col_time TIME NOT NULL DEFAULT '12:34:56.012345678',
col_timestamp TIMESTAMP NOT NULL DEFAULT '2000-01-02 12:34:45',
col_tinyint TINYINT NOT NULL DEFAULT 127,
col_tuple TUPLE<<INT, INT>> NOT NULL DEFAULT 'tuple<<INT,INT>>(0,0)',
col_array_of_tuples TUPLE<<INT, INT>>[] NOT NULL DEFAULT 'TUPLE<<INT,INT>>[tuple<<INT,INT>>(1,2),tuple<<INT,INT>>(2,1)]',
col_uuid UUID NOT NULL DEFAULT '01234567-89ab-cdef-1357-0123456789ab',
col_varbinary VARBINARY NOT NULL DEFAULT '0xaabbccddeeff',
col_varchar VARCHAR NOT NULL DEFAULT 'This is a variable length string'
);

```

### Create a Table for Real-Time Analysis

This example assumes you plan to ingest large quantities of time series data for rapid querying. The most important columns that you plan to reference in queries frequently include:

- `created_at` — A granular timestamp column that represents the primary time reference in the table.
- `user_id` — An integer column to identify each unique user.
- `sell_id` — An integer column representing each transaction by a user.

```

CREATE TABLE IF NOT EXISTS "transact_data" (
  "created_at" TIMESTAMP TIME KEY BUCKET(1, HOUR) NOT NULL,
  "user_id" BIGINT NOT NULL DEFAULT 0,
  "sell_id" BIGINT NOT NULL,
  "purchase_amount" DOUBLE NOT NULL DEFAULT 0,
  "buyer_name" VARCHAR,
  CLUSTERING KEY "ck" ("sell_id", "user_id")
)

CREATE INDEX "purchase_idx" ON "transact_data" ("purchase_amount");
CREATE INDEX "buyer_idx" ON "transact_data" ("buyer_name")
  USING NGRAM(4);

```

This example table makes use of these Ocient System configurations.

| Column Name                  | Configuration Description                                                                                                                                                                                                                                                                              |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>created_at</code>      | This column is the table <code>{{TimeKey}}</code> , which means that the database indexes for filter optimization. The example configures the TimeKey to partition segments by every hour, <code>(1, HOUR)</code> , which is the approximate time granularity you expect to use in your query filters. |
| <code>user_id,sell_id</code> | The example table identifies these two columns as the table Clustering Key. The database pairs these two columns for quick reference, such as filtering one column for row values of the other.                                                                                                        |
| <code>purchase_amount</code> | The <code>CREATE INDEX SQL</code> statement adds a secondary index to this column. As a <code>BIGINT</code> data type, the <code>purchase_amount</code> column defaults to the <code>INVERTED</code> index.                                                                                            |



| Column Name | Configuration Description                                                                                                                                                                                                                                                                        |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| buyer_name  | This <code>CREATE INDEX</code> SQL statement adds a secondary index to this column. As a <code>VARCHAR</code> column, the <code>buyer_name</code> column normally defaults to a <code>HASH</code> index, but this example specifies it should be an <code>NGRAM</code> index of width 4 instead. |

For details about using the TimeKey and Clustering Key functionality, see [TimeKeys and Clustering Keys](#).

For details about other index types, see [Secondary Indexes](#).

### Create a Table for Geospatial Data

This example assumes you plan to ingest geospatial data linked to time series data using OcidentGeo™. The most important columns that you plan to reference frequently in queries include:

- `route_time` — A granular timestamp column that represents the primary time reference in the table.
- `start_point`, `end_point` — The geospatial point values for the start and destination locations.
- `trip_id` — An integer column to identify each unique trip.
- `vehicle_id` — An integer column that represents each vehicle.

```
CREATE TABLE "trip_details" (
  "route_time" TIMESTAMP TIME KEY BUCKET(14, DAY) NOT NULL,
  "start_point" POINT NULL,
  "end_point" POINT NOT NULL DEFAULT 'POINT(0 0)',
  "route_path" LINESTRING NOT NULL
    DEFAULT 'LINESTRING(0 0, 1 1)',
  "route_area" POLYGON NOT NULL
    DEFAULT 'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))',
  "trip_id" BIGINT NOT NULL,
  "vehicle_id" BIGINT NOT NULL,
  CLUSTERING KEY "ck" ("trip_id", "vehicle_id")
);

CREATE INDEX "start_idx" on "trip_details" ("start_point")
  USING SPATIAL;
CREATE INDEX "end_idx" on "trip_details" ("end_point")
  USING SPATIAL;
```

This example table makes use of these Ocident System configurations.

| Column Name                       | Configuration Description                                                                                                                                                                                                                                                       |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| route_time                        | This column is the table TimeKey, which means that the database indexes for filter optimization. The example configures the TimeKey to partition segments by every two weeks (14, DAY), which is the approximate time granularity that you expect to use in your query filters. |
| trip_id, vehicle_id               | The example table identifies these two columns as the table Clustering Key. The database pairs these two columns for quick reference, such as filtering one column for row values of the other.                                                                                 |
| end_point, route_path, route_area | These columns have <code>POINT</code> , <code>LINESTRING</code> , and <code>POLYGON</code> data types, respectively. Each of these geospatial columns includes default values in their respective formats.                                                                      |



| Column Name           | Configuration Description                                                                                                                                                                                    |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| start_point,end_point | The <code>CREATE INDEX</code> SQL statements for both of these columns assign <code>SPATIAL</code> indexes for quicker reference. This index is the default type for any column with a geospatial data type. |

For details about OcintGeo functionality, see [Geospatial Functions](#).

### Create a Table With Compression Options

This example table includes multiple compression options to ensure minimal storage for specific columns and extra performance.

Most data types except arrays have `COMPRESSION DYNAMIC` applied by default.

```
CREATE TABLE "call_records" (
  "utc_timestamp" TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  "sectorid" VARCHAR
    COMPRESSION ZSTD,
  "address" VARCHAR
    COMPRESSION NONE,
  "call_type" VARCHAR
    COMPRESSION ZSTD
    COMPRESSION GDC(4),
  "access_information" VARCHAR(500)
    COMPRESSION GDC(4),
  "file_name" VARCHAR
    COMPRESSION ZSTD compression_level = 3,
    dictionary_size = 1048576,
  CLUSTERING KEY "ck" ("call_type", "access_information")
)
```

This example table makes use of these Ocint System configurations.

| Column name                  | Configuration Description                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| address                      | This column explicitly removes compression by specifying <code>COMPRESSION NONE</code> in the <code>CREATE TABLE</code> SQL statement.                                                                                                                                                                                                                                                                                                                |
| call_type,access_information | These columns have <code>VARCHAR</code> data types configured with Global Dictionary Compression (GDC). This compression scheme maps the variable-length type rows to integers. Among other benefits, this compression allows both columns to support a Clustering Key for faster querying. The <code>call_type</code> column stacks multiple compression schemes by including both GDC and ZSTD compression to provide additional storage reduction. |
| file_name                    | This column uses ZSTD compression with extra options. The <code>compression_level</code> is set for 3, meaning the column data is slightly more compressed than the default 0. The <code>dictionary_size</code> is set for the maximum value, 1048576, which means greater compression but higher memory demand.                                                                                                                                      |

For details about Ociant compression schemes, see [Table Compression Options](#).

### Create a Table for Various Container Types

This example table includes complex container data types, including a two-dimensional array, a tuple with variable-length values, and a tuple with various data types.

```
CREATE TABLE "container_data" (
  "utc_timestamp" TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  "user_id" BIGINT NOT NULL DEFAULT 0,
  "sell_id" BIGINT NOT NULL DEFAULT 0,
  "point_array_col" POINT[],
  "array_col" VARCHAR[] COMPRESSION GDC(4),
  "2D_array_col" INT[][],
  "tuple_col1" TUPLE<<
    VARCHAR(255),
    VARCHAR(500)
  >>,
  "tuple_col2" TUPLE<<
    INT,
    INT[],
    VARCHAR COMPRESSION GDC(1) COMPRESSION ZSTD,
    VARCHAR
  >>,
  "matrix_col" MATRIX[5][10],
  CLUSTERING INDEX "ck" ("user_id", "sell_id")
)

CREATE INDEX "point_idx" on "container_data" ("point_array_col")
  USING SPATIAL;
CREATE INDEX "tuple_idx" on "container_data" ("tuple_col1"[1])
  USING HASH;
CREATE INDEX "tuple_idx2" on "container_data" ("tuple_col2"[2])
  USING INVERTED;
```

This example table makes use of these Ociant System configurations.

| Column name     | Configuration Description                                                                                                                                                                                                                                                                                                                                                               |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| point_array_col | This column is an array of <code>POINT</code> values that is indexed using a <code>SPATIAL</code> index type as specified by the <code>CREATE INDEX SQL</code> statement.                                                                                                                                                                                                               |
| tuple_col1      | This tuple column contains two variable-length values. Each object in the tuple can support its own compression scheme and index. The <code>CREATE INDEX SQL</code> statement assigns a <code>HASH</code> index to the first value ( <code>VARCHAR(255)</code> ).                                                                                                                       |
| tuple_col2      | This tuple column contains various data types. Each object in the tuple can support its own compression scheme and index. The two <code>VARCHAR</code> objects in the tuple column each have their own unique compression scheme. The <code>CREATE INDEX SQL</code> statement assigns an <code>INVERTED</code> index to the second tuple object, the integer array <code>INT[]</code> . |

For details about Ociant container types, see [Array, Tuple, and Matrix Overview](#).

## Related Links

[Data Types](#)

[Database, Tables, Views, and Indexes](#)

[Table Compression Options](#)

[Secondary Indexes](#)

## 6.5 Manage Distributed Tasks

Distributed tasks are processes that run in the background of the Ocient® System to keep the system running smoothly and efficiently. These tasks provide persistent storage of task information in the global metadata storage cluster of the Ocient System, allowing tasks to be tracked and retried until completion. For details about storage and related concepts, see [Core Elements of an Ocient System](#).

The distributed task infrastructure is completely generic. The system instantiates tasks from factories using generic arguments and establishes communication using interfaces to abstract the business logic in whatever specific protocol is doing the actual work. Tasks must remain unchanged when duplicated and can be restarted on a new node by the infrastructure at any time. Tasks can also be structured recursively, with a parent task comprising several subtasks, which can also have subtasks, and so on.

Distributed tasks provide persistence across outages. To do this, the task infrastructure detects when the node running a task, or the task owner, has become unavailable and moves the task elsewhere. The administrator owner manages node unavailability. This owner is a node in the administrator cluster responsible for monitoring the status of a task and reassigning the task owner when needed. This pairing of administrator and task owners provides load balancing across the administration cluster by removing the need for every task owner to communicate with the administrator owner.

### Task Types

The Ocient System has different task types, such as:

- Data pipeline loading
- Rebuild segments
- Check for disk corruption by verifying segments
- Rebalance data across storage clusters and nodes

For data pipeline loading, see [Load Data](#).

For details about rebuilding segments, checking the disk, and rebalancing data, see [Distributed Tasks](#).

### Task Characteristics

The characteristics of a task are a factory, serialized arguments, a location type, and a location identifier:

- Factory — A task class implementation that the Ocient System instantiates generically and executes the actual work of the task.
- Arguments — The inputs to a task the Ocient System stores generically as a string. These inputs can contain anything the task needs, e.g., flags modifying execution modes, identifiers on which to run, etc.
- Location type — The nature of the set of nodes where the task can run. This location can be any node in the system, any node in a specific cluster, or a specific node.
- Location identifier — The specific location entity when the task location type is a cluster or node.

Tasks have optional pre- and post-condition checks and must implement an asynchronous execute method. You can also override cancel and status poll methods.

### Task Execution

Execute tasks using a pair of nodes: administrator and task owners. The administrator owner must be an active node in the metadata storage cluster. This node is responsible for starting and monitoring the task. The task owner

can be any node that meets the location constraints specified by the task creator and executes the actual task. With the exception of subtasks, the Ocient System assigns both owners upon task creation, chosen randomly to balance the load.

To see the disposition of tasks in the system, see the `sys.subtasks` system catalog table. For details, see `sys.subtasks`.

### **Administrator Owner**

The administration owner manages and monitors all steps in the life cycle of their own tasks. After the Ocient System adds a task to the cluster state and assigns an administrator owner, the Administrator Node sends a message to the task owner to submit or poll the current task status. After the task has started successfully, the administrator owner periodically polls the task owner for the task status, updating it in the cluster state as needed.

When a task completes, the administrator owner sets its state to a terminal status and no longer polls or interacts with the task. The Ocient System retains historical tasks in the cluster state until a configurable limit is reached to prevent the state from growing unboundedly.

### **Task Owner**

The task owner maintains an in-memory map of running tasks by the identifier in the health protocol. The Ocient System pools tasks for execution status as needed. When the Ocient System receives a poll task request from an administrator owner, a task owner checks for the existence of the task, starting it if necessary and returning the current status.

The Ocient System starts a task by instantiating a task object using the factory type and then executing its pre-condition check, method, and post-condition check. If at any point the system encounters a failure, the system stops execution.

When a task completes, the task owner stores the results in memory until it receives an indication from the administrator owner that it is safe to remove the results.

### **Handle Outages**

#### **Administrator Owner**

An important responsibility of the administrator owner is to detect and handle task owner outages. If a poll fails for any reason, the administrator owner reassigns the owner in the cluster state and starts the task in the new location. Tasks must remain unchanged when duplicated and resume where they stopped when restarted.

If an administrator owner fails, the administrator leader detects the connectivity change and reassigns the administrator owner. The new administrator owner begins polling the task owner when it receives the update, so the ownership change should not interfere with task execution.

#### **Task Owner**

Generally, the task owner does not need to handle outages. However, in extreme situations, e.g., network splits, an administrator owner might not be able to communicate with its task owner and orphan a task. To prevent orphaned tasks from running in multiple locations in the system, the task owner times out and cancels a running task if it has not heard from the administrator owner in multiple poll cycles.

### **Subtasks**

When you execute a task using a SQL statement, the task performs the operation by executing one or multiple subtasks. Thus, when you query the system catalog tables for subtasks, you must look for numerous subtasks and their individual progress rather than just one task.

You can also execute tasks as a series of subtasks, that run sequentially, in parallel, or some combination of the two. The series comprises a parent task with a number of child tasks. Each child task has a specified ordering. The Ocient System executes tasks with the same ordering in parallel. And, the system executes groups of child tasks in sequence according to ascending order. The Ocient System assigns an administrator owner when the child task

should begin running. The system executes parent tasks directly on the administrator owner. These tasks are only responsible for watching the status of their running child tasks and starting the next child tasks when applicable. Parent tasks complete after all their children tasks complete.

To see the disposition of subtasks in the system, you can query the `sys.subtasks` system catalog table. For details, see `sys.subtasks`.

### Task Cancellation

A client can request to gracefully cancel a task by setting an `is_canceled` flag on the task object. The administrator owner evaluates this state on its next poll cycle and notifies the task owner that the task should be canceled. The exact mechanism by which the running task is canceled varies by type, as some tasks might need to communicate with other protocols to stop execution.

Canceling a parent task causes all of its subtasks to be canceled. A subtask cannot be canceled individually.

If the task is successfully canceled, execution ends with a `CANCELLED` status and reverts back to the administrator owner and Raft state using the normal success or failure paths.

For example, cancel the task named `my_task`.

```
CANCEL TASK 'my_task';
```

For the full syntax, see [Distributed Tasks](#).

### Related Links

[Core Elements of an Ociant System](#)

[Distributed Tasks](#)

[System Catalog](#)

## 6.6 Remove Records from an Ociant System

Ociant® supports multiple options for removing records from a database. Here, you can find the different methods for removing data along with their advantages and drawbacks to help you determine the method that best fits your needs.

All options for removing data are destructive and cannot be undone after execution. Therefore, be cautious when removing any Ociant records from your database.

### Required Permissions

To remove data, a user must have the `DELETE` privilege for the specified table. For details on privileges, see the [Data Control Language \(DCL\) Statement Reference](#).

### Remove Records Using DELETE

The `DELETE` SQL statement removes specific individual rows from a table based on the `WHERE` filter clause.

Unlike `TRUNCATE`, the `DELETE` statement does not restore the disk space of deleted rows without further system operations. In general, `DELETE` is a good option when removing small row sets, not entire tables.

If a `DELETE` statement has no `WHERE` filter, it removes all rows in the table.

The `DELETE` statement can support concurrent operations to remove rows on the same table, which might impact performance.

For details about the `DELETE` statement, see [Database, Tables, Views, and Indexes](#).

## Remove Data Using DELETE with a Filter

In this example, the DELETE statement removes all rows from the `products` table with a value less than 1 in the `rating` column.

```
DELETE FROM products WHERE rating < 1;
```

## Remove Data Using Joined Tables

You can use DELETE with a JOIN statement to help filter rows for removal based on a condition that applies to the second table.

The statement removes rows only from the first table and not the second joined table.

This example deletes rows after performing a JOIN operation on the geospatial values in two tables:

- `point_table` contains a column of geospatial point values.
- `us_state_table` contains columns of all abbreviated U.S. state names and polygon values representing their boundaries.

The JOIN operation uses the [Spatial Relationships](#) function to evaluate if each point value in `point_table` is contained within any of the polygon values in the `us_state_table`.

The example deletes any rows in `point_table` that are within the boundaries of Texas, abbreviated as 'TX'.

```
DELETE FROM point_table p
  JOIN us_state_table s ON ST_INTERSECTS(s.polygon_wkt, p.point_wkt)
WHERE s.state LIKE 'TX';
```

## Data Durability Considerations

To remove data, the data must be queryable in the system. For example, streaming data systems might experience a delay before rows can be deleted while they are in progress of the loading process to become fully durable.

In most cases, data should be loaded and queryable within a matter of seconds.

## Related Links

[Key Concepts](#)

[Ingest Data with Legacy LAT Reference](#)

[Query Ocident](#)

---

## 7 System Administration

Learn how to install and set up an Ocident® System, including hardware and software requirements and ongoing maintenance.

See these pages for key concepts to understand the Ocident System design and setup.

### Installation

These topics provide tutorials for installing Ocident on different system deployments. The reference pages in this section explain Ocident requirements and recommended system configurations.

- [Install an Ocident System](#)
- [Amazon Web Services Ocident Installation](#)
- [Configure Storage Spaces](#)
- [Ocident Software Upgrade](#)

### Maintenance

Maintenance topics include tutorials for managing system hardware for stable performance. Also, you can learn how to add new hardware and restore the system for cases when drives or nodes fail or when a system requires expansion.

- [Guide to Rebuilding Segments](#)
- [Startup and Shutdown Procedure](#)
- [Power Safety Recommendations](#)
- [NVMe Drive Firmware Upgrade Process](#)
- [Replace an NVMe Node Drive](#)
- [Expand and Rebalance System](#)
- [Replace Nodes](#)
- [Backup and Restore](#)

### Monitoring

These topics explain Ocident System monitoring, including error logging, system statistics, and supported tools for system administration.

- [Log Monitoring](#)
- [Statistics Monitoring](#)
- [System Information REST Endpoints](#)
- [Set Up System Monitoring with the TIG Stack and Kapacitor](#)
- [Monitor Segment Group Transfers](#)

### System Configuration

This set of topics explains how to inspect and alter the system configuration.

- [Inspect the Current Configuration](#)
- [Node Configuration with the Administrator Role](#)
- [Manage the Network Configuration of an Ocident System](#)
- [Secure Connections Using TLS](#)
- [Multiple Storage Clusters for Loading Data](#)
- [Alter Default Data Pipeline Behavior](#)
- [Configure Data Pipeline Logging](#)

### Loading Administration

- [Alter Default Data Pipeline Behavior](#)
- [Configuration Settings for Data Pipelines](#)
- [Configure Data Pipeline Logging](#)

## 7.1 Install an Ocient System

This section provides a quick tutorial for installing and configuring the Ocient® System using your data.

### Installation of the Ocient System

Before installing, check with your systems administrator to ensure that you meet the [Ocient System Requirements](#) for the validated operating systems and hardware needed for optimized Ocient performance.

The installation and configuration of the Ocient System involves these steps:

#### [Ocient Application Installation](#)

This step details how to download and install the Ocient System package.

#### [Ocient System Bootstrapping](#)

This step describes the Ocient bootstrapping, which sets up the initial node configuration and connects the nodes together.

#### [Ocient Application Configuration](#)

This step sets up your Ocient System with storage spaces, storage clusters, and node roles.

### 7.1.1 Ocient Application Installation

You can install the Ocient® software application by installing a single RPM or Debian® package.

Before installation, review your hardware sizing and ensure that it meets your needs. For details, see [Ocient System Hardware Specifications](#). For a list of prerequisites needed for Ocient installation, see the [Ocient System Requirements](#).

For access to Ocient software packages, please contact Ocient Support.

### Application Installation

Commands

Consult the documentation for the package manager of your operating system for specific options and command-line syntax.

#### RPM Installation

Use this command to install the Ocient RPM package from the local filesystem. Include `./` or specify the full path to the RPM file; otherwise, the `dnf` command attempts to search the configured repositories instead of installing the local RPM file. In the following command example, replace `XX` with the Ocient package version of the provided file.

```
sudo dnf install ./ocient-XX-x86_64.rpm
```

#### DEB Installation

Use this command to install the Ocient DEB package from the local filesystem. Include `./` or specify the full path to the DEB file; otherwise the `apt` command attempts to search the configured repositories instead of installing the local DEB file. In the following command example, replace `XX` with the Ocient package version of the provided file.

```
sudo apt install ./ocient-XX-amd64.deb
```

### Next Step

When your system install completes, proceed to [Ocient System Bootstrapping](#).



## Related Links

[Installation Reference](#)

[Install an Ocient System](#)

### 7.1.2 Ocient System Bootstrapping

The Ocient® System runs the bootstrapping process one time per node. This process establishes the node identity, configures networking, and accepts nodes into the system.

The `/var/opt/ocient/bootstrap.conf` file controls the initial configuration of each node. The system uses this file only when the node initially starts.

#### Validate the System Before Bootstrapping

The following steps assume a basic Ocient Hyperscale Data Warehouse™ configuration, where every node has a single network interface.

To verify your system before the bootstrapping process:

- Check with your systems administrator to ensure that you meet the [Operating System Configuration](#) requirements. Before bootstrapping, your system must meet the requirements described in the [Operating System Configuration](#) and [Operating System Configuration](#) sections.
- You can run a system check to validate the compliance of hardware, nodes, and the configuration for the Ocient System. For details, see [Checking System Configuration Before Bootstrapping](#).

#### Initial Configuration

The system automatically initializes the first node bootstrapped in the system with the SQL and Administrator roles. Choose a node to serve as a SQL Node. This node is the initial node in the system. The hardware for each node type differs, so ensure that the node you select is intended for a SQL Node.

For new clusters, if you do not define the configuration for the SQL Nodes in a connectivity pool, the Ocient System assigns each SQL Node to their own unique connectivity pool named `<sql_node_name>_default`, where `<sql_node_name>` is the name of the SQL Node. However, the most common configuration of an Ocient System has all SQL Nodes in one connectivity pool.

For an example of adding SQL Nodes to a connectivity pool, see [Cluster and Node Management](#). Change or troubleshoot the default configuration using the instructions in [Manage the Network Configuration of an Ocient System](#).

#### Ocient System Bootstrapping Process

If you meet the system prerequisites, you can begin bootstrapping your Ocient System.

The bootstrapping process consists of these steps:

1. [Ocient System Bootstrapping](#)
2. [Ocient System Bootstrapping](#)

#### Bootstrap the First Node

Bootstrap the Node

Connect to the initial SQL Node with the username and password of your server and the IP address of your node. This example connects as the administrator `admin` to the IP Address `10.10.10.10`.

```
ssh admin@10.10.10.10
```

Use your preferred text editor with `sudo` to create the `/var/opt/ocient/bootstrap.conf` file as root with these contents.

```
/var/opt/ocient/bootstrap.conf example
```

```
initialSystem: true
```

Start the database.

```
sudo systemctl start rolehostd
```

Verify that the node and the service is active by executing this status command.

```
systemctl status rolehostd
```

```
rolehostd.service - Rolehostd daemon startup
  Loaded: loaded (/etc/systemd/system/rolehostd.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2022-01-26 23:31:36 UTC; 7s ago
  ...
```

If the `rolehostd` service is running, you can also check the Ocient logs on your node. Search and ensure there are no [ERROR] log messages.

```
tail -f /var/opt/ocient/log/rolehostd.log
```

Verify Connection to the SQL Node

At this point, you have a running database with a single node. You should be able to connect to the database using [Connect Using JDBC](#) or [Connect Using pyocient](#) and execute commands.

Every new system starts with a `system` database. To connect to a new system, use the username and password configured in the `bootstrap.conf` file or the username `admin@system` and password `admin` if none was provided.

For example, assume your node named `sql0` has an IP address of `10.10.0.1`. Use the JDBC driver CLI to connect with this connection string.

```
connect to jdbc:ocient://10.10.0.1:4050/system user "myuser@system" using "mypassword";
```

See the [JDBC Manual](#) for details on using the JDBC driver.

To see the roles running on the single node, execute this query.

```
select name, operational_status, software_version, array_agg(service_role_type)
  from sys.node_status as ns
 left join sys.nodes as n on ns.node_id = id
 left join sys.service_roles as sr on sr.node_id = n.id
 group by name, operational_status, software_version;
```

The initial node should be listed as running the `sql`, `admin`, `health`, and `operatorvm` roles. If all of these roles are present and the node is Active, you can proceed to the next step to bootstrap the remaining nodes.

## Bootstrap the Remaining Nodes

Performing the bootstrapping process on the remaining nodes is identical on all nodes. The remaining nodes can be bootstrapped in any order.

On each node, log in using SSH and use your text editor with `sudo` to create the file `/var/opt/ocient/bootstrap.conf` that contains this text by replacing `<FIRST_NODE_IP_ADDRESS>` with the IP address of the initial node you created in Step 1.

```
/var/opt/ocient/bootstrap.conf example
```

```
adminHost: <FIRST_NODE_IP_ADDRESS>
```

`<FIRST_NODE_IP_ADDRESS>` is the DNS name or IP address of the initial node.

You can obtain the IP Address of the initial node by executing `ifconfig` on that node.

If the password for the system administrator has changed, set the correct username `adminUserName` and password `adminPassword` in the bootstrap configuration file `bootstrap.conf`.

On each node, start the database.

```
sudo systemctl start rolehostd
```

When you replace Foundation Nodes, the Ocient System removes the prior node after the creation of the new node. Some queries of the system catalog tables might not return results until the prior node is removed.

At this point, all the remaining nodes are not configured with any roles. After all nodes have been started, you should see them when you execute this query with only the `health` role listed. This query uses the `sys.node_status`, `sys.nodes`, and `sys.service_roles` system catalog tables to retrieve node information for the node name, operation status, version, and all service role types. The query uses the [Sorted Aggregate Functions](#) function to retrieve the service role type for all rows.

```
SELECT name, operational_status, software_version, array_agg(service_role_type)
FROM sys.node_status AS ns
LEFT JOIN sys.nodes AS n ON ns.node_id = id
LEFT JOIN sys.service_roles AS sr ON sr.node_id = n.id
GROUP BY name, operational_status, software_version;
```

## More Bootstrapping Options

For options on bootstrapping your Ocient System, see [Node Bootstrapping Reference](#).

## Next Step

When you have completed the bootstrapping process, you can proceed to the [Ocient Application Configuration](#) page to finish configuring Ocient.

## Related Links

[Installation Reference](#)

[Install an Ocient System](#)

### 7.1.3 Ocient Application Configuration

These steps assume that you have already completed the process for bootstrapping your Ocient® System nodes. If you have not done this yet, perform the [Ocient System Bootstrapping](#) steps before beginning this configuration process.

Now that the nodes have all completed bootstrapping, you must properly configure the nodes to perform their roles. You should run these commands using a SQL connection to one of the SQL Nodes in your system. For details, see [Connect to Ocient](#).

The Ocient application configuration process consists of these steps:

1. [Ocient Application Configuration](#)
2. [Ocient Application Configuration](#)
3. [Ocient Application Configuration](#)
4. [Ocient Application Configuration](#)

## Creating a Storage Space

To create a storage space with 12 segments, use `WIDTH=12` and `PARITY_WIDTH=2`. Two of these segments are the parity portion, and 10 of these segments are the data portion.

```
CREATE STORAGE SPACE teststoragespace WIDTH = 12, PARITY_WIDTH = 2;
```

The storage space name must begin with a letter followed by letters, numbers, and underscores.

You can view all storage spaces in the system by querying the `sys.storage_spaces` system catalog table.

```
SELECT * FROM sys.storage_spaces;
```

For details about creating a storage space to suit your system needs, see [Configure Storage Spaces](#).

## Creating Storage Clusters

You must have at least one storage cluster to configure an Ocient System. A storage cluster is a set of Foundation Nodes with an associated storage space that coordinate to reliably store data and serve queries.

Create a Foundation cluster.

```
CREATE CLUSTER lts_cluster
TYPE LTS storage_space "teststoragespace"
PARTICIPANTS=('lts0', 'lts1', 'lts2', 'lts3', 'lts4', 'lts5', 'lts6', 'lts7', 'lts8', 'lts9', 'lts10', 'lts11');
```

Loader Nodes and SQL Nodes do not need to be placed into a cluster.

At a minimum, the number of participants must be the width specified in the storage space. This number can be greater if you overprovision the system. For details about storage spaces and related settings, overprovisioning, and parity, see [Core Elements of an Ocient System](#).

View all clusters in the system.

```
SELECT * FROM sys.clusters;
```

View all clusters and their associated nodes.

```
SELECT c.name AS cluster_name, c.cluster_type, n.name AS node_name
FROM sys.node_clusters nc
JOIN sys.clusters c ON nc.cluster_id = c.id
JOIN sys.nodes n ON nc.node_id = n.id;
```

You must restart all nodes with the SQL role after adding a storage cluster.

## Assigning Service Roles

Assign service roles to each accepted node. This process defines how each node participates in queries and the management of the system.

View all nodes and their configured roles.

```
SELECT n.name AS node_name, sr.service_role_type
FROM sys.nodes n
LEFT JOIN sys.service_roles sr ON sr.node_id = n.id;
```

The addition of nodes to the storage cluster in the prior step automatically assigns `lts` to the Foundation Nodes that you included.

Add the remaining `sql`, `admin`, and `streamloader` roles to the appropriate nodes.

```
ALTER NODE "sql1" ADD ROLE sql;
ALTER NODE "sql1" ADD ROLE admin;
ALTER NODE "loader1" ADD ROLE streamloader;
```

The Admin role can operate standalone or on the same node as the SQL role.

## Restart the Nodes

With roles assigned to the nodes, you can restart the `rolehostd` process on all nodes. At a shell terminal on each node, run this command.

```
sudo systemctl restart rolehostd
```

If any custom configuration settings are applied to running nodes during other configurations, be sure to restart the `rolehostd` process before proceeding.

You can check the status of nodes using the Status endpoint on each node from a terminal. Assuming IP Addresses 10.10.0.2 and 10.10.0.3 for `sql1` and `lts0`, respectively, you can check the status with these curl commands.

```
curl http://10.10.0.2:9090/v1/status
curl http://10.10.0.3:9090/v1/status
```

After all of the nodes are **Active**, Ocident is ready to use for queries.

## Creating an Ocident System with Multiple Storage Clusters

If you need an Ocident System that requires 10 or more Foundation Nodes in the storage cluster, then you can create multiple storage clusters. For considerations to take into account when you work with multiple storage clusters, see [Multiple Storage Clusters for Loading Data](#).

Create a multiple storage cluster with 12 nodes. Create a storage space `store` with width 5 and parity width 1 using the `CREATE STORAGESPACE` SQL statement.

```
CREATE STORAGESPACE store WIDTH = 5, PARITY_WIDTH = 1;
```

Create multiple clusters using the `CREATE CLUSTER` SQL statements using the storage space `store`. Split the Foundation Nodes among the clusters. `cluster_0` storage cluster has Foundation Nodes `lts0` through `lts5`, while `cluster_1` has `lts6` through `lts11`.

```
CREATE CLUSTER cluster_0 TYPE=lts PARTICIPANTS=("lts0", "lts1", "lts2", "lts3", "lts4", "lts5") STORAGESPACE=store;
CREATE CLUSTER cluster_1 TYPE=lts PARTICIPANTS=("lts6", "lts7", "lts8", "lts9", "lts10", "lts11") STORAGESPACE=store;
```

## Validating the System After Configuration

After configuration, you can run a system check to validate the Ocident installation and bootstrapping. For details, see [Checking System After Configuration](#).

## Next Step

Follow the process in [Ingest Data with Legacy LAT Reference](#) to create a database and a table, then load data from a set of files located on a cloud object storage.

## Related Links

[Installation Reference](#)

## Install an Ocient System

### 7.1.4 Installation Reference

#### 7.1.4.1 Ocient System Sizing

The Ocient® System must be the proper size for your data and query requirements. Follow these steps to determine the size of your Ocient System. Contact Ocient to aid with the Ocient System calculations, established configuration, and node quantities.

1. Gather the uncompressed size of your current database or data and its format. If your data resides in a database, determine the compression ratio of the data. Further, consider whether your data is streaming or file-based. For streaming or time series data, investigate the level of data granularity (for example, minute, hour, or day).
2. Calculate the quantity of data (in Gbps or the number of rows in a specific unit of time) that you plan to load into the Ocient System and at what cadence (for example, continuous loading or specific schedule). Describe the latency you expect between the data creation and when data reaches the destination. As you consider the quantity, determine the percent of NULL data. Also, think about the retention period required for the data.
3. Determine the level (small, medium, or large) of data transformation in the data pipeline loading process. See whether you require any user-defined transformations. Investigate whether your transformations are done programmatically or using SQL. After the initial data load, see if you require more transformations, such as cubes or views.
4. Investigate the types of queries you plan to execute (for example, simple filters, complex aggregation or sorting, complex joins, or transformation). In addition, review your query workload (number of queries in a specific unit of time) and how many queries you expect to execute in parallel. Consider any objectives you have for system readiness to execute queries.
5. Calculate the compressed data volume. Consider your compression requirements for the stored data (for example, Global Dictionary Compression).
6. Add index, statistics, working space, and parity size to the compressed data volume.
7. Use the established storage space configuration in the Ocient System for data nodes, parity nodes, and the number of clusters.
8. Using the known size and level of transformation in loading, determine the total number of cores, memory, and storage for your requirements.
9. Calculate the number of SQL Nodes, Loader Nodes, Foundation Nodes, Metadata Nodes, and supporting nodes. Ocient personnel assist with this step.

The number of specific nodes determines the size of the Ocient System. After you have the latter numbers, then you can purchase, install, and configure the system.

#### Related Links

[Ocient System Requirements](#)

[Ocient System Hardware Specifications](#)

[Table Compression Options](#)

[Global Dictionary Compression](#)

[Secondary Indexes](#)

#### 7.1.4.2 Ocient System Requirements

In order to run the Ocient Hyperscale Data Warehouse™, you must meet several requirements for your system hardware, OS, and software. The requirements ensure that your system can run an optimized and fully functional Ocient® installation.

These requirements apply to the Foundation, Loader, and SQL Nodes.

Consult with Ocient or the System Administrator installing Ocient to determine the optimal configuration for your Ocient deployment.

### Server Hardware Requirements

These are the hardware requirements for the Ocient Hyperscale Data Warehouse node servers. These requirements include the Foundation, Loader, SQL, and Metadata Node types. Data (i.e. non-OS) drive requirements are listed separately.

#### Foundation, Loader, and SQL Nodes

| Component          | Requirements                                                                                       | Notes                                                                                                                                                                                                                                                                                                                                     |
|--------------------|----------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Chassis/System     | 8+ NVMe hotswap bays<br>Dual CPUs<br>M.2 slots for OS drives<br>Redundant power supplies           | {{Supermicro}} SYS-120U-TNR or similar recommended<br>Full bandwidth PCIe connections to NVMe drives recommended (i.e. not attached to RAID card and not multiplexed).<br>Drives must appear as <code>nvme</code> devices.<br>NVMe PCIe lanes balanced between CPUs recommended (i.e. equal number of drives connected to CPU1 and CPU2). |
| CPU                | 2x {{Intel}} Skylake or {{AMD}} Rome or newer x86-64 server CPUs                                   | 26+ cores per CPU recommended.<br>Must support AVX2 instruction set.                                                                                                                                                                                                                                                                      |
| Memory             | 1TB+ recommended<br>All CPU memory channels must be populated to achieve maximum memory bandwidth. | Balanced memory configuration with all memory channels populated required to achieve maximum bandwidth.                                                                                                                                                                                                                                   |
| High Speed Network | 100Gbit dual port ethernet NIC                                                                     | {{NVIDIA}} {{Mellanox}} ConnectX-5 MCX516A-CCAT recommended.<br>Dual ports required for redundancy.                                                                                                                                                                                                                                       |
| OS Drives          | 1TB+ Enterprise NVMe drives (M.2)                                                                  | RAID-1 mirroring recommended.<br>M.2 format recommended to keep all U.2 bays available for data drives.                                                                                                                                                                                                                                   |
| Data Drives        | See Ocient System Requirements.                                                                    | NVMe drives for short-term and long-term data storage.                                                                                                                                                                                                                                                                                    |

#### Metadata Nodes

The optional Metadata Nodes have fewer requirements than the other nodes.

| Component      | Requirements                           | Notes                                                                                                                     |
|----------------|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| Chassis/System | Virtual Machine or Physical Machine    | No specific chassis requirements.<br>The system can be a virtual machine.                                                 |
| CPU            | 16+ cores Xeon-D or similar x86-64 CPU | Must support AVX2 instruction set.<br>You might need to change the virtual machine CPU type to support AVX2 instructions. |
| Memory         | 64GB                                   | None.                                                                                                                     |

| Component | Requirements                                    | Notes                                                                                        |
|-----------|-------------------------------------------------|----------------------------------------------------------------------------------------------|
| Network   | 10 Gbps NIC or virtual NIC                      | Metadata Nodes do not require high-speed networking. Dual ports are required for redundancy. |
| OS Drive  | 512GB Enterprise SSD or SSD-backed storage pool | NVMe is recommended.                                                                         |

### Data Drive Requirements

Ocident strictly requires high-performance enterprise or data center NVMe SSD drives for long-term and short-term data storage. The drives listed here have been validated for functionality and performance. They also support drive self-encryption using the OPAL standard. All data stored is encrypted at rest. Power loss protection (PLP), standard on enterprise and datacenter drives, is also required. An endurance of 0.5 drive writes per day (DWPD) or higher is recommended for Foundation and SQL Nodes, endurance of 1.0 DWPD is recommended for Loader Nodes.

- Solidigm™ D5-P5430 OPAL U.2 3.84TB-15.36TB
- Solidigm™ D7-P5520 OPAL U.2 1.92 TB-15.36TB
- Solidigm™ D7-P5510 OPAL U.2 3.84TB, 7.68TB
- Samsung® PM9A3 U.2 960GB-15.36TB

Solidigm P5510 firmware versions VDV10151 and VDV10152 are known to not work properly with OPAL enabled. Upgrade the firmware to a known working version.

Samsung PM9A3 firmware versions GDD5102Q and GDD5202Q are known to have issues with the `vfio-pci` driver. Upgrade the firmware to a newer version.

Enterprise NVMe drives with similar or better performance characteristics and OPAL (or better) support can be considered for validation.

If the Ocident System detects hardware support for the OPAL standard, then nodes enable full disk encryption on data drives by default. Ocident stores the encryption keys on the operating system drive. For the complete list of directories to back up, see [Backup and Restore](#).

### Compatible Operating System Versions

These operation systems and versions are fully supported by Ocident:

- Red Hat® Enterprise Linux® (RHEL®) version 8\* and version 9
- Ubuntu® 20.04, 22.04, 24.04
- Debian® 11, 12

RHEL versions 8.3 and prior require a kernel upgrade for compatibility. There is a bug in the `uio_pci_generic` driver in the kernel provided with those versions that prevents Ocident from communicating with the NVMe drives.

### Network Requirements

Ocident requires high bandwidth connections between its database nodes to achieve high performance. It is critical for performance that the Foundation and SQL Nodes can simultaneously utilize high bandwidth links without bottlenecks. Generally this means that those nodes share the same switch backplane for best performance.

The minimum recommended link speed is 100Gbps.



## Redundancy

Redundant networking is recommended for production use cases. On the server side, dual port NICs are required to support redundancy. Multi-chassis link aggregation (MLAG/CLAG/MCLAG) using redundant switches is the recommended form of redundancy.

## Network Switch

These are the requirements for the network switches to support Ocient. No other advanced L2 or L3 features are required.

1. Enough 100G ports for all Foundation Nodes on the same backplane for best performance.
2. Multi-chassis link aggregation for redundancy (MLAG/CLAG/MCLAG depending on vendor terminology).

## Security Requirements

To secure data stored on disk, Ocient supports and recommends SEDs (Self-Encrypting Drives) that meet the OPAL (or better) standard. Using drives that support this standard, Ocient software enables encryption of data at rest on the drives. This encryption prevents unwanted data leaks, as any time a disk is removed from the system it becomes locked and the data can only be recovered with a key-encrypting-key that is stored off-drive.

To secure network communications, see the [Ocient Security Guide](#).

## Additional Software Requirements

Ocient depends directly on these software packages:

- `python3`
- `sudo`
- `lz4`
- `cronie`
- `logrotate`
- `gzip`
- `pciutils`
- `ca-certificates`
- `libquadmath` (RHEL), `libquadmath0` (Ubuntu/Debian)

For database nodes, these packages are installed automatically as part of the package dependencies.

Ocient also depends on a separate installation of Java® 21, where the `openjdk-21-jre-headless` package is recommended.

## Next Step

If your system meets these requirements, proceed to [Ocient Application Installation](#).

## Related Links

[Ocient Software Upgrade](#)

[Install an Ocient System](#)

### 7.1.4.3 Ocient System Hardware Specifications

Depending on your deployment modality and your purchase agreement with Ocient®, the Ocient System can run on hardware bearing the Ocient brand. Here, you can learn about the specifications for the hardware. However, an Ocient System can run on generic hardware that meets the system requirements. For details, see [Ocient System Requirements](#).

The Ocient System contains Foundation Nodes, Loader Nodes, and SQL Nodes. For details about these node types, see [Core Elements of an Ocient System](#).

For sizing considerations, see [Ocient System Sizing](#). Then, you can further optimize the system by using compression at the table level. For details, see [Table Compression Options](#). After you have the hardware and size defined, you can install the Ocient System. For details, see [Install an Ocient System](#).

## Foundation Node Specifications

### Components

This table provides the components and their descriptions and quantities for the hardware.

| Component                     | Description                                                                                                                                | Quantity |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|----------|
| Chassis                       | Supermicro® Hyper A+ Server AS -2126HS-TN — 24 NVMe Drive 2.5” drive baysFor details, see the Hyper A+ Server AS -2126HS-TN specification. | 1        |
| Processors                    | Dual AMD® EPYC™ 9654 Processors — 96 cores per processor                                                                                   | 2        |
| Memory                        | 96GB DDR5-4800 RDIMM                                                                                                                       | 24       |
| 100Gbps NIC                   | NVIDIA® Mellanox® 100Gbps Network Card — MCX623106AN-CDAT PCIE 2port 100GB QSFP28                                                          | 1        |
| Trusted Platform Module (TPM) | TPM 2.0                                                                                                                                    | 1        |
| Boot Drive RAID Carrier       | AOC-SLG4-2H8M2-O, 2x Hybrid Gen 4 NVMe/SATA M.2 RAID 0/1 Carrier, Standard                                                                 | 1        |
| DCMS License                  | Supermicro Data Center Management Suite License                                                                                            | 1        |
| Boot Drive                    | Samsung® PM983 1.9TB 1 DWPD NVMe PCIe3x4 M.2 22x110 SED                                                                                    | 3        |
| Storage Drives                | Solidigm™ D5-P5430 [7.68TB, 15.36TB, 30.2TB] NVMe U2 15 mm SED                                                                             | 12       |

### Physical Specifications

This table provides the specifications and their corresponding values for the hardware.

| Specification                   | Value                               |
|---------------------------------|-------------------------------------|
| Form Factor                     | 2U Rackmount                        |
| Height                          | 3.5” (88.9 mm)                      |
| Width                           | 17.2” (437 mm)                      |
| Depth                           | 31.74” (806.2 mm)                   |
| Package                         | 9.96” (H) x 26.46” (W) x 43.31” (D) |
| Net Weight                      | 45 lbs (20.5 kg)                    |
| Gross Weight                    | 75 lbs (34 kg)                      |
| Operating Temperature:          | 10°C to 35°C (50°F to 95°F)         |
| Non-operating Temperature       | -40°C to 70°C (-40°F to 158°F)      |
| Operating Relative Humidity     | 8% to 90% (non-condensing)          |
| Non-operating Relative Humidity | 5% to 95% (non-condensing)          |
| Power supplies                  | 2000W Redundant Power Supplies      |
| Peak Load Wattage               | 1350W                               |
| Average Load Wattage            | 810W (varies based on workload)     |
| Min Load Wattage                | 240W                                |

### Performance Specifications

This table provides different foundation hardware performance specifications for Foundation Nodes.

| Node Type                                     | Media (GB) | Total Data Drives (SSDs) (TB) | 4K Read IOPS per Drive (Number of Reads) | 4K Read IOPS (All Drives) (Number of Reads) | Sequential Read (128KB) Per Drive (Gbps) | Sequential Read (128KB) All Drives (Gbps) |
|-----------------------------------------------|------------|-------------------------------|------------------------------------------|---------------------------------------------|------------------------------------------|-------------------------------------------|
| Foundation Node                               | 102.13     | 7.6812 92.16                  | 717,000                                  | 8,604,000                                   | 54.4                                     | 652.8                                     |
| Large Storage Optimized Foundation Node       | 102.13     | 15.362 184.32                 | 935,000                                  | 11,220,000                                  | 56.0                                     | 672                                       |
| Extra Large Storage Optimized Foundation Node | 102.13     | 30.722 368.64                 | 971,000                                  | 11,652,000                                  | 56.0                                     | 672                                       |
| Double Extra Large Storage Optimized          |            |                               |                                          |                                             |                                          |                                           |

Loader Node and SQL Node Specifications

The components, specifications, and sizing for Loader Nodes and SQL Nodes are the same.

Components

This table provides the components and their descriptions and quantities for the hardware.

| Component  | Description                                                                                                                               | Quantity |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------|----------|
| Chassis    | Supermicro Hyper A+ Server AS -2126HS-TN — 24 NVMe Drive 2.5” drive baysFor details, see the Hyper A+ Server AS -2126HS-TN specification. | 1        |
| Processors | Dual AMD EPYC 9654 Processors — 96 cores per processor                                                                                    | 2        |
| Memory     | 96GB DDR5-4800 RDIMM                                                                                                                      | 24       |

| Component                     | Description                                                                | Quantity |
|-------------------------------|----------------------------------------------------------------------------|----------|
| 100 Gbps NIC                  | Mellanox 100Gbps Network Card — MCX623106AN-CDAT PCIE 2port 100GB QSFP28   | 1        |
| Trusted Platform Module (TPM) | TPM 2.0                                                                    | 1        |
| Boot Drive RAID Carrier       | AOC-SLG4-2H8M2-O, 2x Hybrid Gen 4 NVMe/SATA M.2 RAID 0/1 Carrier, Standard | 1        |
| DCMS License                  | Supermicro Data Center Management Suite License                            | 1        |
| Boot Drive                    | Samsung PM983 1.9TB 1 DWPD NVMe PCIe3x4 M.2 22x110 SED                     | 1        |
| Storage Drives                | Solidigm D7-P5520 1.92TB NVMe U2 15 mm SED                                 | 8        |

### Physical Specifications

This table provides the specifications and their corresponding values for the hardware.

| Specification                   | Value                               |
|---------------------------------|-------------------------------------|
| Form Factor                     | 2U Rackmount                        |
| Height                          | 3.5" (88.9 mm)                      |
| Width                           | 17.2" (437 mm)                      |
| Depth                           | 31.74" (806.2 mm)                   |
| Package                         | 9.96" (H) x 26.46" (W) x 43.31" (D) |
| Net Weight                      | 45 lbs (20.5 kg)                    |
| Gross Weight                    | 75 lbs (34 kg)                      |
| Operating Temperature:          | 10°C to 35°C (50°F to 95°F)         |
| Non-operating Temperature       | -40°C to 70°C (-40°F to 158°F)      |
| Operating Relative Humidity     | 8% to 90% (non-condensing)          |
| Non-operating Relative Humidity | 5% to 95% (non-condensing)          |
| Power supplies                  | 2000W Redundant Power Supplies      |
| Peak Load Wattage               | 1050W                               |
| Average Load Wattage            | 630W (varies based on workload)     |
| Min Load Wattage                | 220W                                |

### Performance Specification

This table provides the hardware performance specification for a Loader Node and SQL Node.

| Node Type          | Cores | Memory (TB) | Drives (TB) | Drives (Number) | Total Data Storage (TB) | 4K Read IOPS                | 4K Read IOPS                   | Sequential Read          | Sequential Read           |
|--------------------|-------|-------------|-------------|-----------------|-------------------------|-----------------------------|--------------------------------|--------------------------|---------------------------|
|                    |       |             |             |                 |                         | per Drive (Number of Reads) | (All Drives) (Number of Reads) | (128KB) Per Drive (Gbps) | (128KB) All Drives (Gbps) |
| Loader or SQL Node | 192   | 2.3         | 1.92        | 8               | 15.36                   | 700,000                     | 5,600,000                      | 42.4                     | 339.2                     |

**Related Links**[Install an Ocient System](#)[Operating System Configuration](#)[Checking System Configuration Before Bootstrapping](#)[Ocient System Sizing](#)[Table Compression Options](#)**7.1.4.4 Operating System Configuration**

The Ocient® System Configuration procedure assumes that these steps have been performed on the Foundation Nodes and Loader Nodes in the database cluster. Some of these steps require root user permissions.

Use **sudo** when performing these steps in your command-line interface.

Prepare the Operating System Configuration

You must prepare the operating system with this configuration to run Ocient software:

- Configure Network interface controllers (NICs) and firewall for use by Ocient.
- Set the timezone to UTC.
- If OPAL drives are installed, unlock or revert them to factory state.
- Optional: Mount points `/opt/ocient` and `/var/opt/ocient` are mounted on separate storage devices.

Confirm the Ocient Installation

To check the installation, run this command on each node to show the software version.

```
/opt/ocient/current/rolehostd --build-info
```

If the software is not installed or is the wrong version, see the [Ocient Application Installation](#) instructions.

Apply Required Kernel Settings

In order to achieve high-performance levels, you must set specific kernel settings. You cannot set these settings at runtime. Therefore, the system must pass these parameters to the kernel at boot time.

**Automatically Apply Kernel Settings**

You can automatically configure the kernel settings and apply them by running the `ockernelparams` script in the Ocient software package. This script detects the current system hardware and adjusts the `grub` bootloader kernel parameters. Then, the script updates the `grub` configuration with `grub2-mkconfig` or `update-grub` (depending on the Linux distribution).

Run this command and specify the appropriate node role.

```
/opt/ocient/scripts/ockernelparams --node-role {sql,loader,foundation,metadata}
```

You can also manually configure kernel settings. See [Operating System Configuration](#) for more information.

**Check Current Kernel Settings**

Run this command to show the current settings for the kernel that is running.

You need to reboot after any changes are made to kernel parameters in the bootloader configuration.

```
cat /proc/cmdline
```

```
BOOT_IMAGE=(hd0,gpt2)/boot/vmlinuz-4.18.0-305.19.1.el8_4.x86_64 root=UUID=165df3b2-9221-43be-967d-d91483c0
```

The exact values can vary depending on the DRAM of the machine and the intended role for the node.

Check that the number of free 1G huge pages matches the number specified at the command line. You can check this number using this command.

```
cat /sys/kernel/mm/hugepages/hugepages-1048576kB/nr_hugepages
```

*Output*

```
1024
```

The returned value should match your configuration.

### Manual Kernel Settings

This section describes the optional kernel settings that you can manually configure. To automatically configure these settings, see [Operating System Configuration](#).

You must add these settings that follow the **grub** bootloader kernel parameters for the listed applicable node types.

#### Setting: transparent\_hugepages

Ocident requires that **transparent\_hugepages** be disabled because Ocident software needs strict control over which huge pages it allocates.

#### *Applicable Node Types:*

- Foundation Nodes
- Loader Nodes
- SQL Nodes

#### *Kernel setting:*

```
transparent_hugepage=never
```

#### Setting: isolcpus

To give Ocident software exclusive use of the majority of the cores on the system, the kernel is isolated from the cores using the **isolcpus** parameter. This isolation ensures that the system executes the kernel code only on a small number of cores where Ocident is not running logic that is critical for performance.

The pattern is to isolate all but the last two physical cores of each CPU from the kernel for CPUs with 26 or more cores. This means that two cores (4 threads) per CPU are available to the kernel.

For CPUs with fewer than 26 cores, all but the last physical core of each CPU is isolated.

#### *Applicable Node Types:*

- Foundation Nodes

#### *Kernel setting:*

- Dual 28-core CPUs (Intel® 6348)

```
isolcpus=0-25,28-53,56-81,84-109
```

- Dual 32-core CPUs (Intel 6338)

```
isolcpus=0-29,32-61,64-93,96-125
```

- Dual 96-core CPUs (AMD® 9654)

```
isolcpus=0-93,96-189,192-285,288-381
```

#### Setting: Huge Pages

Ocident utilizes huge pages to streamline the management of large amounts of memory in the system. The system uses 2MB and 1GB huge pages, depending on the node type. These huge pages must be reserved at boot time

because otherwise, the system memory quickly becomes too fragmented to allocate the number of huge pages needed at runtime. These settings are the expected huge page settings.

Metadata Nodes do not use huge pages.

| Node Type                           | Installed DRAM | Huge Page Kernel Parameters                                                                     |
|-------------------------------------|----------------|-------------------------------------------------------------------------------------------------|
| Foundation or SQL                   | 768 GB         | hugepagesz=1G hugepages=384<br>hugepagesz=2M hugepages=16384                                    |
| Foundation or SQL                   | 1024 GB        | hugepagesz=1G hugepages=680<br>hugepagesz=2M hugepages=16384                                    |
| Foundation or SQL                   | 1152 GB        | hugepagesz=1G hugepages=768<br>hugepagesz=2M hugepages=16384                                    |
| Foundation or SQL                   | 1536 GB        | hugepagesz=1G hugepages=1024<br>hugepagesz=2M hugepages=16384                                   |
| Foundation or SQL                   | 2048 GB        | hugepagesz=1G hugepages=1364<br>hugepagesz=2M hugepages=16384                                   |
| Foundation or SQL                   | 2304 GB        | hugepagesz=1G hugepages=1536<br>hugepagesz=2M hugepages=16384                                   |
| Loader                              | 768 GB         | hugepagesz=1G hugepages=384<br>hugepagesz=2M hugepages=16384                                    |
| Loader                              | 1024 GB        | hugepagesz=1G hugepages=512<br>hugepagesz=2M hugepages=16384                                    |
| Loader                              | 1152 GB        | hugepagesz=1G hugepages=576<br>hugepagesz=2M hugepages=16384                                    |
| Loader                              | 1536 GB        | hugepagesz=1G hugepages=768<br>hugepagesz=2M hugepages=16384                                    |
| Loader                              | 2048 GB        | hugepagesz=1G hugepages=1024<br>hugepagesz=2M hugepages=16384                                   |
| Loader                              | 2304 GB        | hugepagesz=1G hugepages=1152<br>hugepagesz=2M hugepages=16384                                   |
| Metadata (standalone administrator) | N/A            | No huge pages. Use the<br>noHugePages parameter when<br>bootstrapping node:noHugePages:<br>true |

## Prepare NVMe Drives

These steps are for Foundation Nodes, Loader Nodes, and SQL Nodes only.

To prepare NVMe drives on Foundation Nodes, Loader Nodes, and SQL Nodes, these nodes must have the state where they are connected to the `uio_pci_generic` or `vfio-pci` driver and not contain a partition table. To initialize the drives for use by Ocient that have been previously partitioned, you must perform these steps.

1. Identify the OS drives. The OS drives have partitions that are mounted to mount points such as `/`. In this example, `nvme0n1` is the OS drive.

```
lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
nvme0n1      259:0    0  1.8T  0 disk
nvme0n1p1    259:1    0   512M  0 part /boot/efi
nvme0n1p2    259:2    0  1.8T  0 part /
```

2. Ensure all drives are bound to the NVMe driver so the NVMe drives show up as standard NVMe devices. List the NVMe drives to identify the drive identifiers.

```
sudo /opt/ocient/scripts/nvme-driver-util.sh bind-nvme
```

```
/opt/ocient/scripts/nvme-driver-util.sh
```

| BDF          | Numa Node | Driver name | Device name |
|--------------|-----------|-------------|-------------|
| 0000:01:00.0 | 0         | nvme        | nvme0n1     |
| 0000:3b:00.0 | 0         | nvme        | -           |
| 0000:3c:00.0 | 0         | nvme        | -           |
| 0000:5e:00.0 | 0         | nvme        | -           |
| 0000:5f:00.0 | 0         | nvme        | -           |
| 0000:86:00.0 | 1         | nvme        | -           |
| 0000:87:00.0 | 1         | nvme        | -           |
| 0000:88:00.0 | 1         | nvme        | -           |
| 0000:89:00.0 | 1         | nvme        | -           |

3. For each NVMe drive that you need to erase and prepare for Ocident (not the OS drive), run this command by replacing X and Y to match the device name.

```
sudo nvme format /dev/nvmeXnY format --ses=1
```

This command formats the specified drive and erases all data on the drive. Ensure that the OS drive is not formatted if it is an NVMe drive.

Some drives do not support the `nvme format` command. In that case, you can overwrite the partition table by zeroing out the beginning of the drive with this command. This command does not securely erase the whole drive, but does allow it to be used by Ocident.

```
sudo dd if=/dev/zero of=/dev/nvmeXnY bs=1024K count=1024
```

4. Bind the drives to the UIO or VFIO driver by running this command. The script automatically chooses the appropriate driver after detecting if the IOMMU setting is enabled. Any NVMe drives with mounted partitions (i.e. the OS drive) remain on the NVMe driver. Show all expected NVMe drives on the node attached to the UIO or VFIO driver.

```
sudo /opt/ocient/scripts/nvme-driver-util.sh bind-vfio
```

```
/opt/ocient/scripts/nvme-driver-util.sh
```

| BDF          | Numa Node | Driver name | Device name |
|--------------|-----------|-------------|-------------|
| 0000:01:00.0 | 0         | nvme        | nvme0n1     |
| 0000:3b:00.0 | 0         | vfio-pci    | -           |
| 0000:3c:00.0 | 0         | vfio-pci    | -           |
| 0000:5e:00.0 | 0         | vfio-pci    | -           |
| 0000:5f:00.0 | 0         | vfio-pci    | -           |
| 0000:86:00.0 | 1         | vfio-pci    | -           |
| 0000:87:00.0 | 1         | vfio-pci    | -           |
| 0000:88:00.0 | 1         | vfio-pci    | -           |
| 0000:89:00.0 | 1         | vfio-pci    | -           |

#### Clear Files from Any Previous Ocident Setup

To bootstrap a node that was previously part of an Ocident System, you must clear a set of files from any previous setup of Ocident from the `/var/opt/ocient/` directory. These commands remove these files.

```
sudo rm -f /var/opt/ocient/rolehostd.conf
sudo rm -f /var/opt/ocient/ocient.key
sudo rm -f /var/opt/ocient/ocient.crt
sudo rm -f /var/opt/ocient/ocient.crt.original
```



```
sudo rm -f /var/opt/ocient/*.raft
sudo rm -f /var/opt/ocient/bootstrap.completed
```

Do not remove files in the `localKeyStore` directory if OPAL drives are present and encryption is not disabled. If these files are removed, the drive encryption must be reset using an external utility.

#### Related Links

[Ocient Software Upgrade](#)

[Install an Ocient System](#)

#### 7.1.4.5 Checking System Configuration Before Bootstrapping

The `precheck` command helps capture critical node health states and metrics before bootstrapping your Ocient® installation.

You can run the command as shown in this example. The flags included after the `precheck` command are optional. See [Checking System Configuration Before Bootstrapping](#) for descriptions.

```
$ sudo precheck --print-level verbose --node-type metadata --logfile-location /home/admin
```

The command invokes a script from the command line on operating systems supported by Ocient. The script sends output to standard output (STDOUT) and standard error (STDERR). To specify the file option, use this flag: `--logfile-location <directory>` where `<directory>` is the directory where the `precheck` command creates the log file. Critically, this script captures information about huge page configuration.

#### Optional Flags for Command Execution

Use the `precheck` command to run the script that checks the system configuration. You can execute the command using the optional flags shown in this table.

| Flag Name                       | Values            | Default                          | Description                                                                                | Example                                                                                                                                                                                                           |
|---------------------------------|-------------------|----------------------------------|--------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>--logfile-location</code> | Non-empty string  | <code>/var/opt/ocient/log</code> | Name of the directory that contains the log file created from the execution of the script. | <code>--logfile-location /user/query/logs</code> generates a log file in the <code>/user/query/logs</code> directory. If this directory does not exist, the command issues a warning.                             |
| <code>--print-level</code>      | verbose or silent | verbose                          | Denotes the level of detail in the output. See the “Outputs” section.                      | <code>--print-level verbose</code> provides a detailed report with information about the checks that Ocient performs. <code>--print_level silent</code> provides the overall status from the check of the system. |

| Flag Name                           | Values                                        | Default                                              | Description                                                                                                                                                                                               | Example                                                                                                            |
|-------------------------------------|-----------------------------------------------|------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| <code>-node-type</code>             | metadata, lts, stream-loader, sql, or unknown | unknown                                              | Denotes the current node type. If you do not specify this setting, the script tries to infer node type based on metrics it collects.                                                                      | <code>--node-type lts</code> runs checks with values and constraints that pertain to <code>anltsnode</code> .      |
| <code>-timeout</code>               | Any integer                                   | 3                                                    | Denotes the default timeout (in seconds) to run any command-line command in the script. In most cases, the default is enough, so you do not need to specify this setting.                                 | <code>--timeout 4</code> runs checks with a timeout of 4 seconds for each command.                                 |
| <code>-nvme-driver-util-path</code> | Non-empty string                              | <code>/opt/ocient/scripts/nvme-driver-util.sh</code> | Path to <code>nvme-driver-util.sh</code> that the precheck script uses. The default value is the location of the script on most installations, so in most cases, you do not need to specify this setting. | <code>--nvme-driver-util-path /opt/ocient/test_dir/nvme-driver-util.sh</code> sets the path to the utility script. |
| <code>-json</code>                  | Not applicable                                | False                                                | If you specify the <code>--json</code> flag, the command prints output in JSON format to STDOUT. If you do not specify this flag, the command prints human-readable output to STDOUT.                     | <code>--json</code> prints output in JSON format to STDOUT.                                                        |

### Command Output

The command has two types of output, silent or verbose, which you can specify by using the `--print-level` flag at the command line.

#### Silent

The `silent` value for the `--print-level` flag only produces a status code. The code is 0 for success, 1 for error, and 2 for warning.

#### Verbose

The `verbose` value for the `--print-level` flag prints either human-readable output by default or output in JSON format (if you specify the `--json` flag). The output contains all status details.

This example shows human-readable output.

```

Warnings:
- isolcpus not configured - recommended for performance.
- No eligible NVMe payload drives found for node.
- bootstrap.conf file does not exist. Create a file at /var/opt/ocient/bootstrap.conf with the correct boo
- SELinux is enabled. This might prevent Ocient from functioning properly. Please disable or set to permis

Memory - OK
=====
ramconfig - SUCCESS: RAM configuration is adequate.

Firmware and Hardware - ERROR
=====
nonbootdrives - WARNING: No eligible NVMe payload drives found for node
...

```

This example shows output in JSON format.

```

{"status": 1,
  "components": {
    "enforce_status": {
      "status": 2,
      "message": "WARNING: SELinux is enabled. This might prevent Ocient from functioning properly. Please
    }
    "bootstrap": {
      "status": 2,
      "message": "WARNING: bootstrap.conf file does not exist. Create a file at /var/opt/ocient/bootstrap.
    }
    ...
  }
}

```

Related Links

[Ocient System Bootstrapping](#)

[Ocient Application Configuration](#)

#### 7.1.4.6 Checking System After Configuration

The `postcheck` command helps capture critical node health states and metrics after the Ocient® configuration process.

You can run the command as shown in this example. The flags included after the `precheck` command are optional. See [Checking System Configuration After Bootstrapping](#) for descriptions.

```
$ sudo /usr/local/bin/postcheck --dsn ocient://ur-sql0:4050/system;user=admin@system;password=admin
```

The command invokes a script from the command line on operating systems supported by Ocient. The script sends output to standard output (STDOUT) and standard error (STDERR) with the option to send a text file or a JSON file. To specify the file option, use this flag: `--logfile-location <directory>` where `<directory>` is the directory where the `postcheck` command creates the log file.

Before you begin, ensure that all steps in the [Ocient System Bootstrapping](#) process are complete.

Prerequisites

The script requires Java® and the Ocient JDBC JAR to run. In most cases, you should use this script on client servers rather than database nodes.

To get the post-check script onto a client, perform the `scp` operation on the wheel onto the server and run `python3 -m pip install --no-index <path_to_wheel>/postcheck*.whl`, and replace `<path_to_wheel>` with the path

to the directory of the wheel file on your server.

#### Flags for Command Execution

Use the **postcheck** command to run the script that checks the system configuration after you configure Ocient. You can execute the command using these flags.

| Flag Name                      | Required | Values            | Default                          | Description                                                                                                                                                                    | Example                                                                                                                                                                                                                        |
|--------------------------------|----------|-------------------|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-dsn</code>              | Yes      | Many              | Not applicable                   | Data source name (DSN). The script prompts for your credentials.                                                                                                               | <code>--dsn ocient://host:port/pathco</code><br>you to the specified cluster where <code>host</code> is the host name, <code>port</code> is the port number, and <code>path</code> is the path on the host for the connection. |
| <code>-logfile-location</code> | No       | Non-empty string  | <code>/var/opt/ocient/log</code> | Name of directory that contains the log file created from the execution of the script.                                                                                         | <code>--logfile-location /user/query/logs</code><br>generates a log file in the <code>/user/query/logs</code> directory. If this directory does not exist, the command issues a warning.                                       |
| <code>-print-level</code>      | No       | verbose or silent | verbose                          | Denotes the level of detail in the output. See the “Outputs” section.                                                                                                          | <code>--print-level verbose</code><br>provides a detailed report with information about the checks that Ocient performs. <code>--print-level silent</code> provides the overall status from the check of the system.           |
| <code>-timeout</code>          | No       | Any integer       | 3                                | Denotes the default timeout (in seconds) for running any command-line command in the script. In most cases, the default is enough, so you do not need to specify this setting. | <code>--timeout 4</code><br>runs checks with a timeout of 4 seconds for each command.                                                                                                                                          |

| Flag Name              | Required | Values           | Default                                    | Description                                                                                                                                                                                               | Example                                                                                                                                                                    |
|------------------------|----------|------------------|--------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-jar-path</code> | No       | Non-empty string | <code>/opt/ocient/current/jdbc4.jar</code> | Defines the path to the Ocient JDBC JAR. The default value is the usual location of the JAR file on an Ocient System. This path must be accurate for the <code>postcheck</code> command to run correctly. | <code>--jar-path /opt/ocient/test_dir/jdbc4.jar</code><br>the JAR at <code>/opt/ocient/test_dir/jdbc4.jar</code> run JDBC commands for the <code>postcheck</code> command. |
| <code>-json</code>     | No       | Not applicable   | False                                      | If you specify the <code>-json</code> flag, the command prints output in JSON format to STDOUT. If you do not specify this flag, the command prints human-readable output to STDOUT.                      | <code>--json</code> prints output in JSON format to STDOUT.                                                                                                                |

### Command Output

The command has two types of output, silent or verbose, which you can specify by using the `--print-level` flag at the command line.

#### Silent

The `silent` value for the `--print-level` flag only produces a status code. The code is 0 for success, 1 for error, and 2 for warning.

#### Verbose

The `verbose` value for the `--print-level` flag prints either human-readable output by default or output in JSON format (if you specify the `--json` flag). The output contains all status details.

This example shows human-readable output.

Warnings:

```
- Foundation Nodes storage are non-uniform.
```

```
OS Configuration - OK
```

```
=====
```

```
Huge Page Settings - OK
```

```
isolcpus settings - OK
```

```
Memory - OK
```

```
=====
```

```
Foundation Nodes: 1024 GB
```

```
SQL Nodes: 1024 GB
```

```
Loader Nodes: 1536 GB
```

```
Total DRAM: 123120 GB
...
```

This example shows output in JSON format.

```
{
  "status": 2,
  "components": {
    "firmware": {
      "status": 0,
      "message": "SUCCESS: NVMe drive firmware is consistent across nodes."
    }
  },
  "symmetry": {
    "status": 2,
    "message": "WARNING: non-uniform configuration."
  },
  "storage": {
    "status": 2,
    "foundation": {
      "status": 0,
      "message": "SUCCESS: storage consistent across Foundation Nodes."
    },
    "foundation1": 16TB,
    ...
  }
  ...
}
```

Related Links

[Ocient System Bootstrapping](#)

[Ocient Application Configuration](#)

#### 7.1.4.7 Node Bootstrapping Reference

Bootstrap a Node

Connect to the Node

To bootstrap an Ocient® node, connect to the node with your server username and password and the IP address of your node.

For example, connect as administrator **admin** to the IP Address 10.10.10.10.

```
ssh admin@10.10.10.10
```

Configure the Bootstrap Configuration File

Use your preferred text editor with **sudo** to create the `/var/opt/ocient/bootstrap.conf` file as root.

For a list of bootstrapping options, see [Node Bootstrapping Reference](#).

##### Option 1: Bootstrap the initial node in a system

Use these `bootstrap.conf` settings to create the initial node in a system.

`/var/opt/ocient/bootstrap.conf` **example**

```
initialSystem: true
```

##### Option 2: Bootstrap a new node on a system with an initial node

Use these `bootstrap.conf` settings to create another node in a system, replacing the IP Address 10.10.10.10 with the IP address or hostname of the initial node.

`/var/opt/ocient/bootstrap.conf` example

```
adminHost: 10.10.10.10
```

Start the Node

Start the database.

```
sudo systemctl start rolehostd
```

Verify that the node and the service are active by running this status command.

```
systemctl status rolehostd
```

```
rolehostd.service - Rolehostd daemon startup
  Loaded: loaded (/etc/systemd/system/rolehostd.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2022-01-26 23:31:36 UTC; 7s ago
  ...
```

You have bootstrapped the node.

Supported Options

You can edit the `/var/opt/ocient/bootstrap.conf` file to include a number of other options for advanced system configurations.

The Ocient `bootstrap.conf` file can also include these optional parameters. For examples of these options, see the [Node Bootstrapping Reference](#).

| Option                       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>name</code>            | The default name of the node in the database is the hostname. If that name is not appropriate, you can assign a different name.                                                                                                                                                                                                                                                                                                                                                                              |
| <code>nodeAddress</code>     | The default address of this node is the hostname of the node, or the IP address if <code>useDns</code> setting is <code>false</code> . If the system has multiple interfaces or you want an alternative setup, then you can specify the node address. This address is the address that all the other nodes in the cluster use to contact this node.                                                                                                                                                          |
| <code>externalAddress</code> | It might be desirable to have the SQL Nodes listen on a different external address than the system uses internally. The <code>externalAddress</code> parameter can either be a single address or a list of addresses. Example of single address: <code>externalAddress: 10.10.10.1</code> Example of a list of addresses: <code>externalAddress: ["10.10.10.1", "20.20.20.2"]</code> This option applies to REST endpoints only. For REST endpoints, see <a href="#">System Information REST Endpoints</a> . |
| <code>adminUserName</code>   | The default user, specified as <code>admin@system</code> , when you create an Ocient System. If a different administrator user should be used to add this node to an existing system, use this option to indicate the user.                                                                                                                                                                                                                                                                                  |
| <code>adminPassword</code>   | If the administrator password is not the default, then use this option to specify the password.                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>logLevel</code>        | The logging level for the system. Defaults to <code>DEBUG</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                           |

| Option            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| connectivityPools | The configuration of SQL Nodes in a connectivity pool. You can use this configuration to configure the connectivity pool for the first SQL Node in a cluster. Other nodes cannot use this option. If there are no entries in the <code>bootstrap.conf</code> file, then the Ocident System adds the SQL Node to a new connectivity pool with a default name. Example of the connectivity pool named <code>test_pool</code> configuration: <code>connectivityPools</code> : |

```
test_pool:
sourceAddress: 1.2.3.4/12
sourcePort: 1 # optional
priority: 1
listenAddress: 111.1.1.1
listenPort: 4050
advertisedAddress: localhost
advertisedPort: 4050 # optional |
```

**systemStorageSpace** | The configuration for storage space has width and parity width for the Ocident System. If you do not configure the storage space, the system sets the width to 3 and parity to 2, the default values. For details about width and parity width, see [Core Elements of an Ocident System](#). To set these options using SQL syntax, see [Cluster and Node Management](#). |

**useOpal** | When you set this value to `false`, this option prevents the `rolehostd` service from taking ownership of and enabling Opal locking on all NVMe drives that support it. Bootstrap with this setting using an external Key Management System (KMS). |

#### Example bootstrap.conf File

This example shows a `bootstrap.conf` file that you can use as a template.

```
# initialSystem: true           # Uncomment for the first node in a
                                # multi-cluster system.
                                # Defaults to false. Cannot be
                                # true if initialSingleClusterSystem
                                # is also true.

# initialSingleClusterSystem: true # Uncomment for the first node in a
                                # single-cluster system.
                                # Defaults to false. Cannot be
                                # true if initialSystem is also true.

# systemName: productionSystem  # The name of the Ocident System. This
                                # configuration applies only when the
                                # initialSystem or initialSingleClusterSystem
                                # configuration is set to true.

# adminHost: sql.example.com     # Hostname or IP address of an administrator
                                # node, generally the first node. Not
                                # required for the initial node, but required
                                # for all other nodes.

# name: elmo                    # The name of this node.
                                # Defaults to hostname.
```



```

# nodeAddress: sql1.example.com # The Hostname or IP address of this node.
                                # Defaults to hostname, and if that does not
                                # work, the bootstrapping process uses the IP
                                # address of the system.

# externalAddress: ["10.10.10.1", "20.20.20.2"]

# useDns: false                 # Use DNS to determine the hostname of this
                                # node. Defaults to true. When you set this
                                # configuration to false, the bootstrapping
                                # process uses the IP address of the system.

# adminUserName: bert@system    # A system administrator username.
                                # Defaults to "admin@system".
# adminPassword: ernie          # The password for the administrator user.
                                # Defaults to "admin".

# logLevel: info                # Log level for the system.
                                # Defaults to debug.

# connectivityPools:            # Define connectivity pool named test_pool.
#   test_pool:
#     sourceAddress: 1.2.3.4/12 # The source IP address, which is
                                # the IP address in CIDR notation of
                                # the client that connects to the SQL Node.

#     sourcePort: 1             # The optional source port number, which is the
                                # port number of the client. This parameter
                                # defaults to none, which means any port number.
#     priority: 1               # The priority of the connection. A higher number
                                # indicates a higher priority.
#     listenAddress: 111.1.1.1 # The IP address for listening.
#     listenPort: 4050          # The port number for listening.
#     advertisedAddress: localhost # The IP address to return to the client.
#     advertisedPort: 4050      # The optional port number to return to the client.
                                # If you specify NULL for this parameter, then the
                                # database uses the value of the listen_port parameter.

# systemStorageSpace:          # Define the system storage space.
#   width: 3                    # The width is the total number of segments to use in
                                # each segment group.
#   parityWidth: 2              # The parity width is the number of parity coding bits
                                # to use for each segment group, or the fault tolerance.

```

#### Related Links

[Checking System Configuration Before Bootstrapping](#)

[Operating System Configuration](#)

[Cluster and Node Management](#)

## 7.2 Amazon Web Services Ocient Installation

Deploy a CloudFormation template to install an Ocient® System in Amazon® Web Services<sup>SM</sup> (AWS<sup>SM</sup>) .

For details about AWS concepts, see these links:

- [What is Amazon EC2?](#)
- [What is IAM?](#)
- [What is AWS CloudFormation?](#)
- [A Primer for Amazon Networking - VPC, Availability Zones & Subnets](#)

Ocient supports deploying an Ocient System in AWS for pilot or testing purposes. There are no guarantees on data durability. Stopping Amazon EC2<sup>SM</sup> instances will result in permanent data loss.

Steps for deploying an Ocient System in AWS:

1. Prepare AWS resources.
2. Prepare a CloudFormation template.
3. Run the CloudFormation template with your customized parameters.
4. Wait for nodes to be provisioned.

### Target Configuration

The standard Ocient implementation of the CloudFormation template deploys an Ocient cluster with the following number and type of nodes. Contact Ocient Support for a CloudFormation template. Note that a template can be created for any number of nodes.

| Node Type            | Instance Type |
|----------------------|---------------|
| Foundation Nodes (3) | i3en.metal    |
| Loader Nodes (1)     | i3en.metal    |
| SQL Nodes (1)        | r5d.metal     |

This diagram shows an example of an Ocient cluster in AWS. The EC2 nodes (SQL, Loader, Foundation) are deployed within a single subnet of an Amazon Virtual Private Cloud<sup>SM</sup> (VPC). Each type of node is assigned a separate security group (`sg1`, `sg2`, `sg3`).

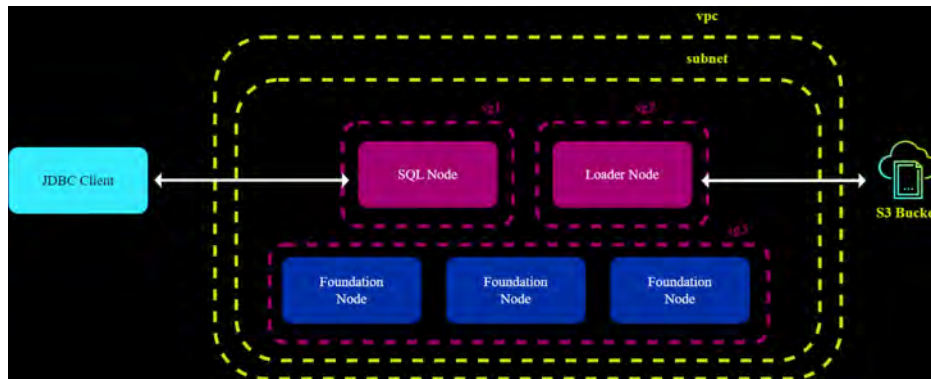


Figure 5: Virtual private cloud for loading data from an S3 bucket using a JDBC client

### AWS Prerequisites

Prior to running the CloudFormation template, a set of AWS resources must be initialized. These resources are the input parameters to the template. You must choose the Region (e.g., `N. Virginia - US East 1`) where you will be deploying your Ocient System and an associated availability zone (e.g., `us-east-1a`).

### Parameters

You pass the following parameters as input parameters to the CloudFormation template.

| Name                           | Example                    | Description                                             |
|--------------------------------|----------------------------|---------------------------------------------------------|
| SystemName                     | test                       | Name of the Ocient System (prefix for all nodes)        |
| ImageId                        | ami-abc0123a1230abcef      | Ocient AMI (contains Ocient software)                   |
| AvailabilityZone               | us-east-1a                 | Where the Ocient System will be deployed                |
| KeyName                        | myKeyPair                  | Name of an AWS KeyPair (used for SSH to nodes)          |
| SubnetId                       | subnet-0abc1230001230abc   | Preset range of IP addresses for Ocient nodes           |
| SQLNodeSecurityGroupIds        | sg-00011123000123abc       | Defines access to the SQL Nodes                         |
| FoundationNodeSecurityGroupIds | sg-012300abc12300123       | Defines access to Foundation Nodes                      |
| LoaderNodeSecurityGroupIds     | sg-0abcedf00123abcde       | Defines access to the Loader Nodes                      |
| LoaderNodeIamInstanceProfile   | AmazonS3ReadOnlyAccessRole | Passes privileges to the Loader Nodes                   |
| MetadataNodeSecurityGroupIds   | sg-a123123bc12300123       | Defines access to the Metadata Nodes                    |
| LogLevel                       | info                       | (optional) Sets the initial logging level for all nodes |
| StorageSpaceWidth              | 3                          | The storage space width of the storage cluster          |
| StorageSpaceParityWidth        | 1                          | The storage space parity width of the storage cluster   |

Many of the parameters are AWS resources. For each of those parameters, follow these steps to request the valid data to fill out the CloudFormation template. Save each parameter value to a temporary text file.

### Ocient Amazon Machine Image (AMI)

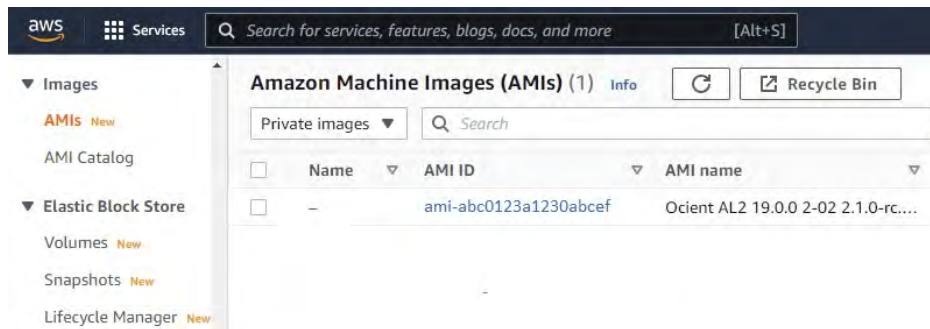


Figure 6: Amazon Machine Images (AMIs)

1. Contact Ocient Support to request the latest AMI.
2. The Ocient team shares the AMI with your AWS account.
3. Navigate to **EC2** in the AWS Management Console.
4. Ensure that the appropriate region is set on the top navigation bar.
5. Select **AMIs** under **Images** from the left navigation.
6. Select **Private Images** from the drop-down.
7. The screen displays an image, such as **ami-abc0123a1230abcef**, with a name similar to Ocient 22.0.0.
8. Save the image name.

### Key Pair

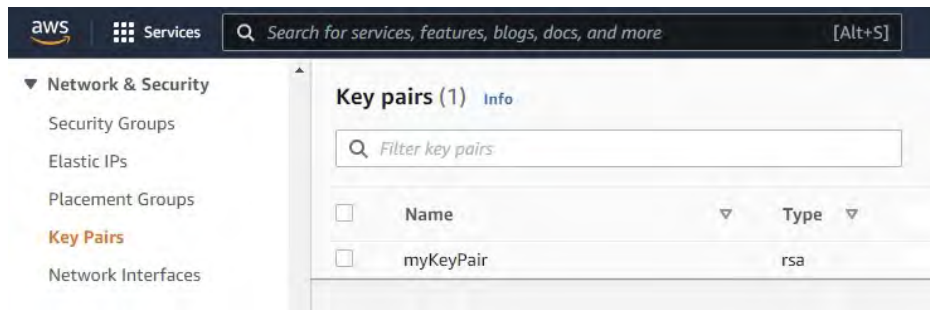


Figure 7: Key pairs

To attach credentials for SSH access to the EC2 nodes, you must select a Key Pair. For details about key pairs, see [Amazon EC2 key pairs and Linux instances](#).

1. View Key Pairs in the console by entering *key pairs* in the search bar.
2. Choose an existing key pair or create a new one.
3. Save the name of the Key Pair.

### Subnet

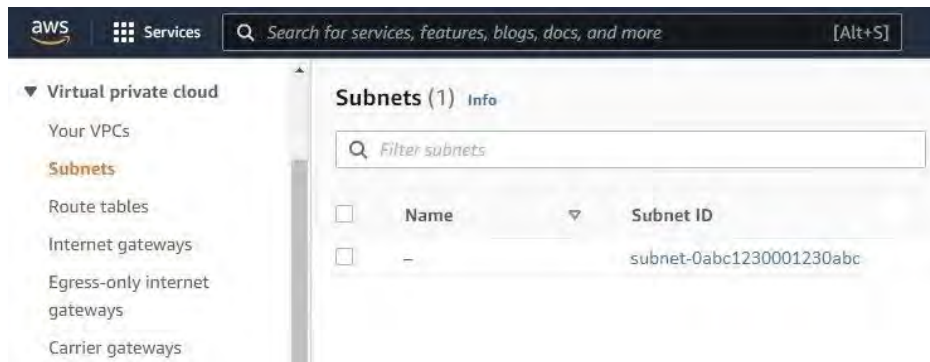


Figure 8: Subnets

A single subnet is used to determine the range of IPs used for the EC2 nodes.

1. View available subnets in the console by entering *subnet* in the search bar.
2. Choose an existing subnet in the chosen region or create a new subnet.
3. Save the Subnet ID.

The chosen subnet must match the chosen availability zone for the CloudFormation template to deploy correctly.

### Security Groups

The CloudFormation template allows different security groups to be assigned for each node type.

The main requirements are to provide access to:

- Port 4050 on the SQL Nodes for JDBC client connectivity
- Port 8080 on the Loader Nodes for LAT client access
- Port 22 on all nodes for SSH access (recommended)

Create an individual security group with rules for these port requirements and apply them to the specified node types.

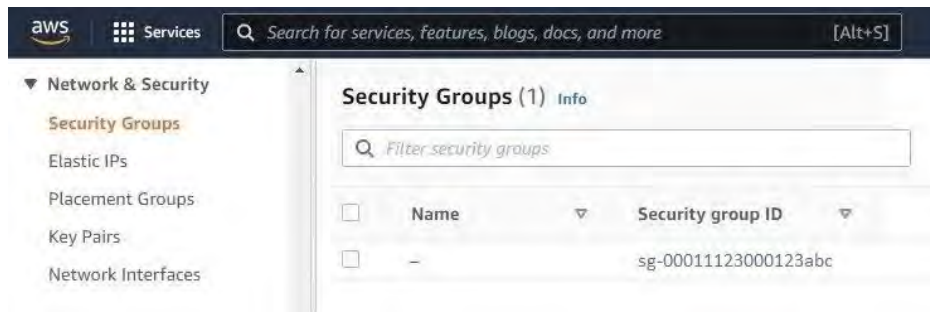


Figure 9: Security Groups

1. View Security Groups in the AWS console by entering *security groups* in the search bar.
2. Choose one or more existing security groups or create them as needed.
3. Save the Security group ID or multiple identifiers if more than one security group is created.

### Identity Access Management (IAM) Instance Role

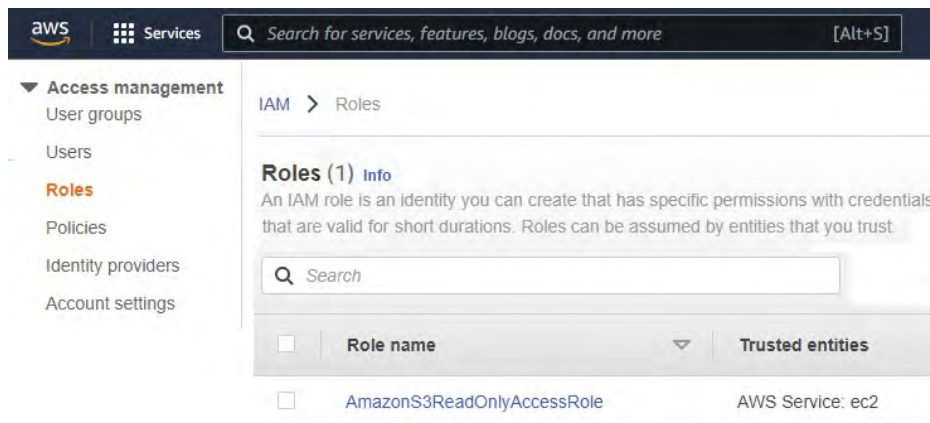


Figure 10: IAM Roles

An IAM Instance Role is used to pass an IAM Policy to the Loader Node. For details about configuring IAM roles, see [Configuring IAM roles for Amazon EC2](#).

1. View a list of Roles in the AWS console by entering *roles* in the search bar.
2. Choose an existing role or create a new one as needed.
3. Save the name of the role.

Minimum permissions for the policy should include Get/List permissions for Amazon S3 (where data is stored). For example, use the AWS standard policy `AmazonS3ReadOnlyAccess`.

You might optionally create a custom policy to limit access to a specific named resource (bucket).

### Deploy the Ocient CloudFormation Template

1. Contact Ocient Support to obtain the latest CloudFormation template (`ocient-cf.yaml`).
2. Deploy the Ocient CloudFormation template using the Create Stack command.
  1. Run the Create Stack command in the AWS console using **CloudFormation** → **Stacks** → **Create**. A set of input screens guides you to specify the location of the template file.
  2. Choose a template file.

3. Enter the name of the stack and enter the values for the parameters from the temporary text file.

Figure 11: Create stack

Or, you can run the Create Stack command using the AWS command line interface. For example:

```
aws cloudformation create-stack \
  --stack-name cloudformation-test \
  --template-body file://ocient-cf.yaml \
  --parameters \"ParameterKey=SystemName,ParameterValue=test\" \
  \"ParameterKey=ImageId,ParameterValue=ami-abc0123a1230abcef\" \
  \"ParameterKey=AvailabilityZone,ParameterValue=us-east-1a\" \
  \"ParameterKey=KeyName,ParameterValue=myKeyPair\" \
  \"ParameterKey=SubnetId,ParameterValue=subnet-0abc1230001230abc\" \
  \"ParameterKey=SQLNodeSecurityGroupIds,ParameterValue=sg-00011123000123abc\" \
  \"ParameterKey=FoundationNodeSecurityGroupIds,ParameterValue=sg-012300abc12300123\" \
  \"ParameterKey=LoaderNodeSecurityGroupIds,ParameterValue=sg-0abcedf00123abcde\" \
  \"ParameterKey=LoaderNodeIamInstanceProfile,ParameterValue=AmazonS3ReadOnlyAccessRole\" \
  \"ParameterKey=MetadataNodeSecurityGroupIds,ParameterValue=sg-a123123bc12300123\" \
  \"ParameterKey=LogLevel,ParameterValue=info\" \
  \"ParameterKey=StorageSpaceWidth,ParameterValue=3\" \
  \"ParameterKey=StorageSpaceParityWidth,ParameterValue=1\"
```

3. Accept the default configuration and click **Create Stack**. AWS creates the CloudFormation stack and provisions a set of Ocient nodes as EC2 instances.
4. Track the status of the CloudFormation stack in the AWS console using **CloudFormation** → **Stacks** → **Events**.
5. View the provisioned EC2 instances in the AWS console using **EC2** → **Instances**.



After approximately 20 minutes, the instances are stable and accessible using SSH.

During this time, each instance restarts at least once as kernel parameters are set up.

The initialization process can be observed in the `/var/log/startup-script.log` file on each instance.

Initialization is complete when the `startup-script.log` file on the `sql01` node contains this message.

text

```
Initialization successfully completed.
```

The Ocient System configuration is now complete with these items:

- Bootstrapping is completed.
- A storage space exists with all Foundation Nodes as members.
- The streamloader role has been applied to all Loader Nodes.
- The SQL and Administrator roles have been applied to the SQL Nodes.
- The Administrator role has been applied to all Metadata Nodes.
- The `rolehostd` service has been restarted for all role changes to take effect.

## Related Links

[Ingest Data with Legacy LAT Reference](#)

[Query Ocient](#)

## 7.3 Configure Storage Spaces

When setting up an Ocient® System, a storage cluster represents a set of Foundation Nodes in the system. A storage space is a set of parameters within a storage cluster that defines how the system administers data and fault tolerance.

Creating a storage space is an important step for launching an Ocient System that must be done after installing hardware and bootstrapping nodes but before you start loading and querying data. To see how creating a storage space fits into the Ocient installation process, see [Ocient Application Configuration](#).

This tutorial explains how to set up and configure storage spaces to meet the needs of your Ocient System.

When an Ocient System starts, the system automatically creates a default system storage space for persistent metadata, such as system catalog tables. This default storage space is immutable, meaning you cannot delete or modify it.

The name of the metadata storage space is `systemStorageSpace` in the `sys.storage_spaces` system catalog table.

## User-Defined Storage Spaces

For storage other than metadata, you must create one or more separate storage spaces. The creation of storage spaces is necessary before loading data or performing other operations. User-defined storage spaces are configurable for extra resiliency or storage based on your system needs.

A storage space represents how the storage cluster spreads data across Foundation Nodes to balance storage and fault tolerance. In general terms, the storage cluster configuration defines how the storage cluster behaves for these attributes:

- Regular data storage operations (see [Configure Storage Spaces](#))
- Query resiliency (see [Configure Storage Spaces](#))
- Load resiliency (see [Configure Storage Spaces](#))

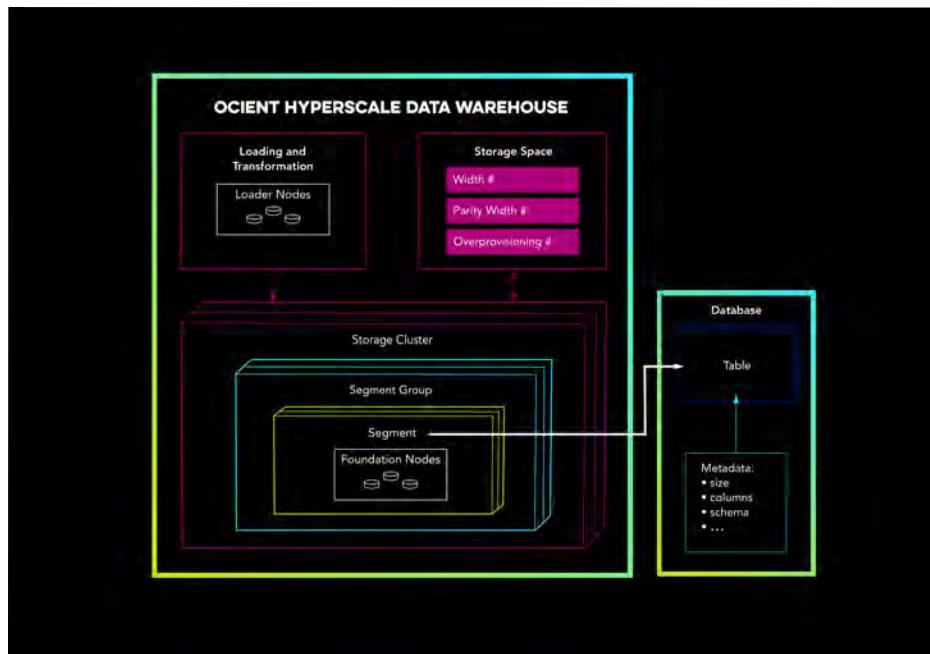


Figure 12: Ocient Hyperscale Data Warehouse contains Foundation Nodes in segments grouped into segment groups that are grouped into storage clusters. Tables with metadata are stored in segments

You can configure a storage space to assign nodes to these job roles by using DDL statements (see [Cluster and Node Management](#)).

Planning out your storage space carefully is important because you cannot change a storage space configuration after creation.

### Segment Group Width

Storage space parameter: `WIDTH`

The Segment Group Width determines the number of segments that comprise a segment group. Functionally, this setting defines how many Foundation Nodes perform read and write operations for a segment group.

Within a Segment Group, the system assigns each segment to a different node. Hence, the Segment Group Width cannot exceed the number of nodes in your system.

In most circumstances, Segment Group Width should comprise most of your Foundation Nodes. On system startup, the default Segment Group Width is three, which is the minimum number of nodes required for an Ocient System.

### Parity Width

Storage space parameter: `PARITY_WIDTH`

You can assign a subset of the Segment Group Width to Parity Width.

The Parity Width of a storage space determines its fault tolerance, defining the number of parity coding bits to use for each segment group. This number determines how many nodes can fail before the cluster is disabled.

Parity Width protects system querying capabilities. If any nodes fail, the system can execute and complete queries as long as the number of failed nodes is less than or equal to the Parity Width. When deployed in conjunction with overprovisioned nodes, Parity Width also provides fault tolerance for loading operations.

The `PARITY WIDTH` requires additional storage overhead, which is calculated by the formula  $(\text{PARITY WIDTH}) / (\text{SEGMENT GROUP WIDTH} - \text{PARITY WIDTH})$ .



## Overprovision

The Ociant System overprovisions any Foundation Nodes in the cluster in excess of the Segment Group Width by default. These nodes can include any nodes not configured in the storage space or any nodes added later.

Overprovisioned nodes protect loading operations from node failure, although they must be used in conjunction with Parity Width. In the event of node failures, loading operations can continue as long as the number of failed nodes does not exceed the Parity Width nodes or the number of overprovisioned nodes. For this reason, providing fault tolerance for loading requires a balance of Parity Width and overprovisioned nodes.

Unlike Segment Group Width and Parity Width, you do not explicitly define a value for overprovisioning. Instead, any Foundation Nodes in the cluster in excess of the number allocated toward Segment Group Width or Parity Width become overprovisioned by default.

## Storage Space Configuration Examples

This section demonstrates different configurations for storage spaces. The examples each go through different scenarios for node failures to show the fault tolerance of each setup.

For information on the syntax for storage spaces, see [Cluster and Node Management](#).

### 10-Node Cluster

This example assumes you have a storage cluster of 10 Foundation Nodes.

```
CREATE STORAGE SPACE ociant WIDTH = 10, PARITY_WIDTH = 2;
```

- The WIDTH = 10 parameter means the system stores a segment group on 10 different nodes.
- The PARITY\_WIDTH = 2 parameter means each segment in the group contains enough parity bits to restore lost data for up to two nodes. In effect, this means parity bits comprise 20 percent of storage.

#### Tolerance Scenarios

This storage space setup has no overprovisioning, making loading operations less resilient. This setup results in these outcomes if nodes become disabled:

- If one node fails, loading fails, and querying can continue.
- If two nodes fail, loading fails, and querying can continue.
- If three nodes fail, loading and querying both fail.

### 12-Node Cluster

This example assumes a storage cluster of 12 Foundation Nodes.

```
CREATE STORAGE SPACE ociant WIDTH = 10, PARITY_WIDTH = 2;
```

Note that this DDL statement is the same as the [Configure Storage Spaces](#), but the cluster in this example has two additional nodes.

- The WIDTH = 10 parameter means the system stores a segment group on 10 of the 12 nodes.
- The two remaining nodes are overprovisioned.
- The PARITY\_WIDTH = 2 parameter means each segment in the group contains enough parity bits to restore lost data for up to two nodes. In effect, this means parity bits comprise 16.6 percent of storage.

#### Tolerance Scenarios

This setup provides a level of resiliency for both querying and loading. The setup results in these outcomes if nodes become disabled:

- If one node fails, loading and querying continues.
- If two nodes fail, loading and querying continues.
- If three nodes fail, loading and querying both fail.

## 12-Node Cluster with More Parity Width

This example assumes a storage cluster of 12 Foundation Nodes with more nodes allocated to parity.

```
CREATE STORAGE SPACE ocident WIDTH = 10, PARITY_WIDTH = 3;
```

- The WIDTH = 10 parameter means the system stores a segment group on 10 of the 12 nodes.
- The two remaining nodes are overprovisioned.
- The PARITY\_WIDTH = 3 parameter means each segment in the group contains enough parity bits to restore lost data for up to three nodes. In effect, this means parity bits comprise 42 percent of storage.

### Tolerance Scenarios

This setup provides resiliency for loading and especially for querying. The setup results in these outcomes if nodes become disabled:

- If one node fails, both loading and querying continue.
- If two nodes fail, both loading and querying continue.
- If three nodes fail, loading fails, and querying continues.
- If four nodes fail, both loading and querying fail.

## 15-Node Cluster

This example assumes a cluster of 15 Foundation Nodes. The configuration in this setup includes five overprovisioned nodes, exceeding the Parity Width nodes. This number of overprovisioned nodes would be inefficient and unable to provide the full benefit in a real system, but this example demonstrates what happens if you added extra nodes at a later time to a cluster, which the storage space would recognize as overprovisioned.

```
CREATE STORAGE SPACE ocident WIDTH = 10, PARITY_WIDTH = 3;
```

- The WIDTH = 10 parameter means the system stores a segment group on 10 of the 15 nodes.
- The five remaining nodes are overprovisioned.
- The PARITY\_WIDTH = 3 parameter means each segment in the group contains enough parity bits to restore lost data for up to three nodes. In effect, this means parity bits comprise 20 percent of storage.

### Tolerance Scenarios

This setup provides resiliency for both querying and loading, but having more than three overprovisioned nodes provides no benefit because it exceeds three Parity Width nodes.

This setup results in these outcomes if nodes become disabled:

- If one node fails, loading and querying continue.
- If two nodes fail, loading and querying continue.
- If three nodes fail, loading and querying continue.
- If four nodes fail, loading and querying both fail because this breaches the Parity Width tolerance.

### Related Links

[Core Elements of an Ocident System](#)

[Cluster and Node Management](#)

## 7.4 Ocident Software Upgrade

To upgrade the Ocident® software, you must upgrade the DEB or RPM package on each node in the cluster. Versions of the Ocident software must match across the cluster, so upgrades must be done in an all-or-nothing fashion. Contact Ocident Support for assistance with the upgrade process.

The Ocient System does not support the downgrade of the software.

Consult the release notes for the target version. The version might have changes to any of the supported operating systems, the required operating system configuration, or the `rolehostd` configuration.

### Ocient System Prerequisites

- An existing Ocient installation is present on each node in the cluster.
- The RPM or DEB file is locally available on the file system.
- No jobs are running on the Ocient cluster.
- All nodes have stopped the `rolehostd` service. For details about stopping the `rolehostd` service, see the [Startup and Shutdown Procedure](#).

### RPM Upgrade

Consult the documentation of the package manager of your operating system for specific options and command-line syntax.

Use this command to upgrade using an Ocient RPM package from the local file system. Include `./` or specify the full path to the RPM file, otherwise the `dnf` command attempts to search the configured repositories instead of installing the local RPM file. In this example, replace `XX` with the Ocient package version of the provided file.

```
sudo dnf install ./ocient-XX-x86_64.rpm
```

After the upgrade is complete, you must run the startup procedures on all nodes. For details, see [Startup and Shutdown Procedure](#).

### DEB Upgrade

Use this command to upgrade using an Ocient DEB package from the local file system. Include `./` or specify the full path to the DEB file, otherwise the `apt` command attempts to search the configured repositories instead of installing the local DEB file. In this example, replace `XX` with the Ocient package version of the provided file.

```
sudo apt install ./ocient-XX-amd64.deb
```

After the upgrade is complete, you must run the startup procedures on all nodes. For details, see the [Startup and Shutdown Procedure](#).

### Related Links

[Maintenance Overview](#)

[Ocient Hyperscale Data Warehouse Release Notes](#)

## 7.5 Maintenance Overview

Ocient® System Administrators need to perform maintenance for three primary reasons:

- When the Ocient software requires action such as rebuilding lost or damaged segments
- When firmware or OS software updates are required
- When drive, memory, or system hardware failures occur

The maintenance documentation begins with Ocient software-related maintenance and proceeds to hardware-related maintenance.

- [Guide to Rebuilding Segments](#)
- [Startup and Shutdown Procedure](#)

- [Power Safety Recommendations](#)
- [NVMe Drive Firmware Upgrade Process](#)
- [Replace an NVMe Node Drive](#)
- [Replace Nodes](#)
  - [Replace Nodes](#)
  - [Replace Nodes](#)
  - [Replace Nodes](#)
  - [Replace Nodes](#)
- [Backup and Restore](#)

### 7.5.1 Guide to Rebuilding Segments

Data in an Ocident® System is erasure coded to provide resilience to disk and Foundation Node failures. The resilience of a system depends on the parity width of the storage space, which represents the number of associated disks (or nodes) that can fail without interruption of service or data loss.

For details about storage spaces and parity width, see [Configure Storage Spaces](#).

#### Data Segment Statuses

You can check the status of your system segment groups by querying the `sys.segment_groups` system catalog table. For details, see System Catalog.

This table describes the states for data segments.

| Status     | Description                                                                                                                                  | Recovery Process                                                                                                                                  |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| INTACT     | The normal and operable status.                                                                                                              | No recovery needed.                                                                                                                               |
| DAMAGED    | The segment failed a checksum, meaning the segment data is corrupted and unusable.                                                           | If you have sufficient parity width, you can recover a damaged segment by invoking a rebuild segment task.                                        |
| MISSING    | The segment is on a node or disk that is currently offline. If the node or disk rejoins the cluster, it can transition to the INTACT status. | When a disk or Foundation Node is permanently removed, you can perform a rebuild task to recover the data. This requires sufficient parity width. |
| REBUILDING | The segment is in recovery after damage or missing segment data.                                                                             | Recovery is already in progress.                                                                                                                  |

#### Recovery Considerations

If a segment has the DAMAGED or MISSING status, queries can proceed by reconstructing the data on demand using the remaining erasure-coded data in the segment group. Having a non-INTACT status means that I/O performance is significantly reduced.

To restore full performance, you need to run a segment rebuild task. Segment rebuilding is not automatic. The system administrator must manually invoke it.

A segment rebuild fails, and data is lost completely if the number of segments with the DAMAGED status in a segment group exceeds the parity width of the storage space. To avoid data loss:

- Provision the parity width of the storage space at or above the number of expected concurrent node failures.
- Rebuild damaged or permanently missing segments as soon as possible.

#### Checking for Abnormal Segments

You can find any segments that need a rebuild by querying for segment groups with an abnormal status.

#### Examples

## Finding Faulty Segment Groups

This example query finds any segment groups that have the DAMAGED or MISSING statuses.

```
SELECT * FROM sys.segment_groups WHERE status IN ('DAMAGED', 'MISSING');
```

Output

| "id"                 | "cluster_id"                          | "segment_type" | "status" | "primary_owr" |
|----------------------|---------------------------------------|----------------|----------|---------------|
| "720576045199095878" | "074c32ec-4f92-4718-ada1-5eb55eafdc3" | TKT_SEGMENT    | DAMAGED  |               |
| "716072445571725367" | "074c32ec-4f92-4718-ada1-5eb55eafdc3" | TKT_SEGMENT    | DAMAGED  |               |

## Finding Clusters and Nodes with Faulty Segment Groups

Inspect the count of damaged groups by cluster and node.

```
SELECT
  c.name AS cluster_name,
  n.name AS node_name,
  g.status AS segment_group_status,
  seg.status AS segment_status,
  seg.kind,
  COUNT(*) AS segment_count
FROM
  sys.segment_groups g
LEFT JOIN
  sys.clusters c
  ON c.id = g.cluster_id
LEFT JOIN
  sys.stored_segments seg
  ON seg.segment_group_id = g.id
LEFT JOIN
  sys.nodes n
  ON n.id = seg.node_id
WHERE
  g.status <> 'INTACT'
  AND (seg.status <> 'INTACT' OR seg.status IS NULL)
GROUP BY 1,2,3,4,5
ORDER BY 1,2,3,4,5;
```

Output

| "cluster_name"     | "node_name" | "segment_group_status" | "segment_status" | "kind"  | "segment_count" |
|--------------------|-------------|------------------------|------------------|---------|-----------------|
| foundation_cluster | foundation0 | DAMAGED                |                  | VIRTUAL | 2               |

## Starting a Segment Rebuild Task

A user with System Administrator privileges can start a segment rebuild using the CREATE TASK TYPE REBUILD SQL statement. You cannot cancel a segment rebuild task after it is started.

Most commonly, a rebuild task repairs all damaged or missing segments across the system.

### Example

Create a rebuild task.

```
CREATE TASK TYPE REBUILD;
```

The system can continue to perform queries while rebuilding segments, but the process can impact I/O performance.

## Advanced Rebuild Commands

Rebuild tasks can also execute on specific Foundation Nodes or clusters. For information on fine-tuning rebuild tasks, see [Distributed Tasks](#).

## Checking Rebuild Task Status

Monitor the status of current and past segment rebuild tasks from the `sys.subtasks` system catalog table. For details, see System Catalog.

```
SELECT * FROM sys.subtasks WHERE task_type = 'rebuild';
```

This table describes the statuses for a rebuild task.

| Status   | Status Detail        | Description                                                                                                     | Next Steps                                                                                                                                                                                  |
|----------|----------------------|-----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| complete | no_work              | The segment groups were already available and healthy. No rebuilding was needed.                                | None                                                                                                                                                                                        |
| complete | complete             | Rebuild completed successfully.                                                                                 | None                                                                                                                                                                                        |
| running  | rebuild_in_progress  | Rebuild is in process.                                                                                          | You can monitor the progress of the rebuild task by checking the <code>percent_complete</code> and <code>elapsed_seconds</code> in the <code>sys.rebuild_tasks</code> system catalog table. |
| failed   | rebuild_not_possible | The number of missing or damaged segments exceeds the parity width, and the data cannot be recovered currently. | If missing segments are available on an offline drive, you can attempt another rebuild task when that drive is made available to the system. Otherwise, you cannot recover the data.        |
| failed   | rebuild_no_space     | There is not enough space available to rebuild the segment.                                                     | You can complete the rebuild by truncating other data to free up space.                                                                                                                     |
| failed   | failed_on_node       | The cluster lost its connection to the node before the rebuild was completed.                                   | This is a transient error. You can retry the rebuild task.                                                                                                                                  |
| failed   | error                | An unexpected internal error occurred.                                                                          | Review error message details using the <code>error_message</code> column in the <code>sys.rebuild_tasks</code> system catalog table.                                                        |

## Related Links

[Distributed Tasks](#)

[Compute-Adjacent Input and Output on Large Working Sets](#)

### 7.5.2 Startup and Shutdown Procedure

Ocient® operates as a distributed system, so properly shutting down the system and preventing sudden power loss to nodes is important to ensure data integrity is maintained.

Execution of this process requires System Administrator access to the database and `sudo` privilege on the node being started or shut down. This procedure is not applicable to fully managed Ocient Systems.

#### Startup Procedure

The startup procedure can be used for starting the `rolehostd` process on a running node as well as powering on nodes in an Ocient System. In most installations, the `rolehostd` process is configured to automatically start on system boot up.

The `rolehostd` process is the main executable for an Ocient System. During the startup of the `rolehostd` process, every non-Administrator Node reaches out to one of the Administrator Nodes. These nodes require a “quorum” for the Ocient System to startup and operate. To establish a quorum, a majority of Administrator Nodes must be actively running the `rolehostd` process. For example, when an Ocient System has three Administrator Nodes, at least two must be active to establish quorum and service requests from other nodes.

Nodes in an Ocient System should be started in the following sequence for smoothest operation:

Queries are not completed until Foundation Nodes are online.

1. Start the `rolehostd` process on all dedicated **Administrator Nodes** and **SQL Nodes** using the `systemctl start rolehostd` command.
2. Start the `rolehostd` process on all **Foundation Nodes** using the `systemctl start rolehostd` command.
3. Start the `rolehostd` process on all **Loader Nodes** using the `systemctl start rolehostd` command.
4. After the `rolehostd` process has finished starting on all Loader Nodes, start the Loading and Transformation service using the `systemctl start lat` command.

#### Shutdown Procedure

The shutdown procedure can be used for stopping the `rolehostd` process as well as powering off nodes in the Ocient System.

Properly sequencing the shutdown of nodes in an Ocient System ensures a smooth power down process and reduces the appearance of errors to users in loading and querying as the system goes offline. The `rolehostd` process must be running on nodes with the Administrator Role for Global Dictionary Compression (GDC) lookup to work during loading and querying of data. If these nodes are shut down first, this will cause GDC lookups to fail, resulting in failures for both loading and querying of data.

It is recommended to stagger powering on of nodes by few seconds to reduce the surge in power distribution circuits.

When you execute `systemctl kill -s SIGKILL rolehostd`, the Ocient System sends SIGKILL to the `rolehostd` process. The node stops immediately without waiting for any queries or loading that are active on the node to finish. Some potential errors might be visible to the user. To avoid the errors, use the *quiescing* process.

The Ocient System should be stopped in the following sequence:

1. Stop the Loading and Transformation service on all Loader Nodes using the `systemctl stop lat` command.
2. Stop the `rolehostd` process on all Loader Nodes the `systemctl kill -s SIGKILL rolehostd` command.
3. Stop the `rolehostd` process on all Foundation Nodes the `systemctl kill -s SIGKILL rolehostd` command.  
Any queries issued after this point will fail.
4. Stop the `rolehostd` process on all SQL Nodes using the `systemctl kill -s SIGKILL rolehostd` command.
5. Stop the `rolehostd` process on all Administrator Nodes the `systemctl kill -s SIGKILL rolehostd` command.

## Quiescing Nodes

Quiescing nodes can be used for gracefully stopping the `rolehostd` process without interrupting loading or running queries. This process is most useful for maintenance purposes or for node upgrades. *Quiescing* refers to the graceful shut down of a node, whereas *quiesced* refers to a node that has completed quiescing and is now fully shut down.

After the node shutdown, quiescing nodes stop interacting with the rest of the system. Only quiesce one node at a time. When you quiesce multiple nodes at the same time, the quiesce process can freeze or lose forward progress for queries and loading. At startup, the node automatically rejoins the system and participates normally.

To issue a quiesce, execute `systemctl stop rolehostd`. This command sends the `SIGTERM` signal to the `rolehostd` process. The node begins the quiesce process, waits for all relevant queries or loading to finish, and shuts down. If a long-running query causes the quiesce to freeze, kill the query to finish the quiesce process.

During the quiesce process, a SQL Node continues to accept new connections from clients, such as JDBC or pyocient. However, when these clients run queries, explains, or DDL updates, the Ocient System redirects the clients to another online SQL Node, if one exists. A user might connect or run a query with the `force` option, which overrides the redirect behavior and forces the command to run on the SQL Node in the quiesce process. In this case, the Ocient System does not guarantee the successful completion of the query.

A Foundation Node in the quiesce process waits for any segment rebuilds that are in progress to finish. When you interrupt or fail the rebuild, the quiesce process finishes shortly.

When a Loader Node is in the middle of loading and you initiate the quiesce process on this node, stop the Loading and Transform (LAT) service that is on the same node.

Quiesce has a default time limit of 30 minutes, controlled by the `systemd` service. After a node quiesces for 30 minutes, the `systemd` service sends the `SIGKILL` signal to `rolehostd` and forcefully kills the node. To change this timeout, edit the `TimeoutStopSec` parameter in the `rolehostd` `systemd` service file.

## Related Links

[NVMe Drive Firmware Upgrade Process](#)

### 7.5.3 Power Safety Recommendations

Like any distributed data processing system, power safety in Ocient® is important to ensure data integrity. Data integrity in Ocient comprises newly loaded data that is in the process of being written to storage and metadata files that contain information about the state of the system.

Ensure that you have Power Loss Protection included in the NVMe drives. The Ocient System does provide some protection against power loss. So, follow these best practices for power management.

## NVMe SSD Power Loss Protection (PLP)

The Ocient Hyperscale Data Warehouse uses NVMe Solid State Drives (SSD) to store data.

To improve I/O performance, SSDs cache data in a volatile memory temporary buffer before it is flushed to non-volatile flash memory. During a normal shutdown, the host system notifies the SSD that the system is shutting down so it can flush data to non-volatile storage. However, unexpected power loss can result in loss or corruption of this data if this volatile memory is not preserved. To protect against this data loss, NVMe drives validated for use with Ocient Systems must include Power Loss Protection (PLP) mechanisms. PLP are standard in enterprise and datacenter grade NVMe drives. They are typically implemented using super capacitors that can hold enough power to allow safe writing of cached data to non-volatile memory in the event of a power loss to ensure data integrity.

## Ocient Distributed System Protections

While data is loaded it temporarily resides in Memory (RAM) or Disk Caches. The data stored in either of these locations is safely persisted to disks during a planned shutdown of nodes. The operating system or software might not be able to persist this data to disk during an unplanned power loss or system crash, so Ocient is designed to handle this scenario gracefully.



Ocient provides redundancy mechanisms using clustering to prevent metadata loss due to a sudden power outage on a single node. The Administrator cluster consists of multiple nodes performing the Administrator role that maintain system metadata. This prevents metadata loss when a single node is lost or damaged.

Foundation storage clusters use parity configuration to protect data in the event of sudden power loss to a Foundation Node. In such an event, inflight queries might fail but without any permanent loss of data. Dynamic rebuilding of missing data allows a configurable number of Foundation Nodes to lose power and Ocient can continue to serve queries. If power loss damages a Foundation Node or causes data corruption, the data on the damaged Foundation Node can be rebuilt with the information available in parity bits on other nodes to fully recover lost data.

This might not hold true in simultaneous power loss on more than one Administrator Node or simultaneous power loss on more Foundation Nodes than the number of parity bits in the storage cluster. Such coordinated power loss could leave an Ocient System in a state where Administrator metadata is corrupted or damaged data on Foundation Nodes cannot be fully rebuilt. To prevent simultaneous power loss, the following best practices are recommended.

### Power Safety Best Practices

The risk of complete system loss can be greatly reduced by avoiding sudden loss of power to an Ocient System. Best practices to reduce this risk are:

- NVMe Drives must include PLP.
- Each Ocient node should be configured with redundant power supplies where each power supply is serviced by independent electrical circuits.
- A rack or Data Center (DC) level Uninterruptible Power System (UPS) should be available to provide constant power to Ocient nodes during a sudden power outage.
- When power loss occurs and the UPS system cannot provide power for an extended period of time, you must properly shut down Ocient nodes by using the [Startup and Shutdown Procedure](#).
- Ocient Systems should be hosted in Tier III or Tier IV Data Centers. These Data Centers provide redundant power and cooling systems and guarantee minimal down time.

### Related Links

[Install an Ocient System](#)

## 7.5.4 NVMe Drive Maintenance

### 7.5.4.1 NVMe Drive Firmware Upgrade Process

The Ocient® software uses NVMe drives for storage because of their higher I/O throughput than spinning media drives. These drives are referred to as data disks or data drives. When you run Ocient software, the Ocient System accesses these data drives using the User Mode I/O (UIO) driver during normal operation.

This procedure guides a System Administrator through binding the drives back to the NVMe driver and upgrading the drive firmware. The procedure includes steps for upgrading the firmware of only Intel® NVMe drives. For other drive models, follow a similar process using the firmware upgrade process from the manufacturer.

Stop the `rolehostd` process prior to working on the NVMe drives. Otherwise, when the `rolehostd` process is running, the process becomes unstable because it has open file handles on the drives.

Execution of this process requires the sudo privilege on the node where the drive firmware is updated.

#### Prerequisites

Perform the firmware upgrade on one node of any specified type (Foundation, Loader, SQL) at a time. You can execute the firmware upgrade in parallel on different types of nodes.

Perform the upgrade during a maintenance window because issues with the upgrade might prevent the node from returning to service.

The prerequisites for this procedure are:

1. The following packages are installed on the node undergoing firmware upgrade:
  - `smartmontools`
  - `nvme-cli`
  - `intelmas`
  - `sedutil`
2. The target firmware version for the upgrade is known and documented.
3. If you upgrade the data drive firmware on a Loader Node, prior to beginning the procedure, take the node out of the loading path by updating the LAT configuration. Before you shut down the `rolehostd` process, drain the pages that are held by the node.

The following example illustrates the firmware update commands used for an Intel NVMe SSD. Please substitute the appropriate command for your drive manufacturer in each of the steps that use the `intelmas` command.

#### Firmware Upgrade Procedure

The following steps require `sudo` permissions on the node.

#### Prepare for the Firmware Upgrade

1. Log in to the node as an administrator user.
2. If you upgrade the drive firmware on a Foundation Node or Loader Node, save the status of the drives for later in the procedure. After you replace `<IP_ADDRESS_OF_THE_NODE>` with the IP address of your node, save the output of this command.

```
curl -s http://<IP_ADDRESS_OF_THE_NODE>:9090/v1/stats | jq -r '.[ ] | select (.name == "localStorageService.device.status" ) | "(.device) (.value)" '
```

Alternatively, you can run a query on the SQL Node to display the status of the individual storage device and firmware version. After you replace `<NODE_NAME>` with the name of your node, connect to any SQL Node on the system and run this query.

```
SELECT
  s.id AS serial_number,
  s.pci_address,
  s.device_status,
  s.firmware_version
FROM
  sys.nodes n
JOIN sys.storage_device_status s
  ON n.id = s.node_id
WHERE
  n.name = '<NODE_NAME>';
```

Sample output:

| serial_number      | pci_address  | device_status | firmware_version |
|--------------------|--------------|---------------|------------------|
| CVMD6356009M800GGN | 0000:3b:00.0 | ACTIVE        | VDV10184         |
| CVMD636400BZ800GGN | 0000:3c:00.0 | ACTIVE        | VDV10184         |
| CVMD636400JA800GGN | 0000:5e:00.0 | ACTIVE        | VDV10184         |
| CVMD6365003C800GGN | 0000:5f:00.0 | ACTIVE        | VDV10184         |
| CVMD6365003P800GGN | 0000:86:00.0 | ACTIVE        | VDV10184         |
| CVMD63650092800GGN | 0000:87:00.0 | ACTIVE        | VDV10184         |

```
PHLJ0431019J8POHGN|0000:88:00.0|ACTIVE      |VDV10184
PHLJ106500018POHGN|0000:89:00.0|ACTIVE      |VDV10170
```

3. If you upgrade the drive firmware on a SQL Node, save the output of the `df` command.

Stop the `rolehostd` Process and Attach the Data Drives

1. Stop the `rolehostd` process to ensure that the drive replacement does not impact system stability.

```
sudo systemctl stop rolehostd
```

2. Prevent the `rolehostd` process from starting after reboot.

```
sudo systemctl disable rolehostd
```

3. Confirm that the output of this command indicates that the `rolehostd` process is not running and that the process auto start on reboot setting is disabled.

```
sudo systemctl status rolehostd
```

When you upgrade the drive firmware, you can impact data reliability while the `rolehostd` process is running on the node.

4. Attach the data drives to the NVMe driver.

```
sudo /opt/ocient/scripts/nvme-driver-util.sh bind-nvme
```

5. Ensure that all drives are not the operating system drive and are attached to the NVMe driver in the output of this command. Any drives that did not bind to the NVMe driver in the previous step do not upgrade, so you must investigate any failure to bind to the NVMe driver.

```
/opt/ocient/scripts/nvme-driver-util.sh
```

**INCORRECT:** Eight data drives are attached to the UIO driver. This is the incorrect system state for performing a firmware upgrade.

```
./opt/ocient/scripts/nvme-driver-util.sh
NVMe device status
```

```
use option 'bind-uio' to bind uio_pci_generic driver to Ociant Payload/unpartitioned drives
use option 'bind-nvme' to bind nvme driver to drives
```

| BDF          | Numa Node | Driver name     | Device name |
|--------------|-----------|-----------------|-------------|
| 0000:01:00.0 | 0         | nvme            | nvme0n1     |
| 0000:3b:00.0 | 0         | uio_pci_generic | -           |
| 0000:3c:00.0 | 0         | uio_pci_generic | -           |
| 0000:5e:00.0 | 0         | uio_pci_generic | -           |
| 0000:5f:00.0 | 0         | uio_pci_generic | -           |
| 0000:86:00.0 | 1         | uio_pci_generic | -           |
| 0000:87:00.0 | 1         | uio_pci_generic | -           |
| 0000:88:00.0 | 1         | uio_pci_generic | -           |
| 0000:89:00.0 | 1         | uio_pci_generic | -           |

**CORRECT:** The data drives are attached to the NVMe driver. This is the correct system state for performing a firmware upgrade.

```
./opt/ocient/scripts/nvme-driver-util.sh
NVMe device status
```

```
use option 'bind-uio' to bind uio_pci_generic driver to Ociant Payload/unpartitioned drives
```

use option 'bind-nvme' to bind nvme driver to drives

| BDF          | Numa Node | Driver name | Device name |
|--------------|-----------|-------------|-------------|
| 0000:01:00.0 | 0         | nvme        | nvme0n1     |
| 0000:3b:00.0 | 0         | nvme        | nvme1n1     |
| 0000:3c:00.0 | 0         | nvme        | nvme2n1     |
| 0000:5e:00.0 | 0         | nvme        | nvme3n1     |
| 0000:5f:00.0 | 0         | nvme        | nvme4n1     |
| 0000:86:00.0 | 1         | nvme        | nvme5n1     |
| 0000:87:00.0 | 1         | nvme        | nvme6n1     |
| 0000:88:00.0 | 1         | nvme        | nvme7n1     |
| 0000:89:00.0 | 1         | nvme        | nvme8n1     |

#### Perform Firmware Upgrade

1. Run this command and review the `FirmwareUpdateAvailable` field to identify the drives that need a firmware upgrade. Save the results of this step for comparison later in the process.

Example

```
sudo intelmas show -d Index, SerialNumber, DeviceStatus, Capacity, FirmwareUpdateAvailable -intelssd
```

Drive 8 needs the firmware update, but drive 7 does not need the update.

Output

```
- 7 Intel SSD DC P4510 Series PHLJ0431019J8POHGN -

Capacity : 8001.56 GB
DeviceStatus : Healthy
FirmwareUpdateAvailable : The selected drive contains current firmware as of this tool release.
Index : 7
SerialNumber : PHLJ0431019J8POHGN

- 8 Intel SSD DC P4510 Series PHLJ106500018POHGN -

Capacity : 8001.56 GB
DeviceStatus : Healthy
FirmwareUpdateAvailable : Firmware=VDV10170 Bootloader=VB1B0172
Index : 8
SerialNumber : PHLJ106500018POHGN
```

2. Execute this command for every drive in the previous step that indicates an old firmware in the `FirmwareUpdateAvailable` field. Replace `<DRIVE_INDEX>` with the number shown in the `Index` field from the output of the previous command for all drives that require the update.

```
sudo intelmas load -intelssd <DRIVE_INDEX>
```

3. Ensure that the output shows `Firmware updated successfully`. This is the sample log output for the upgrade.

```
- intelmas load -intelssd 8
WARNING! You have selected to update the drives firmware!
Proceed with the update? (Y|N): Y
Checking for firmware update...

- Intel SSD DC P4510 Series PHLJ106500018POHGN -

Status : Firmware updated successfully. Please reboot the system.
```

4. After you upgrade all drives that require the firmware upgrade, reboot the node using the `reboot` command.

After the node completes the reboot, log in to the node as an administrator.

5. Attach the data drives to the NVMe driver using this command.

```
sudo /opt/ocient/scripts/nvme-driver-util.sh bind-nvme
```

6. Compare the output of this command with the output of the same command captured earlier in this process. Ensure that the reported drive status and capacity are the same as before the upgrade and that the firmware shown is current.

```
sudo intelmas show -d Index, SerialNumber, DeviceStatus, Capacity, FirmwareUpdateAvailable -intelssd
```

7. Reattach the data drives to the UIO driver.

```
sudo /opt/ocient/scripts/nvme-driver-util.sh bind-uio
```

Enable the `rolehostd` Process and Perform Final Checks

1. Enable the `rolehostd` process to start after reboot and then start the process.

```
sudo systemctl enable rolehostd && sudo systemctl start rolehostd
```

2. Ensure that the `rolehostd` process is enabled and started.

```
sudo systemctl status rolehostd
```

3. If you upgrade the firmware of the drive on a Foundation Node or Loader Node, ensure that the output of this command matches the output of the same command captured earlier in this process.

```
curl -s http://<IP_ADDRESS_OF_THE_NODE>:9090/v1/stats | jq -r '[] | select (.name == "localStorageService
```

Alternatively, you can run a query on the SQL Node to capture the individual device status and firmware version. Ensure that the drives and statuses in this query match the results from the query you executed earlier in this process. After you replace the node name `<NODE_NAME>` with the name of your node, connect to any SQL Node on the system and execute this query.

```
SELECT
    s.id as serial_number,
    s.pci_address,
    s.device_status,
    s.firmware_version
FROM
    sys.nodes n
JOIN sys.storage_device_status s
    ON n.id = s.node_id
WHERE
    n.name = '<NODE_NAME>';
```

Sample output:

| serial_number      | pci_address  | device_status | firmware_version |
|--------------------|--------------|---------------|------------------|
| -----              | -----        | -----         | -----            |
| CVMD6356009M800GGN | 0000:3b:00.0 | ACTIVE        | VDV10184         |
| CVMD636400BZ800GGN | 0000:3c:00.0 | ACTIVE        | VDV10184         |
| CVMD636400JA800GGN | 0000:5e:00.0 | ACTIVE        | VDV10184         |
| CVMD6365003C800GGN | 0000:5f:00.0 | ACTIVE        | VDV10184         |
| CVMD6365003P800GGN | 0000:86:00.0 | ACTIVE        | VDV10184         |
| CVMD63650092800GGN | 0000:87:00.0 | ACTIVE        | VDV10184         |
| PHLJ0431019J8POHGN | 0000:88:00.0 | ACTIVE        | VDV10184         |
| PHLJ106500018POHGN | 0000:89:00.0 | ACTIVE        | VDV10184         |

4. If you upgrade the firmware of the drive on a SQL Node, ensure that the output of the `df` command shows the same mount points as those captured earlier in this process.

Related Links

System Catalog

#### 7.5.4.2 Replace an NVMe Node Drive

Ocient® software uses NVMe drives for storage because of their higher I/O throughput than spinning media drives. These drives are referred to as data disks or drives.

In Ocient, these drives are accessed through a User Mode I/O (UIO) driver, so they cannot be accessed using the common set of Linux® commands (e.g., `df/du/ls/file/parted/etc`).

Working on these drives requires the `rolehostd` process to be stopped on the node, because the process has open file handles on the drives and making changes to the drive can make the process unstable.

Execution of this process requires `sudo` privilege on the Foundation Node on which the drive is being replaced.

#### Detection and Alerting

There are two approaches for detecting issues with NVMe drives on a Foundation Node. The device status of all drives on a specific Foundation Node can be checked using a SQL query of the system catalog to ensure all are active. Secondly, the presence of NVMe drives on the PCI bus can be inspected. You can run each command on a Foundation Node to assess the health of the drives on that node. Depending on the failure mode of a device, either of these methods could reveal an issue with an NVMe drive.

In the case of an operating system drive failure, contact Ocient Support.

#### Check Drive Status Using System Catalog Tables

SQL queries on the `sys.storage_device_status` system catalog table provide a status of all payload drives. Connect to any SQL Node on the system and execute this query, replacing `<NODE_NAME>` with the name of your node.

```
SELECT
  n.name AS node_name,
  s.node_id,
  s.id AS serial_number,
  s.pci_address,
  s.device_status,
  s.device_model
FROM
  sys.nodes n
JOIN sys.storage_device_status s
  ON n.id = s.node_id
WHERE
  n.name = '<NODE_NAME>';
```

#### Output

| node_name | node_id                              | serial_number      | pci_address  | device_status | device_model |
|-----------|--------------------------------------|--------------------|--------------|---------------|--------------|
| lts2      | f4845f46-dd0a-48bd-a2ed-614d5029e5df | CVMD6356009M800GGN | 0000:3b:00.0 | ACTIVE        | PCIe Data C  |
| lts2      | f4845f46-dd0a-48bd-a2ed-614d5029e5df | CVMD636400BZ800GGN | 0000:3c:00.0 | ACTIVE        | PCIe Data C  |
| lts2      | f4845f46-dd0a-48bd-a2ed-614d5029e5df | CVMD636400JA800GGN | 0000:5e:00.0 | ACTIVE        | PCIe Data C  |
| lts2      | f4845f46-dd0a-48bd-a2ed-614d5029e5df | CVMD6365003C800GGN | 0000:5f:00.0 | ACTIVE        | PCIe Data C  |
| lts2      | f4845f46-dd0a-48bd-a2ed-614d5029e5df | CVMD6365003P800GGN | 0000:86:00.0 | ACTIVE        | PCIe Data C  |
| lts2      | f4845f46-dd0a-48bd-a2ed-614d5029e5df | CVMD63650092800GGN | 0000:87:00.0 | ACTIVE        | PCIe Data C  |

|      |                                                                             |             |
|------|-----------------------------------------------------------------------------|-------------|
| lts2 | f4845f46-dd0a-48bd-a2ed-614d5029e5df CVMD6365009H800GGN 0000:88:00.0 ACTIVE | PCIe Data C |
| lts2 | f4845f46-dd0a-48bd-a2ed-614d5029e5df CVMD6371005A800GGN 0000:89:00.0 FAILED | PCIe Data C |

The possible storage device statuses are **ACTIVE** or **FAILED**. In this sample output, the device in PCI Address 0000:89:00.0 needs to be replaced.

If an NVMe drive is completely unavailable, it might not appear in the `sys.storage_device_status` table. For example, if a drive is not seated properly or if the node has rebooted and the drive is undetectable by Ocint. In this case, see the [Replace an NVMe Node Drive](#) to identify the serial number and PCI address of the drive.

In the event that a drive has failed or is experiencing issues, errors are logged in `/var/log/messages` and `/var/opt/ocient/log/rolehostd.log*` files. Inspection of these log files can assist with root cause analysis of the drive failure.

#### Troubleshooting

1. To locate the failed drive in the chassis, use the query results from the [Replace an NVMe Node Drive](#) query to identify the PCI Address of the failed drive.
2. Refer to the chassis diagram for your system to identify the slot of the failed drive based on the PCI address. Three common chassis and drive bay mappings are listed below:

#### Supermicro SYS-1029U-TN10RT Hot swap bays

|              |              |              |              |              |
|--------------|--------------|--------------|--------------|--------------|
| NVMe 3c:00.0 | NVMe 5f:00.0 | NVMe 61:00.0 | NVMe 87:00.0 | NVMe 89:00.0 |
| NVMe 3b:00.0 | NVMe 5e:00.0 | NVMe 60:00.0 | NVMe 86:00.0 | NVMe 88:00.0 |

#### Recovery

##### Prerequisites

- An NVMe drive of the same model and the same or larger capacity as the failed drive.
- The drive firmware is upgraded to the latest version using the [NVMe Drive Firmware Upgrade Process](#).

##### Replacement Procedure

1. Log in to the Foundation Node that needs a drive replacement as an administrator user and stop the `rolehostd` process on this node to ensure that the drive replacement does not impact stability of the process:

```
sudo systemctl stop rolehostd
```

2. Ensure that the `rolehostd` process has stopped. The result of this command should indicate that the process is Stopped.

```
sudo systemctl status rolehostd
```

3. If using encrypted OPAL drives, backup the `localKeyStore` directory. This directory can be empty or absent on systems that do not have OPAL drives.

```
sudo cp -pR /var/opt/ocient/localKeyStore /root
```

4. Physically remove the failed drive from the host and execute `nvme-driver-util.sh` to ensure that the failed drive is not showing up in the output.

```
/opt/ocient/scripts/nvme-driver-util.sh
```

##### Output

```
[admin@go-lts9 ~]$ /opt/ocient/scripts/nvme-driver-util.sh
NVMe device status
```

use option 'bind-uio' to bind uio\_pci\_generic driver to Ocident Payload/unpartitioned drives  
 use option 'bind-nvme' to bind nvme driver to drives

| BDF          | Numa Node | Driver name     | Device name |
|--------------|-----------|-----------------|-------------|
| 0000:3b:00.0 | 0         | uio_pci_generic | -           |
| 0000:3c:00.0 | 0         | uio_pci_generic | -           |
| 0000:5e:00.0 | 0         | uio_pci_generic | -           |
| 0000:5f:00.0 | 0         | uio_pci_generic | -           |
| 0000:86:00.0 | 1         | uio_pci_generic | -           |
| 0000:87:00.0 | 1         | uio_pci_generic | -           |
| 0000:88:00.0 | 1         | uio_pci_generic | -           |

[admin@go-lts9 ~]\$

5. Insert the replacement drive in the chassis and ensure that it shows up in output of the `nvme-driver-util.sh` script.

```
/opt/ocient/scripts/nvme-driver-util.sh
```

#### Output

```
[admin@go-lts9 ~]$ /opt/ocient/scripts/nvme-driver-util.sh
NVMe device status
```

use option 'bind-uio' to bind uio\_pci\_generic driver to Ocident Payload/unpartitioned drives  
 use option 'bind-nvme' to bind nvme driver to drives

| BDF          | Numa Node | Driver name     | Device name |
|--------------|-----------|-----------------|-------------|
| 0000:3b:00.0 | 0         | uio_pci_generic | -           |
| 0000:3c:00.0 | 0         | uio_pci_generic | -           |
| 0000:5e:00.0 | 0         | uio_pci_generic | -           |
| 0000:5f:00.0 | 0         | uio_pci_generic | -           |
| 0000:86:00.0 | 1         | uio_pci_generic | -           |
| 0000:87:00.0 | 1         | uio_pci_generic | -           |
| 0000:88:00.0 | 1         | uio_pci_generic | -           |
| 0000:89:00.0 | 1         | nvme            | -           |

[admin@go-lts9 ~]\$

6. Run this command by replacing X and Y to match the device name using the replacement drive.

```
sudo nvme format /dev/nvmeXnY format --ses=1
```

This command formats the specified drive and erases all data on the drive. If the OS drive is an NVMe drive, ensure that it is not formatted.

7. Execute the `nvme-driver-util.sh` script to bind drives to the UIO driver:

```
sudo /opt/ocient/scripts/nvme-driver-util.sh bind-uio
```

8. Execute `nvme-driver-util.sh` script and confirm that the new drive is bound to `uio_pci_generic` driver.

```
/opt/ocient/scripts/nvme-driver-util.sh
```

#### Output



```
[admin@go-lts9 ~]$ /opt/ocient/scripts/nvme-driver-util.sh
NVMe device status
```

```
use option 'bind-uio' to bind uio_pci_generic driver to Ocident Payload/unpartitioned drives
use option 'bind-nvme' to bind nvme driver to drives
```

```
BDF          Numa Node    Driver name      Device name
0000:3b:00.0  0             uio_pci_generic  -
0000:3c:00.0  0             uio_pci_generic  -
0000:5e:00.0  0             uio_pci_generic  -
0000:5f:00.0  0             uio_pci_generic  -
0000:86:00.0  1             uio_pci_generic  -
0000:87:00.0  1             uio_pci_generic  -
0000:88:00.0  1             uio_pci_generic  -
0000:89:00.0  1             uio_pci_generic  -
[admin@go-lts9 ~]$
```

9. Start the rolehostd process.

```
sudo systemctl start rolehostd
```

10. Check the output of the sys.storage\_device\_status catalog table by connecting to a SQL Node and running by following query after replacing the <NODE\_NAME> with the name of your node. Ensure that all drives appear on the node with status of ACTIVE.

```
SELECT
  n.name AS node_name,
  s.node_id,
  s.id AS serial_number,
  s.pci_address,
  s.device_status,
  s.model
FROM
  sys.nodes n
JOIN sys.storage_device_status s
  ON n.id = s.node_id
WHERE
  n.name = '<NODE_NAME>';
```

Output

| node_name | node_id                              | serial_number      | pci_address  | device_status | model       |
|-----------|--------------------------------------|--------------------|--------------|---------------|-------------|
| lts2      | f4845f46-dd0a-48bd-a2ed-614d5029e5df | CVMD6356009M800GGN | 0000:3b:00.0 | ACTIVE        | PCIe Data C |
| lts2      | f4845f46-dd0a-48bd-a2ed-614d5029e5df | CVMD636400BZ800GGN | 0000:3c:00.0 | ACTIVE        | PCIe Data C |
| lts2      | f4845f46-dd0a-48bd-a2ed-614d5029e5df | CVMD636400JA800GGN | 0000:5e:00.0 | ACTIVE        | PCIe Data C |
| lts2      | f4845f46-dd0a-48bd-a2ed-614d5029e5df | CVMD6365003C800GGN | 0000:5f:00.0 | ACTIVE        | PCIe Data C |
| lts2      | f4845f46-dd0a-48bd-a2ed-614d5029e5df | CVMD6365003P800GGN | 0000:86:00.0 | ACTIVE        | PCIe Data C |
| lts2      | f4845f46-dd0a-48bd-a2ed-614d5029e5df | CVMD63650092800GGN | 0000:87:00.0 | ACTIVE        | PCIe Data C |
| lts2      | f4845f46-dd0a-48bd-a2ed-614d5029e5df | CVMD6365009H800GGN | 0000:88:00.0 | ACTIVE        | PCIe Data C |
| lts2      | f4845f46-dd0a-48bd-a2ed-614d5029e5df | CVMD637100GA800GGN | 0000:89:00.0 | ACTIVE        | PCIe Data C |

As shown in the sample output, the device replaced at PCI Address 0000:89:00.0 is now active. This indicates that it has been successfully replaced.

Next Steps: Rebuild Segments

If the failed drive had data on it prior to failure, a rebuild of data segments stored on the replaced drive is required after the drive is replaced. Refer to the [Guide to Rebuilding Segments](#) for instructions on how to rebuild the missing segments.

#### Alternative Methods to Identify Failed Drives

If the results of querying the `sys.storage_device_status` table do not indicate which drive has failed, alternative methods can be used to detect the failing drive. These methods require the command-line utility `jq`, but you can view the full results of the `curl` commands without it.

##### Option 1: Cross Reference API Results to Identify the PCI Address

You can cross-reference the result of two API commands to determine the PCI address of the failed drive that is not appearing. A drive that does not appear in the `stats` API or catalog table can be present in the `sysconfig` API output.

The output of the following API shows NVMe storage device statuses. Any device status other than 10 indicates a potential issue. Replace `<IP_ADDRESS_OF_THE_NODE>` with the IP Address of your node and inspect the results.

```
curl -s http://<IP_ADDRESS_OF_THE_NODE>:9090/v1/stats | jq -r '[] |select (.name == "localStorageService."
```

#### Output

```
$ curl -s http://1.2.3.4:9090/v1/stats | jq -r '[] |select (.name == "localStorageService.device.status"
BTLJ913605YR2POBGN 10
BTLJ913608AY2POBGN 10
BTLJ913608FR2POBGN 10
BTLJ91360D6Q2POBGN 10
BTLJ91360D6Y2POBGN 10
BTLJ916306LS2POBGN 10
BTLJ92020KP22POBGN 10
BTLJ92030AJM2POBGN 10
PHLJ9403002N2POBGN 10
PHLJ941100RW2POBGN 10
```

The possible device status values are listed in the table below:

| Value | Device Status   |
|-------|-----------------|
| 0     | INVALID_UNKNOWN |
| 10    | ACTIVE          |
| 20    | UNINITIALIZED   |
| 61    | CORRUPT         |
| 80    | FAILED          |

#### Check Device Presence

The output of the following API will show the configured NVMe drives on the Foundation Node. Depending on the failure mode, the output of this command can show a reduced number of drives. For example, if the Foundation Node chassis has 12 drive bays populated and only 11 appear in this result, it would indicate that one drive is not visible to Ocident. Run the following command by replacing `<IP_ADDRESS_OF_THE_NODE>` with the IP Address of your node:

```
curl -s http://<IP_ADDRESS_OF_THE_NODE>:9090/v1/sysconfig | jq -r '.storage_devices[] | select (.use != "LOCAL_FILE_SYST
```

#### Output

```
$ curl -s http://1.2.3.4:9090/v1/sysconfig | jq -r '.storage_devices[] | select (.use != "LOCAL_FILE_SYST
BTLJ913608AY2POBGN 0000:3b:00.0
BTLJ913605YR2POBGN 0000:3c:00.0
BTLJ913608FR2POBGN 0000:5e:00.0
```

```

PHLJ9403002N2POBGN 0000:5f:00.0
BT LJ92020KP22POBGN 0000:60:00.0
BT LJ92030AJM2POBGN 0000:61:00.0
PHLJ941100RW2POBGN 0000:86:00.0
BT LJ91360D6Y2POBGN 0000:87:00.0
BT LJ916306LS2POBGN 0000:88:00.0
BT LJ91360D6Q2POBGN 0000:89:00.0

```

If the address of the failed drive cannot be determined in this way, proceed to Option 2 to identify the location of the drive.

#### Option 2: Inspect Drives Using Operating System Commands

If drives are not appearing in the Ocident outputs, they might be unavailable to the Ocident service. In this case, directly examine the drives on the system with standard disk utilities and operating system commands to find the serial number of drives that meet the following criteria:

1. The drive is not the operating system boot drive
2. The drive is not listed in the `sys.storage_device_status` table

Any drives that meet this criteria are not in use by Ocident and might need to be replaced or repaired. Identify the Serial Number and PCI Addresses of these drives in order to complete the replacement procedure.

Related Links

System Catalog

### 7.5.5 Expand and Rebalance System

If your system needs more resources, you can expand it by installing additional nodes and drives. The steps for expanding a system depend on the type of node you are adding.

These sections explain the steps for adding each Ocident® node type. Foundation Nodes include additional steps for adding more storage drives and rebalancing the system.

For information on how to replace nodes that have failed or are damaged, see [Replace Nodes](#).  
 For information on replacing drives that have failed or are damaged, see [Replace an NVMe Node Drive](#).

#### Prerequisites

These prerequisites apply to adding any node type:

- The process requires either the System Administrator role or the granted UPDATE privilege on SYSTEM.
- :CommentAny new nodes should have the same OS kernel version as other nodes in the system.
- :CommentAny new nodes should have the same Ocident System version as the other nodes. For details, see [Ocident Application Installation](#).
- Ensure that your system meets the requirements outlined in [Ocident System Bootstrapping](#).
- Before adding nodes, stop any query or loading process running on your system.

#### Add Foundation Nodes

A core component of an Ocident® System, the Foundation Nodes store user data in Ocident and perform the bulk of query processing. By adding more Foundation Nodes, a system can support extra storage capacity as well as better performance for query processing.

This process requires rebalancing the data to distribute the query processing across all available hardware, including newly installed disks or nodes.

System Considerations

- No individual cluster should have more than two petabytes of storage.

- Each cluster should have the same number of Foundation Nodes.

### Tutorial

Create the `/var/opt/ocient/bootstrap.conf` file on the new node using this YAML example. This YAML file should follow a similar format and parameters as the `bootstrap.conf` files on your other nodes.

Replace the `<FIRST_NODE_ADDRESS>` with the IP or Hostname of a node running the Administrator Role on your system. If you are not using DNS, use `nodeAddress` instead of `adminHost`.

Use a user account with System Administrator privileges for your system for the `adminUserName` and `adminPassword`.

```
adminHost: <FIRST_NODE_ADDRESS>
adminUserName: my_admin
adminPassword: example_password
```

Start the new node by using this command.

```
sudo systemctl start rolehostd
```

This process takes about a minute to complete. This step accepts the node into the system, but it has not been assigned a role or a storage cluster.

To validate that the node is on the system, execute this query using the `sys.nodes` system catalog table.

```
SELECT name, status
FROM sys.nodes
ORDER BY 1;
```

### Output

| name           | status   |
|----------------|----------|
| admin01        | ACCEPTED |
| admin02        | ACCEPTED |
| admin03        | ACCEPTED |
| loader01       | ACCEPTED |
| loader02       | ACCEPTED |
| loader03       | ACCEPTED |
| foundation01   | ACCEPTED |
| foundation02   | ACCEPTED |
| foundation03   | ACCEPTED |
| foundation04   | ACCEPTED |
| foundation05   | ACCEPTED |
| foundation06   | ACCEPTED |
| foundation07   | ACCEPTED |
| foundation08   | ACCEPTED |
| foundation09   | ACCEPTED |
| foundation10   | ACCEPTED |
| foundation11   | ACCEPTED |
| foundation12   | ACCEPTED |
| foundation_new | ACCEPTED |
| sql01          | ACCEPTED |
| sql02          | ACCEPTED |

The output shows the new node, named `foundation_new` in this example, listed as `ACCEPTED` in the `status` column.

Execute this `ALTER CLUSTER SQL` statement to add the new node to a storage cluster. In this example, replace `storage_cluster_1` and `foundation_new` with your cluster name and node name, respectively.

```
ALTER CLUSTER "storage_cluster_1"
ADD PARTICIPANTS "foundation_new";
```

This example adds only one new node. For information on adding multiple nodes, see [Cluster and Node Management](#).

Restart the `rolehostd` process on all nodes in the system by running the following series of commands at the shell terminal.

First, stop the `rolehostd` process on all nodes.

```
sudo systemctl kill -s SIGKILL rolehostd
```

Confirm that the `rolehostd` process is no longer running.

```
sudo systemctl status rolehostd
```

Start the `rolehostd` process on the system again.

```
sudo systemctl start rolehostd
```

You can verify the new node is active by executing this SQL query after you connect to the database.

```
SELECT n.name,
       ns.operational_status
FROM sys.node_status ns
JOIN sys.nodes n ON ns.node_id = n.id
ORDER BY n.name;
```

| name           | operational_status |
|----------------|--------------------|
| admin01        | Active             |
| admin02        | Active             |
| admin03        | Active             |
| loader01       | Active             |
| loader02       | Active             |
| loader03       | Active             |
| foundation01   | Active             |
| foundation02   | Active             |
| foundation03   | Active             |
| foundation04   | Active             |
| foundation05   | Active             |
| foundation06   | Active             |
| foundation07   | Active             |
| foundation08   | Active             |
| foundation09   | Active             |
| foundation10   | Active             |
| foundation11   | Active             |
| foundation12   | Active             |
| foundation_new | Active             |
| sql01          | Active             |
| sql02          | Active             |

Perform a rebalance task to distribute your data across your newly expanded system evenly. For details, see [Expand and Rebalance System](#).

### Add More Drives to Foundation Nodes

Foundation Nodes can support extra storage by adding additional NVMe drives. This tutorial shows the steps to integrate new drives into an existing Ocident System.

System Considerations

- For best performance, all Foundation Nodes should have equal storage capacity.
- No individual cluster should have more than two petabytes of storage.

#### Prerequisites

- The process requires **systemctl** access on your system OS.
- Any new NVMe drives added to the system must be blank and unpartitioned.

#### Tutorial

Shut down the **rolehostd** process from the shell prompt.

```
sudo systemctl stop rolehostd
```

Install the new storage drives in your system.

Restart the node from the shell prompt.

```
sudo systemctl restart rolehostd
```

The **rolehostd** process recognizes the new drive.

To confirm the new drives are active and running, you can connect to your system and query the **sys.storage\_device\_status** system catalog table.

To use this example query, replace **<NODE\_NAME>** with the name of the node where you are adding a drive.

```
SELECT
  n.name AS node_name,
  s.node_id,
  s.id AS serial_number,
  s.pci_address,
  s.device_status,
  s.device_model
FROM
  sys.nodes n
JOIN sys.storage_device_status s
  ON n.id = s.node_id
WHERE
  n.name = '<NODE_NAME>';
```

#### Output

| node_name   | node_id                              | serial_number                        | pci_address                                          |
|-------------|--------------------------------------|--------------------------------------|------------------------------------------------------|
| foundation0 | 9330a0b3-b3b7-4503-949c-043b196c0cc4 | 6156962f-fcf6-4299-bbd7-2618fc6f1d00 | /var/opt/ocient/6156962f-fcf6-4299-bbd7-2618fc6f1d00 |
| foundation0 | 9330a0b3-b3b7-4503-949c-043b196c0cc4 | e6a4b24f-0d49-4704-b7ce-18af163c0701 | /var/opt/ocient/e6a4b24f-0d49-4704-b7ce-18af163c0701 |

This output lists the serial numbers of all drives in the **foundation0** node, including their statuses.

Perform a rebalance task to distribute your data across your newly expanded system evenly. For details, see [Expand and Rebalance System](#).

#### Rebalance System

**REBALANCE** task execution redistributes your data evenly across your segment groups and clusters. Rebalance your system if you have recently installed new hardware, particularly new nodes or drives.

#### Prerequisites

- Only one rebalance task can execute at a time on the system. The system logs an error if you try to start a second rebalance task.
- You must have the System Administrator role, or be granted the UPDATE privilege on SYSTEM.

## Tutorial

Execute a `SELECT SQL` statement to view how the system has distributed the existing data using the `sys.tables` and `sys.nodes` system catalog tables.

```
SELECT t.name AS table_name,
       n.name AS node_name,
       su.used_bytes FROM sys.storage_used su
  JOIN sys.tables t ON t.id = su.table_id
  JOIN sys.nodes n ON n.id = su.node_id
ORDER BY t.name,
       n.name;
```

## Output

| table_name | node_name | used_bytes |
|------------|-----------|------------|
| table0     | lts0      | 48000000   |
| table0     | lts1      | 11000000   |
| table0     | lts2      | 79000000   |
| table0     | lts3      | 73000000   |
| table0     | lts4      | 85000000   |
| table0     | lts5      | 50000000   |
| table0     | lts6      | 51000000   |
| table0     | lts7      | 41000000   |
| table1     | lts0      | 66000000   |
| table1     | lts1      | 77000000   |
| table1     | lts2      | 73000000   |
| table1     | lts3      | 43000000   |
| table1     | lts4      | 76000000   |
| table1     | lts5      | 51000000   |
| table1     | lts6      | 71000000   |
| table1     | lts7      | 105000000  |

The query output shows how data is distributed across your nodes.

Execute the `REBALANCE` task named `rebalance_task` to reorganize all data in a balanced state using the `CREATE TASK SQL` statement.

```
CREATE TASK rebalance_task TYPE REBALANCE;
```

For details about the syntax for creating tasks, see [Distributed Tasks](#).

View the status of the tasks as the rebalance operation executes using the `sys.subtasks` system catalog table.

```
SELECT id,
       name,
       start_time,
       task_type,
       execution_type,
       status,
       details
FROM sys.subtasks
WHERE task_type = 'rebalance'
ORDER BY start_time DESC;
```

## Output

| id | name | start_time | task_type | execution_type |
|----|------|------------|-----------|----------------|
|----|------|------------|-----------|----------------|

|                                      |                            |           |               |
|--------------------------------------|----------------------------|-----------|---------------|
| ed68a070-d7f7-4c50-9b94-84b04b4503c2 | 2024-04-17 21:44:20.268000 | rebalance | intra_cluster |
| 9e6c3374-6e9c-4a84-9bf8-ba1db3d866eb | 2024-04-17 21:44:20.268000 | rebalance | intra_cluster |
| aebb75d8-e066-4813-b7e8-c272e5a3b8e9 | 2024-04-17 21:44:14.768000 | rebalance | inter_cluster |
| 5cec5095-8b3a-4868-a10d-35fe6a32e998 | 2024-04-17 21:44:14.564000 | rebalance | NULL          |

The output shows when the **REBALANCE** task is finished.

When the **REBALANCE** task runs, segments are in the **REBUILDING** state. After the Ocint System completes this task, all segments should transition to the **INTACT** state.

Execute the query from Step 1 again to see how the system reorganized the data across nodes.

```
SELECT t.name,
       n.name,
       su.used_bytes
FROM sys.storage_used su
     JOIN sys.tables t ON t.id = su.table_id
     JOIN sys.nodes n ON n.id = su.node_id
ORDER BY t.name,
         n.name;
```

*Output*

| name   | name_1 | used_bytes |
|--------|--------|------------|
| table0 | lts0   | 48000000   |
| table0 | lts1   | 39000000   |
| table0 | lts2   | 71000000   |
| table0 | lts3   | 53000000   |
| table0 | lts4   | 85000000   |
| table0 | lts5   | 50000000   |
| table0 | lts6   | 51000000   |
| table0 | lts7   | 41000000   |
| table1 | lts0   | 66000000   |
| table1 | lts1   | 77000000   |
| table1 | lts2   | 73000000   |
| table1 | lts3   | 43000000   |
| table1 | lts4   | 76000000   |
| table1 | lts5   | 68000000   |
| table1 | lts6   | 71000000   |
| table1 | lts7   | 88000000   |

The output shows the redistributed data.

The **REBALANCE** task redistributes data across segment groups, so there can be some variance in data across nodes.

## Add Loader Nodes

Adding more Loader Nodes to your system can improve the throughput of data loading and resiliency against loading failure. By having extra Loader Nodes, you can also dedicate sets of nodes for specific pipelines.

System Considerations



Before starting this process, ensure you meet the requirements in the [Expand and Rebalance System](#) section.

#### Tutorial

Stop any active data pipelines. Execute this SQL statement by replacing `pipeline_name` with the name of your data pipeline.

```
STOP PIPELINE <pipeline_name>;
```

Create the `/var/opt/ocient/bootstrap.conf` file on the new Loader Node using this YAML example. This YAML file should follow a similar format and parameters to the `bootstrap.conf` file on your other nodes.

Replace the `<FIRST_NODE_ADDRESS>` with the IP address or hostname of a node running the Administrator Role on your system. If you are not using a DNS, use `nodeAddress` instead of `adminHost`.

Specify a user account with System Administrator privileges for your system for the username `adminUserName` and password `adminPassword`.

```
adminHost: <FIRST_NODE_ADDRESS>
adminUserName: my_admin
adminPassword: example_password
```

Start the new node by using this command.

```
sudo systemctl start rolehostd
```

This process takes about a minute to complete. This step accepts the node into the system, but does not assign a role.

Add the `streamloader` role to the new node.

Execute this SQL statement by replacing `<new_node_name>` with the new node name.

```
ALTER NODE <new_node_name> ADD ROLE streamloader;
```

:CommentRestart the `rolehostd` process on the replacement node by running this command at the shell terminal on the replacement node.

```
sudo systemctl restart rolehostd
```

Restart the pipeline. Execute this SQL statement by replacing `pipeline_name` with the name of your data pipeline. Optionally, specify the Loader Node by name with the `USING LOADERS` keywords to prioritize it for this pipeline.

```
START PIPELINE <pipeline_name> USING LOADERS <new_node_name>;
```

If you use the legacy LAT service, you must stop loading and copy the LAT configuration files to any new Loader Nodes. For more information, see [Replace Nodes](#).

This step is unnecessary for systems that use Ocident data pipelines for loading.

## Add SQL Nodes

Adding more SQL Nodes to your system can improve query optimization and processing, particularly for aggregation and join operations. Extra SQL Nodes also provide system resiliency, especially when assigned the `admin` role.

#### System Considerations

Before starting this process, ensure you meet the requirements in the [Expand and Rebalance System](#) section.

#### Tutorial

Create the `/var/opt/ocient/bootstrap.conf` file on the new SQL Node using this YAML example. This YAML file should follow a similar format and parameters to the `bootstrap.conf` file on your other nodes.

Replace the `<FIRST_NODE_ADDRESS>` with the IP address or hostname of a node running the Administrator Role on your system. If you are not using a DNS, use `nodeAddress` instead of `adminHost`.

Specify a user account with System Administrator privileges for your system for the username `adminUserName` and password `adminPassword`.

```
adminHost: <FIRST_NODE_ADDRESS>
adminUserName: my_admin
adminPassword: example_password
```

Start the new node by using this command.

```
sudo systemctl start rolehostd
```

This process takes about a minute to complete. This step accepts the node into the system, but does not assign a role.

Add the `sql` role to the new node.

Execute this SQL statement by replacing `<new_node_name>` with the new node name.

```
ALTER NODE <new_node_name> ADD ROLE sql;
```

Optionally, you can also assign the `admin` role to the SQL Node. :CommentAt least one SQL Node must always fulfill this role. By default, the system assigns the `admin` role to the first SQL Node in the system.

For more information about the `admin` role, see [Node Configuration with the Administrator Role](#).

Execute this SQL statement by replacing `<new_node_name>` with the name of your new node.

```
ALTER NODE <new_node_name> ADD ROLE admin;
```

Restart the `rolehostd` process on the replacement node by running this command at the shell terminal on the replacement node.

```
sudo systemctl restart rolehostd
```

Assign the new SQL Node to a connectivity pool with the [Cluster and Node Management](#) statement.

In this example, the statement assigns the SQL Node `sql2` to the connectivity pool `cp1` with the IP address `111.1.1.1` and port number `4050` for listening. Specify the local IP address and port number `4050` to return to the client.

```
ALTER CONNECTIVITY_POOL cp1
  ADD PARTICIPANTS(
    NODE sql2
    LISTEN_ADDRESS '111.1.1.1'
    LISTEN_PORT 4050
    ADVERTISED_ADDRESS 'localhost'
    ADVERTISED_PORT 4050);
```

## Related Links

[Replace Nodes](#)

[Replace an NVMe Node Drive](#)

### 7.5.6 Replace Nodes

An Ocident® System requires a full node replacement in some scenarios. For example, the OS drive on a node might fail and necessitate a reinstallation and addition of the node. Other events, such as a complete node loss due to a hardware failure of the motherboard or damage to NVMe storage drives, can also cause you to replace a node.

An administrator should be familiar with the different types of nodes described in the [Core Elements of an Ocident System](#). There are many similarities between the procedures for different node types, and the differences are clearly noted in the replacement procedures.

This procedure begins with **Replacing Nodes** then guides a System Administrator through replacing any of the Ocient node types.

This procedure also includes instructions about the replacement of an OS drive (also called a boot drive) on an Ocient node because the procedure for the replacement is identical to the replacement of the entire node. Both require reinstallation of the operating system and restoration of the configuration backup files to reestablish the identity of the Ocient System.

You must work carefully with nodes that also perform the Administrator Role, because you must maintain Administrator consensus.

Execution of this process requires System Administrator access to the Ocient System Database and `sudo` privilege on the node being replaced.

### Detection and Alerting

You must monitor all nodes such that the loss of a node on the system triggers a node availability alert in the monitoring system.

This table shows the potential consequences of the loss of each type of node.

#### SQL Node loss

| Node Type | Impact of Node Loss                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SQL       | <p>Might impact query performance in an Ocient System.</p> <p>If the only remaining SQL Node is lost in a system, you cannot query that system.</p> <p>If the SQL Node performs the Administrator Role, then the resiliency of the system could reduce to hardware failure. You must maintain the number of nodes that perform the Administrator Role at all times to maintain the resiliency of the system.</p> |

#### Foundation Node Loss

| Node Type  | Impact of Node Loss                                                                                                                                                                                              |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Foundation | <p>Can impact query performance or the load of new data.</p> <p>The parity configuration of the storage space in the Ocient System determines if new data can be stored while a Foundation Node is disabled.</p> |

#### Loader Node Loss

| Node Type | Impact of Node Loss                                                                                                                                                                                                                                                         |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Loader    | <p>Can impact the throughput of the data load in an Ocient System.</p> <p>If the remaining Loader Nodes are insufficient to handle the incoming data, the data can lead to a backlog.</p> <p>If the only remaining Loader Node is lost in a system, data loading stops.</p> |

## Troubleshooting

You must replace a node in the event of a hardware failure of certain components. You can replace an OS drive, data drives, and RAM on most nodes. A node replacement might be necessary in the case of a CPU, motherboard, or Network Interface Card (NIC) replacement. The troubleshooting steps to recover an existing node are specific to the hardware model of the node.

Please contact Ociant Support for troubleshooting issues with a node.

## Recovery

### Prerequisites

- The SQL Nodes on the Ociant System are active and running Ociant (except for any SQL Nodes that are being replaced as part of this procedure).
- The OS has been installed and configured on the replacement node or OS drive for the node to be replaced.
- The replacement node has been installed with the same version of the Ociant Software as the rest of the Ociant System according to the [Ociant Application Installation](#).
- The `bootstrap.conf` file for the node has been created in the `/var/opt/ocient` directory on the replacement node. For details, see [Ociant System Bootstrapping](#).
- For Loader Nodes only:
  - The replacement node has been installed with the same version of the Loading and Transformation software as the rest of the Loader Nodes as described in [LAT Packaging and Installation](#).
  - The following files are available from the node before the failure. If these files are not available, please contact Ociant Support for assistance.

```
* /etc/lat/lat.conf
* /etc/lat/log4j2.xml
* /opt/lat/.lat-data/*
```

### Replacement Procedure Overview

The replacement procedure is similar for all node types, but there are differences due to the way that different node types coordinate with one another.

The general procedure is:

1. Remove the failed node from the Ociant System.
2. Register the new node with the system and Accept Node.
3. Assign roles and add to clusters where appropriate.
4. Start the new node.

If you want to reuse the name of a node, then you must rename the failed node and set the IP address. To perform these actions, use these statements.

Rename the node from `nodename` to `newnodename`.

```
ALTER NODE "nodename" RENAME TO "newnodename";
```

Set the IP address for the new node.

```
ALTER NODE "newnodename" SET ADDRESS 'newnodename';
```

## SQL Node Replacement Procedure

Step 1: Check SQL Node for the Administrator Role.

Before you replace a SQL Node, check whether the node performs the Administrator Role in addition to the SQL Node. If the node performs the Administrator Role, then follow this additional step to add this role to the new node.

Most SQL Nodes are configured to also perform the Administrator Role. When you remove nodes that perform the Administrator Role, you must replace that node with another node that performs the Administrator Role so that you maintain the level of redundancy in the RAFT protocol of the administrator.

Check the results by using this query.

```
SELECT
    n.name AS node_name,
    sr.service_role_type
FROM sys.nodes n
LEFT JOIN sys.service_roles sr ON sr.node_id = n.id
WHERE sr.service_role_type = 'admin';
```

```
node_name service_role_type
-----
```

```
----
```

```
sql0 admin
sql1 admin
sql2 admin
```

Step 2: Remove the Failed SQL Node from the Ocient System.

Remove the node role using the `ALTER NODE SQL` statement by replacing `sql0` with the node name of your node.

```
ALTER NODE "sql0" REMOVE ROLE sql;
```

Delete the node using the `DROP NODE SQL` statement by replacing `sql0` with the node name of your node.

```
DROP NODE "sql0";
```

If the failed node still exists in the Ocient System, the system does not add the replacement node.

Step 3: Register the New Node with the Ocient System.

Follow the [Replacing Nodes](#) to register the new node. This procedure uses the `rolehostd` process to start any new nodes.

When bootstrapping the replacement node, you can reuse the original node name or assign a new one. The node name of the newly registered node is used in subsequent steps.

Step 4: Add the SQL Role to the New Node.

To add a role, you must be a system-level user. Execute the following SQL statement from the SQL prompt, replacing `<NODE_NAME_OF_THE_NEW_NODE>` with the new node name.

```
ALTER NODE "<NODE_NAME_OF_THE_NEW_NODE>" ADD ROLE sql;
```

Step 5: Add the Administrator Role. (Only for SQL Nodes with the Administrator Role)

If the node had the Administrator Role in Step 1, then add the Administrator Role back to the node. Execute the following SQL statement from the SQL prompt, replacing `<NODE_NAME_OF_THE_NEW_NODE>` with the new node name.

```
ALTER NODE "<NODE_NAME_OF_THE_NEW_NODE>" ADD ROLE admin;
```

Step 6: Restart the Node.

Restart the `rolehostd` process on the replacement node by running this command at the shell terminal on the replacement node.

```
sudo systemctl restart rolehostd
```

## Loader Node Replacement Procedure

Step 1: Remove the Failed Loader Node from the Ocient System.

Remove the node role using the `ALTER NODE` SQL statement by replacing `loader0` with the name of your node.

```
ALTER NODE "loader0" REMOVE ROLE streamloader;
```

Delete the node using the `DROP NODE` SQL statement, replacing `loader0` with the name of your node.

```
DROP NODE "loader0";
```

If the failed node still exists in the Ocient System, the system does not add the replacement node.

Step 2: Register the New Node with the Ocient System.

Follow the [Replacing Nodes](#) to register the new node.

When bootstrapping the replacement node, you can reuse the original node name or assign a new one. The node name of the newly registered node will be used in subsequent steps.

Step 3: Add the `streamloader` Role to the New Node.

To add a role, you must be a system-level user. Execute the following SQL statement, replacing `<NODE_NAME_OF_THE_NEW_NODE>` with the new node name.

```
ALTER NODE "<NODE_NAME_OF_THE_NEW_NODE>" add role streamloader;
```

When adding the `streamloader` role, the node must be restarted if it is already running in order for the change to take effect.

Step 4: Configure the LAT Service.

Stop the LAT service by executing the following command at the shell prompt.

```
sudo systemctl stop lat
```

The LAT server configuration should be restored from the previous configuration. If a backup is unavailable, please contact Ocient Support for assistance or reference the other LAT configuration on the cluster to modify the default files.

- Copy the `lat.conf` backup file to `/etc/lat/lat.conf`.
- Copy the `log4j2.xml` backup file to `/etc/lat/log4j2.xml`.

Ensure that these two files have the appropriate ownership and permissions by executing the following commands.

```
chown lat:lat /etc/lat/lat.conf /etc/lat/log4j2.xml
chmod 644 /etc/lat/lat.conf /etc/lat/log4j2.xml
```

Step 5: Restart the Node.

Restart the `rolehostd` process on the replacement node by running this command at the shell terminal on the replacement node.

```
sudo systemctl restart rolehostd
```

Start the LAT service.

```
sudo systemctl start lat
```

Step 6: Update the LAT Pipeline if Required.

If a continuously streaming pipeline such as loading from Apache® Kafka® is operating on the Ocint System, then the Ocint System requires you to update the restored LAT to run the same pipeline to restore full loading throughput. This process is different if a long-running File Load is in process.

### Kafka pipeline procedure

| Pipeline Type  | Procedure                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Kafka Pipeline | <ol style="list-style-type: none"> <li>1. From the LAT Client, create the pipeline on the new LAT Node. This ensures the pipeline_id matches the existing LAT processes.</li> <li>2. Start the pipeline on the new LAT Node. The new Loader Node joins the existing Loader Nodes, and the load automatically rebalances across all participating Loaders with the same pipeline_id</li> </ol> |

### File load pipeline

| Pipeline Type      | Procedure                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| File Load Pipeline | File loads from sources like S3, local file system, or NFS require rebalancing of partitions in the event that a node is lost or added back to an Ocint System. In the event a node is lost, it is recommended that a rebalance is performed and the file load completes using the remaining nodes. The newly added node is not expected to participate in an already active file load. |

## Foundation Node Replacement Procedure

Step 1: Register the New Node with the Ocint System.

Follow the [Replacing Nodes](#) to register the new node. When bootstrapping the new node, you can reuse the original node name or assign a new one. This tutorial uses the node name of the newly registered node in subsequent steps. These examples use `foundation_new` for the new node name.

If you want to reuse the original node name (the name of the node being removed), you must first rename the original node to prevent conflict. For example, if the original node was named `foundation_original`, this can be performed with the SQL statement `ALTER NODE "foundation_original" RENAME TO "foundation_to_drop";`

Step 2: Add the New Node to a Specific Storage Cluster.

To operate as a Foundation Node, the node must be added to a storage cluster. To add a node to a storage cluster, you must be a system-level user. Execute the following SQL statement from the SQL prompt by replacing `storage_cluster_1` and `foundation_new` with your storage cluster name and node name, respectively.

```
ALTER CLUSTER "storage_cluster_1" ADD PARTICIPANTS "foundation_new";
```

Step 3: Restart the New Node.

Restart the `rolehostd` process on the new node by running this command at the shell terminal on the new node.

```
sudo systemctl restart rolehostd
```

The node should now be operational.

Step 4: Remove the Failed Foundation Node from the Ocident System.

Remove the original node from all clusters to which it belongs. For example, if the existing node is `foundation_original` and your storage cluster name is `storage_cluster_1`, execute the following SQL statement.

```
ALTER CLUSTER "storage_cluster_1" DROP PARTICIPANTS "foundation_original";
```

Delete the node using the `DROP NODE` SQL statement, replacing `foundation_original` with the name of your node.

```
DROP NODE "foundation_original";
```

If you renamed the failed original node in Step 1, be sure to use the correct name when dropping from the storage cluster and dropping the node.

Step 5: Rebuild Segments.

In most cases, after removing the original node, administrators follow the [Guide to Rebuilding Segments](#) to rebuild segments that were on the replaced Foundation Node.

---

## Metadata Node Replacement Procedure

A Metadata Node is a node that only performs the Administrator Role.

Step 1: Remove the Failed Administrator Node from the Ocident System.

Delete the node using the `DROP NODE` SQL statement by replacing `standaloneAdmin00` with the name of your node.

```
DROP NODE "standaloneAdmin00";
```

If the failed node still exists in the Ocident System, the system does not add the replacement node.

Step 2: Register the New Node with the Ocident System.

Follow the [Replacing Nodes](#) to register the new node.

When bootstrapping the replacement node, you can reuse the original node name or assign a new one. This tutorial uses the node name of the newly registered node in subsequent steps.

Step 3: Add the Administrator Role to the New Node.

To add a role, you must be a system-level user. Execute the following SQL statement from the SQL prompt, replacing `<NODE_NAME_OF_THE_NEW_NODE>` with the new node name.

```
ALTER NODE "<NODE_NAME_OF_THE_NEW_NODE>" add role admin;
```

Step 4: Restart the Node.

Restart the `rolehostd` process on the replacement node by running this command at the shell terminal on the replacement node.

```
sudo systemctl restart rolehostd
```

---



## Bootstrapping Node Procedure

To add a node to the system, follow these steps after meeting the prerequisites described in [Ocient System Bootstrapping](#). These steps summarize the bootstrapping process.

All prerequisites should be validated prior to bootstrapping. Ensure that you have set the correct `adminUserName` and `adminPassword` in the `bootstrap.conf` file. Any advanced configuration parameters should also be specified in the `bootstrap.conf` file.

**Step 1:** Create a `/var/opt/ocient/bootstrap.conf` file on the node by replacing the `<FIRST_NODE_ADDRESS>` with the IP or Hostname of a node running the Administrator Role and the correct user name and password for your system.

```
adminHost: <FIRST_NODE_ADDRESS>
adminUserName: my_admin
adminPassword: example_password
```

**Step 2:** Start the new node.

```
sudo systemctl start rolehostd
```

This action accepts the node into the system and prepares it for role assignment.

### Related Links

[Cluster and Node Management](#)

[Cluster and Node Management](#)

[Cluster and Node Management](#)

[System Catalog](#)

### 7.5.7 Backup and Restore

Ocient® can back up and restore a single node or a multi-node system. The backup and restore procedures can replace a node in an existing Ociant cluster due to an OS drive failure or hardware upgrade.

Exact backup and restore procedures depend on the disk layout of your Ociant nodes. Consult Ociant Support or the system administrator that installed your Ociant cluster to determine the best backup strategy for your deployment.

### Related Links

[Maintenance Overview](#)

## 7.6 Monitoring

### 7.6.1 Log Monitoring

Contact Ociant® Support for assistance before altering the system logging configuration in the `rolehostd.conf` file.

The Ociant Hyperscale Data Warehouse™ logs events using these severities:

- Error
- Warn
- Info

- Verbose
- Debug
- Extended Debug

By default, the data warehouse creates the text log file `/var/opt/ocient/rolehostd.log` and logs events at the Info level and above.

Database logging supports extensive customization and integration with third-party tools. You can customize using the `rolehostd` config file. The database can return events in text, JSON, or Graylog® Extended Log Format (GELF) formats, or the database can return events in a file, to standard output (`stdout`), or to a UDP port. You can customize based on the log source.

## Log Formats

The default log format is text. These are some example log entries.

```
[2020-06-09T18:11:25.701594] [ INFO] [role001] roleHost: All roles started
[2020-06-09T18:11:25.701881] [ INFO] [role001] roleHost: Starting all endpoints
[2020-06-09T18:11:25.701925] [ INFO] [role001] roleHost: Starting endpoint 127.0.0.1:17900
[2020-06-09T18:11:25.704905] [ INFO] [role001] roleHost: Starting endpoint 127.0.0.1:17600
[2020-06-09T18:11:25.708437] [ INFO] [tcpU008] roleHost: All endpoints started
[2020-06-09T18:11:25.709236] [ INFO] [role001] host: -----
[2020-06-09T18:11:25.709286] [ INFO] [role001] host: Rolehostd successfully started
[2020-06-09T18:11:25.709305] [ INFO] [role001] host: -----
```

Each log line contains the time, severity, thread, and the source of the log (for example, `roleHost` or `host`).

JSON-formatted logs contain the same information but in a structured format.

```
{"sys":"oc1.admin0", "timestamp":1591726285.701594684, "thread":"role001", "lvl":" INFO", "src":"roleHost"}
{"sys":"oc1.admin0", "timestamp":1591726285.701881144, "thread":"role001", "lvl":" INFO", "src":"roleHost"}
{"sys":"oc1.admin0", "timestamp":1591726285.701925519, "thread":"role001", "lvl":" INFO", "src":"roleHost"}
{"sys":"oc1.admin0", "timestamp":1591726285.704905192, "thread":"role001", "lvl":" INFO", "src":"roleHost"}
{"sys":"oc1.admin0", "timestamp":1591726285.708437413, "thread":"tcpU008", "lvl":" INFO", "src":"roleHost"}
{"sys":"oc1.admin0", "timestamp":1591726285.709236006, "thread":"role001", "lvl":" INFO", "src":"host", "m"}
{"sys":"oc1.admin0", "timestamp":1591726285.709286683, "thread":"role001", "lvl":" INFO", "src":"host", "m"}
{"sys":"oc1.admin0", "timestamp":1591726285.709305246, "thread":"role001", "lvl":" INFO", "src":"host", "m"}
```

GELF-formatted log entries follow the format defined on the GELF via UDP page. This is an example GELF format.

```
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.701594684, "_thread":"role001", "level":"5"}
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.701881144, "_thread":"role001", "level":"5"}
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.701925519, "_thread":"role001", "level":"5"}
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.704905192, "_thread":"role001", "level":"5"}
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.708437413, "_thread":"tcpU008", "level":"5"}
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.709236006, "_thread":"role001", "level":"5"}
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.709286683, "_thread":"role001", "level":"5"}
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.709305246, "_thread":"role001", "level":"5"}
```

The database always returns message content in the `short_message` field of the GELF format.

This table maps Ocint severities to GELF system log (syslog) severities.

| Ocint Severity | Syslog Severity | Syslog Numeric Value |
|----------------|-----------------|----------------------|
| Error          | Error           | 3                    |
| Warn           | Warning         | 4                    |
| Info           | Notice          | 5                    |
| Verbose        | Info            | 6                    |

| Ocient Severity          | Syslog Severity | Syslog Numeric Value |
|--------------------------|-----------------|----------------------|
| Debug and Extended Debug | Debug           | 7                    |

## Log Appenders

A log appender is the destination for log entries. The database can send log entries to a file, a UDP port, or to `stdout`.

When logs are sent to the file appender, they grow continuously unless you use the `truncate` option in the configuration. You can use tools such as `logrotate` to manage log files.

This is a sample `logrotate` configuration.

```
/var/opt/ocient/log/rolehostd.log {
    rotate 5
    daily
    delaycompress
    compress
    copytruncate
    missingok
    minsize 1M
    maxsize 100M
}
```

Test the UDP appender using the Linux® `netcat` command.

```
$ netcat -ul 9000
```

## Log Configuration

An Ocient System supports various logging settings to meet different needs.

Set log configuration settings in `rolehostd.conf`, a YAML file normally found in the system `/var/opt/ocient/` directory. You can change configuration settings to suit your system logging needs in this file.

For information on checking the status of nodes, see [Statistics Monitoring](#).

If you are using an Ocient Simulator, logging behavior and directories might differ from these descriptions. For details on how logging works in the Ocient Simulator, see [Ocient Simulator](#).

### rolehostd Example

This is an abbreviated `rolehostd.conf` example in YAML format. Normally, this file contains other system configurations, but this tutorial focuses on the `loggingConfig` section, which defines logging behaviors and scopes.

```
loggingConfig:
  '': # Default scope (all logs)
    appender:
      logFilePath: /path/to/rolehostd.log
    appenders:
      - appender:
          allowLevel: warn
          logFilePath: /path/to/error.log
      - appender:
          allowLevel: warn
          format: json
          logFilePath: /path/to/error.json
```

```

dump: # Dump-specific logs
  allowLevel: debug
  appender:
    logFilePath: /path/to/dump.log

query: # Query-specific logs
  allowLevel: info
  appender:
    format: json
    logFilePath: /path/to/query.json

security: # Security-specific logs
  allowLevel: info
  appender:
    format: json
    logFilePath: /path/to/security.json

network-logs: # Example network logging
  allowLevel: info
  appender:
    type: UDP
    server: graylog.internal
    port: 12201
    format: gelf
    compressed: true

```

### Scope Definition

The top level of the logging configuration is the scope, which determines the corresponding log sources for each configuration setup.

### Default Scope

An empty scope string ('') applies its configuration settings to all log entries the system returns.

### Named Scopes

Named scopes apply to log entries where the first component matches the name. Useful scopes include:

- **query** — Contains information on all queries and statements executed on the system, including users, time, and other details.
- **security** — Contains information on all connections made to the system, successful or unsuccessful.
- **dump** — Contains information on system failures.

### allowLevel

The `allowLevel` parameter specifies which severity levels of logs to capture.

```
allowLevel: value
```

Severity `value` definitions are:

- **debug** — Most verbose (includes all logs).
- **info** — Includes informational messages and greater severities.
- **verbose** — More verbose than the `info` severity.
- **warn** — Includes warnings and errors only.
- **error** — Includes only error messages.

### appender

Appenders define the format and directory where the database sends log files. You can configure either one or many appenders for each scope.

## Single Appender

For a single appender, use the **appender** key.

```
appender:
  key: value
  [ ... ]
```

## Multiple Appenders

For multiple output destinations, use the **appenders** key, followed by a list of **appender** keys.

```
appenders:
- appender:
  key: value
  [ ... ]
- appender:
  key: value
  [ ... ]
[ -...]
```

### format

Defines the output format of the log files of an appender.

```
appender:
  format: format_type
```

Supported **format\_type** values are:

- **text** — Human-readable text format (default).
- **json** — Structured JSON format.
- **gelf** — GELF format for integration with Graylog.

### logFilePath

Defines the destination directory for log files of the appender. The **directory** value is a file path for the appender log files, for example: `/path/to/rolehostd.log`.

```
appender:
  logFilePath: directory
```

### type

Determines whether the system submits error logs through **stdout** or to a separate server using a TCP or UDP protocol.

```
appender: # standard output example
  type: stdout

appender: # UDP example
  type: UDP
  server: graylog-server.example.com
  port: 12201
  chunkSize: 1024 # Optional
  compressed: true # Optional

appender: # TCP example
  type: TCP
  server: 192.158.1.38
  port: 12201
```

### server

Required for TCP or UDP types.

The destination DNS name or IP address where the database sends log entries.

```
appender:
  server: 192.158.1.38
```

port

Required for TCP or UDP types.

The port where the database sends log entries. The default value is 9000.

```
appender:
  port: 12201
```

chunkSize

Optional for UDP type.

The largest packet size in bytes that the system sends to the server. The database discards log entries larger than this value.

```
appender:
  chunkSize: 1024
```

compressed

Optional for UDP type.

If you set this option to **true**, the system compresses packets. For details about log compression, see the GELF documentation. Otherwise, the packets are uncompressed.

```
appender:
  compressed: true
```

truncate

If you set this option to **true**, the system truncates the log file when the system starts. Otherwise, the system appends new logs to the file. The default value is **false**.

```
appender:
  truncate: true
```

## Related Links

[Install an Ocident System](#)

[Set Up System Monitoring with the TIG Stack and Kapacitor](#)

### 7.6.2 Statistics Monitoring

The Ocident Hyperscale Data Warehouse™ maintains many internal statistics. You can access the statistics on each node in the database using a REST interface that listens on port 9090 by default. For the endpoints, see [System Information REST Endpoints](#).

Explore the statistics using a command such as curl.

```
curl http://oc1-lts0:9090/v1/stats
```

The resulting statistics return in JSON format. This excerpt shows only a partial list of the results.

```
[{
  {
    "name": "localStorageService.device[SDM00001A616].spaceFree",
    "node": "oc1-lts0",
    "time": 1591737983812323,
    "units": "bytes",
    "value": 356599726080
  },
  {
    "name": "localStorageService.device[SDM00001A616].spaceFreePct",
    "node": "oc1-lts0",
    "time": 1591737983812323,
    "units": "none",
    "value": 4631469015491373042
  },
  {
    "name": "localStorageService.device[SDM00001A616].spaceTotal",
    "node": "oc1-lts0",
    "time": 1591737983812323,
    "units": "bytes",
    "value": 800149299200
  }
}]
```

Because each node can produce a very large number of statistics, the REST interface supports wildcard filtering. Return statistics that match the regular expression `^localStorageService*` using this curl command.

```
curl -G --data-urlencode 'filter=^localStorageService*' http://oc1-lts0:9090/v1/stats
```

The resulting subset of statistics returns in JSON format.

```
[
  {
    "name": "localStorageService.device.segmentTableEntries",
    "time": "1684931806764193",
    "timestamp": "2023-05-24T12:36:46.764Z",
    "node": "sql0",
    "value": 1,
    "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
  },
  {
    "name": "localStorageService.device.spaceFree",
    "time": "1684931806764193",
    "timestamp": "2023-05-24T12:36:46.764Z",
    "node": "sql0",
    "value": 0,
    "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
  },
  {
    "name": "localStorageService.device.spaceFreePct",
    "time": "1684931806764193",
    "timestamp": "2023-05-24T12:36:46.764Z",

```

```

    "node": "sql0",
    "value": 0.000000,
    "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
  },
  {
    "name": "localStorageService.device.spaceTotal",
    "time": "1684931806764193",
    "timestamp": "2023-05-24T12:36:46.764Z",
    "node": "sql0",
    "value": 2139074560,
    "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
  },
  {
    "name": "localStorageService.device.status",
    "time": "1684931806764193",
    "timestamp": "2023-05-24T12:36:46.764Z",
    "node": "sql0",
    "value": 10,
    "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
  },
  {
    "name": "localStorageService.device.segmentTableEntries",
    "time": "1684931806764193",
    "timestamp": "2023-05-24T12:36:46.764Z",
    "node": "sql0",
    "value": 1,
    "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
  },
  {
    "name": "localStorageService.device.spaceFree",
    "time": "1684931806764193",
    "timestamp": "2023-05-24T12:36:46.764Z",
    "node": "sql0",
    "value": 0,
    "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
  },
  {
    "name": "localStorageService.device.spaceFreePct",
    "time": "1684931806764193",
    "timestamp": "2023-05-24T12:36:46.764Z",
    "node": "sql0",
    "value": 0.000000,
    "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
  },
  {
    "name": "localStorageService.device.spaceTotal",
    "time": "1684931806764193",
    "timestamp": "2023-05-24T12:36:46.764Z",
    "node": "sql0",
    "value": 2139074560,
    "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
  },
  {
    "name": "localStorageService.device.status",
    "time": "1684931806764193",

```



```

    "timestamp": "2023-05-24T12:36:46.764Z",
    "node": "sql0",
    "value": 10,
    "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
  }
]

```

The specific set of statistics is subject to change.

## Retrieve Statistics Using Telegraf

Telegraf™ is a commonly used monitoring tool. This example shows the configuration file for Telegraf that gathers statistics from a database node where the statistic name includes `localStorageService`.

```

; Read formatted metrics from one or more HTTP endpoints
[[inputs.http]]
  interval = "30s"

  urls = [
    "http://oc1-lts0:9090/v1/stats?filter=localStorageService"
  ]

  ; HTTP method
  method = "GET"

  data_format = "json"
  json_name_key = "name"
  json_time_key = "time"
  json_time_format = "unix_us"
  json_timezone = "America/Chicago"
  tag_keys = [
    "node",
    "silo",
    "core"
  ]
  [inputs.http.tags]
    source = "rolehostd"

```

## Related Links

[Install an Ocident System](#)

### 7.6.3 System Information REST Endpoints

The Ocident Hyperscale Data Warehouse™ maintains many internal statistics. You can access the statistics on each node in the database using a REST interface that listens on port 9090 by default. For an overview of monitoring statistics, see [Statistics Monitoring](#).

#### 7.6.3.0.1 GET `http://oc1-lts0:9090/v1/stats`

**Description:** Retrieve the statistics on each node in the database.

#### Request Parameters

| Name                | Type                | Required | Description                |
|---------------------|---------------------|----------|----------------------------|
| <code>filter</code> | <code>string</code> | No       | Filter for the statistics. |

**Example Request**

```
curl --location --request GET 'http://oc1-lts0:9090/v1/stats' \
--header 'Accept: application/json'
```

**Example Response**

```
[{
  "name": "vm.stats.pdfCacheSize",
  "time": "1686335017884363",
  "timestamp": "2023-06-09T18:23:37.884Z",
  "node": "sql0",
  "value": 4680936
},
{
  "name": "localStorageService.device.spaceFree",
  "time": "1686335145388588",
  "timestamp": "2023-06-09T18:25:45.388Z",
  "node": "sql0",
  "value": 0,
  "device": "4418fbc3-fcf9-5cfe-b2b3-b41a8247a600"
}]
```

**7.6.3.0.2 GET http://oc1-lts0:9090/v1/version**

**Description:** Retrieve the version of the software running.

**Request Parameters**

*No parameters*

**Example Request**

```
curl --location --request GET 'http://oc1-lts0:9090/v1/version' \
--header 'Accept: application/json'
```

**Example Response**

```
{
  "version": "24.0.0",
  "timestamp": "20231110.212417",
  "git_commit": "178ba7a7549bd40c137035c5cd4f10c55fd31b00",
  "build_user": "",
  "build_type": "DEBUG",
  "build_flags": "fb,lld_flags,mcmodel_medium,tls_model_initial_exec"
}
```

**7.6.3.0.3 GET http://oc1-lts0:9090/v1/status**

**Description:** Retrieve the status of the software running.

**Request Parameters**

*No parameters*

### Example Request

```
curl --location --request GET 'http://oc1-lts0:9090/v1/status' \
--header 'Accept: application/json'
```

### Example Response

**Active**

#### 7.6.3.0.4 GET http://oc1-lts0:9090/v1/sysconfig

**Description:** Retrieve the configuration about this node in JSON format.

### Request Parameters

*No parameters*

### Example Request

```
curl --location --request GET 'http://oc1-lts0:9090/v1/sysconfig' \
--header 'Accept: application/json'
```

### Example Response

```
{
  "system_name": "abc7d45f-1e9c-4210-811d-ffg6dbd82297",
  "node_name": "sql0",
  "node_uuid": "578c11a2-1996-523f-a371-c393b4e5ff2e",
  "hostname": "username-dev0",
  "roles": [
    "administration",
    "health",
    "vm",
    "cmdComp"
  ],
  "failed_roles": [],
  "endpoints": [
    {
      "name": "adminopenendpoint",
      "address": "localhost",
      "port": 19000
    },
    {
      "name": "restendpoint",
      "address": "localhost",
      "port": 8000
    },
    {
      "name": "adminsecureendpoint",
      "address": "localhost",
      "port": 18500
    },
    {
      "name": "dataendpoint",
      "address": "localhost",
      "port": 16500
    }
  ],
}
```

```

    {
      "name": "sqlexternal",
      "address": "localhost",
      "port": 3070
    }
  ],
  "totalram": 450948103168,
  "processors": 120,
  "hugepages": [
    {
      "size": 1257152,
      "total": 0,
      "free": 0,
      "reserved": 0
    },
    {
      "size": 1060041824,
      "total": 0,
      "free": 0,
      "reserved": 0
    }
  ],
  "storage_devices": [
    {
      "id": "4425fad3-fcf9-5cfd-f3b3-b47a8457a300",
      "address": "/address/4425fad3-fcf9-5cfd-f3b3-b47a8457a300.dat",
      "type": "SATA",
      "use": "PAYLOAD",
      "pci_address": "",
      "mount_point": "",
      "device_path": ""
    },
    {
      "id": "9ae82aa7-ab73-5g86-bb34-13b27945c701",
      "address": "/address/9ae82aa7-ab73-5g86-bb34-13b27945c701.dat",
      "type": "SATA",
      "use": "PAYLOAD",
      "pci_address": "",
      "mount_point": "",
      "device_path": ""
    }
  ],
  "connections": [
    {
      "address": "111.1.1.1",
      "remote_port": 45598,
      "state": "connected",
      "locally_initiated": false
    }
  ]
}

```

#### 7.6.3.0.5 GET http://oc1-lts0:9090/v1/dbconfig

**Description:** Retrieve the configuration of the system as a whole in JSON format.

### Request Parameters

*No parameters*

### Example Request

```
curl --location --request GET 'http://oc1-lts0:9090/v1/dbconfig' \
--header 'Accept: application/json' \
--data-raw '{"filter":"string"}'
```

### Example Response

```
{
  "system_name": "abc7d45f-1e9c-4210-811d-ffg6dbd82297",
  "node_name": "sql0",
  "node_uuid": "578c11a2-1996-523f-a371-c393b4e5ff2e",
  "hostname": "username-dev0",
  "roles": [
    "administration",
    "health",
    "vm",
    "cmdComp"
  ],
  "failed_roles": [],
  "endpoints": [
    {
      "name": "adminopenendpoint",
      "type": "FULL",
      "address": "localhost",
      "port": 18000
    },
    {
      "name": "restendpoint",
      "type": "FULL",
      "address": "localhost",
      "port": 8000
    },
    {
      "name": "adminsecureendpoint",
      "type": "FULL",
      "address": "localhost",
      "port": 18500
    },
    {
      "name": "dataendpoint",
      "type": "FULL",
      "address": "localhost",
      "port": 16500
    },
    {
      "name": "sqlexternal",
      "type": "FULL",
      "address": "localhost",
      "port": 3070
    }
  ],
}
```

```

"totalram": 450948103168,
"processors": 120,
"hugepages": [
  {
    "size": 1257152,
    "total": 0,
    "free": 0,
    "reserved": 0
  },
  {
    "size": 1060041824,
    "total": 0,
    "free": 0,
    "reserved": 0
  }
],
"storage_devices": [
  {
    "id": "4425fad3-fcf9-5cfd-f3b3-b47a8457a300",
    "address": "/address/4425fad3-fcf9-5cfd-f3b3-b47a8457a300.dat",
    "type": "SATA",
    "use": "PAYLOAD",
    "pci_address": "",
    "mount_point": "",
    "device_path": ""
  },
  {
    "id": "9ae82aa7-ab73-5g86-bb34-13b27945c701",
    "address": "/address/9ae82aa7-ab73-5g86-bb34-13b27945c701.dat",
    "type": "SATA",
    "use": "PAYLOAD",
    "pci_address": "",
    "mount_point": "",
    "device_path": ""
  }
],
"connections": [
  {
    "address": "111.1.1.1",
    "remote_port": 45598,
    "state": "connected",
    "locally_initiated": false
  }
]
}

```

#### 7.6.3.0.6 GET http://oc1-lts0:9090/v1/config

**Description:** Retrieve values of configuration parameters.

#### Request Parameters

*No parameters*

#### Example Request

```
curl --location --request GET 'http://oc1-lts0:9090/v1/config' \
--header 'Accept: application/json'
```

### Example Response

```
{
  "siloType": "xg:runtime:physicalSiloSet_t",
  "tlbfsInfo.enabled": "false",
  "tlbfsInfo.mountPoint": "/mnt/name/hugepages1G",
  "tlbfsInfo.hugePageSize": "1073741824",
  "tlbfsInfo.hugePageCount": "512",
  "memory.warningHpMemoryUsageRatio": "0.8000000000000004",
  "memory.preferredHpMemoryUsageRatio": "0.9499999999999996",
  "memory.warningHeapMemoryUsageRatio": "0.9000000000000002",
  "silos.numSilos": "2",
  "silos.coreSets.computeCoreMask": "0b11111111111000",
  "silos.coreSets.ioCoreMask": "0b1000000111100000",
  "silos.coreSets.networkCoreMask": "0b10000000000111100",
  "silos.coreSets.generalCoreMask": "0b100000000000000111",
  "gdsClientParameters_t.batchTimer": "150000000n",
  "gdsClientParameters_t.maxBatchCount": "4096",
  "ports.adminsecure": "18500",
  "ports.data": "16500",
  "ports.hsi": "16900",
  "ports.ingress": "15300",
  "ports.rest": "8000",
  "ports.sql": "3070",
  "ports.adminopen": "18000",
  "operatorvm.vmProtocolParameters.vmClusterRaftFileParameters.fileName": "sql0-vmCluster.raft"
}
```

#### 7.6.4 Set Up System Monitoring with the TIG Stack and Kapacitor

One method to monitor the Ocient Hyperscale Data Warehouse™ is to use the TIG (Telegraf™, InfluxDB® and Grafana®) Stack. While these instructions are specifically written for this toolset, you can also use this guide as the foundation to set up an alternate stack if necessary.

This guide uses InfluxDB, Telegraf, Kapacitor™, and Grafana for a complete monitoring solution.

This guide specifically focuses on host-level and Ocient® software metrics. Monitoring additional components (e.g., Apache® Kafka®) is outside of the scope of this document.

These instructions were created using these versions of each component.

| Component | Version     |
|-----------|-------------|
| InfluxDB  | 1.8 (OSS)   |
| Telegraf  | 1.16.1      |
| Kapacitor | 1.5.6 (OSS) |
| Grafana   | 7.4.3       |

### Prerequisites

This guide assumes that:

1. SSH access and root-level privileges are available to each server running Ocient Software.

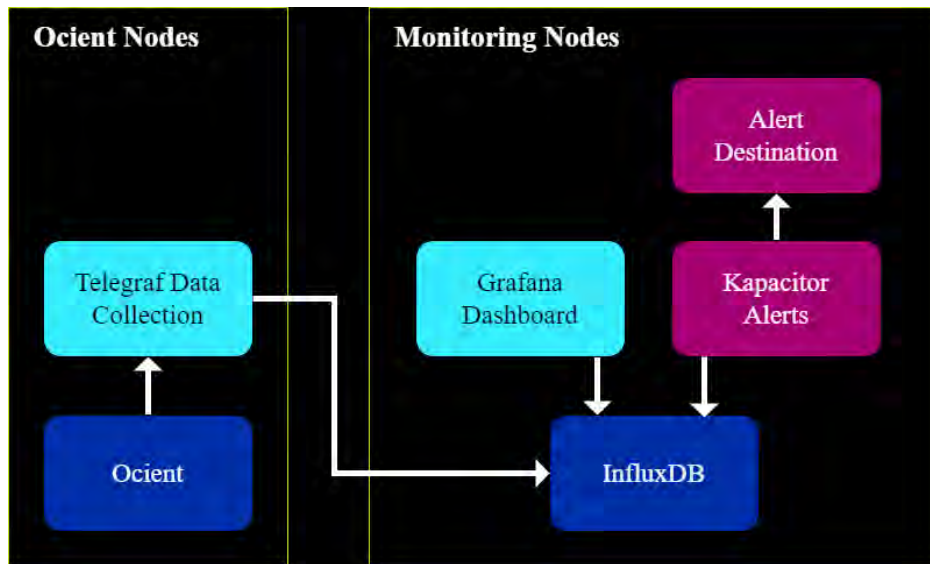


Figure 13: Monitoring example with Telegraf data collection into InfluxDB with Grafana dashboard and Kapacitor alerts

2. InfluxDB, Kapacitor, and Grafana are available to be installed in a virtual machine or container. It is important to note that each of these systems should be run independently. For example, do not run InfluxDB, Kapacitor, and Grafana on the same virtual machine.
3. All Ociant software is currently deployed and running as expected.
4. Logging configuration is set up as specified in [Log Monitoring](#).
5. The following ports must be open for InfluxDB, Kapacitor, Telegraf, and Grafana. InfluxDB is the most critical as each component communicates with it. Always refer to the latest documentation for each product as the definitive reference.

| Component | Default Ports                                   | Components Requiring Access  |
|-----------|-------------------------------------------------|------------------------------|
| InfluxDB  | 8086 8088 (optional, used for backup utilities) | Telegraf, Kapacitor, Grafana |
| Telegraf  | None                                            | None                         |
| Kapacitor | 9092                                            | None                         |
| Grafana   | 3000                                            | None                         |

### Step 1: Deploy InfluxDB

InfluxDB is a timeseries database that provides persistence for the host-level and Ociant software metrics.

1. Refer to current instructions from InfluxDB on how to install the service: <https://docs.influxdata.com/influxdb/v2.0/install/> <https://docs.influxdata.com/influxdb/v2.0/get-started/>
2. Document the IP address of the machine where InfluxDB is installed.
3. Ensure that InfluxDB is running properly.

```
> sudo systemctl status influxdb
influxdb.service - InfluxDB is an open-source, distributed, time series database
Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset: enabled)
Active: active (running) since Tue 2021-09-14 17:17:20 UTC; 3 weeks 2 days ago
Docs:
Main PID: 32687 (influxd)
Tasks: 18 (limit: 4915)
```



```
CGroup: /system.slice/influxdb.service
32687 /usr/bin/influxd -config /etc/influxdb/influxdb.conf/
```

4. Set the InfluxDB service to start on boot.

```
/> sudo systemctl enable influxdb
```

5: It is highly recommended that critical components of monitoring infrastructure, including InfluxDB, are also monitored. Monitoring the monitoring stack is outside of the scope of this document. Refer to available resources on potential solutions to monitor InfluxDB.

## Step 2: Deploy and Configure Telegraf

Telegraf is used for metrics collection. It runs on each of the Admin, Loader, Foundation, SQL Nodes. The following instructions highlight the common configuration elements, followed by the elements that are specific to different node types.

All Nodes

1. Install Telegraf on each of the Admin, Loader, Foundation, and SQL Nodes. Refer to the Telegraf installation instructions available from InfluxDB. At this point, do not start the Telegraf service or generate the default configuration.
2. For **all nodes**, create the Telegraf configuration file (typically located at `/etc/telegraf/telegraf.conf`). Open the file in an editor.

- Copy and paste the following contents into the file.
- Replace the designated placeholders (denoted with a `$`) with the specific values for the specified environment:
  - `CLUSTER_NAME` = — Customer-selected name of a provided cluster for monitoring identification
  - `ROLE` = — One of `admin`, `loader`, `lts`, `sql`, or `stream_proc`, depending on the node type
  - `INFLUX_URL` = — URL of the InfluxDB instance (e.g., `http://10.6.0.4:8086`)

```
; /etc/telegraf/telegraf.conf
[global_tags]
  cluster = "$CLUSTER_NAME"
  cluster_role = "$ROLE" # one of admin, loader (aka indexer), lts (aka foundation), sql, stream_proc

[agent]
  interval = "15s" # time
  metric_batch_size = 1000 # size
  metric_buffer_limit = 100000 # size
  collection_jitter = "3s" # time
  flush_interval = "7s" # time
  flush_jitter = "3s" # time
  round_interval = false

[outputs]

[[outputs.influxdb]]
  urls = [ "$INFLUX_URL" ]
  database = "telegraf"
  precision = "1s"
  timeout = "10s"
```

Save the file.

3. For **all nodes**, create a new file under the `telegraf.d` directory (typically `/etc/telegraf/telegraf.d`) named `host.conf`.

- Copy and paste the following contents into the file.

- Replace the designated placeholders (denoted with an `$`) with the specific values for the given environment:
  - `BOOT_DISK` — Device name of the boot disk (e.g., `sda`). You can determine the boot disk by running an `lsblk` command.
  - `NET_INT_BONDO_WILDCARD` — The wildcard for matching the underlying network interfaces within `bond0` (e.g., `eno*`). If it is unclear, refer to `/proc/net/bonding/bond0` and reference the interfaces noted in the Telegraf documentation.
  - `NET_INT_BOND1_WILDCARD` — The wildcard for matching the underlying network interfaces within `bond1` (e.g., `enp*`). If it is unclear, refer to `/proc/net/bonding/bond1` and reference the interfaces noted in the Telegraf documentation.

```
; /etc/telegraf/telegraf.d/host.conf
[[inputs.cpu]]
    percpu = true
    totalcpu = true
    fielddrop = [ "time_*" ]
[[inputs.disk]]
    ignore_fs = ["tmpfs", "devtmpfs", "vfat"]
[[inputs.diskio]]
    skip_serial_number = true
    devices = ["$BOOT_DISK"]
    [inputs.diskio.tags]
        disk_type="os"

[[inputs.mem]]

[[inputs.net]]
    interfaces = ["$NET_INT_BONDO_WILDCARD", "$NET_INT_BOND1_WILDCARD"]

[[inputs.net]]
    interfaces = ["bond0"]
    [inputs.net.tags]
        primary_interface_10Gb = "true"
    [inputs.net.tagdrop]
        interface = [ "all" ]

[[inputs.net]]
    interfaces = ["bond1"]
    [inputs.net.tags]
        primary_interface_100Gb = "true"
    [inputs.net.tagdrop]
        interface = [ "all" ]

[[inputs.system]]
[[inputs.swap]]

[[inputs.linux_sysctl_fs]]
```

Save the file.

4. For **all nodes**, create a new file under the telegraf scripts directory (typically under `/etc/telegraf/scripts`) named `uio_pci_generic.sh`.

- Copy and paste the following contents into the file.

```
; /etc/telegraf/scripts/uio_pci_generic.sh
; A script used to enumerate the PCI devices on the system.
;
```

```
; !/bin/bash

DEVICES=$(find /sys/bus/pci/drivers/uio_pci_generic/0000\:* | awk -F '/' '{ print $NF }')

for d in $DEVICES; do
    echo pci_devices,device=\"$d\" present=1
done
```

### Foundation and SQL Nodes

Create a new file under the `telegraf.d` directory (typically `/etc/telegraf/telegraf.d`) named `rolehostd.conf`.

- Copy and paste the following contents into the file.
- Replace the designated placeholders (denoted with an `$`) with the specific values for the specified environment:
  - `NODE_BONDO_IP_ADDRESS` = IP address associated with `bond0`

```
[inputs.http]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/status"
  ]
  method = "GET"
  timeout = "5s"
  name_override = "rolehostd_status"
  data_format = "value"
  data_type = "string"

[[inputs.http_response]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/status"
  ]
  method = "GET"
  response_timeout = "5s"
  name_override = "rolehostd_status_response"
  response_string_match = "Active"

[[inputs.http]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/stats"
  ]
  method = "GET"
  timeout = "5s"
  name_override = "rolehostd"
  json_time_key = "time"
  json_time_format = "unix_us"
  json_string_fields = ["units"]
  data_format = "json"
  tagexclude = ["url", "node"]
  tag_keys = [
    "name",
    "socket",
    "device"
  ]

[[inputs.http]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/version"
  ]
```

```

method = "GET"
timeout = "5s"
name_override = "rolehostd_version"
json_query = "{build_type,version}"
json_string_fields = ["build_type","version"]
data_format = "json"

[[inputs.http]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/operatorssummary"
  ]
  method = "GET"
  timeout = "5s"
  name_override = "rolehostd_operatorssummary"
  data_format = "json"

[[inputs.exec]]
  commands = ['sh /etc/telegraf/scripts/uio_pci_generic.sh']
  data_format = "influx"

[[inputs.tail]]
  name_override = "tail_query_json"
  files = ["/var/opt/ocient/log/query.json", "/var/opt/ocient/query.json"]
  data_format = "json"
  json_query = "{src,msg.user,msg.database,msg.service_class,msg.client_version,msg.client_ip,msg.timestamp}"
  tag_keys = ["src","user","database","service_class","code","major_driver_version","resultset_cached"]
  json_string_fields = ["client_version","client_ip","timestamp_start","timestamp_execstart","timestamp_

[[processors.converter]]
  namepass=["tail_query_json"]
  [processors.converter.fields]
    integer = ["timestamp_start","timestamp_execstart","timestamp_complete"]

```

You can also use Filebeat to forward the `query.json` file to a monitoring platform of your choice.

#### Admin and Loader Nodes

Create a new file under the `telegraf.d` directory (typically `/etc/telegraf/telegraf.d`) named `rolehostd.conf`.

- Copy and paste the following contents into the file.
- Replace the designated placeholders (denoted with an `$`) with the specific values for the given environment:
  - `NODE_BONDO_IP_ADDRESS` = IP address associated with bond0

```

; /etc/telegraf/telegraf.d/rolehostd.conf
[[inputs.http]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/status"
  ]
  method = "GET"
  timeout = "5s"
  name_override = "rolehostd_status"
  data_format = "value"
  data_type = "string"

[[inputs.http_response]]
  urls = [

```

```

    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/status"
  ]
  method = "GET"
  response_timeout = "5s"
  name_override = "rolehostd_status_response"
  response_string_match = "Active"

[[inputs.http]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/stats"
  ]
  method = "GET"
  timeout = "5s"
  name_override = "rolehostd"
  json_time_key = "time"
  json_time_format = "unix_us"
  json_string_fields = ["units"]
  data_format = "json"
  tagexclude = ["url", "node"]
  tag_keys = [
    "name",
    "socket",
    "device"
  ]

[[inputs.http]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/version"
  ]
  method = "GET"
  timeout = "5s"
  name_override = "rolehostd_version"
  json_query = "{build_type,version}"
  json_string_fields = ["build_type","version"]
  data_format = "json"

[[inputs.exec]]  commands = ['sh /etc/telegraf/scripts/uio_pci_generic.sh']
  data_format = "influx"

[[inputs.tail]]
  name_override = "tail_query_json"
  files = ["/var/opt/ocient/log/query.json","/var/opt/ocient/query.json"]
  data_format = "json"
  json_query = "{src,msg.user,msg.database,msg.service_class,msg.client_version,msg.client_ip,msg.timestamp}"
  tag_keys = ["src","user","database","service_class","code","major_driver_version","resultset_cached"]
  json_string_fields = ["client_version","client_ip","timestamp_start","timestamp_execstart","timestamp_

[[processors.converter]]
  namepass=["tail_query_json"]
  [processors.converter.fields]
    integer = ["timestamp_start","timestamp_execstart","timestamp_complete"]

```

Loader Nodes

Loader Nodes also run the Loading and Transformation service and require additional Telegraf configuration to capture those metrics.

- Create a new file under the `telegraf.d` directory (typically `/etc/telegraf/telegraf.d`) named `lat.conf`.
- Copy and paste the following contents into the file.

```
; /etc/telegraf/telegraf.d/lat.conf
[[inputs.jolokia2_agent]]
  urls = ["http://localhost:8080/v2/metrics"]

[[inputs.jolokia2_agent.metric]]
  name = "pipeline"
  mbean = "lat:type=pipeline"

[[inputs.jolokia2_agent.metric]]
  name = "partitions"
  mbean = "lat:type=partitions"

[[inputs.procstat]]
  systemd_unit = "lat.service"
```

### Start Telegraf

When the configuration is properly entered on all nodes, Telegraf must be started.

1. On **each node**, start the Telegraf service.

```
/> sudo systemctl start telegraf
```

2. Ensure the service came up properly by checking the status.

```
/> sudo systemctl status telegraf
telegraf.service - The plugin-driven server agent for reporting metrics into InfluxDB
Loaded: loaded (/usr/lib/systemd/system/telegraf.service; enabled; vendor preset: disabled)
Active: active (running) since Thu 2021-08-05 22:04:06 UTC; 2 months 3 days ago
Docs: https://github.com/influxdata/telegraf
Main PID: 2651775 (telegraf)
Tasks: 103 (limit: 1589476)
Memory: 221.2M
CGroup: /system.slice/telegraf.service
2651775 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -config-directory /etc/telegraf/telegraf.d
```

3. Set the Telegraf service to start on boot.

```
/> sudo systemctl enable telegraf
```

### Step 3: Deploy and Configure Kapacitor

Kapacitor is used to process time series data, detect anomalies, and trigger alerts. As the preferred alerting mechanisms for every customer will vary, refer to the Kapacitor documentation on [how to send alerts through specific tools](#).

1. Install Kapacitor using the instructions from InfluxDB. <https://docs.influxdata.com/kapacitor/v1.6/introduction/installation/>
2. Create the Kapacitor configuration file (typically located at `/etc/kapacitor/kapacitor.conf`). Open the file in an editor.
  - Copy and paste the following contents into the file.
  - Replace the designated placeholders (denoted with an `$`) with the specific values for the given environment:
    - `KAPACITOR_HOST_OR_IP` = — Hostname or IP address associated with the Kapacitor instance

– INFLUX\_URL = — URL of the InfluxDB instance (e.g., <http://10.6.0.4:8086>)

```
; /etc/kapacitor/kapacitor.conf
hostname = "$KAPACITOR_HOST_OR_IP"
data_dir = "/var/lib/kapacitor"
skip-config-overrides = false

[logging]
# Destination for logs
# Can be a path to a file or 'STDOUT', 'STDERR'.
file = "/var/log/kapacitor/kapacitor.log"
level = "ERROR"

[load]
enabled = true
dir = "/etc/kapacitor/load"

[replay]
# Where to store replay files, aka recordings.
dir = "/var/lib/kapacitor/replay"

[storage]
boltdb = "/var/lib/kapacitor/kapacitor.db"

[[influxdb]]
enabled = true
default = true
name = "influx"
urls = ["$INFLUX_URL"]
username = "admin"
password = ""
timeout = "10s"

startup-timeout = "60s"

# Subscription mode is either "cluster" or "server"
subscription-mode = "server"

# Which protocol to use for subscriptions
# one of 'udp', 'http', or 'https'.
subscription-protocol = "http"

# Subscriptions resync time interval
# Useful if you want to subscribe to new created databases
# without restart Kapacitord
subscriptions-sync-interval = "1m0s"

; INSERT ALERT DESTINATION CONFIGS HERE

[stats]
enabled = true
stats-interval = "10s"
database = "_kapacitor"
retention-policy= "autogen"
```

Add Kapacitor Alerts and Templates





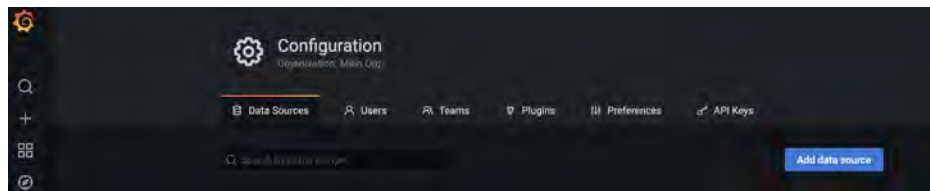


Figure 14: Configuration of data sources



Figure 15: Add data source

- Ensure the data source is marked as the Default.
  - Under the HTTP section, the URL should be the same as the INFLUX\_URL used in previous sections.
  - The query language should be set to the default (InfluxQL).
6. Under the InfluxDB Details section, specify *telegraf* as the database name.
  7. When the configuration appears to be correct, click **Save and Test**.
  8. A green banner should appear that indicates the connection to the data source is working properly. If this message does not display, consult the Grafana documentation or contact Ocient Support.

### Import Dashboards

Ocient provides sample Grafana dashboards upon request. Contact Ocient Support for details.

Within the bundle, relevant contents are located within the **grafana** directory.

Grafana only allows one file at a time.

1. Open Grafana in a web browser.
2. In the Grafana UI, click the **+** (“Create”) tab on the left side of the screen. Select the **Import** option.
3. On the Import page, click the **Upload JSON File** button.
4. Select the file representing the dashboard you wish to import.
5. On the dashboard summary page, verify the settings are correct. Click **Import** to complete the process.
6. Repeat steps 3-6 for each dashboard that needs to be added.

### Related Links

[Log Monitoring](#)

[Statistics Monitoring](#)

Figure 16: Configuration settings

Figure 17: InfluxDB details for database access

Figure 18: Connection status

*Filebeat is a trademark of Elasticsearch BV.*

### 7.6.5 Monitor Segment Group Transfers

You can perform segment group transfers in the Ocint® System. Use the data in the `sys.segment_group_transfers` system catalog table to monitor this activity.

#### Segments, Segment Groups, and Segment Group Transfers

A segment is the primary storage unit. Individual segments combine together to form segment groups. For details, see [Core Elements of an Ocint System](#).

In a system that has multiple storage clusters, the Ocint System might experience a data imbalance between clusters. In this case, you might need to transfer segment group ownership between clusters. Transferring segment groups enables you to transfer the physically stored segments between clusters. During or after the segment group transfer, you can monitor the activity using the `sys.segment_group_transfers` system catalog table.

#### Segment Group Transfers System Catalog Table

The database contains the system catalog table named `sys.segment_group_transfers` with these columns:

- `id` — A UUID that represents the transfer identifier.
- `src_cluster_id` — The UUID of the source cluster.
- `dst_cluster_id` — The UUID of the destination cluster.
- `status` — The current status of the transfer.
  - `ACTIVE` — The transfer is ongoing.
  - `COMPLETE` — The transfer is complete and the segment groups are owned by the destination cluster.
  - `FAILED` — The transfer is not complete and the segment groups are owned by the source cluster. A transfer can be in this state due to an error that prevented the system from transferring segment groups or due to a canceled transfer.
- `segment_group_ids` — An array of uint64 segment group identifiers.
- `src_committed_osn` — The ownership sequence number (OSN) where the source cluster no longer returns the segment groups in the transfer for queries.
- `dst_committed_osn` — The OSN where the destination cluster added the segment groups. The destination cluster does not return the segment groups for queries unless the query uses at least the `src_committed_osn` value for the OSN of the source cluster.

The database removes segment group transfers from the system upon completion.

#### Related Links

[System Catalog](#)

## 7.7 System Configuration

### 7.7.1 Inspect the Current Configuration

Ocint® includes a variety of configuration settings that control how the system behaves. The vast majority of these won't change in a typical deployment. However, there is a small set more commonly changed to adjust for different workloads, hardware, and operating requirements.

Configuration is primarily done through DDL or DCL and can be inspected using the [System Catalog Reference](#).

You can contact Ocint Support to change configuration settings that apply across the entire system or to a subset of nodes. Each Ocint node reads configuration settings during startup. Therefore, configuration changes require restarts for the new values to take effect. For details, contact Ocint Support.

## Inspecting System Configuration

Administrators typically need to know the effective configuration of the system and what has been overridden.

- Configuration Overrides — a list of the overrides applied by administrators
- Effective Configuration — the complete list of all configuration parameters and their current values on each node

### Viewing Configuration Overrides

A system catalog table is provided that lists all DDL Configuration Overrides that are applied to the Ocient System. This table includes the configuration target type as well as the scope to which it is applied.

In the event that the scope is not `SYSTEM` (See Advanced Configuration for `CLUSTER` and `NODE` scopes), the `scope_id` will indicate the `cluster_id` or `node_id` to which the configuration override applies.

### Example

The following query retrieves all config overrides applied on the system and provides a friendly node name or cluster name where an override is limited in scope:

```
SELECT
  cfg.scope_type,
  n.name AS node_name,
  c.name AS cluster_name,
  cfg.key,
  cfg.value
FROM
  sys.config cfg
  LEFT JOIN sys.nodes n
    ON n.id = cfg.scope_id and cfg.scope_type = 'NODE'
  LEFT JOIN sys.clusters c
    ON c.id = cfg.scope_id and cfg.scope_type = 'CLUSTER';
```

### Output

| scope_type | node_name  | cluster_name    |
|------------|------------|-----------------|
| NODE       | sales-lts8 | NULL            |
| SYSTEM     | NULL       | NULL            |
| CLUSTER    | NULL       | storageCluster1 |
| NODE       | sales-lts8 | NULL            |
| SYSTEM     | NULL       | NULL            |
| ...        |            |                 |

### Viewing Effective Configuration Using System Table

A system catalog table is provided that lists the effective configuration of each node.

### Example

The following query retrieves the effective overrides for the SQL role on the specified node:

```
SELECT
  n.name,
  nc.key,
  nc.value
FROM
  sys.node_config nc
  JOIN sys.nodes n
    ON nc.node_id = n.id
  WHERE n.name = 'sql0'
  AND key LIKE 'sql%';
```

*Output*

| name | key                                             | value            |
|------|-------------------------------------------------|------------------|
| sql0 | sql.blobstoreParameters.regionAllocationUnit    | 17179869184      |
| sql0 | sql.blobstoreParameters.pageUtilizationExpand   | 0.75             |
| sql0 | sql.blobstoreParameters.pageUtilizationContract | 0.25             |
| sql0 | sql.blobstoreParameters.maxFileSize             | 0                |
| sql0 | sql.blobstoreParameters.minFileSize             | 128MiB           |
| sql0 | sql.blobstoreParameters.blobPageSize            | 128KiB           |
| sql0 | sql.blobstoreParameters.stripeWidth             | 0                |
| sql0 | sql.blobstoreParameters.coreMask                | 0b10000000       |
| sql0 | sql.cacheMinFreeMemoryPercent                   | 0.109999999      |
| sql0 | sql.cacheMinSize                                | 4096             |
| sql0 | sql.cacheMinKeepTime                            | 720              |
| sql0 | sql.purgeCacheFrequency                         | 300              |
| sql0 | sql.pdfReservedMem                              | 0                |
| sql0 | sql.numLevels                                   | 3                |
| sql0 | sql.maxShutdownWaitTime                         | 1800000          |
| sql0 | sql.shortQueryMillis                            | -1               |
| sql0 | sql.timeLimit                                   | 1800000          |
| sql0 | sql.forceNetworkedVirtualTables                 | false            |
| sql0 | sql.cacheLimit                                  | 17179869184      |
| sql0 | sql.optimizer                                   | TKTOptimizer     |
| sql0 | sql.generator                                   | TKTPlanGenerator |
| sql0 | sql.validator                                   | TKTValidator     |
| sql0 | sql.concurrencyTarget                           | 5                |

## Viewing Effective Configuration Using REST Endpoints

Administrators can also view the effective system configuration using the config endpoint, a REST interface available on each node. This endpoint shows every configuration parameter for the node and its current value. It reflects the active values and might not reflect configuration overrides applied since the last node restart.

The config endpoint is found on each node at `http://<node_hostname_or_ip_address>:9090/v1/config` and it returns a JSON representation of the active system configuration of the targeted node.

**Example:**

```
curl http://10.0.1.7:9090/v1/config
```

**Example Response**

```
{
  "siloType": "xg::runtime::physicalSiloSet_t",
  "tlbfsInfo.enabled": "true",
  "gdsClientParameters_t.batchTimer": "150000000n",
  "gdsClientParameters_t.maxBatchCount": "4096"
  ...
  "operatorvm.vmsiloMask": "0x3",
  "operatorvm.vmParameters.levels": "0x4",
  "operatorvm.vmParameters.statsCacheSize": "4GiB",
  "operatorvm.vmProtocolParameters.routerPollRate": "250000000n",
  "operatorvm.vmProtocolParameters.bufferedDatablockTimeout": "12000000000n"
  ...
  "lts.ioOpParameters.bufferPool.backingSlabSize": "2MiB",
  "lts.ioOpParameters.bufferPool.bufferPools[0].pool.size": "4KiB",
```

```

    "lts.ioOpParameters.bufferPool.bufferPools[0].pool.count": "98304",
    "lts.ioOpParameters.lbaCacheSize": "64",
    "lts.targetFillRatio": "1",
    "lts.pdfReservedMem": "4GiB"
    ...
}

```

## Related Links

System Catalog

### 7.7.2 Node Configuration with the Administrator Role

Nodes with the **admin** role have the additional responsibility to store important metadata for compression and system configuration. In most cases, an Ocient Hyperscale Data Warehouse™ system should have the majority of SQL Nodes assigned to the **admin** role.

#### Administrator Node Redundancy

To ensure high availability and consistency, an Ocient® System must always have a minimum number of operational nodes with **admin** privileges.

Redundancy for the system configuration and dictionary compression operates using a majority present model. In a system with  $N$  nodes configured with the **admin** role, at least  $N / 2 + 1$  must be present and active for the system to function correctly, where  $N$  is the total number of nodes, and  $N / 2$  denotes the division floor (i.e., rounded down to the nearest integer).

This table shows the number of nodes with **admin** privileges that can fail before the system experiences problems. Both columns comprise only nodes assigned the **admin** role, not the entire system node count.

| Number of Nodes | Node Failures Tolerated |
|-----------------|-------------------------|
| 1               | 0                       |
| 2               | 0                       |
| 3               | 1                       |
| 4               | 1                       |
| 5               | 2                       |
| 6               | 2                       |
| 7               | 3                       |

Various core system functions risk failure if the number of nodes running the **admin** role falls below the tolerance threshold. Operations that could fail include:

- System startup
- System configuration
- Query execution
- Data loading

The majority present model for nodes with the **admin** role is different than parity encoding used for Foundation Nodes.

For more information on parity width for Foundation Nodes, see [Configure Storage Spaces](#).

## Administrator Node Metadata

This table describes additional metadata files stored by nodes running the `admin` role. If you remove the `admin` role from a node, the system removes these files.

| File path                                           | Description                                                                         |
|-----------------------------------------------------|-------------------------------------------------------------------------------------|
| <code>/var/opt/ocient/metadataStorage.raft</code>   | Definitions for users, network configuration, and database objects.                 |
| <code>/var/opt/ocient/globalDataStorage.raft</code> | Compression dictionaries for the <code>COMPRESSION</code> <code>GDC</code> columns. |

## Configure Nodes with the Administrator Role

To configure a node to run the `admin` role, execute this SQL statement.

```
ALTER NODE nodename ADD ROLE admin;
```

It is generally a good practice to have all SQL Nodes in your system assigned to run the `admin` role. You can configure additional nodes with the `admin` role to adjust the level of redundancy to your chosen level.

## Related Links

[Cluster and Node Management](#)

[Configure Storage Spaces](#)

[Expand and Rebalance System](#)

### 7.7.3 Manage the Network Configuration of an Ocient System

Control the grouping of SQL Nodes in an Ocient® System by managing network traffic using connectivity pools. For advanced configuration, you can expand a connectivity pool by the number of nodes or you can change the network configuration of the system.

## SQL Nodes Belong to Connectivity Pools

A connectivity pool is a pool of potential SQL Nodes for client connection. The pool contains all the required information to connect to a node in that pool. For a description of the metadata of a connectivity pool, see the [Cluster and Node Management](#) statement.

All SQL Nodes in the system must belong to a connectivity pool. If you do not configure the connectivity pools during bootstrapping, the Ocient System assigns each SQL Node to its own connectivity pool named with the node name concatenated by `default`. For example, for the node named `sql0`, the system creates a connectivity pool named `sql0_default`. For bootstrapping the Ocient System, see [Ocient System Bootstrapping](#).

## Connectivity Pool Considerations

For different networks, you can connect a SQL Node to multiple networks at the same time. Connectivity pools enable each SQL Node to specify the IP address and port number for listening in each network.

Connectivity pools assume that the IP address for each SQL Node remains static. They do not support dynamic IP addresses. If the IP address of your SQL Node changes, you must manually update it by using an [Cluster and Node Management](#) SQL statement.

You can direct network traffic to groups of SQL Nodes by creating connectivity pools with numerous nodes. Also, you can use connectivity pools to isolate SQL Nodes, which allows you to connect to a subset of nodes and redirect network traffic between them. For example, you can configure the network so that the Ocient System sends only certain SQL queries to certain SQL Nodes.

You connect directly to SQL Nodes. You must know the correct connection information to the SQL Node related to the connectivity pool where you want to create the connection. SQL Nodes only listen to IP addresses and port numbers for the connectivity pools associated with the SQL Nodes. Each node can have multiple connectivity pools assigned to it. Assign different IP addresses to SQL Nodes based on the network (e.g., VPN, internal network). You can configure the configuration to reflect your network whether the network is on-site, a virtual public cloud, or a remote network.

Connectivity pools can use the same IP addresses and port numbers across pools. A pool can also use the IP address of the source client connection to distribute the SQL queries in the case where the SQL Node connection information matches multiple connectivity pools.

When you connect to a SQL Node using a specific IP address, the node finds all connectivity pools associated with that IP address. The SQL Node then identifies all the other nodes in the connectivity pool and uses that set of nodes as a redirect group to distribute the SQL query. When a SQL Node matches multiple pools by the IP address and port number combination, the node uses the source connection information (or the information you used to connect) to find a more relevant connectivity pool. The SQL Node uses the chosen connectivity pool to redirect SQL queries to the group of nodes.

### Initial Configuration Setup

For the initial configuration, a common Ocint System configuration has all SQL Nodes in one connectivity pool. However, you can structure the network of nodes based on your initial needs. You can change your initial configuration setup later as your needs change.

Create your initial configuration of the network using the see the [Cluster and Node Management](#) statement or specify the configuration in the `bootstrap.conf` file. For details about the `bootstrap.conf` file, see [Node Bootstrapping Reference](#).

If you do not specify the configuration of the SQL Nodes during the initial Ocint System setup, the system assigns nodes to default connectivity pools. In this case, each node always has a connectivity pool that isolates the node. A common Ocint System configuration has all nodes in one connectivity pool. Thus, you can create a new connectivity pool using the `CREATE CONNECTIVITY_POOL` statement to add all nodes in the network to it. Then, remove all the default pools using the [Cluster and Node Management SQL](#) statement.

For advanced configuration, you can expand the connectivity or change the network configuration of the system.

### Expand the Network of an Ocint System

You can expand the connectivity pool of an Ocint System by adding SQL Nodes. Then, remove the automatically generated connectivity pool that is no longer needed.

For example, with an Ocint System that has SQL Nodes `sql0`, `sql1`, and `sql3` in the connectivity pool named `test_pool`, add a new SQL Node `sql4` to the pool that belongs to the default pool named `sql0_default`. The nodes have the IP address `111.1.1.1` and port number `4050` for listening. Specify the local IP address and port number `4050` to return to the client.

```
ALTER CONNECTIVITY_POOL test_pool
  ADD PARTICIPANTS(
    (NODE sql4
      LISTEN_ADDRESS '111.1.1.1'
      LISTEN_PORT 4050
      ADVERTISED_ADDRESS 'localhost'
      ADVERTISED_PORT 4050));
```

Remove the default connectivity pool `sql0_default`.

```
DROP CONNECTIVITY_POOL IF EXISTS sql0_default;
```

### Change the Network Configuration of an Ocint System



You can configure IP addresses that the Ocient System uses to connect to the external client or to other nodes in the system. Configure IP addresses for external client connections using the [Cluster and Node Management SQL](#) statement. Configure IP addresses for connections with other nodes in the system using the [Cluster and Node Management SQL](#) statement.

If you add a new SQL Node to an existing Ocient System, configure a new system, or add a SQL role using the [Cluster and Node Management SQL](#) statement, you must configure the connectivity pools using the [Cluster and Node Management SQL](#) statements. If you add a SQL Node to any connectivity pool, then you must remove the default connectivity pool where the node previously belonged using the [Cluster and Node Management SQL](#) statement.

### Connectivity Pool Troubleshooting

When you experience issues with connection to the configured connectivity pools, you have two ways to manipulate the network configuration temporarily to address the issues.

You can use the `rolehostd.conf` file. When you cannot establish the connection using the connectivity pool, you can use this configuration in the `rolehostd.conf` file to correct the configuration of the SQL Nodes. This configuration specifies that nodes listen using the IP address `0.0.0.0` and port number `4050` to enable recovery.

```
sql:
  defaultListen: true
```

You must restart the `rolehostd` process after you add this configuration.

```
sudo systemctl restart rolehostd
```

Or, you can change the IP address of the SQL Nodes in the connectivity pool.

For existing clusters, the Ocient System configures all SQL Nodes to listen to the IP address `0.0.0.0`, which allows client connections to manually create the initial connectivity pools.

### Related Links

[Install an Ocient System](#)

[Ocient System Bootstrapping](#)

[Node Bootstrapping Reference](#)

[Cluster and Node Management](#)

[Cluster and Node Management](#)

[Cluster and Node Management](#)

[Cluster and Node Management](#)

## 7.8 Secure Connections Using TLS

The SQL service endpoint can make use of TLS.

The SQL service always has TLS functionality enabled and active.

The database allows connections either unencrypted, or using TLS/SSL. The certificates used to establish SSL connections are located in the directory specified by `certificateStore/path` in the YAML the `rolehostd.conf` file, which is `/var/opt/ocient` by default. The certificate files need to exist on the SQL Nodes for connecting using JDBC.

```
/var/opt/ocient/server.crt
/var/opt/ocient/server.key
/var/opt/ocient/server-ca.crt
```

The server first looks for the certificate or key files named `server.crt` and `server.key` in that directory. Those files should be stored in PEM format. If those files do not exist, the database uses the same `ocient.crt` and `ocient.key` that are used for the internal cluster authentication. The `server.crt` and `server.key` files should be used to provide user-defined certificates to the database.

When these files are in place, restart the services and they will automatically make use of TLS and the specified certificate.

By default, the database allows both encrypted and unencrypted database (JDBC) connections, even if the specified `server.crt` and `server.key` exist.

## Related Links

[Connect to Ocient](#)

## 7.9 Multiple Storage Clusters for Loading Data

You can load data in the Ocient Hyperscale Data Warehouse™ that contains multiple storage clusters. Take these considerations into account for loading in a multi-cluster environment and the related troubleshooting.

### Multi-Cluster Environment Considerations

- Loader Nodes send data to all Foundations Nodes in the Ocient® System.
- The distribution of data loaded to each cluster is proportional to its total capacity.
- Every Loader Node loads to every cluster that belongs to the storage space of the target tables. Depending on the location of the tables, the database or schema could contain multiple storage spaces.
- A storage cluster can only belong to one storage space.

Dropping a cluster is not supported in an Ocient System that contains multiple storage clusters. However, you can drop a cluster in a system that contains only one cluster.

### Troubleshooting Considerations

- If a cluster shuts down, the load stops and provides results after the cluster is available again.
- The load is tolerant of Loader Node failures as the load can make progress when at least one Loader Node is still available.
- Before adding or removing a node, storage drive, or cluster, you must stop the load. Stop the LAT and any `CREATE TABLE AS SELECT` or `INSERT INTO SELECT SQL` statements. Otherwise, the load fails.

Contact Ocient Support for help with a complex multiple storage cluster setup.

## Related Links

[Core Elements of an Ocient System](#)

[Cluster and Node Management](#)

## 7.10 Loading Administration

### 7.10.1 Alter Default Data Pipeline Behavior

An administrator might configure some parameters that control the behavior of data pipelines to modify the default behavior across the Ocient® System or specific nodes. You can use the `ALTER SYSTEM ALTER CONFIG SQL` statements to control the configuration parameters.

### Syntax

```
ALTER SYSTEM ALTER CONFIG SET 'sql.pipelineparameters.parameter_name' 'parameter_value';
```

| Parameter       | Data Type | Description                                                                          |
|-----------------|-----------|--------------------------------------------------------------------------------------|
| parameter_name  | string    | The name of the pipeline parameter, either general or specific to the BINARY format. |
| parameter_value | string    | The value for the pipeline parameter.                                                |

### General Parameters

| General Parameter Name                                   | Default | Data Type | Required Value and Description                                                                                 |
|----------------------------------------------------------|---------|-----------|----------------------------------------------------------------------------------------------------------------|
| sql.pipelineparameters.targetTimePerExtractorTask        | 600     | UINT64_T  | Target execution time in seconds for <code>run_extractor</code> tasks for file loads.                          |
| sql.pipelineparameters.defaultPartitionsPerExtractorTask | 10      | INT       | Number of partitions per task. If you set this parameter to -1, the number is based on available CPU cores.    |
| sql.pipelineparameters.defaultExtractorCores             | 1       | INT       | Number of cores to use in a task. If you set this parameter to -1, the number is based on available CPU cores. |

### Binary Format Parameters

You can modify these parameters to change the default behavior of the BINARY data format extraction.

| Binary Format Parameter Name                                 | Default | Data Type | Required Value and Description                                                                                                         |
|--------------------------------------------------------------|---------|-----------|----------------------------------------------------------------------------------------------------------------------------------------|
| sql.pipelineparameters.extract.binary.defaultEndianness      | big     | string    | The endianness of transforms when you do not specify it in another way. Supported values are <code>big</code> or <code>little</code> . |
| sql.pipelineparameters.extract.binary.defaultCharacterSet    | UTF16   | string    | Determines the default character set to use when you load data in the BINARY format.                                                   |
| sql.pipelineparameters.extract.binary.defaultAutoTrimPadding | True    | Boolean   | Determines the default for whether a padding character should be trimmed when decoding data in the BINARY format to a string.          |

| Binary Format<br>Parameter Name                                            | Default           | Data Type         | Required Value and<br>Description                                                                                                                                                                                                                 |
|----------------------------------------------------------------------------|-------------------|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>sql.pipelineparameters.extract.binary.defaultPaddingCharacter</code> | <code>CHAR</code> | <code>CHAR</code> | Determines the default padding character to trim from the end of a string when decoding data in the <code>BINARY</code> format. This parameter is effective only if the <code>extract.binary.defaultAutoTrim</code> is set to <code>true</code> . |

### Examples

Set the target execution time for the extractor task to 600 seconds.

```
ALTER SYSTEM ALTER CONFIG SET 'sql.pipelineparameters.targetTimePerExtractorTask' 600;
```

Set the default endianness of transforms to little endian for the `BINARY` extract.

```
ALTER SYSTEM ALTER CONFIG SET 'sql.pipelineparameters.extract.binary.defaultEndianness' 'little';
```

The configuration values only go into effect after the `ALTER SQL` statement succeeds and you restart the affected nodes.

### Related Links

[Load Data](#)

[Data Types for Data Pipelines](#)

[Data Formats for Data Pipelines](#)

[Configuration Settings for Data Pipelines](#)

#### 7.10.2 Configuration Settings for Data Pipelines

Data pipeline functionality enables the use of SQL statements to load data. You can use a variety of configuration settings to manage the load. See this table for the settings that you can use.

| Configuration Setting<br>Parameter Name                     | Default           | Data Type            | Description                                                                                                                                                                                                                                                                |
|-------------------------------------------------------------|-------------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>streamloader.extractorEngineParameters.enabled</code> | <code>true</code> | <code>BOOLEAN</code> | Whether the Loader Node attempts to bootstrap the extractor <code>{{Java}}</code> process for data pipelines on start-up. If you set this parameter to <code>true</code> , the Loader Node does not become active until the Java loading process has successfully started. |

| Configuration Setting                                                                              |         |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------------------------------------------------------------------------------------|---------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameter Name                                                                                     | Default | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>streamloader.extractorEngineParameters.heapMemoryConfiguration</code>                        | 82m     | String    | The <code>{{JVM}}</code> memory configuration to use for the extractor Java process. Use strings representing data sizes in SI format: 1 KiB, 5 MiB, etc. You must set the three memory options in the same statement. <b>Example</b><br><code>ALTER SYSTEM ALTER CONFIG SET</code><br><code>'streamloader.extractorEngineParameters.heapMemoryConfiguration = '64g',</code><br><code>'streamloader.extractorEngineParameters.offHeapMemoryConfiguration = '64g',</code><br><code>'streamloader.extractorEngineParameters.restPort = '64g';</code> |
| <code>streamloader.extractorEngineParameters.restPort</code>                                       | 8000    | INT       | Use this parameter with the <code>streamloader.extractorEngineParameters</code> to form the REST port for the extractor. (External monitoring services like <code>{{Telegraf}}</code> can query the REST server for loading process metrics.)                                                                                                                                                                                                                                                                                                      |
| <code>streamloader.extractorEngineParameters.portOffset</code>                                     | 0       | INT       | Use this parameter with the <code>streamloader.extractorEngineParameters</code> to form the REST port of the extractor. (External monitoring services like <code>Telegraf</code> can query the REST server for loading process metrics.)                                                                                                                                                                                                                                                                                                           |
| <code>streamloader.extractorEngineParameters.configurationOption.expect.empty.fail</code>          | FALSE   | BOOLEAN   | Set this value to <code>false</code> to fail a file source data pipeline that finds no files to load instead of the load completing successfully with no data loaded.                                                                                                                                                                                                                                                                                                                                                                              |
| <code>streamloader.extractorEngineParameters.configurationOption.pipeline.preview.rowsLimit</code> | 1000    | INT       | The rows limit number of rows a <code>PREVIEW PIPELINE SQL</code> statement can return. For example, when you use this statement, if the specified <code>LIMIT</code> value exceeds this number, then the loading process throws an error.                                                                                                                                                                                                                                                                                                         |

| Configuration Setting                                         |                                         |                      |                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------------------------------------------|-----------------------------------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameter Name                                                | Default                                 | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>streamloader.extractorEngineParameters.configDir</code> | <code>EngineParameters.configDir</code> | <code>VARCHAR</code> | The default location of the directory containing JARs for data pipeline functions. To install and enable a third-party library for using data pipeline functions, install the JAR package at this location on all Loader Nodes. Then, add the chosen fully-qualified class name to the <code>IMPORT</code> clause of the data pipeline function. For details, see <a href="#">Data Pipelines</a> . |
| <code>streamloader.extractorEngineParameters.configDir</code> | <code>EngineParameters.configDir</code> | <code>BOOLEAN</code> | The size of the source record, in bytes, that the loading process tolerates before throwing an error.                                                                                                                                                                                                                                                                                              |
| <code>streamloader.extractorEngineParameters.configDir</code> | <code>EngineParameters.configDir</code> | <code>VARCHAR</code> | The default region for {{AWS}} S3 file sources.                                                                                                                                                                                                                                                                                                                                                    |
| <code>streamloader.extractorEngineParameters.configDir</code> | <code>EngineParameters.configDir</code> | <code>BOOLEAN</code> | Style is access to use path-style access for all S3 file sources. The default is a virtual-hosted style.                                                                                                                                                                                                                                                                                           |
| <code>streamloader.extractorEngineParameters.configDir</code> | <code>EngineParameters.configDir</code> | <code>BOOLEAN</code> | Style is access to use path-style access for S3 file sources that specify endpoints. (The system still uses virtual-hosted style access for sources without endpoint overrides unless you                                                                                                                                                                                                          |
| <code>streamloader.extractorEngineParameters.configDir</code> | <code>EngineParameters.configDir</code> | <code>INT</code>     | The number of retries for the S3 file source client.                                                                                                                                                                                                                                                                                                                                               |
| <code>streamloader.extractorEngineParameters.configDir</code> | <code>EngineParameters.configDir</code> | <code>INT</code>     | The time in seconds for read operations, in seconds, for the S3 file source client. 0 means do not perform a timeout.                                                                                                                                                                                                                                                                              |
| <code>streamloader.extractorEngineParameters.configDir</code> | <code>EngineParameters.configDir</code> | <code>INT</code>     | Maximum number of allowed concurrent requests for the S3 file source client.                                                                                                                                                                                                                                                                                                                       |

| Configuration Setting                                                                                      | Default | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                          |
|------------------------------------------------------------------------------------------------------------|---------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>streamloader.extractorEngineParameters.configuration.option.awssdk.backoffStrategy.maxBackoff</code> | 20      | INT       | The retry backoff delay, in seconds, for calculating the time the S3 file source client waits before retrying a failed request. The range of values that the calculated time can achieve increases exponentially with each failure. You can set the maximum value using <code>streamloader.extractorEngineParameters.configuration.option.awssdk.backoffStrategy.maxBackoff</code> . |
| <code>streamloader.extractorEngineParameters.configuration.option.awssdk.maxPendingConnections</code>      | 2000    | INT       | The maximum number of pending connection acquires that the S3 file source client allows.                                                                                                                                                                                                                                                                                             |
| <code>streamloader.extractorEngineParameters.configuration.option.awssdk.connectTimeout</code>             | 10      | INT       | The timeout, in seconds, that the S3 file source client waits when initially establishing a connection.                                                                                                                                                                                                                                                                              |
| <code>streamloader.extractorEngineParameters.configuration.option.awssdk.connectIdleTimeout</code>         | 800     | INT       | The maximum timeout, in seconds, that the system allows a connection established by the S3 file source client to remain open while idle.                                                                                                                                                                                                                                             |
| <code>streamloader.extractorEngineParameters.configuration.option.awssdk.connectAcquisitionTimeout</code>  | 800     | INT       | The acquisition timeout, in seconds, that the S3 file source client waits when acquiring a connection from the pool.                                                                                                                                                                                                                                                                 |
| <code>streamloader.extractorEngineParameters.configuration.option.kafka.pollTimeout</code>                 | 1000    | INT       | The amount of time, in milliseconds, that a consumer for a pipeline with an <code>{{Kafka}}</code> source waits for records to become available during a poll operation.                                                                                                                                                                                                             |

| Configuration Setting<br>Parameter Name                                                                                 | Default           | Data Type | Description                                                                                                                                                                                                                                                                                                      |
|-------------------------------------------------------------------------------------------------------------------------|-------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>streamloader.extractorEngineParameters.configurationOption.kafka.assign.timeout.ms</code>                         | 2000              | BIGINT    | The amount of time, in milliseconds, that a PREVIEW PIPELINE SQL statement waits for partitions to be assigned to the consumer on a pipeline with a Kafka source. If the system does not assign any partitions when the timeout elapses, then the PREVIEW PIPELINE statement terminates without data processing. |
| <code>streamloader.extractorEngineParameters.configurationOption.kafka.activatePartitionTimeout</code>                  | 5000              | BIGINT    | The amount of time, in milliseconds, that a PREVIEW PIPELINE SQL statement waits for a partition to receive records on a pipeline with a Kafka source.                                                                                                                                                           |
| <code>streamloader.extractorEngineParameters.configurationOption.arrow.max.native.memory.usage.bytes</code>             | 2975858756        | BIGINT    | The default usage in bytes of native memory usage, in bytes, for the {{Parquet}} reader.                                                                                                                                                                                                                         |
| <code>streamloader.extractorEngineParameters.configurationOption.arrow.max.java.buffer.memory.usage.bytes</code>        | 15887691004       | BIGINT    | The default usage in bytes of maximum buffer memory usage, in bytes, for the Parquet reader.                                                                                                                                                                                                                     |
| <code>streamloader.extractorEngineParameters.configurationOption.hadoop.fs.filesystem.access.allowed.directories</code> | <code>/tmp</code> | VARCHAR   | A semicolon-separated list of directories where you can load server file system data. Each directory must exist. All files must be located within one of the directories. By default, you can load from the <code>/tmp</code> directory.                                                                         |

## Related Links

[Load Data](#)

[Alter Default Data Pipeline Behavior](#)

### 7.10.3 Configure Data Pipeline Logging

You can troubleshoot data pipelines using the standard logging mechanism in Ocient®.

Set these log levels in consultation with Ocient Support because this setting might impact performance.

#### Modify the Level of Logging

Execute these two SQL statements in this order to bump up the log message.



First, set the log level.

```
ALTER SYSTEM ALTER LOG LEVEL SET 'log4j' 'EDEBUG';
```

Then, set the level for the logging configuration.

```
ALTER SYSTEM ALTER CONFIG SET 'streamloader.extractorEngineParameters.loggingConfig.com.ocient.extractor.a
```

If the log level of the first SQL statement is lower than the log level of the second SQL statement, the system loses log entries.

It is also possible to repeat the second SQL statement for different package names, such as:

- ALTER SYSTEM ALTER CONFIG SET 'streamloader.extractorEngineParameters.loggingConfig.com.ocient.extractor.a  
'Info';
- ALTER SYSTEM ALTER CONFIG SET 'streamloader.extractorEngineParameters.loggingConfig.com.ocient.extractor.a  
gine.allowLevel' 'Trace';

### Valid Log Levels

This table provides an overview of the log levels you can specify for the ALTER SYSTEM ALTER LOG LEVEL SET SQL statements with the `streamloader.extractorEngineParameters.loggingConfig` keys. The necessary log level for `log4j` must be set accordingly to at least the value specified in the column for the `log4j` level.

| Log Level of<br><code>streamloader.extractorEngineParameters.loggingConfig</code> | Derived Level for <code>log4j</code> |
|-----------------------------------------------------------------------------------|--------------------------------------|
| FATAL                                                                             | ERROR                                |
| ERROR                                                                             | ERROR                                |
| WARN                                                                              | WARN                                 |
| INFO                                                                              | INFO                                 |
| DEBUG                                                                             | DEBUG                                |
| TRACE                                                                             | EDEBUG                               |

### Reset the Log Level

To reset the log level after you reproduce a problem, execute these SQL statements.

```
ALTER SYSTEM ALTER CONFIG RESET 'streamloader.extractorEngineParameters.loggingConfig.com.ocient.extractor.a
ALTER SYSTEM ALTER CONFIG RESET 'log4j';
```

### Related Links

[Monitor Data Pipelines](#)

[Data Pipeline Loading Errors](#)

## 7.11 Air-Gapped Environment with an Ocient System

An air gap is a security measure that physically isolates a computer or network from any other network, including the internet. This isolation removes wired or wireless connections between the isolated system and any external and potentially unsecured networks. Organizations that handle highly sensitive information, such as government agencies, financial institutions, and critical infrastructure operators, use air-gapped environments to protect against various security incidents.

Ocient® supports using an Ocient System in an air-gapped environment without any special configuration. Here, you can find the general workflow for using an Ocient System in an air-gapped environment and the required installation steps.

## Work with an Ociant System in an Air-Gapped Environment

This workflow shows the high-level steps to set up an Ociant System in an air-gapped environment.

Ensure that your drive firmware and operating system is up to date.

Install and configure an Ociant System in a connected environment where you can download the required files from the internet.

Disassemble the cluster.

Pack and deliver the hardware to your secured location.

Reassign a new IP address.

Reassign a new hostname.

Bootstrap the system again.

## Install an Ociant System in an Air-Gapped Environment

Follow these steps to install the Ociant System in an air-gapped environment. These steps assume that you have met the system requirements for installation.

### Bootstrap the First Node

#### Bootstrap the Node

Connect to the initial SQL Node with the username and password of your server and the IP address of your node. This example connects as the administrator **admin** to the IP Address **10.10.10.10**.

```
ssh admin@10.10.10.10
```

Use your preferred text editor with **sudo** to create the **/var/opt/ocient/bootstrap.conf** file as root with these contents.

```
/var/opt/ocient/bootstrap.conf example
```

```
initialSystem: true
```

Start the database.

```
sudo systemctl start rolehostd
```

Verify that the node and the service is active by executing this status command.

```
systemctl status rolehostd
```

```
rolehostd.service - Rolehostd daemon startup
   Loaded: loaded (/etc/systemd/system/rolehostd.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-01-26 23:31:36 UTC; 7s ago
   ...
```

If the **rolehostd** service is running, you can also check the Ociant logs on your node. Search and ensure there are no **[ERROR]** log messages.

```
tail -f /var/opt/ocient/log/rolehostd.log
```

### Verify Connection to the SQL Node

At this point, you have a running database with a single node. You should be able to connect to the database using **JDBC** or **pyocient** and execute commands.

Every new system starts with a **system** database. To connect to a new system, use the username and password configured in the **bootstrap.conf** file or the username **admin@system** and password **admin** if none was provided.

For example, assume your node named `sql0` has an IP address of `10.10.0.1`. Use the JDBC driver CLI to connect with this connection string.

```
connect to jdbc:ocient://10.10.0.1:4050/system user "myuser@system" using "mypassword";
```

To see the roles running on the single node, execute this query.

```
select name, operational_status, software_version, array_agg(service_role_type)
  from sys.node_status as ns
 left join sys.nodes as n on ns.node_id = id
 left join sys.service_roles as sr on sr.node_id = n.id
 group by name, operational_status, software_version;
```

The initial node should be listed as running the `sql`, `admin`, `health`, and `operatorvm` roles. If all of these roles are present and the node is Active, you can proceed to the next step to bootstrap the remaining nodes.

### Bootstrap the Remaining Nodes

Performing the bootstrapping process on the remaining nodes is identical on all nodes. The remaining nodes can be bootstrapped in any order.

On each node, log in using SSH and use your text editor with `sudo` to create the file `/var/opt/ocient/bootstrap.conf` that contains this text by replacing `<FIRST_NODE_IP_ADDRESS>` with the IP address of the initial node you created in Step 1.

`/var/opt/ocient/bootstrap.conf` example

```
adminHost: <FIRST_NODE_IP_ADDRESS>
```

`<FIRST_NODE_IP_ADDRESS>` is the DNS name or IP address of the initial node.

You can obtain the IP Address of the initial node by executing `ifconfig` on that node.

If the password for the system administrator has changed, set the correct username `adminUserName` and password `adminPassword` in the bootstrap configuration file `bootstrap.conf`.

On each node, start the database.

```
sudo systemctl start rolehostd
```

When you replace Foundation Nodes, the Ocint System removes the prior node after the creation of the new node. Some queries of the system catalog tables might not return results until the prior node is removed.

At this point, all the remaining nodes are not configured with any roles. After all nodes have been started, you should see them when you execute this query with only the `health` role listed. This query uses the `sys.node_status`, `sys.nodes`, and `sys.service_roles` system catalog tables to retrieve node information for the node name, operation status, version, and all service role types. The query uses the `ARRAY_AGG` function to retrieve the service role type for all rows.

```
SELECT name, operational_status, software_version, array_agg(service_role_type)
  FROM sys.node_status AS ns
 LEFT JOIN sys.nodes AS n ON ns.node_id = id
 LEFT JOIN sys.service_roles AS sr ON sr.node_id = n.id
 GROUP BY name, operational_status, software_version;
```

## Remove a Cluster in an Air-Gapped Environment

Use this information and workflow to remove a cluster by shutting down all Ocident processes and erasing the drives and data.

The system administrator must download the `sedutil-cli` utility used in this workflow.

You can classify the drives in an Ocident System in two categories based on their usage.

- **Operating System (OS) drives:** These drives contain the installation of the operating system and software, including the Ocident Hyperscale Data Warehouse™ (OHDW). A system might have a single physical OS drive or an OS installed on a RAID disk created by using more than one drive. The OS drives on nodes with the administrator role (Metadata and possibly SQL Nodes) also store configuration information related to the OHDW that can include:
  - Node names
  - Node IP addresses
  - User data mapping information for compressed columns
  - Encryption keys for the data drives (unless the keys are under the control of an external key management system)
- **Data drives:** The data drives are present in all types of nodes except the node running only the administrator role. On the Foundation Node, the data drives store tables. On SQL Nodes, the data drives store transient query information. And, on Loader Nodes, the data drives store transient loading information.

The data drives in the OHDW are exclusively NVMe drives. The OS drives can be NVMe or SSD. These drives can support The Computing Group (TCG) Opal Specification or not support it. The type of drive (Opal-supported or not) determines how the system removes all data on the drive. Follow these steps to remove a cluster for Opal-supported drives. To remove data for Opal-supported drives, follow these steps.

These steps irreversibly remove the data. Follow these steps after you ensure that you no longer need the applicable data from the system.

Stop the Ocident processes running on the nodes using these commands.

For Loader Nodes, use this command. The command combines two commands that stop all processes related to loading.

```
sudo systemctl disable lat && sudo systemctl stop lat
```

For all types of nodes, use this command. The command combines two commands where the first command stops processing on all nodes and the second command stops the main Ocident System process.

```
sudo systemctl disable rolehostd && sudo systemctl kill -s SIGKILL rolehostd
```

Bind the data drives to an NVMe driver so that the drives become visible to the OS.

```
sudo /opt/ocident/scripts/nvme-driver-util.sh bind-nvme
```

Run this command for each data drive. The impacted drives are the drives displayed as their association change from `uio` drivers to `nvme` drivers. You can see the `nvme` drive association by running the `/opt/ocident/scripts/nvme-driver-util.sh` script. All data drives are erased.

You can move drives to another node with the `sedutil-cli` utility if the nodes are already powered off and you cannot power them on. Use the same command to erase the drive: `sedutil-cli -n --revertTPer admin /dev/nvmeXn1`.

```
sedutil-cli -n --revertTPer admin /dev/nvmeXn1
```

You cannot securely erase the OS drives while the OS is running. You can either use the secure erase facility from BIOS or move drives to an external host with a utility approved by your organization for the remove operation.

---

## 8 Ocient Simulator

Ocient® provides a simulated database system upon request for development or local workloads with no performance requirements.

This simulator system is a lightweight package that emulates a complete Ocient System with these features:

- SQL port for connection using any of the supported connectors from Ocient.
- No NVMe drive or Linux® huge page requirements.

To request access to an Ocient simulator, please contact Ocient Support.

### Prerequisites

The Ocient simulator has these hardware and software requirements:

#### Hardware

- 8GB system RAM
- 64-bit Intel® architecture CPU (x86)
- Allocated disk storage equal to twice the size of the data to load.

If you run the Ocient Simulator on a virtual machine, you must configure your virtual machine CPU to support AVX2 instructions by setting the CPU type to `host`.

#### Software

Docker® Engine version 20.10.10 or greater with Docker Compose version 1.9.0 or greater.

If you do not have the minimum version of Docker Engine installed, you can use this workaround to run the Ocient Simulator with an older version.

Add the YAML option `security_opt: ["seccomp:unconfined"]` to the LAT service in the `docker-compose.yml` file in between the `image` and `hostname` keys.

```
lat:
  image: "ocient/lat/simulator:3.0.2"
  security_opt: ["seccomp:unconfined"]
  hostname: "lat"
```

Add the command-line argument `--security-opt seccomp=unconfined` immediately after the `docker run` command in both `jdbc.sh` and `lat_client.sh` scripts.

```
exec docker run --security-opt seccomp=unconfined -it --rm -v $(pwd):/tmp/ocient -w /tmp/ocient --network
```

```
exec docker run --security-opt seccomp=unconfined $env_arg --rm -v $(pwd):/tmp/ocient -w /tmp/ocient --net
```

### Install the Ocient Simulator

Ocient distributes its simulator as the tarball `simulator_archive.tar`. Upon receiving the file, you can extract it and load the container images into the Docker daemon with the provided `load_all_images.sh` file.

Some file names might differ based on the simulator release version.

From a terminal window, create a new empty directory.

```
mkdir simulator_workspace
```

And then change the directory to the new one.

```
cd simulator_workspace
```

Copy or move the tarball file into the new directory.

Extract the file.

```
tar xf simulator_archive.tar
```

The simulator package contains these files.

| File Name                                                                         | Description                                                                                           |
|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| configure.sql                                                                     | The system configuration script that bootstraps the simulator.                                        |
| docker-compose.yml                                                                | The Docker Compose file that defines the containers that constitute the simulator system.             |
| lat_client.tar, lat_simulator.tar, ocient_jdbc_image.tar, rolehostd_simulator.tar | These tarballs contain various components of the Ocient System.                                       |
| jdbc.sh                                                                           | The JDBC CLI launch script for connection to the SQL interface of the simulator.                      |
| lat_client.sh                                                                     | The Loading and Transformation (LAT) client launch script for connection to the LAT of the simulator. |
| load_all_images.sh                                                                | This script loads all the tarball files into the Docker daemon.                                       |

Load the Ocient System tarball files that form the simulator.

```
./load_all_images.sh
```

If you do not have access privileges to the Docker daemon, you might receive the **Permission denied** error when you run **docker** commands. Use **sudo** or add your user to the Docker group.

This output shows an example of a successful load operation. The exact output of your simulator might differ depending on what version you download.

```
a0c3d7b00a1e: Loading layer [=====>] 59.26MB/59.26MB
5d75db409355: Loading layer [=====>] 329.6MB/329.6MB
b4f0ab1562a2: Loading layer [=====>] 3.072kB/3.072kB
6b6be244d498: Loading layer [=====>] 94.76MB/94.76MB
Loaded image: ocient/lat/simulator:2.3.0
fb4617b63504: Loading layer [=====>] 29.39MB/29.39MB
f5745a52e122: Loading layer [=====>] 4.608kB/4.608kB
d2d33759728d: Loading layer [=====>] 12MB/12MB
f6533a5b289e: Loading layer [=====>] 10.24kB/10.24kB
a3f69486f426: Loading layer [=====>] 23.64MB/23.64MB
84ff92691f90: Loading layer [=====>] 10.24kB/10.24kB
Loaded image: ocient/lat_client:2.3.0
Loaded image: ocient/jdbc:2.100
475a54c2a93d: Loading layer [=====>] 65.52MB/65.52MB
314fd346d653: Loading layer [=====>] 1.32GB/1.32GB
Loaded image: ocient/rolehostd/simulator:23.0.0-1680124029
```

## Start and Configure the Cluster

Each Ocient simulator is a series of nodes that run in their own individual Docker containers. The **docker compose** command enables networking, general administration, and orchestration of these nodes.

The commands you use based on this section might differ with your Docker version. These steps use the `docker compose` command for the simulator setup. If you are using an older version of Docker with `docker-compose` (a dash instead of a space), use that instead.

Start the cluster using the `docker compose` command.

```
docker compose up -d
```

The cluster startup displays output that is similar to this output.

```
[+] Running 7/7
Network simulator_workspace_ocientdbnet      Created           0.1s
Container simulator_workspace-sql-1          Started           20.2s
Container simulator_workspace-foundation0-1   Started           15.4s
Container simulator_workspace-foundation2-1   Started           17.4s
Container simulator_workspace-foundation1-1   Started           16.4s
Container simulator_workspace-loader-1        Started           14.4s
Container simulator_workspace-lat-1           Started            7.1s
```

Ensure that you run every command from the same directory where you extracted the `simulator_archive.tar.gz` archive.

Using the bundled JDBC CLI, connect to the cluster and run the contents of the `configure.sql` file that is distributed with the simulator.

The JDBC CLI is bundled with the simulator as the `jdbc.sh` script. Run this script with no arguments after starting the cluster.

```
./jdbc.sh
```

After loading the JDBC CLI, the system prompts you to enter a username and password. The default simulator username and password are `admin@system` and `admin`.

You can also pass these credentials later in a `CONNECT` statement, for example:  
`CONNECT TO jdbc:ociant://sql:4050/system USER admin@system USING admin;`

The interface switches to the JDBC CLI interactive mode, as indicated by the `Ocient>` prompt. Run the configuration file.

JDBC CLI

```
source configure.sql;
```

The configuration displays output that is similar to this output.

```
Connected to jdbc:ociant://sql:4050/system
Modified 0 rows
Modified 0 rows
Modified 0 rows
Warning: Node must be restarted for changes to take effect (the streamloader role has been added to the co
Modified 0 rows
Modified 0 rows
Modified 0 rows
```

After completing its modifications, the simulator warns that it needs to restart for the changes to take effect. Quit the interactive mode.

JDBC CLI



---

```
quit
```

Restart the cluster.

```
docker compose restart
```

After the nodes restart, verify that the simulator has started and the `test` database is accessible by connecting to it with the JDBC CLI.

Launch the JDBC connector again.

```
./jdbc.sh
```

Connect to the `test` database in the JDBC CLI.

JDBC CLI

```
CONNECT TO jdbc:ociant://sql:4050/test;
```

The system displays a message to confirm a successful connection.

JDBC CLI

```
Connected to jdbc:ociant://sql:4050/test
```

## Load Data with the LAT Client

The simulator cluster includes an LAT Node, which can load data from external sources, such as an Amazon® Web Services<sup>SM</sup> (AWS<sup>SM</sup>) S3 bucket. This section has the steps to create a basic table and pipeline with the LAT client, which you can then use to load data from a sample data set into a simulator database.

If you are not already in the JDBC CLI interactive mode, start it and connect to the `test` database using the steps in [Ocient Simulator](#).

Create a table to load. From the JDBC CLI, enter this SQL CREATE TABLE statement.

```
CREATE TABLE loading.data_type_coverage (  
  col_bigint BIGINT NULL,  
  col_binary BINARY(2) NULL,  
  col_boolean BOOL NULL,  
  col_char CHAR(4) NULL,  
  col_date DATE NULL,  
  col_decimal DECIMAL(18,4) NULL,  
  col_double DOUBLE NULL,  
  col_float FLOAT NULL,  
  col_int INT NULL,  
  col_int_array INT[] NULL,  
  col_ipv4 IPV4 NULL,  
  col_ip IP NULL,  
  col_smallint SMALLINT NULL,  
  col_point POINT NULL,  
  col_linestring LINESTRING NULL,  
  col_polygon POLYGON NULL,  
  col_time TIME NULL,  
  col_timestamp TIMESTAMP NULL,  
  col_tinyint TINYINT NULL,  
  col_tuple tuple<<INT, VARCHAR(255)>> NULL,  
  col_uuid UUID NULL,  
  col_varbinary VARBINARY(255) NULL,  
  col_varchar VARCHAR(255) NULL  
);
```

---

Create a pipeline to load the data using the LAT client. Exit the JDBC CLI by entering the `quit` command.

Create a new file named `pipeline.json` in the directory where you are running the simulator.

Copy this JSON text and paste it into the `pipeline.json` file.

```
{
  "version": 2,
  "workers": 1,
  "source": {
    "compression": "gzip",
    "type": "s3",
    "endpoint": "https://s3.us-east-1.amazonaws.com",
    "bucket": "ocient-examples",
    "file_groups": {
      "test": {
        "prefix": "all_data_types/small",
        "file_matcher_syntax": "glob",
        "file_matcher_pattern": "**output*.csv.gz",
        "sort_type": "lexicographic"
      }
    }
  },
  "sink": {
    "type": "ocient",
    "remotes": ["loader:5050"]
  },
  "extract": {
    "test": {
      "record_type": "delimited",
      "headers": [
        "bigint",
        "binary",
        "boolean",
        "char",
        "date",
        "decimal",
        "double",
        "float",
        "int",
        "int_array",
        "ipv4",
        "ip",
        "smallint",
        "point",
        "linestring",
        "polygon",
        "time",
        "timestamp",
        "tinyint",
        "tuple",
        "uuid",
        "varbinary",
        "varchar"
      ],
      "record_delimiter": "\n",
      "field_delimiter": ",",
    }
  }
}
```

```

        "field_optionally_enclosed_by": "\"",
        "num_header_lines": 1
    },
    "transform": {
        "file_groups": {
            "test": {
                "tables": {
                    "test.loading.data_type_coverage": {
                        "columns": {
                            "col_bigint": "bigint",
                            "col_binary": "binary",
                            "col_boolean": "boolean",
                            "col_char": "char",
                            "col_date": "date",
                            "col_decimal": "decimal",
                            "col_double": "to_number(double)",
                            "col_float": "to_number(float)",
                            "col_int": "int",
                            "col_int_array": "parse_array(int_array, '[', ',', '1')",
                            "col_ipv4": "ipv4",
                            "col_ip": "ip",
                            "col_smallint": "smallint",
                            "col_point": "point",
                            "col_linestring": "linestring",
                            "col_polygon": "polygon",
                            "col_time": "time",
                            "col_timestamp": "replace(replace(timestamp, 'T', ' '), 'Z', ' ')",
                            "col_tinyint": "tinyint",
                            "col_tuple": "to_tuple(tuple, '(', ',', ')",
                            "col_uuid": "uuid",
                            "col_varbinary": "varbinary",
                            "col_varchar": "varchar"
                        }
                    }
                }
            }
        }
    }
}

```

Execute this command to create the pipeline by using the LAT client script.

```
./lat_client.sh pipeline create --pipeline pipeline.json
```

Unlike the JDBC CLI, the LAT client is not interactive. You must issue each command for the LAT client as a separate `./lat_client` command.

Start the pipeline to load the data.

```
./lat_client.sh pipeline start
```

The sample data set can take about 30 seconds to fully load.

Inspect the tables using the JDBC CLI to validate the load of the data set. Launch the JDBC CLI.

```
./jdbc.sh
```

Connect to the `test` database.

JDBC CLI

```
CONNECT TO jdbc:ociant://sql:4050/test;
```

Verify the `loading.data_type_coverage` table is present.

JDBC CLI

```
LIST TABLES;
```

Verify the row count in the table using this SQL statement.

```
SELECT COUNT(*) FROM loading.data_type_coverage;
```

If the load operation is complete, the output table shows a count of 100,000 rows.

```
count(*)
```

```
-----  
100000
```

```
Fetches 1 row
```

When you are confident your data is fully loaded, you can dismantle your pipeline.

Stop the pipeline.

```
./lat_client.sh pipeline stop
```

Delete the pipeline.

```
./lat_client.sh pipeline delete
```

Now that your Ocient simulator is loaded with data, you can add integrations for user interfaces, visualizations, and other applications. For details, see [Ocient Integrations](#).

## Container Configuration

This section explains the configuration setup and options for the Ocient simulator.

### Ports

The container exposes additional ports that can be useful depending on the use case for the system.

| Exposed Port  | Description                                                                        |
|---------------|------------------------------------------------------------------------------------|
| 4050          | The JDBC port of the cluster.                                                      |
| 8080          | Ocient LAT service. When you load new data, the LAT client connects to this port.  |
| 9090 and 9091 | The REST interfaces for the SQL and Foundation (LTS) Node processes, respectively. |

### Environment Variables

You can set these environment variables to affect certain aspects of the Ocient simulator.

| Environment variable             | Description                                                                                                                                                                                                                          |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LAT_MEMORY and LAT_MEMORY_DIRECT | Configures the memory used by LAT when it is running. For details about these environment variables, see the <a href="#">LAT Service Configuration</a> page. The LAT Node is the only node in the cluster affected by this variable. |

### Example

```
services:
  lat:
    environment:
      LAT_MEMORY: '32G'
```

### Resource Limits (CPU, Memory, Disk Usage)

Docker supports a variety of options to limit container access to host resources. Refer to the [Docker Runtime options with Memory, CPUs, and GPUs](#) page for the full list of options. This section explains how constraints on specific host resources affect the Ocient simulator.

In all of these code examples, replace **IMAGE** with the filename of your Docker image.

The Ocient System applies Docker resource configuration changes individually on each of the nodes in the `docker-compose.yml` file (such as `sql`, `foundation`, `loader`, `lat`).

### CPU

CPU can be safely limited without adverse effects. Most workloads benefit from having multiple CPU cores available (four or more).

For example, to add CPU limits to the `sql` node, add this text to the `docker-compose.yml` file in the simulator package. Replace the '4' with the CPU limit that you want.

```
services:
  sql:
    image: IMAGE
    deploy:
      resources:
        limits:
          cpus: '4'
```

### Memory (RAM)

The Ocient simulator requires sufficient RAM to execute all workloads. Stability or uptime of the system is not guaranteed in low or out-of-memory conditions. If you need to limit the memory of the running Docker container, it is recommended to benchmark the container memory usage on your workload and provide at least 1GB of RAM above the maximum observed.

When limiting memory of the Ocient Simulator, do not use the `--oom-kill-disable` option, otherwise the database can become unresponsive for a significant period of time after running out of memory.

To add memory limits to the `sql` node, add this text to the `docker-compose.yml` file in the simulator package. Replace the 8g with the memory limit that you want.

```

services:
  sql:
    image: IMAGE
    deploy:
      resources:
        limits:
          memory: 8g

```

## Disk Usage

When running the simulator without mounted data, you can specify the size in bytes of the data files created. The data files must be large enough to fit any data sets that you load. The file size defaults to 2GiB.

To specify a different file size, you can configure the `rolehostd` startup command in the `docker-compose.yml` file. Add the command option `loopbackDeviceSize` with a new value to configure a new file size.

In this example, the file size on the `foundation0` node is configured to 16GiB.

```

services:
  sql:
    image: IMAGE
    command: "--bootstrap '{adminHost: \"sql\", ipAddress: \"foundation0\", loopbackDeviceSize: \"16GiB\"}'"

```

If you want to specify a new file size value, you must do it before you create your cluster.

While you cannot resize files on a running cluster, you can change this value by deleting and recreating your cluster. This process is irreversible and erases any existing databases and tables.

To resize file on an existing cluster:

1. Stop your cluster by running `docker compose down`.
2. Clear the `volumes` directory by running `rm -rf volumes/`.
3. If you have not yet edited the `docker-compose.yml` file, edit the file to include new `loopbackDeviceSize` values.
4. Restart your cluster by running `docker compose up -d`.

## Simulator Logging

The Ocient simulator sends all logs to stdout and stderr as well as any logging driver configured for the host Docker installation or the container. This logging allows for easy inspection of container logs using the `docker logs` command.

For details on logging with Docker, consult the logging section in the Docker documentation.

## Setting Logging Levels

You can modify the logging level for each node in the cluster by configuring the `rolehostd` startup command in the `docker-compose.yml` file. Add the command option `logLevel` with a new log level.

Supported values include `error`, `info`, `warn`, `verbose`, or `debug`. Defaults to `info`.

For details, see [Log Monitoring](#).

## Example

```

services:
  sql:
    image: IMAGE
    command: "--bootstrap '{adminHost: \"sql\", ipAddress: \"foundation0\", logLevel: \"debug\"}'"

```

---

## Other Logging Configurations

You can customize your logging configuration for each node by making changes to the `docker-compose.yml` file. In this example, the `sql` node has special configurations for its `max-size` and `max-file` values, meaning the container should have no more than five log files of 10MB in size.

### Example

```
services:
  sql:
    image: IMAGE
    logging:
      driver: json-file
      options:
        max-size: 10m
        max-file: 5
```

Use `docker logs` to retrieve the logs. In this example, the `--tail` option limits the output to include only the last 10 lines of logs.

For more options, see the [docker compose logs](#) page.

```
docker logs --tail 10 CONTAINER
```

## Related Links

[Commands Supported by the Ocient JDBC CLI Program](#)

[Database, Tables, Views, and Indexes](#)

---

## 9 Ocient Architecture

The Ocient Hyperscale Data Warehouse™ is a modern, scalable distributed system capable of enabling complex OLAP-style analysis on the world's largest structured and semi-structured datasets using SQL. It features a purpose-built storage and I/O layer designed for highly parallel interaction with modern NVMe SSD drives in a Compute Adjacent Storage Architecture® (CASA) that leverages erasure coding for cost-efficient reliability.

On top of this is a fully integrated execution engine and optimizer that were each built from scratch using modern design principles in a manner that deeply integrates with the storage and I/O layer. The result is a state-of-the-art data warehouse that can deliver outstanding, and even unprecedented, analytical capability on datasets with record sets numbering in the trillions, quadrillions, and beyond.

This section includes these topics:

- [Core Elements of an Ocient System](#)
- [Compute-Adjacent Input and Output on Large Working Sets](#)
- [Query Performance Optimizations](#)
- [Plan Optimization](#)
- [Exabyte Scalability Design](#)
- [Loading Characteristics and Concepts](#)

### 9.1 Core Elements of an Ocient System

#### The System and Nodes

The Ocient® System is the collection of all of the nodes within a single environment. This architecture diagram is an example of one Ocient System. Across systems, the number of nodes can vary along with the total storage capacity. Within a single system, users can define varying numbers of databases, storage groups, users, etc.

An Ocient System is made up of a collection of distributed nodes which each play a predefined role in the data warehouse. The different roles are generally responsible for storing and analyzing data, administration, and moving data into Ocient. These nodes are connected to each other through a high-speed 100 Gbps network connection. In addition to storing user data, the data warehouse also stores metadata about the system and data for its own internal operation and recovery. This collection of nodes, their data, and the interconnection between them make up the Ocient System.

The end user, analyzing data in Ocient, connects to this system through a JDBC or a Python-based client. From there, they interact with the database objects, such as tables and views. The architecture itself is abstracted from users, allowing them to focus on SQL queries and results.

While the number of nodes in an environment might vary, the inclusion of and purpose of the different roles do not. The following sections briefly describe the components in the architecture.

#### Node Types

##### SQL Nodes

A SQL Node is responsible for receiving incoming SQL requests and parsing the statements. This node also serves as the interface for System Administrators and Database Administrators to configure and maintain an Ocient System. Administrators can connect to a SQL Node using the Ocient command-line interface or their chosen client.

When a SQL Node receives a query, it creates a plan for it using one of two optimization methods. After there is a plan for the query, the SQL Node distributes it to the Foundation Nodes. When the Foundation Nodes finish their processing, they return data back to the SQL Node for further processing and packaging the result set to the client. When the result set is finalized, the SQL Node returns the data to the client. Beyond the Foundation Nodes, further processing is done on the SQL Nodes to process intermediate result sets delivered by the Foundation Nodes. Depending on the query, the SQL Node is responsible for a different proportion of the overall work of the query. Common processing done here are aggregations and joins that happen across the intermediate results delivered by the Foundation Nodes.



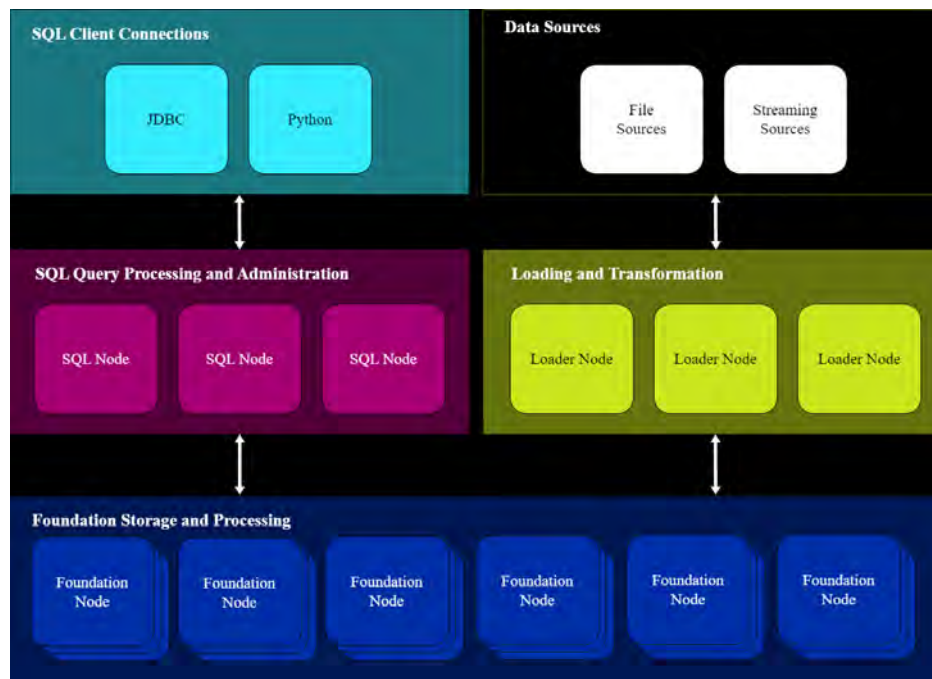


Figure 19: Ociant architecture diagram that shows the relationship between data sources, loading and transformation of the data, and data storage

Administrators also use DDL or DCL statements to make changes to an Ociant System. When submitted to a SQL Node, the node executes the SQL statement across the system or forwards the SQL statement to the node assigned to execute it.

### Foundation Nodes

The Foundation Nodes store the user data in Ociant and perform the bulk of query processing.

A key architectural concept of Ociant is the Compute Adjacent Storage Architecture®, which collocates data and processing where possible. The Foundation Node is the central element of this architectural principle. When a query is deployed to the Foundation Node, it performs as much of the query as it can with the data on the node before having to join, aggregate, or compare it to other data. It returns that result up to the SQL Node for further processing, packaging, and returning to the user.

Foundation Nodes contain the majority of the storage in an Ociant System and are also typically the largest in number.

### Loader Nodes

Loader Nodes are responsible for extracting, transforming, indexing, and loading data ingested by Ociant from batch file sources as well as streaming data sources such as Apache® Kafka®. A user can specify the extraction source details and supply a transformation pipeline that manipulates the structured or semi-structured source data before loading it into relational form into Ociant tables.

Loader Nodes operate in a horizontal scale-out fashion allowing the loading system to scale to fit various ingestion requirements. Transparently to the end user, the Loader Node also manages exactly-once guarantees with data as it is loaded and converts pages into segments in the Foundation Nodes.

### Flows and Networking Between Nodes

Data and communication flows across the nodes of an Ociant System for a variety of different purposes. There are two networks that connect nodes, a 100 Gbps high speed network and a 10 Gbps network. These are used in

different ways for data flows and administrative purposes.

## A Query in Ocient

When a user issues a query against the database, their SQL statement moves from the JDBC client to the SQL Node. The SQL Node parses and optimizes the query before handing the plan off to the Foundation Nodes. The Foundation Nodes do the first level of processing against the data before handing the results back to the SQL Node. The SQL Node further processes the data, constructs the results, and returns them to the client.

- The connection between the SQL Nodes and Foundation Nodes is typically a 100 Gbps network connection. The speed of the connection between the client and the SQL Node is subject to the speed of that network (outside Ocient).
- Every Foundation Node is connected to every SQL Node.

## Administration Flows

When administrators configure or make changes to an Ocient System, they do it using DDL and DCL using a SQL client. The SQL client relays the statement to a SQL Node, which parses them into commands that the Administrator Role on the SQL Node executes. This can impact the SQL Nodes, Foundation Nodes, or Loader Nodes, depending on the type of change or operation.

- The administration flow connections with the Ocient System are 10 Gbps connections.
- Every node is connected to the SQL Nodes that perform the Administrator Role.

## Loading Flow

When the load job is defined and started, the Loader Nodes read data from the source files, stages the data, processes that data, and moves the data into the Foundation Nodes. The Loading and Transformation process is responsible for managing this process and ensuring data is properly indexed and loaded.

- The network connections between the Loader and the Foundation Nodes are 100 Gbps.
- The Loader Nodes connect to every Foundation Node.

## Database, Tables, and Views

Similar to other databases, an Ocient Hyperscale Data Warehouse™ is a grouping of tables, views, users, and data within one Ocient System. As mentioned above, a single system can have multiple databases. Tables and views are defined within each database. While Ocient stores data on disk in columnar format, creating and working with tables in Ocient is just like any other relational database. Ocient supports a wide variety of standard SQL data types along with an expanding set of geospatial data types.

Read more about:

- [Data Definition Language \(DDL\) Statement Reference](#)
- [General SQL Syntax](#)
- [Functions Overview](#)

## TimeKey and Clustering Key

Each table in Ocient can contain a TimeKey® column that is specified when defining the table. The TimeKey column is used to partition the data within the Foundation Nodes. Time partitioning is an important performance mechanism used by the Ocient Hyperscale Data Warehouse. Because many, if not most, queries specify some time filter for the results, time partitioning allows Ocient to quickly skip data from irrelevant times.

When defining a table, users can also specify a clustering key containing one or more of the columns to further subdivide records on disk. This allows the database to quickly find records with the same key values.

Read more in [TimeKeys and Clustering Keys](#).

## Resilience to Hardware Failure

In order to provide reliability in the case of a hardware outage, Ocient uses erasure coding. Erasure coding is a mechanism to organize and compute parity blocks so that the system can rebuild missing data. This means that Ocient does not store redundant copies of the data for reliability. As a result, an Ocient System requires less overall storage than if it were storing one or more copies of the data for failover.

When a user defines their storage configuration, they specify the width of the group and the parity width. These values can vary based on how many Foundation Nodes are in the system and how resilient users want to make the Ocient System.

## Segments and Segment Groups

Segments in Ocient are storage units that contain rows and are divided into a fixed number of coding blocks with a defined size. Segments are typically sized on the order of gigabytes. You set the segment size when you define tables in Ocient. The size of the segment must be a multiple of the coding block size. The coding block is the unit of parity calculation and the smallest unit of recovery as well.

Multiple segments combine to form Segment Groups. A Segment Group has a fixed number of segments, named the width, which is the number of segments in the group. It also has a pre-defined number of parity blocks per set of data blocks for resiliency. Each segment has a defined index in the group. The Segment Group is physically stored in a storage cluster. Segment Groups can also be nested in directories named Segment Directory Groups.

When the user configures an Ocient System, they need to define at least one storage space and storage cluster. A storage cluster is a set of Foundation Nodes with an associated storage space. Also, these nodes coordinate together to store segment groups in a reliable manner. The storage cluster also has a width in the number of nodes. The width of the cluster and the Segment Groups stored on it must be the same. At the storage space level, the user defines the width and parity width of the space. Segments and their Segment Groups are stored across these storage spaces and clusters. To learn more about erasure coding, see [Compute-Adjacent Input and Output on Large Working Sets](#).

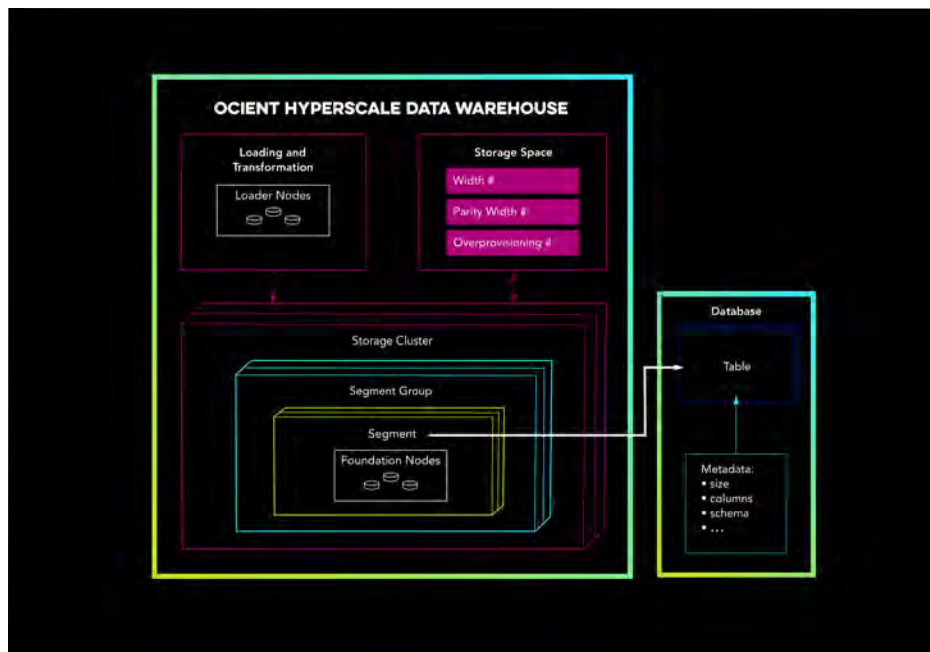


Figure 20: Ocient Hyperscale Data Warehouse contains Foundation Nodes in segments grouped into segment groups that are grouped into storage clusters. Tables with metadata are stored in segments.

For details, see [Configure Storage Spaces](#).

## Related Links

- [Ocient Architecture](#)
- [Compute-Adjacent Input and Output on Large Working Sets](#)
- [Key Concepts](#)

## Related Videos

- [At the Whiteboard with Ocient: Deployment Options](#)
- [At the Whiteboard with Ocient: Data Reliability](#)

## 9.2 Compute-Adjacent Input and Output on Large Working Sets

The storage of the data warehouse and I/O layer is built using a Compute Adjacent Storage Architecture® (CASA). When contrasted against modern cloud-based database engines, this means that the majority of the data to be analyzed is stored on NVMe SSDs directly attached to a significant fraction of the computational capacity of the system using the local PCIe busses of the nodes that make up the *foundation level* of the database engine.

This approach, which is similar to a traditional MPP architecture, is designed to capitalize on the performance characteristics of NVMe SSDs when executing queries against a large *working set* of data. It is important to note that when executing a query, a specific record could be included as part of the results but I/O and computational effort must be undertaken to make that determination. A working set, therefore, is defined as the set of records stored in the database that are considered as part of query execution, even if those records are eventually filtered out or not returned. A “large” working set is one that is substantially larger, even by multiple orders of magnitude, than the available DRAM on the set of nodes performing the computations. When this is the case, *remote-object* database engines that store the majority of their data in a separate storage layer are substantially affected by their inability to keep all the potentially-addressed records near and accessible to computation at the same time. Depending on the query load and their exact engine design and scheduling, query execution times negatively suffer from both simple bottlenecks due to network link capacity and “cache thrashing” that produces expensive extra drive I/O.

Large working sets can be realized in two major ways:

1. Queries that regularly consider a substantial fraction of the entire data set. For example, regular querying or analysis of all data spanning years can mean a single query addresses most of the records in the database.
2. A large number of smaller queries that all consider a disjoint subset of the whole data set. When these queries are executed in parallel the sum total of the disjoint sets of addressed records can also approach most of the records in the database.

The database engine in the Ocient® System is a world-class relational database engine that can go toe-to-toe with other modern engines on small working sets but won’t necessarily out-perform them. After all, well known algorithms like sorting and hashing can only be made so fast when everything is in memory. But Ocient’s performance distinctions derive from its ability to scale CASA to keep all of the data close to the computation.

### NVMe SSDs in CASA

One of the biggest drivers for the development of the Ocient Hyperscale Data Warehouse™ was the ongoing shift from spinning HDDs to NVMe SSDs for data storage. SSD prices continue to fall, relative to HDDs, and there remains little doubt that they will eventually supplant HDDs for all but esoteric or extreme bulk storage applications.

To ride this storage wave, the Ocient engine has been built to minimize the cost of using these drives as the primary storage tier of the engine while maximizing their performance. There are two major aspects to cost management:

1. *Minimization of the total number of stored bytes needed to achieve reliable access to data in a performant manner.* This is achieved using a novel and tunable *erasure coding* scheme in conjunction with a variety of compression techniques. This is detailed more thoroughly in the following sections.
2. *Consideration for endurance management in I/O operations against the NVMe SSDs.* The Ocient storage and execution engine is designed to minimize the number of I/O write operations undertaken against the storage media in order to maximize its lifetime and reduce drive replacement rate due to endurance wear. At a high level, this usually means minimizing small or random writes to the drives and instead favoring larger and

sequential ones. To achieve this, the storage layer uses a specially-written and database-aware file system underneath storage structures designed to capitalize on the large volume of low-mutability data being stored. Using these mechanisms, the storage layer is able to collate writes in a manner that produces relatively large (from the drive's point of view) ranges of contiguous blocks, ordered in the hundreds of KiB to hundreds of MiB in size.

## Erasure Coding

Of particular note in the storage and I/O layer of the Ocident Hyperscale Data Warehouse is the novel use of a tunable erasure coding scheme for data storage.

You can think of erasure coding as a generalization of simple XOR parity calculations. In a simple error correction scheme, you can apply the XOR operation on a sequence of bits:  $B_0, B_1, B_2, \dots, B_n$  such that:  $B_0 \text{ XOR } B_1 \text{ XOR } \dots \text{ XOR } B_n = B_x$ . An easily provable property of the XOR operation is that for any single  $B_i$  with  $0 \leq i \leq n$ , that  $B_0 \text{ XOR } \dots \text{ XOR } B_{i-1} \text{ XOR } B_{i+1} \text{ XOR } \dots \text{ XOR } B_n \text{ XOR } B_x = B_i$ . You can determine any single bit by performing the XOR operation on the other original bits and the extra bit together. In the context of error correction, this means that for the cost of an extra bit of storage (or memory) a system is able to detect, and even correct for, the loss of a single bit.

There are a variety of error correction schemes described in computer science and found in the industry. In the application of this concept to database storage, the Ocident storage layer extends it from individual bits, and beyond bytes, to an error correction unit of the 4KiB block (which is notably aligned with the I/O resolution of standard NVMe SSD drives). It also further extends from enabling single loss or failure to a fully tunable approach that uses **XOR**, **P+Q**, or **Cauchy-Reed-Solomon** coding that allows the user to specify exactly how many faults: one, two, or  $K$ , respectively. Like the simple XOR formulation, these schemes utilize mathematical computations (in this case, polynomials over a finite/Galois field) to compute additional blocks of data that enable reconstructing a configurable number of missing blocks.

In database and storage systems that utilize replication to provide data reliability and availability, it is common to describe storage in terms of the number of replicas (e.g., 3x replication allows for up to three failures). The obvious implication of replication is that each replica is a full copy of the original data and represents a multiple of the minimal data storage required. In an erasure coding scheme, the overhead for  $K$  tolerable faults is calculated as  $(N + K)/N$ , where  $N$  is the total width and  $K$  is parity width. If  $N$  is 10 (take that as a given for the moment) then to support a maximum of 2 faults (the equivalent of 3x replication, which includes the original data and two copies) the system needs only  $12 / 10 = 1.2x$  times the original storage size, contrasted with the 3x of the replication scheme.

The utilization of space-efficient erasure coding can be found in a variety of storage and transmission systems but, in a classic space-time trade off, the space savings come at a computational cost. The approach taken in the Ocident storage layer is novel, at least for databases, and is designed to mitigate the computational overhead of erasure coding.

If one considers a “file” of some size as a unit of storage, a basic approach for applying an erasure coding scheme to provide loss tolerance to this file would be to break it into  $N$  chunks and perform the requisite computation to produce an additional  $K$  chunks. At read time, the system would select an available subset of the  $N+K$  chunks (possibly preferring the original  $N$ ) and perform the **rebuild** computation to produce the original file for access. In this framing the unit of fault tolerance is the individual file, and rebuilding computational overhead can be found in the read path.

However, in a database designed for OLAP-style queries, the potential for the overhead of any rebuilding computation in the read path even when no drives or nodes are offline is troublesome. Additionally, when the system is in a degraded state, the potential need to rebuild an entire file only to retrieve a small subset of its data is expensive as well. The Ocident storage layer, by directly implementing the erasure coding scheme and integrating with it, avoids these issues in two major ways.

The first advantage is the unit of storage. Ocident uses storage units named **segments**, which are not chunked into  $N$  pieces. Instead,  $N$  separate and similarly sized segments are considered together as a unit named a **segment group**. The erasure coding scheme is then applied across the segments to produce  $K$  additional “segments” worth of redundancy information which, in a very loose application of the term, is named **parity data**.

The newly generated parity data is then broken up and evenly distributed amongst the original segments (in a particular and calculated way) such that each segment is now approximately  $(N+K)/N$  times larger than it originally was, and any missing segment from a particular segment group can be reconstructed when one has access to any set of  $N$  segments from the same group. The Ocident storage layer ensures that each segment from a group is stored on a separate physical node, thereby providing data availability in the event of up to  $K$  node losses.

Another important feature of this scheme is due to the fact that a given record is always stored in a single segment. Because the segments themselves are not chunked, any given record is guaranteed to exist intact on some node/drive, and readable with no overhead incurred when that drive is online. Put another way, in the non-degraded state, the reliability scheme implemented incurs no computational cost.

The second advantage of the Ocident System is each segment acts as a self-contained unit of information, containing not only database records but also metadata and indexes on those records. These sub-units within the segment, named *segment parts* are individually addressable and independently rebuildable with 4KiB resolution. The implication of this is that in a degraded state (where a node or drive is missing), it is not strictly necessary to rebuild all missing segments to service a specified query. Instead, the storage layer is capable of addressing and rebuilding only the parts and even single blocks of parts required for it. This often greatly reduces the overhead and performance impact of a degraded state.

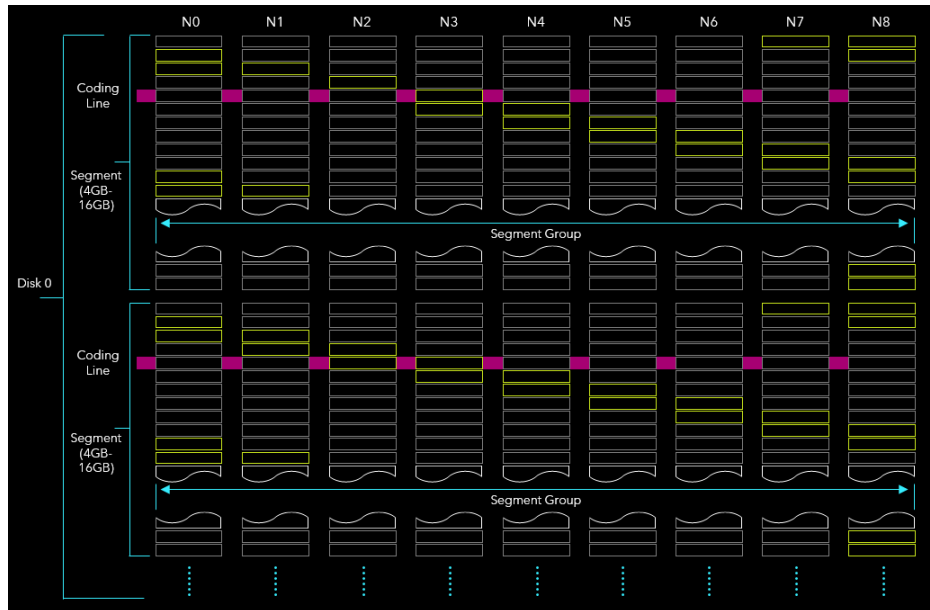


Figure 21: Erasure coding example where  $N = 7$  and  $K = 2$

### Choice of “N”

Earlier, it was taken as a given that  $N$  is equal to 10. The exact choice of this value is somewhat arbitrary but is generally bounded by system-level consensus constraints as well as the calculated cost of the network transit during rebuild operations. As  $N$  increases, the storage overhead for the data reliability for some specified  $K$  *decreases* because  $(N+K)/N$  approaches 1. However, the total amount of information that must be considered and moved over the network for, as well as the actual computational cost of, a rebuild *increases*. Real-world usage has found that an  $N$  somewhere between 8 and 12 provides a good tradeoff between these two opposing considerations.

### NVMe Userspace Optimizations

A final important aspect of the Ocident storage and I/O layer is the database engine’s direct interaction with the SSDs storing segments. The term “direct” in this context can be contrasted against how a more standard approach might look. In many database systems, record storage is achieved using a set of files stored on a disk in some file system (such as EXT4 or NTFS or ZFS). And generally speaking, reading and writing of those files will be performed using kernel-provided system calls/services like `read()`, `write()`, and `mmap()`. Obviously this is a fine

approach in general, but when performance is of utmost importance, the overhead of the system calls, memory copies to and from kernel memory, as well as the implementation of the underlying file systems all have detrimental, and sometimes unpredictable, effects. For example, it is well known that the Linux® kernel's EXT4 implementation takes global locks during certain metadata operations which can have a significant detrimental effect on parallelism and throughput.

Some databases eschew standard files and perform some form of I/O directly against kernel block devices in order to avoid file system overhead. This is surely an improvement, but the kernel's block scheduler, extra memory copies to/from kernel memory, and the overhead of system calls still exist.

In order to overcome these performance impacts, the Ocient storage layer directly interacts with the NVMe SSDs that store segments in two important ways:

1. Each NVMe drive is decoupled from the standard Linux block device driver and instead attached to a device driver that enables low-level interaction with the device using the PCIe protocol. The implications of this approach are that the storage layer and database engine can directly communicate the NVMe protocol to the devices *and* do so using user memory. This completely eliminates all system calls and memory copies required to do I/O with the NVMe drives. The database provides direct pointers to its own virtual memory for I/O operations and the drives directly write and read to and from that memory.

An interesting result from this approach is that the Ocient storage layer and engine were entirely unaffected by the Linux kernel's mitigations for the Spectre and Meltdown classes of attacks. This is because the primary mechanism of the mitigation resulted in a substantial increase in the overhead of the user-to-kernel transition associated with system calls. Other database engines with more standard approaches saw in many cases appreciable performance degradation.\*

2. Modern NVMe SSDs are capable of a million 4KiB random reads per second. This is strikingly and substantially more than HDDs (where even the best drives can produce only hundreds). However, for most economical SSDs, these rates can only be achieved when large numbers of parallel in-flight requests are maintained over a substantial fraction of the drive's address space. The Ocient storage layer implementation, in conjunction with specially tailored data and metadata structures, not only promote this parallelism to be evoked during query execution, but also are written to minimize "latency bubbles" by always having a replacement I/O operation teed up for a drive when it completes a previous one.

## Columnar Compression

As a column-oriented data warehouse, Ocient is able to leverage per-column data locality and similarity to achieve excellent data compression.

- For fixed-length columns, a combination of delta-delta compression, run-length encoding, and null elimination provide significant storage reduction for many datatypes.
- For variable-length columns, the Ocient System can apply per-row compression to reduce the size of large values.
- Both fixed and variable-length columns can benefit from whole-column compression using a shared dictionary trained on a per-column and per-segment basis.

Column compression works best when combined with secondary indexes on frequently filtered columns.

## Related Links

[Secondary Indexes](#)

[Key Concepts](#)

## Related Videos

At the Whiteboard with Ocient: Compute Adjacent Storage Architecture™



## 9.3 Query Performance Optimizations

### Specialized Indexes

The storage engine, execution engine, and optimizer are all aware of two special concepts that can bring about significant query performance benefits. Neither of these specializations are strictly required in any table or query, but the Ocident® engine capitalizes on the fact that for the datasets with trillions or quadrillions of records they are often advantageous.

### Time Keys

Much of the data in hyperscale databases is produced by machines, devices, loggers, and sensors. The records these entities emit often have a temporal nature: they are timestamped or otherwise sequenced relative to each other. At query time, it is often the case that queries themselves have temporal predicates (e.g., “select all records for this user *for this year*”) as well.

When these two conditions are met, the storage layer, execution engine, optimizer, and loading infrastructure all have special handling built-in for more efficiently processing these records. The time-based processing of records shares similarities with zone maps in that they can be used to avoid I/O for non-intersecting sets of data when predicates against certain columns are present in a query. However, the specific temporal nature of the data is further utilized to make loading more efficient and also impacts segment grouping and data roll off.

### Clustering Keys

For many data sets there is often a commonly queried set of columns that form the basis for a variety of analytical tasks. For example, in a network security database recording metadata of IP traffic, it might be the case that a substantial fraction of interesting queries are concerned with the public destination IP address, represented as a column in a record, in addition to other constraints or filters.

When this is true for a subset of the columns in a given table, a **clustering key** can be defined, which enables additional specialized handling across the storage layer, execution engine, optimizer, and loading infrastructure. A clustering key shares some similarities with the “partitioning key” concept found in other databases in that it is defined in terms of one or more columns in the table and also factors into locality decisions for storage. However, unlike most systems, the clustering key *is not* a strict partitioning, and is instead only a “hint” that promotes locality for storage to improve I/O efficiency for queries that locate records using it. Because it is only a (usually followed) hint, the storage layer and engine do not suffer from issues related to partitions outsizeing others or forced locating of segments on particular nodes.

### Global Dictionary Compression

Global Dictionary Compression, or **GDC**, benefits both storage utilization and query throughput for columns with low cardinality ( $\leq \sim 1$  million distinct values). When enabled for a given column, that column’s values are mapped to a succinct global enumeration that is stored on disk and used at query time. GDC is often enabled for variable-length columns representing an enumerated set of potential values (e.g., city, county, and country names, or classification tags). In these scenarios it provides significant compression and also enables a number of optimizations in query and join processing. In this latter category, the optimizer is able to delay or even eliminate the decompression step during execution because there exists a single global key-to-value mapping. This often results in a significant reduction in string data processing as well as network and memory bandwidth.

Like other forms of compression, GDC is completely transparent to end users. New values are added to the global enumeration automatically, and compressed values are mapped back to their original value in query results.

### Specialized Types

#### Complex Types: Tuples, Arrays, Matrixes

In many analytical workloads, semi-structured and multi-dimensional data must be analyzed. Ocident offers complex data types to efficiently store and analyze this type of information, in many cases dramatically reducing query response times when compared to normalized schema designs. Each complex type allows groupings of elements to be



stored and analyzed in a denormalized fashion with the parent record, and combinations of these complex types might also be used. This avoids the repetitive joins across billions of records required for nested data in a normalized schema improving query response times. Finally, Ociant's indexing strategies extend to complex types as described here.

The complex types supported in Ociant include Arrays, Matrixes, and Tuples. An array is an ordered list of elements. The element can have any data type including a tuple. Array types provide operators for efficient set operations and standard SQL semantics like `SOME` and `ALL`. Matrixes are multi-dimensional arrays that share many of the same benefits as Arrays when denormalizing schemas or working with matrix data for mathematical operations.

Tuples are structured objects in a single column created from individual data elements of the same or different underlying types. Arrays of Tuples provide the ability to denormalize structured elements that can then be jointly queried and indexed for improved performance in some cases. Because the array of elements can be stored in a columnar fashion and associated with the parent row, they need not be joined to the parent table repeatedly to query them. Performance is improved and working set memory consumption is reduced on queries that would otherwise need to join tens of billions of rows to potentially trillions of rows. The flexibility of these composable complex types creates more options for database administrators to structure data and optimize performance at scale.

## Secondary Indexes

A variety of secondary index types are used to improve query performance. Secondary indexes are built and searched on a per-segment basis, allowing different segments of a given table to be indexed differently where appropriate.

For fixed-length column types, an I/O-optimized inverted secondary index enables fast equality and range queries. Variable-length columns support hash and N-gram indexes that enable fast equality and `LIKE` filters, respectively.

Complex types like tuples and matrixes are a concise way to model certain datasets. In order to support efficient filtering on these types and their sub-components, filters are pushed into the I/O layer and use secondary indexes when possible.

You can create indexes on arrays of both fixed and variable-length elements to optimize contains, overlaps, and other similar operations. Additionally, `for_all()/for_some()` syntax allows other filters to be pushed down and applied to arrays using an index, without requiring `unnest`. In particular, `for_all/for_some(varchar_array_col) LIKE <filter>` utilizes N-gram indexes to optimize textual queries over arrays of strings.

## Per-Segment I/O Pipelines

The I/O layer is designed to minimize the amount of on-disk data read and processed for each query by compiling a custom *I/O pipeline* for each segment of data the query accesses. An I/O pipeline is a directed acyclic graph that represents an efficient I/O plan for the particular columns and filters of that query combined with indexes and data that can vary from segment to segment. This plan adjusts for data demographic changes over time, and for schema or index changes also. Just like the query plan itself, efficient pipelines are critical for performance.

A simple I/O pipeline example is illustrated here:

This graphic example shows the I/O process stages that go into action when the Ociant System compiles a pipeline. These stages are as follows:

1. When a query is in the IO pipeline, each segment first searches indexes to minimize the data that must be read and processed.
2. The indexes help to quickly filter the rows. When possible, filter predicates are pushed down and evaluated in-line, further reducing the number of rows needed to read even when an appropriate index does not exist.
3. The system reads rows that pass the filters in parallel.
4. The system returns the query output.

During execution, *I/O pipeline elements* work together to search indexes, read column data, and apply filters to return the exact set of matching rows from each segment. Much like CPU instruction pipelining, the I/O pipeline attempts to mitigate latency while fully utilizing fast storage devices requiring deep I/O queues. The graph for a sophisticated query can orchestrate dozens of pipeline steps in series and in parallel. This layer works to keep disks saturated by coordinating I/O across segments with read-ahead and a shared priority queue.

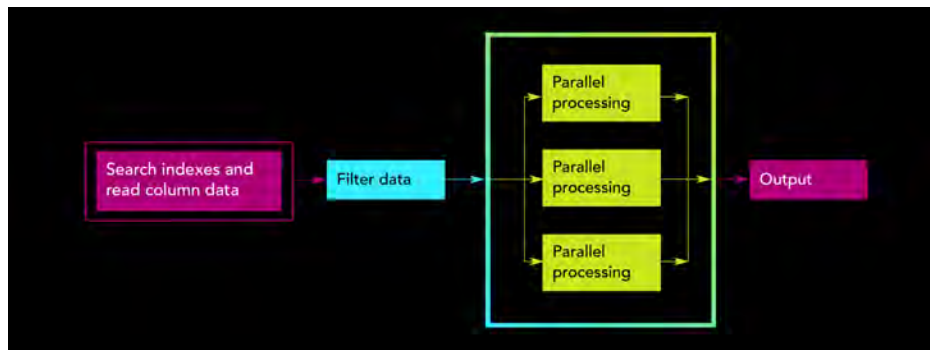


Figure 22: Simple I/O pipeline example that contains a pipeline and its elements

A good I/O pipeline determines the set of cluster key and secondary indexes to use, and an optimal ordering based on index selectivity to filter rows at maximum speed. When possible, filter predicates are pushed down and evaluated in-line, enabling optimizations even when an appropriate index does not exist.

### Optimizing I/O with Bloom Filter and Index-Based Join Push-Down

A pair of advanced techniques are employed to dynamically reduce the amount of I/O needed when processing certain types of joins.

- **Index-based join push-down** — As a running query narrows the collection of potentially matching values, an updated list of filters can be dynamically pushed down into running I/O pipelines. When a matching index is present, the I/O layer can begin using it mid-query to reduce I/O. Otherwise, this technique has the same benefits of early filtering as a pushed-down Bloom filter.
- **Bloom filter push-down** — When one side of a join is fully determined, the execution engine can build a Bloom filter that gets pushed down into running I/O pipelines to enable early filtering of unneeded values at the lowest level. As a probabilistic data structure, Bloom filters can be tuned to provide filtering benefits for even high-cardinality sets.

Both of these techniques reduce I/O on dependent columns, memory and storage bandwidth for unnecessary rows, and unneeded join processing.

## Query Execution

### Parallelism

The storage layer and I/O access patterns to promote parallelism would be of limited impact if the data processing working on the actual data was not itself highly parallel. To that end, the execution engine (which is named the **Virtual Machine or VM**) is designed to maintain parallel and disjoint flow of data during data processing.

A query **plan** is a (mostly) directed graph of **plan operators** in a tree-like arrangement. At the leaves of the tree are the raw I/O operations undertaken to bring records off of SSDs and into memory for processing. Internally, iterative processing, computation, filtering, and joining is undertaken by successive operators to produce the final results for the query. The execution engine is named the VM because it structures plan operators as assembly instructions (such as ADD and MOV) to be scheduled and executed.

The fundamental organizational unit of this execution engine follows a design pattern named **resource-aligned parallelism** where a series of independent execution units, strongly coupled with hardware resources like cores, memory, and disks, disjointly process independent **data partitions** of records that flow independently and in parallel within the engine. Because minimal to no data sharing is required across data partitions, the execution engine not only maximizes resource utilization and throughput, but also scales very well as CPUs continue to include larger numbers of cores.

The number of data partitions across any edge within the plan tree is usually, but not always, equal to the number of distinct **VM cores**. Each VM core is a dedicated and isolated CPU core that does not process any other database

or system threads (this is achieved using a Linux® kernel feature named “cpu isolation”). Then, each plan operator is locally “compiled” into a number of **operator instances** that is equal to the local number of VM cores. In this way, any individual operation within a plan can be spread over all allocated computational units to independently process the parallel disjoint subsets of records it is to handle.

Of particular note is the attention paid to data skew. In order to maximize aggregate throughput across all parallel operator instances it is required that they perform approximately the same amount of work. If work is not evenly distributed across all parallel processors, then query execution is limited to the slowest one while the others have no work. Even partitioning of data is easy to achieve in some cases, and impossible in others (it depends on the operation and the data), but the Ocient engine undertakes a variety of special efforts to evenly balance data flow, even in non-trivial cases.

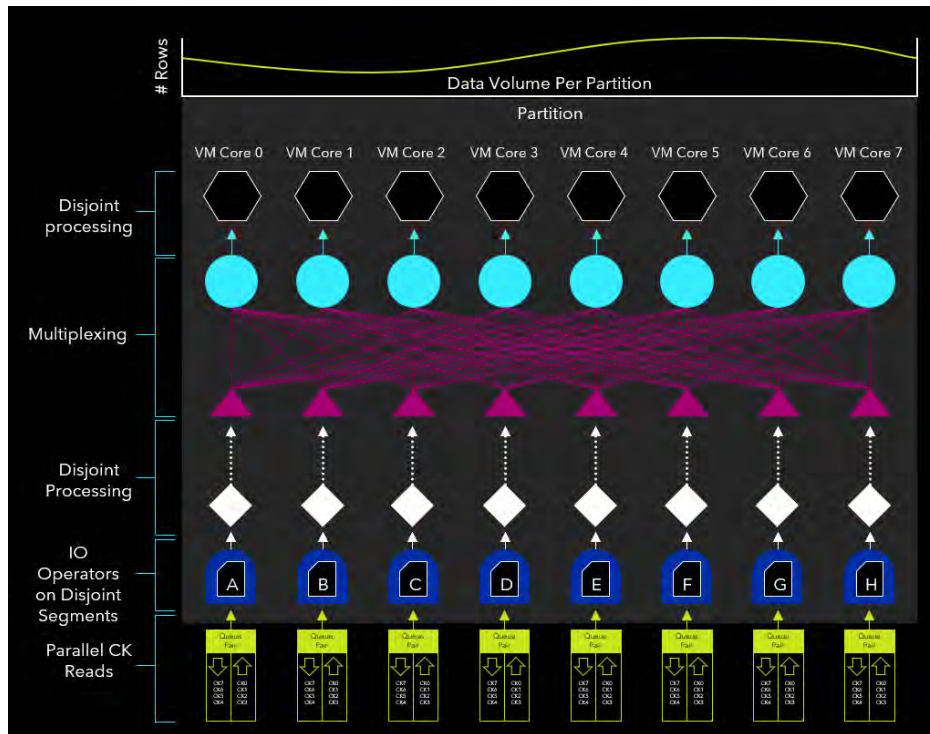


Figure 23: Parallel VM core operation with balanced data flow

An interesting example of this is found in sort execution. If one were given a set of integers to sort, a straightforward approach to parallelize the sort operation would be to split the set into a sequence of disjoint subsets  $S_0, S_1, S_2, \dots, S_n$  where for all  $n$ , the integers in  $S_n$  are all less than the integers in  $S_{n+1}$ . Then, one could sort each subset, in parallel, to end up with a totally ordered sequence of integers across all sub-sequences. Put another way, the system splits and groups the integers at boundary values on the number line, then sorts each interval independently. The problem of data skew arises when the number of rows per interval (or subset) is not roughly equivalent, with the worst case being where one interval has all the integers and the others all have none. The problem is one of boundary selection: any naive choice can be subject to skew. To address this in the sort case, the database engine, in conjunction with information provided by the optimizer based on metadata and statistics, *adjusts* the boundaries for the ranges such that each boundary has a roughly equivalent number of rows.

### Implementation Efficiency

When discussing parallelism, you might have noticed several terms that sound very operating-system like: threads, disjoint cores, compilation, and so on. In fact, the database engine directly implements many general pieces of functionality that are usually in the domain of the operating system, but with database-particular features that outperform standard system-provided ones.

1. *Specialized hugepage-backed memory allocators.* Designed for high throughput and low lock contention, these

allocators are backed by hugepage memory, not only deriving the benefits of fewer TLB misses, but also making them suitable for userspace I/O operations because their virtual to physical address mapping is fixed. Special efforts were made to make them extremely fast and efficient within the resource-aligned parallelism pattern.

2. *Cooperative per-VM-core Scheduler.* This component runs per VM core, allocating CPU time to operator instances on that core to process data. In this framing, each operator instance can be considered as a parallel thread on its core. However, the scheduler is not a generic thread scheduler, but instead has access and visibility into inter-operator data flow, concurrent query memory utilization, and workload management limits. This means that it has a much richer set of information with which to make intelligent scheduling decisions that promote maximal total system throughput.
3. *NUMA-aware Data Processing.* Modern multi-socket systems utilize a non-uniform memory architecture (NUMA), whereby individual CPUs (and the cores in them) have both local memory and remote memory. The implication of this is that a particular memory access made against local memory by code executing on a core is not only lower latency than a remote one, but also avoids an expensive hop across a dedicated memory bus with a finite amount of bandwidth. Because most commodity CPU packages directly integrate a PCIe bus as well, it is the case that not only is there local and remote memory, but local and remote devices as well. To account for this, the VM groups locally-connected RAM, disks, and cores into a functional unit named a *silo* that for all practical purposes is treated as an isolated and independent processing unit. Generally speaking, records for a particular disk will be read only into local memory, and only processed on local cores, all within that silo. This avoids expensive cross-socket communication and greatly improves throughput.

## Related Links

[Secondary Indexes](#)

[Understanding Data Types](#)

## Related Videos

At the Whiteboard with Ocident: SQL at Scale

## 9.4 Plan Optimization

A foundational element of any high-performance database is its plan optimization engine. Recall that a *query plan* is in many ways similar to a generic computer program. A key distinction, however, is that its machine code is in the language of relational algebra, set operations, I/O, network, and other database primitives. When a database converts SQL queries into a query plan, it must always generate a plan that produces the correct query results.

An over-used joke at Ocident® is that if you could return incorrect results, you could make a blazingly fast database by simply returning random numbers for every query. By way of analogy, this correct result requirement is the same as saying that a C++ compiler must always produce x86 machine code that correctly executes the originally provided instructions. But just as a given C++ program can have many different possible representations in x86 machine code (different compilers, optimizations, etc) it is also the case that a given SQL query can have many (infinitely, actually) different query plans.

In a database, the optimizer's job is to take an already-correct plan that is naively generated from a SQL statement and produce an alternative plan that will emit the same results. This is normally viewed as a minimization problem where the alternative plan is "less than" the original plan in some metric (usually estimated run time) and database engines have evolved extremely varied approaches to solving it. Some challenges to consider:

1. The search space for the minimization problem is infinite. This means that any algorithm must eventually converge to some answer but do so in a reasonable amount of time. It should not spend 100 seconds to shave 1 second off of a 2 second query.
2. Plans are compared to each other based on some metric like execution time, but this metric is not trivial to build. Having a metric function  $M(p)$  that returns a useful number and is cheap enough to compute is a challenge in and of itself. Adding to this challenge is the additional considerations beyond measuring the computational cost of plan execution, namely disk and network I/O costs as well.

For these challenges, most database engines incorporate some sort of statistical approach. Statistics on the tables being queried (calculated either at data load time, or periodically updated) are used to bound execution and I/O estimates. The optimizer of the Ocient System does this as well. While basic statistics like row counts are certainly considered, more advanced mathematical approaches are used as well:

1. *Multi-dimensional probability density functions.* If a table has  $j$  columns, then it can make use of a  $j$  dimensional “table space” function  $f(v_0, v_1, v_2, \dots, v_j)$  that maps a particular set of column values to the probability of those values. The Ocient System can then compute approximate row counts in the table by taking integrals of  $f$  in some hyper region of the table space.
2. *HyperLogLog.* In addition to row counts, the ability to estimate the number of *distinct* values is important as well. The basic idea behind HyperLogLog is that if a set has a uniformly randomized set of values, and  $n$  is the maximum number of leading zeros in the binary representations of all values in the set, then a good estimate of the set’s cardinality is  $2^n$ . Here, each column is considered a set, and this statistical approach can estimate the number of distinct values for each column.

With such mathematical machinery available, you can see how the beginnings of a metric function for a plan can be built, but a few important details remain. Firstly, while the statistical approaches can make for good estimates, the Ocient System must be able to transform from the domain of SQL types into numbers for each supported SQL type. This is sometimes straightforward in cases like integral and floating point types, but becomes arbitrary and less obvious with things like variable length strings and arrays. Secondly, and more importantly, Ocient must also be able to calculate the metric on *sequences* of operators, each of which can be considered as a separate transformation of input to output. Put another way, if given a sequence of plan operators where each operator is successively operating on rows  $T \rightarrow P_0 \rightarrow P_1 \rightarrow P_2 \dots$ , then the raw column cardinality estimates of the columns in  $T$  might be of limited use in calculating a metric on  $P_2$  because  $P_0$  and  $P_1$  can drastically alter the statistics of the original table.

In order to account for this, the optimizer must have mathematical models of each operator that can be applied to *input statistics* to produce *output statistics*. Then, successive application of operators can be used to build up plan metrics comprised of per-operator metrics. Clearly, successive approximation incurs increasing error. But an interesting property of using a metric in this manner is that the exact error is not important. Instead, all that is necessary is that the accumulated errors in any estimation must not invalidate the relative comparison between two plans.

The topic of plan optimization is large and complicated, and the Ocient optimizer is a key and integral component of its approach. It incorporates not only detailed implementations but also an integrated view of the distributed nature of the query execution against NVMe drives. Each iterative version of the optimizer also includes an ever-increasing set of optimizations, giving it a sophisticated tool chest of transforms to apply to plans as well.

## Related Links

[Query Ocient](#)

[Key Concepts](#)

## 9.5 Exabyte Scalability Design

### Cluster of Clusters

Various system components in Ocient® have each been designed to scale not only with hardware capabilities as technology progresses, but also in number of nodes storing and processing data. Like any distributed system, issues of fault tolerance, availability, and communication all become increasingly challenging as the number of units increases.

The Ocient scalability approach can be broadly described as deriving from **clusters of clusters**: smaller, locally connected clusters that manage their own availability and local state independently and without consideration to any other clusters in the system. The design goal of this is to minimize global state and global synchronization in any aspect of query processing, fault management, data loading, or data ownership.

There are different kinds of clusters: **metadata** clusters, **storage** clusters, **VM** clusters, and **loading** clusters. An Ocient deployment can contain zero or more of each. With respect to scalability, each cluster runs its own Raft-based consensus protocol and maintains its own local shared state with it. Individual nodes can belong to multiple clusters

and participate in multiple disjoint consensus protocols at the same time, but system scalability derives from the fact that no node has any global view of system state, and indeed, no such global state exists.

This allows Ocident to quantify the upper bound on the  $N$  parameter described in the erasure coding section. Recall that  $N$  is the number of segments in a segment group. In a **storage cluster**, segment ownership and availability is confined to and maintained in the cluster-local consensus protocol. A necessary implication of data management rules for atomic (but still local) state changes is that *a particular segment group must be stored entirely within one storage cluster*. And because the Ocident System cannot place two segments from the same group onto the same node, the value  $N$  cannot exceed the number of nodes in a target storage cluster. This finally means that the upper bound on  $N$  is the practical limit on the consensus protocol implementation, coupled with the design goal of not allowing clusters to be “too big”: somewhere in the range of 20 - 40, depending on deployment and schema details.

### Query Trees (and Plan Trees)

Query execution necessarily spans multiple VM and storage clusters. A key phase of query processing, prior to any actual computation is the **query probe**, which establishes the **query tree** to be used for that particular query. This tree fixes, for the particular execution with which it is associated, the set of nodes that provide which segments across *all* storage clusters, as well as the set of nodes that co-participate in the distributed execution of the query. The former is important because as nodes and drives come and go, and as new data is loaded and old data is removed, it is imperative that all segments that should be included in a query are actually included exactly one time. The latter is important because each node might need to know exactly which subset of peers in the network are performing which aspects of query execution. In both cases, the probe phase enables query execution to proceed without any global coordination or synchronization.

This graphic shows a query execution tree, which can scale up by increasing the number of clusters (represented horizontally in the graphic).



Figure 24: Example query execution tree that shows four levels of multiple clusters

At the top level of the query execution tree is the root operator, which has zero or more additional plan levels below



it, depending on the complexity of the query. Each circle represents individual nodes, which provide segments across the storage clusters or specific tasks for the execution of the query.

Vertically, the graphic shows different plan levels that fan out among the nodes for the stages of the query plan, actions such as transformations, sorting, grouping or other operations. The leaf level at the bottom of the graphic represents the I/O operation that reads the queried data from storage.

When the system initiates query execution, each cluster executes its operations in parallel. For the most part, clusters operate in isolation unless they need to communicate for operations such as a join or shuffle.

When referring to the query plan, each level is delineated by the presence of a special **gather operator** that represents the network boundary between levels. All the plan operators between root and the first gather operator are said to *execute at level 1*. Any plan operators that exist between the first gather operator and the second gather operator execute at level 2, and so on, until the final gather operator and the set of operators below it until the leaf I/O operators that represent the leaf level.

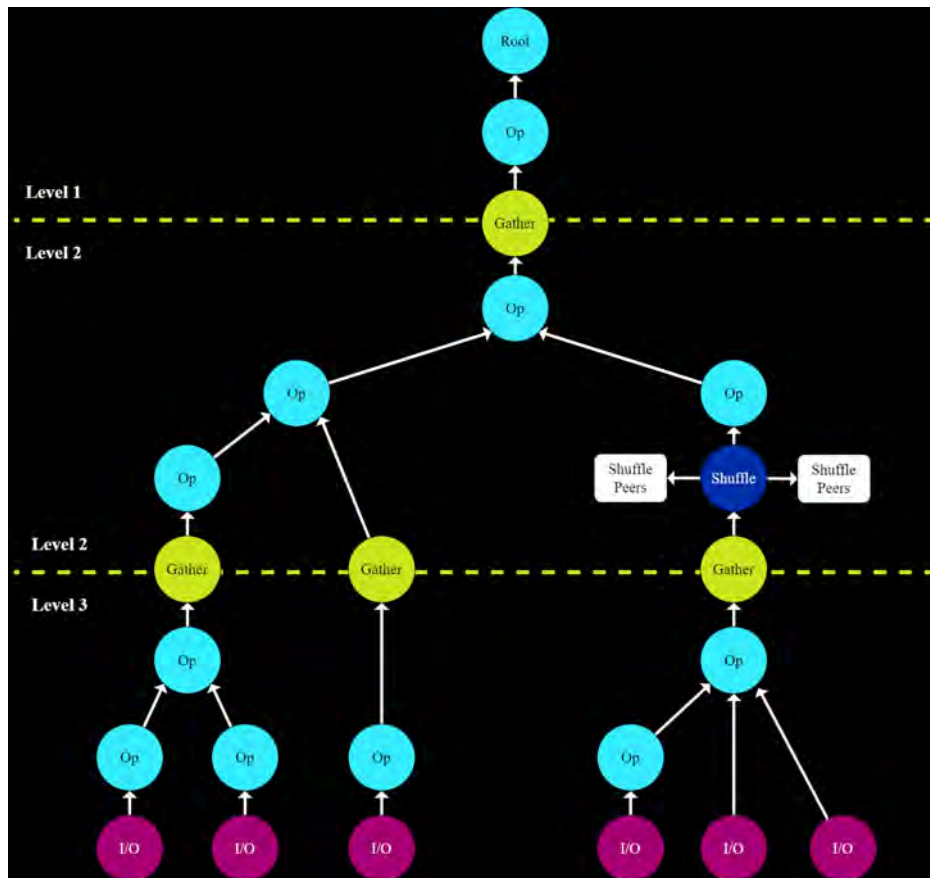


Figure 25: Query tree operators for three levels

When referring to the query tree, each gather operator represents a fan-out from one level N gather operator to some fixed number of N+1 nodes. During execution, a specific level N node is responsible for coordinating and amalgamating partial query results from the level N+1 nodes below it. By definition, there is exactly one level 1 node in a query, fanning out to some number of level 2 nodes, each of which itself fans out to some number of level 3 nodes, and so on. The fan-out factor is closely aligned with the cluster sizes in use.

In both cases, the tree structure promotes scalable query execution in a fault-tolerant way.

## Related Links

[Multiple Storage Clusters for Loading Data](#)

## Query Ocient

### Related Videos

At the Whiteboard with Ocient: Scale

## 9.6 Loading Characteristics and Concepts

When considering trillions or quadrillions of distinct records being stored and accessed as part of an ever-growing data warehouse, the manner in which data is loaded is important. The loading infrastructure of the data warehouse must not only provide strong correctness and durability guarantees in service of modern data pipelines, but must do so at very high load rates as well. For example, if an individual record averages 1000 bytes in size and the data warehouse must store 100 trillion records, the size of the records to be loaded is  $10^{14} * 10^3 = 10^{17}$  bytes, or 100 petabytes. Note that this is *pre-compression* because this is the volume of data that will be presented, row by row, to the data warehouse. If this information represents 1 year's worth of records, the loading system must be prepared to handle a steady-state average of ~25 Gbps. When the data rate is not completely uniform in time (perhaps displaying cyclic behavior associated with the day/night cycle) it is easy to imagine the required load rate can measure in the 100's, or even 1000's of Gbps.

Obviously the sizes and rates are arbitrary, but they are not unrealistic. Consequently a high-throughput, scalable, and robust loading infrastructure is a first-class citizen in an Ocient Hyperscale Data Warehouse™, on par complexity-wise with the execution engine itself.

### Record-Level Resolution

The distinction between “stream” and “batch” is somewhat arbitrary, depending on the system in consideration, but in the Ocient® loading infrastructure the fundamental unit of loading is a single record in a table. Put another way, the various guarantees listed here are made on a record-by-record basis and not on batches of records. Supporting this is the concept of a **record stream** of which the loading infrastructure can manage an unbounded number. A record stream is an ordered sequence of records, and loading considers each record in a stream independently of each other.

### Low Latency Querying

A key design goal of the loading system was a low **time to query-ability** or **TTQ**. Strict guarantees are not made but this usually means that any record presented to the loading system is available to be queried on the order of seconds. An interesting property of the implementation is that the variance on TTQ *decreases* as the aggregate record rate *increases*. This is because the loading system maintains internal structures named **pages** that accumulate records, and the faster they fill the more quickly they are made available for querying.

### Low Latency Durability

A necessary requirement of low-latency querying is the ability to ensure that any particular record is stored in a fault-tolerant manner. To guarantee correctness of query results it must be the case that when a record is considered during query execution (either included or filtered out) it must be considered for *all* subsequent queries until it is deleted by the user. The **durability** guarantee made by the loading system must come before query-ability (and therefore is measured in seconds on a record-by-record basis) and matches the system's configured fault tolerance. For example, if the erasure coding scheme in use has a K value of 2, then the loading system must store all records in a manner that allows for at least 2 losses before they can be made queryable.

The unit of durability in the loading system is named a **page** and within the loading and storage system, a page shares many characteristics with the segments: each page is a self-contained set of records and their own local metadata and statistics, and the consensus-based ownership scheme mediated by the storage clusters is the same for both. However, there are some key distinctions:

1. **Typical Size** — A page is usually substantially smaller than a segment. While segments can be multiple gigabytes in size, pages are more commonly around 100 megabytes.



2. **Replication** — Segments are grouped together into segment groups and mutually erasure coded for fault tolerance. Pages, however, are *replicated* within the storage layer. This is less space efficient for the same fault tolerance, but pages are short-lived: they exist only long enough to be converted into segments, which are denser and richer in metadata. Because they do not exist for very long, the extra storage overhead of the replication scheme is inconsequential, and the loading system opts not to waste the computational effort computing the erasure-coded parity data.

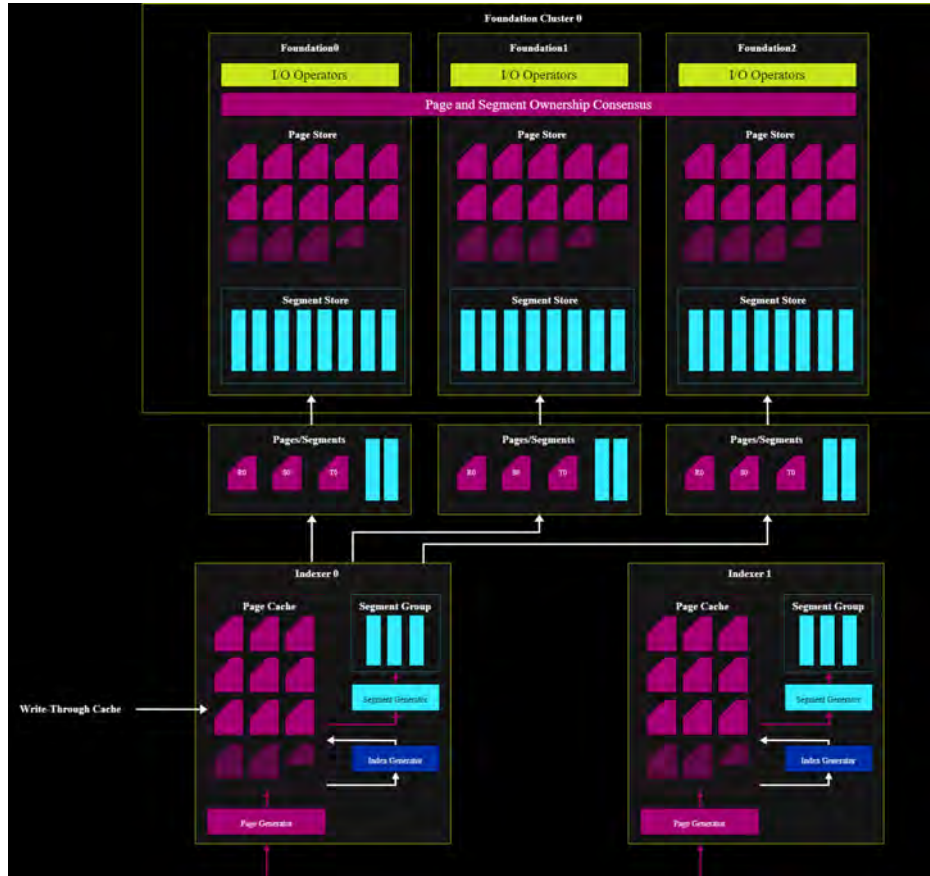


Figure 26: Loading page to segment design

### Exactly-Once Guarantee

When the record stream sources can support it, the Ocident loading infrastructure is capable of providing exactly-once semantics for every record loaded. This means that the record source and the query layer can be guaranteed that every record presented for load is not only not lost (no “gaps” exist) but also not duplicated. On the surface this seems like an obvious and simple requirement, but at millions of records per second in a fault-tolerant distributed system, it is extremely non-trivial to provide this guarantee.

The loading systems of other database engines can also provide this guarantee but often with caveats that might not be acceptable. For example, executing a sequence of `INSERT INTO` statements within the context of a transaction is a possible approach, but distributed transaction processing does not easily scale to millions of records per second. Beyond that there is the issue of how to manage inter-transaction guarantees that place additional requirements on the data sources.

If transactions are not in use, it is generally not possible to provide exactly-once guarantees without a bi-directional contract executed by the record source and the loading system. In a very common form of this contract, the loading system provides the exactly-once guarantee by guaranteeing it will de-duplicate any records presented more than once (i.e. the loading system is *idempotent* with respect to record insertions). Then, the source can safely re-send any records the loading system does not indicate it has received. So long as the source resends anything it has not

been explicitly told was received (and made durable) the two endpoints can together guarantee every record will be present exactly one time.

When this approach is taken, the loading system's job becomes simple deduplication. On some systems, this is achieved using unique record identifiers. First, the source must be able to indicate a column or set of columns whose values taken together for any particular record uniquely identify it. Then, for each record inserted, a hash set or other structure is consulted. If the unique identifier is not present, it is inserted, if it is, the record is silently dropped. While technically cheap to implement, this scheme suffers because the set structure cannot be allowed to grow indefinitely, and therefore some time or size based bound is placed on the exactly-once guarantee. An additional significant issue is that it is challenging and expensive to ensure the set structure is replicated or distributed amongst all nodes participating in the loading system. This presents correctness and scalability challenges to the exactly-once guarantee.

The Ocident loading system takes the idempotent deduplication approach, but performs its deduplication checks in a different manner. The crux of the scheme is that most systems that aim for exactly-once guarantees are also able to replay their record streams *in the same order*. This is true for queuing systems such as Apache® Kafka® as well as file sources such as Apache® Parquet™ files stored in Amazon® Web Services<sup>SM</sup> (AWS<sup>SM</sup>) S3. Due to this property, the Ocident deduplication scheme tracks a ***durability horizon*** for each distinct record stream. Recall that a record stream is an *ordered* sequence of records. The durability horizon is an ever-advancing offset into a stream that indicates the record at which the data warehouse guarantees all previous records up to and including that record, for that stream, are guaranteed durable. Using this scheme, the source can easily identify the set of records it should send or re-send. In the loading system, deduplication becomes cheap in both time and space: the system need only store the durability horizon per stream (instead of an unbounded set), and deduplication checks are as simple as a basic interval check comparing whether or not sequences of records stored in pages overlap.

### Linear Scalability at High Rates

In order to enable the extremely high record and bit rates associated with hyperscale data pipelines, it is important that the loading system be capable of scaling linearly to address arbitrary loading complexity. The Ocident loading layer achieves true linear scalability due to a shared-nothing approach between nodes executing loading operations. Being linearly scalable, it is relatively straightforward to design a fault-tolerant deployment that can handle any arbitrary rate by simply multiplying by the capabilities of a single node for a specified schema.

With that said, it is almost impossible to specify an exact achievable load rate on a per-node basis in advance. This is because the process is bound by the computational complexity of calculating indexes and metadata (which are schema-specified), the width of the erasure coding scheme (which are user-specified), and the bit rate being handled (which is both schema *and* data-dependent). With that said, each Loader Node is usually bounded by memory bandwidth, and it is not uncommon to be capable of steady-state load rates per node measuring in dozens of gigabits per second or millions of records per second. Usually, nodes are also capable of bursting to higher rates: 10's of millions of records per second and approaching 100 Gbps.

### Conclusion

The architecture is designed to maximize performance on the world's largest datasets. The massively parallel Compute Adjacent Storage Architecture® features storage and query processing optimizations to achieve scalable performance on commodity hardware with NVMe SSD storage. The custom-built, fully-integrated SQL execution engine and optimizer deeply integrate with the storage and I/O layer. The result is a state-of-the-art data warehouse that can deliver outstanding performance loading and querying record sets numbering in the trillions, quadrillions, and beyond.

### Related Links

[Ingest Data with Legacy LAT Reference](#)

[Key Concepts](#)

---

## 10 SQL Reference

This SQL Reference section includes details about the different data types, functions, and operators available when writing queries for Ocident®. For an introduction to the general query syntax and working with queries in Ocident, see [Query Ocident](#).

For information on the SQL syntax conventions used throughout the Ocident documentation, see the [SQL Syntax Conventions](#) page.

### Data Types

- [Data Types](#)

### Functions & Operators

- [Aggregate Functions](#)
- [Sorted Aggregate Functions](#)
- [Window Aggregate Functions](#)
- [Scalar Data Conversion Functions](#)
- [Date and Time Functions](#)
- [Time Zone Functions](#)
- [Formatting Functions](#)
- [Math Functions and Operators](#)
- [Character and Binary Functions](#)
- [Matrix Functions and Operators](#)
- [Tuple Functions and Operators](#)
- [Other Functions and Expressions](#)
- [Array Functions and Operators](#)
- [HyperLogLog Functions](#)
- [Network Type Functions](#)

### Geospatial Functions & Operators

- [Attribute Functions](#)
- [Conversion Functions](#)
- [Linestring Functions](#)
- [Linestring Constructors](#)
- [Point Constructors](#)
- [Polygon Constructors](#)
- [Spatial Measurement](#)
- [Spatial Operators](#)
- [Spatial Relationships](#)
- [Spatiotemporal Measurement](#)
- [Spatiotemporal Operators](#)

### DDL SQL Statements

- [Cluster and Node Management](#)
- [Database, Tables, Views, and Indexes](#)
- [Data Pipelines](#)
- [Distributed Tasks](#)
- [Machine Learning Models](#)
- [Users, Groups, and Service Classes](#)
- [Data Integrity and Storage](#)
- [Query Analysis](#)
- [Statistics Cache Management](#)

**DCL SQL Statements**

- [Data Control Language \(DCL\) Statement Reference](#)

**Machine Learning Execution**

- [Machine Learning Model Functions](#)

**Other SQL Reference**

- [Errors and Warnings](#)
- [System Catalog Reference](#)

**10.1 Alphabetical SQL Functions List**

| Function Name                         | Category Name                                  | Function Description                                                                                         |
|---------------------------------------|------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| <a href="#">ABS</a>                   | Math Functions                                 | Returns the absolute value of a specified floating-point number.                                             |
| <a href="#">ABS</a>                   | Matrix Functions                               | Returns magnitude of one-dimensional matrix/vector.                                                          |
| <a href="#">ACOS</a>                  | Math Functions                                 | Returns the inverse cosine of a specified floating-point number.                                             |
| <a href="#">ACOSH</a>                 | Math Functions                                 | Returns the hyperbolic arc-cosine of a specified floating-point number.                                      |
| <a href="#">ADD_MONTHS</a>            | Date and Time Functions                        | Adds the specified number of months to the date.                                                             |
| <a href="#">ANY_VALUE</a>             | Aggregate Functions                            | Returns an arbitrary, non-NULL value from the input column.                                                  |
| <a href="#">APPROX_COUNT_DISTINCT</a> | Aggregate Functions                            | Approximate distinct count by using hyper-log-log (95% confidence interval that the value is within 4.5%).   |
| <a href="#">APPROX_SUM</a>            | Aggregate Functions                            | Allows the aggregation engine to use a faster, non-deterministic ordering to summate floating-point columns. |
| <a href="#">ARRAY[]</a>               | Array Functions                                | PostgreSQL®-compliant constructor. The type of the array is deduced from the elements.                       |
| <a href="#">ARRAY_AGG</a>             | Sorted Aggregate Functions                     | Returns an array containing every row from the expression.                                                   |
| <a href="#">ARRAY_APPEND</a>          | Array Functions                                | Add value to the back of an array.                                                                           |
| <a href="#">ARRAY_ARGMAX</a>          | Array Functions                                | Returns the corresponding argmax of the array as a BIGINT index.                                             |
| <a href="#">ARRAY_ARGMIN</a>          | Array Functions                                | Returns the corresponding argmin of the array as a BIGINT index.                                             |
| <a href="#">ARRAY_CAT</a>             | Array Functions                                | Concatenate 2 arrays into a new one.                                                                         |
| <a href="#">ARRAY_CAP</a>             | Special Data Pipeline Transformation Functions | Restrict the length of an array to a maximum number of elements.                                             |
| <a href="#">ARRAY_COMPACT</a>         | Special Data Pipeline Transformation Functions | Removes NULL values from the array.                                                                          |
| <a href="#">ARRAY_CONTAINS</a>        | Special Data Pipeline Transformation Functions | Returns <b>true</b> if the array contains the specified value.                                               |
| <a href="#">ARRAY_DISTINCT</a>        | Special Data Pipeline Transformation Functions | Return all unique values in an array.                                                                        |

| Function Name                   | Category Name                                  | Function Description                                                                                                                  |
|---------------------------------|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ARRAY_LENGTH</a>    | Array Functions                                | Return the number of elements of a given array.                                                                                       |
| <a href="#">ARRAY_MAX</a>       | Array Functions                                | Returns the corresponding maximum of the array.                                                                                       |
| <a href="#">ARRAY_MIN</a>       | Array Functions                                | Returns the corresponding minimum of the array.                                                                                       |
| <a href="#">ARRAY_POSITION</a>  | Array Functions                                | Returns the position of the first matching scalar in the array.                                                                       |
| <a href="#">ARRAY_POSITIONS</a> | Array Functions                                | Returns all elements stored in the right array and returns their respective positions in the left array.                              |
| <a href="#">ARRAY_PREPEND</a>   | Array Functions                                | Add value to the front of an array.                                                                                                   |
| <a href="#">ARRAY_REMOVE</a>    | Array Functions                                | remove value from the array.                                                                                                          |
| <a href="#">ARRAY_REPLACE</a>   | Array Functions                                | replace value by another in an array.                                                                                                 |
| <a href="#">ARRAY_SORT</a>      | Special Data Pipeline Transformation Functions | Sort and return the input array based on the natural ordering of its elements or the specified Lambda function.                       |
| <a href="#">ARRAY_SUM</a>       | Array Functions                                | Returns the sum of the array. The array must be one-dimensional and contain numeric values. NULL values do not contribute to the sum. |
| <a href="#">ARRAY_TO_STRING</a> | Array Functions                                | Converts array to a list of elements separated by 'delimiter'.                                                                        |
| <a href="#">ASCII</a>           | Character and Binary Functions                 | Returns the ASCII code value of the leftmost character of the character value.                                                        |
| <a href="#">ASIN</a>            | Math Functions                                 | Returns the inverse sine of a specified floating-point number.                                                                        |
| <a href="#">ASINH</a>           | Math Functions                                 | Returns the hyperbolic arc-sine of a specified floating-point number.                                                                 |
| <a href="#">ATAN</a>            | Math Functions                                 | Returns the inverse tangent of a specified floating-point number.                                                                     |
| <a href="#">ATAN2</a>           | Math Functions                                 | Returns the inverse tangent of two numeric, floating-point values.                                                                    |
| <a href="#">ATANH</a>           | Math Functions                                 | Returns the hyperbolic arc-tangent of a specified floating-point number.                                                              |
| <a href="#">AVG</a>             | Aggregate Functions                            | Average, or arithmetic mean, over the set.                                                                                            |
| <a href="#">BICDF</a>           | Math Functions                                 | The cumulative distribution function of the standard bivariate normal distribution.                                                   |
| <a href="#">BIGINT</a>          | Scalar Data Conversion Functions               | Casts the argument to a value of type <code>bigint</code> .                                                                           |
| <a href="#">BINARY</a>          | Scalar Data Conversion Functions               | Parses a hexadecimal string (such as 0x54ab) to create a binary value.                                                                |
| <a href="#">BIPDF</a>           | Math Functions                                 | Letters can be of either case. The probability density function of the standard bivariate normal distribution.                        |
| <a href="#">BIT_LENGTH</a>      | Character and Binary Functions                 | Returns the length of the character value in bits.                                                                                    |
| <a href="#">BITAND</a>          | Math Functions                                 | Alias for the BITFUNC syntax <code>BITFUNC('AND', x, y)</code> .                                                                      |

| Function Name                 | Category Name                                  | Function Description                                                                                                                                                                            |
|-------------------------------|------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">BITFUNC</a>       | Math Functions                                 | Performs a variety of bit operations. Can be any of these string literals: 'AND', 'OR', or 'XOR'.                                                                                               |
| <a href="#">BITNOT</a>        | Math Functions                                 | Returns the bitwise negation of <code>integral_x</code> .                                                                                                                                       |
| <a href="#">BITOR</a>         | Math Functions                                 | Alias for the BITFUNC syntax <code>BITFUNC('OR', x, y)</code> .                                                                                                                                 |
| <a href="#">BITXOR</a>        | Math Functions                                 | Alias for the BITFUNC syntax <code>BITFUNC('XOR', x, y)</code> .                                                                                                                                |
| <a href="#">BOOLAND</a>       | Math Functions                                 | Alias for the BOOLFUNC syntax <code>BOOLFUNC('AND', x, y)</code> .                                                                                                                              |
| <a href="#">BOOLEAN</a>       | Scalar Data Conversion Functions               | Parses a string to create a Boolean value. The string must contain either true or false. It is case-insensitive.                                                                                |
| <a href="#">BOOLFUNC</a>      | Math Functions                                 | Performs a Boolean logical evaluation on arguments <code>x</code> and <code>y</code> . Can be any of these string literals: 'AND', 'OR', or 'XOR'.                                              |
| <a href="#">BOOLNOT</a>       | Math Functions                                 | Returns the logical negation of the <code>BOOLFUNC</code> function.                                                                                                                             |
| <a href="#">BOOLOR</a>        | Math Functions                                 | Alias for the BOOLFUNC syntax <code>BOOLFUNC('OR', x, y)</code> .                                                                                                                               |
| <a href="#">BOOLXOR</a>       | Math Functions                                 | Alias for the BOOLFUNC syntax <code>BOOLFUNC('XOR', x, y)</code> .                                                                                                                              |
| <a href="#">BTRIM</a>         | Character and Binary Functions                 | Alias for <code>TRIM</code> .                                                                                                                                                                   |
| <a href="#">BYTE</a>          | Scalar Data Conversion Functions               | Casts the argument to a value of type <code>byte</code> .                                                                                                                                       |
| <a href="#">CASE</a>          | Conditional Functions                          | <code>CASE</code> operates similarly to conditional scripting in other programming languages, allowing it to function like an <code>if / then / else</code> statement or as a switch statement. |
| <a href="#">CASE WHEN</a>     | Special Data Pipeline Transformation Functions | Returns the <b>result</b> value based on whether an expression is <b>true</b> .                                                                                                                 |
| <a href="#">CAST_TO_ARRAY</a> | Array Functions                                | Casts the elements of the array to another type.                                                                                                                                                |
| <a href="#">CAST_TO_TUPLE</a> | Tuple Functions                                | Converts a tuple into another tuple of a different type.                                                                                                                                        |
| <a href="#">CBRT</a>          | Math Functions                                 | Returns the cube root of the numeric value <code>x</code> .                                                                                                                                     |
| <a href="#">CDF</a>           | Math Functions                                 | The cumulative distribution function of the standard normal distribution. Returns the probability that a random sample is less than or equal to the specified value.                            |
| <a href="#">CEIL</a>          | Math Functions                                 | Returns the nearest integer greater than or equal to <code>x</code> .                                                                                                                           |
| <a href="#">CEILING</a>       | Math Functions                                 | Alias for <code>CEIL</code> .                                                                                                                                                                   |
| <a href="#">CENTURY</a>       | Date and Time Functions                        | Returns the number of centuries.                                                                                                                                                                |
| <a href="#">CHAR</a>          | Array Functions                                | Converts array to its string representation.                                                                                                                                                    |
| <a href="#">CHAR</a>          | Scalar Data Conversion Functions               | Creates a string version of the numeric value.                                                                                                                                                  |

| Function Name                  | Category Name                  | Function Description                                                                                                                                                       |
|--------------------------------|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CHAR                           | Tuple Functions                | Converts a tuple to its string representation.                                                                                                                             |
| CHAR_LENGTH                    | Character and Binary Functions | Alias for LENGTH.                                                                                                                                                          |
| CHARACTER_LENGTH               | Character and Binary Functions | Alias for LENGTH.                                                                                                                                                          |
| CHR                            | Character and Binary Functions | Converts an integer value to a string.                                                                                                                                     |
| COALESCE                       | Conditional Functions          | Evaluates to the first argument that is not NULL, or NULL if all arguments are NULL.                                                                                       |
| COMMIT                         | System Functions               | Returns the most recent commit hash of the database to which the client is currently connected.                                                                            |
| CONCAT                         | Character and Binary Functions | Concatenates two values, which must both be either binary, hash, or string data types. This function is equivalent to the    operator.                                     |
| CONVERT_LOCAL_TIMESTAMP_TO_UTC | Time Zone Functions            | The function converts a timestamp in a specified local time zone to the UTC time zone.                                                                                     |
| CONVERT_UTC_TIMESTAMP_TO_LOCAL | Time Zone Functions            | The function converts a timestamp from the UTC time zone to a specified local time zone.                                                                                   |
| CORR                           | Aggregate Functions            | Alias for CORRELATION.                                                                                                                                                     |
| CORRELATION                    | Aggregate Functions            | Sample correlation.                                                                                                                                                        |
| CORRELATIONP                   | Aggregate Functions            | Population correlation.                                                                                                                                                    |
| COS                            | Math Functions                 | Returns the cosine of x.                                                                                                                                                   |
| COSH                           | Math Functions                 | Returns the hyperbolic cosine of x.                                                                                                                                        |
| COT                            | Math Functions                 | Returns the cotangent of x.                                                                                                                                                |
| COUNT                          | Aggregate Functions            | Number of rows in the set.                                                                                                                                                 |
| COVAR_POP                      | Aggregate Functions            | Alias for COVARIANCEP.                                                                                                                                                     |
| COVAR_SAMP                     | Aggregate Functions            | Alias for COVARIANCE.                                                                                                                                                      |
| COVARIANCE                     | Aggregate Functions            | Sample covariance.                                                                                                                                                         |
| COVARIANCEP                    | Aggregate Functions            | Population covariance.                                                                                                                                                     |
| CROSS_ENTROPY_LOSS             | Array Functions                | Returns the cross entropy loss of two arrays.                                                                                                                              |
| CROSS_ENTROPY_LOSS             | Matrix Functions               | Returns the cross entropy loss of two one-dimensional matrixes or vectors.                                                                                                 |
| CUME_DIST                      | Window Aggregate Functions     | Returns a number $0 < n \leq 1$ and can be used to calculate the percentage of values less than or equal to the current value in the group.                                |
| CURDATE                        | Date and Time Functions        | Alias for CURRENT_DATE.                                                                                                                                                    |
| CURRENT_DATABASE               | System Functions               | Alias for DATABASE.                                                                                                                                                        |
| CURRENT_DATE                   | Date and Time Functions        | Returns the current date in the format YYYY-MM-DD.                                                                                                                         |
| CURRENT_NODE                   | System Functions               | Returns the name of the SQL Node where the current query executes. The name of the node corresponds to the name column in the <code>sys.nodes</code> system catalog table. |
| CURRENT_NODE_ID                | System Functions               | Returns the identifier of the SQL Node where the current query executes.                                                                                                   |
| CURRENT_SCHEMA                 | System Functions               | Returns the name of the current schema.                                                                                                                                    |



| Function Name     | Category Name                    | Function Description                                                                                                                                                                                                                                                                 |
|-------------------|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CURRENT_SYSTEM    | System Functions                 | Returns the name of the system.                                                                                                                                                                                                                                                      |
| CURRENT_TIMESTAMP | Date and Time Functions          | Returns the current date and time as a TIMESTAMP value (e.g., YYYY-MM-DD hhss.mmm).                                                                                                                                                                                                  |
| CURRENT_USER      | System Functions                 | Returns the user for the current connection.                                                                                                                                                                                                                                         |
| DATABASE          | System Functions                 | Returns the name of the database to which the client is currently connected.                                                                                                                                                                                                         |
| DATE              | Scalar Data Conversion Functions | Parses a string in the form 'YYYY-MM-DD' to create a date. Extra characters are ignored.                                                                                                                                                                                             |
| DATE_PART         | Date and Time Functions          | Alias for EXTRACT.                                                                                                                                                                                                                                                                   |
| DATE_TRUNC        | Date and Time Functions          | Returns the date or timestamp entered, truncated to the specified precision.                                                                                                                                                                                                         |
| DATEADD           | Date and Time Functions          | Adds a specified number value (as a signed integer) to a specified date part of an input date value, and then returns that modified value.                                                                                                                                           |
| DATEDIFF          | Date and Time Functions          | This function returns an INT representing the difference between two date or time values in a specified date or time unit.                                                                                                                                                           |
| DAY               | Date and Time Functions          | Alias for DAY_OF_MONTH.                                                                                                                                                                                                                                                              |
| DAY_OF_MONTH      | Date and Time Functions          | Extracts the day-of-month portion of a timestamp or date as an integer.                                                                                                                                                                                                              |
| DAY_OF_WEEK       | Date and Time Functions          | Returns an integer, in the range of 1 to 7, that represents the day of the week.                                                                                                                                                                                                     |
| DAY_OF_YEAR       | Date and Time Functions          | Returns an integer in the range 1 to 366 that represents the day of the year.                                                                                                                                                                                                        |
| DAYS              | Scalar Data Conversion Functions | Converts an integral value to an interval value of type days to be used in date calculations.                                                                                                                                                                                        |
| DECADE            | Date and Time Functions          | The decade is the year divided by 10.                                                                                                                                                                                                                                                |
| DECIMAL           | Scalar Data Conversion Functions | Casts the argument to a value of type decimal.                                                                                                                                                                                                                                       |
| DEGREES           | Math Functions                   | Returns the corresponding angle in degrees for x in radians.                                                                                                                                                                                                                         |
| DELTA             | Window Aggregate Functions       | Computes the finite difference between successive values of expression under the specified ordering. This is a backwards difference, which means that, at degree one, the value for a given row is the difference between the value of expression for that row and the previous row. |
| DENSE_RANK        | Window Aggregate Functions       | Assigns a number to each row in the result set with equal values having the same number. There will be no gaps between ranks.                                                                                                                                                        |



| Function Name | Category Name                                  | Function Description                                                                                                      |
|---------------|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| DERIVATIVE    | Window Aggregate Functions                     | Computes the difference quotient between successive values of expression with respect to expression2.                     |
| DET           | Matrix Functions                               | Returns the determinant of the matrix as a double.                                                                        |
| DIV           | Math Functions                                 | Returns the result of x divided by y. If y is zero, returns NULL.                                                         |
| DOT           | Matrix Functions                               | Returns dot product of two one-dimensional matrixes/vectors.                                                              |
| DOUBLE        | Scalar Data Conversion Functions               | Casts the argument to a value of type double.                                                                             |
| DOW           | Date and Time Functions                        | Alias for DAY_OF_WEEK.                                                                                                    |
| DOY           | Date and Time Functions                        | Alias for DAY_OF_YEAR.                                                                                                    |
| EIGEN         | Matrix Functions                               | Returns eigenvalues and eigenvalues of a square matrix as a vector of pairs.                                              |
| ELEMENT_AT    | Special Data Pipeline Transformation Functions | Returns the element value of the array or tuple at the specified index.                                                   |
| ENDSWITH      | Character and Binary Functions                 | Returns true if x ends with y and false otherwise.                                                                        |
| EOMONTH       | Date and Time Functions                        | Returns the last day of the timestamp or date.                                                                            |
| EPOCH         | Date and Time Functions                        | The number of seconds after 1970-01-01 00:00:00 UTC.                                                                      |
| ERF           | Math Functions                                 | The error function is used for measurements that follow a normal distribution.                                            |
| ERFC          | Math Functions                                 | The complement of the error function. $ERFC(x) = 1 - ERF(x)$ .                                                            |
| EXP           | Math Functions                                 | Returns the exponential of x (e raised to the power of x).                                                                |
| EXPLODE_OUTER | Special Data Pipeline Transformation Functions | Expands a one-dimensional or multidimensional array into its elements with one element per row of output from the system. |
| EXTRACT       | Date and Time Functions                        | Extract a component from a timestamp or date.                                                                             |
| FILTER        | Special Data Pipeline Transformation Functions | Filters elements in an array based on the logic in a lambda expression.                                                   |
| FIRST_VALUE   | Window Aggregate Functions                     | Returns the first value in the ordered result set.                                                                        |
| FLATTEN       | Special Data Pipeline Transformation Functions | Transforms an N-dimensional array into an N-1-dimensional array.                                                          |
| FLOAT         | Scalar Data Conversion Functions               | Casts the argument to a value of type float.                                                                              |
| FLOOR         | Math Functions                                 | Returns the nearest integer less than or equal to x.                                                                      |
| FROBENIUS     | Matrix Functions and Operators                 | Returns the Frobenius norm of a matrix.                                                                                   |
| GAMMA         | Math Functions                                 | <a href="#">Gamma function.</a>                                                                                           |
| GREATEST      | Conditional Functions                          | Returns the largest non-NULL value of all the arguments, or NULL if all the arguments are NULL.                           |

| Function Name                                         | Category Name                    | Function Description                                                                                                                                                                                                                      |
|-------------------------------------------------------|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">HASH</a>                                  | Scalar Data Conversion Functions | Creates a fixed length binary value, with length <code>&lt;length&gt;</code> , from a string, i.e. <code>0x1234abcd</code> . Zero extended if the string does not have enough bytes, truncated if it has too many.                        |
| <a href="#">HEXBINX</a>                               | Math Functions                   | Returns the x-coordinate of the center of the nearest hexagonal bin to the point (x, y).                                                                                                                                                  |
| <a href="#">HEXBINY</a>                               | Math Functions                   | Returns the y-coordinate of the center of the nearest hexagonal bin to the point (x, y).                                                                                                                                                  |
| <a href="#">HINGE_LOSS</a>                            | Array Functions                  | Returns the hinge loss of two arrays.                                                                                                                                                                                                     |
| <a href="#">HINGE_LOSS</a>                            | Matrix Functions                 | Returns the hinge loss of two one-dimensional matrixes or vectors.                                                                                                                                                                        |
| <a href="#">HLL_SKETCH_CREATE</a>                     | HyperLogLog Functions            | Creates an HLL sketch from the data on a specified aggregated column. Returns a <code>HASH((2<sup>log2k</sup>) + 8)</code> data representation of the sketch that you can store in a separate column.                                     |
| <a href="#">HLL_SKETCH_GET_ESTIMATE</a>               | HyperLogLog Functions            | The scalar function converts a sketch into a distinct count estimate of a sketch value. Returns the distinct count estimate as a <code>BIGINT</code> .                                                                                    |
| <a href="#">HLL_SKETCH_GET_ESTIMATE_BOUND</a>         | HyperLogLog Functions            | Takes a <code>HLL_SKETCH</code> column or an integral <code>log2k</code> literal value and returns the resulting bounding 95-percent confidence interval error proportion as a <code>DOUBLE</code> .                                      |
| <a href="#">HLL_SKETCH_TO_STRING</a>                  | HyperLogLog Functions            | The <code>HLL_SKETCH_TO_STRING</code> scalar function takes a <code>HLL_SKETCH</code> column or value and returns a string summary of the sketch.                                                                                         |
| <a href="#">HLL_SKETCH_UNION</a> (aggregate function) | HyperLogLog Functions            | Merges multiple sketches in a single column into a unified sketch. All sketches must have the same precision. This function is an aggregate function and operates on a column.                                                            |
| <a href="#">HLL_SKETCH_UNION</a> (scalar function)    | HyperLogLog Functions            | Merges two sketches into a new combined sketch. This function is a scalar function and operates row-wise. The scalar function merges two sketch columns with heterogeneous precisions into a sketch with the lower of the two precisions. |
| <a href="#">HOUR</a>                                  | Date and Time Functions          | Extracts the hour portion of a timestamp as an integer.                                                                                                                                                                                   |
| <a href="#">HOURS</a>                                 | Scalar Data Conversion Functions | Converts an integral value to an interval value of type hours.                                                                                                                                                                            |
| <a href="#">IDENTITY_MATRIX</a>                       | Matrix Functions                 | Returns an identity matrix of the specified dimension.                                                                                                                                                                                    |

| Function Name          | Category Name                                  | Function Description                                                                                                                              |
|------------------------|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| IERF                   | Math Functions                                 | The inverse of the ERF error function.                                                                                                            |
| IERFC                  | Math Functions                                 | The inverse of the complement of the error function.                                                                                              |
| IF                     | Special Data Pipeline Transformation Functions | Returns T if the expression X evaluates to <b>true</b> , or the function returns F if X evaluates to <b>false</b> .                               |
| IF_NULL                | Conditional Functions                          | Alias for COALESCE.                                                                                                                               |
| INITCAP                | Character and Binary Functions                 | For each word in the provided string, capitalize the first character if it is alphabetic.                                                         |
| INSTR                  | Character and Binary Functions                 | Returns the index position of the first occurrence where the character value char_substring appears in the character value char by ignoring case. |
| INTEGER                | Scalar Data Conversion Functions               | Casts the argument to a value of type integer.                                                                                                    |
| INVERSE                | Matrix Functions                               | Returns inverse of a square, invertible matrix.                                                                                                   |
| IP                     | Network Type Functions                         | Casts an IP data type from an IPV4 data type expression.                                                                                          |
| IPV4                   | Network Type Functions                         | Casts an IPV4 address from an IPV6 address.                                                                                                       |
| ISDATE                 | Date and Time Functions                        | Returns TRUE if the input argument can be successfully cast to a date.                                                                            |
| IS_IPV4                | Network Type Functions                         | Tests whether the database can convert the IP value to the IPV4 data type.                                                                        |
| ISODOW                 | Date and Time Functions                        | Extracts the day of the week based on ISO 8601, which ranges from Monday (1) to Sunday (7).                                                       |
| JSON_EXTRACT_PATH_TEXT | Character and Binary Functions                 | Returns the value for the key-value pair referenced by a series of path elements in a JSON string.                                                |
| KURTOSIS               | Aggregate Functions                            | The sample over the set.                                                                                                                          |
| KURTOSISP              | Aggregate Functions                            | The population over the set.                                                                                                                      |
| LAG                    | Window Aggregate Functions                     | Returns the row which is the specified number backward from the current row. Default is 1 if offset is omitted.                                   |
| LAST_VALUE             | Window Aggregate Functions                     | Returns the last value in the ordered result set.                                                                                                 |
| LCASE                  | Character and Binary Functions                 | Alias for LOWER.                                                                                                                                  |
| LEAD                   | Window Aggregate Functions                     | Returns the row which is the specified number forward from the current row. Default is 1 if offset is omitted.                                    |
| LEAKYRELU              | Math Functions                                 | Returns the leaky rectified linear unit function of x.                                                                                            |
| LEAST                  | Conditional Functions                          | Returns the smallest non-NULL value of all arguments, or NULL if all arguments are NULL.                                                          |

| Function Name    | Category Name                                  | Function Description                                                                                                                                            |
|------------------|------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LEFT             | Character and Binary Functions                 | Return the number of characters in the string equal to the value integer. If the integer is negative, the function returns all but the last integer characters. |
| LEFT_SHIFT       | Math Functions                                 | Returns x shifted to the left by y bits.                                                                                                                        |
| LENGTH           | Character and Binary Functions                 | For character data types, this value is in terms of characters. For binary data types, this value is in terms of bytes.                                         |
| LN               | Math Functions                                 | Returns the natural logarithm of x.                                                                                                                             |
| LOCATE           | Character and Binary Functions                 | Alias for POSITION. Returns the index position of the first occurrence of the character value substring in the character value string.                          |
| LOG              | Math Functions                                 | Returns the base 10 logarithm of x. The optional base argument specifies the numeral system to use. If unspecified, the function defaults to base 10.           |
| LOG_GAMMA        | Math Functions                                 | The natural logarithm of the absolute value of the gamma function.                                                                                              |
| LOG_LOSS         | Array Functions                                | Returns the log loss of two arrays.                                                                                                                             |
| LOG_LOSS         | Matrix Functions                               | Returns the log loss of two one-dimensional matrixes or vectors.                                                                                                |
| LOG2             | Math Functions                                 | Returns the base 2 logarithm of x.                                                                                                                              |
| LOGITS_LOSS      | Array Functions                                | Returns the logits loss of two arrays.                                                                                                                          |
| LOGITS_LOSS      | Matrix Functions                               | Returns the logits loss of two one-dimensional matrixes or vectors.                                                                                             |
| LOWER            | Character and Binary Functions                 | Alias for LCASE. Convert string to lowercase.                                                                                                                   |
| LPAD             | Character and Binary Functions                 | Pad the input text to the specified length with the pad string on the left side.                                                                                |
| LTRIM            | Character and Binary Functions                 | Removes leading blanks from the string value string.                                                                                                            |
| LUPQ_DECOMP      | Matrix Functions                               | Returns LUPQ decomposition of a square matrix A as a tuple of 4 matrixes where $PAQ = LU$ .                                                                     |
| MAP_KEYS         | Special Data Pipeline Transformation Functions | Returns the keys in the specified JSON string.                                                                                                                  |
| MAP_VALUES       | Special Data Pipeline Transformation Functions | Returns the values in the specified JSON string.                                                                                                                |
| MATRIX_DIM       | Matrix Functions                               | Returns the dimensions of the specified matrix as a tuple of (row, col) integers.                                                                               |
| MAKE_MATRIX_IJ   | Matrix Functions                               | Creates an $ij$ matrix with elements $e_{00}, \dots, e_{ij}$ .                                                                                                  |
| MAKEDATETIME     | Date and Time Functions                        | Returns a timestamp consisting of the specified date and time.                                                                                                  |
| MATRIX_FROM_TEXT | Matrix Functions                               | Creates a matrix from the specified string.                                                                                                                     |

| Function Name                      | Category Name                                  | Function Description                                                                                           |
|------------------------------------|------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| <a href="#">MATRIX_TRACE</a>       | Matrix Functions                               | Returns trace of a square matrix as a double.                                                                  |
| <a href="#">MAX</a>                | Aggregate Functions                            | Maximum value in the specified column.                                                                         |
| <a href="#">MD5</a>                | Character and Binary Functions                 | Returns the hexadecimal string (all lowercase) representing the md5 hash of char.                              |
| <a href="#">METADATA</a>           | Special Data Pipeline Transformation Functions | Extracts the metadata value for the specified key from available metadata for the pipeline.                    |
| <a href="#">MICROSECONDS</a>       | Scalar Data Conversion Functions               | Converts an integral value to an interval value of type microseconds.                                          |
| <a href="#">MID</a>                | Character and Binary Functions                 | Alias for SUBSTRING.                                                                                           |
| <a href="#">MILLISECOND</a>        | Date and Time Functions                        | Extracts the millisecond portion of a timestamp as an integer.                                                 |
| <a href="#">MILLISECONDS</a>       | Scalar Data Conversion Functions               | Converts an integral value to an interval value of type milliseconds.                                          |
| <a href="#">MIN</a>                | Aggregate Functions                            | Minimum value in the specified column.                                                                         |
| <a href="#">MINUTE</a>             | Date and Time Functions                        | Extracts the minute portion of a timestamp or date as an integer.                                              |
| <a href="#">MINUTES</a>            | Scalar Data Conversion Functions               | Converts an integral value to an interval value of type minutes.                                               |
| <a href="#">MOD</a>                | Math Functions                                 | Returns the remainder from x divided by y.                                                                     |
| <a href="#">MONTH</a>              | Date and Time Functions                        | Extracts the month portion of a timestamp or date as an integer.                                               |
| <a href="#">MONTH_NAME</a>         | Date and Time Functions                        | Returns the calendar name in English of the month for the specified date.                                      |
| <a href="#">MONTHS</a>             | Scalar Data Conversion Functions               | Converts an integral value to an interval value of type months to be used in date calculations.                |
| <a href="#">MONTHS_BETWEEN</a>     | Date and Time Functions                        | Returns the difference between the two dates or timestamps in months as a DOUBLE.                              |
| <a href="#">MSECS</a>              | Date and Time Functions                        | The seconds field, including fractional parts. The function multiplies the seconds part of the value by 1,000. |
| <a href="#">MURMUR3</a>            | Conditional Functions                          | Returns a 32-bit MurmurHash3 hash of the input value as an INTEGER data type.                                  |
| <a href="#">NANOS_TO_TIMESTAMP</a> | Date and Time Functions                        | Convert a number of nanoseconds into a timestamp equivalent to the duration after the epoch time.              |
| <a href="#">NANOSECONDS</a>        | Scalar Data Conversion Functions               | Converts an integral value to an interval value of type nanoseconds.                                           |
| <a href="#">NEXT_DAY</a>           | Date and Time Functions                        | Returns the closest date after a specified date that lies on a specific day of the week.                       |
| <a href="#">NOW</a>                | Date and Time Functions                        | Alias for CURRENT_TIMESTAMP.                                                                                   |

| Function Name                         | Category Name                                  | Function Description                                                                                                                                                                           |
|---------------------------------------|------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">NTH_VALUE</a>             | Window Aggregate Functions                     | Returns the nth value in the ordered result set.                                                                                                                                               |
| <a href="#">NULL_IF</a>               | Conditional Functions                          | Returns the NULL value if two arguments are equal; otherwise, returns the first argument.                                                                                                      |
| <a href="#">NULL_MATRIX</a>           | Matrix Functions                               | Returns a null matrix of the given (row, col) dimensions.                                                                                                                                      |
| <a href="#">OCTET_LENGTH</a>          | Character and Binary Functions                 | Returns the length in bytes of a character or binary value.                                                                                                                                    |
| <a href="#">PARSE_DELIMITED_ARRAY</a> | Special Data Pipeline Transformation Functions | Converts a string of text data representing an array into a <code>CHAR[]</code> .                                                                                                              |
| <a href="#">PERCENT_RANK</a>          | Window Aggregate Functions                     | The value returned is $0 < n \leq 1$ and can be used to calculate the percentage of values less than the current group, excluding the highest value.                                           |
| <a href="#">PERCENTILE</a>            | Window Aggregate Functions                     | Returns the value that corresponds to the specified percentile ( $0 \leq n \leq 1$ ) within the group.                                                                                         |
| <a href="#">PI</a>                    | Math Functions                                 | Returns the constant value of $\pi$ .                                                                                                                                                          |
| <a href="#">PMOD</a>                  | Math Functions                                 | Returns the smallest non-negative equivalence class of $x \% y$ .                                                                                                                              |
| <a href="#">POSITION</a>              | Character and Binary Functions                 | Alias for <code>LOCATE</code> .                                                                                                                                                                |
| <a href="#">POWER</a>                 | Math Functions                                 | Returns x raised to the power of y.                                                                                                                                                            |
| <a href="#">PROBIT</a>                | Math Functions                                 | The inverse of the cumulative distribution function.                                                                                                                                           |
| <a href="#">PRODUCT</a>               | Aggregate Functions                            | Product over the set.                                                                                                                                                                          |
| <a href="#">QR_DECOMP</a>             | Matrix Functions                               | Returns QR decomposition of a matrix as a tuple of 2 matrixes.                                                                                                                                 |
| <a href="#">QUARTER</a>               | Date and Time Functions                        | Returns an integer between 1 and 4 that represents the quarter of the year in which the specified date falls.                                                                                  |
| <a href="#">RADIANS</a>               | Math Functions                                 | Returns the corresponding angle in radians for x in degrees.                                                                                                                                   |
| <a href="#">RAND</a>                  | Math Functions                                 | Takes no argument and returns a random DOUBLE value in the range [0, 1).                                                                                                                       |
| <a href="#">RANK</a>                  | Window Aggregate Functions                     | Assigns a number to each row in the result set with equal values having the same number. There can be gaps between ranks.                                                                      |
| <a href="#">RATIO_TO_REPORT</a>       | Window Aggregate Functions                     | Computes the ratio of a value to the sum of the set of values.                                                                                                                                 |
| <a href="#">RECORD_UUID</a>           | Other Transformation Functions                 | Returns string that represents a unique identifier for the record in a given pipeline for a given file_group or topic.                                                                         |
| <a href="#">REDUCE</a>                | Special Data Pipeline Transformation Functions | Applies a merge function to a starting value and all elements in the array, and then reduces the array to a single value. Optionally, specify a finish function for the returned single value. |

| Function Name  | Category Name                    | Function Description                                                                                                                                       |
|----------------|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| REGEXP_COUNT   | Character and Binary Functions   | Searches a string for all occurrences of a regular expression pattern.                                                                                     |
| REGEXP_INSTR   | Character and Binary Functions   | Searches a string using a regular expression pattern and returns an integer representing the start position or end position of the substring that matches. |
| REGEXP_REPLACE | Character and Binary Functions   | Searches a string for all occurrences of a regular expression pattern.                                                                                     |
| REGEXP_SUBSTR  | Character and Binary Functions   | Returns one substring from a string that matches a specified regular expression pattern.                                                                   |
| RELU           | Math Functions                   | Returns the rectified linear unit function of x.                                                                                                           |
| REPEAT         | Character and Binary Functions   | Repeats the character value char a number of times equal to num.                                                                                           |
| REPLACE        | Character and Binary Functions   | Replaces all occurrences of <b>substr_to_remove</b> in the character value string with <b>substr_to_replace</b> .                                          |
| REVERSE        | Character and Binary Functions   | Reverse the input string.                                                                                                                                  |
| RIGHT          | Character and Binary Functions   | Return the number of trailing characters in the string equal to the value integer.                                                                         |
| RIGHT_SHIFT    | Math Functions                   | Returns x shifted to the right by y bits.                                                                                                                  |
| ROUND          | Date and Time Functions          | Returns the specified date or timestamp, rounded to the specified precision.                                                                               |
| ROUND          | Math Functions                   | Returns x rounded to the nearest integer.                                                                                                                  |
| ROW_NUMBER     | Window Aggregate Functions       | Assigns a unique number to each row in the result set.                                                                                                     |
| RPAD           | Character and Binary Functions   | Pad the input text to the specified length with the pad string on the right side.                                                                          |
| RSUBSTRING     | Character and Binary Functions   | Returns the substring from the right side of a string, based on a specified length.                                                                        |
| RTRIM          | Character and Binary Functions   | Removes leading blanks from the string value string.                                                                                                       |
| SECOND         | Date and Time Functions          | Extracts the seconds portion of a timestamp as an integer.                                                                                                 |
| SECONDS        | Scalar Data Conversion Functions | Converts an integral value to an interval value of type seconds.                                                                                           |
| SHA1           | Character and Binary Functions   | Uses the SHA-1 cryptographic hash function to convert a string into a 40-character string representing the hexadecimal value of a 160-bit checksum.        |
| SHOW           | System Functions                 | The SHOW function enables you to explore the database and its metadata for user-defined items.                                                             |



| Function Name                       | Category Name                     | Function Description                                                                                                                                       |
|-------------------------------------|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">SIGN</a>                | Math Functions                    | Returns the positive (+1), zero (0), or negative (-1) sign of x.                                                                                           |
| <a href="#">SIN</a>                 | Math Functions                    | Returns the sine of x.                                                                                                                                     |
| <a href="#">SINH</a>                | Math Functions                    | Returns the hyperbolic sine of x.                                                                                                                          |
| <a href="#">SKEW</a>                | Aggregate Functions               | The sample over the set.                                                                                                                                   |
| <a href="#">SMALLINT</a>            | Scalar Data Conversion Functions  | Casts the argument to a value of type <b>smallint</b> .                                                                                                    |
| <a href="#">SOFTMAX</a>             | Array Functions                   | Returns the softmax of the array.                                                                                                                          |
| <a href="#">SOFTMAX</a>             | Matrix Functions                  | Returns the softmax of a one-dimensional matrix or vector.                                                                                                 |
| <a href="#">SPACE</a>               | Character and Binary Functions    | Returns a string of repeated spaces equal to the number value, repeat.                                                                                     |
| <a href="#">SPLIT_PART</a>          | Character and Binary Functions    | Split the value string based on the delimiter value. The function returns a substring from the split operation based on the index value (starting from 1). |
| <a href="#">SPLIT_TO_ARRAY</a>      | Character and Binary Functions    | Splits a string into an array of substrings.                                                                                                               |
| <a href="#">SQRT</a>                | Math Functions                    | Returns the square root of x.                                                                                                                              |
| <a href="#">SQUARE</a>              | Math Functions                    | Returns the square of x.                                                                                                                                   |
| <a href="#">ST_ADDPOINT</a>         | Geospatial Linestring Constructor | Adds a POINT to the given LINESTRING at the specified 0-indexed location.                                                                                  |
| <a href="#">ST_ANGLE</a>            | Geospatial Spatial Measurement    | Calculates the angle between two lines.                                                                                                                    |
| <a href="#">ST_AREA</a>             | Geospatial Spatial Measurement    | Returns the area of the specified geospatial object in the specified unit of measurement.                                                                  |
| <a href="#">ST_ASBINARY</a>         | Geospatial Conversion Functions   | Returns the well-known binary (WKB) representation of the specified geography. Alias of ST_ASWKB.                                                          |
| <a href="#">ST_ASEWKT</a>           | Geospatial Conversion Functions   | Returns a string that represents geographic coordinates of a specified POINT in the specified format.                                                      |
| <a href="#">ST_ASGEOJSON</a>        | Geospatial Conversion Functions   | Alias of ST_ASBINARY.                                                                                                                                      |
| <a href="#">ST_ASLATLONTEXT</a>     | Geospatial Conversion Functions   | Returns the GeoJSON representation of the specified geography using the IETF standards.                                                                    |
| <a href="#">ST_ASTEXT</a>           | Geospatial Conversion Functions   | Alias of ST_ASTEXT.                                                                                                                                        |
| <a href="#">ST_ASWKB</a>            | Geospatial Conversion Functions   | Alias of ST_ASTEXT.                                                                                                                                        |
| <a href="#">ST_ASWKT</a>            | Geospatial Conversion Functions   | Alias of ST_ASWKT and ST_EWKT. Returns the WKT representation of the specified geography.                                                                  |
| <a href="#">ST_AZIMUTH</a>          | Geospatial Spatial Measurement    | Returns the azimuth of the line from <b>point1</b> to <b>point2</b> in radians.                                                                            |
| <a href="#">ST_BOUNDINGDIAGONAL</a> | Geospatial Spatial Operators      | Returns the diagonal LINESTRING from the minimum point to the maximum point of the bounding box that ST_ENVELOPE returns.                                  |



| Function Name                       | Category Name                    | Function Description                                                                                                                                                                                                         |
|-------------------------------------|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ST_BUFFER</a>           | Geospatial Spatial Operators     | Returns a geography that contains all points where the distance from the geography is less than or equal to the specified distance.                                                                                          |
| <a href="#">ST_CENTROID</a>         | Geospatial Point Constructors    | The geographic center of mass is calculated by taking the average of all points on a three-dimensional sphere, projecting the resultant point onto the sphere, and converting it back to latitude and longitude coordinates. |
| <a href="#">ST_CLOSESTPOINT</a>     | Geospatial Spatial Operators     | Returns the two-dimensional POINT of one specified geospatial object that is closest to a second specified geospatial object.                                                                                                |
| <a href="#">ST_CLUSTERDBSCAN</a>    | Geospatial Spatial Relationships | Returns the cluster number for each input geography, based on a two-dimensional implementation of the density-based spatial clustering of applications with noise (DBSCAN) algorithm.                                        |
| <a href="#">ST_CONTAINS</a>         | Geospatial Spatial Relationships | Returns TRUE if the first geographic argument, geo1, contains the second geographic argument, geo2.                                                                                                                          |
| <a href="#">ST_CONTAINSPROPERLY</a> | Geospatial Spatial Relationships | Returns true if geo2 lies entirely in the interior of geo1, and does not intersect or touch the boundary or exterior points.                                                                                                 |
| <a href="#">ST_CONVEXHULL</a>       | Geospatial Spatial Operators     | The convex hull is the smallest convex geometry that encloses the input geometry.                                                                                                                                            |
| <a href="#">ST_COORDDIM</a>         | Geospatial Attribute Functions   | Alias for ST_NDIMS or ST_NDIMENSION. Returns an INTEGER of the coordinate dimension of the specified geography.                                                                                                              |
| <a href="#">ST_COVEREDBY</a>        | Geospatial Spatial Relationships | Returns TRUE if no POINT in geo1 is outside of geo2.                                                                                                                                                                         |
| <a href="#">ST_COVERS</a>           | Geospatial Spatial Relationships | Returns TRUE if no POINT in geo2 is outside of geo1.                                                                                                                                                                         |
| <a href="#">ST_CROSSES</a>          | Geospatial Spatial Relationships | Returns TRUE if two geospatial objects meet these criteria: The intersection of the geospatial interiors is not empty. The intersection is not equal to geo1 or geo2. Neither geospatial object is a single POINT.           |
| <a href="#">ST_DIFFERENCEARRAY</a>  | Geospatial Spatial Operators     | Returns an array containing any geospatial objects that are present in the first specified geospatial argument that are not found in the second geospatial argument.                                                         |
| <a href="#">ST_DIMENSION</a>        | Geospatial Attribute Functions   | Returns an INTEGER that represents the dimension of the specified geography.                                                                                                                                                 |

| Function Name                          | Category Name                     | Function Description                                                                                                                                                       |
|----------------------------------------|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ST_DISJOINT</a>            | Geospatial Spatial Relationships  | Returns true if the specified geographies have no intersection, including boundaries. Both geographic arguments can be different types.                                    |
| <a href="#">ST_DISTANCE</a>            | Geospatial Spatial Measurement    | Returns the minimum distance between the specified arguments.                                                                                                              |
| <a href="#">ST_DISTANCE</a>            | Geospatial Spatiotemporal Measu   | Returns the two-dimensional interpolated minimum simultaneous distance between two <b>LINESTRING-TIMESTAMP</b> array pairs in the specified unit of measurement.           |
| <a href="#">ST_DISTANCESPHERE</a>      | Geospatial Spatial Measurement    | Returns the minimum distance between the specified arguments using a spherical computation.                                                                                |
| <a href="#">ST_DISTANCESPHEROID</a>    | Geospatial Spatial Measurement    | Returns the minimum distance between the specified arguments using a spheroid computation.                                                                                 |
| <a href="#">ST_DWITHIN</a>             | Geospatial Spatial Relationships  | Returns TRUE if the geographies are within a specified distance in meters.                                                                                                 |
| <a href="#">ST_ENDPOINT</a>            | Geospatial Linestring Constructor | Returns the endpoint of a specified <b>LINESTRING</b> . The returned value is a <b>POINT</b> .                                                                             |
| <a href="#">ST_ENVELOPE</a>            | Geospatial Spatial Operators      | Returns a <b>POLYGON</b> that represents the minimum bounding box for the specified geography.                                                                             |
| <a href="#">ST_EQUALS</a>              | Geospatial Spatial Relationships  | Returns TRUE if both geographies are spatially equal.                                                                                                                      |
| <a href="#">ST_EUCLIDEANDISTANCE3D</a> | Geospatial Spatial Measurement    | Returns the minimum distance between two geospatial points with altitude values.                                                                                           |
| <a href="#">ST_EXPAND</a>              | Geospatial Spatial Operators      | Returns the bounding box of a specified geospatial value, which is expanded by a specified length.                                                                         |
| <a href="#">ST_EXTERIORRING</a>        | Geospatial Spatial Operators      | Returns a <b>LINESTRING</b> that represents the exterior ring of a provided <b>POLYGON</b> value.                                                                          |
| <a href="#">ST_FLIPCOORDINATES</a>     | Geospatial Spatial Operators      | Returns a new geographic object with the X and Y coordinates switched using the specified argument.                                                                        |
| <a href="#">ST_FORCE2D</a>             | Geospatial Spatial Operators      | Convert a geographic object into a two-dimensional geography.                                                                                                              |
| <a href="#">ST_FORCECCW</a>            | Geospatial Polygon Constructors   | Creates a standardized polygon from an existing one. Ocint® defines standardized as the exterior being counterclockwise (CCW) and all holes being clockwise (CW) oriented. |
| <a href="#">ST_GEOPOINT</a>            | Geospatial Point Constructors     | Creates and returns a <b>POLYGON</b> geography with a single point, defined by the longitude and latitude specified for the function.                                      |

| Function Name                        | Category Name                    | Function Description                                                                                                                                                                                                             |
|--------------------------------------|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ST_GEOHASH</a>           | Geospatial Conversion Functions  | Returns a string that represents the geohash of the input POINT.                                                                                                                                                                 |
| <a href="#">ST_GEOMETRYTYPE</a>      | Geospatial Attribute Functions   | Returns a string representing the geometry type of the input value.                                                                                                                                                              |
| <a href="#">ST_HAUSDORFFDISTANCE</a> | Geospatial Spatial Measurement   | Returns the Hausdorff distance between two geographies in a specified measurement.                                                                                                                                               |
| <a href="#">ST_INTERIORRINGN</a>     | Geospatial Spatial Operators     | Returns a LINESTRING representing the interior ring of the specified POLYGON, which is specified by its index. (1-indexed)                                                                                                       |
| <a href="#">ST_INTERSECTALL</a>      | Geospatial Spatial Operators     | Returns the intersection of all geographies in the specified array. All geographies in the array must be the same type.                                                                                                          |
| <a href="#">ST_INTERSECTION</a>      | Geospatial Spatiotemporal Measu  | Returns a tuple that represents the intersection of a spatiotemporal LINESTRING with a static geography.                                                                                                                         |
| <a href="#">ST_INTERSECTIONARRAY</a> | Geospatial Spatial Operators     | Returns a geography that represents the point-set intersection of two geographies.                                                                                                                                               |
| <a href="#">ST_INTERSECTS</a>        | Geospatial Spatial Relationships | Returns TRUE if the specified geographies have an intersection, including boundaries.                                                                                                                                            |
| <a href="#">ST_ISCCW</a>             | Geospatial Spatial Relationships | Alias for ST_ISPOLYGONCCW.                                                                                                                                                                                                       |
| <a href="#">ST_ISCLOSED</a>          | Geospatial Spatial Relationships | Returns TRUE if an input POLYGON has an exterior that is counter-clockwise.                                                                                                                                                      |
| <a href="#">ST_ISEMPY</a>            | Geospatial Attribute Functions   | Returns TRUE if the specified geography value is empty, such as 'POLYGON EMPTY'.                                                                                                                                                 |
| <a href="#">ST_ISPOLYGONCCW</a>      | Geospatial Spatial Relationships | Returns TRUE if an input LINESTRING has starting and ending points that are equal.                                                                                                                                               |
| <a href="#">ST_ISPOLYGONCW</a>       | Geospatial Spatial Relationships | Returns TRUE if an input POLYGON has an exterior that is clockwise.                                                                                                                                                              |
| <a href="#">ST_ISRING</a>            | Geospatial Spatial Relationships | Returns TRUE if the specified LINESTRING is closed and does not intersect itself.                                                                                                                                                |
| <a href="#">ST_ISSIMPLE</a>          | Geospatial Spatial Relationships | For LINESTRING values, this function returns FALSE if any line segments intersect anywhere besides the endpoints. For POLYGON values, the function returns FALSE if either the exterior ring or any interior hole is not simple. |
| <a href="#">ST_ISVALID</a>           | Geospatial Spatial Relationships | Returns TRUE if the specified geospatial value is a well-formed and valid geography according to the OGC standards.                                                                                                              |

| Function Name                             | Category Name                     | Function Description                                                                                                                                                            |
|-------------------------------------------|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ST_LENGTH</a>                 | Geospatial Spatial Measurement    | Returns the length of the specified line in the specified measurement unit.                                                                                                     |
| <a href="#">ST_LENGTH2D</a>               | Geospatial Spatial Measurement    | Alias for ST_LENGTH.                                                                                                                                                            |
| <a href="#">ST_LINEFROMEWKT</a>           | Geospatial Linestring Constructor | Creates a LINESTRING from the specified CHAR.                                                                                                                                   |
| <a href="#">ST_LINEFROMGEOJSON</a>        | Geospatial Linestring Constructor | Creates a LINESTRING represented by the specified GeoJSON.                                                                                                                      |
| <a href="#">ST_LINEFROMTEXT</a>           | Geospatial Linestring Constructor | Creates a LINESTRING from a specified CHAR. The CHAR must be a LINESTRING value in WKT format.                                                                                  |
| <a href="#">ST_LINEFROMWKB</a>            | Geospatial Linestring Constructor | Creates a LINESTRING from the specified BINARY. The BINARY value must be a LINESTRING in WKB format.                                                                            |
| <a href="#">ST_LINEGETALLTIMESATPOINT</a> | Geospatial Spatiotemporal Measu   | Returns a timestamp array of all times when the specified LINESTRING value intersects the specified POINT value.                                                                |
| <a href="#">ST_LINEGETPOINTATTIME</a>     | Geospatial Spatiotemporal Measu   | Returns a POINT within the bounds of the specified LINESTRING that corresponds to the interpolated point at the specified TIMESTAMP value.                                      |
| <a href="#">ST_LINEGETTIMEATPOINT</a>     | Geospatial Spatiotemporal Measu   | Returns the interpolated time of the specified POINT on the specified LINESTRING that is paired with a TIMESTAMP ARRAY.                                                         |
| <a href="#">ST_LINEINTERPOLATEPOINT</a>   | Geospatial Linestring Constructor | Returns a POINT along a LINESTRING based on a specified fraction of its total length.                                                                                           |
| <a href="#">ST_LINELOCATEPOINT</a>        | Geospatial Linestring Constructor | Similar to ST_LINEINTERPOLATEPOINT, this function computes a fraction based on where a specified POINT is located along the length of a specified LINESTRING.                   |
| <a href="#">ST_LINESTRING</a>             | Geospatial Linestring Constructor | Creates a LINESTRING based on the specified inputs.                                                                                                                             |
| <a href="#">ST_LINESUBSTRING</a>          | Geospatial Linestring Constructor | Returns a LINESTRING that is a substring of a specified line that starts and ends at the specified fractions of its total length.                                               |
| <a href="#">ST_LONGESTLINE</a>            | Geospatial Spatial Operators      | Returns the longest LINESTRING between two given geospatial arguments.                                                                                                          |
| <a href="#">ST_LONGESTLINE</a>            | Geospatial Spatiotemporal Measu   | With the specified two LINESTRING-TIMESTAMP ARRAY pairs, this function returns a two-point LINESTRING that represents the maximum distance between points at a concurrent time. |

| Function Name                               | Category Name                     | Function Description                                                                                                                                                                  |
|---------------------------------------------|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ST_MAKEENVELOPE</a>             | Geospatial Spatial Operators      | Returns a POLYGON with vertices that represent the minimum bounding box for the specified coordinates.                                                                                |
| <a href="#">ST_MAKELINE</a>                 | Geospatial Linestring Constructor | Alias for ST_LINESTRING.                                                                                                                                                              |
| <a href="#">ST_MAKEPOINT</a>                | Geospatial Point Constructors     | Alias for ST_POINT.                                                                                                                                                                   |
| <a href="#">ST_MAKEPOLYGON</a>              | Geospatial Polygon Constructors   | Alias for ST_POLYGON.                                                                                                                                                                 |
| <a href="#">ST_MAXDISTANCE</a>              | Geospatial Spatial Measurement    | Returns maximum distance between the specified arguments.                                                                                                                             |
| <a href="#">ST_MAXDISTANCE</a>              | Geospatial Spatiotemporal Measu   | Returns the two-dimensional interpolated maximum cote temporal distance between two LINESTRING-TIMESTAMP array pairs in the specified unit of measurement.                            |
| <a href="#">ST_MEMSIZE</a>                  | Geospatial Attribute Functions    | Returns an INTEGER representing the number of bytes in memory required to store the specified geography.                                                                              |
| <a href="#">ST_MINIMUMBOUNDINGCIRCLE</a>    | Geospatial Spatial Operators      | Returns the smallest circle POLYGON that contains the specified geographic object.                                                                                                    |
| <a href="#">ST_MULTIDIFFERENCEARRAY</a>     | Geospatial Spatial Operators      | Returns an array of geographies that represents the parts of the union of the geographies in the first array that do not intersect with the union of geographies in the second array. |
| <a href="#">ST_MINIMUMDISTANCETOSURFACE</a> | Geospatial Spatial Measurement    | Calculates the shortest distance between any point along a Euclidean line segment in three-dimensional space and the surface of the Earth.                                            |
| <a href="#">ST_MULTIINTERSECTIONARRAY</a>   | Geospatial Spatial Operators      | When you specify two arrays of geospatial objects, this function returns an array of any intersections.                                                                               |
| <a href="#">ST_MULTISYMDIFFERENCEARRAY</a>  | Geospatial Spatial Operators      | When you specify two arrays of geospatial objects, this function returns an array of any geospatial values that do not intersect.                                                     |
| <a href="#">ST_MULTIUNIONARRAY</a>          | Geospatial Spatial Operators      | When you specify two arrays of geospatial objects, this function returns one array that represents a union of all geographies in both arrays.                                         |
| <a href="#">ST_NDIMENSION</a>               | Geospatial Attribute Functions    | Alias for ST_COORDDIM.                                                                                                                                                                |
| <a href="#">ST_NDIMS</a>                    | Geospatial Attribute Functions    | Alias for ST_COORDDIM.                                                                                                                                                                |
| <a href="#">ST_NPOINTS</a>                  | Geospatial Attribute Functions    | Returns an INTEGER representing the number of POINT values in a specified geography.                                                                                                  |
| <a href="#">ST_NRINGS</a>                   | Geospatial Spatial Operators      | Returns the number of rings of the specified POLYGON, including both interior and exterior rings.                                                                                     |
| <a href="#">ST_NUMINTERIORRING</a>          | Geospatial Spatial Operators      | Returns the number of interior rings of the specified POLYGON.                                                                                                                        |

| Function Name                         | Category Name                     | Function Description                                                                                                                        |
|---------------------------------------|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ST_NUMINTERIORRINGS</a>   | Geospatial Spatial Operators      | Returns the number of interior rings of the specified POLYGON.                                                                              |
| <a href="#">ST_NUMPOINTS</a>          | Geospatial Attribute Functions    | Returns an INTEGER representing the number of POINT values in a specified geography.                                                        |
| <a href="#">ST_OVERLAPS</a>           | Geospatial Spatial Relationships  | Returns TRUE if both geographic arguments are of the same dimension and they intersect each other, but neither contains the other.          |
| <a href="#">ST_PERIMETER</a>          | Geospatial Spatial Measurement    | Returns the length of the exterior (outer ring) of the POLYGON in the specified unit of measurement.                                        |
| <a href="#">ST_PERIMETER2D</a>        | Geospatial Spatial Measurement    | Alias for ST_PERIMETER.                                                                                                                     |
| <a href="#">ST_POINT</a>              | Geospatial Point Constructors     | Creates a POINT from the specified input arguments.                                                                                         |
| <a href="#">ST_POINTFROMEWKT</a>      | Geospatial Point Constructors     | Alias for ST_POINT. Creates a POINT using an EWKT-formatted CHAR as an input argument.                                                      |
| <a href="#">ST_POINTFROMGEOHASH</a>   | Geospatial Point Constructors     | Creates a POINT represented by the specified geohash.                                                                                       |
| <a href="#">ST_POINTFROMGEOJSON</a>   | Geospatial Point Constructors     | Creates a POINT represented by the specified GeoJSON value as an input argument.                                                            |
| <a href="#">ST_POINTFROMTEXT</a>      | Geospatial Point Constructors     | Alias for ST_POINT(char).                                                                                                                   |
| <a href="#">ST_POINTFROMWKB</a>       | Geospatial Point Constructors     | Alias for ST_POINT(binary).                                                                                                                 |
| <a href="#">ST_POINTINSIDECIRCLE</a>  | Geospatial Spatial Relationships  | Returns TRUE if the geographic object is inside a circle that is centered at the specified point coordinates and with the specified radius. |
| <a href="#">ST_POINTN</a>             | Geospatial Linestring Constructor | Returns the POINT value at a specified index of the given LINESTRING.                                                                       |
| <a href="#">ST_POINTONSURFACE</a>     | Geospatial Spatial Operators      | Returns a POINT guaranteed to intersect the specified geospatial object.                                                                    |
| <a href="#">ST_POLYGON</a>            | Geospatial Polygon Constructors   | Creates a POLYGON.                                                                                                                          |
| <a href="#">ST_POLYGONFROMEWKT</a>    | Geospatial Polygon Constructors   | Creates a POLYGON using an EWKT-formatted CHAR as an input argument. Alias for the ST_POLYGON constructor.                                  |
| <a href="#">ST_POLYGONFROMGEOJSON</a> | Geospatial Polygon Constructors   | Creates a POLYGON from the specified POINT, POINT array, LINESTRING, or POLYGON geography.                                                  |
| <a href="#">ST_POLYGONFROMTEXT</a>    | Geospatial Polygon Constructors   | Alias for ST_POLYGON(char).                                                                                                                 |
| <a href="#">ST_POLYGONFROMWKB</a>     | Geospatial Polygon Constructors   | Alias for ST_POLYGON(binary).                                                                                                               |
| <a href="#">ST_PROJECT</a>            | Geospatial Spatial Operators      | Returns a POINT by projecting a distance and an azimuth value from the specified starting POINT value.                                      |
| <a href="#">ST_REDUCEPRECISION</a>    | Geospatial Spatial Operators      | Returns a new geospatial object with all POINT values rounded to the specified decimal precision.                                           |

| Function Name                                 | Category Name                     | Function Description                                                                                                                                                                |
|-----------------------------------------------|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ST_RELATE</a>                     | Geospatial Spatial Relationships  | Returns the DE-9IM intersection string that represents the nature of the intersection with the specified geographies.                                                               |
| <a href="#">ST_REMOVEPOINT</a>                | Geospatial Linestring Constructor | Removes a POINT value at a specified index from the specified line.                                                                                                                 |
| <a href="#">ST_REMOVEREPEATEDPOINTS</a>       | Geospatial Spatial Operators      | Returns a new geospatial object with no repeated POINT values.                                                                                                                      |
| <a href="#">ST_REVERSE</a>                    | Geospatial Spatial Operators      | Returns a new geospatial object with the vertexes reversed.                                                                                                                         |
| <a href="#">ST_SEGMENTIZE</a>                 | Geospatial Spatial Operators      | Returns a geospatial object that the function modifies to have no segment longer than the specified max_segment_length in meters.                                                   |
| <a href="#">ST_SETPPOINT</a>                  | Geospatial Linestring Constructor | Replaces a POINT value in a given LINESTRING at a specified index. The function returns the altered LINESTRING with the replaced point.                                             |
| <a href="#">ST_SHORTESTLINE</a>               | Geospatial Spatial Operators      | Returns the shortest LINESTRING between two specified geospatial arguments.                                                                                                         |
| <a href="#">ST_SHORTESTLINE</a>               | Geospatial Spatiotemporal Measu   | When you specify two LINESTRING-TIMESTAMP ARRAY pairs, this function returns a LINESTRING with two points that represents the minimum distance between points at a concurrent time. |
| <a href="#">ST_SIMPLIFY</a>                   | Geospatial Spatial Operators      | Returns a simplified version of the specified geography, which is either a POINT or LINESTRING.                                                                                     |
| <a href="#">ST_SIMPLIFYARRAY</a>              | Geospatial Spatial Operators      | Returns a POLYGON array that represents a simplified version of the specified geography, which is either a POINT, LINESTRING, or POLYGON.                                           |
| <a href="#">ST_SNAPTOGRID</a>                 | Geospatial Spatial Operators      | Returns a new geography value with all POINT values rounded to the specified precisions.                                                                                            |
| <a href="#">ST_SRID</a>                       | Geospatial Attribute Functions    | Returns the EPSG code of the spatial reference identifier (SRID) of the input geography.                                                                                            |
| <a href="#">ST_STARTPOINT</a>                 | Geospatial Linestring Constructor | Returns the starting POINT value of the line.                                                                                                                                       |
| <a href="#">ST_SYMDIFFERENCEARRAY</a>         | Geospatial Spatial Operators      | Returns a geographic array that contains the parts that are not common between two geographic objects, geo1 and geo2.                                                               |
| <a href="#">ST_TOTALSECONDSININTERSECTION</a> | Geospatial Spatiotemporal Measu   | Returns the total number of seconds spent in the intersection result calculated by the spatiotemporal version of ST_INTERSECTION.                                                   |



| Function Name                   | Category Name                    | Function Description                                                                                                       |
|---------------------------------|----------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ST_TOUCHES</a>      | Geospatial Spatial Relationships | Returns TRUE if the only POINT values in common between the two geographic arguments lie in the union of their boundaries. |
| <a href="#">ST_UNIONARRAY</a>   | Geospatial Spatial Operators     | Performs a union of the input geography values to produce a geographic array.                                              |
| <a href="#">ST_WHOLEEARTH</a>   | Geospatial Polygon Constructors  | Returns the database internal representation of the whole earth polygon.                                                   |
| <a href="#">ST_WITHIN</a>       | Geospatial Spatial Relationships | Alias for ST_CONTAINS.                                                                                                     |
| <a href="#">ST_X</a>            | Geospatial Attribute Functions   | Returns the x value of the specified POINT.                                                                                |
| <a href="#">ST_XMAX</a>         | Geospatial Attribute Functions   | Returns the maximum x value of the specified geography.                                                                    |
| <a href="#">ST_XMIN</a>         | Geospatial Attribute Functions   | Returns the minimum x value of the specified geography.                                                                    |
| <a href="#">ST_Y</a>            | Geospatial Attribute Functions   | Returns the y value of the specified POINT.                                                                                |
| <a href="#">ST_YMAX</a>         | Geospatial Attribute Functions   | Returns the maximum y value of specified geography.                                                                        |
| <a href="#">ST_YMIN</a>         | Geospatial Attribute Functions   | Returns the minimum y value of specified geography.                                                                        |
| <a href="#">STARTSWITH</a>      | Character and Binary Functions   | Returns true if string starts with substring and false otherwise.                                                          |
| <a href="#">STDDEV</a>          | Aggregate Functions              | Alias for STDEV.                                                                                                           |
| <a href="#">STDDEV_POP</a>      | Aggregate Functions              | Alias for STDEVP.                                                                                                          |
| <a href="#">STDDEV_SAMP</a>     | Aggregate Functions              | Alias for STDEV.                                                                                                           |
| <a href="#">STDEV</a>           | Aggregate Functions              | Sample standard deviation.                                                                                                 |
| <a href="#">STDEVP</a>          | Aggregate Functions              | Population standard deviation.                                                                                             |
| <a href="#">STRING_AGG</a>      | Sorted Aggregate Functions       | Returns a string concatenated from every row from the expression. The delimiter argument is optional.                      |
| <a href="#">STRING_TO_ARRAY</a> | Array Functions                  | Converts the string representation of an array (e.g., 'int[1,2,NULL]') into an array.                                      |
| <a href="#">STRING_TO_TUPLE</a> | Tuple Functions                  | Converts the string representation of a tuple (e.g 'tuple<<INT,BIGINT,CHAR>>(1,2,NULL)') into a tuple.                     |
| <a href="#">STRPOS</a>          | Character and Binary Functions   | Equivalent to using LOCATE as LOCATE(substring, string). Note the reversed argument order.                                 |
| <a href="#">SUBSTR</a>          | Character and Binary Functions   | Alias for SUBSTRING.                                                                                                       |
| <a href="#">SUBSTRING</a>       | Character and Binary Functions   | Returns the substring of a character or binary value.                                                                      |
| <a href="#">SUBNET</a>          | Network Type Functions           | Computes the prefix from an IP or IPV4 value and the size of the prefix.                                                   |
| <a href="#">SUM</a>             | Aggregate Functions              | Sum over the set.                                                                                                          |
| <a href="#">SVD_DECOMP</a>      | Matrix Functions                 | Returns SVD decomposition of a matrix as a tuple of 3 matrixes.                                                            |
| <a href="#">SWEP</a>            | Aggregate Functions              | The population over the set.                                                                                               |
| <a href="#">TAN</a>             | Math Functions                   | Returns the tangent of x.                                                                                                  |
| <a href="#">TANH</a>            | Math Functions                   | Returns the hyperbolic tangent of x.                                                                                       |



| Function Name      | Category Name                                  | Function Description                                                                                                                                      |
|--------------------|------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| TIME               | Scalar Data Conversion Functions               | Parses the string to create a time value. The string must be in the form 'HH:MM[.SSSSSSSS] '.                                                             |
| TIMESTAMP          | Scalar Data Conversion Functions               | Parses a string and makes a timestamp value. The string must be in the format 'YYYY-MM-DD[HH:MM][.SSSSSSSS] '.                                            |
| TIMESTAMP_TO_NANOS | Date and Time Functions                        | Convert timestamp into nanoseconds after epoch as BIGINT.                                                                                                 |
| TO_ARRAY_LENGTH    | Special Data Pipeline Transformation Functions | Create the specified number of copies of any JSON object in an array.                                                                                     |
| TO_CHAR            | Character and Binary Functions                 | Converts a numeric, date, or timestamp value into a CHAR date type.                                                                                       |
| TO_DATE            | Formatting Functions                           | Converts a character value with the specified format to a DATE type.                                                                                      |
| TO_NUMBER          | Formatting Functions                           | Converts a character value with the specified format to a DECIMAL type.                                                                                   |
| TO_TIMESTAMP       | Formatting Functions                           | Converts a character value with the specified format to a TIMESTAMP type.                                                                                 |
| TRANSFORM          | Special Data Pipeline Transformation Functions | Transforms an array based on the logic in a lambda expression.                                                                                            |
| TRANSLATE          | Character and Binary Functions                 | Replaces specified characters in a provided string with a separate set of characters.                                                                     |
| TRANSPOSE          | Matrix Functions                               | Returns transpose of the matrix.                                                                                                                          |
| TRIM               | Character and Binary Functions                 | Alias for BTRIM. Trim leading and trailing blanks from the string.                                                                                        |
| TRUNC              | Math Functions                                 | Returns x truncated to y decimal places.                                                                                                                  |
| TRUNCATE           | Math Functions                                 | Alias for TRUNC.                                                                                                                                          |
| TUPLE()            | Tuple Functions                                | Construct a tuple with the specified elements. Types of the tuple are inferred from the inner elements.                                                   |
| TUPLE«»            | Tuple Functions                                | Construct a tuple with the specified elements. NULL is also supported as an element.                                                                      |
| TYPE[]             | Array Functions                                | Construct an array of SQL type TYPE giving the elements.                                                                                                  |
| UCASE              | Character and Binary Functions                 | Alias for UPPER.                                                                                                                                          |
| UNNEST             | Array Functions                                | Expand each element in an input array out to an individual row.                                                                                           |
| UPPER              | Character and Binary Functions                 | Convert string to upper case.                                                                                                                             |
| USECS              | Date and Time Functions                        | The seconds part of a time value, including fractional parts, returned as an integer. The function multiplies the seconds part of the value by 1,000,000. |

| Function Name | Category Name                                  | Function Description                                                                                       |
|---------------|------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| UUID          | Scalar Data Conversion Functions               | Parses the string and makes a Universally Unique Identifier (UUID) value. The string must be a valid UUID. |
| VAR_POP       | Aggregate Functions                            | Alias for VARIANCEP.                                                                                       |
| VAR_SAMP      | Aggregate Functions                            | Alias for VARIANCE.                                                                                        |
| VARIANCE      | Aggregate Functions                            | Sample variance.                                                                                           |
| VARIANCEP     | Aggregate Functions                            | Population variance.                                                                                       |
| VECTOR_ARGMAX | Matrix Functions                               | Returns the argmax of a one-dimensional matrix or vector.                                                  |
| VECTOR_ARGMIN | Matrix Functions                               | Returns the argmin of a one-dimensional matrix or vector.                                                  |
| VECTOR_MAX    | Matrix Functions                               | Returns the maximum of elements in a one-dimensional matrix/vector.                                        |
| VECTOR_MIN    | Matrix Functions                               | Returns the minimum of elements in a one-dimensional matrix/vector.                                        |
| VECTOR_SUM    | Matrix Functions                               | Returns the sum of elements in a one-dimensional matrix/vector.                                            |
| VERSION       | System Functions                               | Returns the version of the database to which the client is currently connected.                            |
| WEEK          | Date and Time Functions                        | Returns the ISO-8601 week number, as an integer, of the specified timestamp or date value.                 |
| WEEKS         | Scalar Data Conversion Functions               | Converts an integral value to an interval value of type weeks to be used in date calculations.             |
| WIDTH_BUCKET  | Special Data Pipeline Transformation Functions | Creates N equal-width buckets in the range [min,max) as a histogram.                                       |
| YEAR          | Date and Time Functions                        | Extracts the year portion of a timestamp or date as an integer.                                            |
| YEARS         | Scalar Data Conversion Functions               | Converts an integral value to an interval value of type years.                                             |
| ZERO_MATRIX   | Matrix Functions                               | Returns a zero matrix of the specified (row, col) dimensions.                                              |
| ZIP_WITH      | Special Data Pipeline Transformation Functions | Merges two or more input arrays, element-wise, into a single array using the trailing combining function.  |
| ZN            | Conditional Functions                          | If x is NULL, returns 0. Otherwise, returns x.                                                             |
| ZSCORE        | Window Aggregate Functions                     | Zscore of the sample based on the stddev() function.                                                       |
| ZSCOREP       | Window Aggregate Functions                     | Zscore of the sample based on the stddevp() function.                                                      |

## 10.2 Categorized SQL Functions List

| Function Name | Category Name       | Function Description                                        |
|---------------|---------------------|-------------------------------------------------------------|
| ANY_VALUE     | Aggregate Functions | Returns an arbitrary, non-NULL value from the input column. |

| Function Name                         | Category Name       | Function Description                                                                                         |
|---------------------------------------|---------------------|--------------------------------------------------------------------------------------------------------------|
| <a href="#">APPROX_COUNT_DISTINCT</a> | Aggregate Functions | Approximate distinct count by using hyper-log-log (95% confidence interval that the value is within 4.5%).   |
| <a href="#">APPROX_SUM</a>            | Aggregate Functions | Allows the aggregation engine to use a faster, non-deterministic ordering to summate floating-point columns. |
| <a href="#">AVG</a>                   | Aggregate Functions | Average, or arithmetic mean, over the set.                                                                   |
| <a href="#">CORR</a>                  | Aggregate Functions | Alias for CORRELATION.                                                                                       |
| <a href="#">CORRELATION</a>           | Aggregate Functions | Sample correlation.                                                                                          |
| <a href="#">CORRELATIONP</a>          | Aggregate Functions | Population correlation.                                                                                      |
| <a href="#">COUNT</a>                 | Aggregate Functions | Number of rows in the set.                                                                                   |
| <a href="#">COVAR_POP</a>             | Aggregate Functions | Alias for COVARIANCEP.                                                                                       |
| <a href="#">COVAR_SAMP</a>            | Aggregate Functions | Alias for COVARIANCE.                                                                                        |
| <a href="#">COVARIANCE</a>            | Aggregate Functions | Sample covariance.                                                                                           |
| <a href="#">COVARIANCEP</a>           | Aggregate Functions | Population covariance.                                                                                       |
| <a href="#">KURTOSIS</a>              | Aggregate Functions | The sample over the set.                                                                                     |
| <a href="#">KURTOSISP</a>             | Aggregate Functions | The population over the set.                                                                                 |
| <a href="#">MAX</a>                   | Aggregate Functions | Maximum value in the specified column.                                                                       |
| <a href="#">MIN</a>                   | Aggregate Functions | Minimum value in the specified column.                                                                       |
| <a href="#">PRODUCT</a>               | Aggregate Functions | Product over the set.                                                                                        |
| <a href="#">SKEW</a>                  | Aggregate Functions | The sample over the set.                                                                                     |
| <a href="#">STDDEV</a>                | Aggregate Functions | Alias for STDEV.                                                                                             |
| <a href="#">STDDEV_POP</a>            | Aggregate Functions | Alias for STDEVP.                                                                                            |
| <a href="#">STDDEV_SAMP</a>           | Aggregate Functions | Alias for STDEV.                                                                                             |
| <a href="#">STDEV</a>                 | Aggregate Functions | Sample standard deviation.                                                                                   |
| <a href="#">STDEVP</a>                | Aggregate Functions | Population standard deviation.                                                                               |
| <a href="#">SUM</a>                   | Aggregate Functions | Sum over the set.                                                                                            |
| <a href="#">SWEP</a>                  | Aggregate Functions | The population over the set.                                                                                 |
| <a href="#">VAR_POP</a>               | Aggregate Functions | Alias for VARIANCEP.                                                                                         |
| <a href="#">VAR_SAMP</a>              | Aggregate Functions | Alias for VARIANCE.                                                                                          |
| <a href="#">VARIANCE</a>              | Aggregate Functions | Sample variance.                                                                                             |
| <a href="#">VARIANCEP</a>             | Aggregate Functions | Population variance.                                                                                         |
| <a href="#">ARRAY[]</a>               | Array Functions     | PostgreSQL®-compliant constructor. The type of the array is deduced from the elements.                       |
| <a href="#">ARRAY_APPEND</a>          | Array Functions     | Add value to the back of an array.                                                                           |
| <a href="#">ARRAY_ARGMAX</a>          | Array Functions     | Returns the corresponding argmax of the array as a BIGINT index.                                             |
| <a href="#">ARRAY_ARGMIN</a>          | Array Functions     | Returns the corresponding argmin of the array as a BIGINT index.                                             |
| <a href="#">ARRAY_CAT</a>             | Array Functions     | Concatenate 2 arrays into a new one.                                                                         |
| <a href="#">ARRAY_LENGTH</a>          | Array Functions     | Return the number of elements of a given array.                                                              |
| <a href="#">ARRAY_MAX</a>             | Array Functions     | Returns the corresponding maximum of the array.                                                              |
| <a href="#">ARRAY_MIN</a>             | Array Functions     | Returns the corresponding minimum of the array.                                                              |
| <a href="#">ARRAY_POSITION</a>        | Array Functions     | Returns the position of the first matching scalar in the array.                                              |

| Function Name          | Category Name                  | Function Description                                                                                                                                  |
|------------------------|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| ARRAY_POSITIONS        | Array Functions                | Returns all elements stored in the right array, and return their respective positions in the left array.                                              |
| ARRAY_PREPEND          | Array Functions                | Add value to the front of an array.                                                                                                                   |
| ARRAY_REMOVE           | Array Functions                | remove value from the array.                                                                                                                          |
| ARRAY_REPLACE          | Array Functions                | replace value by another in an array.                                                                                                                 |
| ARRAY_SUM              | Array Functions                | Returns the sum of the array. The array must be one-dimensional and contain numeric values. NULL values do not contribute to the sum.                 |
| ARRAY_TO_STRING        | Array Functions                | Converts array to a list of elements separated by 'delimiter'.                                                                                        |
| CAST_TO_ARRAY          | Array Functions                | Casts the elements of the array to another type.                                                                                                      |
| CHAR                   | Array Functions                | Converts array to its string representation.                                                                                                          |
| CROSS_ENTROPY_LOSS     | Array Functions                | Returns the cross entropy loss of two arrays.                                                                                                         |
| HINGE_LOSS             | Array Functions                | Returns the hinge loss of two arrays.                                                                                                                 |
| LOG_LOSS               | Array Functions                | Returns the log loss of two arrays.                                                                                                                   |
| LOGITS_LOSS            | Array Functions                | Returns the logits loss of two arrays.                                                                                                                |
| SOFTMAX                | Array Functions                | Returns the softmax of the array.                                                                                                                     |
| STRING_TO_ARRAY        | Array Functions                | Converts the string representation of an array (e.g., 'int[1,2,NULL]') into an array.                                                                 |
| TYPE[]                 | Array Functions                | Construct an array of SQL type TYPE giving the elements.                                                                                              |
| ASCII                  | Character and Binary Functions | Returns the ASCII code value of the leftmost character of the character value.                                                                        |
| BIT_LENGTH             | Character and Binary Functions | Returns the length of the character value in bits.                                                                                                    |
| BTRIM                  | Character and Binary Functions | Alias for TRIM.                                                                                                                                       |
| CHAR_LENGTH            | Character and Binary Functions | Alias for LENGTH.                                                                                                                                     |
| CHARACTER_LENGTH       | Character and Binary Functions | Alias for LENGTH.                                                                                                                                     |
| CHR                    | Character and Binary Functions | Converts an integer value to a string.                                                                                                                |
| CONCAT                 | Character and Binary Functions | Concatenates two values, which must both be either binary, hash, or string data types. This function is equivalent to the    operator.                |
| ENDSWITH               | Character and Binary Functions | Returns true if x ends with y and false otherwise.                                                                                                    |
| INITCAP                | Character and Binary Functions | For each word in the provided string, capitalize the first character if it is alphabetic.                                                             |
| INSTR                  | Character and Binary Functions | Returns the index position of the first occurrence where the character value char_substring appears in the character value char by ignoring the case. |
| JSON_EXTRACT_PATH_TEXT | Character and Binary Functions | Returns the value for the key-value pair referenced by a series of path elements in a JSON string.                                                    |
| LCASE                  | Character and Binary Functions | Alias for LOWER.                                                                                                                                      |

| Function Name  | Category Name                  | Function Description                                                                                                                                            |
|----------------|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LEFT           | Character and Binary Functions | Return the number of characters in the string equal to the value integer. If the integer is negative, the function returns all but the last integer characters. |
| LENGTH         | Character and Binary Functions | For character data types, this value is in terms of characters. For binary data types, this value is in terms of bytes.                                         |
| LOCATE         | Character and Binary Functions | Alias for POSITION. Returns the index position of the first occurrence of the character value substring in character value string.                              |
| LOWER          | Character and Binary Functions | Alias for LCASE. Convert string to lowercase.                                                                                                                   |
| LPAD           | Character and Binary Functions | Pad the input text to the specified length with the pad string on the left side.                                                                                |
| LTRIM          | Character and Binary Functions | Removes leading blanks from the string value string.                                                                                                            |
| MD5            | Character and Binary Functions | Returns the hexadecimal string (all lowercase) representing the md5 hash of char.                                                                               |
| MID            | Character and Binary Functions | Alias for SUBSTRING.                                                                                                                                            |
| OCTET_LENGTH   | Character and Binary Functions | Returns the length in bytes of a character or binary value.                                                                                                     |
| REGEXP_COUNT   | Character and Binary Functions | Searches a string for all occurrences of a regular expression pattern.                                                                                          |
| REGEXP_INSTR   | Character and Binary Functions | Searches a string using a regular expression pattern and returns an integer representing the start position or end position of the substring that matches.      |
| REGEXP_REPLACE | Character and Binary Functions | Searches a string for all occurrences of a regular expression pattern.                                                                                          |
| REGEXP_SUBSTR  | Character and Binary Functions | Returns one substring from a string that matches a specified regular expression pattern.                                                                        |
| POSITION       | Character and Binary Functions | Alias for LOCATE.                                                                                                                                               |
| REPEAT         | Character and Binary Functions | Repeats the character value char a number of times equal to num.                                                                                                |
| REPLACE        | Character and Binary Functions | Replaces all occurrences of substr_to_remove in the character value string with substr_to_replace.                                                              |
| REVERSE        | Character and Binary Functions | Reverse the input string.                                                                                                                                       |
| RIGHT          | Character and Binary Functions | Return the number of trailing characters in the string equal to the value integer.                                                                              |
| RPAD           | Character and Binary Functions | Pad the input text to the specified length with the pad string on the right side.                                                                               |
| RSUBSTRING     | Character and Binary Functions | Returns the substring from the right side of a string based on a specified length.                                                                              |

| Function Name | Category Name                  | Function Description                                                                                                                                                  |
|---------------|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RTRIM         | Character and Binary Functions | Removes leading blanks from the string value string.                                                                                                                  |
| SHA1          | Character and Binary Functions | Uses the SHA-1 cryptographic hash function to convert a string into a 40-character string representing the hexadecimal value of a 160-bit checksum.                   |
| SPACE         | Character and Binary Functions | Returns a string of repeated spaces equal to the number value, repeat.                                                                                                |
| SPLIT_PART    | Character and Binary Functions | Split the value string based on the delimiter value. The function returns a substring from the split operation based on the index value (starting from 1).            |
| STARTSWITH    | Character and Binary Functions | Returns true if string starts with substring and false otherwise.                                                                                                     |
| STRPOS        | Character and Binary Functions | Equivalent to using LOCATE as LOCATE(substring, string). Note the reversed argument order.                                                                            |
| SUBSTR        | Character and Binary Functions | Alias for SUBSTRING.                                                                                                                                                  |
| SUBSTRING     | Character and Binary Functions | Returns the substring of a character or binary value.                                                                                                                 |
| TO_CHAR       | Character and Binary Functions | Converts a numeric, date, or timestamp value into a CHAR date type.                                                                                                   |
| TRANSLATE     | Character and Binary Functions | Replaces specified characters in a provided string with a separate set of characters.                                                                                 |
| TRIM          | Character and Binary Functions | Alias for BTRIM. Trim leading and trailing blanks from the string.                                                                                                    |
| UCASE         | Character and Binary Functions | Alias for UPPER.                                                                                                                                                      |
| UPPER         | Character and Binary Functions | Convert string to upper case.                                                                                                                                         |
| CASE          | Conditional Functions          | CASE operates similarly to conditional scripting in other programming languages, allowing it to function like an if / then / else statement or as a switch statement. |
| COALESCE      | Conditional Functions          | Evaluates to the first argument that is not NULL, or NULL if all arguments are NULL.                                                                                  |
| GREATEST      | Conditional Functions          | Returns the largest non-NULL value of all the arguments, or NULL if all the arguments are NULL.                                                                       |
| IF_NULL       | Conditional Functions          | Alias for COALESCE.                                                                                                                                                   |
| LEAST         | Conditional Functions          | Returns the smallest non-NULL value of all arguments, or NULL if all arguments are NULL.                                                                              |
| MURMUR3       | Conditional Functions          | Returns a 32-bit MurmurHash3 hash of the input value as an INTEGER data type.                                                                                         |
| NULL_IF       | Conditional Functions          | Returns the NULL value if two arguments are equal; otherwise, returns the first argument.                                                                             |

| Function Name                     | Category Name           | Function Description                                                                                                                       |
|-----------------------------------|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ZN</a>                | Conditional Functions   | If x is NULL, returns 0. Otherwise, returns x.                                                                                             |
| <a href="#">ADD_MONTHS</a>        | Date and Time Functions | Adds the specified number of months to the date.                                                                                           |
| <a href="#">CENTURY</a>           | Date and Time Functions | Returns the number of centuries.                                                                                                           |
| <a href="#">CURDATE</a>           | Date and Time Functions | Alias for CURRENT_DATE.                                                                                                                    |
| <a href="#">CURRENT_DATE</a>      | Date and Time Functions | Returns the current date in the format YYYY-MM-DD.                                                                                         |
| <a href="#">CURRENT_TIMESTAMP</a> | Date and Time Functions | Returns the current date and time as a TIMESTAMP value (e.g., YYYY-MM-DD hhss.mmm).                                                        |
| <a href="#">DATE_PART</a>         | Date and Time Functions | Alias for EXTRACT.                                                                                                                         |
| <a href="#">DATE_TRUNC</a>        | Date and Time Functions | Returns the date or timestamp entered, truncated to the specified precision.                                                               |
| <a href="#">DATEADD</a>           | Date and Time Functions | Adds a specified number value (as a signed integer) to a specified date part of an input date value, and then returns that modified value. |
| <a href="#">DATEDIFF</a>          | Date and Time Functions | This function returns an INT representing the difference between two date or time values in a specified date or time unit.                 |
| <a href="#">DAY</a>               | Date and Time Functions | Alias for DAY_OF_MONTH.                                                                                                                    |
| <a href="#">DAY_OF_MONTH</a>      | Date and Time Functions | Extracts the day-of-month portion of a timestamp or date as an integer.                                                                    |
| <a href="#">DAY_OF_WEEK</a>       | Date and Time Functions | Returns an integer, in the range of 1 to 7, that represents the day of the week.                                                           |
| <a href="#">DAY_OF_YEAR</a>       | Date and Time Functions | Returns an integer in the range 1 to 366 that represents the day of the year.                                                              |
| <a href="#">DECADE</a>            | Date and Time Functions | The decade is the year divided by 10.                                                                                                      |
| <a href="#">DOW</a>               | Date and Time Functions | Alias for DAY_OF_WEEK.                                                                                                                     |
| <a href="#">DOY</a>               | Date and Time Functions | Alias for DAY_OF_YEAR.                                                                                                                     |
| <a href="#">EOMONTH</a>           | Date and Time Functions | Returns the last day of the timestamp or date.                                                                                             |
| <a href="#">EPOCH</a>             | Date and Time Functions | The number of seconds after 1970-01-01 00:00:00 UTC.                                                                                       |
| <a href="#">EXTRACT</a>           | Date and Time Functions | Extract a component from a timestamp or date.                                                                                              |
| <a href="#">HOUR</a>              | Date and Time Functions | Extracts the hour portion of a timestamp as an integer.                                                                                    |
| <a href="#">ISDATE</a>            | Date and Time Functions | Returns TRUE if the input argument can be successfully cast to a date.                                                                     |
| <a href="#">ISODOW</a>            | Date and Time Functions | Extracts the day of the week based on ISO 8601, which ranges from Monday (1) to Sunday (7).                                                |
| <a href="#">MAKEDATETIME</a>      | Date and Time Functions | Returns a timestamp consisting of the specified date and time.                                                                             |
| <a href="#">MILLISECOND</a>       | Date and Time Functions | Extracts the millisecond portion of a timestamp as an integer.                                                                             |



| Function Name      | Category Name                  | Function Description                                                                                                                                      |
|--------------------|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| MINUTE             | Date and Time Functions        | Extracts the minute portion of a timestamp or date as an integer.                                                                                         |
| MONTH              | Date and Time Functions        | Extracts the month portion of a timestamp or date as an integer.                                                                                          |
| MONTH_NAME         | Date and Time Functions        | Returns the calendar name in English of the month for the specified date.                                                                                 |
| MONTHS_BETWEEN     | Date and Time Functions        | Returns the difference between the two dates or timestamps in months as a DOUBLE.                                                                         |
| MSECS              | Date and Time Functions        | The seconds field, including fractional parts. The function multiplies the seconds part of the value by 1,000.                                            |
| NANOS_TO_TIMESTAMP | Date and Time Functions        | Convert a number of nanoseconds into a timestamp equivalent to the duration after the epoch time.                                                         |
| NEXT_DAY           | Date and Time Functions        | Returns the closest date after a specified date that lies on a specific day of the week.                                                                  |
| NOW                | Date and Time Functions        | Alias for CURRENT_TIMESTAMP.                                                                                                                              |
| QUARTER            | Date and Time Functions        | Returns an integer between 1 and 4 that represents the quarter of the year in which the specified date falls.                                             |
| ROUND              | Date and Time Functions        | Returns the specified date or timestamp, rounded to the specified precision.                                                                              |
| SECOND             | Date and Time Functions        | Extracts the seconds portion of a timestamp as an integer.                                                                                                |
| TIMESTAMP_TO_NANOS | Date and Time Functions        | Convert timestamp into nanoseconds after epoch as BIGINT.                                                                                                 |
| USECS              | Date and Time Functions        | The seconds part of a time value, including fractional parts, returned as an integer. The function multiplies the seconds part of the value by 1,000,000. |
| WEEK               | Date and Time Functions        | Returns the ISO-8601 week number, as an integer, of the specified timestamp or date value.                                                                |
| YEAR               | Date and Time Functions        | Extracts the year portion of a timestamp or date as an integer.                                                                                           |
| TO_DATE            | Formatting Functions           | Converts a character value with the specified format to a DATE type.                                                                                      |
| TO_NUMBER          | Formatting Functions           | Converts a character value with the specified format to a DECIMAL type.                                                                                   |
| TO_TIMESTAMP       | Formatting Functions           | Converts a character value with the specified format to a TIMESTAMP type.                                                                                 |
| ST_COORDDIM        | Geospatial Attribute Functions | Alias for ST_NDIMS or ST_NDIMENSION. Returns an INTEGER of the coordinate dimension of the specified geography.                                           |



| Function Name                   | Category Name                   | Function Description                                                                                     |
|---------------------------------|---------------------------------|----------------------------------------------------------------------------------------------------------|
| <a href="#">ST_DIMENSION</a>    | Geospatial Attribute Functions  | Returns an INTEGER that represents the dimension of the specified geography.                             |
| <a href="#">ST_GEOMETRYTYPE</a> | Geospatial Attribute Functions  | Returns a string representing the geometry type of the input value.                                      |
| <a href="#">ST_ISEMPTY</a>      | Geospatial Attribute Functions  | Returns TRUE if the specified geography value is empty, such as 'POLYGON EMPTY'.                         |
| <a href="#">ST_MEMSIZE</a>      | Geospatial Attribute Functions  | Returns an INTEGER representing the number of bytes in memory required to store the specified geography. |
| <a href="#">ST_NDIMENSION</a>   | Geospatial Attribute Functions  | Alias for ST_COORDDIM.                                                                                   |
| <a href="#">ST_NDIMS</a>        | Geospatial Attribute Functions  | Alias for ST_COORDDIM.                                                                                   |
| <a href="#">ST_NPOINTS</a>      | Geospatial Attribute Functions  | Returns an INTEGER representing the number of POINT values in a specified geography.                     |
| <a href="#">ST_NUMPOINTS</a>    | Geospatial Attribute Functions  | Returns an INTEGER representing the number of POINT values in a specified geography.                     |
| <a href="#">ST_SRID</a>         | Geospatial Attribute Functions  | Returns the EPSG code of the spatial reference identifier (SRID) of the input geography.                 |
| <a href="#">ST_X</a>            | Geospatial Attribute Functions  | Returns the x value of the specified POINT.                                                              |
| <a href="#">ST_XMAX</a>         | Geospatial Attribute Functions  | Returns the maximum x value of the specified geography.                                                  |
| <a href="#">ST_XMIN</a>         | Geospatial Attribute Functions  | Returns the minimum x value of the specified geography.                                                  |
| <a href="#">ST_Y</a>            | Geospatial Attribute Functions  | Returns the y value of the specified POINT.                                                              |
| <a href="#">ST_YMAX</a>         | Geospatial Attribute Functions  | Returns the maximum y value of specified geography.                                                      |
| <a href="#">ST_YMIN</a>         | Geospatial Attribute Functions  | Returns the minimum y value of specified geography.                                                      |
| <a href="#">ST_ASBINARY</a>     | Geospatial Conversion Functions | Returns the well-known binary (WKB) representation of the specified geography. Alias of ST_ASWKB.        |
| <a href="#">ST_ASEWKT</a>       | Geospatial Conversion Functions | Returns a string that represents geographic coordinates of a specified POINT in the specified format.    |
| <a href="#">ST_ASGEOJSON</a>    | Geospatial Conversion Functions | Alias of ST_ASBINARY.                                                                                    |
| <a href="#">ST_ASLATLONTEXT</a> | Geospatial Conversion Functions | Returns the GeoJSON representation of the specified geography using the IETF standards.                  |
| <a href="#">ST_ASTEXT</a>       | Geospatial Conversion Functions | Alias of ST_ASTEXT.                                                                                      |
| <a href="#">ST_ASWKB</a>        | Geospatial Conversion Functions | Alias of ST_ASTEXT.                                                                                      |
| <a href="#">ST_ASWKT</a>        | Geospatial Conversion Functions | Alias of ST_ASWKT and ST_EWKT. Returns the WKT representation of the specified geography.                |

| Function Name                           | Category Name                     | Function Description                                                                                                                                          |
|-----------------------------------------|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ST_GEOHASH</a>              | Geospatial Conversion Functions   | Returns a string that represents the geohash of the input POINT.                                                                                              |
| <a href="#">ST_ADDPOINT</a>             | Geospatial Linestring Constructor | Adds a POINT to the specified LINESTRING at the specified 0-indexed location.                                                                                 |
| <a href="#">ST_ENDPOINT</a>             | Geospatial Linestring Constructor | Returns the endpoint of a specified LINESTRING. The returned value is a POINT.                                                                                |
| <a href="#">ST_LINEFROMEWKT</a>         | Geospatial Linestring Constructor | Creates a LINESTRING from the specified CHAR.                                                                                                                 |
| <a href="#">ST_LINEFROMGEOJSON</a>      | Geospatial Linestring Constructor | Creates a LINESTRING represented by the specified GeoJSON.                                                                                                    |
| <a href="#">ST_LINEFROMTEXT</a>         | Geospatial Linestring Constructor | Creates a LINESTRING from a specified CHAR. The CHAR must be a LINESTRING value in WKT format.                                                                |
| <a href="#">ST_LINEFROMWKB</a>          | Geospatial Linestring Constructor | Creates a LINESTRING from the specified BINARY. The BINARY value must be a LINESTRING in WKB format.                                                          |
| <a href="#">ST_LINEINTERPOLATEPOINT</a> | Geospatial Linestring Constructor | Returns a POINT along a LINESTRING based on a specified fraction of its total length.                                                                         |
| <a href="#">ST_LINELOCATEPOINT</a>      | Geospatial Linestring Constructor | Similar to ST_LINEINTERPOLATEPOINT, this function computes a fraction based on where a specified POINT is located along the length of a specified LINESTRING. |
| <a href="#">ST_LINESTRING</a>           | Geospatial Linestring Constructor | Creates a LINESTRING based on the specified inputs.                                                                                                           |
| <a href="#">ST_LINESUBSTRING</a>        | Geospatial Linestring Constructor | Returns a LINESTRING that is a substring of a specified line that starts and ends at the specified fractions of its total length.                             |
| <a href="#">ST_MAKELINE</a>             | Geospatial Linestring Constructor | Alias for ST_LINESTRING.                                                                                                                                      |
| <a href="#">ST_POINTN</a>               | Geospatial Linestring Constructor | Returns the POINT value at a specified index of the specified LINESTRING.                                                                                     |
| <a href="#">ST_REMOVEPOINT</a>          | Geospatial Linestring Constructor | Removes a POINT value at a specified index from the specified line.                                                                                           |
| <a href="#">ST_SETPPOINT</a>            | Geospatial Linestring Constructor | Replaces a POINT value in a specified LINESTRING at a specified index. The function returns the altered LINESTRING with the replaced point.                   |
| <a href="#">ST_STARTPOINT</a>           | Geospatial Linestring Constructor | Returns the starting POINT value of the line.                                                                                                                 |

| Function Name                         | Category Name                   | Function Description                                                                                                                                                                                                         |
|---------------------------------------|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ST_CENTROID</a>           | Geospatial Point Constructors   | The geographic center of mass is calculated by taking the average of all points on a three-dimensional sphere, projecting the resultant point onto the sphere, and converting it back to latitude and longitude coordinates. |
| <a href="#">ST_GEOPOINT</a>           | Geospatial Point Constructors   | Creates and returns a POLYGON geography with a single point, defined by the longitude and latitude specified for the function.                                                                                               |
| <a href="#">ST_MAKEPOINT</a>          | Geospatial Point Constructors   | Alias for ST_POINT.                                                                                                                                                                                                          |
| <a href="#">ST_POINT</a>              | Geospatial Point Constructors   | Creates a POINT from the specified input arguments.                                                                                                                                                                          |
| <a href="#">ST_POINTFROMEWKT</a>      | Geospatial Point Constructors   | Alias for ST_POINT. Creates a POINT using an EWKT-formatted CHAR as an input argument.                                                                                                                                       |
| <a href="#">ST_POINTFROMGEOHASH</a>   | Geospatial Point Constructors   | Creates a POINT represented by the specified geohash.                                                                                                                                                                        |
| <a href="#">ST_POINTFROMGEOJSON</a>   | Geospatial Point Constructors   | Creates a POINT represented by the specified GeoJSON value as an input argument.                                                                                                                                             |
| <a href="#">ST_POINTFROMTEXT</a>      | Geospatial Point Constructors   | Alias for ST_POINT(char).                                                                                                                                                                                                    |
| <a href="#">ST_POINTFROMWKB</a>       | Geospatial Point Constructors   | Alias for ST_POINT(binary).                                                                                                                                                                                                  |
| <a href="#">ST_FORCECCW</a>           | Geospatial Polygon Constructors | Creates a standardized polygon from an existing one. Ocint® defines standardized as the exterior being counterclockwise (CCW) and all holes being clockwise (CW) oriented.                                                   |
| <a href="#">ST_MAKEPOLYGON</a>        | Geospatial Polygon Constructors | Alias for ST_POLYGON.                                                                                                                                                                                                        |
| <a href="#">ST_POLYGON</a>            | Geospatial Polygon Constructors | Creates a POLYGON.                                                                                                                                                                                                           |
| <a href="#">ST_POLYGONFROMEWKT</a>    | Geospatial Polygon Constructors | Creates a POLYGON using an EWKT-formatted CHAR as an input argument. Alias for the ST_POLYGON constructor.                                                                                                                   |
| <a href="#">ST_POLYGONFROMGEOJSON</a> | Geospatial Polygon Constructors | Creates a POLYGON from the specified POINT, POINT array, LINestring, or POLYGON geography.                                                                                                                                   |
| <a href="#">ST_POLYGONFROMTEXT</a>    | Geospatial Polygon Constructors | Alias for ST_POLYGON(char).                                                                                                                                                                                                  |
| <a href="#">ST_POLYGONFROMWKB</a>     | Geospatial Polygon Constructors | Alias for ST_POLYGON(binary).                                                                                                                                                                                                |
| <a href="#">ST_WHOLEEARTH</a>         | Geospatial Polygon Constructors | Returns the database internal representation of the whole earth polygon.                                                                                                                                                     |
| <a href="#">ST_ANGLE</a>              | Geospatial Spatial Measurement  | Calculates the angle between two lines.                                                                                                                                                                                      |
| <a href="#">ST_AREA</a>               | Geospatial Spatial Measurement  | Returns the area of the specified geospatial object in the specified unit of measurement.                                                                                                                                    |
| <a href="#">ST_AZIMUTH</a>            | Geospatial Spatial Measurement  | Returns the azimuth of the line from <b>point1</b> to <b>point2</b> in radians.                                                                                                                                              |
| <a href="#">ST_DISTANCE</a>           | Geospatial Spatial Measurement  | Returns the minimum distance between the specified arguments.                                                                                                                                                                |

| Function Name                               | Category Name                  | Function Description                                                                                                                                                 |
|---------------------------------------------|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ST_DISTANCESPHERE</a>           | Geospatial Spatial Measurement | Returns the minimum distance between the specified arguments using a spherical computation.                                                                          |
| <a href="#">ST_DISTANCESPHEROID</a>         | Geospatial Spatial Measurement | Returns the minimum distance between the specified arguments using a spheroid computation.                                                                           |
| <a href="#">ST_EUCLIDEANDISTANCE3D</a>      | Geospatial Spatial Measurement | Returns the minimum distance between two geospatial points with altitude values.                                                                                     |
| <a href="#">ST_HAUSDORFFDISTANCE</a>        | Geospatial Spatial Measurement | Returns the Hausdorff distance between two geographies in a specified measurement.                                                                                   |
| <a href="#">ST_LENGTH</a>                   | Geospatial Spatial Measurement | Returns the length of the specified line in the specified measurement unit.                                                                                          |
| <a href="#">ST_LENGTH2D</a>                 | Geospatial Spatial Measurement | Alias for ST_LENGTH.                                                                                                                                                 |
| <a href="#">ST_MAXDISTANCE</a>              | Geospatial Spatial Measurement | Returns maximum distance between the specified arguments.                                                                                                            |
| <a href="#">ST_MINIMUMDISTANCETOSURFACE</a> | Geospatial Spatial Measurement | Calculates the shortest distance between any point along a Euclidean line segment in three-dimensional space and the surface of the Earth.                           |
| <a href="#">ST_PERIMETER</a>                | Geospatial Spatial Measurement | Returns the length of the exterior (outer ring) of the POLYGON in the specified unit of measurement.                                                                 |
| <a href="#">ST_PERIMETER2D</a>              | Geospatial Spatial Measurement | Alias for ST_PERIMETER.                                                                                                                                              |
| <a href="#">ST_BOUNDINGDIAGONAL</a>         | Geospatial Spatial Operators   | Returns the diagonal LINESTRING from the minimum point to the maximum point of the bounding box that ST_ENVELOPE returns.                                            |
| <a href="#">ST_BUFFER</a>                   | Geospatial Spatial Operators   | Returns a geography that contains all points where the distance from the geography is less than or equal to the specified distance.                                  |
| <a href="#">ST_CLOSESTPOINT</a>             | Geospatial Spatial Operators   | Returns the two-dimensional POINT of one specified geospatial object that is closest to a second specified geospatial object.                                        |
| <a href="#">ST_CONVEXHULL</a>               | Geospatial Spatial Operators   | The convex hull is the smallest convex geometry that encloses the input geometry.                                                                                    |
| <a href="#">ST_DIFFERENCEARRAY</a>          | Geospatial Spatial Operators   | Returns an array containing any geospatial objects that are present in the first specified geospatial argument that are not found in the second geospatial argument. |
| <a href="#">ST_ENVELOPE</a>                 | Geospatial Spatial Operators   | Returns a POLYGON that represents the minimum bounding box for the specified geography.                                                                              |
| <a href="#">ST_EXPAND</a>                   | Geospatial Spatial Operators   | Returns the bounding box of a specified geospatial value, which is expanded by a specified length.                                                                   |

| Function Name                              | Category Name                | Function Description                                                                                                                                                                  |
|--------------------------------------------|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ST_EXTERIORRING</a>            | Geospatial Spatial Operators | Returns a LINESTRING that represents the exterior ring of a provided POLYGON value.                                                                                                   |
| <a href="#">ST_FLIPCOORDINATES</a>         | Geospatial Spatial Operators | Returns a new geographic object with the X and Y coordinates switched using the specified argument.                                                                                   |
| <a href="#">ST_FORCE2D</a>                 | Geospatial Spatial Operators | Convert a geographic object into a two-dimensional geography.                                                                                                                         |
| <a href="#">ST_INTERIORRINGN</a>           | Geospatial Spatial Operators | Returns a LINESTRING representing the interior ring of the specified POLYGON, which is specified by its index. (1-indexed)                                                            |
| <a href="#">ST_INTERSECTALL</a>            | Geospatial Spatial Operators | Returns the intersection of all geographies in the specified array. All geographies in the array must be the same type.                                                               |
| <a href="#">ST_INTERSECTIONARRAY</a>       | Geospatial Spatial Operators | Returns a geography that represents the point-set intersection of two geographies.                                                                                                    |
| <a href="#">ST_LONGESTLINE</a>             | Geospatial Spatial Operators | Returns the longest LINESTRING between two specified geospatial arguments.                                                                                                            |
| <a href="#">ST_MAKEENVELOPE</a>            | Geospatial Spatial Operators | Returns a POLYGON with vertices that represent the minimum bounding box for the specified coordinates.                                                                                |
| <a href="#">ST_MINIMUMBOUNDINGCIRCLE</a>   | Geospatial Spatial Operators | Returns the smallest circle POLYGON that contains the specified geographic object.                                                                                                    |
| <a href="#">ST_MULTIDIFFERENCEARRAY</a>    | Geospatial Spatial Operators | Returns an array of geographies that represents the parts of the union of the geographies in the first array that do not intersect with the union of geographies in the second array. |
| <a href="#">ST_MULTIINTERSECTIONARRAY</a>  | Geospatial Spatial Operators | When you specify two arrays of geospatial objects, this function returns an array of any intersections.                                                                               |
| <a href="#">ST_MULTISYMDIFFERENCEARRAY</a> | Geospatial Spatial Operators | When you specify two arrays of geospatial objects, this function returns an array of any geospatial values that do not intersect.                                                     |
| <a href="#">ST_MULTIUNIONARRAY</a>         | Geospatial Spatial Operators | When you specify two arrays of geospatial objects, this function returns one array that represents a union of all geographies in both arrays.                                         |
| <a href="#">ST_NRINGS</a>                  | Geospatial Spatial Operators | Returns the number of rings of the specified POLYGON, including both interior and exterior rings.                                                                                     |
| <a href="#">ST_NUMINTERIORRING</a>         | Geospatial Spatial Operators | Returns the number of interior rings of the specified POLYGON.                                                                                                                        |
| <a href="#">ST_NUMINTERIORRINGS</a>        | Geospatial Spatial Operators | Returns the number of interior rings of the specified POLYGON.                                                                                                                        |

| Function Name                           | Category Name                    | Function Description                                                                                                                                                                  |
|-----------------------------------------|----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ST_POINTONSURFACE</a>       | Geospatial Spatial Operators     | Returns a POINT guaranteed to intersect the specified geospatial object.                                                                                                              |
| <a href="#">ST_PROJECT</a>              | Geospatial Spatial Operators     | Returns a POINT by projecting a distance and an azimuth value from the specified starting POINT value.                                                                                |
| <a href="#">ST_REDUCEPRECISION</a>      | Geospatial Spatial Operators     | Returns a new geospatial object with all POINT values rounded to the specified decimal precision.                                                                                     |
| <a href="#">ST_REMOVEREPEATEDPOINTS</a> | Geospatial Spatial Operators     | Returns a new geospatial object with no repeated POINT values.                                                                                                                        |
| <a href="#">ST_REVERSE</a>              | Geospatial Spatial Operators     | Returns a new geospatial object with the vertexes reversed.                                                                                                                           |
| <a href="#">ST_SEGMENTIZE</a>           | Geospatial Spatial Operators     | Returns a geospatial object that the function modifies to have no segment longer than the specified <code>max_segment_length</code> in meters.                                        |
| <a href="#">ST_SHORTESTLINE</a>         | Geospatial Spatial Operators     | Returns the shortest LINESTRING between two specified geospatial arguments.                                                                                                           |
| <a href="#">ST_SIMPLIFY</a>             | Geospatial Spatial Operators     | Returns a simplified version of the specified geography, which is either a POINT or LINESTRING.                                                                                       |
| <a href="#">ST_SIMPLIFYARRAY</a>        | Geospatial Spatial Operators     | Returns a POLYGON array that represents a simplified version of the specified geography, which is either a POINT, LINESTRING, or POLYGON.                                             |
| <a href="#">ST_SNAPTOGRID</a>           | Geospatial Spatial Operators     | Returns a new geography value with all POINT values rounded to the specified precisions.                                                                                              |
| <a href="#">ST_SYMDIFFERENCEARRAY</a>   | Geospatial Spatial Operators     | Returns a geographic array that contains the parts that are not common between two geographic objects, <code>geo1</code> and <code>geo2</code> .                                      |
| <a href="#">ST_UNIONARRAY</a>           | Geospatial Spatial Operators     | Performs a union of the input geography values to produce a geographic array.                                                                                                         |
| <a href="#">ST_CLUSTERDBSCAN</a>        | Geospatial Spatial Relationships | Returns the cluster number for each input geography, based on a two-dimensional implementation of the density-based spatial clustering of applications with noise (DBSCAN) algorithm. |
| <a href="#">ST_CONTAINS</a>             | Geospatial Spatial Relationships | Returns TRUE if the first geographic argument, <code>geo1</code> , contains the second geographic argument, <code>geo2</code> .                                                       |
| <a href="#">ST_CONTAINSPROPERLY</a>     | Geospatial Spatial Relationships | Returns true if <code>geo2</code> lies entirely in the interior of <code>geo1</code> and does not intersect or touch the boundary or exterior points.                                 |
| <a href="#">ST_COVEREDBY</a>            | Geospatial Spatial Relationships | Returns TRUE if no POINT in <code>geo1</code> is outside of <code>geo2</code> .                                                                                                       |

| Function Name                        | Category Name                    | Function Description                                                                                                                                                                                                             |
|--------------------------------------|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ST_COVERS</a>            | Geospatial Spatial Relationships | Returns TRUE if no POINT in geo2 is outside of geo1.                                                                                                                                                                             |
| <a href="#">ST_CROSSES</a>           | Geospatial Spatial Relationships | Returns TRUE if two geospatial objects meet these criteria: The intersection of the geospatial interiors is not empty. The intersection is not equal to geo1 or geo2. Neither geospatial object is a single POINT.               |
| <a href="#">ST_DISJOINT</a>          | Geospatial Spatial Relationships | Returns true if the specified geographies have no intersection, including boundaries. Both geographic arguments can be different types.                                                                                          |
| <a href="#">ST_DWITHIN</a>           | Geospatial Spatial Relationships | Returns TRUE if the geographies are within a specified distance in meters.                                                                                                                                                       |
| <a href="#">ST_EQUALS</a>            | Geospatial Spatial Relationships | Returns TRUE if both geographies are spatially equal.                                                                                                                                                                            |
| <a href="#">ST_INTERSECTS</a>        | Geospatial Spatial Relationships | Returns TRUE if the specified geographies have intersection, including boundaries.                                                                                                                                               |
| <a href="#">ST_ISCCW</a>             | Geospatial Spatial Relationships | Alias for ST_ISPOLYGONCCW.                                                                                                                                                                                                       |
| <a href="#">ST_ISCLOSED</a>          | Geospatial Spatial Relationships | Returns TRUE if an input POLYGON has an exterior that is counter-clockwise.                                                                                                                                                      |
| <a href="#">ST_ISPOLYGONCCW</a>      | Geospatial Spatial Relationships | Returns TRUE if an input LINESTRING has starting and ending points that are equal.                                                                                                                                               |
| <a href="#">ST_ISPOLYGONCW</a>       | Geospatial Spatial Relationships | Returns TRUE if an input POLYGON has an exterior that is clockwise.                                                                                                                                                              |
| <a href="#">ST_ISRING</a>            | Geospatial Spatial Relationships | Returns TRUE if the specified LINESTRING is closed and does not intersect itself.                                                                                                                                                |
| <a href="#">ST_ISSIMPLE</a>          | Geospatial Spatial Relationships | For LINESTRING values, this function returns FALSE if any line segments intersect anywhere besides the endpoints. For POLYGON values, the function returns FALSE if either the exterior ring or any interior hole is not simple. |
| <a href="#">ST_ISVALID</a>           | Geospatial Spatial Relationships | Returns TRUE if the specified geospatial value is a well-formed and valid geography according to the OGC standards.                                                                                                              |
| <a href="#">ST_OVERLAPS</a>          | Geospatial Spatial Relationships | Returns TRUE if both geographic arguments are of the same dimension and they intersect each other, but neither contains the other.                                                                                               |
| <a href="#">ST_POINTINSIDECIRCLE</a> | Geospatial Spatial Relationships | Returns TRUE if the geographic object is inside a circle that is centered at the specified point coordinates and with the specified radius.                                                                                      |



| Function Name                 | Category Name                         | Function Description                                                                                                                                                                |
|-------------------------------|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ST_RELATE                     | Geospatial Spatial Relationships      | Returns the DE-9IM intersection string that represents the nature of the intersection with the specified geographies.                                                               |
| ST_TOUCHES                    | Geospatial Spatial Relationships      | Returns TRUE if the only POINT values in common between the two geographic arguments lie in the union of their boundaries.                                                          |
| ST_WITHIN                     | Geospatial Spatial Relationships      | Alias for ST_CONTAINS.                                                                                                                                                              |
| ST_DISTANCE                   | Geospatial Spatiotemporal Measurement | Returns the two-dimensional interpolated minimum simultaneous distance between two LINESTRING-TIMESTAMP array pairs in the specified unit of measurement.                           |
| ST_INTERSECTION               | Geospatial Spatiotemporal Measurement | Returns a tuple that represents the intersection of a spatiotemporal LINESTRING with a static geography.                                                                            |
| ST_LINEGETALLTIMESATPOINT     | Geospatial Spatiotemporal Measurement | Returns a timestamp array of all times when the specified LINESTRING value intersects the specified POINT value.                                                                    |
| ST_LINEGETPOINTATTIME         | Geospatial Spatiotemporal Measurement | Returns a POINT within the bounds of the specified LINESTRING that corresponds to the interpolated point at the specified TIMESTAMP value.                                          |
| ST_LINEGETTIMEATPOINT         | Geospatial Spatiotemporal Measurement | Returns the interpolated time of the specified POINT on the specified LINESTRING that is paired with a TIMESTAMP ARRAY.                                                             |
| ST_LONGESTLINE                | Geospatial Spatiotemporal Measurement | With the specified two LINESTRING-TIMESTAMP ARRAY pairs, this function returns a two-point LINESTRING that represents the maximum distance between points at a concurrent time.     |
| ST_MAXDISTANCE                | Geospatial Spatiotemporal Measurement | Returns the two-dimensional interpolated maximum cotemporal distance between two LINESTRING-TIMESTAMP array pairs in the specified unit of measurement.                             |
| ST_SHORTESTLINE               | Geospatial Spatiotemporal Measurement | When you specify two LINESTRING-TIMESTAMP ARRAY pairs, this function returns a LINESTRING with two points that represents the minimum distance between points at a concurrent time. |
| ST_TOTALSECONDSININTERSECTION | Geospatial Spatiotemporal Measurement | Returns the total number of seconds spent in the intersection result calculated by the spatiotemporal version of ST_INTERSECTION.                                                   |



| Function Name                                         | Category Name         | Function Description                                                                                                                                                                                                                      |
|-------------------------------------------------------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">HLL_SKETCH_CREATE</a>                     | HyperLogLog Functions | Creates an HLL sketch from the data on a specified aggregated column. Returns a $\text{HASH}((2^{\log 2k} + 8))$ data representation of the sketch that you can store in a separate column.                                               |
| <a href="#">HLL_SKETCH_GET_ESTIMATE</a>               | HyperLogLog Functions | The scalar function converts a sketch into a distinct count estimate of a sketch value. Returns the distinct count estimate as a <b>BIGINT</b> .                                                                                          |
| <a href="#">HLL_SKETCH_GET_ESTIMATE_BOUND</a>         | HyperLogLog Functions | Takes a <b>HLL_SKETCH</b> column or an integral log2k literal value and returns the resulting bounding 95-percent confidence interval error proportion as a <b>DOUBLE</b> .                                                               |
| <a href="#">HLL_SKETCH_TO_STRING</a>                  | HyperLogLog Functions | The <b>HLL_SKETCH_TO_STRING</b> scalar function takes a <b>HLL_SKETCH</b> column or value and returns a string summary of the sketch.                                                                                                     |
| <a href="#">HLL_SKETCH_UNION</a> (aggregate function) | HyperLogLog Functions | Merges multiple sketches in a single column into a unified sketch. All sketches must have the same precision. This function is an aggregate function and operates on a column.                                                            |
| <a href="#">HLL_SKETCH_UNION</a> (scalar function)    | HyperLogLog Functions | Merges two sketches into a new combined sketch. This function is a scalar function and operates row-wise. The scalar function merges two sketch columns with heterogeneous precisions into a sketch with the lower of the two precisions. |
| <a href="#">ABS</a>                                   | Math Functions        | Returns the absolute value of a specified floating-point number.                                                                                                                                                                          |
| <a href="#">ACOS</a>                                  | Math Functions        | Returns the inverse cosine of a specified floating-point number.                                                                                                                                                                          |
| <a href="#">ACOSH</a>                                 | Math Functions        | Returns the hyperbolic arc-cosine of a specified floating-point number.                                                                                                                                                                   |
| <a href="#">ASIN</a>                                  | Math Functions        | Returns the inverse sine of a specified floating-point number.                                                                                                                                                                            |
| <a href="#">ASINH</a>                                 | Math Functions        | Returns the hyperbolic arc-sine of a specified floating-point number.                                                                                                                                                                     |
| <a href="#">ATAN</a>                                  | Math Functions        | Returns the inverse tangent of a specified floating-point number.                                                                                                                                                                         |
| <a href="#">ATAN2</a>                                 | Math Functions        | Returns the inverse tangent of two numeric, floating-point values.                                                                                                                                                                        |
| <a href="#">ATANH</a>                                 | Math Functions        | Returns the hyperbolic arc-tangent of a specified floating-point number.                                                                                                                                                                  |
| <a href="#">BICDF</a>                                 | Math Functions        | The cumulative distribution function of the standard bivariate normal distribution.                                                                                                                                                       |

| Function Name | Category Name  | Function Description                                                                                                                                                 |
|---------------|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BIPDF         | Math Functions | The probability density function of the standard bivariate normal distribution.                                                                                      |
| BITAND        | Math Functions | Alias for the BITFUNC syntax <code>BITFUNC('AND', x, y)</code> .                                                                                                     |
| BITFUNC       | Math Functions | Performs a variety of bit operations. Can be any of these string literals: 'AND', 'OR', or 'XOR'.                                                                    |
| BITNOT        | Math Functions | Returns the bitwise negation of <code>integral_x</code> .                                                                                                            |
| BITOR         | Math Functions | Alias for the BITFUNC syntax <code>BITFUNC('OR', x, y)</code> .                                                                                                      |
| BITXOR        | Math Functions | Alias for the BITFUNC syntax <code>BITFUNC('XOR', x, y)</code> .                                                                                                     |
| BOOLAND       | Math Functions | Alias for the BOOLFUNC syntax <code>BOOLFUNC('AND', x, y)</code> .                                                                                                   |
| BOOLFUNC      | Math Functions | Performs a Boolean logical evaluation on arguments x and y. Can be any of these string literals: 'AND', 'OR', or 'XOR'.                                              |
| BOOLNOT       | Math Functions | Returns the logical negation of the <code>BOOLFUNC</code> function.                                                                                                  |
| BOOLOR        | Math Functions | Alias for the BOOLFUNC syntax <code>BOOLFUNC('OR', x, y)</code> .                                                                                                    |
| BOOLXOR       | Math Functions | Alias for the BOOLFUNC syntax <code>BOOLFUNC('XOR', x, y)</code> .                                                                                                   |
| CBRT          | Math Functions | Returns the cube root of the numeric value x.                                                                                                                        |
| CDF           | Math Functions | The cumulative distribution function of the standard normal distribution. Returns the probability that a random sample is less than or equal to the specified value. |
| CEIL          | Math Functions | Returns the nearest integer greater than or equal to x.                                                                                                              |
| CEILING       | Math Functions | Alias for <code>CEIL</code> .                                                                                                                                        |
| COS           | Math Functions | Returns the cosine of x.                                                                                                                                             |
| COSH          | Math Functions | Returns the hyperbolic cosine of x.                                                                                                                                  |
| COT           | Math Functions | Returns the cotangent of x.                                                                                                                                          |
| DEGREES       | Math Functions | Returns the corresponding angle in degrees for x in radians.                                                                                                         |
| DIV           | Math Functions | Returns the result of x divided by y. If y is zero, returns NULL.                                                                                                    |
| ERF           | Math Functions | The error function is used for measurements that follow a normal distribution.                                                                                       |
| ERFC          | Math Functions | The complement of the error function. $ERFC(x) = 1 - ERF(x)$ .                                                                                                       |
| EXP           | Math Functions | Returns the exponential of x (e raised to the power of x).                                                                                                           |
| FLOOR         | Math Functions | Returns the nearest integer less than or equal to x.                                                                                                                 |
| GAMMA         | Math Functions | <a href="#">Gamma function</a> .                                                                                                                                     |

| Function Name | Category Name  | Function Description                                                                                                                                  |
|---------------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| HEXBINX       | Math Functions | Returns the x-coordinate of the center of the nearest hexagonal bin to the point (x, y).                                                              |
| HEXBINY       | Math Functions | Returns the y-coordinate of the center of the nearest hexagonal bin to the point (x, y).                                                              |
| IERF          | Math Functions | The inverse of the ERF error function.                                                                                                                |
| IERFC         | Math Functions | The inverse of the complement of the error function.                                                                                                  |
| LEAKYRELU     | Math Functions | Returns the leaky rectified linear unit function of x.                                                                                                |
| LEFT_SHIFT    | Math Functions | Returns x shifted to the left by y bits.                                                                                                              |
| LN            | Math Functions | Returns the natural logarithm of x.                                                                                                                   |
| LOG           | Math Functions | Returns the base 10 logarithm of x. The optional base argument specifies the numeral system to use. If unspecified, the function defaults to base 10. |
| LOG_GAMMA     | Math Functions | The natural logarithm of the absolute value of the gamma function.                                                                                    |
| LOG2          | Math Functions | Returns the base 2 logarithm of x.                                                                                                                    |
| MOD           | Math Functions | Returns the remainder from x divided by y.                                                                                                            |
| PI            | Math Functions | Returns the constant value of $\pi$ .                                                                                                                 |
| PMOD          | Math Functions | Returns the smallest non-negative equivalence class of $x \% y$ .                                                                                     |
| POWER         | Math Functions | Returns x raised to the power of y.                                                                                                                   |
| PROBIT        | Math Functions | The inverse of the cumulative distribution function.                                                                                                  |
| RADIANS       | Math Functions | Returns the corresponding angle in radians for x in degrees.                                                                                          |
| RAND          | Math Functions | Takes no argument and returns a random DOUBLE value in the range [0, 1).                                                                              |
| RELU          | Math Functions | Returns the rectified linear unit function of x.                                                                                                      |
| RIGHT_SHIFT   | Math Functions | Returns x shifted to the right by y bits.                                                                                                             |
| ROUND         | Math Functions | Returns x rounded to the nearest integer.                                                                                                             |
| SIGN          | Math Functions | Returns the positive (+1), zero (0), or negative (-1) sign of x.                                                                                      |
| SIN           | Math Functions | Returns the sine of x.                                                                                                                                |
| SINH          | Math Functions | Returns the hyperbolic sine of x.                                                                                                                     |
| SQRT          | Math Functions | Returns the square root of x.                                                                                                                         |
| SQUARE        | Math Functions | Returns the square of x.                                                                                                                              |
| TAN           | Math Functions | Returns the tangent of x.                                                                                                                             |
| TANH          | Math Functions | Returns the hyperbolic tangent of x.                                                                                                                  |
| TRUNC         | Math Functions | Returns x truncated to y decimal places.                                                                                                              |
| TRUNCATE      | Math Functions | Alias for TRUNC.                                                                                                                                      |

| Function Name                      | Category Name    | Function Description                                                                        |
|------------------------------------|------------------|---------------------------------------------------------------------------------------------|
| <a href="#">ABS</a>                | Matrix Functions | Returns magnitude of one-dimensional matrix or vector.                                      |
| <a href="#">CROSS_ENTROPY_LOSS</a> | Matrix Functions | Returns the cross entropy loss of two one-dimensional matrixes or vectors.                  |
| <a href="#">DET</a>                | Matrix Functions | Returns the determinant of the matrix as a double.                                          |
| <a href="#">DOT</a>                | Matrix Functions | Returns dot product of two one-dimensional matrixes/vectors.                                |
| <a href="#">EIGEN</a>              | Matrix Functions | Returns eigenvalues and eigenvalues of a square matrix as a vector of pairs.                |
| <a href="#">FROBENIUS</a>          | Matrix Functions | Returns the Frobenius norm of a matrix.                                                     |
| <a href="#">HINGE_LOSS</a>         | Matrix Functions | Returns the hinge loss of two one-dimensional matrixes or vectors.                          |
| <a href="#">IDENTITY_MATRIX</a>    | Matrix Functions | Returns an identity matrix of the specified dimension.                                      |
| <a href="#">INVERSE</a>            | Matrix Functions | Returns inverse of a square, invertible matrix.                                             |
| <a href="#">LOG_LOSS</a>           | Matrix Functions | Returns the log loss of two one-dimensional matrixes or vectors.                            |
| <a href="#">LOGITS_LOSS</a>        | Matrix Functions | Returns the logits loss of two one-dimensional matrixes or vectors.                         |
| <a href="#">LUPQ_DECOMP</a>        | Matrix Functions | Returns LUPQ decomposition of a square matrix A as a tuple of 4 matrixes where $PAQ = LU$ . |
| <a href="#">MATRIX_DIM</a>         | Matrix Functions | Returns the dimensions of the specified matrix as a tuple of (row, col) integers.           |
| <a href="#">MAKE_MATRIX_IJ</a>     | Matrix Functions | Creates an <code>ij</code> matrix with elements <code>e_00, ..., e_ij</code> .              |
| <a href="#">MATRIX_FROM_TEXT</a>   | Matrix Functions | Creates a matrix from the specified string.                                                 |
| <a href="#">MATRIX_TRACE</a>       | Matrix Functions | Returns trace of a square matrix as a double.                                               |
| <a href="#">NULL_MATRIX</a>        | Matrix Functions | Returns a NULL matrix of the specified (row, col) dimensions.                               |
| <a href="#">QR_DECOMP</a>          | Matrix Functions | Returns QR decomposition of a matrix as a tuple of 2 matrixes.                              |
| <a href="#">SOFTMAX</a>            | Matrix Functions | Returns the softmax of a one-dimensional matrix or vector.                                  |
| <a href="#">SVD_DECOMP</a>         | Matrix Functions | Returns SVD decomposition of a matrix as a tuple of 3 matrixes.                             |
| <a href="#">TRANSPOSE</a>          | Matrix Functions | Returns transpose of the matrix.                                                            |
| <a href="#">VECTOR_ARGMAX</a>      | Matrix Functions | Returns the argmax of a one-dimensional matrix or vector.                                   |
| <a href="#">VECTOR_ARGMIN</a>      | Matrix Functions | Returns the argmin of a one-dimensional matrix or vector.                                   |
| <a href="#">VECTOR_MAX</a>         | Matrix Functions | Returns the maximum of elements in a one-dimensional matrix/vector.                         |
| <a href="#">VECTOR_MIN</a>         | Matrix Functions | Returns the minimum of elements in a one-dimensional matrix/vector.                         |
| <a href="#">VECTOR_SUM</a>         | Matrix Functions | Returns the sum of elements in a one-dimensional matrix/vector.                             |

| Function Name                | Category Name                    | Function Description                                                                                                                                                                           |
|------------------------------|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ZERO_MATRIX</a>  | Matrix Functions                 | Returns a zero matrix of the given (row, col) dimensions.                                                                                                                                      |
| <a href="#">IP</a>           | Network Type Functions           | Casts an IP data type from an IPV4 data type expression.                                                                                                                                       |
| <a href="#">IPV4</a>         | Network Type Functions           | Casts an IPV4 address from an IPV6 address.                                                                                                                                                    |
| <a href="#">IS_IPV4</a>      | Network Type Functions           | Tests whether the database can convert the IP value to the IPV4 data type.                                                                                                                     |
| <a href="#">SUBNET</a>       | Network Type Functions           | Computes the prefix from an IP or IPV4 value and the size of the prefix.                                                                                                                       |
| <a href="#">BIGINT</a>       | Scalar Data Conversion Functions | Casts the argument to a value of type <b>bigint</b> .                                                                                                                                          |
| <a href="#">BINARY</a>       | Scalar Data Conversion Functions | Parses a hexadecimal string (such as 0x54ab) to create a binary value. Letters can be of either case.                                                                                          |
| <a href="#">BOOLEAN</a>      | Scalar Data Conversion Functions | Parses a string to create a Boolean value. The string must contain either true or false. It is case-insensitive.                                                                               |
| <a href="#">BYTE</a>         | Scalar Data Conversion Functions | Casts the argument to a value of type byte.                                                                                                                                                    |
| <a href="#">CHAR</a>         | Scalar Data Conversion Functions | Creates a string version of the numeric value.                                                                                                                                                 |
| <a href="#">DATE</a>         | Scalar Data Conversion Functions | Parses a string in the form 'YYYY-MM-DD' to create a date. Extra characters are ignored.                                                                                                       |
| <a href="#">DAYS</a>         | Scalar Data Conversion Functions | Converts an integral value to an interval value of type days to be used in date calculations.                                                                                                  |
| <a href="#">DECIMAL</a>      | Scalar Data Conversion Functions | Casts the argument to a value of type decimal.                                                                                                                                                 |
| <a href="#">DOUBLE</a>       | Scalar Data Conversion Functions | Casts the argument to a value of type double.                                                                                                                                                  |
| <a href="#">FLOAT</a>        | Scalar Data Conversion Functions | Casts the argument to a value of type float.                                                                                                                                                   |
| <a href="#">HASH</a>         | Scalar Data Conversion Functions | Creates a fixed length binary value, with length <b>&lt;length&gt;</b> , from a string, i.e. 0x1234abcd. Zero extended if the string does not have enough bytes, truncated if it has too many. |
| <a href="#">HOURS</a>        | Scalar Data Conversion Functions | Converts an integral value to an interval value of type hours.                                                                                                                                 |
| <a href="#">INTEGER</a>      | Scalar Data Conversion Functions | Casts the argument to a value of type integer.                                                                                                                                                 |
| <a href="#">IP</a>           | Scalar Data Conversion Functions | Creates a value of type IPV4 or IPV6 from a string.                                                                                                                                            |
| <a href="#">IPV4</a>         | Scalar Data Conversion Functions | Parses a string containing an IP address and creates a value of type IPV5.                                                                                                                     |
| <a href="#">MICROSECONDS</a> | Scalar Data Conversion Functions | Converts an integral value to an interval value of type microseconds.                                                                                                                          |
| <a href="#">MILLISECONDS</a> | Scalar Data Conversion Functions | Converts an integral value to an interval value of type milliseconds.                                                                                                                          |

| Function Name                  | Category Name                                  | Function Description                                                                                                      |
|--------------------------------|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <a href="#">MINUTES</a>        | Scalar Data Conversion Functions               | Converts an integral value to an interval value of type minutes.                                                          |
| <a href="#">MONTHS</a>         | Scalar Data Conversion Functions               | Converts an integral value to an interval value of type months to be used in date calculations.                           |
| <a href="#">NANOSECONDS</a>    | Scalar Data Conversion Functions               | Converts an integral value to an interval value of type nanoseconds.                                                      |
| <a href="#">SECONDS</a>        | Scalar Data Conversion Functions               | Converts an integral value to an interval value of type seconds.                                                          |
| <a href="#">SMALLINT</a>       | Scalar Data Conversion Functions               | Casts the argument to a value of type <b>smallint</b> .                                                                   |
| <a href="#">TIME</a>           | Scalar Data Conversion Functions               | Parses the string to create a time value. The string must be in the form 'HH:MM[.SSSSSSSS] '.                             |
| <a href="#">TIMESTAMP</a>      | Scalar Data Conversion Functions               | Parses a string and makes a timestamp value. The string must be in the format 'YYYY-MM-DD[HH:MM][.SSSSSSSS] '.            |
| <a href="#">UUID</a>           | Scalar Data Conversion Functions               | Parses the string and makes a UUID value. The string must be a valid UUID.                                                |
| <a href="#">WEEKS</a>          | Scalar Data Conversion Functions               | Converts an integral value to an interval value of type weeks to be used in date calculations.                            |
| <a href="#">YEARS</a>          | Scalar Data Conversion Functions               | Converts an integral value to an interval value of type years.                                                            |
| <a href="#">ARRAY_AGG</a>      | Sorted Aggregate Functions                     | Returns an array containing every row from the expression.                                                                |
| <a href="#">STRING_AGG</a>     | Sorted Aggregate Functions                     | Returns a string concatenated from every row from the expression. The delimiter argument is optional.                     |
| <a href="#">ARRAY_CAP</a>      | Special Data Pipeline Transformation Functions | Restrict the length of an array to a maximum number of elements.                                                          |
| <a href="#">ARRAY_COMPACT</a>  | Special Data Pipeline Transformation Functions | Removes NULL values from the array.                                                                                       |
| <a href="#">ARRAY_CONTAINS</a> | Special Data Pipeline Transformation Functions | Returns <b>true</b> if the array contains the specified value.                                                            |
| <a href="#">ARRAY_DISTINCT</a> | Special Data Pipeline Transformation Functions | Return all unique values in an array.                                                                                     |
| <a href="#">ARRAY_SORT</a>     | Special Data Pipeline Transformation Functions | Sort and return the input array based on the natural ordering of its elements or the specified Lambda function.           |
| <a href="#">CASE WHEN</a>      | Special Data Pipeline Transformation Functions | Returns the <b>result</b> value based on whether an expression is <b>true</b> .                                           |
| <a href="#">ELEMENT_AT</a>     | Special Data Pipeline Transformation Functions | Returns the element value of the array or tuple at the specified index.                                                   |
| <a href="#">EXPLODE_OUTER</a>  | Special Data Pipeline Transformation Functions | Expands a one-dimensional or multidimensional array into its elements with one element per row of output from the system. |
| <a href="#">FILTER</a>         | Special Data Pipeline Transformation Functions | Filters elements in an array based on the logic in a lambda expression.                                                   |

| Function Name         | Category Name                                     | Function Description                                                                                                                                                                           |
|-----------------------|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FLATTEN               | Special Data Pipeline                             | Transforms an N-dimensional array into an N-1-dimensional array.                                                                                                                               |
| IF                    | Transformation Functions<br>Special Data Pipeline | Returns T if the expression X evaluates to <b>true</b> , or the function returns F if X evaluates to <b>false</b> .                                                                            |
| MAP_KEYS              | Transformation Functions<br>Special Data Pipeline | Returns the keys in the specified JSON string.                                                                                                                                                 |
| MAP_VALUES            | Transformation Functions<br>Special Data Pipeline | Returns the values in the specified JSON string.                                                                                                                                               |
| METADATA              | Special Data Pipeline<br>Transformation Functions | Extracts the metadata value for the specified key from available metadata for the pipeline.                                                                                                    |
| PARSE_DELIMITED_ARRAY | Special Data Pipeline<br>Transformation Functions | Converts a string of text data representing an array into a <b>CHAR[]</b> .                                                                                                                    |
| REDUCE                | Special Data Pipeline<br>Transformation Functions | Applies a merge function to a starting value and all elements in the array, and then reduces the array to a single value. Optionally, specify a finish function for the returned single value. |
| TO_ARRAY_LENGTH       | Special Data Pipeline<br>Transformation Functions | Create the specified number of copies of any JSON object in an array.                                                                                                                          |
| TRANSFORM             | Special Data Pipeline<br>Transformation Functions | Transforms an array based on the logic in a lambda expression.                                                                                                                                 |
| WIDTH_BUCKET          | Special Data Pipeline<br>Transformation Functions | Creates N equal-width buckets in the range <b>[min,max)</b> as a histogram.                                                                                                                    |
| ZIP_WITH              | Special Data Pipeline<br>Transformation Functions | Merges two or more input arrays, element-wise, into a single array using the trailing combining function.                                                                                      |
| COMMIT                | System Functions                                  | Returns the most recent commit hash of the database to which the client is currently connected.                                                                                                |
| CURRENT_DATABASE      | System Functions                                  | Alias for DATABASE.                                                                                                                                                                            |
| CURRENT_NODE          | System Functions                                  | Returns the name of the SQL Node where the current query executes. The name of the node corresponds to the name column in the <b>sys.nodes</b> system catalog table.                           |
| CURRENT_NODE_ID       | System Functions                                  | Returns the identifier of the SQL Node where the current query executes.                                                                                                                       |
| CURRENT_SCHEMA        | System Functions                                  | Returns the name of the current schema.                                                                                                                                                        |
| CURRENT_SYSTEM        | System Functions                                  | Returns the name of the system.                                                                                                                                                                |
| CURRENT_USER          | System Functions                                  | Returns the user for the current connection.                                                                                                                                                   |
| DATABASE              | System Functions                                  | Returns the name of the database to which the client is currently connected.                                                                                                                   |
| SHOW                  | System Functions                                  | The SHOW function enables you to explore the database and its metadata for user-defined items.                                                                                                 |



| Function Name                  | Category Name              | Function Description                                                                                                                                                                                                                                                                 |
|--------------------------------|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| VERSION                        | System Functions           | Returns the version of the database to which the client is currently connected.                                                                                                                                                                                                      |
| UNNEST                         | Array Functions            | Expand each element in an input array out to an individual row.                                                                                                                                                                                                                      |
| CONVERT_LOCAL_TIMESTAMP_TO_UTC | Time Zone Functions        | The function converts a timestamp in a specified local time zone to the UTC time zone.                                                                                                                                                                                               |
| CONVERT_UTC_TIMESTAMP_TO_LOCAL | Time Zone Functions        | The function converts a timestamp from the UTC time zone to a specified local time zone.                                                                                                                                                                                             |
| CAST_TO_TUPLE                  | Tuple Functions            | Converts a tuple into another tuple of a different type.                                                                                                                                                                                                                             |
| CHAR                           | Tuple Functions            | Converts a tuple to its string representation.                                                                                                                                                                                                                                       |
| STRING_TO_TUPLE                | Tuple Functions            | Converts the string representation of a tuple (e.g 'tuple«INT,BIGINT,CHAR»(1,2,NULL)') into a tuple.                                                                                                                                                                                 |
| TUPLE()                        | Tuple Functions            | Construct a tuple of specified elements. Types of the tuple are inferred from the inner elements.                                                                                                                                                                                    |
| TUPLE«»                        | Tuple Functions            | Construct a tuple of specified elements. NULL is also supported as an element.                                                                                                                                                                                                       |
| CUME_DIST                      | Window Aggregate Functions | Returns a number $0 < n \leq 1$ and can be used to calculate the percentage of values less than or equal to the current value in the group.                                                                                                                                          |
| DELTA                          | Window Aggregate Functions | Computes the finite difference between successive values of expression under the specified ordering. This is a backwards difference, which means that, at degree one, the value for a given row is the difference between the value of expression for that row and the previous row. |
| DENSE_RANK                     | Window Aggregate Functions | Assigns a number to each row in the result set with equal values having the same number. There will be no gaps between ranks.                                                                                                                                                        |
| DERIVATIVE                     | Window Aggregate Functions | Computes the difference quotient between successive values of expression with respect to expression2.                                                                                                                                                                                |
| FIRST_VALUE                    | Window Aggregate Functions | Returns the first value in the ordered result set.                                                                                                                                                                                                                                   |
| LAG                            | Window Aggregate Functions | Returns the row which is the specified number backward from the current row. Default is 1 if offset is omitted.                                                                                                                                                                      |
| LAST_VALUE                     | Window Aggregate Functions | Returns the last value in the ordered result set.                                                                                                                                                                                                                                    |



| Function Name                   | Category Name              | Function Description                                                                                                                                 |
|---------------------------------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">LEAD</a>            | Window Aggregate Functions | Returns the row which is the specified number forward from the current row. Default is 1 if offset is omitted.                                       |
| <a href="#">NTH_VALUE</a>       | Window Aggregate Functions | Returns the nth value in the ordered result set.                                                                                                     |
| <a href="#">PERCENT_RANK</a>    | Window Aggregate Functions | The value returned is $0 < n \leq 1$ and can be used to calculate the percentage of values less than the current group, excluding the highest value. |
| <a href="#">PERCENTILE</a>      | Window Aggregate Functions | Returns the value that corresponds to the specified percentile ( $0 \leq n \leq 1$ ) within the group.                                               |
| <a href="#">RANK</a>            | Window Aggregate Functions | Assigns a number to each row in the result set with equal values having the same number. There can be gaps between ranks.                            |
| <a href="#">RATIO_TO_REPORT</a> | Window Aggregate Functions | Computes the ratio of a value to the sum of the set of values.                                                                                       |
| <a href="#">ROW_NUMBER</a>      | Window Aggregate Functions | Assigns a unique number to each row in the result set.                                                                                               |
| <a href="#">ZSCORE</a>          | Window Aggregate Functions | Zscore of the sample based on the <code>stddev()</code> function.                                                                                    |
| <a href="#">ZSCOREP</a>         | Window Aggregate Functions | Zscore of the sample based on the <code>stddevp()</code> function.                                                                                   |

### 10.3 Aggregate Functions

Aggregate functions compute a single result from a group of rows.

You can use the `DISTINCT` keyword with any aggregation function, such as `COUNT(DISTINCT col)`.

#### Aggregate Functions

| Function                              | Syntax                                  | Purpose                                                                                                                                                                                                                                                                                                                               |
|---------------------------------------|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ANY_VALUE</a>             | <code>ANY_VALUE(col)</code>             | Returns an arbitrary, non-NULL, value from the input column. The function gives preference to any non-NULL values in the column and returns NULL only if all rows in the input column are NULL. The <code>ANY_VALUE</code> function is explicitly undefined if you use it for window aggregation and the function generates an error. |
| <a href="#">APPROX_COUNT_DISTINCT</a> | <code>APPROX_COUNT_DISTINCT(col)</code> | Approximate distinct count by using hyper-log-log (95% confidence interval that the value is within 4.5%)                                                                                                                                                                                                                             |

| Function     | Syntax                  | Purpose                                                                                                                                                                                                                                                                                          |
|--------------|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| APPROX_SUM   | APPROX_SUM(col)         | Allows the aggregation engine to use a faster, non-deterministic, ordering to summate floating-point columns. This summation might lead to minute differences in the result on the order of the machine epsilon. For other column types, the aggregation engine uses the standard SUM algorithm. |
| AVG          | AVG(col)                | Average, or arithmetic mean, over the set                                                                                                                                                                                                                                                        |
| CORRELATION  | CORRELATION(col1,col2)  | Sample correlation                                                                                                                                                                                                                                                                               |
| CORR         | CORR(col1,col2)         | Alias for CORRELATION                                                                                                                                                                                                                                                                            |
| CORRELATIONP | CORRELATIONP(col1,col2) | Population correlation                                                                                                                                                                                                                                                                           |
| COUNT        | COUNT(col)              | Number of rows in the set where values in col are not null                                                                                                                                                                                                                                       |
| COUNT        | COUNT(*)                | Total number of rows in the table                                                                                                                                                                                                                                                                |
| COVARIANCE   | COVARIANCE(col1,col2)   | Sample covariance                                                                                                                                                                                                                                                                                |
| COVAR_SAMP   | COVAR_SAMP(col1, col2)  | Alias for COVARIANCE                                                                                                                                                                                                                                                                             |
| COVARIANCEP  | COVARIANCEP(col1,col2)  | Population covariance                                                                                                                                                                                                                                                                            |
| COVAR_POP    | COVAR_POP(col1, col2)   | Alias for COVARIANCEP                                                                                                                                                                                                                                                                            |
| MAX          | MAX(col)                | Maximum value in the given column                                                                                                                                                                                                                                                                |
| MIN          | MIN(col)                | Minimum value in the given column                                                                                                                                                                                                                                                                |
| STDEVP       | STDEVP(col)             | Population standard deviation                                                                                                                                                                                                                                                                    |
| STDDEV_POP   | STDDEV_POP(col)         | Alias for STDEVP                                                                                                                                                                                                                                                                                 |
| STDEV        | STDEV(col)              | Sample standard deviation                                                                                                                                                                                                                                                                        |
| STDDEV       | STDDEV(col)             | Alias for STDEV                                                                                                                                                                                                                                                                                  |
| STDDEV_SAMP  | STDDEV_SAMP(col)        | Alias for STDEV                                                                                                                                                                                                                                                                                  |
| SUM          | SUM(col)                | Sum over the set                                                                                                                                                                                                                                                                                 |
| VARIANCE     | VARIANCE(col)           | Sample variance                                                                                                                                                                                                                                                                                  |
| VAR_SAMP     | VAR_SAMP(col)           | Alias for VARIANCE                                                                                                                                                                                                                                                                               |
| VARIANCEP    | VARIANCEP(col)          | Population variance                                                                                                                                                                                                                                                                              |
| VAR_POP      | VAR_POP(col)            | Alias for VARIANCEP                                                                                                                                                                                                                                                                              |
| PRODUCT      | PRODUCT(col)            | Product over the set                                                                                                                                                                                                                                                                             |
| SKEW         | SKEW(col)               | The sample over the set                                                                                                                                                                                                                                                                          |
| SWEP         | SWEP(col)               | The population over the set                                                                                                                                                                                                                                                                      |
| KURTOSIS     | KURTOSIS(col)           | The sample over the set                                                                                                                                                                                                                                                                          |
| KURTOSISP    | KURTOSISP(col)          | The population over the set                                                                                                                                                                                                                                                                      |

### Related Links

[Sorted Aggregate Functions](#)

[Math Functions and Operators](#)

[Query Ociant](#)

## 10.4 Array Functions and Operators

### UNNEST

Expands each element in an input array into an individual row.

For example, the UNNEST function on an array column of type `ARRAY(INT)` with values `[2, 6]` yields two result rows with integers 2 and 6.

The values of the other columns in each input row are unchanged in each corresponding output row. You can specify multiple array columns to unnest the specified arrays from each row in parallel.

### Syntax

```
UNNEST( array_column [, ... ] ≤ )
  [ WITH <unnest_options> ]
```

```
<unnest_options> ::=
  { ORDINALITY [ ord_identifier ]
  | VALUE [ val_identifier ]
  | NULL_INPUT }
```

| Argument       | Data Type | Description                                                                                                |
|----------------|-----------|------------------------------------------------------------------------------------------------------------|
| array_column   | ARRAY     | The array column to expand.                                                                                |
| ord_identifier | CHAR      | When you use this argument in an <b>ORDINALITY</b> clause, the argument is the name for the output column. |
| val_identifier | CHAR      | When you use this argument in a <b>VALUE</b> clause, the argument is the name for the unnested column.     |

The UNNEST function supports these options.

| Options                     | Description                                                                                                                                                                                                                                                                                                                             |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ORDINALITY [ord_identifier] | Optional. Generates a new integer column that corresponds to the index position of each output element in the original array. The <b>ord_identifier</b> value specifies the name of the output ordinality column. If you do not specify this clause, the column name defaults to the original column name with the suffix <b>_ord</b> . |
| VALUE [val_identifier]      | Optional. Rename the unnested column to the new name specified in the <b>val_identifier</b> argument. If you do not specify this clause, the column name defaults to the original column name with the suffix <b>_val</b> .                                                                                                             |
| NULL_INPUT                  | Optional. When the input column array is empty or NULL, this clause specifies to replace each output row with an array with one NULL element. This replacement preserves the contents of other input rows. By default, this option is not enabled.                                                                                      |

These examples demonstrate how the UNNEST function operates on arrays and other objects. Most of the examples use the data rows from this table.

| a | b      | c      | d   | e    | f                        |
|---|--------|--------|-----|------|--------------------------|
| 1 | [1, 2] | [6, 9] | []  | NULL | [[1, 2], [5], []]        |
| 2 | [7, 5] | [4]    | [3] | [8]  | [[6], [9, 4, 7], [3, 8]] |

### Examples

#### Expand a Column

This example performs a simple UNNEST on column **b**.

```
SELECT UNNEST(b);
```

*Output*

```
b_val
-----
1
2
7
5
```

#### Expand a Column Using the ORDINALITY Option

In this example, the `ORDINALITY` option adds a second return column that represents the index position of the value in the input array.

```
SELECT UNNEST(b) WITH ORDINALITY;
```

*Output*

```
b_val  b_ord
-----
1      1
2      2
7      1
5      2
```

#### Expand a Column with Empty Data

In this example, the `UNNEST` query returns only one row because one of the two rows in column `d` is an empty array. If you use the `NULL_INPUT` option, the query returns a second row with the `NULL` value.

```
SELECT UNNEST(d);
```

*Output*

```
d_val
-----
3
```

#### Expand a Column with NULL Data

`UNNEST` also does not return array values that are `NULL` unless you specify the `NULL_INPUT` option.

```
SELECT UNNEST(e);
```

*Output*

```
e_val
-----
8
```

#### Expand a Column with Multiple Array Layers

The `UNNEST` function expands only one array layer.

The empty array remains in the returned values.

```
SELECT UNNEST(f);
```

*Output*

```
f_val
-----
[1,2]
[5]
[]
[6]
[9,4,7]
[3,8]
```

### Expand Columns Using Multiple UNNEST Statements

Multiple UNNEST statements in a query produce a per-row Cartesian product of all the unnested values.

```
SELECT UNNEST(b), UNNEST(c);
```

*Output*

```
b_val  c_val
-----
1      6
1      9
2      6
2      9
7      4
5      4
```

### Expand Columns Using Multiple UNNEST Statements in Reverse Order

This example uses UNNEST on the same columns but in reverse order.

```
SELECT UNNEST(c), UNNEST(b);
```

*Output*

```
c_val  b_val
-----
6      1
6      2
9      1
9      2
4      7
4      5
```

### Expand Multiple Columns

In this example, the query unnests two array columns in parallel. Even though it does not include the `NULL_INPUT` option, the query still returns NULL for column `c` to correspond with column `b`.

```
SELECT UNNEST(b, c);
```

*Output*

```
b_val  c_val
-----
1      6
2      9
7      4
5      NULL
```

### Expand Multiple Array Columns

This query unnests two array columns.

```
SELECT UNNEST(b, d);
```

*Output*

| b_val | d_val |
|-------|-------|
| 1     | NULL  |
| 2     | NULL  |
| 7     | 3     |
| 5     | NULL  |

### Expand Values of Arrays Within Arrays

This example has two layers of UNNEST to capture values of arrays within arrays.

```
SELECT UNNEST(f_val) FROM (UNNEST(f));
```

*Output*

| f_val |
|-------|
| 1     |
| 2     |
| 5     |
| 6     |
| 9     |
| 4     |
| 7     |
| 3     |
| 8     |

### Expand an Empty Array

This example unnests an empty integer array.

```
SELECT UNNEST(INT[] ());
```

*Output*

| col_val |
|---------|
|---------|

The query returns a column of type INT but no rows.

### Expand an Empty Array Column Without the NULL\_INPUT Option

This example selects the non-array column **a** and an empty array column. The query returns no rows because empty or NULL array values are not returned unless you specify the NULL\_INPUT option.

```
SELECT a, UNNEST(INT[] ());
```

*Output*

| a | col_val |
|---|---------|
|---|---------|

### Expand a Column Without the NULL\_INPUT Option

This query does not include the NULL\_INPUT option, which causes the result to omit the NULL array values in column **e**.

```
SELECT a, UNNEST(e);
```

*Output*

```
a  e_val
-----
1  8
2  8
```

### Expand a Column with the NULL\_INPUT Option

This query includes the `NULL_INPUT` option, which means that the query returns the `NULL` values in column `e`.

```
SELECT a, UNNEST(e) WITH NULL_INPUT;
```

*Output*

```
a  e_val
-----
1  8
1  NULL
2  8
2  NULL
```

### Expand Arrays with NULL Rows

This example attempts to unnest two arrays, one of which is cast as `NULL`, while the other is empty.

In both cases, the query returns a column of type `INTEGER` with no rows because there are no values to unnest.

```
SELECT UNNEST(CAST(NULL AS INT[]));
```

```
SELECT UNNEST(INT[] ());
```

*Output*

```
col_val
-----
```

### Other Array Functions

| Function                 | Syntax                           | Purpose                                                                                                                                                                      |
|--------------------------|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Constructor              | TYPE                             | Construct an array of SQL type <code>TYPE</code> giving the elements. <code>NULL</code> is also supported as an element.                                                     |
| Constructor (2)          | array[e1, e2, ..., en]           | {{PostgreSQL}}-compliant constructor. The type of the array is deduced from the elements.                                                                                    |
| array concatenation      | array_cat(array, array)          | Concatenate 2 arrays into a new one.                                                                                                                                         |
| array length             | array_length(array)              | Return the number of elements of a specified array.                                                                                                                          |
| array softmax            | softmax(array)                   | Returns the softmax of the array. The array must be one-dimensional and contain float or double values. The system transfers <code>NULL</code> -valued indices as 0.0.       |
| array cross entropy loss | cross_entropy_loss(array, array) | Returns the cross entropy loss of two arrays. Both arrays must be one-dimensional and contain float or double values. <code>NULL</code> values do not contribute to the sum. |

| Function                 | Syntax                                                                  | Purpose                                                                                                                                                                                                                   |
|--------------------------|-------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| array log loss           | <code>log_loss(array, array)</code>                                     | Returns the log loss of two arrays. Both arrays must be one-dimensional and contain float or double values. NULL values do not contribute to the sum.                                                                     |
| array logits loss        | <code>logits_loss(array, array)</code>                                  | Returns the logits loss of two arrays. Both arrays must be one-dimensional and contain float or double values. NULL values do not contribute to the sum.                                                                  |
| array hinge loss         | <code>hinge_loss(array, array)</code>                                   | Returns the hinge loss of two arrays. Both arrays must be one-dimensional and contain float or double values. NULL values do not contribute to the sum.                                                                   |
| array sum                | <code>array_sum(array)</code>                                           | Returns the sum of the array. The array must be one-dimensional and contain numeric values. NULL values do not contribute to the sum.                                                                                     |
| array min/max            | <code>array_min(array),</code><br><code>array_max(array)</code>         | Returns the corresponding minimum or maximum of the array. This function only considers the first dimension of the array. If the array contains only NULL values, then the function returns NULL.                         |
| array argmin/argmax      | <code>array_argmin(array),</code><br><code>array_argmax(array)</code>   | Returns the corresponding argmin or argmax of the array as a BIGINT index. This function only considers the first dimension of the array. If the array contains only NULL values, then the function returns NULL.         |
| array casting            | <code>cast_to_array(array, format)</code>                               | Casts the elements of the array to another type, as specified by the format string. The format is: 'ARRAY(INT)' for an int array, 'ARRAY(CHAR)' for a char array, and so on. Currently, only numeric casts are supported. |
| scalar position          | <code>array_position(array, scalar, pos = 1)</code>                     | Returns the position of the first matching scalar in the array. pos indicates the position to start searching, if not specified it defaults to 1.                                                                         |
| array elements positions | <code>array_positions(array, array )</code>                             | Same as <code>array_position()</code> . But this looks for all elements stored in the right array and returns their respective positions in the left array.                                                               |
| array append             | <code>array_append(array, value)</code>                                 | Add value to the back of an array                                                                                                                                                                                         |
| array prepend            | <code>array_prepend(value, array)</code>                                | Add value to the front of an array                                                                                                                                                                                        |
| array remove             | <code>array_remove(array, value)</code>                                 | remove value from the array                                                                                                                                                                                               |
| array replace            | <code>array_replace(array, old_value,</code><br><code>new_value)</code> | replace value by another in an array                                                                                                                                                                                      |
| convert array to string  | <code>char(array)</code>                                                | Converts array to its string representation.                                                                                                                                                                              |



| Function                | Syntax                                                     | Purpose                                                                                                                                                               |
|-------------------------|------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| convert array to string | <code>array_to_string(array, delimiter [, nullstr])</code> | Converts array to a list of elements separated by 'delimiter'. Nested arrays are flattened. NULL values are either ignored or replaced by <i>nullstr</i> if provided. |
| convert string to array | <code>string_to_array(str, format)</code>                  | Converts the string representation of an array (e.g., ' <code>int[1,2,NULL]</code> ') into an array. The format string is the same as in <code>cast_to_array</code> . |

### Function Examples

| Function                     | SQL statement                                             | Result                                          |
|------------------------------|-----------------------------------------------------------|-------------------------------------------------|
| type constructor             | <code>int</code>                                          | int array with values 1 and 2                   |
| constructor with no type     | <code>array[1,2]</code>                                   | bigint array with values 1 and 2                |
| <code>array_cat</code>       | <code>array_cat(array[2, 3], array[4, 5])</code>          | <code>array[2, 3, 4, 5]</code>                  |
| <code>array_length</code>    | <code>array_length(array[2, 3])</code>                    | 2                                               |
| <code>cast_to_array</code>   | <code>cast_to_array(array[2.5, 3.4], 'ARRAY(INT)')</code> | int                                             |
| <code>array_position</code>  | <code>array_position(array[1, 2, 1], 1)</code>            | 1                                               |
|                              | <code>array_position(array[1, 2, 3, 4, 1], 1, 3)</code>   | 5                                               |
| <code>array_positions</code> | <code>array_positions(array[1, 2, 3, 4, 1], 1)</code>     | int                                             |
| <code>array_append</code>    | <code>array_append(array[1,2], 3)</code>                  | <code>array[1,2,3]</code>                       |
| <code>array_prepend</code>   | <code>array_prepend(0, array[1,2])</code>                 | <code>array[0,1,2]</code>                       |
| <code>array_remove</code>    | <code>array_remove(array[1,2], 1)</code>                  | <code>array[2]</code>                           |
| <code>array_replace</code>   | <code>array_replace(array[1,2,4],4,3)</code>              | <code>array[1,2,3]</code>                       |
| <code>softmax</code>         | <code>softmax(float)</code>                               | <code>float[0.25, 0.25, 0.25, 0.0, 0.25]</code> |
| <code>array_sum</code>       | <code>array_sum(int)</code>                               | 6                                               |
| <code>array_min</code>       | <code>array_min(int)</code>                               | -2                                              |
| <code>array_argmax</code>    | <code>array_argmax(int)</code>                            | 1                                               |
| <code>char()</code>          | <code>char(array[1,2])</code>                             | ' <code>bigint[1,2]</code> '                    |
| <code>array_to_string</code> | <code>array_to_string(int[1,2,3,NULL,5], ',', '*')</code> | ' <code>1,2,3,*,5</code> '                      |
| <code>string_to_array</code> | <code>string_to_array('int[1,2]', 'ARRAY(INT)')</code>    | int                                             |

### Array Operators

| Operator               | Syntax                              | Purpose                                                             |
|------------------------|-------------------------------------|---------------------------------------------------------------------|
| contains scalar        | <code>array @&gt; scalar</code>     | Tell whether the array contains the scalar.                         |
| contains elements      | <code>array @&gt; array</code>      | Tell whether the left array contains all elements from right array. |
| scalar is contained by | <code>scalar &lt;@ array</code>     | Tell whether the array contains the scalar.                         |
| array is contained by  | <code>array &lt;@ array</code>      | Tell whether the right array contains all elements from left.       |
| overlap                | <code>array &amp;&amp; array</code> | Tell whether any of the elements between the arrays are common.     |

| Operator        | Syntax                                     | Purpose                                                                                                         |
|-----------------|--------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| access operator | <code>array[element_index]</code>          | Returns the element stored in the given index position. Index starts at 1. Access out of bounds returns a NULL. |
| slice operator  | <code>array[left_index:right_index]</code> | Returns the subarray ranging from left_index to right_index inclusive. Index starts at 1.                       |

### About the Array Slice Operator

- Each index starts at 1.
- If the array or either index value is NULL, the result of slicing is NULL.
- If you exclude the left index, it is equivalent to using 1. If you exclude the right index, it is equivalent to `len(array)`. If you exclude both, it returns the full array.
- Values below zero for the left index convert to 1 and values beyond the array length for the right index convert to `len(array)`. `arrayVal[-1:4]` is equivalent to `arrayVal[1:4]`, and `int[] (1, 2, 3) [1:6]` is equivalent to `int[] (1, 2, 3) [1:3]`.
- Ranges completely out of array bounds return an empty array. `array[1, 2] [4:5] = array[]`
- Sequential slices slice each dimension of multidimensional arrays. `array[array[1, 2, 3], array[4, 5, 6]] [1:1] [2:3] = array[array[2,3]]`
- More than N sequential slices of an N-dimensional array, for example, three sequential slices on a two-dimensional array return an empty array. `array[1, 2, 3] [:] [:] = []`
- When slicing multidimensional arrays, you cannot combine slicing with the access operator. Any access operator `[n]` converts to `[:n]`. `arrayVal[4] [1:6] = arrayVal[:4] [1:6]`
- Slicing an array of tuples such as `tuple<<int,int>>[tuple<<int,int>>(1,2),tuple<<int,int>>(3,4)] [1:1] [2:3]` slices `[1:1]` on the array and `[2:3]` on the tuple elements within the sliced array, and returns the value `tuple<<int>>[tuple<<int>>(2)]`.

### Operator Examples

| Operator        | SQL Predicate                                          | Result |
|-----------------|--------------------------------------------------------|--------|
| Contains        | <code>array[1, 4, 3] @&gt; array[3, 1]</code>          | True   |
|                 | <code>array[1, 4, 3] @&gt; array[3, 1, 3]</code>       | True   |
|                 | <code>array[3, 1, 3] @&gt; 1</code>                    | True   |
| Is contained by | <code>array[2, 5] &lt;@ array[2, 5, 3, 7, 9]</code>    | True   |
|                 | <code>array[2, 2, 5] &lt;@ array[2, 5, 3, 7, 9]</code> | True   |
|                 | <code>1 &lt;@ array[2, 5, 1]</code>                    | True   |
| Overlap         | <code>array[1 ,2, 3] &amp;&amp; array[1 ,6, 4]</code>  | True   |

### Multidimensional arrays

Multidimensional arrays are supported. As seen in these examples, each dimension, or length, at each level does not have to be the same. NULL values are allowed at each dimension of an array.

### Multidimensional Examples

| Function                                 | SQL statement                                     | Result                                             |
|------------------------------------------|---------------------------------------------------|----------------------------------------------------|
| type constructor two-dimensional         | <code>int[] [] (int[] (1, NULL), array[1])</code> | two-dimensionalINTarray with values[1, NULL]and[1] |
| constructor with no type two-dimensional | <code>array[int[] (1), array[int(2)]]</code>      | two-dimensionalINTarray with values[1]and[2]       |

| Function                                     | SQL statement                                          | Result                                                      |
|----------------------------------------------|--------------------------------------------------------|-------------------------------------------------------------|
| two-dimensional array with NULL values       | <code>array[array[1, NULL], NULL]</code>               | two-dimensionalBIGINTarray with values[1, NULL]andNULL      |
| two-dimensional array with different lengths | <code>array[array[1], array[1,2], array[1,2,3]]</code> | two-dimensionalBIGINTarray with values[1],[1,2], and[1,2,3] |

### Related Links

[Matrix Functions and Operators](#)

[Tuple Functions and Operators](#)

[Math Functions and Operators](#)

[Sorted Aggregate Functions](#)

[Data Types](#)

[Query Ocient](#)

## 10.5 Character and Binary Functions

All string functions support Unicode characters. Functions that transform character case are locale-sensitive.

These functions support the UTF-8 format:

- RTRIM
- LTRIM
- LEFT
- RIGHT
- TRIM
- TRANSLATE
- RPAD
- LPAD
- SUBSTRING

Index position values for character and string functions begin at position 1. For example, the "H" in the string "Hello" is at position 1.

### ASCII

Returns the ASCII code value of the leftmost character of the character value.

The ASCII function only supports ASCII characters in the input argument.

### Syntax

`ASCII(char)`

### Example

```
SELECT ASCII('a');
```

*Output:* 97

### Example

```
SELECT ASCII('bc');
```

Output: 98

## BIT\_LENGTH

Returns the length of the character value in bits.

### Syntax

```
BIT_LENGTH(char)
```

### Example

```
SELECT BIT_LENGTH('a');
```

\*Output: 8

### Example

```
SELECT BIT_LENGTH('ab');
```

Output: 16

### Example

```
SELECT BIT_LENGTH('ab4');
```

Output: 24

## BTRIM

Alias for [Character / Binary Functions](#).

## CHR

Converts an integer value to a string.

The value is first sign-extended to 8 bytes. Then, if it can be represented as 1 byte, a string is returned with that one byte.

Otherwise, a string is returned of length 2 bytes, 4 bytes, or 8 bytes with the bytes set to the big-endian representation of the integer, depending on how many high order zero bytes there are in the integer.

### Syntax

```
CHR(integer)
```

### Example

```
SELECT CHR(97);
```

Output: a

## CHAR\_LENGTH

Alias for [Character / Binary Functions](#).

## CHARACTER\_LENGTH

Alias for [Character / Binary Functions](#).

## CONCAT

Concatenates two or more values, all of which must be binary, hash, or string data types.

For strings, as long as one argument is a character value, the other arguments are implicitly cast to a character value.

This function is equivalent to the || operator.

### Syntax

```
CONCAT(value1, value2 [, ...])
```

| Arguments              | DataType          | Description                                                                                                            |
|------------------------|-------------------|------------------------------------------------------------------------------------------------------------------------|
| value1, value2 [, ...] | BINARY,HASHorCHAR | Two or more values to concatenate.If any argument is a character value, the others are also cast to a character value. |

### Example

```
SELECT CONCAT('ocient',' ','data',' ','warehouse');
```

Output: ocient data warehouse

### Example

```
SELECT 'ocient' || ' data warehouse';
```

Output: ocient data warehouse

## ENDSWITH

Returns true if x ends with y and false otherwise.

### Syntax

```
ENDSWITH(char1, char2)
```

| Argument | DataType | Description                                                       |
|----------|----------|-------------------------------------------------------------------|
| char1    | CHAR     | A string to evaluate for whether it ends withchar2.               |
| char2    | CHAR     | A string to evaluate for whetherchar1ends with it as a substring. |

### Example

```
SELECT ENDSWITH('ocient data warehouse','warehouse');
```

Output: true

### Example

```
SELECT ENDSWITH('ocient data warehouse','db');
```

Output: \*false

### Example

```
SELECT ENDSWITH('tamaño','o');
```

Output: true

## INITCAP

For each word in the specified string, capitalize the first character if it is alphabetic. The system defines words as alphanumeric strings separated by non-alphanumeric characters. The system converts all other alphabetic characters to lowercase.

### Syntaxes

```
INITCAP(char_string)
INITCAP(char_string, delimiter)
```

| Argument    | Data Type | Description                                                                                                                                                                                                                                                                                                                                      |
|-------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| char_string | CHAR      | A string to evaluate for initial capitalization.                                                                                                                                                                                                                                                                                                 |
| delimiter   | CHAR      | Optional. One or more non-alphanumeric characters that specify the delimiter between words. The characters include all the ASCII printable non-alphanumeric characters except backquotes and equals=’. Non-ASCII characters are not included. ≤ If you specify alphabetic characters for the delimiter, the function converts them to lowercase. |

### Example

```
SELECT INITCAP('ocient');
```

Output: Ocient

## INSTR

Returns the index position of the first occurrence where the character value **char\_substring** appears in the character value **char** by ignoring the case.

### Syntax

```
INSTR(char, char_substring)
```

| Argument       | Data Type | Description                                                                            |
|----------------|-----------|----------------------------------------------------------------------------------------|
| char           | CHAR      | A string to evaluate for the first index position where <b>char_substring</b> appears. |
| char_substring | CHAR      | A string to evaluate for where it first appears in <b>char</b> .                       |

### Example

```
SELECT INSTR('ocient dw dw', 'ocient');
```

\*Output: 1

```
SELECT INSTR('ocient dw dw', 'dw');
```

\*Output: 8

**JSON\_EXTRACT\_PATH\_TEXT**

Returns the value for the key-value pair referenced by a series of path elements in a JSON string.

**Syntax**

```
JSON_EXTRACT_PATH_TEXT (
    json_string, path [, path2 [, ...] ] [, null_if_invalid ] )
```

| Argument                | DataType | Description                                                                                                                                                                                                                                                                                                                               |
|-------------------------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| json_string             | CHAR     | A JSON string.                                                                                                                                                                                                                                                                                                                            |
| path [, path2 [, ...] ] | CHAR     | One or more path elements in a JSON string. You must include at least one path element up to a maximum of five, meaning the function can extract from paths nested up to five levels deep in the JSON string. Path elements are case-sensitive. If a specified path element does not exist in the JSON string, the function returns NULL. |
| null_if_invalid         | BOOLEAN  | Optional. If you set this value to TRUE, the function returns NULL if the JSON string is invalid. If you set this value to FALSE, the function returns an error when the JSON string is invalid. The default value is FALSE.                                                                                                              |

**Examples****Extract Values from Nested Paths**

This example extracts the value based on two specified paths in the JSON string, **n4** and **n6**.

```
SELECT JSON_EXTRACT_PATH_TEXT(
    '{"n2":{"n3":1},"n4":{"n5":99,"n6":"circle"}}',
    'n4', 'n6'
);
```

Output: "circle"

**Return NULL from Invalid JSON**

In this example, the query requests the same nested paths, but the JSON is invalid because all of the keys lack quotation marks.

The query returns NULL because the **null\_if\_invalid** argument equals TRUE.

```
SELECT JSON_EXTRACT_PATH_TEXT(
    '{n2:{n3:1},n4:{n5:99,n6:"circle"}}',
    'n4', 'n6',
    TRUE
);
```

Output: NULL

**LCASE**

Alias for [Character / Binary Functions](#).

**LEFT**

Return the number of characters in the string equal of the value **integer**. If **integer** is negative, the function returns all but the last **integer** characters.

**Syntax**

```
LEFT(char, integer)
```

| Argument | DataType | Description                                                                                                                                                     |
|----------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| char     | CHAR     | A string to be modified by returning the number of characters equal to <b>integer</b> .                                                                         |
| integer  | INT      | The number of characters to return from the <b>char</b> string. If <b>integer</b> is negative, the function returns all but the last <b>integer</b> characters. |

**Example**

```
SELECT LEFT('ocient data warehouse', 8);
```

\*Output: ocient d

**Example**

```
SELECT LEFT('ocient data warehouse', -3);
```

\*Output: ocient data wareho

**LENGTH**

Alias for CHAR\_LENGTH and CHARACTER\_LENGTH.

Returns the length of the value. For character data types, this value is in terms of characters. For binary data types, this value is in terms of bytes.

**Syntax**

```
LENGTH(character_or_binary_value)
```

**Example**

```
SELECT LENGTH('ocient data warehouse');
```

Output: 21

**LOCATE**

Alias for POSITION.

Returns the index position of the first occurrence of the character value **substring** in the character value **string**.

Optionally, you can also include an additional value **offset** to offset the LOCATE function by the specified number of spaces.

Index positions begin at 1.

**Syntax**

```
LOCATE(substring, string [, offset] )
```



| Argument  | DataType | Description                                                                                                                                                                                                |
|-----------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| substring | CHAR     | A substring to be found for its first position in string.                                                                                                                                                  |
| string    | CHAR     | A string to be evaluated for the first position of <b>substring</b> .                                                                                                                                      |
| offset    | INT      | Optional. A number of index positions in <b>string</b> to offset the search for the <b>substring</b> value. This index position must be a positive integer, and it starts from the left of <b>string</b> . |

**Example**

```
SELECT LOCATE('ware', 'ocient data warehouse');
```

\*Output: 13

**Example**

```
SELECT LOCATE('e', 'ocient');
```

\*Output: 4

**Example**

In this example, the index starts at position 5. This means the function skips the initial “e” in the string. Instead, it returns the second e at index position 15.

```
SELECT LOCATE('e', 'ocient database', 5);
```

Output: 15

**LOWER**

Alias for LCASE.

Convert string to lowercase.

**Syntax**

```
LOWER(char)
```

**Example**

```
SELECT LOWER('Ocient');
```

Output: ocient

```
SELECT
```

```
LOWER('OCIENT');
```

Output: ocient

**LPAD**

Pad the input text to the specified length with the pad string on the left side. If text is longer than length, it is truncated to length characters. If the argument pad is not provided, the space character is used.

**Syntax**

```
LPAD(string, length [, pad_character] )
```

| Argument      | DataType | Description                                                                                                                                                                                                           |
|---------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| string        | CHAR     | A string to be padded on its left side with additional characters, so that it equals the <code>length</code> value.                                                                                                   |
| length        | INT      | The number of characters for <code>string</code> to total. If the <code>string</code> is already longer than <code>length</code> , the function truncates <code>string</code> to equal the <code>length</code> value. |
| pad_character | CHAR     | Optional. A string of one or more characters to use to pad <code>string</code> to equal the <code>length</code> value. If not provided, <code>pad_character</code> uses a whitespace character to pad.                |

#### Example

```
SELECT LPAD('ocient data warehouse',30,'ab');
```

\*Output: ababababaoocient data warehouse

#### Example

```
SELECT LPAD('ocient data warehouse',6);
```

\*Output: ocient

### LTRIM

Removes leading blanks from the string value `string`.

Alternatively, you can specify a second string value `trim_character`. If you specify the `trim_character` value, the LTRIM function removes all leading instances of the `trim_character` value from the `string`.

#### Syntax

```
LTRIM(string [, trim_character] )
```

| Argument       | DataType | Description                                                                                                                                                                                                                                                                                                 |
|----------------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| string         | CHAR     | A string to be trimmed on its left side.                                                                                                                                                                                                                                                                    |
| trim_character | CHAR     | Optional. A string of one or more characters to be trimmed from the left side of <code>string</code> . Each character is trimmed individually, not as a cohesive substring. If you do not specify this argument, this value defaults to removing all whitespace from the left side of <code>string</code> . |

#### Example

```
SELECT LTRIM(' ocient');
```

\*Output:: \*ocient

#### Example

```
SELECT LTRIM('aaaaaocient','a');
```

\*Output: ocient

#### Example

In this example, all characters “abeo” are removed from the left side of the string.

```
SELECT LTRIM('aaeabe ocient','abeo ');
```

\*Output: cient

### MD5

Returns the hexadecimal string (all lowercase) representing the md5 hash of `char`.

#### Syntax

```
MD5(char)
```

#### Example

```
SELECT md5('ocient');
```

Output: 438f03cf6e9ddf8793e02db25f2d2f88

### MID

Alias for [Character / Binary Functions](#).

### OCTET\_LENGTH

Returns the length in bytes of a character or binary value.

#### Syntax

```
OCTET_LENGTH(value)
```

#### Example

```
SELECT OCTET_LENGTH('a');
```

\*Output: \*1

#### Example

```
SELECT OCTET_LENGTH('ocient');
```

\*Output: 6

### POSITION

Alias for [Character / Binary Functions](#).

### REGEXP\_COUNT

Searches a string for all occurrences of a regular expression pattern. The function returns an integer representing the number of times the regular expression pattern occurs in the string.

#### Syntax

```
REGEXP_COUNT( string, pattern [, position [, parameters ] ] )
```

| Argument   | DataType | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| string     | CHAR     | The string to search, specified as a character string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| pattern    | CHAR     | The regular expression pattern to use in the search, specified as a character string.                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| position   | INT      | Optional. The position, specified as a positive integer, that represents the position within the <b>string</b> where to begin the search, based on the number of characters. If this integer exceeds the number of characters in <b>string</b> , then the function returns 0. The default value is 1.                                                                                                                                                                                                                        |
| parameters | CHAR     | Optional. Parameters, specified as a character string, that contains one or more characters representing regular expression options for pattern matching. Supported options are: <b>c</b> — Perform case-sensitive matching. The <b>Ocient</b> System enables this type of matching by default. <b>i</b> — Perform case-insensitive matching. <b>p</b> — Interpret the pattern using the Perl Compatible Regular Expression (PCRE) dialect. For details, see PCRE. The Ocient System enables this interpretation by default. |

### Example

This example searches the 'ABABDaSGRESaB' string for the count of occurrences of the 'AB' string by ignoring the case sensitivity. The search starts at position 1.

```
SELECT REGEXP_COUNT('ABABDaSGRESaB', 'AB', 1, 'i');
```

Output: 3

### REGEXP\_INSTR

Searches a string using a regular expression pattern and returns an integer representing the start position or end position of the substring that matches.

The function returns 0 if no match is found.

The REGEXP\_INSTR function is similar to the [Character / Binary Functions](#) function, but it allows greater precision with regular expressions.

### Syntax

```
REGEXP_INSTR( source_string, pattern [, position [, occurrence] [, option [, parameters] ] ] )
```

| Argument                | DataType | Description                                                                                                                                                                                                                                                                                                                                                                      |
|-------------------------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>string</code>     | CHAR     | The string to search, specified as a character string. Note that SQL escape sequences for string literals can override regular expression escape sequences. For details, see <a href="#">General SQL Syntax</a> .                                                                                                                                                                |
| <code>pattern</code>    | CHAR     | The regular expression pattern to use in the search, specified as a character string.                                                                                                                                                                                                                                                                                            |
| <code>position</code>   | INT      | Optional. A positive integer that represents the position within the <code>string</code> where to begin the search, based on the number of characters. This value alters only the start position for the match, not the returned string position. If this integer exceeds the number of characters in <code>string</code> , then the function returns 0. The default value is 1. |
| <code>occurrence</code> | INT      | Optional. A positive integer that represents the occurrence of a positive pattern match to return. The default value is 1, which means the function returns the first substring that matches the regular expression pattern.                                                                                                                                                     |
| <code>option</code>     | INT      | Optional. An integer that specifies whether to return the position of the matching start position or the end position. Your choices are: 0: Returns the start position of the match. 1: Returns the end position of the match + 1. The function treats any nonzero integer value as 1. The default value is 0.                                                                   |

| Argument   | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| parameters | CHAR      | Optional. A string containing one or more characters that represents the regular expression options for pattern matching. Supported options are: <b>c</b> — Perform case-sensitive matching. The Ociant System enables this type of matching by default. <b>i</b> — Perform case-insensitive matching. <b>e</b> — Extract the substring using a regular expression subpattern. This subpattern is enclosed in parentheses in the regular expression. The function uses the full regular expression pattern for the match but returns only the first subpattern match. <b>p</b> — Interpret the pattern using the Perl Compatible Regular Expression (PCRE) dialect. For details, see PCRE. The Ociant System enables this interpretation by default. |

### Example

The query searches the website URL for the substring preceded by the # character. The function includes optional arguments to start the search at position 9 and match the first occurrence. The last specified optional argument directs the function to return the ending position of the matching substring.

```
SELECT REGEXP_INSTR(
  'https://docs.ocient.com/character-binary-functions#J28jB',
  '#\w+',
  9,
  1,
  1
);
```

Output: 57

### REGEXP\_REPLACE

Searches a string for all occurrences of a regular expression pattern. The function replaces every match occurrence of the pattern with a new string.

The REGEXP\_REPLACE function is similar to the [Character / Binary Functions](#) and [Character / Binary Functions](#) functions, but it allows greater precision with regular expressions.

#### Syntax

```
REGEXP_REPLACE( string, pattern [, replace_string [ , position [, parameters ] ] ] )
```

| Argument                    | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>string</code>         | CHAR      | The string to search, specified as a character string. Note that SQL escape sequences for string literals can override regular expression escape sequences. For details, see <a href="#">General SQL Syntax</a> .                                                                                                                                                                                                                                                                                               |
| <code>pattern</code>        | CHAR      | The regular expression pattern to use in the search, specified as a character string.                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>replace_string</code> | CHAR      | Optional. A string to replace all occurrences of the regular expression pattern <code>instring</code> . This string can include references to capture groups in the regular expression <code>pattern</code> . To reference capture groups, use <code>a\$</code> followed by the group number. For example, <code>\$1</code> references the first capture group, and <code>\$2</code> references the second. The default value is an empty string <code>''</code> .                                              |
| <code>position</code>       | INT       | Optional. A positive integer that represents the position within the string to begin the search, based on the number of characters. If this integer exceeds the number of characters <code>instring</code> , then the function returns the original <code>string</code> . The default value is 1.                                                                                                                                                                                                               |
| <code>parameters</code>     | CHAR      | Optional. A string containing one or more characters representing regular expression options for pattern matching. Supported options are: <code>c</code> — Perform case-sensitive matching. The Ocint System enables this type of matching by default. <code>i</code> — Perform case-insensitive matching. <code>p</code> — Interpret the pattern using the Perl Compatible Regular Expression (PCRE) dialect. For details, see <a href="#">PCRE</a> . The Ocint System enables this interpretation by default. |

## Examples

### Replace Text Using a Substring

The query replaces the matching substring `'#J28jB'` with the characters `'#OFqD_'`. The search begins at position 9.

```
SELECT REGEXP_REPLACE(
  'https://docs.ocient.com/character-binary-functions#J28jB',
  '#\w+',
  '#OFqD_',
  9
);
```

Output: [https://docs.ocient.com/character-binary-functions#0FqD\\_](https://docs.ocient.com/character-binary-functions#0FqD_)

### Replace Text Using Multiple Capture Groups

This example uses three capture groups to take an unformatted phone number and convert it into the (XXX) XXX-XXXX format. The `replace_string` argument references each capture group as \$1, \$2, and \$3.

```
SELECT REGEXP_REPLACE(
  '5558675309',
  '(\d{3})(\d{3})(\d{4})',
  '($1) $2-$3'
);
```

Output: (555) 867-5309

### REGEXP\_SUBSTR

Returns one substring from a string that matches a specified regular expression pattern.

REGEXP\_SUBSTR is similar to the [Character / Binary Functions](#) function, but it allows greater precision with regular expressions.

If the pattern produces no matches, the function returns an empty string.

#### Syntax

```
REGEXP_SUBSTR( string, pattern [, position [, occurrence [, parameters ] ] ] )
```

| Argument   | DataType | Description                                                                                                                                                                                                                                                                                                |
|------------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| string     | CHAR     | The string to search, specified as a character string. Note that SQL escape sequences for string literals can override regular expression escape sequences. For details, see <a href="#">General SQL Syntax</a> .                                                                                          |
| pattern    | CHAR     | The regular expression pattern to use in the search, specified as a character string.                                                                                                                                                                                                                      |
| position   | INT      | Optional. A positive integer that represents the position within the string to begin the search, based on the number of characters. If this integer is greater than the number of characters in <code>string</code> , then the function returns the original <code>string</code> . The default value is 1. |
| occurrence | INT      | Optional. A positive integer that represents the occurrence of a positive pattern match to return. If this value exceeds the number of matches, then the function returns NULL. The default value is 1, which means the function returns the first substring that matches the regular expression pattern.  |



| Argument   | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| parameters | CHAR      | Optional. A string containing one or more characters representing regular expression options for pattern matching. Supported options are: <b>c</b> — Perform case-sensitive matching. The Ociant System enables this type of matching by default. <b>i</b> — Perform case-insensitive matching. <b>e</b> — Extract the substring using a regular expression subpattern. This subpattern is enclosed in parentheses in the regular expression. The function uses the full regular expression pattern for the match, but it returns only the first subpattern match. If there is no subexpression in the <b>pattern</b> argument, then the <b>parameter</b> is ignored. <b>p</b> — Interpret the pattern using the Perl Compatible Regular Expression (PCRE) dialect. For details, see PCRE. The Ociant System enables this interpretation by default. |

### Example

The query returns the first occurrence of the #J28jB string by using the regular expression pattern '#\w+'. The search starts at position 9.

```
SELECT REGEXP_SUBSTR(
  'https://docs.ocient.com/character-binary-functions#J28jB',
  '#\w+',
  9,
  1
);
```

Output: #J28jB

### REPEAT

Repeats the **char** string **num** times without spaces.

#### Syntax

```
REPEAT(char, num)
```

### Example

Repeat the **a** string five times.

```
SELECT REPEAT('a', 5);
```

Output: aaaaa

### REPLACE

Replaces all occurrences of `substr_to_remove` in the character value `string` with `substr_to_replace`.

#### Syntax

```
REPLACE(string, substr_to_remove, substr_to_replace)
```

| Argument                       | Data Type | Description                                                                                                                                                                                                            |
|--------------------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>string</code>            | CHAR      | A string to alter by replacing all instances of <code>substr_to_remove</code> with <code>substr_to_replace</code> .                                                                                                    |
| <code>substr_to_remove</code>  | CHAR      | A substring to remove from <code>string</code> , replacing all instances with the <code>substr_to_replace</code> value. If <code>substr_to_remove</code> is the empty string, the system returns <code>string</code> . |
| <code>substr_to_replace</code> | CHAR      | A substring to replace all instances of <code>substr_to_remove</code> .                                                                                                                                                |

#### Example

```
SELECT REPLACE('abcbcabcbabc', 'ab', '$');
```

Output: \$c\$c\$c\$c

### REVERSE

Reverse the input string.

#### Syntax

```
REVERSE(char)
```

#### Example

```
SELECT REVERSE('abcd');
```

Output: dcba

### RIGHT

Return the number of trailing characters in the string equal to the value `integer`.

#### Syntax

```
RIGHT(char, integer)
```

| Argument             | Data Type | Description                                                                                                                                                                                   |
|----------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>char</code>    | CHAR      | A string to evaluate to return the number of trailing characters equal to <code>integer</code> .                                                                                              |
| <code>integer</code> | INT       | The number of characters of return from the end of the <code>char</code> string. If <code>integer</code> is negative, the function returns all but the first <code>integer</code> characters. |

#### Example

```
SELECT RIGHT('ociant data warehouse', 8);
```

\*Output: arehouse

### Example

```
SELECT RIGHT('ociant data warehouse', -3);
```

\*Output: ent data warehouse

## RPAD

Pad the input text to the specified length with the pad string on the right side. If text is longer than length, it is truncated to **length** characters. If the argument pad is not provided, the space character is used.

## Syntax

```
RPAD(character_value_text, integral_value_length [, character_value_pad] )
```

| Argument      | Data Type | Description                                                                                                                                                                                                           |
|---------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| string        | CHAR      | A string to be padded on its right side with additional characters, so that it equals the <code>length</code> value.                                                                                                  |
| length        | INT       | The number of characters for <code>string</code> to total. If the <code>string</code> is already longer than <code>length</code> , the function truncates <code>string</code> to equal the <code>length</code> value. |
| pad_character | CHAR      | Optional. A string of one or more characters to use to pad <code>string</code> to equal the <code>length</code> value. If not provided, <code>pad_character</code> uses a whitespace character to pad.                |

### Example

```
SELECT RPAD('ocient data warehouse',30);
```

\*Output: ocient data warehouse

### Example

```
SELECT RPAD('ocient data warehouse',30,'ab');
```

```
*Output: ocient data warehouseababababa
```

## RSUBSTRING

Returns the substring from the right side of a string, based on a specified length.

## Syntax

```
RSUBSTRING(string, integer)
```

| Argument | DataType | Description                                                              |
|----------|----------|--------------------------------------------------------------------------|
| string   | CHAR     | A string to be evaluated to return a substring based on theintegervalue. |

| Argument | DataType | Description                                                                                                     |
|----------|----------|-----------------------------------------------------------------------------------------------------------------|
| integer  | INT      | The number of characters to return from the right side of the <b>string</b> value. This value must be positive. |

**Example**

```
SELECT RSUBSTRING('ocient data warehouse',14);
```

Output: data warehouse

**RTRIM**

Removes trailing blanks from the string value **string**.

Alternatively, you can specify a second string value **trim\_character**. If you specify the **trim\_character** value, the RTRIM function removes all trailing instances of the **trim\_character** value from the **string**.

**Syntax**

```
RTRIM(string [, trim_character] )
```

| Argument       | DataType | Description                                                                                                                                                                                                                                                                                       |
|----------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| string         | CHAR     | A string to be trimmed on its right side.                                                                                                                                                                                                                                                         |
| trim_character | CHAR     | Optional. A string of one or more characters to be trimmed from the right side of <b>string</b> . Each character is trimmed individually, not as a cohesive substring. If you do not specify this argument, this value defaults to removing all whitespace from the right side of <b>string</b> . |

**Example**

```
SELECT RTRIM('ocient ');
```

\*Output: ocient

**Example**

```
SELECT RTRIM('ocientaaaaa','a');
```

\*Output: ocient

**SHA1**

Uses the SHA-1 cryptographic hash function to convert a string into a 40-character string representing the hexadecimal value of a 160-bit checksum.

**Syntax**

```
SHA1(string)
```

| Argument | DataType | Description                                                      |
|----------|----------|------------------------------------------------------------------|
| string   | CHAR     | A string to convert using the SHA-1 cryptographic hash function. |

**Example**

```
SELECT SHA1('Ocient');
```

Output: 0c2a9a042b9f047f875c3414e7a4f4c53efbe082

**SPACE**

Returns a string of repeated spaces equal to the number value, **repeat**. You can join this to another string by using the [Character / Binary Functions](#) function.

**Syntax**

```
SPACE(repeat)
```

**Example**

```
SELECT CONCAT(SPACE(10), 'end');
```

Output: end

**SPLIT\_PART**

Split the value **string** based on the **delimiter** value. The function returns a substring from the split operation based on the **index** value (starting from 1).

**Syntax**

```
SPLIT_PART(string, delimiter, index)
```

| Argument  | DataType | Description                                                                 |
|-----------|----------|-----------------------------------------------------------------------------|
| string    | CHAR     | A string to be split based on the <b>delimiter</b> value.                   |
| delimiter | CHAR     | The character to use as a delimiter in <b>string</b> .                      |
| index     | INT      | The index of the substring to return. The first index position starts at 1. |

**Example**

```
SELECT SPLIT_PART('id|name|address', '|', 1);
```

\*Output: id

**Example**

```
SELECT SPLIT_PART('id|name|address', '|', 2);
```

\*Output: name

**Example**

```
SELECT SPLIT_PART('id,name,address', ',', 3);
```

\*Output: address

**SPLIT\_TO\_ARRAY**

Splits a string into an array of substrings.

**Syntax**

```
SPLIT_TO_ARRAY(string, [ delimiter ])
```

| Argument  | DataType | Description                                                                        |
|-----------|----------|------------------------------------------------------------------------------------|
| string    | CHAR     | A string to split into an array of substrings.                                     |
| delimiter | CHAR     | Optional. The delimiter that divides the string. The default value is a comma (,). |

**Example**

This example specifies | as the delimiter to split the input string.

```
SELECT SPLIT_TO_ARRAY('AB|CD|EF', '|');
```

Output: ['AB', 'CD', 'EF']

**STARTSWITH**

Returns true if string starts with substring and false otherwise.

**Syntax**

```
STARTSWITH(string, substring)
```

| Argument  | DataType | Description                                     |
|-----------|----------|-------------------------------------------------|
| string    | CHAR     | A string to search.                             |
| substring | CHAR     | A substring to find at the beginning of string. |

**Example**

```
SELECT STARTSWITH('ocient', 'o');
```

\*Output: true

**Example**

```
SELECT STARTSWITH('ocient', 'c');
```

Output: false

**STRPOS**

Equivalent to using [Character / Binary Functions](#) as LOCATE(substring, string). Note the reversed argument order.

**Syntax**

```
STRPOS(string, substring)
```

| Argument  | DataType | Description                                                           |
|-----------|----------|-----------------------------------------------------------------------|
| string    | CHAR     | A string to be evaluated for the first position of <b>substring</b> . |
| substring | CHAR     | A substring to be found for its first position in <b>string</b> .     |

**Example**

```
SELECT STRPOS('ocient data warehouse', 'house');
```

Output: 17

**SUBSTR**

Alias for **Character / Binary Functions**.

**SUBSTRING**

Alias for SUBSTR and MID.

Returns the substring of a character or binary value that starts with the position specified by the second argument and that ends with the position specified by one less than the sum of the second and third arguments. When the sum of the second and third arguments is less than two, the function returns the empty string.

**Syntax**

```
SUBSTRING(string, start_position [, length] )
```

| Argument       | DataType | Description                                                                                                                                                                                |
|----------------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| string         | CHAR     | A string to be truncated based on the <b>start_position</b> and <b>end_position</b> values.                                                                                                |
| start_position | INT      | The starting position to return a substring. The first index position starts at 1.                                                                                                         |
| length         | INT      | Optional. The number of characters from the <b>start_position</b> to include in the returned substring. If unspecified, SUBSTRING returns all characters after the <b>start_position</b> . |

**Example**

```
SELECT SUBSTRING('ocient data warehouse', 8);
```

\*Output: data warehouse

**Example**

```
SELECT SUBSTRING('ocient data warehouse', 8, 4);
```

Output: data

**TO\_CHAR**

Converts a numeric, date, or timestamp value into a **CHAR** date type.

TO\_CHAR(value, format)

| Argument | Data Type                             | Description                                                                                                       |
|----------|---------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| value    | DATE, TIMESTAMP, or all numeric types | A numeric, date, or timestamp value to be converted into a CHAR date type.                                        |
| format   | CHAR                                  | The format used for the CHAR conversion. For information on data type formats, see the Formatting Functions page. |

#### Example

```
SELECT TO_CHAR(20200610132514/1000000, '9999-99-99');
```

\*Output: 2020-06-10

#### Example

```
SELECT TO_CHAR(20200610132514%1000000, '99:99:99');
```

Output: 13:25:14

## TRANSLATE

Replaces specified characters in a provided string with a separate set of characters.

Characters specified in the `char_to_remove` set are replaced with characters in the `char_to_replace` set based on the corresponding index position.

#### Syntax

TRANSLATE(string, char\_to\_remove, char\_to\_replace)

| Argument        | Data Type | Description                                                                                                                                                                                                                                                                                                                                                       |
|-----------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| string          | CHAR      | A string to have specific characters transposed based on the <code>char_to_remove</code> and <code>char_to_replace</code> values.                                                                                                                                                                                                                                 |
| char_to_remove  | CHAR      | The characters to be removed from <code>string</code> . Characters specified in the <code>char_to_remove</code> set are replaced with characters in the <code>char_to_replace</code> set based on the corresponding index position. If <code>char_to_remove</code> is longer than <code>char_to_replace</code> , occurrences of the extra characters are removed. |
| char_to_replace | CHAR      | The characters to replace the removed characters in <code>char_to_remove</code> .                                                                                                                                                                                                                                                                                 |

#### Example

```
SELECT TRANSLATE('abcdef', 'ace', 'ghi');
```

Output: gbhdif



## TRIM

Alias for BTRIM.

Trim leading and trailing blanks from the string.

Alternatively, you can specify a second string value `trim_char`. If a `trim_char` value is provided, the TRIM function removes all leading and trailing instances of the `trim_char` value from the string.

### Syntax

```
TRIM(string [, trim_char] )
```

| Argument               | Data Type | Description                                                                                                                          |
|------------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------|
| <code>string</code>    | CHAR      | A string to be trimmed of leading and trailing whitespace, or any other characters specified by the <code>trim_char</code> argument. |
| <code>trim_char</code> | CHAR      | Optional. If specified, this removes an alternative substring from the start or end of the <code>string</code> value.                |

### Example

```
SELECT TRIM(' trimmed string ');
```

Output: trimmed string

### Example

```
SELECT TRIM('aaaaaaatrimmed stringaaaaaaa', 'a');
```

Output: trimmed string

## UCASE

Alias for [Character / Binary Functions](#).

## UPPER

Alias for UCASE.

Convert string to upper case.

### Syntax

```
UPPER(character_value)
```

### Example

```
SELECT UPPER('ocient');
```

Output: OCIENT

### Related Links

[General SQL Syntax](#)

[Formatting Functions](#)

## 10.6 Data Types

The Ocient® System supports a variety of data types for SQL functions and querying. For information on supported data types, see [Data Types](#). Data type requirements for a specific function can be found in [SQL Reference](#) section.

SQL statements, such as DDL and DCL commands, use generic data types numeric and string. For details, see [Data Types](#).

### Data Types for SQL Functions

The following data types are supported in table columns and queries.

This table provides more information on the data types supported by the Ocient System. The example value column gives SQL input examples for the respective data types.

The data types in this table apply to SQL functions.

The default column values for the Tuple, Matrix, and Array values in a **CREATE TABLE** statement require an alternate syntax than the examples shown in this table.

For examples of how to specify these default values, see [CREATE TABLE SQL Statement Examples](#).

| Name                                         | Description                                                                                                                                                                                               | Example Value                                                                                                 |
|----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| ARRAY                                        | An array of the specified type in the format <code>TYPE[]</code> . The <code>TYPE</code> type can be any supported data type besides ARRAY. For example, <code>INT[]</code> creates an array of integers. | <code>INT[] (1,2,3) `` [] ()</code> can be used for empty arrays. For example: <code>BIGINT[] ()</code>       |
| BIGINT                                       | 8-byte signed integer                                                                                                                                                                                     | 9876543210                                                                                                    |
| BINARY(N) or HASH(N)                         | Fixed-length binary array with length N                                                                                                                                                                   | <code>BINARY(4):'0x01234567'</code> , <code>BINARY(3):'0xabcd'</code>                                         |
| BOOLEAN                                      | 1-byte logical Boolean (false or true)                                                                                                                                                                    | TRUE,FALSE                                                                                                    |
| CHARACTER(N) or CHAR(N) or CHAR or CHARACTER | Variable-length character string. Length N is only for compatibility. The Ocient System does not use this argument.                                                                                       | <code>CHAR(16):'This is a string'</code>                                                                      |
| DATE                                         | 4-byte calendar date (year, month, day)                                                                                                                                                                   | <code>'2020-02-02'</code> , <code>'2000-01-01'</code>                                                         |
| DECIMAL(P,S)                                 | Exact numerical with precision P and scale S                                                                                                                                                              | 123.45                                                                                                        |
| DOUBLE PRECISION or DOUBLE                   | 8-byte double precision floating-point number                                                                                                                                                             | 3.141592                                                                                                      |
| INT                                          | 4-byte signed integer                                                                                                                                                                                     | 123456789                                                                                                     |
| IPV4                                         | 4-byte Internet Protocol version 4                                                                                                                                                                        | <code>'127.0.0.1'</code>                                                                                      |
| IP                                           | 16-byte Internet Protocol version 6 (can also hold IPV4)                                                                                                                                                  | <code>'0123:4567:89ab:cdef:0123:4567:89ab:cdef'</code>                                                        |
| LINESTRING                                   | Geometric type composed of N points                                                                                                                                                                       | <code>'LINESTRING(0 0,2 0)'</code>                                                                            |
| MATRIX                                       | A one or two-dimensional mathematical matrix used for matrix calculations.                                                                                                                                | <code>A Matrix[2][2]:{ {0, 0}, {0, 0} }</code><br><code>A Matrix [2][4]:{ {0, 0, 0, 0}, {0, 0, 0, 0} }</code> |
| POINT                                        | Geometric point in 2 dimensions                                                                                                                                                                           | <code>'POINT(-87.6410 41.8841)'</code>                                                                        |
| POLYGON                                      | Geometric type composed of N closed linestrings                                                                                                                                                           | <code>'POLYGON((0 0,2 0,5 5,0 2,0 0), (0 0,1 0,2 2,0 2,0 0))'</code>                                          |
| REAL or FLOAT or SINGLE PRECISION            | 4-byte single precision floating-point number                                                                                                                                                             | 2.718                                                                                                         |
| SMALLINT                                     | 2-byte signed integer                                                                                                                                                                                     | 32767                                                                                                         |

| Name                     | Description                                                       | Example Value                                                                                               |
|--------------------------|-------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| TIME                     | 8-byte time of day in nanoseconds                                 | '12:34:56.012345678'                                                                                        |
| TIMESTAMP                | 8-byte date and time in nanoseconds with no associated time zone. | '2000-01-02 12:34:45'                                                                                       |
| TINYINT or BYTE          | 1-byte signed integer                                             | 127                                                                                                         |
| TUPLE«type1, type2, ...» | Tuple of elements of different types.                             | tuple<<INT,INT>>(1,2) An Array of Tuples: tuple<<INT,INT>>[] (tuple<<INT,INT>>(1,2), tuple<<INT,INT>>(3,4)) |
| UUID                     | 16-byte universally unique identifier                             | '01234567-89ab-cdef-1357-0123456789ab'                                                                      |
| VARBINARY(N)             | Variable-length binary array with maximum length of N             | VARBINARY(6):'0xaabbccddeeff'                                                                               |
| VARCHAR(N)               | Variable-length character string with maximum length of N         | VARCHAR(4):'This is a variable length string'                                                               |

### Data Types for SQL Statements

DDL, DCL, and General SQL Syntax statements define parameter data types as either numeric or string. These data types are simplified terms that follow these rules:

- **numeric** — Any numeric value. This value defaults to the **BIGINT** data type if it is an integer with no decimal parts. If the number includes decimal parts, it defaults to **DOUBLE**.
- **string** — A series of one or more characters. This value defaults to the **VARCHAR** data type.

### Data Type Considerations

Integer literals in SQL statements are of type **BIGINT** unless explicitly cast to another type.

Floating point literals in SQL statements are of type **DOUBLE** unless explicitly cast to another type.

**TRUE** and **FALSE** can be used in SQL statements as valid **BOOLEAN** literals.

Floating point values must be strictly numeric. Infinity and NaN values are unsupported.

The decimal type supports a maximum precision of 31 and a scale of one less than the precision.

Interval types can only be used within an expression. If the final result of an outermost expression is an interval type, the Ociant System automatically converts it to a **BIGINT** and the value loses the units information.

- “yyyy” : The year as a four-digit number.
- “MM” : The month, from 01 through 12.
- “dd” : The day of the month, from 01 through 31.
- “HH” : The hour, using a 24-hour clock from 00 to 23.
- “mm” : The minute, from 00 through 59.
- “ss” : The second, from 00 through 59.
- “FFFFFFFF” : If non-zero, the billionths of a second in a date and time value.

### Data Type Compatibility

The Ociant System casts data types to the highest common type when possible in expressions to produce an output of that type. These data-type promotions occur in these hierarchies. This promotion often applies to **COALESCE**, **CONCAT**, and **||**, and other operators.

### Numeric Types

**DOUBLE**  
**FLOAT**  
**DECIMAL**  
**BIGINT**

INT  
SMALLINT  
TINYINT

### Geospatial Types

POLYGON  
LINESTRING  
POINT

Geospatial functions might have semantic differences when you execute these functions with different types that have higher precedence than the type promotion of its arguments.

### Size Limits

Variable-length columns have limits to how large the value can be.

Data types with a 124 KiB size limit:

- `TUPLE«type1, type2, ...»` : Tuple value with a `VARCHAR(N)` or `VARBINARY(N)` type

Data types with a 512 MiB size limit:

- `TYPE[]` : The inner values of the array are subject to the size limits of the inner type.
- `LINESTRING`
- `POLYGON`
- `VARBINARY(N)`
- `VARCHAR(N)`

When you query data, if the database must analyze a value that exceeds 124 KiB at query execution time, the system increases memory usage. Large computed values are non-freeable during query execution and the system cannot offload to temporary disk. Queries that operate on computed large values with many rows can cause the system to run out of memory and result in the kill of the query.

### Array

Array is a SQL type container that stores multiple elements of the same SQL type. Arrays can have zero or any number of values. For instance, an INT array can contain many integers including NULL. Arrays in Ocient® can store any SQL type; i.e: IPV4, IP, BOOLEAN, BYTE, SHORT, INT, LONG, FLOAT, DOUBLE, TIME, TIMESTAMP, CHAR, DATE, BINARY, HASH, POINT, UUID, DECIMAL and TUPLE.

Arrays have special functions and operators that are described in the following sections:

- [Array Functions and Operators](#)
- [Array Functions and Operators](#)

### Tuple

Tuple is a SQL type container that is meant to store elements of different types. Ocient supports tuple using any of the supported SQL types; i.e: IPV4, IP, BOOLEAN, BYTE, SHORT, INT, BIGINT, FLOAT, DOUBLE, TIME, TIMESTAMP, CHAR, DATE, BINARY, HASH, POINT, UUID and DECIMAL. In Ocient, tuples can either exist as single columns or as SQL array elements. There is currently no support for storing nested tuples or tuples of arrays.

Tuple functions and operators are described in the following sections:

- [Tuple Functions and Operators](#)
- [Tuple Functions and Operators](#)

## Matrix

Matrix is a SQL-type container. It is a fixed-size, one or two-dimensional array (mathematical matrix) of the same fixed-size SQL data types. Currently, Matrix only supports storing doubles. Matrix dimensions must be non-zero.

Matrix functions and operators are described in the following sections:

- [Matrix Functions and Operators](#)
- [Matrix Functions and Operators](#)

## Geospatial Data Types

Ocient supports three different geospatial geographies: POLYGONS, LINESTRINGS, and POINTS.

- A POLYGON can be constructed with a closed LINESTRING or an outer shell and array of inner rings.
- A LINESTRING represents a series of POINTS connected by line segments. It can be constructed with either LINESTRINGS or POINTS.
- A POINT represents a point in space defined by an (x, y)/(long, lat) coordinate pair.

Polygon semantics in the Ocient System follow the OGC standard (ISO 19125-1) for LinearRings, where the exterior and holes can be meaningfully oriented either counter-clockwise or clockwise. To conceptualize this, imagine walking along the exterior of the polygon in the order of its defined points, so everything to your left is the “interior” of the polygon. A counter-clockwise polygon is the typical orientation where, using an example of a state boundary, the interior of the state corresponds to the interior of the polygon. If you define the state boundary using the clockwise orientation, the interior of the polygon represents everything except the state.

If a counter-clockwise exterior polygon has holes, the holes should be clockwise-oriented.

Ensure that both the data and polygon literals are oriented in the way you intend when you use them in queries.

## Geography Equivalent

Many other geospatial implementations use an object-oriented approach to geospatial data types by implementing a GEOGRAPHY union type that might contain any combination of these types: POINT, LINESTRING, or POLYGON.

Ocient supports GEOGRAPHY slightly differently by using POLYGON as an implicit union type. This container type means unclosed POLYGON types with no holes are equivalent to LINESTRING types, and that a POLYGON might contain a single POINT. This definition of a POLYGON differs from other implementations, where such POLYGON types would be considered degenerate. During loading, Ocient is able to convert incoming GEOGRAPHY data to the appropriate POLYGON equivalent. Ocient geospatial functions appropriately handle these cases as needed.

## Spatial Reference Identifier (SRID)

Similar to geography union types, many other geospatial implementations define an attribute table of column identifiers and an associated SRID. The system uses the SRID lookup to define semantics of functions such as `st_area`, which uses different calculations if the specified polygon is planar, defined on an ideal sphere, or on a spheroid. The identifiers also specify units of measure and the number of variables used to define a point in the reference system. However, Ocient only supports the storage of latitude and longitude coordinates for its geography types, and uses only GCS WGS 84 (EPSG code 4326) semantics. Rather than applying a transformation like you might in other implementations to change units or model accuracy, the geospatial measurement functions in the Ocient System allow you to specify the units of measure directly, as well as whether to use a faster idealized sphere model of the Earth, or a more accurate spheroidal model.

Ensure that you set the units and spheroidal model correctly when performing critical measurements. Also, be careful when you export data from other databases to Ocient so that the incoming data is in GCS WGS 84 format. Ensure that you use the Loading and Transformation pipeline to prepend or strip SRID definitions from EWKT or EWKB formatted data. Internally, the Ocient System has no concept of SRIDs, however it does support parsing and

emitting SRID-prepended data for compatibility with external tools. When the database parses SRID-prepended data, the system ignores the SRID component and parses the data as a GCS WGS 84 defined geography.

## Geospatial Functions

Each geospatial geography has a number of functions and operators that can be used in SQL queries to perform analyses. Some functions apply to specific geography types. For more function and operator specifics, see [Geospatial Functions](#).

## Related Links

[Date and Time Functions](#)

[Character and Binary Functions](#)

[Query Ocient](#)

[CREATE TABLE SQL Statement Examples](#)

## 10.7 Date and Time Functions

### Basic Date and Time Functions

#### ADD\_MONTHS

Adds the specified number of months to the date. Equivalent to using a date and adding a `MONTH(value)`.

#### Syntax

```
ADD_MONTHS(time, int)
```

| Argument    | DataType        | Description                                                                                                         |
|-------------|-----------------|---------------------------------------------------------------------------------------------------------------------|
| <i>time</i> | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function modifies by adding a specified number of months. |
| <i>int</i>  | INT             | The number of months to add to the <i>time</i> value.                                                               |

#### Example

```
SELECT ADD_MONTHS('2022-11-15 04:18:00',2);
```

*Output:* 2023-01-15

#### CENTURY

Returns the number of centuries.

#### Syntax

```
CENTURY(time)
```

| Argument    | DataType        | Description                                                                                                        |
|-------------|-----------------|--------------------------------------------------------------------------------------------------------------------|
| <i>time</i> | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function computes to return the number of its centuries. |

**Example**

In this example, the current date is 2023-01-03.

```
SELECT CENTURY(CURRENT_DATE());
```

*Output:* 21

**CURDATE**

Alias for CURRENT\_DATE.

**CURRENT\_DATE**

Alias for CURDATE.

Returns the current date in the format YYYY-MM-DD.

**Syntax**

```
CURRENT_DATE
```

**Example**

```
SELECT CURRENT_DATE;
```

*Output:* 2023-01-03

**CURRENT\_TIME**

Returns the current time as a TIME value (e.g., hh:mm:ss.mm).

**Syntax**

```
CURRENT_TIME
```

**Example**

```
SELECT CURRENT_TIME;
```

*Output:* 19:40:04

**CURRENT\_TIMESTAMP**

Alias for **Date / Time Functions**.

Returns the current date and time as a TIMESTAMP value (e.g., YYYY-MM-DD hh:mm:ss.mmm).

**Syntax**

```
CURRENT_TIMESTAMP
```

**Example**

```
SELECT CURRENT_TIMESTAMP;
```

*Output:* 2023-01-03 11:00:03.316674397

**DATE\_PART**

Alias for **Date / Time Functions**.

**DATE\_TRUNC**

Returns the date or timestamp entered, truncated to the specified precision.

**Syntax**

```
DATE_TRUNC(precision, time)
```

| Argument  | DataType          | Description                                                                                                                                                                                                                                                                                                                                                                     |
|-----------|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| precision | CHAR              | The timestamp unit used to truncate the returned value. For the <code>precision</code> input, acceptable values are: <code>NANOSECOND</code> , <code>MICROSECOND</code> , <code>MILLISECOND</code> , <code>SECOND</code> , <code>MINUTE</code> , <code>HOUR</code> , <code>DAY</code> , <code>WEEK</code> , <code>MONTH</code> , <code>QUARTER</code> , and <code>YEAR</code> . |
| time      | TIMESTAMP or DATE | A time value, specified as a timestamp or date, which the function truncates to the specified precision. The <code>time</code> input supports precision values <code>DAY</code> or larger.                                                                                                                                                                                      |

**Example**

This example uses `CURRENT_TIMESTAMP` set to 2023-01-03 11:00:59.114058182.

```
SELECT DATE_TRUNC('NANOSECOND', CURRENT_TIMESTAMP());
```

*Output:* 2023-01-03 11:01:05.280975906

**Example**

This example uses `CURRENT_TIMESTAMP` set to 2023-01-01 11:00:59.114058181.

```
SELECT DATE_TRUNC('MONTH', CURRENT_TIMESTAMP());
```

*Output:* 2023-01-01 00:00:00.000000000

**Example**

This example uses `CURRENT_TIMESTAMP` set to 2023-01-01 11:00:59.114058181.

```
SELECT DATE_TRUNC('CENTURY', CURRENT_TIMESTAMP());
```

*Output:* 2001-01-01 00:00:00.000000000

**DAY**

Alias for [Date / Time Functions](#).

**DAY\_OF\_WEEK**

Returns an integer, in the range of 1 to 7, that represents the day of the week. The value 1 is Sunday, and 7 is Saturday.

**Syntax**

```
DAY_OF_WEEK(time)
```



| Argument | DataType        | Description                                                                                          |
|----------|-----------------|------------------------------------------------------------------------------------------------------|
| time     | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function computes for its day of the week. |

**Example**

```
SELECT DAY_OF_WEEK(CURRENT_DATE());
```

*Output: 3*

**DAY\_OF\_YEAR**

Alias for DOY.

Returns an integer in the range 1 to 366 that represents the day of the year.

**Syntax**

```
DAY_OF_YEAR(time)
```

| Argument | DataType        | Description                                                                                      |
|----------|-----------------|--------------------------------------------------------------------------------------------------|
| time     | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function computes for its day of year. |

**Example**

This example uses CURRENT\_TIMESTAMP set to 2023-01-03 11:00:59.114058182.

```
SELECT DAY_OF_YEAR(CURRENT_TIMESTAMP());
```

*Output: 3*

**DAY\_OF\_MONTH**

Alias for DAY.

Extracts the day-of-month portion of a timestamp or date as an integer.

**Syntax**

```
DAY_OF_MONTH(time)
```

| Argument | DataType        | Description                                                                                           |
|----------|-----------------|-------------------------------------------------------------------------------------------------------|
| time     | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function computes for its day of the month. |

**Example**

This example uses CURRENT\_TIMESTAMP set to 2023-01-03 11:00:59.114058182.

```
SELECT DAY(CURRENT_TIMESTAMP());
```

Output: 3

## DECADE

The decade is the year divided by 10.

### Syntax

```
DECADE(time)
```

| Argument | Data Type       | Description                                                                                 |
|----------|-----------------|---------------------------------------------------------------------------------------------|
| time     | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function computes for its decade. |

### Example

This example uses CURRENT\_DATE set to 2023-01-03.

```
SELECT DECADE(CURRENT_DATE());
```

Output: 202

## DOW

Alias for [Date / Time Functions](#).

## DOY

Alias for [Date / Time Functions](#).

## EOMONTH

Returns the last day of the month using the specified timestamp or date. If you specify both arguments, this function returns the last day of the month for the resulting timestamp or date after the system adds the specified number of months to the first argument value.

### Syntax

```
EOMONTH(time [, int])
```

| Argument | Data Type       | Description                                                                                                                        |
|----------|-----------------|------------------------------------------------------------------------------------------------------------------------------------|
| time     | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function computes for the last day of the month.                         |
| int      | INT             | Optional. An integer value that represents the number of months to add to the time value. If unspecified, the value defaults to 0. |

### Examples

This example returns the last day of the month for the current timestamp.

```
SELECT EOMONTH(CURRENT_TIMESTAMP(),0);
```

Output: 2023-02-27

This example returns the last day of the month in November in 2022 by using the EOMONTH function to add one month to the last day of October in 2022.

```
SELECT EOMONTH(DATE('2022-10-31'), 1);
```

Output: 2022-11-30

## EPOCH

The number of seconds after 1970-01-01 00:00:00 UTC.

### Syntax

```
EPOCH(time)
```

| Argument | Data Type         | Description                                                                                                                          |
|----------|-------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| time     | TIMESTAMP or DATE | A time value, specified as a timestamp or date, which the function computes for the number of seconds after 1970-01-01 00:00:00 UTC. |

### Example

```
SELECT EPOCH(CURRENT_TIMESTAMP());
```

Output: 1672743901

## EXTRACT

Extract a component from a timestamp or date.

### Syntax

```
EXTRACT(precision FROM time)
```

| Argument  | Data Type         | Description                                                                                                                                                                      |
|-----------|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| time      | TIMESTAMP or DATE | A time value, specified as a timestamp or date, which the function computes for the component to be extracted.                                                                   |
| precision | CHAR              | The timestamp unit to be extracted. For the <code>precision</code> input, acceptable values are: <code>NANOSECOND</code> , <code>MICROSECOND</code> , <code>MILLISECOND</code> . |

### Example

This example uses `CURRENT_TIMESTAMP` set to 2023-01-03 11:00:59.114058182.

```
SELECT EXTRACT(HOUR FROM CURRENT_TIMESTAMP());
```

Output: 11.0

## HOUR

Extracts the hour portion of a timestamp as an integer.

### Syntax

```
HOUR(time)
```

| Argument | DataType        | Description                                                                                     |
|----------|-----------------|-------------------------------------------------------------------------------------------------|
| time     | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function computes for its hour value. |

### Example

This example uses CURRENT\_TIMESTAMP set to 2023-01-03 11:00:59.114058182.

```
SELECT HOUR(CURRENT_TIMESTAMP());
```

*Output:* 11

## ISODOW

Extracts the day of the week based on ISO 8601, which ranges from Monday (1) to Sunday (7).

### Syntax

```
ISODOW(timestamp or date)
```

| Argument | DataType        | Description                                                                                            |
|----------|-----------------|--------------------------------------------------------------------------------------------------------|
| time     | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function computes for its day-of-week value. |

### Example

This example uses CURRENT\_TIMESTAMP set to 2023-01-03 11:00:59.114058182.

```
SELECT ISODOW(CURRENT_TIMESTAMP());
```

*Output:* 2

## ISDATE

Returns TRUE if the input argument can be successfully cast to a date, FALSE otherwise.

### Syntax

```
ISDATE(char)
```

| Argument | DataType | Description                |
|----------|----------|----------------------------|
| char     | CHAR     | A string to cast as aDATE. |

### Example

```
SELECT ISDATE('2023-01-03');
```

*Output:* TRUE

## MAKEDATETIME

Returns a timestamp consisting of the specified date and time.

### Syntax

```
MAKEDATETIME(date, time)
```

| Argument | DataType | Description                           |
|----------|----------|---------------------------------------|
| date     | CHAR     | A date string for timestamp creation. |
| time     | CHAR     | A time string for timestamp creation. |

### Example

```
SELECT MAKEDATETIME('2022-11-02', '01:24:58');
```

Output: 2022-11-02 01:24:58.000000000

## MILLISECOND

Extracts the millisecond portion of a timestamp as an integer.

### Syntax

```
MILLISECOND(timestring)
```

| Argument   | DataType  | Description                                                                                            |
|------------|-----------|--------------------------------------------------------------------------------------------------------|
| timestring | TIMESTAMP | A time value, specified as a timestamp or date, which the function extracts for the millisecond value. |

### Example

This example uses CURRENT\_TIMESTAMP set to 2023-02-22 13:59:25.286.

```
SELECT MILLISECOND(CURRENT_TIMESTAMP());
```

Output: 286

## MINUTE

Extracts the minute portion of a timestamp or date as an integer.

### Syntax

```
MINUTE(time)
```

| Argument | DataType        | Description                                                                                       |
|----------|-----------------|---------------------------------------------------------------------------------------------------|
| time     | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function extracts for the minute value. |

### Example

This example uses CURRENT\_TIMESTAMP set to 2023-02-22 14:00:40.764.

```
SELECT MINUTE(CURRENT_TIMESTAMP());
```

Output: 0

## MONTH\_NAME

Returns the calendar name in English of the month for the specified date.

Syntax

```
MONTH_NAME(time)
```

| Argument | DataType        | Description                                                                                 |
|----------|-----------------|---------------------------------------------------------------------------------------------|
| time     | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function computes its month name. |

## Example

This example uses CURRENT\_TIMESTAMP set to 2023-02-22 13:59:25.286.

```
SELECT MONTH_NAME(CURRENT_TIMESTAMP());
```

Output: February

## MONTHS\_BETWEEN

Returns the difference between the two dates or timestamps in months as a DOUBLE. The fractional months component is based on a 31-day month. If the two dates have the same day or are both the last day of the month, this function returns a whole number instead.

Syntax

```
MONTHS_BETWEEN(time1, time2)
```

| Argument | DataType        | Description                                                                                                        |
|----------|-----------------|--------------------------------------------------------------------------------------------------------------------|
| time1    | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function compares to time2 for the difference in months. |
| time2    | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function compares to time1 for the difference in months. |

## Example

```
SELECT MONTHS_BETWEEN(CURRENT_TIMESTAMP(), ADD_MONTHS(CURRENT_TIMESTAMP(), 11));
```

Output: -11

## MONTH

Extracts the month portion of a timestamp or date as an integer.

Syntax

**MONTH**(time)

| Argument | DataType        | Description                                                                                      |
|----------|-----------------|--------------------------------------------------------------------------------------------------|
| time     | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function computes for its month value. |

**Example**

This example uses CURRENT\_TIMESTAMP set to 2023-02-22 13:59:25.286.

```
SELECT MONTH(CURRENT_TIMESTAMP());
```

Output: 2

**MSECS**

The seconds field, including fractional parts. This is the same as EXTRACT(MILLISECONDS FROM time).

**Syntax****MSECS**(time)

| Argument | DataType        | Description                                                                                                     |
|----------|-----------------|-----------------------------------------------------------------------------------------------------------------|
| time     | TIMESTAMPorDATE | A time value, specified as a timestamp or date. The function multiplies the seconds part of the value by 1,000. |

**Example**

```
SELECT MSECS(CURRENT_TIMESTAMP());
```

Output: 37637.648

**NANOS\_TO\_TIMESTAMP**

Convert a number of nanoseconds into a timestamp equivalent to the duration after the epoch time, 1970-01-01 00:00:00 UTC.

**Syntax****NANOS\_TO\_TIMESTAMP**(time)

| Argument | DataType | Description                                                                                                                                     |
|----------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| ns       | INT      | A number of nanoseconds, specified as an integer, which the function converts into a timestamp equivalent to the duration after the epoch time. |

**Example**

```
SELECT NANOS_TO_TIMESTAMP(10000000);
```

Output: 1969-12-31 16:00:00.010

**NEXT\_DAY**

Returns the closest date after a specified date that lies on a specific day of the week.

**Syntax**

```
NEXT_DAY(date, character)
```

| Argument  | DataType                | Description                                                                                                                                                                                                                               |
|-----------|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| date      | DATE,TIMESTAMP, or CHAR | A time value, specified as a date, timestamp, or string, which the function uses to return the closest date for a specific day of the week. Can be any type that can be cast to DATE.                                                     |
| character | CHAR                    | A string representing a day of the week. The string must match the first three characters (case insensitive) of the English name of any day of the week. If this string does not match the prefix of any day, this function returns NULL. |

**Example**

```
SELECT NEXT_DAY('2023-02-27', 'sat');
```

*Output:* 2023-03-04

**NOW**

Alias for [Date / Time Functions](#).

**QUARTER**

Returns an integer between 1 and 4 representing the quarter of the year in which the specified date falls.

**Syntax**

```
QUARTER(time)
```

| Argument | DataType          | Description                                                                                                    |
|----------|-------------------|----------------------------------------------------------------------------------------------------------------|
| time     | TIMESTAMP or DATE | A time value, specified as a timestamp or date, which the function computes to return the quarter of the year. |

**Example**

```
SELECT QUARTER(DATE('2023-02-08 09:00:00.000'));
```

*Output:* 1

**ROUND**

Returns the specified date or timestamp, rounded to the specified precision. The precision argument behaves like [Date / Time Functions](#). The function rounds up values to the specified precision.

**Syntax**



```
ROUND(time, precision)
```

| Argument  | Data Type         | Description                                                                                                                                                                                       |
|-----------|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| precision | CHAR              | The timestamp unit used to round the returned value. For the <code>precision</code> input, acceptable values are: <code>NANOSECOND</code> , <code>MICROSECOND</code> , <code>MILLISECOND</code> . |
| time      | TIMESTAMP or DATE | A time value, specified as a timestamp or date, which the function rounds to the specified precision.                                                                                             |

### Example

```
SELECT ROUND(CURRENT_TIMESTAMP(), 'HOUR');
```

Output: 2023-02-08 09:00:00.000

### SECOND

Extracts the seconds portion of a timestamp as an integer.

#### Syntax

```
SECOND(time)
```

| Argument | Data Type         | Description                                                                                          |
|----------|-------------------|------------------------------------------------------------------------------------------------------|
| time     | TIMESTAMP or TIME | A time value, specified as a timestamp or date, which the function uses to return the seconds value. |

### Example

This example uses `CURRENT_TIMESTAMP` set to 2023-02-28 13:17:30.243.

```
SELECT SECOND(CURRENT_TIMESTAMP());
```

Output: 30

### TIMESTAMP\_TO\_NANOS

Convert timestamp into nanoseconds after epoch as BIGINT.

#### Syntax

```
TIMESTAMP_TO_NANOS(time)
```

| Argument | Data Type         | Description                                                                                                        |
|----------|-------------------|--------------------------------------------------------------------------------------------------------------------|
| time     | TIMESTAMP or DATE | A time value, specified as a timestamp or date, which the function uses to return the nanoseconds after the epoch. |

**Example**

This example uses CURRENT\_TIMESTAMP set to 2023-02-28 13:20:12.373.

```
SELECT TIMESTAMP_TO_NANOS(CURRENT_TIMESTAMP());
```

Output: 1677619212373708904

**TO\_TIMESTAMP**

For information on using the TO\_TIMESTAMP conversion function, see the [Formatting Functions](#) page.

**USECS**

Returns the seconds part of a time value, including fractional parts, as an integer. Same as EXTRACT(MICROSECONDS FROM time).

**Syntax**

```
USECS(time)
```

| Argument | Data Type       | Description                                                                                                         |
|----------|-----------------|---------------------------------------------------------------------------------------------------------------------|
| time     | TIMESTAMPorDATE | A time value, specified as a timestamp or date. The function multiplies the seconds part of the value by 1,000,000. |

**Example**

This example uses CURRENT\_TIMESTAMP set to 2023-02-28 13:32:17.438.

```
SELECT USECS(CURRENT_TIMESTAMP());
```

Output: 17438041

**WEEK**

Returns the ISO-8601 week number, as an integer, of the specified timestamp or date value. The week starts on Monday and the first week of a year contains January 4 of that year.

For details, see the ISO week date definition.

**Syntax**

```
WEEK(time)
```

| Argument | Data Type       | Description                                                                                        |
|----------|-----------------|----------------------------------------------------------------------------------------------------|
| time     | TIMESTAMPorDATE | A time value, specified as a timestamp or date, which the function uses to return the week number. |

**Example**

This example uses CURRENT\_TIMESTAMP set to 2023-02-28 13:37:37.128.

```
SELECT WEEK(CURRENT_TIMESTAMP());
```

Output: 9

## YEAR

Extracts the year portion of a timestamp or date as an integer.

### Syntax

```
YEAR(time)
```

| Argument | Data Type         | Description                                                                                  |
|----------|-------------------|----------------------------------------------------------------------------------------------|
| time     | TIMESTAMP or DATE | A time value, specified as a timestamp or date, which the function uses to extract the year. |

### Example

This example uses CURRENT\_TIMESTAMP set to 2023-02-28 13:37:37.128.

```
SELECT YEAR(CURRENT_TIMESTAMP());
```

*Output:* 2023

## Advanced Date and Time Functions

### DATEADD

This function adds a specified number value (as a signed integer) to a specified **datepart** of an input date value, and then returns that modified value.

The data type of the returned value for this function is dynamic. The return type depends on the argument supplied for **date**. If the value for **date** is a string literal **date**, DATEADD returns a **datetime** value. If another valid input data type is supplied for **date**, DATEADD returns the same data type. DATEADD raises an error if the string literal seconds scale exceeds three decimal place positions (.nnn) or if the string literal contains the time zone offset part.

### Syntax

```
DATEADD (datepart, number, date)
```

| Argument | Data Type             | Description                                                                                                                                                                                                                                                                                 |
|----------|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| datepart | CHAR                  | The part of date to which DATEADD adds an integer number. See the Datepart Values table for a list of supported arguments.                                                                                                                                                                  |
| number   | INT                   | An expression that can resolve to an INT that DATEADD adds to the specified datepart value of date. DATEADD accepts user-defined variable values for number. DATEADD truncates a specified number value that has a decimal fraction. In this situation, it does not round the number value. |
| date     | TIME, TIMESTAMP, DATE | A column expression, expression, string literal, or user-defined variable. A string literal value must resolve to a date or timestamp. Use four-digit years to avoid ambiguity issues.                                                                                                      |

DATEADD does not accept user-defined variable equivalents for the `datepart`\*`*`arguments.

## Examples

This example shows each result from incrementing different `datepart` values by 1.

```
SELECT 'year', DATEADD(year,1, '2007-01-01 13:10:10.1111111')
      UNION ALL
SELECT 'quarter', DATEADD(quarter,1, '2007-01-01 13:10:10.1111111')
      UNION ALL
SELECT 'month', DATEADD(month,1, '2007-01-01 13:10:10.1111111')
      UNION ALL
SELECT 'dayofyear', DATEADD(dayofyear,1, '2007-01-01 13:10:10.1111111')
      UNION ALL
SELECT 'day', DATEADD(day,1, '2007-01-01 13:10:10.1111111')
      UNION ALL
SELECT 'week', DATEADD(week,1, '2007-01-01 13:10:10.1111111')
      UNION ALL
SELECT 'weekday', DATEADD(weekday,1, '2007-01-01 13:10:10.1111111')
      UNION ALL
SELECT 'hour', DATEADD(hour,1, '2007-01-01 13:10:10.1111111')
      UNION ALL
SELECT 'minute', DATEADD(minute,1, '2007-01-01 13:10:10.1111111')
      UNION ALL
SELECT 'second', DATEADD(second,1, '2007-01-01 13:10:10.1111111')
      UNION ALL
SELECT 'millisecond', DATEADD(millisecond,1, '2007-01-01 13:10:10.1111111')
      UNION ALL
SELECT 'microsecond', DATEADD(microsecond,1, '2007-01-01 13:10:10.1111111')
      UNION ALL
SELECT 'nanosecond', DATEADD(nanosecond,1, '2007-01-01 13:10:10.1111111');
```

## Output

|             |                               |
|-------------|-------------------------------|
| week        | 2007-01-08 13:10:10.111111100 |
| millisecond | 2007-01-01 13:10:10.112111100 |
| second      | 2007-01-01 13:10:11.111111100 |
| day         | 2007-01-02 13:10:10.111111100 |
| minute      | 2007-01-01 13:11:10.111111100 |
| dayofyear   | 2007-01-02 13:10:10.111111100 |
| quarter     | 2007-04-01 13:10:10.111111100 |
| month       | 2007-02-01 13:10:10.111111100 |
| hour        | 2007-01-01 14:10:10.111111100 |
| weekday     | 2007-01-02 13:10:10.111111100 |
| year        | 2008-01-01 13:10:10.111111100 |
| nanosecond  | 2007-01-01 13:10:10.111111101 |
| microsecond | 2007-01-01 13:10:10.111112100 |

Fetches 13 rows

In this example, each statement increments `datepart` by a number large enough to increment the next higher unit of date or time.

```
SELECT 'quarter', DATEADD(quarter,4, '2007-01-01 01:01:01.1111111')
      UNION ALL
SELECT 'month', DATEADD(month,13, '2007-01-01 01:01:01.1111111')
      UNION ALL
```

```

SELECT 'dayofyear', DATEADD(dayofyear,365,'2007-01-01 01:01:01.1111111')
      UNION ALL
SELECT 'day', DATEADD(day,365,'2007-01-01 01:01:01.1111111')
      UNION ALL
SELECT 'week', DATEADD(week,5,'2007-01-01 01:01:01.1111111')
      UNION ALL
SELECT 'weekday', DATEADD(weekday,31,'2007-01-01 01:01:01.1111111')
      UNION ALL
SELECT 'hour', DATEADD(hour,23,'2007-01-01 01:01:01.1111111')
      UNION ALL
SELECT 'minute', DATEADD(minute,59,'2007-01-01 01:01:01.1111111')
      UNION ALL
SELECT 'second', DATEADD(second,59,'2007-01-01 01:01:01.1111111');

```

#### Output

|           |                               |
|-----------|-------------------------------|
| day       | 2008-01-01 01:01:01.111111100 |
| weekday   | 2007-02-01 01:01:01.111111100 |
| second    | 2007-01-01 01:02:00.111111100 |
| month     | 2008-02-01 01:01:01.111111100 |
| quarter   | 2008-01-01 01:01:01.111111100 |
| dayofyear | 2008-01-01 01:01:01.111111100 |
| week      | 2007-02-05 01:01:01.111111100 |
| hour      | 2007-01-02 00:01:01.111111100 |
| minute    | 2007-01-01 02:00:01.111111100 |

Fetches 9 rows

## DATEDIFF

This function returns an INT representing the difference between two date or time values, in a specified date or time unit.

If only a time value is assigned to a date data type variable, **DATEDIFF** sets the value of the missing **datepart** to the default value: 1900-01-01. If only a date value is assigned to a variable of a time or date data type, **DATEDIFF** sets the value of the missing time part to the default value: 00:00:00. If either **startdate** or **enddate** have only a time part and the other only a **datepart**, **DATEDIFF** sets the missing time and **datepart** to the default values.

If **startdate** and **enddate** have different date data types, and one has more time parts or fractional seconds precision than the other, **DATEDIFF** sets the missing parts of the other to 0.

### Syntax

```
DATEDIFF ( datepart, startdate, enddate)
```

| Argument  | DataType            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| datepart  | CHAR                | The units in which the function returns the difference between the <code>startdate</code> and <code>enddate</code> . Commonly used <code>datepart</code> units include <code>month</code> or <code>second</code> . The <code>datepart</code> value cannot be specified in a variable, or in a quoted string like <code>'month'</code> . If <code>datepart</code> has a DATE value, and <code>startdate</code> and <code>enddate</code> are both assigned a TIME value, then the function returns 0. See the Datepart Values table for a list of supported * arguments. |
| startdate | TIME,TIMESTAMP,DATE | A starting date or time value to determine the difference from the <code>enddate</code> value. If only a time value is assigned, <code>DATEDIFF</code> sets the value of the missing <code>datepart</code> to the default value: 1900-01-01                                                                                                                                                                                                                                                                                                                            |
| enddate   | TIME,TIMESTAMP,DATE | An ending date or time value to determine the difference from the <code>startdate</code> value. If only a time value is assigned, <code>DATEDIFF</code> sets the value of the missing <code>datepart</code> to the default value: 1900-01-01                                                                                                                                                                                                                                                                                                                           |

For `millisecond`, the maximum difference between `startdate` and `enddate` is 24 days, 20 hours, 31 minutes, and 23.647 seconds.

For `second`, the maximum difference is 68 years, 19 days, 3 hours, 14 minutes, and 7 seconds.

Each specific `datepart` name and abbreviations for that `datepart` name return the same value.

### Example

The example shows how `DATEDIFF` calculates the difference between the two values based on the various `datepart` values.

```
SELECT 'year', DATEDIFF(year, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:00.0000000')
      UNION ALL
SELECT 'quarter', DATEDIFF(quarter, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:00.0000000')
      UNION ALL
SELECT 'month', DATEDIFF(month, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:00.0000000')
      UNION ALL
SELECT 'dayofyear', DATEDIFF(dayofyear, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:00.0000000')
      UNION ALL
SELECT 'day', DATEDIFF(day, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:00.0000000')
      UNION ALL
SELECT 'week', DATEDIFF(week, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:00.0000000')
      UNION ALL
SELECT 'hour', DATEDIFF(hour, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:00.0000000')
      UNION ALL
SELECT 'minute', DATEDIFF(minute, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:00.0000000')
```

**UNION ALL**

```
SELECT 'second', DATEDIFF(second, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:00.0000000');
```

*Output*

```
day          580
quarter      7
year         2
dayofyear    580
month        20
hour         13897
week         83
minute       833761
second       50025601
```

**Related Links**
[Time Zone Functions](#)
[Formatting Functions](#)

## 10.8 Formatting Functions

**Formatting Considerations**

- The format string can be any combination of characters and format patterns for each function specified in the [Formatting Functions](#) and [Formatting Functions](#) tables.
- Patterns can be uppercase or lowercase, but not mixed case. YYYY or yyyy matches YYYY, but not YyYy.
- Any characters in the format string that are not part of a pattern match any character in that position in TO\_DATE, TO\_NUMBER, and TO\_TIMESTAMP.
- You can escape string literals that contain format patterns with quotes. For example: YYYY"a literal part YYYY"

**Date and Time Formatting Functions****Date and Timestamp Formatting Patterns**

| Pattern                   | Description                                                                                                                  |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------|
| Y,YYY                     | 4 digit year with comma                                                                                                      |
| YYYY                      | 4 digit year                                                                                                                 |
| YYY                       | Last 3 digits of a year. Values between 000-599 are assumed to be the year 2XXX and 600-999 are assumed to be the year 1XXX. |
| YY                        | Last 2 digits of a year. Values between 00-69 are assumed to be the year 20XX and 70-99 are assumed to be 19XX.              |
| Y                         | Last digit of a year. Assumed to be year 200X.                                                                               |
| BC, bc, AD, or ad         | An era indicator BC or AD in upper or lower case                                                                             |
| B.C., b.c., A.D., or a.d. | An era indicator B.C. or A.D. in upper or lower case                                                                         |
| CC                        | Century value 1-99. Is ignored unless YY, Y, IY, or I are set                                                                |
| MM                        | Month of the year value 01-12                                                                                                |
| MONTH                     | Upper case English month name                                                                                                |
| Month                     | English Month name with first letter capitalized                                                                             |
| month                     | lower case English month name                                                                                                |
| RM                        | Upper case month of year in Roman numerals                                                                                   |
| rm                        | Lower case month of year in Roman numerals                                                                                   |

| Pattern                   | Description                                                                                                                                          |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| MON                       | 3 letter upper case abbreviation of an English month name                                                                                            |
| Mon                       | 3 letter abbreviation of an English month name with first letter capitalized                                                                         |
| mon                       | 3 letter lower case abbreviation of an English month name                                                                                            |
| W                         | Week of month 1-5. For example: Days 15-21 are in the third week of the month.                                                                       |
| DD                        | Day of month 1-31                                                                                                                                    |
| WW                        | Week of year 1-53                                                                                                                                    |
| DDD                       | Day of year 1-366                                                                                                                                    |
| D                         | Day of week 1-7 where Sunday is 1. This value is ignored outside of to_char.                                                                         |
| IYYY                      | 4 digit ISO-8601 week-numbered year. See for more details.                                                                                           |
| IYY                       | Last 3 digits of an ISO-8601 week-numbered year. Values between 000-599 are assumed to be the year 2XXX and 600-999 are assumed to be the year 1XXX. |
| IY                        | Last 2 digits of an ISO-8601 week-numbered year. Values between 00-69 are assumed to be the year 20XX and 70-99 are assumed to be 19XX.              |
| I                         | Last digit of an ISO-8601 week-numbered year. Assumed to be the year 200X.                                                                           |
| IDDD                      | Day of an ISO-8601 week-numbered year 001-371                                                                                                        |
| IW                        | Week of an ISO-8601 week-numbered year 01-53                                                                                                         |
| ID                        | Day of an ISO-8601 week 01-07 where Monday is 1                                                                                                      |
| HH or HH12                | Hour of day 01-12                                                                                                                                    |
| HH24                      | Hour of day 0-23                                                                                                                                     |
| AM, am, PM, or pm         | Meridiem indicator AM or PM in upper or lower case                                                                                                   |
| A.M., a.m., P.M., or p.m. | Meridiem indicator A.M. or P.M. in upper or lower case                                                                                               |
| SSSS                      | Seconds after midnight 0-86399                                                                                                                       |
| MI                        | Minute of hour 0-59                                                                                                                                  |
| SS                        | Second of minute 0-59                                                                                                                                |
| MS                        | Milliseconds 000-999                                                                                                                                 |
| US                        | Microseconds 000000-999999                                                                                                                           |
| NS                        | Nanoseconds 000000000-999999999                                                                                                                      |
| J                         | Julian Day (days after November 24, 4714 BC at midnight)                                                                                             |

The following modifiers can also be applied to a format pattern.

#### Date and Timestamp Formatting Modifiers

| Modifier | Description                                                                                                              |
|----------|--------------------------------------------------------------------------------------------------------------------------|
| FM       | Has no effect on TO_DATE and TO_TIMESTAMP. Will not be considered a literal, but will not otherwise affect the behavior. |
| TH       | Upper case ordinal number suffix. For example: 'DDTH'                                                                    |
| th       | Lower case ordinal number suffix. For example: 'DDth'                                                                    |



| Modifier | Description                                                                                                                                                                                                                           |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FX       | Global prefix. By default, TO_DATE and TO_TIMESTAMP collapse all spaces to a single space. If FX is specified, spaces are not collapsed. FX must be at the beginning of the format string to be applied. For example: 'FX YYYY-MM-DD' |
| TM       | Has no effect on TO_DATE and TO_TIMESTAMP. Will not be considered a literal, but will not otherwise affect the behavior.                                                                                                              |
| SP       | Has no effect on TO_DATE and TO_TIMESTAMP. Will not be considered a literal, but will not otherwise affect the behavior.                                                                                                              |

### Date and Time notes

- You can use time specifiers such as HH12 in TO\_DATE. Their format will be validated, but their values will not affect the resulting date.
- ISO 8601 formats for dates cannot be mixed with traditional formats for years, months, and days in TO\_DATE and TO\_TIMESTAMP
- Values for MS, US, and NS are scaled up if they do not have leading zeros. TO\_TIMESTAMP('0.3', 'S.NS') corresponds to 300000000 NS and TO\_TIMESTAMP('0.000000003', 'S.NS') corresponds to 3 NS.
- Conflicting information for the same pattern will cause an exception. ex: TO\_DATE('05 31', 'DD DD')
- Conflicting information across different patterns such as DD and DDD will generally give precedence to the pattern that appears higher in the table.

### TO\_TIMESTAMP

Converts a character value with the specified format to a TIMESTAMP type.

#### Syntax

```
TO_TIMESTAMP(character_value, character_format)
```

| Arguments        | DataType | Description                                                                                                                                            |
|------------------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| character_value  | CHAR     | A string of a value to be converted to a TIMESTAMP type. The string must match the provided character_format.                                          |
| character_format | CHAR     | The format pattern to be used to convert the character_value. See the Formatting Functions section for more information about accepted pattern values. |

### Examples

```
SELECT TO_TIMESTAMP('2022/12/01', 'YYYY/MM/DD');
```

Output: 2022-12-01 00:00:00.000

```
SELECT TO_TIMESTAMP('2023-02-28 13:43:20.403', 'YYYY/MM/DD HH24:MI:SS');
```

Output: 2023-02-28 13:43:20.000

**TO\_DATE**

Converts a character value with the specified format to a DATE type.

**Syntax**

```
TO_DATE(character_value, character_format)
```

| Arguments        | Data Type | Description                                                                                                                                            |
|------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| character_value  | CHAR      | A string of a value to be converted to a DATE type. The string must match the provided character_format.                                               |
| character_format | CHAR      | The format pattern to be used to convert the character_value. See the Formatting Functions section for more information about accepted pattern values. |

**Example**

```
SELECT TO_DATE('2022/12/01', 'YYYY/MM/DD');
```

Output: 2022-12-01

**Number Formatting Functions****Number Formatting Patterns**

| Pattern | Description                                                                                                                                              |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9       | A digit.                                                                                                                                                 |
| 0       | A digit. In TO_CHAR, includes leading zeros.                                                                                                             |
| . or D  | A decimal point.                                                                                                                                         |
| , or G  | A group/thousand separator.                                                                                                                              |
| PR      | Negative value in braces. For example: -1 = <1>                                                                                                          |
| S or SG | A + or - sign for positive or negative, respectively.                                                                                                    |
| MI      | A - sign if the number is negative.                                                                                                                      |
| PL      | A + sign if the number is positive.                                                                                                                      |
| V       | Shifts the decimal place to the by the number of digits after V in the format. Cannot be mixed with . or D. For example: TO_NUMBER('32', '9V99') => 0.32 |
| RN      | Roman numeral. Not supported by TO_NUMBER.                                                                                                               |
| EEEE    | Exponent for scientific notation. Not supported by TO_NUMBER.                                                                                            |
| L       | Currency symbol.                                                                                                                                         |

**Number Formatting Notes**

- You cannot mix PR with other sign indicators.
- Any character that does not match the corresponding group in the format string is ignored. TO\_NUMBER('1&2', '999') = 12.
- If PR is present, it must be after every 0 or 9 in the format string.
- Similar to TO\_DATE and TO\_TIMESTAMP, the FM is not considered a literal, but will not otherwise affect the behavior of TO\_NUMBER.

## TO\_NUMBER

Converts a character value with the specified format to a `DECIMAL` type.

### Syntax

```
TO_NUMBER(character_value, character_format)
```

| Arguments                     | DataType          | Description                                                                                                                                                                          |
|-------------------------------|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>character_value</code>  | <code>CHAR</code> | A string of a value to be converted to a <code>DECIMAL</code> type. The string must match the provided <code>character_format</code> .                                               |
| <code>character_format</code> | <code>CHAR</code> | The format pattern to be used to convert the <code>character_value</code> . See the <a href="#">Formatting Functions</a> section for more information about accepted pattern values. |

### Example

```
SELECT TO_NUMBER('5230.87', '9999V99');
```

Output: 5230.87

### Related Links

[Date and Time Functions](#)

[Time Zone Functions](#)

[Character and Binary Functions](#)

## 10.9 HyperLogLog Functions

The HyperLogLog (HLL) sketch family of functions provide an approximate count of the number of unique elements in one or more columns.

HLL functionality is similar to running a query using a `COUNT(DISTINCT col)` clause and uses the same mechanisms as the [Aggregate Functions](#) `APPROX_COUNT_DISTINCT`.

The tradeoff for HLL sketches is accuracy. For further explanation of this tradeoff in accuracy, see the [Apache® DataSketches](#) documentation. Each sketch is an approximate and compact representation of the original data, and it introduces a small margin of error. The [Ocient®](#) HLL implementation is based on the algorithm outlined in the research paper [HyperLogLog in Practice](#). This implementation uses `log2k = 11` by default, which provides a percent error of approximately 5% at the 95% confidence interval.

### HLL Sketch Functions in the SQL Statement

Sketches are useful on aggregated tables that require estimates of distinct counts.

This example creates a table for the new sketch columns `ip_address_sketch` and `user_id_sketch` as part of a `CREATE TABLE AS SELECT` statement. The new table inserts values from the `master_agg_table` table.

```
CREATE TABLE my_sketch_table (
  col_a INT,
  col_b VARCHAR(255),
  ip_address_sketch HLL_SKETCH(11),
  user_id_sketch HLL_SKETCH(11)
)
```

```

AS (
  SELECT
    col_a,
    col_b,
    HLL_SKETCH_CREATE(ip_address) as ip_address_sketch,
    HLL_SKETCH_CREATE(user_id) as user_id_sketch
FROM
  master_agg_table
  GROUP by 1,2);

```

After the database creates the new table, you can query for the approximate count of distinct values.

This example merges the sketch columns by using an `HLL_SKETCH_UNION` function. After the database merges the columns, the database converts sketch values to a distinct count estimate by using the `HLL_SKETCH_GET_ESTIMATE` function.

```

SELECT
  HLL_SKETCH_GET_ESTIMATE(HLL_SKETCH_UNION(ip_address_sketch)) AS ip_address_sketch,
  HLL_SKETCH_GET_ESTIMATE(HLL_SKETCH_UNION(user_id_sketch)) AS user_id_sketch
FROM
  my_sketch_table;

```

### HLL Accuracy with log2k Parameter

In addition to the default `log2k = 11` implementation, you can specify a `log2k` value in the range of `[10, 16]` when creating an HLL Sketch.

This parameter controls the number of buckets used in the HLL algorithm, which controls the accuracy of the sketch upper bounded by  $1.04 / \sqrt{k}$ . The tradeoff with larger `log2k` values is that the sketch size grows exponentially. See this table for specifics on how the precision values affect size and accuracy.

| Precision    | Uncompressed Size | 95% CI |
|--------------|-------------------|--------|
| 10           | 1032 B            | ±6.50% |
| 11 (default) | 2056 B            | ±4.60% |
| 12           | 4104 B            | ±3.25% |
| 13           | 8200 B            | ±2.30% |
| 14           | 16392 B           | ±1.63% |
| 15           | 32776 B           | ±1.15% |
| 16           | 65544 B           | ±0.81% |

By default, `log2k` values larger than 11 are compressed on disk using ZSTD compression.

When creating or referencing a HLL Sketch column in a `CREATE TABLE`, `CTAS` or `IAS` statement, you can use the type alias `HLL_SKETCH(log2k)` instead of the internal `HASH((2^log2k) + 8)`. This type alias is supported in `EXPORT` statements as well.

### Example

In this example, a `CREATE TABLE` statement creates two HLL Sketch columns with specified `log2k` precision values.

- `sketch_a` has a precision value of 10, meaning it has low precision and low storage requirements.
- `sketch_b` has a precision value of 15, meaning it has high precision and high storage requirements.

Note that the precision values specified in both `HLL_SKETCH_CREATE` statements match the precision values in the `CREATE TABLE` columns.

```
CREATE TABLE my_sketch_table (
  sketch_a HLL_SKETCH(10),
  sketch_b HLL_SKETCH(15)
)
AS (
  SELECT
    HLL_SKETCH_CREATE(col_a, 10) as sketch_a,
    HLL_SKETCH_CREATE(col_b, 15) as sketch_b
FROM
  master_agg_table
GROUP BY 1,2);
```

## Supported HLL Sketch Functions

### HLL\_SKETCH\_CREATE

Creates an HLL sketch from the data on a specified aggregated column. Returns a  $\text{HASH}((2^{\log 2k}) + 8)$  data representation of the sketch that you can store in a separate column.

The HLL algorithm depends on the hash value of its input. When merging multiple sketches, it is important that each sketch derives from the same type in order to get accurate results. `1::BIGINT` might hash differently than `1::INT`, and thus, the HLL algorithm might not view these two hashes as referring to the same value 1. For accurate results, users should cast numeric columns or expressions to a consistent type before they are used in a `HLL_SKETCH_CREATE` function, especially if they will be merged later on.

### Syntax

```
HLL_SKETCH_CREATE(agg_col, [ log2k ] )
```

| Arguments | DataType                      | Description                                                                                                                                                                                                                                                                                                                                                                                |
|-----------|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| agg_col   | All data types are supported. | An aggregated column for use when you create a sketch.                                                                                                                                                                                                                                                                                                                                     |
| log2k     | Integral literal              | Optional. This parameter controls the number of buckets in the HLL Sketch, which affects the accuracy and storage of the sketch. This value must be an integer literal in the range of [10, 16]. It cannot be a reference to a column. By default, log2k values larger than 11 are compressed on disk using ZSTD compression. For details about this parameter, see HyperLogLog Functions. |

### Example

This example uses `HLL_SKETCH_CREATE` as a window aggregate to generate 100 sketches that contain one value in each sketch.

```
SELECT MOD(c1, 9) AS m,
       HLL_SKETCH_CREATE(c1, 12) OVER (PARTITION BY c1) AS sketch
FROM sys.dummy100;
```

**HLL\_SKETCH\_UNION (aggregate function)**

Merges multiple sketches in a single column into a unified sketch. All sketches must have the same precision. This function is an aggregate function and operates on a column.

**Syntax**

```
HLL_SKETCH_UNION(agg_sketch_col)
```

| Arguments      | DataType          | Description                                                            |
|----------------|-------------------|------------------------------------------------------------------------|
| agg_sketch_col | HLL_SKETCH<log2k> | A column that contains multiple sketches to be merged into one sketch. |

**Example**

This example performs two merges. First, this code uses HLL\_SKETCH\_CREATE as a window aggregate to generate 100 sketches that contain one value in each sketch. The code merges the 100 sketches into nine sketches, and then the final SELECT statement merges the nine sketches into a single sketch. Also, the example uses the HLL\_SKETCH\_GET\_ESTIMATE function to retrieve the estimated distinct count from the merged sketches.

```
WITH hll_sketch AS (
  SELECT MOD(c1, 9) AS m,
         HLL_SKETCH_CREATE(c1) OVER (PARTITION BY c1) AS sketch
  FROM sys.dummy100
),
hll_merged AS (
  SELECT HLL_SKETCH_UNION(hll_sketch.sketch) AS sketch
  FROM hll_sketch
  GROUP BY m
)
SELECT HLL_SKETCH_GET_ESTIMATE(HLL_SKETCH_UNION(hll_merged.sketch))
FROM hll_merged;
```

*Output: 97*

**HLL\_SKETCH\_UNION (scalar function)**

Merges two sketches to a new combined sketch. This function is a scalar function and operates row-wise. The scalar function merges two sketch columns with heterogeneous precisions into a sketch with the lower of the two precisions.

**Syntax**

```
HLL_SKETCH_UNION(sketch_col1, sketch_col2)
```

| Arguments   | DataType          | Description              |
|-------------|-------------------|--------------------------|
| sketch_col1 | HLL_SKETCH<log2k> | A column of HLL sketches |
| sketch_col2 | HLL_SKETCH<log2k> | A column of HLL sketches |

**Example**

This example creates two sketches from the dummy table. The first uses the default precision of 11, and the second uses a precision of 14. The second sketch is offset by 1000, so there are 2000 unique values being sketched overall.

The query merges the two sketches using the two argument scalar union, which combines them into a new sketch. This unified sketch has a precision of 11, the lower of the two input precisions.

The query then decodes the sketch, which has the expected result of  $2000 \pm 92$ .

```

SELECT HLL_SKETCH_GET_ESTIMATE(
    HLL_SKETCH_UNION(
        HLL_SKETCH_CREATE(c1),
        HLL_SKETCH_CREATE((c1 + 1000)::INT, 14)
    )
) AS sketch_col
FROM sys.dummy1000;

```

Output: 2000 ± 92

## HLL\_SKETCH\_GET\_ESTIMATE

The `HLL_SKETCH_GET_ESTIMATE` scalar function converts a sketch into a distinct count estimate of a sketch value. Returns the distinct count estimate as a `BIGINT`.

To return a distinct count across a column of sketch values, you should first merge the sketches by using the `HLL_SKETCH_UNION` function.

### Syntax

```
HLL_SKETCH_GET_ESTIMATE(sketch)
```

| Arguments | DataType          | Description                                                 |
|-----------|-------------------|-------------------------------------------------------------|
| sketch    | HLL_SKETCH<log2k> | A sketch value to convert to an approximate distinct count. |

### Example

```

WITH hll_sketch AS (
    SELECT MOD(c1, 9) AS m,
           HLL_SKETCH_CREATE(c1) OVER (PARTITION BY c1) AS sketch
    FROM sys.dummy100
),
hll_merged AS (
    SELECT HLL_SKETCH_UNION(hll_sketch.sketch) AS sketch
    FROM hll_sketch
    GROUP BY m
)
SELECT HLL_SKETCH_GET_ESTIMATE(HLL_SKETCH_UNION(hll_merged.sketch))
FROM hll_merged;

```

Output: 97

## HLL\_SKETCH\_GET\_ESTIMATE\_BOUND

The `HLL_SKETCH_GET_ESTIMATE_BOUND` scalar function takes a `HLL_SKETCH` column or an integral `log2k` literal value and returns the resulting bounding 95-percent confidence interval error proportion as a `DOUBLE`. If you provide an integral `log2k` value, it must be an integral literal and cannot be a column.

### Syntax

```
HLL_SKETCH_GET_ESTIMATE_BOUND(sketch_or_integral)
```

| Arguments          | DataType                              | Description                                     |
|--------------------|---------------------------------------|-------------------------------------------------|
| sketch_or_integral | HLL_SKETCH<log2k> or integral literal | A sketch value, column, or an integral literal. |

**Example**

```
SELECT HLL_SKETCH_GET_ESTIMATE_BOUND(HLL_SKETCH_CREATE(c1, 14))
FROM sys.dummy100;
```

Output: 0.0163

**HLL\_SKETCH\_TO\_STRING**

The HLL\_SKETCH\_TO\_STRING scalar function takes a HLL\_SKETCH column or value and returns a string summary of the sketch.

**Syntax**

```
HLL_SKETCH_TO_STRING(sketch)
```

| Arguments | Data Type         | Description                                        |
|-----------|-------------------|----------------------------------------------------|
| sketch    | HLL_SKETCH<log2k> | A sketch value or column to summarize as a string. |

**Example**

```
SELECT HLL_SKETCH_TO_STRING(HLL_SKETCH_CREATE(c1)) AS summary
FROM sys.dummy10000;
```

Output:

```
### HLL SKETCH SUMMARY:
Log2k           : 11
Lower bound     : 9540.000000
Estimate        : 10000
Upper bound     : 10460.000000
Number of values seen : 10000
Minimum index   : 114
Value at minimum index : 0
Maximum index   : 1892
Value at maximum index : 14
```

This summary contains this information:

- The log2k value. For more information, see [HyperLogLog Functions](#).
- The lower, estimate, and upper bound of the sketch.
- The number of values seen in the sketch.
- The minimum and maximum bucket indexes and their values in the sketch.

The output fields in this summary are for internal purposes, except for the log2k value.

**Bibliography**

Heule, Stefan, Marc Nunkesser, and Alex Hall. “HyperLogLog in Practice: Algorithmic Engineering of a State of The Art Cardinality Estimation Algorithm.” In *Proceedings of the EDBT 2013 Conference*. Genoa, Italy, 2013.

**Related Links**

[Aggregate Functions](#)

[Generate Tables Using sys.dummy](#)



## 10.10 Geospatial Functions

The Ocint Hyperscale Data Warehouse™ has OcintGeo™ that supports three different geospatial geographies: POLYGON, LINESTRING, and POINT.

- A POLYGON can be constructed with a closed LINESTRING or an outer shell and array of inner rings.
- A LINESTRING represents a series of points connected by line segments. It can be constructed with either LINESTRING or POINT data types.
- A POINT represents a point in space defined by an (x, y) or (longitude, latitude) coordinate pair.

Each LINESTRING or POLYGON value can be up to a maximum of 512 MB in size. This means a LINESTRING or POLYGON can contain approximately 32 million point values.

These supported geospatial data types require Well-Known Text (WKT) formatting. For formatting examples, see [Data Types](#) or WKT Representation of Geometry. The OcintGeo functionality reference also contains examples with supported formatting and syntax.

Each geospatial geography has a number of functions and operators that can be used in SQL queries to perform analyses. Some functions apply to specific geography types. For operators specific to geospatial data types, refer to [Geospatial Functions](#).

OcintGeo uses the authalic radius of the Earth, 6371007.1810824 meters, as the reference for many measurement calculations.

The authalic radius reference might cause slight differences in measurements compared to other geospatial systems that use a different radius definition.

### Geospatial Operators

OcintGeo supports special operators that perform operations on two different geographies. These can be used in queries to compare geographies or calculate values based on the geographies.

| Operator              | Syntax                                   | Purpose                                                                                    |
|-----------------------|------------------------------------------|--------------------------------------------------------------------------------------------|
| intersects            | <code>geoA &amp;&amp; geoB</code>        | Returns TRUE if the bounding box of geography A intersects with the bounding box of B.     |
| contained             | <code>geoA @ geoB</code>                 | Returns TRUE if the bounding box of geography A is contained within the bounding box of B. |
| equal                 | <code>geoA ~= geoB</code>                | Returns TRUE if the bounding box of geography A is the same as bounding box of B.          |
| contains              | <code>geoA ~ geoB</code>                 | Returns TRUE if the bounding box of geography A contains the bounding box of B.            |
| distance              | <code>geoA &lt;-&gt; geoB</code>         | Returns the distance in meters between the two geography parameters.                       |
| bounding box distance | <code>geoA &lt;#&gt; geoB</code>         | Returns the distance in meters between the bounding box of the two geography parameters.   |
| centroid distance     | <code>geoA &lt;&lt;-&gt;&gt; geoB</code> | Returns the distance in meters between the centroids of the two geography parameters.      |

## Geospatial Functions Overview

OcientGeo supports many geospatial functions that operate on POINT, LINESTRING, and POLYGON values. Each function, its expected inputs, and return type are described in the following section.

## Geospatial Filtering

Queries that involve geospatial filtering can make use of functions in the [Spatial Relationships](#) section.

To optimize these queries, OcientGeo provides a **SPATIAL** index that you can apply to columns with geospatial data types. For details about the **SPATIAL** index, see [Secondary Indexes](#).

## Attribute Functions

Attribute functions return descriptive information on the specified data set.

- ST\_COORDDIM
- ST\_DIMENSION
- ST\_GEOMETRYTYPE
- ST\_ISEMPTY
- ST\_MEMSIZE
- ST\_NDIMS
- ST\_NDIMENSION
- ST\_NPOINTS
- ST\_NUMPOINTS
- ST\_SRID
- ST\_X
- ST\_XMAX
- ST\_XMIN
- ST\_Y
- ST\_YMAX
- ST\_YMIN

## Conversion Functions

Conversion functions transform a specified operand to a different data type.

- ST\_ASBINARY
- ST\_ASWKB
- ST\_ASJSON
- ST\_ASEWKT
- ST\_ASLATLONTEXT
- ST\_ASTEXT
- ST\_ASWKT
- ST\_GEOHASH

## Linestring Constructors

LINESTRING constructors use geospatial data to create a LINESTRING object.

- ST\_LINEFROMTEXT
- ST\_LINEFROMJSON
- ST\_LINEFROMWKB
- ST\_LINEFROMEWKT
- ST\_LINESTRING
- ST\_MAKELINE

### Linestring Functions

LINESTRING functions can perform alterations or access descriptive information on LINESTRING objects.

- ST\_ADDPOINT
- ST\_ENDPOINT
- ST\_LINEINTERPOLATEPOINT
- ST\_LINELOCATEPOINT
- ST\_LINESUBSTRING
- ST\_POINTN
- ST\_REMOVEPOINT
- ST\_SETPPOINT
- ST\_STARTPOINT

### Point Constructors

POINT constructors use geospatial data to create a POINT object.

- ST\_CENTROID
- ST\_GEOPOINT
- ST\_MAKEPOINT
- ST\_POINT
- ST\_POINTFROMEWKT
- ST\_POINTFROMGEOHASH
- ST\_POINTFROMGEOJSON
- ST\_POINTFROMTEXT
- ST\_POINTFROMWKB

### Polygon Constructors

POLYGON constructors use geospatial data to create a POLYGON object.

- ST\_FORCECCW
- ST\_MAKEPOLYGON
- ST\_POLYGON
- ST\_POLYGONFROMGEOJSON
- ST\_POLYGONFROMWKB
- ST\_POLYGONFROMEWKT
- ST\_POLYGONFROMTEXT
- ST\_WHOLEEARTH

### Spatial Measurement

Spatial measurement functions can perform basic calculations on geospatial data, such as measuring the distance between two POINT objects or the area of a POLYGON object.

- ST\_AREA
- ST\_ANGLE
- ST\_AZIMUTH
- ST\_DISTANCE
- ST\_DISTANCESPHERE
- ST\_DISTANCESPHEROID
- ST\_EUCLIDEANDISTANCE3D
- ST\_HAUSDORFFDISTANCE
- ST\_LENGTH
- ST\_LENGTH2D
- ST\_MINIMUMDISTANCETOSURFACE
- ST\_MAXDISTANCE
- ST\_PERIMETER

- ST\_PERIMETER2D

### Spatial Operators

Spatial operators perform geometry calculations on geospatial data to return a different type of geospatial data.

- ST\_BOUNDINGDIAGONAL
- ST\_BUFFER
- ST\_CLOSESTPOINT
- ST\_CONVEXHULL
- ST\_DIFFERENCEARRAY
- ST\_ENVELOPE
- ST\_EXPAND
- ST\_EXTERIORRING
- ST\_FLIPCOORDINATES
- ST\_FORCE2D
- ST\_INTERIORRINGN
- ST\_INTERSECTALL
- ST\_INTERSECTIONARRAY
- ST\_LONGESTLINE
- ST\_MAKEENVELOPE
- ST\_MINIMUMBOUNDINGCIRCLE
- ST\_MULTIDIFFERENCEARRAY
- ST\_MULTIINTERSECTIONARRAY
- ST\_MULTISYMDIFFERENCEARRAY
- ST\_MULTIUNIONARRAY
- ST\_NRINGS
- ST\_NUMINTERIORRING
- ST\_NUMINTERIORRINGS
- ST\_POINTONSURFACE
- ST\_PROJECT
- ST\_REDUCEPRECISION
- ST\_REMOVEREPEATEDPOINTS
- ST\_REVERSE
- ST\_SEGMENTIZE
- ST\_SHORTESTLINE
- ST\_SIMPLIFY
- ST\_SIMPLIFYARRAY
- ST\_SNAPTOGRID
- ST\_SYMDIFFERENCEARRAY
- ST\_UNIONARRAY

### Spatial Relationships

Spatial relationship functions use arguments to test for different types of spatial relationships.

- ST\_CLUSTERDBSCAN
- ST\_CONTAINS
- ST\_CONTAINSPROPERLY
- ST\_COVERS
- ST\_COVEREDBY
- ST\_CROSSES
- ST\_DISJOINT
- ST\_DWITHIN
- ST\_EQUALS
- ST\_INTERSECTS
- ST\_ISCCW

- ST\_ISPOLYGONCCW
- ST\_ISPOLYGONCW
- ST\_ISCLOSED
- ST\_ISRING
- ST\_ISSIMPLE
- ST\_ISVALID
- ST\_OVERLAPS
- ST\_POINTINSIDECIRCLE
- ST\_RELATE
- ST\_TOUCHES
- ST\_WITHIN

### Spatiotemporal Measurement

Spatiotemporal measurement functions can perform basic calculations on geospatial data paired with `TIMESTAMP` data.

- ST\_DISTANCE
- ST\_MAXDISTANCE
- ST\_TOTALSECONDSININTERSECTION

### Spatiotemporal Operators

Spatiotemporal operators perform calculations on geospatial data using an array of `TIMESTAMP` arguments.

- ST\_LONGESTLINE
- ST\_LINEGETALLTIMESATPOINT
- ST\_LINEGETPOINTATTIME
- ST\_LINEGETTIMEATPOINT
- ST\_INTERSECTION
- ST\_SHORTESTLINE

### Related Videos

At the Whiteboard with Ocient: Geospatial Analytics

#### 10.10.1 Attribute Functions

OcientGeo™ attribute functions return descriptive information on the specified data set.

#### ST\_COORDDIM

Alias for `ST_NDIMS` or `ST_NDIMENSION`.

Returns an `INTEGER` of the coordinate dimension of the specified geography.

#### Syntax

```
ST_COORDDIM(geo)
```

| Argument | DataType                    | Description                                             |
|----------|-----------------------------|---------------------------------------------------------|
| geo      | POINT,LINestring, orPOLYGON | The geospatial object used to calculate the dimensions. |

#### Example

```
SELECT ST_COORDDIM(ST_POLYGON('POLYGON((1 1, 2 1, 2 2, 1 1))'));
```

*Output: 2*

## ST\_DIMENSION

Returns an **INTEGER** that represents the dimension of the specified geography. This table describes the result values of the **ST\_DIMENSION** function.

| <b>ST_DIMENSION Result Value</b> | <b>Dimension Type</b> |
|----------------------------------|-----------------------|
| -1                               | Empty geography value |
| 0                                | POINT                 |
| 1                                | LINESTRING            |
| 2                                | POLYGON               |

The **ST\_DIMENSION** function always returns 0 for non-empty **POINT** objects.

For non-empty **LINESTRING** and **POLYGON** objects, the **ST\_DIMENSION** function returns the dimension value equal to the greatest dimension represented by their bounding points. For example, a **LINESTRING** or **POLYGON** object that contains only a single point returns 0. Similarly, a **POLYGON** object can return 1 if it contains insufficient points to represent a closed polygon.

### Syntax

```
ST_DIMENSION(geo)
```

| <b>Argument</b> | <b>Data Type</b>              | <b>Description</b>                                      |
|-----------------|-------------------------------|---------------------------------------------------------|
| geo             | POINT, LINESTRING, or POLYGON | The geospatial object used to calculate the dimensions. |

### Examples

In this example, the function evaluates a **LINESTRING** with an empty **POINT** value.

```
SELECT ST_DIMENSION(ST_LINESTRING('POINT EMPTY'));
```

*Output:* -1

In this example, the function evaluates a **LINESTRING** with a single **POINT** value.

```
SELECT ST_DIMENSION(ST_LINESTRING('ST_POINT(0 0)'));
```

*Output:* 0

This example evaluates an empty **LINESTRING**.

```
SELECT ST_DIMENSION(ST_LINESTRING('LINESTRING EMPTY'));
```

*Output:* -1

In this example, the function again evaluates a **LINESTRING** with a single **POINT** value.

```
SELECT ST_DIMENSION(ST_LINESTRING('LINESTRING(0 0)'));
```

*Output:* 0

In this example, the function evaluates a **LINESTRING** with multiple **POINT** values.

```
SELECT ST_DIMENSION(ST_LINESTRING('LINESTRING(0 0, 1 1)'));
```

*Output:* 1

This example evaluates an empty **POLYGON**.

```
SELECT ST_DIMENSION(ST_POLYGON('POLYGON EMPTY'));
```

Output: -1

This example evaluates a POLYGON with a single POINT.

```
SELECT ST_DIMENSION(ST_POLYGON('POLYGON((1 1))'));
```

Output: 0

This example evaluates a POLYGON with two POINT values.

```
SELECT ST_DIMENSION(ST_POLYGON('POLYGON((0 0, 1 1))'));
```

Output: 1

This example evaluates a POLYGON with multiple POINT values, enough to close the POLYGON.

```
SELECT ST_DIMENSION(ST_POLYGON('POLYGON((0 0, 1 0, 1 1, 0 0))'));
```

Output: 2

## ST\_GEOMETRYTYPE

Returns a STRING that represents the geometry of the input geospatial value. Supported values are ST\_POINT, ST\_LINESTRING, and ST\_POLYGON. This function enforces the same strict dimension types as the [Attribute Functions](#) function. If geo is NULL, then the function returns NULL.

### Syntax

```
ST_GEOMETRYTYPE(geo)
```

| Argument | DataType                 | Description                                                |
|----------|--------------------------|------------------------------------------------------------|
| geo      | POINT,LINestring,POLYGON | A geospatial value to be identified by its dimension type. |

### Example

```
SELECT ST_GEOMETRYTYPE(ST_POINT(1, 2));
```

Output: ST\_POINT

## ST\_ISEMPTY

Returns TRUE if the specified geography value is empty, such as 'POLYGON EMPTY'. A NULL input value returns NULL.

### Syntax

```
ST_ISEMPTY(geo)
```

| Argument | DataType                 | Description                                       |
|----------|--------------------------|---------------------------------------------------|
| geo      | POINT,LINestring,POLYGON | A geospatial value to be examined if it is empty. |

### Example

```
SELECT ST_ISEMPTY(ST_POLYGON('POLYGON EMPTY'));
```

Output: TRUE

**ST\_MEMSIZE**

Returns an INTEGER representing the number of bytes in memory required to store the specified geography.

**Syntax**

```
ST_MEMSIZE(geo)
```

| Argument | DataType                 | Description                                           |
|----------|--------------------------|-------------------------------------------------------|
| geo      | POINT,LINestring,POLYGON | A geospatial value to be measured by its data memory. |

**Example**

```
SELECT ST_MEMSIZE(ST_POINT(1,1));
```

*Output:* 16

**ST\_NDIMS or ST\_NDIMENSION**

Alias for [Attribute Functions](#).

**ST\_NPOINTS or ST\_NUMPOINTS**

Returns an INTEGER representing the number of POINT values in a specified geography.

If the specified value is a POLYGON, this function counts the number of POINT values in both the exterior and any holes.

**Syntax**

```
ST_NPOINTS(geo)
```

```
ST_NUMPOINTS(geo)
```

| Argument | DataType                 | Description                                                      |
|----------|--------------------------|------------------------------------------------------------------|
| geo      | POINT,LINestring,POLYGON | A geospatial value to be examined for its count of POINT values. |

**Example**

```
SELECT ST_NPOINTS(
  ST_POLYGON(
    'POLYGON((1 2, 1 3, 1 2))');
```

*Output:* 3

**ST\_SRID**

Returns the EPSG code of the spatial reference identifier (SRID) of the input geography. All Ocient® geographies are of type GCS WGS 84, which is identified by the value 4326. There is no way to set a different SRID.

**Syntax**

```
ST_SRID(geo)
```



| Argument | DataType                 | Description                                          |
|----------|--------------------------|------------------------------------------------------|
| geo      | POINT,LINestring,POLYGON | A geospatial value to be analyzed for its EPSG code. |

**Example**

```
SELECT ST_SRID(ST_MAKEPOINT(1.1, 3.11));
```

*Output:* 4326

**ST\_X**

Returns the x value, or longitude, of the specified POINT. The returned value is a DOUBLE type.

**Syntax**

```
ST_X(point)
```

| Argument | DataType | Description                                                       |
|----------|----------|-------------------------------------------------------------------|
| point    | POINT    | A geospatial point to be analyzed for its x value, the longitude. |

**Example**

```
SELECT ST_X(ST_POINT(3, 5));
```

*Output:* 3

**ST\_XMAX**

Returns the maximum x value of the specified geography. The returned value is a DOUBLE type.

**Syntax**

```
ST_XMAX(geo)
```

| Argument | DataType                 | Description                                                           |
|----------|--------------------------|-----------------------------------------------------------------------|
| geo      | POINT,LINestring,POLYGON | One or more geospatial points to be analyzed for the maximum x value. |

**Example**

In this example, the ST\_XMAX function returns 2 because it is the largest x value of the specified points.

```
SELECT ST_XMAX(
    ST_POLYGON(ST_LINestring('LINestring(2 5, 1 2, 1 4, 0 3)')));
```

*Output:* 2

**ST\_XMIN**

Returns the minimum x value of the specified geography.

**Syntax**

```
ST_XMIN(geo)
```

| Argument | DataType                 | Description                                                           |
|----------|--------------------------|-----------------------------------------------------------------------|
| geo      | POINT,LINestring,POLYGON | One or more geospatial points to be analyzed for the minimum x value. |

### Example

In this example, the ST\_XMIN function returns 0 because it is the smallest x value of the specified points.

```
SELECT ST_XMIN(
  ST_POLYGON(ST_LINestring('LINestring(2 5, 1 2, 1 4, 0 3)')));
```

*Output:* 0

### ST\_Y

Returns the y value, or latitude, of the specified POINT. The returned value is a DOUBLE type.

#### Syntax

```
ST_Y(point)
```

| Argument | DataType | Description                                                      |
|----------|----------|------------------------------------------------------------------|
| point    | POINT    | A geospatial point to be analyzed for its y value, the latitude. |

### Example

```
SELECT ST_Y(ST_POINT(3, 5));
```

*Output:* 5

### ST\_YMAX

Returns the maximum y value of specified geography.

#### Syntax

```
ST_YMAX(geo)
```

| Argument | DataType                 | Description                                                           |
|----------|--------------------------|-----------------------------------------------------------------------|
| geo      | POINT,LINestring,POLYGON | One or more geospatial points to be analyzed for the maximum y value. |

### Example

In this example, the ST\_YMAX function returns 5 because it is the largest y value of the specified points.

```
SELECT ST_YMAX(
  ST_POLYGON(ST_LINestring('LINestring(2 5, 1 2, 1 4, 0 3)')));
```

*Output:* 5

**ST\_YMIN**

Returns the minimum y value of specified geography.

**Syntax**

```
ST_YMIN(geo)
```

| Argument | Data Type                | Description                                                           |
|----------|--------------------------|-----------------------------------------------------------------------|
| geo      | POINT,LINestring,POLYGON | One or more geospatial points to be analyzed for the minimum y value. |

**Example**

In this example, the ST\_YMIN function returns 2 because it is the smallest y value of the specified points.

```
SELECT ST_YMIN(
    ST_POLYGON(ST_LINestring('LINestring(2 5, 1 2, 1 4, 0 3)')));
```

*Output:* 2

**Related Links**

[Data Types](#)

[Linestring Constructors](#)

[Point Constructors](#)

[Polygon Constructors](#)

**10.10.2 Conversion Functions**

OcientGeo™ conversion functions transform a specified operand to a different data type.

**ST\_ASBINARY**

Alias of ST\_ASWKB.

Returns the well-known binary (WKB) representation of the specified geography.

**Syntax**

```
ST_ASBINARY(geo [, XDR_or_NDR ] )
```

| Argument   | Data Type                   | Description                                                                                                                                                                                              |
|------------|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo        | POINT,LINestring, orPOLYGON | The geospatial object used to calculate the dimensions.                                                                                                                                                  |
| XDR_or_NDR | CHAR                        | Optional. This argument determines whether the resulting WKB is in little endian (NDR) or big endian (XDR) format.If you specify this argument, it must be eitherXDRorNDR. Otherwise, it defaults toNDR. |

**Example**

```
SELECT ST_ASBINARY(ST_POINT(2, 2), 'NDR');
```

*Output:* 010100000000000000000000400000000000000040

**ST\_ASGEOJSON**

Returns the GeoJSON representation of the specified geography using the IETF standards.

```
ST_ASGEOJSON( geo [, planar_conversion ] )
```

| Argument          | DataType                    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo               | POINT,LINestring, orPOLYGON | The geospatial object used to create a GeoJSON representation.≤ The GeoJSON specification has certain requirements for aLINestringandPOLYGON.In keeping with this specification, aPOLYGONinstance that represents aLINestringresults in aLINestringGeoJSON instance.Similarly, aPOLYGONinstance that represents aPOINTresults in aPOINTGeoJSON instance.ALINestringinstance that represents aPOINTresults in aPOINTGeoJSON instance.Unlike the GeoJSON specification, a specifiedPOLYGONdoes not need to follow the Right-Hand Rule. If a specifiedPOLYGONcontains any holes that are not validPOLYGONobjects, then the function omits those holes in the resulting GeoJSON instance.                                                                                                                                                                       |
| planar_conversion | BOOLEAN                     | Optional. A Boolean value that determines if the geodesic edges of the specified geography are converted to planar representation before the function applies the GeoJSON format. IETF standards state that all GeoJSON objects are to be interpreted as explicitly planar. This argument allows preservation of the original geodesic shape.Ifplanar_conversionis specifiedTRUE, points will be added such that the resulting GeoJSON represents a planar shape within 10m of the original geospatial object. Additionally, if the original geospatial object crossed the international dateline, it will be split at the point where it crosses. This will generate a GeoJSON object of eitherMultiLineStringorMultiPolygontype, depending on the original type of the geospatial object.If you do not specify this argument, the value defaults toFALSE. |

## Examples

This example converts a POINT value into a GeoJSON representation.

```
SELECT ST_ASGEOJSON(ST_POINT(2, 2), false);
```

Output:

```
{"type": "Point", "coordinates": [2, 2]}
```

This example converts an empty POINT value.

```
SELECT ST_ASGEOJSON(ST_POINTFROMTEXT('POINT EMPTY'), false);
```

Output:

```
{"type": "Point", "coordinates": []}
```

This example converts a LINESTRING value into a GeoJSON representation.

```
SELECT ST_ASGEOJSON(ST_LINESTRING(ST_POINT(1.234, 2), ST_POINT(3.456, 5)));
```

Output:

```
{"type": "LineString", "coordinates": [[1.234, 2], [3.456, 5]]}
```

This example again converts a LINESTRING value.

```
SELECT ST_ASGEOJSON(ST_LINESTRING(ARRAY[ST_POINT(1.234, 2)]));
```

Output:

```
{"type": "Point", "coordinates": [1.234, 2]}
```

This example converts an empty LINESTRING value into a GeoJSON representation.

```
SELECT ST_ASGEOJSON(ST_LINEFROMTEXT('LINESTRING EMPTY'), false);
```

Output:

```
{"type": "LineString", "coordinates": []}
```

This example converts a POLYGON value into a GeoJSON representation.

```
SELECT ST_ASGEOJSON(
  ST_POLYGON(
    ARRAY [ST_POINT(1.234, 2), ST_POINT(3.456, 5), ST_POINT(5.678, 2), ST_POINT(1.234, 2)],
    ARRAY [ARRAY[ST_POINT(2, 2.5), ST_POINT(2.1, 2.6), ST_POINT(2.2, 2.5), ST_POINT(2, 2.5)],
    ARRAY [ST_POINT(3, 2.5), ST_POINT(3.1, 2.6), ST_POINT(3.2, 2.5), ST_POINT(3, 2.5)]]), false);
```

Output:

```
{"type": "Polygon", "coordinates": [[[1.234, 2], [3.456, 5], [5.678, 2], [1.234, 2]], [[2, 2.5], [2.1, 2.6], [2.2, 2.5], [2, 2.5]]]}
```

This example again converts a POLYGON value.

```
SELECT ST_ASGEOJSON(
  ST_POLYGON(
    ARRAY [ST_POINT(1.234, 2), ST_POINT(3.456, 5), ST_POINT(5.678, 2), ST_POINT(1.234, 2)],
    ARRAY [ARRAY[ST_POINT(2, 2.5), ST_POINT(2.1, 2.6), ST_POINT(2.2, 2.5), ST_POINT(3, 2.5)],
    ARRAY [ST_POINT(3, 2.5), ST_POINT(3.2, 2.5), ST_POINT(3, 2.5)]]), false);
```

Output:

```
{"type": "Polygon", "coordinates": [[[1.234, 2], [3.456, 5], [5.678, 2], [1.234, 2]]]}
```

This example converts a two-point POLYGON, which results in a LINESTRING type in the GeoJSON representation.

```
SELECT ST_ASGeoJSON(
  ST_POLYGON(
    ST_LineString(ST_Point(1.234, 2), ST_Point(3.456, 5)),
    false);
```

Output:

```
{"type": "LineString", "coordinates": [[1.234, 2], [3.456, 5]]}
```

This example converts a single-point POLYGON, which results in a POINT type in the GeoJSON representation.

```
SELECT ST_ASGeoJSON(ST_POLYGON(ST_Point(1.234, 2)), false);
```

Output:

```
{"type": "Point", "coordinates": [1.234, 2]}
```

This example converts an empty POLYGON value into a GeoJSON representation.

```
SELECT ST_ASGeoJSON(ST_PolygonFromText('POLYGON EMPTY'), false);
```

Output:

```
{"type": "Polygon", "coordinates": []}
```

## ST\_ASLATLONTEXT

Returns a string that represents geographic coordinates of a specified POINT in the specified format.

```
ST_ASLATLONTEXT(point, [ output_format ] )
```

| Argument | DataType | Description                                                                                                |
|----------|----------|------------------------------------------------------------------------------------------------------------|
| point    | POINT    | A POINT value used to calculate geographic coordinates. If point is NULL, then this function returns NULL. |

| Argument      | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| output_format | CHAR      | Optional. A string that includes option flags to set the format and decimal precision for the output of the coordinates. Option flags must be uppercase to be recognized. Accepted option flags are: <b>D</b> —Represents coordinate degrees. If you specify a format string for <b>output_format</b> , this value is required. If <b>M</b> is omitted, this option flag sets the decimal precision. <b>M</b> —Represents minutes of latitude or longitude distance. If <b>S</b> is omitted, this option flag sets the decimal precision. <b>S</b> —Represents seconds of latitude or longitude distance. <b>C</b> —Represents the cardinal direction. If you specify this value, the appropriate cardinal direction is represented as N/S/E/W. If you omit this value, coordinates that are south or west are represented as negative values while east or north remain positive values. For all option flags besides <b>C</b> , you can repeat the option characters and use them with decimal points to set the format and decimal precision. The number of characters after the decimal point indicates the precision, while the number of characters before the decimal indicates the total width of the formatted number, including the decimal point. If the formatted number is shorter than the requested width, the system pads the value with leading spaces. Characters other than <b>D</b> , <b>M</b> , <b>S</b> , <b>C</b> , and <b>.</b> are passed through as string characters. If you do not specify <b>output_format</b> or leave it as empty, the value defaults to the format <b>D°M' 'S.SSS"C</b> . |

### Example

```
SELECT ST_ASLATLONTEXT(ST_POINT(10.2342342,-2.32498), 'D.D degrees MM.MM minutes C');
```

Output: '2.0 degrees 19.50 minutes S 10.0 degrees 14.05 minutes E'

### ST\_ASTEXT

Alias of ST\_ASWKT and ST\_EWKT.

Returns the well-known text (WKT) representation of the specified geography.

If you use `ST_ASEWKT`, the database prepends the resultant string with `SRID=4326;`.

`ST_ASTEXT(geo)`

| Argument | DataType                    | Description                                                                                                                                                                                                                                                                                                 |
|----------|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo      | POINT,LINestring, orPOLYGON | A geospatial object that is to be returned as a WKT value.If you specify aLINestringorPOLYGONthat contains a singlePOINT, the function returns the WKT representation as aPOINT.If you specify aPOLYGONthat has an exterior that is not closed, the function returns the WKT representation as aLINestring. |

### Example

```
SELECT ST_ASTEXT(ST_POLYGON(ST_LINestring('LINestring(1 2, 1 3, 1 3, 1 2)')));
```

*Output:* POLYGON((1 2, 1 3, 1 3, 1 2))

### ST\_ASEWKT

For usage, see [Conversion Functions](#).

### ST\_ASWKT

Alias of [Conversion Functions](#).

### ST\_ASWKB

Alias of [Conversion Functions](#).

### ST\_GEOHASH

Returns a string that represents the geohash of the input POINT. This function also accepts an optional second argument to specify the length of the geohash result.

If you specify any argument as NULL, then the function returns NULL.

`ST_GEOHASH(point [, precision ] )`

| Argument  | DataType | Description                                                                                                                                                                                                                      |
|-----------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| point     | POINT    | APOINTvalue that is to be converted into a geohash.                                                                                                                                                                              |
| precision | INTEGER  | Optional. A numeric value that specifies the geohash precision, which determines the length of the output string.This number must be in the range of 1 to 20. If you do not specify this value, <b>precision</b> defaults to 20. |

### Example



```
SELECT ST_GEOHASH(ST_POINT(-126, 48));
```

Output: c0w3hc0w3hf1s70w3hf1s70w3

### Related Links

[Data Types](#)

[Attribute Functions](#)

### 10.10.3 Linestring Constructors

OcientGeo™ LINESTRING constructors use geospatial data to create a LINESTRING object.

#### ST\_LINEFROMTEXT

Alias for ST\_LINESTRING(CHAR).

Creates a LINESTRING from a specified CHAR. The CHAR must be a LINESTRING value in WKT format.

#### Syntax

```
ST_LINEFROMTEXT(char)
```

| Argument | DataType | Description                                                        |
|----------|----------|--------------------------------------------------------------------|
| char     | CHAR     | A character value in WKT format to be used to create a LINESTRING. |

#### Example

```
SELECT ST_LINEFROMTEXT('LINESTRING(1 2)');
```

Output: LINESTRING(1.000000 2.000000)

#### ST\_LINEFROMWKB

Alias for ST\_LINESTRING(BINARY).

Creates a LINESTRING from the specified BINARY. The BINARY value must be a LINESTRING in WKB format.

#### Syntax

```
ST_LINEFROMWKB(binary)
```

| Argument | DataType | Description                                                     |
|----------|----------|-----------------------------------------------------------------|
| binary   | BINARY   | A binary value in WKB format to be used to create a LINESTRING. |

#### Example

```
SELECT ST_LINEFROMWKB(
  BINARY(
    '0x0102000000020000000000000000f03f000000000000400000000000084000000000001040'
  )
);
```

Output: LINESTRING(1 2,3 4)

**ST\_LINEFROMEWKT**

Creates a **LINESTRING** from the specified **CHAR**.

The **CHAR** value must be a **LINESTRING** in EWKT format.

**ST\_LINEFROMEWKT** is functionally the same as **ST\_LINESTRING(CHAR)**.

The input string must include an **SRID=...; value**. However, the database ignores this component as all Ocint® geography types are SRID 4326.

**Syntax**

```
ST_LINEFROMEWKT('SRID=value;char')
```

| Argument | DataType | Description                                                                                                                                                |
|----------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| char     | CHAR     | ALINESTRINGvalue in WKT format to be used to create aLINESTRING.To function properly, you must prefix this argument with an SRID value in the same string. |
| value    | INTEGER  | Any integer value. This is required, however the database ignores this component and defaults to SRID 4326.                                                |

**Example**

```
SELECT ST_LINEFROMEWKT('SRID=4326;LINESTRING(1 2)');
```

*Output:* LINESTRING(1.000000 2.000000)

**ST\_LINEFROMGEOJSON**

Creates a **LINESTRING** represented by the specified GeoJSON. The specified GeoJSON can represent a GeoJSON **POINT** or **LINESTRING** type.

Valid GeoJSON formats follow IETF standards, and you can generate them by using [Conversion Functions](#). If you specify an invalid GeoJSON, the behavior of the function is undefined.

**Syntax**

```
ST_LINEFROMGEOJSON(geojson [, geodesic ])
```

| Argument | DataType | Description                                                                                                                            |
|----------|----------|----------------------------------------------------------------------------------------------------------------------------------------|
| geojson  | CHAR     | A GeoJSON value that represents aPOINTorLINESTRINGtype that is used to create aLINESTRING.Ifgeojsonis NULL, the function returns NULL. |

| Argument | DataType | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geodesic | BOOLEAN  | Optional. A <b>BOOLEAN</b> value that determines if the specified <b>geojson</b> should be converted from a planar to a geodesic representation. If you specify <b>geodesic</b> as <b>TRUE</b> , the function adds points to the resulting <b>LINESTRING</b> such that it remains within 10 meters of the original planar line. If unspecified, <b>geodesic</b> defaults to <b>FALSE</b> . If <b>geodesic</b> is <b>NULL</b> , the function returns <b>NULL</b> . |

### Examples

In this example, the function converts an empty GeoJSON value of a **LINESTRING**.

```
SELECT ST_LINEFROMGEOJSON('{"type":"LineString","coordinates":[]}',false);
```

Output: **LINESTRING EMPTY**

In this example, the function converts a GeoJSON value of a **LINESTRING**.

```
SELECT ST_LINEFROMGEOJSON(
  '{"type":"LineString","coordinates":[[1,1],[1,5],[5,5],[5,1]]}',
  false);
```

Output: **LINESTRING(1 1, 1 5, 5 5, 5 1)**

### ST\_LINESTRING

Alias for **ST\_MAKELINE**.

Creates a **LINESTRING** based on the specified inputs.

#### ST\_LINESTRING(geoArray)

Creates a **LINESTRING** from the specified array of either **LINESTRING** or **POINT** values.

#### Syntax

```
ST_LINESTRING(geo_array)
```

| Argument  | DataType                                                               | Description                                                                                             |
|-----------|------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|
| geo_array | ARRAY of geographies, either <b>LINESTRING</b> or <b>POINT</b> values. | An array of <b>LINESTRING</b> or <b>POINT</b> values to be used to create a unified <b>LINESTRING</b> . |

### Example

```
SELECT ST_LINESTRING(
  ST_LINESTRING [] (ST_LINESTRING(ST_POINT(1, 2), ST_POINT(3, 4))));
```

Output: **LINESTRING(1.000000 2.000000, 3.000000 4.000000)**

## ST\_LINESTRING(binary)

Creates a **LINESTRING** from the specified **BINARY**. The **BINARY** value must be a **LINESTRING** in WKB format.

## Syntax

ST\_LINestring(binary)

| Argument | Data Type | Description                                                                                           |
|----------|-----------|-------------------------------------------------------------------------------------------------------|
| binary   | BINARY    | ABINARY value to be used to create a LINESTRING. The BINARY value must be a LINESTRING in WKB format. |

### Example

```
SELECT ST_LINestring(BINARY('0x01020000002000000000000000f03f000000000000040000000000084000000000
```

*Output:* LINESTRING(1 2,3 4)

## ST\_LINestring(char)

Creates a **LINESTRING** from the specified **CHAR**. The **CHAR** value must be a **LINESTRING** in WKT format.

## Syntax

ST\_LINestring(char)

| Argument | Data Type | Description                                                                                 |
|----------|-----------|---------------------------------------------------------------------------------------------|
| char     | CHAR      | ACHARvalue to be used to create aLINESTRING. The argument must be aLINESTRINGin WKT format. |

### Example

```
SELECT ST_LINestring('LINestring(1 2)');
```

*Output:* LINESTRING(1.000000 2.000000)

**ST\_LINestring**(geo1, geo2)

Creates a **LINESTRING** consisting of two separate geographic arguments, both of which must be either a **POINT** or **LINESTRING**. You can mix geographic types, meaning one argument can be a **POINT** while the other is a **LINESTRING**.

## Syntax

ST\_LINestring(geo1, geo2)

| Argument  | Data Type         | Description                                                                                                              |
|-----------|-------------------|--------------------------------------------------------------------------------------------------------------------------|
| geo1,geo2 | POINTorLINESTRING | Two separate geographic arguments to be consolidated into a singleLINESTRING.Inputs can be the same type or one of each. |

### Example

```
SELECT ST_LINESTRING(ST_POINT(1, 2), ST_POINT(1,3));
```

*Output:* LINESTRING(1.000000 2.000000, 1.000000 3.000000)

### ST\_MAKELINE

Alias for [Linestring Constructors](#).

### ST\_MAKELINE(geo1, geo2)

Alias for [Linestring Constructors](#).

### ST\_MAKELINE(CHAR)

Alias for [Linestring Constructors](#).

### ST\_MAKELINE(geoArray)

Alias for [Linestring Constructors](#).

### Related Links

[Data Types](#)

[Linestring Functions](#)

[Conversion Functions](#)

#### 10.10.4 Linestring Functions

OcientGeo™ LINESTRING functions can perform alterations or access descriptive information on LINESTRING objects.

### ST\_ADDPOINT

Adds a POINT to the given LINESTRING at the specified 0-indexed location.

#### Syntax

```
ST_ADDPOINT(geo_linestring, geo_point_to_add [, location ] )
```

| Argument         | DataType   | Description                                                                                                                                                                                                                            |
|------------------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo_linestring   | LINESTRING | A linestring to be altered by adding the <code>geo_point_to_add</code> at a specified location. The function throws an error if <code>geo_linestring</code> is any data type except for <code>LINESTRING</code> or <code>NULL</code> . |
| geo_point_to_add | POINT      | A point value to be added to the <code>geo_linestring</code> .                                                                                                                                                                         |

| Argument | DataType | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| location | INTEGER  | Optional. Represents a numeric 0-indexed location in the <code>geo_linestring</code> where <code>geo_point_to_add</code> is appended. If you do not specify this value, the function defaults to appending the <code>geo_point_to_add</code> argument to the end of <code>geo_linestring</code> . A location value of -1 also appends to the end. The function throws an error if the <code>location</code> argument is outside the bounds of the <code>geo_linestring</code> or if it is a value of less than -1. |

**Example**

```
SELECT ST_ADDPOINT(
  ST_LINESTRING('LINESTRING(1 2, 1 3)'),
  ST_POINT(1, 4),
  1);
```

Output: LINESTRING(1.000000 2.000000, 1.000000 4.000000, 1.000000 3.000000)

**ST\_ENDPOINT**

Returns the endpoint of a specified LINESTRING. The returned value is a POINT.

**Syntax**

```
ST_ENDPOINT(geo)
```

| Argument | DataType   | Description                                                                                                                                                                                         |
|----------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo      | LINESTRING | A line to be computed to return its endpoint. If this value is a POLYGON, POINT, or is empty, then the function returns NULL. If this value is a non-geospatial type, the function throws an error. |

**Example**

```
SELECT ST_ENDPOINT(ST_LINESTRING('LINESTRING(1 2, 1 3, 1 4)'));
```

Output: POINT(1,4)

**ST\_LINEINTERPOLATEPOINT**

Returns a POINT along a LINESTRING based on a specified fraction of its total length.

**Syntax**

```
ST_LINEINTERPOLATEPOINT(line, fraction)
```

| Argument | DataType   | Description                                                                                                                                                                                                                 |
|----------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| line     | LINESTRING | A line used to compute an interpolated point.                                                                                                                                                                               |
| fraction | DOUBLE     | A decimal value between 0 and 1 that represents a fraction of the total length of the line. The function returns a point at this fraction of the length. The function throws an error if this value is not between 0 and 1. |

### Example

In this example, the function computes a point halfway (0.5) along the specified line.

```
SELECT ST_LINEINTERPOLATEPOINT(ST_LINESTRING('LINESTRING(1 2, 2 1)'), 0.5);
```

Output: POINT(1.5 1.5)

### ST\_LINELOCATEPOINT

Similar to [Linestring Functions](#), this function computes a fraction based on where a specified POINT is located along the length of a specified LINESTRING.

#### Syntax

```
ST_LINELOCATEPOINT(line, point)
```

| Argument | DataType   | Description                                                                                                                                                                    |
|----------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| line     | LINESTRING | A LINESTRING used to compute a fraction based off the interpolated point value.                                                                                                |
| point    | POINT      | A POINT used to calculate a fraction based on where it is located along the line value. If the function cannot interpolate this value from the line, the function returns 0.0. |

### Example

In this example, the function computes a fraction that represents where the point (1.5, 1.5) lies along the line.

```
SELECT ST_LINELOCATEPOINT(
  ST_LINESTRING('LINESTRING(1 2, 2 1)'),
  ST_POINT(1.5, 1.5));
```

Output: 0.5

### ST\_LINESUBSTRING

Returns a LINESTRING that is a substring of a specified line that starts and ends at the specified fractions of its total length.

#### Syntax

```
ST_LINESUBSTRING(line, start_fraction, end_fraction)
```

| Argument       | DataType   | Description                                                                                                                                                                                                                                                  |
|----------------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| line           | LINESTRING | ALINESTRINGused to compute a substring line based on thestart_fractionandend_fractionvalues.                                                                                                                                                                 |
| start_fraction | DOUBLE     | A decimal value between0and1that represents a fractional point of the total length of theline. The function uses this fractional point as the start point for the substring.The function throws an error if thestart_fractionis greater than theend_fraction |
| end_fraction   | DOUBLE     | A decimal value between0and1that represents a fractional point of the total length ofline. The function uses this fractional point as the endpoint for the substring.                                                                                        |

### Example

In this example, the function generates a subsection of the provided linestring, spanning from the **start\_fraction** value (0.5) that represents the halfway point of the original linestring, to the **end\_fraction** value (1.0) that represents the endpoint.

```
SELECT ST_LINESUBSTRING(ST_LINESTRING('LINESTRING(1 2, 2 1)'), 0.5, 1.0);
```

*Output:* LINESTRING(1.50 1.50, 2.0 1.0)

### ST\_POINTN

Returns the POINT value at a specified index of the specified LINESTRING. If out of bounds, the function returns NULL.

#### Syntax

```
ST_POINTN(geo, index)
```

| Argument | DataType   | Description                                                                                                                                                                                                                                                 |
|----------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo      | LINESTRING | ALINESTRINGthat is used to return one of its point values.If you specify aPOINTorPOLYGONvalue, this function returns NULL.                                                                                                                                  |
| index    | INTEGER    | An index location of one of the point values in thegeovalue.This is one-based, meaning the first index location starts at1.Negative values count backward from the end of the linestring. In other words, an index value of-1returns that last point value. |

### Example

```
SELECT ST_POINTN(
  ST_LINESTRING(
```



```
ST_POINT [] (ST_POINT(1, 2), ST_POINT(1, 3), ST_POINT(1, 4)),
3);
```

Output: ST\_POINT(1,4)

## ST\_REMOVEPOINT

Removes a POINT value at a specified index from the specified line.

### Syntax

```
ST_REMOVEPOINT(geo, index)
```

| Argument | DataType   | Description                                                                                                                                           |
|----------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo      | LINESTRING | ALINESTRINGthat is to be altered by removing one of its point values.                                                                                 |
| index    | INTEGER    | An index of one of the point values in thegeovalue.This is zero-based, meaning the first index location starts at0and the last index islength(geo)-1. |

### Example

```
SELECT ST_REMOVEPOINT(ST_LINESTRING('LINESTRING(1 2, 1 3, 1 4)'), 1);
```

Output: LINESTRING(1.0 2.0, 1.0 4.0)

## ST\_SETPPOINT

Replaces a POINT value in a specified LINESTRING at a specified index. The function returns the altered LINESTRING with the replaced point.

### Syntax

```
ST_SETPPOINT(geo, index, geo_point_to_replace)
```

| Argument             | DataType   | Description                                                                                                                                                                                                                                                   |
|----------------------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo                  | LINESTRING | ALINESTRINGthat is to be altered with a new point value.                                                                                                                                                                                                      |
| index                | INTEGER    | An index location of one of the point values in thegeolinestring.This is zero-based, meaning the first index location starts at0.Negative values count backward from the end of thegeovalue. In other words, an index value of-1returns the last point value. |
| geo_point_to_replace | POINT      | APOINTvalue to replace the value at the specifiedindexlocation.If this value is not aPOINTtype or NULL, the function throws an error.                                                                                                                         |

### Example

In this example, the function replaces the point value located at the -2 index location, meaning it is the second from the last in the sequence.

```
SELECT ST_SETPPOINT(
  ST_LINESTRING('LINESTRING(1 2, 1 3, 1 4)'),
  -2,
  ST_POINT(1, 5));
```

Output: LINESTRING(1.000000 2.000000, 1.000000 5.000000, 1.000000 4.000000)

## ST\_STARTPOINT

Returns the starting POINT value of the line. If this value is empty, the function returns NULL. If you specify a POLYGON, POINT, or NULL, then the function returns NULL. If you specify a non-geography value, the function throws an error.

### Syntax

```
ST_STARTPOINT(geo)
```

| Argument | DataType   | Description                                                                                                                                                                                           |
|----------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo      | LINESTRING | A line to be computed to return its starting point.If this value is aPOLYGON,POINT, or is empty, then the function returns NULL.If this value is a non-geospatial type, the function throws an error. |

### Example

```
SELECT ST_STARTPOINT(ST_LINESTRING('LINESTRING(1 2, 1 3, 1 4)'));
```

Output: ST\_POINT(1,2)

### Related Links

[Data Types](#)

[Linestring Constructors](#)

## 10.10.5 Point Constructors

OcientGeo™ POINT constructors use geospatial data to create a POINT object.

## ST\_CENTROID

Returns a POINT that is the geographic center of mass of a specified geospatial object.

The geographic center of mass is calculated by taking the average of all points on a three-dimensional sphere, projecting the resultant point onto the sphere, and converting it back to latitude and longitude coordinates.

### Syntax

```
ST_CENTROID( geo [, use_spheroid ] )
```

| Argument | DataType                    | Description                                                                     |
|----------|-----------------------------|---------------------------------------------------------------------------------|
| geo      | POINT,LINestring, orPOLYGON | A geospatial object that the function analyzes to compute its geometric center. |

| Argument     | DataType | Description                                                                                                                                                                            |
|--------------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| use_spheroid | BOOLEAN  | Optional.If this argument isTRUE, the function computes the centroid using a spheroid model instead of a spherical model.If you do not specify this value, the value defaults toFALSE. |

**Example**

```
SELECT ST_CENTROID(ST_POLYGON('POLYGON((1 1, 2 1, 2 2, 1 1))'));
```

Output: POINT(1.66 1.33)

**ST\_GEOGPOINT**

Alias for ST\_POLYGON(ST\_LINESTRING(ARRAY(ST\_POINT(lon, lat)))).

Creates and returns a POLYGON geography with a single point, defined by the longitude and latitude specified for the function.

**Syntax**

```
ST_GEOGPOINT(lon, lat)
```

| Argument | DataType | Description                                                              |
|----------|----------|--------------------------------------------------------------------------|
| lon      | DOUBLE   | A longitude coordinate to be used to create aPOLYGONwith a single point. |
| lat      | DOUBLE   | A latitude coordinate to be used to create aPOLYGONwith a single point.  |

**Example**

```
SELECT ST_GEOGPOINT(41.882548, -87.63922);
```

Output: POLYGON((41.882548 -87.63922))

**ST\_MAKEPOINT**

Alias for [Point Constructors](#).

**ST\_POINT**

Creates a POINT from the specified input arguments.

**ST\_POINT(binary)**

Alias for ST\_POINTFROMWKB.

Creates a POINT from the specified BINARY value, which must represent a point in WKB format.

**Syntax**

```
ST_POINT(binary)
```

| Argument | DataType | Description                                           |
|----------|----------|-------------------------------------------------------|
| binary   | BINARY   | A binary value in WKB format that represents a POINT. |

**Example**

```
SELECT ST_POINT(BINARY('0x010100000000000000000000f03f00000000000040'));
```

Output: POINT(1 2)

**ST\_POINT(char)**

Creates a POINT from a CHAR value that represents a point in WKT format.

**Syntax**

```
ST_POINT(char)
```

| Argument | DataType | Description                                   |
|----------|----------|-----------------------------------------------|
| char     | CHAR     | A CHAR that represents a POINT in WKT format. |

**Example**

```
SELECT ST_POINT('POINT(1 2)');
```

Output: POINT(1 2)

**ST\_POINT(lon, lat)**

Creates a POINT defined with the values provided for the longitude and latitude coordinates.

**Syntax**

```
ST_POINT(lon, lat)
```

| Argument | DataType | Description                                            |
|----------|----------|--------------------------------------------------------|
| lon      | DOUBLE   | A DOUBLE value that represents a longitude coordinate. |
| lat      | DOUBLE   | A DOUBLE value that represents a latitude coordinate.  |

**Example**

```
SELECT ST_POINT(87.6386, 41.8843);
```

Output: POINT(87.6386 41.8843)

**ST\_POINTFROMEWKT**

Alias for [Point Constructors](#).

Creates a POINT using an EWKT-formatted CHAR as an input argument.

**Syntax**

```
ST_POINTFROMEWKT(srid_char)
```

| Argument  | DataType | Description                                                                                                                                                                                                                                                                                                                                                                         |
|-----------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| srid_char | CHAR     | A string that includes an Spatial Reference Identifier (SRID) value and an EWKT-formatted value that represents a POINT. Separate the values in the string by a semicolon using this format: 'SRID = value; point_value'. The input string must include the SRID = value; component. However, the database ignores this component as all {{Ocint}} geography types are SRID = 4326. |

### Example

```
SELECT ST_POINTFROMEWKT('SRID=4326;POINT(1 2)');
```

Output: POINT(1 2)

### ST\_POINTFROMGEOJSON

Creates a POINT represented by the specified GeoJSON value as an input argument. A second optional BOOLEAN argument determines if the specified GeoJSON should be converted from a planar to a geodesic representation.

If you specify a NULL value for either argument, the function returns NULL.

### Syntax

```
ST_POINTFROMGEOJSON(geojson [, geodesic ] )
```

| Argument | DataType | Description                                                                                                                                                                                                                                                          |
|----------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo_json | STRING   | A string of a GeoJSON. Valid GeoJSON formats follow IETF standards, which you can generate using the Conversion Functions function. If you specify an invalid GeoJSON, the behavior of the function is undefined. The function returns NULL if any argument is NULL. |
| geodesic | BOOLEAN  | Optional. If you set this argument to TRUE, the specified GeoJSON is converted from a planar to a geodesic representation. This does not affect POINT geospatial objects. The default value is FALSE.                                                                |

### Examples

In this example, the function captures a POINT from an empty GeoJSON representation.

```
SELECT ST_POINTFROMGEOJSON('{ "type": "Point", "coordinates": [] }', false);
```

Output: POINT EMPTY

In this example, the function captures a POINT from a GeoJSON representation.

```
SELECT ST_POINTFROMGEOJSON('{"type":"Point","coordinates":[1,1]}',false);
```

Output: POINT(1 1)

## ST\_POINTFROMGEOHASH

Creates a POINT represented by the specified geohash.

### Syntax

```
ST_POINTFROMGEOHASH(geohash)
```

| Argument | DataType | Description                                                                                                                                                                                                                                                  |
|----------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geohash  | STRING   | A string of a valid geohash. Valid geohashes contain only the characters 0123456789bcdefghjkmnpqrstvwxyz. If you specify an invalid geohash, the function throws an <code>INVALID_ARGUMENT</code> error. The function returns NULL if this argument is NULL. |

### Examples

```
SELECT ST_POINTFROMGEOHASH('c0w3hfls70w3hfls70w3');
```

Output: POINT(-126.00000000000003 48.00000000000007)

```
SELECT ST_POINTFROMGEOHASH('kzn21');
```

Output: POINT(42.60498046875 -5.60302734375)

## ST\_POINTFROMTEXT

Alias for [Point Constructors](#).

## ST\_POINTFROMWKB

Alias for [Point Constructors](#).

### Related Links

[Data Types](#)

[Conversion Functions](#)

### 10.10.6 Polygon Constructors

OcientGeo™ POLYGON constructors use geospatial data to create a POLYGON object.

## ST\_FORCECCW

Creates a standardized polygon from an existing one. Ocient® defines standardized as the exterior being counter-clockwise (CCW) and all holes being clockwise (CW) oriented.

### Syntax

```
ST_FORCECCW(polygon)
```

| Argument | DataType | Description                                                                                    |
|----------|----------|------------------------------------------------------------------------------------------------|
| polygon  | POLYGON  | APOLYGONto be converted on its exterior counterclockwise and clockwise for any interior holes. |

**Example**

```
SELECT ST_FORCECCW(ST_POLYGON('POLYGON((0 0,0 1,1 1,1 0,0 0))'));
```

Output: POLYGON((0 0,1 0,1 1,0 1,0 0))

**ST\_MAKEPOLYGON**

Alias for **Polygon Constructors**.

**ST\_POLYGON**

Creates a POLYGON.

**ST\_POLYGON(geo)**

Creates a POLYGON from the specified POINT, POINT array, LINESTRING, or POLYGON geography.

**Syntax**

```
ST_POLYGON(geo)
```

| Argument | DataType | Description                                                                     |
|----------|----------|---------------------------------------------------------------------------------|
| geo      | POLYGON  | APOINT,POINTarray,LINestring, orPOLYGONvalue to be converted to aPOLYGONobject. |

**Example**

```
SELECT ST_POLYGON(
  ST_LINESTRING(
    ST_POINT [] (
      ST_POINT(1, 2),
      ST_POINT(1, 3),
      ST_POINT(1, 4),
      ST_POINT(1, 2)))
);
```

Output: POLYGON((1 2,1 3,1 4,1 2))

**ST\_POLYGON(char)**

Alias for ST\_MAKEPOLYGON(char).

Creates a POLYGON from the specified CHAR representing a POLYGON in WKT format.

**Syntax**

```
ST_POLYGON(char)
```





```
SELECT ST_POLYGON(
  ST_LINESTRING(ST_POINT [] (ST_POINT(1, 2))),
  ST_LINESTRING [] (ST_LINESTRING(ST_POINT [] (ST_POINT(1, 2)))));
```

Output: POLYGON((1.000000 2.000000), (1.000000 2.000000))

## ST\_POLYGONFROMWKB

Alias for [Polygon Constructors](#).

## ST\_POLYGONFROMEWKT

Alias for the [Polygon Constructors](#) constructor.

Creates a POLYGON using an EWKT-formatted CHAR as an input argument.

### Syntax

```
ST_POLYGONFROMEWKT(char)
```

| Argument | DataType | Description                                                                                                                                                                                                                                                                                                                                      |
|----------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| char     | CHAR     | A string that includes an SRID value and an EWKT-formatted value that represents a POLYGON. Separate the values in the string by a semicolon in this format: 'SRID = value; polygon_value' The input string must include the SRID = value; component. However, the database ignores this component as all Ocint geography types are SRID = 4326. |

### Example

```
SELECT ST_POLYGONFROMEWKT('SRID=4326;POLYGON((1 2,1 3,1 4,1 2))');
```

Output: POLYGON((1 2,1 3,1 4,1 2))

## ST\_POLYGONFROMGEOJSON

Creates a POLYGON represented by the specified GeoJSON value as an input argument.

### Syntax

```
ST_POLYGONFROMGEOJSON(geojson [, geodesic ] )
```

| Argument | DataType | Description                                                                                                                                                                                                                                                                               |
|----------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo_json | STRING   | A string of a GeoJSON. Valid GeoJSON formats follow IETF standards, which you can generate using the <code>Conversion Functions</code> function. If you specify an invalid GeoJSON, the behavior of the function is undefined. The function returns NULL if any argument is a NULL value. |

| Argument | DataType | Description                                                                                                                                                                                                                                                                                |
|----------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geodesic | BOOLEAN  | Optional. If you specify <code>geodesic</code> as <code>TRUE</code> , the function adds points to the resulting <code>POLYGON</code> such that it remains within 10 meters of the original planar polygon. If you do not specify this argument, the value defaults to <code>FALSE</code> . |

### Examples

In this example, the function produces an empty `POLYGON` value from the GeoJSON representation.

```
SELECT ST_POLYGONFROMGEOJSON('{"type":"Polygon","coordinates":[]}',false);
```

*Output:* POLYGON EMPTY

In this example, the function produces a `POLYGON` value from the GeoJSON representation.

```
SELECT ST_POLYGONFROMGEOJSON(
  '{"type":"Polygon","coordinates":[[1,1],[1,5],[5,5],[5,1]]}',
  false);
```

*Output:* POLYGON((1 1, 1 5, 5 5, 5 1))

### ST\_POLYGONFROMTEXT

Alias for [Polygon Constructors](#).

### ST\_WHOLEEARTH

Returns the database internal representation of the whole earth polygon. The [Spatial Operators](#) function returns this `POLYGON` value, with the envelope set as the entire earth.

GIS specifications do not describe a canonical polygon for the whole earth, but this value serves as the Ocient-canonical polygon. The whole earth polygon is also equivalent to `ST_POLYGON('POLYGON((0 0,0 0,0 0,0 0))')`, which can have a separate meaning in other GIS implementations.

#### Syntax

```
ST_WHOLEEARTH()
```

#### Example

```
SELECT ST_WHOLEEARTH();
```

*Output:* POLYGON((0.0 0.0, 0.0 0.0, 0.0 0.0, 0.0 0.0))

### Related Links

[Data Types](#)

[Conversion Functions](#)

### 10.10.7 Spatial Measurement

OcientGeo™ spatial measurement functions can perform basic calculations on geospatial data, such as measuring the distance between two `POINT` objects or the area of a `POLYGON` object.

**ST\_AREA**

Returns the area of the specified geospatial object in the specified unit of measurement.

**Syntax**

```
ST_AREA(geo [, units] [, use_spheroid ] )
```

| Argument     | DataType                    | Description                                                                                                               |
|--------------|-----------------------------|---------------------------------------------------------------------------------------------------------------------------|
| geo          | POINT,LINestring, orPOLYGON | A geospatial object to be measured in the specified measurement unit.                                                     |
| units        | STRING                      | Optional.Supported units are 'FEET', 'KILOMETERS', 'MILES', and 'METERS'.The default value is 'METERS'.                   |
| use_spheroid | BOOLEAN                     | Optional.If you set this value toTRUE, this function uses a spheroid model instead of a spherical model.Defaults toFALSE. |

**Example**

```
SELECT ST_AREA(
  ST_POLYGON(ST_LINestring('LINestring(2 3, 2 4, 4 2, 2 3)'),
    'MILES');
```

*Output:* 196932665.2369081

**ST\_ANGLE**

Calculates the angle between two lines. The lines are represented by three POINT values, including one point for each line (a, b), and a third point (c) representing the vertex where they intersect. The function calculates the spherical angle as formed by the two great circles defined by (a, c) and (b, c).

ST\_ANGLE returns the value in radians, in the range [0, 2pi], excluding the value 2pi.

If either a or b is the same or exactly antipodal to c, the function returns NULL. If any argument is NULL, the function returns NULL.

**Syntax**

```
ST_ANGLE(endpoint_a, vertex_c, endpoint_b)
```

| Argument   | DataType | Description                                                                                                                                        |
|------------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| endpoint_a | POINT    | APOINTthat represents an endpoint for a line between its position and thevertex_cvalue.                                                            |
| vertex_c   | POINT    | APOINTthat represents the intersection between two lines. The lines comprise the endpoints betweenendpoint_aandvertex_c, andendpoint_bandvertex_c. |
| endpoint_b | POINT    | APOINTthat represents an endpoint for a line between its position and thevertex_cvalue.                                                            |

**Example**

```
SELECT ST_ANGLE(ST_POINT(1,0), ST_POINT(0,0), ST_POINT(1,1));
```

Output: 5.49778714378213

## ST\_AZIMUTH

Returns the azimuth of the line from point1 to point2 in radians.

Returns NULL if either of the points are empty. Returns 0 if both of the points are the same.

### Syntax

```
ST_AZIMUTH(point1, point2)
```

| Argument | DataType | Description                                                    |
|----------|----------|----------------------------------------------------------------|
| point1   | POINT    | An endpoint for a line to be calculated for its azimuth value. |
| point2   | POINT    | An endpoint for a line to be calculated for its azimuth value. |

### Example

```
SELECT ST_AZIMUTH(ST_POINT(1,1), ST_POINT(2,2));
```

Output: 0.785398163397448

## ST\_DISTANCE

Returns the minimum distance between the specified arguments.

The ST\_DISTANCE function returns the minimum distance in the specified unit of measurement. The use\_spheroid optional argument enables the computation to use a spheroid model rather than a spherical one.

### Syntax

```
ST_DISTANCE(geo1, geo2 [, units] [, use_spheroid] )
```

| Argument     | DataType                    | Description                                                                                                                                       |
|--------------|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| geo1         | POINT,LINestring, orPOLYGON | A geospatial object to be measured for its minimum distance to geo2.                                                                              |
| geo2         | POINT,LINestring, orPOLYGON | A geospatial object to be measured for its minimum distance to geo1.                                                                              |
| units        | STRING                      | Optional.Supported units include 'FEET', 'KILOMETERS', 'MILES', and 'METERS'.If you do not specify this argument, the value defaults to 'METERS'. |
| use_spheroid | BOOLEAN                     | Optional.If you set this value to TRUE, this function uses a spheroid model instead of a spherical model.Defaults to FALSE.                       |

### Example

```
SELECT ST_DISTANCE(
  ST_LINESTRING('LINESTRING(2 3, 2 4)'),
  ST_LINESTRING('LINESTRING(1 2, 1 4, 1 5)'),
```

```
'METERS',
true);
```

Output: 111050.10339481

To use ST\_DISTANCE as a spatiotemporal function to calculate the minimum simultaneous distance between two linestring-timestamp array pairs, see the [Spatiotemporal Measurement](#) function in [Spatiotemporal Measurement](#).

### ST\_DISTANCESPHERE

Returns the minimum distance between the specified arguments using a spherical computation.

This function is identical to ST\_DISTANCE with the `use_spheroid` argument set to FALSE.

```
ST_DISTANCESPHERE(geo1, geo2 [, units] )
```

See [Spatial Measurement](#) for usage example.

### ST\_DISTANCESPHEROID

Returns the minimum distance between the specified arguments using a spheroid computation.

This function is identical to ST\_DISTANCE with the `use_spheroid` argument set to TRUE.

```
ST_DISTANCESPHEROID(geo1, geo2 [, units] )
```

See [Spatial Measurement](#) for usage example.

### ST\_EUCLIDEANDISTANCE3D

Returns the minimum distance between two geospatial points with altitude values. The distance represents the length of a Euclidean line between the points.

#### Syntax

```
ST_EUCLIDEANDISTANCE3D(point1, altitude1, point2, altitude2 [, units] )
```

| Argument  | DataType | Description                                                                                                                                                              |
|-----------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| point1    | POINT    | A geospatial point that represents a Euclidean line with <b>point2</b> .                                                                                                 |
| altitude1 | DOUBLE   | Distance from the surface of the Earth for <b>point1</b> . This argument uses meters, unless you specify a different unit of measurement with the <b>units</b> argument. |
| point2    | POINT    | A geospatial point that represents a Euclidean line with <b>point1</b> .                                                                                                 |
| altitude2 | DOUBLE   | Distance from the surface of the Earth for <b>point2</b> . This argument uses meters, unless you specify a different unit of measurement with the <b>units</b> argument. |

| Argument | DataType | Description                                                                                                                                                                                                         |
|----------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| units    | VARCHAR  | Optional. This argument specifies the unit of measurement for the distance input arguments and the function output. Supported units are 'FEET', 'KILOMETERS', 'MILES', and 'METERS'. The default value is 'METERS'. |

**Example**

```
SELECT ST_EUCLIDEANDISTANCE3D(
  ST_POINT(0,0),1000.0,
  ST_POINT(0,1),1000.0,
  'METERS');
```

*Output:* 111211.09372321835

**ST\_HAUSDORFFDISTANCE**

Returns the Hausdorff distance between two geographies in a specified measurement. If any argument is NULL or empty, the function returns NULL.

The two geographies can be POINT, LINESTRING, or POLYGON values. The system promotes these geographies to filled polygons. Behavior is undefined if the resulting polygons are not convex.

**Syntax**

```
ST_HAUSDORFFDISTANCE(geo1, geo2 [, units] [, use_spheroid] )
```

| Argument     | DataType                      | Description                                                                                                                      |
|--------------|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| geo1         | POINT, LINESTRING, or POLYGON | A geospatial object to be measured for its Hausdorff distance to geo2.                                                           |
| geo2         | POINT, LINESTRING, or POLYGON | A geospatial object to be measured for its Hausdorff distance to geo1.                                                           |
| units        | STRING                        | Optional. Supported units are 'FEET', 'KILOMETERS', 'MILES', and 'METERS'. The default value is 'METERS'.                        |
| use_spheroid | BOOLEAN                       | Optional. If you set this argument to TRUE, this function uses a spheroid model instead of a spherical model. Defaults to FALSE. |

**Example**

```
SELECT ST_HAUSDORFFDISTANCE(
  ST_POLYGON('POLYGON((1 1, 2 2, 1 2, 1 1))'),
  ST_POLYGON('POLYGON((0 0, 2 1, 2 0, 0 0))'),
  'METERS',
  TRUE);
```

*Output:* 156899.5136831999

## ST\_LENGTH

Returns the length of the specified line in the specified measurement unit.

Returns 0 for all geography types except for LINESTRING. To return the total length of a POLYGON, use [Spatial Measurement](#).

### Syntax

```
ST_LENGTH(geo [, units] [, use_spheroid] )
```

| Argument     | DataType   | Description                                                                                                                      |
|--------------|------------|----------------------------------------------------------------------------------------------------------------------------------|
| geo          | LINESTRING | A line that is to be measured for its length. The function accepts POINT and POLYGON data types, but returns 0.                  |
| units        | STRING     | Optional. Supported units are 'FEET', 'KILOMETERS', 'MILES', and 'METERS'. The default value is 'METERS'.                        |
| use_spheroid | BOOLEAN    | Optional. If you set this argument to TRUE, this function uses a spheroid model instead of a spherical model. Defaults to FALSE. |

### Example

```
SELECT ST_LENGTH(
  ST_LINESTRING('LINESTRING(1 2, 1 4)'),
  'METERS',
  false);
```

Output: 221154.8586057658

## ST\_LENGTH2D

Alias for [Spatial Measurement](#).

## ST\_MINIMUMDISTANCETOSURFACE

Calculates the shortest distance between any point along a Euclidean line segment in three-dimensional space and the surface of the Earth. Two input points and their altitudes define this line segment. The line can go above or below the surface of the Earth.

The function takes the difference of both points from the center of the Earth to calculate the distance between the nearest point of the line and the surface of the Earth.

The function returns the distance in meters or another unit of measurement if specified. If the returned distance is positive, the shortest distance is above the surface. If the result is negative, the shortest distance is below the surface.

### Syntax

```
ST_MINIMUMDISTANCETOSURFACE(point1, altitude1, point2, altitude2 [, units] )
```

| Argument | DataType | Description                                                      |
|----------|----------|------------------------------------------------------------------|
| point1   | POINT    | A geospatial point that represents a Euclidean line with point2. |

| Argument  | Data Type | Description                                                                                                                                                                                                         |
|-----------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| altitude1 | DOUBLE    | Distance from the surface of the Earth for <code>point1</code> . This argument uses meters, unless you specify a different unit of measurement with the <code>units</code> argument.                                |
| point2    | POINT     | A geospatial point that represents a Euclidean line with <code>point1</code> .                                                                                                                                      |
| altitude2 | DOUBLE    | Distance from the surface of the Earth for <code>point2</code> . This argument uses meters, unless you specify a different unit of measurement with the <code>units</code> argument.                                |
| units     | VARCHAR   | Optional. This argument specifies the unit of measurement for the distance input arguments and the function output. Supported units are 'FEET', 'KILOMETERS', 'MILES', and 'METERS'. The default value is 'METERS'. |

### Examples

This example takes a line from two points and calculates the distance of the nearest point to the surface of the Earth.

```
SELECT ST_MINIMUMDISTANCETOSURFACE(
  ST_POINT(0,0),1000.0,
  ST_POINT(0,1),1000.0,
  'METERS');
```

Output: 757.3734914669767

In this example, the output is negative, meaning the smallest distance is actually below the surface of the Earth.

```
SELECT ST_MINIMUMDISTANCETOSURFACE(
  ST_POINT(0,0),1000.0,
  ST_POINT(0,5),1000.0,
  'METERS');
```

Output: -5064.738915202208

### ST\_MAXDISTANCE

Returns maximum distance between the specified arguments.

`ST_MAXDISTANCE` returns the maximum distance in the specified unit of measurement. The `use_spheroid` optional argument enables the computation to use a spheroid model rather than a spherical one.

#### Syntax

```
ST_MAXDISTANCE(geo1, geo2 [, units] [, use_spheroid] )
```

| Argument | Data Type                   | Description                                                                        |
|----------|-----------------------------|------------------------------------------------------------------------------------|
| geo1     | POINT,LINestring, orPOLYGON | A geospatial object to be measured for its maximum distance to <code>geo2</code> . |
| geo2     | POINT,LINestring, orPOLYGON | A geospatial object to be measured for its maximum distance to <code>geo1</code> . |



| Argument     | DataType | Description                                                                                                                      |
|--------------|----------|----------------------------------------------------------------------------------------------------------------------------------|
| units        | STRING   | Optional. Supported units are 'FEET', 'KILOMETERS', 'MILES', and 'METERS'. The default value is 'METERS'.                        |
| use_spheroid | BOOLEAN  | Optional. If you set this argument to TRUE, this function uses a spheroid model instead of a spherical model. Defaults to FALSE. |

### Example

```
SELECT ST_MAXDISTANCE(
  ST_POLYGON(ST_LINESTRING('LINESTRING(1 2, 1 4, 1 7, 1 2)'),
  ST_POLYGON(ST_LINESTRING('LINESTRING(1 2, 1 4, 1 5, 1 2)'),
    'METERS',
    false);
```

Output: 555975.2598904524

To use ST\_MAXDISTANCE as a spatiotemporal function to calculate the maximum simultaneous distance between two linestring-timestamp array pairs, see the [Spatiotemporal Measurement](#) function in [Spatiotemporal Measurement](#).

### ST\_PERIMETER

Returns the length of the exterior (outer ring) of the POLYGON in the specified unit of measurement.

Returns 0 for all geography types except for POLYGON. To return the length of a LINESTRING, use [Spatial Measurement](#).

#### Syntax

```
ST_PERIMETER(geo, [, units] [, use_spheroid] )
```

| Argument     | DataType | Description                                                                                                                           |
|--------------|----------|---------------------------------------------------------------------------------------------------------------------------------------|
| geo          | POLYGON  | A POLYGON that is to be measured for its total perimeter length. The function accepts POINT and LINESTRING data types, but returns 0. |
| units        | STRING   | Optional. Supported units are 'FEET', 'KILOMETERS', 'MILES', and 'METERS'. The default value is 'METERS'.                             |
| use_spheroid | BOOLEAN  | Optional. If you set this argument to TRUE, this function uses a spheroid model instead of a spherical model. Defaults to FALSE.      |

### Example

```
SELECT ST_PERIMETER(
  ST_POLYGON(ST_LINESTRING('LINESTRING(1 2, 1 3, 1 3, 1 2)'),
    'METERS',
    false);
```

*Output:* 222390.1039561809

## ST\_PERIMETER2D

Alias for [Spatial Measurement](#).

### Related Links

[Data Types](#)

[Spatial Operators](#)

[Spatial Relationships](#)

## 10.10.8 Spatial Operators

OcientGeo™ spatial noperators perform geometry calculations on geospatial data to return a different type of geospatial data.

### ST\_BOUNDINGDIAGONAL

Returns the diagonal `LINESTRING` from the minimum point to the maximum point of the bounding box that [Spatial Operators](#) returns.

#### Syntax

```
ST_BOUNDINGDIAGONAL(geo)
```

| Argument | DataType                    | Description                                                                     |
|----------|-----------------------------|---------------------------------------------------------------------------------|
| geo      | POINT,LINESTRING, orPOLYGON | A geospatial object to be measured for the diagonal length of its bounding box. |

#### Example

```
SELECT ST_BOUNDINGDIAGONAL(
  ST_POLYGON(ST_LINESTRING('LINESTRING(1 2, 1 3, 1 3, 1 2)'))
);
```

*Output:* LINESTRING(1 2,1 3)

### ST\_BUFFER

Returns a geography that contains all points where the distance from the geography is less than or equal to the specified distance. The function simplifies the geometry before buffering unless you specify the `simplification_tolerance` argument with a zero or negative value.

The output of the function is an array of polygons. For POINT and LINESTRING data types, the returned array contains one polygon only.

#### Syntax

```
ST_BUFFER(geo, radius [, num_segments_per_quarter_circle]
  [, simplification_tolerance] [, maximum_radius_error]
  [, join_style] [, miter_limit]
  [, endcap_style] [, side_style])
```

| Argument                                     | Data Type                     | Description                                                                                                                                                                                                                                                                                                                          |
|----------------------------------------------|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>geo</code>                             | POINT, LINESTRING, or POLYGON | A geospatial object, specified as a point, linestring, or polygon, which the function buffers.                                                                                                                                                                                                                                       |
| <code>radius</code>                          | DOUBLE                        | Radius, specified as a double, that indicates the distance to buffer the geospatial object in meters. Negative values cause polygons to shrink. Also, negative values cause the result to be empty for points and linestrings.                                                                                                       |
| <code>num_segments_per_quarter_circle</code> | BIGINT                        | Optional. The number of segments to use per quarter circle when parts of the buffer are round, specified as a BIGINT. Only use if the <code>join_style</code> or <code>endcap_style</code> value is set to <code>ROUND</code> . Defaults to 8.                                                                                       |
| <code>simplification_tolerance</code>        | DOUBLE                        | Optional. Value that the function passes to the <code>ST_SIMPLIFY</code> function prior to buffering. If you specify a zero or negative value, the function skips simplification of the geometry.                                                                                                                                    |
| <code>maximum_radius_error</code>            | DOUBLE                        | Optional. Maximum radius error, specified as a double. The returned buffer might be slightly larger than the <code>radius</code> value by this amount. This value is required because arcs of great circles cannot be parallel. Use this argument for linestrings and polygons only. Defaults to 10.0.                               |
| <code>join_style</code>                      | STRING                        | Optional. The type of join to use at the corners where segments of the geometry meet, specified as a string. Supported styles are <code>ROUND</code> , <code>BEVEL</code> , and <code>MITER</code> . Use this argument for linestrings and polygons only. Defaults to <code>ROUND</code> .                                           |
| <code>miter_limit</code>                     | DOUBLE                        | Optional. Miter limit, specified as a double. Use this argument only if the <code>join_style</code> value is set to <code>MITER</code> . A mitered corner is cut off at a distance of <code>miter_ratio * radius</code> from the corner point of the geometry. Use this argument for linestrings and polygons only. Defaults to 1.0. |
| <code>endcap_style</code>                    | STRING                        | Optional. The type of end cap for use at the ends of linestrings, specified as a string. Supported styles are <code>ROUND</code> , <code>SQUARE</code> , and <code>FLAT</code> . Use this argument for linestrings only. Defaults to <code>ROUND</code> .                                                                            |

| Argument   | DataType | Description                                                                                                                                                                                        |
|------------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| side_style | STRING   | Optional. The side, specified as a string, that denotes the side for buffering the geometry. Supported styles are BOTH, LEFT, and RIGHT. Use this argument for linestrings only. Defaults to BOTH. |

**Example**

```
SELECT ST_BUFFER(
  ST_POLYGON('POLYGON((0 0, 1 0, 1 1, 0 0))'), 2000, 8, 500.0, 10.0,
  'MITER', 2.0);
```

*Output:*

```
[POLYGON((2.202700241764562E-18 -0.017986411844962975, 1.0179863625702694 -0.0179863616840571,
1.0179892379654352 1.031486286416526, 1.009550207729886 1.0349816974070565, -0.03498171178141835
-0.009548649645081482, -0.03148633724650228 -0.01798632287046757, 2.202700241764562E-18 -0.017986411844962
```

**ST\_CLOSESTPOINT**

Returns the two-dimensional POINT of one specified geospatial object that is closest to a second specified geospatial object. This function works with any geospatial type.

If either geospatial argument is empty, the function returns NULL.

**Syntax**

```
ST_CLOSESTPOINT(geo1, geo2 [, use_spheroid] )
```

| Argument     | DataType                      | Description                                                                                                                   |
|--------------|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| geo1         | POINT, LINESTRING, or POLYGON | A geospatial object to be computed to determine its closestPOINT value relative to geo2.                                      |
| geo2         | POINT, LINESTRING, or POLYGON | A second geospatial object to be compared to geo1 to find the closestPOINT value of geo1.                                     |
| use_spheroid | BOOLEAN                       | Optional. If you set this value to TRUE, this function uses a spheroid model instead of a spherical model. Defaults to FALSE. |

**Example**

```
SELECT ST_CLOSESTPOINT(
  ST_POLYGON(ST_LINESTRING('LINESTRING(2 3, 2 4, 2 4, 2 3)'),
  ST_POLYGON(ST_LINESTRING('LINESTRING(1 2, 1 4, 1 5, 1 2)')));
```

*Output:* POINT(2,3)

**ST\_CONVEXHULL**

Returns a POLYGON representing the convex hull of the input geometry.

The convex hull is the smallest convex geometry that encloses the input geometry. If the geo argument is empty, the function returns an empty POLYGON.

**Syntax**

```
ST_CONVEXHULL(geo)
```

| Argument | Data Type                     | Description                                                                                                                                                                                                                                 |
|----------|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo      | POINT, LINESTRING, or POLYGON | A geospatial object to be computed for a POLYGON of its convex hull. If geo is a POINT or a single-point LINESTRING or POLYGON, the function returns a POLYGON containing only that point value. If geo is NULL, the function returns NULL. |

**Example**

```
SELECT ST_CONVEXHULL(
  ST_POLYGON('POLYGON((0 0,1 0,1 1,0.5 0.5,0.3 0.3,0 1,0 0))'));
```

*Output:* POLYGON((0.0 0.0, 1.0 0.0, 1.0 1.0, 0.0 1.0, 0.0 0.0))

**ST\_DIFFERENCEARRAY**

Returns an array containing any geospatial objects that are present in the first specified geospatial argument that are not found in the second geospatial argument.

**Syntax**

```
ST_DIFFERENCEARRAY(geo1, geo2)
```

| Argument | Data Type                     | Description                                                                                                                                                           |
|----------|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo1     | POINT, LINESTRING, or POLYGON | A geospatial object to be compared to geo2 to find any unique geospatial values.                                                                                      |
| geo2     | POINT, LINESTRING, or POLYGON | A second geospatial object used to compare with geo1 to find unique values. The function captures only objects that are unique to geo1, and not those unique to geo2. |

**Example**

In this example, the function returns LINESTRING(2.0 2.0, 3.0 3.0) because that portion of the first LINESTRING argument is not present in the second argument.

```
SELECT ST_DIFFERENCEARRAY(
  ST_LINESTRING('LINESTRING(1 1, 2 2, 3 3)'),
  ST_LINESTRING('LINESTRING(1 1, 2 2)'));
```

*Output:* [LINESTRING(2.0 2.0, 3.0 3.0)]

**ST\_ENVELOPE**

Returns a POLYGON that represents the minimum bounding box for the specified geography. If the specified argument is empty, the function returns an empty POLYGON.

A single POINT argument results in a single POINT.

**Syntax**

```
ST_ENVELOPE(geo)
```

| Argument | DataType                    | Description                                                              |
|----------|-----------------------------|--------------------------------------------------------------------------|
| geo      | POINT,LINestring, orPOLYGON | A geospatial object to be computed for the vertices of its bounding box. |

**Example**

```
SELECT ST_ENVELOPE(
  ST_POLYGON(ST_LINestring('LINestring(10 10, 0 0)')));
```

*Output:* POLYGON((0 0,10 0,10 10,0 10,0 0))

**ST\_EXPAND**

Returns the bounding box of a specified geospatial value, which is expanded by a specified length. The unit of length is in meters.

**Syntax**

```
ST_EXPAND(geo1, units_to_expand)
```

| Argument        | DataType                    | Description                                                                                                           |
|-----------------|-----------------------------|-----------------------------------------------------------------------------------------------------------------------|
| geo             | POINT,LINestring, orPOLYGON | A geospatial object to be computed for the vertices of its bounding box with the added <code>units_to_expand</code> . |
| units_to_expand | DOUBLE                      | An additional distance to expand the bounding box of <code>geo</code> . Specify this argument only in meters.         |

**Example**

```
ST_EXPAND(ST_POLYGON('POLYGON((1 1, 2 1, 2 2, 1 1))'), 1.0);
```

*Output:* POLYGON((0.99 0.99, 1.33 0.99, 1.66 0.99, 2.01 0.99, 2.00 2.00, 1.66 2.00, 1.33 2.00, 0.99 2.00, 0.99 0.99))

**ST\_EXTERIORRING**

Returns a LINestring that represents the exterior ring of a provided POLYGON value. If you specify a geography value that is not a POLYGON, the function returns NULL.

**Syntax**

```
ST_EXTERIORRING(polygon)
```

| Argument | DataType | Description                                                        |
|----------|----------|--------------------------------------------------------------------|
| polygon  | POLYGON  | A POLYGON object to be computed for a LINestring of its perimeter. |

**Example**

```
SELECT ST_EXTERIORRING(ST_POLYGON('POLYGON((1 1, 3 1, 2 2, 1 1))'));
```

*Output:* LINESTRING(1.0 1.0, 3.0 1.0, 2.0 2.0, 1.0 1.0)

### ST\_FLIPCOORDINATES

Returns a new geographic object with the X and Y coordinates switched using the specified argument. The return type is the same as the specified type.

#### Syntax

```
ST_FLIPCOORDINATES(geo)
```

| Argument | DataType                    | Description                                                |
|----------|-----------------------------|------------------------------------------------------------|
| geo      | POINT,LINestring, orPOLYGON | A geographic object for switching its X and Y coordinates. |

#### Example

```
ST_FLIPCOORDINATES(ST_POINT('POINT(1 3)'));
```

*Output:* POINT(3.0 1.0)

### ST\_FORCE2D

Convert a geographic object into a two-dimensional geography. The output representations only have the X and Y coordinates.

The returned type is the same as the specified type.

#### Syntax

```
ST_FORCE2D(geo1)
```

| Argument | DataType                    | Description                                                      |
|----------|-----------------------------|------------------------------------------------------------------|
| geo      | POINT,LINestring, orPOLYGON | A geographic object to be converted to a two-dimensional object. |

#### Example

```
SELECT ST_FORCE2D(ST_POINT('POINT(1 1)'));
```

*Output:* POINT(1.0 1.0)

### ST\_INTERIORRINGN

Returns a **LINestring** representing the interior ring of the specified **POLYGON**, which is specified by its index. (1-indexed)

Returns **NULL** if the index is out of range, or if specified a non-**POLYGON** geography.

#### Syntax

```
ST_INTERIORRINGN(polygon, index)
```

| Argument | Data Type | Description                                                                                                                                                                          |
|----------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| polygon  | POLYGON   | APOLYGONObject to be computed for its interior ring.                                                                                                                                 |
| index    | INTEGER   | An index that represents the specific interior ring of the <code>polygon</code> value to be captured. Interior-ring indexes are 1-based, meaning the first object has an index of 1. |

**Example**

```
SELECT ST_INTERIORRINGN(
  ST_POLYGON(
    ST_LINESTRING(ST_POINT [] (ST_POINT(1, 2))),
    ST_LINESTRING [] (ST_LINESTRING(ST_POINT [] (ST_POINT(1, 2))))) ,
  1);
```

Output: LINESTRING(1 2)

**ST\_INTERSECTALL**

Returns the intersection of all geographies in the specified array. All geographies in the array must be the same type. The `ST_INTERSECTALL` function ignores NULL values in the array. The output geography has the same type as the input geometries.

**Syntax**

```
ST_INTERSECTALL(array)
```

| Argument | Data Type                              | Description                                                      |
|----------|----------------------------------------|------------------------------------------------------------------|
| array    | ARRAY of POINT, LINESTRING, or POLYGON | An array of geographic objects, which all must be the same type. |

**Example**

```
SELECT ST_INTERSECTALL(
  ST_POINT [] (ST_POINT(0, 0), ST_POINT(1, 1), ST_POINT(2, 2)));
```

Output: POINT[]

**ST\_INTERSECTIONARRAY**

Returns a geography that represents the point-set intersection of two geographies.

If the two geographies are different types, the function returns an array of the smallest geography. For example, if `geo1` is a POINT, and `geo2` is a POLYGON, the function returns an array of POINT values.

**Syntax**

```
ST_INTERSECTIONARRAY(geo1, geo2)
```

| Argument | Data Type                     | Description                                                                             |
|----------|-------------------------------|-----------------------------------------------------------------------------------------|
| geo1     | POINT, LINESTRING, or POLYGON | A geographic object to be compared to <code>geo2</code> to find a set of intersections. |
| geo2     | POINT, LINESTRING, or POLYGON | A geographic object to be compared to <code>geo1</code> to find a set of intersections. |



**Example**

This example compares two polygons to find any intersections. Because both objects are **POLYGON** types, the result is also a **POLYGON** containing all intersection points.

```
SELECT ST_INTERSECTIONARRAY(
  ST_POLYGON(ST_LINESTRING(ST_POINT(1,2),ST_POINT(3,4))),
  ST_POLYGON(ST_LINESTRING(ST_POINT(1,2),ST_POINT(4,5))));
```

*Output:* [POLYGON((1, 2))]

**ST\_LONGESTLINE**

Returns the longest **LINESTRING** between two specified geospatial arguments.

**Syntax**

```
ST_LONGESTLINE(geo1, geo2, use_spheroid)
```

| Argument     | DataType                    | Description                                                                                                              |
|--------------|-----------------------------|--------------------------------------------------------------------------------------------------------------------------|
| geo1         | POINT,LINestring, orPOLYGON | A geospatial object for the determination of the longestLINestringrelative togeo2.                                       |
| geo2         | POINT,LINestring, orPOLYGON | A geospatial object for the determination of the longestLINestringrelative togeo1.                                       |
| use_spheroid | BOOLEAN                     | Optional.If you set this value toTRUE, this function usesa spheroid model instead of a spherical model.Defaults toFALSE. |

**Example**

```
SELECT ST_LONGESTLINE(
  ST_MAKELINE('LINestring(0 3, 1 3)'),
  ST_MAKELINE('LINestring(2 3, 0 3)'));
```

*Output:* LINestring(0.0 3.0, 2.0 3.0)

To use **ST\_LONGESTLINE** as a spatiotemporal function to calculate the longest line between two points at a concurrent time, see the [Spatiotemporal Operators](#) function.

**ST\_MAKEENVELOPE**

Alias for **ST\_ENVELOPE(ST\_LINESTRING(ST\_POINT(xmin, ymin), ST\_POINT(xmax, ymax)))**.

Returns a **POLYGON** with vertices representing the minimum bounding box for the specified coordinates.

**Syntax**

```
ST_MAKEENVELOPE(xmin, ymin, xmax, ymax)
```

| Argument | DataType | Description                               |
|----------|----------|-------------------------------------------|
| xmin     | DOUBLE   | The minimum x value for the bounding box. |
| ymin     | DOUBLE   | The minimum y value for the bounding box. |
| xmax     | DOUBLE   | The maximum x value for the bounding box. |

| Argument | DataType | Description                               |
|----------|----------|-------------------------------------------|
| ymax     | DOUBLE   | The maximum y value for the bounding box. |

**Example**

```
SELECT ST_MAKEENVELOPE(0, 0, 10, 10);
```

*Output:* POLYGON((0.0 0.0, 10.0 0.0, 10.0 10.0, 0.0 10.0, 0.0 0.0))

**ST\_MINIMUMBOUNDINGCIRCLE**

Returns the smallest circle POLYGON that contains the specified geographic object.

The function requires an additional argument to define the number of segment for inclusion in each quarter of the returned circle.

**Syntax**

```
ST_MINIMUMBOUNDINGCIRCLE(geo, num_segs_per_qt_circ)
```

| Argument             | DataType                    | Description                                                                                      |
|----------------------|-----------------------------|--------------------------------------------------------------------------------------------------|
| geo                  | POINT,LINestring, orPOLYGON | A geospatial object for the determination of the smallest possible circlePOLYGONto encompass it. |
| num_segs_per_qt_circ | INTEGER                     | The number of segments to be included in each quarter of the returned circlePOLYGON.             |

**Example**

```
SELECT ST_MINIMUMBOUNDINGCIRCLE(ST_POLYGON('POLYGON((1 1, 2 1, 2 2, 1 1))'), 3);
```

*Output:* POLYGON((1.49 2.91, 0.27 0.79, 2.72 0.79, 1.49 2.91))

**ST\_MULTIDIFFERENCEARRAY**

Returns an array of geographies representing the parts of the union of the geographies in the first array that do not intersect with the union of geographies in the second array.

This operation is not symmetric, meaning ST\_MULTIDIFFERENCEARRAY(array1, array2) does not produce the same results as ST\_MULTIDIFFERENCEARRAY(array2, array1) unless both arrays are identical.

The function ignores NULL types. For types besides NULL, all geographies in both arrays must be of the same type.

The output geographies are the same type as the input geographies.

**Syntax**

```
ST_MULTIDIFFERENCEARRAY(array1, array2)
```

| Argument | DataType                                  | Description                                                                                                                                                      |
|----------|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| array1   | ARRAYofPOINT,LINestring, orPOLYGONvalues. | An array of geographic objects to be compared to the geographic objects inarray2.The function returns any values in this array that do not intersect witharray2. |

| Argument | DataType                                  | Description                                                                                                                                       |
|----------|-------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| array2   | ARRAYofPOINT,LINestring, orPOLYGONvalues. | An array of geographic objects to be compared to the geographic objects inarray1.The function returns only values that do not intersect inarray1. |

**Example**

```
ST_MULTIDIFFERENCEARRAY(
  POINT[] (ST_POINT(0, 0), ST_POINT(1, 1)),
  POINT[] (ST_POINT(1, 1), ST_POINT(2, 2)));
```

Output: ['POINT(0.0 0.0)']

**ST\_MULTIINTERSECTIONARRAY**

When you specify two arrays of geospatial objects, this function returns an array of any intersections. The function treats each specified array as a union of all its geospatial values.

All geographies in both arrays must be of the same type, not counting NULL values, which the function ignores.

The output geographies are the same type as the input geographies.

**Syntax**

```
ST_MULTIINTERSECTIONARRAY(array1, array2)
```

| Argument | DataType                                  | Description                                                                                                                                                                                |
|----------|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| array1   | ARRAYofPOINT,LINestring, orPOLYGONvalues. | An array of geographic objects to be compared to the geographic objects inarray2for any intersections.The data types in this array must match the data types inarray2unless they are NULL. |
| array2   | ARRAYofPOINT,LINestring, orPOLYGONvalues. | An array of geographic objects to be compared to the geographic objects inarray1for any intersections.The data types in this array must match the data types inarray1unless they are NULL. |

**Example**

```
SELECT ST_MULTIINTERSECTIONARRAY(
  POINT[] (ST_POINT(0, 0), ST_POINT(1, 1)),
  POINT[] (ST_POINT(1, 1), ST_POINT(2, 2)));
```

Output: ['POINT(1.0 1.0)']

**ST\_MULTISYMDIFFERENCEARRAY**

Alias for ST\_MULTIUNIONARRAY(array1, array2) - ST\_MULTIINTERSECTIONARRAY(array1, array2).

When you specify two arrays of geospatial objects, this function returns an array of any geospatial values that do not intersect. The function treats each specified array as a union of all its geospatial values.

All geographies in both arrays must be of the same type, not counting NULL values, which the function ignores.

The output geographies are the same type as the input geographies.

#### Syntax

```
ST_MULTISYMDIFFERENCEARRAY(array1, array2)
```

| Argument | Data Type                                      | Description                                                                                                                                                                                                    |
|----------|------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| array1   | ARRAY of POINT, LINESTRING, or POLYGON values. | An array of geographic objects to be compared to the geographic objects in array2 for any values that do not intersect. The data types in this array must match the data types in array2 unless they are NULL. |
| array2   | ARRAY of POINT, LINESTRING, or POLYGON values. | An array of geographic objects to be compared to the geographic objects in array1 for any values that do not intersect. The data types in this array must match the data types in array1 unless they are NULL. |

#### Example

```
SELECT ST_MULTISYMDIFFERENCEARRAY(
  POINT[] (ST_POINT(0, 0), ST_POINT(1, 1)),
  POINT[] (ST_POINT(1, 1), ST_POINT(2, 2)));
```

Output: ['POINT(0.0 0.0)', 'POINT(2.0 2.0)']

### ST\_MULTIUNIONARRAY

When you specify two arrays of geospatial objects, this function returns one array that represents a union of all geographies in both arrays.

All geographies in both arrays must be the same type. The function ignores NULL values in the arrays. The output geographies are the same type as the input geographies.

#### Syntax

```
ST_MULTIUNIONARRAY(array1, array2)
```

| Argument | Data Type                                      | Description                                                                                                                                       |
|----------|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| array1   | ARRAY of POINT, LINESTRING, or POLYGON values. | An array of geographic objects to be combined with array2. The data types in this array must match the data types in array2 unless they are NULL. |
| array2   | ARRAY of POINT, LINESTRING, or POLYGON values. | An array of geographic objects to be combined with array1. The data types in this array must match the data types in array1 unless they are NULL. |

#### Example

```
SELECT ST_MULTIUNIONARRAY(
  POINT[] (ST_POINT(0, 0), ST_POINT(1, 1)),
  POINT[] (ST_POINT(1, 1), ST_POINT(2, 2)));
```

*Output:* ['POINT(1.0 1.0)', 'POINT(2.0 2.0)', 'POINT(0.0 0.0)']

## ST\_NRINGS

Returns the number of rings of the specified **POLYGON**, including both interior and exterior rings. If you specify a geography object that is not a **POLYGON**, this function returns 0.

### Syntax

ST\_NRINGS(geo)

| Argument | DataType | Description                                                   |
|----------|----------|---------------------------------------------------------------|
| geo      | POLYGON  | An object to be computed to return its total number of rings. |

### Example

```
SELECT ST_NRINGS(
  ST_POLYGON('POLYGON((1.0 2.0, 1.0 4.0, 1.0 5.0, 1.0 2.0))');
```

*Output:* 1

## ST\_NUMINTERIORRINGS or ST\_NUMINTERIORRING

Returns the number of interior rings of the specified **POLYGON**. If you specify a geography object that is not a **POLYGON**, this function returns 0.

### Syntax

ST\_NUMINTERIORRINGS(polygon)

ST\_NUMINTERIORRING(polygon)

| Argument | DataType | Description                                                            |
|----------|----------|------------------------------------------------------------------------|
| polygon  | POLYGON  | An object to be computed to return its total number of interior rings. |

### Example

```
SELECT ST_NUMINTERIORRINGS(
  ST_POLYGON(
    ST_LINESTRING(ST_POINT [] (ST_POINT(1, 2))),
    ST_LINESTRING [] (ST_LINESTRING(ST_POINT [] (ST_POINT(1, 2))))));
```

*Output:* 1

## ST\_POINTONSURFACE

Returns a **POINT** guaranteed to intersect the specified geospatial object.

### Syntax

ST\_POINTONSURFACE(geo1)

| Argument | DataType                    | Description                                             |
|----------|-----------------------------|---------------------------------------------------------|
| geo      | POINT,LINestring, orPOLYGON | An object to be computed for an intersectionPOINTvalue. |

**Example**

```
SELECT ST_POINTONSURFACE(ST_POLYGON('POLYGON((1 1, 2 1, 2 2, 1 1))'));
```

Output: POINT(1.0 1.0)

**ST\_PROJECT**

Returns a POINT by projecting a distance and an azimuth value from the specified starting POINT value. The distance must be in meters, and azimuth must be in radians.

**Syntax**

```
ST_PROJECT(point, distance, azimuth)
```

| Argument | DataType | Description                                                                                    |
|----------|----------|------------------------------------------------------------------------------------------------|
| point    | POINT    | A starting point for locating a returned destination by using the distance and azimuth values. |
| distance | DOUBLE   | A distance value in meters.                                                                    |
| azimuth  | DOUBLE   | An azimuth value in radians.                                                                   |

**Example**

```
SELECT ST_PROJECT(ST_POINT(1, 3), 100, 100);
```

Output: POINT(0.99 3.00)

**ST\_REDUCEPRECISION**

Returns a new geospatial object with all POINT values rounded to the specified decimal precision.

**Syntax**

```
ST_REDUCEPRECISION(geo, precision)
```

| Argument  | DataType                    | Description                                                                                                                                                                                                                 |
|-----------|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo       | POINT,LINestring, orPOLYGON | A geospatial object to be rounded to the specified precision value.                                                                                                                                                         |
| precision | DOUBLE                      | A decimal value that defines the level of precision for returned values. For example, a precision of 0.01 returns values rounded to the hundredth decimal. Do not use digits other than 0 and 1 to represent the precision. |

**Example**

```
SELECT ST_REDUCEPRECISION(ST_POINT('POINT(1.21 1.34)'), 0.1);
```

Output: POINT(1.2 1.3)

**ST\_REMOVEREPEATEDPOINTS**

Returns a new geospatial object with no repeated POINT values. The returned type is the same as the specified type.

**Syntax**

```
ST_REMOVEREPEATEDPOINTS(geo)
```

| Argument | Data Type                     | Description                                                    |
|----------|-------------------------------|----------------------------------------------------------------|
| geo      | POINT, LINESTRING, or POLYGON | A geospatial object for removal of any duplicate POINT values. |

**Example**

```
SELECT ST_REMOVEREPEATEDPOINTS(ST_LINESTRING('LINESTRING(1 2, 1 2, 2 1, 1 0)'));
```

*Output:* LINESTRING(1.0 2.0, 2.0 1.0, 1.0 0.0)

**ST\_REVERSE**

Returns a new geospatial object with the vertexes reversed. The returned type is the same as the specified type.

**Syntax**

```
ST_REVERSE(geo)
```

| Argument | Data Type                     | Description                              |
|----------|-------------------------------|------------------------------------------|
| geo      | POINT, LINESTRING, or POLYGON | A geospatial object for vertex reversal. |

**Example**

```
SELECT ST_REVERSE(ST_LINESTRING('LINESTRING(1 2, 1 10)'));
```

*Output:* LINESTRING(1.0 10.0, 1.0 2.0)

**ST\_SEGMENTIZE**

Returns a geospatial object that the function modifies to have no segment longer than the specified `max_segment_length` in meters.

**Syntax**

```
ST_SEGMENTIZE(geo, max_segment_length)
```

| Argument           | Data Type                     | Description                                                                                                    |
|--------------------|-------------------------------|----------------------------------------------------------------------------------------------------------------|
| geo                | POINT, LINESTRING, or POLYGON | A geospatial object for segment modification.                                                                  |
| max_segment_length | DOUBLE                        | The maximum length of a segment to modify the specified <code>geo</code> object. This value must be in METERS. |

**Example**

```
SELECT ST_SEGMENTIZE(
  ST_POLYGON('POLYGON((1 1, 2 1, 2 2, 1 1))'),
  15000.0);
```

*Output:* POLYGON((1.0 1.0, 2.0 1.0, 2.0 2.0, 1.49 1.50, 1.0 1.0))

**ST\_SHORTESTLINE**

Returns the shortest **LINESTRING** between two specified geospatial arguments.

**Syntax**

```
ST_SHORTESTLINE(geo1, geo2, use_spheroid)
```

| Argument     | DataType                    | Description                                                                                                                                    |
|--------------|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| geo1         | POINT,LINESTRING, orPOLYGON | A geospatial object for the determination of the shortest <b>LINESTRING</b> relative togeo2.                                                   |
| geo2         | POINT,LINESTRING, orPOLYGON | A geospatial object for the determination of the shortest <b>LINESTRING</b> relative togeo1.                                                   |
| use_spheroid | BOOLEAN                     | Optional.If you set this argument to <b>TRUE</b> , this function uses a spheroid model instead of a spherical model.Defaults to <b>FALSE</b> . |

**Example**

```
SELECT ST_SHORTESTLINE(
  ST_MAKELINE('LINESTRING(4 3, 10 3)'),
  ST_MAKELINE('LINESTRING(0 1, 0 3)'));
```

*Output:* LINESTRING(0.0 3.0, 4.0 3.0)

To use **ST\_SHORTESTLINE** as a spatiotemporal operator to calculate the shortest line between two points at a concurrent time, see the [Spatiotemporal Operators](#) function in the Spatiotemporal section.

**ST\_SIMPLIFY**

Returns a simplified version of the specified geography, which is either a **POINT** or **LINESTRING**.

The function simplifies the input geography by replacing nearly straight edges that contain multiple points with a single straight edge. The input geography does not change by more than the specified tolerance, which you specify as a **DOUBLE** in meters.

**Syntax**

```
ST_SIMPLIFY(geo, tolerance)
```

| Argument  | DataType                  | Description                                                                                  |
|-----------|---------------------------|----------------------------------------------------------------------------------------------|
| geo       | POINTor <b>LINESTRING</b> | A geospatial object to be simplified.                                                        |
| tolerance | <b>DOUBLE</b>             | A value that represents the maximum number of meters to simplify in the specified geography. |

**Example**

```
SELECT ST_SIMPLIFY(
  ST_LINESTRING('LINESTRING(0 0, 0.05 0, 0.1 0, 0.15 0, 2 0)'), 1);
```

*Output:* LINESTRING(0 0, 2 0)



## ST\_SIMPLIFYARRAY

Returns a POLYGON array that represents a simplified version of the specified geography, which is either a POINT, LINESTRING, or POLYGON.

The function simplifies the input geography by replacing nearly straight edges that contain multiple points with a single straight edge. The input geography does not change by more than the specified tolerance, which you specify as a DOUBLE in meters.

### Syntax

```
ST_SIMPLIFYARRAY(geo, tolerance)
```

| Argument  | Data Type                     | Description                                                                                  |
|-----------|-------------------------------|----------------------------------------------------------------------------------------------|
| geo       | POINT, LINESTRING, or POLYGON | A geospatial object to be simplified.                                                        |
| tolerance | DOUBLE                        | A value that represents the maximum number of meters to simplify in the specified geography. |

### Example

In this example, ST\_SIMPLIFYARRAY evaluates a POLYGON with an outer ring of 13 points and an inner ring with four points. The example sets a tolerance argument of 10,000 meters, meaning that is the maximum distance that the function simplifies.

```
SELECT ST_SIMPLIFYARRAY(
    ST_POLYGON(
        'POLYGON((0 0, 1 0, 1 0.49, 1.01 0.49, 1.01 0.51, 1 0.51, 1 1, 0
        ),
        10000);
```

Output: \* [POLYGON((0.0 0.5000000762285838, 1.0 0.500000076228584, 1.0 1.0, 0.0 1.0, 0.0 0.5000000762285838)), POLYGON((1.0 0.4998004375649452, 0.5 0.4, 0.0 0.4998004375649452, 0.0 0.0, 1.0 0.0, 1.0 0.4998004375649452))]

As shown in the output, the ST\_SIMPLIFYARRAY function has reduced the size of the outer POLYGON from 13 points to 5.

Note that the function actually increases the number of points in the inner ring, going from four to six. The increase is due to how close the inner ring is to the edges of the outer ring. Taken altogether, the function still reduced the total number of points in the POLYGON.

## ST\_SNAPTOGRID

Returns a new geography value with all POINT values rounded to the specified precisions. This function is similar to [Spatial Operators](#) but it allows you to specify the precision for the x and y coordinates separately.

### Syntax

```
ST_SNAPTOGRID(geo1, precision_x, precision_y)
```

| Argument | Data Type                     | Description                                                          |
|----------|-------------------------------|----------------------------------------------------------------------|
| geo      | POINT, LINESTRING, or POLYGON | A geospatial object to be rounded to the specified precision values. |

| Argument    | DataType | Description                                                                                                                                                                                                             |
|-------------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| precision_x | DOUBLE   | A decimal value representing the level of precision for the x value. For example, a precision of 0.01 returns values rounded to the hundredth decimal. Do not use digits other than 0 and 1 to represent the precision. |
| precision_y | DOUBLE   | A decimal value representing the level of precision for the y value. For example, a precision of 0.1 returns values rounded to the tenth decimal. Do not use digits other than 0 and 1 to represent the precision.      |

**Example**

```
SELECT ST_SNAPTOGRID(
  ST_MAKEPOINT(1.1, 3.11), 1, 0.1);
```

Output: POINT(1.0 3.1)

**ST\_SYMDIFFERENCEARRAY**

Returns a geographic array that contains the parts that are not common between two geographic objects, **geo1** and **geo2**.

**Syntax**

```
ST_SYMDIFFERENCEARRAY(geo1, geo2)
```

| Argument | DataType                      | Description                                                                   |
|----------|-------------------------------|-------------------------------------------------------------------------------|
| geo1     | POINT, LINESTRING, or POLYGON | A geographic object to be compared to the geographic objects in <b>geo2</b> . |
| geo2     | POINT, LINESTRING, or POLYGON | A geographic object to be compared to the geographic objects in <b>geo1</b> . |

**Example**

```
SELECT ST_SYMDIFFERENCEARRAY(
  ST_LINESTRING('LINESTRING(1 1, 2 2)'),
  ST_LINESTRING('LINESTRING(1 1, 2 2, 3 3)'));
```

Output: [LINESTRING(2.0 2.0, 3.0 3.0)]

**ST\_UNIONARRAY**

Performs a union of the input geography values to produce a geographic array.

**Syntax**

```
ST_UNIONARRAY(geo1, geo2)
```

| Argument | DataType                      | Description                                                                   |
|----------|-------------------------------|-------------------------------------------------------------------------------|
| geo1     | POINT, LINESTRING, or POLYGON | A geographic object to be unified with the geographic values in <b>geo2</b> . |

| Argument | DataType                    | Description                                                           |
|----------|-----------------------------|-----------------------------------------------------------------------|
| geo2     | POINT,LINestring, orPOLYGON | A geographic object to be unified with the geographic values in geo1. |

**Example**

```
SELECT ST_UNIONARRAY(
  ST_LINestring('LINestring(0 0, 1 1, 2 2)'),
  ST_LINestring('LINestring(1 1, 2 2, 3 3)'));
```

Output: ['LINestring(0.0 0.0, 1.0 1.0, 2.0 2.0, 3.0 3.0)']

**Related Links**

[Data Types](#)

[Spatial Measurement](#)

[Spatial Relationships](#)

**10.10.9 Spatial Relationships**

OcientGeo™ spatial relationship functions use arguments to test for different types of spatial relationships.

**ST\_CLUSTERDBSCAN**

Returns the cluster number for each input geography based on a two-dimensional implementation of the density-based spatial clustering of applications with noise (DBSCAN) algorithm.

This function is a window function because the resulting value for an individual row depends on other rows and not just itself. No frame specification is allowed.

This function supports sort orders, but it is unnecessary unless deterministic resolution of tie-breaking in cluster determination is required. This tie-breaking happens only in situations where a point on the boundary of a cluster is equally valid for multiple clusters. In these cases, the result depends on the sort order of the data. For this reason, you might want to sort the data to force the deterministic behavior by using an **ORDER BY** in the **OVER** clause.

The function returns NULL for any row not included in a cluster.

**Syntax**

```
ST_CLUSTERDBSCAN(geo, eps, min_neighbors)
```

| Argument      | DataType                    | Description                                                                                                                              |
|---------------|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| geo           | POINT,LINestring, orPOLYGON | A geospatial object to be grouped into a cluster based on the DBSCAN algorithm.                                                          |
| eps           | DOUBLE                      | A numeric argument that controls the separation between objects before they are considered a new cluster. The eps argument is in meters. |
| min_neighbors | BIGINT                      | The minimum number of objects within the appropriate distance of each other that need to exist before it is considered a cluster.        |

**Example**

```
SELECT ST_CLUSTERDBSCAN(ST_POINT(1.5, 1.5), 1.0, 1) OVER();
```

Output: 1

## ST\_CONTAINS

Returns TRUE if the first geographic argument, **geo1**, contains the second geographic argument, **geo2**.

**geo1** contains **geo2** only if all of these criteria are met:

- No POINT values of **geo2** are located at the exterior of **geo1**.
- At least one POINT in the interior of **geo2** lies in the interior of **geo1**.

Geographies do not contain their boundaries, therefore all POINT values in a LINESTRING are interior, except for the endpoints, which are regarded as exterior boundaries.

If either **geo1** or **geo2** is NULL, then the function returns NULL.

### Syntax

```
SELECT ST_CONTAINS(geo1, geo2)
```

| Argument | DataType                      | Description                                                                                                                        |
|----------|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| geo1     | POINT, LINESTRING, or POLYGON | A geospatial object to be compared to geo2. The POINT values in geo1 are evaluated to determine if they are contained within geo2. |
| geo2     | POINT, LINESTRING, or POLYGON | A geospatial object to be compared to geo1. The POINT values in geo2 are evaluated to determine if they are contained within geo1. |

### Example

```
SELECT ST_CONTAINS(
  ST_LINEFROMTEXT('LINESTRING(0 0, 1 1, 2 2)'),
  ST_POINT(1, 1));
```

Output: true

## ST\_CONTAINSPROPERLY

This function is similar to ST\_CONTAINS, but it has stricter criteria.

Returns true if **geo2** lies entirely in the interior of **geo1** and does not intersect or touch the boundary or exterior points.

Returns NULL if either **geo1** or **geo2** is NULL.

### Syntax

```
ST_CONTAINSPROPERLY(geo1, geo2)
```

| Argument | DataType                      | Description                                                                                                                        |
|----------|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| geo1     | POINT, LINESTRING, or POLYGON | A geospatial object to be compared to geo2. The POINT values in geo1 are evaluated to determine if they are contained within geo2. |

| Argument | DataType                    | Description                                                                                                                        |
|----------|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| geo2     | POINT,LINestring, orPOLYGON | A geospatial object to be compared to geo1. The POINT values in geo2 are evaluated to determine if they are contained within geo1. |

### Examples

In this example, the first geographic object is a **POLYGON** representing a simple square (0 0, 1 0, 1 1, 0 1). The second geographic object is a small **LINestring** that spans from the point (0.25 0.25) to the point (0.75 0.75).

Because the line is located entirely within the polygon, never touching its boundaries, the function evaluates to **true**.

```
SELECT ST_CONTAINSProPERLY(
  ST_MakePOLYGON(
    ST_MakeLINE(
      ARRAY [ST_POINT(0, 0), ST_POINT(1, 0), ST_POINT(1, 1), ST_POINT(0, 1), ST_POINT(0, 0)]),
    ST_MakeLINE(ST_POINT(0.25, 0.25), ST_POINT(0.75, 0.75)));
```

*Output:* TRUE

The second geographic object is slightly altered in this example, where the line spans from (0 0) to (1 1). Both of these endpoints touch the boundaries of the polygon, so the function now evaluates to **false**.

```
SELECT ST_CONTAINSProPERLY(
  ST_MakePOLYGON(
    ST_MakeLINE(
      ARRAY [ST_POINT(0, 0), ST_POINT(1, 0), ST_POINT(1, 1), ST_POINT(0, 1), ST_POINT(0, 0)]),
    ST_MakeLINE(ST_POINT(0, 0), ST_POINT(1, 1)));
```

*Output:* FALSE

### ST\_COVERS

Returns TRUE if no POINT in geo2 is outside of geo1.

Returns NULL if either geo1 or geo2 is NULL.

#### Syntax

```
ST_COVERS(geo1, geo2)
```

| Argument | DataType                    | Description                                                                                                                        |
|----------|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| geo1     | POINT,LINestring, orPOLYGON | A geospatial object to be compared to geo2. The POINT values in geo2 are evaluated to determine if they are contained within geo2. |
| geo2     | POINT,LINestring, orPOLYGON | A geospatial object to be compared to geo1. The POINT values in geo2 are evaluated to determine if they are contained within geo1. |

### Example

```
SELECT ST_COVERS(
  ST_MakePOLYGON(
    ST_MakeLINE(
```

```
ARRAY [ST_POINT(0, 0), ST_POINT(1, 0), ST_POINT(1, 1), ST_POINT(0, 1), ST_POINT(0, 0)])),
ST_MAKELINE(ST_POINT(0, 0), ST_POINT(1, 1)));
```

Output: TRUE

## ST\_COVEREDBY

This function operates similarly to ST\_COVERS, except the arguments are interchanged.

Returns TRUE if no POINT in **geo1** is outside of **geo2**.

Returns NULL if either **geo1** or **geo2** is NULL.

### Syntax

```
ST_COVEREDBY(geo1, geo2)
```

| Argument | Data Type                   | Description                                                                                                                                            |
|----------|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo1     | POINT,LINestring, orPOLYGON | A geospatial object to be compared to <b>geo2</b> .ThePOINTvalues in <b>geo1</b> are evaluated to determine if they are contained within <b>geo2</b> . |
| geo2     | POINT,LINestring, orPOLYGON | A geospatial object to be compared to <b>geo1</b> .ThePOINTvalues in <b>geo1</b> are evaluated to determine if they are contained within <b>geo2</b> . |

### Example

```
SELECT ST_COVEREDBY(
  ST_POINT(1,1),
  ST_POLYGON('POLYGON((1 1, 2 1, 2 2, 1 1))'));
```

Output: TRUE

## ST\_CROSSES

Returns TRUE if two geospatial objects meet this criteria:

- The intersection of the geospatial interiors is not empty.
- The intersection is not equal to **geo1** or **geo2**.
- Neither geospatial object is a single POINT.

Returns NULL if either argument is NULL.

### Syntax

```
ST_CROSSES(geo1, geo2)
```

| Argument | Data Type                   | Description                                                                       |
|----------|-----------------------------|-----------------------------------------------------------------------------------|
| geo1     | POINT,LINestring, orPOLYGON | A geospatial object to be compared to <b>geo2</b> to evaluate whether they cross. |
| geo2     | POINT,LINestring, orPOLYGON | A geospatial object to be compared to <b>geo1</b> to evaluate whether they cross. |

### Example

```
SELECT ST_CROSSES(
  ST_LINESTRING(ST_POINT(0,0), ST_POINT(2,2)),
  ST_LINESTRING(ST_POINT(1,0), ST_POINT(1,5)));
```

Output: TRUE

## ST\_DISJOINT

Returns **true** if the specified geographies have no intersection, including boundaries. Both geographic arguments can be different types.

This function is the opposite of the [Spatial Relationships](#) function.

### Syntax

```
ST_DISJOINT(geo1, geo2)
```

| Argument | Data Type                     | Description                                                                           |
|----------|-------------------------------|---------------------------------------------------------------------------------------|
| geo1     | POINT, LINESTRING, or POLYGON | A geospatial object to be compared to <b>geo2</b> to evaluate whether they intersect. |
| geo2     | POINT, LINESTRING, or POLYGON | A geospatial object to be compared to <b>geo1</b> to evaluate whether they intersect. |

### Example

```
SELECT ST_DISJOINT(
  ST_POLYGON(ST_LINESTRING('LINESTRING(2 3, 2 4, 2 4, 2 3)'),
  ST_POLYGON(ST_LINESTRING('LINESTRING(1 2, 1 4, 1 5, 1 2)')));
```

Output: TRUE

## ST\_DWITHIN

Returns **TRUE** if the geographies are within a specified distance in meters.

### Syntax

```
ST_DWITHIN(geo1, geo2, distance_meters)
```

| Argument        | Data Type                     | Description                                                                                                                           |
|-----------------|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| geo1            | POINT, LINESTRING, or POLYGON | A geospatial object to be compared to <b>geo2</b> to evaluate whether they are within the specified distance <b>distance_meters</b> . |
| geo2            | POINT, LINESTRING, or POLYGON | A geospatial object to be compared to <b>geo1</b> to evaluate whether they are within the specified distance <b>distance_meters</b> . |
| distance_meters | DOUBLE                        | A numeric value that represents the number of meters to evaluate the distance between <b>geo1</b> and <b>geo2</b> .                   |

### Example

```
SELECT ST_DWITHIN(
  ST_POINT('POINT(1 1)'),
  ST_POINT('POINT(0 0)'),
  100);
```

Output: FALSE

## ST\_EQUALS

Returns TRUE if both geographies are spatially equal.

### Syntax

```
ST_EQUALS(geo1, geo2)
```

| Argument | DataType                    | Description                                                                    |
|----------|-----------------------------|--------------------------------------------------------------------------------|
| geo1     | POINT,LINestring, orPOLYGON | A geospatial object to be compared to geo2 to evaluate whether they are equal. |
| geo2     | POINT,LINestring, orPOLYGON | A geospatial object to be compared to geo1 to evaluate whether they are equal. |

### Example

```
SELECT ST_EQUALS(
  ST_POLYGON(ST_LINestring('LINestring(10 10, 0 0, 0 0, 10 10)'),
  ST_POLYGON(ST_LINestring('LINestring(10 10, 0 0, 0 0, 10 10)')));
```

Output: True

## ST\_INTERSECTS

Returns TRUE if the specified geographies intersect, including boundaries.

This function is the opposite of the [Spatial Relationships](#) function.

### Syntax

```
ST_INTERSECTS(geo1, geo2)
```

| Argument | DataType                    | Description                                                                    |
|----------|-----------------------------|--------------------------------------------------------------------------------|
| geo1     | POINT,LINestring, orPOLYGON | A geospatial object to be compared to geo2 to evaluate whether they intersect. |
| geo2     | POINT,LINestring, orPOLYGON | A geospatial object to be compared to geo1 to evaluate whether they intersect. |

### Example

```
SELECT ST_INTERSECTS(
  ST_POLYGON(ST_LINestring('LINestring(2 3, 2 4, 2 4, 2 3)'),
  ST_POLYGON(ST_LINestring('LINestring(1 2, 1 4, 1 5, 1 2)')));
```

Output: FALSE



**ST\_ISCCW**

Alias for [Spatial Relationships](#).

**ST\_ISCLOSED**

Returns TRUE if an input LINESTRING has starting and ending points that are equal.

If you specify a POLYGON or POINT value, the function returns **true** if the specified geography is not empty.

**Syntax**

```
ST_ISCLOSED(line)
```

| Argument | DataType                      | Description                                                                                                                                                                                                                                     |
|----------|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| line     | POINT, LINESTRING, or POLYGON | If you specify a LINESTRING object, the function evaluates it to determine whether its starting and ending points are equal. If you specify a POLYGON or POINT value, the function returns <b>true</b> if the specified geography is not empty. |

**Example**

```
SELECT ST_ISCLOSED(
  ST_LINESTRING('LINESTRING(1 2, 1 3, 1 2)'));
```

Output: TRUE

**ST\_ISPOLYGONCW**

Alias for `BOOLNOT(ST_ISCCW(poly))`.

Returns TRUE if an input POLYGON has an exterior that is clockwise.

If the POLYGON argument is empty, this function returns TRUE. If the argument is NULL, this function returns NULL.

**Syntax**

```
ST_ISPOLYGONCW(poly)
```

| Argument | DataType | Description                                                                                        |
|----------|----------|----------------------------------------------------------------------------------------------------|
| poly     | POLYGON  | A POLYGON object to evaluate to determine whether its point values align in a clockwise direction. |

**Example**

```
SELECT ST_ISPOLYGONCW(
  ST_POLYGON('POLYGON((0 0,1 0,1 1,0 1,0 0))'));
```

Output: FALSE

**ST\_ISPOLYGONCCW**

Alias for ST\_ISCCW.

Returns TRUE if an input POLYGON has an exterior that is counter-clockwise.

If the POLYGON argument is empty, this function returns TRUE. If the argument is NULL, this function returns NULL.

**Syntax**

```
ST_ISPOLYGONCCW(poly)
```

| Argument | DataType | Description                                                                                              |
|----------|----------|----------------------------------------------------------------------------------------------------------|
| poly     | POLYGON  | APOLYGONObject to evaluate to determine whether its point values align in a counter-clockwise direction. |

**Example**

```
SELECT ST_ISPOLYGONCCW(
  ST_POLYGON('POLYGON((0 0,1 0,1 1,0 1,0 0))'));
```

*Output:* TRUE

**ST\_ISRING**

Returns TRUE if the specified LINESTRING is closed and does not intersect itself. This function is equivalent to evaluating both [Spatial Relationships](#) and [Spatial Relationships](#).

If you specify a value other than LINESTRING, the function throws an error.

**Syntax**

```
ST_ISRING(geo)
```

| Argument | DataType   | Description                                                                                                                      |
|----------|------------|----------------------------------------------------------------------------------------------------------------------------------|
| line     | LINESTRING | ALINESTRINGObject to evaluate to determine whether its starting and ending points are equal and whether it has no intersections. |

**Example**

```
SELECT ST_ISRING(
  ST_LINESTRING('LINESTRING(1 2, 2 4, 1 3, 1 2)'));
```

*Output:* TRUE

**ST\_ISSIMPLE**

For LINESTRING values, this function returns FALSE if any line segments intersect anywhere besides the endpoints.

For POLYGON values, the function returns FALSE if either the exterior ring or any interior hole is not simple.

For POINT values, the function always returns TRUE.

**Syntax**

**ST\_ISSIMPLE(geo)**

| Argument | DataType                    | Description                                                                                                                                                                                                                                                                                                                                  |
|----------|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo      | POINT,LINestring, orPOLYGON | A geospatial object to evaluate to determine if any points intersect.ForLINestringvalues, this function returnsFALSEif any line segments intersect anywhere besides the endpoints.ForPOLYGONvalues, the function returnsFALSEif either the exterior ring or any interior hole is not simple.ForPOINTvalues, the function always returnsTRUE. |

**Example**

```
SELECT ST_ISSIMPLE(
  ST_POLYGON('POLYGON((1 2, 3 4, 3 5, 1 2))');
```

*Output:* TRUE

**ST\_ISVALID**

Returns TRUE if the specified geospatial value is a well-formed and valid geography according to the OGC standards.

LINestring values are not valid if they are composed by more than one POINT with the same coordinates.

POLYGON values are valid if they are composed by one external ring (shell) and zero or more internal rings. A ring is a LINestring where only the first and last POINT intersect to each other. All internal rings need to be contained inside the shell, and none of the rings should intersect with each other in more than one POINT.

**Syntax**

**ST\_ISVALID(geo1)**

| Argument | DataType                    | Description                                                                    |
|----------|-----------------------------|--------------------------------------------------------------------------------|
| geo      | POINT,LINestring, orPOLYGON | A geospatial object to evaluate to determine if it is a well-formed geography. |

**Example**

```
SELECT ST_ISVALID(
  ST_LINestring('LINestring(1 2, 2 4)');
```

*Output:* TRUE

**ST\_OVERLAPS**

Returns TRUE if both geographic arguments are of the same dimension and intersect, but neither contains the other. Returns NULL if either geo1 or geo2 is NULL.

**Syntax**

**ST\_OVERLAPS(geo1, geo2)**

| Argument | DataType                    | Description                                                                      |
|----------|-----------------------------|----------------------------------------------------------------------------------|
| geo1     | POINT,LINestring, orPOLYGON | A geospatial object for evaluation if it is shaped similarly and intersectsgeo2. |
| geo2     | POINT,LINestring, orPOLYGON | A geospatial object for evaluation if it is shaped similarly and intersectsgeo1. |

**Example**

```
SELECT ST_OVERLAPS(
  ST_LINestring(ST_POINT(0,0), ST_POINT(2,2)),
  ST_LINestring(ST_POINT(1,1), ST_POINT(3,3)));
```

*Output:* TRUE

**ST\_POINTINSIDECIRCLE**

Returns TRUE if the geographic object is inside a circle centered at the specified point coordinates and with the specified radius.

**Syntax**

```
ST_POINTINSIDECIRCLE(geo, center_x, center_y, circle_radius)
```

| Argument      | DataType                    | Description                                                                                        |
|---------------|-----------------------------|----------------------------------------------------------------------------------------------------|
| geo           | POINT,LINestring, orPOLYGON | A geospatial object for evaluation if it is contained within a circle of the specified dimensions. |
| center_x      | DOUBLE                      | The x coordinate of the center of the circle to evaluate.                                          |
| center_y      | DOUBLE                      | The y coordinate of the center of the circle to evaluate.                                          |
| circle_radius | DOUBLE                      | The radius in meters of the circle to evaluate.                                                    |

**Example**

```
SELECT ST_POINTINSIDECIRCLE(
  ST_POINT(1, 3), 1, 2, 3000000);
```

*Output:* TRUE

**ST\_RELATE**

Returns the DE-9IM intersection string that represents the nature of the intersection with the specified geographies. If you specify an intersection pattern as the third argument, this function returns TRUE if the intersection of the geographies satisfies the pattern and FALSE otherwise.

**Syntax**

```
ST_RELATE(geo1, geo2, [, intersectionPattern])
```

| Parameter           | Data Type                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo1                | POINT,LINestring, orPOLYGON | A geospatial object to relate withgeo2.                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| geo2                | POINT,LINestring, orPOLYGON | A geospatial object to relate withgeo1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| intersectionPattern | STRING                      | Optional.Intersection pattern to query the nature of the intersection ofgeo1andgeo2against.Must contain exactly 9 characters, corresponding to the component (interior, boundary, or exterior) intersections considered by DE-9IM. Supported characters are:T— Components intersect.F— Components do not intersect.0— Component intersection is zero-dimensional.1— Component intersection is 1-dimensional.2— Component intersection is 2-dimensional.*— Specify no restriction on the component intersection. |

### Examples

Determine the nature of the intersection between the point (0.0, 0.0) and the line from -1 0 to 1 0. This function returns the DE-9IM string 0FFFFF102, which means that the boundaries do not intersect at all, but there is evidence of an intersection at the zero, one, and two-dimensional spaces.

```
SELECT ST_RELATE(ST_POINT(0.0, 0.0),
  ST_LINestring('LINestring(-1 0, 1 0)'));
```

Output: 0FFFFF102

Determine whether the intersection between the point (0.0, 0.0) and line from -1 0 to 1 0 follows the DE-9IM pattern string TF\*FFF102. This function returns TRUE, which means the nature of the intersection between these two geographies conforms to this pattern.

```
SELECT ST_RELATE(ST_POINT(0.0, 0.0),
  ST_LINestring('LINestring(-1 0, 1 0)'), 'TF*FFF102');
```

Output: TRUE

### ST\_TOUCHES

Returns TRUE if the only POINT values in common between the two geographic arguments lie in the union of their boundaries.

#### Syntax

```
ST_TOUCHES(geo1, geo2)
```

| Argument | DataType                    | Description                                           |
|----------|-----------------------------|-------------------------------------------------------|
| geo1     | POINT,LINestring, orPOLYGON | A geospatial object for evaluation if it touchesgeo2. |
| geo2     | POINT,LINestring, orPOLYGON | A geospatial object for evaluation if it touchesgeo1. |

**Example**

```
SELECT ST_TOUCHES(
  ST_LINESTRING('LINESTRING(1.0 3.0, 0.0 3.0)'),
  ST_LINESTRING('LINESTRING(1.0 3.0, 2.0 3.0)');
```

*Output:* TRUE

**ST\_WITHIN**

Alias for [Spatial Relationships](#).

**Related Links**

[Data Types](#)

[Spatial Measurement](#)

[Spatial Operators](#)

**10.10.10 Spatiotemporal Measurement**

OcientGeo™ spatiotemporal measurement functions can perform basic calculations on geospatial data that is paired with **TIMESTAMP** data.

**ST\_DISTANCE**

Returns the two-dimensional interpolated minimum simultaneous distance between two **LINESTRING-TIMESTAMP** array pairs in the specified unit of measurement.

If any argument is **NULL** or empty, the function returns **NULL**. If either **LINESTRING-TIMESTAMP** array pair has a mismatched number of points and timestamps, the function returns an error.

Behavior is undefined if either timestamp array is not monotonically increasing.

```
ST_DISTANCE(geo1, ts_arr1, geo2, ts_arr2, units [, use_spheroid] )
```

| Argument     | DataType        | Description                                                                                                                                                  |
|--------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo1         | LINESTRING      | A geospatial object to be evaluated for its minimum cotemporal distance from geo2.                                                                           |
| ts_arr1      | TIMESTAMP ARRAY | A timestamp array that is paired with geo1.                                                                                                                  |
| geo2         | LINESTRING      | A geospatial object to be evaluated for its minimum cotemporal distance from geo1.                                                                           |
| ts_arr2      | TIMESTAMP ARRAY | A timestamp array that is paired with geo2.                                                                                                                  |
| units        | STRING          | The unit of measurement used for the minimum cotemporal distance. Accepted options include “FEET”, “KILOMETERS”, “MILES”, or “METERS”. Defaults to “METERS”. |
| use_spheroid | BOOLEAN         | Optional. If you set this argument to <b>TRUE</b> , this function uses a spheroid model instead of a spherical model. Defaults to <b>FALSE</b> .             |

**Example**

```
SELECT ST_DISTANCE(
  ST_LINESTRING('linestring(0 0,0 12)'),
  TIMESTAMP [] (
    '2020-07-04 07:00:00.000000000',
    '2020-07-04 11:00:00.000000000'),
  ST_LINESTRING('linestring(0 0,6 0)'),
  TIMESTAMP [] (
    '2020-07-04 08:00:00.000000000',
    '2020-07-04 10:00:00.000000000'),
  'MILES',
  TRUE);
```

Output: 206.124

To use `ST_DISTANCE` as a spatial function to calculate the minimum distance between two points, see the [Spatial Measurement](#) function in the Spatial Measurement section.

**ST\_MAXDISTANCE**

Returns the two-dimensional interpolated maximum cotemporal distance between two `LINESTRING-TIMESTAMP` array pairs in the specified unit of measurement.

If any argument is `NULL` or empty, the function returns `NULL`. If either `LINESTRING-TIMESTAMP` array pair has a mismatched number of points and timestamps, the function returns an error.

Behavior is undefined if either timestamp array is not monotonically increasing.

```
ST_MAXDISTANCE(geo1, ts_arr1, geo2, ts_arr2, units [, use_spheroid] )
```

| Argument     | DataType        | Description                                                                                                                                                  |
|--------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geo1         | LINESTRING      | A geospatial object to be evaluated for its maximum cotemporal distance from geo2.                                                                           |
| ts_arr1      | TIMESTAMP ARRAY | A <code>TIMESTAMP ARRAY</code> that is paired with geo1.                                                                                                     |
| geo2         | LINESTRING      | A geospatial object to be evaluated for its maximum cotemporal distance from geo1.                                                                           |
| ts_arr2      | TIMESTAMP ARRAY | A <code>TIMESTAMP ARRAY</code> that is paired with geo2.                                                                                                     |
| units        | STRING          | The unit of measurement used for the maximum cotemporal distance. Accepted options include “FEET”, “KILOMETERS”, “MILES”, or “METERS”. Defaults to “METERS”. |
| use_spheroid | BOOLEAN         | Optional. If you set this argument to <code>TRUE</code> , this function uses a spheroid model instead of a spherical model. Defaults to <code>FALSE</code> . |

**Example**

```
SELECT ST_MAXDISTANCE(
  ST_LINESTRING('linestring(0 0,0 12)'),
  TIMESTAMP[] ('2020-07-04 07:00:00.000000000', '2020-07-04 11:00:00.000000000'),
  ST_LINESTRING('linestring(0 0,6 0)'),
  TIMESTAMP[] ('2020-07-04 08:00:00.000000000', '2020-07-04 10:00:00.000000000'),
  'MILES',
  TRUE);
```

Output: 743.83

To use ST\_MAXDISTANCE as a spatial function to calculate the maximum distance between two points, see the [Spatial Measurement](#) function in the Spatial Measurement section.

## ST\_TOTALSECONDSININTERSECTION

Returns the total number of seconds spent in the intersection result calculated by the spatiotemporal version of [Spatiotemporal Operators](#).

Execute this function only with the result from ST\_INTERSECTION. If the input is an empty, correctly typed tuple, this function returns 0 seconds. If the input is NULL, this function returns NULL.

ST\_TOTALSECONDSININTERSECTION(tp1)

| Argument | DataType                              | Description                                                                                      |
|----------|---------------------------------------|--------------------------------------------------------------------------------------------------|
| tp1      | Tuple<br>of LINESTRING-TIMESTAMPpairs | A tuple<br>of LINESTRING-TIMESTAMPpairs used<br>to evaluate how long an intersection<br>existed. |

### Example

```
SELECT ST_TOTALSECONDSININTERSECTION(
  ST_INTERSECTION(
    ST_LINESTRING('LINESTRING(1 1,2 2,1 1)'),
    TIMESTAMP[] (
      '2020-07-04 07:00:00.000000000',
      '2020-07-04 11:00:00.000000000',
      '2020-07-04 12:00:00.000000000'),
    ST_LINESTRING('LINESTRING(2 2,1 1)')));
```

Output: 18000.0

### Related Links

[Data Types](#)

[Spatiotemporal Operators](#)

#### 10.10.11 Spatiotemporal Operators

OcientGeo™ spatiotemporal operators perform calculations on geospatial data using an array of TIMESTAMP arguments.



## ST\_LONGESTLINE

With the specified two `LINESTRING-TIMESTAMP` ARRAY pairs, this function returns a two-point `LINESTRING` that represents the maximum distance between points at a concurrent time.

If the two `LINESTRING` values do not overlap temporally, this function returns `NULL`.

### Syntax

```
ST_LONGESTLINE(linestring1, ts_arr1, linestring2, ts_arr2 [, use_spheroid] )
```

| Argument                  | DataType                     | Description                                                                                                                                                  |
|---------------------------|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>linestring1</code>  | <code>LINESTRING</code>      | A geospatial object to be evaluated for its maximum cotemporal distance from <code>linestring2</code> .                                                      |
| <code>ts_arr1</code>      | <code>TIMESTAMP ARRAY</code> | A timestamp array that is paired with <code>linestring1</code> . Behavior is undefined if this timestamp array is not monotonically increasing.              |
| <code>linestring2</code>  | <code>LINESTRING</code>      | A geospatial object to be evaluated for its maximum cotemporal distance from <code>linestring1</code> .                                                      |
| <code>ts_arr2</code>      | <code>TIMESTAMP ARRAY</code> | A timestamp array that is paired with <code>linestring2</code> . Behavior is undefined if this timestamp array is not monotonically increasing.              |
| <code>use_spheroid</code> | <code>BOOLEAN</code>         | Optional. If you set this argument to <code>TRUE</code> , this function uses a spheroid model instead of a spherical model. Defaults to <code>FALSE</code> . |

### Example

```
SELECT ST_LONGESTLINE(
  ST_MAKELINE(ST_POINT(0, 0), ST_POINT(0, 12)),
  TIMESTAMP [] (
    '2020-07-04 07:00:00.000000000',
    '2020-07-04 11:00:00.000000000'),
  ST_MAKELINE(ST_POINT(0, 0), ST_POINT(8, 0)),
  TIMESTAMP [] (
    '2020-07-04 08:00:00.000000000',
    '2020-07-04 10:00:00.000000000'));
```

Output: `LINESTRING(0 9, 8 0)`

To use `ST_LONGESTLINE` as a spatial function to calculate the longest line between two points, see the [Spatial Operators](#) function.

## ST\_LINEGETALLTIMESATPOINT

Returns a timestamp array of all times when the specified `LINESTRING` value intersects the specified `POINT` value. If the line never intersects the point, then the function returns an empty timestamp array.

If the specified inputs are `NULL`, the function returns `NULL`.

This function cannot represent intervals of time where the line was stationary at a point, therefore use this function only for mobile paths. If the path is stationary, use the [Spatiotemporal Operators](#) function as it can represent intervals of time spent intersecting a stationary `LINESTRING`.

### Syntax

```
ST_LINEGETALLTIMESATPOINT(line, timestamp_arr, point)
```

| Argument      | DataType        | Description                                                                                                                                                                                                                                                                                                       |
|---------------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| line          | LINESTRING      | A geospatial <code>LINESTRING</code> to be evaluated for all time values in the <code>timestamp_arr</code> when it intersects the specified <code>point</code> value. If this value is empty, the function returns an empty timestamp array. This value must be the same size as the <code>timestamp_arr</code> . |
| timestamp_arr | TIMESTAMP ARRAY | A <code>TIMESTAMP ARRAY</code> that is paired with the <code>line</code> argument. The <code>timestamp_arr</code> must be the same size as the <code>line</code> value. Behavior is undefined if the timestamp array is not monotonically increasing.                                                             |
| point         | POINT           | A geospatial point to be evaluated for the time values when the point intersects the <code>line</code> value.                                                                                                                                                                                                     |

### Example

```
SELECT ST_LINEGETALLTIMESATPOINT(
  ST_MAKELINE(ST_POINT(0, 0), ST_POINT(4, 0), ST_POINT(0, 0)),
  TIMESTAMP [] (
    '2020-07-04 07:00:00.000000000',
    '2020-07-04 08:00:00.000000000',
    '2020-07-04 09:00:00.000000000'),
  ST_POINT(0, 0));
```

Output: `TIMESTAMP[] ('2020-07-04 07:00:00.000000000', '2020-07-04 09:00:00.000000000')`

### ST\_LINEGETPOINTATTIME

Returns a `POINT` within the bounds of the specified `LINESTRING` that corresponds to the interpolated point at the specified `TIMESTAMP` value.

The specified `TIMESTAMP` value does not need to be an actual value explicitly included in the array. The function interpolates the corresponding point, which means that the function approximates the point coordinates from the surrounding values.

The `LINESTRING` and `TIMESTAMP` array must be the same size. If the timestamp array is not monotonically increasing, the behavior is undefined.

### Syntax

```
ST_LINEGETPOINTATTIME(line, timestamp_arr, time_for_point)
```

| Argument       | DataType        | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| line           | LINESTRING      | A geospatial <code>LINESTRING</code> to be evaluated for an interpolated point, based on <code>the_time_for_point</code> value. This value must be the same size as <code>the_timestamp_arr</code> .                                                                                                                                                                                                                                                                              |
| timestamp_arr  | TIMESTAMP ARRAY | A <code>TIMESTAMP ARRAY</code> that is paired with <code>the_line</code> argument. <code>timestamp_arr</code> must be the same size as <code>the_line</code> value. Behavior is undefined if the timestamp array is not monotonically increasing.                                                                                                                                                                                                                                 |
| time_for_point | TIMESTAMP       | A time value to approximate a <code>POINT</code> value on <code>the_line</code> argument, based on interpolating the values in <code>timestamp_arr</code> . If this time value precedes the first value in <code>timestamp_arr</code> , the function returns the first <code>POINT</code> on <code>line</code> . Similarly, if this time value comes after the last value in <code>timestamp_arr</code> , the function returns the last <code>POINT</code> on <code>line</code> . |

### Example

```
SELECT ST_LINEGETPOINTATTIME(
  ST_MAKELINE(ST_POINT(0, 0), ST_POINT(4, 0)),
  TIMESTAMP[] ('2020-07-04 07:00:00.000000000', '2020-07-04 08:00:00.000000000'),
  TIMESTAMP('2020-07-04 07:30:00.000000000'));
```

Output: POINT(2, 0)

### ST\_LINEGETTIMEATPOINT

Returns the interpolated time of the specified `POINT` on the specified `LINESTRING` that is paired with a `TIMESTAMP ARRAY`.

The `LINESTRING` and `TIMESTAMP ARRAY` must be the same size.

### Syntax

```
ST_LINEGETTIMEATPOINT(line, timestamp_arr, point_for_time)
```

| Argument | DataType   | Description                                                                                                                                                                           |
|----------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| line     | LINESTRING | A geospatial line to be evaluated for an interpolated time value based on <code>the_point_for_time</code> value. This value must be the same size as <code>the_timestamp_arr</code> . |

| Argument       | DataType        | Description                                                                                                                                                                                                   |
|----------------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| timestamp_arr  | TIMESTAMP ARRAY | A timestamp array that is paired with thelineargument.timestamp_arrmust be the same size as thelinevalue.Behavior is undefined if the timestamp array is not monotonically increasing.                        |
| point_for_time | POINT           | APOINTvalue to approximate aTIMESTAMPvalue on thelineargument, based on interpolating the values intimestamp_arr.Iflineintersects thisPOINTvalue multiple times, then the function returns the earliest time. |

### Example

```
SELECT ST_LINEGETTIMEATPOINT(
  ST_MAKELINE(ST_POINT(0, 0), ST_POINT(4, 0)),
  TIMESTAMP[] ('2020-07-04 07:00:00.000000000', '2020-07-04 08:00:00.000000000'),
  ST_POINT(2, 0));
```

Output: TIMESTAMP('2020-07-04 07:30:00.000000000')

## ST\_INTERSECTION

Returns a tuple that represents the intersection of a spatiotemporal LINESTRING with a static geography.

If the specified static geography is a POINT, the returned value is a tuple that contains a paired POINT array and TIMESTAMP array. If the specified static geography is a LINESTRING or POLYGON, then the returned value is a LINESTRING array paired with a two-dimensional TIMESTAMP array.

### Syntax

```
ST_INTERSECTION(line, timestamp_arr, intersection_point)
```

| Argument      | DataType        | Description                                                                                                                                                                          |
|---------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| line          | LINESTRING      | A geospatial line to be evaluated for all intersections with theintersection_point.If this argument is empty, the function returns NULL.                                             |
| timestamp_arr | TIMESTAMP ARRAY | ATIMESTAMP ARRAYthat is paired with thelineargument.timestamp_arrmust be the same size as thelinevalue.Behavior is undefined if the timestamp array is not monotonically increasing. |

| Argument           | Data Type                     | Description                                                                                                                                                                                              |
|--------------------|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| intersection_point | POINT, LINESTRING, or POLYGON | A geographic object to evaluate for all intersections with the <code>line</code> argument. If this value is empty, the function returns an empty tuple. If the value is NULL, the function returns NULL. |

### Example

```
SELECT ST_INTERSECTION(
  ST_LINESTRING('LINESTRING(1 1,2 2,3 3)'),
  TIMESTAMP[] ('2020-07-04 07:00:00.000000000', '2020-07-04 11:00:00.000000000',
    '2020-07-04 12:00:00.000000000'),
  ST_POLYGON('POLYGON((2 2,3 2,3 3,3 2,2 2))');
```

*Output:* TUPLE(LINESTRING[] (ST\_LINESTRING('LINESTRING(2 2)'), ST\_LINESTRING('LINESTRING(3 3)'),  
 ARRAY[ARRAY[TIMESTAMP '2020-07-04 11:00:00.000000000'], ARRAY[TIMESTAMP '2020-07-04 12:00:00.000000000']])

### ST\_SHORTESTLINE

When you specify two LINESTRING-TIMESTAMP ARRAY pairs, this function returns a LINESTRING with two points that represents the minimum distance between points at a concurrent time.

If the two linestrings do not overlap temporally, this function returns NULL.

### Syntax

```
ST_SHORTESTLINE(linestring1, ts_arr1, linestring2, ts_arr2 [, use_spheroid ])
```

| Argument     | Data Type       | Description                                                                                                                      |
|--------------|-----------------|----------------------------------------------------------------------------------------------------------------------------------|
| linestring1  | LINESTRING      | A geospatial object to be evaluated for its minimum cotemporal distance from <code>linestring2</code> .                          |
| ts_arr1      | TIMESTAMP ARRAY | A timestamp array that is paired with <code>linestring1</code> .                                                                 |
| linestring2  | LINESTRING      | A geospatial object to be evaluated for its minimum cotemporal distance from <code>linestring1</code> .                          |
| ts_arr2      | TIMESTAMP ARRAY | A timestamp array that is paired with <code>linestring2</code> .                                                                 |
| use_spheroid | BOOLEAN         | Optional. If you set this argument to TRUE, this function uses a spheroid model instead of a spherical model. Defaults to FALSE. |

### Example

```
SELECT ST_SHORTESTLINE(
  ST_MAKELINE(ST_POINT(0, 0), ST_POINT(4, 0)),
  TIMESTAMP[] ('2020-07-04 07:00:00.000000000', '2020-07-04 08:00:00.000000000'),
  ST_MAKELINE(ST_POINT(3, 0), ST_POINT(4, 8)),
  TIMESTAMP[] ('2020-07-04 07:00:00.000000000', '2020-07-04 07:30:00.000000000'));
```

*Output:* LINESTRING(0.09295649430902601 0, 3.0461764508256914 0.37184471667525754)

To use ST\_SHORTESTLINE as a spatial function to calculate the shortest line between two points, see the [Spatial Operators](#) function.

## Related Links

[Data Types](#)

[Spatiotemporal Measurement](#)

## 10.11 Machine Learning Model Functions

OcientML™ functionality supports these SQL functions for machine learning models.

The scope of a machine learning model is the schema.

## Supported Machine Learning Models

Supported machine learning models are divided into these categories.

### Regression Models

- [Regression Models](#)
- [Regression Models](#)
- [Regression Models](#)
- [Regression Models](#)
- [Regression Models](#)
- [Regression Models](#)

### Classification Models

- [Classification Models](#)
- [Classification Models](#)
- [Classification Models](#)
- [Classification Models](#)
- [Classification Models](#)
- [Classification Models](#)

### Clustering and Dimension Reduction Models

- [Clustering and Dimension Reduction Models](#)
- [Clustering and Dimension Reduction Models](#)
- [Clustering and Dimension Reduction Models](#)
- [Clustering and Dimension Reduction Models](#)

### Other Models

- [Other Models](#)
- [Other Models](#)

## Execute a Query Using a Machine Learning Model

To create a machine learning model and manage the model, see [Machine Learning Models](#) for the corresponding syntaxes. After you have the model, you can execute a query using the model with this syntax.

### Syntax

```

SELECT <model name>(expression, ...) FROM | **Option** | **Notes** |
| ---/---/
| `metrics` | The model supports the `metrics` option for all regression types. If this option is set to `true`
| `threshold` | Most regression model types support the `threshold` option, which enables soft-thresholding.
| `gamma` | Some regression model types support the `gamma` option. The value of this option is the Tikhonov
| `weighted` | Some regression model types also support weighted least squares. The model types do not support

:::info
Model option names are case-sensitive.
:::

#### Simple Linear Regression {.unnumbered}

Model Type: `SIMPLE LINEAR REGRESSION`

Trains a new machine learning model of type `<model_type>` on the result set returned by the SQL SELECT statement.

```sql
CREATE MLMODEL <model_name>
TYPE <model_type> ON(
  <SQL select statement>
)
[options(<option list>)]

```

The result set used as input to the model must have two numeric columns. The first column is the independent variable (referred to as  $x$ ). The second column is the dependent variable (referred to as  $y$ ). The model finds the least squares best fit for  $y = ax + b$ .

The simple linear regression model type also supports the `yIntercept` option, which is the y-intercept of the resulting best-fit line. If you do not specify this option, the model does not force the y-intercept to be any particular value, and the model uses least squares to find the best value instead. If you force the y-intercept to be a particular value, the model uses least squares to find the best fit with that constraint.

## Model Options

Optional

**metrics** — If you set this option to `true`, the model collects quality metrics such as the coefficient of determination ( $r^2$ ) and the root mean squared error (RMSE). This option defaults to `false`.

**yIntercept** — If you set this option, then the option must be a numeric value. The system forces the specific y-intercept (i.e., the model value when  $x$  is zero).

**threshold** — This option enables soft thresholding. If you specify this option, then the option must be a positive numeric value. After the model calculates the coefficients, if any are greater than the threshold value, the threshold value is subtracted from them. If any coefficients are less than the negation of the threshold value, the model adds the threshold value to them. For any coefficients that are between the negative and positive threshold values, the model sets those coefficients to zero.

## Execute the Model

Create a simple linear regression model with y-intercept 10. Collect metrics for the model execution by setting the `metrics` option to `true`.

```

CREATE MLMODEL my_model
TYPE SIMPLE LINEAR REGRESSION ON (
  SELECT

```

```

    x1,
    y
FROM public.my_table
)
options(
  'yIntercept' -> '10',
  'metrics' -> 'true'
);

```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model after training, the model takes a single numeric argument that represents  $x$  and returns  $ax + b$ .

```
SELECT my_model(col1) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.simple_linear_regression_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in System Catalog.

## Multiple Linear Regression

Model Type: MULTIPLE LINEAR REGRESSION

Multiple linear regression means that there is a vector of independent variables and the dependent variable is a scalar-valued function of the vector input, and that it is linear in all vector components.

The model uses the result set as an input. The result set has  $N$  columns that must be numeric. The first  $N - 1$  columns are the independent variables (it can be considered a single independent variable that is a vector). The last column is the dependent variable. The model finds the least squares best fit for  $y = a_1 * x_1 + a_2 * x_2 + \dots + b$ , or in vector notation,  $y = \mathbf{ax} + b$ , where  $\mathbf{a}$  and  $\mathbf{x}$  are vectors and the multiplication is a dot product.

## Model Options

Optional

**metrics** — If you set this option to `true`, the model collects quality metrics such as the coefficient of determination ( $r^2$ ), the adjusted coefficient of determination, and the root mean squared error (RMSE). This option defaults to `false`.

**threshold** — This option enables soft thresholding. If you specify this option, the option must be a positive numeric value. After the model calculates the coefficients, if any coefficients are greater than the threshold value, the model subtracts the threshold value from the coefficients. If any coefficients are less than the negation of the threshold value, the model adds the threshold value to the coefficients. For any coefficients that are between the negative and positive threshold values, the model sets the coefficients to zero.

**weighted** — If you set this option to `true`, the model performs weighted least squares regression, where each sample has a weight or importance associated with it. In this case, there is an extra numeric column after the dependent variable that has the weight for the sample. This option defaults to `false`.

**gamma** — If you set this option, the option must be a matrix. The value represents a Tikhonov gamma matrix used for regularization. For details, see Tikhonov regularization. The model uses this option for ridge regression.

**yIntercept** — If you set this option, then the option must be a numeric value. The system forces the specific y-intercept (i.e., the model value when  $x$  is zero).



## Execute the Model

Create a multiple linear regression model. Collect metrics for the model execution by setting the `metrics` option to `true`.

```
CREATE MLMODEL my_model
TYPE MULTIPLE LINEAR REGRESSION ON (
  SELECT
    x1,
    x2,
    x3,
    y
  FROM public.my_table
)
options(
  'metrics' -> 'true'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model after training, the independent variables are provided to the model function execution and the function returns the estimate of the dependent variable.

```
SELECT my_model(col1, col2, col3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.multiple_linear_regression_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in System Catalog.

## Vector Autoregression

Model Type: VECTOR AUTOREGRESSION

Vector auto-regression is a model that estimates the next value of multiple variables based on some number of lags of all variables, as a group. The model tries to build the following:

Estimate  $\langle x_1(t), x_2(t) \rangle$  based on  $x_1(t-1)$ ,  $x_2(t-1)$ ,  $x_1(t-2)$ , and  $x_2(t-2)$

In this example,  $x_1(t)$  means the value of  $x_1$  at time  $t$ , and  $x_1(t-1)$  means the value of  $x_1$  at time  $t-1$  (typically the previous sample time). The syntax  $\langle x_1(t), x_2(t) \rangle$  demonstrates that the result of the model is a row vector that contains all the predictions of the model and that all predictions rely on all lags of all variables.

When you create a model, the input result set must have one more column than the number of lags. Each column must be a row vector of a size equal to the number of variables. The first column is the un-lagged values, e.g.,  $\{x_1, x_2, x_3\}$ . The second column is the first lag for all variables, e.g.,  $\{x_1\_lag1, x_2\_lag2, x_3\_lag3\}$ .

## Model Options

Required

**numVariables** — Specify this option as a positive integer for the number of variables in the model.

**numLags** — Specify this option as a positive integer for the number of lags in the model.

Optional

**metrics** — If you set this option to `true`, the function collects the metric for the coefficient of determination ( $r^2$ ). This option defaults to `false`.

**threshold** — This option enables soft thresholding. If you specify this option, then the option must be a positive numeric value. After the model calculates the coefficients, if any of them are greater than the threshold value, the threshold value is subtracted from them. If any coefficients are less than the negation of the threshold value, the model adds the threshold value to them. For any coefficients that are between the negative and positive threshold values, the model sets those coefficients to zero.

## Execute the Model

Create a vector autoregression model with three variables and four lags on each one. Collect metrics for the model execution by setting the **metrics** option to **true**.

```
CREATE MLMODEL my_model
TYPE VECTOR AUTOREGRESSION ON (
  SELECT
    {{x1, x2, x3}},
    {{x1_lag1, x2_lag1, x3_lag1}},
    {{x1_lag2, x2_lag2, x3_lag2}},
    {{x1_lag3, x2_lag3, x3_lag3}},
    {{x1_lag4, x2_lag4, x3_lag4}}
  FROM (
    SELECT
      x1, x2, x3,
      LAG(x1, 1) OVER(ORDER BY t) as x1_lag1,
      LAG(x1, 2) OVER(ORDER BY t) as x1_lag2,
      LAG(x1, 3) OVER(ORDER BY t) as x1_lag3,
      LAG(x1, 4) OVER(ORDER BY t) as x1_lag4,
      LAG(x2, 1) OVER(ORDER BY t) as x2_lag1,
      LAG(x2, 2) OVER(ORDER BY t) as x2_lag2,
      LAG(x2, 3) OVER(ORDER BY t) as x2_lag3,
      LAG(x2, 4) OVER(ORDER BY t) as x2_lag4,
      LAG(x3, 1) OVER(ORDER BY t) as x3_lag1,
      LAG(x3, 2) OVER(ORDER BY t) as x3_lag2,
      LAG(x3, 3) OVER(ORDER BY t) as x3_lag3,
      LAG(x3, 4) OVER(ORDER BY t) as x3_lag4
    FROM public.my_table
  )
)
options(
  'metrics' -> 'true',
  'numVariables' -> '3',
  'numLags' -> '4'
);
```

After you create the model, you can see its details by querying the **sys.machine\_learning\_models** and **sys.machine\_learning\_model\_options** system catalog tables.

When you execute the model after training, the number of arguments you specify must be equal to the number of lags. Each of those arguments must be a row vector that contains lags for all model variables. The first argument is the first lag. The second argument is the second lag, and so on. In this example, the unlagged value is the first lag, which means that the model must predict the next value.

```
SELECT my_model(
  {{x1_lag1, x2_lag1, x3_lag1}},
  {{x1_lag2, x2_lag2, x3_lag2}},
  {{x1_lag3, x2_lag3, x3_lag3}},
  {{x1_lag4, x2_lag4, x3_lag4}}) as predicted,
  {{x1, x2, x3}} as actual
```

```

FROM (
  SELECT x1, x2, x3,
    LAG(x1, 1) OVER(ORDER BY t) as x1_lag1,
    LAG(x1, 2) OVER(ORDER BY t) as x1_lag2,
    LAG(x1, 3) OVER(ORDER BY t) as x1_lag3,
    LAG(x1, 4) OVER(ORDER BY t) as x1_lag4,
    LAG(x2, 1) OVER(ORDER BY t) as x2_lag1,
    LAG(x2, 2) OVER(ORDER BY t) as x2_lag2,
    LAG(x2, 3) OVER(ORDER BY t) as x2_lag3,
    LAG(x2, 4) OVER(ORDER BY t) as x2_lag4,
    LAG(x3, 1) OVER(ORDER BY t) as x3_lag1,
    LAG(x3, 2) OVER(ORDER BY t) as x3_lag2,
    LAG(x3, 3) OVER(ORDER BY t) as x3_lag3,
    LAG(x3, 4) OVER(ORDER BY t) as x3_lag4
  FROM my_table
);

```

After you execute a model, you can find the details about the execution results in the `sys.vector_autoregression_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in System Catalog.

## Polynomial Regression

Model Type: POLYNOMIAL REGRESSION

Polynomial regression means that there are one to many independent variables and one dependent variable and that should be modeled in terms of an *n*th-degree polynomial of the independent variables.

The `order` option must be set to a positive integer value that specifies the degree of the polynomial to use. If you specify the option `negativePowers` and set it to `true`, then the model also includes terms with negative exponents, but still with the restriction that the sum of the absolute value of the exponents in a term will be less than or equal to the value specified on the `order` option. Regardless of whether you use the `negativePowers` option, the model computes a coefficient for every possible term that meets this restriction. When you use the `negativePowers` option, the model contains many more terms. For example, a quadratic model over two independent variables has six terms, but when you use the `negativePowers` option, the model has 13 terms.

The result set you specify as input to the model has *N* columns, which must all be numeric. The first *N*-1 columns are the independent variables (it can be considered a single independent variable that is a vector). The last column is the dependent variable. The model finds the least squares best fit of a sum of all possible combinations of terms where the degree is less than or equal to the value of the `order` option. For example, with two independent variables (*x1* and *x2*) and `order` set to 2, the model is  $y = a1*x1^2 + a2*x2^2 + a3*x1*x2 + a4*x1 + a5*x2 + b$ .

## Model Options

Required

**order** — This option is the degree of the polynomial and must be set to a positive integer.

Optional

**metrics** — If you set this option to `true`, the model collects quality metrics such as the coefficient of determination ( $r^2$ ), the adjusted coefficient of determination, and the root mean squared error (RMSE). This option defaults to `false`.

**threshold** — This option enables soft thresholding. If you specify this option, then the option must be a positive numeric value. After the model calculates the coefficients, if any of them are greater than the threshold value, the threshold value is subtracted from them. If any coefficients are less than the negation of the threshold value, the

model adds the threshold value to them. For any coefficients that are between the negative and positive threshold values, the model sets those coefficients to zero.

**weighted** — If you set this option to **true**, the model performs weighted least squares regression, where each sample has an associated weight. When weighted, there is an extra numeric column after the dependent variable that has the weight for the sample. This option defaults to **false**.

**negativePowers** — If you set this option to **true**, the model includes independent variables raised to negative powers. These variables are called Laurent polynomials. The model generates all possible terms such that the sum of the absolute value of the power of each term in each product is less than or equal to the order. For example, with two independent variables and the order set to 2, the model is:  $y = a_1*x_1^2 + a_2*x_1^{-2} + a_3*x_2^2 + a_4*x_2^{-2} + a_5*x_1*x_2 + a_6*x_1^{-1}*x_2 + a_7*x_1*x_2^{-1} + a_8*x_1^{-1}*x_2^{-1} + a_9*x_1 + a_{10}*x_1^{-1} + a_{11}*x_2 + a_{12}*x_2^{-1} + b$ .

**gamma** — If you specify this option, the value must be a matrix. The value represents a Tikhonov gamma matrix that is used for regularization. For details, see Tikhonov regularization.

**yIntercept** — If you set this option, then the option must be a numeric value. The system forces the specific y-intercept (i.e., the model value when x is zero).

## Execute the Model

Create a polynomial regression model of degree 3. Collect metrics for the model execution by setting the **metrics** option to **true**.

```
CREATE MLMODEL my_model
TYPE POLYNOMIAL REGRESSION ON (
  SELECT
    x1,
    x2,
    x3,
    y
  FROM public.my_table
)
options(
  'order' -> '3',
  'metrics' -> 'true'
);
```

After you create the model, you can see its details by querying the **sys.machine\_learning\_models** and **sys.machine\_learning\_model\_options** system catalog tables.

When you execute the model after training, the independent variables are provided to the model function execution, and the function returns the estimate of the dependent variable.

```
SELECT my_model(col1, col2, col3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the **sys.polynomial\_regression\_models** system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in System Catalog.

## Linear Combination Regression

Model Type: **LINEAR COMBINATION REGRESSION**

Linear combination regression is a model built on top of m independent variables and a single dependent variable. But in this case, the function used to perform least-squares regression is a linear combination of functions that you specify. The general form is  $y = c_0 + c_1 * f_1(x_1, x_2, \dots) + f_2(x_1, x_2, \dots) + \dots$ .

The model determines the number of independent variables based on the number of columns in the SQL statement that the model is based on. There is always a column for the dependent variable, and there can be a weight column

(see the **weighted** option). So, the number of independent variables is either one or two less than the number of columns in the result of the input SQL statement. The number of user-specified functions for the model must be specified by defining **function1**, **function2**, and so on (keys in the options dictionary). As long as consecutive function key names exist, the model includes these names. The model always includes a constant term. The value strings for the **functionN** keys must be specified in SQL syntax and should refer to **x1**, **x2**, ... for the model input independent variables.

The result set you specify as input to the model has **N** columns, which must all be numeric. The first **N-1** columns are independent variables (it can be considered a single independent variable that is a vector). The last column is the dependent variable. The model finds the least squares best fit for a model of the form  $y = a_1 * f_1(x_1, x_2, \dots, x_n) + a_2 * f_2(x_1, x_2, \dots, x_n) + \dots + a_n * f_n(x_1, x_2, \dots, x_n)$ , where **f1**, **f2**, ..., **fn** are functions that are provided in a required option.

## Model Options

### Required

**functionN** — You must specify the first function (**f1**) using a key named '**function1**'. Subsequent functions must use keys with names that use subsequent values of **N**. You must specify functions in SQL syntax and should use the variables **x1**, **x2**, ..., **xn** to refer to the 1st, 2nd, and nth independent variables, respectively. For example, '**function1**' -> '**sin(x1 \* x2 + x3)**', '**function2**' -> '**cos(x1 \* x3)**'.

### Optional

**metrics** — If you set this option to **true**, the model collects quality metrics such as the coefficient of determination ( $r^2$ ), the adjusted coefficient of determination, and the root mean squared error (RMSE). This option defaults to **false**.

**threshold** — This option enables soft thresholding. If you specify this option, the option must be a positive numeric value. After the model calculates the coefficients, if any coefficients are greater than the threshold value, the model subtracts the threshold value from the coefficients. If any coefficients are less than the negation of the threshold value, the model adds the threshold value to the coefficients. For any coefficients between the negative and positive threshold values, the model sets the coefficients to zero.

**weighted** — If you set this option to **true**, the model performs weighted least squares regression, where each sample has an associated weight or importance. When weighted, there is an extra numeric column after the dependent variable that represents the weight of the sample. This option defaults to **false**.

**gamma** — If you specify this option, it must be a matrix. The value represents a Tikhonov gamma matrix used for regularization. For details, see Tikhonov regularization.

**yIntercept** — If you set this option, then the option must be a numeric value. The system forces the specific y-intercept (i.e., the model value when **x** is zero).

## Execute the Model

Create a linear combination regression with two functions that exhibit a wave-like pattern.

```
CREATE MLMODEL my_model
TYPE LINEAR COMBINATION REGRESSION
ON (
  SELECT
    x1,
    x2,
    x3,
    y1
  FROM public.my_table
)
options(
  'function1' -> 'sin(x1 * x2 + x3)',
```

```
'function2' -> 'cos(x1 * x3)'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model after training, the independent variables are provided to the model function execution, and the function returns the estimate of the dependent variable.

```
SELECT my_model(col1, col2, col3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.linear_combination_regression_model` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in System Catalog.

## Nonlinear Regression

Model Type: **NONLINEAR REGRESSION**

Nonlinear regression finds best-fit parameters of an arbitrary function using an arbitrary loss function. This model type essentially provides direct access to capabilities that both logistic regression and support vector machines rely on. The first  $N - 1$  columns are numeric independent variables, and the last column is the numeric dependent variable. For faster, lower quality models, reduce the `popSize`, `initialIterations`, and `subsequentIterations` options. Conversely, for slower, higher quality models, increase the values for these same options.

### Model Options

Required

**numParameters** — Specify this option as a positive integer. This value specifies the number of different parameters to optimize, i.e., how many different `aN` variables there are in the user-specified function.

**function** — Specify the name of the function to fit the data in SQL syntax. Use `a1`, `a2`, ... to refer to the parameters for optimization. Use `x1`, `x2`, ... to refer to the input features. The model does not allow some SQL functions. The model allows only scalar expressions that can be represented internally as postfix expressions. Most notably, the model does not allow some functions that are rewritten as CASE statements (like `least()` and `greatest()`). If your function is not allowed, the model displays an error message.

Optional

**metrics** — If you set this option to true, the model calculates the coefficient of determination ( $r^2$ ), the adjusted  $r^2$ , and the root mean squared error (RMSE). However, the model calculates these quality metrics using the least squares loss function, and not the user-specified loss function because these metrics really only make sense for least squares.

**lossFunction** — If you specify this option, this parameter dictates the nonlinear optimizer the loss function uses on a per-sample basis. Then, the actual loss function is the sum of this function applied to all samples. The model should use the variable `y` to refer to the dependent variable in the training data and the variable `f` to refer to the computed estimate for a specified sample. The default is least squares, which could be specified as `(f-y)*(f-y)`.

**popSize** — If you specify this option, the value must be a positive integer. This option sets the population size for the particle swarm optimization (PSO) part of the algorithm. It defaults to 100.

**minInitParamValue** — If you specify this option, the value must be a floating-point number. This option sets the minimum for initial parameter values in the optimization algorithm. It defaults to -10.

**maxInitParamValue** — If you specify this option, the value must be a floating-point number. This option sets the maximum for initial parameter values in the optimization algorithm. It defaults to 10.

**initialIterations** — If you specify this option, the value must be a positive integer. This option sets the number of PSO iterations for the first PSO pass. This option defaults to 500.

**subsequentIterations** — If you specify this option, the value must be a positive integer. This value sets the number of subsequent PSO iterations after the initial pass. This option defaults to 100.

**momentum** — If you specify this option, the value must be a positive floating point number. This parameter controls how much PSO iterations move away from the local best value to explore new territory. This option defaults to 0.1.

**gravity** — If you specify this option, the value must be a positive floating point number. This parameter controls how much PSO iterations are drawn back towards the local best value. This option defaults to 0.01.

**lossFuncNumSamples** — If you specify this option, the value must be a positive integer. This parameter controls how many points the model samples when estimating the loss function. This option defaults to 1000.

**numGAAttempts** — If you specify this option, the value must be a positive integer. This parameter controls how many GA crossover possibilities the model tries. This option defaults to 10 million.

**maxLineSearchIterations** — If you specify this option, the value must be a positive integer. This parameter controls the maximum allowed number of iterations when the model runs the line search part of the algorithm. This option defaults to 200.

**minLineSearchStepSize** — If you specify this option, the value must be a positive floating-point number. This parameter controls the minimum step size that the line search algorithm takes. This option defaults to 1e-5.

**ridgeCoefficient** — If you specify this option, the value must be a double data type. This option is the ridge coefficient for the loss function. The default behavior is the function ignores this option, effectively setting this option to 0.0.

**lassoCoefficient** — If you specify this option, the value must be a double data type. This option is the lasso coefficient for the loss function. The default behavior is the function ignores this option, effectively setting this option to 0.0.

**samplesPerThread** — If you specify this option, the value must be a positive integer. This parameter controls the target number of samples sent to each thread. Each thread independently computes a logistic regression model, and the models are all combined at the end. The option defaults to 1 million.

## Execute the Model

Create a nonlinear regression that fits five parameters **a1**, **a2**, **a3**, **a4**, **a5** with two independent variables **x1**, **x2**.

```
CREATE MLMODEL my_model
TYPE NONLINEAR REGRESSION
ON (
  SELECT
    x1,
    x2,
    y1
  FROM public.my_table
)
options(
  'numParameters' -> '5',
  'function' -> 'a1 * sin(a2 * x1 + a3) + a4 + a5 * x2'
);
```

After you create the model, you can see its details by querying the **sys.machine\_learning\_models** and **sys.machine\_learning\_model\_options** system catalog tables.

When the model executes, pass **N - 1** independent variables, and the model returns the estimate of the dependent variable.

```
SELECT my_model(x1, x2) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the **sys.nonlinear\_regression\_models** system catalog table.



For details, see the description of the associated system catalog tables in the Machine Learning section in the System Catalog.

## Related Links

[Machine Learning in Ocint](#)

[Classification Models](#)

[Machine Learning Models](#)

### 10.11.1 Classification Models

OcientML™ supports classification models that involve understanding and grouping large data sets into preset categories or subpopulations. With the help of pre-classified training data sets, machine learning classification models leverage various algorithms to classify future data sets into respective and relevant categories.

To create the model, use the `CREATE MLMODEL` syntax. For details, see [Machine Learning Model Functions](#).

Model option names are case-sensitive.

## K-Nearest Neighbors Classification

Model Type: `K NEAREST NEIGHBORS`

K-nearest neighbors (KNN) is a classification algorithm, where the first  $N - 1$  inputs are the features, which must be numeric. The last input column is a label, which can be any data type.

There is no training step for KNN. Instead, when you create the model, the model saves a copy of all input data to a table, so that when the model is executed in a later SQL statement, a snapshot of the data the model is supposed to use is available. You can override both the weight function and the distance function.

## Model Options

Required

**k** — This option must be a positive integer that specifies how many closest points to use for classifying a new point.

Optional

**dataReduction** — If you specify this option to `true`, you enable the data reduction feature that finds a smaller subset of rows to build the KNN model over, enough to ensure that the accuracy of the model is still high. By default, the execution stops when the database finds a model that is 90 percent accurate over the training data or when the reduced model increases to 1,000 rows of data. You can override these defaults by setting the **maxReducedRows** or **targetAccuracy** options.

**distance** — If you specify this option, the value must be a function in SQL syntax for calculating the distance between a point used for classification and points in the training data set. This function should use the variables `x1`, `x2`, ... for the 1st, 2nd, ... features in the training data set, and `p1`, `p2`, ... for the features in the point for classification. If you do not specify this option, the option defaults to the Euclidian distance.

**maxReducedRows** — If you specify this option, then you must set the **dataReduction** option to `true`. Specify the maximum number of rows for data reduction as a positive integer. The default value is 1,000 rows of data.

**metrics** — If you set this option to `true`, the model calculates the percentage of samples that are correctly classified by the model and saves this information in a catalog table. The default value is `false`.

**normalize** — If you set this option to `true`, the model normalizes the data before the start of training. Normalization means to shift the mean to 0, and scale the data so the standard deviation is 1.



**targetAccuracy** — If you specify this option, then you must set the **dataReduction** option to **true**. Specify the model accuracy over the training data with a **double** value in the range (0,1]. The default value is 90 percent accuracy, or 0.90.

**weight** — If you specify this option, the value must be a function in SQL syntax for calculating the weight of a neighbor. The function should use the variable *d* for distance. By default, the distance is set to  $1.0/(d+0.1)$ , thus avoiding division by zero on exact inputs and still allowing neighbors to have some influence.

## Execute the Model

Create a KNN model with 8 closest points for classification and distance function `power(x1 - p1, 2) + power(x2 - p2, 2) + power(x3 - p3, 2)`.

```
CREATE MLMODEL my_model
TYPE K NEAREST NEIGHBORS
ON (
  SELECT
    x1,
    x2,
    x3,
    y1
  FROM public.my_table
)
options(
  'k' -> '8',
  'distance' -> 'power(x1 - p1, 2) + power(x2 - p2, 2) + power(x3 - p3, 2)'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model, the model executes with *N* - 1 features as input and returns a label. The model chooses the label from the class with the highest score. The model scores classes by summing the weights from the nearest *k* points in the training data.

```
SELECT my_model(x1, x2, x3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.k_nearest_neighbors_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the System Catalog.

## Naive Bayes Classification

Model Type: **NAIVE BAYES**

Naive Bayes is a classification algorithm. The input is *N* - 1 feature columns, and the last column is a label column. All columns can be any data type. The label column must be discrete. The feature columns can be discrete or continuous. When you use continuous feature columns, you must specify which columns are continuous (see options).

Naive Bayes works by assuming that all features are equally important in the classification and that there is no correlation between features. With those assumptions, the algorithm computes all frequency information and saves it in three tables you create using SQL **SELECT** statements.

## Model Options

Optional

**metrics** — If you set this option to **true**, the model calculates the percentage of samples correctly classified by the model and saves this information in a catalog table. This option defaults to **false**.

**continuousFeatures** — If you specify this option, the value must be a comma-separated list of the feature indexes that are continuous numeric variables. Indexes start at 1.

## Execute the Model

Create a Naive Bayes model with feature indexes 1,3.

```
CREATE MLMODEL my_model
TYPE NAIVE BAYES
ON (
  SELECT
    x1,
    x2,
    x3,
    y1
  FROM public.my_table
)
options(
  'continuousFeatures' -> '1,3'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model, you specify  $N - 1$  feature input arguments and the model returns the most likely class. The returned class is based on computing the class with the highest probability, given prior knowledge of the feature values. In other words, the class  $y$  has the highest value of  $P(y \mid x_1, x_2, \dots, x_n)$ .

```
SELECT my_model(col1, col2, col3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.naive_bayes_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in System Catalog.

## Decision Tree

Model Type: DECISION TREE

The decision tree is a classification model. The first  $N - 1$  input columns are features and can be any data type. All non-numeric features must be discrete and contain no more than the configured `distinctCountLimit` number of unique values. This limit is in place to prevent the internal model representation from growing too large. Numeric features are discrete by default and have the same limitation on the number of unique values, but they can be marked as continuous with the `continuousFeatures` option. For continuous features, the model builds the decision tree by dividing the values into two ranges instead of using discrete, unique values. The last input column is the label and can be any data type. The label must also have no more than the `distinctCountLimit` number of unique values.

When you create the model, you specify all features first, and then specify the label as the last column in the result set.

You can use secondary indexes on discrete feature columns to greatly speed up the training of a decision tree model.

## Model Options

Optional

**continuousFeatures** — If you set this option, the value must be a comma-separated list of the feature indexes that are continuous numeric variables. Indexes start with 1. In the default state, the model considers no features as continuous.

**distinctCountLimit** — If you set this option, the value must be a positive integer. This value sets the limit for how many distinct values a non-continuous feature and the label can contain. This option defaults to 256.

**doPrune** — If you set this option to **true**, the model uses Pessimistic Error Pruning (PEP) to prune the tree after training. This option defaults to **false**.

**featureArray** — If you set this option to **true**, all features must be present in the one array column and all arrays in that column have the same length. This option defaults to **false**.

**featureArrayElements** — If you set this option, the value must be a comma-separated list of the feature indexes. If you set the **featureArray** option to **true**, this list determines the elements of the arrays to use for training. By default, the model uses all elements.

**maxCellsToFetch** — If you set this option, the value must be a positive integer. This value determines the memory threshold to switch from training with system memory to training with SQL queries in the database. In-memory training is generally faster, but is limited by the available SQL Node memory. If the size of a training data subset exceeds this value, then the system performs training operations using SQL queries. This value defaults to 33,554,432 (calculated as  $32 * 1024 * 1024$ ).

**maxDepth** — If you set this option, the value must be a positive integer. This value sets the maximum allowable depth of the decision tree (the maximum number of features to split on). The default is unspecified, which means there is no maximum depth.

**maxRows** — If you set this option, the value must be a positive integer. The decision tree trains using a subset of data with no more than the specified number of rows. If you set this option, ensure that the model does not select different subsets of data in different parts of the training by not setting the **noSnapshot** option.

**maxThreads** — If you set this option, the value must be a positive integer. This value indicates the maximum number of parallel threads to use while the model trains. The default value is 1.

**metrics** — If you set this option to **true**, the model also calculates the percentage of samples correctly classified by the model and saves this information in a catalog table. This option defaults to **false**.

**noSnapshot** — If you set this option to **true**, the database does not create an intermediate table that stores the result of the specified SQL statement, which the model uses for training. This option defaults to **false**. In this case, the database creates and uses the intermediate table. Setting this option to **true** is useful when the training set is fixed. If the training set is a table with modifications, set this option to **false** as the decision tree trainer uses different data sets in different parts of the tree. Likewise, if the training set consists of a query that returns 100 rows, then set this option to **false** because there is no guarantee that running that query twice generates the same 100 rows each time.

**numSplits** — If you set this option, the value must be an integer greater than 1. This value sets the maximum number of branches a continuous feature can have. The default value is 32.

**ROCNumSamples** — If you set the option, you must also set the **metrics** option. This positive integer indicates the number of samples for the model to use for the area under the ROC curve. The default value is 10.

**skipLimitCheck** — If you set this option to **true**, the model skips cardinality checks that throw errors when columns have too many values. The limit that this option checks is the same one that is specified by the **distinctCountLimit** option. This option defaults to **false**.

**weighted** — If you set this option, the model considers weights for labels. If you set this option value to **true**, you must specify an additional column as a **double** in the training data for label weights. Rows with the same labels must have the same weights. If you set this value to **auto**, the model calculates weights automatically by weighing each label according to the ratio of the count of the most frequent label to the count of the specified label. As a result, the most frequent label has a weight of 1.0, and the other label weights are higher. This option defaults to **false**, which means all labels have equal weight.

## Execute the Model

Create a decision tree model with feature indexes 1,3.

```
CREATE MLMODEL my_model
TYPE DECISION TREE ON (
  SELECT
    c1,
    c2,
    c3,
    y1
  FROM public.my_table
)
options(
  'continuousFeatures' -> '1,3'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model, you must specify the  $N - 1$  features as parameters. The model returns the expected label.

```
SELECT my_model(col1, col2, col3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.decision_tree_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in System Catalog.

## Random Forest

Model Type: RANDOM FOREST

Random forest is a classification model consisting of multiple decision trees. The result consists of the most common label among the tree results and an array of pairs of labels and their frequencies, sorted in descending order by frequency.

The model breaks ties in various ways depending on the type of label. For strings, the model uses lexicographic order (reverse of the usual alphabetic order), so `C` comes before `A`, for example. For Booleans, the model chooses `true` before `false`. For numeric types, the model chooses the largest number.

When you call the model to make classification predictions, you can optionally use soft voting by adding an extra Boolean argument to the statement, e.g., `house(x, y, true)`. This extra argument must be a Boolean literal, either `true` (soft voting) or `false` (hard voting). If you do not specify this value, the default value is `false` (hard voting).

In soft voting, each decision tree reports a list of possible results and their confidence factors. The random forest model adds the confidence factors from the decision trees and normalizes them to add up to 1.0. The model sorts the results by descending confidence, then by descending values.

In hard voting, the model does not account for confidence factors. Each decision tree makes its own class prediction as a vote. The model selects the prediction with the most votes from the trees.

## Model Options

Required

**numChildren** — Number of child decision trees.

Optional

**continuousFeatures** — If you set this option, the value must be a comma-separated list of the feature indexes that are continuous numeric variables. Indexes start with 1.

**distinctCountLimit** — If you set this option, the value must be a positive integer. This value limits how many distinct values a non-continuous feature and the label can contain. The default value is 256.

**featureArray** — If you set this option to **true**, all features must be present in one array column where all arrays are the same length. The default value is **false**.

**fractionSelected** — The proportion of rows the model uses to train each decision tree. The value is a **double** that must be in the interval (0, 1]. You cannot set this option if you also set the **rowsPerChild** option to a positive value. The default behavior is that the model uses all available rows.

**inputsPerChild** — Number of features used to create each child decision tree. The default value is the number of features you specify for the forest divided by 3 and rounded up.

**maxDepth** — If you set this option, the value must be a positive integer. This value sets the maximum allowable depth of the decision tree. The default value is 3.

**maxThreads** — The maximum number of parallel threads to use while the model trains decision trees. This value must be a positive integer. The default value is 16.

**metrics** — If you set this option to **true**, the model also calculates the percentage of samples that are correctly classified by the model for the random forest and saves this information in a catalog table. This option is always set to **false** for the child trees.

**noSnapshot** — If you set this option to **true**, the data source must not change. In this case, the database does not create an intermediate table that stores the result of the specified SQL statement, which the model uses for training a random forest. Child decision trees always have this option set to **true**, so the database does not create a separate intermediate table for each decision tree. The default value is **false**. Setting this option to **true** is useful when the training set is fixed. If the training set is a table with modifications, set this option to **false** as the decision tree trainer uses different data sets in different parts of the tree. Likewise, if the training set consists of a query that returns 100 rows, then set this option to **false** because there is no guarantee that running that query twice generates the same 100 rows each time.

**requiredFeatures** — A comma-separated list of integers as strings representing specific features where the first feature has the value 1. The model uses these features in every decision tree in the forest. The default behavior is that the decision tree in the forest can train on any feature that is in the list.

**rowsPerChild** — If you set this option to a positive integer, the number represents the number of rows (from a random sample) to use for each decision tree. If you set this option to 0, each child uses all available rows. The default value is 0. You cannot set this option to a positive value if you also set the **fractionSelected** option.

**skipLimitCheck** — If you set this option to **true**, the model skips cardinality checks that throw errors when columns have too many values. The limit that this option checks is the same one that is specified by the **distinctCountLimit** option. This option defaults to **false**.

## Execute the Model

Create a random forest model with four child decision trees and two features randomly chosen by the model for each tree. The **training\_table** table contains the training data set. Collect metrics for the model execution by setting the **metrics** option to **true**.

```
CREATE MLMODEL test_model
TYPE RANDOM FOREST
ON (
  SELECT *
  FROM public.training_table
)
options(
  'numChildren' -> '4',
  'inputsPerChild' -> '2'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models`, `sys.machine_learning_model_options`, and `sys.random_forest_models` system catalog tables.

Execute this model using three columns `a`, `b`, and `c`, from `large_table` table that contains the whole data set.

```
SELECT test_model(a, b, c) FROM large_table;
```

Optionally, you can execute the model using soft voting by including an extra Boolean argument.

```
SELECT test_model(a, b, c, true) FROM large_table;
```

The output of the model is a tuple containing the most common label as the first element. The second element is an array of tuples, where the first element is the label and the second is the frequency. The model sorts the array by frequency from highest to lowest, where the sum of frequencies is 1. When two labels have the same frequency, the model breaks the tie according to lexicographic order.

To retrieve the most common label, use `[]` with an index of 1, `[1]`, for example, `test_model(a,b,c)[1]`.

After you execute a model, you can find the results in the output of the model function execution.

For details, see the description of the associated system catalog tables in the Machine Learning section in System Catalog.

## Logistic Regression

Model Type: LOGISTIC REGRESSION

This model fits a logistic curve to the data across any number of classes greater than one.

The first  $N - 1$  inputs are features and must be numeric. Features can be one-hot encoded. The last input column is the class or label. You must have at least one non-NULL label in the result set for model creation. The model best fits the logistic curve using a negative log likelihood loss function. The model uses an algorithm that is a combination of particle swarm optimization, line search, and genetic algorithms to find the best-fit parameters.

For faster, lower-quality models, try reducing the `popSize`, `initialIterations`, and `subsequentIterations` options. Conversely, for slower, higher-quality models, try increasing the values for these same options.

## Model Options

Optional

**metrics** — If you set this option to `true`, the model calculates the percentage of samples that are correctly classified by the model and saves this information in the `sys.logistic_regression_models` system catalog table. This option defaults to `false`.

**popSize** — If you specify this option, the value must be a positive integer. This value sets the population size for the particle swarm optimization (PSO) part of the algorithm. This option defaults to 100.

**minInitParamValue** — If you specify this option, the value must be a floating point number. This value sets the minimum for initial parameter values in the optimization algorithm. This option defaults to -10.

**maxInitParamValue** — If you specify this option, the value must be a floating point number. This value sets the maximum for initial parameter values in the optimization algorithm. This option defaults to 10.

**initialIterations** — If you specify this option, the value must be a positive integer. This value sets the number of PSO iterations for the first PSO pass. This option defaults to 500.

**subsequentIterations** — If you specify this option, the value must be a positive integer. This value sets the number of PSO iterations for subsequent iterations of the PSO algorithm. This option defaults to 100.

**momentum** — If you specify this option, the value must be a positive floating point. This parameter controls how much PSO iterations move away from the local best value to explore new territory. This option defaults to 0.1.

**gravity** — If you specify this option, the value must be a positive floating point. This parameter controls how much PSO iterations are drawn back towards the local best value. This option defaults to 0.01.

**lossFuncNumSamples** — If you specify this option, the value must be a positive integer. This parameter controls how many points are sampled when estimating the loss function. This option defaults to 1000.

**numGAAttempts** — If you specify this option, the value must be a positive integer. This parameter controls how many GA crossover possibilities the model tries. This option defaults to 10 million.

**maxLineSearchIterations** — If you specify this option, the value must be a positive integer. This parameter controls the maximum allowed number of iterations when the model runs the line search part of the algorithm. This option defaults to 200.

**minLineSearchStepSize** — If you specify this option, the value must be a positive floating point. This parameter controls the minimum step size the line search algorithm ever takes. This option defaults to 1e-5.

**samplesPerThread** — If you specify this option, the value must be a positive integer. This parameter controls the target number of samples sent to each thread. Each thread independently computes a logistic regression model, and the threads are all combined at the end. This option defaults to 1 million.

## Execute the Model

Create a logistic regression model. Collect metrics for the model execution by setting the **metrics** option to **true**.

```
CREATE MLMODEL my_model
TYPE LOGISTIC REGRESSION
ON (
  SELECT
    x1,
    x2,
    x3,
    y1
  FROM public.my_table
)
options(
  'metrics' -> 'true'
);
```

After you create the model, you can see its details by querying the **sys.machine\_learning\_models** and **sys.machine\_learning\_model\_options** system catalog tables.

When you execute this model after training, you must specify the features as input and the label as the output. The label can be any data type.

```
SELECT my_model(col1, col2, col3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the **sys.logistic\_regression\_models** system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the System Catalog.

## Support Vector Machine

Model Type: SUPPORT VECTOR MACHINE

Support Vector Machine (SVM) essentially finds a hypersurface (the hypersurface is a curve in 2-dimensional space) that correctly splits the data into any number of classes greater than one and maximizes the margin around the hypersurface. By default, SVM finds a hyperplane to split the data (the hyperplane is a straight line in 2-dimensional space). SVM uses a hinge loss function to balance the two objectives of finding a hyperplane with a wide margin while minimizing the number of incorrectly classified points.

The first  $N - 1$  input columns are the features and must be numeric. The last column is the label and can be any arbitrary type. You must have at least one non-NULL label in the result set for model creation.

For faster, lower-quality models, reduce the `popSize`, `initialIterations`, and `subsequentIterations` options. Conversely, for slower, higher-quality models, increase the values for these same options.

## Model Options

### Optional

**metrics** — If you set this option to true, the model also calculates the percentage of samples that are correctly classified by the model and saves this information in a catalog table. This option defaults to false.

**regularizationCoefficient** — If you specify this option, the value must be a valid floating point number. This option is used to control the balance of finding a wide margin and minimizing incorrectly classified points in the loss function. When this value is larger (and positive) it makes having a wide margin around the hypersurface more important relative to the incorrectly classified points. Because of how Ocint® implements SVM, the values for this parameter are likely different than values used in other common SVM implementations. This option defaults to 1.0 / 1000000.0.

**functionN** — By default, SVM uses a linear kernel. If you use a different kernel, you must provide a list of functions that are summed together, just like with linear combination regression. You must specify the first function using a key named 'function1'. Subsequent functions must use keys with names that use subsequent values of N. You must specify functions in SQL syntax and use the variables x1, x2, ..., xn to refer to the 1st, 2nd, and nth independent variables, respectively. You can specify the default linear kernel as: 'function1' → 'x1', 'function2' → 'x2', and so on. The model always adds a constant term equivalent to 'functionN' → '1.0' that you do not need to specify explicitly.

**popSize** — If you set this option, the value must be a positive integer. This value sets the population size for the particle swarm optimization (PSO) part of the algorithm. This option defaults to 100.

**minInitParamValue** — If you set this option, the value must be a floating point number. Sets the minimum for initial parameter values in the optimization algorithm. This option defaults to -10.

**maxInitParamValue** — If you set this option, the value must be a floating point number. Sets the maximum for initial parameter values in the optimization algorithm. This option defaults to 10.

**initialIterations** — If you set this option, the value must be a positive integer. Sets the number of PSO iterations for the first PSO pass. This option defaults to 500.

**subsequentIterations** — If you set this option, the value must be a positive integer. Sets the number of PSO iterations for subsequent PSO iterations after the initial pass. This option defaults to 100.

**momentum** — If you set this option, the value must be a positive floating point number. This parameter controls how much PSO iterations move away from the local best value to explore new territory. This option defaults to 0.1.

**gravity** — If you set this option, the value must be a positive floating point number. This parameter controls how much PSO iterations are drawn back towards the local best value. This option defaults to 0.01.

**lossFuncNumSamples** — If you set this option, the value must be a positive integer. This parameter controls how many points the model samples when estimating the loss function. This option defaults to 1000.

**numGAAttempts** — If you set this option, the value must be a positive integer. This parameter controls how many GA crossover possibilities the model tries. This option defaults to 10 million.

**maxLineSearchIterations** — If you set this option, the value must be a positive integer. This parameter controls the maximum allowed number of iterations when running the line search part of the algorithm. This option defaults to 200.

**minLineSearchStepSize** — If you set this option, the value must be a positive floating point number. This parameter controls the minimum step size the line search algorithm ever takes. This option defaults to 1e-5.

**samplesPerThread** — If you set this option, the value must be a positive integer number. This parameter controls the target number of samples the model sends to each thread. Each thread independently computes a logistic regression model, and the models are all combined at the end. This option defaults to 1 million.



## Execute the Model

Create a support vector machine model.

```
CREATE MLMODEL my_model
TYPE SUPPORT VECTOR MACHINE ON (
  SELECT
    c1,
    c2,
    c3,
    y1
  FROM public.my_table
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model, the  $N - 1$  features must be passed as parameters. The model returns the expected label.

```
SELECT my_model(col1, col2, col3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.support_vector_machine_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the System Catalog.

## Related Links

[Clustering and Dimension Reduction Models](#)

[Other Models](#)

[Machine Learning Models](#)

### 10.11.2 Clustering and Dimension Reduction Models

OcientML™ supports these machine learning models for clustering and dimension reduction.

To create the model, use the `CREATE MLMODEL` syntax. For details, see [Machine Learning Model Functions](#).

Model option names are case-sensitive.

## Principal Component Analysis

Model Type: `PRINCIPAL COMPONENT ANALYSIS`

Principal Component Analysis (PCA) is typically not used as a model on its own. PCA is most commonly used on the inputs to other models. PCA serves two purposes.

1. PCA normalizes all numeric feature data. Some types of models are sensitive to the scale of numeric features, and when different features have different scales, the results end up skewed. PCA normalizes all features to the same scale.
2. PCA is used for dimensionality reduction. PCA computes linear combinations of the original features to put the most signal into a smaller number of new features.

The input result set when creating a PCA model is  $N$  numeric columns, which are all features. There is no label or dependent variable. After you create a PCA model, the `sys.principal_component_analysis_models` system

catalog table contains information on the percentage of the signal in each PCA feature. You can use this information to figure out how many of the output features to keep.

Also, after you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

## Principal Component Analysis and Logistic Regression

You can also use PCA models as inputs to other models. For example, if you have three features and you want to use PCA to reduce the number to two features, you can execute the following SQL statements.

```
CREATE MLMODEL reduceTo2
TYPE PRINCIPAL COMPONENT ANALYSIS ON (
  SELECT
    c1,
    c2,
    c3
  FROM public.my_table
);
```

You can use this model as input for another model, for example, logistic regression.

```
CREATE MLMODEL binaryClass
TYPE LOGISTIC REGRESSION ON (
  SELECT
    reduceTo2(c1, c2, c3, 1),
    reduceTo2(c1, c2, c3, 2),
    label
  FROM ...
);
```

To correctly use this model later, you must pass the original features through the PCA model when you execute the logistic regression model.

```
SELECT
  binaryClass(
    reduceTo2(x1, x2, x3, 1),
    reduceTo2(x1, x2, x3, 2)
  )
FROM ... ;
```

Similarly, to create a PCA analysis over four variables, execute this SQL statement:

```
CREATE MLMODEL my_model
TYPE PRINCIPAL COMPONENT ANALYSIS ON (
  SELECT
    c1,
    c2,
    c3,
    c4
  FROM public.my_table
);
```

## Model Options

### K-Means Clustering

Model Type: KMEANS

K-means is an unsupervised clustering algorithm. All of the columns in the input result set are features, and there is no label. All the input columns must be numeric. The algorithm finds **k** points such that all points are classified by the closest **k** points. Distance calculations are Euclidean by default.

## Model Options

Required

**k** — This option must be a positive integer. The option specifies the algorithm for how many clusters to make.

Optional

**epsilon** — If you specify this option, the value must be a valid positive floating point value. When the maximum distance that a centroid moves from one iteration of the algorithm to the next is less than this value, the algorithm terminates. This parameter defaults to 1e-8.

**l1norm** — If you set this option to **true**, the model uses the L1 distance instead of the Euclidian distance. This option defaults to **false**.

**normalize** — If you set this option to **true**, the model normalizes the data before the start of training. Normalization means to shift the mean to 0, and scale the data so the standard deviation is 1.

## Execute the Model

Create a K-means model with 8 clusters.

```
CREATE MLMODEL my_model
TYPE KMEANS
ON (
  SELECT
    x1,
    x2,
    x3,
    x4
  FROM public.my_table
)
options(
  'k' -> '8'
);
```

Also, after you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

Because there are no labels for clusters, when you execute this function after training with the same number (and same order) of features as input, the result is an integer that specifies the cluster to which the point belongs.

```
SELECT my_model(x1, x2, x3, x4) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.k_means_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the System Catalog.

## Gaussian Mixture

Model Type: GAUSSIAN MIXTURE MODEL

Gaussian Mixture Models (GMMs) find the mixture of **N** Gaussian distributions that best fit the data. The most common use is unsupervised clustering, but there are some key differences compared to K-means clustering:

- GMMs can handle clusters that are not circular, i.e., clusters in the shape of ovals or higher-dimensional analogs with variances in different directions.

- Clusters can have an arbitrary rotation, i.e., they can have covariance.
- GMMs can handle instances when clusters are not as equally likely. If a point is right between two clusters, it is more likely to be more common in the training data.
- GMMs can show the probability of a new point belonging to each cluster instead of providing only a single cluster value.

GMMs generally handle more complex data than K-means models, but they require more training and processing time during execution.

## Model Options

Required

**numDistributions** — Must be a positive integer. This value specifies the number of clusters of Gaussian distributions for the model to make.

Optional

**epsilon** — If you specify this option, the value must be a valid positive floating point value. When the maximum distance that the entire best model moves in its n-dimensional space is less than this value, the algorithm terminates. This parameter defaults to 1e-8 (0.00000001 as a decimal number).

## Execute the Model

Create a GMM with three clusters of Gaussian distributions.

```
CREATE MLMODEL gmm
TYPE GAUSSIAN MIXTURE MODEL
ON (
  SELECT
    x,
    y,
    z
  FROM my_table
)
options(
  'numDistributions' -> '3'
);
```

Also, after you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute this function after training with the same number (and same order) of features as input, the result is an integer that specifies the probable cluster to which the point belongs.

```
SELECT gmm(x,y,z) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.gaussian_mixture_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the System Catalog.

## Linear Discriminant Analysis

Model Type: LINEAR DISCRIMINANT ANALYSIS

Linear Discriminant Analysis (LDA) is a dimension-reduction technique similar to PCA. While PCA is unsupervised, LDA can classify and use class labels to find linear combinations of features that best separate the classes.

The input result set when you create an LDA model is N numeric columns that are all features. The last column is a class label and can be any data type. After you create an LDA model, the `sys.linear_discriminant_analysis_models`

system catalog table contains information on the percentage of the signal that is in each LDA feature. You can use this information to figure out how many of the output features to keep.

Also, after you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

## Linear Discriminant Analysis and Logistic Regression

You can use LDA models as inputs to other models. For example, to reduce the number of features from three to two using LDA, you can execute these SQL statements.

```
CREATE MLMODEL reduceTo2
TYPE LINEAR DISCRIMINANT ANALYSIS ON (
  SELECT
    c1,
    c2,
    c3,
    label
  FROM public.my_table
);
```

You can use this model as input for another model, for example, logistic regression.

```
CREATE MLMODEL binaryClass
TYPE LOGISTIC REGRESSION ON (
  SELECT
    reduceTo2(c1, c2, c3, 1),
    reduceTo2(c1, c2, c3, 2),
    label
  FROM ...
);
```

To correctly use this model later, you must pass the original features through the LDA model when you execute the logistic regression model.

```
SELECT
  binaryClass(
    reduceTo2(x1, x2, x3, 1),
    reduceTo2(x1, x2, x3, 2)
  )
FROM ... ;
```

Similarly, to create an LDA model over four variables, execute this SQL statement.

```
CREATE MLMODEL my_model
TYPE LINEAR DISCRIMINANT ANALYSIS ON (
  SELECT
    c1,
    c2,
    c3,
    c4,
    label
  FROM public.my_table
);
```

## Model Options

### Execute the Model

When you execute the trained LDA model, you must provide the same original input features in the same order, followed by a positive integer argument that specifies the LDA component to return. The LDA component index starts at 1.

```
SELECT
  my_model(col1, col2, col3, col4, 2) as component2,
  my_model(col1, col2, col3, col4, 3) as component3,
FROM public.my_table;
```

## Related Links

[Classification Models](#)

[Other Models](#)

[Machine Learning Models](#)

### 10.11.3 Other Models

OcientML™ supports other machine learning models for advanced analysis that include association rules and feedforward neural network.

To create the model, use the `CREATE MLMODEL` syntax. For details, see [Machine Learning Model Functions](#).

Model option names are case-sensitive.

## Association Rules

Model Type: ASSOCIATION RULES

The association rules model trains itself over rows of arrays to suggest other associated values. The model returns values commonly associated in a set that are absent in the provided value set. In practical use, this model is often used to help in retail transactions by suggesting related items or services for purchase based on similar past transactions.

## Model Options

### Execute the Model

Create an association rules model. When you create the model, there should be a single input column, an array of any inner type.

```
CREATE MLMODEL my_model
TYPE ASSOCIATION RULES ON (
  SELECT
    array_agg(product)
  FROM public.my_table
  GROUP BY customer_id
);
```

Also, after you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model, you must provide an array with the same inner type as the input column. The array represents the current state of the transaction.

You must also provide a second argument that is a positive integer that represents the association ranking of the item to return. The value of 1 indicates to add the first most-associated item to the transaction.

The third and fourth arguments are optional constant Boolean values.

If the third argument is set to `true`, the model adds the most associated item to the transaction based on other transactions that have the same items in the current transaction. Otherwise, only some of the items must be the same. The third argument defaults to `true`.

If the fourth argument is set to `true`, the model counts duplicate items in a transaction only one time. Otherwise, the model counts duplicate items multiple times. To specify the fourth argument, you must specify the third argument. The default value for the fourth argument is `true`, which means that duplicate items are counted one time.

```
SELECT my_model(array['sausage', 'mushrooms'], 1);
SELECT my_model(array['sausage', 'mushrooms'], 1, true, true);
```

The output from the model is a tuple ( `v`, `p`, `c` ). The definitions of these tuple values are:

`v` — The associated item determined by the model based on your input arguments.

`p` — The proportion of arrays in the training data that contain the items of the input array and the associated item `v`, as compared to arrays in the training data that have the input items but not necessarily the associated item `v`.

`c` — The count of occurrences of the associated item `v` in the training data arrays based on your input arguments.

After you execute a model, you can find the details about the execution results in the `sys.association_rules_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the System Catalog.

## Feedforward Neural Network

Model Type: `FEEDFORWARD NETWORK`

The feedforward network model is a neural network model where data moves from the inputs through hidden layers to the outputs. Feedforward neural networks are fully connected (sometimes named multilayer perceptrons). The number of inputs is determined by the number of columns in the input result set. Each input must be numeric. The last column in the input result set is the target variable. For models with one output, the column is also numeric. For models with multiple outputs, the result must be a 1xN matrix (a row vector).

Common uses of multiple output models are:

- Multi-class classification — Multiple outputs are one-hot encoded values that represent the class of the record. The model uses results with `argmax` to select the highest probability class.
- Probability modeling — Multiple output values represent probabilities between 0 and 1 that sum to 1.
- Multiple numeric prediction — Multiple output values represent different numeric values to predict against.

For faster, lower-quality models, reduce the `popSize`, `initialIterations`, and `subsequentIterations` options. Conversely, for slower, higher-quality models, increase the values for these same options.

## Model Options

Required

**hiddenLayers** — You must set this option to a positive integer that specifies how many hidden layers to use.

**hiddenLayerSize** — You must set this option to a positive integer that specifies the number of nodes in each hidden layer.

**outputs** — You must set this option to a positive integer that specifies the number of outputs.

**lossFunction** — This option specifies the loss function that all hidden layer nodes and all output layer nodes use. This function can be one of several predefined loss functions or a user-defined loss function.

The predefined loss functions are `squared_error` (regression), `vector_squared_error` (vector-valued regression), `log_loss` (binary classification with target values of 0 and 1), `logits_loss` (binary classification with target values of 0 and 1), `hinge_loss` (binary classification with target values of -1 and 1), and `cross_entropy_loss` (multi-class

classification). If the value for this required parameter is none of these strings, the model assumes a user-defined loss function. The user-defined loss function specifies the per-sample loss. Then, the actual loss function is the sum of this function applied to all samples. The model should use the variable **y** to refer to the dependent variable in the training data, and the model should use the variable **f** to refer to the computed estimate for a given sample.

#### Optional

**activationFunction** — If you set this option, the values are **linear**, **relu** (rectified linear unit), **leakyrelu** (leaky rectified linear unit), **tanh** (hyperbolic tangent function), or **sigmoid** (fast sigmoid approximation). This option defaults to **relu**. This option affects all layers except the output layer.

**outputActivationFunction** — If you set this option, the values are **linear**, **relu** (rectified linear unit), **leakyrelu** (leaky rectified linear unit), **tanh** (hyperbolic tangent function), or **sigmoid** (fast sigmoid approximation). Different activation functions have different output ranges. The chosen activation function should match the dependent variable of your data. For example, if the dependent variable can be anything, then choose the **linear** value. If the dependent variable is always positive, then choose the **relu** value. If your outputs range from -1 to 1 or you perform hinge loss classification, **tanh** is a good option because the hyperbolic tangent function has the same range. But, if your outputs range from 0 to 1 or you perform log loss classification, **sigmoid** is a better choice for the same reason. This option defaults to **linear**. The option only sets the activation function for the output layer.

**metrics** — If you set this option to **true**, the model calculates the average value of the loss function.

**useSoftMax** — If you set this option to **true**, the model applies a softmax function to the output of the output layer before computing the loss function. This option defaults to **true** if the **lossFunction** is set to **cross\_entropy\_loss** and **false** otherwise.

**popSize** — If you set this option, the value must be a positive integer. Sets the population size for the particle swarm optimization (PSO) part of the algorithm. This option defaults to 100.

**minInitParamValue** — If you set this option, the value must be a floating-point number. Sets the minimum for initial parameter values in the optimization algorithm. This option defaults to -1.

**maxInitParamValue** — If you set this option, the value must be a floating-point number. Sets the maximum for initial parameter values in the optimization algorithm. This option defaults to 1.

**initialIterations** — If you set this option, the value must be a positive integer. Sets the number of PSO iterations for the first PSO pass. This option defaults to 50.

**subsequentIterations** — If you set this option, the value must be a positive integer. Sets the number of PSO iterations for subsequent PSO iterations after the initial pass. This option defaults to 10.

**momentum** — If you set this option, the value must be a positive floating-point number. This parameter controls how much PSO iterations move away from the local best value to explore new territory. This option defaults to 0.1.

**gravity** — If you set this option, the value must be a positive floating-point number. This parameter controls how much PSO iterations are drawn back towards the local best value. This option defaults to 0.01.

**lossFuncNumSamples** — If you set this option, the value must be a positive integer. This parameter controls how many points the model samples when estimating the loss function. This option defaults to 1000.

**numGAAttempts** — If you set this option, the value must be a positive integer. This parameter controls how many GA crossover possibilities the model tries. This option defaults to 10,000.

**maxLineSearchIterations** — If you set this option, the value must be a positive integer. This parameter controls the maximum allowed number of iterations when running the line search part of the algorithm. This option defaults to 20.

**minLineSearchStepSize** — If you set this option, the value must be a positive floating-point number. This parameter controls the minimum step size of the line search algorithm. This option defaults to 0.01.

**samplesPerThread** — If you set this option, the value must be a positive integer number. This parameter controls the target number of samples the model sends to each thread. Each thread independently computes a logistic regression model, and the models are all combined at the end. This option defaults to 1 million.



## Execute the Model

Create a neural network to perform multi-class classification for three possible classes. `y1`, `y2`, `y3` are one-hot encoded outputs. If the values are 1 and the rest are 0, the value 1 denotes the class that the training data belongs to from the three classes.

```
CREATE MLMODEL my_model
TYPE FEEDFORWARD NETWORK
ON (
  SELECT
    x1,
    x2,
    {{y1,y2,y3}}
  FROM public.my_table
)
options(
  'hiddenLayers' -> '1',
  'hiddenLayerSize' -> '8',
  'outputs' -> '3',
  'lossFunction' -> 'cross_entropy_loss',
  'activationFunction' -> 'relu',
  'useSoftmax' -> 'true'
);
```

Also, after you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model later, pass `N - 1` input variables and the model returns the estimate of the target variable. In the case of multiple outputs, the result is a `1xN` matrix (a row vector). If the model uses multiple outputs to perform multi-class classification, use `argmax` to get the integer that represents the class.

```
SELECT argmax(my_model(x1, x2)) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.feedforward_network_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the System Catalog.

## Related Links

[Machine Learning Model Functions](#)

[Machine Learning Models](#)

## 10.12 Math Functions and Operators

### Math Operators

Ocient® supports addition, subtraction, multiplication, and division between any two numeric types. Division involving two integral types always results in an integral type.

### Decimal Precision

These rules determine the precision `p` and scale `s` for the result of a specified expression that involves decimals `d1` and `d2`, with precisions `p1` and `p2`, and scales `s1` and `s2`:

| Operation             | Precision                               | Scale                  |
|-----------------------|-----------------------------------------|------------------------|
| addition, subtraction | $\max(s1, s2) + \max(p1-s1, p2-s2) + 1$ | $\max(s1, s2)$         |
| multiplication        | $p1 + p2$                               | $s1 + s2$              |
| division              | <i>See note.</i>                        | $\max(6, s1 + p2 + 1)$ |

For division, the resulting precision is the minimum necessary to hold the whole number portion of the result. If that portion does not exist because the absolute value of the result is less than one, the resulting precision is one greater than the scale. For other operations, the result might still have a lower precision (but the same scale) than specified by these rules when the result would otherwise have a leading insignificant zero. If the resulting precision exceeds 31, the database decreases both precision and scale until the precision is 31, except in division, where the minimal scale is always six. If the resulting precision still exceeds 31, the database truncates it to 31. The database returns an overflow error when the result exceeds the available integral portion of the result. When the database decreases the resulting scale, it truncates the excess fractional digits in the result.

### Math Operators for Time Intervals

Ocient supports math operators for time interval values (hour, minute, second, etc.), including when used with time-series data types such as TIME, TIMESTAMP, and DATE. You can use addition, subtraction, multiplication, and division on time intervals like any other numeric value.

Division operations can produce inaccurate results because the system truncates the precision of the result to a whole number. For details, see [Math Functions and Operators](#).

### Examples

#### Add Interval Value to Timestamp

This example uses the MINUTES cast function to add six minutes to the TIMESTAMP value returned from the TIMESTAMP cast function of the '2023-04-05 12:15:05' string.

```
SELECT TIMESTAMP('2023-04-05 12:15:05') + (MINUTES(6));
```

Output: 2023-04-05 12:21:05.000

#### Add Fractional Interval Value to Timestamp

This example shows how division can produce an inaccurate result. The division operation in the example must produce a whole number. In other words, the system calculates two minutes as  $5 / 2 = 2$  instead of two and a half minutes as  $5 / 2 = 2.5$ .

```
SELECT TIMESTAMP('2023-04-05 12:15:05') + (MINUTES(5) / 2);
```

Output: 2023-04-05 12:17:05.000

### Math Functions

These scalar math functions operate on a single value and return a single value. They can appear anywhere where an ordinary value or condition can appear.

For a list of common math symbols supported in Ocient, see [Math Functions and Operators](#).

### ABS

Returns the absolute value of a specified floating-point number.

### Syntax

**ABS(x)****Example**

```
SELECT ABS(-250.5);
```

*Output:* 250.5

**ACOS**

Returns the inverse cosine of a specified floating-point number.

**Syntax****ACOS(x)****Example**

```
SELECT ACOS(0.25);
```

*Output:* 1.318116071652818

**ACOSH**

Returns the hyperbolic arc-cosine of a specified floating-point number.

**Syntax****ACOSH(x)****Example**

```
SELECT ACOSH(2);
```

*Output:* 1.3169578969248166

**ASIN**

Returns the inverse sine of a specified floating-point number.

**Syntax****ASIN(x)****Example**

```
SELECT ASIN(0.25);
```

*Output:* 0.25268025514207865

**ASINH**

Returns the hyperbolic arc-sine of a specified floating-point number.

**Syntax****ASINH(x)****Example**

```
SELECT ASINH(2);
```

*Output:* 1.4436354751788103

**ATAN**

Returns the inverse tangent of a specified floating-point number.

**Syntax**

```
ATAN(x)
```

**Example**

```
SELECT ATAN(0.25);
```

*Output:* 0.24497866312686414

**ATAN2**

Returns the inverse tangent of two numeric, floating-point values.

**Syntax**

```
ATAN2(x, y)
```

**Example**

```
SELECT ATAN2(0.50, 1);
```

*Output:* 0.4636476090008061

**ATANH**

Returns the hyperbolic arc-tangent of a specified floating-point number.

**Syntax**

```
ATANH(x)
```

**Example**

```
SELECT ATANH(0.25);
```

*Output:* 0.25541281188299536

**BICDF**

The cumulative distribution function of the standard bivariate normal distribution, which returns the probability that a random sample of (A, B) is less than *x* and less than *y* (*A* <= *x* and *B* <= *y*). The correlation is set to 0 in the case where you specify two arguments only.

You can specify an optional **rho** coefficient to calculate the correlation.

For details, see the bivariate cumulative distribution function.

**Syntax**

```
BICDF(x, y [, rho])
```

**Example**

```
SELECT BICDF(1,2);
```

*Output:* 0.8222040420815764

**BIPDF**

The probability density function of the standard bivariate normal distribution returns the relative likelihood that a random sample would equal the specified values. The correlation is set to 0 in the case where you specify two arguments only.

You can specify an optional **rho** coefficient to calculate the correlation.

For details, see the bivariate probability density function.

**Syntax**

```
BIPDF(x, y, [, rho])
```

**Example**

```
SELECT BIPDF(1,2);
```

*Output:* 0.013064233284684893

**BITFUNC**

Performs a variety of bit operations.

The **char\_func** can be any of these string literals: 'AND', 'OR', or 'XOR'. This argument determines the bitwise operation that is performed on arguments **x** and **y**.

**Syntax**

```
BITFUNC(char_func, x, y)
```

| Argument       | Data Type | Description                                                                                                                                                                                                                                       |
|----------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| character_func | CHAR      | This argument determines the bitwise operation performed on arguments <b>x</b> and <b>y</b> . Can be any of these string literals: 'AND', 'OR', or 'XOR'. To perform bit operations using a NOT operator, use the <b>Math Functions</b> function. |
| <b>x</b>       | INT       | A numeric value to be evaluated by the function.                                                                                                                                                                                                  |
| <b>y</b>       | INT       | A numeric value to be evaluated by the function.                                                                                                                                                                                                  |

**Example**

```
SELECT BITFUNC('AND', 11, 12);
```

*Output:* 8

**Example**

```
SELECT BITFUNC('OR', 11, 12);
```

*Output:* 15

**Example**

```
SELECT BITFUNC('XOR', 11, 12);
```

*Output:* 7

**BITAND**

Alias for the **Math Functions** syntax `BITFUNC('AND', x, y)`.

**BITNOT**

Returns the bitwise negation of `integral_x`.

**Syntax**

```
BITNOT(integral_x)
```

**Example**

```
SELECT BITNOT(1);
```

*Output:* -2

**BITOR**

Alias for the **Math Functions** syntax `BITFUNC('OR', x, y)`.

**BITXOR**

Alias for the **Math Functions** syntax `BITFUNC('XOR', x, y)`.

**BOOLFUNC**

Performs a Boolean logical evaluation on arguments `x` and `y`. Numbers equal to 0 evaluate to **false**, and numbers not equal to 0 evaluate to **true**.

**Syntax**

```
BOOLFUNC(char_func, x, y)
```

| Argument  | DataType | Description                                                                                                                                                                                                                                               |
|-----------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| char_func | CHAR     | This argument determines the Boolean operation performed on arguments <code>x</code> and <code>y</code> . Can be any of these string literals: 'AND', 'OR', or 'XOR'. To perform operations using a NOT operator, use the <b>Math Functions</b> function. |
| x         | INT      | A numeric value for evaluation by the function.                                                                                                                                                                                                           |
| y         | INT      | A numeric value for evaluation by the function.                                                                                                                                                                                                           |

**Example**

```
SELECT BOOLFUNC('AND',0,1);
```

*Output:* false

**Example**

```
SELECT BOOLFUNC('OR',0,1);
```

*Output:* true

**Example**

```
SELECT BOOLFUNC('XOR',0,1);
```

*Output:* true

## BOOLAND

Alias for the **Math Functions** syntax `BOOLFUNC('AND', x, y)`.

## BOOLNOT

Returns the logical negation of the **Math Functions** function. Numbers equal to 0 evaluate to **true**, and numbers not equal to 0 evaluate to **false**.

### Syntax

```
BOOLNOT(numeric_x)
```

### Example

```
SELECT BOOLNOT(0);
```

*Output:* true

## BOOLOR

Alias for the **Math Functions** syntax `BOOLFUNC('OR', x, y)`.

## BOOLXOR

Alias for the **Math Functions** syntax `BOOLFUNC('XOR', x, y)`.

## CBRT

Returns the cube root of the numeric value **x**.

### Syntax

```
CBRT(x)
```

### Example

```
SELECT CBRT(64);
```

*Output:* 4.0

## CDF

The cumulative distribution function of the standard normal distribution. Returns the probability that a random sample is less than or equal to the specified value.

For details about the cumulative distribution function, see Cumulative distribution function.

### Syntax

```
CDF(x)
```

### Example

```
SELECT CDF(1.5);
```

*Output:* 0.9331927987311419

**CEIL**

Returns the nearest integer greater than or equal to **x**.

**Syntax**

```
CEIL(x)
```

**Example**

```
SELECT CEIL(23.45);
```

*Output:* 24.0

**CEILING**

Alias for CEIL.

**COS**

Returns the cosine of **x**.

**Syntax**

```
COS(x)
```

**Example**

```
SELECT COS(2);
```

*Output:* -0.4161468365471424

**COSH**

Returns the hyperbolic cosine of **x**.

**Syntax**

```
COSH(x)
```

**Example**

```
SELECT COSH(2);
```

*Output:* 3.7621956910836314

**COT**

Returns the cotangent of **x**.

**Syntax**

```
COT(x)
```

**Example**

```
SELECT COT(2);
```

*Output:* -0.45765755436028577

**DEGREES**

Returns the corresponding angle in degrees for **x** in radians.

**Syntax**



DEGREES(x)

### Example

```
SELECT DEGREES(1.5);
```

*Output:* 85.94366926962348

### DIV

Returns the result of x divided by y. If y is zero, returns NULL.

### Syntax

DIV(x, y)

### Example

```
SELECT DIV(45, 9);
```

*Output:* 5

### ERF

The error function is used for measurements that follow a normal distribution. Use this function to find the probability that the error of a single measurement lies between [-a, a]. For details about the error function, see Error function.

### Syntax

ERF(x)

### Example

```
SELECT ERF(8);
```

*Output:* 1.0

### ERFC

The complement of the error function.  $ERFC(x) = 1 - ERF(x)$ .

### Syntax

ERFC(x)

### Example

```
SELECT ERFC(8);
```

*Output:* 1.1224297172982928E-29

### EXP

Returns the exponential of x (e raised to the power of x).

### Syntax

EXP(x)

### Example

```
SELECT EXP(1);
```

*Output:* 2.718281828459045

**FLOOR**

Returns the nearest integer less than or equal to **x**.

**Syntax**

```
FLOOR(x)
```

**Example**

```
SELECT FLOOR(22.75);
```

*Output:* 22.0

**GAMMA**

Compute the gamma function of the specified value. For details about the gamma function, see the Gamma function.

**Syntax**

```
GAMMA(x)
```

**Example**

```
SELECT GAMMA(9.0);
```

*Output:* 40319.999999999999

**HEXBINX**

Returns the x-coordinate of the center of the nearest hexagonal bin to the point (**x**, **y**). Hexagonal bins have a side length of 1.

**Syntax**

```
HEXBINX(x, y)
```

**Example**

```
SELECT HEXBINX(5, 7);
```

*Output:* 4.5

**HEXBINY**

Returns the y-coordinate of the center of the nearest hexagonal bin to the point (**x**, **y**). Hexagonal bins have a side length of 1.

**Syntax**

```
HEXBINY(x, y)
```

**Example**

```
SELECT HEXBINY(5, 7);
```

*Output:* 7.794

**IERF**

The inverse of the ERF error function.

**Syntax**

```
IERF(x)
```

**Example**

```
SELECT IERF(0);
```

*Output:* 0.0

### IERFC

The inverse of the complement of the error function.

#### Syntax

```
IERFC(x)
```

#### Example

```
SELECT IERFC(1);
```

*Output:* 0.0

### LEAKYRELU

Returns the leaky rectified linear unit function of *x*.

If unspecified, the optional *numeric\_value\_slope* argument defaults to 0.01.

#### Syntax

```
LEAKYRELU(x [, numeric_value_slope] )
```

#### Example

```
SELECT LEAKYRELU(-40);
```

*Output:* -0.4

### LEFT\_SHIFT

Returns *x* shifted to the left by *y* bits. If *y* is less than 0, then *x* shifts to the right by the absolute value. The function throws an out-of-range error if you attempt a shift into the sign bit.

#### Syntax

```
LEFT_SHIFT(x, y)
```

#### Example

```
SELECT LEFT_SHIFT(1,2);
```

*Output:* 4

### LN

Returns the natural logarithm of *x* (base *e* or 2.171828...).

#### Syntax

```
LN(x)
```

#### Example

```
SELECT LN(2);
```

*Output:* 0.6931471805599453

**LOG2**

Returns the base 2 logarithm of **x**.

**Syntax**

```
LOG2(x)
```

**Example**

```
SELECT LOG2(2);
```

*Output:* 1.0

**LOG**

Returns the base 10 logarithm of **x**.

The optional **base** argument specifies the numeral system to use. If unspecified, the function defaults to base 10.

**Syntax**

```
LOG([ base,] x)
```

**Example**

```
SELECT LOG(2);
```

*Output:* 0.3010299956639812

**LOG\_GAMMA**

The natural logarithm of the absolute value of the gamma function.

**Syntax**

```
LOG_GAMMA(x)
```

**Example**

```
SELECT LOG_GAMMA(10);
```

*Output:* 12.80182748008147

**MOD**

Returns the remainder from **x** divided by **y**.

**Syntax**

```
MOD(x, y)
```

**Example**

```
SELECT MOD(12, 10);
```

*Output:* 2

**PI**

Returns the constant value of  $\pi$ .

**Syntax**

```
PI()
```

**Example**

```
SELECT PI();
```

*Output:* 3.141592653589793

## PMOD

Returns the smallest non-negative equivalence class of  $x \% y$ .

### Syntax

```
PMOD(x, y)
```

### Example

```
SELECT PMOD(12,10);
```

*Output:* 2

## POWER

Returns  $x$  raised to the power of  $y$ . The return type `FLOAT` is returned for a `FLOAT` input, `DECIMAL` for `DOUBLE` input, and `BIGINT` otherwise.

### Syntax

```
POWER(x, y)
```

### Example

```
SELECT POWER(2, 4);
```

*Output:* 16

## PROBIT

The inverse of the cumulative distribution function. The function accepts only decimal values between 0 and 1. For details about the probit function, see the Probit model.

### Syntax

```
PROBIT(x)
```

### Example

```
SELECT PROBIT(0.4);
```

*Output:* -0.2533471031357998

## RADIANS

Returns the corresponding angle in radians for  $x$  in degrees.

### Syntax

```
RADIANS(x)
```

### Example

```
SELECT RADIANS(2);
```

*Output:* 0.03490658503988659

**RAND**

Takes no argument and returns a random DOUBLE value in the range [0, 1).

**Syntax**

```
RAND()
```

**Example**

```
SELECT RAND();
```

*Output:* 0.6053105024272647

**RELU**

Returns the rectified linear unit function of **x**.

**Syntax**

```
RELU(x)
```

**Example**

```
SELECT RELU(-1);
```

*Output:* 0.0

**RIGHT\_SHIFT**

Returns **x** shifted to the right by **y** bits. If **y** is less than 0, then **x** shifts to the left by the absolute value of **y** in bits. The function throws an out-of-range error if you attempt a shift into the sign bit.

**Syntax**

```
RIGHT_SHIFT(x, y)
```

**Example**

```
SELECT RIGHT_SHIFT(5, 1);
```

*Output:* 2

**ROUND**

Returns **x** rounded to the nearest integer.

**Syntax**

```
ROUND(x)
```

**Example**

```
SELECT ROUND(12.49);
```

*Output:* 12.0

**SIGN**

Returns the positive (+1), zero (0), or negative (-1) sign of **x**.

**Syntax**

```
SIGN(x)
```

**Example**

```
SELECT SIGN(+12);
```

*Output:* 1

#### Example

```
select SIGN(0);
```

*Output:* 0

#### Example

```
SELECT SIGN(-12);
```

*Output:* -1

### SIN

Returns the sine of **x**.

#### Syntax

```
SIN(x)
```

#### Example

```
SELECT SIN(2);
```

*Output:* 0.9092974268256817

### SINH

Returns the hyperbolic sine of **x**.

#### Syntax

```
SINH(x)
```

#### Example

```
SELECT SINH(2);
```

*Output:* 3.626860407847019

### SQRT

Returns the square root of **x**.

#### Syntax

```
SQRT(x)
```

#### Example

```
SELECT SQRT(81);
```

*Output:* 9

### SQUARE

Returns the square of **x**.

#### Syntax

```
SQUARE(x)
```

#### Example

```
SELECT SQUARE(7);
```

*Output:* 49

## TAN

Returns the tangent of *x*.

### Syntax

```
TAN(x)
```

### Example

```
SELECT TAN(1.75);
```

*Output:* -5.52037992250933

## TANH

Returns the hyperbolic tangent of *x*.

### Syntax

```
TANH(x)
```

### Example

```
SELECT TANH(1.75);
```

*Output:* 0.9413755384972874

## TRUNC

Returns *x* truncated to *y* decimal places.

### Syntax

```
TRUNC(x,y)
```

### Example

```
SELECT TRUNC(1.23456,3);
```

*Output:* 1.234

## TRUNCATE

Alias for TRUNC.

### Related Links

[Array Functions and Operators](#)

[Matrix Functions and Operators](#)

[Tuple Functions and Operators](#)

[Formatting Functions](#)

[Data Types](#)

[Query Ocient](#)



## 10.13 Matrix Functions and Operators

The Ociant Hyperscale Data Warehouse™ supports SQL-based tools for working with multi-dimensional data, including functions to construct, transform, and analyze matrices and vectors. Matrix functions include arithmetic operations, linear algebra (e.g., transpose, inverse, eigenvalues), and statistical tools.

Matrixes store their values in DOUBLE format. To define a matrix data type in SQL, use the syntax `MATRIX[rows][columns]`, where the bracketed numbers specify the number of matrix rows and columns.

### Example

This example creates a basic table `matrix_example` with a matrix column `col_matrix` that contains matrixes with two rows and three columns.

```
CREATE TABLE matrix_example (
  id INT PRIMARY KEY,
  col_matrix MATRIX[2][3] NOT NULL
);
```

Insert data for two matrixes into the matrix column.

```
INSERT INTO matrix_example (id, col_matrix) VALUES
(1, '{{1, 2, 3}, {4, 5, 6}}'),
(2, '{{0, 0, 0}, {7, 8, 9}}');
```

For more DDL examples that use matrixes, see [CREATE TABLE SQL Statement Examples](#).

For details about working with matrix values, see [Data Types](#).

### Matrix Functions

In the examples, `_R{}` and `_C{}` are vector notation, meaning `_R{ 1,2,3 }` is a row vector with values 1, 2, and 3.

| Function                  | Syntax                                              | Purpose                                                                       | Example                                                     | Result                                           |
|---------------------------|-----------------------------------------------------|-------------------------------------------------------------------------------|-------------------------------------------------------------|--------------------------------------------------|
| Constructor               | <code>make_matrix_ixj(i, j, e_00, ..., e_ij)</code> | Creates an <code>ixj</code> matrix with elements <code>e_00, ..., e_ij</code> | <code>make_matrix_1x3(1,3,1,2,3)</code>                     | <code>{0,2,0,3,0 }</code>                        |
| Constructor (2)           | <code>matrix_from_text(string)</code>               | Creates a matrix from the specified string                                    | <code>matrix_from_text('{{1,2,3}, {1,2,3}}')</code>         | <code>{ {1.0,2.0,3.0}, {1.0,2.0,3.0} }</code>    |
| + (matrix addition)       | <code>matrix + matrix</code>                        | Performs matrix addition                                                      | <code>{ {1,2,3}, {4,5,6} } + { {4,5,6}, {1,2,3} }</code>    | <code>{ {5.0,7.0,9.0}, {5.0,7.0,9.0} }</code>    |
| - (matrix subtraction)    | <code>matrix - matrix</code>                        | Performs matrix subtraction                                                   | <code>{ {1,2,3}, {4,5,6} } - { {4,5,6}, {1,2,3} }</code>    | <code>{ {-3.0,-3.0,-3.0}, {3.0,3.0,3.0} }</code> |
| * (scalar multiplication) | <code>numeric * matrix</code>                       | Performs scalar multiplication                                                | <code>2 * { {4,5,6}, {1,2,3} }</code>                       | <code>{ {2.0,4.0,6.0}, {8.0,10.0,12.0} }</code>  |
| * (matrix multiplication) | <code>matrix * matrix</code>                        | Performs matrix multiplication                                                | <code>{ {1,2,3}, {4,5,6} } * { {1,2}, {3,4}, {5,6} }</code> | <code>{ {22.0,28.0}, {49.0,64.0} }</code>        |
| / (scalar division)       | <code>matrix / numeric</code>                       | Performs scalar division                                                      | <code>{ {1,2,3}, {4,5,6} } / 2</code>                       | <code>{ {0.5,1.0,1.5}, {2.0,2.5,3.0} }</code>    |
| transpose                 | <code>transpose(matrix)</code>                      | Returns transpose of the matrix                                               | <code>transpose( { {1,2,3}, {4,5,6} } )</code>              | <code>{ {1.0,4.0}, {2.0,5.0}, {3.0,6.0} }</code> |
| determinant               | <code>det(matrix)</code>                            | Returns the determinant of the matrix as a double                             | <code>det( { {1,2}, {3,4} } )</code>                        | <code>-2.0</code>                                |

| Function           | Syntax                             | Purpose                                                                          | Example                                                                                                                                                                                                                  | Result                       |
|--------------------|------------------------------------|----------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------|
| inverse            | inverse(matrix)                    | Returns inverse of a square, invertible matrix                                   | inverse( { {1,2}, {3,4} } )                                                                                                                                                                                              | { {-2.0, 1.0}, {1.5, -0.5} } |
| trace              | matrix_trace(matrix)               | Returns trace of a square matrix as a double                                     | matrix_trace( { {1,2,3}, {4,5,6}, {7,8,9} } )                                                                                                                                                                            | 15.0                         |
| vector sum         | vector_sum(matrix)                 | Returns the sum of elements in a one-dimensional matrix/vector                   | vector_sum( _R{ 1,2,3 } )vector_sum( _C{ 1,2,3 } )                                                                                                                                                                       | 6.0                          |
| vector min         | vector_min(matrix)                 | Returns the minimum of elements in a one-dimensional matrix/vector               | vector_min( _R{ 1,2,3 } )vector_min( _C{ 1,2,3 } )                                                                                                                                                                       | 1.0                          |
| vector max         | vector_max(matrix)                 | Returns the maximum of elements in a one-dimensional matrix/vector               | vector_max( _R{ 1,2,3 } )vector_max( _C{ 1,2,3 } )                                                                                                                                                                       | 3.0                          |
| vector argmin      | vector_argmin(matrix)              | Returns the argmin of a one-dimensional matrix or vector. The index starts at 0. | vector_argmin( _R{1, -1, 5, 0})                                                                                                                                                                                          | 1                            |
| vector argmax      | vector_argmax(matrix)              | Returns the argmax of a one-dimensional matrix or vector. The index starts at 0. | vector_argmax( _C{1, -1, 5, 0})                                                                                                                                                                                          | 2                            |
| softmax            | softmax(matrix)                    | Returns the softmax of a one-dimensional matrix or vector.                       | softmax( _R{1, 1})                                                                                                                                                                                                       | {{0.5, 0.5}}                 |
| cross entropy loss | cross_entropy_loss(matrix, matrix) | Returns the cross entropy loss of two one-dimensional matrixes or vectors.       | cross_entropy_loss( _R{ 0.2, 0.5, 0.8 }, _C{ 0,1,0 } )cross_entropy_loss( _C{ 0.2, 0.5, 0.8 }, _R{ 0,1,0 } )cross_entropy_loss( _R{ 0.2, 0.5, 0.8 }, _R{ 0,1,0 } )cross_entropy_loss( _C{ 0.2, 0.5, 0.8 }, _C{ 0,1,0 } ) | 0.69                         |
| log loss           | log_loss(matrix, matrix)           | Returns the log loss of two one-dimensional matrixes or vectors.                 | log_loss( _R{ 0.2, 0.5, 0.8 }, _C{ 0,1,0 } )log_loss( _C{ 0.2, 0.5, 0.8 }, _R{ 0,1,0 } )log_loss( _R{ 0.2, 0.5, 0.8 }, _R{ 0,1,0 } )log_loss( _C{ 0.2, 0.5, 0.8 }, _C{ 0,1,0 } )                                         | 2.53                         |

| Function                                                                                                                                                                                                                                                                                                                                      | Syntax                      | Purpose                                                             | Example                                         | Result |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|---------------------------------------------------------------------|-------------------------------------------------|--------|
| logits loss                                                                                                                                                                                                                                                                                                                                   | logits_loss(matrix, matrix) | Returns the logits loss of two one-dimensional matrixes or vectors. | logits_loss( _R{ 0.2, 0.5, 0.8 }, _C{ 0,1,0 } ) |        |
| <pre>logits_loss( _C{ 0.2, 0.5, 0.8 }, _R{ 0,1,0 } ) logits_loss( _R{ 0.2, 0.5, 0.8 }, _R{ 0,1,0 } ) logits_loss( _C{ 0.2, 0.5, 0.8 }, _C{ 0,1,0 } )   2.44  </pre>                                                                                                                                                                           |                             |                                                                     |                                                 |        |
| hinge loss   hinge_loss(matrix, matrix)   Returns the hinge loss of two one-dimensional matrixes or vectors.                                                                                                                                                                                                                                  |                             |                                                                     |                                                 |        |
| hinge_loss( _R{ 0.2, 0.5, 0.8 }, _C{ 0,1,0 } ) hinge_loss( _C{ 0.2, 0.5, 0.8 }, _R{ 0,1,0 } ) hinge_loss( _R{ 0.2, 0.5, 0.8 }, _R{ 0,1,0 } ) hinge_loss( _C{ 0.2, 0.5, 0.8 }, _C{ 0,1,0 } )   3.5                                                                                                                                             |                             |                                                                     |                                                 |        |
| dot product   dot(matrix, matrix)   Returns dot product of two one-dimensional matrixes/vectors   dot( _R{ 1,2,3 }, _C{ 1,2,3 } )dot( _C{ 1,2,3 }, _R{ 1,2,3 } )dot( _R{ 1,2,3 }, _R{ 1,2,3 } )dot( _C{ 1,2,3 }, _C{ 1,2,3 } )   14.0                                                                                                         |                             |                                                                     |                                                 |        |
| magnitude   abs(matrix)   Returns magnitude of one-dimensional matrix/vector   abs( _R{ 1,2,3 } )abs( _C{ 1,2,3 } )   3.74                                                                                                                                                                                                                    |                             |                                                                     |                                                 |        |
| LUPQ decomp   LUPQ_decomp(matrix)   Returns LUPQ decomposition of a square matrix A as a tuple of 4 matrixes where PAQ = LU                                                                                                                                                                                                                   |                             |                                                                     |                                                 |        |
| LUP decomp   LUP_decomp(matrix)   Returns LUP decomposition of a square matrix as a tuple of 3 matrixes                                                                                                                                                                                                                                       |                             |                                                                     |                                                 |        |
| QR decomp   qrdecomp(matrix)   Returns QR decomposition of a matrix as a tuple of 2 matrixes                                                                                                                                                                                                                                                  |                             |                                                                     |                                                 |        |
| SVD decomp   SVD_decomp(matrix)   Returns SVD decomposition of a matrix as a tuple of 3 matrixes                                                                                                                                                                                                                                              |                             |                                                                     |                                                 |        |
| eigenvalues/vectors   eigen(matrix)   Returns eigenvalues and eigenvalues of a square matrix as a vector of pairs                                                                                                                                                                                                                             |                             |                                                                     |                                                 |        |
| identity matrix   identity_matrix(unsigned int)   Returns an identity matrix of the specified dimension   identity_matrix(3)   { {1.0,0.0,0.0}, {0.0,1.0,0.0}, {0.0,0.0,1.0} }                                                                                                                                                                |                             |                                                                     |                                                 |        |
| NULL matrix   null_matrix(unsigned int, unsigned int)   Returns a NULL matrix of the specified (row, col) dimensions   null_matrix(2, 3)   NULL                                                                                                                                                                                               |                             |                                                                     |                                                 |        |
| zero matrix   zero_matrix(unsigned int, unsigned int)   Returns a zero matrix of the specified (row, col) dimensions   zero_matrix(2, 3)   { {0.0,0.0,0.0}, {0.0,0.0,0.0} }                                                                                                                                                                   |                             |                                                                     |                                                 |        |
| matrix dimension   matrix_dim(matrix)   Returns the dimensions of the specified matrix as a tuple of (row, col) integers. When the input matrix is NULL, the function returns NULL.   matrix_dim(_R{1, 2, 3})   TUPLE<INT,INT>(1, 3)                                                                                                          |                             |                                                                     |                                                 |        |
| Frobenius norm   frobenius(matrix)   Returns the Frobenius norm of a matrix. This norm is the matrix norm of anM` ` x ` `N* *matrix defined as the square root of the sum of the absolute squares of its elements. (1) a ^2 is the same as a^2, where a is a real number.   frobenius(matrix_from_text('{{3,4},{8,6}}'))   11.180339887498949 |                             |                                                                     |                                                 |        |

## Matrix Operators

| Operator | Syntax                    | Purpose                                                                                                                                                    |
|----------|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <        | matrix < matrix           | Less than comparison                                                                                                                                       |
| >        | matrix > matrix           | Greater than comparison                                                                                                                                    |
| <=       | matrix <= matrix          | Less than or equal to comparison                                                                                                                           |
| >=       | matrix >= matrix          | Greater than or equal to comparison                                                                                                                        |
| =        | matrix = matrix           | Equal to comparison                                                                                                                                        |
| <>       | matrix <> matrix          | Not equal to comparison                                                                                                                                    |
| [.]      | matrix[rowIndex,colIndex] | Access matrix at the specified row and column. 1-Indexed. Accessing a NULL matrix or using a NULL index returns NULL. Access out of bounds returns a NULL. |

| Operator | Syntax              | Purpose                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [:,:]    | matrix[r1:c1,r2:c2] | <p>Slice matrix at the specified row range and column range. The index starts at 1. Slicing a NULL matrix or using a NULL index returns NULL. Start indices omitted or below 1 become 1. End indices omitted or above the number of rows/columns become the number of rows/columns. Example</p> <p>—matrix_10X10[:10,-1:20] is equivalent to matrix[1:10,1:10]. Ranges completely out of bounds throw an INVALID_ARGUMENT error. Sequential slices like matrix[1:1,1:1] [1:1,1:1] throw an error. Slices cannot be combined with access. matrix[1,1:10] is not allowed, but matrix_10X10[1:1,1:10] is. Trying to slice with non-constant indexes throws an error.</p> |

### Operator Examples

| Operator | SQL Predicate                                | Result |
|----------|----------------------------------------------|--------|
| <        | { {1,2,3}, {4,5,6} } < { {4,5,6}, {1,2,3} }  | true   |
|          | { {1,2,3}, {4,5,6} } < { {1,2,2}, {1,2,3} }  | false  |
|          | { {1,2,3}, {4,5,6} } < { {1,2,3}, {4,5,6} }  | false  |
| >        | { {4,5,6}, {4,5,6} } > { {1,2,3}, {7,8,9} }  | true   |
|          | { {1,2,3}, {4,5,6} } > { {4,5,6}, {1,2,3} }  | false  |
|          | { {1,2,3}, {4,5,6} } > { {1,2,3}, {4,5,6} }  | false  |
| <=       | { {1,2,3}, {4,5,6} } <= { {4,5,6}, {1,2,3} } | true   |
|          | { {1,2,3}, {4,5,6} } <= { {1,2,2}, {1,2,3} } | false  |
|          | { {1,2,3}, {4,5,6} } <= { {1,2,3}, {4,5,6} } | true   |
| >=       | { {4,5,6}, {4,5,6} } >= { {1,2,3}, {7,8,9} } | true   |
|          | { {1,2,3}, {4,5,6} } >= { {1,5,6}, {1,2,3} } | false  |
|          | { {1,2,3}, {4,5,6} } >= { {1,2,3}, {4,5,6} } | true   |
| =        | { {1,2,3}, {4,5,6} } = { {1,2,3}, {4,5,6} }  | true   |

| Operator               | SQL Predicate                                | Result                   |
|------------------------|----------------------------------------------|--------------------------|
| <>                     | { {1,2,3}, {4,5,6} } = { {1,2,4}, {4,5,6} }  | false                    |
|                        | { {1,2,3}, {4,5,6} } <> { {1,2,3}, {4,5,6} } | false                    |
|                        | { {1,2,3}, {4,5,6} } <> { {1,2,4}, {4,5,6} } | true                     |
|                        | { {1,2,3}, {4,5,6} } [1,2]                   | 2.0                      |
| matrix_access_operator | { {1,2,3}, {4,5,6} } [NULL,2]                | NULL                     |
| matrix_slice_operator  | { {1,2,3}, {4,5,6} } [1:,2:3]                | { {2.0,3.0}, {5.0,6.0} } |
|                        | { {1,2,3}, {4,5,6} } [1:NULL,2:3]            | NULL                     |

## Bibliography

- (1) Golub, Gene H., and Charles F. Van Loan. *Matrix Computations*. 3rd ed, Johns Hopkins University Press, 1996.

## Related Links

[Array Functions and Operators](#)

[Math Functions and Operators](#)

[Tuple Functions and Operators](#)

[Data Types](#)

[Query Ocient](#)

## 10.14 Network Type Functions

The Ocient® System supports these functions for IP and IPV4 data types.

### IP

Casts the IPV4 data type to the IP data type. The returned value is the IPv6-mapped IPv4 address.

### Syntax

```
IP(ipv4_col)
```

| Argument | Data Type  | Description                             |
|----------|------------|-----------------------------------------|
| ipv4_col | CHARorIPV4 | IP address in theIPV4data type to cast. |

### Example

Cast the IPv4 address 128.1.2.3 to the IPv6-mapped IPv4 address.

```
SELECT IP(IPV4('128.1.2.3'));
```

### Output

```
ip(ipv4(('128.1.2.3')))
```

```
-----
128.1.2.3
```

```

Fetched 1 row
```

## IPV4

Returns the IPv4 portion of the IPv6 address.

You can only cast the IP data type to IPv4 if the IPv6 value has an IPv4 semantic equivalent. The database can convert these IPv6 address formats to IPv4:

- IPv4-compatible IPv6 ::x.x.x.x
- IPv6-mapped IPv4 ::ffff:x.x.x.x
- NAT64-prefixed 64:ff9b::x.x.x.x

### Syntax

```
IPV4(ip_col)
```

| Argument | Data Type | Description                                                                                                                                                                                          |
|----------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ip_col   | IP        | IPv6-mapped IP address in the IP data type to cast. If the IPv6 address is not IPv4-compatible, IPv6-mapped, or NAT64-prefixed, the cast returns an <code>INVALID DATA TYPE CONVERSION</code> error. |

### Example

Cast the IPv6-mapped IP address ::ffff:128.1.2.3 to the IPV4 data type. Return the IPv4 portion of the address.

```
SELECT IPV4(IP('::ffff:128.1.2.3'));
```

### Output

```
ipv4(ip(('::ffff:128.1.2.3')))  
-----  
128.1.2.3
```

Fetches 1 row

## IS\_IPV4

The IS\_IPV4 function tests whether the database can convert the IP value to the IPV4 data type. The database can convert these IPv6 address formats to IPv4:

- IPv4-compatible IPv6 ::x.x.x.x
- IPv6-mapped IPv4 ::ffff:x.x.x.x
- NAT64-prefixed 64:ff9b::x.x.x.x

The function returns a Boolean value (`true` or `false`) that indicates whether the conversion works from the specified IP address to the IPv4 address.

### Syntax

```
IS_IPV4(ip_col)
```

| Argument | Data Type | Description                                             |
|----------|-----------|---------------------------------------------------------|
| ip_col   | IP        | Specified IP address for conversion to an IPv4 address. |

### Example

Determine whether the `::ffff:128.1.2.3` string is a valid IPv4 address. Use the `IP` function to cast the string as an IP address.

```
SELECT IS_IPV4(IP('::ffff:128.1.2.3'));
```

*Output*

```
is_ipv4_convertible(ip(('::ffff:128.1.2.3')))  
-----  
false  
  
Fetched 1 row
```

## SUBNET

The `SUBNET` function computes the prefix from an `IP` or `IPv4` value and the size of the prefix. The function returns a value of the same type as the `ip_col` argument.

### Syntax

```
SUBNET(ip_col, prefix_size)
```

| Argument                 | Data Type | Description                                                                                                                                                                                                                                                                                                                  |
|--------------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ip_col</code>      | IPorIPv4  | The IP address for the prefix calculation.                                                                                                                                                                                                                                                                                   |
| <code>prefix_size</code> | Integer   | The size of the subnet network prefix. For <code>IPv4</code> addresses, the size must be in the range <code>[0, 32]</code> , and for <code>IP</code> addresses, the size must be in the range <code>[0, 128]</code> . If the prefix exceeds these range values, the function returns an <code>INVALID ARGUMENT</code> error. |

### Example

Calculate the prefix for the NAT64-prefixed IP address `64:ff9b::128.2.3.4` with prefix size `98`. Use the `IP` function to cast the string as an IP address.

```
SELECT SUBNET(IP('64:ff9b::128.2.3.4'), 98);
```

*Output*

```
subnet(ip(('64:ff9b::128.2.3.4')), (98))  
-----  
64:ff9b:0:0:0:0:8000:0  
  
Fetched 1 row
```

### Related Links

[Data Types](#)

## 10.15 Scalar Data Conversion Functions

These functions convert one data type to another.

## Scalar Data Conversion Functions

| Function | Syntax                                                 | Purpose                                                                                                            |
|----------|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| BYTE     | BYTE(numeric value)                                    | Casts the argument to a value of type byte                                                                         |
| SMALLINT | BYTE(character value)                                  | Parses the string as a byte value                                                                                  |
|          | SMALLINT(numeric value)                                | Casts the argument to a value of type smallint                                                                     |
| INTEGER  | SMALLINT(character value)                              | Parses the string as a smallint value                                                                              |
|          | INT(numeric value) or<br>INTEGER(numeric value)        | Casts the argument to a value of type integer                                                                      |
| BIGINT   | INT(character value) or<br>INTEGER(character value)    | Parses the string as an integer value                                                                              |
|          | BIGINT(numeric value)                                  | Casts the argument to a value of type bigint                                                                       |
|          | BIGINT(character value)                                | Parses the string as a bigint value                                                                                |
|          | BIGINT(time value)                                     | Creates a bigint with the number of milliseconds represented by the time argument                                  |
| FLOAT    | BIGINT(timestamp value)                                | Creates a bigint with the number of milliseconds after the {{Unix}} epoch corresponding to the timestamp argument  |
|          | FLOAT(numeric value)                                   | Casts the argument to a value of type float                                                                        |
| DOUBLE   | FLOAT(character value)                                 | Parses the string as a float value                                                                                 |
|          | DOUBLE(numeric value)                                  | Casts the argument to a value of type double                                                                       |
| DECIMAL  | DOUBLE(character value)                                | Parses the string as a double value                                                                                |
|          | DECIMAL(numeric value)                                 | Casts the argument to a value of type decimal                                                                      |
|          | DECIMAL(character value)                               | Parses the string and creates a decimal value                                                                      |
|          | DECIMAL(numeric value, integer value, integer value)   | Casts the first argument to a decimal value with precision of the second argument and scale of the third argument  |
| CHAR     | DECIMAL(character value, integer value, integer value) | Parses the first argument to a decimal value with precision of the second argument and scale of the third argument |
|          | STR                                                    | CHAR(numeric value)                                                                                                |
|          | CHAR(character value)                                  | None                                                                                                               |
|          | CHAR(timestamp)                                        | Creates a string representation of the timestamp                                                                   |
|          | CHAR(date)                                             | Creates a string representation of the date                                                                        |
|          | CHAR(time)                                             | Creates a string representation of the time                                                                        |
|          | CHAR(boolean)                                          | Creates a string representation of the Boolean value                                                               |
|          | CHAR(binary)                                           | Creates a hexadecimal string representation of the binary value                                                    |
|          | CHAR(ip)                                               | Creates a string representation of the IPV4 or IPV6 value                                                          |



| Function | Syntax                   | Purpose                                                                                                                                                                                |
|----------|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BINARY   | CHAR(ipv4)               | Creates a string representation of the IPV4 value                                                                                                                                      |
|          | CHAR(hash)               | Creates a string representation of the fixed-binary hash value                                                                                                                         |
|          | CHAR(point)              | Creates a string representation of the point value                                                                                                                                     |
|          | CHAR(uuid)               | Creates a string representation of the Universally Unique Identifier (UUID)                                                                                                            |
|          | BINARY(character value)  | Parses a hexadecimal string (such as <code>0x54ab</code> ) to create a binary value. Letters can be of either case.                                                                    |
| BOOLEAN  | BINARY(binary)           | None                                                                                                                                                                                   |
|          | BINARY(hash)             | Converts the fixed length binary value (hash) to a variable length binary value                                                                                                        |
|          | BOOLEAN(character value) | Parses a string to create a Boolean value. The string must contain either true or false. It is case-insensitive.                                                                       |
| DATE     | BOOLEAN(boolean)         | None                                                                                                                                                                                   |
|          | DATE(character value)    | Parses a string in the form 'YYYY-MM-DD' to create a date. Extra characters are ignored.                                                                                               |
|          | DATE(date)               | None                                                                                                                                                                                   |
| HASH     | DATE(timestamp)          | Truncates a timestamp to be a date value                                                                                                                                               |
|          | HASH(char, length)       | Creates a fixed length binary value, with length , from a string, i.e. <code>0x1234abcd</code> . Zero extended if the string does not have enough bytes, truncated if it has too many. |
|          | HASH(binary, length)     | Creates a fixed length binary value, with length , from a variable length binary value. Zero extended if the string does not have enough bytes, truncated if it has too many.          |
| IP       | IP(character value)      | Creates a value of type IPV4 or IPV6 from a string                                                                                                                                     |
| IPV4     | IPV4(character value)    | Parses a string containing an IP address and creates a value of type IPV4                                                                                                              |
| TIME     | IPV4(ipv4)               | no-op                                                                                                                                                                                  |
|          | TIME(character value)    | Parses the string to create a time value. The string must be in the form 'HH:MM[.SSSSSSSS]'                                                                                            |
|          | TIME(bigint)             | Creates a time value representing a certain number of milliseconds after midnight                                                                                                      |
|          | TIME(time)               | None                                                                                                                                                                                   |
|          | TIME(timestamp)          | Makes a time value from the time component of a timestamp                                                                                                                              |

| Function  | Syntax                     | Purpose                                                                                                       |
|-----------|----------------------------|---------------------------------------------------------------------------------------------------------------|
| TIMESTAMP | TIMESTAMP(character value) | Parses a string and makes a timestamp value. The string must be in the format 'YYYY-MM-DD[HH:MM][.SSSSSSSS]'. |
|           | TIMESTAMP(date)            | Makes a timestamp from the date by setting the time component to 00:00:00.0                                   |
|           | TIMESTAMP(timestamp)       | None                                                                                                          |
|           | TIMESTAMP(bigint value)    | Creates a timestamp from a bigint representing the number of milliseconds after the Unix epoch (1/1/1970).    |
| UUID      | UUID(character value)      | Parses the string and makes a UUID value. The string must be a valid UUID.                                    |
|           | UUID(uuid)                 | None                                                                                                          |
| WEEKS     | WEEKS(integral value)      | Converts an integral value to an interval value of type weeks for date calculations.                          |
|           | Weeks(weeks)               | None                                                                                                          |
| DAYS      | DAYS(integral value)       | Converts an integral value to an interval value of type days for date calculations.                           |
|           | DAYS(days)                 | None                                                                                                          |
|           | DAYS(weeks)                | Converts an interval value of type weeks to an interval value of type days representing the same time span.   |
|           |                            |                                                                                                               |
| MONTHS    | MONTHS(integral value)     | Converts an integral value to an interval value of type months for date calculations.                         |
|           | MONTHS(months)             | None                                                                                                          |
|           | MONTHS(years)              | Converts an interval value of type years to an interval value of type months representing the same time span  |
| YEARS     | YEARS(integral value)      | Converts an integral value to an interval value of type years                                                 |
|           | YEARS(years)               | None                                                                                                          |
| HOURS     | HOURS(integral value)      | Converts an integral value to an interval value of type hours                                                 |
|           | HOURS(hours)               | None                                                                                                          |
|           | HOURS(days)                | Converts an interval value of type days to an interval value of type hours representing the same time span.   |
|           | HOURS(weeks)               | Converts an interval value of type weeks to an interval value of type hours representing the same time span.  |
|           |                            |                                                                                                               |
| MINUTES   | MINUTES(integral value)    | Converts an integral value to an interval value of type minutes                                               |
|           | MINUTES(minutes)           | None                                                                                                          |

| Function     | Syntax                       | Purpose                                                                                                               |
|--------------|------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| SECONDS      | MINUTES(hours)               | Converts an interval value of type hours to an interval value of type minutes representing the same time span         |
|              | MINUTES(days)                | Converts an interval value of type days to an interval value of type minutes representing the same time span          |
|              | MINUTES(weeks)               | Converts an interval value of type weeks to an interval value of type minutes representing the same time span         |
|              | SECONDS(integral value)      | Converts an integral value to an interval value of type seconds                                                       |
|              | SECONDS(seconds)             | None                                                                                                                  |
|              | SECONDS(minutes)             | Converts an interval value of type minutes to an interval value of type seconds representing the same time span       |
|              | SECONDS(hours)               | Converts an interval value of type hours to an interval value of type seconds representing the same time span         |
|              | SECONDS(days)                | Converts an interval value of type days to an interval value of type seconds representing the same time span          |
|              | SECONDS(weeks)               | Converts an interval value of type weeks to an interval value of type seconds representing the same time span         |
|              |                              |                                                                                                                       |
| MILLISECONDS | MILLISECONDS(integral value) | Converts an integral value to an interval value of type milliseconds                                                  |
|              | MILLISECONDS(milliseconds)   | None                                                                                                                  |
|              | MILLISECONDS(seconds)        | Converts an interval value of type seconds to an interval value of type milliseconds, representing the same time span |
|              | MILLISECONDS(minutes)        | Converts an interval value of type minutes to an interval value of type milliseconds representing the same time span  |
|              | MILLISECONDS(hours)          | Converts an interval value of type hours to an interval value of type milliseconds representing the same time span    |
|              | MILLISECONDS(days)           | Converts an interval value of type days to an interval value of type milliseconds representing the same time span     |
|              | MILLISECONDS(weeks)          | Converts an interval value of type weeks to an interval value of type milliseconds representing the same time span    |
|              |                              |                                                                                                                       |

| Function     | Syntax                       | Purpose                                                                                                                   |
|--------------|------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| MICROSECONDS | MICROSECONDS(integral value) | Converts an integral value to an interval value of type microseconds                                                      |
|              | MICROSECONDS(milliseconds)   | Converts an interval value of type milliseconds to an interval value of type microseconds representing the same time span |
|              | MICROSECONDS(seconds)        | Converts an interval value of type seconds to an interval value of type microseconds representing the same time span      |
|              | MICROSECONDS(minutes)        | Converts an interval value of type minutes to an interval value of type microseconds representing the same time span      |
|              | MICROSECONDS(hours)          | Converts an interval value of type hours to an interval value of type microseconds representing the same time span        |
|              | MICROSECONDS(days)           | Converts an interval value of type days to an interval value of type microseconds representing the same time span         |
|              | MICROSECONDS(weeks)          | Converts an interval value of type weeks to an interval value of type microseconds representing the same time span        |
|              | MICROSECONDS(microseconds)   | None                                                                                                                      |
| NANOSECONDS  | NANOSECONDS(integral value)  | Converts an integral value to an interval value of type nanoseconds                                                       |
|              | NANOSECONDS(microseconds)    | Converts an interval value of type microseconds to an interval value of type nanoseconds representing the same time span  |
|              | NANOSECONDS(milliseconds)    | Converts an interval value of type milliseconds to an interval value of type nanoseconds representing the same time span  |
|              | NANOSECONDS(seconds)         | Converts an interval value of type seconds to an interval value of type nanoseconds representing the same time span       |
|              | NANOSECONDS(minutes)         | Converts an interval value of type minutes to an interval value of type nanoseconds representing the same time span       |
|              | NANOSECONDS(hours)           | Converts an interval value of type hours to an interval value of type nanoseconds representing the same time span         |
|              | NANOSECONDS(days)            | Converts an interval value of type days to an interval value of type nanoseconds representing the same time span          |
|              | NANOSECONDS(nanoseconds)     | None                                                                                                                      |

| Function | Syntax                   | Purpose                                                                                                           |
|----------|--------------------------|-------------------------------------------------------------------------------------------------------------------|
|          | NANOSECONDS(weeks)       | Converts an interval value of type weeks to an interval value of type nanoseconds representing the same time span |
|          | NANOSECONDS(nanoseconds) | None                                                                                                              |

### Implicit Conversions

In certain situations, Ocient® SQL can automatically convert expressions to a specific data type without an explicit SQL statement.

### Cast from Strings

You can cast string literals to any other data type by placing the name of the data type before the string literal.

#### Examples

##### Cast a Double

Cast the value 1.5 from a string to a floating point value.

```
SELECT DOUBLE '1.5';
```

*Output:* 1.5

##### Cast an Array

Cast the array INT[0,1,2,3] from a string to an array of integers.

```
SELECT INT[] 'INT[0,1,2,3]';
```

*Output:* ['0','1','2','3']

##### Cast a Point Value

Cast the point value POINT(0 0) to the geospatial data type POINT. In this case, the cast keyword is ST\_POINT, and the data type is POINT.

```
SELECT ST_POINT 'POINT(0 0)';
```

*Output:* POINT(0.0 0.0)

### Cast from Date and Time Strings

You can cast date and time values to their interval value with the INTERVAL keyword.

#### Retrieve Interval Values from Various Date and Time Strings

Retrieve the interval value 5 from the '5 DAYS' string.

```
SELECT INTERVAL '5 DAYS';
```

*Output:* 5

Retrieve the interval value 2 from the '2 WEEKS' string.

```
SELECT INTERVAL '2 WEEKS';
```

*Output:* 2

Retrieve the interval value 10 from the '10 SECONDS' string.

```
SELECT INTERVAL '10 SECONDS';
```

*Output:* 10

## CAST Functions

These SQL statements allow casting an expression to any other data type.

### CAST

Transforms an expression to any other data type.

This function operates similarly to the [Scalar Data Conversion Functions](#).

#### Syntax

```
CAST[!] expr [AS] data_type
```

| Argument  | DataType | Description                                                                               |
|-----------|----------|-------------------------------------------------------------------------------------------|
| expr      | Any      | A column or value to transform to another data type.                                      |
| data_type | String   | Any of the Ocient-supported data types. For a full list, see <a href="#">Data Types</a> . |

As shown in the syntax examples, CAST can support the optional ! operator.

The ! operator returns a NULL value if the operation cannot cast to the specified data type. This behavior prevents situations where the system would normally raise an error.

### Examples

#### Cast an Integer to Varchar

Transform the value 2 as a VARCHAR string.

```
SELECT CAST(2 AS VARCHAR);
```

*Output:* '2'

#### Cast a Varchar to Date

Transform the '2020-02-02' string to the DATE data type.

```
SELECT CAST('2020-02-02' AS DATE);
```

*Output:* '2020-02-02'

#### Cast an Incompatible Type with the ! Operator

Transform the 'a' string to an integer, or INT data type, using the ! operator.

```
SELECT CAST!('a' AS INT);
```

*Output:* NULL

This operation generates NULL because the database cannot transform this value to an integer. Without adding this operator, the statement raises an error because the database cannot transform the input expression to a numeric type.

### :: Operator

Transforms an expression to any other data type.

This function operates similarly to the [Scalar Data Conversion Functions](#) function.

:: Syntax

```
expr ::[!] data_type
```

| Argument  | Data Type | Description                                                                              |
|-----------|-----------|------------------------------------------------------------------------------------------|
| expr      | Any       | A column or value to transform to another data type.                                     |
| data_type | String    | Any of the Ocint-supported data types. For a full list, see <a href="#">Data Types</a> . |

As shown in the syntax examples, :: can support the optional ! operator.

The ! operator returns a NULL value if the operation cannot cast to the specified data type. This behavior prevents situations where the system would normally raise an error.

## Examples

### Cast an Integer to Float

Transform the '2' string to a floating point number.

```
SELECT '2' :: FLOAT;
```

Output: 2.0

### Cast to an Incompatible Type with the ! Operator

Transform the 'a' string to an integer, or INT data type, using the ! operator.

```
SELECT 'a' ::! INT;
```

Output: NULL

This operation generates NULL because the database cannot transform this value to an integer. Without adding this operator, the statement raises an error because the database cannot transform the input expression to a numeric type.

## Related Links

[Date and Time Functions](#)

[Character and Binary Functions](#)

[Math Functions and Operators](#)

[Formatting Functions](#)

## 10.16 Sorted Aggregate Functions

Aggregate functions compute a single result from a group of rows. These aggregate functions support *DISTINCT* which filters out repeated values, and the standard *ORDER BY* syntax, which dictates the ordering of the aggregation.

### Syntax

```
AGGREGATE([DISTINCT] arg1, arg2, ... [ORDER BY ...])
```

The ORDER BY syntax clause must be after all arguments to the aggregate function.

| Function   | Syntax                      | Purpose                                                                                                                                                                                                  |
|------------|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ARRAY_AGG  | ARRAY_AGG(expr)             | Returns an array containing every row from the expression. <b>Example</b><br>SELECT array_agg(c1 ORDER BY c1 DESC) FROM sys.dummy10<br>Output: int[10,9,8,7,6,5,4,3,2,1]                                 |
| STRING_AGG | STRING_AGG(expr, delimiter) | Returns a string concatenated from every row from the expression. The delimiter argument is optional. <b>Example</b><br>SELECT string_agg(char(c1), ' ' ORDER BY c1 ASC) FROM sys.dummy10<br>Output: "1" |

## Related Links

[Aggregate Functions](#)

## 10.17 Time Zone Functions

The Ocident® cluster system administrator is responsible for keeping the OS time zone database (e.g., on Ubuntu®, `tzdata`) up to date. Outdated time zone database files might cause unexpected results due to IANA rule updates around name changes, DST observance, and UTC offsets for time zones.

The `TIMESTAMP` data type has no associated time zone value. The database administrator tracks the associated time zone with a `TIMESTAMP` column, whether through an associated `time_zone` `VARCHAR` column or other documentation. Ocident recommends storing all `TIMESTAMP` data types in the UTC time zone to avoid ambiguity and using these functions to convert to arbitrary time zones. You can still store `TIMESTAMP` data types associated with an arbitrary time zone and use these functions to convert to UTC and from UTC to another time zone.

### CONVERT\_UTC\_TIMESTAMP\_TO\_LOCAL

The function converts a timestamp from the UTC time zone to a specified local time zone.

This function can take the `timezone` argument as a `VARCHAR` literal, or the argument can be a column. When the argument is a literal, you can arbitrarily capitalize it. When the argument is a column, the values in the column must be capitalized according to the canonical IANA time zone names, such as 'America/Chicago'.

#### Syntax

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(timestamp,timezone)
```

| Argument  | Data Type            | Description                                                                                                                        |
|-----------|----------------------|------------------------------------------------------------------------------------------------------------------------------------|
| timestamp | TIMESTAMP or VARCHAR | Timestamp to convert that you specify as a <code>TIMESTAMP</code> or <code>VARCHAR</code> that you typecast in a timestamp format. |
| timezone  | VARCHAR              | <code>VARCHAR</code> that contains the full name of the target time zone.                                                          |



## Examples

### Convert a Constant Time Zone Value

Convert timestamp 2022-11-06 02:09:00 to the local time in Chicago.

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(Timestamp '2022-11-06 02:09:00', 'America/Chicago');
```

Output

```
CONVERT_UTC_TIMESTAMP_TO_LOCAL(timestamp(('2022-11-06 02:09:00')), ('America/Chicago')) -
----- 2022-11-05 21:09:00.000000000
```

### Convert a Timestamp Value in a Column

Convert the timestamp in the `ts` column to the local time in Chicago using the `test_timezone` table.

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts, 'America/Chicago') FROM test_timezone;
```

Output

```
CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts, ('America/Chicago')) -
-----
2023-01-09 17:23:12.173997444
```

### Convert a Timestamp Value and Time Zone Name in Columns

Convert the timestamp in the `ts` column to the local time specified by the `timezone_name` column using the `test_timezone` table.

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts, timezone_name) FROM test_timezone;
```

Output

```
CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts, timezone_name) -
-----
2023-01-09 17:43:21.193229991
```

### Convert a Timestamp Value Using Other Date and Time Functions

Use the `TO_TIMESTAMP` function to convert a numeric timestamp to the local time in Chicago.

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(TO_TIMESTAMP(1534587890), 'America/Chicago');
```

Output

```
CONVERT_UTC_TIMESTAMP_TO_LOCAL(to_timestamp((1534587890)), ('America/Chicago')) -
----- 2018-08-18 05:24:50.000000000
```

Use a UTC timestamp from the output of the `CONVERT_LOCAL_TIMESTAMP_TO_UTC` function with Chicago time and convert it back to local time in Chicago.

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(CONVERT_LOCAL_TIMESTAMP_TO_UTC(Timestamp '2022-11-09 01:59:00', 'America/Chicago'), 'America/Chicago');
```

Output

```
CONVERT_UTC_TIMESTAMP_TO_LOCAL(CONVERT_LOCAL_TIMESTAMP_TO_UTC(timestamp(('2022-11-09 01:59:00')), ('America/Chicago')), ('America/Chicago')) -
-----
2022-11-09 01:59:00.000000000
```

Extract the hour portion of the local Chicago time using the output of the `TO_TIMESTAMP` function on a numeric timestamp.

```
SELECT HOUR(CONVERT_UTC_TIMESTAMP_TO_LOCAL(TO_TIMESTAMP(1534587890), 'America/Chicago'));
```

Output

```
hour(CONVERT_UTC_TIMESTAMP_TO_LOCAL(to_timestamp((1534587890)), ('America/Chicago'))) -
----- 5
```

### Convert a Timestamp During Table Creation

Create a table using a SELECT statement that converts a timestamp in local Chicago time.

```
CREATE TABLE test_timezone_2 (ts timestamp, b int)
  AS (SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(Timestamp '2022-11-06 02:09:00', 'America/Chicago'),
      1 FROM sys.dummy1);
```

Modified 1 row

```
SELECT * FROM test_timezone_2;          1
```

Output

```
ts_1          b_1 ----- 2022-11-05
21:09:00.000000000
```

### Convert a Timestamp During an Insert Operation

Insert a timestamp in local Chicago time into the test\_timezone\_2 table.

```
INSERT INTO test_timezone_2 SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(Timestamp '2022-11-06 02:09:00', 'America/Chicago'),
  1 FROM sys.dummy1;
```

Modified 1 row

### Convert a Timestamp in a Query Filter

Use timestamp, a numeric constant value in local Chicago time, as a filter on the column ts in the test\_timezone table.

```
SELECT * FROM test_timezone
  WHERE ts=CONVERT_UTC_TIMESTAMP_TO_LOCAL(Timestamp '2022-11-06 14:09:00', 'America/Chicago');
```

Output

```
ts          b ----- 2022-11-06
08:09:00.000000000      1
```

Use timestamp, a runtime value from the table column ts in local Chicago time, as a filter on the test\_timezone table.

```
SELECT * FROM test_timezone
  WHERE CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts::timestamp, 'America/Chicago')='2022-11-06 08:09:00.000000000'
```

Output

```
ts          b ----- 2022-11-06
08:09:00.000000000      1
```

### Convert a Timestamp in the Grouping Part of the Query

Group the query results by a timestamp converted to local time in Chicago.

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts::timestamp, 'America/Chicago')
FROM test_timezone GROUP BY CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts::timestamp, 'America/Chicago');
```

Output

```
CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts, ('America/Chicago')) -----
2022-11-05 04:09:00.000000000 2022-11-09 02:09:00.000000000 2022-11-04 04:09:00.000000000 2022-11-06
02:09:00.000000000
```

Using a grouping, omit the 2022-11-10 02:09:00 timestamp using the HAVING clause.

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts::timestamp, 'America/Chicago')
FROM test_timezone GROUP BY CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts::timestamp, 'America/Chicago')
HAVING CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts::timestamp, 'America/Chicago') != '2022-11-10 02:09:00';
```

Output

```
CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts, ('America/Chicago')) -----
2022-11-06 02:09:00.000000000 2022-11-04 04:09:00.000000000 2022-11-05 04:09:00.000000000
```

## CONVERT\_LOCAL\_TIMESTAMP\_TO\_UTC

The function converts a timestamp in a specified local time zone to the UTC time zone. In the case of ambiguous timestamps, the database returns the earliest of the two possible times in the UTC time zone. Ambiguous timestamps refer to the times when the DST time zone change happens, and 1 hour can transform to either 1 or +1 and 1 or -1. Ambiguous timestamps can also refer to non-existent times such as 2023-03-12 02:01:00 'America/New\_York' which are similarly converted to the earliest possible UTC time, 2023-03-12 07:00:00 'UTC', due to the skipping of hour 02:00:00 entirely with DST spring forward.

This function can take the `timezone` argument as a `VARCHAR` literal, or the argument can be a column. When the argument is a literal, you can arbitrarily capitalize it. When the argument is a column, the values in the column must be capitalized according to the canonical IANA time zone names, such as 'America/Chicago'.

### Syntax

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(timestamp, timezone)
```

| Argument  | Data Type            | Description                                                                                              |
|-----------|----------------------|----------------------------------------------------------------------------------------------------------|
| timestamp | TIMESTAMP or VARCHAR | Timestamp to convert that you specify as a TIMESTAMP or VARCHAR that you typecast in a timestamp format. |
| timezone  | VARCHAR              | VARCHAR that contains the full name of the target time zone.                                             |

### Examples

#### Convert a Constant Time Zone Value

Convert timestamp 2022-11-06 02:09:00 to Chicago UTC time.

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(Timestamp '2022-11-06 02:09:00', 'America/Chicago');
```

Output

```
CONVERT_LOCAL_TIMESTAMP_TO_UTC(timestamp(('2022-11-06 02:09:00')), ('America/Chicago')) -
2022-11-06 08:09:00.000000000
```

### Convert a Timestamp Value in a Column

Convert the timestamp in the `ts` column to Chicago UTC time using the `test_timezone` table.

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts, 'America/Chicago') FROM test_timezone;
```

Output

```
CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts, ('America/Chicago')) -
2023-01-09 17:23:12.173997444
```

### Convert a Timestamp Value and Time Zone Name in Columns

Convert the timestamp in the `ts` column to the UTC time specified by the `timezone_name` column using the `test_timezone` table.

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts, timezone_name) FROM test_timezone;
```

Output

```
CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts, timezone_name) - 2023-
01-09 17:43:21.193229991
```

### Convert a Timestamp Value Using Other Date and Time Functions

Use the `TO_TIMESTAMP` function to convert a numeric timestamp to the Chicago UTC time.

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(TO_TIMESTAMP(1534587890), 'America/Chicago');
```

Output

```
CONVERT_LOCAL_TIMESTAMP_TO_UTC(to_timestamp((1534587890)), ('America/Chicago')) -
2018-08-18 15:24:50.000000000
```

Use a local timestamp from the output of the `CONVERT_UTC_TIMESTAMP_TO_LOCAL` function with Chicago time and convert it back to UTC time.

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(TO_TIMESTAMP(CONVERT_UTC_TIMESTAMP_TO_LOCAL(Timestamp '2022-11-06 01:59:00.000000000', 'America/Chicago')));
```

Output

```
CONVERT_LOCAL_TIMESTAMP_TO_UTC(CONVERT_UTC_TIMESTAMP_TO_LOCAL(timestamp(('2022-
11-06 01:59:00')), ('America/Chicago')), ('America/Chicago')) -
2022-11-06 01:59:00.000000000
```

### Convert a Timestamp During Table Creation

Create a table using a `SELECT` statement that converts a timestamp in Chicago UTC time.

```
CREATE TABLE test_timezone (ts timestamp, b int)
AS (SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(Timestamp '2022-11-09 02:09:00', 'America/Chicago'),
1 FROM sys.dummy1);
```

Modified 1 row

```
SELECT * FROM test_timezone;
```

Output

```
ts                                b                                2022-11-09
08:09:00.000000000                1
```

## Convert a Timestamp During an Insert Operation

Insert a timestamp in Chicago UTC time into the `test_timezone` table.

```
INSERT INTO test_timezone SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(Timestamp '2022-11-06 02:09:00', 'America/Chicago')
  1 from sys.dummy1;
```

Modified 1 row

## Convert a Timestamp in a Query Filter

Use a UTC timestamp in Chicago as a numeric constant value to filter on the column `ts` in the `test_timezone` table.

```
SELECT * FROM test_timezone
  WHERE ts=CONVERT_LOCAL_TIMESTAMP_TO_UTC(Timestamp '2022-11-06 02:09:00', 'America/Chicago');
```

Output

| ts                 | b |            |
|--------------------|---|------------|
| 08:09:00.000000000 | 1 | 2022-11-06 |

Use timestamp in Chicago UTC time, a runtime value from the table column `ts`, as a filter on the `test_timezone` table.

```
SELECT * FROM test_timezone
  WHERE CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts::timestamp, 'America/Chicago')='2022-11-06 08:09:00.000000000'
```

Output

| ts                 | b |            |
|--------------------|---|------------|
| 08:09:00.000000000 | 1 | 2022-11-06 |

## Convert a Timestamp in the Grouping Part of the Query

Group the query results by a timestamp converted to UTC time in Chicago.

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts::timestamp, 'America/Chicago')
  FROM test_timezone GROUP BY CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts::timestamp, 'America/Chicago');
```

Output

| CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts, ('America/Chicago'))                                                                 |
|-------------------------------------------------------------------------------------------------------------------------|
| 2022-11-09 14:09:00.000000000 2022-11-06 14:09:00.000000000 2022-11-04 14:09:00.000000000 2022-11-05 14:09:00.000000000 |

Using a grouping, omit the 2022-11-05 14:09:00 timestamp using the `HAVING` clause.

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts::timestamp, 'America/Chicago')
  FROM test_timezone GROUP BY CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts::timestamp, 'America/Chicago')
  HAVING CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts::timestamp, 'America/Chicago') != '2022-11-05 14:09:00';
```

Output

| CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts, ('America/Chicago'))                                   |
|-------------------------------------------------------------------------------------------|
| 2022-11-09 14:09:00.000000000 2022-11-06 14:09:00.000000000 2022-11-04 14:09:00.000000000 |

## Related Links

[General SQL Syntax](#)

[Date and Time Functions](#)

[Conversion Functions](#)

[Database, Tables, Views, and Indexes](#)

[Database, Tables, Views, and Indexes](#)

## 10.18 Tuple Functions and Operators

### Tuple Functions

| Function                | Syntax                                                       | Purpose                                                                                                                                   |
|-------------------------|--------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| Constructor             | <code>tuple«TYPE1,TYPE2, ..., TYPEn»(e1, e2, ..., en)</code> | Construct a tuple given elements. NULL is also supported as an element.                                                                   |
| Implicit Constructor    | <code>tuple(e1, e2, ..., en)</code>                          | Construct a tuple with the specified elements. Types of the tuple are inferred from the inner elements.                                   |
| convert tuple to string | <code>char(tuple)</code>                                     | Converts a tuple to its string representation.                                                                                            |
| convert string to tuple | <code>string_to_tuple(str, format)</code>                    | Converts the string representation of a tuple (e.g., 'tuple«INT,BIGINT,CHAR»(1,2,NULL)') into a tuple.                                    |
| tuple casting           | <code>cast_to_tuple(tuple, format)</code>                    | Converts a tuple into another tuple of a different type (e.g., <code>cast_to_tuple(tuple«int,int»(1,2), 'TUPLE(BIGINT,DOUBLE)')</code> ). |

### Tuple Function Examples

| Function                             | SQL statement                                                               | Result                                 |
|--------------------------------------|-----------------------------------------------------------------------------|----------------------------------------|
| type constructor                     | <code>tuple«INT,BIGINT»(1,2)</code>                                         | tuple holding INT(1) and BIGINT(2)     |
| type constructor for array of tuples | <code>tuple«INT,BIGINT»</code>                                              | array holding 2 tuples: [(1,2), (1,2)] |
| <code>char()</code>                  | <code>char(tuple«INT,BIGINT»(1,2))</code>                                   | 'tuple«INT,BIGINT»(1,2)'               |
| <code>string_to_array</code>         | <code>string_to_array('tuple«INT,BIGINT»(1,2)', 'TUPLE«INT,BIGINT»')</code> | [tuple«INT,BIGINT»(1,2)]               |

### Tuple Operators

| Operator        | Syntax                                     | Description                                                                                                             |
|-----------------|--------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| access operator | <code>tuple[element_index]</code>          | Returns the element stored in the specified index position. The index starts at 1. Access out of bounds returns a NULL. |
| slice operator  | <code>tuple[left_index:right_index]</code> | Returns the subtuple ranging from left_index to right_index inclusive. The index starts at 1.                           |

### About the Tuple Slice Operator

- Each index starts at 1.
- If the tuple is NULL or either index value is NULL, the result of the slicing is NULL.

- If you omit the left index, it is equivalent to using 1. If you omit the right index, it is equivalent to `len(tuple)`. If you omit both, it returns the full tuple. `tuple<<int,int,int>>(1, 2, 3)[:]` is equivalent to `tuple<<int,int,int>>(1, 2, 3)`.
- Values below zero for the left index convert to 1, and values beyond the tuple length for the right index convert to `len(tuple)`. `tupleVal[-1:4]` is equivalent to `tupleVal[1:4]`, and `tuple<<int,int,int>>(1, 2, 3)[1:6]` is equivalent to `tuple<<int,int,int>>(1, 2, 3)[1:3]`.
- Ranges completely out of tuple bounds throw an invalid argument error, e.g., `tuple<<int,int>>(1,2)[4:5]`.
- Sequential slices slice each dimension of tuples. `tuple<<tuple<<int,int,int>>,tuple<<int,int,int>>>>(tuple<<int,int,int>>(1, 2, 3), tuple<<int,int,int>>(4, 5, 6))[1:1][2:3] = tuple<<tuple<<int,int>>>>(tuple<<int,int>>>(2, 3))`
- More than N sequential slices of an N-dimensional tuple, for example, three sequential slices on a two-dimensional tuple throw an error. `tuple<<int>>(1)[:][:]`
- You cannot combine slicing with the access operator. If you attempt such a combination, any access operator `[n]` converts to `[:n]`. `tupleVal[4][1:6] = tupleVal[:4][1:6]`.
- Slicing with non-constant indexes throws an error.
- Slicing a tuple of arrays such as `tuple<<int[],double[]>>(int[1,2,3],double[4,5,6])[1:1][2:3]` slices `[1:1]` on the tuple and `[2:3]` on the resulting inner arrays, resulting in `tuple<<int[]>>(int[2,3])`.
- Slicing a tuple with a mix of containers and primitives such as `tuple<<int,int[]>>(1,int[1,2,3])` throws an error if, at any given depth, the slice is not valid for one or more elements. Slicing the tuple with `[:][2:3]` fails because you cannot slice an integer, and the resultant tuple from `[:]` includes an integer. However, slicing with `[2:2][2:3]` returns `tuple<<int[]>>(int[2,3])` because the tuple resulting from the `[2:2]` slice contains only elements that the next slice range can slice.

## Operator Examples

| Operator        | SQL Predicate                                          | Result                                   |
|-----------------|--------------------------------------------------------|------------------------------------------|
| access operator | <code>tuple«CHAR,BIGINT»('SQL',2)[1]</code>            | 'SQL'                                    |
| access operator | <code>tuple«CHAR,BIGINT»('SQL',2)[2]</code>            | NULL                                     |
| slice operator  | <code>tuple«CHAR,BIGINT,BIGINT»('SQL',1,2)[1:2]</code> | <code>tuple«CHAR,BIGINT»('SQL',1)</code> |

## Related Links

[Array Functions and Operators](#)

[Matrix Functions and Operators](#)

[Math Functions and Operators](#)

[Data Types](#)

[Query Ocient](#)

## 10.19 Window Aggregate Functions

Window functions operate on a set of rows and return a single value for each row from the underlying query. Like aggregate functions, window functions operate on a set of rows, but they do not reduce the number of rows returned by the query.

Unlike regular aggregate functions, window functions do not group rows into a single output row. Instead, the rows retain their separate identities, and the window function produces aggregate calculations that access more than just the current row in the query result.

The term window describes the set of rows on which the window function operates. A window function returns a value from the rows in a window.

The database uses the `ORDER BY` and `PARTITION BY` clauses to define the scope of the window.

### Syntax

Window Aggregate functions follow this syntax with `window_function` representing any of the functions listed in [Window Aggregate Functions](#).

```

window_function( arg1, arg2 [,...] [ IGNORE NULLS ] )
  OVER (
    [ PARTITION BY part_list ]
    [ ORDER BY order_list [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] ]
    [ <ROWS_or_RANGE_clause> ] )

<ROWS_or_RANGE_clause> ::=
{ ROWS | RANGE }
  { <window_frame_preceding>
  | BETWEEN { <window_frame_preceding> | <window_frame_following> }
  AND { <window_frame_preceding> | <window_frame_following> }
  }

<window_frame_preceding> ::=
{ UNBOUNDED PRECEDING | row_integer PRECEDING | CURRENT ROW }

<window_frame_following> ::=
{ UNBOUNDED FOLLOWING | row_integer FOLLOWING | CURRENT ROW }

```

| Parameter         | Data Type | Description                                                                                |
|-------------------|-----------|--------------------------------------------------------------------------------------------|
| arg1, arg2 [,...] | String    | The arguments of the window function. The arguments depend on the type of window function. |
| part_list         | String    | Specifies the column by which the rows are partitioned for the window function.            |
| order_list        | String    | Specifies the column by which the rows are ordered.                                        |
| row_integer       | Numeric   | An integer representing the number of rows to operate from the current row.                |

## Window Aggregate Keywords

Window aggregate functions use these clauses and keywords to designate the parameters for the query and output.

### PARTITION BY

This clause defines a set of rows within a result set. **PARTITION BY** operates similarly to a [General SQL Syntax](#) clause, except it does not reduce the number of rows in the result set.

### ORDER BY

Sorts the result set in ascending or descending order based on one or more specified columns.

An **ORDER BY** clause must be included on all window functions to ensure your query results are deterministic. Some window functions require an **ORDER BY** clause. See the syntax for each function for specifics. For details, see [General SQL Syntax](#).

### ASC or DESC

Specifies whether the values in the specified column should be sorted in **ASC** ascending or **DESC** descending order. **ASC** is the default sort order.



## NULLS FIRST, NULLS LAST, or IGNORE NULLS

Specifies if the database sorts NULL rows first **NULLS FIRST**, last **NULLS LAST**, or whether the database ignores NULLs **IGNORE NULLS**. The default is **NULLS LAST**.

## ROWS or RANGE

Limit the rows within the partition by specifying start and end points within the partition. Specify a range of rows with respect to the current row by logical association using **RANGE** or physical association using **ROWS**.

The **RANGE** clause logically limits the rows within a partition by specifying a range of values with respect to the value in the current row. The system defines preceding or following rows based on the ordering in the **ORDER BY** clause. If you do not specify this clause, the default is ascending order.

The **ROWS** clause limits the rows within a partition by specifying a fixed number of rows preceding or following the current row.

Specifying a window frame by the **RANGE** keyword with the current row **CURRENT ROW** keyword includes all rows with the same values as the current row in the **ORDER BY** clause.

**ROWS** and **RANGE** clauses can include these options.

| Keyword Clause               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UNBOUNDED PRECEDING</b>   | Specifies that the window starts at the first row of the partition. This clause specifies a window starting point only.                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>UNBOUNDED FOLLOWING</b>   | Specifies that the window ends at the last row of the partition. This clause specifies a window endpoint only. For example: <b>RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING</b> This window frame starts with the current row and ends with the last row of the partition.                                                                                                                                                                                                                                                |
| <b>row_integer PRECEDING</b> | The <b>row_integer</b> value specifies the number of rows or values to precede the current row. You cannot use the <b>PRECEDING</b> keyword in a <b>RANGE</b> clause.                                                                                                                                                                                                                                                                                                                                                          |
| <b>row_integer FOLLOWING</b> | The <b>row_integer</b> value specifies the number of rows or values to follow the current row. When you specify the <b>FOLLOWING</b> keyword as the window starting point, the endpoint must also use a <b>FOLLOWING</b> clause with a row value. For example: <b>ROWS BETWEEN 2 FOLLOWING AND 10 FOLLOWING</b> This window frame starts with the second row following the current row and ends with the tenth row following the current row. You cannot use the <b>FOLLOWING</b> keyword in a <b>RANGE</b> clause.            |
| <b>CURRENT ROW</b>           | When you specify the <b>CURRENT ROW</b> keyword in a <b>ROWS</b> clause, the window frame starts or ends at the current row. When you specify this keyword in a <b>RANGE</b> clause, the window frame includes all rows with equal values. For example: <b>ROWS BETWEEN 2 PRECEDING AND CURRENT ROW</b> This clause means that the window function operates on a frame of three rows in size, where the first two rows precede the current row (inclusive). You can specify the current row as either a starting or end point. |

## Supported Window Aggregate Functions

These functions are supported for window frames. Some examples use the `sys.dummy` virtual table to populate data. For details, see [Generate Tables Using sys.dummy](#).

You cannot use the `DISTINCT` keyword with window aggregation functions.

## CUME\_DIST

Returns a positive value less than one that represents a percentage of values that are less than or equal to the current value in the group.

### Syntax

```
CUME_DIST (
    OVER ( [ PARTITION BY part_list ] ORDER BY order_list )
```

| Parameter  | Data Type | Description                                                                     |
|------------|-----------|---------------------------------------------------------------------------------|
| part_list  | Any       | Specifies the column by which the rows are partitioned for the window function. |
| order_list | Any       | Specifies the column by which the rows are ordered for the window function.     |

### Example

This example shows the cumulative distribution values for a table with generated numbers from one to 10.

```
SELECT x, CUME_DIST()
    OVER (ORDER BY x) AS d
FROM (SELECT c1 AS x FROM sys.dummy10);
```

#### Output

```
| "x" | "d" |
|-----|-----|
| 1 | 0.1 |
| 2 | 0.2 |
| 3 | 0.3 |
| 4 | 0.4 |
| 5 | 0.5 |
| 6 | 0.6 |
| 7 | 0.7 |
| 8 | 0.8 |
| 9 | 0.9 |
| 10 | 1.0 |
```

## DELTA

Computes the finite difference between successive values of expression under the specified ordering. This is a backward difference, meaning the delta value for a specified row is the difference between the value of that row and the previous row.

The function requires two rows of input data to calculate a first-degree difference, three rows of input data to calculate a second-degree difference, and so on.

However, rather than returning NULL for the first few rows, the function returns the correct values, such that using DELTA of degree `n` followed by DELTA of degree `-n` cancel each other out to return the initial result (other than

floating point error accumulation). In general, using this window function over larger result sets with large degrees worsens floating point error accumulation.

Integer degree values in the range  $[-3, 3]$  are generally stable up to approximately 5 million rows. Fractional degree values in the range  $[-3, 3]$  are generally stable up to approximately 1 million rows. The stable degree range is wider over smaller result sets and narrower over larger result sets. For example, fractional and integer degrees are stable over the range  $[-4, 4]$  up to approximately 80,000 rows.

The function treats any NULL values in the input data as zero, which might cause unexpected results. Filter out NULL values before you use the window function.

### Syntax

```
DELTA ( expression [, degree ])
  OVER (
    [ PARTITION BY expr_list ]
    ORDER BY order_list )
```

| Parameter  | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                   |
|------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| expression | Numeric   | The column or expression to use for the DELTA calculation.                                                                                                                                                                                                                                                                                                                                    |
| degree     | Numeric   | Optional. If degree is set to a positive integer greater than 1, the window function calculates the higher-order finite difference without using nested DELTA calls. If negative, the calculation is an anti-difference (i.e., discrete integration). You can specify fractional values, either positive or negative, and these values mean fractional differences or sums. The default is 1. |
| part_list  | Any       | Specifies the column by which the rows are partitioned for the window function.                                                                                                                                                                                                                                                                                                               |
| order_list | Any       | Specifies the column by which the rows are ordered for the window function.                                                                                                                                                                                                                                                                                                                   |

### Examples

#### Calculate Difference Values Using Basic Numbers

This example shows the delta values for a table with generated numbers from one to 10.

```
SELECT y, DELTA(y, 1)
  OVER (ORDER BY y) AS d
FROM (SELECT DOUBLE(c1 * c1) AS y FROM sys.dummy10);
```

#### Output

```
| "y" | "d" |
|-----|-----|
| 1   | 1   |
| 4   | 3   |
| 9   | 5   |
```

```
| 16 | 7 |
| 25 | 9 |
| 36 | 11 |
| 49 | 13 |
| 64 | 15 |
| 81 | 17 |
| 100 | 19 |
```

### Calculate Difference Values Using Partitioned Sales Data

This example uses the `DELTA` function with a table of regional sales data to find the difference in sales by region and year.

```
SELECT
    region,
    year,
    sales_amount,
    DELTA(sales_amount) OVER (PARTITION BY region ORDER BY year) AS sales_delta
FROM
    regional_sales
ORDER BY
    region,
    year;
```

#### Output

| "region" | "year" | "sales_amount" | "sales_delta" |
|----------|--------|----------------|---------------|
| North    | 2018   | 12000          | 12000         |
| North    | 2019   | 15000          | 3000          |
| North    | 2020   | 17000          | 2000          |
| North    | 2021   | 16000          | -1000         |
| North    | 2022   | 18000          | 2000          |
| South    | 2018   | 10000          | 10000         |
| South    | 2019   | 14000          | 4000          |
| South    | 2020   | 15000          | 1000          |
| South    | 2021   | 17000          | 2000          |
| South    | 2022   | 16000          | -1000         |

### DENSE\_RANK

Assigns a rank to each row in the result set with equal values having the same number. There are no gaps between ranks.

#### Syntax

```
DENSE_RANK (
    OVER ( [ PARTITION BY part_list ] ORDER BY expr_list )
```

| Parameter  | Data Type | Description                                                                     |
|------------|-----------|---------------------------------------------------------------------------------|
| part_list  | Any       | Specifies the column by which the rows are partitioned for the window function. |
| order_list | Any       | Specifies the column by which the rows are ordered for the window function.     |

## Example

This example generates the rank for a series of ten random numbers. Use the [Math Functions and Operators](#) function to generate the random numbers.

```
WITH RandomNumbers AS (
    SELECT RAND() AS num
    FROM sys.dummy10
)
SELECT num, DENSE_RANK()
    OVER (ORDER BY num) AS n
FROM RandomNumbers;
```

### Output

| num                  | n  |
|----------------------|----|
| 0.021744504292815926 | 1  |
| 0.11531389663905277  | 2  |
| 0.17818297230968436  | 3  |
| 0.3009914646699445   | 4  |
| 0.366713904443239    | 5  |
| 0.7801256114813641   | 6  |
| 0.8735727062571587   | 7  |
| 0.8969752860565523   | 8  |
| 0.9679736914744647   | 9  |
| 0.9910291449541664   | 10 |

## DERIVATIVE

The function calculates the difference between successive values of two numeric expressions, representing a backward difference (e.g., the difference between the current and previous row). It can be considered as computing the derivative of one expression with respect to another.

The function must use an **ORDER BY** clause, and it cannot use a **PARTITION BY** clause.

**DERIVATIVE** requires two arguments, **expression** and **expression2**, which must be included in the **ORDER BY** clause.

The function computes the difference in **expression** values between consecutive rows and divides this by the difference in **expression2** values over the same rows. If **expression** represents some measured value and **expression2** is time, the result is the rate of change (derivative) of the value with respect to time.

The function can compute higher-degree derivatives (second, third, etc.). To calculate the **n**th-degree derivative, it uses a backward difference formula that compares the current row to the previous **n** rows.

For example:

- A second-degree derivative requires three rows of data: the current row, the previous row, and the one before the previous row.
- A third-degree derivative would require four rows, and so on.

Unlike some functions that return NULL for rows that lack sufficient preceding data, the **DERIVATIVE** function is designed to return correct values even for initial rows. This behavior ensures that if you apply a derivative of degree **n** followed by its inverse (derivative of degree **-n**), you recover the original values with minimal error (except for floating point inaccuracies).

In general, using this function over larger result sets with larger degrees worsens the floating point error. Integer degree values in the range of three are generally stable up to approximately 5 million rows. Fractional degree values in the range of three are generally stable up to approximately 1 million rows. The stable degree range is wider over smaller result sets and narrower over larger result sets. For example, fractional and integer degrees are stable over the range **[-4, 4]** up to approximately 80,000 rows.

For higher-order calls, the sample points for **expression** must be equally spaced, or the function might return incorrect results. In other words, the DELTA executions between consecutive **expression2** values must always be the same. If this is not true, use some form of interpolation to create an evenly-spaced series of samples before you use the window function.

The function treats any NULL values in the input data for either **expression** or **expression2** arguments as zero, which might cause unexpected results. Filter out NULL values before you use the window function.

### Syntax

```
DERIVATIVE ( expression [, degree ] )
  OVER ( ORDER BY expression2 )
```

| Parameter   | Data Type | Description                                                                                                                                                                                                                                                                                                                                          |
|-------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| expression  | Numeric   | The column or expression to use for the DERIVATIVE calculation compared to expression2.                                                                                                                                                                                                                                                              |
| expression2 | Numeric   | The column or expression to use for the DERIVATIVE calculation compared to expression.                                                                                                                                                                                                                                                               |
| degree      | Numeric   | Optional. If you set degree to a positive integer greater than one, the window function calculates the higher-order difference quotients (discrete derivatives) without using nested DERIVATIVE executions. If the degree argument is a negative integer, the calculation is a discrete anti-differentiation or integration. The default value is 1. |

### Examples

#### Calculate the Derivative Using Basic Numbers

This example shows the derivative values for a basic table with values x and y.

```
SELECT x, y, DERIVATIVE(y, 1)
  OVER (ORDER BY x) AS d
FROM (SELECT DOUBLE(2 * x + 1) AS y, x
      FROM (SELECT c1/2.0 AS x FROM sys.dummy10));
```

#### Output

```
| "x" | "y" | "d" |
|----|-----|-----|
| 0.5 | 2.0 | 4.0 |
| 1.0 | 3.0 | 2.0 |
| 1.5 | 4.0 | 2.0 |
| 2.0 | 5.0 | 2.0 |
| 2.5 | 6.0 | 2.0 |
| 3.0 | 7.0 | 2.0 |
| 3.5 | 8.0 | 2.0 |
| 4.0 | 9.0 | 2.0 |
```

```
| 4.5 | 10.0 | 2.0 |
| 5.0 | 11.0 | 2.0 |
```

### Calculate the Antiderivative Using a Negative Degree

This example uses a negative degree to produce an antiderivative.

```
SELECT x, y, DERIVATIVE(y, -1)
      OVER (ORDER BY x) AS d
      FROM (SELECT DOUBLE(2 * x + 1) AS y, x
            FROM (SELECT c1/2.0 AS x FROM sys.dummy10));
```

*Output*

```
| "x" | "y" | "d" |
|-----|-----|-----|
| 0.5 | 2.0 | 1.0 |
| 1.0 | 3.0 | 2.5 |
| 1.5 | 4.0 | 4.5 |
| 2.0 | 5.0 | 7.0 |
| 2.5 | 6.0 | 10.0 |
| 3.0 | 7.0 | 13.5 |
| 3.5 | 8.0 | 17.5 |
| 4.0 | 9.0 | 22.0 |
| 4.5 | 10.0 | 27.0 |
| 5.0 | 11.0 | 32.5 |
```

### FIRST\_VALUE

Returns the first value in the ordered result set.

#### Syntax

```
FIRST_VALUE ( expression )
      OVER ( [ PARTITION BY p_list ] [ ORDER BY order_list ] )
```

| Parameter  | Data Type | Description                                                                     |
|------------|-----------|---------------------------------------------------------------------------------|
| expression | Any       | The column or expression value to use for the <b>FIRST_VALUE</b> calculation.   |
| part_list  | Any       | Specifies the column by which the rows are partitioned for the window function. |
| order_list | Any       | Specifies the column by which the rows are ordered for the window function.     |

### Example

This example uses a table with generated numbers from one to 10. Of this group, the function determines the first value from the ordering.

```
SELECT y, FIRST_VALUE(y)
      OVER (ORDER BY y) AS first_value
      FROM (SELECT c1 AS y FROM sys.dummy10);
```

*Output*

| "y" | "first_value" |
|-----|---------------|
| 1   | 1             |
| 2   | 1             |
| 3   | 1             |
| 4   | 1             |
| 5   | 1             |
| 6   | 1             |
| 7   | 1             |
| 8   | 1             |
| 9   | 1             |
| 10  | 1             |

## LAG

Returns a row backward from the current row (one row back unless you specify a different number value).

### Syntax

```
LAG (value_expr [, offset ])
    OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

| Parameter  | Data Type | Description                                                                                                                    |
|------------|-----------|--------------------------------------------------------------------------------------------------------------------------------|
| value_expr | Any       | The column or expression to use for the LAG calculation.                                                                       |
| offset     | Numeric   | Optional. This number determines how many rows back from the current row from which to return a value. The default value is 1. |
| part_list  | Any       | The column or expression value to use for the LAG calculation.                                                                 |
| order_list | Any       | Specifies the column by which the rows are ordered for the window function.                                                    |

## Examples

### Calculate Lag Using Basic Numbers

This example uses a table with generated numbers from one to 10. Of this group, the LAG function returns the last value for the specified column y.

```
SELECT y, LAG(y)
    OVER (ORDER BY y) AS wf
FROM (SELECT c1 AS y FROM sys.dummy10);
```

### Output

| "y" | "wf" |
|-----|------|
| 1   |      |
| 2   | 1    |
| 3   | 2    |
| 4   | 3    |
| 5   | 4    |
| 6   | 5    |
| 7   | 6    |



|  |    |  |   |  |
|--|----|--|---|--|
|  | 8  |  | 7 |  |
|  | 9  |  | 8 |  |
|  | 10 |  | 9 |  |

Calculate Lag Using an Offset

In this example, the offset value 5 changes the number of rows back from the current row from which to return a value.

```
SELECT y, LAG(y, 5)
      OVER (ORDER BY y) AS wf
FROM (SELECT c1 AS y FROM sys.dummy10);
```

Output

|  |       |  |       |  |
|--|-------|--|-------|--|
|  | "y"   |  | "wf"  |  |
|  | ----- |  | ----- |  |
|  | 1     |  |       |  |
|  | 2     |  |       |  |
|  | 3     |  |       |  |
|  | 4     |  |       |  |
|  | 5     |  |       |  |
|  | 6     |  | 1     |  |
|  | 7     |  | 2     |  |
|  | 8     |  | 3     |  |
|  | 9     |  | 4     |  |
|  | 10    |  | 5     |  |

LAST\_VALUE

Returns the last value in the ordered result set.

Syntax

```
LAST_VALUE ( expression )
      OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

| Parameter  | Data Type | Description                                                                     |
|------------|-----------|---------------------------------------------------------------------------------|
| expression | Any       | The column or expression value to use for theLAST_VALUEcalculation.             |
| part_list  | Any       | Specifies the column by which the rows are partitioned for the window function. |
| order_list | Any       | Specifies the column by which the rows are ordered for the window function.     |

Example

This example uses a table with generated numbers from one to 10. Of this group, the function determines the last value from the ordering.

```
SELECT y, LAST_VALUE(y)
      OVER (ORDER BY y) AS last_value
FROM (SELECT c1 AS y FROM sys.dummy10);
```

Output

| "y" | "last_value" |
|-----|--------------|
| 1   | 1            |
| 2   | 2            |
| 3   | 3            |
| 4   | 4            |
| 5   | 5            |
| 6   | 6            |
| 7   | 7            |
| 8   | 8            |
| 9   | 9            |
| 10  | 10           |

## LEAD

Returns a row forward from the current row (one row forward unless you specify a different number value).

### Syntax

```
LEAD (value_expr [, offset ])
    OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

| Parameter  | Data Type | Description                                                                                                            |
|------------|-----------|------------------------------------------------------------------------------------------------------------------------|
| value_expr | Any       | The value to use for the LEAD calculation.                                                                             |
| offset     | Numeric   | Optional. This number determines how many rows forward from the current row to return a value. The default value is 1. |
| part_list  | Any       | Specifies the column by which the rows are partitioned for the window function.                                        |
| order_list | Any       | Specifies the column by which the rows are ordered for the window function.                                            |

## Examples

### Calculate the Lead Value Using Basic Numbers

This example uses a table with generated numbers from one to 10. Of this group, the LEAD function returns the last value for the specified column y.

```
SELECT y, LEAD(y)
    OVER (ORDER BY y) AS lead_value
FROM (SELECT c1 AS y FROM sys.dummy10);
```

### Output

| "y" | "lead_value" |
|-----|--------------|
| 1   | 2            |
| 2   | 3            |
| 3   | 4            |
| 4   | 5            |
| 5   | 6            |
| 6   | 7            |

|    |    |
|----|----|
| 7  | 8  |
| 8  | 9  |
| 9  | 10 |
| 10 |    |

### Calculate the Lead Value Using an Offset

In this example, using a table of generated numbers with values from one to 10, specify the offset value 5 in the LEAD function to change the number of rows forward from the current row to return a value.

```
SELECT y, LEAD(y, 5)
      OVER (ORDER BY y) AS lead_value
FROM (SELECT c1 AS y FROM sys.dummy10);
```

*Output*

|       |              |
|-------|--------------|
| "y"   | "lead_value" |
| ----- | -----        |
| 1     | 6            |
| 2     | 7            |
| 3     | 8            |
| 4     | 9            |
| 5     | 10           |
| 6     |              |
| 7     |              |
| 8     |              |
| 9     |              |
| 10    |              |

### NTH\_VALUE

Returns the *n*-th value in the ordered result set.

#### Syntax

```
NTH_VALUE (expr, offset)
      OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

| Parameter  | Data Type | Description                                                                                                                 |
|------------|-----------|-----------------------------------------------------------------------------------------------------------------------------|
| expr       | Any       | The column or expression from which to retrieve a value.                                                                    |
| offset     | Numeric   | A positive integer that indicates the row number, starting from the first row in the window. This integer must be positive. |
| part_list  | Any       | Specifies the column by which the rows are partitioned for the window function.                                             |
| order_list | Any       | Specifies the column by which the rows are ordered for the window function.                                                 |

### Examples

#### Calculate the Fifth Value Using Bounded Rows

This example uses a table with generated numbers from one to 10. Of this group, the NTH\_VALUE function returns the fifth value for the specified column y.

For the first four rows, the window contains fewer than five rows, so the function returns NULL because the fifth value is not yet available.

```
SELECT y, NTH_VALUE(y, 5)
       OVER (ORDER BY y) AS nth
FROM (SELECT c1 AS y FROM sys.dummy10);
```

*Output*

| "y" | "nth" |
|-----|-------|
| 1   |       |
| 2   |       |
| 3   |       |
| 4   |       |
| 5   | 5     |
| 6   | 5     |
| 7   | 5     |
| 8   | 5     |
| 9   | 5     |
| 10  | 5     |

### Calculate the Fifth Value Using Unbounded Rows

To include future rows in the window frame, you can modify the window frame as shown in this example.

```
SELECT y, NTH_VALUE(y, 5)
       OVER ( ORDER BY y ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS nth
FROM
  (SELECT c1 AS y FROM sys.dummy10);
```

*Output*

| "y" | "nth" |
|-----|-------|
| 1   | 5     |
| 2   | 5     |
| 3   | 5     |
| 4   | 5     |
| 5   | 5     |
| 6   | 5     |
| 7   | 5     |
| 8   | 5     |
| 9   | 5     |
| 10  | 5     |

## PERCENTILE

Returns the value corresponding to the specified percentile ( $0 \leq n \leq 1$ ) within the group.

The set of values is treated as a continuous distribution, so it is possible that the computed value cannot appear in the result set.

### Syntax

```
PERCENTILE (expr, percentile)
  OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

| Parameter               | Data Type | Description                                                                                                                    |
|-------------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------|
| <code>expr</code>       | Any       | The column or expression to use for the <code>PERCENTILE</code> calculation.                                                   |
| <code>percentile</code> | Float     | A percentile ( $0 \leq n \leq 1$ ) to determine the value to return by its position within the ordering of <code>expr</code> . |
| <code>part_list</code>  | Any       | Specifies the column by which the rows are partitioned for the window function.                                                |
| <code>order_list</code> | Any       | Specifies the column by which the rows are ordered for the window function.                                                    |

### Example

This example finds the value at approximately 90 percent of the range of the expression values. For the source data, use a table of generated numbers from one to 10.

The value returns 9.1 because the window column starts at 1. It would be 9 if the column started at 0.

```
SELECT y, PERCENTILE(y, 0.9)
      OVER (ORDER BY y) AS percentile_value
FROM (SELECT c1 AS y FROM sys.dummy10);
```

#### Output

| "y" | "percentile_value" |
|-----|--------------------|
| 1   | 9.1                |
| 2   | 9.1                |
| 3   | 9.1                |
| 4   | 9.1                |
| 5   | 9.1                |
| 6   | 9.1                |
| 7   | 9.1                |
| 8   | 9.1                |
| 9   | 9.1                |
| 10  | 9.1                |

### PERCENT\_RANK

Returns a percentage value for each row representing the percentage of values less than the current group, excluding the highest value. The highest value in a group is always 1.

#### Syntax

```
PERCENT_RANK ()
      OVER ( [ PARTITION BY expr_list ] ORDER BY order_list )
```

### Example

This example generates the percent rank of the table with ten generated numbers from one to 10.

```
SELECT y, PERCENT_RANK()
      OVER (ORDER BY y) AS percent_rank
FROM (SELECT c1 AS y FROM sys.dummy10);
```

#### Output

| "y" | "percent_rank"     |
|-----|--------------------|
| 1   | 0.0                |
| 2   | 0.1111111111111111 |
| 3   | 0.2222222222222222 |
| 4   | 0.3333333333333333 |
| 5   | 0.4444444444444444 |
| 6   | 0.5555555555555556 |
| 7   | 0.6666666666666666 |
| 8   | 0.7777777777777778 |
| 9   | 0.8888888888888888 |
| 10  | 1.0                |

## RANK

Assigns a rank number to each row in the result set with equal values having the same number. There can be gaps between ranks. Results are similar to the ranking used in sporting events.

### Syntax

```
RANK (
    OVER ( [ PARTITION BY expr_list ] ORDER BY order_list )
```

| Parameter  | Data Type | Description                                                                     |
|------------|-----------|---------------------------------------------------------------------------------|
| part_list  | Any       | Specifies the column by which the rows are partitioned for the window function. |
| order_list | Any       | Specifies the column by which the rows are ordered for the window function.     |

### Example

This example ranks a series of ten random numbers. Use the [Math Functions and Operators](#) function to generate the random numbers.

```
WITH RandomNumbers AS (
    SELECT RAND() AS num
    FROM sys.dummy10
)
SELECT num, RANK() OVER (ORDER BY num) AS n
FROM RandomNumbers;
```

### Output

| num                 | n  |
|---------------------|----|
| 0.0003210869634629  | 1  |
| 0.03250125317575761 | 2  |
| 0.09413438896602266 | 3  |
| 0.130728439355569   | 4  |
| 0.376078282337459   | 5  |
| 0.5049826734614208  | 6  |
| 0.5642048500743658  | 7  |
| 0.5989180494879046  | 8  |
| 0.9077625673606013  | 9  |
| 0.9428422123895421  | 10 |

## RATIO\_TO\_REPORT

Computes the ratio of a value to the sum of the set of values. For example, the `RATIO_TO_REPORT` function calculates the ratio of the salary of an employee relative to the total salaries in the department where the employee belongs.

### Syntax

```
RATIO_TO_REPORT ( ratio_expression )
    OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

| Parameter        | Data Type | Description                                                                       |
|------------------|-----------|-----------------------------------------------------------------------------------|
| ratio_expression | Numeric   | A column or expression to calculate the ratio for each row compared to the total. |
| part_list        | Any       | Specifies the column by which the rows are partitioned for the window function.   |
| order_list       | Any       | Specifies the column by which the rows are ordered for the window function.       |

### Example

This example shows a ratio of each salary of an individual employee to the total salaries.

```
SELECT name, salaries, RATIO_TO_REPORT(salaries)
    OVER (ORDER BY salaries) AS ratio
FROM salaries;
```

### Output

| name            | salaries | ratio               |
|-----------------|----------|---------------------|
| Vasquez, Gustav | 70000.0  | 0.08484848484848485 |
| Ho, Chen        | 75000.0  | 0.09090909090909091 |
| Kiwi, Marie     | 110000.0 | 0.13333333333333333 |
| Smith, Roberts  | 140000.0 | 0.1696969696969697  |
| Porter, Polly   | 180000.0 | 0.21818181818181817 |
| Svenson, Olaf   | 250000.0 | 0.30303030303030304 |

## ROW\_NUMBER

Assigns a unique number to each row in the result set.

### Syntax

```
ROW_NUMBER (
    OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

| Parameter | Data Type | Description                                                                     |
|-----------|-----------|---------------------------------------------------------------------------------|
| part_list | Any       | Specifies the column by which the rows are partitioned for the window function. |

| Parameter  | Data Type | Description                                                                 |
|------------|-----------|-----------------------------------------------------------------------------|
| order_list | Any       | Specifies the column by which the rows are ordered for the window function. |

## Examples

### Determine the Unique Number of Each Row in a Table

This example assigns a row number to a basic table of fruit prices based on the price order.

```
SELECT fruit, price, ROW_NUMBER()
OVER (ORDER BY price) AS row_no
FROM fruits;
```

#### Output

| "fruit"    | "price" | "row_no" |
|------------|---------|----------|
| banana     | 0.5     | 1        |
| kiwi       | 0.75    | 2        |
| orange     | 1.5     | 3        |
| mango      | 1.75    | 4        |
| apple      | 1.99    | 5        |
| strawberry | 3.99    | 6        |

### Determine the Unique Number of Each Row Using a Partitioned Table

This example uses a table of sales data for two regions. The ROW\_NUMBER function orders the sales team members by their region and sales amounts.

```
SELECT
    salesperson,
    region,
    sales_amount,
    SUM(sales_amount) OVER (PARTITION BY region) AS total_sales_by_region,
    ROW_NUMBER() OVER (PARTITION BY region ORDER BY sales_amount DESC) AS sales_rank_in_region
FROM
    sales;
```

#### Output

| "salesperson" | "region" | "sales_amount" | "total_sales_by_region" | "sales_rank_in_region" |
|---------------|----------|----------------|-------------------------|------------------------|
| Eve           | North    | 2500           | 5000                    | 1                      |
| Bob           | North    | 1500           | 5000                    | 2                      |
| Alice         | North    | 1000           | 5000                    | 3                      |
| Dave          | South    | 3000           | 5000                    | 1                      |
| Carol         | South    | 2000           | 5000                    | 2                      |

## ZSCORE

Calculates the z-score of the sample based on the standard deviation (similar to the [Aggregate Functions](#) aggregate function).

The standard deviation assumes that the data is a sample of a larger set, meaning it corrects for bias.

If your data set is comprehensive and represents an entire group, use the [Window Aggregate Functions](#) function instead.



**Syntax**

```
ZSCORE (expr)
  OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

| Parameter  | Data Type | Description                                                                     |
|------------|-----------|---------------------------------------------------------------------------------|
| expr       | Numeric   | A column or expression to calculate the standard deviation for each row.        |
| part_list  | Any       | Specifies the column by which the rows are partitioned for the window function. |
| order_list | Any       | Specifies the column by which the rows are ordered for the window function.     |

**Example**

This example calculates the standard deviation in a range of salaries.

```
SELECT name, salaries, ZSCORE(salaries)
OVER (ORDER BY salaries) AS std_dev
FROM salaries;
```

*Output*

| name            | salaries | std_dev            |
|-----------------|----------|--------------------|
| Vasquez, Gustav | 70000.0  |                    |
| Ho, Chen        | 75000.0  | 0.7071076862849652 |
| Kiwi, Marie     | 110000.0 | 1.1470789011835154 |
| Smith, Roberts  | 140000.0 | 1.2593337909146212 |
| Porter, Polly   | 180000.0 | 1.4100478781464227 |
| Svenson, Olaf   | 250000.0 | 1.634473657490547  |

**ZSCOREP**

Calculates the z-score of the sample based on the population standard deviation (similar to the [Aggregate Functions](#) aggregate function).

The population standard deviation assumes that the data contains the entire population, meaning there is no correction for bias. Use it when your data set is comprehensive and represents the entire group (i.e. population) you are analyzing.

If your data set is a sample of a larger set, use the [Window Aggregate Functions](#) function.

**Syntax**

```
ZSCOREP (expr)
  OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

| Parameter | Data Type | Description                                                                     |
|-----------|-----------|---------------------------------------------------------------------------------|
| expr      | Numeric   | A column or expression to calculate the standard deviation for each row.        |
| part_list | Any       | Specifies the column by which the rows are partitioned for the window function. |

| Parameter  | Data Type | Description                                                                 |
|------------|-----------|-----------------------------------------------------------------------------|
| order_list | Any       | Specifies the column by which the rows are ordered for the window function. |

### Example

This example calculates the population standard deviation in a range of salaries.

```
SELECT name, salaries, ZSCOREP(salaries)
OVER (ORDER BY salaries) AS std_devp
FROM salaries;
```

### Output

| name            | salaries | std_devp           |
|-----------------|----------|--------------------|
| Vasquez, Gustav | 70000.0  |                    |
| Ho, Chen        | 75000.0  | 1.0000012800024576 |
| Kiwi, Marie     | 110000.0 | 1.4048790013060097 |
| Smith, Roberts  | 140000.0 | 1.4541534063682966 |
| Porter, Polly   | 180000.0 | 1.5764814535323706 |
| Svenson, Olaf   | 250000.0 | 1.7904761837110903 |

### Related Links

[Math Functions and Operators](#)

[General SQL Syntax](#)

[Generate Tables Using sys.dummy](#)

## 10.20 Other Functions and Expressions

The Ocient® System supports the following miscellaneous functions and expressions.

### CASE

CASE operates similarly to conditional scripting in other programming languages, allowing it to function like an if / then / else statement or as a switch statement.

CASE sets one or more WHEN conditions to be evaluated. When a condition is evaluated as true, the database applies the specified action in a THEN clause. If no matches are found, the database applies the action in the ELSE clause. If no ELSE clause is specified and no matches are found, the result is NULL.

This evaluation creates a new column where the values are expressions inside the THEN clause.

### Syntax

*--To use CASE like an if / then statement:*

```
CASE <caseCondition> ELSE expression | NULL END
```

```
<caseCondition> ::=
```

```
WHEN search_condition | expression THEN expression | NULL
```

*--To use CASE like a switch statement:*

```
CASE switch_expression <switchCondition> ELSE expression | NULL END
```

```
<switchCondition> ::=
WHEN switch_value THEN expression | NULL
```

| Arguments         | Description                                                                         |
|-------------------|-------------------------------------------------------------------------------------|
| search_condition  | A condition including a column name and a Boolean expression to evaluate.           |
| switch_expression | A column name to be used to evaluate aswitch_value.                                 |
| switch_value      | A value to be evaluated for theswitch_expression.                                   |
| expression        | An expression to return based on evaluating thesearch_conditionorswitch_expression. |

### Example

This example uses conditional logic with **CASE** to categorize individuals into different groups based on their ages.

```
SELECT age,
CASE
  WHEN age < 13 THEN 'child'
  WHEN age < 19 THEN 'teen'
  ELSE 'adult'
END AS age_grp
FROM sys.people
ORDER BY age;
```

### Example

This example uses switch conditions to categorize age groups.

```
SELECT age,
CASE age
  WHEN 13 THEN 'thirteen'
  WHEN 14 THEN 'fourteen'
  WHEN 15 THEN 'fifteen'
  WHEN 16 THEN 'sixteen'
  WHEN 17 THEN 'seventeen'
  WHEN 18 THEN 'eighteen'
  WHEN 19 THEN 'nineteen'
  ELSE 'not teens'
END
FROM sys.people;
```

## COALESCE

Returns the first argument that is not NULL or NULL if all arguments are NULL. Arguments must be the same type and the returned value is the most compatible type.

COALESCE can work with mixed-type numeric arguments, which means the lower precedence types are cast to the highest one. For example, when using **TIMESTAMP** and **DATE** data types, all values are cast to a timestamp.

### Syntax

```
COALESCE(value1, value2 [,...])
```

| Argument     | Data Type                                                          | Description                                               |
|--------------|--------------------------------------------------------------------|-----------------------------------------------------------|
| value [,...] | All data types are supported. All values must be of the same type. | Values for comparison to find the value that is not NULL. |

**Example**

```
SELECT COALESCE(null,null,13,null,8);
```

Output: 13

**COMMIT**

Returns the most recent commit hash of the database to which the client is currently connected.

**Syntax**

```
COMMIT()
```

**Example**

```
SELECT COMMIT();
```

Output: cd0d53e3aa

**CURRENT\_DATABASE**

Alias for [Other Functions and Expressions](#).

**CURRENT\_NODE**

Returns the name of the SQL Node where the current query executes. The name of the node corresponds to the `name` column in the `sys.nodes` system catalog table.

**Syntax**

```
CURRENT_NODE()
```

**Example**

```
SELECT CURRENT_NODE();
```

Output: sql0

**CURRENT\_NODE\_ID**

Returns the identifier of the SQL Node where the current query executes. The name of the node corresponds to the `id` column in the `sys.nodes` system catalog table.

**Syntax**

```
CURRENT_NODE_ID()
```

**Example**

```
SELECT CURRENT_NODE_ID();
```

Output: f690c14a-4e4f-4143-8e5e-20c90b60e15c

**CURRENT\_SCHEMA**

Returns the name of the current schema.

**Syntax**

```
CURRENT_SCHEMA()
```

**Example**

```
SELECT CURRENT_SCHEMA();
```

*Output:* schema@database

## CURRENT\_SYSTEM

Returns the name of the system. You can set the name using the [Cluster and Node Management SQL](#) statement.

If you never set the system name, the Ocint System initializes the name of the system as a random identifier, the Universally Unique Identifier (UUID).

### Syntax

```
CURRENT_SYSTEM()
```

### Example

```
SELECT CURRENT_SYSTEM();
```

*Output:* 2677b88c-7f07-470c-be3e-e1bafddc1221

## CURRENT\_USER

Returns the user for the current connection.

### Syntax

```
CURRENT_USER()
```

### Example

```
SELECT CURRENT_USER();
```

*Output:* user@database

## DATABASE

Alias for CURRENT\_DATABASE.

Returns the name of the database to which the client is currently connected.

### Syntax

```
DATABASE()
```

### Example

```
SELECT DATABASE();
```

*Output:* test

## GREATEST

Returns the largest non-NULL value of all the arguments, or NULL if all the arguments are NULL. All values must be comparable to each other.

### Syntax

```
GREATEST(value [,...])
```

| Argument     | Data Type                                                                                             | Description                                                         |
|--------------|-------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------|
| value [,...] | All data types are supported. All values must either be of the same type or comparable to each other. | A series of values to compare to find the largest value of the set. |

**Example**

```
SELECT GREATEST(12,13,5,8);
```

*Output:* 13

**IF\_NULL**

Returns the first argument that is not NULL or NULL if all arguments are NULL. Arguments must be the same type and the returned value is the most compatible type.

IF\_NULL can work with mixed-type numeric arguments, which means the lower precedence types are cast to the highest one. For example, when using **TIMESTAMP** and **DATE** data types, all values are cast to a timestamp.

**Syntax**

```
IF_NULL(value1, value2)
```

| Argument     | Data Type                                                          | Description                                               |
|--------------|--------------------------------------------------------------------|-----------------------------------------------------------|
| value [,...] | All data types are supported. All values must be of the same type. | Values for comparison to find the value that is not NULL. |

**Example**

```
SELECT IF_NULL(null, 5)
```

*Output:* 5

**LEAST**

Returns the smallest non-NULL value of all arguments, or NULL if all arguments are NULL. All values must be comparable to each other.

**Syntax**

```
LEAST(value [,...])
```

| Argument     | Data Type                                                                                             | Description                                                          |
|--------------|-------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| value [,...] | All data types are supported. All values must either be of the same type or comparable to each other. | A series of values to compare to find the smallest value of the set. |

**Example**

```
SELECT LEAST(12,13,5,8);
```

*Output:* 5

**MURMUR3**

Returns a 32-bit MurmurHash3 hash of the input value as an **INTEGER** data type. If you specify a NULL input value, the function returns 0. This function is primarily useful for partitioning data.

**Syntax**

```
MURMUR3(value)
```

| Argument | Data Type                         | Description                                        |
|----------|-----------------------------------|----------------------------------------------------|
| value    | All SQL data types are supported. | Specified value for the MurmurHash3 hash function. |

**Example**

```
SELECT MURMUR3(null);
```

Output: 0

**Example**

```
SELECT MURMUR3('fred');
```

Output: 331477181

**NULL\_IF**

Returns the NULL value if two arguments are equal; otherwise, returns the first argument.

**Syntax**

```
NULL_IF(value1, value2)
```

**Example**

```
SELECT NULL_IF(13,12);
```

Output: 13

**Example**

```
SELECT NULL_IF(13,13);
```

Output: NULL

**SHOW**

The SHOW function lets you explore the database and its metadata for user-defined items. For details, see Information Schema.

**Syntax**

```
SHOW <item>;
```

*-- <item> can be any of the following:*

```
{  COLUMNS,
   DATA_TYPES,
   DATABASES,
   GROUPS,
   INDEXES,
   RESERVED_WORDS,
   SCHEMATA,
   TABLES,
   USERS,
   VIEWS }
```

**Examples**

This example uses SHOW to explore databases on the system.

```
SHOW DATABASES;
```

*Output*

```
database_name
created
-----
test
2022-12-19 19:52:34.576426348
```

This example explores table data.

```
SHOW TABLES;
```

*Output*

```
table_catalog
table_schema
table_name
table_type
is_insertable_into
created_at
-----
test
test_schema
test_table
BASE TABLE
YES
2022-12-20 14:45:13.858144104
```

## TYPE\_STRIP

Returns the SQL type of the specified value.

### Syntax

```
TYPE_STRIP(value)
```

| Argument | Data Type                     | Description                                                                                            |
|----------|-------------------------------|--------------------------------------------------------------------------------------------------------|
| value    | All data types are supported. | Specified value for the determination of the type. You can specify columns, literals, and expressions. |

### Examples

```
SELECT TYPE_STRIP(_R{1,2,3});
```

Output: MATRIX[1][3]

```
SELECT TYPE_STRIP(current_user());
```

Output: CHAR

```
SELECT TYPE_STRIP(NULL);
```

Output: NULL

```
SELECT TYPE_STRIP(NULL::INT[]);
```

Output: INT[]



**VERSION**

Returns the version of the database to which the client is currently connected.

**Syntax**

```
VERSION()
```

**Example**

```
SELECT VERSION();
```

*Output:* 22.0.0-20230105.2353

**ZN**

If *x* is NULL, returns 0. Otherwise, returns *x*.

**Syntax**

```
ZN(x)
```

**Example**

```
SELECT ZN(null);
```

*Output:* 0

**Example**

```
SELECT ZN(12);
```

*Output:* 12

**Related Links**

[Database, Tables, Views, and Indexes](#)

[System Catalog](#)

[Database Administration](#)

**10.21 Data Definition Language (DDL) Statement Reference**

DDL statements allow users to run commands and administrative operations on the Ocident® System. You can execute DDL statements over a database connection to manage nodes, storage spaces, tables and views, system configuration, users and groups, underlying storage segments, and more.

**Quoting Identifiers**

In all DDL statements, identifiers not in double quotes must begin with a letter and can only contain letters, numbers, and underscores. Identifiers are the names of databases, tables, nodes, etc. For details, see [Identifiers](#). The Ocident System internally converts identifiers to lower-case. Identifiers in double quotes can contain any characters besides newline and carriage return, can begin with any character, and are not case-adjusted. To use an identifier with the same name as any keyword, it must be in double quotes.

String literals must be enclosed in single quotes. A single quote within a string can be escaped as `\"`. A string literal can be preceded by `e` to enable additional escape sequences (ex: `e'\n'`).

DDL statements are supported for these categories:

**Cluster and Node Management**

- **CLUSTER**
  - CREATE CLUSTER

- DROP CLUSTER
- ALTER CLUSTER
  - \* ALTER CLUSTER ADD PARTICIPANTS
  - \* ALTER CLUSTER DROP PARTICIPANTS
  - \* ALTER CLUSTER ALTER CONFIG SET
  - \* ALTER CLUSTER ALTER LOG LEVEL SET
  - \* ALTER CLUSTER RENAME
- **STORAGESPACE**
  - CREATE STORAGESPACE
  - DROP STORAGESPACE
  - ALTER STORAGESPACE RENAME
- **NODE**
  - DROP NODE
  - ALTER NODE
    - \* ALTER NODE ADD ROLE
    - \* ALTER NODE REMOVE ROLE
    - \* ALTER NODE ALTER CONFIG SET
    - \* ALTER NODE RENAME
    - \* ALTER NODE ALTER LOG LEVEL SET
    - \* ALTER NODE ALTER METRIC LEVEL
    - \* ALTER NODE SET ADDRESS
- **CONNECTIVITY POOL**
  - CREATE CONNECTIVITY\_POOL
  - DROP CONNECTIVITY\_POOL
  - ALTER CONNECTIVITY\_POOL
    - \* ALTER CONNECTIVITY\_POOL SET
    - \* ALTER CONNECTIVITY\_POOL RENAME TO
    - \* ALTER CONNECTIVITY\_POOL ADD PARTICIPANTS
    - \* ALTER CONNECTIVITY\_POOL ALTER PARTICIPANT
    - \* ALTER CONNECTIVITY\_POOL DROP PARTICIPANTS
    - \* ALTER CONNECTIVITY\_POOL SET SSO INTEGRATION
    - \* ALTER CONNECTIVITY\_POOL REMOVE SSO INTEGRATION
- **SSO INTEGRATION**
  - CREATE SSO INTEGRATION
  - DROP SSO INTEGRATION
  - ALTER SSO INTEGRATION
- **SYSTEM**
  - ALTER SYSTEM ALTER CONFIG SET
  - ALTER SYSTEM RENAME TO
  - ALTER SYSTEM ALTER METRIC LEVEL
  - ALTER SYSTEM ALTER SECURITY

## Database, Tables, Views, and Indexes

- **DATABASE**
  - CREATE DATABASE
  - DROP DATABASE
  - ALTER DATABASE
    - \* ALTER DATABASE RENAME
    - \* ALTER DATABASE SET SSO INTEGRATION
    - \* ALTER DATABASE ALTER SSO INTEGRATION
    - \* ALTER DATABASE REMOVE SSO INTEGRATION
    - \* ALTER DATABASE ALTER SECURITY
- **TABLE**
  - CREATE TABLE
  - CREATE TABLE AS SELECT (CTAS)

- CREATE TABLE AS SELECT USING LOADERS
- DROP TABLE
- ALTER TABLE
  - \* ALTER TABLE RENAME
  - \* ALTER TABLE RENAME COLUMN
  - \* ALTER TABLE ADD COLUMN
  - \* ALTER TABLE ALTER COLUMN COMPRESSION
  - \* ALTER TABLE ALTER REDUNDANCY
  - \* ALTER TABLE DROP COLUMN
  - \* ALTER TABLE STREAMLOADER\_PROPERTIES
  - \* ALTER TABLE DISABLE INDEX
  - \* ALTER TABLE ENABLE INDEX
- DELETE FROM TABLE
- EXPORT TABLE
- INSERT INTO TABLE
  - \* INSERT INTO TABLE USING LOADERS
- TRUNCATE TABLE
- **VIEW**
  - CREATE VIEW
  - DROP VIEW
  - ALTER VIEW RENAME
  - ALTER VIEW AS
  - EXPORT VIEW
- **INDEX**
  - CREATE INDEX
  - DROP INDEX

## Data Pipelines

- **PIPELINE**
  - CREATE PIPELINE
  - DROP PIPELINE
  - PREVIEW PIPELINE
  - START PIPELINE
  - STOP PIPELINE
  - ALTER PIPELINE RENAME
  - EXPORT PIPELINE
  - CREATE PIPELINE FUNCTION
  - DROP PIPELINE FUNCTION

## Distributed Tasks

- **TASK**
  - CREATE TASK
  - CANCEL TASK

## Machine Learning Models

- CREATE MLMODEL
- ALTER MLMODEL
- DROP MLMODEL
- EXPORT MLMODEL
- REFRESH MLMODEL

## Users, Groups, and Service Classes

- **USER**

- CREATE USER
- DROP USER
- ALTER USER
  - \* ALTER USER SET
- **GROUP**
  - CREATE GROUP
  - DROP GROUP
  - ALTER GROUP
    - \* ALTER GROUP USER
    - \* ALTER GROUP RENAME
    - \* ALTER GROUP SET SERVICE CLASS
    - \* ALTER GROUP ALTER SECURITY
- **SERVICE CLASS**
  - CREATE SERVICE CLASS
  - DROP SERVICE CLASS
  - ALTER SERVICE CLASS
    - \* ALTER SERVICE CLASS RENAME
    - \* ALTER SERVICE CLASS SET
    - \* ALTER SERVICE CLASS RESET
  - ALTER QUERY

## Data Integrity and Storage

- ALTER SEGMENT QUARANTINE

## Query Analysis

- EXPLAIN
- EXPLAIN PIPELINE

## Statistics Cache Management

### 10.21.1 Cluster and Node Management

Cluster and Node Management commands allow administrators to manage storage clusters, create and manage storage spaces, create and manage nodes, and apply configuration changes to nodes, clusters, or the overall system.

## CLUSTER

### CREATE CLUSTER

**CREATE CLUSTER** creates a new cluster in the current database. The cluster name must be distinct from the name of any existing tables in the database.

To create a cluster, the logged-in user must be a system-level user.

For details about creating clusters, see [Ocient Application Configuration](#).

For Foundation clusters, if the system is configured with a single-cluster configuration, when you add a second Foundation cluster, the system automatically changes to a multi-cluster configuration. This change is irreversible and requires a full system restart to fully take effect.

### Syntax

```
CREATE CLUSTER [ IF NOT EXISTS ] cluster_name <create_definition>
```

```
<create_definition> ::=
```

```

TYPE [=] foundation
| PARTICIPANTS [=] (node_name1, node_name2, ...)
| STORAGESPACE [=] storage_space_name

```

| Parameter    | Data Type | Description                            |
|--------------|-----------|----------------------------------------|
| cluster_name | string    | The name of the new cluster to create. |

Define Clusters ( <create\_definition> )

You must include these configuration settings when you create a new cluster.

| Configuration | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TYPE          | string    | The supported value is: <b>foundation</b> . Processes query operators and handles storage on the system.                                                                                                                                                                                                                                                                                                                          |
| PARTICIPANTS  | string    | A list of one or more node names included in the new cluster. The number of participant nodes in a cluster must be equal to or greater than the storage-space width. For details, see Cluster and Node Management. Adding more than the minimum number of participants to a cluster overprovisions the cluster and allows replacing or dropping participant nodes. Nodes cannot be assigned as participants to multiple clusters. |
| STORAGESPACE  | string    | The name of the storage space for the cluster.                                                                                                                                                                                                                                                                                                                                                                                    |

## Examples

This example creates a Foundation cluster named **lts\_cluster0** using the storage space named **my\_storage\_space** with three participant nodes.

```

CREATE CLUSTER lts_cluster0
  TYPE=foundation
  PARTICIPANTS=(node1,node2,node3)
  STORAGESPACE=my_storage_space;

```

## DROP CLUSTER

**DROP CLUSTER** removes an existing cluster from the system.

To remove a cluster, you must be a system-level user.

### Syntax

```

DROP CLUSTER [ IF EXISTS ] cluster_name

```

| Parameter    | Data Type | Description                                  |
|--------------|-----------|----------------------------------------------|
| cluster_name | string    | The name of the specified cluster to remove. |

### Example

This example removes an existing cluster named `lts_cluster0`.

```
DROP CLUSTER lts_cluster0;
```

### ALTER CLUSTER

Add Nodes To An Existing Cluster

`ALTER CLUSTER ADD PARTICIPANTS` adds the specified nodes to an existing cluster in the system. This operation can overprovision the storage space if necessary.

For the system to recognize newly added nodes, you must restart the cluster. To do this, you can follow the steps in [Expand and Rebalance System](#).

### Syntax

```
ALTER CLUSTER [ IF EXISTS ] cluster_name
  ADD PARTICIPANTS [=] ( node_name1 [,...] );
```

| Parameter          | Data Type | Description                                          |
|--------------------|-----------|------------------------------------------------------|
| cluster_name       | string    | The name of the specified cluster to alter.          |
| node_name1, [,...] | string    | The name of one or more nodes to add to the cluster. |

### Example

This example adds the node `node3` to a cluster named `example_cluster`, which already contains `node1` and `node2`. Following this operation, `example_cluster` consists of three nodes.

```
ALTER CLUSTER example_cluster ADD PARTICIPANTS (node3);
```

Remove Nodes from an Existing Cluster

`ALTER CLUSTER DROP PARTICIPANTS` drops nodes from an existing cluster.

This SQL statement fails for a Foundation cluster. Dropping the specified nodes brings the number of participants below the total width of the storage space associated with that cluster.

### Syntax

```
ALTER CLUSTER [ IF EXISTS ] cluster_name
  DROP PARTICIPANTS [=] ( node_name1 [,...] );
```

| Parameter          | Data Type | Description                                             |
|--------------------|-----------|---------------------------------------------------------|
| cluster_name       | string    | The name of the specified cluster to alter.             |
| node_name1, [,...] | string    | The name of one or more nodes to drop from the cluster. |

## Example

This example drops participant node `node3` from a cluster named `example_cluster`, which contains `node1`, `node2`, and `node3`. Following this operation, the participants in `example_cluster` consist of the nodes `node1` and `node2`.

```
ALTER CLUSTER example_cluster DROP PARTICIPANTS (node3);
```

## Cluster Scope Configuration

`ALTER CLUSTER ALTER CONFIG SET` sets a configuration override for the configuration at the cluster scope. For an explanation of how configuration overrides work, see [Inspect the Current Configuration](#).

If the specified path to the configuration setting does not exist, the system considers it as a new configuration and appends it to the collection of existing configurations. To reset a configuration override for a specific key, use the `RESET` keyword.

## Syntax

```
-- To set a new configuration override:
ALTER CLUSTER [ IF EXISTS ] cluster_name
    ALTER CONFIG SET string_definition [=] string_value [...]
```

```
-- To reset a configuration override:
ALTER CLUSTER [ IF EXISTS ] clusterName
    ALTER CONFIG RESET [ string_definition [...] ]
```

| Parameter                      | DataType | Description                                                                                                                  |
|--------------------------------|----------|------------------------------------------------------------------------------------------------------------------------------|
| <code>cluster_name</code>      | string   | The name of the specified cluster to alter.                                                                                  |
| <code>string_definition</code> | string   | The name of a configuration parameter to alter. Contact Ocident@support for details about altering configuration parameters. |
| <code>string_value</code>      | string   | The new value to set for the specified configuration parameter.                                                              |

## Examples

This example alters the base configuration for cluster `cluster_name` and sets the certificate for all nodes to a certificate named `cert_name.crt`.

```
ALTER CLUSTER cluster_name
    ALTER CONFIG SET 'certificateStore.cert' = 'cert_name.crt';
```

This example resets the certificate configuration for cluster `cluster_name`.

```
ALTER CLUSTER cluster_name
    ALTER CONFIG RESET 'certificateStore.cert';
```

This example alters the configuration at the cluster scope for cluster `cluster_name` and service role `lts`.

```
ALTER CLUSTER cluster_name
    ALTER CONFIG SET 'lts.numLevels' = '3';
```

## Cluster Scope Log Level Configuration

`ALTER CLUSTER cluster_name ALTER LOG LEVEL SET` sets the log level for a particular logger for all nodes in the provided cluster. For an explanation of how configuration overrides work, see [Inspect the Current Configuration](#).

## Syntax

```
-- To set a new configuration override:
ALTER CLUSTER cluster_name
    ALTER LOG LEVEL SET [ <logger_parameters> [,...] ]

<logger_parameters> ::=
    { 'ALL' | logger_name [=] logging_level }

-- To reset a configuration override:
ALTER CLUSTER cluster_name ALTER LOG LEVEL RESET [logger_name [, ...] ]
```

| Parameter    | Data Type | Description                                 |
|--------------|-----------|---------------------------------------------|
| cluster_name | string    | The name of the specified cluster to alter. |

#### Logger Parameters (<logger\_parameters>)

| Configuration | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| logger_name   | string    | Use <b>ALL</b> for the root logging configuration. Otherwise, the <b>logger_name</b> parameter can be any string specifying a logger name for all nodes in the provided cluster.                                                                                                                                                                                                                        |
| logger_level  | string    | The level of severity to be assigned to the <b>logger_name</b> . <b>logger_level</b> must be one of the following options: <b>'EDEBUG'</b>   <b>'DEBUG'</b>   <b>'VERBOSE'</b>   <b>'INFO'</b>   <b>'WARN'</b>   <b>'ERROR'</b> . See the <a href="#">Log Monitoring</a> page for more details on log levels. To reset a logger to the default log level ( <b>INFO</b> ), use the <b>RESET</b> keyword. |

#### Examples

This example alters the log level at the cluster scope. This sets the query log level for all nodes in the cluster to **DEBUG**.

```
ALTER CLUSTER cluster_name
    ALTER LOG LEVEL SET 'query' = 'DEBUG';
```

This example resets the log level for the query logger for the entire cluster.

```
ALTER CLUSTER cluster_name
    ALTER LOG LEVEL RESET 'query';
```

#### Rename Cluster

Rename an existing cluster in the system.

#### Syntax

```
ALTER CLUSTER old_cluster_name RENAME TO new_cluster_name
```



| Parameter        | Data Type | Description                                 |
|------------------|-----------|---------------------------------------------|
| old_cluster_name | string    | The name of the specified cluster to alter. |
| new_cluster_name | string    | The new name to assign to the cluster.      |

### Example

This example renames an existing cluster named `lts-cluster-0` to `lts-cluster-1`.

```
ALTER CLUSTER "lts-cluster-0" RENAME TO "lts-cluster-1";
```

## STORAGESPACE

### CREATE STORAGESPACE

`CREATE STORAGESPACE` creates a new storage space. The name of the storage space must be distinct from the name of any existing storage spaces in the system.

To create a storage space, you must possess the `CREATE STORAGESPACE` privilege for the current system.

It is recommended that the total width of a production system be smaller than the number of Foundation Nodes in the cluster so that loading can continue even in the event of a node outage.

### Syntax

```
CREATE STORAGESPACE [ IF NOT EXISTS ] storage_space_name
    [ WIDTH [=] width_integer], [ PARITY_WIDTH [=] pw_integer ]
```

| Parameter          | Data Type | Description                                                                                                                                               |
|--------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| storage_space_name | string    | The name of the new storage space. A storage space name must begin with a letter followed by letters, numbers, and underscores.                           |
| width_integer      | integer   | The total number of nodes to use in each segment group as it is written to disk. Width cannot be greater than the number of nodes in the storage cluster. |

| Parameter               | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pw_integer</code> | integer   | The number of parity coding bits to use for each segment group. This number must be less than <code>width</code> . The specified storage space can still complete queries even if nodes are disabled as long as the number of disabled nodes does not exceed the <code>parity_width</code> . For example, a storage space with <code>awidth = 5</code> and <code>aparity_width = 2</code> can have up to two nodes become disabled, and the system still uses the three remaining nodes to execute queries. However, if a third node goes down, the system can no longer process query operations. |

### Example

This example creates a new storage space named `ocient`.

```
CREATE STORAGE SPACE ocient WIDTH = 10, PARITY_WIDTH = 2;
```

### DROP STORAGE SPACE

`DROP STORAGE SPACE` removes an existing storage space, along with all associated tables and views. The SQL statement fails if one or more storage clusters are linked to the storage space.

To remove a storage space, you must possess the `DROP STORAGE SPACE` privilege for the storage space. Note that this action cannot be undone.

### Syntax

```
DROP STORAGE SPACE [ IF EXISTS ] storage_space_name [, ...]
```

| Parameter                       | Data Type | Description                                                                                                                                             |
|---------------------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>storage_space_name</code> | string    | The name of a storage space to drop. You can drop multiple storage spaces by specifying additional storage space names and separating each with commas. |

### Example

Remove an existing storage space named `ocient_ss`.

```
DROP STORAGE SPACE ocient_ss;
```

Remove multiple storage spaces.

```
DROP STORAGE SPACE ocient_ss1, ocient_ss2;
```

### ALTER STORAGE SPACE

`ALTER STORAGE SPACE RENAME` renames an existing storage space.

To rename a storage space, the logged-in user must be a system-level user.

**Syntax**

```
ALTER STORAGESPACE [ IF EXISTS ] old_storage_space_name RENAME TO new_storage_space_name
```

| Parameter              | DataType | Description                                   |
|------------------------|----------|-----------------------------------------------|
| old_storage_space_name | string   | The name of a storage space to rename.        |
| new_storage_space      | string   | The new name for the specified storage space. |

**Example**

To rename an existing storage space named `teststoragespace` to `storage-space-1`.

```
ALTER STORAGESPACE "teststoragespace" RENAME TO "storage-space-1";
```

**NODE****DROP NODE**

`DROP NODE` removes an existing node from the system.

A node cannot be removed from the system if it is still part of a cluster or one of the two remaining Administration Nodes.

**Syntax**

```
DROP NODE [ IF EXISTS ] node_name [, ...]
```

| Parameter | DataType | Description                                                                                                                  |
|-----------|----------|------------------------------------------------------------------------------------------------------------------------------|
| node_name | string   | The name of a node to drop. You can drop multiple nodes by specifying additional node names and separating each with commas. |

**Example**

Remove a node named `example_node`.

```
DROP NODE example_node;
```

Remove multiple nodes.

```
DROP NODE example_node1, example_node2;
```

**ALTER NODE**

`ALTER NODE RENAME TO`

`ALTER NODE RENAME` renames an existing node.

To rename a node, you must be a system-level user.

Renaming a node can cause a mismatch between the name of the node and the hostname of the node. Furthermore, you have to update users and scripts to use the new name of the node. Use this statement only to reverse these situations that can arise naturally (for example, you must replace a node with a node that has a different name).

If you have a DNS configuration for the node, you must reconfigure the node address after renaming the node by using the [Cluster and Node Management SQL](#) statement.

### Syntax

```
ALTER NODE [ IF EXISTS ] old_node_name RENAME TO new_node_name
```

| Parameter     | Data Type | Description                          |
|---------------|-----------|--------------------------------------|
| old_node_name | string    | The name of a node to rename.        |
| new_node_name | string    | The new name for the specified node. |

### Example

This example renames an existing node named `sql0` to `sql1`.

```
ALTER NODE sql0 RENAME TO sql1;
```

### ALTER NODE ADD ROLE

ALTER NODE ADD ROLE adds a role to a node. Each node can have one or more roles that it performs.

To add a role, you must be a system-level user.

### Syntax

```
ALTER NODE [ IF EXISTS ] node_name ADD ROLE [ IF NOT EXISTS ] <node_role>
```

```
<node_role> ::=
{ sql | streamloader | operatorvm | admin | health }
```

| Parameter | Data Type | Description                  |
|-----------|-----------|------------------------------|
| node_name | string    | The name of a node to alter. |

### Node Roles (<node\_role>)

This table explains the supported roles for nodes:

| Role         | Description                                                                                                        | Notes                                                                                                              |
|--------------|--------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| streamloader | The <code>streamloader</code> role allows the node to continually import a large amount of data into the database. | When adding the <code>streamloader</code> role, the node must be restarted in order for the change to take effect. |
| sql          | The <code>sql</code> role accepts a SQL statement, parses the statement, and compiles the statement.               | You can remove this role only if there is at least one other SQL Node present.                                     |
| operatorvm   | The <code>operatorvm</code> role manages database query plans and executes queries.                                | Added or removed automatically together with the <code>sql</code> role.                                            |

| Role                | Description                                                                                                                                                                            | Notes                                                                                                 |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| <code>lts</code>    | The nodes with the <code>lts</code> role store data on disk in segments in a column-oriented fashion. The <code>lts</code> role is associated with the Foundation Nodes on the system. | Can only be added or removed by adding or removing the node as a participant in a Foundation cluster. |
| <code>admin</code>  | The <code>admin</code> role accepts the administrative protocol and updates system metadata.                                                                                           | Can only be removed if there are at least two other Administrative Nodes.                             |
| <code>health</code> | A node with the <code>health</code> role is responsible for maintaining performance-related counters and statistics.                                                                   | Present initially on every node. Cannot be removed.                                                   |

### Example

To add SQL role to the node `sql1`:

```
ALTER NODE sql1 ADD ROLE sql;
```

ALTER NODE REMOVE ROLE

ALTER NODE REMOVE ROLE removes a role from a node. For a list of roles, see [Cluster and Node Management](#).

Some node roles cannot be removed using an ALTER NODE SQL statement, including `health`, `lts`, and `operatorvm`.

To remove a role, you must be a system-level user.

The node must be restarted in order for this change to take effect.

If you remove a SQL role from a SQL Node, the database removes the node from the connectivity pool. You must restart all SQL Nodes so that the other SQL Nodes understand that the node is no longer a SQL Node.

When you remove the `admin` role, you must also remove the `/var/opt/ocient/metadataStorage.raft` file.

### Syntax

```
ALTER NODE [ IF EXISTS ] node_name REMOVE ROLE [ IF EXISTS ]
{ sql | streamloader | admin }
```

| Parameter              | Data Type | Description                  |
|------------------------|-----------|------------------------------|
| <code>node_name</code> | string    | The name of a node to alter. |

### Example

This example removes the `sql` role from a node.

```
ALTER NODE my_node REMOVE ROLE sql;
```

ALTER NODE ALTER CONFIG SET

ALTER NODE ALTER CONFIG SET sets a configuration override for the configuration at the node scope. For an explanation of how configuration overrides work, see [Inspect the Current Configuration](#).

If the specified path to the configuration setting does not exist, it will be considered a new configuration and appended to the collection of existing configurations. To reset a configuration override for a specific key, use the RESET keyword.

### Syntax

```
-- To set a new configuration override:
ALTER NODE [ IF EXISTS ] node_name
    ALTER CONFIG SET string_definition [=] string_value [,...]

-- To reset a configuration override:
ALTER NODE [ IF EXISTS ] node_name ALTER CONFIG RESET string_definition [,...]
```

| Parameter         | DataType | Description                                                                                                                     |
|-------------------|----------|---------------------------------------------------------------------------------------------------------------------------------|
| node_name         | string   | The name of the specified node to alter.                                                                                        |
| string_definition | string   | The name of a configuration parameter to alter. Contact {{Ocient}} Support for details about altering configuration parameters. |
| string_value      | string   | The new value to be set for the specified configuration parameter.                                                              |

### Examples

This example sets the certificate name for this node to `cert_name.crt`.

```
ALTER NODE node2 ALTER CONFIG SET 'certificateStore.cert' = 'cert_name.crt';
```

This example resets the certificate configuration for the node `node2`.

```
ALTER NODE node2 ALTER CONFIG RESET 'certificateStore.cert';
```

This example alters the configuration at the node scope for node `node2` for service role `lts`.

```
ALTER NODE node2 ALTER CONFIG SET 'lts.numLevels' = '3';
```

### ALTER NODE ALTER LOG LEVEL SET

ALTER NODE node\_name ALTER LOG LEVEL SET sets the log level for a particular logger on the specified node. For an explanation of how configuration overrides work, see [Inspect the Current Configuration](#).

The logger name specified can be any string specifying a logger name, or ALL for the root logging configuration. See the [Log Monitoring](#) for more details.

To reset a logger to the default log level (INFO), use the RESET keyword.

### Syntax

```
-- To set a new configuration override:
ALTER NODE [ IF EXISTS ] node_name ALTER LOG LEVEL SET { <logger_parameters> [,...] }

<logger_parameters> ::=
    'ALL' | logger_name [=] { 'EDEBUG' | 'DEBUG' | 'VERBOSE' | 'INFO' | 'WARN' | 'ERROR' }

-- To reset a configuration override:
ALTER NODE [ IF EXISTS ] node_name ALTER LOG LEVEL RESET [ logger_name [,...] ]
```

| Parameter           | Data Type | Description                                                                                                                                                                                                                                                                                                                       |
|---------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| node_name           | string    | The name of the specified node to alter.                                                                                                                                                                                                                                                                                          |
| <logger_parameters> | string    | Use <b>ALL</b> for the root logging configuration. Otherwise, the <b>logger_name</b> parameter can be any string specifying a logger name assigned to a severity level. See the <a href="#">Error Monitoring</a> page for more details. To reset a logger to the default log level ( <b>INFO</b> ), use the <b>RESET</b> keyword. |

### Examples

This example alters the log level at the node scope. The example sets the query log level for this particular node to **DEBUG**.

```
ALTER NODE node_name ALTER LOG LEVEL SET 'query' = 'DEBUG';
```

This example resets log level for the query logger on the specified node.

```
ALTER NODE node_name ALTER LOG LEVEL RESET 'query';
```

### ALTER NODE ALTER METRIC LEVEL

Alters the reporting level of various internal metrics on a specific node. For details about internal metrics, see [Statistics Monitoring](#).

To set metrics reporting on all nodes across the system, see [Cluster and Node Management](#).

### Syntax

```
ALTER NODE [ IF EXISTS ] node_name ALTER METRIC LEVEL
  { SET <set_options> | RESET <reset_options> }

<set_options> ::=
  [ LIKE | REGEX ] metric_to_alter [ ,... ] [=] { 'INFO' | 'DEBUG' }

<reset_options> ::=
  { [ LIKE | REGEX ] metric_to_reset [ ,... ] | ALL }
```

| Parameter | Data Type | Description                  |
|-----------|-----------|------------------------------|
| node_name | string    | The name of a node to alter. |

### Set Metric Levels <set\_options>

The **SET** SQL statement assigns one or more specified metrics to the specified reporting level.

To set levels for a range of statistics, you can use wildcard characters with the **LIKE** keyword. Alternatively, you can use regular expressions with the **REGEX** keyword.

| Parameter                    | Data Type | Description                                                                                                                                                                                                                              |
|------------------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>metric_to_alter</code> | string    | The names of one or more metrics to alter for logging. Enclose any metric names in single quotes. The supported logging level values for metrics are <code>INFO</code> and <code>DEBUG</code> . Either keyword must be in single quotes. |

#### Reset Metric Levels <reset\_options>

The `RESET SQL` statement reverts one or more specified metrics to the default log level.

You can use wildcard characters with the `LIKE` keyword. Alternatively, you can use regular expressions with the `REGEX` keyword.

Use the `ALL` keyword to revert all metrics to the default log level.

| Parameter                    | Data Type | Description                                                                                                                                     |
|------------------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>metric_to_reset</code> | string    | The names of one or more metrics to reset by reverting their log level to the default. To revert all metrics, use the <code>ALL</code> keyword. |

#### Examples

In this example, the `ALTER NODE SQL` statement sets the `resultCache.queries` metric on the node `oc1` to the `DEBUG` level.

```
ALTER NODE oc1 ALTER METRIC LEVEL
SET 'resultCache.queries' = 'DEBUG';
```

This example uses the `LIKE` wildcard keyword to capture any metrics prefixed by `localStorageService.device.smart.` and assign them to the `INFO` level.

```
ALTER NODE oc1 ALTER METRIC LEVEL
SET LIKE 'localStorageService.device.smart.%' 'INFO';
```

Similar to the previous example, this SQL statement uses regular expressions to capture any metrics prefixed by `localStorageService.device.smart..`

```
ALTER NODE oc1 ALTER METRIC LEVEL
SET REGEX 'localStorageService.device.smart.*' 'INFO';
```

This example alters the level of multiple metrics.

```
ALTER NODE oc1 ALTER METRIC LEVEL
SET 'resultCache.queries' = 'DEBUG',
    'resultCache.data' = 'INFO',
    LIKE 'localStorageService.device.smart.%' 'INFO';
```

This example resets the same metrics.

```
ALTER NODE oc1 ALTER METRIC LEVEL
RESET 'resultCache.queries',
      'resultCache.data',
    LIKE 'localStorageService.device.smart.%;
```

#### ALTER NODE SET ADDRESS



**ALTER NODE** *node\_name* **SET ADDRESS** *ip\_address* changes the internal IP address or hostname for any node. You must restart the system after you change the address. If you change the address of an Administrator Node, then you must update the `bootstrap.conf` file for every node with the new address.

For details about this file, see [Ocint System Bootstrapping](#).

### Syntax

```
ALTER NODE [ IF EXISTS ] node_name SET ADDRESS node_address
```

| Parameter           | Data Type | Description                                                                                                                                                           |
|---------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>node_name</i>    | string    | The name of a node to alter.                                                                                                                                          |
| <i>node_address</i> | string    | The address for a node. This address can be an internal IP address (e.g., 111.11.11.111) or a DNS hostname with the node address matching the node name (e.g., sql1). |

### Examples

This example changes the IP address to 111.11.11.111 for the node named `my_node`.

```
ALTER NODE my_node SET ADDRESS '111.11.11.111';
```

This example changes the SQL Node `sql1` to use the DNS address `sql1`.

```
ALTER NODE sql1 SET ADDRESS 'sql1';
```

## CONNECTIVITY POOL

### CREATE CONNECTIVITY\_POOL

**CREATE CONNECTIVITY\_POOL** creates a connectivity pool from participant SQL Nodes for a client connection. Connectivity pools enable you to change the IP address of the client connection. Client redirects happen only within connectivity pools. For details about connectivity pools and managing them, see [Manage the Network Configuration of an Ocint System](#).

All SQL Nodes must be part of at least one connectivity pool. SQL Nodes only listen using the values set for the `listen_ip` and `listen_port` parameters.

The database uses the `source_ip`, `source_port`, and `priority` parameters when multiple connectivity pools exist that a client can use for the connection. In this case, the database uses the connectivity pool with the highest priority.

The database commits all changes made from these SQL statements after you restart all the SQL Nodes.

For information on all connectivity pools on the system, query the system catalog tables for Connectivity Pools. To see the connectivity pool assignment for each node, query the system catalog table for Connectivity Pool Participants.

### Syntax

```
CREATE CONNECTIVITY_POOL [IF NOT EXISTS] pool_name <create_definition>
```

```
<create_definition> ::=
    SOURCE_ADDRESS [=] source_ip
    | [ SOURCE_PORT [=] source_port ]
    | PRIORITY [=] priority
    | [ SSO INTEGRATION [=] sso_name ]
    | PARTICIPANTS (
        (NODE [=] node_name
```

```

LISTEN_ADDRESS [=] listen_ip
LISTEN_PORT [=] listen_port
ADVERTISED_ADDRESS [=] advertised_ip
[ ADVERTISED_PORT [=] advertised_port ]
[ OPENAPI_PORT [=] openapi_port ), ... )

```

| Parameter       | Data Type | Description                                                                                                                                                                                                            |
|-----------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pool_name       | string    | The name of the connectivity pool.                                                                                                                                                                                     |
| source_ip       | string    | The source IP address, which is the IP address in CIDR notation of the client that connects to the SQL Node.                                                                                                           |
| source_port     | integer   | Optional. The source port number is the port number of the client. This parameter defaults to none, which means any port number.                                                                                       |
| priority        | integer   | The priority of the connection. A higher number indicates a higher priority.                                                                                                                                           |
| sso_name        | string    | The name of an SSO integration to use for the connectivity pool. For details, see Cluster and Node Management.                                                                                                         |
| node_name       | string    | The name of the SQL Node.                                                                                                                                                                                              |
| listen_ip       | string    | The IP address for listening. ≤ Connectivity pools do not support dynamic IP addresses. If the IP address of your SQL Node changes, you must manually update it by using an Cluster and Node Management SQL statement. |
| listen_port     | integer   | The port number for listening.                                                                                                                                                                                         |
| advertised_ip   | string    | The IP address to return to the client.                                                                                                                                                                                |
| advertised_port | integer   | Optional. The port number to return to the client. If you specify NULL for this parameter, then the database uses the value of the listen_port parameter.                                                              |
| openapi_port    | integer   | The specified port for HTTP Query API-compliant specifications and endpoints.                                                                                                                                          |

## Examples

Create the connectivity pool named `test_pool` with source address `1.2.3.4/32` and priority 1 for one SQL Node named `sql0`. The node has the IP address `111.1.1.1` and port number 4050 for listening. Specify the local IP address to return to the client.

```

CREATE CONNECTIVITY_POOL IF NOT EXISTS test_pool
SOURCE_ADDRESS '1.2.3.4/32'
PRIORITY 1
PARTICIPANTS (
  (NODE sql0

```

```

LISTEN_ADDRESS '111.1.1.1'
LISTEN_PORT 4050
ADVERTISED_ADDRESS 'localhost');

```

Create the connectivity pool named `test_pool` with source address `1.2.3.4/32`, port number `44`, and priority `1` for one SQL Node named `sql0`. The node has the IP address `111.1.1.1` and port number `4050` for listening. Specify the local IP address and port number `4050` to return to the client.

```

CREATE CONNECTIVITY_POOL IF NOT EXISTS test_pool
SOURCE_ADDRESS '1.2.3.4/32'
SOURCE_PORT 44
PRIORITY 1
PARTICIPANTS (
  (NODE sql0
    LISTEN_ADDRESS '111.1.1.1'
    LISTEN_PORT 4050
    ADVERTISED_ADDRESS 'localhost'
    ADVERTISED_PORT 4050));

```

Create the connectivity pool named `test_pool` with source address `1.2.3.4/32`, port number `44`, priority `2`, and three participant SQL Nodes: `sql0`, `sql1`, and `sql2`. The nodes have the IP address `111.1.1.1` and port number `4050` for listening. Specify the local IP address and port number `4050` to return to the client.

```

CREATE CONNECTIVITY_POOL IF NOT EXISTS test_pool
SOURCE_ADDRESS '1.2.3.4/32'
SOURCE_PORT 44
PRIORITY 2
PARTICIPANTS (
  (NODE sql0
    LISTEN_ADDRESS '111.1.1.1'
    LISTEN_PORT 4050
    ADVERTISED_ADDRESS 'localhost'
    ADVERTISED_PORT 4050),
  (NODE sql1
    LISTEN_ADDRESS '111.1.1.1'
    LISTEN_PORT 4050
    ADVERTISED_ADDRESS 'localhost'
    ADVERTISED_PORT 4050),
  (NODE sql2
    LISTEN_ADDRESS '111.1.1.1'
    LISTEN_PORT 4050
    ADVERTISED_ADDRESS 'localhost'
    ADVERTISED_PORT 4050));

```

## DROP CONNECTIVITY\_POOL

`DROP CONNECTIVITY_POOL` removes the specified connectivity pool.

All nodes must belong to a connectivity pool. The execution of this SQL statement does not permit the removal of a pool such that a node becomes an orphan.

## Syntax

```

DROP CONNECTIVITY_POOL [ IF EXISTS ] pool_name

```

| Parameter | Data Type | Description                                  |
|-----------|-----------|----------------------------------------------|
| pool_name | string    | The name of the connectivity pool to remove. |

### Example

Remove the connectivity pool named `test_pool`.

```
DROP CONNECTIVITY_POOL test_pool;
```

### ALTER CONNECTIVITY\_POOL

ALTER CONNECTIVITY\_POOL SET

ALTER CONNECTIVITY\_POOL SET sets the metadata of a connectivity pool that includes the source IP address, source port number, priority, and node participants.

### Syntax

```
ALTER CONNECTIVITY_POOL [ IF EXISTS ] pool_name SET <connectivity_pool_definition>
```

```
<connectivity_pool_definition> ::=
    SOURCE_ADDRESS [=] source_ip |
    SOURCE_PORT [=] source_port |
    PRIORITY [=] priority |
    PARTICIPANTS(
        (NODE [=] node_name
          LISTEN_ADDRESS [=] listen_ip
          LISTEN_PORT [=] listen_port
          ADVERTISED_ADDRESS [=] advertised_ip
          [ADVERTISED_PORT [=] advertised_port] ), ...)
```

For parameter definitions, see [Cluster and Node Management](#).

To directly edit the LISTEN\_ADDRESS, LISTEN\_PORT, ADVERTISED\_ADDRESS, or ADVERTISED\_PORT parameters, you must either remove the participants and add them back with new values for these parameters using the ALTER CONNECTIVITY\_POOL DROP PARTICIPANTS and ALTER CONNECTIVITY\_POOL ADD PARTICIPANTS statements respectively, or use the ALTER CONNECTIVITY\_POOL SET PARTICIPANTS statement.

### Examples

This example sets the priority to 2 of the connectivity pool named `test_pool`.

```
ALTER CONNECTIVITY_POOL test_pool SET PRIORITY = 2;
```

This example sets the participants of the connectivity pool named `test_pool` to the list of participants with SQL Nodes `sql1` and `sql2`. When you execute this statement, the database resets the prior list of node participants to the new list. The nodes have the IP address 111.1.1.1 and port number 4050 for listening. Specify the local IP address and port number 4050 to return to the client.

```
ALTER CONNECTIVITY_POOL test_pool
SET PARTICIPANTS(
    (NODE sql1
      LISTEN_ADDRESS '111.1.1.1'
      LISTEN_PORT 4050
      ADVERTISED_ADDRESS 'localhost'
      ADVERTISED_PORT 4050),
    (NODE sql2
      LISTEN_ADDRESS '111.1.1.1'
      LISTEN_PORT 4050
```

```
ADVERTISED_ADDRESS 'localhost'
ADVERTISED_PORT 4050));
```

View the node identifiers of the participants of the `test_pool` connectivity pool by querying the node identifier `node_id` in the `sys.connectivity_pool_participants` and `sys.connectivity_pools` system catalog tables.

```
SELECT cpp.node_id
FROM sys.connectivity_pool_participants cpp
INNER JOIN sys.connectivity_pools cp
ON cpp.id = cp.id
WHERE cp.name = 'test_pool';
```

**ALTER CONNECTIVITY\_POOL RENAME TO**

**ALTER CONNECTIVITY\_POOL RENAME TO** SQL statement renames the existing connectivity pool.

#### Syntax

```
ALTER CONNECTIVITY_POOL [ IF EXISTS ] pool_name RENAME TO new_pool_name
```

| Parameter                  | Data Type | Description                                 |
|----------------------------|-----------|---------------------------------------------|
| <code>pool_name</code>     | string    | The name of the existing connectivity pool. |
| <code>new_pool_name</code> | string    | The new name of the connectivity pool.      |

#### Examples

Rename the `test_pool` connectivity pool to `new_test_pool`.

```
ALTER CONNECTIVITY_POOL test_pool RENAME TO new_test_pool;
```

**ALTER CONNECTIVITY\_POOL ADD PARTICIPANTS**

**ALTER CONNECTIVITY\_POOL ADD PARTICIPANTS** adds one or more nodes to the participant list of an existing connectivity pool.

For information on all connectivity pools on the system, query the system catalog tables for Connectivity Pools. To see the connectivity pool assignment for each node, query the system catalog table for Connectivity Pool Participants.

#### Syntax

```
ALTER CONNECTIVITY_POOL [ IF EXISTS ] pool_name
  ADD PARTICIPANTS(
    (NODE [=] node_name
      LISTEN_ADDRESS [=] listen_ip
      LISTEN_PORT [=] listen_port
      ADVERTISED_ADDRESS [=] advertised_ip
      [ADVERTISED_PORT [=] advertised_port] ), ...)
  )
```

#### Examples

Add one node `sql5` to the participant list of the connectivity pool `test_pool`. The node has the IP address `111.1.1.1` and port number `4050` for listening. Specify the local IP address and port number `4050` to return to the client.

```
ALTER CONNECTIVITY_POOL test_pool
  ADD PARTICIPANTS(
    NODE sql5
    LISTEN_ADDRESS '111.1.1.1'
    LISTEN_PORT 4050
    ADVERTISED_ADDRESS 'localhost'
    ADVERTISED_PORT 4050);
```

Add two nodes `sql6` and `sql7` to the participants list of the connectivity pool `test_pool`. The nodes have the IP address `111.1.1.1` and port number `4050` for listening. Specify the local IP address and port number `4050` to return to the client.

```
ALTER CONNECTIVITY_POOL test_pool
  ADD PARTICIPANTS(
    (NODE sql6
      LISTEN_ADDRESS '111.1.1.1'
      LISTEN_PORT 4050
      ADVERTISED_ADDRESS 'localhost'
      ADVERTISED_PORT 4050),
    (NODE sql7
      LISTEN_ADDRESS '111.1.1.1'
      LISTEN_PORT 4050
      ADVERTISED_ADDRESS 'localhost'
      ADVERTISED_PORT 4050));
```

#### ALTER CONNECTIVITY\_POOL ALTER PARTICIPANT

`ALTER CONNECTIVITY_POOL ALTER PARTICIPANT` modifies the configurations of one SQL Node assigned to the existing connectivity pool.

This SQL statement can alter only one configuration parameter of a participant node at a time.

For information on all connectivity pools on the system, query the system catalog tables for Connectivity Pools. To see the connectivity pool assignment for each node, query the system catalog table for Connectivity Pool Participants.

#### Syntax

```
ALTER CONNECTIVITY_POOL [ IF EXISTS ] pool_name
  ALTER PARTICIPANT node_name
  SET <connectivity_pool_definition>

<connectivity_pool_definition> ::=
  { LISTEN_ADDRESS [=] listen_ip
  | LISTEN_PORT [=] listen_port
  | ADVERTISED_ADDRESS [=] advertised_ip
  | ADVERTISED_PORT [=] advertised_port
  | OPENAPI_PORT [=] openapi_port }
```

| Parameter                    | Data Type | Description                                                                                                                                                  |
|------------------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pool_name</code>       | string    | The name of the connectivity pool.                                                                                                                           |
| <code>node_name</code>       | string    | The name of the SQL Node to alter.                                                                                                                           |
| <code>listen_ip</code>       | string    | The IP address for listening.                                                                                                                                |
| <code>listen_port</code>     | numeric   | The port number for listening.                                                                                                                               |
| <code>advertised_ip</code>   | string    | The IP address to return to the client.                                                                                                                      |
| <code>advertised_port</code> | numeric   | The port number to return to the client. If you specify NULL for this parameter, then the database uses the value of the <code>listen_port</code> parameter. |
| <code>openapi_port</code>    | numeric   | The specified port for HTTP Query API-compliant specifications and endpoints.                                                                                |

#### Example

Modify the SQL Node `sql0` to use the new `listen_port` value 4051.

```
ALTER CONNECTIVITY_POOL cp_test
  ALTER PARTICIPANT sql0
  SET listen_port 4051;
```

### ALTER CONNECTIVITY\_POOL DROP PARTICIPANTS

`ALTER CONNECTIVITY_POOL DROP PARTICIPANTS` removes one or more nodes from the participant list of an existing connectivity pool. If you remove the last node of a connectivity pool, the Ocint System automatically removes the connectivity pool.

For information on all connectivity pools on the system, query the system catalog tables for Connectivity Pools. To see the connectivity pool assignment for each node, query the system catalog table for Connectivity Pool Participants.

All nodes must belong to a connectivity pool. The execution of this SQL statement does not permit the removal of a node such that it becomes an orphan.

### Syntax

```
ALTER CONNECTIVITY_POOL [ IF EXISTS ] pool_name
  DROP PARTICIPANTS (node_name, ...)
```

### Examples

Remove one node `sql5` from the connectivity pool `test_pool`.

```
ALTER CONNECTIVITY_POOL IF EXISTS test_pool DROP PARTICIPANTS sql5;
```

Remove nodes `sql6` and `sql7` from the connectivity pool `test_pool`.

```
ALTER CONNECTIVITY_POOL test_pool DROP PARTICIPANTS (sql6, sql7);
```

### ALTER CONNECTIVITY\_POOL SET SSO INTEGRATION

`ALTER CONNECTIVITY_POOL SET SSO INTEGRATION` assigns a connectivity pool to use a specific SSO integration.

Connectivity pools look for the specified SSO integration name for each database you are trying to connect to. If the database doesn't support that SSO integration name, it uses the default SSO.

### Syntax

```
ALTER CONNECTIVITY_POOL [ IF EXISTS ] pool_name
  SET SSO INTEGRATION sso_name
```

| Parameter              | Data Type | Description                                 |
|------------------------|-----------|---------------------------------------------|
| <code>pool_name</code> | string    | The name of the existing connectivity pool. |
| <code>sso_name</code>  | string    | The new name of the SSO integration.        |

### Example

```
ALTER CONNECTIVITY_POOL cp_test SET SSO INTEGRATION sso_test;
```

### ALTER CONNECTIVITY\_POOL REMOVE SSO INTEGRATION

`ALTER CONNECTIVITY_POOL REMOVE SSO INTEGRATION` removes an assigned SSO integration.

### Syntax

```
ALTER CONNECTIVITY_POOL [ IF EXISTS ] pool_name
  REMOVE SSO INTEGRATION sso_name
```

| Parameter | DataType | Description                                 |
|-----------|----------|---------------------------------------------|
| pool_name | string   | The name of the existing connectivity pool. |
| sso_name  | string   | The name of the existing SSO integration.   |

### Example

This example removes the SSO integration `sso_test` from the connectivity pool `cp_test`.

```
ALTER CONNECTIVITY_POOL cp_test REMOVE SSO INTEGRATION sso_test;
```

## SSO INTEGRATION

### CREATE SSO INTEGRATION

Creates a new SSO integration protocol, which can connect to databases as a database integration (see [Database, Tables, Views, and Indexes](#)) or connect using a connectivity pool (see [Cluster and Node Management](#)).

#### Syntax

```
CREATE SSO INTEGRATION [IF NOT EXISTS ] sso_name
    PROTOCOL sso_protocol <sso_property> [, ... ]
```

```
<sso_property> ::=
    -- literal or string value
    property_name = value |
    -- list of values
    property_name = [ value [, ...] ] |
    -- map of values
    property_name = { key = value [, ...] }
```

| Parameter    | Type   | Description                                                                                                                                                                       |
|--------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sso_name     | string | The identifier of the SSO integration to create.                                                                                                                                  |
| sso_protocol | string | Supported values are: <code>oidc</code> , <code>openid</code> , and <code>openid_connect</code> . All these values are aliases of each other and use the OAuth 2.0 Authorization. |



| Parameter     | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| property_name | string | One or more properties to include in the SSO integration. Each SSO property must be specified either as a literal, list, or map. See the SSO Properties table for the requirements for each supported SSO parameter. SSO properties also have these rules: You can place any string key or value between double quotations (e.g., "default_group"). String values that contain characters other than [a-zA-Z]   [0-9]   '_' require double quotations (e.g., "this.is.a.complex-\$string"). Using a NULL value clears the existing configuration of a list or map property (e.g., user_claim_ids = NULL). |

## SSO Properties

### Example

This example creates an SSO integration protocol with the specified OpenID issuer "https://accounts.google.com", client identifier, and default group. Each of these parameters is required to create a new SSO integration. The example assumes preset values for the client identifier and default group.

```
CREATE SSO INTEGRATION sso_test PROTOCOL oidc
  issuer = "https://accounts.google.com",
  client_id = example_database_app_id,
  default_group = example_database_group;
```

## DROP SSO INTEGRATION

Drops an SSO integration protocol.

### Syntax

```
DROP SSO INTEGRATION [ IF EXISTS ] sso_name
```

| Parameter | Type   | Description                                    |
|-----------|--------|------------------------------------------------|
| sso_name  | string | The identifier of the SSO integration to drop. |

### Example

This SQL statement drops the SSO integration protocol sso\_test.

```
DROP SSO INTEGRATION sso_test;
```

## ALTER SSO INTEGRATION

Alters an SSO integration protocol by renaming it or modifying its properties.

```
ALTER SSO INTEGRATION sso_name [ IF EXISTS ] {
  RENAME TO new_name |
  SET <sso_property> [, ... ]
}
```

```
<sso_property> ::=
  -- literal or string value
  property_name = value |
  -- list of values
  property_name = [ value [, ...] ] |
  -- map of values
  property_name = { key = value [, ...] }
```

| Parameter     | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sso_name      | string | The identifier of the SSO integration to alter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| new_name      | string | The new name of the SSO integration.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| property_name | string | One or more properties to include in the SSO integration. Each SSO property must be specified either as a literal, list, or map. See the <a href="#">Cluster and Node Management</a> table to see the requirements for each supported SSO parameter. SSO properties also have these rules: You can place any string key or value between double quotations (e.g., "default_group"). String values that contain characters other than [a-zA-z]   [0-9]   '_' require double quotations (e.g., "this.is.a.complex-\$string"). Using a NULL value clears the existing configuration of a list or map property (e.g., user_claim_ids = NULL). |

## Examples

### Alter SSO Integration Properties

This example alters the SSO integration to use different properties, changing the OpenID client identifier and the default group to "group2". The example assumes a preset value for the client identifier.

```
ALTER SSO INTEGRATION sso_test SET
  client_id = different_database_app_id,
  default_group = "group2";
```

### Rename an SSO Integration

This example renames the SSO integration to sso\_test\_2.

```
ALTER SSO INTEGRATION sso_test RENAME TO sso_test_2;
```

## SYSTEM

## ALTER SYSTEM ALTER CONFIG SET

ALTER SYSTEM ALTER CONFIG SET sets an override for the configuration at the system scope. For an explanation of how configuration overrides work, see [Inspect the Current Configuration](#).

If the specified path to the configuration setting does not exist, it is considered a new configuration and appended to the collection of existing configurations. To reset a configuration override for a specific parameter, use the RESET keyword.

This configuration override is set in the base node configuration by default. To apply the override to a specific service role, prefix the key with the service role name.

### Syntax

```
-- To set a new configuration override:
ALTER SYSTEM
    ALTER CONFIG SET parameter_name [=] parameter_value [,...]

-- To reset a configuration override:
ALTER SYSTEM
    ALTER CONFIG RESET [ parameter_name [,...] ]
```

| Parameter       | DataType | Description                                                                                                                                                                                                                                                                                                           |
|-----------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| parameter_name  | string   | The name of a configuration parameter to alter. This parameter should be a string literal (i.e., enclosed in single quotes). Contact Ocient Support for details about altering configuration parameters.                                                                                                              |
| parameter_value | any      | The new value to be set for the specified configuration parameter. The data type of this value depends on the configuration parameter. This can be a string literal that can be cast to the data type of the configuration parameter. For example, the string literal 'true' is equivalent to the Boolean value true. |

### Examples

This example alters the base configuration of the system scope. The example sets the certificate name for each node in the entire system to `cert_name.crt`.

```
ALTER SYSTEM ALTER CONFIG SET 'certificateStore.cert' = 'cert_name.crt';
```

This example resets the certificate configuration for the entire system.

```
ALTER SYSTEM ALTER CONFIG RESET 'certificateStore.cert';
```

This example alters the configuration for service role `sql`. The `purgeCacheFrequency` setting determines the frequency in seconds to purge the result-set cache of stale values.

```
ALTER SYSTEM ALTER CONFIG SET 'sql.purgeCacheFrequency' = 600;
```

### System Scope Log Level Configuration

ALTER SYSTEM ALTER LOG LEVEL SET sets the log level for a particular logger for all nodes in the system. For an explanation of how configuration overrides work, see [Inspect the Current Configuration](#).

The logger name specified can be any string specifying a logger name, or ALL for the root logging config. See the [Log Monitoring](#) for more details.

#### Syntax

```
-- To set a new configuration override:
ALTER SYSTEM ALTER LOG LEVEL SET { <logger_parameters> [,...] }

<logger_parameters> ::=
    'ALL' | logger_name [=] { 'EDEBUG' | 'DEBUG' | 'VERBOSE' | 'INFO' | 'WARN' | 'ERROR' }

-- To reset a configuration override:
ALTER SYSTEM ALTER LOG LEVEL RESET [logger_name [, ...]]
```

| Parameter           | Data Type | Description                                                                                                                                                                                                                                                                                  |
|---------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <logger_parameters> | string    | Use ALL for the root logging config. Otherwise, the logger_name parameter can be any string specifying a logger name assigned to a severity level. See the <a href="#">Error Monitoring</a> page for more details. To reset a logger to the default log level (INFO), use the RESET keyword. |

#### Example

This example alters the log level at the system scope.

```
ALTER SYSTEM ALTER LOG LEVEL SET 'ALL' = 'DEBUG';
```

### ALTER SYSTEM RENAME TO

ALTER SYSTEM RENAME TO sets the system name. The system name appears in the sys.system\_information system catalog table. You can also see the current system name by using the [Other Functions and Expressions](#) function.

#### Syntax

```
ALTER SYSTEM RENAME TO system_name
```

| Parameter   | Data Type | Description                    |
|-------------|-----------|--------------------------------|
| system_name | string    | The name of the Ocient System. |

#### Example

```
ALTER SYSTEM RENAME TO "productionSystem";
```

### ALTER SYSTEM ALTER METRIC LEVEL

Alters the reporting level of various internal metrics across the system, including all nodes. For details about internal metrics, see [Statistics Monitoring](#).

To set metrics reporting for individual nodes, see [Cluster and Node Management](#).

#### Syntax

```

ALTER SYSTEM ALTER METRIC LEVEL
  { SET <set_options> | RESET <reset_options> }

<set_options> ::=
  [ LIKE | REGEX ] metric_to_alter [ ,... ] [=] { 'INFO' | 'DEBUG' }

<reset_options> ::=
  { [ LIKE | REGEX ] metric_to_reset [ ,... ] | ALL }

```

#### Set Metric Levels <set\_options>

The SET SQL statement assigns one or more specified metrics to the specified reporting level. You can assign metrics to INFO or DEBUG levels.

To set levels for a range of statistics, you can use wildcard characters with the LIKE keyword. Alternatively, you can use regular expressions with the REGEX keyword.

| Parameter       | DataType | Description                                                                                                                                                                                             |
|-----------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| metric_to_alter | string   | The names of one or more metrics to alter for logging. Enclose any metric names in single quotes. The supported logging levels for metrics are INFO and DEBUG. Either keyword must be in single quotes. |

#### Reset Metric Levels <reset\_options>

The RESET SQL statement reverts one or more specified metrics to the default reporting level.

You can use wildcard characters with the LIKE keyword. Alternatively, you can use regular expressions with the REGEX keyword.

Use the ALL keyword to revert all metrics to the default level.

| Parameter       | DataType | Description                                                                                                                    |
|-----------------|----------|--------------------------------------------------------------------------------------------------------------------------------|
| metric_to_reset | string   | The names of one or more metrics to reset by reverting their level to the default. To revert all metrics, use the ALL keyword. |

#### Examples

In this example, the ALTER SYSTEM SQL statement sets the `resultCache.queries` metric to the DEBUG level.

```

ALTER SYSTEM ALTER METRIC LEVEL
  SET 'resultCache.queries' = 'DEBUG';

```

This example uses the LIKE wildcard to capture any metrics prefixed by `localStorageService.device.smart.` and assign them to the INFO level.

```

ALTER SYSTEM ALTER METRIC LEVEL
  SET LIKE 'localStorageService.device.smart.%' 'INFO';

```

Similar to the previous example, this SQL statement uses regular expressions to capture any metrics prefixed by `localStorageService.device.smart..`

```
ALTER SYSTEM ALTER METRIC LEVEL
SET REGEX 'localStorageService.device.smart.*' 'INFO';
```

This example alters the level of multiple metrics.

```
ALTER SYSTEM ALTER METRIC LEVEL
SET 'resultCache.queries' = 'DEBUG',
   'resultCache.data' = 'INFO',
   LIKE 'localStorageService.device.smart.%' 'INFO';
```

This example resets the same metrics.

```
ALTER SYSTEM ALTER METRIC LEVEL
RESET 'resultCache.queries',
      'resultCache.data',
      LIKE 'localStorageService.device.smart.%;
```

## ALTER SYSTEM ALTER SECURITY

Set security settings using the ALTER SYSTEM ALTER SECURITY SQL statement. Replace <security\_setting> with the security setting and <value> with the value.

You can inspect the current values of system configuration settings using the sys.config and sys.node\_config system catalog tables.

### Syntax

```
ALTER SYSTEM ALTER SECURITY <security_setting> [=] <value>
```

| Parameter        | DataType | Description                                                                                                                                |
|------------------|----------|--------------------------------------------------------------------------------------------------------------------------------------------|
| security_setting | string   | The security setting with values:password_minimum_length`password_comp details about these values, seeDatabase Password Security Settings. |
| value            | numeric  | An integer to represent one of the security settings. For details about this value, seeDatabase Password Security Settings.                |

### Examples

#### Set a Security Setting for All Databases

Set the password lifetime to 30 days for the entire system.

```
ALTER SYSTEM ALTER SECURITY password_lifetime_days = 30;
```

#### Set Multiple Security Settings

Set multiple security settings in a single SQL statement. In this case, set the password lifetime to 30 days and the minimum password character length to 12. As with other similar SQL statements, the = character is optional.

```
ALTER SYSTEM ALTER SECURITY
password_lifetime_days 30,
password_minimum_length 12;
```

## Related Links

[Other Functions and Expressions](#)

[CREATE TABLE SQL Statement Examples](#)

[Database Password Security Settings](#)

### 10.21.2 Database, Tables, Views, and Indexes

This group of DDL SQL statements allows database administrators to manage elements of the database and schema design. Database administrators can create and modify databases, including SSO authentication settings. These statements also create, modify, drop, and export Tables, Views, and Indexes in the database and truncate segments on a Table. Additional statements are available for unique settings like segment redundancy, compression settings, and streamloader properties.

## DATABASE

### CREATE DATABASE

CREATE DATABASE creates a new database. The database name must be distinct from the name of any existing database in the system.

To create a database, you must have the CREATE DATABASE privilege for the current system.

#### Syntax

```
CREATE DATABASE [ IF NOT EXISTS ] database_name
```

| Parameter     | Type   | Description                                                                                                                                                                            |
|---------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| database_name | string | A unique identifier for the database. The system generates an error if a duplicate name is provided. <b>system</b> is a reserved database name and can neither be created nor dropped. |

#### Example

To create a new database named **ocient**:

```
CREATE DATABASE ocient;
```

### DROP DATABASE

DROP DATABASE removes an existing database. This SQL statement also disconnects all users currently connected to the database.

To remove a database, you must have the DROP DATABASE privilege for the current database.

You cannot drop a database while it has any PIPELINE in a running status.

The DROP DATABASE SQL statement removes the existing database and all created users, tables, and views. This action cannot be undone.

#### Syntax

```
DROP DATABASE [ IF EXISTS ] database_name [, ...]
```

| Parameter     | Type   | Description                                                                                                                                                                                                                              |
|---------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| database_name | string | An identifier for the database to be dropped. You can drop multiple databases by specifying additional database names and separating each with commas. <b>system</b> is a reserved database name and can neither be created nor dropped. |

### Example

Remove an existing database named **ocient**.

```
DROP DATABASE ocient;
```

## ALTER DATABASE

### ALTER DATABASE RENAME

**ALTER DATABASE RENAME** renames an existing database.

To rename a database, you must have the **ALTER DATABASE** privilege for the database.

### Syntax

```
ALTER DATABASE old_database_name RENAME TO new_database_name
```

| Parameter         | Type   | Description                                    |
|-------------------|--------|------------------------------------------------|
| old_database_name | string | The old identifier of the database for rename. |
| new_database_name | string | The new identifier of the database for rename. |

### Example

Rename an existing database named **oracle** to **ocient**.

```
ALTER DATABASE oracle RENAME TO ocient;
```

### ALTER DATABASE SET SSO INTEGRATION

**ALTER DATABASE SET SSO INTEGRATION** configures the database to authenticate using an external SSO provider. This SSO integration is the default for connections unless you use a connectivity pool or specify a different provider.

To set a connection, you must be a system-level user or a database administrator and have an open connection to the database.

This SQL statement is an alias for [Database, Tables, Views, and Indexes](#).

See [Authentication Methods](#) for details about configuring SSO protocols.

If your Ocient® System is version 25.0 or later, you can create multiple SSO integrations for each database. An SSO integration assigned to the database by the **ALTER DATABASE** SQL statement is the primary SSO connection, unless you connect to the database with a connectivity pool that has a different SSO integration assigned to it.

### Syntax



```
ALTER DATABASE database
  ALTER SSO INTEGRATION sso_name
```

| Parameter | Type   | Description                                       |
|-----------|--------|---------------------------------------------------|
| database  | string | The identifier of the database for configuration. |
| sso_name  | string | The identifier of the SSO integration to use.     |

### Example

This example sets an example database to use the SSO integration named `sso_test`.

```
ALTER DATABASE example_database
  SET SSO INTEGRATION sso_test;
```

### ALTER DATABASE ALTER SSO INTEGRATION

`ALTER DATABASE ALTER SSO CONNECTION` configures the database to authenticate using an external SSO provider. This SSO integration is the default for connections unless you use a connectivity pool or specify a different provider.

To alter a connection, you must be a system-level user or a database administrator and have an open connection to the database.

This SQL statement is an alias for [Database, Tables, Views, and Indexes](#).

See [Authentication Methods](#) for details about configuring SSO protocols.

If your Ocient System is version 25.0 or later, you can create multiple SSO integrations for each database. An SSO integration assigned to the database by the `ALTER DATABASE` SQL statement is the primary SSO connection, unless you connect to the database with a connectivity pool that has a different SSO integration assigned to it.

### Syntax

```
ALTER DATABASE database
  ALTER SSO INTEGRATION sso_name
```

| Parameter | Type   | Description                                       |
|-----------|--------|---------------------------------------------------|
| database  | string | The identifier of the database for configuration. |
| sso_name  | string | The identifier of the SSO integration to use.     |

### Example

This example alters an example database to use the SSO integration named `sso_test`.

```
ALTER DATABASE example_database
  ALTER SSO INTEGRATION sso_test;
```

### ALTER DATABASE REMOVE SSO INTEGRATION

`ALTER DATABASE REMOVE SSO INTEGRATION` removes an existing SSO integration as the default connection protocol for the database. This action effectively undoes the [Database, Tables, Views, and Indexes](#) SQL statement.

To remove a connection, you must be a system-level user or a database administrator.

### Syntax

```
ALTER DATABASE database REMOVE SSO INTEGRATION
```

| Parameter | Type   | Description                                  |
|-----------|--------|----------------------------------------------|
| database  | string | The identifier of the database for deletion. |

### Example

Remove the default connection from the database named `example_database`.

```
ALTER DATABASE example_database REMOVE SSO INTEGRATION;
```

### ALTER DATABASE ALTER SECURITY

Sets the security settings at the database level using the `ALTER DATABASE ALTER SECURITY` SQL statement. Replace `<security_setting>` with the security setting and `<value>` with the value.

### Syntax

```
ALTER DATABASE database ALTER SECURITY <security_setting> [=] <value>
```

| Parameter        | Data Type | Description                                                                                                                                                                          |
|------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| database         | string    | The identifier of the database for setting security settings.                                                                                                                        |
| security_setting | string    | The security setting with values: <code>password_minimum_length</code> , <code>password_complexity</code> . For details about these values, see Database Password Security Settings. |
| value            | numeric   | An integer to represent one of the security settings. For details about this value, see Database Password Security Settings.                                                         |

### Example

Set the password lifetime to 20 days for the database `example_db`.

```
ALTER DATABASE example_db
  ALTER SECURITY password_lifetime_days = 20;
```

## TABLE

### CREATE TABLE

Creates a new table in the current database.

The table name must be distinct from the name of any existing tables in the database unless the `REPLACE` keyword is specified. To use `REPLACE` in the `CREATE TABLE` statement, you must have `DELETE` privileges.

By default, columns are nullable unless otherwise specified.

For faster query results, you can define one TimeKey® for the table, which must be a timestamp, date, or time column, with a specified bucket resolution. Tables with a specified TimeKey can perform query operations faster, especially if they involve time filtering.

You can specify a Clustering Key composed of one or more fixed-length columns. Designating columns as cluster keys that are frequently referenced in queries can greatly improve performance.

For details about defining TimeKeys and clustering indexes, see [TimeKeys and Clustering Keys](#).

See the [Data Types](#) section for table-supported data types.

To create a table, you must have the CREATE TABLE privilege for the current database.

For examples, see [CREATE TABLE SQL Statement Examples](#).

### Syntax

```
CREATE [ OR REPLACE ] TABLE [ IF NOT EXISTS ] table_name
  [ ( <column_definition> [, ...] | <clustering_definition> ) ]
  [ [ WITH ] <create_option> [, ...] ]
  [ AS (query) ]

<column_definition> ::=
  column_name [ data_type ] [ <timekey_definition>
  | <column_constraint> [, ...] ]

<timekey_definition> ::=
  TIME KEY BUCKET (bucket_granularity, bucket_value) [ <column_constraint> ]

<column_constraint> ::=
  TIME KEY BUCKET( bucket_granularity, bucket_value )
  | [ NOT ] NULL
  | DEFAULT literal
  | COMMENT comment
  | COMPRESSION GDC [ (compression_value), EXISTING schema_name ]
  | COMPRESSION [ compression_scheme ]

<clustering_definition> ::=
  CLUSTERING KEY key_name (ck_col1, ck_col2 [, ...])
    [, INDEX index_name (idx_col1, idx_col2 [, ...])
    [, INDEX ... ] ]

<create_option> ::=
  STORAGE SPACE storage_space_name
  | SEGMENT SIZE segment_value
  | REDUNDANCY segment_part (redundancy_scheme)
  | STREAMLOADER_PROPERTIES streamloader_json
  | COMMENT '<string>'
```

| Parameter  | Type   | Description                                                                                                                                                                            |
|------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| table_name | string | A unique identifier for the table. <code>table_name</code> must be distinct from the name of any existing tables in the database unless the <code>REPLACE</code> keyword is specified. |
| query      | string | A <code>SELECT</code> query used to load data into the newly created table. For details, see <a href="#">Database, Tables, Views, and Indexes</a> .                                    |

### Column Definition (<column\_definition>)

The parameters listed here are required for defining each column in a table.

| Parameter                | Type   | Description                                                                                         |
|--------------------------|--------|-----------------------------------------------------------------------------------------------------|
| <code>column_name</code> | string | An identifier for a column to be included in the newly created table.                               |
| <code>data_type</code>   | string | The data type of a specified column. For a list of supported data, see <a href="#">Data Types</a> . |

#### Clustering Key and Index Definition (<clustering\_definition>)

The parameters listed here are required for defining a Clustering Key or clustering indexes on a table.

For details about how to apply clustering columns, see [TimeKeys and Clustering Keys](#).

#### TimeKey Definition (<timekey\_definition>)

This syntax example is for a single TimeKey column, which can be included in the column definition of a **CREATE TABLE** statement. For the full syntax, see [Database, Tables, Views, and Indexes](#).

For details about using TimeKeys, see [TimeKeys and Clustering Keys](#).

#### Column Constraint (<column\_constraint>)

The parameters listed here include constraints and other configurations for individual columns. For best performance, one column with date or time data in each table should be defined as the **TIME KEY** with a specified `bucket_value`.

For details about TimeKey columns, see [TimeKeys and Clustering Keys](#).

| Parameter                       | Type    | Description                                                                                                                                                                                                                                |
|---------------------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>bucket_granularity</code> | numeric | The granularity of the TimeKey column based on the specified time type in <code>bucket_value</code> .                                                                                                                                      |
| <code>bucket_value</code>       | string  | The time type to parse the TimeKey column. Supported values include: [ <b>DAY</b> , <b>HOURLY</b> , <b>MINUTE</b> , <b>SECOND</b> ] For example, <code>BUCKET(1, DAY)</code> sets the time-bucket granularity to a fixed width of one day. |

| Parameter                      | Type                            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------------------|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>literal</code>           | depends on the column data type | If specified as a constraint, sets the default value for the column. The value must be a literal that is enclosed in quotes. Do not include an expression with this argument to cast this value to the data type of the column. The system automatically attempts type coercion to convert the literal to the data type of the column. For example, this CREATE TABLE statement includes a default value for its Universally Unique Identifier (UUID) column: <code>CREATE TABLE example_table (`col1` UUID NOT NULL DEFAULT '00000000-0000-0000-0000-000000000000');</code> The supported geospatial data types POINT, LINESTRING, and POLYGON require WKT formatting. For formatting examples, see the Well-known text representation of geometry. |
| <code>comment</code>           | string                          | An optional comment for the column.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>compression_value</code> | numeric                         | An integer value of either 1, 2, or 4 is used to define the storage of COMPRESSION GDC. The compression of the values is defined as follows:- A <code>compression_value</code> of 1 can hold up to 255 unique values.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

- A `compression_value` of 2 can hold up to 65535 unique values.

- A `compression_value` of 4 can hold millions of unique values. For tuple columns, compression is specified

`schema_name` | string | Fully qualified column name or a column name if you specify a column in the same table. If EXISTING is specified as a compression constraint, the compression GDC reuses the system lookup table for the `schema_name` column, rather than creating a new one. The existing column must be in the same database as the new column. `schema_name` should include the schema, table, and column names, each separated by periods and enclosed in double-quotation marks, for example: `"schema_name"."table_name"."column_name"` If an existing column is specified, any new column that uses the existing column's system lookup table must be deleted before the existing column can be deleted. |

`compression_scheme` | string | The type of compression used for the column. Supported values include: [COMPRESSION NONE | COMPRESSION ZSTD | COMPRESSION DYNAMIC] If no compression setting is specified, the compression defaults to COMPRESSION DYNAMIC for fixed-length columns and COMPRESSION NONE for variable-length columns. COMPRESSION ZSTD applies to VARCHAR columns as well as other data types. For the COMPRESSION DYNAMIC setting, the Ocint System applies LZ4 compression only if the column data is dynamically determined to be compressible. For fixed-length columns, COMPRESSION DYNAMIC applies delta-delta compression. For details about Ocint-supported compression schemes, see Table Compression Options. |

### Create Option (<create\_option>)

The parameters listed here include various options to configure table storage space, segments, redundancy, stream-loading, and indexes.

| Parameter                       | Type    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>storage_space_name</code> | string  | An identifier for the <code>STORAGESPACE</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>segment_value</code>      | numeric | A value to define the size of the segment.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>segment_part</code>       | string  | Specifies the segment part redundancy of the table. Supported values include: { <code>DATA</code>   <code>MANIFEST</code>   <code>INDEX</code>   <code>SUMMARY_STATS</code>   <code>STATS</code> } These settings are defined as follows: <code>DATA</code> : The actual data for the table. <code>MANIFEST</code> : Header information stored about the data, which describes how to locate any specified cluster of rows within the data. <code>INDEX</code> : The index of the data used for quicker lookups and better query performance. <code>SUMMARY_STATS</code> : A collection of statistics on the data that includes compression, row count, and average column size. <code>STATS</code> : Used in the optimizer, the probability density function and combinable distinct estimators used to make better optimizations to query plans. |
| <code>redundancy_scheme</code>  | string  | The redundancy scheme. Supported values include: { <code>COPY</code>   <code>PARITY</code> } These settings are defined as follows: <code>COPY</code> : A copy of the bytes is stored throughout the storage cluster to ensure redundancy. This option uses more storage but is faster during rebuilds and node outages. <code>PARITY</code> : Using the parity encoding specified on the storage cluster, this option uses parity bits to ensure redundancy for the data. This option uses less storage but is slower during rebuilds and node outages.                                                                                                                                                                                                                                                                                           |
| <code>streamloader_json</code>  | string  | A JSON string that defines the streamloader parameters. For details, see <a href="#">Database, Tables, Views, and Indexes</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

### Example

This example creates a new table in the current database and schema named `trades`.

The table uses the `TIMESTAMP` column `created_at` as the TimeKey, with the granularity set at 1 hour. The columns `ticker_symbol` and `t_type` are defined as the table clustering keys. The example also includes a streamloader property `pageQueryExclusionDuration` to delay how soon data pages that were recently added can be included in query results.

```
CREATE TABLE trades (
  id UUID NOT NULL,
  ticker_symbol VARCHAR(255) NOT NULL COMPRESSION GDC(2),
  t_type VARCHAR(255) NOT NULL COMPRESSION GDC(1),
  raw_ticker_data VARCHAR(255),
  created_at TIMESTAMP NOT NULL TIME KEY BUCKET(1, HOUR),
  array_of_tuples tuple<<varchar(2040), byte, bigint, double, timestamp, date, time, decimal(3,2), st_point>>,
  CLUSTERING INDEX idx_ticker_symbol_type (ticker_symbol, t_type),
  INDEX idx_type (t_type)
)
STORAGESPACE ocient_storage,
REDUNDANCY DATA (PARITY),
REDUNDANCY MANIFEST (COPY),
STREAMLOADER_PROPERTIES '{
  "pageQueryExclusionDuration" : "2700s"
}';
```

## CREATE TABLE AS SELECT (CTAS)

CTAS provides the ability to create and load a new table from the result of a query on one or more existing tables. The first column of the query result maps to the first column of the new table definition, the second column maps to the second column of the new table, and so on. The new table is available for querying after it has been created, and the entire result set from the query has been loaded into the table. When you receive a response to the CTAS SQL statement, the load is complete and the table is ready.

When you create a table from a **SELECT** SQL statement, the schema for the table can be automatically determined based on the query results. You can override this behavior with an alternative schema, provided the query results can automatically be cast to the target column types. CTAS also supports all syntax options for the new table. CTAS does not support default values and explicit nullable definitions on the column of the table.

CTAS statements support secondary and prefix indexes.

To create a table, you must have both the **CREATE TABLE** privilege for the current database and the **SELECT** privilege on all referenced tables and views.

For syntax and parameter information, see [Database, Tables, Views, and Indexes](#).

### Default Table Definitions

By default, a new table created with a CTAS statement retains column names, data types, and nullable definitions from the queried table. You can override this configuration with alternate table definitions in the CTAS statement.

Tables created by a CTAS statement do not inherit some table definitions from the original table, including the following:

- TimeKey
- Clustering Key and Clustering indexes
- Secondary indexes
- Column compression
- Optional table configurations (see [Database, Tables, Views, and Indexes](#))

To include these table definitions, you must explicitly specify them in the CTAS statement.

## Examples

These CTAS examples select columns from the `original_table` table that this **CREATE TABLE** statement defines. This table contains these columns:

- `col_int` — Non-nullable integer with the default value 123456789
- `col_bigint` — Non-nullable 8-byte signed integer

- `col_id` — Non-nullable integer
- `col_point` — Non-nullable point with the default value `POINT(0 0)`
- `col_timestamp` — Non-nullable TimeKey with granularity set at 1 day
- `col_varchar` — Variable-length character string with a maximum length of 255 characters and Zstandard compression

The table has a Clustering Key `ck` using the `col_bigint` and `col_id` columns. It also has two secondary indexes: a hash index `idx_01` on the `col_varchar` column and a spatial index on the `idx_02` column.

```
CREATE TABLE original_table (
  col_int INT NOT NULL DEFAULT 123456789,
  col_bigint BIGINT NOT NULL,
  col_id INT NOT NULL,
  col_point POINT NOT NULL DEFAULT 'POINT(0 0)',
  col_timestamp TIMESTAMP TIME KEY BUCKET (1,DAY) NOT NULL,
  col_varchar VARCHAR(255) COMPRESSION ZSTD,
  CLUSTERING KEY ck (col_bigint, col_id)
)
WITH
  INDEX idx_01 (col_varchar) USING HASH,
  INDEX idx_02 (col_point) USING SPATIAL;
```

### CTAS Using All Columns from a Base Table

This example shows a basic CTAS statement that inherits most of its table definition from the `original_table` table.

```
CREATE TABLE basic_ctas AS (
  SELECT * FROM original_table );
```

The new `basic_ctas` table includes all the columns and data types from the original table definition. However, it does not include the segment keys, indexes, or the compression on the `col_varchar` column. The `EXPORT TABLE` SQL statement shows the differences in the `basic_ctas` table.

```
EXPORT TABLE basic_ctas;
```

#### Output

```
CREATE TABLE basic_ctas (
  "col_int" INT NOT NULL,
  "col_bigint" BIGINT NOT NULL,
  "col_id" INT NOT NULL,
  "col_point" POINT NOT NULL,
  "col_timestamp" TIMESTAMP NOT NULL,
  "col_varchar" VARCHAR(536870912) COMPRESSION DYNAMIC NULL
)
REDUNDANCY cde (PARITY),
REDUNDANCY column_metadata (COPY),
REDUNDANCY data (PARITY),
REDUNDANCY index (PARITY),
REDUNDANCY pdf (PARITY),
REDUNDANCY skip_lists (COPY),
REDUNDANCY stats (PARITY),
REDUNDANCY summary_stats (PARITY),
REDUNDANCY table_of_contents (COPY),
STORAGESPACE "ss0",
SEGMENTSIZ 4;
```

In this output, the table options for `REDUNDANCY`, `STORAGESPACE`, and `SEGMENTSIZ` are all default table settings.



### CTAS Using a Full Table Definition

This example CTAS statement includes a more detailed table definition. The definition includes new columns for the TimeKey, Clustering Key, and secondary indexes. The example also makes various changes from the `original_table` schema:

- Different column default value
- Different compression scheme (dynamic compression)
- New TimeKey granularity of 1 hour
- Three columns in the Clustering Key
- Different secondary index types (NGRAM and SPATIAL)

```
CREATE TABLE complex_ctas (
  col_amt INT NOT NULL,
  col_phone BIGINT NOT NULL COMPRESSION DYNAMIC,
  col_id INT NOT NULL,
  col_point POINT NOT NULL DEFAULT 'POINT(0 0)',
  col_timestamp TIMESTAMP TIME KEY BUCKET (1,HOUR) NOT NULL,
  col_varchar VARCHAR(255) COMPRESSION DYNAMIC,
  CLUSTERING KEY ck (col_amt, col_phone, col_id)
)
WITH
  INDEX idx_01 (col_varchar) USING NGRAM(3),
  INDEX idx_02 (col_point) USING SPATIAL
AS (
  SELECT * FROM original_table );
```

### CTAS Using a Subset of Table Columns

This example selects only a subset of columns, `col_int`, `col_bigint`, and `col_id`, from the original table to insert into the new `subset` table. The example also specifies alternate table options for `REDUNDANCY` and `SEGMENTSIZ`.

```
CREATE TABLE subset (
  col_amt INT NOT NULL,
  col_phone BIGINT NOT NULL,
  col_id INT NOT NULL
)
WITH
  REDUNDANCY cde (PARITY),
  SEGMENTSIZ 3
AS (
  SELECT
    col_int,
    col_bigint,
    col_id
FROM
  original_table );
```

Due to limitations of the JDBC API, the reported modified row count might not be accurate for tables larger than two billion rows.

### CTAS Using Transformations on Columns

This example performs various transformation functions on the original columns as it selects them for the new table. These include:

- `col_int_multiply` column is the multiplication of the `col_int` values by 10.
- `col_month_add` column is the result of adding three months to each `col_timestamp` column value.

- `col_year` column is the extraction of the year value from each `col_timestamp` column value.
- `col_substring` column contains the first three characters from each `col_varchar` column value.

```
CREATE TABLE calc_table (
  col_int_multiply INT NOT NULL,
  col_month_add TIMESTAMP NOT NULL,
  col_year INT NOT NULL,
  col_substring VARCHAR(255)
)
AS (
  SELECT
    col_int * 10,
    ADD_MONTHS(col_timestamp, 3),
    YEAR(col_timestamp),
    SUBSTRING(col_varchar, 1, 3)
FROM
  original_table );
```

## CTAS USING LOADERS

Specify one or more Loader Nodes for executing the CTAS SQL statement. If you do not use this option, the Ocient System uses all Loader Nodes that are live to execute the SQL statement.

This statement is useful for managing loading operations, particularly when balancing multiple loads of different sizes and resource requirements. Alternatively, this statement can also help simplify small batch loads by sourcing the data from a single Loader Node.

### Syntax

| Parameter                 | Type   | Description                                                                                                                                                                                                                                                                                                 |
|---------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>streamloader</code> | string | A unique name for the Loader Node. Identify the names of Loader Nodes from <code>thesys.nodestable</code> by using this query: <code>SELECT name FROM sys.nodes</code> ; If the name of the Loader Node contains special characters, you must enclose it in quotes, such as <code>"stream-loader1"</code> . |
| <code>query</code>        | string | A <code>SELECT</code> query that defines values or a table and any of its columns to use for data in the specified <code>table_name</code> table.                                                                                                                                                           |

For the query to execute successfully, the specified names of the Loader Nodes must:

- Identify nodes that are live.
- Identify nodes that have the Loader role.

### Examples

Create a table named `my_schema.my_ctas_table_2` with a clustering index named `idx` on the `int_col` column with the values in the `int_col` column in the table named `my_schema.my_table`. Use the Loader Node named `stream-loader1` to execute this SQL statement.

```
CREATE TABLE my_schema.my_ctas_table_2 (
  CLUSTERING INDEX idx (int_col)
)
```

```

USING LOADERS "stream-loader1"
AS (SELECT int_col FROM my_schema.my_table);

```

In this example, execute the same CTAS SQL statement with two Loader Nodes named `stream-loader2` and `stream-loader3`.

```

CREATE TABLE my_schema.my_ctas_table_2 (
    CLUSTERING INDEX idx (int_col)
)
USING LOADERS "stream-loader2","stream-loader3"
AS (SELECT int_col FROM my_schema.my_table);

```

## DROP TABLE

**DROP TABLE** removes one or more existing tables in the current database, along with all associated views.

This action cannot be undone.

To remove a table, the logged-in user must be a system-level user or have the **DELETE TABLE** privileges for the table.

### Syntax

```

DROP TABLE [ IF EXISTS ] table_name [, ...]

```

| Parameter  | Type   | Description                                                                                                                           |
|------------|--------|---------------------------------------------------------------------------------------------------------------------------------------|
| table_name | string | A unique identifier for the table. You can drop multiple tables by specifying additional table names and separating each with commas. |

### Examples

This example drops an existing table in the current database and schema named `employees`.

```

DROP TABLE employees;

```

In this example, drop two tables in the current database and schema named `employees` and `departments`.

```

DROP TABLE employees, departments;

```

When you drop multiple tables, and none of them exist in the database, the database returns an error for each missing table. Use the **IF EXISTS** statement to convert the error to a warning. If you execute the **DROP TABLE** statement and only some of the tables exist while other tables are missing, the database drops the existing tables and returns warnings for each missing table.

## ALTER TABLE

### ALTER TABLE RENAME

**ALTER TABLE RENAME** renames an existing table.

To rename a table, you must have the **ALTER TABLE** privilege for the table.

### Syntax

```

ALTER TABLE [ IF EXISTS ] old_table_name RENAME TO new_table_name

```

| Parameter      | Type   | Description                             |
|----------------|--------|-----------------------------------------|
| table_name     | string | A unique identifier for the table.      |
| old_table_name | string | The name of the table to alter.         |
| new_table_name | string | The new name to replace old_table_name. |

### Example

This example renames an existing table in the current database and schema named `us.employees` to `mid_west_employees`:

```
ALTER TABLE us.employees RENAME TO mid_west_employees;
```

This example renames an existing table in the current database named `us.employees` to `north_america.employees`:

```
ALTER TABLE us.employees RENAME TO north_america.employees;
```

### ALTER TABLE RENAME COLUMN

ALTER TABLE RENAME COLUMN renames an existing column.

To rename a column, you must have the ALTER TABLE privilege for the table.

### Syntax

```
ALTER TABLE [ IF EXISTS ] table_name RENAME COLUMN old_column_name TO new_column_name
```

| Parameter       | Type   | Description                                     |
|-----------------|--------|-------------------------------------------------|
| table_name      | string | A unique identifier for the table.              |
| old_column_name | string | The name of the table column to alter.          |
| new_column_name | string | The new column name to replace old_column_name. |

### Example

This example renames an existing column name in the table `employees` in the current database and schema to `first_name`:

```
ALTER TABLE employees RENAME COLUMN name TO first_name;
```

### ALTER TABLE ADD COLUMN

ALTER TABLE ADD COLUMN adds a new column to the table.

To add a column, you must have the ALTER TABLE privilege for the table.

New columns must either be nullable or specify a default value.

For a defined list of column parameters, see [Database, Tables, Views, and Indexes](#). For constraints, see [Database, Tables, Views, and Indexes](#).

### Syntax

```
ALTER TABLE [ IF EXISTS ] table_name ADD COLUMN <column_definition>;
```

```
<column_definition> ::=
    column_name [ data_type ] [ <column_constraint> [, ...] ]
```

```
<column_constraint> ::=
    TIME KEY BUCKET(bucket_granularity [, bucket_value ] )
    | [ NOT ] NULL
```

```
| DEFAULT literal
| COMMENT comment
| COMPRESSION GDC [ (compression_value), EXISTING schema_name ]
| COMPRESSION [ compression_scheme ]
```

| Parameter  | Type   | Description                        |
|------------|--------|------------------------------------|
| table_name | string | A unique identifier for the table. |

## Examples

This example adds a BIGINT column to the `employees` table in the current database and schema with the default value of 0.

```
ALTER TABLE employees ADD COLUMN
    new_column BIGINT
    NOT NULL
    DEFAULT 0;
```

This example adds a column that is nullable.

```
ALTER TABLE employees ADD COLUMN
    new_column BIGINT
    NULL;
```

## ALTER TABLE ALTER COLUMN COMPRESSION

ALTER TABLE ALTER COLUMN COMPRESSION alters an existing column in the table to change its compression scheme.

Supported compression schemes are COMPRESSION NONE, COMPRESSION DYNAMIC, and COMPRESSION ZSTD.

Altering the compression setting of a column only affects compression for data loaded after you execute the SQL statement.

For details about Ocint-supported compression schemes, see [Table Compression Options](#).

## Syntax

```
ALTER TABLE [ IF EXISTS ] table_name ALTER COLUMN column_name
    SET COMPRESSION [ compression_scheme ];
```

| Parameter   | Type   | Description                                           |
|-------------|--------|-------------------------------------------------------|
| table_name  | string | The name of the table containing the column to alter. |
| column_name | string | The name of the column to alter.                      |

| Parameter                       | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>compression_scheme</code> | string | <p>Supported values for the compression schemes are:<br/> <b>COMPRESSION NONE</b> specifies no compression applied.<br/> <b>COMPRESSION DYNAMIC</b> applies only if the column data is dynamically determined to be compressible.<br/> <b>COMPRESSION ZSTD</b> applies to VARCHAR columns as well as other data types. For this option only, you can specify these additional parameters:<br/> <b>compression_level</b>—≤ This value signifies how much compression the data receives. The default value is 0. The full range of values is from -7 through 15. The database uses less memory when this value is lower, whereas more memory when this value is larger. Larger values provide better compression.<br/> <b>dictionary_size</b>—≤ Dictionary size specified as a positive integer that signifies the size of the shared compression dictionary in bytes. The default value is 32768 (32K). The full range of values is from 4096 (4K) through 1048576 (1MB). This value denotes the amount of memory consumed during segment generation. In general, larger values provide better compression but use more memory.</p> |

### Examples

This example alters the compression scheme for the column `employee_name` in the table `employees` in the current database and schema.

```
ALTER TABLE employees ALTER COLUMN employee_name
SET COMPRESSION LZ4;
```

This example alters the compression scheme to Zstandard for the column `employee_name` in the table `employees` in the current database and schema.

```
ALTER TABLE employees ALTER COLUMN employee_name
SET COMPRESSION ZSTD
compression_level=5,
dictionary_size=32768;
```

### ALTER TABLE ALTER REDUNDANCY

`ALTER TABLE ALTER REDUNDANCY` alters the segment part redundancy for future segments of an existing table.

Note that altering a segment part redundancy setting only affects data loaded after applying the SQL statement.

```
ALTER TABLE [ IF EXISTS ] table_name ALTER REDUNDANCY segment_part (redundancy_scheme)
```

| Parameter         | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| table_name        | string | The name of the table that you want to alter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| segment_part      | string | Specifies the segment part redundancy of the table. Supported values include:{ DATA   MANIFEST   INDEX   PDF   CDE   STATS }These settings are defined as follows:DATA: The actual data for the table.MANIFEST: Header information stored about the data, which describes how to locate any given cluster of rows within the data.INDEX: The index of the data used for quicker lookups and better query performance.STATS: Used in the optimizer, the probability density function and combinable distinct estimators used to make better optimizations to query plans. |
| redundancy_scheme | string | The redundancy scheme. Supported values include:{ COPY   PARITY }These settings are defined as follows:COPY— The system stores a copy of the bytes throughout the storage cluster to ensure redundancy. This option uses more storage but is faster during rebuilds and node outages.PARITY— The system uses the parity encoding specified on the storage cluster, and uses parity bits to ensure redundancy for the data. This option uses less storage but is slower during rebuilds and node outages.                                                                 |

### Example

This example alters the STATS part to COPY redundancy.

```
ALTER TABLE employees ALTER REDUNDANCY STATS (COPY);
```

### ALTER TABLE DROP COLUMN

ALTER TABLE DROP COLUMN drops an existing column from the table.

You cannot remove the TimeKey column and the clustering key columns from the table.

When you remove a column, the database does not remove or free any actual data.

```
ALTER TABLE [ IF EXISTS ] table_name DROP COLUMN column_name [ IF EXISTS ]
```

| Parameter   | Type   | Description                     |
|-------------|--------|---------------------------------|
| table_name  | string | The name of the table to alter. |
| column_name | string | The name of the column to drop. |

### Example

This example removes a column named `address` from the table `employees`.

```
ALTER TABLE employees DROP COLUMN address;
```

### ALTER TABLE STREAMLOADER\_PROPERTIES

`ALTER TABLE STREAMLOADER_PROPERTIES` resets the table streamloader properties to the provided string. The properties string must be in valid JSON format.

The database registers streamloader changes dynamically. Therefore, you do not need to restart nodes or take other actions for the changes to take effect.

Any properties not specified in the string default to the system-wide setting.

```
ALTER TABLE [ IF EXISTS ] table_name STREAMLOADER_PROPERTIES streamloader_json
```

| Parameter         | Type   | Description                                                                                  |
|-------------------|--------|----------------------------------------------------------------------------------------------|
| table_name        | string | The name of the table to alter.                                                              |
| streamloader_json | string | The streamloader properties to alter. See this table for a list of all supported properties. |

### Configuring Streamloader Properties

`STREAMLOADER_PROPERTIES` is a field on the table metadata that must be written as a JSON string in order to be read properly. The database can dynamically render any changes to `STREAMLOADER_PROPERTIES` with the `ALTER TABLE` SQL statement. You can set Loader Node properties for a new table as a parameter in the `CREATE TABLE` SQL statement. You do not need to restart the database node for the changes to take effect.

#### Per-Table Streamloader Properties

| Parameter                  | Data type                                                                                                                                                                                                                         | Description                                                                                                                                                                                                                                                                                                         |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pageQueryExclusionDuration | Integer in nanoseconds(ns) or a string with the suffix <code>ns</code> , <code>us</code> , <code>ms</code> , or <code>s</code> appended. For example: <code>"10s"</code> = 10 seconds, <code>"1000us"</code> = 1,000 microseconds | Per-table configuration for the time interval for pages that should be excluded from queries. The database excludes pages with time column values that are greater than the duration of the query. A value of 0 means the database does not exclude any pages. By default, this value is set to 0 if not specified. |

### Example

This example sets the Loader Node properties of the table `employees` to `{"pageQueryExclusionDuration" : "30s"}`. This means that any pages added less than 30 seconds ago will not be included in query results.

```
ALTER TABLE employees
  STREAMLOADER_PROPERTIES '{
    "pageQueryExclusionDuration" : "30s"
  }';
```



## ALTER TABLE DISABLE INDEX

The **ALTER TABLE DISABLE INDEX** statement instructs future queries not to use the specified indexes, but existing segments and new segments continue to have the index available in case you enable the index again.

All secondary indexes except for secondary clustering key indexes can be disabled. Trying to disable other types of indexes generates an error.

You can specify the index by name or UUID.

### Syntax

```
ALTER TABLE [ IF EXISTS ] table_name DISABLE INDEX
    { index_name_or_uuid | IN (index_name_or_uuid [,...]) }
```

| Parameter          | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| table_name         | string | The name of the table to alter.                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| index_name_or_uuid | string | The name or UUID of the index to disable. If you specify the index by name, the name must match an existing index. If you specify a UUID instead, it does not have to match an existing index. Therefore, you can disable a dropped index using its UUID, which ensures it is not used within old segments that were loaded with the index. You can get a list of index names and UUIDs in your database by referencing the sys.indexes table in the system catalog. |

### Examples

This example disables an existing index named `current_idx` on the table `employees`.

```
ALTER TABLE employees DISABLE INDEX current_idx;
```

This example disables an existing or dropped index with the UUID `5c15d8de-36fa-4055-9bdc-3f1750aabeea0`.

```
ALTER TABLE employees DISABLE INDEX '5c15d8de-36fa-4055-9bdc-3f1750aabeea0';
```

This example disables both indexes `current_idx` and `other_idx` on the table `employees`.

```
ALTER TABLE employees DISABLE INDEX IN (current_idx, other_idx);
```

## ALTER TABLE ENABLE INDEX

The **ALTER TABLE ENABLE INDEX** statement reverts the operation performed by the **ALTER TABLE DISABLE INDEX** statement.

### Syntax

```
ALTER TABLE [ IF EXISTS ] table_name ENABLE INDEX index_name_or_uuid
```

| Parameter  | Type   | Description                     |
|------------|--------|---------------------------------|
| table_name | string | The name of the table to alter. |

| Parameter          | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| index_name_or_uuid | string | The name or UUID of the index to enable. If you specify the index by name, the name must match an existing index. If you specify a UUID instead, it does not have to match an existing index. Therefore, you can disable a dropped index using its UUID, which ensures it is not used within old segments that were loaded with the index. You can get a list of index names and UUIDs in your database by referencing the sys.indexes table in the system catalog. |

### Examples

This example enables an existing index named `current_idx` on the table `employees`.

```
ALTER TABLE employees ENABLE INDEX current_idx;
```

This example enables an existing or dropped index with the UUID `5c15d8de-36fa-4055-9bdc-3f1750aabeea0`.

```
ALTER TABLE employees ENABLE INDEX '5c15d8de-36fa-4055-9bdc-3f1750aabeea0';
```

This example enables both indexes `current_idx` and `other_idx` on the table `employees`.

```
ALTER TABLE employees ENABLE INDEX IN (current_idx, other_idx);
```

### DELETE FROM TABLE

Removes rows from the specified table.

You can use the `WHERE` clause to specify the rows to remove. If a `DELETE` SQL statement lacks the `WHERE` clause, then the database deletes all rows in the table.

To use this statement, you must have the `DELETE` privilege for the table.

For details and examples, see [Remove Records from an Ocient System](#).

`DELETE` actions cannot be undone.

If a `DELETE` operation fails during execution, the database rolls back the changes and returns to its original state.

Due to limitations of the JDBC API, the reported modified row count might not be accurate for `DELETE` operations that are larger than two billion rows.

### Syntax

```
DELETE FROM table_name [ WITH cte ] [ WHERE <filter_clause> ]
```

| Parameter  | Type   | Description                                                                   |
|------------|--------|-------------------------------------------------------------------------------|
| table_name | string | The name of the table, specified as a string, indicates where to delete rows. |

| Parameter       | Type   | Description                                                                                                                                                                                                                                                                               |
|-----------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| cte             | string | A common table expression that defines temporary data for the <code>DELETE</code> statement. For details about using common table expressions, see <a href="#">General SQL Syntax</a> .                                                                                                   |
| <filter_clause> | None   | A logical combination of predicates that filter the rows to delete based on one or more columns. For details, see the <a href="#">General SQL Syntax</a> clause. The <code>DELETE SQL</code> statement removes all rows from a table if you do not include the <code>WHERE</code> clause. |

## Examples

### Delete Rows from the Table with Filter Criteria

This `DELETE SQL` statement removes all rows in the `movies` table that have a budget of less than 10000.

```
DELETE FROM movies WHERE budget < 10000;
```

### Delete Rows from the Table Using a Common Table Expression

This example uses a common table expression using the `WITH` keyword to find rows representing all transactions that occurred before 2022 that are less than \$100. The `DELETE SQL` statement receives the results from the common table expression. Then, the database executes this statement to delete the corresponding rows.

```
DELETE FROM transactions
  WITH old_transactions AS (
    SELECT
      transaction_id
    FROM
      transactions
    WHERE transaction_date < '2022-01-01'
      AND amount < 100
  )
 WHERE transaction_id IN (
   SELECT transaction_id
   FROM old_transactions );
```

## EXPORT TABLE

`EXPORT TABLE` shows the `CREATE TABLE` statement for an existing table in the current database.

To export a table, you must have the `SELECT TABLE` privilege for the table.

```
EXPORT TABLE table_name
```

| Parameter  | Type   | Description                                    |
|------------|--------|------------------------------------------------|
| table_name | string | The name of the table that you want to export. |

## Example

This example exports an existing table in the current database and schema named `trades`.

```
EXPORT TABLE trades;
```

### Output

```
CREATE TABLE "admin@system"."trade_test" (
  "id" UUID NOT NULL,
  "ticker_symbol" VARCHAR(1048576) COMPRESSION GDC(2) NOT NULL,
  "t_type" VARCHAR(1048576) COMPRESSION GDC(1) NOT NULL,
  "raw_ticker_data" VARCHAR(1048576) COMPRESSION DYNAMIC NULL,
  "created_at" TIMESTAMP TIME KEY BUCKET(1, HOUR) NOT NULL,
  "array_of_tuples" TUPLE<<VARCHAR(1048576) COMPRESSION DYNAMIC,TINYINT,BIGINT,DOUBLE PRECISION,TIMESTAMP>>
  CLUSTERING INDEX "idx_ticker_symbol_type" ("ticker_symbol", "t_type"),
  INDEX "idx_type" ("t_type")
)
REDUNDANCY cde (PARITY),
REDUNDANCY manifest (COPY),
REDUNDANCY pdf (PARITY),
REDUNDANCY stats (PARITY),
REDUNDANCY column_metadata (COPY),
REDUNDANCY index (PARITY),
REDUNDANCY summary_stats (PARITY),
REDUNDANCY skip_lists (COPY),
REDUNDANCY data (PARITY),
STORAGESPACE "storage",
SEGMENTSIZ 4,
STREAMLOADER_PROPERTIES '{"pageQueryExclusionDuration" : "30s"}';
CREATE INDEX "new_idx" ON "admin@system"."trade_test" ("raw_ticker_data") USING HASH;
```

## INSERT INTO TABLE

INSERT INTO inserts rows into a table in the current database using the results of a SQL query. The column list of the query must match the columns in the table and the query results. The first column of the query result maps to the first column of the existing table definition, the second column maps to the second column of the existing table, and so on.

The column list defaults to all columns in the table if you do not specify any column names. The database does not require every column in the table to be populated from the query. If you do not choose a column to be populated, the database inserts default values for the column. The database inserts NULL values if the column does not have a default value and the column can contain NULL. If neither a default value nor NULL can be inserted, the operation fails.

Due to limitations of the JDBC API, the reported modified row count might not be accurate for insert operations that are larger than two billion rows.

### Syntax

```
INSERT INTO table_name [ ( col1, col2 [, ...] ) ]
  [ WITH cte ] { query | VALUES [ <rows_to_insert> ] }

<rows_to_insert> ::=
  ( row1_col1, row1_col2 [, ...] ),
  ( row2_col1, row2_col2 [, ...] )
  [, ...]
```

| Parameter                  | Type   | Description                                                                                                                                                                             |
|----------------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| table_name                 | string | The name of the table for insertion.                                                                                                                                                    |
| col1, col2 [, ... ]        | string | The specific columns for insertion of new values.                                                                                                                                       |
| cte                        | string | A common table expression that defines temporary data for the <code>INSERT</code> statement. For details about using common table expressions, see <a href="#">General SQL Syntax</a> . |
| query                      | string | A <code>SELECT</code> query that defines values or a table and any of its columns that should be inserted into the specified <code>table_name</code> .                                  |
| row_col1, row_col2 [, ...] | string | The specific values to insert into columns in the table.                                                                                                                                |

## Examples

### Insert Values from One Column

This example inserts the columns from `system.table_b` into `system.table_a`.

```
INSERT INTO system.table_a SELECT * FROM system.table_b;
```

### Insert Values from Multiple Columns

This example inserts column `system.table_b.id_col_b` into `system.table_a.id_col_a` and `system.table_b.int_col_b` into `system.table_a.int_col_a`.

```
INSERT INTO system.table_a (id_col_a, int_col_a)
  SELECT id_col_b, int_col_b FROM system.table_b;
```

### Insert Literal Values

You can also insert multiple rows of literal values. The `CREATE TABLE` statement is included to demonstrate the table schema.

```
CREATE TABLE sales (
  product INT NOT NULL,
  quantity INT NOT NULL,
  sale_date DATE NOT NULL
);

INSERT INTO sales (product, quantity, sale_date)
  VALUES
    (1, 10, '2023-01-15'),
    (2, 5, '2023-01-20'),
    (1, 8, '2023-02-05');
```

### Insert Values Using a Common Table Expression

In this example, a common table expression performs calculations on the `sales` table before inserting rows into the `monthly_sales_summary` table.

The example uses the `monthly_sales_summary` table created by this `CREATE TABLE` statement with these non-nullable columns:

- `product_id` — Product identifier
- `month` — Month part of the date
- `total_quantity` — Total quantity of the product

```
CREATE TABLE monthly_sales_summary (
    "product_id" INT NOT NULL,
    "month" DATE NOT NULL,
    "total_quantity" INT NOT NULL
);
```

The common table expression following the WITH keyword extracts the month from the sale date `sale_date` and calculates the sum of the quantity sold `total_quantity` of the product from the `sales` table before inserting this data. Then, the INSERT SQL statement specifies to insert the data into the `monthly_sales_summary` table.

```
INSERT INTO monthly_sales_summary (product_id, month, total_quantity)
WITH monthly_totals AS (
    SELECT
        product,
        DATE_TRUNC('month', sale_date) AS month,
        SUM(quantity_sold) AS total_quantity
    FROM
        sales
    GROUP BY
        product,
        DATE_TRUNC('month', sale_date)
)
SELECT
    product,
    month,
    total_quantity
FROM
    monthly_totals;
```

## INSERT INTO TABLE USING LOADERS

Specify one or more Loader Nodes for executing the INSERT INTO SQL statement. If you do not use this option, the Ocient System uses all Loader Nodes that are live to execute the SQL statement.

This statement is useful for managing loading operations, particularly when balancing multiple loads of different sizes and resource requirements. Alternatively, this statement can also help simplify small batch loads by sourcing the data from a single Loader Node.

### Syntax

```
INSERT INTO TABLE table_name USING LOADERS streamloader [, ... ]
    query
```

| Parameter                 | Type   | Description                                                                                                                                                                                                                                                                                                  |
|---------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>streamloader</code> | string | A unique name for the Loader Node. Identify the names of Loader Nodes from <code>thesys.nodestable</code> by using this query: <code>SELECT name FROM sys.nodes</code> ; If the name of the streamloader contains special characters, you must enclose it in quotes, such as <code>"stream-loader1"</code> . |

For the query to execute successfully, the specified names must:

- Identify nodes that are live.
- Identify nodes that have the Loader role.

### Examples

This example inserts the column `system.table_b.id_col_b` into `system.table_a.id_col_a` and `system.table_b.int_col_b` into `system.table_a.int_col_a`. Use the Loader Node named `stream-loader1` to execute this SQL statement.

```
INSERT INTO system.table_a (id_col_a, int_col_a)
  USING LOADERS "stream-loader1"
  SELECT id_col_b, int_col_b FROM system.table_b;
```

In this example, execute the same SQL statement with two Loader Nodes named `stream-loader2` and `stream-loader3`.

```
INSERT INTO system.table_a (id_col_a, int_col_a)
  USING LOADERS "stream-loader2","stream-loader3"
  SELECT id_col_b, int_col_b FROM system.table_b;
```

### TRUNCATE TABLE

**TRUNCATE TABLE** removes some or all records from an existing table in the current database. The system deletes the truncated data, but the table and its schema remain intact in the system even if all data is deleted. If the entire table is truncated, Global Dictionary Compression tables remain in place.

To truncate a table, you must have the **DELETE** privilege for the table.

To remove a subset of rows from a table, you can use the [Database, Tables, Views, and Indexes SQL](#) statement.

For details and examples of using **TRUNCATE**, see [Remove Records from an Ocient System](#).

This action cannot be undone.

### Syntax

```
TRUNCATE TABLE table_name
```

```
TRUNCATE TABLE table_name WHERE segment_group_id = <ID>
```

```
TRUNCATE TABLE table_name WHERE segment_group_id in (<ID>, ...)
```

| Parameter  | Type   | Description                        |
|------------|--------|------------------------------------|
| table_name | string | The name of the table to truncate. |

### Examples

This example truncates an existing table in the current database and schema named `students`.

```
TRUNCATE TABLE students;
```

This example truncates an existing table in the current database named `us.students`.

```
TRUNCATE TABLE us.students;
```

This example truncates a single segment group from an existing table in the current database named `students`.

```
TRUNCATE TABLE students WHERE segment_group_id = 123456789;
```

This example truncates a number of segment groups from an existing table in the current database named `us.students`.

```
TRUNCATE TABLE us.students WHERE segment_group_id IN (1,2,3,4,5);
```

## VIEW

### CREATE VIEW

**CREATE VIEW** creates a new view in the current database or replaces an existing view. For view creation, the name of the view must be distinct from the name of any existing views in the database.

To create a view, you must have both the **CREATE VIEW** privilege for the current database and the **SELECT** privilege on all directly referenced tables and views in the query.

You must have **DROP** privileges on the view. You also must have **CREATE VIEW** privileges on the database.

#### Syntax

```
CREATE [ OR REPLACE ] VIEW [ IF NOT EXISTS ] view_name AS query
```

| Parameter | Type   | Description                                                                       |
|-----------|--------|-----------------------------------------------------------------------------------|
| view_name | string | A distinct identifier used to name the view.                                      |
| query     | string | A <b>SELECT</b> query that defines the data from a table used to create the view. |

#### Example

This example creates an existing view in the current database and schema named `option_trades`.

```
CREATE VIEW option_trades
AS SELECT * FROM trades WHERE type = 'OPTIONS';
```

### DROP VIEW

**DROP VIEW** removes one or more existing views in the current database.

To remove a view, you must have the **DROP VIEW** privilege for the view.

#### Syntax

```
DROP VIEW [ IF EXISTS ] view_name [, ...]
```

| Parameter | Type   | Description                                                                                                                          |
|-----------|--------|--------------------------------------------------------------------------------------------------------------------------------------|
| view_name | string | The identifier of the view to drop. You can drop multiple views by specifying additional view names and separating each with commas. |

#### Example

This example drops an existing view in the current database and schema named `star_employees`.

```
DROP VIEW star_employees;
```

This example drops two views named `star_employees` and `bad_employees`.



```
DROP VIEW star_employees, bad_employees;
```

## ALTER VIEW RENAME

ALTER VIEW RENAME renames an existing view.

To rename a view, you must have the ALTER VIEW privilege for the view.

### Syntax

```
ALTER VIEW [ IF EXISTS ] old_view_name RENAME TO new_view_name
```

| Parameter     | Type   | Description                           |
|---------------|--------|---------------------------------------|
| old_view_name | string | The identifier of the view to rename. |
| new_view_name | string | The new name for the specified view.  |

### Examples

This example renames an existing view in the current database and schema named `star_employees` to `star_mid_west_employees`.

```
ALTER VIEW star_employees RENAME TO star_mid_west_employees;
```

This example renames an existing view in the current database named `us.star_employees` to `us.star_mid_west_employees`.

```
ALTER VIEW us.star_employees RENAME TO us.star_mid_west_employees;
```

## ALTER VIEW AS

ALTER VIEW AS modifies an inner query of the existing view.

To modify the query for an existing view, you must be a system-level user or have the ALTER VIEW privilege for the view. You must also have the SELECT privilege on all referenced tables and views in the new query.

### Syntax

```
ALTER VIEW [ IF EXISTS ] view_name AS query
```

| Parameter | Type   | Description                                                                   |
|-----------|--------|-------------------------------------------------------------------------------|
| view_name | string | The identifier for the view to alter.                                         |
| query     | string | A SELECT query that defines the new data from a table used to alter the view. |

### Example

This example alters a view in the current database and schema named `star_employees`.

```
ALTER VIEW star_employees AS SELECT * FROM sys.tables;
```

## EXPORT VIEW

EXPORT VIEW shows the CREATE VIEW statement for an existing view in the current database.

To export a view, the logged-in user must be a system-level user or have the READ VIEW right for the view.

### Syntax

```
EXPORT VIEW view_name
```

| Parameter | Type   | Description                            |
|-----------|--------|----------------------------------------|
| view_name | string | The identifier for the view to export. |

### Example

This example exports an existing view in the current database and schema named **students**.

```
EXPORT VIEW students;
```

## INDEX

### CREATE INDEX

CREATE INDEX creates a new secondary index.

Indexes help optimize database queries when created on columns that are frequently referenced. For more information on how Ocint indexes operate, see [Secondary Indexes](#).

Creating an index does not trigger re-indexing of existing segments. Only segments generated after the CREATE INDEX is issued contain the new index.

Indexes can be created on columns containing various different data types as long as the requirements are met. Please note that depending on the data type, the system can assign different index types by default if you decline to specify which index type to use. The name must be distinct from the name of any existing index on the table.

You can apply indexes regardless of whether they have GDC compression.

### Syntax

```
CREATE INDEX [ IF NOT EXISTS ] index_name ON table (column_name) [ USING <index_type> ]
```

```
<index_type> ::=
    INVERTED | HASH | NGRAM [ (n_value) ] | SPATIAL
```

| Parameter   | Type    | Description                                                                                                                                                                       |
|-------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| index_name  | string  | An identifier for the index to create. The name must be distinct from the name of any existing index on the table.                                                                |
| table       | string  | The name of the table for the index.                                                                                                                                              |
| column_name | string  | The name of the column for the index. Identical indexes on the same column are not allowed. A column can only have multiple indexes if they are of different types or parameters. |
| n_value     | integer | Optional. When used with an NGRAM index, this numeric value specifies the character length of the substrings to be indexed. If unspecified, this value defaults to 3.             |

Index Types (<index\_type>)

Ocient supports four index types alongside the clustering index: **INVERTED**, **HASH**, **NGRAM**, and **SPATIAL**.

An index notionally stores a mapping of a column value to the rows that contain that value, and the index type differentiates the format in which the column values are stored and accessed.

Unless an index type is explicitly specified with a **USING** clause, the data type of a column determines a default index type that the system creates. For information on index type defaults, see [Secondary Indexes](#).

For container data types (e.g., arrays and tuples), the index stores the internal elements of the container, and is used on predicates that target the internal values. However, a mapping of **NULL** column values is generally stored for both scalar and container data types, so the index can always be used for **column IS NULL** predicates.

| Index Type      | Primary Data Types                                                          | Primary Usage Description                                                                                                                                                                                                                                                                                                                                              |
|-----------------|-----------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>INVERTED</b> | Fixed-length numeric columns.                                               | Stores whole column value internally, meaning its storage size is approximately the same as the width of the data type. Supports lookups using strict equality or range comparisons. For details, see the <a href="#">Understanding Secondary Indexes</a> section.                                                                                                     |
| <b>HASH</b>     | Variable-length character columns                                           | Stores a hash of the indexed column value rather than the full value. Primarily used for exact comparisons. For details, see the <a href="#">Understanding Secondary Indexes</a> section.                                                                                                                                                                              |
| <b>NGRAM</b>    | Variable-length character columns                                           | Stores substrings equal in size to its <b>n_value</b> . Storage requirements can greatly vary depending on column data size, width and cardinality. Supports exact string comparison and filters including <b>LIKE</b> , <b>NOT LIKE</b> , <b>SIMILAR TO</b> and <b>NOT SIMILAR TO</b> . For details, see the <a href="#">Understanding Secondary Indexes</a> section. |
| <b>SPATIAL</b>  | Geospatial columns<br>( <b>POINT</b> , <b>LINestring</b> , <b>POLYGON</b> ) | Groups geographic objects for bounding-box filtering. For details, see the <a href="#">Secondary Indexes</a> section.                                                                                                                                                                                                                                                  |

For further description and examples of the index types, see [Secondary Indexes](#).

### Examples

This example creates an index named **new\_idx** on the **address** column of the table. Because **address** is a **VARCHAR** column, this index defaults to the **HASH** index type.

```
CREATE INDEX new_idx ON employees (address);
```

This example creates an index of type **NGRAM** on the **address** column. As the **NGRAM** has no specified **n\_value**, it defaults to indexing substrings of three characters long.

```
CREATE INDEX ngram_address_idx ON employees (address) USING NGRAM;
```

This example creates an index on a component of the **tuple\_col** column. As this column is of data type **INT**, the index defaults to using the **INVERTED** type.

```
CREATE INDEX tuple_index ON employees (tuple_col[1]);
```

This example creates an index on a component of the `point_col` column. As this column is of data type `POINT`, the index defaults to using the `SPATIAL` type.

```
CREATE INDEX spatial_index ON employees (point_col)
```

## DROP INDEX

`DROP INDEX` drops a secondary index on a table.

After an index is dropped, new segments that are generated do not contain the new index. However, no existing segments will be altered. This means that until a segment is rebuilt, you can still use the removed index internally and the system does not reclaim the storage space the removed index occupied.

### Syntax

```
DROP INDEX [ IF EXISTS ] index_name ON table_name
```

| Parameter               | Type   | Description                                   |
|-------------------------|--------|-----------------------------------------------|
| <code>index_name</code> | string | An identifier for the index to drop.          |
| <code>table_name</code> | string | The name of the table with the index to drop. |

### Example

This example drops the index named `new_idx` on the `employees` table.

```
DROP INDEX new_idx ON employees;
```

## Related Links

[General SQL Syntax](#)

[Database Password Security Settings](#)

[System Catalog](#)

### 10.21.3 Data Pipelines

Data pipelines enable you to load data from your chosen source into the Ocient Hyperscale Data Warehouse™. You can preview a data pipeline using the `PREVIEW PIPELINE` SQL statement to test the creation statement and any required data transformation. Then, you can create a data pipeline using the `CREATE PIPELINE` statement. To manage the load, use the `START PIPELINE` statement to start the load and the `STOP PIPELINE` statement to stop it. You can rename a data pipeline using the `ALTER PIPELINE RENAME` statement. To see the full definition of a created pipeline, use the `EXPORT PIPELINE` statement. When you finish with the load, you can use the `DROP PIPELINE` statement to remove the data pipeline. You can create user-defined data pipeline functions using the `CREATE PIPELINE FUNCTION` statement and remove the function using the `DROP PIPELINE FUNCTION` statement. Also, you can administer privileges for data pipelines and data pipeline functions. For details, see [Data Control Language \(DCL\) Statement Reference](#).

## CREATE PIPELINE

`CREATE PIPELINE` defines a data pipeline that you can execute with the `START PIPELINE` SQL statement. Specify the type of load, data source, and data format.

You must have the `ALTER` privilege on the pipeline. For details, see [Data Control Language \(DCL\) Statement Reference](#).

### Syntax

```

CREATE [ OR REPLACE ] [ CONTINUOUS | BATCH ] PIPELINE [ IF NOT EXISTS ] pipeline_name
    [ <advanced_pipeline_options> ]
    [ BAD_DATA_TARGET (<kafka_bad_data_target>) ]
    SOURCE (<s3_source> | <filesystem_source> | <kafka_source>)
    EXTRACT
        (<delimited_extract_options> | <json_extract_options> |
         <parquet_extract_options> | <binary_extract_options>)
        [ <general_extract_options> ]
    INTO destination_table_name
    SELECT
        expression as col_alias_target,
        expression2 as col_alias_target2,
        ...
    [ WHERE filter_expression ]

/*****/
/* SOURCE OPTIONS */
/*****/
s3_source:
    S3
        BUCKET bucket_name
        <s3_source_paths>
        <file_based_source_options>
        <file_monitor_options>
        [ ENDPOINT s3_endpoint ]
        [ ACCESS_KEY_ID access_key_credentials ]
        [ SECRET_ACCESS_KEY secret_key_credentials ]
        [ MAX_CONCURRENCY parallel_connections ]
        [ READ_TIMEOUT num_seconds ]
        [ REGION region ]
        [ REQUEST_DEPTH num_requests ]
        [ REQUEST_RETRIES num_retries ]
        [ HEADERS headers ]

filesystem_source:
    FILESYSTEM
        <file_based_source_paths>
        <file_based_source_options>
        <file_monitor_options>

s3_source_paths:
    <file_based_source_paths> | <object_key_source_paths>

file_based_source_paths:
    (FILTER | FILTER_GLOB | FILTER_REGEX) file_filter
    [ PREFIX file_prefix ]

object_key_source_paths:
    OBJECT_KEY object_key

kafka_source:
    KAFKA
        TOPIC topic_name
        BOOTSTRAP_SERVERS bootstrap_servers
        [ WRITE_OFFSETS write_offsets ]

```

```

    [ CONFIG config_option ]
    [ AUTO_OFFSET_RESET ]

file_based_source_options:
    [ COMPRESSION_METHOD 'gzip' ]
    [ SORT_BY ('filename' | 'created' | 'modified')
      [ SORT_DIRECTION (ASC | DESC) ] ]
    [ START_TIMESTAMP start_timestamp ]
    [ END_TIMESTAMP end_timestamp ]
    [ START_FILENAME start_filename ]
    [ END_FILENAME end_filename ]

file_monitor_options:
    FILE_MONITOR
    <general_monitor_options>
    <sqs_monitor_options> | <kafka_monitor_options>

general_monitor_options:
    MONITOR_TYPE ['sqs' | 'kafka']
    [ POLLING_INTERVAL_SECOND polling_interval_second ]
    [ BATCH_MIN_FILE_COUNT batch_min_file_count ]
    [ BATCH_TIMEOUT_SECOND batch_timeout_second ]
    [ DEDUPLICATION_BOUNDARY_TARGET_HOURS deduplication_boundary_target_hours ]

sqs_monitor_options:
    SQS_QUEUE_URL sqs_queue_url
    SQS_ENDPOINT sqs_endpoint
    [ ACCESS_KEY_ID access_key_id ]
    [ SECRET_ACCESS_KEY secret_access_key ]
    [ REGION region ]

kafka_monitor_options:
    BOOTSTRAP_SERVERS bootstrap_servers
    TOPIC topic
    AUTO_OFFSET_RESET ['latest' | 'earliest']
    [ GROUP_ID group_id ]
    [ CONSUME_BATCH_SIZE consume_batch_size ]
    [ CONSUME_TIMEOUT_MS consume_timeout_ms ]
    [ FILE_INFO_SELECTORS file_info_selectors ]

/*****/
/* EXTRACT OPTIONS */
/*****/
general_extract_options:
    [ CHARSET_NAME charset_name ]
    [ COLUMN_DEFAULT_IF_NULL column_default_if_null ]

delimited_extract_options:
    FORMAT ('delimited' | 'csv')
    [ COMMENT_CHAR comment_char ]
    [ EMPTY_FIELD_AS_NULL empty_field_as_null ]
    [ ESCAPE_CHAR escape_char ]
    [ FIELD_DELIMITER field_delimiter ]
    [ FIELD_OPTIONALLY_ENCLOSED_BY enclosure_char ]
    [ NUM_HEADER_LINES num_header_lines ]

```

```

[ RECORD_DELIMITER record_delimiter ]
[ SKIP_EMPTY_LINES skip_empty_lines ]
[ OPEN_ARRAY open_array_char ]
[ CLOSE_ARRAY close_array_char ]
[ ARRAY_ELEMENT_DELIMITER array_delimiter ]
[ OPEN_OBJECT open_object_char ]
[ CLOSE_OBJECT close_object_char ]

json_extract_options:
    FORMAT 'json'

parquet_extract_options:
    FORMAT 'parquet'

binary_extract_options:
    FORMAT 'binary'
    RECORD_LENGTH length_in_bytes
    [ ENDIANNESS ('big' | 'little') ]
    [ AUTO_TRIM_PADDING auto_trim_padding ]
    [ PADDING_CHARACTER padding_character ]

/*****
/* BAD_DATA_TARGET OPTIONS */
*****/
kafka_bad_data_target:
    KAFKA
    TOPIC topic_name
    BOOTSTRAP_SERVERS bootstrap_servers
    [ CONFIG config_option ]

/*****
/* ADVANCED OPTIONS */
*****/
advanced_pipeline_options:
    [ CORES processing_cores ]
    [ PARTITIONS file_partitions ]
    [ BATCH_SIZE number_of_rows ]
    [ RECORD_NUMBER_FORMAT record_number_format ]

```

The FILE\_MONITOR option only works with an Amazon® Web Services<sup>SM</sup> (AWS<sup>SM</sup>) S3 and file system source. This option is required for continuous data pipelines that you create using the CONTINUOUS keyword. For Apache® Kafka®, you cannot use this option.

## Pipeline Identity and Naming

The name of a data pipeline is unique in an Ocient® System. Reference the pipeline in other SQL statements like START PIPELINE, STOP PIPELINE, and DROP PIPELINE using the name of the pipeline. The SQL statement throws an error if a pipeline with the same name already exists unless you specify the IF NOT EXISTS option. You can rename a pipeline with the ALTER PIPELINE SQL statement.

## Update a Pipeline

You can update a pipeline with the OR REPLACE clause in the CREATE PIPELINE SQL statement. Use this clause when you want to continue loading from the current place in a continuous load, but you need to modify transformations

or other settings. If you specify the **OR REPLACE** clause and the pipeline already exists, the database replaces the original pipeline object with the options specified in the new **CREATE OR REPLACE PIPELINE** statement. When you replace an existing pipeline, the pipeline retains its current position in the source data so that data is not duplicated when the pipeline is resumed with a **START PIPELINE SQL** statement. First, you must stop a pipeline before executing the **CREATE OR REPLACE SQL** statement.

### Batch and Continuous Data Pipeline Modes

You can define pipelines in either **BATCH** or **CONTINUOUS** mode.

- File sources (e.g., **s3**, **filesystem**) support **BATCH** and **CONTINUOUS** modes. File-based loads default to **BATCH** mode if you do not specify this keyword.
- Kafka only supports **CONTINUOUS** mode. Loads with a Kafka source default to **CONTINUOUS** mode if you do not specify this keyword.
- When you execute the **START PIPELINE SQL** statement, for **BATCH** mode, the system creates a static list of files with the **PENDING** status in the **sys.pipeline\_files** system catalog table. For **CONTINUOUS** mode, the monitor appends new incoming files to the list of files in the **sys.pipeline\_files** system catalog table.

### Required Options

You must define certain options in every pipeline. Optional options need to be present only if you use specific functions. A pipeline must contain the **SOURCE**, **EXTRACT FORMAT**, and **INTO table\_name ... SELECT** statements.

- [Data Pipelines](#)
- [Data Pipelines](#)
- [Data Formats for Data Pipelines](#)
- [Transform Data in Data Pipelines](#)

These options depend on each other: **IF NOT EXISTS** and **OR REPLACE SQL** statements are mutually exclusive.

### SELECT Statement and Data Transformation

Use the **INTO table\_name ... SELECT SQL** statement in the **CREATE PIPELINE SQL** statement to specify how to transform the data. The **SELECT** statement includes a set of one or more expressions and a target column name in the form **expression as column\_name**. The **expression** part in the statement contains a source field reference (e.g., **\$1** or **\$my\_field.subfield**) and, optionally, the transformation function you want to apply.

If your data is nested in arrays, you can use special transformation functions, such as the [Special Data Pipeline Transformation Functions](#) function, to expand the data into individual rows.

For details about data transformation and supported transformation functions, see [Transform Data in Data Pipelines](#).

For details about data types and casting, see [Data Types for Data Pipelines](#) and [Data Formats for Data Pipelines](#).

You can also specify metadata values, such as filename, to load in the **SELECT SQL** statement. For details, see [Load Metadata Values in Data Pipelines](#).

This is an example of a transformation statement snippet.

```
...
FORMAT json
INTO public.orders
SELECT
    TIMESTAMP(BIGINT($created_timestamp)) as ordertime,
    metadata('filename') as source_filename,
    $order_number as ordernumber,
    $customer.first_name as fname,
    LEFT($customer.middle_initial,1) as minitial,
    $customer.last_name as lname,
    $postal_code as postal_code,
    $promo_code as promo_code,
```



```

$order_total as ordertotal,
DECIMAL($tax,8,2) as tax,
CHAR[]($line_items[].product_name) as product_names,
CHAR[]($line_items[].sku) as skus

```

Optionally, you can filter the load by using the **WHERE** clause in the form **WHERE filter\_expression**. This expression should evaluate to a Boolean value or NULL and can include more than one filter expression. The system loads rows that contain data matching the filter criteria when the expression evaluates to **true**. The system does not load the rows when the expression evaluates to **false** or NULL. You can include any transform in the **WHERE** clause as in the **SELECT** clause.

This is an example of a filter snippet that loads order data for customers whose last name starts with the letter **a**. Use the **COALESCE** function to return only non-NULL values.

```

...
FORMAT json
INTO public.orders
SELECT
  TIMESTAMP(BIGINT($created_timestamp)) as ordertime,
  metadata('filename') as source_filename,
  $order_number as ordernumber,
  $customer.first_name as fname,
  LEFT($customer.middle_initial,1) as minitial,
  $customer.last_name as lname,
  $postal_code as postal_code,
  $promo_code as promo_code,
  $order_total as ordertotal,
  DECIMAL($tax,8,2) as tax,
  CHAR[]($line_items[].product_name) as product_names,
  CHAR[]($line_items[].sku) as skus
WHERE
  STARTSWITH(COALESCE($customer.last_name, ''), 'a')

```

### NULL and Default Value Handling

The Ocint System has specific ways to handle NULL and default values in the data pipeline. Consider this information when you prepare data to load and write the **CREATE TABLE** and **CREATE PIPELINE SQL** statements.

To load NULL values, insert the NULL value in the data. In this case, if you specify **NOT NULL** for the target column in the **CREATE TABLE** statement, the data pipeline fails to load.

If you omit a column in the **SELECT SQL** statement, the data pipeline loads the default value for the column. If you do not specify the default value, the pipeline loads a NULL value. If you specify **NOT NULL** for the target column, the data pipeline also fails to load.

You can use the **DEFAULT** keyword to load a default value. In this case, if the column does not have a default value, the pipeline fails to load. If the pipeline loads a NULL into a column with a specified default value, you can use **COALESCE(<value>, DEFAULT)** to insert the default value instead of the NULL value, where **<value>** is the NULL column. You can modify the load of one column at a time in this way.

This table describes the data pipeline behavior for NULL or omitted column values.

| Value in the Data Pipeline                 | Nullable Target Column                | Default Value in Target Column   | Resulting Data Pipeline Behavior                                                                                                                                       |
|--------------------------------------------|---------------------------------------|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NULL                                       | No                                    | Value might or might not be set. | The pipeline uses the default value if the default exists and you specify the <code>COLUMN_DEFAULT_IF_NULL</code> option. Otherwise, the pipeline fails.               |
| NULL                                       | Yes                                   | Value might or might not be set. | The pipeline uses the default value if the default exists and you specify the <code>COLUMN_DEFAULT_IF_NULL</code> option. Otherwise, the pipeline uses the NULL value. |
| Omitted column in the SELECT SQL statement | NULL value might or might not be set. | Yes                              | The pipeline uses the default value.                                                                                                                                   |
| Omitted column in the SELECT SQL statement | No                                    | No                               | The pipeline fails.                                                                                                                                                    |
| Omitted column in the SELECT SQL statement | Yes                                   | No                               | The pipeline uses the NULL value.                                                                                                                                      |

### Required Privileges

You must have the `CREATE PIPELINE` privilege on the underlying database and the `VIEW` privilege on each table in the pipeline definition to execute the `CREATE PIPELINE` SQL statement. The table must already exist.

See the `START PIPELINE` SQL statement for the required privileges to execute a pipeline.

For details, see [Data Control Language \(DCL\) Statement Reference](#).

### Limitations

A pipeline can contain, at most, 4,194,304 files. When loading more than 4,194,304 files, you must partition files into smaller batches and serially execute multiple pipelines. Use `PREFIX` and `FILTER` settings to control these partitions. It is not recommended to create multiple pipelines with a total of more than 4,194,304 files across the entire system simultaneously.

### Examples

Load JSON Data from Kafka

This example loads JSON data from Kafka using the `CREATE PIPELINE` SQL statement. Use the bootstrap server `192.168.0.1:9092` using the Kafka topic `orders`. Load data into the `public.orders` table. Specify the data to load as these JSON selectors:

- Identifier `$id`
- User identifier `$user_id`
- Product identifier `$product_id`
- Subtotal amount `$subtotal`
- Tax `$tax`
- Total amount `$total`
- Discount amount `$discount`
- Created time `$created_at`
- Quantity `$quantity`

```

CREATE PIPELINE orders_pipeline
  SOURCE KAFKA
    BOOTSTRAP_SERVERS '192.168.0.1:9092'
    TOPIC 'orders'
  EXTRACT
    FORMAT json
INTO public.orders
SELECT
  $id as id,
  $user_id as user_id,
  $product_id as product_id,
  $subtotal as subtotal,
  $tax as tax,
  $total as total,
  $discount as discount,
  $created_at as created_at,
  $quantity as quantity;

```

For a complete tutorial, see [Data Pipeline Load of JSON Data from Kafka](#).

#### Load Delimited Data from S3

This example loads delimited data in CSV format from S3. Use the <https://s3.us-east-1.amazonaws.com> endpoint with the `ocient-examples` bucket and path `metabase_samples/csv/orders.csv`. Specify one header line with the `NUM_HEADER_LINES` option. Load data into the `public.orders` table. Specify the data to load using the column numbers:

- Identifier
- User identifier
- Product identifier
- Subtotal amount
- Tax
- Total amount
- Discount amount
- Created time
- Quantity

```

CREATE PIPELINE orders_pipeline
  SOURCE S3
    ENDPOINT 'https://s3.us-east-1.amazonaws.com'
    BUCKET 'ocient-examples'
    FILTER 'metabase_samples/csv/orders.csv'
  EXTRACT
    FORMAT csv
    NUM_HEADER_LINES 1
INTO public.orders
SELECT
  $1 as id,
  $2 as user_id,
  $3 as product_id,
  $4 as subtotal,
  $5 as tax,
  $6 as total,
  $7 as discount,
  $8 as created_at,
  $9 as quantity;

```

For a complete tutorial, see [Data Pipeline Load of CSV Data from S3](#).

#### Continuous File Load of Delimited Data in CSV Files

Create a data pipeline that uses continuous file loading with delimited data in CSV files. Specify to use 32 partitions and 16 cores with the `PARTITIONS` and `CORES` options, respectively. The source is S3 with the `http://endpoint.ocient.com` endpoint and `cs_data` bucket. Specify the filter `'**.csv'` to find all files with a filename that matches a glob pattern without subdirectories, for example, `data.csv`. The system filters filenames with subdirectories such as `data/data_sample.csv`.

For continuous file loading, specify the `FILE_MONITOR` option. Use the Kafka monitor by specifying the `MONITOR_TYPE` option. Use 15 seconds as the timeout for waiting before creating another loading job with pending files with the `BATCH_TIMEOUT_SECOND` option. Using the `BATCH_MIN_FILE_COUNT` option, specify a minimum of 10 pending files to start a new load. Then, specify 10 hours for looking back for file deduplication with the `DEDUPLICATION_BOUNDARY_TARGET_HOURS` option.

Specify the `test-broker:9092` bootstrap server, `cfl_kafka_ten_adtech_flat_small` Kafka topic, and reset the offset to the smallest offset using the `earliest` value of the `AUTO_OFFSET_RESET` option. Set the client group identifier to `84079bf1-bdc4-4b10-ba12-41ba6b17dffe`. Allow a maximum of 10 messages to poll each time with a timeout of 1,000 milliseconds using the `CONSUME_BATCH_SIZE` and `CONSUME_TIMEOUT_MS` options, respectively.

The format is `csv` with the record delimiter as the newline character `\n`. There are 39 fields in the CSV data. Load data in the `public.ad_sessions` table.

The `SELECT` statement identifies the columns to load by number. For each column, transform each column using cast functions. For details, see [Scalar Data Conversion Functions](#) for each function. See [Date and Time Functions](#) for the `TO_TIMESTAMP` function.

```
CREATE CONTINUOUS PIPELINE test_continuous_pipeline
PARTITIONS 32
CORES 16
SOURCE S3
  ENDPOINT 'http://endpoint.ocient.com/'
  BUCKET 'cs_data'
  FILTER '**.csv'
  FILE_MONITOR(
    MONITOR_TYPE kafka
    BATCH_TIMEOUT_SECOND 15
    BATCH_MIN_FILE_COUNT 10
    DEDUPLICATION_BOUNDARY_TARGET_HOURS 10
    BOOTSTRAP_SERVERS 'test-broker:9092'
    TOPIC 'cfl_kafka_ten_adtech_flat_small'
    AUTO_OFFSET_RESET 'earliest'
    GROUP_ID '84079bf1-bdc4-4b10-ba12-41ba6b17dffe'
    CONSUME_BATCH_SIZE 10
    CONSUME_TIMEOUT_MS 1000 )
EXTRACT
  FORMAT csv
  RECORD_DELIMITER e'\n'
  NUM_FIELDS 39
AS source_field
INTO public.ad_sessions
SELECT
  TO_TIMESTAMP(CHAR($1), 'yyyy-MM-dd HH:mm:ss.SSSSSS', 'java') AS event_date_time,
  CHAR(NULL_IF($2, '')) AS device_model,
  TINYINT($8) AS device_user_age,
  BOOLEAN($10) AS device_ad_tracking_disabled,
  BINARY(NULL_IF($11, '')) AS device_mac,
  INT($14) AS ip_zip,
```

```

    FLOAT($19)           AS ip_zip_latitude,
    DOUBLE($20)          AS ip_zip_longitude,
    BIGINT($21)          AS session_id,
    SMALLINT($32)        AS session_response_latency,
    DECIMAL($34, 10, 1)  AS session_transaction_revenue,
    CHAR($39)            AS session_app_name
FROM source_field;
```

SOURCE Options

File-Based Source Options

Options that apply to both the S3 and FILESYSTEM sources.

| Option Key | Default     | Required or<br>Optional | Data Type | Description |
|------------|-------------|-------------------------|-----------|-------------|
| FILTER     | FILTER_GLOB | FILTER_REGEX            | None      | Required    |

| Option Key | Default | Required or Optional | Data Type                | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------|---------|----------------------|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PREFIX     | None    | Optional             | STRINGorARRAY OF STRINGS | <p>Specify a prefix in addition to a filter. The prefix must end with a forward slash. The Ocient System uses the more specific option between the specified prefix and filter.Paths include a leading forward slash.The system applies the prefix to S3 sources to restrict the scope of listing operations and improve performance. Use this option along with FILTER_REGEX to enforce the listing prefix of your choice.When you apply a list of prefixes, the system loads the union of the set of files found in each of the paths specified by the prefix.≤This option is not available for continuous data pipelines.<b>Examples:</b>List all CSV files in the2024subdirectory offiles.PREFIX<br/> '/files/2024/'<br/> FILTER<br/> '**/*.csv'List all JSON files of subdirectory2024/09/in theordersdirectory.PREFIX<br/> '/data/orders/2024/09/'<br/> FILTER_REGEX<br/> './orders/*.json'List all JSON files in multiple subdirectories.PREFIX<br/> ['/data/orders/2024/09/',<br/> '/data/orders/2024/10/']</p> |

| Option Key | Default | Required or Optional | Data Type                | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------|---------|----------------------|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OBJECT_KEY | None    | Optional             | STRINGorARRAY OF STRINGS | Specify an S3 object key to load. You cannot use this option if the PREFIX option or any FILTER option is set.≤ Object keys do not include a leading forward slash.The pipeline loads the object key or keys listed in the pipeline. This way can be faster for loading data by avoiding listing of files in large directories.Object keys must not include any of the special characters:*?\{\} [] ≤This option is not available for continuous data pipelines. <b>Examples:</b> Load a single object from the designated bucket at the specified object key.OBJECT_KEY 'order_data/jsonl/orders_202 a list of objects from the designated bucket at the specified object keys.OBJECT_KEY ['order_data/jsonl/orders_20 'order_data/jsonl/orders_202 |

| Option Key         | Default  | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------|----------|----------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| COMPRESSION_METHOD | NONE     | Optional             | STRING    | The method the Ocient System uses to decompress file data. The supported option is <code>gzip</code> . To load uncompressed data, omit the <code>COMPRESSION_METHOD</code> option. ≤ The <code>COMPRESSION_METHOD</code> option is only applicable to file-based sources. Kafka pipelines automatically decompress data based on the Kafka topic configuration.                                                                                |
| SORT_BY            | filename | Optional             | STRING    | The sort criteria for sorting the file list before the load. Supported options are: <code>filename</code> — Sort files lexicographically by the <code>filename</code> . <code>created</code> — Sort files based on the timestamp that indicates the creation of the file. <code>modified</code> — Sort files based on the timestamp that indicates the modification of the file. ≤ This option is not available for continuous data pipelines. |
| SORT_DIRECTION     | ASC      | Optional             | STRING    | The sort direction, either <code>ASC</code> ascending or <code>DESC</code> descending, determines the sort order. If you specify this option, you must also specify the <code>SORT_BY</code> option. ≤ This option is not available for continuous data pipelines.                                                                                                                                                                             |



| Option Key      | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----------------|---------|----------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| START_TIMESTAMP | None    | Optional             | TIMESTAMP | <p>The ISO-8601-compliant date or date time that is used as the lower bound to filter files for batch pipelines. The time zone should match the file metadata. (inclusive)When you use this option with the SORT_BY option with the value set to filename or modified, then the system uses the modification timestamp for the file to compare the value. When you use this option with the SORT_BY option and the created value, then the system uses the creation timestamp for the file to compare the value. You can use this option without the END_TIMESTAMP option. If you specify the END_TIMESTAMP option, you must specify the START_TIMESTAMP option before the END_TIMESTAMP option. ≤ This option is not available for continuous data pipelines.</p> |

| Option Key    | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------|---------|----------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| END_TIMESTAMP | None    | Optional             | TIMESTAMP | <p>The ISO-8601-compliant date or date time that is used as the upper bound to filter files for batch pipelines. The time zone should match the file metadata. (inclusive)When you use this option with the SORT_BY option with the value set to filename or modified, then the system uses the modification timestamp for the file to compare the value. When you use this option with the SORT_BY option and the created value, then the system uses the creation timestamp for the file to compare the value. You can use this option without the START_TIMESTAMP option. If you specify the END_TIMESTAMP option, you must specify the START_TIMESTAMP option before the END_TIMESTAMP option. ≤ This option is not available for continuous data pipelines.</p> |

| Option Key     | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------|---------|----------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| START_FILENAME | None    | Optional             | STRING    | <p>The filename string that is the lower bound to filter for files lexicographically for batch pipelines. (inclusive) Use the full path of the file, such as <code>'/dir/load_file.json'</code>. If the file is located in the top-most directory, start the path with a slash/, such as <code>'/load_file.json'</code>. You can use this option without the <code>END_FILENAME</code> option. If you specify the <code>END_FILENAME</code> option, the value of the <code>START_FILENAME</code> option must be smaller than the value for the <code>END_FILENAME</code> option lexicographically.</p> |

| Option Key   | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------|---------|----------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| END_FILENAME | None    | Optional             | STRING    | The filename string that is the upper bound to filter for files lexicographically for batch pipelines. (inclusive) Use the full path of the file, such as <code>'/dir/load_file.json'</code> . If the file is located in the top-most directory, start the path with a slash/, such as <code>'/load_file.json'</code> . You can use this option without the START_FILENAME option. If you specify the END_FILENAME option, the value of the START_FILENAME option must be smaller than the value for the END_FILENAME option lexicographically. |

### S3 Source Options

You can apply these options to data sources of the `SOURCE S3` type, which include S3 and S3-compatible services.

| Option Key | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                 |
|------------|---------|----------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BUCKET     | None    | Required             | STRING    | The name of the bucket in AWS S3.                                                                                                                                                                           |
| ENDPOINT   | None    | Optional             | STRING    | The endpoint URI for the S3-compatible service API. (e.g., <code>https://s3.us-east-2.amazonaws.com</code> ). If you provide the ENDPOINT option, the Ocient System ignores settings for the REGION option. |

| Option Key        | Default               | Required or Optional | Data Type | Description                                                                                                                                                                                                                                   |
|-------------------|-----------------------|----------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| REGION            | 'us-east-1'           | Optional             | STRING    | The region that the Ocient System uses for AWS access. If you specify the <code>ENDPOINT</code> option, the system ignores this option.                                                                                                       |
| ACCESS_KEY_ID     |                       | Optional             | STRING    | The access key identification for AWS credentials. If you specify this option, you must also specify the <code>SECRET_ACCESS_KEY</code> option. The Ocient System uses anonymous credentials when you specify an empty value for this option. |
| SECRET_ACCESS_KEY |                       | Optional             | STRING    | The secret key for AWS credentials. If you specify this option, you must also specify the <code>ACCESS_KEY_ID</code> option. The Ocient System uses anonymous credentials when you specify an empty value for this option.                    |
| MAX_CONCURRENCY   | 30                    | Optional             | INTEGER   | Determines the number of parallel connections the Ocient System uses to communicate with the AWS S3 service. This option does not require modification in most cases. Contact Ocient Support to modify these values.                          |
| READ_TIMEOUT      | 0 (unlimited timeout) | Optional             | INTEGER   | The number of seconds until a read operation times out. This option does not require modification in most cases. Contact Ocient Support to modify these values.                                                                               |

| Option Key      | Default | Required or Optional | Data Type | Description                                                                                                                                                                                            |
|-----------------|---------|----------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| REQUEST_DEPTH   | 500     | Optional             | INTEGER   | The upper boundary of requests that the Ocient System handles concurrently. This option does not require modification in most cases. Contact Ocient Support to modify these values.                    |
| REQUEST_RETRIES | 10      | Optional             | INTEGER   | Number of times the AWS SDK retries failing requests before the Ocient System throws an error. This option does not require modification in most cases. Contact Ocient Support to modify these values. |

| Option Key | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------|---------|----------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HEADERS    | None    | Optional             | STRING    | The headers to send with every request. This option is a JSON-formatted string. Represent the chosen header names as keys with corresponding values as scalars or lists of scalars. The system converts scalars that are not strings to strings. During the load, the system maps each element in a list to the header name represented by the corresponding key. Examples: HEADERS '{"x-amz-request-payer": "requester"}' returns this header for each request: x-amz-request-payer: requester. HEADERS '{"header-name": ["list", "of", "values"]}' returns this header for each request: header-name: list, of, values. |

### FILESYSTEM Source Options

No options exist specific to the file system source (`SOURCE FILESYSTEM`) except for general [Data Pipelines](#).

When you load data using `SOURCE FILESYSTEM`, the files must be addressable from all of your Loader Nodes. The Ocint System uses the specified path in the pipeline `PREFIX` and `FILTER` options to select the files to load.

A shared view of the files you want to load must be available to all Loader Nodes involved in a pipeline. For example, you can use a Network File System (NFS) mount available to all Loader Nodes at a common path on each node.

### Example

This example `CREATE PIPELINE` SQL statement snippet contains a `FILESYSTEM` source and filters to all CSV files in the `/tmp/sample-data/` directory on each of the Loader Nodes.

```
CREATE PIPELINE ...
  SOURCE FILESYSTEM
    PREFIX '/tmp/sample-data/'
```

```

    FILTER '**/*.csv'
  EXTRACT
    FORMAT delimited
  INTO public.orders
  SELECT
    $1 as username,
    $2 as subtotal,
    ...

```

### KAFKA Source Options

You can apply these options to Kafka data sources (`SOURCE KAFKA`).

For compression, you do not need to specify a compression option in Kafka-based pipelines, because the Ocient System handles the compression type automatically. Records produced to the Kafka broker with a `compression.type` setting or with the `compression.type` set on the topic automatically decompress when the loading process consumes the records. The loading process uses built-in headers in Kafka to determine the required decompression during extraction.

| Option Key        | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                               |
|-------------------|---------|----------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TOPIC             | None    | Required             | STRING    | The name of the Kafka topic indicates where to consume records.                                                                                                                                                                                                           |
| BOOTSTRAP_SERVERS | None    | Required             | STRING    | A comma-delimited list of IP:port pairs that contain the IP addresses and the associated port numbers of the Kafka Brokers. You can also use a hostname instead of the IP address. <b>Example:</b> <code>BOOTSTRAP_SERVERS = '198.51.100.1:9092,198.51.100.1:9092'</code> |
| WRITE_OFFSETS     | true    | Optional             | BOOLEAN   | Indicates whether the Kafka consumer should write its durably-made record offsets to the Kafka Broker.                                                                                                                                                                    |
| CONFIG            | {       |                      |           |                                                                                                                                                                                                                                                                           |

```

  "enable.auto.commit": false,
  "key.deserializer": "org.apache.kafka.common.serialization.ByteArrayDeserializer",
  "value.deserializer": "org.apache.kafka.common.serialization.ByteArrayDeserializer",
  "group.id": -} | Optional | STRING | The consumer configuration that the Kafka consumers should use. This

```

`AUTO_OFFSET_RESET` | None | Optional | STRING | Determine which action to take for the Kafka configuration when there is no initial offset in the offset store or the specified offset is out of range. Supported values are: `'smallest'` or `'earliest'`— Automatically reset the offset value to the smallest value. `'largest'` or `'latest'`— Automatically reset the offset value to the largest value. `'error'`— Throw an error. |



For the consumer configuration, to create a secure connection from the Kafka consumer to a Kafka Broker, set the "security.protocol" key along with any SSL or SASL keys.

If a certificate file is required, you must add it to the truststore used by the JVM® on all Loader Nodes. The truststore path must be identical on all Loader Nodes. The Kafka configuration can reference this truststore path.

### Kafka CONFIG Option Defaults

When you configure the CONFIG option for a Kafka source, the Ocient System merges the values you specify with default values that Ocient uses when the system creates a Kafka Consumer.

You cannot override some values that Ocient provides using the CONFIG option.

| Key                | Default Value                                              | Override Allowed |
|--------------------|------------------------------------------------------------|------------------|
| group.id           | <database name>-<pipeline name>                            | Yes              |
| enable.auto.commit | false                                                      | No               |
| key.deserializer   | "org.apache.kafka.common.serialization.StringDeserializer" | No               |
| value.deserializer | "org.apache.kafka.common.serialization.StringDeserializer" | No               |

### Continuous File Loading Source Options

Specify these options for data pipelines that use S3 and file system sources with the CONTINUOUS mode. For a Kafka source, do not use these options.

### General File Monitor Options

The FILE\_MONITOR option is required for data pipelines with the CONTINUOUS mode and S3 and file system sources.

| Option Key       | Default   | Required or Optional | Data Type | Description                                                                                                                                                                                               |
|------------------|-----------|----------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MONITOR_TYPE     | None      | Required             | STRING    | The type of monitor. If the specified string is not one of the defined monitors, compilation fails. Valid values are: <code>s3</code> , <code>kafka</code> .                                              |
| POLLING_INTERVAL | 10SECOND  | Optional             | INT       | Number of seconds to consume the event topic. Valid value range: 10 to 120.                                                                                                                               |
| BATCH_TIMEOUT    | 300SECOND | Optional             | INT       | When the number of pending files is smaller than <code>BATCH_MIN_FILE_COUNT</code> , the number of seconds to wait before creating another loading job with pending files. Valid value range: 10 to 1800. |

| Option Key                          | Default | Required or Optional | Data Type | Description                                                                             |
|-------------------------------------|---------|----------------------|-----------|-----------------------------------------------------------------------------------------|
| BATCH_MIN_FILE_COUNT                | 1000    | Optional             | INT       | The minimum number of pending files to start a new load. Valid value range: 10 to 2000. |
| DEDUPLICATION_BOUNDARY_TARGET_HOURS | 24      | Optional             | INT       | The number of hours to look back for file deduplication. Valid value range: 0 to 48.    |

### SQS Monitor Options

Use these options when you set the MONITOR\_TYPE option to **sqs** for Amazon® SQS.

| Option Key        | Default   | Required or Optional | Data Type | Description                                                                                                                                                                                                      |
|-------------------|-----------|----------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SQS_ENDPOINT      | None      | Required             | STRING    | The endpoint URL for the client. For example: <code>http://localhost:32769</code>                                                                                                                                |
| SQS_QUEUE_URL     | None      | Required             | STRING    | The URL of the target queue. For example: <code>http://localhost:32769/00000000000000000000000000000000</code>                                                                                                   |
| ACCESS_KEY_ID     | <empty>   | Optional             | STRING    | Access key identifier for SQS authentication. <b>ACCESS_KEY_ID</b> and <b>SECRET_ACCESS_KEY</b> must be provided as a pair. If you do not provide a secret, continuous file loading reverts to anonymous access. |
| SECRET_ACCESS_KEY | <empty>   | Optional             | STRING    | The secret access key for SQS authentication.                                                                                                                                                                    |
| REGION            | us-east-1 | Optional             | STRING    | The region for SQS operation.                                                                                                                                                                                    |

### Kafka Monitor Options

Use these options when you set the MONITOR\_TYPE option to **kafka** for Kafka.

| Option Key         | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------|---------|----------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BOOTSTRAP_SERVERS  | None    | Required             | STRING    | The bootstrap servers for the Kafka consumer configuration. The string contains a list of brokers as a comma-separated list of the broker hostnames or broker hostnames and port number combinations, in the format <code>hostname:port</code> .                                                                                                                                                                                                                                                                 |
| TOPIC              | None    | Required             | STRING    | The name of the Kafka topic indicates where to consume records.                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| AUTO_OFFSET_RESET  | None    | Required             | STRING    | Action to take for the Kafka configuration when there is no initial offset in the offset store or the chosen offset is out of range: <code>'smallest'</code> , <code>'earliest'</code> —Automatically reset the offset to the smallest offset. <code>'largest'</code> , <code>'latest'</code> —Automatically reset the offset to the largest offset. <code>'error'</code> —Trigger an error ( <code>ERR_AUTO_OFFSET_RESET</code> ) retrieved by consuming messages and checking <code>'message-&gt;err'</code> . |
| GROUP_ID           | CFL     | Optional             | STRING    | Client group identifier for the Kafka configuration. All clients share the same group. The identifier belongs to the same group and therefore shares the same committed offset.                                                                                                                                                                                                                                                                                                                                  |
| CONSUME_BATCH_SIZE | None    | Optional             | INT       | The maximum number of messages to poll each time.                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

| Option Key         | Default                                                                                                                                                          | Required or Optional | Data Type     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CONSUME_TIMEOUT    | 1000MS                                                                                                                                                           | Optional             | INT           | The operation timeout, in milliseconds, that controls how long the consume request waits for the response.                                                                                                                                                                                                                                                                                                                                                                                                                          |
| FILE_INFO_SELECTOR | <code>["Records"][1].s3.object.key, "Records"[1].eventTime, "Records"[1].s3.object.size]</code>                                                                  | Optional             | ARRAY<STRING> | Defines three JSON selectors for the filename, the last modification timestamp of the file, and the file size of an incoming message in a monitor. The selector must follow this order. The default value works for <code>s3:ObjectCreated:Put</code> event. If you override this default configuration, you can parse custom JSON messages. The format of the individual fields must still adhere to S3 standards. For details, see Amazon Simple Storage Service Event message structure. The message must be in the JSON format. |
| CONFIG             | <code>{ "enable.auto.commit": false, "group.id": "CFL", "max.poll.interval.ms": "600000", "heartbeat.interval.ms": "6000", "session.timeout.ms": "600000"</code> |                      |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

| Option Key | Default  | Required or Optional | Data Type                                                                                                                                                                                                                                              | Description |
|------------|----------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| }          | Optional | STRING               | The consumer configuration that the Kafka consumers should use. This option is a JSON-formatted string. Certain values within this configuration are fixed, whereas the Ocident System provides other values with a default value that you can modify. |             |

The same Kafka CONFIG option override considerations apply. For details, see [Data Pipelines](#).

## EXTRACT Options

### General Extract Options

You can specify these options on any of the allowed **FORMAT** types.

| Option Key | Default | Required or Optional | Data Type | Description                                                                                |
|------------|---------|----------------------|-----------|--------------------------------------------------------------------------------------------|
| FORMAT     | None    | Required             | STRING    | Specifies the format of the files to load. Supported values are: delimited, csv, json, bin |

| Option Key             | Default                                            | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------------------------|----------------------------------------------------|----------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CHARSET_NAME           | Binary<br>format:ibm1047All<br>other formats:utf-8 | Optional             | STRING    | Specifies the character set for decoding data from the source records into character data. Use this character set when you load data into VARCHAR columns or when you apply CHAR transformation functions. Defaults to utf-8 for all formats except BINARY, which defaults to ibm1047. You can configure the default value for BINARY formatted data using a SQL statement such as: ALTER SYSTEM ALTER CONFIG SET 'sql.pipelineParameters.extra = 'ibm1047' |
| COLUMN_DEFAULT_is_null | false NULL                                         | Optional             | BOOLEAN   | Specifies whether pipelines should load the column default value when the result of a series of transforms is NULL. If you set this option to false (the default), the pipeline loads NULL values into columns. If you set this option to true, the pipeline loads the defined default value of the column when the result of the execution of transformation on the column is NULL.                                                                        |

| Option Key   | Default                                                           | Required or Optional | Data Type           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------|-------------------------------------------------------------------|----------------------|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NULL_STRINGS | Delimited or CSV<br>format: ['null',<br>'NULL']JSON<br>format: [] | Optional             | ARRAY OF<br>STRINGS | Specifies string values that should represent a NULL value when extracted from the source records. Use this <code>incsv, delimited</code> , and <code>json</code> formats to convert specific values to NULL instead of requiring individual transform function calls to <code>NULL_IF</code> with those values. This option applies to source data immediately after extraction. If the result of transformations is one of these strings, you must use <code>NULL_IF</code> to transform to NULL with the specified string values. |

### Delimited and CSV Extract Options

You can specify these options for `FORMAT delimited` or `FORMAT csv` data formats, which are aliases.

For details on working with Delimited data, see [Data Formats for Data Pipelines](#).

| Option Key   | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                   |
|--------------|---------|----------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| COMMENT_CHAR | NULL    | Optional             | STRING    | Specifies the character used to comment out a record in the source file. The load skips records where the first character of a record is equal to this character. Set this option to NULL or '' to turn off the detection of these control characters. Example: <code>COMMENT_CHAR '#'</code> |

| Option Key          | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------|---------|----------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EMPTY_FIELD_AS_NULL | NULL    | Optional             | BOOLEAN   | Specifies whether the Ocient System should extract an empty source field as NULL or a missing value. When this option is set to <b>true</b> , the Ocient System treats empty fields as NULL. Otherwise, the system treats fields as a missing value. For string-type fields, a missing value is equivalent to an empty string. Define an empty field as two consecutive delimiters (e.g., the second field is empty in <code>abc,,xyz</code> ). If a field is explicitly an empty string as indicated by quote characters, the Ocient System treats the field as an empty string, not an empty field (e.g., the second field is an empty string in <code>abc,"",xyz</code> ). The <code>CHAR(\$1)</code> transformation function handles both NULLs and empty strings and passes them through. Beware that the <code>NULL_IF(\$1, '')</code> transformation function directly loads a NULL for both NULLs <i>and</i> missing values. This transformation can override the behavior of <code>EMPTY_FIELD_AS_NULL</code> . |



| Option Key                     | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------------------|---------|----------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FIELD_OPTIONALLY" _ENCLOSED_BY |         | Optional             | STRING    | Also known as the “quote character,” this option specifies the character for optionally enclosing fields. Fields enclosed by this character can include delimiters, the enclosure character, or the escape character.Set this option to NULL or ' 'to turn off the detection of these control characters. <b>Examples:</b> Use a double quote character:<br>FIELD_OPTIONALLY_ENCLOSED_BY = " "Use a single quote character:<br>FIELD_OPTIONALLY_ENCLOSED_BY = ' ' |
| ESCAPE_CHAR                    | "       | Optional             | STRING    | Specifies the escape character within fields enclosed by theFIELD_OPTIONALLY_ENCLOSED_BY character. Use this option to escape the enclosure character or escape character.Set this option to NULL or ' 'to turn off the detection of these control characters. <b>Examples:</b> Use a double quote as the escape character.<br>ESCAPE_CHAR = " "Use a single quote as the escape character.<br>ESCAPE_CHAR = ' '                                                  |

| Option Key       | Default | Required or Optional | Data Type                | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------|---------|----------------------|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FIELD_DELIMITER  | ','     | Optional             | STRINGorARRAY OF STRINGS | Specifies a character or list of possible characters for delimiting fields. The default value sets the field delimiter to only a comma. The value must be one byte.The Ocient System automatically interprets the values you specify as C-style escaped strings. This means you do not need to specify an escape string ( <b>e'some value'</b> ) is not required to specify control characters. This differs from the default string behavior in Ocient. For details, seeGeneral SQL Syntax.FIELD_DELIMITERand FIELD_DELIMITERS are aliases. <b>Examples:</b> Use a tab character:FIELD_DELIMITER = '\t'Use a pipe character:FIELD_DELIMITER = ' 'Use either a pipe or a comma character:FIELD_DELIMITER = [' ',' '] |
| NUM_HEADER_LINES | 0       | Optional             | INTEGER                  | Specifies the number of header lines, typically 0 or 1. The Ocient System skips this number of lines and does not load them as data when files are processed. Use this option when your data includes a row of header values.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

| Option Key       | Default        | Required or Optional | Data Type                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------|----------------|----------------------|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RECORD_DELIMITER | ['\r\n', '\n'] | Optional             | STRING or ARRAY OF STRINGS | Specifies the string or an array of strings for delimiting records. The file is split into individual records using this character during processing. Common values include '\r\n' and '\n'. The value must be one or two bytes. The Ociant System automatically interprets the values you specify as C-style escaped strings. An escape string (e'some value') is not required to specify control characters. This differs from the default strings behavior in Ociant. The system chooses the first specified delimiter and uses that delimiter for the rest of the data. Data with mixed delimiters is not supported. For details, see General SQL Syntax. RECORD_DELIMITER and RECORD_DELIMITERS are aliases. <b>Examples:</b> Use a linefeed character: RECORD_DELIMITER = '\n' Use a carriage return and linefeed character sequence: RECORD_DELIMITER = '\r\n' |
| SKIP_EMPTY_LINES | false          | Optional             | BOOLEAN                    | Specifies whether or not to skip empty lines.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

| Option Key  | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------|---------|----------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CLOSE_ARRAY | ' ] '   | Optional             | STRING    | <p>Specifies the character that indicates the end of an array in <code>acsvordelimitedfield</code>. Use this option to parse array data types. Specify the <code>OPEN_ARRAY</code> option also when using this option. Set this option to <code>NULL</code> or <code>' '</code> to turn off the detection of these control characters. If you set this option to either of these characters, the system also turns off the detection of these characters for the <code>OPEN_ARRAY</code> option. <b>Example:</b> C source data such as <code>val1, "{1,2,3}", val2</code> to an array when referenced as <code>\$2[ ].CLOSE_ARRAY</code></p> <p><code>' }'      OPEN_ARRAY</code></p> <p><code>' {'</code></p> <p><code>ARRAY_ELEMENT_DELIMITER</code></p> <p><code>' , '</code></p> |

| Option Key   | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------|---------|----------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OPEN_ARRAY   | '['     | Optional             | STRING    | Specifies the character that indicates the start of an array in <code>acsvordelimitedfield</code> . Use this option to parse array data types. Specify the <code>CLOSE_ARRAY</code> option also when using this option. Set this option to NULL or '' to turn off the detection of these control characters. If you set this option to either of these characters, the system also turns off the detection of these characters for the <code>CLOSE_ARRAY</code> option. <b>Example:</b><br>source data such as<br><code>val1,"[1,2,3]",val2</code> to an array when referenced as<br><code>\$2[].OPEN_ARRAY</code><br><code>'[' CLOSE_ARRAY</code><br><code>']'</code><br><code>ARRAY_ELEMENT_DELIMITER</code><br><code>','</code> |
| CLOSE_OBJECT | '}'     | Optional             | STRING    | Specifies the character that indicates the end of a tuple in a field. Use this option to parse tuple data types. Specify the <code>OPEN_OBJECT</code> option also when using this option. Set this option to NULL or '' to turn off the detection of these control characters. If you set this option to either of these characters, the system also turns off the detection of these characters for the <code>OPEN_OBJECT</code> option.                                                                                                                                                                                                                                                                                          |

| Option Key              | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------------------|---------|----------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OPEN_OBJECT             | '{'     | Optional             | STRING    | Specifies the character that indicates the start of a tuple in a field. Use this option to parse tuple data types. Specify the CLOSE_OBJECT option also when using this option. Set this option to NULL or '' to turn off the detection of these control characters. If you set this option to either of these characters, the system also turns off the detection of these characters for the CLOSE_OBJECT option. |
| ARRAY_ELEMENT_DELIMITER |         | Optional             | STRING    | Specifies the character that separates values in an array within a CSV delimited field. Use this option to parse array data types. Set this option to NULL or '' to turn off the detection of these control characters. <b>Example:</b> Convert source data such as <code>val1, "[1;2;3]", val2</code> to an array when referenced as <code>\$2[ ].ARRAY_ELEMENT_DELIMITER = ';' ;</code>                           |

When you specify the escape character, you often have to use an escape sequence. This action follows standard SQL rules.

#### JSON Extract Options

No options exist for JSON data record extraction (**FORMAT json**).

For details about JSON-formatted data, see [Data Formats for Data Pipelines](#).

#### PARQUET Extract Options

No options exist for Apache® Parquet™ data record extraction (**FORMAT parquet**).

For details about Parquet-formatted data, see [Data Formats for Data Pipelines](#).

When you use the **FORMAT PARQUET** option with an AWS S3 Source, the **ENDPOINT** option is required.

#### BINARY Extract Options

You can apply these options to binary data record extraction (**FORMAT BINARY**).

For details about BINARY-formatted data, see [Data Formats for Data Pipelines](#).

The general option **CHARSET\_NAME** has a different default value for **FORMAT BINARY**.

| Option Key        | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                             |
|-------------------|---------|----------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RECORD_LENGTH     | None    | Required             | INTEGER   | Specifies the fixed size in bytes of each record in the source data. The Ociant System splits the binary data into binary chunks according to this length value and processes them individually.                                                                                        |
| ENDIANNESS        | 'big'   | Optional             | STRING    | Specifies the endianness used to interpret multi-byte sequences in various transforms of binary data. Accepted values are 'big' and 'little'.                                                                                                                                           |
| AUTO_TRIM_PADDING | TRUE    | Optional             | BOOL      | Determines if padding characters should be trimmed after decoding the binary data into string data. If you set this option to TRUE, the Ociant System trims all instances of the PADDING_CHARACTER value from the end of a string after the system decodes the string from BINARY type. |

| Option Key        | Default                                                                                                                                                                                                                                                                                                                                                                                              | Required or Optional | Data Type | Description |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|-----------|-------------|
| PADDING_CHARACTER | (space)  <br>Optional   STRING<br>  The padding<br>character from<br>the string after<br>the system<br>decodes the<br>string<br>from BINARY type. You<br>can change the<br>default padding<br>character<br>for BINARY formatted<br>data using a SQL<br>statement such<br>as: ALTER<br>SYSTEM ALTER<br>CONFIG SET<br>'sql.pipelineParameters.extract.binaryFormat.defaultPaddingCharacter'<br>= ' * ' |                      |           |             |

The Ociant System trims the default padding character of a space from the end of the text data in binary data.

## Bad Data Targets

Bad data represents records that the Ociant System could not load due to errors in the transformations or invalid data in the source records. You can provide options for a bad data target that the Ociant System uses during pipeline execution to capture the records that are not loaded. The original bytes that the pipeline tried to load are captured in the bad data target along with the metadata about the error, such as the error message or source.

Kafka is the only supported bad data target.

### Kafka Bad Data Target

When you use Kafka as a bad data target, the Ociant System produces the original bytes of the source record into the Kafka topic of your choice. The Ociant System includes the metadata about the record in the header of the record as it is sent to Kafka. You can configure the Kafka topic on your Kafka Brokers using the retention and partition settings of your choice.

In the event that the Kafka Broker is unreachable when the Ociant System attempts to produce a bad data record to the bad data target, the system logs an error on the Loader Node and the pipeline continues.

## Example

This example CREATE PIPELINE SQL statement snippet contains a bad data target definition using the BAD\_DATA\_TARGET option.

```
CREATE PIPELINE ...
  BAD_DATA_TARGET
  KAFKA
  TOPIC 'orders_errors'
  BOOTSTRAP_SERVERS '111.11.111.1:9092,111.11.111.2:9092'
```



```
        CONFIG '{"compression.type": "gzip"}'
SOURCE
    ...
EXTRACT
    ...
INTO public.orders
SELECT
    $order.billing.name as username,
    $order.subtotal as subtotal,
    ...
```

Kafka Bad Data Target Options

| Option Key        | Default                        | Required or Optional | Data Type | Description                                                                                                                                                                                             |
|-------------------|--------------------------------|----------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TOPIC             | None                           | Required             | STRING    | The name of the Kafka topic where the Ocient System should produce bad data records.                                                                                                                    |
| BOOTSTRAP_SERVERS | None                           | Required             | STRING    | A comma-delimited list of IP:port pairs that contain the IP addresses of the Kafka Brokers and the associated port number.<br><b>Example:</b> BOOTSTRAP_SERVERS = '111.11.111.1:9092,111.11.111.1:9092' |
| CONFIG            | '{"compression.type": "none"}' | Optional             | STRING    | The producer configuration that the Kafka producer should use. This option is a JSON-formatted string.                                                                                                  |

Advanced Pipeline Tuning Options

You can use pipeline tuning options to control the parallelism or batching dynamics of your pipelines. This tuning can throttle the resources used on a pipeline or increase parallel processing across Loader Nodes. These options are advanced settings that might require a detailed understanding of the underlying mechanics of the loading infrastructure in the Ocient System to employ. Due to the inherent nature of each source type, the behavior of these parameters can differ between file-based and Kafka-based loads.



| Option Key | Default                                                        | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------|----------------------------------------------------------------|----------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CORES      | The maximum number of CPU cores available on each Loader Node. | Optional             | INTEGER   | <p>Maximum number of processing threads that the Ocient System uses during execution on each Loader Node. The system creates this number of threads on each Loader Node. The Ocient System automatically determines the default value by finding the number of cores of a Loader Node. You can use this option for performance tuning. The calculation for maximum parallelism of a pipeline is: <code>number_of_loaders * CORES</code>.</p> <p><b>About Kafka Partitions and Parallelism</b></p> <p>For Kafka Loads, this option determines the number of Kafka Consumers created on each Loader Node. For Kafka Pipelines, the recommendation is that <code>number_of_loaders * CORES</code> equals the number of Kafka topic partitions. If this number exceeds the number of Kafka topic partitions, the work might spread unevenly across Loader Nodes. If this number is less than the number of Kafka topic partitions, some Kafka Consumers might receive uneven amounts of work. In this case, use a value for <code>number_of_loaders * CORES</code> that is an</p> |

| Option Key | Default                      | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|------------|------------------------------|----------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PARTITIONS | Equal to the value of CORES. | Optional             | INTEGER   | Specifies the number of partitions over which to split the file list. Not applicable to Kafka loads. The Ocient System automatically sets a default value based on the configured value for the CORES option. You can use this option for performance tuning. The number of partitions determines how many buckets of work the Ocient System generates for each batch of files processed on a Loader Node. The pipeline processes this number of partitions in parallel using the specified number of cores. If you specify fewer partitions than cores, some cores are not fully utilized, and resources are wasted. If you specify more partitions than cores, the Ocient System divides partitions in a round-robin fashion over the available cores. |

| Option Key | Default                                                                                     | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------|---------------------------------------------------------------------------------------------|----------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BATCH_SIZE | A dynamic value, determined by the Ociant System for each pipeline to maximize performance. | Optional             | INTEGER   | Number of rows in the batch to load at one time. The Ociant System automatically calculates a dynamic value depending on the table columns and the utilization of internal buffers to transfer records to the database backend. You can use this option to turn off the dynamic adjustments for performance tuning. Only change this setting in rare cases where loading performance is slower than expected, and you have a large record size. If this setting is improperly set, pipelines might fail with out-of-memory exceptions. You can configure the default value (for the batch payload target) using a SQL statement such as: <code>ALTER SYSTEM ALTER CONFIG SET 'streamloader.extractorEngine' '65536'</code> |

| Option Key           | Default                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RECORD_NUMBER_FORMAT | For loads that do not use the Special Data Pipeline Transformation function, the default is [19, 45, 0]. For Kafka loads that do not use the EXPLODE_OUTER function, the default is [0, 64, 0]. For loads that use the EXPLODE_OUTER function, the default is the specified load-type-specific default value with 13 subtracted from the record index bits. The system adds these bits to the bits for rows within a record. For example, the default for file loads that use this function is [19, 32, 13]. | Optional             | ARRAY     | <p>The 64-bit record number for each record of the load. This number uniquely identifies a row within its partition. The format is an array with three values in the format [&lt;file_index_bits&gt;, &lt;record_index_bits&gt;, &lt;rows_per_record_index_bits&gt;].</p> <p>file index bits&lt;file_index_bits&gt;value is the number of bits used to represent the file index within a partition. The record index bits&lt;record_index_bits&gt;value is the number of bits used to represent the record index within a file. The rows per record index bits&lt;rows_per_record_index_bits&gt;value is the number of bits used to represent the row within a record. The system uses this value with the EXPLODE_OUTER function. These three values must sum to 64.</p> <p><b>Example</b> RECORD_NUMBER_FORMAT [10, 54, 0] Set the number of file index bits to 10 and the number of record index bits to 54, allowing up to <math>2^{10}</math> files and <math>2^{54}</math> records per file. The system does not support the EXPLODE_OUTER function in this configuration because the rows per record index bits are 0.</p> |

## DROP PIPELINE

**DROP PIPELINE** removes an existing pipeline in the current database. You cannot remove a pipeline that is running. You must have the **DROP** privilege on the pipeline to execute this SQL statement. For details, see [Data Control Language \(DCL\) Statement Reference](#).

When you drop a pipeline, the Ocient System also removes the associated system catalog information, such as pipeline errors, events, files, partitions, and metrics.

### Syntax

```
DROP PIPELINE [ IF EXISTS ] pipeline_name [, ...]
```

| Parameter     | DataType | Description                                                                                                                                            |
|---------------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| pipeline_name | string   | The name of the specified pipeline to remove. You can drop multiple pipelines by specifying additional pipeline names and separating each with commas. |

### Examples

Remove an existing pipeline named `ad_data_pipeline`.

```
DROP PIPELINE ad_data_pipeline;
```

Remove an existing pipeline named `ad_data_pipeline` or return a warning if the Ocient System does not find the pipeline in the database.

```
DROP PIPELINE IF EXISTS ad_data_pipeline;
```

## PREVIEW PIPELINE

**PREVIEW PIPELINE** enables you to view the results of loading data for a specific **CREATE PIPELINE** SQL statement without creating a whole data pipeline and without storing those results in the target table. Using this SQL statement, you can iterate quickly and modify the syntax as needed to achieve your expected results. After you confirm your expected results, you can use the same syntax in the body of the **CREATE PIPELINE** statement with the appropriate source.

A table must exist in the database to serve as the target of your **PREVIEW PIPELINE** statement. This table ensures the pipeline matches the column types of the target table. However, the execution of this statement does not load data into the target table.

### Preview Sources

The **SOURCE INLINE** source type is available only for the **PREVIEW PIPELINE** SQL statement. You cannot create a data pipeline with inline source data.

Other source types defined in the **CREATE PIPELINE** statement, **S3**, **KAFKA**, and **FILESYSTEM**, are compatible with the **PREVIEW PIPELINE** statement. The extract options vary by the source type to mirror the **CREATE PIPELINE** statement. The database returns 10 records by default.

### Preview Error Handling

Pipeline-level errors cause the **PREVIEW PIPELINE** SQL statement to fail. The Ocient System returns an error and no result set. However, the Ocient System accumulates record-level errors that occur during the execution of this statement in a single warning that the system returns along with the result set. Each line of the warning describes a

record-level error in human-readable form or as a JSON blob, depending on the value of the `SHOW_ERRORS_AS_JSON` option. Rows or columns that encounter record-level errors have `NULL` values in the result set.

### Preview Limitations

Limitations of this SQL statement are:

- Before executing a `PREVIEW PIPELINE` SQL statement, you must create a table for the Ocient System to have context for the preview.
- The `COLUMN_DEFAULT_IF_NULL` option from the `CREATE PIPELINE` SQL statement has no effect on the `PREVIEW PIPELINE` SQL statement.
- The `PREVIEW PIPELINE` SQL statement does not honor the assignment of a service class based on text matching.
- These source options are not supported:
  - `START_TIMESTAMP`
  - `END_TIMESTAMP`
  - `START_FILENAME`
  - `END_FILENAME`
- When you execute two duplicate `PREVIEW PIPELINE` statements for a specific Kafka topic, the two statements share a consumer group. If the topic is small, one or both of the result sets might only be a partial result.

### Syntax

```
PREVIEW PIPELINE pipeline_name
```

```

[ MODE mode ]
[ SHOW_ERRORS_AS_JSON show_errors_as_json ]

SOURCE [ INLINE ] (inline_string | <s3_source> | <filesystem_source> | <kafka_source>)
[ LIMIT limit ]
```

```

EXTRACT
  FORMAT csv
  RECORD_DELIMITER record_delimiter
  FIELD_DELIMITERS ['delim1', 'delim2', ...]
  [ INTERMEDIATE_VALUES intermediate_values ]
```

```
INTO created_tablename
```

```

SELECT
  preview_column_formula AS preview_column_name, ...
```

Though this syntax shows the CSV format, you can use the `PREVIEW PIPELINE` statement with the other formats also.

| Parameter                      | Data Type  | Description                                                                                                                |
|--------------------------------|------------|----------------------------------------------------------------------------------------------------------------------------|
| <code>pipeline_name</code>     | string     | The name of the specified data pipeline for the preview.                                                                   |
| <code>created_tablename</code> | identifier | The identifier for the name of the table that you create before executing the <code>PREVIEW PIPELINE</code> SQL statement. |



| Parameter              | Data Type  | Description                                                                                                                                                                                                                                                                                                          |
|------------------------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| preview_column_formula | identifier | The identifier for the formula of the data to load. For example, for the data in the first field of the inline source, use \$1. If you need to add a transformation, you can use functions to transform data, such as CONCAT(\$1, \$2), to load the concatenation of the first two fields in the inline source data. |
| preview_column_name    | identifier | The name of the column in the target table.                                                                                                                                                                                                                                                                          |

### SQL Statement Options

| Option Key          | Default     | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------|-------------|----------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MODE                | 'transform' | Optional             | string    | Indicates whether to perform a validation of the PREVIEW PIPELINE SQL statement. Valid values are: 'validate' and 'transform'. Set this option to 'validate' for checking that the creation of the data pipeline succeeds. If the pipeline is valid, the statement produces no output; otherwise, it returns an error. Set this option to 'transform' to retrieve a preview of the results of the pipeline. |
| SHOW_ERRORS_AS_JSON | false       | Optional             | Boolean   | Indicates whether to show errors. Values are true or false. If the value is true, the Ocident System returns record-level errors as JSON blobs rather than human-readable messages.                                                                                                                                                                                                                         |

| Option Key          | Default | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                               |
|---------------------|---------|----------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SOURCE_INLINE       | None    | Optional             | string    | The string that contains data for the preview of the data pipeline load. For example, source data can be 'oci,ent,ocient ware,house' where   is the record delimiter and , is the field delimiter. For special characters, such as \t, use an escape sequence such as 'oci,ent,oci\tent ware,house'.                                      |
| LIMIT               | 10      | Optional             | INT       | The number of rows, specified as an integer, to return in the preview results for sources with many rows. The default value is 10 rows.                                                                                                                                                                                                   |
| INTERMEDIATE_VALUES | false   | Optional             | Boolean   | Indicates whether to capture intermediate values during a transformation sequence. Values are true or false. If the value is true, the Ocient System appends an extra column to the result set. Each value in the column contains a JSON blob that describes the intermediate values processed for each column after each transformation. |

You must specify at least one column name in the **SELECT** part of the syntax. The name of the specified column must match the name of the column in the created table. The number of columns in the **SELECT** part can be less than those in the created table.

For definitions of other extract options, see the **CREATE PIPELINE SQL** statement options in [Data Pipelines](#).

## Examples

### Preview Pipeline Using CSV Format

Preview the load of two rows of data. First, create a table to serve as the context for the load. The **previewload** table contains three columns with these data types: string, integer, and Boolean.

```
CREATE TABLE previewload (col1 VARCHAR, col2 INT, col3 BOOLEAN);
```

Create the preview pipeline `testpipeline` with this data: `'hello,2,true|bye,3,false'`. Specify the CSV extract format, `|` record delimiter, and the `,` field delimiter. Load the data without transformation.

```
PREVIEW PIPELINE testpipeline
  SOURCE INLINE 'hello,2,true|bye,3,false'
  EXTRACT
    FORMAT CSV
    RECORD_DELIMITER '|'
    FIELD_DELIMITERS [',']
  INTO previewload
  SELECT
    $1 AS col1,
    $2 AS col2,
    $3 AS col3;
```

*Output*

```
col1 col2 col3
-----
hello 2 true bye 3 false
Fetched 2 rows
```

Delete the `previewload` table.

```
DROP TABLE previewload;
```

### Preview Pipeline Using CSV Format with Escape Characters

Preview the load of two rows of data. First, create a table to serve as the context for the load. The `previewload` table contains three columns with these data types: string, integer, and Boolean.

```
CREATE TABLE previewload (col1 VARCHAR, col2 INT, col3 BOOLEAN);
```

Create the preview pipeline `testpipeline` with this data: `'hello\tworld,2,true|bye\tworld,3,false'`. Specify the CSV extract format, `|` record delimiter, and `,` field delimiter. Load the data without transformation. In this case, the data contains the special character `\t`. You must escape the character by using the escape sequence `e`.

```
PREVIEW PIPELINE testpipeline
  SOURCE INLINE e'hello\tworld,2,true|bye\tworld,3,false'
  EXTRACT
    FORMAT CSV
    RECORD_DELIMITER '|'
    FIELD_DELIMITERS [',']
  INTO previewload
  SELECT
    $1 AS col1,
    $2 AS col2,
    $3 AS col3;
```

*Output*

```
col1 col2 col3
-----
hello world 2 true bye world 3 false
Fetched 2 rows
```

Delete the `previewload` table.

```
DROP TABLE previewload;
```

### Preview Pipeline Using CSV Format with Transformation

Create a table to serve as the context for the load. The `previewload` table contains three string columns.

```
CREATE TABLE previewload (col1 VARCHAR, col2 VARCHAR, col3 VARCHAR);
```

Create the preview pipeline `testpipeline` with this data: `'hello,world|bye,world'`. Specify the CSV extract format, `|` record delimiter, and the `,` field delimiter. Load the data with a transformation to concatenate the two strings and return the result in the third column.

```
PREVIEW PIPELINE testpipeline
  SOURCE INLINE 'hello,world|bye,world'
  EXTRACT
    FORMAT CSV
    RECORD_DELIMITER '|'
    FIELD_DELIMITERS [',']
  INTO previewload
  SELECT
    $1 AS col1,
    $2 AS col2,
    CONCAT($1,$2) AS col3;
```

*Output*

| col1       | col2      | col3     |
|------------|-----------|----------|
| helloworld | bye world | byeworld |

hello world

helloworld bye world byeworld

Fetches 2 rows

The third column contains the concatenated result of the first two columns.

Delete the `previewload` table.

```
DROP TABLE previewload;
```

### Preview Pipeline Using the Kafka Source

Create the `previewload` table with these columns:

- `id` — Non-NULL integer
- `salut` — Non-NULL string
- `name` — Non-NULL string
- `surname` — Non-NULL string
- `zipcode` — Non-NULL integer
- `age` — Non-NULL integer
- `rank` — Non-NULL integer

```
CREATE TABLE previewload (
  id INT NOT NULL,
  salut VARCHAR(3) NOT NULL,
  name VARCHAR(10) NOT NULL,
  surname VARCHAR(10) NOT NULL,
  zipcode INT NOT NULL,
  age INT NOT NULL,
  rank INT NOT NULL);
```

Create the preview pipeline `test_small_kafka_simple_csv`. Specify the `ddl_csv` topic. Indicate that the Kafka consumer should not write its durably-made record offsets to the Kafka Broker by using the `WRITE_OFFSETS` option set to `false`. Specify the bootstrap server as `servername:0000` and configuration options as `"auto.offset.reset": "earliest"` by using the `BOOTSTRAP_SERVERS` and `CONFIG` options, respectively. Limit the returned results to three rows by using the `LIMIT` option. Specify the CSV extract format and `\n` record delimiter by using the `FORMAT` and `RECORD_DELIMITER` extract options, respectively.

```
PREVIEW PIPELINE test_small_kafka_simple_csv
SOURCE KAFKA
  TOPIC 'ddl_csv'
  WRITE_OFFSETS false
  BOOTSTRAP_SERVERS 'servername:0000'
  CONFIG '{"auto.offset.reset": "earliest"}'
  LIMIT 3
EXTRACT
  FORMAT csv
  RECORD_DELIMITER '\n'
INTO previewload
SELECT
  INT($1) AS id,
  CHAR($2) AS salut,
  CHAR($3) AS name,
  CHAR($4) AS surname,
  INT($5) AS zipcode,
  INT($6) AS age,
  INT($7) AS rank;
```

```
id salut name surname zipcode age rank
```

```
105 Mr JmhsuxofspX Uaofgayjugb 85573 29 2 101 Mr Ijmmtdkdyh Yqbxqnkgidp 52393 43 1 109 Mr Bigohpwwmr
Qcxgakpkoeu 74420 1 3
```

```
Fetchd 3 rows
```

## START PIPELINE

START PIPELINE begins the execution of the specified data pipeline that extracts data and loads it into the target tables specified by the CREATE PIPELINE SQL statement.

When you execute the START PIPELINE SQL statement, the Ocint System creates a static list of files in the `sys.pipeline_files` system catalog table and marks them with the `PENDING` status. After the system assigns a file to an underlying task, the system marks the file as `QUEUED`. After the system verifies that the file exists, the system marks the file as `LOADING` to signify that a Loader Node has started reading the source data. Finally, upon successfully loading the file, the system transitions the status of the file to the terminal status `LOADED`.

A Kafka pipeline never enters the `COMPLETED` state in the `information_schema.pipeline_status` view. Instead, the pipeline remains running after you start the pipeline until you stop it or the pipeline reaches the specified error limit using the `ERROR LIMIT` option.

You must have the `EXECUTE` privilege on the pipeline and the `INSERT` privilege on any table that is a target in the pipeline. For details, see [Data Control Language \(DCL\) Statement Reference](#).

### Syntax

```
START PIPELINE pipeline_name
[ ERROR
  [ LIMIT <integer_value> ]
  [ SCOPE PIPELINE ]
  [ FILE_ERROR (FAIL | SKIP_MISSING_FILE | TOLERATE) ] ]
[ USING LOADERS <loader_names> ]
[ ON COMPLETION (NO_FLUSH | FLUSH_AND_WAIT | FLUSH_AND_RETURN) ]
```

| Parameter     | DataType | Description                              |
|---------------|----------|------------------------------------------|
| pipeline_name | string   | The name of the specified data pipeline. |

## SQL Statement Options

| Option Key                    | Default  | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------------------|----------|----------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ERROR<br>LIMIT<integer_value> | 0        | Optional             | integer   | Error log option that determines the number of record-level errors that can occur during the execution of a pipeline that the load tolerates before the whole pipeline execution fails.<integer_value>is a number greater than or equal to -1. When you set<integer_value>to -1, the load tolerates an unlimited number of record-level errors.By default, continuous pipelines tolerate an unlimited number of record-level errors, whereas batch pipelines tolerate zero errors. |
| ERROR<br>SCOPE<scope_value>   | PIPELINE | Optional             | string    | Error log option that defines the scope at which the load applies the specified error limit.<scope_value>supports thePIPELINEkeyword, which is the scope for the error limit for the whole pipeline. When you set this option, if the pipeline reaches the error limit, the database rolls back all data loaded by the pipeline.                                                                                                                                                   |

| Option Key | Default | Required or Optional | Data Type | Description |
|------------|---------|----------------------|-----------|-------------|
|------------|---------|----------------------|-----------|-------------|

|                                   |      |          |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------------------------------|------|----------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ERROR<br>FILE_ERROR<error_action> | FAIL | Optional | string | For pipelines that load data from S3 or local file sources, this error configuration option determines how to treat unrecoverable file-level errors. Examples of unrecoverable file-level errors are:The file is listed when the pipeline starts but is missing later during the load.The Gzip file is corrupted and cannot be decompressed.The file cannot be downloaded from the source.Record-level error that is not tolerable occurs when tokenizing or transforming data in the file.<error_action>can be one of these keywords:FAIL—Fail the whole pipeline because of a file-level error.SKIP_MISSING_FILE—Only tolerate errors that occur due to missing files. If a file exists in the list when the pipeline starts but is missing later during the load, skip the file and continue with the next file.TOLERATE—Tolerate all unrecoverable file-level errors. In this mode, the load also tolerates an unlimited number of record-level errors.TheFAILED,SKIPPED, andLOADED_WITH_ERRORSfile statuses appear in thesys.pipeline_filesystem catalog tables, respectively, and indicate how the pipeline handled the |
|-----------------------------------|------|----------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

| Option Key                     | Default | Required or Optional | Data Type       | Description                                                                                                                                                                                                                                                                                         |
|--------------------------------|---------|----------------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| USING LOAD-<br>ERSloader_names | none    | Optional             | list of strings | Specify one or more names of Loader Nodes as a comma-separated list for executing theSTART PIPELINESQL statement. If you do not use this option, the Ocient System uses all of the Loader Nodes that are active to execute the pipeline.You can find node names in thesys.nodesystem catalog table. |



| Option Key                     | Default  | Required or Optional | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------------------------|----------|----------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ON COMPLETION<completion_mode> | NO_FLUSH | Optional             | string    | Completion type option that specifies the behavior when the pipeline finishes loading. This option determines when the remaining pages are converted into Segments.<br><b>NO_FLUSH</b> —Do not force a flush of pages. Rely on watermarks and timeouts to trigger final conversion to Segments.<br><b>FLUSH_AND_WAIT</b> —Trigger a flush of pages, initiating final conversion to Segments. The pipeline blocks and waits for the conversion to Segments to complete before marking the pipeline as <b>COMPLETED</b> .<br><b>FLUSH_AND_RETURN</b> —Trigger a flush of pages, initiating the final conversion to Segments. The Ocient System marks the pipeline as <b>COMPLETED</b> immediately following the flush without waiting for conversion to Segments to complete. |

For the query to execute successfully, the specified node names must identify nodes that have:

- **ACTIVE** operational status
- **streamloader** role

When you execute the **START PIPELINE** SQL statement, the Ocient System creates a static list of files only for batch pipelines and a dynamic list for continuous pipelines in the **sys.pipeline\_files** system catalog table.

### Examples

Start an existing pipeline named **ad\_data\_pipeline** with default settings.

```
START PIPELINE ad_data_pipeline;
```

Start an existing pipeline named `ad_data_pipeline` with error tolerance (tolerate 10 errors before aborting the pipeline). For details about error tolerance, see [Error Tolerance in Data Pipelines](#).

```
START PIPELINE ad_data_pipeline
  ERROR LIMIT 10;
```

Data pipelines log a message for each pipeline error to the `sys.pipeline_errors` system catalog table, even if you do not specify the `ERROR` option. Use `BAD_DATA_TARGET` settings to capture the original source data.

Start an existing pipeline named `ad_data_pipeline` using the Loader Node named `stream-loader1`.

```
START PIPELINE ad_data_pipeline
  USING LOADERS "stream-loader1";
```

### Resume a Pipeline with File Loading

In many cases, a file-based pipeline stops executing before completion. You cannot resume a pipeline in a `COMPLETED` status.

To resume a pipeline, use the `START PIPELINE SQL` statement. Before you resume a pipeline, the status of the pipeline must be `CREATED`, `FAILED`, or `STOPPED`. When a pipeline resumes, individual files remain in their most recent status as defined in the `sys.pipeline_files` system catalog table. For `BATCH` pipelines, the Ocint System does not add new files to the eligible file list when the pipeline resumes.

If you modify the contents of files during the loading process, the Ocint System might experience issues with deduplication that cause duplicated rows or missing data. Avoid modifying files after you start a pipeline for the first time.  
Creating new files on your data source does not impact deduplication logic.

The `START` operation groups files using their `extractor_task_id` and `stream_source_id` identifiers. The `stream_source_id` uniquely identifies partitions (i.e., an ordered list of files), and `extractor_task_id` identifies the batch that loads a group of partitions.

#### File Statuses

The Ocint System considers files with the statuses `LOADED`, `LOADED_WITH_ERRORS`, or `SKIPPED` to be in the terminal status, whereas other file statuses are still in process.

- **Completed Batches** — If all the files in a particular batch have terminal status, then the pipeline does not attempt to reload the batch. These files have been completely processed, so the Ocint System ignores modifications to these files.
- **In-Process Batches** — If at least one file in a particular batch does not have terminal status, then the pipeline reloads the entire batch. The pipeline reprocesses the in-process batches and relies on row deduplication to prevent duplication of rows in the target tables.
  - Modifications to files in an in-process batch can but are not guaranteed to be picked up by a restart.
  - Modifications to any files in this batch with the `LOADED`, `LOADED_WITH_ERRORS`, or `SKIPPED` statuses might cause issues with deduplication, leading to duplicate or missed data.
- **Pending Files** — The Ocint System does not assign all `PENDING` files to a partition. The pipeline attempts to load these files after reloading any in-progress batches.

#### Load Duplicate Data from Files

Sometimes you might want to load the same data multiple times. If you want to load a second copy of the source data, you can either:

- Drop and recreate the pipeline to reset the `sys.pipeline_files` system catalog table.
- Create a second pipeline with a new name and the same configuration.

When you truncate the target tables and restart the pipeline, the Ocint System does not reload the data.

### Restart with Kafka Loading

Ocint relies on the offset management and consumer group behavior in Kafka to deliver exactly-once loading semantics and to control the Ocint pipeline behavior.

#### Kafka Offsets and Consumer Group Identifiers

If you set the `WRITE_OFFSETS` option to `true` (default value is `true`), the Kafka consumers commit offsets back to Kafka after data is considered durable in the database. The Kafka Broker stores these offsets as the last committed offset for the group identifier `group.id`.

For each pipeline, the group identifier defaults to `<database-name>-<pipeline-name>`, where the `<database-name>` is the name of your database and `<pipeline-name>` is the name of the data pipeline. In most use cases, you should not manually change the `group.id` field for a pipeline.

Any Kafka pipeline that has the same `group.id` starts consuming from its last committed offset, or if you do not set the value, the pipeline uses the Kafka `auto.offset.reset` policy to determine where to start. For details, see Kafka offset management.

If you want to start loading from the beginning of a topic, configure an unused `group.id` field (or use a `group.id` field that did not commit any of its offsets back) and ensure the `auto.offset.reset` Kafka configuration is appropriately set in the `CONFIG` option.

#### Kafka Pipeline Deduplication

The committed offset of a Kafka partition lags slightly behind the rows that have been loaded into Ocint. These lags do not cause an issue with data duplication. If you stop a pipeline before it can commit its most recent durable offset to Kafka, restarting the same pipeline starts loading from the last committed offset. However, the database deduplicates records sent twice for the same pipeline.

Ocint deduplicates Kafka data for the specified combination of pipeline identifier, Kafka topic, and the Kafka partition number.

While the consumer group offsets manage where the pipeline resumes loading, the Ocint System enforces the exactly-once loading of a Kafka partition only if you stop or restart a pipeline with the same pipeline identifier.

If you drop a pipeline and create a new one with the same name, the Ocint System creates a new pipeline identifier. This action does not deduplicate data against data loaded in the original pipeline. To preserve deduplication, instead of dropping the pipeline, use the `CREATE OR REPLACE PIPELINE SQL` statement with the original pipeline name and the pipeline correctly deduplicates against the original data.

Do not run multiple pipelines concurrently with the same consumer group identifier. This action leads to unpredictable data duplication. If you want to increase the number of consumers that read from a Kafka topic, increase the value of the `CORES` parameter.

#### Load Duplicate Data on Kafka

Sometimes you might want to load the same data multiple times. If you want to load a second copy of the source data from Kafka, you can either:

- Drop the pipeline, recreate it with the same name, and reset the consumer group offsets manually.
- Create a new pipeline with a different name and load from the beginning of the topic.

## Pipeline Database Dependency

Each pipeline belongs to a database. You cannot drop a database that has a running pipeline. To drop a database, ensure that all pipelines in the database are in a non-running status.

## Pipeline Table Dependency

Each pipeline has a target table. You cannot drop a table that has a running pipeline. To drop a table, ensure that all pipelines that are loading data into the table are in a non-running status.

## STOP PIPELINE

STOP PIPELINE stops the execution of the pipeline and its associated tasks. After you stop a pipeline, you can execute the START PIPELINE SQL statement on the pipeline to run the pipeline again. Regardless, the load deduplicates any records previously loaded in the same pipeline.

You must have the EXECUTE privilege on the pipeline to execute this SQL statement. For details, see [Data Control Language \(DCL\) Statement Reference](#).

### Syntax

```
STOP PIPELINE pipeline_name
```

| Parameter     | DataType | Description                              |
|---------------|----------|------------------------------------------|
| pipeline_name | string   | The name of the specified data pipeline. |

### Example

Stop an existing pipeline named `ad_data_pipeline`.

```
STOP PIPELINE ad_data_pipeline;
```

You can see the status of the parent tasks in the `sys.tasks` system catalog table and see the status of the child tasks in the `sys.subtasks` system catalog table.

## ALTER PIPELINE RENAME

ALTER PIPELINE RENAME TO SQL statement changes the name of the pipeline object, while retaining its identifier, options, and other metadata. The Ocint System reflects this change in the `sys.pipelines` system catalog table. Then, you must use the new name when you refer to the pipeline in SQL statements.

You must have the ALTER privilege on the pipeline to execute this SQL statement. For details, see [Data Control Language \(DCL\) Statement Reference](#).

### Syntax

```
ALTER PIPELINE [ IF EXISTS ] pipeline_original_name RENAME TO pipeline_new_name
```

| Parameter              | DataType | Description                             |
|------------------------|----------|-----------------------------------------|
| pipeline_original_name | string   | The name of the existing data pipeline. |
| pipeline_new_name      | string   | The new name of the data pipeline.      |

### Example

Rename an existing pipeline named `ad_data_pipeline` to `renamed_pipeline`.

```
ALTER PIPELINE ad_data_pipeline RENAME TO renamed_pipeline;
```

## EXPORT PIPELINE

EXPORT PIPELINE returns the CREATE PIPELINE SQL statement used to create the pipeline object. You can use the output of this statement to recreate an identical pipeline when you remove the original pipeline.

The execution of this statement censors sensitive S3 values like ACCESS\_KEY\_ID and SECRET\_ACCESS\_KEY and Kafka [Consumer Configuration](#) password-type fields. The database replaces them with \*\*\*\*\*.

To execute this statement, you must have the VIEW privilege on the pipeline and any table the pipeline targets. For details, see [Data Control Language \(DCL\) Statement Reference](#).

### Syntax

```
EXPORT PIPELINE pipeline_name
```

| Parameter     | DataType | Description                              |
|---------------|----------|------------------------------------------|
| pipeline_name | string   | The name of the specified data pipeline. |

### Example

Export an existing pipeline in the database ad\_data\_pipeline.

```
EXPORT PIPELINE ad_data_pipeline;
```

## CREATE PIPELINE FUNCTION

CREATE PIPELINE FUNCTION enables you to define a function for loading data. Define the function behavior using the Apache® Groovy™ language. For details about this language, see The Apache Groovy Programming Language.

Function arguments and output are strongly typed and immutable.

You can test the execution of your function using the PREVIEW PIPELINE SQL statement.

The Ocient System does not support the overload of function names.

### Syntax

```
CREATE [ OR REPLACE ] PIPELINE FUNCTION [ IF NOT EXISTS ]
    function_name( input_argument [, ...] )
    LANGUAGE GROOVY
    RETURNS output_argument_definition
    IMPORTS [ library_name [, ...] ]
    AS $$
        groovy_declaration
    $$
```

| Parameter     | Type   | Description                                         |
|---------------|--------|-----------------------------------------------------|
| function_name | string | A unique identifier for the data pipeline function. |

| Parameter                               | Type   | Description                                                                                                                                                                                                                                                                                       |
|-----------------------------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>input_argument</code>             | string | The name of one or more input arguments of the function. Specify data types for input arguments according to the support data types defined in <a href="#">Data Types for Data Pipelines</a> . For the data type declaration, use <code>NOT NULL</code> where applicable for maximum performance. |
| <code>output_argument_definition</code> | string | The type definition of the output from the function.                                                                                                                                                                                                                                              |
| <code>library_name</code>               | string | The name of one or more <code>{{Java}}</code> libraries. You can include libraries by using the <code>IMPORTS</code> clause or specifying the fully-qualified class (e.g., <code>java.lang.Integer</code> ) path in the source definition.                                                        |
| <code>groovy_declaration</code>         | string | The Groovy definition of the function.                                                                                                                                                                                                                                                            |

### Install and Enable Third-Party Libraries

You can use the default list of supported third-party libraries or additional third-party libraries that you install.

#### Supported Libraries

Data pipeline functions can import classes from the default list of supported third-party libraries. This table provides the resources for each supported library package.

| Library Package                                      | Resource                                                         |
|------------------------------------------------------|------------------------------------------------------------------|
| <code>java.lang.*</code>                             | <code>java.lang</code>                                           |
| <code>java.util.*</code>                             | <code>java.util</code>                                           |
| <code>java.nio.ByteBuffer.*</code>                   | <code>ByteBuffer</code>                                          |
| <code>groovy.json.*</code>                           | <code>groovy.json</code>                                         |
| <code>groovy.xml.*</code>                            | <code>groovy.xml</code>                                          |
| <code>groovy.yaml.*</code>                           | <code>groovy.yaml</code>                                         |
| <code>org.apache.groovy.datetime.extensions.*</code> | <code>org.apache.groovy.datetime.extensions</code>               |
| <code>org.apache.groovy.dateutil</code>              | <code>org.apache.groovy.dateutil.extensions</code>               |
| <code>com.ocient.streaming.data.types.*</code>       | <code>Data Types for User-Defined Data Pipeline Functions</code> |

#### Additional Libraries

You can install and enable additional third-party libraries to import for use in your data pipeline functions.

You must install the JAR package on all Loader Nodes in the `/opt/ocient/current/lib/extractorengine_udt` folder.

Then, add the fully qualified class name in the function import list as part of the `library_name` parameter. For example, to reference the `ByteBuffer` class from the `com.fastbuffer` package, specify `com.fastbuffer.ByteBuffer` in the `library_name` parameter and use the class in the Groovy definition as `var x = new com.fastbuffer.ByteBuffer()`.

### Groovy Data Type Mapping

For the Groovy definition, the Ociant System maps its SQL data type to the corresponding Groovy data type. Your Groovy code should use the Groovy data type defined in this table for any input arguments and output.

| SQL Data Type              | Groovy Data Type                                 |
|----------------------------|--------------------------------------------------|
| BIGINT                     | java.lang.Long                                   |
| BINARY(N) or HASH(N)       | byte[]                                           |
| BOOLEAN                    | java.lang.Boolean                                |
| CHAR(N) or VARCHAR(N)      | java.lang.String                                 |
| DATE                       | java.time.LocalDate                              |
| DECIMAL(P,S)               | com.ocient.streaming.data.types.Decimal          |
| DOUBLE                     | java.lang.Double                                 |
| INT                        | java.lang.Integer                                |
| IPV4                       | java.net.Inet4Address                            |
| IP                         | java.net.Inet6Address                            |
| ST_POINT                   | com.ocient.streaming.data.types.gis.STPoint      |
| ST_LINestring              | com.ocient.streaming.data.types.gis.STLinestring |
| ST_POLYGON                 | com.ocient.streaming.data.types.gis.STPolygon    |
| FLOAT                      | java.lang.Float                                  |
| SMALLINT                   | java.lang.Short                                  |
| TIME                       | com.ocient.streaming.data.types.Time             |
| TIMESTAMP                  | com.ocient.streaming.data.types.Timestamp        |
| BYTE                       | java.lang.Byte                                   |
| TUPLE<<type1, type2, ...>> | com.ocient.streaming.data.types.OcientTuple      |
| TYPE[]                     | java.util.List<TYPE>                             |
| UUID                       | java.util.UUID                                   |
| VARBINARY(N)               | byte[]                                           |
| VARCHAR(N)                 | java.lang.String                                 |

### Example

Create the `sort_function` data pipeline function to sort an array of integers. The function has two input arguments: `value`, a non-NULL array of integers, and `ascending`, the sort order. The function returns a non-NULL array of integers.

The function imports these Java libraries:

- `java.lang.Integer`
- `java.util.ArrayList`
- `java.util.Collections`
- `java.util.Comparator`
- `java.util.List`

Define the Groovy code. Because the input arguments do not change, the example Groovy code first copies the `value` argument, sorts the copied list according to the sort order, and returns the sorted array.

```
CREATE PIPELINE FUNCTION
  sort_function( value INT[] NOT NULL, ascending BOOLEAN NOT NULL)
LANGUAGE GROOVY
RETURNS INT[] NOT NULL
IMPORTS [
  "java.lang.Integer",
  "java.util.ArrayList",
  "java.util.Collections",
  "java.util.Comparator",
  "java.util.List"
]
```

```

AS $$
  /* Make a copy of the list. */
  var sorted = new ArrayList<Integer>((List<Integer>)value);

  /* Sort the array elements according to the specified order. */
  Collections.sort(
    sorted,
    ascending ? Comparator.naturalOrder() : Comparator.reversed());

  /* Return the sorted array. */
  return sorted;
$$;

```

View the creation information about the `sort_function` function using the `sys.pipeline_functions` system catalog table. This statement returns the function name, return type, argument names, data types of the arguments, and the list of imported libraries.

```

SELECT name, return_type, argument_names, argument_types, imported_libraries
FROM sys.pipeline_functions;

```

## DROP PIPELINE FUNCTION

DROP PIPELINE FUNCTION removes an existing pipeline function.

You must have the DROP privilege on the pipeline function to execute this SQL statement. For details, see [Data Control Language \(DCL\) Statement Reference](#).

### Syntax

```
DROP PIPELINE FUNCTION [ IF EXISTS ] function_name [, ...]
```

| Parameter     | Data Type | Description                                                                                                                                                          |
|---------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| function_name | string    | The name of the specified data pipeline function to remove. You can drop multiple pipelines by specifying additional function names and separating each with commas. |

### Examples

#### Remove the Existing Pipeline Function

Remove an existing pipeline function named `sort_function`.

```
DROP PIPELINE FUNCTION sort_function;
```

#### Remove an Existing Pipeline Function by Checking for Existence

Remove an existing pipeline function named `sort_function` or return a warning if the Ocient System does not find the function in the database.

```
DROP PIPELINE FUNCTION IF EXISTS sort_function;
```

### Related Links

[Load Data](#)

[Data Pipeline Load of CSV Data from S3](#)

[Data Pipeline Load of Parquet Data from S3](#)



[Transform Data in Data Pipelines](#)

[Load Metadata Values in Data Pipelines](#)

[Data Control Language \(DCL\) Statement Reference](#)

[Identifiers](#)

#### 10.21.4 Distributed Tasks

You can manage distributed tasks in the Ocient® System. These tasks are processes that run in the background of the system to keep it running smoothly and efficiently. These DDL commands create tasks to rebuild damaged or missing segment data and check data integrity. For information about managing these tasks, see [Manage Distributed Tasks](#).

### CREATE TASK

Launches a new task.

To create a rebuild or check disk task, you must be assigned the System Administrator role.

#### Syntax

```
CREATE TASK {
  <rebuild_segment_task_definition>
  | <check_disk_task_definition>
  | <rebalance_task_definition>
}

<rebuild_segment_task_definition> ::=
  [name] TYPE REBUILD [ OPTIONS ('location_type' [, 'location_name' [, <rebuild_args> ]])]

<check_disk_task_definition> ::=
  [name] TYPE CHECK_DISK
  [ OPTIONS ('location_type' [, 'location_name' [, <check_disk_args> ]])]

<rebalance_task_definition> ::=
  [name] TYPE REBALANCE
```

Rebuilding Segment Task ( <rebuild\_segment\_task\_definition> )

Performing a rebuild segment operation reconstructs data segments by using erasure-coded data in the same segment group. This task allows segments that are in DAMAGED or MISSING status to be restored to full performance.

For details about rebuild tasks, see the [Guide to Rebuilding Segments](#). To create a rebuild segment task, you must have UPDATE privileges on the system.

Specifying locationType of SYSTEM means every segment in the system is rebuilt. Alternatively, you can specify individual clusters or nodes by name to rebuild their segments.

| Parameter | Data Type | Description                          |
|-----------|-----------|--------------------------------------|
| name      | string    | An optional identifier for the task. |

| Parameter     | Data Type | Description                                                                                                                                                                                                                                                                 |
|---------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| location_type | string    | Specify the type of location for rebuild from these selections: 'SYSTEM'   'CLUSTER' The value definitions are: <b>SYSTEM</b> — Every segment in the system is rebuilt. <b>CLUSTER</b> — Only the cluster specified by the <code>location_name</code> parameter is rebuilt. |
| location_name | string    | The name of a specific cluster to rebuild.                                                                                                                                                                                                                                  |

#### Rebuild Arguments ( <rebuild\_args> )

Additional arguments for the rebuild task. These arguments must be in a valid JSON format. All fields are optional. You must specify all arguments in a single JSON object.

```
<rebuild_args> ::=
```

```
{
  "node": "node_name",
  "segment_groups": segment_array,
  "table_id": "table_id_string"
}
```

| Parameter      | Data Type | Description                                                                                                                                                                                                                                          |
|----------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| node           | string    | If included, this option limits the rebuild process to only rebuild segments on the node specified by <code>node_name</code> . If not included, the system rebuilds all segment groups on the cluster or system.                                     |
| segment_groups | string    | If included, this option limits the rebuild process to only the identifiers for the rebuild segment groups specified in <code>segment_array</code> . If not included, the system rebuilds all segment groups on the cluster or system.               |
| table_id       | string    | If included, this option limits the rebuild process to only rebuild segment groups that belong to the table identifier specified in <code>table_id_string</code> . If not included, the system rebuilds all segment groups on the cluster or system. |

#### Checking Disk Task ( <check\_disk\_task\_definition> )

A **CHECK\_DISK** task verifies the segments on a node by computing checksums.

| Parameter     | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name          | string    | An optional identifier for the task.                                                                                                                                                                                                                                                                                                                                                                                              |
| location_type | string    | The type of location for the check disk task to be performed on. Supported <code>location_type</code> values include: [ 'SYSTEM'   'CLUSTER'   'NODE' ] The value definitions are:<br><b>SYSTEM</b> — The task includes all segments.<br><b>CLUSTER</b> — The task covers only a cluster specified by the <code>location_name</code> .<br><b>NODE</b> — The task covers only a node specified by the <code>location_name</code> . |
| location_name | string    | The specific name of the location where the check disk task is to be performed. This argument is required if the <code>location_type</code> argument is set to 'CLUSTER' or 'NODE'.                                                                                                                                                                                                                                               |

#### Check Disk Arguments (<check\_disk\_args>)

Arguments must be in valid JSON format. All fields are optional. You must specify all arguments in a single JSON object.

```
<check_disk_args> ::=
'{
  "fix": true,
  "sample": false,
  "only_unhealthy": false,
  "storage_ids": ["02c65e30-3651-478e-96bf-9935c1bc8fb4"]
}'
```

| Parameter      | Data Type | Description                                                                                                                                                                                                                                                                                      |
|----------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fix            | Boolean   | If set to <code>true</code> , this option modifies the storage cluster state based on the result of the checksum verification. For example, if you set the <code>fix</code> option, the system moves a segment that fails the checksum from the <b>INTACT</b> state to the <b>DAMAGED</b> state. |
| sample         | Boolean   | If set to <code>true</code> , this option checks only a sample of data in each segment.                                                                                                                                                                                                          |
| only_unhealthy | Boolean   | If set to <code>true</code> , this option checks only stored segments that are not in the <b>INTACT</b> state.                                                                                                                                                                                   |
| storage_ids    | string    | This option limits the disk check operation only to include the specified segment storage identifiers. Otherwise, the system checks all segments on the node.                                                                                                                                    |

## Examples

Rebuild all segments on a system.

```
CREATE TASK TYPE rebuild;
```

```
CREATE TASK TYPE rebuild OPTIONS ('SYSTEM');
```

Rebuild all segments on a system for a specified table with the table identifier 529a2e9d-d06c-46cd-a93b-624d3bed1c08.

```
CREATE TASK TYPE rebuild OPTIONS ('SYSTEM', '{"table_id": "529a2e9d-d06c-46cd-a93b-624d3bed1c08"}');
```

Rebuild segments on Foundation cluster my\_lts\_cluster1.

```
CREATE TASK TYPE rebuild OPTIONS ('CLUSTER', 'my_lts_cluster1');
```

Rebuild segments on a Foundation Node my\_node1.

```
CREATE TASK TYPE rebuild OPTIONS ('CLUSTER', 'my_lts_cluster1', '{"node": "my_node1"}');
```

Rebuild segments on a segment group with the identifier 53.

```
CREATE TASK TYPE rebuild OPTIONS ('CLUSTER', 'my_lts_cluster1', '{"segment_groups": [53]}');
```

Rebuild only the segment in segment group 53 on a Foundation Node my\_node.

```
CREATE TASK TYPE rebuild OPTIONS ('CLUSTER', 'my_lts_cluster1', '{"node": "my_node1", "segment_groups": [53]}');
```

Compute the checksum on all segments on storage node my\_node1.

```
CREATE TASK TYPE check_disk OPTIONS ('NODE', 'my_node1');
```

Compute a checksum on a sample of blocks for each segment on storage cluster my\_lts\_cluster1.

```
CREATE TASK TYPE check_disk OPTIONS ('CLUSTER', 'my_lts_cluster1', '{"sample": true}');
```

Compute a checksum on all nodes in the system and modify the state based on the result.

```
CREATE TASK TYPE check_disk OPTIONS ('SYSTEM', '{"fix": true}');
```

Compute a checksum for a segment with storage identifier 529a2e9d-d06c-46cd-a93b-624d3bed1c08 on the storage node my\_node1.

```
CREATE TASK TYPE check_disk OPTIONS ('NODE', 'my_node1', '{"storage_ids": ["529a2e9d-d06c-46cd-a93b-624d3bed1c08"]}');
```

Rebalance Task (<rebalance\_task\_definition>)

Rebalancing evenly distributes data across disks in the system. Run this task if data is not balanced properly across the system, such as after adding additional nodes or drives.

For details about rebalancing, see [Expand and Rebalance System](#).

## CANCEL TASK

Cancels a running task.

To cancel a task, you must be assigned the System Administrator role.

## Syntax

```
CANCEL TASK ('task_name' | 'task_id')
```

| Parameter | Data Type | Description                                                                                                                                                                                                                       |
|-----------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| task_name | string    | The name specified on task creation. This applies only if an optional name was created for the task.                                                                                                                              |
| task_id   | string    | The unique identifier for the task in the system tables. For rebuild tasks, you can find the unique identifier in the sys.result_cache table. For check disk tasks, you can find the unique identifier in the sys.subtasks table. |

### Example

Cancel a task named my\_task.

```
CANCEL TASK 'my_task';
```

### Related Links

[System Catalog](#)

[Guide to Rebuilding Segments](#)

## 10.21.5 Machine Learning Models

OcientML™ functionality enables you to create a machine learning model, rename the model, export the syntax for the model creation, retrain the model, execute a query against the model, and drop the model.

### CREATE MLMODEL

Train a new machine learning model of type <model\_type> on the result set returned by the SQL SELECT statement. After the database creates the model, <model\_name> becomes a callable function in SQL SELECT statements.

#### Syntax

```
CREATE [ OR REPLACE ] MLMODEL <model_name> TYPE <model_type>
ON( <SQL SELECT statement> ) [options(<option_list>)]
```

model\_name

| Parameter  | Data Type | Description                      |
|------------|-----------|----------------------------------|
| model_name | VARCHAR   | The name of the model to create. |

model\_type

| Parameter  | Data Type | Description                                   |
|------------|-----------|-----------------------------------------------|
| model_type | VARCHAR   | The type of machine learning model to create. |

These models are supported. You can find full descriptions of each model in [Regression Models](#), [Classification Models](#), [Clustering and Dimension Reduction Models](#), or [Other Models](#).

- SIMPLE LINEAR REGRESSION
- MULTIPLE LINEAR REGRESSION
- POLYNOMIAL REGRESSION

- LINEAR COMBINATION REGRESSION
- VECTOR AUTOREGRESSION
- KMEANS
- KNN (K Nearest Neighbors)
- LOGISTIC REGRESSION
- NAIVE BAYES
- NONLINEAR REGRESSION
- FEEDFORWARD NETWORK
- PRINCIPAL COMPONENT ANALYSIS
- LINEAR DISCRIMINANT ANALYSIS
- SUPPORT VECTOR MACHINE
- DECISION TREE
- GAUSSIAN MIXTURE MODEL
- ASSOCIATION RULES

option\_list

| Parameter   | DataType | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| option_list | VARCHAR  | The options for the specified machine learning model that is specified as a comma-separated list in the format:<option name 1> -> <value 1>, <option name 2> -> <value 2>, and so on. Names and values must be all enclosed in single quotes and are case sensitive with the exception that Boolean values can be true,false,TRUE, orFALSE. Refer to the respective model for the full options list.Example options list:options(`'yIntercept' -> '10',``'metrics' -> 'true'``) |

The SQL SELECT statement that serves as the basis for the model must return rows that fit the specified requirements of the model. For example, in multiple linear regression, the first N columns are the independent variables, and the last column is the dependent variable.

You cannot create a machine learning model with an existing schema and name combination.

### Example

Assume you created the `mldata` table that contains the data for the model. Then, you can create the `my_model` machine learning model based on that data.

```
CREATE MLMODEL my_model
TYPE SIMPLE LINEAR REGRESSION ON (
  SELECT
    x1,
    y
  FROM mldata
)
options(
  'yIntercept' -> '10',
  'metrics' -> 'true'
);
```

## ALTER MLMODEL

Rename a machine learning model. Use the IF EXISTS clause to ignore any models that do not exist.

### Syntax

```
ALTER MLMODEL [IF EXISTS] <model name> RENAME TO <new model name>
```

| Parameter      | Data Type | Description                      |
|----------------|-----------|----------------------------------|
| model name     | VARCHAR   | The name of the model to rename. |
| new model name | VARCHAR   | The new name of the model.       |

### Example

```
ALTER MLMODEL my_model RENAME TO my_model_slr;
```

## DROP MLMODEL

Drop a machine learning model. Use the IF EXISTS clause to ignore any models that do not exist.

### Syntax

```
DROP MLMODEL [IF EXISTS] model_name [, ...]
```

| Parameter  | Data Type | Description                                                                                                                 |
|------------|-----------|-----------------------------------------------------------------------------------------------------------------------------|
| model name | VARCHAR   | The name of the model to drop. You can drop multiple models by specifying additional names and separating each with commas. |

### Example

Drop a machine learning model my\_model.

```
DROP MLMODEL my_model;
```

Drop multiple machine learning models.

```
DROP MLMODEL my_model1, my_model2;
```

## EXPORT MLMODEL

Return the SQL statement that can recreate the machine learning model.

### Syntax

```
EXPORT MLMODEL <model name>
```

| Parameter  | Data Type | Description                      |
|------------|-----------|----------------------------------|
| model name | VARCHAR   | The name of the model to create. |

### Example

```
EXPORT MLMODEL my_model;
```

### Output

```
export
```

```
-----
CREATE MLMODEL "my_schema"."my_model" TYPE simple linear regression ON(select x, y from mldemo.slr) OPTIONS
```

```

Fetched 1 row
```

The output SQL statement includes the schema explicitly.

## REFRESH MLMODEL

Retrain a machine learning model without changing any model options.

### Syntax

```
REFRESH MLMODEL <model name>
```

| Parameter  | Data Type | Description                       |
|------------|-----------|-----------------------------------|
| model name | VARCHAR   | The name of the model to retrain. |

### Example

```
REFRESH MLMODEL my_model;
```

### Related Links

[Machine Learning in Ocint](#)

[Machine Learning Model Functions](#)

[Regression Models](#)

[Classification Models](#)

[Clustering and Dimension Reduction Models](#)

[Other Models](#)

## 10.21.6 Users, Groups, and Service Classes

Users, Groups, and Service Classes are managed using DDL SQL statements that allow administrators to create, drop, or alter each type of database object. These commands also allow administrators to associate service classes with groups and control service class settings for workload management.

### USER

#### CREATE USER

CREATE USER creates a new user, which is scoped to the database of the active connection processing the DDL statement. This means that usernames can be reused across different databases.

### Syntax

```
CREATE USER [ IF NOT EXISTS ] fully_qualified_username <user_definition> [, ...] ]
```

```
<user_definition> ::=
  PASSWORD [=] 'password_string' |
  [ FIRST_NAME [=] 'first_name_string' ] |
  [ LAST_NAME [=] 'last_name_string' ] |
  [ EMAIL [=] 'email_string'
```



| Parameter                             | Data Type | Description                                                                                                                                                                                                                                                                               |
|---------------------------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>fully_qualified_username</code> | string    | A fully qualified username (FQUN) that follows the pattern <code>user_name@database_name</code> and uniquely identifies a user in {{Ocint}}. An FQUN can reference a user associated with a different database than the one you are connected to. For FQUN examples, see the table below. |

If you specify only a `user_name` without a `database_name`, Ocint assumes the user is associated with the database of the active connection.

Define User Parameters (<user\_definition>)

You can apply the following optional parameters when creating a new user object.

| Parameter                      | Data Type | Description                                                  |
|--------------------------------|-----------|--------------------------------------------------------------|
| <code>password_string</code>   | string    | A password for the user account. This parameter is required. |
| <code>first_name_string</code> | string    | Optional. The first name of the user.                        |
| <code>last_name_string</code>  | string    | Optional. The last name of the user.                         |
| <code>email_string</code>      | string    | Optional. The email of the user.                             |

### Example

This example creates a new user named `jmack@test-database` with the password `pass1234`.

```
CREATE USER
  "jmack@test-database" PASSWORD = 'pass1234',
  FIRST_NAME = 'Johnny',
  LAST_NAME = 'Mack';
```

### DROP USER

DROP USER removes an existing user from the system.

To remove a user, you must possess the DROP USER privilege for the user.

```
DROP USER [ IF EXISTS ] fully_qualified_username [,...]
```

| Parameter                | Data Type | Description                                                                                                                                                                                                                                                                                                                                   |
|--------------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fully_qualified_username | string    | A fully qualified username (FQUN) that follows the pattern <code>user_name@database_name</code> and uniquely identifies a user in Ocint. You can drop multiple users by specifying additional usernames and separating each with commas. An FQUN can reference a user associated with a different database than the one you are connected to. |

If you specify only a `user_name` without a `database_name`, Ocint assumes the user is associated with the database of the active connection.

### Example

This example removes a user named `jmack@test-database`.

```
DROP USER "jmack@test-database";
```

This example drops multiple users.

```
DROP USER "jmack@test-database", "jcollins@test-database";
```

### ALTER USER

Change the status of a user using the `ALTER USER` SQL statement.

#### Required Privileges

To change the state of a user, you must have `ALTER` privileges on the user or one of these roles:

- System Administrator Role
- Security Administrator Role
- Administrator Role on the database of the user

Users cannot change their own state.

### Syntax

```
ALTER USER <user_name> { ENABLE | DISABLE | PASSWORD_EXPIRED }
```

| Parameter | Data Type | Description           |
|-----------|-----------|-----------------------|
| user_name | string    | The name of the user. |

### Examples

#### Enable a User

Enable a user named `example_user`.

```
ALTER USER example_user ENABLE;
```

#### Disable a User

Disable a user named `example_user`.

```
ALTER USER example_user DISABLE;
```

### Expire the Password of a User

Change the `example_user` user to the `password_expired` state. This action requires the user to update their password on their next successful login.

```
ALTER USER example_user PASSWORD_EXPIRED;
```

### ALTER USER SET

Changes the password of an existing user.

#### Syntax

```
ALTER USER [ IF EXISTS ] fully_qualified_username SET PASSWORD = password_string
```

| Parameter                             | Data Type | Description                                                                                                                                                                                                                                   |
|---------------------------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>fully_qualified_username</code> | string    | A fully qualified username (FQUN) that follows the pattern <code>user_name@database_name</code> and uniquely identifies a user in Ocint. An FQUN can reference a user associated with a different database than the one you are connected to. |
| <code>password_string</code>          | string    | A password for the user account. There are no restrictions on the string used for a password.                                                                                                                                                 |

If you specify only a `user_name` without a `database_name`, Ocint assumes the user is associated with the database of the active connection.

### Example

This example changes the password of user `"jmack@test-database"` to `'newpass'`.

```
ALTER USER "jmack@test-database" SET PASSWORD = 'newpass';
```

## GROUP

### CREATE GROUP

`CREATE GROUP` creates a new group. The name must be distinct from the name of any existing group in the database.

```
CREATE GROUP [ IF NOT EXISTS ] group_name
```

| Parameter               | Data Type | Description                                                  |
|-------------------------|-----------|--------------------------------------------------------------|
| <code>group_name</code> | string    | A group name unique from any existing group in the database. |

### Example

This example creates a new group named `group1`.

```
CREATE GROUP "group1";
```

## DROP GROUP

DROP GROUP removes an existing group from the system.

To remove a group, you must possess the DROP GROUP privilege for the group.

### Syntax

```
DROP GROUP [ IF EXISTS ] group_name [, ...]
```

| Parameter  | Data Type | Description                                                                                                                             |
|------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| group_name | string    | The identifier of the group to drop. You can drop multiple groups by specifying additional group names and separating each with commas. |

### Example

This example removes a group named group1.

```
DROP GROUP "group1";
```

This example removes multiple groups.

```
DROP GROUP "group1", "group2";
```

## ALTER GROUP

### ALTER GROUP USER

ALTER GROUP adds or removes users from the group using the USER keyword.

### Syntax

```
ALTER GROUP [ IF EXISTS ] groupname { ADD | DROP } USER username [, ...]
```

| Parameter  | Data Type | Description                                                        |
|------------|-----------|--------------------------------------------------------------------|
| group_name | string    | The identifier of the group to alter.                              |
| username   | string    | A username that uniquely identifies a user in the specified group. |

If you specify only a `user_name` without a `database_name`, Ocient assumes the user is associated with the database of the active connection.

### Example

This example adds user1 to an existing group named group1.

```
ALTER GROUP "group1" ADD USER "user1";
```

### ALTER GROUP RENAME

ALTER GROUP renames an existing group by including the RENAME TO keyword.

### Syntax

```
ALTER GROUP [ IF EXISTS ] group_name RENAME TO new_group_name
```

| Parameter      | Data Type | Description                           |
|----------------|-----------|---------------------------------------|
| group_name     | string    | The identifier of the group to alter. |
| new_group_name | string    | The new name for the group.           |

### Example

This example renames an existing group `group1` to `group2`.

```
ALTER GROUP "group1" RENAME TO "group2";
```

### ALTER GROUP SET SERVICE CLASS

Set a service class for the specified group.

### Syntax

```
ALTER GROUP group_name SET SERVICE CLASS sc_name
```

| Parameter  | Data Type | Description                                                                                                                                                        |
|------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| group_name | string    | The identifier of the group to alter.                                                                                                                              |
| sc_name    | string    | The new name of the service class to assign to the group. To unset a service class from a group and restore defaults, specify “DEFAULT” as the service class name. |

### Example

```
ALTER GROUP "wlm_test" SET SERVICE CLASS "high_priority";
```

### ALTER GROUP ALTER SECURITY

Set the security settings at the group level using the `ALTER GROUP ALTER SECURITY` SQL statement. Replace `<security_setting>` with the security setting and `<value>` with the value.

### Syntax

```
ALTER GROUP group_name ALTER SECURITY <security_setting> [=] <value>
```

| Parameter        | Data Type | Description                                                                                                                                                                          |
|------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| group_name       | string    | The identifier of the group for setting security settings.                                                                                                                           |
| security_setting | string    | The security setting with values: <code>password_minimum_length</code> , <code>password_complexity</code> . For details about these values, see Database Password Security Settings. |
| value            | numeric   | An integer to represent one of the security settings. For details about this value, see Database Password Security Settings.                                                         |

### Example

Set the password lifetime to 10 days for the group `example_group`.

```
ALTER GROUP example_group
  ALTER SECURITY password_lifetime_days = 10;
```

## SERVICE CLASS

### CREATE SERVICE CLASS

A service class defines a set of limits on various system parameters.

The Ocint System applies service classes per group. By default, all groups are in the DEFAULT service class, which has no limits.

If a user belongs to multiple groups with different service classes, the system uses the first service class alphabetically by service class name.

#### Syntax

```
CREATE SERVICE CLASS [ IF NOT EXISTS ] sc_name [ <service_class_definition> [, ...] ]
```

```
<service_class_definition> ::=
  MAX_TEMP_DISK_USAGE [=] percentage |
  MAX_ELAPSED_TIME [=] time |
  MAX_CONCURRENT_QUERIES [=] queries |
  MAX_ROWS_RETURNED [=] rows |
  SCHEDULING_PRIORITY [=] priority |
  CACHE_MAX_BYTES [=] cache_max_bytes |
  CACHE_MAX_TIME [=] cache_max_time |
  MAX_ELAPSED_TIME_FOR_CACHING [=] max_elapsed_time_for_caching |
  MAX_COLUMNS_IN_RESULT_SET [=] max_columns_in_result_set |
  PRIORITY_ADJUSTMENT_FACTOR [=] priority_adjustment_factor |
  PRIORITY_ADJUSTMENT_TIME [=] priority_adjustment_time |
  MIN_PRIORITY [=] priority |
  MAX_PRIORITY [=] priority |
  STATEMENT_TEXT { LIKE | REGEX } [=] statement_text_pattern
```

| Parameter | Data Type | Description                          |
|-----------|-----------|--------------------------------------|
| sc_name   | string    | The identifier of the service class. |

Define Service Classes ( <service\_class\_definition> )

When you create a new service class, the Ocint System sets any omitted service class definitions to the maximum or least restrictive value. Each definition should be comma-separated. When you alter a service class definition, the system modifies only the definition you specify.

See the [Workload Management Walkthrough](#) for more details.

#### Example

This example creates a service class named `sc_name`.

```
CREATE SERVICE CLASS sc_name
  MAX_TEMP_DISK_USAGE = 80,
  MAX_ELAPSED_TIME = 100,
  MAX_CONCURRENT_QUERIES = 10,
  MAX_ROWS_RETURNED = 100,
  SCHEDULING_PRIORITY = 5.0,
  CACHE_MAX_BYTES = 1000,
```

```
CACHE_MAX_TIME = 25,
MAX_ELAPSED_TIME_FOR_CACHING = 50;
```

## DROP SERVICE CLASS

Remove an existing service class from the system.

When dropping a service class, you can use the **FORCE** flag to drop a service class and unset it from any dependent groups. If **FORCE** is not used, the function throws an error if there are dependent groups.

### Syntax

```
DROP SERVICE CLASS sc_name [ IF EXISTS ] [ FORCE ]
```

| Parameter | Data Type | Description                          |
|-----------|-----------|--------------------------------------|
| sc_name   | string    | The identifier of the service class. |

### Example

```
DROP SERVICE CLASS "my_sc";
```

## ALTER SERVICE CLASS

### ALTER SERVICE CLASS RENAME

Rename a service class.

### Syntax

```
ALTER SERVICE CLASS [ IF EXISTS ] sc_name RENAME TO new_name
```

| Parameter | Data Type | Description                                     |
|-----------|-----------|-------------------------------------------------|
| sc_name   | string    | The old identifier of the service class.        |
| new_name  | string    | The new identifier for thesc_nameservice class. |

### Example

```
ALTER SERVICE CLASS "my_sc" RENAME TO "my_new_sc";
```

### ALTER SERVICE CLASS SET

Alter a service class with new parameter definitions by using the **SET** keyword.

### Syntax

```
ALTER SERVICE CLASS [ IF EXISTS ] sc_name
  SET <service_class_definition> [, ...]

<service_class_definition> ::=
  MAX_TEMP_DISK_USAGE [=] percentage |
  MAX_ELAPSED_TIME [=] time |
  MAX_CONCURRENT_QUERIES [=] queries |
  MAX_ROWS_RETURNED [=] rows |
  SCHEDULING_PRIORITY [=] priority |
  CACHE_MAX_BYTES [=] cache_max_bytes |
  CACHE_MAX_TIME [=] cache_max_time |
  MAX_ELAPSED_TIME_FOR_CACHING [=] max_elapsed_time_for_caching |
```

```

MAX_COLUMNS_IN_RESULT_SET [=] max_columns_in_result_set |
PRIORITY_ADJUSTMENT_FACTOR [=] priority_adjustment_factor |
PRIORITY_ADJUSTMENT_TIME [=] priority_adjustment_time |
MIN_PRIORITY [=] priority |
MAX_PRIORITY [=] priority |
STATEMENT_TEXT { LIKE | REGEX } [=] statement_text_pattern

```

| Parameter | Data Type | Description                          |
|-----------|-----------|--------------------------------------|
| sc_name   | string    | The identifier of the service class. |

For the service class definition parameters <service\_class\_definition>, see [Users, Groups, and Service Classes](#).

### Examples

This example changes the value of MAX\_ROWS\_RETURNED to 51 on the service class named `sc_name`.

```
ALTER SERVICE CLASS "sc_name" SET MAX_ROWS_RETURNED = 51;
```

This example makes dynamic priority adjustments at the service class level for a priority adjustment time of 15 seconds, priority adjustment factor of 0.75, minimum priority of 2.0, and maximum priority of 5.0.

```

ALTER SERVICE CLASS "sc_name" SET PRIORITY_ADJUSTMENT_TIME = 15;
ALTER SERVICE CLASS "sc_name" SET PRIORITY_ADJUSTMENT_FACTOR = 0.75;
ALTER SERVICE CLASS "sc_name" SET MIN_PRIORITY = 2.0;
ALTER SERVICE CLASS "sc_name" SET MAX_PRIORITY = 5.0;

```

This example sets the service class `sc_name` to handle queries matching the regular expression `.*my_table.*`.

```
ALTER SERVICE CLASS "sc_name" SET STATEMENT_TEXT REGEX ".*my_table.*";
```

### ALTER SERVICE CLASS RESET

Alter a service class by restoring default values for one or more specified settings.

#### Syntax

```
ALTER SERVICE CLASS [ IF EXISTS ] sc_name RESET <service_class_definition> [, ...]
```

```

<service_class_definition> ::=
NAME [=] name |
MAX_TEMP_DISK_USAGE [=] percentage |
MAX_ELAPSED_TIME [=] time |
MAX_CONCURRENT_QUERIES [=] queries |
MAX_ROWS_RETURNED [=] rows |
SCHEDULING_PRIORITY [=] priority |
CACHE_MAX_BYTES [=] cache max bytes |
CACHE_MAX_TIME [=] cache max time |
MAX_ELAPSED_TIME_FOR_CACHING [=] max elapsed time for caching |
MAX_COLUMNS_IN_RESULT_SET [=] max columns in result set |
PRIORITY_ADJUSTMENT_FACTOR [=] priority adjustment factor |
PRIORITY_ADJUSTMENT_TIME [=] priority adjustment time |
MIN_PRIORITY [=] priority |
MAX_PRIORITY [=] priority |
STATEMENT_TEXT { LIKE | REGEX } [=] statement_text_pattern

```

| Parameter | Data Type | Description                          |
|-----------|-----------|--------------------------------------|
| sc_name   | string    | The identifier of the service class. |



For descriptions of the <service\_class\_definition> parameters, see [Users, Groups, and Service Classes](#).

### Example

This example resets the value of MAX\_ROWS\_RETURNED on the service class named `sc_name`.

```
ALTER SERVICE CLASS "sc_name" RESET MAX_ROWS_RETURNED;
```

## ALTER QUERY

Set the priority of a SQL query that is running. This priority takes effect immediately. The priority must remain within service class priority limits. The Ocint System overrides other preset priorities for this query, including any existing dynamic priority adjustments.

### Syntax

```
ALTER QUERY query_id SET PRIORITY [=] priority
```

| Parameter | Data Type | Description                                      |
|-----------|-----------|--------------------------------------------------|
| query_id  | numeric   | The identifier of the SQL query that is running. |
| priority  | numeric   | The priority of the specified SQL query.         |

### Example

This example sets the priority of the SQL query 34566 to 2.

```
ALTER QUERY 34566 SET PRIORITY 2;
```

### Related Links

[Manage Users, Groups, and Roles](#)

[Data Control Language \(DCL\) Statement Reference](#)

[Database Password Security Settings](#)

## 10.21.7 Data Integrity and Storage

Ocint® provides several capabilities to check on the quality of data on a node and to quarantine segment groups that are problematic for any reason. Related to data integrity is the ability to rebuild tasks. For details, see [Distributed Tasks](#).

### ALTER SEGMENT QUARANTINE

ALTER SEGMENT QUARANTINE isolates segment groups from segment generation or queries. Use this SQL statement if segments are causing unexpected results or crashes. The segments can be un-quarantined by setting quarantine\_level to none.

### Syntax

```
-- To quarantine a single segment group:
ALTER SEGMENT QUARANTINE FOR { <quarantine_level> }
    WHERE SEGMENT_GROUP_ID = seg_id

-- To quarantine multiple segment groups:
ALTER SEGMENT QUARANTINE FOR { <quarantine_level> }
    WHERE SEGMENT_GROUP_ID in ( seg_id [,...] )
```

```
<quarantine_level> ::=
segment_generation | query | all | none
```

| Parameter | Data type | Description                                                                                                                                                                                                                                                                                                                  |
|-----------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| seg_id    | numeric   | The identification numbers for segment groups. Depending on your <code>ALTER SEGMENT QUARANTINE</code> statement, the Ocident System quarantines the specified segment groups for having an existing quarantine removed. For details about finding abnormal segments, see the <a href="#">Guide to Rebuilding Segments</a> . |

Using Quarantine Levels ( `<quarantine_level>` )

You can place quarantines at these segment levels.

| Level              | Description                                                                                                                                                                                                                                              |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| segment_generation | Isolates the specified segments from segment generation. This statement prevents corrupt data from causing problems while still leaving the data available for future analysis.                                                                          |
| query              | Isolates segment groups from subsequent queries. Data that is quarantined at the query level is not returned by database queries. This can be useful to prevent corrupt data from causing query problems while leaving it available for future analysis. |
| all                | Isolates the specified segments from segment generation and subsequent queries.                                                                                                                                                                          |
| none               | Removes any quarantines from the specified segments.                                                                                                                                                                                                     |

## Examples

This example quarantines a single segment group from `segment_generation`.

```
ALTER SEGMENT QUARANTINE FOR segment_generation WHERE SEGMENT_GROUP_ID = 123456789;
```

This example quarantines multiple segment groups from `segment_generation`.

```
ALTER SEGMENT QUARANTINE FOR segment_generation WHERE SEGMENT_GROUP_ID IN (1,2,3,4,5);
```

This example removes a quarantine from a single-segment group.

```
ALTER SEGMENT QUARANTINE FOR none WHERE SEGMENT_GROUP_ID = 123456789;
```

This example removes a quarantine from multiple segment groups.

```
ALTER SEGMENT QUARANTINE FOR none WHERE SEGMENT_GROUP_ID IN (1,2,3,4,5);
```

This example quarantines at the `query` level.

```
ALTER SEGMENT QUARANTINE FOR query WHERE SEGMENT_GROUP_ID = 123456789;
```

This example removes a quarantine at the `query` level.

```
ALTER SEGMENT QUARANTINE FOR none WHERE SEGMENT_GROUP_ID = 123456789;
```

This example quarantines at both the query and segment-group levels.

```
ALTER SEGMENT QUARANTINE FOR all WHERE SEGMENT_GROUP_ID = 123456789;
```

## Related Links

[Errors and Warnings](#)

### 10.21.8 Query Analysis

You can analyze queries by commands that instruct Ocint® to return plan-related information for a database administrator or analyst to evaluate. The results of these commands are vital in performance optimization activities on an Ocint System.

## EXPLAIN

EXPLAIN shows the optimized query plan. The database optimizes the query normally, but it does not execute. Instead, the database returns the query plan in the specified format.

### Syntax

```
EXPLAIN [ PROTO | JSON | DEBUG ] query
```

| Parameter | Data type | Description                                   |
|-----------|-----------|-----------------------------------------------|
| query     | string    | A valid SELECT statement on a database table. |

This table defines the explain plan formats.

| Format | Description                                                                                             |
|--------|---------------------------------------------------------------------------------------------------------|
| PROTO  | Includes the same information as the JSON format, but it is presented in a human-readable presentation. |
| JSON   | The default option. Returns the optimized query plan in JSON format.                                    |
| DEBUG  | The DEBUG format is more compact but does not include some information about each operator.             |

## Examples

This example shows the optimized plan for the query `SELECT * FROM names` in JSON format.

```
EXPLAIN SELECT * FROM names;
```

This EXPLAIN example generates a lengthy output showing the execution plan for the query. This execution plan shows how the database scans the tables referenced by the statement and calculates the memory usage and total execution cost.

This example shows the optimized plan for the query `SELECT * FROM names` in DEBUG format, which produces a more compact execution plan.

```
EXPLAIN DEBUG SELECT * FROM names;
```

### Output

```

ROOT_OP @ (0x7eae71515c40), ID: 94921393, md: 2, Out (1): time_x |
  REORDER_OP @ (0x7f4c19ee1c40), ID: 94921392, md: 0, Out (1): time_x | no-op reorder
    RENAME_OP @ (0x7f49b1bc2c40), ID: 94921390, md: 2, Out (1): time_x | Old2New: (loc.time_x -> time_x)
      GATHER_OP @ (0x7f49b7886a40), ID: 94921389, md: 2, Out (1): loc.time_x |
        GATHER_OP @ (0x7f49b78866c0), ID: 94921388, md: 2, Out (1): loc.time_x |
          RANDOM_SHUFFLE_OP @ (0x7f486825ba40), ID: 94921394, md: 2, Out (1): loc.time_x |
            INDEX_OP @ (0x7f49b792f040), ID: 94921387, md: 1, Out (1): loc.time_x |

```

## EXPLAIN PIPELINE

EXPLAIN PIPELINE shows the pipeline the system uses for compiling a query. The system compiles the pipeline for the specified segments, and if no segment is specified, it picks random segments with the tables.

To explain a pipeline, you must be a system-level user or possess the **SELECT TABLE** privilege for the table used in the query.

### Syntax

```

-- To show pipeline for a random segment:
EXPLAIN PIPELINE query

-- To show pipeline for a specific segment:
EXPLAIN PIPELINE FOR SEGMENT 'storage_id' ON query

-- To show pipeline for multiple specific segments:
EXPLAIN PIPELINE FOR SEGMENT IN ('storage_id' [, ...]) ON query

```

| Parameter  | Data type | Description                                   |
|------------|-----------|-----------------------------------------------|
| query      | string    | A valid SELECT statement on a database table. |
| storage_id | string    | A valid segment storage identifier.           |

### Examples

This example shows the pipeline on the query **SELECT \* FROM names** for a random segment.

```
EXPLAIN PIPELINE SELECT * FROM names;
```

This example shows the pipeline for the segment with the identifier **529a2e9d-d06c-46cd-a93b-624d3bed1c08**.

```
EXPLAIN PIPELINE FOR SEGMENT '529a2e9d-d06c-46cd-a93b-624d3bed1c08'
ON SELECT * FROM names;
```

This example shows the pipeline for segments with these identifiers: **529a2e9d-d06c-46cd-a93b-624d3bed1c08**, **378eac93-2e3d-4ce6-bf1f-d8b7a78fe585**.

```
EXPLAIN PIPELINE FOR SEGMENT IN (
  '529a2e9d-d06c-46cd-a93b-624d3bed1c08',
  '378eac93-2e3d-4ce6-bf1f-d8b7a78fe585')
ON SELECT * FROM names;
```

All the examples generate lengthy output that includes the execution plan for the query for the specified segments. This execution plan shows how the database scans the tables referenced by the statement and calculates the memory usage and total execution cost.

### Related Links

[SQL Reference](#)

### 10.21.9 Statistics Cache Management

System administrators can invalidate internal statistics caches used by the SQL optimizer. The optimizer then immediately sends new requests to Foundation Nodes to repopulate statistics for the underlying data in tables.

The Ocient® System calculates two primary types of statistics. The system calculates Cardinality Distinct Estimates (CDEs) by performing a union operation on the information stored in the table on Foundation Nodes. Also, the system calculates Probability Density Functions (PDFs) by sampling the table data local to the Foundation Node. Then, the system performs a union operation on the CDEs and PDFs on SQL Nodes when stored in the statistics cache. Notably, the union operation for the PDF is not associative and its implementation is not deterministic. The union operation for the CDE is associative, commutative, and deterministic.

The use of these SQL statements in system administration settings is rare. Do not use these statements for day-to-day maintenance of Ocient clusters. Mostly, you can use these commands when large amounts of data have been loaded or removed from the Ocient System quickly, and the system administrator wants to recalculate statistics immediately rather than wait for the configured statistics cache refresh timer to reflect the changes. These syntaxes help to stabilize the optimizer for new data distributions quickly.

#### Invalidate Optimizer Statistics Cache

`INVALIDATE STATS` clears the entire optimizer statistics cache on all SQL Nodes.

If a table has not undergone any row delta after the last statistics cache refresh, the Ocient System does not recalculate statistics by sampling the underlying table. PDF statistics might still change slightly due to non-deterministic union operations. If the table has undergone a row change, the Ocient Sytem recalculates statistics from scratch.

Query performance might be affected until the statistics cache repopulates, specifically for optimization time. For tables that have undergone a row delta, the optimizer temporarily uses the old statistics file again until the recalculated file is available.

#### Syntax

```
INVALIDATE STATS
```

#### Example

Clear the entire optimizer statistics cache on all SQL Nodes.

```
INVALIDATE STATS;
```

#### Invalidate All Base Statistics

The `INVALIDATE STATS FILES` SQL statement forcibly invalidates the CDE and PDF files of the Foundation Node for all tables regardless of changes in the row delta, which triggers the system to recalculate the statistics.

The optimizer statistics cache continues to use its current value until the Foundation Nodes have finished recalculating the new statistics, after which the cache replaces the table entry with the new statistics. This operation should have minimal impact on query optimization times.

Depending on the size of the system, the system might take minutes to hours to recalculate all statistics on the system fully and for new statistics to populate in the optimizer cache.

#### Syntax

```
INVALIDATE STATS FILES
```

#### Example

Invalidate all statistics

```
INVALIDATE STATS FILES;
```

### Invalidate All Base Statistics For a Specific Table

The `ALTER table_name INVALIDATE STATS` SQL statement forcibly invalidates the CDE and PDF files of the Foundation Node for the specified table regardless of changes in the row delta, which triggers the system to recalculate the statistics.

The optimizer statistics cache continues to use its current value until the Foundation Nodes have finished recalculating the new statistics, after which the cache replaces the table entry with the new statistics. This operation should have minimal impact on query optimization times.

#### Syntax

```
ALTER TABLE table_name INVALIDATE STATS
```

| Parameter  | Data Type | Description            |
|------------|-----------|------------------------|
| table_name | string    | The name of the table. |

#### Example

Force the system to recalculate the statistics for the table named `test_table`.

```
ALTER TABLE test_table INVALIDATE STATS;
```

#### Related Links

[Database, Tables, Views, and Indexes](#)

[Plan Optimization](#)

[Query Performance Optimizations](#)

## 10.22 Data Control Language (DCL) Statement Reference

In order to assign privileges to users, administrators can use Data Control Language (DCL) statements. You can write and execute DCL statements like any other SQL statements against the Ocient® System and databases. DCL statements allow you to quickly and easily grant or revoke privileges from users. DCL statements work off two main concepts: Grants and Revokes. Statements that begin with **GRANT** give the associated privileges to a user or group. Statements that begin with **REVOKE** remove those privileges from a user or group.

A full list of privileges can be found in the [Data Control Language \(DCL\) Command Reference](#).

Supported DCL SQL statements are:

- GRANT PRIVILEGE
- REVOKE PRIVILEGE
- GRANT ROLE
- REVOKE ROLE

### Grant Privileges

The **GRANT** statement grants privileges to a user or group. The privileges that you can grant in the **GRANT** statement map to the tables and differ per object. The **WITH GRANT OPTION** keyword specifies that the grantee can also grant privileges on that object.

To grant privileges, you must have:

- **VIEW** privileges for the specified user or group being granted new privileges.
- **SYSAUTH** privileges over the system or database object that the specified user or group is being granted privileges on.

## Syntax

```

GRANT {
  { CREATE { <systemObjects> } | VIEW { TABLE | VIEW }
    | ALTER { TABLE | VIEW }
    | DROP { TABLE | VIEW }
    | SELECT | SYSAUTH | USE | ALL }
    ON SYSTEM
  | { CREATE { <databaseObjects> } | VIEW { TABLE | VIEW }
    | ALTER { TABLE | VIEW }
    | DROP { TABLE | VIEW }
    | SELECT | ALTER | DROP | SYSAUTH | USE | ALL }
    ON DATABASE database_name
  | { VIEW | ALTER | DROP | SYSAUTH | ALL }
    ON { GROUP | USER } object_name
  | { SELECT | DROP | SYSAUTH | ALL }
    ON MLMODEL object_name
  | { VIEW | EXECUTE | ALTER | DROP | SYSAUTH | ALL }
    ON PIPELINE object_name
  | { VIEW | DROP | SYSAUTH | ALL }
    ON PIPELINE FUNCTION object_name
  | { VIEW | SELECT | ALTER | DROP | SYSAUTH | ALL }
    ON VIEW object_name
  | { VIEW | SELECT | ALTER | LOAD | DROP | DELETE | SYSAUTH | INSERT | ALL }
    ON TABLE object_name
  | { VIEW | SELECT | ALL }
    ON TABLE sys.system_catalog_table
}
  TO { { GROUP | USER } object_name | PUBLIC } [ WITH GRANT OPTION ]

<systemObjects> ::=
  DATABASE | TABLE | VIEW | USER | GROUP
<databaseObjects> ::=
  TABLE | VIEW | MLMODEL | PIPELINE | PIPELINE FUNCTION | USER | GROUP

```

When you grant any privilege to a user on a system or database object other than the VIEW privilege, the database implicitly grants the VIEW privilege on the specified object to the user, as well as associated object types (e.g., DROP VIEW ON DATABASE also grants VIEW VIEW ON DATABASE and VIEW ON DATABASE).

## Examples

### Grant Privilege on a Table

This example grants privileges for the SELECT SQL statement on a table to a trusted group.

```
GRANT SELECT ON TABLE company_data TO GROUP trusted_employees;
```

### Grant Privilege on a Database

This example grants the SELECT privilege for all tables and views on the database\_name database to a trusted group.

```
GRANT SELECT ON DATABASE database_name TO GROUP trusted_employees;
```

### Grant Privilege on the System

This example grants the SELECT privilege for all tables and views on all databases in the system to a trusted group.

```
GRANT SELECT ON SYSTEM TO GROUP trusted_employees;
```

For examples of granting privileges for individual users, see [Data Control Language \(DCL\) Command Reference](#).

## Revoke Privileges

The REVOKE statement revokes privileges from a user or group.

To revoke privileges, you must have:

- VIEW privileges for the specified user or group having their privileges revoked.
- SYSAUTH privileges over the system or database object that the specified user or group is having their privileges revoked from.

```
REVOKE [ GRANT OPTION FOR ] {
  { CREATE { <systemObjects> } | VIEW { TABLE | VIEW }
    | ALTER { TABLE | VIEW }
    | DROP { TABLE | VIEW }
    | SELECT | SYSAUTH | USE | ALL }
  ON SYSTEM
  | { CREATE { <databaseObjects> } | VIEW { TABLE | VIEW }
    | ALTER { TABLE | VIEW }
    | DROP { TABLE | VIEW }
    | SELECT | ALTER | DROP | SYSAUTH | USE | ALL }
  ON DATABASE database_name
  | { VIEW | ALTER | DROP | SYSAUTH | ALL }
  ON { GROUP | USER } object_name
  | { SELECT | DROP | SYSAUTH | ALL }
  ON MLMODEL object_name
  | { VIEW | EXECUTE | ALTER | DROP | SYSAUTH | ALL }
  ON PIPELINE object_name
  | { VIEW | DROP | SYSAUTH | ALL }
  ON PIPELINE FUNCTION object_name
  | { VIEW | SELECT | ALTER | DROP | SYSAUTH | ALL }
  ON VIEW object_name
  | { VIEW | SELECT | ALTER | LOAD | DROP | DELETE
    | SYSAUTH | INSERT | ALL }
  ON TABLE object_name
  | { VIEW | SELECT | ALL }
  ON TABLE sys.system_catalog_table
}
FROM { [ USER | GROUP ] object_name | PUBLIC }

<systemObjects> ::=
  DATABASE | TABLE | VIEW | USER | GROUP
<databaseObjects> ::=
  TABLE | VIEW | MLMODEL | PIPELINE | PIPELINE FUNCTION | USER | GROUP
```

When you revoke the VIEW privilege on a system or database object, the database implicitly revokes all privileges on the specified object from the user.

If you revoke the VIEW privilege on an associated object type, the system revokes the privileges for that object type. For example, grant the DROP TABLE privilege to the user by using the GRANT DROP TABLE ON DATABASE SQL statement. Due to implicit granting by the system, this action causes the user to have DROP TABLE, VIEW TABLE, and VIEW privileges. If you revoke the VIEW TABLE privilege using the REVOKE VIEW TABLE ON DATABASE SQL statement, then the system revokes the privileges for the associated type. In this case, the user only retains the VIEW privilege on the database.



**Example**

This example revokes the SELECT privilege on a table from an untrusted group.

```
REVOKE SELECT ON TABLE company_data FROM GROUP untrusted;
```

For examples of revoking privileges for individual users, see [Data Control Language \(DCL\) Command Reference](#).

**Ocient Privileges Reference**

These tables describe the privilege options on each object in Ocient and the allowed privileges.

As of version 23.0, Ocient DCL has replaced the TRUNCATE privilege with the DELETE privilege. Any TRUNCATE privileges, whether manually granted or assigned as part of a user role, should automatically convert to DELETE privileges following a system upgrade to version 23.0 or later.

**Ocient System Privileges**

| Privilege                       | Description                                                                                                                                                                                                                                                                                  |
|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CREATE [ DATABASE, USER, GROUP] | The privilege to create the specified object type. On creation, the creating user inherits all privileges on the object. In the case of the DATABASE object, this is enforced by granting the creating user the newly created database administrator role as described in the Roles section. |
| CREATE [TABLE, VIEW]            | The privilege to create the specified object type on any database where the user can connect. On creation, the creating user inherits all privileges on the object. In order to apply the privilege, the user has to be connected to a user-defined database.                                |
| DROP [TABLE, VIEW]              | The privilege to drop the specified object type on any database where the user can connect. In order to apply the privilege, the user has to be connected to a user-defined database.                                                                                                        |
| ALTER [TABLE, VIEW]             | The privilege to modify the specified object type on any database where the user can connect. In order to apply the privilege, the user has to be connected to a user-defined database.                                                                                                      |
| VIEW [TABLE, VIEW]              | The privilege to see and read schema information of the specified object type on any database where the user can connect.                                                                                                                                                                    |
| SELECT                          | The privilege to read data on any table or view where the user has the VIEW privilege.                                                                                                                                                                                                       |
| SYSAUTH                         | The privilege to grant and revoke all privileges on the system.                                                                                                                                                                                                                              |
| USE                             | The ability to connect to the system.                                                                                                                                                                                                                                                        |

**Database Privileges**

| Privilege                                                               | Description                                                                                                                |
|-------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| CREATE [TABLE, VIEW, MLMODEL, PIPELINE, PIPELINE FUNCTION, USER, GROUP] | The privilege to create the specified object type. The user that creates the object inherits all privileges on the object. |

| Privilege           | Description                                                                                               |
|---------------------|-----------------------------------------------------------------------------------------------------------|
| DROP [TABLE, VIEW]  | The privilege to drop the specified object type on this database.                                         |
| ALTER [TABLE, VIEW] | The privilege to modify the specified object type on this database.                                       |
| VIEW [TABLE, VIEW]  | The privilege to see and read schema information of the specified object type on this database.           |
| SELECT              | The privilege to read data on any table or view, where the user has the VIEW privilege, in the database.  |
| USE                 | Ability to connect to the database. Users created in this database are implicitly granted this privilege. |
| ALTER               | The ability to issue ALTER DATABASE statements on this database.                                          |
| DROP                | The ability to drop this database.                                                                        |
| SYSAUTH             | The ability to grant and revoke all privileges on the database.                                           |

### View Privileges

| Privilege | Description                                                |
|-----------|------------------------------------------------------------|
| VIEW      | Ability to see and read schema information about the view. |
| SELECT    | Ability to read data from the view.                        |
| ALTER     | Ability to modify the view.                                |
| DROP      | Ability to drop the view.                                  |
| SYSAUTH   | Ability to grant privileges on the view.                   |

### Table Privileges

| Privilege | Description                                                 |
|-----------|-------------------------------------------------------------|
| VIEW      | Ability to see and read schema information about the table. |
| SELECT    | Ability to read data from the table.                        |
| ALTER     | Ability to modify the table.                                |
| LOAD      | Ability to load data into the table.                        |
| DROP      | Ability to drop the table.                                  |
| DELETE    | Ability to delete and truncate the table.                   |
| SYSAUTH   | Ability to grant and revoke privileges on the table.        |
| INSERT    | Ability to insert data into the table.                      |

### Machine Learning Model Privileges

| Privilege | Description                                               |
|-----------|-----------------------------------------------------------|
| ALTER     | Ability to rename the model.                              |
| SELECT    | Ability to read data and descriptive data from the model. |
| DROP      | Ability to drop the model.                                |
| SYSAUTH   | Ability to grant and revoke privileges on the model.      |

### Data Pipeline Privileges

| Privilege | Description                                       |
|-----------|---------------------------------------------------|
| VIEW      | See and read information schema about a pipeline. |
| EXECUTE   | Start and stop the execution of a pipeline.       |
| ALTER     | Rename and replace a pipeline.                    |
| DROP      | Drop a pipeline.                                  |
| SYSAUTH   | Grant and revoke privileges on a pipeline.        |

### Data Pipeline Function Privileges

| Privilege | Description                                                                              |
|-----------|------------------------------------------------------------------------------------------|
| VIEW      | See and read information schema about a data pipeline function in system catalog tables. |
| DROP      | Drop a data pipeline function.                                                           |
| SYSAUTH   | Grant and revoke privileges on a data pipeline function.                                 |

### User Privileges

| Privilege | Description                                                |
|-----------|------------------------------------------------------------|
| VIEW      | Ability to read information about the user.                |
| ALTER     | Ability to modify the user.                                |
| DROP      | Ability to drop the user.                                  |
| SYSAUTH   | Ability to grant rights and revoke privileges on the user. |

### Group Privileges

| Privilege | Description                                  |
|-----------|----------------------------------------------|
| VIEW      | Ability to read information about the group. |
| ALTER     | Ability to alter the group.                  |
| DROP      | Ability to drop the group.                   |
| SYSAUTH   | Ability to grant privileges on the group.    |

### The System Catalog and Object or View Visibility

The `sys.privileges` table in the system catalog exposes the privileges in the system. This table displays this information:

- Timestamp of the grant
- Grantor
- Grantee
- Privilege granted
- Object type
- Object id
- Grantable

You can grant and revoke System Catalog **VIEW** and **SELECT** privileges just like ordinary tables. By default, everyone has **VIEW** and **SELECT** privileges on all system catalog tables.

You can see the objects within the system catalog tables only if you have sufficient privileges on those objects. These objects require a **VIEW** or **SELECT** privilege or membership in a group or role to provide visibility. When you execute queries against a system catalog table, you do not see nor know the existence of the objects to which you do not have access.

Views offer a similar functionality because the database does not check privileges to the underlying tables and views after you create a view. You can create a table with sensitive information and restrict visibility by creating a view on top of the table with only certain rows or columns. Someone else with the **SELECT** privilege to the view can query the view, even without any privileges to the underlying table.

## Roles

Similar to groups, users can inherit privileges by being granted one of the predefined roles in Ocident. The names of and privileges assigned to roles in Ocident are predefined by the system. Roles can be applicable to the Ocident System or one of the user-defined databases. The roles in Ocident are as follows:

### System Roles

- **Security Administrator** — Can read, create, and modify all users and groups.
- **System Administrator** — Can read, create, and modify system objects such as tables, clusters, databases, etc. Can see and delete any queries system-wide.
- **System Analyst** — Read-only access to the entire system. Can see all queries in the system.

### Database Roles

- **Database Administrator** — Can read, create, and modify database objects such as tables, clusters, databases, etc. This role can also create users for the database. Can see and delete all queries in the database.
- **Database Analyst** — Read-only rights to database objects and data within the database. Can see all queries in the database.

For specifics on each Ocident role, see [Data Control Language \(DCL\) Command Reference](#).

## Grant Role Membership

Grants a role to a user or group.

### Syntax

```
GRANT ROLE role_name TO { USER | GROUP } user_or_group_name
```

| Parameter          | DataType | Description                                                                                     |
|--------------------|----------|-------------------------------------------------------------------------------------------------|
| role_name          | string   | The name of a role to be granted to the specified user or group. Enclose this string in quotes. |
| user_or_group_name | string   | The name of a user or group to be granted the specified role.                                   |

### Example

This example grants `user1` the system administrator role.

```
GRANT ROLE "System Administrator" TO USER user1;
```

## Revoke Role Membership

Revokes a role from a user or group.

### Syntax

```
REVOKE ROLE role_name FROM { USER | GROUP } user_or_group_name
```

| Parameter          | Data Type | Description                                                                                      |
|--------------------|-----------|--------------------------------------------------------------------------------------------------|
| role_name          | string    | The name of a role to be revoked for the specified user or group. Enclose this string in quotes. |
| user_or_group_name | string    | The name of a user or group to have the specified role revoked.                                  |

### Example

This example revokes the system administrator role from `user1`.

```
REVOKE ROLE "System Administrator" FROM USER user1;
```

### Default Role Privileges

#### Security Administrator Privileges

| Target                 | Privileges                      |
|------------------------|---------------------------------|
| System                 | SYSAUTH, CREATE USER            |
| Database               | SYSAUTH, CREATE [ USER, GROUP ] |
| View                   | SYSAUTH                         |
| Table                  | SYSAUTH                         |
| Machine Learning Model | SYSAUTH                         |
| Data Pipeline          | SYSAUTH                         |
| Data Pipeline Function | SYSAUTH                         |
| User                   | VIEW, ALTER, DROP, SYSAUTH      |
| Group                  | VIEW, ALTER, DROP, SYSAUTH      |

#### System Administrator Privileges

| Target                 | Privileges                                                                            |
|------------------------|---------------------------------------------------------------------------------------|
| System                 | USE, CREATE [ DATABASE, USER ]                                                        |
| Database               | USEALTERDROPCREATE [ TABLE, VIEW, MLMODEL, USER, GROUP, PIPELINE, PIPELINE FUNCTION ] |
| View                   | VIEW, SELECT, ALTER, DROP                                                             |
| Table                  | VIEW, SELECT, ALTER, LOAD, DROP, DELETE, INSERT                                       |
| Machine Learning Model | SELECT, DROP                                                                          |
| Data Pipeline          | VIEW, EXECUTE, ALTER, DROP                                                            |
| Data Pipeline Function | VIEW, DROP                                                                            |
| User                   | VIEW, ALTER, DROP                                                                     |
| Group                  | VIEW, ALTER, DROP                                                                     |

#### System Analyst Privileges

| Target   | Privileges   |
|----------|--------------|
| System   | USE          |
| Database | USE          |
| View     | VIEW, SELECT |

| Target                 | Privileges   |
|------------------------|--------------|
| Table                  | VIEW, SELECT |
| Machine Learning Model | SELECT       |
| Data Pipeline          | VIEW         |
| Data Pipeline Function | VIEW         |
| User                   | VIEW         |
| Group                  | VIEW         |

### Database Administrator Privileges

| Target                 | Privileges                                                                                   |
|------------------------|----------------------------------------------------------------------------------------------|
| Database               | USEALTERDROPCREATE [ TABLE, VIEW, MLMODEL, USER, GROUP, PIPELINE, PIPELINE FUNCTION ]SYSAUTH |
| View                   | VIEW, SELECT, ALTER, DROP, SYSAUTH                                                           |
| Table                  | VIEW, SELECT, ALTER, LOAD, DROP, DELETE, INSERT, SYSAUTH                                     |
| Machine Learning Model | SELECT, DROP, SYSAUTH                                                                        |
| Data Pipeline          | VIEW, EXECUTE, ALTER, DROP, SYSAUTH                                                          |
| Data Pipeline Function | VIEW, DROP, SYSAUTH                                                                          |
| User                   | VIEW, ALTER, DROP, SYSAUTH                                                                   |
| Group                  | VIEW, ALTER, DROP, SYSAUTH                                                                   |

### Database Analyst Privileges

| Target                 | Privileges                         |
|------------------------|------------------------------------|
| Database               | USE                                |
| View                   | VIEW, SELECT                       |
| Table                  | VIEW, SELECT, LOAD, DELETE, INSERT |
| Machine Learning Model | SELECT                             |
| Data Pipeline          | VIEW                               |
| Data Pipeline Function | VIEW                               |
| User                   | VIEW                               |
| Group                  | VIEW                               |

### Query Visibility

The visibility of queries depends on which database you are logged into and your system role.

#### Visibility by Role

Users with no additional role privileges can view only queries that they submit themselves.

Users with granted roles have extra visibility to see queries from any user, either for the entire system or individual databases. This table summarizes the roles that have query privileges.

| Role                          | Query Visibility                                |
|-------------------------------|-------------------------------------------------|
| <b>System Administrator</b>   | Can see and delete any queries system-wide.     |
| <b>System Analyst</b>         | Can see all queries in the system.              |
| <b>Database Administrator</b> | Can see and delete all queries in the database. |
| <b>Database Analyst</b>       | Can see all queries in the database.            |

For an overview of other privileges by role, see [Data Control Language \(DCL\) Command Reference](#).

## Login Impacts on Query Visibility

Along with having role privileges, users also must be logged into the appropriate database to view queries:

- If the user logs in to the **SYSTEM** database, the user can see all queries from all databases.
- If the user logs in to a database other than **SYSTEM**, the user can see queries in only that database.

## Related Links

[Database Administration](#)

[Manage Users, Groups, and Roles](#)

[Object-Type Level Privileges Management](#)

## 10.23 Errors and Warnings

The Ocient® System produces these error and warning codes as part of its operation. Refer to these codes for their description.

You can also execute this SQL statement using the `sys.sql_messages` system catalog table to view the codes and descriptions in the database: `SELECT * FROM sys.sql_messages.`

## Error Codes

The error codes are negative values that split into ranges by number such as -100s, -200s, and so on.

| -1xx Codes        | Description                                                                                                                |
|-------------------|----------------------------------------------------------------------------------------------------------------------------|
| -100              | Unknown error occurred.                                                                                                    |
| -101              | This is an internal error that indicates that a client (driver or connector) received an invalid response from the server. |
| -102              | Internal error that indicates a corruption of the serialized result set sent from the server to the client.                |
| -103              | Internal error that indicates an unexpected condition has happened.                                                        |
| <b>-2xx Codes</b> | <b>Description</b>                                                                                                         |
| -200              | A client attempted a connection with a malformed connection string.                                                        |
| -201              | The connection to the server failed.                                                                                       |
| -202              | The handshake with the server failed.                                                                                      |
| -203              | The connection to the server was unexpectedly closed.                                                                      |
| -204              | A network communications error has occurred.                                                                               |
| -205              | The client tried to connect to a database with a specified name, but no database with that name exists.                    |
| -206              | The target node is offline.                                                                                                |
| <b>-3xx Codes</b> | <b>Description</b>                                                                                                         |
| -300              | An argument to a driver method execution was invalid.                                                                      |
| -301              | A driver method executed, but the input argument object was already closed.                                                |
| -302              | The cursor was not positioned in a valid row.                                                                              |
| -303              | An attempt was made to create a new result set, but the previous result set for the connection was still open.             |
| -304              | The user and password combination was incorrect.                                                                           |
| -305              | A statement that was not a query was passed to <code>executeQuery()</code> .                                               |
| -306              | A statement that was a query was passed to <code>executeUpdate()</code> .                                                  |

| -1xx Codes | Description                                                                                                                                                                                                        |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -307       | A NULL output buffer was passed to a driver method, where a NULL buffer is not allowed.                                                                                                                            |
| -308       | A driver execution was made that is not implemented.                                                                                                                                                               |
| -313       | An invalid parameter marker number was specified.                                                                                                                                                                  |
| -316       | No database connection exists.                                                                                                                                                                                     |
| -317       | Invalid data was detected for a parameter marker.                                                                                                                                                                  |
| -318       | The cursor was not in the required state.                                                                                                                                                                          |
| -319       | An invalid buffer length was specified.                                                                                                                                                                            |
| -320       | There is currently no result set.                                                                                                                                                                                  |
| -321       | A NULL indicator pointer was required but not specified.                                                                                                                                                           |
| -322       | A connection attribute that is not supported was specified in SQLBrowseConnect. The four supported attributes are: "DRIVER", "DSN", "UID", "PWD".                                                                  |
| -323       | An invalid descriptor record index was specified. Valid indices start at 1 and increase to the size specified by executing SQLSetDescField with the SQL_DESC_COUNT field.                                          |
| -324       | An invalid or inconsistent descriptor field type was specified in SQLSetDescRec.                                                                                                                                   |
| -325       | An invalid descriptor field was specified in SQLSetDescField.                                                                                                                                                      |
| -326       | The target handle passed to SQLCopyDesc was an IRD.                                                                                                                                                                |
| -327       | An implicit descriptor handle that was not the original was the specified value in the execution of SQLSetStmtAttr with the SQL_ATTR_APP_PARAM_DESC or SQL_ATTR_APP_ROW_DESC attribute.                            |
| -328       | Divide by zero.                                                                                                                                                                                                    |
| -329       | Invalid floating point operation that resulted in NaN or Inf.                                                                                                                                                      |
| -330       | The action conflicts with the current state of the database.                                                                                                                                                       |
| -331       | The unconditional DELETE FROM statement is rewritten using the TRUNCATE statement. In this case, the database does not create a plan that would appear in the results from the execution of the EXPLAIN statement. |
| -332       | Already in use. Used mostly in DDL when a resource or option is already assigned to the object.                                                                                                                    |
| -333       | Optional value was substituted.                                                                                                                                                                                    |
| -334       | This driver is unknown or the version support has been removed.                                                                                                                                                    |
| -335       | The execution of a client method was invalid.                                                                                                                                                                      |
| -4xx Codes | Description                                                                                                                                                                                                        |
| -400       | One of the referenced columns does not exist.                                                                                                                                                                      |
| -401       | The requested data type conversion was invalid.                                                                                                                                                                    |
| -402       | Unknown data type.                                                                                                                                                                                                 |
| -403       | A column name was ambiguous in the context where it was used.                                                                                                                                                      |
| -404       | Two incompatible data types were specified as return types in a case statement.                                                                                                                                    |
| -405       | Join leads to a duplicate column name.                                                                                                                                                                             |
| -406       | Invalid POINT conversion from NULL.                                                                                                                                                                                |



| -1xx Codes | Description                                                                                                                       |
|------------|-----------------------------------------------------------------------------------------------------------------------------------|
| -407       | Column value is larger than internal limits.                                                                                      |
| -5xx Codes | <b>Description</b>                                                                                                                |
| -500       | Generic syntax error.                                                                                                             |
| -501       | The SQL statement is not valid or not supported in the context where it was used.                                                 |
| -502       | A duplicate common table expression name was detected.                                                                            |
| -503       | The number of columns does not match the SELECT statement.                                                                        |
| -504       | A negative limit was specified.                                                                                                   |
| -505       | A negative offset was specified.                                                                                                  |
| -506       | An incorrect number of arguments was specified in a function.                                                                     |
| -507       | Aggregation of a constant value is invalid.                                                                                       |
| -508       | An argument to a function had an incorrect data type.                                                                             |
| -509       | An invalid comparison was detected.                                                                                               |
| -510       | Aggregation was used in a context where it is not allowed (possibly something in a WHERE clause that belongs in a HAVING clause). |
| -511       | A HAVING clause did not have any aggregation in it.                                                                               |
| -512       | An ORDER BY clause referenced a column by position and that position was out of range.                                            |
| -513       | An expression of type list cannot be returned in a result set.                                                                    |
| -514       | You cannot have an expression of type list within another list.                                                                   |
| -515       | A list expression contains incompatible types.                                                                                    |
| -516       | The function does not exist.                                                                                                      |
| -517       | If the result of an expression is a time interval type, the result must be cast to an integral type.                              |
| -518       | An invalid matrix literal was specified.                                                                                          |
| -519       | A matrix literal had incorrect dimensions.                                                                                        |
| -520       | An invalid query priority was specified.                                                                                          |
| -521       | A column reference in a SELECT statement is invalid due to aggregation.                                                           |
| -522       | Columns for the first and second SELECT statements for the UNION, EXCEPT, or INTERSECT statement are not compatible.              |
| -523       | An expression in the CASE condition is not allowed.                                                                               |
| -524       | An expression in the JOIN condition is not allowed.                                                                               |
| -525       | Aggregation is not allowed in the PARTITION BY or ORDER BY clauses of a windowed aggregate.                                       |
| -526       | A non-constant value was found where a constant was expected in an argument of a windowed aggregate.                              |
| -527       | Range-based framing does not allow the PRECEDING or FOLLOWING keywords.                                                           |
| -528       | ORDER BY with a column ordinal is not allowed in a windowed aggregate.                                                            |
| -529       | A frame specification is not allowed if no ORDER BY clause is specified in a windowed aggregate.                                  |
| -530       | UNBOUNDED PRECEDING is only allowed on the starting frame bound.                                                                  |
| -531       | UNBOUNDED FOLLOWING is only allowed on the ending frame bound.                                                                    |

| <b>-1xx Codes</b> | <b>Description</b>                                                                                                  |
|-------------------|---------------------------------------------------------------------------------------------------------------------|
| -532              | The ending frame bound cannot be less than the starting frame bound.                                                |
| -533              | No frame specification is allowed for this particular windowed aggregate.                                           |
| -534              | A window function requires a positive integer argument.                                                             |
| -535              | The specified window function requires an ORDER BY statement in the OVER() clause.                                  |
| -536              | Invalid aggregation.                                                                                                |
| -537              | Invalid expression.                                                                                                 |
| -538              | A CASE expression within a CASE expression is not allowed.                                                          |
| -539              | Window function was used in a context where it is not allowed (perhaps in a predicate of a WHERE or HAVING clause). |
| <b>-6xx Codes</b> | <b>Description</b>                                                                                                  |
| -600              | A referenced table does not exist.                                                                                  |
| -601              | The specified database already exists.                                                                              |
| -602              | The specified table already exists.                                                                                 |
| -603              | The specified view already exists.                                                                                  |
| -604              | The specified view was not found.                                                                                   |
| -605              | The specified database was not found.                                                                               |
| -606              | The specified storage space was not found.                                                                          |
| -607              | The specified storage space already exists.                                                                         |
| -608              | The specified user does not exist.                                                                                  |
| -609              | The specified user already exists.                                                                                  |
| -610              | The specified password for the new user is invalid.                                                                 |
| -611              | The specified group name already exists.                                                                            |
| -612              | The specified group does not exist.                                                                                 |
| -613              | The specified connection already exists.                                                                            |
| -614              | The specified connection was not found.                                                                             |
| -615              | The referenced translation does not exist.                                                                          |
| -616              | A translation with that name exists.                                                                                |
| -617              | The specified user was not found in this group.                                                                     |
| -618              | The provided storage space has no storage cluster assigned.                                                         |
| -619              | The referenced table is a system table and not allowed for the export operation.                                    |
| -620              | The referenced pipeline does not exist.                                                                             |
| -621              | A pipeline with that name exists.                                                                                   |
| -622              | The pipeline is invalid.                                                                                            |
| -623              | The pipeline is unable to be correctly compiled.                                                                    |
| -624              | The referenced task does not exist.                                                                                 |
| -625              | An index with that type and on that column already exists.                                                          |
| -626              | An index with that name already exists.                                                                             |
| -627              | The referenced table ID no longer exists.                                                                           |
| -628              | A connectivity pool with that name already exists.                                                                  |
| -629              | Cannot find a connectivity pool with the specified name.                                                            |
| -630              | Cannot drop the last connectivity pool from a node.                                                                 |
| <b>-7xx Codes</b> | <b>Description</b>                                                                                                  |
| -700              | The user does not have read authorization on a referenced table.                                                    |
| -701              | The user does not have authorization to create a database.                                                          |

| <b>-1xx Codes</b> | <b>Description</b>                                                       |
|-------------------|--------------------------------------------------------------------------|
| -702              | The user does not have modify authorization on a referenced table.       |
| -703              | The user does not have authorization to create a view.                   |
| -704              | The user does not have modify authorization on a referenced database.    |
| -705              | The user does not have modify authorization on a referenced view.        |
| -706              | The user does not have authorization to drop a referenced table.         |
| -707              | The user does not have authorization to drop a referenced row.           |
| -708              | The user does not have authorization to drop a referenced view.          |
| -709              | The user does not have authorization to drop a referenced database.      |
| -710              | The user does not have authorization to create a table.                  |
| -711              | The user does not have authorization to create a storage space.          |
| -712              | The user does not have authorization to drop a referenced storage space. |
| -713              | The user does not have authorization to create a user.                   |
| -714              | The user does not have authorization to drop a user.                     |
| -715              | The user does not have the authority to create a group.                  |
| -716              | The user does not have the authority to drop the group.                  |
| -717              | The user does not have the authority to create a connection.             |
| -718              | The user does not have the authority to drop the connection.             |
| -719              | The user is not authorized to take this action.                          |
| -720              | The user does not have the authority to modify the group.                |
| -721              | The user does not have the authority to execute a plan.                  |
| -722              | The user does not have the authority to execute an inline plan.          |
| -723              | The user does not have the authority to create a connectivity pool.      |
| -724              | The user does not have the authority to drop the connectivity pool.      |
| -725              | The user does not have the authority to modify the connectivity pool.    |
| -726              | Unable to drop some objects in the request.                              |
| -733              | Session has expired. Refresh is necessary.                               |
| <b>-8xx Codes</b> | <b>Description</b>                                                       |
| -800              | An MLMODEL with that name was not found.                                 |
| -801              | An MLMODEL with that name already exists.                                |
| -802              | The user does not have the authority to create an MLMODEL.               |
| -803              | The user does not have the authority to drop the MLMODEL.                |
| -804              | Unable to create an MLMODEL due to a singular matrix.                    |
| -805              | Unable to create an MLMODEL because the training set is empty.           |

| <b>-1xx Codes</b>  | <b>Description</b>                                                                                                  |
|--------------------|---------------------------------------------------------------------------------------------------------------------|
| -806               | Unable to perform LUP decomposition on this singular matrix.                                                        |
| -807               | The user does not have the authority to modify the MLMODEL.                                                         |
| <b>-9xx Codes</b>  | <b>Description</b>                                                                                                  |
| -900               | Operation was canceled or aborted.                                                                                  |
| -901               | I/O error.                                                                                                          |
| -902               | Scalar Subquery Cardinality Violation.                                                                              |
| -903               | Numeric value out of range.                                                                                         |
| -904               | Plan compilation error.                                                                                             |
| -905               | TKT limit reached.                                                                                                  |
| -906               | Out of memory.                                                                                                      |
| -907               | Segment not available.                                                                                              |
| -908               | System is still initializing.                                                                                       |
| -909               | Out of temporary disk space.                                                                                        |
| -910               | Operation was automatically killed due to drop in the client connection.                                            |
| -911               | Failed to acquire OSN (Ownership Sequence Number). Data for querying is no longer available to the storage cluster. |
| <b>-10xx Codes</b> | <b>Description</b>                                                                                                  |
| -1000              | The privilege target already has the privilege.                                                                     |
| -1001              | The user does not have the authority to grant this privilege.                                                       |
| -1002              | The privilege target does not have that privilege.                                                                  |
| -1003              | The referenced role does not exist.                                                                                 |
| -1004              | The provided object ID was not found.                                                                               |
| -1005              | Create task failure.                                                                                                |
| -1006              | Task already in progress.                                                                                           |
| -1007              | Could not find the supplied service class.                                                                          |
| -1008              | Supplied service class definition is invalid.                                                                       |
| <b>-11xx Codes</b> | <b>Description</b>                                                                                                  |
| -1100              | No available service classes for the user have available query slots.                                               |
| -1101              | The specified node is not found.                                                                                    |
| -1102              | Segment not found.                                                                                                  |
| -1111              | Cluster not found.                                                                                                  |
| -1112              | A cluster with that name exists.                                                                                    |
| -1120              | Query exceeded service class time limit.                                                                            |
| -1121              | Query exceeded session time limit.                                                                                  |
| -1122              | Query exceeded query time limit.                                                                                    |
| -1123              | Query exceeded service class row limit.                                                                             |
| -1124              | Query exceeded session row limit.                                                                                   |
| -1125              | Query exceeded query row limit.                                                                                     |
| -1126              | Query exceeded service class temporary disk limit.                                                                  |
| -1127              | Query exceeded session temporary disk limit.                                                                        |
| -1128              | Query exceeded query temporary disk limit.                                                                          |
| -1129              | Too many columns in result set for the service class.                                                               |
| -1130              | Too many columns in result set for the session.                                                                     |
| -1131              | Too many columns in result set for the query.                                                                       |
| <b>-12xx Codes</b> | <b>Description</b>                                                                                                  |
| -1200              | The security integration is disabled.                                                                               |
| -1201              | The security integration does not exist.                                                                            |

| -1xx Codes | Description                                                                                                                                       |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| -1202      | The request failed due to a network communications error.                                                                                         |
| -1203      | The Single Sign On request identifier could not be found.                                                                                         |
| -1204      | No session associated with the connection exists.                                                                                                 |
| -1205      | The supplied security token is invalid.                                                                                                           |
| -1206      | The supplied security token signature does not match the expected value.                                                                          |
| -1207      | The supplied security token has a signature fingerprint from an unknown entity.                                                                   |
| -1208      | The supplied security token has expired.                                                                                                          |
| -1220      | A network communications error occurred.                                                                                                          |
| -1221      | The OpenID provider rejected the request that indicates an internal server error.                                                                 |
| -1222      | The OpenID provider returned an invalid response that indicates an error with the provider.                                                       |
| -1223      | The request_uri in the Authorization Request returns an error or contains invalid data.                                                           |
| -1224      | The OpenID provider does not support use of the request parameter.                                                                                |
| -1225      | The OpenID provider does not support use of the request_uri parameter.                                                                            |
| -1226      | The OpenID provider discovery document is invalid.                                                                                                |
| -1227      | The database OpenID configuration is invalid.                                                                                                     |
| -1228      | The OpenID ID or access token has expired.                                                                                                        |
| -1229      | The ID or access token was signed by a key not known to the OpenID provider.                                                                      |
| -1230      | The signing key returned by the OpenID provider is invalid.                                                                                       |
| -1231      | The token contains an invalid claim.                                                                                                              |
| -1232      | The token was issued by an unknown OpenID provider.                                                                                               |
| -1233      | The token was issued for an alternate audience.                                                                                                   |
| -1234      | The token signature is invalid, which implies the token was manipulated enroute to the database.                                                  |
| -1235      | The specified ID token is invalid.                                                                                                                |
| -1236      | The specified Access token is invalid.                                                                                                            |
| -1237      | The refresh token returned from the provider is invalid.                                                                                          |
| -1250      | The client is not authorized to request an authorization code using this method.                                                                  |
| -1251      | The resource owner or authorization server denied the request.                                                                                    |
| -1252      | The authorization server does not support obtaining an authorization code using this method.                                                      |
| -1253      | The requested scope is invalid, unknown, or malformed.                                                                                            |
| -1254      | The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed. |
| -1255      | The authorization server encountered an unexpected condition that prevented it from fulfilling the request.                                       |
| -1256      | The authorization server is currently unable to handle the request due to a temporary overloading or maintenance of the server.                   |
| -1257      | Cannot lock table.                                                                                                                                |
| -1258      | Cannot unlock table.                                                                                                                              |
| -1259      | Cannot delete from a join or a view with multiple tables.                                                                                         |

| -1xx Codes   | Description          |
|--------------|----------------------|
| <b>Other</b> | <b>Description</b>   |
| -9999        | Undefined exception. |

### Warning Codes

The warning codes are positive values.

| Warning Code | Description                                                            |
|--------------|------------------------------------------------------------------------|
| +300         | Data received by a JDBC driver was truncated.                          |
| +301         | The driver is below the recommended threshold for the warning version. |
| +900         | A query returned an empty result set.                                  |
| +901         | The query is estimated to have a long execution time.                  |
| +1113        | The referenced object does not exist.                                  |
| +1114        | An object with that name exists.                                       |
| +1115        | Not authorized to take this action.                                    |
| +1116        | Restart the node for changes to take effect.                           |
| +1117        | Complete system restart required for changes to take effect.           |
| +1118        | Statement contains invalid syntax.                                     |
| +1119        | Unable to drop some objects in the request.                            |

### Related Links

[SQL Reference](#)

[Data Definition Language \(DDL\) Statement Reference](#)

[Data Integrity and Storage](#)

[Query Analysis](#)

[System Catalog](#)

#### 10.23.1 Data Pipeline Loading Errors

This table captures error codes and their corresponding error messages that you can encounter when you load data using data pipelines. Also, the table contains the definitions of the parameters in the error messages, descriptions that explain the error messages, and actions you can take to resolve the errors.

| Code  | Error Message                                                                                                    | Message Parameters                                                                                                                                                                     | Description                                                                                           | Resolution                                                                 |
|-------|------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------|
| OLD00 | Key <code>key</code> for class <code>class</code> must be set to a non-NULL value. Details: <code>details</code> | <code>key</code> : The key of the missing property.<br><code>class</code> : The class that has the missing property.<br><code>details</code> : Additional information about the error. | This error occurs when the loading process reads the pipeline definition missing a required property. | Specify a non-NULL value for the specified key in the pipeline definition. |

| Code  | Error Message                                                                                                                               | Message Parameters                                                                                                                                                                                                                                                                                                                | Description                                                                                                           | Resolution                                                                                                                     |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| OLD01 | Value ' <i>value</i> ' is not acceptable with the key <i>key</i> for class <i>class</i> . Expected <i>example</i> . Details: <i>details</i> | <i>value</i> : The value that you set for the specified pipeline definition key. <i>key</i> : The pipeline definition key with an incorrect value. <i>class</i> : The class that contains the property. <i>example</i> : An example or description of the correct value. <i>details</i> : Additional information about the error. | This error occurs when the loading process reads the pipeline definition that contains a key with an incorrect value. | Specify the correct value for the setting corresponding to the specified key in the pipeline definition.                       |
| OLD02 | No class with name ' <i>class</i> ' exists in the pipeline builder packages. Pipeline cannot run.                                           | <i>class</i> : The class that cannot be found.                                                                                                                                                                                                                                                                                    | This error occurs when the loading process reads the pipeline definition that contains an unknown class.              | Ensure that the class exists and belongs to a package specified in the ' <code>engine.builder.packages</code> ' option.        |
| OLE00 | Failed to extract record with index <i>index</i> from ' <i>sourceName</i> '. Details: <i>details</i>                                        | <i>index</i> : The index of the record (index starts at 1) in the source where the error occurred. <i>sourceName</i> : The name of the source resource where the error occurred. <i>details</i> : Additional information about the error.                                                                                         | This error occurs when the extraction of a record fails.                                                              | Provide the correct source data in the accurate format, extract configuration, or compression method.                          |
| OLE01 | <i>dataExtract</i> cannot be extracted from ' <i>sourceData</i> '. Details: <i>details</i>                                                  | <i>dataExtract</i> : The data to extract. <i>sourceData</i> : The source data for the extraction. <i>details</i> : Additional information about the error                                                                                                                                                                         | This error occurs when the extraction of data from a record fails.                                                    | Provide the correct source data in the accurate format, extract configuration, compression method, or transform configuration. |

| Code  | Error Message                                                                                                                                       | Message Parameters                                                                                                                                                                                                                                                           | Description                                                                                                                                                    | Resolution                                                                                                                                                                                                                                                                                                                                                        |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OLO00 | Failed to connect to <i>sourceType</i> source because the specified <i>locationType</i> 'location' was not found. Details: <i>details</i>           | <i>sourceType</i> : The type of the source.<br><i>locationType</i> : The type of the data location argument that was not found.<br><i>location</i> : The value of the data location argument that was not found.<br><i>details</i> : Additional information about the error. | This error occurs when the pipeline cannot find the source data (e.g., cannot find the configured bucket or file path).                                        | In the pipeline definition, specify an existing data location in the source.                                                                                                                                                                                                                                                                                      |
| OLO01 | Failed to connect to <i>sourceType</i> source because the access credentials are unauthorized. Details: <i>details</i>                              | <i>sourceType</i> : The type of source where access is denied.<br><i>details</i> : Additional information about the error.                                                                                                                                                   | This error occurs when the pipeline cannot connect to the source due to invalid credentials.                                                                   | Specify credentials in the pipeline definition with access to the specified source or configure the source to provide access to the pipeline.                                                                                                                                                                                                                     |
| OLO10 | Failed to connect to <i>sourceType</i> source because the file path is empty. The locations searched are: <i>location</i> . Details: <i>details</i> | <i>sourceType</i> : The type of the source.<br><i>location</i> : The value of the data location argument that was not found.                                                                                                                                                 | This error occurs when the pipeline cannot find the source data (e.g., the file path is empty).                                                                | If an empty file list was expected, set the <code>expect.empty.file.list</code> configuration option to <code>true</code> by executing <code>ALTER SYSTEM ALTER CONFIG SET 'streamloader.extractorEngine = 'true''</code> ; . Then, restart the pipeline. Handle the missing file using your own process or configure the pipeline to skip missing files instead. |
| OLO50 | File <i>fileName</i> was not found.                                                                                                                 | <i>fileName</i> : The name of the missing file.                                                                                                                                                                                                                              | This error occurs when the pipeline is configured to fail on missing files, and it cannot find a file in the list.                                             |                                                                                                                                                                                                                                                                                                                                                                   |
| OLS00 | Failed to load transformed value 'value' to sink after <i>transformClassName</i> . Details: <i>details</i>                                          | <i>value</i> : The value that the Loader Node could not load to the sink.<br><i>transformClassName</i> : The class of the executed transform.<br><i>details</i> : Additional information about the error.                                                                    | This error occurs when the Loader Node cannot load a transformed value to the sink. For example, the transformed value might be too large for the sink column. | Check that the sink column is configured correctly or check that the transforms in the column are composed properly to return values that can load to the sink.                                                                                                                                                                                                   |



| Code  | Error Message                                                                                                                                             | Message Parameters                                                                                                                                                                                                                                                                                                                                 | Description                                                                                                                                                                                                                                                                                                                  | Resolution                                                                                                                                    |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| OLS97 | No table with name ‘ <i>tableName</i> ’ exists in sink<br>at <i>ipAddress:port</i> .<br>The available tables are: <i>existingTables</i> .                 | <i>tableName</i> : The fully specified name of the specified table.<br><i>ipAddress</i> : The IP address for the Loader Node of the {{Ocient}} sink.<br><i>port</i> : The external TCP port for the Loader Node of the Ocient sink.<br><i>existingTables</i> : The list of existing tables in the same database and schema as the specified table. | This error occurs when no table with the specified name exists in the sink, but the database and schema of the table do exist.                                                                                                                                                                                               | Specify an existing table in the pipeline definition or create a table with the specified name.                                               |
| OLS98 | The column name ‘ <i>columnName</i> ’ does not exist. The columns available to the pipeline from the <i>tableName</i> table are: <i>existingColumns</i> . | <i>columnName</i> : The name of the column that was not found.<br><i>tableName</i> : The table that was searched.<br><i>existingColumns</i> : The columns in the table that are available to the pipeline.                                                                                                                                         | This error occurs when the target table in the sink does not contain all the columns specified in the pipeline or a column specified as a transform target is not declared in the sink definition of the pipeline. (The database ignores deleted columns. This error does not occur when you load data to a deleted column.) | Specify an existing column in the pipeline definition or add the corresponding column to the target table.                                    |
| OLS99 | Failed to connect to sink<br>at <i>ipAddress:port</i> .<br>Check that a streamloader at that address is reachable.                                        | <i>ipAddress</i> : The IP address for the Loader Node of the Ocient sink.<br><i>port</i> : The external TCP port for the Loader Node of the Ocient sink.                                                                                                                                                                                           | This error occurs when the pipeline cannot connect to its Ocient sink using a Loader Node.                                                                                                                                                                                                                                   | Check that the Loader Node at the specified host and port is reachable and that the host and port of the Ocient sink are configured properly. |

| Code  | Error Message                                                                                                                                          | Message Parameters                                                                                                                                                                                                                                                                                                | Description                                                                                 | Resolution                                                                                                                                                                     |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OLT00 | Cannot perform operation<br><i>operationName</i><br>because the inputs are <i>inputType</i> .<br>Expected inputs are <i>expectedType</i> .             | <i>operationName</i> : The name of the operation that cannot be performed because its inputs are the wrong type. (usually the name of a transform class)<br><i>inputType</i> : The type of inputs that the operation received.<br><i>expectedType</i> : The type of inputs that the operation expects to receive. | This error occurs when transforms in a column are not composed correctly.                   | Add, remove, or replace transforms such that each transform receives input of the correct type and that the Loader Node loads data of the correct type into the target column. |
| OLT01 | Failed to transform value ' <i>value</i> ' using <i>transformClassName</i> .<br>Details: <i>details</i>                                                | <i>value</i> : The value that was not transformed.<br><i>transformClassName</i> : The class of the failed transform.<br><i>details</i> : Additional information about the error.                                                                                                                                  | This error occurs when a transform fails.                                                   | Correct the source data or modify the transform to fit the source data. Alternatively, start the pipeline with a higher error limit so the load skips the failing record.      |
| OLT02 | Value ' <i>value</i> ' is not an acceptable <i>parameterName</i> for <i>transformClassName</i> .<br>Expected <i>example</i> .                          | <i>value</i> : The incorrect value for the specified parameter.<br><i>parameterName</i> : The name of the specified parameter.<br><i>transformClassName</i> : The class of the transform that is not configured correctly.<br><i>example</i> : An example or description of the correct value.                    | This error occurs when the parameter of the transform is not configured correctly.          | Correct the parameter of the transform.                                                                                                                                        |
| OLT03 | Failed to transform value ' <i>value</i> ' using <i>transformClassName</i> because the value is <i>valueDescription</i> .<br>Expected <i>example</i> . | <i>value</i> : The value that was not transformed.<br><i>transformClassName</i> : The class of the failed transform.<br><i>valueDescription</i> : A feature or description of the incorrect value for this transform.<br><i>example</i> : An example or description of the correct value.                         | This error occurs when a transform fails because the data is not correct for the transform. | Correct the source data, modify the transform configuration, or use a different transform.                                                                                     |

| Code  | Error Message                                                                                                                                                                                                                          | Message Parameters                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Description                                                                                                    | Resolution                                                                                                                                                                |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OLT04 | Failed to match value ' <i>value</i> ' to format ' <i>format</i> ' in <i>transformClassName</i> because ' <i>valuePart</i> ' at position <i>valuePartPosition</i> does not match pattern ' <i>pattern</i> '. Expected <i>example</i> . | <i>value</i> : The value that was not parsed with the transform format.<br><i>format</i> : The transform format.<br><i>transformClassName</i> : The class of the failed transform.<br><i>valuePart</i> : The part of the value that was unmatched.<br><i>valuePartPosition</i> : The position (index starts at 1) of the part unmatched in the value.<br><i>pattern</i> : The pattern in the transform format that did not match part of the value.<br><i>example</i> : An example or description of the correct value. | This error occurs when a transformation fails because the value does not match the specified transform format. | Correct the source data or modify the transform format to fit the source data.                                                                                            |
| OLTM0 | Failed to transform ' <i>value</i> ' value using <i>transformClassName</i> for details.                                                                                                                                                | <i>value</i> : The value that was not transformed.<br><i>transformClassName</i> : The name of the failed transform.<br><i>details</i> : Additional information about the error.                                                                                                                                                                                                                                                                                                                                         | This error occurs when an EXPLODE transformation fails.                                                        | Correct the source data or modify the transform to fit the source data. Alternatively, start the pipeline with a higher error limit so the load skips the failing record. |
| OL998 | An error occurred in library <i>library:details</i>                                                                                                                                                                                    | <i>library</i> : The name of the library where the error occurred.<br><i>details</i> : Additional information about the error.                                                                                                                                                                                                                                                                                                                                                                                          | This error occurs when the Ocient System detects an error from an internal library.                            | Diagnose and address the error using the details provided by the library.                                                                                                 |
| OL999 | An error occurred in class <i>class:details</i>                                                                                                                                                                                        | <i>class</i> : The class that threw the exception.<br><i>details</i> : Additional information about the error.                                                                                                                                                                                                                                                                                                                                                                                                          | This error occurs when the Ocient System detects a general error.                                              | If the details of the message are not helpful for you to resolve the error, then contact Ocient Support.                                                                  |

## Related Links

[Errors and Warnings](#)

[Monitor Data Pipelines](#)

## 10.24 Reserved Words

Keywords in Ociant® are specific words that the database can interpret either as SQL syntax or unquoted identifiers. You cannot use a subset of them, named reserved words, as unquoted identifiers without introducing parsing ambiguities in the database. As a result, when you use these words as an identifier, you must surround the reserved word with double quotes.

You can find these words in the sys.reserved\_words system catalog table.

### A

ADD | ALL | ALTER | AND | ANTI | ANY | ARRAY | AS

### B

BAD\_DATA\_TARGET | BALANCE | BETWEEN

### C

CACHE | CALL | CANCEL | CASE | CAST | CHECK | CLASS | CLUSTERING | COLUMN | COLVECTOR | COMMENT | COMPRESSION | COUNT | CREATE | CROSS | CURRENT | CURRENT\_DATABASE | CURRENT\_DATE | CURRENT\_SCHEMA | CURRENT\_TIME | CURRENT\_TIMESTAMP | CURRENT\_USER

### D

DATA | DATABASE | DEFAULT | DELETE | DISABLE | DISTINCT | DROP

### E

ELSE | ENABLE | END | EQUALS | ERROR | EXCEPT | EXECUTE | EXISTING | EXISTS | EXPLAIN | EXPORT | EXTRACT

### F

FALSE | FIRST | FIX | FOLLOWING | FOR | FROM | FULL | FUNCTION

### G

GDC | GRANT

### H

HAVING

### I

IGNORE | ILIKE | IN | INDEX | INNER | INSERT | INTERSECT | INTERVAL | INTO | INVALIDATE | IS

### J

JOIN

### K

KILL

### L

LAST | LATERAL | LEFT | LIKE | LOAD | LOADERS

**M**

MATRIX | MLMODEL | MOVE

**N**

NODE | NOT | NULL | NULLS

**O**

OFFSET | ON | OPTION | OPTIONS | ORDER | OUTER | OVER

**P**

PARTITION | PRAGMA | PRECEDING | PROCEDURE

**R**RANGE | REDUNDANCY | REFRESH | REGEX | REMOVE | RENAME | RESET | REVOKE | RIGHT | ROLE  
| ROW | ROWS | ROWVECTOR**S**SAMPLE | SEGMENT | SELECT | SEMI | SERVICE | SET | SHOW | SIMILAR | SOME | SOURCE | SSO  
INTEGRATION | START | STOP | STORAGE SPACE | SUMMARY\_STATS | SYSAUTH**T**

TABLE | TAG | TASK | THEN | TO | TRACE | TRUE | TRUNCATE | TYPE

**U**

UNBOUNDED | UNION | USE | USING

**V**

VIEW

**W**

WHEN | WHERE | WITH

**Z**

ZSTD

**Related Links**[SQL Reference](#)**10.25 System Catalog Reference**

The Ocident® System Catalog comprises a set of virtual tables and views. Together, these present administrators with metadata that describes the state of the data stored in the system, activity on the system, and the configuration of the system. The `information_schema` schema contains views that present information joined from multiple system catalog tables.

- System Catalog — Listing and description of all system catalog tables included in the system.
- Information Schema — Listing and description of all `information_schema` schema views and `SHOW` commands defined by default.

To explore the system catalog tables, use sample queries for various topics as described in [Discover Insights From System Catalog Tables](#).

### 10.25.1 System Catalog

The system catalog exposes read-only virtual tables that contain metadata about the system.

#### Alphabetical List of Tables

- sys.active\_operator\_instances
- sys.addendum\_directories
- sys.association\_rules\_models
- sys.average\_bb\_sizes
- sys.average\_column\_sizes
- sys.channel\_endpoint\_parameters
- sys.clusters
- sys.column\_cardinalities
- sys.column\_distributions
- sys.columns
- sys.columns\_compression\_info
- sys.completed\_operator\_instances
- sys.completed\_queries
- sys.compute\_configurations
- sys.config
- sys.connectivity\_pool\_participants
- sys.connectivity\_pools
- sys.data\_usage\_by\_file\_type\_and\_table
- sys.databases
- sys.decision\_tree\_models
- sys.degraded\_segment\_groups
- sys.degraded\_segment\_groups\_by\_table
- sys.effective\_merge\_policy\_per\_table
- sys.feedforward\_network\_models
- sys.function\_signatures
- sys.functions
- sys.gaussian\_mixture\_models
- sys.global\_map\_table\_info
- sys.group\_roles
- sys.groups
- sys.index\_advisor\_enabled
- sys.index\_advisors
- sys.index\_columns
- sys.index\_recommendations
- sys.index\_usage
- sys.indexes
- sys.k\_means\_models
- sys.k\_nearest\_neighbors\_models
- sys.linear\_combination\_regression\_models
- sys.linear\_discriminant\_analysis\_models
- sys.load\_errors
- sys.load\_events
- sys.locations
- sys.locks
- sys.logistic\_regression\_models
- sys.lts\_cluster\_info
- sys.machine\_learning\_model\_options

- sys.machine\_learning\_models
- sys.metric\_levels
- sys.multiple\_linear\_regression\_models
- sys.naive\_bayes\_models
- sys.network\_interface\_models
- sys.network\_interface\_usage\_types
- sys.node\_clusters
- sys.node\_config
- sys.node\_network\_interfaces
- sys.node\_status
- sys.nodes
- sys.nodes\_with\_unhealthy\_stored\_segments
- sys.nonlinear\_regression\_models
- sys.oidc\_integrations
- sys.oidc\_sessions
- sys.op\_inst\_debug\_info
- sys.orphaned\_segments
- sys.osn\_acquisitions
- sys.pipeline\_errors
- sys.pipeline\_events
- sys.pipeline\_files
- sys.pipeline\_functions
- sys.pipeline\_metrics
- sys.pipeline\_metrics\_info
- sys.pipeline\_partitions
- sys.pipeline\_tables
- sys.pipeline\_tasks
- sys.pipelines
- sys.plans
- sys.polynomial\_regression\_models
- sys.principal\_component\_analysis\_models
- sys.privileges
- sys.procedures
- sys.queries
- sys.random\_forest\_models
- sys.reserved\_words
- sys.result\_cache
- sys.rights
- sys.roles
- sys.scheduled\_tasks
- sys.security\_integrations
- sys.security\_settings
- sys.segment\_directories
- sys.segment\_group\_transfers
- sys.segment\_groups
- sys.segment\_groups\_merge\_eligible\_per\_table
- sys.segment\_part\_inventory
- sys.segment\_part\_redundancy\_info
- sys.segment\_parts
- sys.segments
- sys.segments\_compression\_info
- sys.service\_classes
- sys.service\_classes\_for\_user
- sys.service\_role\_channel\_endpoints
- sys.service\_role\_status

- sys.service\_roles
- sys.sessions
- sys.simple\_linear\_regression\_models
- sys.sql\_messages
- sys.stats\_files
- sys.storage\_capacity
- sys.storage\_device\_files
- sys.storage\_device\_metrics
- sys.storage\_device\_status
- sys.storage\_scopes
- sys.storage\_spaces
- sys.storage\_used
- sys.stored\_leaf\_segment\_part\_inventory
- sys.stored\_segments
- sys.subtasks
- sys.support\_vector\_machine\_models
- sys.system\_information
- sys.system\_table\_columns
- sys.system\_tables
- sys.table\_cardinalities
- sys.tables
- sys.tasks
- sys.tkt\_table\_clustering\_column\_indices
- sys.tkt\_table\_info
- sys.user\_groups
- sys.user\_mappings
- sys.user\_roles
- sys.users
- sys.vector\_autoregression\_models
- sys.view\_columns
- sys.views
- sys.vl\_columns\_compression\_info

### Alphabetical List of Categories

- Configuration
- Connectivity Pools
- Databases
- Loading
- Machine Learning
- Monitoring
- Network
- Pipelines
- Security
- Security Integrations
- Statistics
- Storage
- System
- System Information
- User Management
- Workload Management

### Catalog Table Details

Configuration

sys.config



This table contains all configuration overrides applied on the system.

| Column Name | Column Type | Column Description                                                                     |
|-------------|-------------|----------------------------------------------------------------------------------------|
| scope_type  | CHAR        | Type of the scope.                                                                     |
| scope_id    | UUID        | Universally Unique IDentifier (UUID) of the storage scope where this override applies. |
| key         | CHAR        | String that represents the configuration parameter.                                    |
| value       | CHAR        | Value for the configuration parameter key.                                             |

sys.node\_config

This table contains the effective node configuration for all nodes in the system.

| Column Name | Column Type | Column Description                                                                            |
|-------------|-------------|-----------------------------------------------------------------------------------------------|
| node_id     | UUID        | Universally Unique IDentifier (UUID) of the node where this configuration exists (sys.nodes). |
| key         | CHAR        | String that represents the configuration parameter.                                           |
| value       | CHAR        | Value for the configuration parameter key.                                                    |

Connectivity Pools

sys.connectivity\_pool\_participants

This table contains the connectivity pool participants and their information.

| Column Name        | Column Type | Column Description                                                                           |
|--------------------|-------------|----------------------------------------------------------------------------------------------|
| id                 | UUID        | Universally Unique IDentifier (UUID) of the connectivity pool that contains the participant. |
| node_id            | UUID        | UUID of the participant node.                                                                |
| created_at         | TIMESTAMP   | Timestamp that indicates the creation of the connectivity pool participant.                  |
| updated_at         | TIMESTAMP   | Timestamp that indicates the last update of the connectivity pool participant.               |
| listen_address     | CHAR        | Address where the participant node listens.                                                  |
| listen_port        | INT         | Port number where the participant node listens.                                              |
| advertised_address | CHAR        | Address that the participant node advertises to the client.                                  |
| advertised_port    | INT         | Optional port number that the participant node advertises to the client.                     |

| Column Name  | Column Type | Column Description                                                                                 |
|--------------|-------------|----------------------------------------------------------------------------------------------------|
| openapi_port | INT         | Optional port number where the API, written according to the OpenAPI Specification (OAS), listens. |

sys.connectivity\_pools

This table contains the connectivity pools and their information (excluding participants).

| Column Name          | Column Type | Column Description                                                      |
|----------------------|-------------|-------------------------------------------------------------------------|
| id                   | UUID        | Universally Unique Identifier (UUID) of the connectivity pool.          |
| name                 | CHAR        | Name of the connectivity pool.                                          |
| created_at           | TIMESTAMP   | Timestamp that indicates the creation of the connectivity pool.         |
| updated_at           | TIMESTAMP   | Timestamp that indicates the last update of the connectivity pool.      |
| source_address       | CHAR        | Source address in CIDR notation for the connectivity pool.              |
| source_port          | INT         | Optional source port of the connectivity pool.                          |
| priority             | INT         | Priority of the connectivity pool.                                      |
| sso_integration_name | CHAR        | Name of the default SSO security integration for the connectivity pool. |

Databases

sys.columns

This table contains all columns in each table in the system.

| Column Name        | Column Type | Column Description                                          |
|--------------------|-------------|-------------------------------------------------------------|
| id                 | UUID        | Universally Unique Identifier (UUID) of the column.         |
| name               | CHAR        | Name of the column.                                         |
| data_type          | CHAR        | Data type of the column (INT, CHAR, BOOLEAN, etc.).         |
| position           | INT         | Position of the column in its table.                        |
| table_id           | UUID        | UUID of the table that contains this column (sys.tables).   |
| nullable           | BOOLEAN     | Specifies whether the values in this column can be NULL.    |
| default_expression | CHAR        | Default value of this column when the value is unspecified. |
| description        | CHAR        | Description of the column.                                  |
| ordinal            | LONG        | Ordinal of the column with respect to its table.            |
| gdc_type           | CHAR        | Type of GDC applied to this column.                         |
| potential_index    | BOOLEAN     | Specifies whether the column is a potential index.          |

## sys.databases

This table contains all databases defined in the system.

| Column Name               | Column Type | Column Description                                      |
|---------------------------|-------------|---------------------------------------------------------|
| id                        | UUID        | Universally Unique IDentifier (UUID) of the database.   |
| name                      | CHAR        | Name of the database.                                   |
| created_at                | TIMESTAMP   | Timestamp that specifies when the database was created. |
| user_can_view_all_queries | BOOLEAN     | Current user can view all queries made in this database |

## sys.functions

This table contains all of the user-defined functions in the system.

| Column Name                    | Column Type | Column Description                                                                                                            |
|--------------------------------|-------------|-------------------------------------------------------------------------------------------------------------------------------|
| id                             | UUID        | Universally Unique IDentifier (UUID) of the user-defined function.                                                            |
| schema                         | CHAR        | Schema of the user-defined function.                                                                                          |
| name                           | CHAR        | Name of the user-defined function.                                                                                            |
| sql_expression                 | CHAR        | For SQL user-defined functions, this is the expression that defines the function.                                             |
| database_id                    | UUID        | UUID of the database of this user-defined function (sys.databases).                                                           |
| created_at                     | TIMESTAMP   | Timestamp of when this user-defined function was created.                                                                     |
| updated_at                     | TIMESTAMP   | Timestamp of when this user-defined function was last updated.                                                                |
| language                       | CHAR        | Language that the user-defined function is written in.                                                                        |
| function_type                  | CHAR        | For non-SQL user-defined functions, the type of function being defined.                                                       |
| filename                       | CHAR        | For non-SQL user-defined functions, the name of the file containing the user-defined function code.                           |
| function_name_in_external_code | CHAR        | For non-SQL user-defined functions, the name of the Java or Python function or method that defines the user-defined function. |
| return_type                    | CHAR        | For non-SQL user-defined functions, the return type of the user-defined function.                                             |

## sys.global\_map\_table\_info

This table contains the global dictionary compression (GDC) details associated with each compressed column.

| Column Name     | Column Type | Column Description                                              |
|-----------------|-------------|-----------------------------------------------------------------|
| table_id        | UUID        | Universally Unique IDentifier (UUID) of the table (sys.tables). |
| column_id       | UUID        | UUID of the column (sys.columns).                               |
| compressed_size | INT         | Compressed column size (in bytes).                              |
| max_count       | LONG        | Maximum amount of unique column values allowed in the column.   |
| current_count   | LONG        | Current amount of unique column values in the column.           |

sys.index\_advisor\_enabled

This table contains all tables and databases where the index advisor is currently enabled.

| Column Name | Column Type | Column Description                                             |
|-------------|-------------|----------------------------------------------------------------|
| id          | UUID        | Universally Unique IDentifier (UUID) of the table or database. |
| name        | CHAR        | Name of the table or database.                                 |
| object_type | CHAR        | Determines whether this object is a database or table.         |

sys.index\_advisors

This table contains raw index advisor information.

| Column Name | Column Type | Column Description                                                                      |
|-------------|-------------|-----------------------------------------------------------------------------------------|
| table_id    | UUID        | Universally Unique IDentifier (UUID) of the table for the potential index (sys.tables). |
| column_id   | UUID        | UUID that represents the column (sys.columns).                                          |
| table_name  | CHAR        | Name of the table.                                                                      |
| column_name | CHAR        | Name of the column.                                                                     |
| index_type  | CHAR        | Type of the potential index.                                                            |
| query_id    | UUID        | UUID that represents the query.                                                         |
| query_time  | TIMESTAMP   | A UNIX timestamp that represents when the query executed.                               |
| sql         | CHAR        | The SQL statement to use for creating this index.                                       |

sys.index\_columns

This table contains the index settings applied to each column.

| Column Name | Column Type | Column Description                                                            |
|-------------|-------------|-------------------------------------------------------------------------------|
| index_id    | UUID        | Universally Unique IDentifier (UUID) that represents the index (sys.indexes). |
| column_id   | UUID        | UUID that represents the column (sys.columns).                                |
| ordinal     | INT         | Ordinal of the column in the index structure.                                 |

| Column Name    | Column Type | Column Description                                                                                          |
|----------------|-------------|-------------------------------------------------------------------------------------------------------------|
| tuple_element  | CHAR        | Specifies the name of the component column that the index applies to if the index exists on a tuple column. |
| ascending      | BOOLEAN     | Indicates whether the values in this column are built in ascending order.                                   |
| column_ordinal | LONG        | Ordinal position of this column in the source table.                                                        |

sys.index\_recommendations

This table contains index recommendations.

| Column Name | Column Type | Column Description                                |
|-------------|-------------|---------------------------------------------------|
| table_name  | CHAR        | Name of the table.                                |
| column_name | CHAR        | Name of the column.                               |
| sql         | CHAR        | The SQL statement to use for creating this index. |
| usage_count | LONG        | Number of times this index would be used.         |

sys.index\_usage

This table contains index usage information.

| Column Name | Column Type | Column Description                                                            |
|-------------|-------------|-------------------------------------------------------------------------------|
| table_id    | UUID        | Universally Unique IDentifier (UUID) of the table where the index is located. |
| index_id    | UUID        | UUID of the index.                                                            |
| segment_id  | UUID        | UUID of the segment with the index.                                           |
| query_id    | UUID        | UUID of the query that could have used the index.                             |
| was_used    | BOOLEAN     | Whether or not the index was used.                                            |

sys.indexes

This table contains all indexes defined on tables.

| Column Name | Column Type | Column Description                                 |
|-------------|-------------|----------------------------------------------------|
| id          | UUID        | Universally Unique IDentifier (UUID) of the index. |
| name        | CHAR        | Name of the index.                                 |
| index_type  | CHAR        | Type of the index.                                 |
| table_id    | UUID        | UUID of the table for the index (sys.tables).      |
| index_use   | CHAR        | Intended use for the index by the system.          |
| ngram_size  | INT         | Size of each N-gram (in bytes).                    |
| enabled     | BOOLEAN     | Whether the index is enabled on this table.        |

## sys.locations

This table contains all of the external table providers.

| Column Name | Column Type | Column Description                                                          |
|-------------|-------------|-----------------------------------------------------------------------------|
| id          | UUID        | Universally Unique IDentifier (UUID) of the location.                       |
| name        | CHAR        | Name of the location.                                                       |
| remote_info | CHAR        | Remote information that defines how to talk to the external table provider. |
| database_id | UUID        | UUID of the database of this location (sys.databases).                      |
| created_at  | TIMESTAMP   | Timestamp that represents when this location was created.                   |
| updated_at  | TIMESTAMP   | Timestamp that represents when this location was last updated.              |

## sys.procedures

This table contains all of the stored procedures in the system.

| Column Name   | Column Type | Column Description                                                                      |
|---------------|-------------|-----------------------------------------------------------------------------------------|
| id            | UUID        | Universally Unique IDentifier (UUID) of the stored procedure.                           |
| schema        | CHAR        | Schema of the stored procedure.                                                         |
| name          | CHAR        | Name of the stored procedure.                                                           |
| database_id   | UUID        | UUID of the database of this stored procedure (sys.databases).                          |
| created_at    | TIMESTAMP   | Timestamp of when this stored procedure was created.                                    |
| updated_at    | TIMESTAMP   | Timestamp of when this stored procedure was last updated.                               |
| language      | CHAR        | Language that the stored procedure is written in.                                       |
| filename      | CHAR        | The name of the file that the database should load when it needs this stored procedure. |
| function_name | CHAR        | The name of the function to execute to execute the stored procedure.                    |

## sys.tables

This table contains all tables defined in the system.

| Column Name | Column Type | Column Description                                              |
|-------------|-------------|-----------------------------------------------------------------|
| id          | UUID        | Universally Unique IDentifier (UUID) of the table (sys.tables). |
| name        | CHAR        | Name of the table.                                              |
| schema      | CHAR        | Name of the schema.                                             |
| database_id | UUID        | UUID of the database (sys.databases).                           |

| Column Name                  | Column Type | Column Description                                                                                          |
|------------------------------|-------------|-------------------------------------------------------------------------------------------------------------|
| storage_space_id             | UUID        | UUID of the storage space (sys.storage_spaces).                                                             |
| maximum_segment_size_gib     | INT         | Maximum size of a segment for the table in GiB.                                                             |
| description                  | CHAR        | Detailed description of the table.                                                                          |
| streamloader_property_string | CHAR        | Additional properties assigned to control the stream loading behavior for the table.                        |
| lts_property_string          | CHAR        | Additional properties assigned to control the Foundation role (formerly “lts role”) behavior for the table. |
| created_at                   | TIMESTAMP   | Timestamp that represents when the table was created.                                                       |
| altered_at                   | TIMESTAMP   | Timestamp that represents when the table’s structure was last modified via DDL.                             |
| rolehostd_version            | CHAR        | The rolehostd version in which this table was created                                                       |
| software_compatible_version  | INT         | The rolehostd version this table is compatible with.                                                        |
| creator_id                   | UUID        | The UUID of the user or group who created the table (sys.users).                                            |

#### sys.tkt\_table\_clustering\_column\_indices

This table contains the columns defined as part of a clustering index.

| Column Name | Column Type | Column Description                                              |
|-------------|-------------|-----------------------------------------------------------------|
| table_id    | UUID        | Universally Unique IDentifier (UUID) of the table (sys.tables). |
| column_id   | UUID        | UUID of the column (sys.columns).                               |

#### sys.tkt\_table\_info

This table contains the columns that are defined as the time key on each user-defined table and the size of the time bucket in use.

| Column Name       | Column Type | Column Description                                                             |
|-------------------|-------------|--------------------------------------------------------------------------------|
| table_id          | UUID        | Universally Unique IDentifier (UUID) of the table (sys.tables).                |
| time_column_id    | UUID        | UUID of the column that represents the time column on the table (sys.columns). |
| time_bucket_width | LONG        | Time bucket width used to segment data in nanoseconds.                         |

#### sys.user\_mappings

This table contains user mappings for locations.

| Column Name     | Column Type | Column Description                                                 |
|-----------------|-------------|--------------------------------------------------------------------|
| id              | UUID        | Universally Unique Identifier (UUID) of the user mapping.          |
| name            | CHAR        | Name of the user mapping.                                          |
| local_userid    | CHAR        | Local user identifier.                                             |
| remote_userid   | CHAR        | Remote user identifier.                                            |
| location_id     | UUID        | UUID of the location for this user mapping.                        |
| database_id     | UUID        | UUID of the database for this user mapping (sys.databases).        |
| created_at      | TIMESTAMP   | Timestamp that represents when this user mapping was created.      |
| updated_at      | TIMESTAMP   | Timestamp that represents when this user mapping was last updated. |
| remote_password | CHAR        | Password for the remote user identifier.                           |

## sys.view\_columns

This table contains all columns in each view in the system.

| Column Name        | Column Type | Column Description                                      |
|--------------------|-------------|---------------------------------------------------------|
| id                 | UUID        | The Universally Unique Identifier (UUID) of the column. |
| name               | CHAR        | The name of the column.                                 |
| data_type          | CHAR        | The data type of the column.                            |
| view_id            | UUID        | The UUID of the view this column comes from.            |
| nullable           | BOOLEAN     | Whether or not this column is nullable.                 |
| default_expression | CHAR        | The default expression of this column, if it exists.    |
| ordinal            | LONG        | The ordinal of this column.                             |

## sys.views

This table contains all user-defined database views created on the system.

| Column Name                            | Column Type | Column Description                                                        |
|----------------------------------------|-------------|---------------------------------------------------------------------------|
| id                                     | UUID        | Universally Unique Identifier (UUID) of the view.                         |
| name                                   | CHAR        | Name of the view.                                                         |
| database_id                            | UUID        | UUID of the database (sys.databases).                                     |
| schema                                 | CHAR        | Schema name where the view exists.                                        |
| query                                  | CHAR        | Query used to generate the view content.                                  |
| description                            | CHAR        | Detailed description of the view.                                         |
| global_dictionary_compression_table_id | UUID        | UUID of the table with the GDC column (sys.tables).                       |
| created_at                             | TIMESTAMP   | Timestamp that represents the date and time for the creation of the view. |



| Column Name | Column Type | Column Description                                                           |
|-------------|-------------|------------------------------------------------------------------------------|
| updated_at  | TIMESTAMP   | Timestamp that represents the date and time for the last update of the view. |
| creator_id  | UUID        | The UUID of the user or group who created the view (sys.users/sys.groups).   |

## Loading

### sys.load\_errors

The **load\_errors** view shows information for the load errors in the system.

| Column Name   | Column Type | Column Description                                                                                                                                         |
|---------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| loader_id     | UUID        | The unique identifier of the Loader Node (sys.nodes).                                                                                                      |
| table_id      | UUID        | If the event is specific to a table, the unique identifier of the table (sys.tables).                                                                      |
| error_message | CHAR        | The message that provides more details about the error.                                                                                                    |
| timestamp     | TIMESTAMP   | The timestamp that represents when this error occurred.                                                                                                    |
| scope_id      | UUID        | If the error is specific to a storage scope, the unique identifier of the scope, which might be the identifier of the query for the results (sys.queries). |

### sys.load\_events

The **load\_events** view shows information for the load events in the system.

| Column Name     | Column Type | Column Description                                                                                                                                         |
|-----------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| loader_id       | UUID        | The unique identifier of the Loader Node (sys.nodes).                                                                                                      |
| table_id        | UUID        | If the event is specific to a table, the unique identifier of the table (sys.tables).                                                                      |
| event_type      | CHAR        | The type of the event.                                                                                                                                     |
| event_message   | CHAR        | The message that provides more details about the event.                                                                                                    |
| event_timestamp | TIMESTAMP   | The timestamp that represents when this event occurred.                                                                                                    |
| scope_id        | UUID        | If the error is specific to a storage scope, the unique identifier of the scope, which might be the identifier of the query for the results (sys.queries). |

## Machine Learning

## sys.association\_rules\_models

This table contains the name of the snapshot table for an association rules model.

| Column Name               | Column Type | Column Description                                                                                |
|---------------------------|-------------|---------------------------------------------------------------------------------------------------|
| machine_learning_model_id | UUID        | Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models). |
| num_arguments             | INT         | Number of arguments in the machine learning model data.                                           |
| table_name                | CHAR        | Name of the snapshot table.                                                                       |

## sys.decision\_tree\_models

This table contains all model parameters for Decision Tree Models in the system.

| Column Name               | Column Type | Column Description                                                                                |
|---------------------------|-------------|---------------------------------------------------------------------------------------------------|
| machine_learning_model_id | UUID        | Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models). |
| case_statement            | CHAR        | The case statement that defines the decision tree.                                                |
| case_statement_confidence | CHAR        | The case statement that defines the decision tree and returns confidences.                        |
| num_arguments             | INT         | Number of arguments in the machine learning model data.                                           |
| correctly_classified      | DOUBLE      | Percentage of training data that the decision tree model correctly classifies.                    |
| area_under_roc            | DOUBLE      | Area under the ROC curve of the decision tree model.                                              |

## sys.feedforward\_network\_models

This table contains all model parameters for Feedforward Neural Network Models in the system.

| Column Name               | Column Type   | Column Description                                                                                |
|---------------------------|---------------|---------------------------------------------------------------------------------------------------|
| machine_learning_model_id | UUID          | Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models). |
| coefficients              | ARRAY(DOUBLE) | List of coefficients associated with the feedforward network model.                               |
| num_arguments             | INT           | Number of arguments in the machine learning model data.                                           |
| average_loss              | DOUBLE        | Average value of the loss function.                                                               |

## sys.gaussian\_mixture\_models

This table contains all model parameters for Gaussian Mixture Models.

| Column Name               | Column Type   | Column Description                                                                                |
|---------------------------|---------------|---------------------------------------------------------------------------------------------------|
| machine_learning_model_id | UUID          | Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models). |
| coefficients              | ARRAY(DOUBLE) | List of coefficients associated with the Gaussian Mixture Model.                                  |
| num_distributions         | INT           | Number of distributions in the Gaussian Mixture Model.                                            |
| average_loss              | DOUBLE        | Average value of the loss function.                                                               |

## sys.k\_means\_models

This table contains all model parameters for K-Means Clustering Models in the system.

| Column Name               | Column Type   | Column Description                                                                                |
|---------------------------|---------------|---------------------------------------------------------------------------------------------------|
| machine_learning_model_id | UUID          | Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models). |
| centroids                 | ARRAY(DOUBLE) | Values that represent the centroid values of each cluster.                                        |
| num_arguments             | INT           | Number of arguments in the machine learning model data.                                           |
| means                     | ARRAY(DOUBLE) | Means of each of the feature columns in the data.                                                 |
| standard_deviations       | ARRAY(DOUBLE) | Standard deviations of each of the feature columns in the data.                                   |

## sys.k\_nearest\_neighbors\_models

This table contains the number of arguments and the target table for K-Nearest Neighbors Models in the system.

| Column Name               | Column Type   | Column Description                                                                                |
|---------------------------|---------------|---------------------------------------------------------------------------------------------------|
| machine_learning_model_id | UUID          | Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models). |
| num_arguments             | INT           | Number of arguments in the machine learning model data.                                           |
| table_name                | CHAR          | Name of the target table for the K Nearest Neighbors (KNN) model.                                 |
| correctly_classified      | DOUBLE        | Percentage of training data that is correctly classified by the KNN model.                        |
| means                     | ARRAY(DOUBLE) | Means of each of the feature columns in the data.                                                 |
| standard_deviations       | ARRAY(DOUBLE) | Standard deviations of each of the feature columns in the data.                                   |

## sys.linear\_combination\_regression\_models

This table contains all model parameters for Linear Combination Regression Models in the system.

| Column Name                  | Column Type   | Column Description                                                                        |
|------------------------------|---------------|-------------------------------------------------------------------------------------------|
| machine_learning_model_id    | UUID          | Universally Unique IDentifier (UUID) of the machine learning model                        |
| coefficients                 | ARRAY(DOUBLE) | (sys.machine_learning_models). List of model coefficients associated with the regression. |
| y_intercept                  | DOUBLE        | y-intercept of the regression.                                                            |
| coefficient_of_determination | DOUBLE        | R <sup>2</sup> value of the regression, if calculated, otherwise NULL.                    |
| num_arguments                | INT           | Number of arguments in the machine learning model data.                                   |
| rmse                         | DOUBLE        | Root mean square error of the regression.                                                 |
| adjusted_r2                  | DOUBLE        | Adjusted R <sup>2</sup> value of the regression.                                          |

#### sys.linear\_discriminant\_analysis\_models

This table contains all model parameters for Linear Discriminant Analysis (LDA) Models in the system.

| Column Name               | Column Type   | Column Description                                                                 |
|---------------------------|---------------|------------------------------------------------------------------------------------|
| machine_learning_model_id | UUID          | Universally Unique IDentifier (UUID) of the machine learning model                 |
| coefficients              | ARRAY(DOUBLE) | (sys.machine_learning_models). List of coefficients associated with the LDA model. |
| num_features              | INT           | Number of features returned by the LDA model.                                      |
| importance                | ARRAY(DOUBLE) | Importance of each of the output features that represent the data.                 |

#### sys.logistic\_regression\_models

This table contains all model parameters for Logistic Regression Models in the system.

| Column Name               | Column Type   | Column Description                                                                                 |
|---------------------------|---------------|----------------------------------------------------------------------------------------------------|
| machine_learning_model_id | UUID          | Universally Unique IDentifier (UUID) of the machine learning model                                 |
| coefficients              | ARRAY(DOUBLE) | (sys.machine_learning_models). List of coefficients associated with the logistic regression model. |
| correctly_classified      | DOUBLE        | Percentage of training data that is correctly classified by the logistic regression.               |
| num_arguments             | INT           | Number of arguments in the machine learning model data.                                            |
| zero_case                 | CHAR          | The case where the logistic regression uses a value of 0.                                          |
| one_case                  | CHAR          | The case where the logistic regression uses a value of 1.                                          |

| Column Name | Column Type | Column Description                          |
|-------------|-------------|---------------------------------------------|
| classes     | ARRAY(CHAR) | Target classes for the logistic regression. |

#### sys.machine\_learning\_model\_options

This table contains options defined for each machine learning model. The database stores these options as key-value pairs.

| Column Name                  | Column Type | Column Description                                                |
|------------------------------|-------------|-------------------------------------------------------------------|
| machine_learning_model_id    | UUID        | UUID of the machine learning model (sys.machine_learning_models). |
| machine_learning_model_key   | CHAR        | Machine learning model configuration option key.                  |
| machine_learning_model_value | CHAR        | Machine learning model configuration option value.                |

#### sys.machine\_learning\_models

This table contains the machine learning models that are defined in the database.

| Column Name                 | Column Type | Column Description                                                                           |
|-----------------------------|-------------|----------------------------------------------------------------------------------------------|
| id                          | UUID        | Universally Unique IDentifier (UUID) of the machine learning model.                          |
| name                        | CHAR        | Name of the machine learning model.                                                          |
| schema                      | CHAR        | Schema of the machine learning model.                                                        |
| machine_learning_model_type | CHAR        | Type of the machine learning model.                                                          |
| database_id                 | UUID        | UUID of the database that contains the machine learning model definition (sys.databases).    |
| on_select                   | CHAR        | The SELECT SQL statement used to create this model.                                          |
| creator_id                  | UUID        | The UUID of the user or group who created the machine learning model (sys.users/sys.groups). |

#### sys.multiple\_linear\_regression\_models

This table contains the y-intercept and coefficient of determination values for the multiple linear regression machine learning models.

| Column Name                  | Column Type | Column Description                                                                                |
|------------------------------|-------------|---------------------------------------------------------------------------------------------------|
| machine_learning_model_id    | UUID        | Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models). |
| y_intercept                  | DOUBLE      | y-intercept of the regression.                                                                    |
| coefficient_of_determination | DOUBLE      | R <sup>2</sup> value of the regression, if calculated, otherwise NULL.                            |

| Column Name | Column Type   | Column Description                               |
|-------------|---------------|--------------------------------------------------|
| rmse        | DOUBLE        | Root mean square error of the regression.        |
| adjusted_r2 | DOUBLE        | Adjusted R <sup>2</sup> value of the regression. |
| slopes      | ARRAY(DOUBLE) | Slopes for each variable in the regression.      |

#### sys.naive\_bayes\_models

This table contains all model parameters for Naive Bayes Classification Models in the system.

| Column Name                 | Column Type | Column Description                                                                                |
|-----------------------------|-------------|---------------------------------------------------------------------------------------------------|
| machine_learning_model_id   | UUID        | Universally Unique Identifier (UUID) of the machine learning model (sys.machine_learning_models). |
| num_arguments               | INT         | Number of arguments in the machine learning model data.                                           |
| result_probability_table    | CHAR        | Name of the internal result probability table.                                                    |
| feature_result_matrix_table | CHAR        | Name of the internal feature result matrix table.                                                 |
| correctly_classified        | DOUBLE      | Percentage of training data that is correctly classified by the Naive Bayes model.                |

#### sys.nonlinear\_regression\_models

This table contains all model parameters for Nonlinear Regression Models in the system.

| Column Name                  | Column Type   | Column Description                                                                                |
|------------------------------|---------------|---------------------------------------------------------------------------------------------------|
| machine_learning_model_id    | UUID          | Universally Unique Identifier (UUID) of the machine learning model (sys.machine_learning_models). |
| coefficients                 | ARRAY(DOUBLE) | List of coefficients associated with the nonlinear regression model.                              |
| coefficient_of_determination | DOUBLE        | R <sup>2</sup> value of the regression, if calculated, otherwise NULL.                            |
| num_arguments                | INT           | Number of arguments in the machine learning model data.                                           |
| rmse                         | DOUBLE        | Root mean square error of the regression.                                                         |
| adjusted_r2                  | DOUBLE        | Adjusted R <sup>2</sup> value of the regression.                                                  |

#### sys.polynomial\_regression\_models

This table contains all model parameters for Polynomial Regression Models in the system.

| Column Name                  | Column Type   | Column Description                                                                                         |
|------------------------------|---------------|------------------------------------------------------------------------------------------------------------|
| machine_learning_model_id    | UUID          | Universally Unique Identifier (UUID) of the machine learning model                                         |
| coefficients                 | ARRAY(DOUBLE) | (sys.machine_learning_models). List of model coefficients associated with the polynomial regression model. |
| y_intercept                  | DOUBLE        | y-intercept of the regression.                                                                             |
| coefficient_of_determination | DOUBLE        | R <sup>2</sup> value of the regression, if calculated, otherwise NULL.                                     |
| num_arguments                | INT           | Number of arguments in the machine learning model data.                                                    |
| rmse                         | DOUBLE        | Root mean square error of the regression.                                                                  |
| adjusted_r2                  | DOUBLE        | Adjusted R <sup>2</sup> value of the regression.                                                           |

sys.principal\_component\_analysis\_models

This table contains all model parameters for Principal Component Analysis (PCA) Models in the system.

| Column Name               | Column Type   | Column Description                                                                 |
|---------------------------|---------------|------------------------------------------------------------------------------------|
| machine_learning_model_id | UUID          | Universally Unique Identifier (UUID) of the machine learning model                 |
| coefficients              | ARRAY(DOUBLE) | (sys.machine_learning_models). List of coefficients associated with the PCA model. |
| num_features              | INT           | Number of features returned by the PCA model.                                      |
| importance                | ARRAY(DOUBLE) | Importance of each of the output features that represents the data.                |
| means                     | ARRAY(DOUBLE) | Means of each of the feature columns in the data.                                  |
| standard_deviations       | ARRAY(DOUBLE) | Standard deviations of each of the feature columns in the data.                    |

sys.random\_forest\_models

This table contains all model parameters for Random Forest Models in the system.

| Column Name                | Column Type | Column Description                                                                                           |
|----------------------------|-------------|--------------------------------------------------------------------------------------------------------------|
| machine_learning_model_id  | UUID        | Universally Unique Identifier (UUID) of the machine learning model                                           |
| vote_expression            | CHAR        | (sys.machine_learning_models). The vote expression that retrieves information from the child decision trees. |
| vote_expression_confidence | CHAR        | The vote expression that retrieves information from the child decision trees and returns confidences.        |

| Column Name          | Column Type | Column Description                                                             |
|----------------------|-------------|--------------------------------------------------------------------------------|
| num_arguments        | INT         | Number of arguments in the machine learning model data.                        |
| num_children         | INT         | Number of child trees in the forest.                                           |
| correctly_classified | DOUBLE      | Percentage of training data that the random forest model correctly classifies. |
| area_under_roc       | DOUBLE      | Area under the ROC curve of the random forest model.                           |

#### sys.simple\_linear\_regression\_models

This table contains the slope, y-intercept, and coefficient of determination values for single linear regression machine learning models.

| Column Name                  | Column Type | Column Description                                                                                   |
|------------------------------|-------------|------------------------------------------------------------------------------------------------------|
| machine_learning_model_id    | UUID        | Universally Unique IDentifier (UUID) of the machine learning model<br>(sys.machine_learning_models). |
| slope                        | DOUBLE      | Slope of the regression.                                                                             |
| y_intercept                  | DOUBLE      | The y-intercept of the regression.                                                                   |
| coefficient_of_determination | DOUBLE      | R <sup>2</sup> value of the regression, if calculated, otherwise NULL.                               |
| rmse                         | DOUBLE      | Root mean square error of the regression.                                                            |
| adjusted_r2                  | DOUBLE      | Adjusted R <sup>2</sup> value of the regression.                                                     |

#### sys.support\_vector\_machine\_models

This table contains all model parameters for Support Vector Machine (SVM) Models in the system.

| Column Name               | Column Type   | Column Description                                                                                   |
|---------------------------|---------------|------------------------------------------------------------------------------------------------------|
| machine_learning_model_id | UUID          | Universally Unique IDentifier (UUID) of the machine learning model<br>(sys.machine_learning_models). |
| coefficients              | ARRAY(DOUBLE) | List of coefficients associated with the SVM model.                                                  |
| correctly_classified      | DOUBLE        | Percentage of training data that is correctly classified by the SVM model.                           |
| num_arguments             | INT           | Number of arguments in the machine learning model data.                                              |
| negative_case             | CHAR          | Case when the SVM model classifies data as negative.                                                 |
| positive_case             | CHAR          | Case when the SVM model classifies data as positive.                                                 |
| classes                   | ARRAY(CHAR)   | Target classes for the SVM model.                                                                    |

#### sys.vector\_autoregression\_models

This table contains all model parameters for Vector Autoregression Models in the system.



| Column Name                  | Column Type   | Column Description                                                                                   |
|------------------------------|---------------|------------------------------------------------------------------------------------------------------|
| machine_learning_model_id    | UUID          | Universally Unique Identifier (UUID) of the machine learning model                                   |
| coefficients                 | ARRAY(DOUBLE) | (sys.machine_learning_models). List of coefficients associated with the vector autoregression model. |
| coefficient_of_determination | DOUBLE        | R <sup>2</sup> value of the autoregression, if calculated, otherwise NULL.                           |

## Monitoring

sys.active\_operator\_instances

This table contains information about operator instances that are active in the database.

| Column Name              | Column Type | Column Description                                                                                 |
|--------------------------|-------------|----------------------------------------------------------------------------------------------------|
| query_id                 | UUID        | A unique identifier of the query to which the operator belongs.                                    |
| operator_id              | UUID        | A unique identifier of the finalized operator.                                                     |
| node_id                  | UUID        | The unique identifier of the node that stores the information about this operator.                 |
| silo_id                  | LONG        | The unique identifier of the silo that stores the information about this operator.                 |
| core_id                  | LONG        | The unique identifier of the virtual machine core that stores the information about this operator. |
| op_runtime_id            | LONG        | The unique identifier for the operator instance that the system assigns at runtime.                |
| op_instance_type         | CHAR        | The name of the type for this operator instance.                                                   |
| bloom_filtered_rows      | LONG        | The rows filtered by Bloom filters.                                                                |
| rows_received            | LONG        | The number of rows fetched by the operator instance.                                               |
| rows_processed           | LONG        | The number of rows processed by this operator instance during execution.                           |
| rows_emitted             | LONG        | The number of rows returned by the operator instance.                                              |
| blocks_emitted           | LONG        | The number of data blocks returned by the operator instance.                                       |
| first_block_receive_time | TIMESTAMP   | The timestamp that specifies the return of the first data block.                                   |
| first_block_emitted_time | TIMESTAMP   | The timestamp that specifies the return of the first data block.                                   |
| num_cycles               | LONG        | The number of cycles that run on this operator instance.                                           |
| num_oom_cycles           | LONG        | The number of run cycles that an operator received for processing out-of-memory issues.            |

| Column Name                      | Column Type | Column Description                                                                                              |
|----------------------------------|-------------|-----------------------------------------------------------------------------------------------------------------|
| num_failed_oom_cycles            | LONG        | The number of out-of-memory cycles received that failed to do any work.                                         |
| num_no_work_oom_cycles           | LONG        | The number of out-of-memory cycles received that reported they had no work to do.                               |
| run_count                        | LONG        | The number of times that the scheduler has reviewed this operator and <b>isRunnable</b> is set to <b>true</b> . |
| no_work_cycles                   | LONG        | The number of times that the read and write cycle executed and performed no work.                               |
| backed_up_cycles                 | LONG        | The number of times a cycle stopped because a backup of the parent exists.                                      |
| leaf_branch_blocked_cycles       | LONG        | The number of times a cycle stopped on a leaf operator because of multi-child scheduling heuristics.            |
| num_eof_cycles                   | LONG        | The number of end-of-file cycles received.                                                                      |
| normal_run_time_in_ms            | DOUBLE      | The time, in milliseconds, of the normal run for the operator instance.                                         |
| dispatch_queue_time_in_ms        | DOUBLE      | The time spent, in milliseconds, processing incoming message events for the operator instance.                  |
| oom_run_time_in_ms               | DOUBLE      | The time, in milliseconds, of the out-of-memory issue for the operator instance.                                |
| max_hp_mem_usage                 | LONG        | The maximum heap memory usage by the operator instance.                                                         |
| num_blocks_written               | LONG        | The number of data blocks written to temporary disk.                                                            |
| num_blocks_read                  | LONG        | The number of data blocks read from temporary disk.                                                             |
| num_fragments_written            | LONG        | The number of written memory fragments.                                                                         |
| num_fragments_read               | LONG        | The number of read memory fragments.                                                                            |
| cycles_over_1_sec                | LONG        | The count of cycles that ran over 1 second for this operator instance.                                          |
| cycles_over_3_sec                | LONG        | The count of cycles that ran over 3 seconds for this operator instance.                                         |
| long_dispatch_queue_events       | LONG        | The count of dispatch queue events processed by the operator with a runtime greater than 0.5 seconds.           |
| total_bin_stream_allocated_bytes | LONG        | The total amount of memory allocated in output blocks for variable-length column data.                          |
| total_bin_stream_used_bytes      | LONG        | The total amount of memory used in output blocks for variable-length column data.                               |
| total_col_stream_allocated_bytes | LONG        | The total amount of memory allocated in output blocks for fixed-length column data.                             |

| Column Name                                  | Column Type | Column Description                                                                                                                                                                                                                                                                                   |
|----------------------------------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| total_col_stream_used_bytes                  | LONG        | The total amount of memory used in output blocks for fixed-length column data.                                                                                                                                                                                                                       |
| blocked_received                             | LONG        | The total number of input data blocks received by this operator instance.                                                                                                                                                                                                                            |
| blocked_created                              | LONG        | The total number of non-trivial output data blocks created by this operator instance.                                                                                                                                                                                                                |
| received_block_bin_stream_unused_space_bytes | LONG        | The total amount of wasted space across all binary streams in received data blocks (includes over-allocations as well as projected columns and dead memory). This number is NULL unless configuration parameters allow recording this statistic for each block because the calculation is expensive. |
| received_block_col_stream_unused_space_bytes | LONG        | The total amount of wasted space across all column streams in received data blocks (includes over-allocations as well as projected columns and dead memory). This number is NULL unless configuration parameters allow recording this statistic for each block because the calculation is expensive. |
| mandatory_alloc_attempts                     | LONG        | The count of cycles where the scheduler enabled mandatory allocation.                                                                                                                                                                                                                                |
| mandatory_alloc_success_count                | LONG        | The count of cycles where the scheduler enabled mandatory allocation and this operator broke an out-of-memory deadlock.                                                                                                                                                                              |
| max_num_pending_blocks                       | LONG        | The maximum number of pending blocks across all partitions during the lifetime of this operator.                                                                                                                                                                                                     |
| max_num_consecutive_backup_cycles            | LONG        | The maximum number of consecutive scheduler cycles where this operator has a backup. The scheduler can use this information to help detect certain resource deadlocks.                                                                                                                               |
| longest_time_since_unblocked_cycle           | LONG        | The maximum time between successful attempts to run this operator from the scheduler. The scheduler can use this information to help detect certain resource deadlocks.                                                                                                                              |
| compiled_level                               | LONG        | The level within a hierarchy of distinct operators to execute. A single plan operator can be compiled into this hierarchy.                                                                                                                                                                           |

| Column Name                                | Column Type | Column Description                                                                           |
|--------------------------------------------|-------------|----------------------------------------------------------------------------------------------|
| additional_json                            | CHAR        | Additional statistics for an operator instance in JSON format.                               |
| num_bloom_filters_sent                     | LONG        | The number of Bloom filters the operator sent to its peers.                                  |
| num_bloom_filters_received                 | LONG        | The number of Bloom filters the operator received from its peers.                            |
| num_key_lists_sent                         | LONG        | The number of potential index join key lists the operator sent to its peers.                 |
| num_key_lists_received                     | LONG        | The number of potential index join key lists the operator received from its peers.           |
| max_hash_table_rows                        | LONG        | The maximum number of rows present in any hash table used by this operator instance.         |
| plan_operator_estimated_output_cardinality | LONG        | The number of rows the optimizer estimated for this plan operator to emit.                   |
| num_delayed_exceptions_generated           | LONG        | The number of rows this operator generated with delayed exception information.               |
| num_pending_blocks_spilled                 | LONG        | The number of queued pending data blocks this operator spilled to disk.                      |
| max_normal_cycle_time_ms                   | DOUBLE      | The time, in milliseconds, of the longest normal run cycle for the operator instance.        |
| max_oom_cycle_time_ms                      | DOUBLE      | The time, in milliseconds, of the longest out-of-memory run cycle for the operator instance. |
| num_rows_received_per_plan_child           | ARRAY(LONG) | The number of rows this operator received from each of its children in the plan.             |
| is_runnable                                | BOOLEAN     | Whether the operator has available data to process.                                          |
| current_num_pending_blocks                 | LONG        | The current number of pending blocks across all partitions of this operator.                 |

sys.completed\_operator\_instances

This table contains information about operator instances that have finalized in the virtual machine.

| Column Name   | Column Type | Column Description                                                    |
|---------------|-------------|-----------------------------------------------------------------------|
| database_name | CHAR        | The database associated with the query to which the operator belongs. |
| user_name     | CHAR        | The user associated with the query to which the operator belongs.     |
| query_id      | UUID        | A unique identifier of the query to which the operator belongs.       |
| operator_id   | UUID        | A unique identifier of the finalized operator.                        |

| Column Name                | Column Type | Column Description                                                                                                          |
|----------------------------|-------------|-----------------------------------------------------------------------------------------------------------------------------|
| node_id                    | UUID        | The unique identifier of the node that stores the information about this operator.                                          |
| silo_id                    | LONG        | The unique identifier of the silo that stores the information about this operator.                                          |
| core_id                    | LONG        | The unique identifier of the virtual machine core that stores the information about this operator.                          |
| op_runtime_id              | LONG        | The unique identifier for the operator instance that the system assigns at runtime.                                         |
| op_instance_type           | CHAR        | The name of the type for this operator instance.                                                                            |
| bloom_filtered_rows        | LONG        | The rows filtered by Bloom filters.                                                                                         |
| rows_received              | LONG        | The number of rows fetched by the operator instance.                                                                        |
| rows_processed             | LONG        | The number of rows processed by this operator instance during execution.                                                    |
| rows_emitted               | LONG        | The number of rows returned by the operator instance.                                                                       |
| blocks_emitted             | LONG        | The number of data blocks returned by the operator instance.                                                                |
| finalization_time          | TIMESTAMP   | The timestamp that represents when the operator completed all work.                                                         |
| first_block_receive_time   | TIMESTAMP   | The timestamp that specifies the receipt of the first data block.                                                           |
| first_block_emitted_time   | TIMESTAMP   | The timestamp that specifies the return of the first data block.                                                            |
| num_cycles                 | LONG        | The number of cycles that run on this operator instance.                                                                    |
| num_oom_cycles             | LONG        | The number of run cycles that an operator received for processing out-of-memory issues.                                     |
| num_failed_oom_cycles      | LONG        | The number of out-of-memory cycles received that failed to do any work.                                                     |
| num_no_work_oom_cycles     | LONG        | The number of out-of-memory cycles received that reported they had no work to do.                                           |
| run_count                  | LONG        | The number of times that the scheduler has reviewed this operator and <code>isRunnable</code> is set to <code>true</code> . |
| no_work_cycles             | LONG        | The number of times that the read and write cycle executed and performed no work.                                           |
| backed_up_cycles           | LONG        | The number of times a cycle stopped because a backup of the parent exists.                                                  |
| leaf_branch_blocked_cycles | LONG        | The number of times a cycle stopped on a leaf operator because of multi-child scheduling heuristics.                        |
| num_eof_cycles             | LONG        | The number of end-of-file cycles received.                                                                                  |

| Column Name                                  | Column Type | Column Description                                                                                                                                                                                                                                                                                   |
|----------------------------------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| normal_run_time_in_ms                        | DOUBLE      | The time, in milliseconds, of the normal run for the operator instance.                                                                                                                                                                                                                              |
| dispatch_queue_time_in_ms                    | DOUBLE      | The time spent, in milliseconds, processing incoming message events for the operator instance.                                                                                                                                                                                                       |
| oom_run_time_in_ms                           | DOUBLE      | The time, in milliseconds, of the out-of-memory issue for the operator instance.                                                                                                                                                                                                                     |
| max_hp_mem_usage                             | LONG        | The maximum heap memory usage by the operator instance.                                                                                                                                                                                                                                              |
| num_blocks_written                           | LONG        | The number of data blocks written to temporary disk.                                                                                                                                                                                                                                                 |
| num_blocks_read                              | LONG        | The number of data blocks read from temporary disk.                                                                                                                                                                                                                                                  |
| num_fragments_written                        | LONG        | The number of written memory fragments.                                                                                                                                                                                                                                                              |
| num_fragments_read                           | LONG        | The number of read memory fragments.                                                                                                                                                                                                                                                                 |
| cycles_over_1_sec                            | LONG        | The count of cycles that ran over 1 second for this operator instance.                                                                                                                                                                                                                               |
| cycles_over_3_sec                            | LONG        | The count of cycles that ran over 3 seconds for this operator instance.                                                                                                                                                                                                                              |
| long_dispatch_queue_events                   | LONG        | The count of dispatch queue events processed by the operator with a runtime greater than 0.5 seconds.                                                                                                                                                                                                |
| total_bin_stream_allocated_bytes             | LONG        | The total amount of memory allocated in output blocks for variable-length column data.                                                                                                                                                                                                               |
| total_bin_stream_used_bytes                  | LONG        | The total amount of memory used in output blocks for variable-length column data.                                                                                                                                                                                                                    |
| total_col_stream_allocated_bytes             | LONG        | The total amount of memory allocated in output blocks for fixed-length column data.                                                                                                                                                                                                                  |
| total_col_stream_used_bytes                  | LONG        | The total amount of memory used in output blocks for fixed-length column data.                                                                                                                                                                                                                       |
| blocked_received                             | LONG        | The total number of input data blocks received by this operator instance.                                                                                                                                                                                                                            |
| blocked_created                              | LONG        | The total number of non-trivial output data blocks created by this operator instance.                                                                                                                                                                                                                |
| received_block_bin_stream_unused_space_bytes | LONG        | The total amount of wasted space across all binary streams in received data blocks (includes over-allocations as well as projected columns and dead memory). This number is NULL unless configuration parameters allow recording this statistic for each block because the calculation is expensive. |

| Column Name                                | Column Type | Column Description                                                                                                                                                                                                                                                                                   |
|--------------------------------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| received_block_col_stream_unused_space     | LONG        | The total amount of wasted space across all column streams in received data blocks (includes over-allocations as well as projected columns and dead memory). This number is NULL unless configuration parameters allow recording this statistic for each block because the calculation is expensive. |
| mandatory_alloc_attempts                   | LONG        | The count of cycles where the scheduler enabled mandatory allocation.                                                                                                                                                                                                                                |
| mandatory_alloc_success_count              | LONG        | The count of cycles where the scheduler enabled mandatory allocation and this operator broke an out-of-memory deadlock.                                                                                                                                                                              |
| max_num_pending_blocks                     | LONG        | The maximum number of pending blocks across all partitions during the lifetime of this operator.                                                                                                                                                                                                     |
| max_num_consecutive_backup_cycles          | LONG        | The maximum number of consecutive scheduler cycles where this operator has a backup. The scheduler can use this information to help detect certain resource deadlocks.                                                                                                                               |
| longest_time_since_unblocked_cycle         | LONG        | The maximum time between successful attempts to run this operator from the scheduler. The scheduler can use this information to help detect certain resource deadlocks.                                                                                                                              |
| compiled_level                             | LONG        | The level within a hierarchy of distinct operators to execute. A single plan operator can be compiled into this hierarchy.                                                                                                                                                                           |
| additional_json                            | CHAR        | Additional statistics for an operator instance in JSON format.                                                                                                                                                                                                                                       |
| num_bloom_filters_sent                     | LONG        | The number of Bloom filters the operator sent to its peers.                                                                                                                                                                                                                                          |
| num_bloom_filters_received                 | LONG        | The number of Bloom filters the operator received from its peers.                                                                                                                                                                                                                                    |
| num_key_lists_sent                         | LONG        | The number of potential index join key lists the operator sent to its peers.                                                                                                                                                                                                                         |
| num_key_lists_received                     | LONG        | The number of potential index join key lists the operator received from its peers.                                                                                                                                                                                                                   |
| max_hash_table_rows                        | LONG        | The maximum number of rows present in any hash table used by this operator instance.                                                                                                                                                                                                                 |
| plan_operator_estimated_output_cardinality | LONG        | The number of rows the optimizer estimated for this plan operator to emit.                                                                                                                                                                                                                           |

| Column Name                      | Column Type | Column Description                                                                           |
|----------------------------------|-------------|----------------------------------------------------------------------------------------------|
| num_delayed_exceptions_generated | LONG        | The number of rows this operator generated with delayed exception information.               |
| num_pending_blocks_spilled       | LONG        | The number of queued pending data blocks this operator spilled to disk.                      |
| max_normal_cycle_time_ms         | DOUBLE      | The time, in milliseconds, of the longest normal run cycle for the operator instance.        |
| max_oom_cycle_time_ms            | DOUBLE      | The time, in milliseconds, of the longest out-of-memory run cycle for the operator instance. |
| num_rows_received_per_plan_child | ARRAY(LONG) | The number of rows this operator received from each of its children in the plan.             |

#### sys.completed\_queries

This table contains information about queries that finished execution in the database.

| Column Name       | Column Type | Column Description                                                                                                               |
|-------------------|-------------|----------------------------------------------------------------------------------------------------------------------------------|
| query_id          | UUID        | A unique identifier of the query that ran.                                                                                       |
| user              | CHAR        | The name of the user that executed the query.                                                                                    |
| database_name     | CHAR        | The name of the database that runs this query.                                                                                   |
| database_id       | UUID        | The unique identifier of the database this query ran in.                                                                         |
| sql               | CHAR        | The executed SQL statement.                                                                                                      |
| sql_text_length   | LONG        | The length of the SQL statement.                                                                                                 |
| referenced_tables | ARRAY(CHAR) | The tables and views referenced by the query.                                                                                    |
| total_time        | LONG        | The total time in milliseconds the query took to run. This time includes generation_time, optimization_time, and execution_time. |
| generation_time   | LONG        | The time in milliseconds it took to generate the query before any processing happened.                                           |
| optimization_time | LONG        | The time in milliseconds the optimizer processed the query and built an optimized plan for the query.                            |
| execution_time    | LONG        | Time in milliseconds the query was executed by the database on the LTS nodes to return the query result.                         |
| timestamp_start   | TIMESTAMP   | A timestamp that represents the point in time when the query entered the system.                                                 |
| parsing_time      | LONG        | The time in milliseconds the query was parsing.                                                                                  |



| Column Name                    | Column Type | Column Description                                                                                                                                                                                                                                                         |
|--------------------------------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| cache_lookup_time              | LONG        | The time in milliseconds the query spent in lookup up matching cached result sets.                                                                                                                                                                                         |
| validation_time                | LONG        | The time in milliseconds the query was validating.                                                                                                                                                                                                                         |
| plangen_time                   | LONG        | The time in milliseconds the query plan was being generated.                                                                                                                                                                                                               |
| queue_time                     | LONG        | The time in milliseconds the query was queued in the system before any processing happened.                                                                                                                                                                                |
| timestamp_optimization_start   | TIMESTAMP   | A timestamp that represents the point in time when optimization of the query started.                                                                                                                                                                                      |
| timestamp_execution_start      | TIMESTAMP   | A timestamp that represents the point in time when execution of the query started.                                                                                                                                                                                         |
| tree_probe_time                | LONG        | The time in milliseconds taken by the VM tree probe during query execution.                                                                                                                                                                                                |
| vm_initialization_time         | LONG        | Time in milliseconds the query was initializing on all participating nodes during execution.                                                                                                                                                                               |
| timestamp_first_byte_sent      | TIMESTAMP   | A timestamp that represents the point in time when the first byte of the result set from the query was returned to the application.                                                                                                                                        |
| timestamp_complete             | TIMESTAMP   | A timestamp that represents the point in time when the execution of the query completed from the client's perspective.                                                                                                                                                     |
| timestamp_execution_complete   | TIMESTAMP   | A timestamp that represents the point in time when the internal execution of the query completed.                                                                                                                                                                          |
| awaiting_client_eof_fetch_time | LONG        | Time in milliseconds that represents the difference between when the client fetched the eof for the query and when the eof was queued internally.                                                                                                                          |
| rows_returned                  | LONG        | The number of rows returned by the query.                                                                                                                                                                                                                                  |
| bytes_returned                 | LONG        | The number of bytes returned by the query.                                                                                                                                                                                                                                 |
| rows_inserted                  | LONG        | The number of rows that the query inserts. This value is NULL when the database does not insert any rows. The value is 0 when the database does not identify any rows to insert from a <code>CREATE TABLE AS SELECT</code> or <code>INSERT AS SELECT SQL</code> statement. |

| Column Name                    | Column Type | Column Description                                                                                                                                                                                                                                                                                                    |
|--------------------------------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rows_deleted                   | LONG        | The number of rows that the query deletes. This value is NULL when the database does not delete any rows. The value is 0 when the database does not identify any rows to delete from a <b>DELETE FROM TABLE SQL</b> statement.                                                                                        |
| update_subquery_execution_time | LONG        | The time in milliseconds taken by the virtual machine (VM) to execute a subquery that identifies the rows for insertion or deletion. This value is NULL when you execute a SQL statement that is not one of these statements: <b>CREATE TABLE AS SELECT</b> , <b>INSERT AS SELECT</b> , or <b>DELETE FROM TABLE</b> . |
| transferred_bytes_per_second   | LONG        | The transfer rate in bytes per second for the result set of the query.                                                                                                                                                                                                                                                |
| code                           | INT         | The SQL code returned at the end of the query.                                                                                                                                                                                                                                                                        |
| state                          | CHAR        | The SQL state returned at the end of the query.                                                                                                                                                                                                                                                                       |
| reason                         | CHAR        | A message associated with the SQL code and SQL state.                                                                                                                                                                                                                                                                 |
| initial_priority               | DOUBLE      | Priority based on the service class, session, and query-level limits.                                                                                                                                                                                                                                                 |
| initial_effective_priority     | DOUBLE      | initial_priority adjusted for total cost and memory usage.                                                                                                                                                                                                                                                            |
| final_effective_priority       | DOUBLE      | Final dynamically adjusted priority.                                                                                                                                                                                                                                                                                  |
| cost_estimate                  | DOUBLE      | The cost estimate of the optimizer.                                                                                                                                                                                                                                                                                   |
| concurrency_service_class_name | CHAR        | Name of the service class this query executes in.                                                                                                                                                                                                                                                                     |
| concurrency_service_class_id   | UUID        | The unique identifier of the service class used for the query.                                                                                                                                                                                                                                                        |
| priority_adjust_factor         | DOUBLE      | The percentage amount by which the database adjusts the priority.                                                                                                                                                                                                                                                     |
| priority_adjust_time           | INT         | How frequently the database adjusts the priority during query execution.                                                                                                                                                                                                                                              |
| cached_query                   | BOOLEAN     | Flag that indicates whether the result was returned from the result set cache.                                                                                                                                                                                                                                        |
| resultset_cached               | BOOLEAN     | Flag that indicates whether the result of the query was stored in the result set cache.                                                                                                                                                                                                                               |
| temp_disk_consumed             | LONG        | Flag that indicates the approximate total disk usage in bytes during query execution.                                                                                                                                                                                                                                 |
| protocol_version               | CHAR        | The version of the client-server protocol for this connection.                                                                                                                                                                                                                                                        |
| client_ip                      | CHAR        | The IP address of the client that executed the query.                                                                                                                                                                                                                                                                 |
| client_name                    | CHAR        | The name of the client that executed the query.                                                                                                                                                                                                                                                                       |

| Column Name                        | Column Type | Column Description                                                                                                                                              |
|------------------------------------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| client_session_id                  | CHAR        | The session id of the client that executed the query.                                                                                                           |
| num_client_threads                 | INT         | The maximum number of client threads determined by the server.                                                                                                  |
| node_id                            | UUID        | The unique identifier of the node that stores the information about this query.                                                                                 |
| participating_nodes                | ARRAY(CHAR) | The nodes that participated in the execution of this query.                                                                                                     |
| ocient_db_version                  | CHAR        | The Ocient database version used to run this query.                                                                                                             |
| ocient_db_commit                   | CHAR        | The Ocient database git commit used to run this query.                                                                                                          |
| ocient_db_dirtied_commit           | CHAR        | The dirtied Ocient database git commit used to run this query.                                                                                                  |
| max_temp_disk_usage                | INT         | The maximum temporary disk usage for this query, expressed as a percentage.                                                                                     |
| max_elapsed_time                   | INT         | The maximum elapsed time for this query, in seconds.                                                                                                            |
| max_rows_returned                  | LONG        | The maximum number of rows this query can return.                                                                                                               |
| num_root_operator_instances        | INT         | The number of created root operator instances.                                                                                                                  |
| approx_system_peak_vm_node_heap    | LONG bytes  | The max observed heap memory in use during execution on a single non-SQL vm node. The memory in use is not necessarily associated with this query.              |
| approx_system_peak_vm_cluster_heap | LONG bytes  | The max observed heap memory in use during execution totaled across all non-SQL vm nodes. The memory in use is not necessarily associated with this query.      |
| approx_system_peak_vm_node_huge    | LONG bytes  | The max observed huge page memory in use during execution on a single non-SQL vm node. The memory in use is not necessarily associated with this query.         |
| approx_system_peak_vm_cluster_huge | LONG bytes  | The max observed huge page memory in use during execution totaled across all non-SQL vm nodes. The memory in use is not necessarily associated with this query. |
| approx_system_peak_sql_node_heap   | LONG bytes  | The max observed heap memory in use during execution on the sql node. The memory in use is not necessarily associated with this query.                          |
| approx_system_peak_sql_node_huge   | LONG bytes  | The max observed huge page memory in use during execution on the sql node. The memory in use is not necessarily associated with this query.                     |
| driver_version                     | CHAR        | The version of the client driver.                                                                                                                               |

| Column Name                       | Column Type | Column Description                                                    |
|-----------------------------------|-------------|-----------------------------------------------------------------------|
| join_order_optimization_time      | LONG        | The time, in milliseconds, the join order of the query was optimized. |
| join_order_optimization_algorithm | CHAR        | The algorithm used for optimizing the join order of the query.        |

sys.metric\_levels

This table contains settings for metric levels.

| Column Name | Column Type | Column Description                                                                               |
|-------------|-------------|--------------------------------------------------------------------------------------------------|
| match       | CHAR        | Metric name match.                                                                               |
| match_type  | CHAR        | The type of match for the name of the metric.                                                    |
| level       | CHAR        | Metric level.                                                                                    |
| node_id     | UUID        | The unique node identifier for this setting. This setting might be NULL for system-wide metrics. |

sys.node\_status

This table contains the status of nodes in the system and the core node monitoring metrics.

| Column Name        | Column Type | Column Description                                                                                                     |
|--------------------|-------------|------------------------------------------------------------------------------------------------------------------------|
| node_id            | UUID        | The Universally Unique IDentifier (UUID) of the node.                                                                  |
| operational_status | CHAR        | The operational status of the node. Values are ACTIVE, STARTING, STOPPING, ERROR, UNKNOWN, or UNREACHABLE.             |
| loadavg_1          | DOUBLE      | Average load on the node over the last 1 minute. Represents the average number of processes in the system run queue.   |
| loadavg_5          | DOUBLE      | Average load on the node over the last 5 minutes. Represents the average number of processes in the system run queue.  |
| loadavg_15         | DOUBLE      | Average load on the node over the last 15 minutes. Represents the average number of processes in the system run queue. |
| max_rss            | LONG        | Maximum resident set size (RSS) of the process on the node in bytes.                                                   |
| heap               | LONG        | Heap memory allocated on the node in bytes.                                                                            |
| software_start     | TIMESTAMP   | Timestamp for the last start of the Ocint software on the node.                                                        |
| system_start       | TIMESTAMP   | Timestamp for the last start of the node hardware.                                                                     |

sys.op\_inst\_debug\_info

This table contains information about the debug information for the operator instance.

| Column Name  | Column Type | Column Description                                                                                 |
|--------------|-------------|----------------------------------------------------------------------------------------------------|
| operator_id  | UUID        | The Universally Unique Identifier (UUID) of the operator                                           |
| node_id      | UUID        | The unique identifier of the node that stores the information about this operator.                 |
| silo_id      | LONG        | The unique identifier of the silo that stores the information about this operator.                 |
| vm_core_id   | LONG        | The unique identifier of the virtual machine core that stores the information about this operator. |
| runtime_id   | INT         | The process-wide unique identifier of this operator instance.                                      |
| key          | CHAR        | The debug information (debugInfo) key.                                                             |
| value        | CHAR        | The debugInfo value.                                                                               |
| op_inst_type | CHAR        | Type of the operator Instance.                                                                     |
| query_id     | UUID        | The UUID of the query that owns the operator.                                                      |

#### sys.osn\_acquisitions

This table contains information about client identifiers for Ownership Storage Number (OSN) acquisitions.

| Column Name      | Column Type | Column Description                                    |
|------------------|-------------|-------------------------------------------------------|
| node_id          | UUID        | The Universally Unique Identifier (UUID) of the node. |
| client_id        | UUID        | The query identifier of the OSN acquisition.          |
| osn              | LONG        | The locked OSN.                                       |
| acquisition_time | TIMESTAMP   | Timestamp that represents when the OSN was acquired.  |

#### sys.plans

This table contains information about the plan for each SQL statement. Not all SQL statements have a corresponding plan (e.g., some DDL statements, SQL statements that have failed parsing or optimization).

| Column Name     | Column Type | Column Description                                                               |
|-----------------|-------------|----------------------------------------------------------------------------------|
| query_id        | UUID        | The unique identifier of the SQL query that executed.                            |
| user            | CHAR        | The name of the user that executed the SQL query.                                |
| database_name   | CHAR        | The name of the database that executes this SQL query.                           |
| timestamp_start | TIMESTAMP   | A timestamp that represents the point in time when the query entered the system. |
| plan            | CHAR        | The plan for the SQL query.                                                      |

sys.queries

This table contains information about queries that the database is currently executing.

| Column Name       | Column Type | Column Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| query_id          | UUID        | A unique identifier of the query that is currently running.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| user              | CHAR        | The name of the user that executes the query.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| database_name     | CHAR        | The name of the database that runs this query.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| database_id       | UUID        | A unique identifier of the database this query is running in.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| status            | CHAR        | The statuses represent how the virtual machine (VM) tracks a specific query. The statuses are: AWAITING_SLOT (The query waits for a service class slot for optimization or execution.), OPTIMIZING (The Ocient System optimizes the query. If the system cannot optimize the query, the query skips this status and changes to the QUEUED status.), QUEUED (The system initializes the query in the VM.), RUNNING (The system executes the query in the VM, either by compiling the result from the Foundation Nodes or by fetching the results from the cache.), and FINALIZING (The query is complete.). |
| sql               | CHAR        | The SQL statement that is executing.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| sql_text_length   | LONG        | The length of the SQL statement.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| referenced_tables | ARRAY(CHAR) | The tables and views referenced by the query.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| total_time        | LONG        | The total time in milliseconds the query has run. This time includes generation_time, optimization_time, and execution_time.                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| generation_time   | LONG        | The time in milliseconds the query was generated or is still generating. This number might increase if generation is in progress.                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| optimization_time | LONG        | The time in milliseconds the optimizer processed or is still processing the query. This number might be 0 if optimization of the query has not started yet. This number might increase if optimization is in progress.                                                                                                                                                                                                                                                                                                                                                                                     |

| Column Name                  | Column Type | Column Description                                                                                                                                                                                                            |
|------------------------------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| execution_time               | LONG        | Time in milliseconds the query executes. This number might be 0 if execution of the query has not started yet. This number might increase if execution of the query is still in progress.                                     |
| timestamp_start              | TIMESTAMP   | A timestamp that represents the point in time when the query entered the system.                                                                                                                                              |
| parsing_time                 | LONG        | Time in milliseconds the query was parsing or is still parsing. This number might be 0 if parsing of the query has not started yet. This number might increase if the query is still parsing.                                 |
| cache_lookup_time            | LONG        | The time in milliseconds the query spent in lookup up matching cached result sets.                                                                                                                                            |
| validation_time              | LONG        | Time in milliseconds the query was validating or is still being validated. This number might be 0 if validation of the query has not started yet. This number might increase if the query is still validating.                |
| plangen_time                 | LONG        | Time in milliseconds the query plan was being generated or is still generating. This number might be 0 if generation of the query plan has not started yet. This number might increase if the query plan is still generating. |
| queue_time                   | LONG        | Time in milliseconds the query was queued or is still queuing. This number might be 0 if queuing of the query has not started yet. This number might increase if the query is still in the queue.                             |
| timestamp_optimization_start | TIMESTAMP   | A timestamp that represents the point in time when optimization of the query started. This value might be 0 if optimization has not started yet.                                                                              |
| timestamp_execution_start    | TIMESTAMP   | A timestamp that represents the point in time when execution of the query started. This value might be 0 if execution has not started yet.                                                                                    |
| tree_probe_time              | LONG        | Time in milliseconds the query was in the VM tree probe stage during execution. This number might be 0 if VM tree probe has not started yet. This number might increase if the probe is still happening.                      |

| Column Name                    | Column Type | Column Description                                                                                                                                                                                                                                                                                                     |
|--------------------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| vm_initialization_time         | LONG        | Time in milliseconds the query was initializing on all participating nodes during execution. This number might be 0 if the VM has not begin initializing the query for execution yet.                                                                                                                                  |
| timestamp_first_byte_sent      | TIMESTAMP   | A timestamp that represents the point in time when the first byte of the result set from the query was returned to the application. This value might be 0 if execution of the query has not started yet or no bytes have been transferred yet.                                                                         |
| rows_returned                  | LONG        | The number of rows returned by the query. This value might be 0 if no rows have been transferred so far.                                                                                                                                                                                                               |
| bytes_returned                 | LONG        | The number of bytes returned by the query. This value might be 0 if no rows have been transferred so far.                                                                                                                                                                                                              |
| rows_inserted                  | LONG        | The number of rows that the query inserts. This value is NULL when the database does not insert any rows. The value is 0 when the database does not identify any rows to insert from a <code>CREATE TABLE AS SELECT</code> or <code>INSERT AS SELECT SQL</code> statement.                                             |
| rows_deleted                   | LONG        | The number of rows that the query deletes. This value is NULL when the database does not delete any rows. The value is 0 when the database does not identify any rows to delete from a <code>DELETE FROM TABLE SQL</code> statement.                                                                                   |
| update_subquery_execution_time | LONG        | The time in milliseconds taken by the VM to execute a subquery that identifies the rows for insertion and deletion. This value is NULL when you execute a SQL statement that is not one of these statements: <code>CREATE TABLE AS SELECT</code> , <code>INSERT AS SELECT</code> , or <code>DELETE FROM TABLE</code> . |
| initial_priority               | DOUBLE      | Priority based on the service class, session, and query level limits.                                                                                                                                                                                                                                                  |
| initial_effective_priority     | DOUBLE      | <code>initial_priority</code> adjusted for total cost and memory usage.                                                                                                                                                                                                                                                |
| effective_priority             | DOUBLE      | Current dynamically adjusted priority.                                                                                                                                                                                                                                                                                 |
| estimated_time                 | INT         | The estimate of the optimizer for the time in milliseconds to process the query.                                                                                                                                                                                                                                       |
| estimated_result_rows          | LONG        | The estimate of the optimizer for the number of rows in the result set.                                                                                                                                                                                                                                                |



| Column Name                     | Column Type | Column Description                                                                                                                                                                       |
|---------------------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| estimated_result_size           | LONG        | The estimate of the optimizer for the size of the result set in bytes.                                                                                                                   |
| concurrency_service_class_name  | CHAR        | Name of the service class this query executes in.                                                                                                                                        |
| concurrency_service_class_id    | UUID        | The unique identifier of the service class used for the query.                                                                                                                           |
| priority_adjust_factor          | DOUBLE      | The percentage amount by which the database adjusts the priority.                                                                                                                        |
| priority_adjust_time            | INT         | How frequently the database adjusts the priority during query execution.                                                                                                                 |
| cached_query                    | BOOLEAN     | Flag that indicates whether the result is returned from the result set cache.                                                                                                            |
| temp_disk_consumed              | LONG        | Flag that indicates the approximate total disk usage in bytes during query execution.                                                                                                    |
| server_ip                       | IPV4        | The IP address of the SQL node that executed the query.                                                                                                                                  |
| protocol_version                | CHAR        | The version of the client-server protocol for this connection.                                                                                                                           |
| driver_version                  | CHAR        | The version of the client driver.                                                                                                                                                        |
| client_ip                       | CHAR        | The IP address of the client that executed the query.                                                                                                                                    |
| client_name                     | CHAR        | The name of the client that executed the query.                                                                                                                                          |
| client_session_id               | CHAR        | The session id of the client that executed the query.                                                                                                                                    |
| num_client_threads              | INT         | The maximum number of client threads determined by the server.                                                                                                                           |
| node_id                         | UUID        | The unique identifier of the node that provided the information about this query.                                                                                                        |
| participating_nodes             | ARRAY(CHAR) | The nodes that are currently known to be participating in the execution of this query. If the query is not in the RUNNING state, this list might not include all nodes that participate. |
| max_temp_disk_usage             | INT         | The maximum temporary disk usage for this query, expressed as a percentage.                                                                                                              |
| max_elapsed_time                | INT         | The maximum elapsed time for this query, in seconds.                                                                                                                                     |
| max_rows_returned               | LONG        | The maximum number of rows this query can return.                                                                                                                                        |
| num_root_operator_instances     | INT         | The number of created root operator instances.                                                                                                                                           |
| approx_system_peak_vm_node_heap | LONG        | The max observed heap memory in use during execution on a single non-SQL vm node. The memory in use is not necessarily associated with this query.                                       |

| Column Name                                          | Column Type | Column Description                                                                                                                                              |
|------------------------------------------------------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| approx_system_peak_vm_cluster_heap_memory_bytes      | LONG        | The max observed heap memory in use during execution totaled across all non-SQL vm nodes. The memory in use is not necessarily associated with this query.      |
| approx_system_peak_vm_node_huge_page_memory_bytes    | LONG        | The max observed huge page memory in use during execution on a single non-SQL vm node. The memory in use is not necessarily associated with this query.         |
| approx_system_peak_vm_cluster_huge_page_memory_bytes | LONG        | The max observed huge page memory in use during execution totaled across all non-SQL vm nodes. The memory in use is not necessarily associated with this query. |
| approx_system_peak_sql_node_heap_memory_bytes        | LONG        | The max observed heap memory in use during execution on the sql node. The memory in use is not necessarily associated with this query.                          |
| approx_system_peak_sql_node_huge_page_memory_bytes   | LONG        | The max observed huge page memory in use during execution on the sql node. The memory in use is not necessarily associated with this query.                     |
| join_order_optimization_time                         | LONG        | The time, in milliseconds, the join order of the query was optimized.                                                                                           |
| join_order_optimization_algorithm                    | CHAR        | The algorithm used for optimizing the join order of the query.                                                                                                  |

## sys.result\_cache

This table contains queries that have cached results in the system.

| Column Name       | Column Type | Column Description                                                                   |
|-------------------|-------------|--------------------------------------------------------------------------------------|
| node_id           | UUID        | The Universally Unique IDentifier (UUID) of the node that holds the cache results.   |
| query_id          | UUID        | The UUID of the query.                                                               |
| database_id       | UUID        | The database of the query.                                                           |
| service_class_id  | UUID        | The UUID of the service class of the query.                                          |
| timestamp         | LONG        | The timestamp in seconds after the UNIX epoch when the result set entered the cache. |
| time              | CHAR        | The human-readable time when the result set entered the cache.                       |
| time_remaining    | LONG        | The amount of time remaining in seconds before the cached result set expires.        |
| referenced_tables | ARRAY(CHAR) | All referenced tables in this query.                                                 |

## sys.scheduled\_tasks

This table contains the details of scheduled tasks in the system.

| Column Name        | Column Type | Column Description                                              |
|--------------------|-------------|-----------------------------------------------------------------|
| id                 | UUID        | The scheduled task identifier.                                  |
| admin_owner_id     | UUID        | The identifier of the node that is managing the scheduled task. |
| task_type          | CHAR        | The type of the task.                                           |
| execution_interval | LONG        | The interval, in milliseconds, between task executions.         |

sys.service\_role\_status

This table contains the status of each service role on the system.

| Column Name       | Column Type | Column Description                                       |
|-------------------|-------------|----------------------------------------------------------|
| node_id           | UUID        | The Universally Unique IDentifier (UUID) of the node.    |
| service_role_id   | UUID        | The UUID of the service role (sys.service_roles).        |
| service_role_name | CHAR        | The human-readable name of the service role on the node. |
| status            | CHAR        | The current status of the role on this node.             |

sys.storage\_device\_metrics

This table contains monitoring metrics collected on each storage device that is controlled by Ocident.

| Column Name | Column Type | Column Description                                                                                                  |
|-------------|-------------|---------------------------------------------------------------------------------------------------------------------|
| id          | CHAR        | Unique and persistent serial number of the drive. This serial number is not a Universally Unique IDentifier (UUID). |
| key         | CHAR        | Name of the measured statistic on the drive.                                                                        |
| value       | CHAR        | Value of the named statistic on the drive.                                                                          |
| transient   | BOOLEAN     | When the value is true, the value indicates that it is transient in nature and is reset when the node is restarted. |
| node_id     | UUID        | The UUID of the node that contains the storage device.                                                              |

sys.storage\_device\_status

This table contains real-time information about the storage devices for each online node.

| Column Name                        | Column Type | Column Description                                                                                                                  |
|------------------------------------|-------------|-------------------------------------------------------------------------------------------------------------------------------------|
| node_id                            | UUID        | The Universally Unique Identifier (UUID) of the node that contains the storage device.                                              |
| device_id                          | INT         | ID of the drive.                                                                                                                    |
| role_name                          | CHAR        | The name of the role or roles active on the node that contains the storage device.                                                  |
| id                                 | CHAR        | Unique and persistent serial number of the drive. This serial number is not a UUID.                                                 |
| pci_address                        | CHAR        | PCI address where the drive is mounted.                                                                                             |
| device_model                       | CHAR        | Model information about the drive.                                                                                                  |
| manufacturer                       | CHAR        | Manufacturer information about the drive.                                                                                           |
| firmware_version                   | CHAR        | Firmware version active on the drive.                                                                                               |
| capacity                           | LONG        | Capacity, in bytes, of the drive.                                                                                                   |
| utilization                        | LONG        | Utilization, in bytes, of the drive.                                                                                                |
| endurance_percentage_used          | DOUBLE      | Value from 0 to 1 that represents an estimate of the percentage of the drive used, as it applies to drive endurance.                |
| device_status                      | CHAR        | Human-readable version of the drive status. Values are ACTIVE, FAILED, or NOT PRESENT.                                              |
| assigned_drive_slots               | ARRAY(INT)  | List of drive slots assigned to this device. The list may be empty if no slot are assigned to a device.                             |
| encryption_drive_locking_supported | BOOLEAN     | Indicates whether encryption drive locking (e.g., OPAL) is enabled on the device.                                                   |
| encryption_drive_locking_enabled   | BOOLEAN     | Indicates whether encryption drive locking (e.g., OPAL) is supported on the device.                                                 |
| encryption_drive_locking_status    | CHAR        | Indicates the status of drive locking (e.g., OPAL): LOCKED, or UNLOCKED. If drive locking is not supported, this value is UNLOCKED. |
| silo_id                            | LONG        | The unique identifier of the silo where this device is located.                                                                     |

## sys.subtasks

This table contains the status and details of subtasks in the system.

| Column Name    | Column Type | Column Description                                |
|----------------|-------------|---------------------------------------------------|
| id             | UUID        | The task identifier.                              |
| name           | CHAR        | The name of the task.                             |
| task_type      | CHAR        | The type of the task.                             |
| execution_type | CHAR        | The type of the task execution.                   |
| location_type  | CHAR        | The type of the location where the task must run. |

| Column Name     | Column Type | Column Description                                         |
|-----------------|-------------|------------------------------------------------------------|
| location_id     | CHAR        | The location where the task must run.                      |
| task_options    | CHAR        | The task arguments.                                        |
| start_time      | TIMESTAMP   | The start time of the task.                                |
| end_time        | TIMESTAMP   | The end time of the task.                                  |
| last_poll_time  | TIMESTAMP   | The last time that the task was polled for a status.       |
| duration        | LONG        | The duration, in milliseconds, of the task.                |
| status          | CHAR        | The current status of the task.                            |
| details         | CHAR        | The details or results of the status for the current task. |
| task_owner_id   | UUID        | The identifier of the node that is running the task.       |
| admin_owner_id  | UUID        | The identifier of the node that is monitoring the task.    |
| parent_task_id  | UUID        | The identifier of the parent task for the current task.    |
| root_task_id    | UUID        | The identifier of the root task for the current task.      |
| database_id     | UUID        | The database identifier.                                   |
| parallelization | CHAR        | The parallelization type of the task.                      |
| state           | BINARY      | The internal state of the task.                            |
| scope           | UUID        | The identifier of the scope for the current task.          |

sys.tasks

The **tasks** view shows the root tasks that have been started in the system.

| Column Name     | Column Type | Column Description                                 |
|-----------------|-------------|----------------------------------------------------|
| id              | UUID        | Task identifier                                    |
| name            | CHAR        | Task name                                          |
| task_type       | CHAR        | Task type                                          |
| execution_type  | CHAR        | Task execution type                                |
| location_type   | CHAR        | Location type where the task must run              |
| location_id     | CHAR        | Location where the task must run                   |
| task_options    | CHAR        | Task arguments                                     |
| start_time      | TIMESTAMP   | Task start time                                    |
| end_time        | TIMESTAMP   | Task end time                                      |
| last_poll_time  | TIMESTAMP   | Last time the task was polled for status           |
| duration        | LONG        | Task duration in milliseconds                      |
| status          | CHAR        | Current task status                                |
| details         | CHAR        | Current task status details or results             |
| task_owner_id   | UUID        | Identifier of the node that is running the task    |
| admin_owner_id  | UUID        | Identifier of the node that is monitoring the task |
| parallelization | CHAR        | Parallelization type of the task                   |
| scope           | UUID        | Identifier of the scope of the task                |

## Network

## sys.channel\_endpoint\_parameters

This table contains details for the endpoints defined on the nodes in the system.

| Column Name    | Column Type | Column Description                                            |
|----------------|-------------|---------------------------------------------------------------|
| id             | UUID        | Universally Unique IDentifier (UUID) of the channel endpoint. |
| node_id        | UUID        | UUID of the node where the endpoint is located (sys.nodes).   |
| name           | CHAR        | Name of the channel endpoint.                                 |
| endpoint_type  | CHAR        | Type of the channel endpoint (FULL or EXTERNAL).              |
| port           | INT         | Port that the channel endpoint is using.                      |
| ip_address     | CHAR        | IP address of the channel endpoint.                           |
| context_type   | CHAR        | Context type of the channel endpoint                          |
| listener_type  | CHAR        | Listener type of the channel endpoint                         |
| initiator_type | CHAR        | Initiator type of the channel endpoint.                       |
| encrypted      | BOOLEAN     | Specifies whether the channel endpoint is encrypted.          |

## sys.network\_interface\_models

This table contains the network devices registered on the system.

| Column Name     | Column Type | Column Description                               |
|-----------------|-------------|--------------------------------------------------|
| id              | UUID        | Universally Unique IDentifier (UUID) of the NIC. |
| manufacturer    | CHAR        | Manufacturer of the NIC.                         |
| model_name      | CHAR        | Model name of the NIC.                           |
| pci_id          | CHAR        | PCI identifier.                                  |
| pci_subsys_id   | CHAR        | PCI Subsystem identifier.                        |
| driver_type     | CHAR        | Driver type of NIC.                              |
| bits_per_second | LONG        | Bits per second.                                 |

## sys.network\_interface\_usage\_types

This table contains the uses of each network device in the system.

| Column Name                | Column Type | Column Description                                                                      |
|----------------------------|-------------|-----------------------------------------------------------------------------------------|
| network_interface_id       | UUID        | Universally Unique IDentifier (UUID) of the network interface (sys.network_interfaces). |
| network_interface_model_id | UUID        | UUID of the network interface model (sys.network_interface_models).                     |
| network_type               | CHAR        | Usage type of NIC (ADMIN, DATA, EXTERNAL, LOCAL_HSI).                                   |

## sys.node\_network\_interfaces

This table contains the network interfaces on each node.

| Column Name                | Column Type | Column Description                                                                                  |
|----------------------------|-------------|-----------------------------------------------------------------------------------------------------|
| network_interface_model_id | UUID        | Universally Unique IDentifier (UUID) of the network interface model (sys.network_interface_models). |
| network_interface_id       | UUID        | UUID of the network interface.                                                                      |
| node_id                    | UUID        | UUID of the node where this interface exists (sys.nodes).                                           |
| interface_type             | CHAR        | Type of interface (PHYSICAL, BOND_SLAVE, BOND_MASTER).                                              |
| address                    | CHAR        | MAC address of the NIC.                                                                             |
| name                       | CHAR        | Name of the NIC.                                                                                    |
| netmask                    | CHAR        | Netmask of the NIC.                                                                                 |
| gateway                    | CHAR        | Gateway of the NIC.                                                                                 |
| pci_address                | CHAR        | PCI address of the NIC.                                                                             |
| master_interface_name      | CHAR        | Master interface name of the NIC.                                                                   |
| bond_type                  | CHAR        | Network bond type of the NIC.                                                                       |

sys.service\_role\_channel\_endpoints

This table contains the channel endpoint parameters that are enabled for each network type and service role.

| Column Name                    | Column Type | Column Description                                                                                 |
|--------------------------------|-------------|----------------------------------------------------------------------------------------------------|
| service_role_id                | UUID        | The Universally Unique IDentifier (UUID) of the service role of this endpoint (sys.service_roles). |
| channel_endpoint_parameters_id | UUID        | UUID of the parameters for this endpoint (sys.channel_endpoint_parameters).                        |
| network_type                   | CHAR        | Network type for this endpoint (ADMIN, DATA, EXTERNAL, LOCAL_HSI).                                 |

Pipelines

sys.pipeline\_errors

While running a pipeline, you might experience errors. These errors are often due to a mismatch between the expectation of the pipeline and the source data. System or networking errors can also occur while the pipeline is running. The sys.pipeline\_errors system catalog table lists extraction, transformation, and pipeline errors.

| Column Name       | Column Type | Column Description                                                                              |
|-------------------|-------------|-------------------------------------------------------------------------------------------------|
| pipeline_id       | UUID        | Universally Unique IDentifier (UUID) of the pipeline (sys.pipelines) that loads this partition. |
| extractor_task_id | UUID        | UUID of the task associated with this pipeline run (sys.subtasks).                              |
| error_index       | LONG        | Error indexes start at 1 and increase monotonically as the Ocint System finds them.             |

| Column Name   | Column Type | Column Description                                                                                                                                                                                                 |
|---------------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| error_type    | CHAR        | The type of error. Values are EXTRACTION, TRANSFORMATION, and PIPELINE_ERROR.                                                                                                                                      |
| error_code    | CHAR        | This 5-character code identifies the type of error.                                                                                                                                                                |
| source_name   | CHAR        | For file loads, this value is the absolute path of the file that includes the S3 bucket. For Kafka loads, the value is the topic name and partition number.                                                        |
| error_message | CHAR        | The error message.                                                                                                                                                                                                 |
| partition_id  | CHAR        | The identifier of the partition. For file loads, the value is the stream source identifier where the file is located. For Apache® Kafka® loads, this value is the partition number.                                |
| record_number | LONG        | The one-based index of the processed record in the source file or Kafka partition defined in the <b>source_name</b> column. This index does not account for headers or skipped lines.                              |
| record_offset | LONG        | The one-based offset in bytes of the record data in the source file for file loads or the Kafka offset for Kafka loads. This offset is -1 for files with non-row-based storage formats (such as Apache® Parquet™). |
| field_index   | INT         | The zero-based index of the expression in the SELECT clause of the CREATE PIPELINE SQL statement.                                                                                                                  |
| column_name   | CHAR        | The name of the column in the target table.                                                                                                                                                                        |
| created_at    | TIMESTAMP   | Timestamp that represents when this error was created.                                                                                                                                                             |

#### sys.pipeline\_events

While your pipeline is running, the pipeline generates events in the sys.pipeline\_events system catalog table to mark significant checkpoints when something has occurred. For details about the lists of files, see the sys.pipeline\_files system catalog table. Pipeline events include messages from many different tasks. These events are the background processes that execute across different Loader Nodes during pipeline operation. For details about tasks, you can query the sys.tasks and sys.subtasks system catalog tables.

| Column Name | Column Type | Column Description                                                    |
|-------------|-------------|-----------------------------------------------------------------------|
| pipeline_id | UUID        | Universally Unique IDentifier (UUID) of the pipeline (sys.pipelines). |
| task_id     | UUID        | UUID of the task associated with this pipeline run (sys.subtasks).    |



| Column Name     | Column Type | Column Description                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| user_id         | UUID        | The user who initiated the event (sys.users).                                                                                                                                                                                                                                                                                                                                                 |
| event_type      | CHAR        | The type of the event. You can generate events with these types: CREATED, REPLACED, RENAMED, STARTED, STOPPING, and STOPPED. The Ocident System generates events with these types automatically: FILE_LISTING_STARTED, FILE_LISTING_COMPLETED, COMPLETED, FAILED, EXTRACTION_STARTED, EXTRACTION_COMPLETED, EXTRACTION_FAILED, SKIPPING_PARTIALLY_LOADED_FILE, and KAFKA_PARTITION_REBALANCE. |
| event_message   | CHAR        | Message that provides more details about the event.                                                                                                                                                                                                                                                                                                                                           |
| event_timestamp | TIMESTAMP   | Timestamp that represents when this event occurred.                                                                                                                                                                                                                                                                                                                                           |

#### sys.pipeline\_files

For file-based loads, the sys.pipeline\_files system catalog table contains one row for each file that shows the status of the file.

| Column Name        | Column Type | Column Description                                                            |
|--------------------|-------------|-------------------------------------------------------------------------------|
| pipeline_id        | UUID        | Universally Unique IDentifier (UUID) of the pipeline (sys.pipelines).         |
| extractor_task_id  | UUID        | The unique identifier of the task requested to load this file (sys.subtasks). |
| name               | CHAR        | Name of the file.                                                             |
| create_timestamp   | TIMESTAMP   | Timestamp that represents when this file is created.                          |
| modified_timestamp | TIMESTAMP   | Timestamp that represents when this file is last modified.                    |
| size               | LONG        | Size of the file.                                                             |

| Column Name      | Column Type | Column Description                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|------------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| status           | CHAR        | The status of the file that updates throughout the process of a pipeline load. Values are: PENDING (listed file but not assigned to an extractor task), QUEUED (assigned to an extractor task but not confirmed the existence of the file), LOADING (confirmed existence of the file and began loading), LOADED (loaded data with complete success), LOADED_WITH_ERRORS (loaded data with at least one error), FAILED (failed to load), and SKIPPED (skipped loading the file). |
| stream_source_id | CHAR        | The unique identifier of the associated partition stream source.                                                                                                                                                                                                                                                                                                                                                                                                                |

## sys.pipeline\_functions

This table contains all of the pipeline functions in the system.

| Column Name          | Column Type    | Column Description                                                                        |
|----------------------|----------------|-------------------------------------------------------------------------------------------|
| id                   | UUID           | The Universally Unique IDentifier (UUID) of the pipeline function.                        |
| name                 | CHAR           | The name of the pipeline function.                                                        |
| database_id          | UUID           | The UUID of the database for this pipeline function (sys.databases).                      |
| language             | CHAR           | The language used to write the pipeline function.                                         |
| return_type          | CHAR           | The return type of the pipeline function.                                                 |
| return_type_nullable | BOOLEAN        | The nullable property of the return type for the pipeline function.                       |
| definition           | CHAR           | The definition of the pipeline function.                                                  |
| creator_id           | UUID           | The UUID of the user or group who created the pipeline function (sys.users/sys.groups).   |
| created_at           | TIMESTAMP      | The timestamp when this pipeline function was created.                                    |
| altered_at           | TIMESTAMP      | The timestamp when this pipeline function was last updated.                               |
| argument_names       | ARRAY(CHAR)    | The argument names of the pipeline function, specified as an array.                       |
| argument_types       | ARRAY(CHAR)    | The argument types of the pipeline function, specified as an array.                       |
| argument_nullability | ARRAY(BOOLEAN) | Whether the argument types for the pipeline function are nullable, specified as an array. |
| imported_libraries   | ARRAY(CHAR)    | The import statements for the pipeline function, specified as an array.                   |

`sys.pipeline_metrics`

Find metrics that measure the performance and activity of a pipeline in the `sys.pipeline_metrics` view. This view contains one row for each metric at a specified point in time. Metrics are either instantaneous or incremental. Instantaneous metrics reflect the current value of a counter at the time of the metric collection. Incremental metrics increase over time and reflect the cumulative value for a specified counter. The first snapshot of a metric appears when it is initialized. Afterward, the system updates the metric every 10 seconds. Each metric has a scope that defines its uniqueness. Do not aggregate metrics across different scopes. The scope columns are `pipeline_id`, `extractor_task_id`, `partition_id`, and `sink_index`. If the value of any of these columns is NULL, the scope applies to all values in that dimension.

| Column Name                    | Column Type | Column Description                                                                                                                                         |
|--------------------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pipeline_id</code>       | UUID        | UUID of the pipeline ( <code>sys.pipelines</code> ) loading this partition.                                                                                |
| <code>extractor_task_id</code> | UUID        | UUID of the task associated with this pipeline run ( <code>sys.subtasks</code> ).                                                                          |
| <code>partition_id</code>      | CHAR        | The partition identifier. For file-based loads, this identifier is the stream source identifier. For Kafka loads, this identifier is the partition number. |
| <code>sink_index</code>        | LONG        | The sink index. This number is always NULL or 0.                                                                                                           |
| <code>name</code>              | CHAR        | The name of the metric.                                                                                                                                    |
| <code>value</code>             | LONG        | The value of the metric.                                                                                                                                   |
| <code>updated_at</code>        | TIMESTAMP   | Timestamp that represents when this metric was updated.                                                                                                    |

`sys.pipeline_metrics_info`

Find the full list of available metrics in the `sys.pipeline_metrics_info` built-in view. This static view provides metric units, type, and descriptions.

| Column Name              | Column Type | Column Description                                       |
|--------------------------|-------------|----------------------------------------------------------|
| <code>name</code>        | CHAR        | Name of the metric.                                      |
| <code>units</code>       | CHAR        | Units of the metric.                                     |
| <code>metric_type</code> | CHAR        | Type of the metric. One of INCREMENTAL or INSTANTANEOUS. |
| <code>description</code> | CHAR        | Description of the metric.                               |

`sys.pipeline_partitions`

For Apache® Kafka® partition-based loads, the `sys.pipeline_partitions` system catalog table contains one row for each partition, which contains the current offsets and record counts.

| Column Name                    | Column Type | Column Description                                                                                          |
|--------------------------------|-------------|-------------------------------------------------------------------------------------------------------------|
| <code>pipeline_id</code>       | UUID        | Universally Unique IDentifier (UUID) of the pipeline ( <code>sys.pipelines</code> ) loading this partition. |
| <code>extractor_task_id</code> | UUID        | UUID of the task associated with this pipeline run ( <code>sys.subtasks</code> ).                           |

| Column Name                   | Column Type | Column Description                                                                                                                                                    |
|-------------------------------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| group_id                      | CHAR        | The group identifier (group.id) of the pipeline that loads this partition.                                                                                            |
| topic_name                    | CHAR        | The name of the Kafka topic where this partition belongs.                                                                                                             |
| partition_number              | LONG        | The partition number on the Kafka topic.                                                                                                                              |
| max_offset                    | LONG        | The current maximum offset on this partition.                                                                                                                         |
| last_committed_durable_offset | LONG        | The last committed offset on the partition that was indicated as durable by the Ocient System. This offset might lag slightly behind the actual durability of tables. |
| lag                           | INT         | The number of records yet to be processed in a partition, calculated as the difference between the maximum offset and the last committed durable offset.              |
| records_processed             | INT         | The number of records processed (but not necessarily made durable) for this partition.                                                                                |
| records_failed                | INT         | The number of failed records for this partition.                                                                                                                      |
| updated_at                    | TIMESTAMP   | The timestamp when the Ocient System last updated the details of this partition.                                                                                      |

#### sys.pipeline\_tables

This table contains all of the table identifiers currently associated with pipeline objects in the system.

| Column Name | Column Type | Column Description                                                    |
|-------------|-------------|-----------------------------------------------------------------------|
| pipeline_id | UUID        | Universally Unique IDentifier (UUID) of the pipeline (sys.pipelines). |
| table_id    | UUID        | UUID of the table (sys.tables).                                       |

#### sys.pipeline\_tasks

The Ocient System implements the internals of a pipeline using distributed tasks. Each pipeline has a unique pipeline identifier `pipeline_id`. Every time that a `START PIPELINE` SQL statement is executed, the Ocient System generates a `run_pipeline` task that serves as the main coordinator task. This task automatically partitions the source data into smaller units and creates a `run_extractor` task for each one. Then, the Ocient System sends each `run_extractor` task and executes it on a Loader Node. In the `sys.pipeline_tasks` system catalog table, you can see the `run_pipeline` and `run_extractor` tasks.

| Column Name    | Column Type | Column Description                                    |
|----------------|-------------|-------------------------------------------------------|
| pipeline_id    | UUID        | Universally Unique IDentifier (UUID) of the pipeline. |
| task_id        | UUID        | Task identifier                                       |
| execution_type | CHAR        | The type of the task.                                 |

| Column Name    | Column Type | Column Description                                         |
|----------------|-------------|------------------------------------------------------------|
| parent_task_id | UUID        | Identifier of the parent task of the task                  |
| location_type  | CHAR        | The type of location where the task must run.              |
| location_id    | CHAR        | The identifier for the location where the task must run.   |
| task_owner_id  | UUID        | Identifier of the node that is running the task            |
| admin_owner_id | UUID        | Identifier of the node that is monitoring the task         |
| status         | CHAR        | Current task status                                        |
| details        | CHAR        | The details or results of the status for the current task. |
| start_time     | TIMESTAMP   | The start time of the task.                                |
| end_time       | TIMESTAMP   | The end time of the task.                                  |
| duration       | LONG        | Task duration in milliseconds                              |

## sys.pipelines

This table contains all of the pipelines in the system.

| Column Name         | Column Type | Column Description                                                                        |
|---------------------|-------------|-------------------------------------------------------------------------------------------|
| id                  | UUID        | Universally Unique IDentifier (UUID) of the pipeline.                                     |
| name                | CHAR        | Name of the pipeline.                                                                     |
| database_id         | UUID        | UUID of the database of the pipeline (sys.databases).                                     |
| loading_mode        | CHAR        | Specifies whether the pipeline runs in a one-time (BATCH) or continuous (CONTINUOUS) way. |
| source_type         | CHAR        | Source of the data (S3, KAFKA, FILESYSTEM).                                               |
| data_format         | CHAR        | Structure of the data (DELIMITED, CSV, JSON).                                             |
| created_at          | TIMESTAMP   | Timestamp that indicates when this pipeline was created.                                  |
| altered_at          | TIMESTAMP   | Timestamp that indicates when this pipeline was last updated.                             |
| creator_id          | UUID        | The UUID of the user or group who created the pipeline (sys.users/sys.groups).            |
| status              | CHAR        | Status of the pipeline (RUNNING, STOPPED, COMPLETED, FAILED).                             |
| status_message      | CHAR        | Status message of the pipeline.                                                           |
| task_id             | UUID        | Universally Unique IDentifier (UUID) of the task (sys.tasks).                             |
| pending_files_count | LONG        | The number of files with the PENDING status (sys.pipeline_files).                         |
| queued_files_count  | LONG        | The number of files with the QUEUED status (sys.pipeline_files).                          |
| loading_files_count | LONG        | The number of files with the LOADING status (sys.pipeline_files).                         |

| Column Name                    | Column Type | Column Description                                                           |
|--------------------------------|-------------|------------------------------------------------------------------------------|
| loaded_files_count             | LONG        | The number of files with the LOADED status (sys.pipeline_files).             |
| loaded_with_errors_files_count | LONG        | The number of files with the LOADED_WITH_ERRORS status (sys.pipeline_files). |
| failed_files_count             | LONG        | The number of files with the FAILED status (sys.pipeline_files).             |
| skipped_files_count            | LONG        | The number of files with the SKIPPED status (sys.pipeline_files).            |

## Security

sys.security\_settings

This table contains security settings.

| Column Name                    | Column Type | Column Description                                                         |
|--------------------------------|-------------|----------------------------------------------------------------------------|
| database_id                    | UUID        | The database for this setting. This might be NULL for system-wide settings |
| group_id                       | UUID        | The group for this setting when the setting is for a group                 |
| password_minimum_length        | INT         | Minimum password length.                                                   |
| password_complexity_level      | INT         | Password complexity requirement.                                           |
| password_lifetime_days         | INT         | Password lifetime in days.                                                 |
| password_no_repeat_count       | INT         | Password repetition count.                                                 |
| password_invalid_attempt_limit | INT         | Invalid password attempt limit.                                            |

## Security Integrations

sys.oidc\_integrations

This table enumerates the configuration for OpenID Connect Single Sign-On integrations (i.e. OKTA®). You can modify this configuration using the ALTER DATABASE :::Iframe{code=""}

::: ALTER SSO INTEGRATION oidc DDL statement where :::Iframe{code=""}

::: is the name of your database.

| Column Name             | Column Type | Column Description                                                                       |
|-------------------------|-------------|------------------------------------------------------------------------------------------|
| security_integration_id | UUID        | The id of this OpenID Connect (OIDC) Single Sign-On integration.                         |
| database_id             | UUID        | The id of the database using this Single Sign-On authorization to this integration.      |
| disabled                | BOOLEAN     | When true, all new connections made using this Single Sign-On integration will fail.     |
| default_group           | CHAR        | The group which all users of this Single Sign-On intergration are granted membership to. |

| Column Name                    | Column Type | Column Description                                                                                                                                                       |
|--------------------------------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| issuer                         | CHAR        | The complete URL for the OAuth 2.0 / OpenID Connect Authorization Server. This is the expected “iss” claim in an access token.                                           |
| client_id                      | CHAR        | The provider-supplied ID of this Single Sign-On integration.                                                                                                             |
| public_client                  | BOOLEAN     | When true, PKCE is used in place of a client authentication method                                                                                                       |
| enable_id_token_authentication | BOOLEAN     | When true, a fat ID Token is sufficient to connect to the database.                                                                                                      |
| user_id_claims                 | ARRAY(CHAR) | The id token claim(s) used to identify users in audit trails.                                                                                                            |
| additional_scopes              | ARRAY(CHAR) | Request scopes for authorization requests.                                                                                                                               |
| additional_audiences           | ARRAY(CHAR) | Additional token audience to accept when validating tokens. This is useful for Authorization Servers without a token exchange capability.                                |
| groups_claim_ids               | ARRAY(CHAR) | The token claims that can be used to map the user to a Database group. You must provide <code>groups_claim_ids</code> if <code>groups_claim_mappings</code> is provided. |
| groups_claim_mappings          | CHAR        | The mappings from Provider group => Database group.                                                                                                                      |
| roles_claim_ids                | ARRAY(CHAR) | The token claims that can be used to map the user to a Database role. You must provide <code>roles_claim_ids</code> if <code>roles_claim_mappings</code> is provided.    |
| roles_claim_mappings           | CHAR        | The mappings from Provider role => Database role.                                                                                                                        |
| blocked_groups                 | ARRAY(CHAR) | Provider groups that are restricted from connecting to the database.                                                                                                     |
| blocked_roles                  | ARRAY(CHAR) | Provider roles that are restricted from connecting to the database.                                                                                                      |
| enable_debug_flow              | BOOLEAN     |                                                                                                                                                                          |
| request_scopes                 | ARRAY(CHAR) | All request scopes for authorization requests.                                                                                                                           |
| scopes_on_refresh              | BOOLEAN     | Include the scope list in the initial refresh-token request                                                                                                              |

#### sys.oidc\_sessions

The sys.oidc\_sessions table contains information about active database connections established using an OpenID Connect (oidc) Single Sign-On integration.

| Column Name | Column Type | Column Description                                 |
|-------------|-------------|----------------------------------------------------|
| id          | UUID        | The unique identifier for this session.            |
| user        | CHAR        | The Fully Qualified User Name (FQUN) of this user. |

| Column Name             | Column Type | Column Description                                                              |
|-------------------------|-------------|---------------------------------------------------------------------------------|
| database_id             | UUID        | The id of the database associated with this session.                            |
| security_integration_id | UUID        | The id of the OpenID Connect (OIDC) Single Sign-On integration that authorized. |
| has_access_token        | BOOLEAN     | True if the identity provider supplied an Access Token.                         |
| has_refresh_token       | BOOLEAN     | True if the identity provider supplied a Refresh Token.                         |
| compact_id_token        | CHAR        | The JWS compact serialization form of the ID Token.                             |
| id_token_json           | CHAR        | The ID Token claims.                                                            |

## sys.security\_integrations

This table contains the OIDC security integrations installed on the system.

| Column Name          | Column Type | Column Description                                                                           |
|----------------------|-------------|----------------------------------------------------------------------------------------------|
| id                   | UUID        | UUID of the OIDC security integration.                                                       |
| database_id          | UUID        | UUID of the database associated with this security integration (sys.databases).              |
| integration_type     | CHAR        | The type of the security integration (SSO, SCIM, LDAP, etc.).                                |
| table_descriptor     | CHAR        | An indirect pointer to the table where you can view the objects of this integration.         |
| sso_integration_name | CHAR        | SSO integration name, if the security integration is of type SSO                             |
| is_database_default  | BOOLEAN     | When true, the security integration is used as the default SSO integration for the database. |

## sys.sessions

This table contains information about active client connections.

| Column Name  | Column Type | Column Description                                                                                                                                                                                                    |
|--------------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| id           | UUID        | The unique identifier for this session.                                                                                                                                                                               |
| user         | CHAR        | The Fully Qualified User Name (FQUN) of this user.                                                                                                                                                                    |
| sso_protocol | CHAR        | The Single Sign-On (SSO) protocol used to authorize the connection request. Null if basic password authentication was used. Tables exists for each SSO protocol using the following naming scheme: “:::Iframe{code=}” |

“:::sessions”. For example, all sessions created using the OpenID Connect (oidc) protocol will be listed in the “sys.oidc\_sessions” table. | | database\_id | UUID | The id of the database associated with this session. | |



default\_schema | CHAR | The default schema used for queries submitted by this user. | | created\_at\_timestamp | LONG | The number of seconds from epoch the session was established. | | duration | LONG | The number of seconds since the session was established. | | expires\_at\_timestamp | LONG | The number of seconds from epoch the session will expire. | | is\_renewable | BOOLEAN | When true, the database will attempt to extend the session when it expires. | | client\_ip | CHAR | The IP address of the client. | | client\_name | CHAR | The human-readable name of the client. | | client\_session\_id | CHAR | An opaque identifier provided by client implementations. Used to correlate events across application boundaries. | | protocol\_version | CHAR | The version of the client-server protocol for this connection. | | driver\_version | CHAR | The version of the client driver. | | tls\_enabled | BOOLEAN | True if TLS is enabled for this client connection. | | node\_id | UUID | The unique identifier of the node the client is connected to. | | connectivity\_pool\_id | UUID | The unique identifier of the connectivity pool the client is connected on. | | group\_names | CHAR | Names of all groups in the session. | | role\_names | CHAR | Names of all roles in the session. | | active\_service\_class\_names | CHAR | Names of all active service classes in the session. | | inactive\_service\_class\_names | CHAR | Names of all inactive service classes in the session. |

#### Statistics

##### sys.average\_bb\_sizes

This table contains the average size of bounding boxes in the database for geospatial data types.

| Column Name | Column Type | Column Description                                              |
|-------------|-------------|-----------------------------------------------------------------|
| table_id    | UUID        | Universally Unique IDentifier (UUID) of the table (sys.tables). |
| column_id   | UUID        | UUID of the column (sys.columns).                               |
| avg_width   | DOUBLE      | Average width of bounding boxes in this column.                 |
| avg_height  | DOUBLE      | Average height of bounding boxes in this column.                |

##### sys.average\_column\_sizes

This table contains the average size of each column in the database.

| Column Name | Column Type | Column Description                                              |
|-------------|-------------|-----------------------------------------------------------------|
| table_id    | UUID        | Universally Unique IDentifier (UUID) of the table (sys.tables). |
| column_id   | UUID        | UUID of the column (sys.columns).                               |
| size        | DOUBLE      | Average size of the column.                                     |

##### sys.column\_cardinalities

This table contains estimates of the number of unique values in each column in the database. These estimates might not reflect recent loading or deletion activity.

| Column Name | Column Type | Column Description                                               |
|-------------|-------------|------------------------------------------------------------------|
| table_id    | UUID        | Universally Unique IDentifier (UUID) of the table (sys.tables).  |
| column_id   | UUID        | UUID of the column (sys.columns).                                |
| cardinality | LONG        | Estimate of the number of unique values in this column.          |
| stats_type  | CHAR        | Represents the type of values, either 'COLUMN' or 'INNER_ARRAY'. |

##### sys.column\_distributions

This table contains some statistical characteristics of the calculated Kernel Density Distribution of data in the database columns.

| Column Name   | Column Type | Column Description                                                                  |
|---------------|-------------|-------------------------------------------------------------------------------------|
| table_id      | UUID        | Universally Unique IDentifier (UUID) of the table (sys.tables).                     |
| column_id     | UUID        | UUID of the column (sys.columns).                                                   |
| bandwidth     | DOUBLE      | Smoothing parameter for the distribution.                                           |
| minimum_value | DOUBLE      | Minimum value found in the distribution.                                            |
| maximum_value | DOUBLE      | Maximum value found in the distribution.                                            |
| stats_type    | CHAR        | Represents the type of values in the distribution, either 'TABLE' or 'INNER_ARRAY'. |

#### sys.columns\_compression\_info

This table contains column-level compression statistics for fixed-length columns. The system calculates statistics from the column data across all segments.

| Column Name             | Column Type | Column Description                                                                                             |
|-------------------------|-------------|----------------------------------------------------------------------------------------------------------------|
| node_id                 | UUID        | Universally Unique IDentifier (UUID) of the node that owns the segments included in these statistics.          |
| table_id                | UUID        | UUID of the table of this column (sys.tables).                                                                 |
| ordinal                 | INT         | Ordinal of the column.                                                                                         |
| is_deltadelta_enabled   | BOOLEAN     | Specifies whether deltadelta compression is enabled.                                                           |
| is_rle_enabled          | BOOLEAN     | Specifies whether RLE compression scheme is enabled.                                                           |
| is_ne_enabled           | BOOLEAN     | Specifies whether the NULL elimination compression scheme is enabled.                                          |
| raw_size                | LONG        | Size of the data in bytes before compression.                                                                  |
| compressed_size         | LONG        | Size of the data in bytes after compression.                                                                   |
| num_deltadelta_blocks   | LONG        | Number of data blocks compressed using the deltadelta compression.                                             |
| num_rle_blocks          | LONG        | Number of data blocks compressed using the RLE (Run-Length Encoding) compression scheme.                       |
| num_nerle_blocks        | LONG        | Number of data blocks compressed using a combination of the RLE and NE (NULL Elimination) compression schemes. |
| num_uncompressed_blocks | LONG        | Number of uncompressed data blocks.                                                                            |
| num_total_blocks        | LONG        | Total number of data blocks.                                                                                   |
| id                      | UUID        | UUID of the column                                                                                             |

| Column Name | Column Type | Column Description |
|-------------|-------------|--------------------|
| name        | CHAR        | Name of the column |

#### sys.segments\_compression\_info

This table contains segment-level compression statistics. The system calculates statistics from the data within the specified segment. All block statistics apply only to fixed-length columns within the segment. Non-block specific statistics (e.g., size information) include both fixed and variable-length columns.

| Column Name               | Column Type | Column Description                                                                                        |
|---------------------------|-------------|-----------------------------------------------------------------------------------------------------------|
| table_id                  | UUID        | Universally Unique Identifier (UUID) of the table relevant to this segment (sys.tables).                  |
| segment_id                | UUID        | UUID for the segment.                                                                                     |
| num_deltadelta_blocks     | LONG        | Number of data blocks compressed using deltadelta compression                                             |
| num_rle_blocks            | LONG        | Number of data blocks compressed using RLE (Run-Length Encoding) compression scheme                       |
| num_nerle_blocks          | LONG        | Number of data blocks compressed using a combination of RLE and NE (Null Elimination) compression schemes |
| num_uncompressed_blocks   | LONG        | Number of uncompressed data blocks                                                                        |
| num_total_blocks          | LONG        | Total number of data blocks                                                                               |
| raw_size                  | LONG        | Size of the data part before any compression and without any parity data                                  |
| compressed_size           | LONG        | Size of the data part after compression and without any parity data + Size of the manifest part           |
| compressed_data_part_size | LONG        | Size of the data part after compression and without any parity data                                       |
| num_cluster_keys          | LONG        | Number of unique cluster keys in the segment                                                              |
| num_time_buckets          | LONG        | Number of time buckets in the segment                                                                     |

#### sys.stats\_files

This table contains information about the on-disk cache for each Foundation Node of the local table probability density functions (PDFs) and array PDFs.

| Column Name | Column Type | Column Description                                                                |
|-------------|-------------|-----------------------------------------------------------------------------------|
| node_id     | UUID        | The Universally Unique Identifier (UUID) of the node. References id in sys.nodes. |
| table_id    | UUID        | The UUID of the table. References id in sys.tables.                               |
| file_id     | UUID        | The UUID of the file.                                                             |

| Column Name        | Column Type | Column Description                                                                                                                                             |
|--------------------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| file_type          | CHAR        | The type of this file. Values are ARRAY_PDF, TABLE_PDF, or TABLE_CDE.                                                                                          |
| file_size          | LONG        | Size in bytes of the file.                                                                                                                                     |
| column_ordinal     | LONG        | Ordinal of the column where the stats file applies. The ordinal references sys.columns for the specified table. The value is NULL when file_type is TABLE_PDF. |
| updated_at         | TIMESTAMP   | Timestamp of the last update to this file.                                                                                                                     |
| is_stale           | BOOLEAN     | Whether any rows are present on disk that were not present when this stats file was last updated                                                               |
| rows_computed_from | LONG        | Number of rows present on the node when this stats file was last updated                                                                                       |
| marked_stale_at    | TIMESTAMP   | Timestamp of this file being marked stale.                                                                                                                     |

#### sys.table\_cardinalities

This table contains estimates of the number of rows in the individual tables in the database. These estimates might not reflect recent loading or deletion activity.

| Column Name | Column Type | Column Description                                              |
|-------------|-------------|-----------------------------------------------------------------|
| table_id    | UUID        | Universally Unique IDentifier (UUID) of the table (sys.tables). |
| cardinality | LONG        | Estimate of the number of rows in this table.                   |

#### sys.vl\_columns\_compression\_info

This table contains column-level compression statistics for variable-length columns. The system calculates statistics from the column data across all segments.

| Column Name            | Column Type | Column Description                                                                                    |
|------------------------|-------------|-------------------------------------------------------------------------------------------------------|
| node_id                | UUID        | Universally Unique IDentifier (UUID) of the node that owns the segments included in these statistics. |
| table_id               | UUID        | UUID of the table (sys.tables).                                                                       |
| ordinal                | INT         | Ordinal of the column.                                                                                |
| is_compression_enabled | BOOLEAN     | Whether compression is enabled.                                                                       |
| raw_size               | LONG        | Size of the data in bytes before compression.                                                         |
| compressed_size        | LONG        | Size of the data in bytes after compression.                                                          |
| num_lz4_rows           | LONG        | Number of lz4 compressed rows.                                                                        |
| num_uncompressed_rows  | LONG        | Number of uncompressed rows.                                                                          |
| num_null_rows          | LONG        | Number of NULL rows.                                                                                  |
| num_total_rows         | LONG        | Total number of rows.                                                                                 |
| id                     | UUID        | UUID of the column                                                                                    |

| Column Name | Column Type | Column Description |
|-------------|-------------|--------------------|
| name        | CHAR        | Name of the column |

Storage

sys.addendum\_directories

This table contains all addendum directories with details about their associated directories and metadata.

| Column Name                       | Column Type | Column Description                                                                                                                                   |
|-----------------------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| id                                | UUID        | The identifier of the addendum directory within the storage cluster state.                                                                           |
| associated_directory_group_id     | CHAR        | The group identifier of the associated directory.                                                                                                    |
| associated_directory_ida_offset   | INT         | The IDA offset (the logical segment in a segment directory group) of the associated directory.                                                       |
| parent_segment_group_id           | CHAR        | The segment group identifier of the associated parent segment (sys.segment_groups).                                                                  |
| parent_ida_offset                 | INT         | The IDA offset of the associated parent segment.                                                                                                     |
| segment_part_inventory_name       | UUID        | The name of the segment part inventory for the addendum part. This name references the id column of the segment_part_inventory system catalog table. |
| segment_part_inventory_storage_id | CHAR        | The storage identifier of the segment part inventory.                                                                                                |
| num_deleted_rows                  | LONG        | If the segment part inventory houses a delete part, this number is the number of rows subtracted from the parent segment.                            |
| is_replica_for_ida_offset         | INT         | The IDA offset of the parent segment that this segment part inventory replicates.                                                                    |

sys.data\_usage\_by\_file\_type\_and\_table

This table contains information about the amount of queryable data by file type.

| Column Name          | Column Type | Column Description                                                               |
|----------------------|-------------|----------------------------------------------------------------------------------|
| segment_type         | CHAR        | The type of the segment.                                                         |
| table_name           | CHAR        | The name for the table.                                                          |
| num_segments         | LONG        | The number of segments of a specific segment type.                               |
| total_rows           | LONG        | The total number of rows for segments of a specific segment type.                |
| avg_rows_per_segment | DOUBLE      | The average amount of rows per segment for a segment of a specific segment type. |

| Column Name   | Column Type | Column Description                                                       |
|---------------|-------------|--------------------------------------------------------------------------|
| total_size_gb | DOUBLE      | The total size in gigabytes for all segments of a specific segment type. |
| avg_size_mb   | DOUBLE      | The average size in megabytes for segments of a specific segment type.   |

#### sys.degraded\_segment\_groups

This table contains information about degraded segment groups and the clusters where they are located.

| Column Name             | Column Type | Column Description                                       |
|-------------------------|-------------|----------------------------------------------------------|
| cluster_id              | UUID        | The unique identifier of the cluster.                    |
| cluster_name            | CHAR        | The name of the cluster.                                 |
| segment_group_status    | CHAR        | The segment group status.                                |
| degraded_segment_groups | LONG        | The total number of damaged segment groups in a cluster. |

#### sys.degraded\_segment\_groups\_by\_table

This table contains information about degraded segment groups in addition to the clusters and tables where they are located.

| Column Name             | Column Type | Column Description                                       |
|-------------------------|-------------|----------------------------------------------------------|
| cluster_id              | UUID        | The unique identifier of the cluster.                    |
| cluster_name            | CHAR        | The name of the cluster.                                 |
| segment_group_status    | CHAR        | The segment group status.                                |
| table_name              | CHAR        | The table name.                                          |
| degraded_segment_groups | LONG        | The total number of damaged segment groups in a cluster. |

#### sys.effective\_merge\_policy\_per\_table

This table computes and displays the effective merge policy (enabled or disabled) for each table in the system. The system bases the policy on the default configuration and any overrides that are present for each table.

| Column Name                  | Column Type | Column Description                                                                                                                                                         |
|------------------------------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| id                           | UUID        | Universally Unique IDentifier (UUID) of the table (sys.tables).                                                                                                            |
| schema                       | CHAR        | Name of the schema.                                                                                                                                                        |
| name                         | CHAR        | Name of the table.                                                                                                                                                         |
| table_lts_property_string    | CHAR        | Additional properties assigned to control the Foundation role (formerly “LTS role”) behavior for the table.                                                                |
| effective_table_merge_policy | CHAR        | The effective merge policy applied to this table based on the value of the global <b>feature_directory_merge</b> feature flag and an optional table merge policy override. |

#### sys.nodes\_with\_unhealthy\_stored\_segments

This table contains information about nodes with damaged, missing, or invalid segments.

| Column Name            | Column Type | Column Description                                                             |
|------------------------|-------------|--------------------------------------------------------------------------------|
| node_id                | UUID        | The unique identifier of the node.                                             |
| node_name              | CHAR        | The name of the node.                                                          |
| status                 | CHAR        | The status of the stored segment. Values include INTACT, MISSING, and DAMAGED. |
| num_unhealthy_segments | LONG        | The total number of unhealthy segments in the node.                            |

sys.orphaned\_segments

This table contains information about orphaned segments in the cluster.

| Column Name  | Column Type | Column Description                                                      |
|--------------|-------------|-------------------------------------------------------------------------|
| storage_id   | UUID        | The Universally Unique Identifier (UUID) for the orphaned segment.      |
| owner        | LONG        | The identifier of the owner for this orphaned segment.                  |
| segment_type | CHAR        | The type of the segment (TKT, PAGE, etc.).                              |
| node_id      | UUID        | The UUID for this setting, which might be NULL for system-wide metrics. |

sys.segment\_directories

This table contains all segment directories with details about parents, associated buckets, and metadata.

| Column Name             | Column Type | Column Description                                                                         |
|-------------------------|-------------|--------------------------------------------------------------------------------------------|
| segment_group_id        | CHAR        | The identifier of the segment group (sys.segment_groups).                                  |
| ida_offset              | INT         | The unique index of the segment.                                                           |
| storage_id              | UUID        | The storage identifier of the segment directory.                                           |
| owner                   | LONG        | The owner identifier of this stored segment.                                               |
| table_id                | UUID        | Universally Unique Identifier (UUID) of the table (sys.tables).                            |
| begin_time              | LONG        | Start time of the bucket of this segment directory in the time bucket column of its table. |
| end_time                | LONG        | End time of the bucket of this segment directory in the time bucket column of its table.   |
| depth                   | INT         | The depth of the segment group. A non-zero value indicates a segment directory group.      |
| parent_segment_group_id | CHAR        | The parent identifier of the segment directory group (sys.segment_groups).                 |

| Column Name       | Column Type | Column Description                                                                                                                                                                                                                                               |
|-------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| parent_ida_offset | INT         | The unique index of the parent segment, if it exists.                                                                                                                                                                                                            |
| parent_storage_id | UUID        | The storage identifier of the parent segment directory, if it exists.                                                                                                                                                                                            |
| parent_owner      | LONG        | The owner of the parent storage identifier. A zero value indicates an unowned storage identifier. Whereas a non-zero value indicates a segment owned after rebuild.                                                                                              |
| row_count         | LONG        | If the directory depth equals 1, then this count is the number of rows in the base leaf segment (not factoring in any DELETE SQL statements). If the depth is greater than 1, then this count is the total number of rows spanned by the intermediate directory. |

#### sys.segment\_group\_transfers

This table contains all segment group transfers between clusters.

| Column Name       | Column Type | Column Description                                                                                                       |
|-------------------|-------------|--------------------------------------------------------------------------------------------------------------------------|
| id                | UUID        | Universally Unique Identifier (UUID) of the segment group transfer.                                                      |
| src_cluster_id    | UUID        | UUID of the source cluster (sys.clusters).                                                                               |
| dst_cluster_id    | UUID        | UUID of the destination cluster (sys.clusters).                                                                          |
| status            | CHAR        | The status of the transfer.                                                                                              |
| segment_group_ids | ARRAY(CHAR) | List of the identifiers of segment groups in this transfer (sys.segment_groups).                                         |
| src_committed_osn | CHAR        | The Ownership Storage Number (OSN) in which the segment groups are considered fully transferred from the source cluster. |
| dst_committed_osn | CHAR        | The OSN in which the segment groups are considered fully transferred from the destination cluster.                       |

#### sys.segment\_groups

This table contains all segment groups stored on the cluster with details about the ownership, associated table, data structure, and other metadata.

| Column Name | Column Type | Column Description                   |
|-------------|-------------|--------------------------------------|
| id          | CHAR        | The identifier of the segment group. |



| Column Name       | Column Type | Column Description                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| cluster_id        | UUID        | Universally Unique Identifier (UUID) of the internal cluster of this segment group.                                                                                                                                                                                                                                                                                                         |
| segment_type      | CHAR        | Type of the segment (TKT, PAGE, etc.).                                                                                                                                                                                                                                                                                                                                                      |
| status            | CHAR        | The availability and health status of the segment group.                                                                                                                                                                                                                                                                                                                                    |
| primary_owner     | UUID        | For a replicated segment group, the UUID of the node that serves the segment (sys.nodes).                                                                                                                                                                                                                                                                                                   |
| loader_id         | UUID        | The streamloader node that wrote the segment group (sys.nodes).                                                                                                                                                                                                                                                                                                                             |
| table_id          | UUID        | UUID of the table (sys.tables).                                                                                                                                                                                                                                                                                                                                                             |
| scope_id          | UUID        | UUID of the storage scope (sys.storage_scopes).                                                                                                                                                                                                                                                                                                                                             |
| block_size        | LONG        | Disk block size used to store segments in this group in bytes.                                                                                                                                                                                                                                                                                                                              |
| begin_time        | LONG        | Minimum bucket value, specified as an integer, for all rows in this bucket. If the bucket column is a date or time value, this column has the same unit of measure as defined by the TimeKey column and can be converted or compared to other dates or times with appropriate conversion functions. When you specify a TIMESTAMP TimeKey, the value represents nanoseconds since the epoch. |
| end_time          | LONG        | Maximum bucket value, specified as an integer, for all rows in this bucket. If the bucket column is a date or time value, this column has the same unit of measure as defined by the TimeKey column and can be converted or compared to other dates or times with appropriate conversion functions. When you specify a TIMESTAMP TimeKey, the value represents nanoseconds since the epoch. |
| coding_algorithm  | CHAR        | Coding algorithm used to erasure code this group (NO_CODING, PQ_PARITY, REED_SOLOMON).                                                                                                                                                                                                                                                                                                      |
| coding_block_size | INT         | The unit size that is erasure coded in bytes.                                                                                                                                                                                                                                                                                                                                               |
| coding_threshold  | INT         | The number of coding blocks required to rebuild all blocks in a coding line.                                                                                                                                                                                                                                                                                                                |
| coding_width      | INT         | The number of coding blocks in a coding line.                                                                                                                                                                                                                                                                                                                                               |
| replication       | INT         | The number of replicas of each segment in the group.                                                                                                                                                                                                                                                                                                                                        |

| Column Name       | Column Type | Column Description                                                                                                                |
|-------------------|-------------|-----------------------------------------------------------------------------------------------------------------------------------|
| parity_cycle      | INT         | Parity cycle that the system calculates by multiplying by the coding_width to return the number of segments in the segment group. |
| created_time      | TIMESTAMP   | Specifies the time, in nanoseconds, when this segment group was created.                                                          |
| rolehostd_version | CHAR        | The version of the rolehostd binary at the time of the segment group generation.                                                  |
| commit_hash       | CHAR        | The commit hash of the rolehostd binary at the time of the segment group generation.                                              |
| timestamp         | CHAR        | The build timestamp or commit timestamp of the rolehostd binary at the time of the segment group generation.                      |
| build_user        | CHAR        | The build user of the rolehostd binary at the time of the segment group generation.                                               |
| depth             | INT         | The depth of the segment directory group. A non-zero value indicates a segment directory group.                                   |
| removal_type      | CHAR        | The method by which a segment group was removed.                                                                                  |
| visibility        | CHAR        | The visibility of the segment group.                                                                                              |

#### sys.segment\_groups\_merge\_eligible\_per\_table

This table computes and displays a count of segment groups and segment directory groups eligible for merging for each table in the Ocint System.

| Column Name                   | Column Type | Column Description                                              |
|-------------------------------|-------------|-----------------------------------------------------------------|
| id                            | UUID        | Universally Unique Identifier (UUID) of the table (sys.tables). |
| schema                        | CHAR        | Name of the schema.                                             |
| name                          | CHAR        | Name of the table.                                              |
| segment_groups_merge_eligible | LONG        | The count of segment groups eligible for merging in the table.  |

#### sys.segment\_part\_inventory

This table contains information about the inventory of markers for segments. For example, a marker denotes whether the database can delete a specific segment part. One segment can have multiple inventories, whereas each inventory contains only one marker.

| Column Name | Column Type | Column Description                                                                              |
|-------------|-------------|-------------------------------------------------------------------------------------------------|
| id          | UUID        | The universally unique identifier (UUID) of the inventory that stores the markers for segments. |

| Column Name             | Column Type | Column Description                                                                                                                                                                        |
|-------------------------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ida_offset              | INT         | The IDA offset (the logical part of the segment) denotes the position within the segment.                                                                                                 |
| parent_segment_group_id | CHAR        | The identifier of the segment group where the parent segment is located.                                                                                                                  |
| parent_ida_offset       | INT         | The IDA offset where the parent segment is located within the segment group.                                                                                                              |
| num_deleted_rows        | LONG        | The number of deleted rows.                                                                                                                                                               |
| operation_id            | UUID        | The UUID of the query where the Ocident System creates the inventory (the <b>DELETE</b> SQL statement). The system shares this UUID across all inventories created in the same operation. |
| subsuming_type          | CHAR        | The type of the marker within the inventory.                                                                                                                                              |
| node_id                 | UUID        | The UUID of the node that contains the latest segment with the part.                                                                                                                      |

sys.segment\_part\_redundancy\_info

This table contains the redundancy strategy for each table and the segment part type.

| Column Name     | Column Type | Column Description                                              |
|-----------------|-------------|-----------------------------------------------------------------|
| table_id        | UUID        | Universally Unique IDentifier (UUID) of the table (sys.tables). |
| part_name       | CHAR        | Name of the segment part.                                       |
| redundancy_type | CHAR        | The redundancy strategy for this part (COPY, PARITY).           |

sys.segment\_parts

This table contains the segment parts associated with each segment group. These segment parts belong to queryable segments. The table does not display parts that belong to quarantined segments or segments not yet activated.

| Column Name        | Column Type | Column Description                                                                           |
|--------------------|-------------|----------------------------------------------------------------------------------------------|
| segment_group_id   | CHAR        | The identifier of the segment group (sys.segment_groups).                                    |
| ida_offset         | INT         | Unique index of the segment. The original copy of this segment part belongs to this segment. |
| name               | CHAR        | Name of the segment part.                                                                    |
| part_type          | CHAR        | Partial identifier of a segment part (DATA, INDEX, MANIFEST, STATS).                         |
| size               | LONG        | Size of the segment part in bytes.                                                           |
| segment_ida_offset | INT         | Unique index of a segment within a segment group.                                            |

| Column Name        | Column Type | Column Description                                                          |
|--------------------|-------------|-----------------------------------------------------------------------------|
| segment_lba_offset | LONG        | Offset of this segment part within the Logical Block Address (LBA) on disk. |

## sys.segments

This table contains the individual segments stored in each storage cluster including basic information about the segment and the segment group where the segment belongs. The table contains queryable segments only. Quarantined segments or segments not yet activated do not appear in this table (they are present in the sys.stored\_segments table).

| Column Name                             | Column Type | Column Description                                                                                                                                                                                                                                                                                                                                                                          |
|-----------------------------------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| segment_group_id                        | CHAR        | The identifier of the segment group (sys.segment_groups).                                                                                                                                                                                                                                                                                                                                   |
| segment_type                            | CHAR        | Type of the segment (TKT or PAGE).                                                                                                                                                                                                                                                                                                                                                          |
| ida_offset                              | INT         | Unique index of the segment in the segment group.                                                                                                                                                                                                                                                                                                                                           |
| row_count                               | LONG        | Count of accessible rows in the segment. If the count is unknown, the value is NULL.                                                                                                                                                                                                                                                                                                        |
| storage_id                              | UUID        | Universally Unique IDentifier (UUID) for the stored segment (sys.stored_segments.storage_id).                                                                                                                                                                                                                                                                                               |
| table_id                                | UUID        | Universally Unique IDentifier (UUID) of the table (sys.tables).                                                                                                                                                                                                                                                                                                                             |
| segment_size                            | LONG        | Size of the segment in bytes.                                                                                                                                                                                                                                                                                                                                                               |
| root_segment_directory_group_id         | CHAR        | The identifier of the root-level segment directory group to which this segment belongs. For root-level directories and non-subsumed segments, this value is the same as the <b>segment_group_id</b> column.                                                                                                                                                                                 |
| root_segment_directory_group_ida_offset | INT         | The unique index of the directory in the root-level directory group to which this segment belongs.                                                                                                                                                                                                                                                                                          |
| begin_time                              | LONG        | Minimum bucket value, specified as an integer, for all rows in this bucket. If the bucket column is a date or time value, this column has the same unit of measure as defined by the TimeKey column and can be converted or compared to other dates or times with appropriate conversion functions. When you specify a TIMESTAMP TimeKey, the value represents nanoseconds since the epoch. |

| Column Name                | Column Type | Column Description                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| end_time                   | LONG        | Maximum bucket value, specified as an integer, for all rows in this bucket. If the bucket column is a date or time value, this column has the same unit of measure as defined by the TimeKey column and can be converted or compared to other dates or times with appropriate conversion functions. When you specify a TIMESTAMP TimeKey, the value represents nanoseconds since the epoch. |
| owner                      | LONG        | Owner identifier of this stored segment.                                                                                                                                                                                                                                                                                                                                                    |
| num_deleted_rows           | LONG        | Number of rows deleted from this segment.                                                                                                                                                                                                                                                                                                                                                   |
| page_size_with_replication | LONG        | Specifies the replication of stored pages.                                                                                                                                                                                                                                                                                                                                                  |

## sys.storage\_capacity

This table contains information about storage capacity for each node.

| Column Name    | Column Type | Column Description                                            |
|----------------|-------------|---------------------------------------------------------------|
| node_id        | UUID        | Universally Unique IDentifier (UUID) of the node (sys.nodes). |
| capacity_bytes | LONG        | Approximate storage capacity of the node in bytes.            |
| cluster_id     | UUID        | UUID of the cluster (sys.clusters).                           |

## sys.storage\_device\_files

This table contains the list of all files for each storage device.

| Column Name           | Column Type | Column Description                                                                                                                                      |
|-----------------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| node_id               | UUID        | The unique identifier of the node.                                                                                                                      |
| device_id             | INT         | The numeric identifier of the drive.                                                                                                                    |
| storage_id            | UUID        | The unique identifier of the file.                                                                                                                      |
| owner                 | INT         | Owner of the storage identifier. The zero value indicates an unowned storage identifier. A non-zero value indicates a segment owned after the rebuild.  |
| segment_type          | CHAR        | The type of the segment (TKT, PAGE, etc.).                                                                                                              |
| parent_id             | UUID        | The unique identifier of the parent file (for addendum parts).                                                                                          |
| normalized_drive_slot | INT         | Theoretical drive slot, if applicable, that was computed from the segment group identifier. The Ocint System expects to allocate the file to this slot. |

| Column Name        | Column Type | Column Description                                                                                                                                                                                         |
|--------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| abnormal_placement | BOOLEAN     | Specifies whether the Ocident System places the segment abnormally. If this value is <code>true</code> , the segment is not located on the drive slot that was computed from the segment group identifier. |
| segment_group_id   | CHAR        | The identifier of the segment group (sys.segment_groups).                                                                                                                                                  |

## sys.storage\_scopes

This table contains the storage scopes defined in the system with the number of rows, page groups, and segment groups in each storage scope.

| Column Name            | Column Type | Column Description                                                 |
|------------------------|-------------|--------------------------------------------------------------------|
| id                     | UUID        | Universally Unique Identifier (UUID) of the storage scope.         |
| row_count              | LONG        | The number of rows that have been currently loaded onto the scope. |
| num_page_groups        | LONG        | The number of page groups currently present in the scope.          |
| num_tkt_segment_groups | LONG        | Number of TKT segment groups currently present in the scope.       |
| cluster_id             | UUID        | UUID of the cluster (sys.clusters).                                |

## sys.storage\_spaces

This table contains information for all defined storage spaces.

| Column Name             | Column Type | Column Description                                                                                            |
|-------------------------|-------------|---------------------------------------------------------------------------------------------------------------|
| id                      | UUID        | Universally Unique Identifier (UUID) of the storage space.                                                    |
| name                    | CHAR        | Name of the storage space.                                                                                    |
| is_system_storage_space | BOOLEAN     | Is this storage space the system storage space                                                                |
| block_size              | INT         | Unit size that is erasure coded in bytes.                                                                     |
| total_width             | INT         | The coding width. (The N in an M of N parity configuration.)                                                  |
| parity_width            | INT         | Within a coding line, the number of blocks dedicated to storing parity information.                           |
| parity_type             | CHAR        | The methodology used to compute parity blocks (P+Q, XOR, REED SOLOMON, REPLICATION, NONE).                    |
| parity_cycles           | INT         | Parity cycles calculated by multiplying by total_width to return the number of segments in the segment group. |

| Column Name      | Column Type | Column Description                                                   |
|------------------|-------------|----------------------------------------------------------------------|
| page_replication | INT         | The total number of page replicas, which includes the original page. |

sys.storage\_used

This table contains information about storage utilization for each node.

| Column Name | Column Type | Column Description                                                 |
|-------------|-------------|--------------------------------------------------------------------|
| node_id     | UUID        | Universally Unique Identifier (UUID) of the node (sys.nodes).      |
| table_id    | UUID        | UUID of the table (sys.tables).                                    |
| used_bytes  | LONG        | Approximate storage utilization of the table on the node in bytes. |
| cluster_id  | UUID        | UUID of the cluster (sys.clusters).                                |

sys.stored\_leaf\_segment\_part\_inventory

This table contains the physical location for the inventory of markers for all segments.

| Column Name             | Column Type | Column Description                                                                                  |
|-------------------------|-------------|-----------------------------------------------------------------------------------------------------|
| segment_group_id        | CHAR        | The identifier of the segment group (sys.segment_groups).                                           |
| cluster_id              | UUID        | Universally Unique Identifier (UUID) of the cluster (sys.clusters).                                 |
| ida_offset              | INT         | Index of the segment in the segment group.                                                          |
| storage_id              | UUID        | UUID of the stored segment.                                                                         |
| owner                   | LONG        | Owner identifier of the stored segment.                                                             |
| root_directory_group_id | CHAR        | Root segment directory identifier, if it has been subsumed.                                         |
| node_id                 | UUID        | UUID of the node (sys.nodes).                                                                       |
| start_osn               | CHAR        | The starting Ownership Sequence Number (OSN) for the latest OSN range of the inventory of markers.  |
| end_osn                 | CHAR        | The ending OSN for the latest OSN range of the inventory of markers.                                |
| kind                    | CHAR        | The kind of segment (DISK, VIRTUAL).                                                                |
| visibility              | CHAR        | The visibility of the inventory of markers.                                                         |
| toc_name                | UUID        | UUID of the inventory of markers. A NULL UUID denotes the main inventory of markers of the segment. |

sys.stored\_segments

This table contains details for each segment that is saved to disk.

| Column Name        | Column Type | Column Description                                                  |
|--------------------|-------------|---------------------------------------------------------------------|
| segment_group_id   | CHAR        | The identifier of the segment group (sys.segment_groups).           |
| cluster_id         | UUID        | Universally Unique Identifier (UUID) of the cluster (sys.clusters). |
| ida_offset         | INT         | Index of the segment in the segment group.                          |
| storage_id         | UUID        | UUID of this stored segment.                                        |
| owner              | LONG        | Owner identifier of this stored segment.                            |
| node_id            | UUID        | UUID of the node (sys.nodes).                                       |
| status             | CHAR        | The current status of the segment (MISSING, INTACT, etc.).          |
| kind               | CHAR        | The kind of segment (DISK, VIRTUAL).                                |
| start_osn          | CHAR        | The start OSN for the latest OSN range of the segment.              |
| end_osn            | CHAR        | The end OSN for the latest OSN range of the segment.                |
| visibility         | CHAR        | The visibility of the stored segment.                               |
| abnormal_placement | BOOLEAN     | Whether the segment is placed abnormally.                           |
| row_count          | LONG        | Count of rows in the segment.                                       |
| segment_size       | LONG        | Size of the segment in bytes.                                       |

## System

## sys.clusters

This table contains all clusters defined in the system.

| Column Name       | Column Type | Column Description                                                        |
|-------------------|-------------|---------------------------------------------------------------------------|
| id                | UUID        | Universally Unique Identifier (UUID) of the cluster (sys.clusters.id).    |
| name              | CHAR        | Name of the cluster.                                                      |
| cluster_type      | CHAR        | Type of the cluster.                                                      |
| storage_space_ids | ARRAY(UUID) | UUIDs of the storage spaces used in this cluster (sys.storage_spaces.id). |

## sys.compute\_configurations

This table contains the compute configurations for nodes and clusters in the system.

| Column Name | Column Type | Column Description                                                                               |
|-------------|-------------|--------------------------------------------------------------------------------------------------|
| cluster_id  | UUID        | Universally Unique Identifier (UUID) of the cluster that this node belongs to (sys.clusters.id). |
| node_id     | UUID        | UUID of the node (sys.nodes.id).                                                                 |
| level       | LONG        | Execution level for this node or cluster.                                                        |
| csn         | LONG        | Compute sequence number.                                                                         |



## sys.locks

This table contains information of the currently queued and granted lock requests.

| Column Name            | Column Type | Column Description                                                                                                                                                                   |
|------------------------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| request_id             | UUID        | The unique identifier of the lock.                                                                                                                                                   |
| owner_identifier       | CHAR        | The unique identifier of the owner of the lock. Should be of format <b>&lt;node uuid with lowercase chars&gt;.&lt;locking system&gt;.&lt;locking reason&gt;.&lt;process uuid&gt;</b> |
| lock_scope_id          | CHAR        | An identifier on the scope of the lock, should be of format <b>&lt;system&gt;.&lt;type&gt;.&lt;target unique identifier&gt;</b>                                                      |
| lock_type              | CHAR        | Whether the lock is to READ or WRITE on the elements in its scope.                                                                                                                   |
| status                 | CHAR        | The status of the lock, QUEUED means that it is waiting to be accepted, GRANTED means it is currently active.                                                                        |
| create_time            | LONG        | The creation time, in milliseconds, of the lock relative to the underlying RAFT Consensus Log. This value is not a UNIX timestamp.                                                   |
| last_refresh_time      | LONG        | The time, in milliseconds, when the lock was last refreshed relative to the underlying RAFT Consensus Log. This value is not a UNIX timestamp                                        |
| priority_id            | UUID        | The identifier for lock prioritization.                                                                                                                                              |
| create_timestamp       | TIMESTAMP   | The timestamp that represents when the lock was created on the Raft leader.                                                                                                          |
| last_refresh_timestamp | TIMESTAMP   | The timestamp that represents when the lock was most recently refreshed on the Raft leader.                                                                                          |

## sys.lts\_cluster\_info

This table contains the storage clusters defined in the system.

| Column Name       | Column Type | Column Description                                                             |
|-------------------|-------------|--------------------------------------------------------------------------------|
| cluster_id        | UUID        | Universally Unique Identifier (UUID) of the storage cluster (sys.clusters.id). |
| storage_space_ids | ARRAY(UUID) | UUIDs of the storage spaces used in this cluster (sys.storage_spaces.id).      |

## sys.node\_clusters

This table contains nodes that are members of each cluster.

| Column Name | Column Type | Column Description                                                             |
|-------------|-------------|--------------------------------------------------------------------------------|
| cluster_id  | UUID        | Universally Unique Identifier (UUID) of the cluster that this node belongs to. |
| node_id     | UUID        | UUID of the node.                                                              |
| ordinal     | INT         | Ordinal of this node.                                                          |

sys.nodes

This table contains all nodes defined in the system.

| Column Name        | Column Type | Column Description                                  |
|--------------------|-------------|-----------------------------------------------------|
| id                 | UUID        | Universally Unique Identifier (UUID) of the node.   |
| unique_node_number | INT         | Ordinal number assigned to the node.                |
| name               | CHAR        | Name of the node.                                   |
| status             | CHAR        | Status of the node (ACCEPTED, UNACCEPTED, INVALID). |
| hostname           | CHAR        | Hostname of the node.                               |
| software_version   | CHAR        | Version of the software running on this node.       |
| kernel_version     | CHAR        | Version of the kernel running on this node.         |
| system_version     | CHAR        | Version of Linux® running on this node.             |
| system_memory      | LONG        | Amount of available system memory in bytes.         |
| sockets            | INT         | Number of CPU sockets.                              |
| cores_per_socket   | INT         | Number of cores per CPU socket.                     |
| hugepages_1gb      | INT         | Number of configured 1GB huge pages.                |
| hugepages_2mb      | INT         | Number of configured 2MB huge pages.                |
| hyperthreaded      | BOOLEAN     | Whether the node is hyperthreaded.                  |

sys.service\_roles

This table contains the service roles defined on each node.

| Column Name       | Column Type | Column Description                                        |
|-------------------|-------------|-----------------------------------------------------------|
| id                | UUID        | Universally Unique Identifier (UUID) of the service role. |
| service_role_type | CHAR        | Type of the service role.                                 |
| node_id           | UUID        | UUID of the node. (sys.nodes.id)                          |
| level             | CHAR        | Execution level of the node.                              |

System Information

sys.function\_signatures

This table contains all function signatures defined in the system.

| Column Name   | Column Type | Column Description                                              |
|---------------|-------------|-----------------------------------------------------------------|
| id            | UUID        | Universally Unique Identifier (UUID) of the function signature. |
| name          | CHAR        | Name of the function.                                           |
| return_type   | CHAR        | Return type of the function.                                    |
| arg_types     | ARRAY(CHAR) | The types of all arguments in the function.                     |
| function_type | CHAR        | Type of the function.                                           |

sys.reserved\_words

This table contains all reserved words defined in Ocient.

| Column Name   | Column Type | Column Description                                   |
|---------------|-------------|------------------------------------------------------|
| reserved_word | CHAR        | A word that is a reserved keyword for use by Ocient. |

sys.sql\_messages

This table contains information about the SQL errors and warnings that the Ocient System can produce.

| Column Name | Column Type | Column Description                                      |
|-------------|-------------|---------------------------------------------------------|
| name        | CHAR        | SQL error or warning name.                              |
| code        | INT         | SQL error or warning code.                              |
| state       | CHAR        | SQL error or warning state.                             |
| reason      | CHAR        | Description of the SQL error or warning.                |
| is_error    | BOOLEAN     | Whether or not this SQL message is an error or warning. |

sys.system\_information

This table contains information about the Ocient System.

| Column Name                         | Column Type | Column Description                                                                                                                                                                                             |
|-------------------------------------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| id                                  | UUID        | Universally Unique Identifier (UUID) for the overall system.                                                                                                                                                   |
| name                                | CHAR        | The name of the system. This system assigns this name randomly using a UUID by default. You can change the name using the <b>ALTER SYSTEM RENAME</b> SQL statement.                                            |
| current_compatible_software_version | INT         | The minimum software version for the system compatibility.                                                                                                                                                     |
| allowed_compatible_software_version | INT         | The software version that prevents the use of features in the newest version of the system to keep backward compatibility with the specified version in the <b>current_compatible_software_version</b> column. |

| Column Name             | Column Type | Column Description                                                                 |
|-------------------------|-------------|------------------------------------------------------------------------------------|
| feature_directory_merge | CHAR        | The value of the <b>directory merge</b> feature flag for segment directory groups. |

## sys.system\_table\_columns

This table contains all columns for tables available in the system catalog.

| Column Name | Column Type | Column Description                                                     |
|-------------|-------------|------------------------------------------------------------------------|
| id          | UUID        | The Universally Unique IDentifier (UUID) of this column (sys.columns). |
| name        | CHAR        | The name of this column.                                               |
| data_type   | CHAR        | The data type of the column.                                           |
| nullable    | BOOLEAN     | Whether or not this column is nullable.                                |
| ordinal     | LONG        | The ordinal of this column.                                            |
| schema_name | CHAR        | The name of the schema to which this column belongs.                   |
| table_name  | CHAR        | The name of the table to which this column belongs.                    |
| description | CHAR        | Description of the column.                                             |

## sys.system\_tables

This table contains all tables available in the system catalog.

| Column Name | Column Type | Column Description                                              |
|-------------|-------------|-----------------------------------------------------------------|
| id          | UUID        | Universally Unique IDentifier (UUID) of the table (sys.tables). |
| schema      | CHAR        | Schema of the table.                                            |
| name        | CHAR        | Name of the table.                                              |
| table_type  | CHAR        | Type of the table.                                              |
| category    | CHAR        | Category of the table.                                          |
| description | CHAR        | Description of the table.                                       |

## User Management

## sys.group\_roles

This table contains roles that belong to each group.

| Column Name | Column Type | Column Description                                              |
|-------------|-------------|-----------------------------------------------------------------|
| group_id    | UUID        | Universally Unique IDentifier (UUID) of the group (sys.groups). |
| role_id     | UUID        | UUID of the role (sys.roles).                                   |

## sys.groups

This table contains information, including the service class, related to each group.

| Column Name      | Column Type | Column Description                                             |
|------------------|-------------|----------------------------------------------------------------|
| id               | UUID        | Universally Unique Identifier (UUID) of the group.             |
| name             | CHAR        | Name of the group.                                             |
| database_id      | UUID        | UUID of the database (sys.databases).                          |
| service_class_id | UUID        | UUID of the service class of this group (sys.service_classes). |

## sys.privileges

This table contains information related to the privileges granted in Ocint.

| Column Name      | Column Type | Column Description                                                                                                    |
|------------------|-------------|-----------------------------------------------------------------------------------------------------------------------|
| timestamp        | TIMESTAMP   | Timestamp of the grant given.                                                                                         |
| grantor          | CHAR        | The user who granted this privilege.                                                                                  |
| grantee          | CHAR        | The user who received this privilege.                                                                                 |
| privilege        | CHAR        | The privilege for the grant.                                                                                          |
| privilege_target | CHAR        | The type of object to which the privilege applies. This value is NULL if the privilege applies to the granted object. |
| object_type      | CHAR        | The type of object on which this privilege was granted.                                                               |
| object_id        | UUID        | Universally Unique Identifier (UUID) of the object.                                                                   |
| grantable        | BOOLEAN     | Whether the user can grant this privilege to another user.                                                            |

## sys.rights

This table contains information related to the rights granted in Ocint.

| Column Name | Column Type | Column Description                                              |
|-------------|-------------|-----------------------------------------------------------------|
| entity_type | CHAR        | Owner of the right (user, group, role).                         |
| entity_id   | UUID        | Universally Unique Identifier (UUID) of the owner.              |
| target_type | CHAR        | Type of the target.                                             |
| target_id   | UUID        | UUID of the target.                                             |
| value       | CHAR        | The right being given (CREATE, READ, UPDATE, DELETE, SECURITY). |

## sys.roles

This table contains roles in the system.

| Column Name | Column Type | Column Description                                |
|-------------|-------------|---------------------------------------------------|
| id          | UUID        | Universally Unique Identifier (UUID) of the role. |

| Column Name | Column Type | Column Description                                 |
|-------------|-------------|----------------------------------------------------|
| database_id | UUID        | UUID of the database of this role (sys.databases). |
| name        | CHAR        | Name of this role.                                 |
| description | CHAR        | Detailed description of this role.                 |

## sys.user\_groups

This table contains the users that belong to groups in Ocident.

| Column Name | Column Type | Column Description                                            |
|-------------|-------------|---------------------------------------------------------------|
| user_id     | UUID        | Universally Unique Identifier (UUID) of the user (sys.users). |
| group_id    | UUID        | UUID of the group where the user belongs (sys.groups).        |

## sys.user\_roles

This table contains roles associated with each user.

| Column Name | Column Type | Column Description                                            |
|-------------|-------------|---------------------------------------------------------------|
| user_id     | UUID        | Universally Unique Identifier (UUID) of the user (sys.users). |
| role_id     | UUID        | UUID of the role (sys.roles).                                 |

## sys.users

This table contains information about users in Ocident.

| Column Name            | Column Type | Column Description                                                                                   |
|------------------------|-------------|------------------------------------------------------------------------------------------------------|
| id                     | UUID        | Universally Unique Identifier (UUID) of the user.                                                    |
| user_name              | CHAR        | Username.                                                                                            |
| database_id            | UUID        | UUID of the database.                                                                                |
| first_name             | CHAR        | First name of the user.                                                                              |
| last_name              | CHAR        | Last name of the user.                                                                               |
| email                  | CHAR        | Email address of the user.                                                                           |
| password_updated_at    | TIMESTAMP   | Timestamp that represents the date and time for the last time the password for this user was updated |
| state                  | CHAR        | State.                                                                                               |
| invalid_login_attempts | INT         | Number of invalid login attempts since last successful login.                                        |

## Workload Management

## sys.service\_classes

This table shows information about the service classes used for workload management in Ocident. For more details on workload management, see the corresponding section of the user documentation.

| Column Name                  | Column Type | Column Description                                                                                                                                                                                   |
|------------------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| id                           | UUID        | The ID of the service class.                                                                                                                                                                         |
| database_id                  | UUID        | The ID of the database that has the service class.                                                                                                                                                   |
| name                         | CHAR        | The name of the service class.                                                                                                                                                                       |
| max_temp_disk_usage          | LONG        | The maximum temporary disk usage for a query running with this service class.                                                                                                                        |
| max_elapsed_time             | LONG        | The maximum time, in seconds, that a query running with this service class can run.                                                                                                                  |
| max_concurrent_queries       | LONG        | The maximum number of queries that can run concurrently with this service class.                                                                                                                     |
| max_rows_returned            | LONG        | The maximum number of rows a query running with this service class can return.                                                                                                                       |
| scheduling_priority          | DOUBLE      | The initial priority for a query running with this service class.                                                                                                                                    |
| cache_max_bytes              | LONG        | The maximum number of bytes in a result set, if it is eligible for cache storage, for a query running with this service class.                                                                       |
| cache_max_time               | LONG        | The maximum amount of time, in seconds, that the system caches rows for a query running with this service class.                                                                                     |
| max_elapsed_time_for_caching | LONG        | The maximum amount of time, in seconds, that a query running with this service class can run and have its result set cached. This value must be higher than the <code>max_elapsed_time</code> value. |
| max_columns_in_result_set    | LONG        | The maximum number of columns allowed in the result set of a query running with this service class.                                                                                                  |
| priority_adjustment_factor   | DOUBLE      | The amount that the system adjusts the query priority in each time interval, as specified by the <code>priority_adjustment_time</code> value, for a query running with this service class.           |
| priority_adjustment_time     | LONG        | The frequency, in seconds, of each priority adjustment for a query running with this service class.                                                                                                  |
| min_priority                 | DOUBLE      | The minimum value of the adjusted priority for a query running with this service class.                                                                                                              |
| max_priority                 | DOUBLE      | The maximum value of the adjusted priority for a query running with this service class.                                                                                                              |

| Column Name                  | Column Type | Column Description                                                                                                                                                                                                                                                                                                              |
|------------------------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| statement_text               | CHAR        | Specifies the comparison string to use for matching statement text to queries. This string takes the format specified by the statement_text_matcher_type field.                                                                                                                                                                 |
| statement_text_matcher_type  | CHAR        | Specifies the type of string comparison to use for matching statement text to queries. This value must be LIKE or REGEX and is required if you specify a 'statement_text' field.                                                                                                                                                |
| half_parallelism             | BOOLEAN     | Determines whether the VM uses half of the available cores for each operator in this query rather than the maximum possible parallelism. Setting this value can reduce overhead and latency for queries, but it significantly decreases data throughput. So, updating this value should be rare outside of Ocient Support work. |
| load_balance_shuffle         | BOOLEAN     | Determines whether the optimizer includes a load-balancing network operator above I/O for each query. This field can add latency to queries, but it will likely increase data throughput. If unset, the choice is left to the optimizer. So, updating this value should be rare outside of Ocient Support work.                 |
| parallelism                  | INT         | Determines the number of parallel cores to use for each operator in the VM. This value must be nonzero. If unset, the choice is left to the VM. So, updating this value should be rare outside of Ocient Support work.                                                                                                          |
| minimize_query_debug_records | BOOLEAN     | This value determines whether the optimizer and VM should attempt to limit debug metrics and logging for queries associated with this service class. Limiting query observability may improve performance for workloads running many queries per second.                                                                        |
| memory_optimal_strategy      | BOOLEAN     | This value determines whether the optimizer and VM should attempt to lower the memory requirements of a query regardless of potential execution time penalties. So, updating this value should be rare outside of Ocient Support work.                                                                                          |

sys.service\_classes\_for\_user



The `service_classes_for_user` view shows groups that each user belongs to and the service class for each group.

| Column Name                     | Column Type | Column Description             |
|---------------------------------|-------------|--------------------------------|
| <code>user_name</code>          | CHAR        | Username.                      |
| <code>group_name</code>         | CHAR        | Name of the group.             |
| <code>service_class_name</code> | CHAR        | The name of the service class. |

## Related Links

[Information Schema](#)

### 10.25.2 Information Schema

The system catalog exposes a set of system views in the `information_schema` schema that contain metadata about the Ocint System. Most of these views follow the SQL standard with additional views specific to the system.

#### Alphabetical List of Views

- `information_schema.columns`
- `information_schema.data_types`
- `information_schema.databases`
- `information_schema.geometry_columns`
- `information_schema.groups`
- `information_schema.index_recommendations`
- `information_schema.indexes`
- `information_schema.information_schema_catalog_name`
- `information_schema.nodes`
- `information_schema.pipeline_status`
- `information_schema.pipelines`
- `information_schema.reserved_words`
- `information_schema.schemata`
- `information_schema.table_privileges`
- `information_schema.table_storage`
- `information_schema.tables`
- `information_schema.users`
- `information_schema.views`

## System View Descriptions

`information_schema.columns`

The `columns` view shows information for all columns in the system.

| Column Name                   | Column Type | Column Description                                       |
|-------------------------------|-------------|----------------------------------------------------------|
| <code>table_catalog</code>    | CHAR        | Name of the database.                                    |
| <code>table_schema</code>     | CHAR        | Name of the schema.                                      |
| <code>table_name</code>       | CHAR        | Name of the table.                                       |
| <code>column_name</code>      | CHAR        | Name of the column.                                      |
| <code>ordinal_position</code> | LONG        | Ordinal of the column with respect to its table.         |
| <code>data_type</code>        | CHAR        | Data type of the column (INT, CHAR, BOOLEAN, etc.).      |
| <code>is_nullable</code>      | CHAR        | Specifies whether the values in this column can be NULL. |

| Column Name    | Column Type | Column Description                                   |
|----------------|-------------|------------------------------------------------------|
| column_default | CHAR        | The default expression of this column, if it exists. |

information\_schema.data\_types

The **data\_types** view shows all data types in the system.

| Column Name | Column Type | Column Description |
|-------------|-------------|--------------------|
| data_type   | CHAR        | Data type.         |

information\_schema.databases

The **databases** view shows all databases where the current user has access.

| Column Name   | Column Type | Column Description                                      |
|---------------|-------------|---------------------------------------------------------|
| database_name | CHAR        | Name of the database.                                   |
| created_at    | TIMESTAMP   | Timestamp that specifies when the database was created. |

information\_schema.geometry\_columns

The **geometry\_columns** view shows information about all geometry type columns in the system.

| Column Name       | Column Type | Column Description                                          |
|-------------------|-------------|-------------------------------------------------------------|
| f_table_catalog   | CHAR        | Name of the database.                                       |
| f_table_schema    | CHAR        | Name of the schema.                                         |
| f_table_name      | CHAR        | Name of the table.                                          |
| f_geometry_column | CHAR        | Name of the column.                                         |
| coord_dimension   | INT         | The coordinate dimension.                                   |
| srid              | INT         | The ID of the spatial reference system.                     |
| type              | CHAR        | Data type of the column (POINT, LINESTRING, POLYGON, etc.). |

information\_schema.groups

The **groups** view shows all groups in the system.

| Column Name   | Column Type | Column Description    |
|---------------|-------------|-----------------------|
| database_name | CHAR        | Name of the database. |
| group_name    | CHAR        | Name of the group.    |

information\_schema.index\_recommendations

This view contains index recommendations.

| Column Name | Column Type | Column Description |
|-------------|-------------|--------------------|
| table_name  | CHAR        | Name of the table. |

| Column Name | Column Type | Column Description                                     |
|-------------|-------------|--------------------------------------------------------|
| column_name | CHAR        | Name of the column.                                    |
| sql         | CHAR        | SQL statement to create or drop the recommended index. |

information\_schema.indexes

The **indexes** view shows all indexes used in the system.

| Column Name   | Column Type | Column Description    |
|---------------|-------------|-----------------------|
| table_catalog | CHAR        | Name of the database. |
| table_schema  | CHAR        | Name of the schema.   |
| table_name    | CHAR        | Name of the table.    |
| index_name    | CHAR        | Name of the index.    |
| index_type    | CHAR        | Type of the index.    |
| column_name   | CHAR        | Name of the column.   |

information\_schema.information\_schema\_catalog\_name

The **information\_schema\_catalog\_name** view shows the current database.

| Column Name  | Column Type | Column Description    |
|--------------|-------------|-----------------------|
| catalog_name | CHAR        | Name of the database. |

information\_schema.nodes

The **nodes** view shows all nodes defined in the system.

| Column Name        | Column Type | Column Description                                                                                         |
|--------------------|-------------|------------------------------------------------------------------------------------------------------------|
| name               | CHAR        | Name of the node.                                                                                          |
| service_role_types | ARRAY(CHAR) | The type of service roles on this node.                                                                    |
| operational_status | CHAR        | The operational status of the node. Values are ACTIVE, STARTING, STOPPING, ERROR, UNKNOWN, or UNREACHABLE. |
| software_version   | CHAR        | Version of the software running on this node.                                                              |

information\_schema.pipeline\_status

To view dynamic information about the status of the pipeline, query from information\_schema.pipeline\_status, or you can use the SHOW PIPELINE\_STATUS command.

| Column Name   | Column Type | Column Description                                   |
|---------------|-------------|------------------------------------------------------|
| database_name | CHAR        | Name of the database where the pipeline was created. |
| pipeline_name | CHAR        | The name of this pipeline.                           |

| Column Name       | Column Type | Column Description                                                                                                                                                                                                                                                             |
|-------------------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| table_names       | ARRAY(CHAR) | An array of fully-qualified table names where the pipeline loads data. This value is always an array with one element.                                                                                                                                                         |
| status            | CHAR        | Status of the pipeline (RUNNING, STOPPED, COMPLETED, FAILED).                                                                                                                                                                                                                  |
| status_message    | CHAR        | The corresponding event message of the last event that the Ocient System sees for this pipeline from the event_message column of the sys.pipeline_events system catalog table.                                                                                                 |
| duration_seconds  | INT         | Duration, in seconds, of how long the pipeline has been running.                                                                                                                                                                                                               |
| files_processed   | LONG        | The number of files that have been processed for file-based loads. (includes files with these statuses: LOADED, LOADED_WITH_ERRORS, and SKIPPED)                                                                                                                               |
| files_failed      | LONG        | The number of files that have failed and have a file status of FAILED for file-based loads.                                                                                                                                                                                    |
| files_remaining   | LONG        | The number of remaining files for file-based loads. (includes these file statuses: PENDING, QUEUED, and LOADING)                                                                                                                                                               |
| files_total       | LONG        | The total number of files for file-based loads.                                                                                                                                                                                                                                |
| fraction_complete | FLOAT       | The estimated fraction completion as a value from 0.0 to 1.0 for file-based batch loads. The calculation is files processed or failed divided by the total files.                                                                                                              |
| records_processed | LONG        | The number of records that the pipeline has processed. The pipeline attempts to process and load a record. This number can be greater than the value of records_loaded due to the deduplication of records that the system processes twice or where record-level errors occur. |
| records_loaded    | LONG        | The number of records that have successfully loaded into the system. This number reflects the records in the system and does not include records that failed to load or were eliminated as duplicates.                                                                         |
| records_failed    | LONG        | Number of records that have failed.                                                                                                                                                                                                                                            |

information\_schema.pipelines

To view static information that stays the same after the pipeline has been created, query from `information_schema.pipelines`, or you can use the `SHOW PIPELINES` command. The columns in the `information_schema.pipelines` view derive from the underlying `sys.pipelines` system catalog table.

| Column Name                | Column Type | Column Description                                                                                                     |
|----------------------------|-------------|------------------------------------------------------------------------------------------------------------------------|
| <code>database_name</code> | CHAR        | Name of the database where the pipeline was created.                                                                   |
| <code>pipeline_name</code> | CHAR        | Name of the pipeline.                                                                                                  |
| <code>loading_mode</code>  | CHAR        | Specifies whether the pipeline runs in a one-time (BATCH) or continuous (CONTINUOUS) way.                              |
| <code>source_type</code>   | CHAR        | Source of the data (S3, KAFKA, FILESYSTEM).                                                                            |
| <code>data_format</code>   | CHAR        | Structure of the data (DELIMITED, CSV, JSON, PARQUET).                                                                 |
| <code>table_names</code>   | ARRAY(CHAR) | An array of fully-qualified table names where the pipeline loads data. This value is always an array with one element. |
| <code>created_at</code>    | TIMESTAMP   | Timestamp of when the pipeline was created.                                                                            |
| <code>altered_at</code>    | TIMESTAMP   | Timestamp that indicates when the pipeline was last altered.                                                           |
| <code>creator_id</code>    | UUID        | Universally Unique Identifier (UUID) of the creator of the pipeline.                                                   |

`information_schema.reserved_words`

The **reserved\_words** view shows all reserved words in the system.

| Column Name                | Column Type | Column Description                                  |
|----------------------------|-------------|-----------------------------------------------------|
| <code>reserved_word</code> | CHAR        | A word that is a reserved keyword for use by Ocint. |

`information_schema.schemata`

The **schemata** view shows information for all schemas in the system.

| Column Name               | Column Type | Column Description    |
|---------------------------|-------------|-----------------------|
| <code>schema_name</code>  | CHAR        | Name of the schema.   |
| <code>catalog_name</code> | CHAR        | Name of the database. |

`information_schema.table_privileges`

The **table\_privileges** view shows information for all table privileges in the system.

| Column Name                | Column Type | Column Description                    |
|----------------------------|-------------|---------------------------------------|
| <code>grantor</code>       | CHAR        | The user who granted this privilege.  |
| <code>grantee</code>       | CHAR        | The user who received this privilege. |
| <code>table_catalog</code> | CHAR        | Name of the database.                 |

| Column Name    | Column Type | Column Description                                         |
|----------------|-------------|------------------------------------------------------------|
| table_schema   | CHAR        | Name of the schema.                                        |
| table_name     | CHAR        | Name of the table.                                         |
| privilege_type | CHAR        | The privilege for the grant.                               |
| is_grantable   | CHAR        | Whether the user can grant this privilege to another user. |

information\_schema.table\_storage

The **tables\_storage** view shows information about the storage used by tables.

| Column Name       | Column Type | Column Description                                |
|-------------------|-------------|---------------------------------------------------|
| table_catalog     | CHAR        | Name of the database.                             |
| table_schema      | CHAR        | Name of the schema.                               |
| table_name        | CHAR        | Name of the table.                                |
| row_count         | LONG        | The number of rows in the table.                  |
| deleted_row_count | LONG        | The number of deleted rows in the table.          |
| segment_count     | LONG        | The number of segments that constitute the table. |
| size              | LONG        | The size of the table in bytes.                   |

information\_schema.tables

The **tables** view shows information for all tables and views in the system, including user-defined and system objects.

| Column Name        | Column Type | Column Description                                                |
|--------------------|-------------|-------------------------------------------------------------------|
| table_catalog      | CHAR        | Name of the database.                                             |
| table_schema       | CHAR        | Name of the schema.                                               |
| table_name         | CHAR        | Name of the table.                                                |
| table_type         | CHAR        | Type of the table.                                                |
| is_insertable_into | CHAR        | Whether an INSERT command can target this table.                  |
| created_at         | TIMESTAMP   | Timestamp that represents when the table was created.             |
| creator_id         | UUID        | Universally Unique Identifier (UUID) of the creator of the table. |

information\_schema.users

The **users** view shows all users in the system.

| Column Name   | Column Type | Column Description    |
|---------------|-------------|-----------------------|
| database_name | CHAR        | Name of the database. |
| user_name     | CHAR        | Username of the user. |

information\_schema.views

The **views** view shows information for all user-defined and system views in the system.

| Column Name     | Column Type | Column Description                                                           |
|-----------------|-------------|------------------------------------------------------------------------------|
| table_catalog   | CHAR        | Name of the database.                                                        |
| table_schema    | CHAR        | Name of the schema.                                                          |
| table_name      | CHAR        | Name of the table.                                                           |
| view_definition | CHAR        | Query used to generate the view content.                                     |
| created_at      | TIMESTAMP   | Timestamp that represents the date and time for the creation of the view.    |
| updated_at      | TIMESTAMP   | Timestamp that represents the date and time for the last update of the view. |
| creator_id      | UUID        | Universally Unique Identifier (UUID) of the creator of the view.             |

### SHOW Commands

The **SHOW** commands are additional commands you can use to retrieve metadata about the system. These commands are tightly coupled with the **information\_schema** schema and offer a more user-friendly syntax alternative. Sometimes, unlike their **information\_schema** counterparts, **SHOW** commands only return user-defined objects.

| SHOW Command          | Description                                                                                              |
|-----------------------|----------------------------------------------------------------------------------------------------------|
| SHOW COLUMNS          | This statement returns columns from user-defined tables and views where the current user has access.     |
| SHOW DATA_TYPES       | This statement returns all data types in the system.                                                     |
| SHOW DATABASES        | This statement returns all databases where the current user has access.                                  |
| SHOW GEOMETRY_COLUMNS | This statement returns all geometry type columns where the current user has access.                      |
| SHOW GROUPS           | This statement returns all groups in the system where the current user has read access.                  |
| SHOW INDEXES          | This statement returns all indexes on user-defined tables where the current user has access.             |
| SHOW NODES            | This statement returns all nodes in the system where the current user has read access.                   |
| SHOW PIPELINE_STATUS  | This statement returns the status of all pipelines in the system where the current user has read access. |
| SHOW PIPELINES        | This statement returns all pipelines in the system where the current user has read access.               |
| SHOW RESERVED_WORDS   | This statement returns all reserved words in the system.                                                 |
| SHOW SCHEMATA         | This statement returns all user-defined schemas where the current user has access.                       |
| SHOW SYSTEM TABLES    | This statement returns all system tables in the <b>sys</b> or <b>information_schema</b> schemas.         |
| SHOW TABLE_PRIVILEGES | This statement returns all table privileges for tables where the current user has access.                |
| SHOW TABLES           | This statement returns all user-defined tables where the current user has access.                        |
| SHOW USERS            | This statement returns all users in the system where the current user has read access.                   |
| SHOW VIEWS            | This statement returns all user-defined views where the current user has access.                         |

### Related Links

## SQL Reference Database Administration System Catalog

**10.25.3 Discover Insights From System Catalog Tables**

You can query the Ocient® System catalog tables to discover important information on system objects such as databases, segments, tables, and indexes. These statistics can be important for monitoring your system to ensure it operates as intended.

However, querying these tables can be challenging because generating insights about your system can involve writing complex queries that involve merging, grouping, and filtering multiple system catalog tables.

Here, you can find examples of useful queries that you can use as templates for discovering insights about your system.

Most of these example queries use the `data_type_coverage` table available with the Ocient Simulator. For details, see [Ocient Simulator](#).

**Segment System Catalog Tables**

For specific column definitions of the system catalog tables referenced in these examples, see the Storage section of the System Catalog page.

**Segment Size**

Check the size of segments and return insights about the segment storage allocation and the number of table rows contained in each segment.

```
SELECT current_timestamp,
       sg.segment_type,
       t.name,
       (COUNT(*)) AS num_segments,
       (SUM(s.row_count)) AS total_rows,
       (AVG(s.row_count)) AS avg_rows_per_segment,
       (DOUBLE(SUM(s.segment_size)) / 1000000000) AS total_size_gb,
       (MAX(s.segment_size / 1000000)) AS max_size,
       (MIN(s.segment_size / 1000000)) AS min_size,
       (AVG(s.segment_size) / 1000000) AS avg_size_mb,
       (STDEV(DOUBLE(s.segment_size) / 1000000)) AS stdev_size_mb
FROM   sys.segment_groups AS sg
       JOIN sys.tables t ON t.id = sg.table_id
       JOIN sys.segments s ON s.segment_group_id = sg.id
GROUP BY
       sg.segment_type, t.name
ORDER BY
       total_size_gb DESC,
       total_rows DESC,
       sg.segment_type,
       t.name;
```

*Output*

| "current_timestamp()"   | "segment_type" | "name"             | "num_segments" | "total_rows" | "avg_row" |
|-------------------------|----------------|--------------------|----------------|--------------|-----------|
| 2024-07-31 16:23:22.863 | TKT_SEGMENT    | data_type_coverage | 3              | 100000       | 3333      |

For information on the system catalog tables queried in this example, see `sys.segments` and `sys.tables`.

**Page and Segment Information**



Check the details for pages and segments in the system.

```

WITH cte AS (
SELECT
    COALESCE(v.schema, t.schema) AS schema,
    COALESCE(v.name, t.name) AS name,
    stored_segments.status,
    stored_segments.kind,
    SUM(CASE WHEN segments.segment_type = 'PAGE' THEN 1 ELSE 0 END) AS pagecount,
    SUM(CASE WHEN segments.segment_type = 'TKT_SEGMENT' THEN 1 ELSE 0 END) AS segcount,
    SUM(CASE WHEN segments.segment_type = 'PAGE' THEN segments.row_count ELSE 0 END) AS page_rows,
    SUM(CASE WHEN segments.segment_type = 'TKT_SEGMENT' THEN segments.row_count ELSE 0 END) AS seg_rows,
    MIN(CASE WHEN segments.segment_type = 'PAGE' THEN sg.begin_time ELSE NULL END) AS minpagetime,
    MAX(CASE WHEN segments.segment_type = 'PAGE' THEN sg.end_time ELSE NULL END) AS maxpagetime,
    MIN(CASE WHEN segments.segment_type = 'TKT_SEGMENT' THEN sg.begin_time ELSE NULL END) AS minsegtime,
    MAX(CASE WHEN segments.segment_type = 'TKT_SEGMENT' THEN sg.end_time ELSE NULL END) AS maxsegtime
FROM
    sys.segments
JOIN sys.stored_segments ON
    segments.segment_group_id = stored_segments.segment_group_id
    AND segments.ida_offset = stored_segments.ida_offset
JOIN sys.segment_groups sg ON
    sg.id = segments.segment_group_id
JOIN sys.tables AS t ON
    t.id = sg.table_id
LEFT JOIN sys.views AS v ON
    v.global_dictionary_compression_table_id = t.id
WHERE
    stored_segments.end_osn = 'OSN_INFINITY'
GROUP BY
    1,
    2,
    stored_segments.status,
    stored_segments.kind )
SELECT
    schema,
    name,
    status,
    kind,
    pagecount,
    segcount,
    page_rows,
    seg_rows,
    page_rows + seg_rows AS tot_rows,
    CEILING(page_rows / (pagecount + .000001)) AS rows_per_page,
    CEILING(seg_rows / (segcount + .000001)) AS rows_per_seg,
    to_timestamp(minpagetime / 1000000000) AS minpagetime,
    to_timestamp(maxpagetime / 1000000000) AS maxpagetime,
    to_timestamp(minsegtime / 1000000000) AS minsegtime,
    to_timestamp(maxsegtime / 1000000000) AS maxsegtime,
    NOW()
FROM
    cte
ORDER BY
    schema,
    name;

```

*Output*

| "schema" | "name"             | "status" | "kind" | "pagecount" | "segcount" | "page_rows" | "seg_rows" |
|----------|--------------------|----------|--------|-------------|------------|-------------|------------|
| loading  | data_type_coverage | INTACT   | DISK   | 0           | 3          | 0           | 100000     |

## Storage Spaces on a System

Check all storage spaces on your system.

```
SELECT *
FROM sys.storage_spaces;
```

*Output*

| "id"                                   | "name"             | "is_system_storage_space" | "block_size" |
|----------------------------------------|--------------------|---------------------------|--------------|
| ce9f9c2e-d3fe-4101-a2b2-6f7f2580327b   | ss0                | false                     | 4096         |
| "7277656e-6465-6c20-7761-732068657265" | systemstoragespace | true                      | 4096         |

## Query Management System Catalog Tables

For specific column definitions of the system catalog tables referenced in these examples, see the Monitoring section of the System Catalog page.

## Running Queries

Check all queries currently running on the system.

The output of the `sys.queries` table depends on what database you are logged into and your system role. For details, see [Data Control Language \(DCL\) Statement Reference](#).

```
SELECT
    queries.query_id,
    queries.initial_priority,
    queries.effective_priority,
    queries.user,
    queries.total_time,
    queries.status,
    queries.rows_returned,
    queries.bytes_returned
FROM
    sys.queries
WHERE
    queries.sql NOT LIKE '%sys.queries%';
```

```
SELECT
    q.query_id,
    q.user,
    q.sql,
    q.status,
    "database_name",
    q.bytes_returned
FROM
    sys.queries q;
```

*Output*

| "query_id"                             | "user"       | "sql"           | "status" | "database_name" |
|----------------------------------------|--------------|-----------------|----------|-----------------|
| "66b27dde-17df-403a-82d7-65e34b6fd07f" | admin@system | "example_query" | RUNNING  | test            |

## Recent Queries

Check the most recent queries completed by the system.

The output of the `sys.completed_queries` table depends on what database you are logged into and your system role. For details, see [Data Control Language \(DCL\) Statement Reference](#).

```
SELECT user,
       timestamp_start,
       total_time,
       state,
       reason,
       sql
FROM sys.completed_queries
ORDER BY timestamp_start DESC
LIMIT 10;
```

### Output

| "user"       | "time_start"              | "total_time" | "state" |                                       |
|--------------|---------------------------|--------------|---------|---------------------------------------|
| admin@system | "2024-08-06T21:51:29.284" | 84           | "00000" | The operation completed successfully  |
| admin@system | "2024-08-06T21:45:33.952" | 58           | "00000" | The operation completed successfully  |
| admin@system | "2024-08-06T21:41:04.521" | 205          | "00000" | The operation completed successfully  |
| admin@system | "2024-08-06T21:38:19.740" | 90           | "00000" | The operation completed successfully  |
| admin@system | "2024-08-06T21:37:35.181" | 5            | "42703" | The reference to referenced column i  |
| admin@system | "2024-08-06T21:37:33.710" | 5            | "42703" | The reference to referenced column i  |
| admin@system | "2024-08-06T21:36:03.692" | 11           | "42703" | The reference to referenced column i  |
| admin@system | "2024-08-06T21:35:51.301" | 4            | "42703" | The reference to column 't.name' is i |
| admin@system | "2024-08-06T21:35:10.331" | 5            | "42703" | The reference to column 'sg.segment_  |
| admin@system | "2024-08-06T21:34:42.970" | 208          | "00000" | The operation completed successfully  |

## Database and Table System Catalog Tables

For specific column definitions of the system catalog tables used in these examples, see the Databases section of the System Catalog page.

### Database Sizes

Check the total size of all databases on your system.

```
SELECT TO_CHAR(SUM(sp.size), '999,999,999,999,999') AS totsize,
       d.name
FROM sys.segment_parts AS sp
JOIN sys.segment_groups AS sg
      ON sp.segment_group_id = sg.id
JOIN sys.tables AS t
      ON sg.table_id = t.id
JOIN sys.databases AS d
      ON t.database_id = d.id
GROUP BY d.name
ORDER BY 1 DESC;
```

### Output

| "totsize" | "name" |
|-----------|--------|
| 204276004 | test   |

## Table Sizes

Check the total size of all tables on the system.

```
WITH
cte AS (
    SELECT
        sg.id AS segment_group_id,
        sp.name AS segment_name,
        sp.part_type AS partition_type,
        sp.size AS segment_size,
        sg.segment_type,
        sg.status,
        t.schema,
        t.name
    FROM sys.segment_parts AS sp
    INNER JOIN sys.segment_groups AS sg ON
        sp.segment_group_id = sg.id
    INNER JOIN sys.tables AS t ON
        sg.table_id = t.id ),
ctefinal AS (
    SELECT schema, name, SUM(segment_size) AS total_table_size
    FROM cte
    GROUP BY schema, name )
SELECT SCHEMA, name, total_table_size, SUM(total_table_size)
    OVER( ORDER BY total_table_size DESC ) AS running_total
FROM ctefinal
ORDER BY total_table_size DESC;
```

## Output

| "schema" | "name"             | "total_table_size" | "running_total" |
|----------|--------------------|--------------------|-----------------|
| loading  | data_type_coverage | 204276004          | 204276004       |

## Table Statistics

This query provides a breakdown of table statistics and partition information.

```
WITH
cte1 AS (
    SELECT
        sg.id AS segment_group_id,
        sp.name AS segment_name,
        sp.part_type AS partition_type,
        sp.size AS segment_size,
        sg.segment_type,
        sg.status,
        t.schema,
        t.name
    FROM sys.segment_parts AS sp
    INNER JOIN sys.segment_groups AS sg ON
        sp.segment_group_id = sg.id
    INNER JOIN sys.tables AS t ON
        sg.table_id = t.id ),
cte AS (
    SELECT
        COUNT(*) AS num_segments,
```

```

        SUM(segment_size) AS partition_size,
        partition_type,
        SUM(SUM(segment_size)) OVER(PARTITION BY schema, name) AS total_table_size,
        SUM(COUNT(*)) OVER(PARTITION BY schema, name) AS total_table_segments,
        schema,
        name
FROM cte1
GROUP BY
    partition_type,
    schema,
    name )
SELECT
    name,
    num_segments,
    partition_size,
    partition_type,
    CAST(FLOAT(partition_size)/(total_table_size + 1) * 100
        AS DECIMAL(6,3)) AS percent_of_total,
    total_table_size,
    total_table_segments
FROM cte
ORDER BY
    name,
    partition_size DESC;

```

#### Output

| "name"             | "num_segments" | "partition_size" | "partition_type"       | "percent_of_total" | " |
|--------------------|----------------|------------------|------------------------|--------------------|---|
| data_type_coverage | 3              | 113397760        | parity_data            | 59.827             |   |
| data_type_coverage | 3              | 75599872         | data                   | 39.885             |   |
| data_type_coverage | 3              | 172032           | skip_lists             | 0.091              |   |
| data_type_coverage | 3              | 172032           | copy_skip_lists        | 0.091              |   |
| data_type_coverage | 3              | 90112            | parity_stats           | 0.048              |   |
| data_type_coverage | 3              | 59210            | stats                  | 0.031              |   |
| data_type_coverage | 3              | 12288            | table_of_contents      | 0.006              |   |
| data_type_coverage | 3              | 12288            | parity_index           | 0.006              |   |
| data_type_coverage | 3              | 12288            | copy_table_of_contents | 0.006              |   |
| data_type_coverage | 3              | 8192             | index                  | 0.004              |   |
| data_type_coverage | 3              | 4096             | parity_summary_stats   | 0.002              |   |
| data_type_coverage | 3              | 906              | column_metadata        | 0.000              |   |
| data_type_coverage | 3              | 906              | copy_column_metadata   | 0.000              |   |
| data_type_coverage | 3              | 710              | summary_stats          | 0.000              |   |

For information on the system catalog tables queried in this example, see `sys.segment_groups` and `sys.segment_parts`.

#### Compression

Check the information on the compression for each column in the specified table.

```

SELECT r."table",
       c.NAME,
       r.is_rle_enabled,
       r.is_ne_enabled,
       r.raw_size,
       r.compressed_size,
       r.is_deltadelta_enabled,

```

```

    r.num_deltadelta_blocks,
    r.num_rle_blocks,
    r.num_nerle_blocks,
    r.num_uncompressed_blocks,
    r.num_total_blocks
FROM sys.columns c,
    ( SELECT t.name AS "table",
        t.id,
        cci.is_deltadelta_enabled,
        cci.is_rle_enabled,
        cci.is_ne_enabled,
        cci.raw_size,
        cci.compressed_size,
        cci.num_deltadelta_blocks,
        cci.num_rle_blocks,
        cci.num_nerle_blocks,
        cci.num_uncompressed_blocks,
        cci.num_total_blocks,
        cci.ordinal
      FROM sys.tables AS t,
            sys.columns_compression_info AS cci
      WHERE t.id = cci.table_id
            AND t.name = 'chicago_docsis_arr_idx' ) AS r
WHERE r.id = c.table_id
      AND r.ordinal = c.ordinal;

```

#### Output

| "table"            | "name"        | "is_rle_enabled" | "is_ne_enabled" | "raw_size" | "compressed_size" |
|--------------------|---------------|------------------|-----------------|------------|-------------------|
| data_type_coverage | col_double    | true             | true            | 900000     | 905216            |
| data_type_coverage | col_int       | true             | true            | 500000     | 507904            |
| data_type_coverage | col_tinyint   | true             | true            | 200000     | 204800            |
| data_type_coverage | col_date      | true             | true            | 500000     | 294912            |
| data_type_coverage | col_decimal   | true             | true            | 1100000    | 1105920           |
| data_type_coverage | col_ip        | true             | true            | 1700000    | 1712128           |
| data_type_coverage | col_point     | true             | true            | 1700000    | 1712128           |
| data_type_coverage | col_timestamp | true             | true            | 900000     | 905216            |
| data_type_coverage | col_bigint    | true             | true            | 900000     | 905216            |
| data_type_coverage | col_binary    | true             | true            | 300000     | 307200            |
| data_type_coverage | col_boolean   | true             | true            | 200000     | 106496            |
| data_type_coverage | col_ipv4      | true             | true            | 500000     | 507904            |
| data_type_coverage | col_time      | true             | true            | 900000     | 905216            |
| data_type_coverage | col_float     | true             | true            | 500000     | 507904            |
| data_type_coverage | col_uuid      | true             | true            | 1700000    | 1712128           |
| data_type_coverage | col_smallint  | true             | true            | 300000     | 307200            |

For information on the system catalog table queried in this example, see sys.columns\_compression\_info.

#### Compression for the Entire Table

This query summarizes the total compression of the specified table. In the example, replace the table name and schema name accordingly.

```

SELECT SUM(raw_size),
       SUM(compressed_size)
FROM (

```

```

SELECT t.name,
       c.name AS colname,
       co.raw_size,
       co.compressed_size,
       SUM(co.compressed_size) OVER(PARTITION BY t.schema, t.name) AS table_size,
       100 - ((100 * co.compressed_size) / co.raw_size) AS col_compress_percent,
       DECIMAL(
         FLOAT(co.compressed_size) * 100 / SUM(co.compressed_size)
         OVER(PARTITION BY t.schema, t.name),
         3,
         1
       ) AS percent_of_table
FROM (
  SELECT table_id,
         ordinal,
         raw_size,
         compressed_size
  FROM sys.columns_compression_info
  UNION ALL
  SELECT table_id,
         ordinal,
         raw_size,
         compressed_size
  FROM sys.vl_columns_compression_info
) AS co
JOIN sys.tables AS t ON co.table_id = t.id
JOIN sys.columns AS c ON c.table_id = t.id
  AND co.ordinal = c.ordinal
WHERE t.name = 'data_type_coverage'
  AND t.schema = 'loading'
ORDER BY co.ordinal
) AS abc;

```

*Output*

|                 |                        |
|-----------------|------------------------|
| "sum(raw_size)" | "sum(compressed_size)" |
| -----           | -----                  |
| 73788914        | 73592306               |

For information on the system catalog table queried in this example, see `sys.columns_compression_info`.

*Index Size*

Check the total size of each index across the different segment groups.

In this example, the table has three secondary indexes added before loading data.

```

SELECT p.name AS uuid,
       p.part_type,
       i.name AS index_name,
       SUM(p.size) AS sum_size
FROM (
  SELECT *
  FROM sys.segment_parts
  WHERE part_type LIKE 'index'
) AS p
  INNER JOIN sys.indexes i ON p.name = i.id
GROUP BY 1,2,3

```

```
ORDER BY 1,2;
```

Output

| "uuid"                                 | "part_type" | "index_name" | "sum_size" |
|----------------------------------------|-------------|--------------|------------|
| "669cbcdf-772f-4c97-b250-0cc9e573cd7b" | index       | idx01        | 2527232    |
| cccf6f5e-96a8-4621-b947-f318d120f6a0   | index       | idx03        | 180224     |
| d75ae71a-207e-4723-a86c-ecffc798c7d0   | index       | idx02        | 3190784    |

For information on the system catalog tables queried in this example, see sys.indexes and sys.segment\_parts.

#### Column Sizes

This query finds the size of all columns for the specified table. Before running this query on your system, replace `table_name` in the example.

```
WITH cte AS (
  SELECT
    table_id,
    ordinal,
    compressed_size
  FROM
    sys.columns_compression_info
  UNION ALL
  SELECT
    table_id,
    ordinal,
    compressed_size
  FROM
    sys.vl_columns_compression_info )
SELECT
  ( SELECT t.name
    FROM sys.tables AS t
    WHERE t.id = cte.table_id ) AS table_name,
  ordinal,
  compressed_size,
  ( SELECT name
    FROM sys.columns AS c
    WHERE c.table_id = cte.table_id
      AND c.ordinal = cte.ordinal ) AS name,
  DOUBLE(compressed_size) / SUM(compressed_size) OVER() AS percent_of_total
FROM cte
WHERE
  table_id IN (
    SELECT id
    FROM sys.tables
    WHERE name LIKE 'table_name' )
ORDER BY compressed_size;
```

Output

| "table_name"       | "ordinal" | "compressed_size" | "name"      | "percent_of_total"    |
|--------------------|-----------|-------------------|-------------|-----------------------|
| data_type_coverage | 25        | 8192              |             | 0.0001082748327179    |
| data_type_coverage | 21        | 8192              |             | 0.0001082748327179    |
| data_type_coverage | 2         | 102400            | col_boolean | 0.0013534354089736837 |
| data_type_coverage | 18        | 204800            | col_tinyint | 0.0027068708179473675 |



|                    |    |          |                                      |
|--------------------|----|----------|--------------------------------------|
| data_type_coverage | 19 | 266240   | 0.003518932063331578                 |
| data_type_coverage | 4  | 294912   | col_date   0.0038978939778442096     |
| data_type_coverage | 12 | 307200   | col_smallint   0.004060306226921052  |
| data_type_coverage | 1  | 307200   | col_binary   0.004060306226921052    |
| data_type_coverage | 10 | 507904   | col_ipv4   0.0067130396285094715     |
| data_type_coverage | 7  | 507904   | col_float   0.0067130396285094715    |
| data_type_coverage | 8  | 507904   | col_int   0.0067130396285094715      |
| data_type_coverage | 3  | 699370   | col_char   0.009243673066151615      |
| data_type_coverage | 23 | 783147   | col_varbinary   0.010350965627260875 |
| data_type_coverage | 6  | 905216   | col_double   0.011964369015327365    |
| data_type_coverage | 17 | 905216   | col_timestamp   0.011964369015327365 |
| data_type_coverage | 16 | 905216   | col_time   0.011964369015327365      |
| data_type_coverage | 0  | 905216   | col_bigint   0.011964369015327365    |
| data_type_coverage | 5  | 1105920  | col_decimal   0.014617102416915785   |
| data_type_coverage | 13 | 1712128  | col_point   0.022629440038039995     |
| data_type_coverage | 22 | 1712128  | col_uuid   0.022629440038039995      |
| data_type_coverage | 11 | 1712128  | col_ip   0.022629440038039995        |
| data_type_coverage | 20 | 1784388  | 0.02358451076706771                  |
| data_type_coverage | 24 | 1788957  | col_varchar   0.023644899891907562   |
| data_type_coverage | 14 | 8500000  | col_linestring   0.11234571265894837 |
| data_type_coverage | 9  | 21477958 | col_int_array   0.28387723505517193  |
| data_type_coverage | 15 | 27739482 | col_polygon   0.3666366910682436     |

This output is based on a query using the table `data_type_coverage`.

#### Column Cardinality

This query finds the cardinality of each column in the specified table. Knowing the cardinality is useful for determining which compression scheme to use for a column.

```
SELECT t.schema,
       t.name,
       c.name,
       c.ordinal,
       cc.cardinality
FROM sys.column_cardinalities AS cc
     LEFT JOIN sys.tables AS t ON cc.table_id = t.id
     LEFT JOIN sys.columns AS c ON cc.column_id = c.id
WHERE t.name LIKE 'data_type_coverage'
ORDER BY c.ordinal;
```

#### Output

| "schema" | "name"             | "name_1"      | "ordinal" | "cardinality" |
|----------|--------------------|---------------|-----------|---------------|
| loading  | data_type_coverage | col_bigint    | 0         | 98551         |
| loading  | data_type_coverage | col_binary    | 1         | 48048         |
| loading  | data_type_coverage | col_boolean   | 2         | 2             |
| loading  | data_type_coverage | col_char      | 3         | 95206         |
| loading  | data_type_coverage | col_date      | 4         | 29634         |
| loading  | data_type_coverage | col_decimal   | 5         | 20059         |
| loading  | data_type_coverage | col_double    | 6         | 53546         |
| loading  | data_type_coverage | col_float     | 7         | 75680         |
| loading  | data_type_coverage | col_int       | 8         | 99319         |
| loading  | data_type_coverage | col_int_array | 9         | 5000855       |
| loading  | data_type_coverage | col_int_array | 9         | 99899         |
| loading  | data_type_coverage | col_ipv4      | 10        | 99908         |

|         |                    |                |    |        |  |
|---------|--------------------|----------------|----|--------|--|
| loading | data_type_coverage | col_ip         | 11 | 99875  |  |
| loading | data_type_coverage | col_smallint   | 12 | 51227  |  |
| loading | data_type_coverage | col_point      | 13 | 100000 |  |
| loading | data_type_coverage | col_linestring | 14 | 100000 |  |
| loading | data_type_coverage | col_polygon    | 15 | 99153  |  |
| loading | data_type_coverage | col_time       | 16 | 99902  |  |
| loading | data_type_coverage | col_timestamp  | 17 | 99829  |  |
| loading | data_type_coverage | col_tinyint    | 18 | 256    |  |
| loading | data_type_coverage | col_uuid       | 22 | 99892  |  |
| loading | data_type_coverage | col_varbinary  | 23 | 68834  |  |
| loading | data_type_coverage | col_varchar    | 24 | 93258  |  |
| loading | data_type_coverage |                |    | 10410  |  |
| loading | data_type_coverage |                |    | 2      |  |
| loading | data_type_coverage |                |    | 1      |  |
| loading | data_type_coverage |                |    | 93333  |  |
| loading | data_type_coverage |                |    | 1      |  |

For information on the system catalog table queried in this example, see `sys.column_cardinalities`.

### Cluster and Node System Catalog Tables

For specific column definitions of the system catalog tables referenced in these examples, see the System section of the System Catalog page.

#### Nodes Present on a System

Check the status of all nodes in the system.

```
SELECT name,
       status
FROM sys.nodes;
```

#### Output

| "name"      | "status" |
|-------------|----------|
| -----       | -----    |
| foundation2 | ACCEPTED |
| sql         | ACCEPTED |
| loader      | ACCEPTED |
| foundation1 | ACCEPTED |
| foundation0 | ACCEPTED |

#### Nodes Roles on a System

Check the roles of each node on the system.

```
SELECT
  n.name, n.status,
  STRING_AGG(DISTINCT c.name, ', ' ORDER BY c.name) in_clusters,
  STRING_AGG(DISTINCT sr.service_role_type, ', ' ORDER BY sr.service_role_type) with_roles
FROM
  sys.nodes n
  LEFT JOIN sys.node_clusters nc ON nc.node_id = n.id
  LEFT JOIN sys.clusters c ON c.id = nc.cluster_id
  LEFT JOIN sys.service_roles sr ON sr.node_id = n.id
GROUP BY
  n.name, n.status
ORDER BY
  n.name;
```

*Output*

| "name"      | "status" | "in_clusters"        | "with_roles"           |               |            |       |
|-------------|----------|----------------------|------------------------|---------------|------------|-------|
| -----       | -----    | -----                | -----                  | -----         | -----      | ----- |
| foundation0 | ACCEPTED | "foundation_cluster" | foundation_cluster-vm" | "health       | lts        |       |
| foundation1 | ACCEPTED | "foundation_cluster" | foundation_cluster-vm" | "health       | lts        |       |
| foundation2 | ACCEPTED | "foundation_cluster" | foundation_cluster-vm" | "health       | lts        |       |
| loader      | ACCEPTED |                      | "health                | streamloader" |            |       |
| sql         | ACCEPTED | initial-vm           | "admin                 | health        | operatorvm |       |

## Node Status

Check the operational status, software versions, and assigned roles of your nodes. Possible status values include ACTIVE, STARTING, STOPPING, ERROR, UNKNOWN, or UNREACHABLE.

```
SELECT name,
       operational_status,
       software_version,
       ARRAY_AGG(service_role_type)
FROM sys.node_status AS ns
     LEFT JOIN sys.nodes AS n ON ns.node_id = id
     LEFT JOIN sys.service_roles AS sr ON sr.node_id = n.id
GROUP BY name,
         operational_status,
         software_version;
```

*Output*

| name        | operational_status | software_version | array_agg(service_role_type:) |
|-------------|--------------------|------------------|-------------------------------|
| -----       | -----              | -----            | -----                         |
| loader      | Active             | 24.0.0           | ['health','streamloader']     |
| foundation2 | Active             | 24.0.0           | ['health','lts','operatorvm'] |

## Foundation Nodes on a Specific Cluster

Check for all Foundation Nodes on your system clusters.

```
SELECT n.name,
       r.name
FROM sys.nodes AS n,
     ( SELECT *
       FROM sys.clusters AS c,
           sys.node_clusters AS nc
       WHERE c.cluster_type = 'Foundation'
             AND c.id = nc.cluster_id ) AS r
WHERE r.node_id = n.id;
```

*Output*

| foundation2 | foundation_cluster |
|-------------|--------------------|
| -----       | -----              |
| foundation0 | foundation_cluster |
| foundation1 | foundation_cluster |

## Drive Status

Check the status of all drives assigned to nodes in the system.

```

SELECT
    n.name AS node_name,
    s.node_id,
    s.id AS serial_number,
    s.pci_address,
    s.device_status,
    s.device_model
FROM
    sys.nodes AS n
JOIN sys.storage_device_status AS s
    ON n.id = s.node_id;

```

Output

| "node_name" | "node_id"                              | "serial_number"                        |          |
|-------------|----------------------------------------|----------------------------------------|----------|
| foundation1 | "6a96b307-57bc-477f-952c-e6cab38a7922" | "72062649-396c-4424-9da5-c90cf0589e00" | /var/opt |
| foundation1 | "6a96b307-57bc-477f-952c-e6cab38a7922" | ee2f0013-c7b7-4750-833e-3d717af85001   | /var/opt |
| loader      | "83f0ecf9-9a31-4a56-85b9-7eb610447185" | ff3186fb-502f-417c-bcab-8b7aa3de8800   | /var/opt |
| loader      | "83f0ecf9-9a31-4a56-85b9-7eb610447185" | "11ff09e2-a6c1-4f76-ae96-3a97eaf02a01" | /var/opt |
| foundation0 | "9330a0b3-b3b7-4503-949c-043b196c0cc4" | "6156962f-fcf6-4299-bbd7-2618fc6f1d00" | /var/opt |
| foundation0 | "9330a0b3-b3b7-4503-949c-043b196c0cc4" | e6a4b24f-0d49-4704-b7ce-18af163c0701   | /var/opt |
| sql         | "1198a86f-549d-4736-ac31-1ba54a2b02e7" | "37203a6c-9deb-47cf-8194-95c4eecd7300" | /var/opt |
| sql         | "1198a86f-549d-4736-ac31-1ba54a2b02e7" | "0921d62f-a31c-4681-9a56-82cb1450a401" | /var/opt |
| foundation2 | babadc9c-f2a3-4512-ab9a-bab3fd88a544   | "842881df-b71c-4cfc-847b-19517e8b2800" | /var/opt |
| foundation2 | babadc9c-f2a3-4512-ab9a-bab3fd88a544   | fc0e490a-a295-491d-b6a6-8b3c153a7301   | /var/opt |

## Roles, Privileges, and Service Class System Catalog Tables

For specific column definitions of the system catalog tables referenced in these examples, see the User Management section of the System Catalog page.

### Group Assignment for Users

Check for all users on the system and their corresponding assigned group.

```

SELECT g.name AS group_name,
    r.user_name
FROM sys.groups AS g,
    ( SELECT u.user_name,
        ug.group_id
      FROM sys.users AS u,
        sys.user_groups AS ug
      WHERE u.id = ug.user_id ) AS r
ORDER BY g.name ASC;

```

Output

| group_name | user_name |
|------------|-----------|
| DBA        | Humphrey  |

### User Roles

Check for all users on the system and their assigned role.

```

SELECT rl.name AS role_name,
    r.user_name
FROM sys.roles AS rl,

```

```
( SELECT u.user_name,
      ur.role_id
  FROM sys.users AS u,
       sys.user_roles AS ur
 WHERE u.id = ur.user_id ) AS r
ORDER BY rl.name ASC;
```

Output

| role_name              | user_name |
|------------------------|-----------|
| system administrator   | Humphrey  |
| system analyst         | Grimey    |
| database administrator | Charles   |
| database analyst       | Edna      |

### User Privileges

Check the privileges for the specific table name. Before running on your system, replace `username` in the example.

```
SELECT t.schema AS db_name,
      t.name AS table_name,
      r.privilege
FROM sys.tables AS t,
     ( SELECT object_type,
           object_id,
           privilege
       FROM sys.PRIVILEGES
       WHERE grantee = 'username' ) AS r
WHERE r.object_id = t.id
ORDER BY t.schema, t.name ASC;
```

Output

| db_name | table_name         | privilege |
|---------|--------------------|-----------|
| loading | data_type_coverage | VIEW      |
| loading | data_type_coverage | SELECT    |

### Privileges for the Specific Table

Check the granted privileges across all database objects of the specific type, such as `GROUP`, `TABLE`, or `DATABASE`. In this example, the query checks for `TABLE` privileges.

```
SELECT p.grantee,
      t.schema,
      t.name,
      p.privilege
FROM sys.tables AS t,
     sys.privileges AS p
WHERE object_type = 'TABLE'
      AND p.object_id = t.id;
```

Output

| admin@system | admin@system | example_table | DELETE  |
|--------------|--------------|---------------|---------|
| admin@system | admin@system | example_table | DROP    |
| admin@system | admin@system | example_table | SYSAUTH |

|              |              |               |        |  |
|--------------|--------------|---------------|--------|--|
| admin@system | admin@system | example_table | LOAD   |  |
| admin@system | admin@system | example_table | INSERT |  |
| admin@system | admin@system | example_table | ALTER  |  |
| admin@system | admin@system | example_table | SELECT |  |
| admin@system | admin@system | example_table | VIEW   |  |

### Groups Assigned to Service Classes

Check for the service class assignment for all groups.

```
SELECT g.name,
       sc.name
FROM sys.groups AS g,
      sys.service_classes AS sc
WHERE g.service_class_id = sc.id;
```

#### Output

|             |               |  |
|-------------|---------------|--|
| group       | sc_name       |  |
| ----- ----- |               |  |
| analyst     | high_priority |  |
| dbadmin     | low_priority  |  |

### Service Class Settings

Check for the settings for the specified service class. Before running on your system, replace the **high priority** service class name in the example.

For details about service class settings, see [Workload Management and Service Classes](#).

```
SELECT *
FROM sys.groups AS g,
      sys.service_classes AS sc
WHERE g.service_class_id = sc.id
      AND sc.name = 'high_priority';
```

#### Output

|                                      |         |                                      |                  |
|--------------------------------------|---------|--------------------------------------|------------------|
| id                                   | name    | database_id                          | service_class_id |
| ----- ----- ----- -----              |         |                                      |                  |
| e9c92a1b-32a8-4f88-ab72-485ff8b24f53 | analyst | e80010d7-6f26-438c-8461-11af309ed8a3 | 83b76f87-6634-4c |

### Service Classes for the Specific User

Check for the service class of the specific user. Before running on your system, replace the **jmack@test** username in this example with the specific username that you want to query.

```
SELECT r2.name,
       sc.name,
       r2.user_name
FROM sys.service_classes AS sc,
      ( SELECT g.name,
              g.service_class_id,
              r.user_name
        FROM sys.groups AS g,
              ( SELECT u.user_name,
                    ug.group_id
                FROM sys.users AS u,
                    sys.user_groups AS ug
                WHERE u.id = ug.user_id
                    AND u.user_name = 'jmack@test' ) AS r ) AS r2;
```

*Output*

| group   | service_class | user_name  |
|---------|---------------|------------|
| analyst | Default       | jmack@test |
| analyst | high_priority | jmack@test |

**Data Pipelines**

## Pipeline Events

While your pipeline is running, the pipeline generates events in the `sys.pipeline_events` system catalog table to mark significant checkpoints when something has occurred. In this example, you run the `CREATE PIPELINE` and `START PIPELINE` SQL statements on the pipeline named `my_pipeline` and let the pipeline complete. As expected, the `CREATED`, `STARTED`, and `COMPLETED` events appear. For details about the lists of files, see the `sys.pipeline_files` system catalog table.

Pipeline events include messages from many different tasks. These events are the background processes that execute across different Loader Nodes during pipeline operation. For details about tasks, you can query the `sys.tasks` and `sys.subtasks` system catalog tables.

```
SELECT *
FROM sys.pipeline_events
ORDER BY event_timestamp;
```

*Output*

| pipeline_id                          | task_id                              | user_id                    |
|--------------------------------------|--------------------------------------|----------------------------|
| d8848467-9262-4bd6-84a7-1380c95f8b8b |                                      | dde90d1b-bfcf-4b48-a251-b9 |
| d8848467-9262-4bd6-84a7-1380c95f8b8b |                                      | dde90d1b-bfcf-4b48-a251-b9 |
| d8848467-9262-4bd6-84a7-1380c95f8b8b |                                      |                            |
| d8848467-9262-4bd6-84a7-1380c95f8b8b |                                      |                            |
| d8848467-9262-4bd6-84a7-1380c95f8b8b | 61007c85-6744-42e7-9401-66e1a594e990 |                            |
| d8848467-9262-4bd6-84a7-1380c95f8b8b | 61007c85-6744-42e7-9401-66e1a594e990 |                            |
| d8848467-9262-4bd6-84a7-1380c95f8b8b | 49c17f79-0108-4b05-aae5-9ae9a13abaca |                            |

Fetches 7 rows

## Pipeline Metrics

Find metrics that measure the performance and activity of a pipeline in the `sys.pipeline_metrics` system catalog table. This table contains one row for each metric at a specified point in time. A common use of this data is to plot metric values over time to examine the behavior of a pipeline.

**Metric Types**

Metrics are either instantaneous or incremental:

- Instantaneous metrics reflect the current value of a counter at the time of the metric collection. These values can increase or decrease based on the current state or rate of the metric.
- Incremental metrics increment over time and reflect the cumulative value for a specified counter.

**Example**

The first snapshot of a metric appears when it is initialized. Afterward, the metric is updated every 10 seconds. The `updated_at` column shows the metric collection time.

For example, if you are interested in the number of the transformed record bytes, you can run this query. The value increases over time as the pipeline runs.

```
SELECT *
FROM sys.pipeline_metrics
WHERE name = 'count.record.bytes.transformed'
ORDER BY updated_at;
```

Output

| pipeline_id                          | extractor_task_id                    | partition_id | sink_index |
|--------------------------------------|--------------------------------------|--------------|------------|
| 0f51f1c0-f251-4407-8393-0f9bcbf18f28 | 034cf49d-364b-42b5-ad0a-37b7c237f16e |              |            |
| 0f51f1c0-f251-4407-8393-0f9bcbf18f28 | 034cf49d-364b-42b5-ad0a-37b7c237f16e |              |            |
| 0f51f1c0-f251-4407-8393-0f9bcbf18f28 | 034cf49d-364b-42b5-ad0a-37b7c237f16e |              |            |
| 0f51f1c0-f251-4407-8393-0f9bcbf18f28 | 034cf49d-364b-42b5-ad0a-37b7c237f16e |              |            |
| 0f51f1c0-f251-4407-8393-0f9bcbf18f28 | 034cf49d-364b-42b5-ad0a-37b7c237f16e |              |            |

Fetches 5 rows

### Metric Scope

Each metric has a scope that defines its uniqueness. Be careful not to aggregate metrics across different scopes.

The scope columns are `pipeline_id`, `extractor_task_id`, `partition_id`, and `sink_index`. If the value of any of these columns is NULL, the scope applies to all values in that dimension.

For example, because the `partition_id` and `sink_index` columns are NULL for the metric `count.record.bytes.transformed`, its scope applies to the pipeline indicated by the `pipeline_id` and `extractor_task_id` across all sinks and partitions.

An `extractor_task_id` is the internal identifier used when the Ocident System executes a pipeline process on a Loader Node. A pipeline might have many of these processes across time and loaders.

### Example

Retrieve the most recent snapshot for each metric and aggregate the values to produce a single value for a pipeline.

```
SELECT pipeline_id, name, SUM(most_recent_value) AS total_value
FROM (
  SELECT DISTINCT * FROM (
    SELECT pipeline_id, extractor_task_id, partition_id, name,
    first_value (value) OVER
      (PARTITION BY pipeline_id, extractor_task_id, partition_id, name
      ORDER BY updated_at DESC) AS most_recent_value
    FROM sys.pipeline_metrics
    WHERE name LIKE 'count%'
  )
)
GROUP BY pipeline_id, name;
```

Output

| pipeline_id                          | name                           | total_value |
|--------------------------------------|--------------------------------|-------------|
| 0c7b32f1-37ca-43f7-96b6-69c266c978a5 | count.file.total               | 64          |
| 0c7b32f1-37ca-43f7-96b6-69c266c978a5 | count.record.bytes.transformed | 33644961    |
| 0c7b32f1-37ca-43f7-96b6-69c266c978a5 | count.file.processed           | 64          |
| 0c7b32f1-37ca-43f7-96b6-69c266c978a5 | count.record.sent              | 100000      |
| 0c7b32f1-37ca-43f7-96b6-69c266c978a5 | count.record.durable           | 100000      |



Fetches 5 rows

This output contains a single row for each metric for each pipeline. If you run a pipeline with 64 files and 100,000 records, the output might look like this. This table explains each column in the output.

### Related Links

[System Catalog](#)

[Data Integrity and Storage](#)

[Query Analysis](#)

[Database, Tables, Views, and Indexes](#)

[Cluster and Node Management](#)

[Users, Groups, and Service Classes](#)

---

## 11 Connection Driver Reference

Ocient® includes multiple ways to connect and issue statements or queries to the database. The most common drivers used for communicating with Ocient are the JDBC driver and the pyocient module. Many third-party tools use the JDBC driver for business intelligence, visualization, and data exploration. These drivers can also be used in custom programs by leveraging the programming language's appropriate mechanisms to use the driver, for example, using a JDBC driver directly in Java®.

- [JDBC Manual](#)
  - [Commands Supported by the Ocient JDBC CLI Program](#)
  - [Supported JDBC Classes and Methods](#)
  - [Data Extract Manual](#)
- [Ocient Python Module: pyocient](#)

### 11.1 JDBC Manual

The Ocient® JDBC Driver and command-line interface (CLI) enable you to connect to Ocient using a JDBC connection. Ensure that you meet the prerequisites before using the Ocient JDBC Driver. Then, invoke the CLI program, configure options, and connect to a database using the driver.

You can also use the data extract tool to extract a result set to delimited files in the target location. For details about data extracting, see [Data Extract Manual](#).

For a list of commands available in the JDBC CLI, see [Commands Supported by the Ocient JDBC CLI Program](#).

#### Prerequisites

| Software | Version                                                                         |
|----------|---------------------------------------------------------------------------------|
| Ocient   | Use the latest Ocient System version.                                           |
| OS       | {{Windows}}, {{Linux}}, or {{macos}}. Use the latest version of each OS system. |
| {{Java}} | See theVersion Compatibility.                                                   |

#### Driver Features

The Ocient JDBC connector supports these features as of the current version.

|                       |                                                                                          |
|-----------------------|------------------------------------------------------------------------------------------|
| Unicode Support       | UTF-8                                                                                    |
| Isolation Levels      | Ocient does not support transactions at this time.                                       |
| Data Types            | Supports all OcientData Types.                                                           |
| Security / Encryption | Uses SSL/TLS to connect to the Ocient System. TLS protocol is available as aJDBC Manual. |

#### Invoke the Ocient JDBC CLI Program

If your system meets all the necessary prerequisites, you can run the JDBC CLI by using the Ocient JDBC JAR file. To do this, follow these steps.

Go to the Ocient Maven repository for all JDBC versions. For more information on which version to pick, see the [Version Compatibility](#) page.

For the JDBC version you want to use, download the JAR file with dependencies. This JAR file follows the format: `ocient-jdbc4-<version number>-jar-with-dependencies.jar`.

Move this JAR file to the directory where your Ocient system is installed.

From the shell terminal, run this command to launch the JDBC CLI.

```
java -classpath <path_to_jar_with_dependencies> com.ocient.cli.CLI [<username> [<password>]]
```

This example runs JDBC version 2.10.

```
java -classpath ./ocient-jdbc4-2.10-jar-with-dependencies.jar com.ocient.cli.CLI testuser testpassword
```

After launching, the JDBC CLI prompts you to enter your username and password.

```
Username: admin@system
Password: admin
```

The interface changes to the Ocient CLI.

```
Ocient> _
```

Connect to your system from the JDBC using a connection string.

Assuming the standard port 4050, a self-signed certificate, and the SQL Node IP address 10.10.1.1, you can connect to the system database with the following connecting string.

```
Ocient> connect to jdbc:ocient://10.10.1.1:4050/system;
```

The CLI responds with a connection message.

```
Connected to jdbc:ocient://10.10.1.1:4050/system
Ocient> _
```

Now that you are connected to your system, you can execute any queries or commands.

For Java version 1.8.0\_144, download and install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8.

## JDBC CLI Configuration Options

The JDBC CLI reads a configuration file consisting of key-value pairs located at `~/.ocient-cli-configuration` with this format.

```
key1=value1
key2=value2
...
keyN=valueN
```

This option is supported.

| Option                             | Comments                                                                                                                                                           | Default                                                         |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|
| <code>cliIdleTimeoutMinutes</code> | Configures the idle timeout for the CLI. The CLI rejects subsequent commands and exits the process after <code>cliIdleTimeoutMinutes</code> minutes of inactivity. | 0 if left unspecified or the configuration file does not exist. |

For supported commands, see [Commands Supported by the Ocient JDBC CLI Program](#).

## Use the Ocient JDBC Driver in Java Programs

First, you must load the Ocient driver class with this statement in a Java program using the JDBC driver.

```
Class.forName("com.ocient.jdbc.JDBCdriver");
```

The driver class is located in the JDBC driver JAR file named `ocient-jdbc4.jar` and must be available in the CLASSPATH defined for the program.

### Connection Encryption (SSL/TLS)

The JDBC driver can use SSL/TLS to connect to Ocient, causing all traffic to be encrypted. Specify the `tls` property on the connect statement to enable TLS support. The `tls` property supports these values.

**unverified** Traffic on the connection is encrypted, but no verification is done on the certificate received from the Ocient System.

**on** Traffic is encrypted, and the JDBC client must be able to verify the certificate received from the Ocient System.

The TLS **on** mode requires that the client knows the **Certificate Authority** that signed the certificate provided by the Ocient System. Typically, this mode requires either that the certificate is signed by a *well known* certificate authority, or the Certificate Authority certificate has been imported into the truststore of the Java system. The Java `keytool` utility is used to manipulate a Java truststore.

[Secure Connections Using TLS](#) discusses how you can configure user-defined certificates for the Ocient System.

### Sample Java Program Using the Ocient JDBC Driver

This sample program illustrates how to use the Ocient JDBC driver to connect to a database, build a prepared SQL statement, execute the query, and loop through the result set.

```
public class OcientJDBCExample {
    public static void main(final String args[]) {
        Class.forName("com.ocient.jdbc.JDBC4Driver");
        Properties props = new Properties();
        props.setProperty("user", "username");
        props.setProperty("password", "pwd");
        props.setProperty("force", "true");
        String url = "jdbc:ocient://192.168.121.82:4050/db";
        Connection conn = DriverManager.getConnection(url, props);
        PreparedStatement pstmt = conn.prepareStatement(
            "select l_orderkey from tpch.lineitem where l_linenumber = ?");
        Pstmt.setInt(1, 4);
        ResultSet rs = pstmt.executeQuery();
        while(rs.next()){
            // do something with row
        }
        rs.close();
        pstmt.close();
        conn.close();
        return;
    }
}
```

For supported classes and methods, see [Supported JDBC Classes and Methods](#).

### Related Links

[Connect Using JDBC](#)

[Data Extract Manual](#)

[Commands Supported by the Ocient JDBC CLI Program](#)

[Supported JDBC Classes and Methods](#).

### 11.1.1 Data Extract Manual

The data extract tool is a part of the JDBC driver to unload data. You can execute the tool directly from the JDBC CLI. The tool extracts a result set to delimited files in the target location. To invoke the JDBC CLI, see the [JDBC Manual](#).

To use the data extract tool, you must have JDBC version 2.63 or higher.

#### General Command Structure

Here is the general structure of an extract command.

```
EXTRACT TO <location_type>{ LOCAL | S3 } [OPTIONS([param=value [,...]])] AS <query>
```

The command is case insensitive. Each extract command must start with **EXTRACT TO**. The location type needs to follow and must be either **LOCAL** or Amazon® Web Services<sup>SM</sup> (AWS<sup>SM</sup>) **S3**. You can enclose additional options within a pair of parenthesis following the word **OPTIONS**. Note that location type is the only required option. Next, the query follows the word **AS**.

This example is a simple general command structure.

```
EXTRACT TO LOCAL OPTIONS(
  file_prefix="/home/user/out/data_",
  file_extension=".csv"
)
AS SELECT c1 FROM sys.dummy10;
```

For supported options, see [Data Extract Manual](#).

#### Specify Options, Quoting, and Escaping Quotes

Here is the general format of options.

```
key1 = value1, key2 = value2, ... , keyN = valueN
```

You need to follow certain guidelines when you specify options. Keys (option names) can only consist of alphanumeric characters and are unquoted. Values can be either quoted (with the reserved character ") or unquoted. If values are unquoted, they can only contain alphanumeric characters. If the value has a non-alphanumeric character, you must quote it with the reserved character ". Note that the single quote character does not work.

```
OPTIONS(file_prefix = "/path/to/dir/result", header_mode = none, file_extension = ".csv")
```

To use the reserved quote character " as an argument, you must escape it with the backslash character \. To use \ as an argument, you must escape it with another \. This code illustrates both of these scenarios.

```
OPTIONS(field_optionally_enclosed_by = "\"", escape = "\\")
```

#### File Naming Conventions

When you use the extract tool, the tool produces a number of files. If the tool extracts with a single thread, the tool names files in this convention: {file\_prefix}\_{file\_number}{file\_extension}{gzip\_extension}

1. **file\_prefix**: Option specified by the user.
2. **file\_number**: This part of the convention is 0 if all results go into one file. However, if **MAX\_ROWS\_PER\_FILE** is set, then rows are placed into one file until that limit is hit. Then, another file generates with an incremented **file\_number**. File number starts from 0.
3. **file\_extension**: Option specified by the user.
4. **gzip\_extension**: If you specify gzip compression, then the tool adds the .gzip suffix.

If you use the multithreaded extract, then the naming convention is: {file\_prefix}{thread\_number}\_{file\_number}{file\_ext

`file_prefix`, `file_extension`, `gzip_extension` is still determined with the set options. `thread_number` is the number of the thread going from 0 to N - 1 where N is the number of threads specified. The file number is now calculated per thread with the number of rows given to that thread. Recall that rows are distributed to threads in a round-robin order, starting with thread 0. So, thread 0 receives the rows 0, N, 2N, 3N, etc. Thread 1 receives the rows 1, N + 1, 2N + 1, 3N + 1, etc.

### Extract Options

| Option            | Usage     | Comments                                                                                                                                                                                                                                                                                                                              | Default                 |
|-------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| LOCATION_TYPE     | LOCAL, S3 | Dictates where the results are extracted to. LOCAL extracts the results to the local machine. {{AWS}} S3 extracts the results to S3. When using S3, other options are required. See S3 options for details.                                                                                                                           | None, must be specified |
| FILE_PREFIX       | LOCAL, S3 | Dictates the prefix used on the results. When extracting to LOCAL, this is the prefix used to determine the path of the results. This can be a relative or full path. When extracting to S3, this is the prefix for the key. In either case, additional file numbers and file extensions are added to generate the complete filename. | results-                |
| FILE_EXTENSION    | LOCAL, S3 | The file extension given to each result file.                                                                                                                                                                                                                                                                                         | .csv                    |
| MAX_ROWS_PER_FILE | LOCAL, S3 | If non-zero, the MAX_ROWS_PER_FILE modifier splits the results into files with maximum MAX_ROWS_PER_FILE in each file.                                                                                                                                                                                                                | NULL                    |
| COMPRESSION       | LOCAL, S3 | Compression type to use. Supported compression types are: NONE ≤ — ≤ No compression GZIP ≤ — ≤ GZip compression BZIP2 ≤ — ≤ bzip2 compression XZ ≤ — ≤ xz compression                                                                                                                                                                 | NONE                    |
| RECORD_DELIMITER  | LOCAL, S3 | Delimiter to use between records. This supports {{Java}} strings, so special characters can be input using escape characters. UTF-16: \u[utf-16 value] or Octal [octal value].                                                                                                                                                        | \n                      |

| Option          | Usage     | Comments                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Default                                                                       |
|-----------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| FIELD_DELIMITER | LOCAL, S3 | Delimiter to use between fields within a record. This supports Java strings, so special characters can be input using escape characters. UTF-16: \u[utf-16 value] or Octal [octal value].                                                                                                                                                                                                                                                                                                                                                                                        | ,                                                                             |
| HEADER_MODE     | LOCAL, S3 | Dictates how to manage headers in result files. Supported values are NONE, ALL_FILES, and FIRST_FILE. NONE — The tool writes all output files without an additional header. ALL_FILES — The tool adds column names as a header in the first row of each output file. Each file has at most MAX_ROWS_PER_FILE + 1 total rows. FIRST_FILE — The tool adds column names as a header in the first row of the first output file. The tool does not add the header to subsequent files. Each file has at most MAX_ROWS_PER_FILE total rows, inclusive of the header in the first file. | NONE                                                                          |
| NULL_FORMAT     | LOCAL, S3 | Format string to use for writing NULL values to the output files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | "" (empty string)                                                             |
| ENCODING        | LOCAL, S3 | Encoding used when writing out data to files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | The default charset of the system, as determined by the Oracle documentation. |
| ESCAPE          | LOCAL, S3 | Character used for escaping quoted fields. Set this to the NULL character (\0) to indicate that the escape character is not specified.                                                                                                                                                                                                                                                                                                                                                                                                                                           | "                                                                             |

| Option                       | Usage     | Comments                                                                                                                                                                                                                                                                    | Default                 |
|------------------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| FIELD_OPTIONALLY_ENCLOSED_BY | LOCAL, S3 | Sometimes, you need to surround fields in a character. For example, the field might have a literal comma. Generally, this character is also known as the quote character. Set this option to the NULL character (\0) to indicate that the quote character is not specified. | "                       |
| BINARY_FORMAT                | LOCAL, S3 | The format with which to encode the BINARY data type. Supports UTF-8, Hexadecimal, and Base64.                                                                                                                                                                              | Hexadecimal             |
| COMPRESSION_BLOCK_SIZE       | LOCAL, S3 | The number of bytes that comprise each block to be compressed; larger blocks result in better compression at the expense of more RAM usage when compressing.                                                                                                                | 4194304                 |
| COMPRESSION_LEVEL            | LOCAL, S3 | An integer value [-1, 9]. Use -1 for the GZip default compression level, 0 for "no compression", or a value [1-9] where 1 indicates fastest compression and 9 indicates best compression.                                                                                   | 1                       |
| NUM_COMPRESSION_THREADS      | LOCAL, S3 | The number of threads to use for compression. Leave unspecified for the default value.                                                                                                                                                                                      | \$(number_of_cores * 2) |
| ESCAPE_UNQUOTED_VALUES       | LOCAL, S3 | Dictates whether to write escape sequences in unquoted values. Only applicable when FIELD_DELIMITER is set to ,.                                                                                                                                                            | false                   |



| Option            | Usage     | Comments                                                                                                                                                                                                                                                                                                                      | Default                                                                  |
|-------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|
| INPUT_ESCAPED     | LOCAL, S3 | Dictates whether the input is already escaped. When this option is set to true, the tool does not add escape sequences, and data is written without changes to the output file. Only applicable when FIELD_DELIMITER is set to ,. Ensure that data is properly escaped, otherwise the extract might produce invalid CSV data. | false                                                                    |
| QUOTE_ALL_FIELDS  | LOCAL, S3 | Dictates whether all written fields are enclosed with quotes. When this option is set to true, the tool encloses all fields with the FIELD_OPTIONALLY_ENCLOSED_BY character.                                                                                                                                                  | false                                                                    |
| BUCKET            | S3        | S3 bucket to use. Ignored if extracting locally. If extracting to S3, this argument is required.                                                                                                                                                                                                                              | None, required for S3.                                                   |
| AWS_KEY_ID        | S3        | AWS Key ID. If empty, the CLI uses the Java AWS SDK default credentials provider chain documented here.                                                                                                                                                                                                                       | ""                                                                       |
| AWS_SECRET_KEY    | S3        | AWS Secret Key. If empty, the CLI uses the Java AWS SDK default credentials provider chain documented here.                                                                                                                                                                                                                   | ""                                                                       |
| REGION            | S3        | S3 region to upload to. Ignored when extracting to LOCAL.                                                                                                                                                                                                                                                                     | US_EAST_2                                                                |
| ENDPOINT          | S3        | Endpoint for S3 upload. Required when extracting to S3. Ignored when extracting to LOCAL. Documentation on endpoint formatting.                                                                                                                                                                                               | None, required for S3                                                    |
| PATH_STYLE_ACCESS | S3        | Whether path style access should be used to access a bucket.                                                                                                                                                                                                                                                                  | true for JDBC version 3.4.1 and false for JDBC version 3.4.0 and earlier |

| Option                    | Usage | Comments                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Default |
|---------------------------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| TRANSLATE_CHARACTERS_MODE | 3     | Character Mode to use for translating characters. Supported values are CHAR and HEX. The tool performs character translation only if you specify TRANSLATE_CHARACTERS_FROM and TRANSLATE_CHARACTERS_TO. The tool replaces the Nth character in TRANSLATE_CHARACTERS_FROM with the Nth character in TRANSLATE_CHARACTERS_TO in the extracted records. When TRANSLATE_CHARACTERS_MODE is set to CHAR, TRANSLATE_CHARACTERS_FROM, and TRANSLATE_CHARACTERS_TO must be equal-length strings of UTF-8 characters. For example: TRANSLATE_CHARACTERS_MODE="CHAR", TRANSLATE_CHARACTERS_MODE is set to HEX, TRANSLATE_CHARACTERS_FROM, and TRANSLATE_CHARACTERS_TO must be comma-separated lists of hexadecimal UTF-8 code points with the same number of list elements. For example: TRANSLATE_CHARACTERS_MODE="HEX", TRANSLATE_ | CHAR    |
| TRANSLATE_CHARACTERS_FROM | 3     | Sequence of UTF-8 characters in the source data to translate to a corresponding character in the TRANSLATE_CHARACTERS_TO option. See the TRANSLATE_CHARACTERS_MODE option for the expected format.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | "       |

| Option                    | Usage     | Comments                                                                                                                                                                         | Default |
|---------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| TRANSLATE_CHARACTERS_FROM | LOCAL, S3 | Sequence of UTF-8 characters to use as a replacement for the characters included in TRANSLATE_CHARACTERS_FROM. See the TRANSLATE_CHARACTERS_MODE option for the expected format. | ""      |

### Examples

OPTIONS(...) is only required when you specify additional options. This example unloads the results of `SELECT c1 FROM sys.dummy10` to the local machine at the relative path `result-0.csv`.

```
EXTRACT TO LOCAL
AS SELECT c1 FROM sys.dummy10;
```

`sys.dummy` creates a virtual table with the specified number for rows. For details, see [Generate Tables Using sys.dummy](#).

This example extracts the results of `SELECT c1 FROM sys.dummy10` to the local machine at the absolute path `/home/user/out/data_0.csv`.

```
EXTRACT TO local OPTIONS(
  file_prefix="/home/user/out/data_",
  file_extension=".csv"
)
AS SELECT c1 FROM sys.dummy10;
```

The result set has 100 rows, and `MAX_ROWS_PER_FILE` is set to 10. Thus, the tool writes 10 files to S3 with the keys `query_results/query_0/_results_{file_number}.csv.gz`, where `file_number` ranges from 0 to 9, inclusive.

### Related Links

[Connect Using JDBC](#)

[JDBC Manual](#)

#### 11.1.2 Commands Supported by the Ocient JDBC CLI Program

Commands are not case-sensitive.

For DDL statements, see [Data Definition Language \(DDL\) Statement Reference](#).

All commands, except `QUIT`, must end with a semicolon at the command line.

You can cancel any query that does not update data by pressing `CTRL+C`.

### CANCEL

Cancels the current query that is still running on the system.

### Syntax

```
CANCEL uuid;
```

| Parameter | Data Type | Description                                                                                                                                                                 |
|-----------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uuid      | String    | A query Universally Unique Identifier (UUID) that is currently running. UseCommands Supported by the Ocient JDBC CLI Programor search thesys.queriestable to find the UUID. |

## CONNECT

Connects to a SQL Node. For details about connection steps, see [Connect Using JDBC](#).

### Syntax

```
CONNECT TO <url> [ USER username USING password ]
```

The <url> parameter must follow this format:

```
jdbc:ocient://node_address1:port1[,node_address2:port2[,...]]/database_name[;property=value[;...]]
```

If you do not specify the **username** and **password**, then the database uses the default credentials specified at startup.

### JDBC URL parameters

| Parameter     | Description                                                                                                                                                                |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| node_address  | A hostname or IP address.The database attempts to connect using eachDNSName:Portpair in the comma-delimited list from left to right until a successful connection happens. |
| port          | Standard JDBC port number is4050.For details about ports, seeOcient Simulator.                                                                                             |
| database_name | The identifier used for your database.                                                                                                                                     |

### Additional JDBC URL Properties

The JDBC URL can include these optional properties in the key-value format **property=value**. Separate each property pair by semicolons.

URL properties are case-insensitive.

To authenticate using Single Sign-On, set the **handshake** property to SSO and leave the **user** and **password** properties empty. For example:  

```
jdbc:ocient://DNSName:Port[,IP2:Port2,...]/databaseName;handshake=SSO;user=;password=;
```

| Parameter     | Description                                                                                 |
|---------------|---------------------------------------------------------------------------------------------|
| defaultSchema | Default schema.                                                                             |
| force         | If set to true (case-sensitive), this parameter disables load-balancing for the connection. |

| Parameter                | Description                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| handshake                | Specifies the handshake protocol used for the connection. Supported options include: "CBC", "GCM", "SSO" or "GCM" (Galois/Counter Mode). This is the default encryption and is the recommended password encryption algorithm. "CBC" (Cipher Block Chaining) for password encryption. "SSO" Single Sign-On.                                                                                                            |
| identityprovider         | An SSO integration established in the database. For details, see Cluster and Node Management.                                                                                                                                                                                                                                                                                                                         |
| logfile                  | The filename to use for JDBC tracing.                                                                                                                                                                                                                                                                                                                                                                                 |
| loglevel                 | If set to ERROR (warnings and errors only) or DEBUG (verbose tracing) and logfile is also set, JDBC tracing is enabled. This parameter is case-sensitive.                                                                                                                                                                                                                                                             |
| longQueryThreshold       | Estimated query runtime in milliseconds before deeper query optimization runs. 0— Use database server default. -1— Never run deeper optimization.                                                                                                                                                                                                                                                                     |
| maxRows                  | Maximum allowed result set size in the number of rows.                                                                                                                                                                                                                                                                                                                                                                |
| maxTempDisk              | Maximum allowed temp disk usage as a percentage (0 - 100).                                                                                                                                                                                                                                                                                                                                                            |
| maxThreadsPerResultSet   | The maximum number of threads the client uses to fetch rows from the server per Result Set, as defined in the official JDBC documentation. In this context, a thread represents a logical connection to the database. Applications that create and operate on Statement objects concurrently might find value in setting this property. Defaults to 0, which effectively creates an unbounded Result Set thread pool. |
| maxTime                  | Maximum allowed runtime of a query in seconds before it is canceled on the server.                                                                                                                                                                                                                                                                                                                                    |
| networkTimeout           | Network connection timeout in milliseconds. If unspecified, this defaults to 10000 milliseconds.                                                                                                                                                                                                                                                                                                                      |
| parallelism              | Limits a query to running on a specified number of cores on each CPU.                                                                                                                                                                                                                                                                                                                                                 |
| password                 | The password for the user.                                                                                                                                                                                                                                                                                                                                                                                            |
| priority                 | Query priority. This sets the priority for queries to run on the server side. If unspecified, this defaults to 1.0.                                                                                                                                                                                                                                                                                                   |
| priorityAdjustFactor     | The default query priority adjustment value. For details, see Commands Supported by the Ocint JDBC CLI Program.                                                                                                                                                                                                                                                                                                       |
| priorityAdjustTime       | The default frequency to adjust the query priority. For details, see Commands Supported by the Ocint JDBC CLI Program.                                                                                                                                                                                                                                                                                                |
| serviceClassName         | Specifies the name of the service class to use for the database session.                                                                                                                                                                                                                                                                                                                                              |
| ssoNumericAddress        | Specifies the SSO callback URL as 127.0.0.1. When this value is false, the URL is localhost. The default value is false.                                                                                                                                                                                                                                                                                              |
| ssoOAuthCodeCallbackPort | If ssoOAuthFlow=authorizationCode, this property specifies the port that the SSO authorization uses. If unspecified, the default value is 7050.                                                                                                                                                                                                                                                                       |

| Parameter                           | Description                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ssoOAuthFlow</code>           | This property is only applicable if SSO authorization is enabled. Forces the driver to use either the “authorizationCode” or “deviceGrant” flow to establish a Single Sign-On session. If this property is not provided, the {{Ocint}} System uses the “authorizationCode” flow when a web browser is available to the client and the “deviceGrant” flow when a web browser is not available to the client. |
| <code>ssoDebugMode</code>           | When set to <code>true</code> , the system records additional log messages related to SSO.                                                                                                                                                                                                                                                                                                                  |
| <code>ssoOktaNativeTokenPath</code> | The path to the {{OKTA}} Native Token (AES-256GCM JWE). The path is relative to your home directory (i.e. ‘~/’).                                                                                                                                                                                                                                                                                            |
| <code>ssoSslCallback</code>         | Specifies whether to use HTTPS instead of HTTP for the SSO callback URL. The default value is <code>false</code> .                                                                                                                                                                                                                                                                                          |
| <code>ssoTimeoutSeconds</code>      | The number of seconds before the SSO connection request times out. The default value is 60.                                                                                                                                                                                                                                                                                                                 |
| <code>statementPooling</code>       | ≤ This property has been removed as of Ocint JDBC version 2.104 and later. When set to <code>ON</code> , recently used statements are cached. Set to <code>OFF</code> when using a third-party application that implements statement pooling                                                                                                                                                                |
| <code>tls</code>                    | Can be set to <code>unverified</code> or <code>on</code> . Enables SSL/TLS encryption for the connection.                                                                                                                                                                                                                                                                                                   |
| <code>timeoutMillis</code>          | Number of milliseconds before cancellable operations are timed out and killed by the driver. 0 means no timeout. Default: 0                                                                                                                                                                                                                                                                                 |
| <code>user</code>                   | The identifier of the user.                                                                                                                                                                                                                                                                                                                                                                                 |

## DESCRIBE TABLE

Lists the columns of the specified table and its associated data types.

### Syntax

```
DESCRIBE TABLE [schema.]table [verbose]
```

| Parameter            | Description                                                                                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>table</code>   | A table identifier.                                                                                                                          |
| <code>schema</code>  | A qualifying schema name for the specified table. If a qualifying schema is not specified for the table, the default name schema is assumed. |
| <code>verbose</code> | If specified, the Ocint System prints the full column metadata for the specified table as returned by <code>GetColumns</code> .              |

## DESCRIBE VIEW

Returns the query text used to create the specified view.

### Syntax

```
DESCRIBE VIEW [schema.]view [verbose]
```

| Parameter | Description                                                                                                                                 |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------|
| view      | A view identifier.                                                                                                                          |
| schema    | A qualifying schema name for the specified table.If a qualifying schema is not specified for the table, the default name schema is assumed. |
| verbose   | If specified, the Ocient System prints the full column metadata for the specified table as returned by GetColumns.                          |

## EXTRACT

Extracts a result set in delimited files to a specified location. For details about using this command, see [Data Extract Manual](#).

### Syntax

```
EXTRACT TO location
  { LOCAL | S3 }
  [ OPTIONS ( [ param=value [ ,... ] ] ) ] AS query
```

## GET JDBC VERSION

Returns the JDBC version of the driver.

### Syntax

```
GET JDBC VERSION
```

## GET SCHEMA

Retrieves the default name schema for tables.

### Syntax

```
GET SCHEMA
```

## KILL

Issues a hard kill command to the virtual machine to terminate a query running on the system. Use only if CANCEL fails.

### Syntax

```
KILL uuid
```

| Parameter | Description                                                                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| uuid      | A query UUID that is currently running. UseCommands Supported by the Ocient JDBC CLI Program, or search thesys.queriestable to find the UUID. |

## LIST ALL QUERIES

List all the current running queries across all the cmdComp servers. This is the equivalent of running `SELECT * FROM sys.queries` on all cmdComp servers.

### Syntax

```
LIST ALL QUERIES
```

## LIST INDEXES

Lists the indexes on the specified table and its columns.

**Syntax**

```
LIST INDEXES [schema.]table [verbose]
```

| Parameter | Description                                                                                                                                 |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------|
| table     | A table identifier.                                                                                                                         |
| schema    | A qualifying schema name for the specified table.If a qualifying schema is not specified for the table, the default name schema is assumed. |
| verbose   | If specified, the Ociant System prints the full column metadata for the specified table as returned by getIndexInfo.                        |

**LIST SYSTEM TABLES**

Lists all system catalog tables in the database.

**Syntax**

```
LIST SYSTEM TABLES [verbose]
```

| Parameter | Description                                                                             |
|-----------|-----------------------------------------------------------------------------------------|
| verbose   | If verbose is specified, the full table metadata, as returned by getTables, is printed. |

**LIST TABLES**

Lists all tables (in all schemata) in the database.

**Syntax**

```
LIST TABLES [verbose]
```

| Parameter | Description                                                                             |
|-----------|-----------------------------------------------------------------------------------------|
| verbose   | If verbose is specified, the full table metadata, as returned by getTables, is printed. |

**LIST VIEWS**

Lists all views (in all schemata) in the database.

**Syntax**

```
LIST VIEWS [verbose]
```

| Parameter | Description                                                                            |
|-----------|----------------------------------------------------------------------------------------|
| verbose   | If verbose is specified, the full table metadata, as returned by getViews, is printed. |

**OUTPUT GIS KML**

Outputs the next query in KML format to the specified filename. This flag affects only the next query.

This syntax prints out all GIS types to the KML file, and non-GIS types are added to the description of elements in the same row. You can upload this KML file to certain visualization tools.

**Syntax**



OUTPUT GIS KML <filename>

| Parameter | Description                    |
|-----------|--------------------------------|
| filename  | A filename for the KML output. |

## OUTPUT NEXT QUERY

Output the next query in CSV format to the specified filename. This flag only affects the next query. All subsequent queries print out their results.

Add APPEND to add output to the specified file instead of overwriting it.

### Syntax

OUTPUT NEXT QUERY filename [APPEND]

| Parameter | Description                    |
|-----------|--------------------------------|
| filename  | A filename for the CSV output. |

## PERFORMANCE

Enables different levels of query result output for measuring benchmarks by using different performance options.

### Syntax

PERFORMANCE [ NETWORK | DATABASE | OFF ]

| Parameter | Description                                                                                                                                                                                                                             |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OFF       | This is the default mode. The system executes a query through all normal stages, including the database, client network, and client processing.                                                                                         |
| NETWORK   | The system executes a query through the database and client network but discards rows before they are processed by the client (nothing is printed to stdout). Use this mode when you think the client network is hindering performance. |
| DATABASE  | The system executes a query through the database only. The database does not send any matching rows to the client. Use this mode when you think the database query execution is affecting performance.                                  |

PERFORMANCE ON mode has been removed as of JDBC version 3.0.

## QUIT

Closes the command-line interface.

### Syntax

QUIT

## REPORT LOG ON

Enables Apache® Log4J 2™ -based metrics collection and redirects reporting events to the configured appenders.

The **REPORT LOG ON** command requires JDBC driver version 2.88 or later.

The Log4j configuration determines the destination for report logs. The Ocient logger name is `com.ocient.util.ReportLogger`. For details about configuring Log4j, see the Log4j documentation.

The default Log4j configuration routes report logs to the `RollingFileAppender`. The default configuration includes these settings:

#### Properties

- `fileName="logs/${sys:ocient.jdbc.sessionId:-unknown}/report.log"`
- `filePattern="logs/${sys:ocient.jdbc.sessionId:-unknown}/%d{yyyy-MM-dd}-%i-report.log.gz"`

#### Rollover Policies

- `TimeBasedTriggeringPolicy`
- `SizeBasedTriggeringPolicy (100MB)`

You can retrieve the default configuration from the `ocient-jdbc4` jar file by using an archive utility like **unzip**. For example:

```
unzip -p <path-to-ocient-jdbc4> log4j2.xml
```

#### Syntax

```
REPORT LOG ON [ interval ]
```

| Parameter             | Description                                                                             |
|-----------------------|-----------------------------------------------------------------------------------------|
| <code>interval</code> | Optional. Sets the Log4j reporting interval in seconds. If unspecified, defaults to 30. |

#### REPORT LOG OFF

Disables file-based metrics collection.

The **REPORT LOG OFF** command requires JDBC driver version 2.88 or later.

#### Syntax

```
REPORT LOG OFF
```

#### SELECT

Retrieves the appropriate result set of a SQL **SELECT** statement.

#### Syntax

```
SQL select statement
```

#### SET MAXROWS

Sets the maximum amount of rows allowed in a result set. Applies only to the statement object that executed the command.

#### Syntax

```
SET MAXROWS num_of_rows
```

| Parameter                | Description                                                                                                                                                                                                                                                                                                            |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>num_of_rows</code> | The maximum amount of rows allowed in a result set. If a query returns more rows than the specified value, the result set is truncated by <code>num_of_rows</code> . To disable this setting, you can either set the maximum rows to 0 or use the <code>RESET</code> keyword (e.g., <code>SET MAXROWS RESET;</code> ). |

## SET ADJUSTFACTOR

Adjusts the query priority by a specified percentage amount at every interval set by the **Commands Supported by the Ocient JDBC CLI Program** command.

The `SET ADJUSTFACTOR` command requires JDBC driver version 2.52 or later.

### Syntax

```
SET ADJUSTFACTOR priorityAdjustFactor
```

| Parameter                         | Description                                                                                                                                                                                                                                                          |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>priorityAdjustFactor</code> | A percentage amount to raise or lower the query priority. Accepted values are in the range of [-1,1]. At every interval set by the <code>priorityAdjTime</code> , the query priority is multiplied by $1 + \text{priorityAdjustFactor}$ to get a new priority value. |

## SET ADJUSTTIME

Modifies how frequently the query priority is adjusted during the course of execution.

The `SET ADJUSTTIME` command requires JDBC driver version 2.52 or later.

### Syntax

```
SET ADJUSTTIME priorityAdjTime
```

| Parameter                    | Description                                                |
|------------------------------|------------------------------------------------------------|
| <code>priorityAdjTime</code> | The frequency for adjusting the query priority in seconds. |

## SET MAXTEMPDISK

Sets the maximum percentage of temporary disk used by queries. Applies only to the statement object that executed the command.

### Syntax

```
SET MAXTEMPDISK percentage
```

| Parameter  | Description                                                                                                                                                                                               |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| percentage | The maximum percentage of temporary disk used by queries. Any queries exceeding this threshold are killed. To disable this setting, use the <b>RESET</b> keyword (e.g., <b>SET MAXTEMPDISK RESET</b> ;;). |

## SET MAXTIME

Sets the maximum query time. Applies only to the statement object that executed the command.

### Syntax

```
SET MAXTIME seconds
```

| Parameter | Description                                                                                                                                                                                         |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| seconds   | The maximum query time. Queries longer than this are killed. To disable this setting, you can either set the maximum time to 0 or use the <b>RESET</b> keyword (e.g., <b>SET MAXTIME RESET</b> ;;). |

## SET PARALLELISM

Sets a limit on the number of cores on each CPU that can be allocated to running the query. Applies only to the statement object that executed the command.

The **parallelism** value limits a query to running on the specified number of cores on each CPU.

### Syntax

```
SET PARALLELISM parallelism
```

| Parameter   | Description                                                                                                                                                                                                                   |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| parallelism | Limits a query to running on a specified number of cores on each CPU. To disable this setting, you can either set the <b>parallelism</b> value to -1 or use the <b>RESET</b> keyword (e.g., <b>SET PARALLELISM RESET</b> ;;). |

## SET PRIORITY

Sets the scheduling priority to use for queries. Applies only to the statement object that executed the command.

### Syntax

```
SET PRIORITY priority
```

| Parameter | Description                                                                                                                                                                                      |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| priority  | The scheduling priority to use for queries. Set from 0 to 100. To disable this setting, you can either set priority to -1 or use the <b>RESET</b> keyword. (e.g., <b>SET PRIORITY RESET</b> ;;). |

## SET SCHEMA

Changes the default schema for tables. Applies only to the statement object that executed the command.

### Syntax

SET SCHEMA schema

| Parameter | Description                        |
|-----------|------------------------------------|
| schema    | The default schema for new tables. |

SET SERVICECLASS

Limits a query to using the specified service class. Applies only to the statement object that executed the command.

#### Syntax

SET SERVICECLASS service\_class\_name

| Parameter          | Description                                                                                                                                                  |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| service_class_name | The service class that a query is limited to using. To disable this setting, you can use the <b>RESET</b> keyword. (e.g., <b>SET SERVICECLASS RESET</b> ;) . |

SOURCE

Reads and executes commands found in the specified file.

#### Syntax

SOURCE filename

| Parameter | Description               |
|-----------|---------------------------|
| filename  | A valid filename or path. |

TIMING

Enables or disables reporting the execution time of each query.

#### Syntax

TIMING { ON | OFF }

#### Related Links

[JDBC Manual](#)

System Catalog

### 11.1.3 Supported JDBC Classes and Methods

The following Java® Classes and Methods are supported by the Ocient® JDBC Driver for use in Java programs.

| Java Class / Interface | Unsupported Methods or Comments                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DriverManager          | There are no unsupported methods. Get a connection with <code>getConnection().getConnection(String, Properties)</code> . The string is a URL and must be of the form: <code>jdbc:ociant://IP:Port/databaseName[:property=value;...]</code> . Properties is a Properties object with the supported properties “user” and “password” (required), and “force”, “loglevel”, and “logfile” (optional). The meanings of these properties is described under the CONNECT CLI command. Any properties specified on the URL will be added to the properties object passed to <code>getConnection()</code> . If a property exists in both places, the value on the URL will take precedence.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Connection             | <code>createArrayOf()</code> <code>createBlob()</code> <code>createClob()</code> <code>createNClob()</code> <code>createSQLXML()</code> <code>createStruct()</code> <code>prepareCall()</code> <code>releaseSavepoint()</code> <code>rollback()</code> <code>setNetworkTimeout()</code> <code>setSavepoint()</code> <code>unwrap()</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Statement              | <code>addBatch()</code> <code>clearBatch()</code> <code>closeOnCompletion()</code> <code>executeBatch()</code> <code>getGeneratedKeys()</code> <code>getMoreResults()</code> <code>setCursorName()</code> <code>setEscapeProcessing()</code> <code>setMaxFieldSize()</code> <code>setPoolable()</code> <code>unwrap()</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| PreparedStatement      | <code>getParameterMetaData()</code> <code>setArray()</code> <code>setAsciiStream()</code> <code>setBinaryStream()</code> <code>setBlob()</code> <code>setCharacterStream()</code> <code>setClob()</code> <code>setNCharacterStream()</code> <code>setNClob()</code> <code>setNString()</code> <code>setRef()</code> <code>setRowId()</code> <code>setSQLXML()</code> <code>setUnicodeStream()</code> <code>setURL()</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| ResultSet              | <code>absolute()</code> <code>afterLast()</code> <code>beforeFirst()</code> <code>cancelRowUpdates()</code> <code>deleteRow()</code> <code>first()</code> <code>getAsciiStream()</code> <code>getBinaryStream()</code> <code>getBlob()</code> <code>getCharacterStream()</code> <code>getClob()</code> <code>getCursorName()</code> <code>getNCharacterStream()</code> <code>getNClob()</code> <code>getNString()</code> <code>getRef()</code> <code>getRowId()</code> <code>getSQLXML()</code> <code>getUnicodeStream()</code> <code>getURL()</code> <code>insertRow()</code> <code>isLast()</code> <code>last()</code> <code>moveToCurrentRow()</code> <code>moveToInsertRow()</code> <code>previous()</code> <code>refreshRow()</code> <code>relative()</code> <code>unwrap()</code> <code>updateArray()</code> <code>updateAsciiStream()</code> <code>updateBigDecimal()</code> <code>updateBinaryStream()</code> <code>updateBlob()</code> <code>updateBoolean()</code> <code>updateByte()</code> <code>updateBytes()</code> <code>updateCharacterStream()</code> <code>updateClob()</code> <code>updateDate()</code> <code>updateDouble()</code> <code>updateFloat()</code> <code>updateInt()</code> <code>updateLong()</code> <code>updateNCharacterStream()</code> <code>updateNClob()</code> <code>updateNString()</code> <code>updateNull()</code> <code>updateObject()</code> <code>updateRef()</code> <code>updateRow()</code> <code>updateRowId()</code> <code>updateShort()</code> <code>updateSQLXML()</code> <code>updateString()</code> <code>updateTime()</code> <code>updateTimestamp()</code> |
| ResultSetMetaData      | <code>unwrap()</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| DatabaseMetaData       | <code>unwrap()</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

## Related Links

[Connect Using JDBC](#)

## 11.2 Ociant Python Module: pyocient

The Ociant® pyocient Python® module supports direct connections to the Ociant System through a command line interface (CLI) as well as Python programs that follow the Python Database API Specification 2.0. To download,

install, and use the pyociant CLI, see [Connect Using pyociant](#).

The pyociant package has these supported features and functionality.

### Prerequisites

To use the pyociant client, you should have this software with the corresponding version installed.

| Software | Version                                                                         |
|----------|---------------------------------------------------------------------------------|
| Ociant   | All Ociant versions are supported.                                              |
| OS       | {{Windows}}, {{Linux}}, or {{macos}}. Use the latest version of each OS system. |
| Python   | Version 3.8 or later.                                                           |

### Module Features

The pyociant connector supports these features as of the current version.

|                       |                                                                                   |
|-----------------------|-----------------------------------------------------------------------------------|
| Unicode Support       | UTF-8                                                                             |
| Isolation Levels      | Ociant does not support transactions at this time.                                |
| Data Types            | Supports all OciantData Types.                                                    |
| Security / Encryption | GCM, CBC, and SSO protocols are supported. TLS protocol is optional as a setting. |

### pyociant CLI

You can make a connection by invoking the pyociant module in a command-line interface with a connection string using this syntax.

#### Syntax

```
$ pyociant [named_arguments] dsn [sql]
```

For details on the parts of a connection string, see these sections.

- [Ociant Python Module: pyociant](#)
- [Ociant Python Module: pyociant](#)
- [Ociant Python Module: pyociant](#)
- [Ociant Python Module: pyociant](#)

### Connect Using a DSN

Required. A Data Source Name (DSN) supplies the user credentials, host, port, and database to establish a connection. This string can also include various optional parameters.

#### Syntax

```
ociant://<user>:<password>@<host>:[<port>] [/<database>] [?'<parameters>'&...]
```

A DSN consists of these arguments.

| Argument   | Definition                                                                     |
|------------|--------------------------------------------------------------------------------|
| <user>     | Required. An Ociant username.                                                  |
| <password> | Required. The password associated with your username for connection to Ociant. |

| Argument     | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <host>       | Required. The Ocident host address. You can specify multiple hosts if they are separated by commas. The system attempts to connect to multiple hosts in order, starting with the leftmost host. For example, <code>host1</code> is the first for the connection attempt in this case: <code>oci://someone:somepassword@host1,host2:4051/mydb</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <port>       | The port number used by your host. The default is 4050.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <database>   | The name of the database for the connection. The default is <code>system</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <parameters> | One or more optional parameters for the connection in the format <code>parameter=value</code> . Supported parameters include: <code>tls</code> : TLS protocol security. You can set the value to <code>off</code> , <code>unverified</code> , or <code>on</code> . The default is <code>off</code> . <code>force</code> : Values are <code>true</code> or <code>false</code> . If you set this value to <code>true</code> , this argument forces the connection to stay on this server. The default is <code>false</code> . <code>handshake</code> : Security handshake protocol. Supported values include <code>cbc</code> . By default, the system does not use <code>cbc.identityprovider</code> : An SSO integration established in the database. For details, see Cluster and Node Management. |

### DSN Example

When you successfully connect with a DSN, your command line switches to interactive mode.

```
$ pyocient oci://user:password@myhost:4050/mydb?tls=on&force=true
Ocident Database
System Version: 22.0, Client Version 1.0.13
>
```

In interactive mode, you can enter SQL commands and queries directly at the command line. This example shows a simple `SELECT` query that you enter at the command line.

```
> SELECT * FROM my_table LIMIT 1;
[
  {
    "time_x": "2022-06-15T13:31:42.824000",
    "col_001": 1.0,
    "col_002": 2.0,
    "col_003": 3,
    "col_004": 4,
    "keyid": 5
  }
]
```

### SQL Queries in the Connection String

Instead of executing queries and commands in the CLI, you can also include SQL scripts directly in your connection string.

A connection string can contain multiple queries as long as each is delineated by semicolons.

You must enclose any SQL statements in a connection string in quotes and specify the statements after the DSN portion.

### Example



This example shows a SQL query embedded in a pyocient connection string. The query processes and returns its values without launching the interactive mode.

```
$ pyocient ocient://user:password@myhost:4050/mydb?tls=on&force=true "SELECT * FROM my_table LIMIT 1"

[
  {
    "time_x": "2022-06-15T13:31:42.824000",
    "col_001": 1.0,
    "col_002": 2.0,
    "col_003": 3,
    "col_004": 4,
    "keyid": 5
  }
]
```

### Use Optional Named Arguments

You can add these optional arguments to a connection string.

```
[-h] [-o OUTFILE | -n] [-c CONFIGFILE | --noconfig]
[-i INFILE] [-l {critical,error,warning,info,debug}]
[--logfile LOGFILE] [-t] [--format {json | table | csv}]
[--nocolor]
```

### Named Argument Definitions

| Named Argument               | Definition                                                                                                                                                                                                                                     |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -o   --outfile OUTFILE       | pyocient records all SQL output to a specified output file. You must also include the file path to the output file, enclosed in quotation marks. The Ocint System saves output values to the file after query completion.                      |
| -n   --noout                 | Prevent all output results.                                                                                                                                                                                                                    |
| -c   --configfile CONFIGFILE | Direct pyocient to use a configuration file. You must include the path location and name of the configuration file, enclosed in quotation marks.                                                                                               |
| --noconfig                   | Direct pyocient not to use a configuration file.                                                                                                                                                                                               |
| -i   --infile INFILE         | Direct pyocient to use an input file that contains SQL statements.                                                                                                                                                                             |
| -l   --loglevel LOGLEVEL     | Sets the logging level. Logging level options include: CRITICAL   ERROR   WARNING   INFO   DEBUG. Defaults to CRITICAL. For details, see Log Monitoring.                                                                                       |
| --logfile LOGFILE            | Direct pyocient to record log events in a specified log file. You must include the path location and name of the log file, enclosed in quotation marks.                                                                                        |
| -t, --time                   | pyocient outputs the execution time for all queries and commands.                                                                                                                                                                              |
| -f, --format                 | Specifies the output format for all SQL queries. Format options include: JSON   TABLE   CSV. JSON— {{JavaScript}} Object Notation. TABLE— A table rendered in ASCII characters. CSV— Comma-separated values format. The default value is JSON. |
| --nocolor                    | When you use pyocient in interactive mode, the interface does not color SQL keywords.                                                                                                                                                          |

## Additional *pyocient* CLI Commands

When running in a CLI, *pyocient* recognizes these commands in addition to the SQL statements and queries.

| Command                                                                           | Description                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>CONNECT TO &lt;dsn&gt; USER &lt;username&gt; USING &lt;password&gt;;</code> | Allows switching to a different DSN or user during an interactive session. The DSN must be enclosed in quotation marks. Otherwise, the command follows the normal format explained in <i>Ociant Python Module: pyocient</i> . You can specify the username and password either in the DSN string or with the <code>USER</code> and <code>USING</code> keywords. |
| <code>SOURCE '&lt;file&gt;';</code>                                               | Executes the input statements from the specified file. The filename must be in single quotes.                                                                                                                                                                                                                                                                   |
| <code>SET FORMAT TABLE;</code>                                                    | Sets the output format. Format options include <code>JSON</code> , <code>TABLE</code> , and <code>CSV</code> . Defaults to <code>JSON</code> . <code>JSON</code> — Javascript Object Notation. <code>TABLE</code> — A table rendered in ASCII characters. <code>CSV</code> — Comma-separated values format.                                                     |
| <code>QUIT;</code>                                                                | Quits the CLI. Alternatively, you can exit the interface by using <code>Ctrl + D</code> .                                                                                                                                                                                                                                                                       |

## *pyocient* API

This Python database API conforms to the Python Database API Specification 2.0, and you can use it to access the Ociant System.

You can also execute this module as a main function, in which case it acts as a primitive CLI for the database.

When you execute this module as a main function, the API takes a connection string in the DSN format. The connection string can also include one or more query strings that execute. For information on using a DSN to connect, see *Ociant Python Module: pyocient*.

The API returns output in JSON format by default.

The database returns any warnings and sends them to the Python warnings module. By default, that module sends warnings to standard output, however you can change the behavior by using that module.

## *pyocient.Connection*

### Constructor

```
class pyocient.Connection(
    dsn=None, user=None, password=None,
    host=None, database=None, tls=None,
    handshake=None, identityprovider=None,
    force=None, configfile=None, session=None)
```

A connection to Ociant. This class is normally constructed by executing the module `connect()`, but you can also construct it directly.

You can specify connection parameters using the `dsn` argument, other keyword arguments, or a mix of both. If the system receives multiple arguments for the same parameters, the keyword arguments take precedence and override the `dsn` value.

Definitions for the connection parameters are listed in *Ociant Python Module: pyocient*. Other parameters are:

- **configfile** — The name of a configuration file in .ini format, where each section either uses default connection settings or a pattern that matches the host or database. Sections are matched in order, so highly detailed sections should precede less detailed sections.
- **session** — A string of an SSO security token.

This code provides an example of an .ini configuration file.

```
[DEFAULT]
tls = unverified

; This matches the specific host and database.
[foo.ocient.com/somedb]
user = panther
password = pink

; This matches the specific host.
[foo.ocient.com]
user = panther
password = pink

; This matches any host in the ocient.com domain.
[*.ocient.com]
user = tom
password = jerry
database = mice

; This matches any host in the ocient.com domain using the default database.
[*.ocient.com]
user = tom
password = jerry
```

The currently supported parameter is:

**tls** — The values are `on`, `off`, or `unverified` in the DSN. You can also set TLS by using `Connection.TLS_NONE`, `Connection.TLS_UNVERIFIED`, or `Connection.TLS_ON` as a keyword parameter.

## Connection Object Methods

### `close()`

Close the connection. Any subsequent queries on the closed connection fail.

### `commit()`

Commit any pending transaction to the database.

### `cursor()`

Return a new cursor object for this connection.

## `pyocient.Cursor`

### Constructor

```
class pyocient.Cursor(conn)
```

A database cursor that can manage a query and its returned results.

Cursors are normally created by executing `cursor()` on a connection, but you can also create them directly by providing a connection.

### Cursor Object Methods

#### **close()**

Close this cursor. The current result set is closed, but you can reuse the cursor for subsequent `execute()` calls.

#### **execute(operation, parameters=None)**

Prepare and execute a database operation (query or command).

Parameters might be provided as a mapping and are bound to variables in the operation. Variables are specified in Python extended format codes, for example: `WHERE name=%(name)`

#### **executemany(operation, parameterlist)**

Prepare a database operation (query or command) and then execute it against all parameter sequences or mappings found in the sequence `parameterlist`.

Parameters might be provided as a mapping and are bound to variables in the operation. Variables are specified in Python extended format codes, for example: `WHERE name=%(name)`

#### **fetchone()**

Fetch the next row of a query result set. The function returns a single row or `None` when no more data rows are available.

#### **fetchmany(size=None)**

Fetch the next set of rows of a query result and return a sequence of sequences as a list of tuples. The database returns an empty sequence when no more rows are available.

The number of rows to fetch per call is specified by the `size` parameter. If this parameter is not specified, the array size of the cursor determines the number of rows to fetch. The method tries to fetch as many rows as indicated by the `size` parameter. If this is not possible due to the specified number of rows not being available, the function returns fewer rows.

#### **fetchall()**

Fetch all (remaining) rows of a query result and return them as a sequence of sequences (e.g., a list of tuples). Note that the array size attribute of the cursor can affect the performance of this operation.

#### **arraysize**

This read-and-write attribute specifies the number of rows to fetch at a time with [Ociant Python Module: pyocient](#). The default value is 1, meaning the attribute fetches a single row at a time.

#### **tables(schema='%', table='%')**

Retrieve the database tables. By default, the database returns all tables.

#### **system\_\_tables(table='%')**

Retrieve the database system tables.

#### **views(view='%')**

Retrieve the database views.

**columns(schema='%', table='%', column='%')**

Retrieve the database columns.

**indexes(schema='%', table='%')**

Retrieve the database indexes.

**getTypeInfo()**

Retrieve the database type information.

**fetchval()**

The `fetchval()` convenience method returns the first column of the first row if there are results, otherwise it returns `None`.

**resolve\_new\_endpoint(new\_host, new\_port)**

Handles mapping to a secondary interface based on the secondary interface mapping saved on this connection.

**Arguments:**

`new_host[string]`: the new host to be remapped `new_port[int]`: the new port to be remapped

**Returns:**

`[tuple(string, int)]`: The actual endpoint to connect to in the format: (host, port).

**redirect(new\_host, new\_port)**

Redirects to the proper secondary interface given a new endpoint.

**Arguments:**

`new_host[string]`: the new host to be redirected `new_port[int]`: the new port to be redirected

**Returns:**

`[Connection]`: A new connection.

**refresh()**

Refresh the session associated with this connection. The server returns a new server session identifier and security token.

**setinputsizes(sizes)**

Used before an **Ociant Python Module: *pyocient*** call to set memory sizes for the specified operation.

For details on usage, see the Python API documentation.

**setoutputsize(size [, column])**

Used before an **Ociant Python Module: *pyocient*** call to set a column buffer size for fetches of columns with large data types, such as LONG, BLOB, etc.

For details on usage, see the Python API documentation.

## Type Objects and Constructors

These constructors create objects to hold special values to comply with defined types in a database schema. When you pass these objects to the cursor methods, the module detects the proper type of input parameter and binds it accordingly.

For details, see the Python documentation on Type Objects and Constructors.

### **Date(*year, month, day*)**

This function constructs an object that holds a date value.

### **Time(*hour, minute, second*)**

This function constructs an object that holds a time value.

### **Timestamp(*year, month, day, hour, minute, second*)**

This function constructs an object that holds a timestamp value.

### **DateFromTicks(*ticks*)**

This function constructs an object that holds a date value from the specified ticks value (number of seconds after the epoch; see the documentation of [the standard Python time module](#) for details).

### **TimeFromTicks(*ticks*)**

This function constructs an object that holds a time value from the specified ticks value (number of seconds after the epoch; see the documentation of the standard Python time module for details).

### **TimestampFromTicks(*ticks*)**

This function constructs an object that holds a timestamp value from the specified ticks value (number of seconds after the epoch; see the documentation of the standard Python time module for details).

### **Binary(*string*)**

This function constructs an object capable of holding a binary (long) string value.

### **STRING type**

This type object describes columns in a database that are string-based (e.g., CHAR).

### **BINARY type**

This type object describes (long) binary columns in a database (e.g., LONG, RAW, BLOB).

### **NUMBER type**

This type object describes numeric columns in a database.

### **DATETIME type**

This type object describes date and time columns in a database.

### **ROWID type**

This type object describes the “Row ID” column in a database.

**Exceptions****Error**

Exception that is the base class of all other error exceptions.

```
exception Error(reason, sql_state='58005', vendor_code=-100)
```

**Warning**

Exception that is the base class of all other warning exceptions.

```
exception Warning(reason='', sql_state='00000', vendor_code=0)
```

**InterfaceError**

Exception raised for errors related to the database interface rather than the database itself.

```
exception InterfaceError(reason, sql_state='58005', vendor_code=-100)
```

**DatabaseError**

Exception raised for errors that are related to the database.

```
exception DatabaseError(reason, sql_state='58005', vendor_code=-100)
```

**InternalError**

Exception raised when the database encounters an internal error, e.g., the cursor is not valid anymore.

```
exception InternalError(reason, sql_state='58005', vendor_code=-100)
```

**OperationalError**

Exception raised for errors related to the operation of the database and not necessarily under the control of the programmer, e.g., an unexpected disconnect occurs, the data source name is not found, etc.

```
exception OperationalError(reason, sql_state='58005', vendor_code=-100)
```

**ProgrammingError**

Exception raised for programming errors, e.g., table not found, syntax error in the SQL statement, wrong number of parameters specified, etc.

```
exception ProgrammingError(reason, sql_state='58005', vendor_code=-100)
```

**IntegrityError**

Exception raised when the relational integrity of the database is affected, e.g., a foreign key check fails.

```
exception IntegrityError(reason, sql_state='58005', vendor_code=-100)
```

**DataError**

Exception raised for errors due to problems with the processed data like division by zero, numeric value out of range, etc.

```
exception DataError(reason, sql_state='58005', vendor_code=-100)
```

**NotSupportedError**

Exception raised when a method or database API that is not supported by the database is used.

```
exception NotSupportedError(reason, sql_state='58005', vendor_code=-100)
```

## Optional DB API Extensions

### **Cursor.rownumber**

This read-only attribute should provide the current 0-based index of the cursor in the result set or **None** if the index cannot be determined.

You can see the index as the index of the cursor in a sequence (the result set). The next fetch operation fetches the row indexed by `.rownumber` in that sequence.

### **Connection.Error, Connection.ProgrammingError, etc.**

All exception classes defined by the DB API standard should be exposed on the `Connection` objects as attributes (in addition to being available at module scope).

These attributes simplify error handling in multi-connection environments.

### **Cursor.connection**

This read-only attribute returns a reference to the `Connection` object on which the cursor was created.

The attribute simplifies writing polymorph code in multi-connection environments.

### **Cursor.\_\_iter\_\_()**

Return self to make cursors compatible with the Python iteration protocol.

## Related Links

[pyocient PyPI](#)



---

## 12 Loading and Transformation

### 12.1 Ingest Data with Legacy LAT Reference

Data Pipelines are now the preferred method for loading data into the Ocident System. For details, see [Load Data](#).

#### Loading and Transformation (LAT) Overview

This section provides an overview of how loading works in Ocident® along with step-by-step examples to illustrate the key aspects of the loading process. A separate LAT Reference provides a detailed explanation of the options for each data format and data source in Ocident.

The key elements in the Loading system are:

- Data Source — an origin source for data such as Amazon® Web Services<sup>SM</sup> (AWS<sup>SM</sup>) S3, Apache® Kafka®, or Apache® Hadoop® Distributed File System (HDFS). Sources can be file or streaming in nature.
- Data Type Extraction — the format for data extraction from the source. Examples include JSON, CSV, fixed width.
- Transformations — the functions used to cleanse, route, and transform the incoming data.
- Indexer — operated by the **streamloader** role on a Loader Node, the Indexer collects transformed records, stores in replicated pages, and converts into segments on the Foundation Nodes.

Reference:

- [LAT Data Types in Loading](#)
- [LAT Pipeline Configuration](#)
- [LAT Client Command Line Interface](#)
- [LAT Metrics](#)
- [LAT Advanced Topics](#)
- [LAT Packaging and Installation](#)

#### Loading Overview

The LAT is a service within an Ocident System that is responsible for fetching data from file or streaming sources, extracting records, transforming them, and routing them to Ocident tables. The LAT runs as a standalone Java® process on Loader Nodes. It runs many parallel workers on high-core count servers to deliver the required parallelism for high-throughput loading. This diagram shows that a set of Loader Nodes sit between the data sources and the Foundation Nodes to control the loading flow and deliver high performance across many parallel workers.

The LAT loads data from two general types of data sources:

- Streaming Sources — continuous loading of data from an ordered stream such as Kafka
- File Sources — a discrete batch of data loaded from files until complete, typically from a file system or object storage solution like AWS S3

The main difference between these modes is whether the task continues indefinitely or ends when the batch of files has been loaded. In both cases, the LAT uses identical transformation and loading processes so that users can manage both with the same mental model and configuration files. Most data types are supported identically in streaming and batch processing modes, however some formats are not suitable for streaming due to their structure.

A complete list of data sources and supported data formats is in [LAT Data Types in Loading](#).

While loading, the LAT coordinates with the Ocident System to manage backpressure of loading new records. New rows accumulate into a storage structure called a “page” which is a column-oriented storage mechanism. These pages are replicated across Foundation Nodes so that the specified redundancy is maintained in the event of a system outage. When enough pages have accumulated in a given time period (as defined by the TimeKey® bucket), the Indexer will convert the pages into segment structures and place these on the Foundation Nodes for permanent storage as Segment Groups that span the width of the storage space.

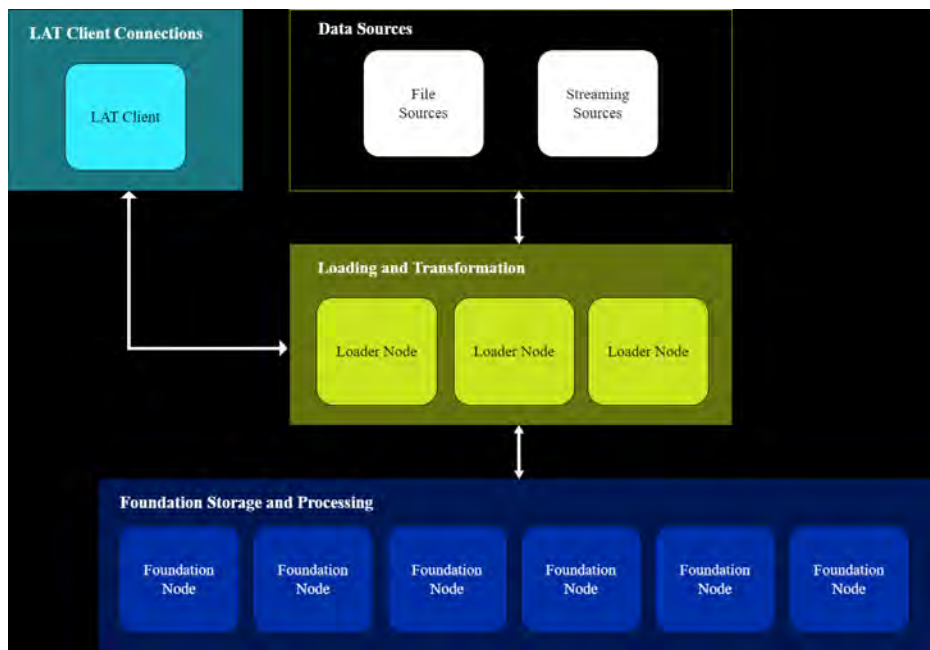


Figure 27: Loading and transformation into foundation storage

This process is transparent to analysts issuing queries. Rows are federated during queries so that pages and Segments are seamlessly presented in result sets.

### Pipelines Overview

The process and configuration that defines how data is loaded is referred to as a “pipeline.” A “pipeline” defines the end-to-end data flow in a loading task including the Source type (e.g., Kafka), the record Extraction for the data type (e.g., CSV extraction), the transformations on each record (e.g., JSON value extraction, data exploding, string concatenation, flattening), and where transformed data should be loaded in Ocint tables. As shown in this diagram, a user sets each of these sections in the pipeline file to control the LAT.

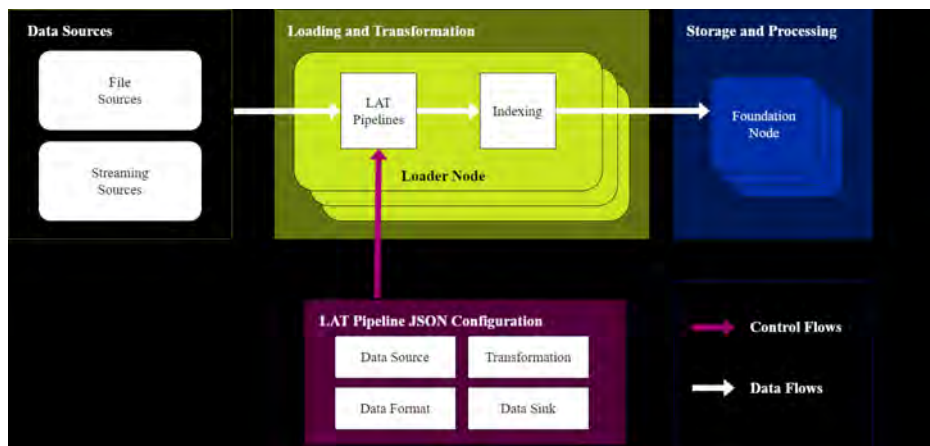


Figure 28: LAT pipeline JSON configuration with data flow from data sources to storage and processing

The LAT pipelines are managed through an HTTP API on the Loader Nodes or through a Command Line Interface (CLI) that provides a convenient way to work with the HTTP API. Because most Ocint Systems include many Loader Nodes, the CLI coordinates pipelines across the specified set of Loader Nodes.

The LAT Client issues the command to the Loader Node to create the new pipeline with the pipeline configuration

file. When started, the LAT uses the pipeline to execute a highly parallelized loading task across all Loader Nodes and Foundation Nodes to load the configured tables.

Detailed instructions for configuring a pipeline are found in [LAT Pipeline Configuration](#).

### Time Ordering

The Ocient Hyperscale Data Warehouse contains timeseries data. Loading in the data warehouse is considerably faster when data is presented in an ordered time sequence according to the TimeKey in a table. For most streaming sources, this is typically accomplished automatically with limited “out of order” data by the nature of the queue.

However, for file based sources, it is not uncommon for data to be collected in folders in a manner that could appear haphazard to the loader. For this reason, it is critical that the `sort_type` setting for file based sources be used correctly to inform the LAT how files should be ordered.

Read more about file sorting in [LAT Source Configuration](#).

### Exactly-Once Guarantees

Loading in Ocient is designed to ensure exactly once processing of data. The LAT operates independent streams of data that each has a monotonically increasing row ID tied to an individual row in a File-based load or a partition offset in a Kafka-based load. Exactly-once processing is made possible through coordination between the loading components around a “Durability Horizon” that represents the highest record durably stored on non-volatile storage in each independent stream. In the event of a node outage or a replay of the data, the Durability Horizon automatically removes duplicate records in an efficient manner.

It is important to understand how the LAT determines the unique row ID for different source types to ensure that data is loaded correctly. This is described in more detail in the LAT Reference Documentation.

Learn more about the underlying approach used in [Loading Characteristics and Concepts](#).

### Dynamic Schema Changes

Ocient also supports schema changes on tables while loading data in a continuously streaming pipeline. Ocient maintains a table version with each running pipeline that serves as a loading contract until the load task complete. This ensures that existing pipelines are not interrupted when columns are added or removed from a table that is receiving new data. Ocient continues loading records using the original table version even after a change has been made to the tables. These dynamic schema changes from `ALTER TABLE` commands allow flexible updates to tables while complex loading processes are active. Pipelines can then be updated to add or remove data elements and match the altered schema, reducing the burden of coordination across systems.

### Loading Examples

The following examples walk through a simple loading example using the LAT for a streaming load off of Kafka and a file load off of S3. These examples are very similar due to the way the LAT uses a common language for all transformation and loading regardless of data source or data type. The final set of examples outline some more complex transformations.

Examples:

- [LAT Load JSON Data from Kafka](#)
- [LAT Load CSV Data from S3](#)
- [LAT Load JSON Data from S3](#)
- [LAT Advanced Loading and Transformations](#)

For more detailed information about LAT settings, see the [LAT Pipeline Configuration](#).

### Related Links

[LAT Overview](#)

[LAT Data Types in Loading](#)

### 12.1.1 Loading by Data Source

#### 12.1.1.1 LAT Load JSON Data from Kafka

Data Pipelines are now the preferred method for loading data into the Ocient System. For details, see [Load Data](#).

A common setup for streaming data into Ocient® is to send JSON documents to Apache® Kafka® and then transform each document into rows in one or more different tables. Ocient's Loading and Transformation capabilities use a simple SQL-like syntax for transforming data. This tutorial will guide users through a simple example load using a small set of data in JSON format. The data in this example is created from a test set for the Metabase<sup>SM</sup> Business Intelligence tool.

Prerequisites

This tutorial assumes that:

1. A Kafka cluster is operational and can be reached by the Ocient Loader Nodes.
2. An Ocient System is installed and configured with an active storage cluster (See the [Ocient Application Configuration](#) guide).
3. The Ocient Loader Nodes are running the latest Loading and Transformation version which is configured to connect to Kafka for stream loading.
4. A default "sink" for the Ocient Loader Nodes is configured on the system.
5. The [LAT Client Command Line Interface](#) is installed.
6. The test data for this Tutorial can be found at the following S3 addresses.

You must be logged into Amazon AWS to download these files.

```
https://ocient-examples.s3.amazonaws.com/metabase_samples/jsonl/orders.jsonl
https://ocient-examples.s3.amazonaws.com/metabase_samples/jsonl/products.jsonl
https://ocient-examples.s3.amazonaws.com/metabase_samples/jsonl/people.jsonl
https://ocient-examples.s3.amazonaws.com/metabase_samples/jsonl/reviews.jsonl
```

#### Step 1: Create a New Database

To begin, you are going to load four example tables in a database. First, connect to a SQL Node using the [Commands Supported by the Ocient JDBC CLI Program](#). Then run the following DDL command:

```
CREATE DATABASE metabase;
```

#### Step 2: Create Tables

To create tables in the new database, first connect to that database (e.g., connect to `jdbc:ocient://sql-node:4050/metabase`), then run the following DDL commands:

```
CREATE TABLE public.orders (
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  user_id INT NOT NULL,
  product_id INT NOT NULL,
  subtotal DOUBLE,
  tax DOUBLE,
  total DOUBLE,
  discount DOUBLE,
  quantity INT,
  CLUSTERING INDEX idx01 (user_id, product_id)
);
```

```
CREATE TABLE public.people (
```

```

created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
id INT NOT NULL,
address VARCHAR(255),
email VARCHAR(255),
password VARCHAR(255),
name VARCHAR(255),
city VARCHAR(255),
longitude DOUBLE,
state VARCHAR(255),
source VARCHAR(255),
birth_date DATE,
zip VARCHAR(255),
latitude DOUBLE,
CLUSTERING INDEX idx01 (id)
);

CREATE TABLE public.products(
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  ean VARCHAR(255),
  title VARCHAR(255),
  category VARCHAR(255) COMPRESSION GDC(2) NOT NULL,
  vendor VARCHAR(255),
  price DOUBLE,
  rating DOUBLE,
  CLUSTERING INDEX idx01 (category)
);

CREATE TABLE public.reviews (
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  product_id INT NOT NULL,
  reviewer VARCHAR(255),
  rating INT,
  body VARCHAR(255),
  CLUSTERING INDEX idx01 (product_id)
);

```

Now, the database tables are created, and you can begin loading data.

### Step 3: Create a Data Pipeline

Data pipelines are created using a simple loading configuration that is submitted to the Transformation Nodes to start loading. Each Kafka topic is routed to one or more Ocint tables, and each column is the result of a transformation applied to the source document.

First, let's inspect the data that is going to be loaded. Each document has a format similar to the following example:

```

/* orders */
{"id": 1, "user_id": 1, "product_id": 14, "subtotal": 37.65, "tax": 2.07, "total": 39.72, "discount": null}
{"id": 2, "user_id": 1, "product_id": 123, "subtotal": 110.93, "tax": 6.1, "total": 117.03, "discount": null}
...

/* products */
{"id": 1, "ean": "1018947080336", "title": "Rustic Paper Wallet", "category": "Gizmo", "vendor": "Swaniawski"}
{"id": 2, "ean": "7663515285824", "title": "Small Marble Shoes", "category": "Doohickey", "vendor": "Balistreri"}
{"id": 3, "ean": "4966277046676", "title": "Synergistic Granite Chair", "category": "Doohickey", "vendor": "Balistreri"}

```

```

...

/* people */
{"id": 1, "address": "9611-9809 West Rosedale Road", "email": "borer-hudson@yahoo.com", "password": "ccca8"}
{"id": 2, "address": "101 4th Street", "email": "williamson-domenica@yahoo.com", "password": "eafc45bf-cf8"}
{"id": 3, "address": "29494 Anderson Drive", "email": "lina.heaney@yahoo.com", "password": "36f67891-34e5-"}
...

/* reviews */
{"id": 1, "product_id": 1, "reviewer": "christ", "rating": 5, "body": "Ad perspiciatis quis et consectetur"}
{"id": 2, "product_id": 1, "reviewer": "xavier", "rating": 4, "body": "Reprehenderit non error architecto"}
{"id": 3, "product_id": 1, "reviewer": "cameron.nitzsche", "rating": 5, "body": "In aut numquam labore fug"}
...

```

As you can see, this is similar to the target schema, but will require some transformation. Most transformations are identical to functions already in Ociant's SQL dialect. To route data to your tables, you must create a pipeline.json file that has the following structure:

```

{
  "version": 2,
  "workers": 4,
  "pipeline_id": "pipeline-metabase",
  "source": {
    "type": "kafka",
    "kafka": {
      "bootstrap.servers": "127.0.0.1:9092",
      "auto.offset.reset": "earliest"
    }
  },
  "transform": {
    "topics": {
      "orders": {
        "tables": {
          "metabase.public.orders": {
            "columns": {
              "id": "id",
              "user_id": "user_id",
              "product_id": "product_id",
              "subtotal": "subtotal",
              "tax": "tax",
              "total": "total",
              "discount": "discount",
              "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\'T\\'HH:mm:ss[.SSS]X')",
              "quantity": "quantity"
            }
          }
        }
      }
    },
    "people": {
      "tables": {
        "metabase.public.people": {
          "columns": {
            "id": "id",
            "address": "address",
            "email": "email",
            "password": "password",

```



```
export LAT_HOSTS="https://10.0.0.1:8443,https://10.0.0.2:8443"
```

If your LAT is running without TLS configured, replace the port number of your LAT Hosts with 8080 and the protocol with `http://`.

Next, check on the status of the LAT:

```
lat_client pipeline status
```

**Example response:**

```
10.0.0.1:8443: Stopped
10.0.0.2:8443: Stopped
```

Success! This confirms that you can reach the LAT from your CLI. If the status is “Running” it means a pipeline is already executing a pipeline. You are next going to update and start your new pipeline.

This example uses secure connections. If you receive an SSL Error when testing, your service cannot be configured to use TLS or you might need to use the `--no-verify` flag if certificate validation fails.

**Step 5: Test the Transformation**

The CLI supports previewing a transformation with an example document and the pipeline file. This makes it easy to test your transformations.

First, save an example document to your file system to use for this test. For this demo, you can download an example file from [https://ocient-examples.s3.amazonaws.com/metabase\\_samples/jsonl/orders.jsonl](https://ocient-examples.s3.amazonaws.com/metabase_samples/jsonl/orders.jsonl) and save it to `~/orders.jsonl`.

Next, make sure the pipeline.json file that you created is stored at `~/pipeline.json`.

Now that both files are available, run the CLI to preview the results. You can pass the preview command the topic name, the pipeline file, and the sample record file. The response contains the transformed data tied to the destination table and a list of any error records.

```
lat_client preview --topic orders --pipeline ~/pipeline.json --records ~/orders.jsonl
```

**Example response:**

```
{
  "tableRecords": {
    "metabase.public.orders": [
      {
        "id": 1,
        "user_id": 1,
        "product_id": 14,
        "subtotal": 37.65,
        "tax": 2.07,
        "total": 39.72,
        "discount": null,
        "created_at": 1549921227892000000,
        "quantity": 2
      },
      {
        "id": 2,
        "user_id": 1,
        "product_id": 123,
        "subtotal": 110.93,
        "tax": 6.1,
        "total": 117.03,
```



```

        "discount": null,
        "created_at": 1526371444580000000,
        "quantity": 3
      },
      {
        "id": 3,
        "user_id": 1,
        "product_id": 105,
        "subtotal": 52.72,
        "tax": 2.9,
        "total": 49.2,
        "discount": 6.42,
        "created_at": 1575670968544000000,
        "quantity": 2
      }
    ],
    "recordErrors": []
  }
}

```

You can see that the data is transformed and the columns to which each transformed value will be mapped. If there are issues in the values, these will appear in the **recordErrors** object. You can quickly update your `pipeline.json` file and preview again. Now, you can inspect different documents to confirm that various states of data cleanliness like missing columns, null values, and special characters are well handled by your transformations.

#### Step 6: Configure and Start the Data Pipeline

With a tested transformation, the next step is to set up and start the data pipeline.

First, configure the pipeline using the `pipeline create` command. This validates and creates the pipeline, but will not take effect until you start the pipeline:

```
lat_client pipeline create --pipeline ~/pipeline.json
```

Example response:

```
10.0.0.1:8443: Created
10.0.0.2:8443: Created
```

In cases where there is an existing pipeline operating, it is necessary to stop the pipeline and remove the original pipeline before creating and starting the new pipeline.

Now that the pipeline has been created on all LAT Nodes, you can start the LAT by running the `pipeline start` commands:

```
lat_client pipeline start
```

Example responses:

```
10.0.0.1:8443: Running
10.0.0.2:8443: Running
```

#### Step 7: Confirm that Loading is Operating Correctly

With your pipeline in place and running, data will immediately begin loading off of the Kafka topics that are configured in the pipeline. If you do not have data in the Kafka topics yet, now would be a good time to start producing data into the topics.

Producing Test Data into Kafka:

For test purposes, [kafkacat](#) is a helpful utility that makes it easy to product records into a topic. For example, if you have a file of sample data `orders.jsonl` in a JSONL format (newline delimited JSON records), you can run the following command to send those records into your Kafka broker:

```
kafkacat -b <broker_ip_address>:9092 -t <topic_name> -T -P -l orders.jsonl
```

Assuming your broker is running at 10.0.0.3 and you want to send data into the four topics defined in your `pipeline.json` definition, you can run:

```
kafkacat -b 10.0.0.3:9092 -t orders -T -P -l orders.jsonl
kafkacat -b 10.0.0.3:9092 -t products -T -P -l products.jsonl
kafkacat -b 10.0.0.3:9092 -t people -T -P -l people.jsonl
kafkacat -b 10.0.0.3:9092 -t reviews -T -P -l reviews.jsonl
```

Each of these commands will push the entire JSONL file of messages into Kafka with one record per line. As these are produced into Kafka, your running pipeline will begin loading them into Ocident.

Observing Loading Progress:

With data in Kafka, our pipeline will begin loading data immediately and streaming any new data into Ocident. To observe this progress, you can monitor the metrics endpoint of the Loading and Transformation Nodes. This can be done manually from a command line or from a tool like Grafana®. For this example, you can run a `curl` command against the endpoint and review the result. For details on metrics, see the [LAT Metrics](#) Documentation.

#### Command:

```
curl https://127.0.0.1:8443/v2/metrics/lat:type=pipeline
```

If your LAT is running without TLS configured, replace the port number of your LAT Hosts with 8080 and the protocol with `http://`.

#### Example response:

```
{
  "request": {
    "mbean": "lat:type=pipeline",
    "type": "read"
  },
  "value": {
    "partitions": [
      {
        "offsets_durable": 1,
        "pushes_errors": 0,
        "pushes_attempts": 1,
        "rows_pushed": 1,
        "offsets_written": 18759,
        "records_buffered": 0,
        "records_errors_column": 0,
        "records_errors_deserialization": 0,
        "source_bytes_buffered": 0,
        "records_errors_transformation": 0,
        "offsets_processed": 18759,
        "partition": "table_orders-0",
        "records_filter_accepted": 1,
        "records_errors_row": 0,
        "records_filter_rejected": 0,
        "records_errors_generic": 0,
        "producer_send_attempts": 0,
        "offsets_pushed": 18759,
```

```

    "pushes_unacknowledged": 0,
    "invalid_state": 0,
    "bytes_pushed": 88,
    "records_errors_total": 0,
    "offsets_buffered": 18759,
    "complete": 0,
    "offsets_end": 1,
    "producer_send_errors": 0
  },
  {
    "offsets_durable": 1,
    "pushes_errors": 0,
    "pushes_attempts": 1,
    "rows_pushed": 1,
    "offsets_written": 2499,
    "records_buffered": 0,
    "records_errors_column": 0,
    "records_errors_deserialization": 0,
    "source_bytes_buffered": 0,
    "records_errors_transformation": 0,
    "offsets_processed": 2499,
    "partition": "table_people-0",
    "records_filter_accepted": 1,
    "records_errors_row": 0,
    "records_filter_rejected": 0,
    "records_errors_generic": 0,
    "producer_send_attempts": 0,
    "offsets_pushed": 2499,
    "pushes_unacknowledged": 0,
    "invalid_state": 0,
    "bytes_pushed": 223,
    "records_errors_total": 0,
    "offsets_buffered": 2499,
    "complete": 0,
    "offsets_end": 1,
    "producer_send_errors": 0
  },
  {
    "offsets_durable": 1,
    "pushes_errors": 0,
    "pushes_attempts": 1,
    "rows_pushed": 1,
    "offsets_written": 199,
    "records_buffered": 0,
    "records_errors_column": 0,
    "records_errors_deserialization": 0,
    "source_bytes_buffered": 0,
    "records_errors_transformation": 0,
    "offsets_processed": 199,
    "partition": "table_products-0",
    "records_filter_accepted": 1,
    "records_errors_row": 0,
    "records_filter_rejected": 0,
    "records_errors_generic": 0,
    "producer_send_attempts": 0,

```

```

    "offsets_pushed": 199,
    "pushes_unacknowledged": 0,
    "invalid_state": 0,
    "bytes_pushed": 145,
    "records_errors_total": 0,
    "offsets_buffered": 199,
    "complete": 0,
    "offsets_end": 1,
    "producer_send_errors": 0
  },
  {
    "offsets_durable": 1,
    "pushes_errors": 0,
    "pushes_attempts": 1,
    "rows_pushed": 1,
    "offsets_written": 1111,
    "records_buffered": 0,
    "records_errors_column": 0,
    "records_errors_deserialization": 0,
    "source_bytes_buffered": 0,
    "records_errors_transformation": 0,
    "offsets_processed": 1111,
    "partition": "table_reviews-0",
    "records_filter_accepted": 1,
    "records_errors_row": 0,
    "records_filter_rejected": 0,
    "records_errors_generic": 0,
    "producer_send_attempts": 0,
    "offsets_pushed": 1111,
    "pushes_unacknowledged": 0,
    "invalid_state": 0,
    "bytes_pushed": 184,
    "records_errors_total": 0,
    "offsets_buffered": 1111,
    "complete": 0,
    "offsets_end": 1,
    "producer_send_errors": 0
  }
],
"paused": 1,
"bytes_buffered": 0,
"workers": 20
},
"timestamp": 1626970368,
"status": 200
}

```

Check Row Counts in Tables:

To confirm that you are seeing results in the target tables, you can also run some simple queries to check row counts. Depending on the streamloader role settings, the time for records to become queryable can vary from a few seconds to minutes:

#### Example Queries:

```

Ociant> SELECT count(*) FROM public.orders;
count(*)

```

```
-----
18760
```

```
Ocient> SELECT count() FROM public.people;
count()
```

```
-----
2500
```

```
Ocient> SELECT count(*) FROM public.products;
count(*)
```

```
-----
200
```

```
Ocient> SELECT count(*) FROM public.reviews;
count(*)
```

```
-----
1112
```

Success! Now you can explore the data in these four tables with any Ocient SQL queries. If more data is pushed into these topics, your pipeline is still running and will automatically load all new data.

#### Check Errors

In this example, all rows load successfully. However, a successful load does not always happen, and you can inspect errors using the LAT Client. Whenever the LAT process fails to parse a file correctly or fails to transform or load a record, the LAT process records an error. The LAT Client includes the `lat_client pipeline errors` command that reports the latest errors on the pipeline.

A full error log is available on the Loader Nodes. These logs report all bad records and the reason that the load fails.

When you load a pipeline from Kafka, the load might route errors to an error topic on the Kafka broker instead of the logs. The LAT Client does not contain the errors sent to the error topic. You can inspect these errors with Kafka utilities instead.

This LAT Client command displays a maximum of 100 error messages.

```
lat_client pipeline errors --max-errors 100 --only-error-messages
```

```
|-----|
| exception_message |
|-----|
| Column name: time1. Message: Failed to evaluate |
| expression. Cause: |
| java.time.format.DateTimeParseException |
|-----|
| Column name: time1. Message: Failed to evaluate |
| expression. Cause: |
| java.time.format.DateTimeParseException |
|-----|
```

The errors indicate that there is an issue parsing the `time1` column. Options exist on the `pipeline errors` command to return JSON and to restrict the response to specific components of the error detail that includes a reference to the source location of this record.

The following command returns JSON that is delimited with newline characters. You can pass the JSON output to `jq` or a file. The JSON includes the source topic or file group, the filename where the error occurred, the offset that indicates the line number or Kafka offset, and the exception message that aids in troubleshooting and identifying the incorrect record in the source data. You can use the `log_original_message` pipeline setting to provide direct access to the parsed source record for errors when appropriate.

```
lat_client pipeline errors --max-errors 100 --json
```

```
{
  "time": "2022-05-17T16:53:50.387386+00:00",
  "topic": "calcs",
  "partition": 0,
  "state": "TRANSFORMATION_ER
  "time": "2022-05-17T16:53:50.404684+00:00",
  "topic": "calcs",
  "partition": 0,
  "state": "TRANSFORMATION_ER
```

Related Links

[LAT Overview](#)

[LAT Data Types in Loading](#)

[LAT Advanced Topics](#)

### 12.1.1.2 LAT Load CSV Data from S3

Data Pipelines are now the preferred method for loading data into the Ocient System. For details, see [Load Data](#).

A common setup for batch loading files into Ocient® is to load from a bucket on Amazon® Web Services<sup>SM</sup> (AWS<sup>SM</sup>) S3 with time partitioned data. In many instances, a batch load is performed on a recurring basis to load new files. The LAT transforms each document into rows in one or more different tables. Ocient's Loading and Transformation capabilities use a simple SQL-like syntax for transforming data. This tutorial will guide users through a simple example load using a small set of data in CSV format. The data in this example is created from a test set for the Metabase<sup>SM</sup> Business Intelligence tool.

Prerequisites

This tutorial assumes that:

1. The Ocient System has network access to S3 from the Loader Nodes.
2. An Ocient System is installed and configured with an active storage cluster (See the [Ocient Application Configuration](#) guide).
3. Loading and Transformation is installed on the Loader Nodes.
4. A default “sink” for the Ocient Loader Nodes is configured on the system.
5. The [LAT Client Command Line Interface](#) is installed.

Step 1: Create a New Database

To begin, load two example tables in a database. First, connect to a SQL Node using the [Commands Supported by the Ocient JDBC CLI Program](#). Then run the following DDL command:

```
CREATE DATABASE metabase;
```

Step 2: Create Tables

To create tables in the new database, first connect to that database (e.g., `connect to jdbc:ocient://sql-node:4050/metabase`), then run the following DDL commands:

```
CREATE TABLE public.orders (
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  user_id INT NOT NULL,
  product_id INT NOT NULL,
  subtotal DOUBLE,
  tax DOUBLE,
  total DOUBLE,
  discount DOUBLE,
  quantity INT,
  CLUSTERING INDEX idx01 (user_id, product_id)
);
```

```
CREATE TABLE public.products(
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  ean VARCHAR(255),
  title VARCHAR(255),
  category VARCHAR(255) COMPRESSION GDC(2) NOT NULL,
  vendor VARCHAR(255),
  price DOUBLE,
  rating DOUBLE,
  CLUSTERING INDEX idx01 (category)
);
```

Now, the database tables are created and you can begin loading data.

### Step 3: Create a Data Pipeline

Data pipelines are created using a simple loading configuration that is submitted to the Transformation Nodes to start loading. File Groups designate a batch of files to load. Each File Group is routed to one or more Ocident tables, and each column is the result of a transformation applied to the source document.

First, inspect the data that you plan to load. Each document has a format similar to the following example:

```
/* orders */
id,user_id,product_id,subtotal,tax,total,discount,created_at,quantity
1,1,14,37.65,2.07,39.72,null,2019-02-11T21:40:27.892Z,2
2,1,123,110.93,6.1,117.03,null,2018-05-15T08:04:04.580Z,3
3,1,105,52.72,2.9,49.2,6.42,2019-12-06T22:22:48.544Z,2
...

/* products */
id,ean,title,category,vendor,price,rating,created_at
1,1018947080336,Rustic Paper Wallet,Gizmo,"Swaniawski, Casper and Hilll",29.46,4.6,2017-07-19T19:44:56.582Z
2,7663515285824,Small Marble Shoes,Dooickey,Balistreri-Ankunding,70.08,0,2019-04-11T08:49:35.932Z
3,4966277046676,Synergistic Granite Chair,Dooickey,"Murray, Watsica and Wunsch",35.39,4,2018-09-08T22:03:12.345Z
...
```

This is similar to the target schema created in Step 2, but it will require some transformation. Most transformations are identical to functions already in Ocident's SQL dialect. To route data to the tables, you need to create a pipeline.json file that has the following structure:

```
{
  "version": 2,
  "workers": 4,
  "source": {
    "compression": "none",
    "type": "s3",
    "endpoint": "https://s3.us-east-1.amazonaws.com",
    "bucket": "ocient-examples",
    "file_groups": {
      "orders": {
        "prefix": "metabase_samples/csv",
        "file_matcher_syntax": "glob",
        "file_matcher_pattern": "**orders*.csv",
        "sort_type": "lexicographic"
      },
      "products": {
        "prefix": "metabase_samples/csv",
```

```

        "file_matcher_syntax": "glob",
        "file_matcher_pattern": "**products*.csv",
        "sort_type": "lexicographic"
    }
}
},
"extract": {
    "orders": {
        "record_type": "delimited",
        "headers": [
            "id",
            "user_id",
            "product_id",
            "subtotal",
            "tax",
            "total",
            "discount",
            "created_at",
            "quantity"
        ],
        "null_strings": [
            "NULL",
            "N/A"
        ],
        "record_delimiter": "\r\n",
        "num_header_lines": 1
    },
    "products": {
        "record_type": "delimited",
        "headers": [
            "id",
            "ean",
            "title",
            "category",
            "vendor",
            "price",
            "rating",
            "created_at"
        ],
        "record_delimiter": "\r\n",
        "num_header_lines": 1
    }
},
"transform": {
    "file_groups": {
        "orders": {
            "tables": {
                "metabase.public.orders": {
                    "columns": {
                        "id": "id",
                        "user_id": "user_id",
                        "product_id": "product_id",
                        "subtotal": "subtotal",
                        "tax": "tax",
                        "total": "total",

```



```

        "discount": "discount",
        "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\'T\\'HH:mm:ss[.SSS]X')",
        "quantity": "quantity"
      }
    }
  },
  "products": {
    "tables": {
      "metabase.public.products": {
        "columns": {
          "id": "id",
          "ean": "ean",
          "title": "title",
          "category": "category",
          "vendor": "vendor",
          "price": "price",
          "rating": "rating",
          "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\'T\\'HH:mm:ss[.SSS]X')",
        }
      }
    }
  }
}

```

The two interesting parts of this pipeline.json file are the way the file groups and the extraction settings are defined.

First, note that each file group sets the S3 endpoint, a bucket, a prefix used for filtering the considered files, and then a file matcher. In this example, there is only a single file, but if there were many files matching the pattern **\*\*orders\*.csv** then they would all be part of the file group. You do not need S3 credentials because this is a public bucket, but if it were private, you can supply credentials in a few different ways.

Next, note the extract section. Here, the example specifies a delimited format. If it were compressed (e.g., gzip), you could specify compression. The example also specifies the headers to associate with each column in the CSV data. Delimited extracts can specify different record delimiters (e.g., \n), specify field delimiters (e.g., |, \t, ,), define how to handle empty fields or to trim whitespace, and specify strings that should be considered NULL. While not used in this file, an example of null strings is provided that would turn the string literal “NULL” or “N/A” into a database NULL.

The final parameter supplied in the example is the sort type for the file load. This informs the LAT how you would like data to be ordered when loading. The ideal sort organizes files in time order according to the defined TimeKey@. This makes more efficient segments and is much faster to load. This example uses the lexicographic sort which orders according to the characters in the file name. Other sort types are available to use file modified time or to extract the timestamp for sorting from the file path or file name.

#### Step 4: Using the Loading and Transformation CLI

With a pipeline.json file ready to go, you can test this pipeline. To test, use the LAT CLI. For these examples, assume that two LATs are configured and set using an environment variable.

First, configure the LAT CLI to use the hosts of the Ocient Loading and Transformation service. You can add these to every CLI command as a flag, but for simplicity you can also set them as environment variables. From a command line, run the following command replacing the IP addresses with the IP addresses of your LAT processes:

```
export LAT_HOSTS="https://10.0.0.1:8443,https://10.0.0.2:8443"
```

If your LAT is running without TLS configured, replace the port number of your LAT Hosts with 8080 and the protocol with `http://`.

Next, check on the status of the LAT:

```
lat_client pipeline status
```

**Example response:**

```
10.0.0.1:8443: Stopped
10.0.0.2:8443: Stopped
```

Success! This confirms that you can reach the LAT from your CLI. If the status is “Running” it means a pipeline is already executing a pipeline. In the next step, you will update and start the new pipeline.

This example uses secure connections. If you receive an SSL Error when testing, your service cannot be configured to use TLS or you might need to use the `--no-verify` flag if certificate validation fails.

**Step 5: Test the Transformation**

The CLI supports previewing a transformation with an example document and the pipeline file. This makes it easy to test your transformations.

First, save an example document to your file system to use for this test. For this demo, you can download an example file from [https://ocient-examples.s3.amazonaws.com/metabase\\_samples/csv/orders.csv](https://ocient-examples.s3.amazonaws.com/metabase_samples/csv/orders.csv) and save it to `~/orders.csv`.

Next, make sure the `pipeline.json` file that you created is stored at `~/pipeline.json`.

Now that both files are available, you can run the CLI to preview the results. Pass the preview command the topic name, the pipeline file, and the sample record file. The response contains the transformed data tied to the destination table and a list of any error records.

Similar to how you can preview records on a Apache® Kafka® topic for file loads, you can supply any one `file_groups` created in the extract section to preview the transformations.

```
lat_client preview --topic orders --pipeline ~/pipeline.json --records ~/orders.csv
```

**Example response:**

```
{
  "tableRecords": {
    "metabase.public.orders": [
      {
        "id": 1,
        "user_id": 1,
        "product_id": 14,
        "subtotal": 37.65,
        "tax": 2.07,
        "total": 39.72,
        "discount": null,
        "created_at": 1549921227892000000,
        "quantity": 2
      },
      {
        "id": 2,
        "user_id": 1,
        "product_id": 123,
        "subtotal": 110.93,
        "tax": 6.1,
```

```

        "total": 117.03,
        "discount": null,
        "created_at": 1526371444580000000,
        "quantity": 3
      },
      {
        "id": 3,
        "user_id": 1,
        "product_id": 105,
        "subtotal": 52.72,
        "tax": 2.9,
        "total": 49.2,
        "discount": 6.42,
        "created_at": 1575670968544000000,
        "quantity": 2
      }
    ],
    "recordErrors": []
  }
}

```

You can see that the data is transformed and the columns to which each transformed value will be mapped. If there are issues in the values, these will appear in the `recordErrors` object. You can quickly update the `pipeline.json` file and preview again. Now, you can inspect different documents to confirm that various states of data cleanliness like missing columns, null values, and special characters are well handled by your transformations.

#### Step 6: Configure and Start the Data Pipeline

With a tested transformation, the next step is to setup and start the data pipeline.

First, configure the pipeline using the `pipeline create` command. This validates and creates the pipeline, but will not take effect until you start the pipeline:

```
lat_client pipeline create --pipeline ~/pipeline.json
```

#### Example response:

```
10.0.0.1:8443: Created
10.0.0.2:8443: Created
```

In cases where there is an existing pipeline operating, it is necessary to stop the pipeline and remove the original pipeline before creating and starting the new pipeline.

Now that the pipeline has been created on all LAT Nodes, you can start the LAT by running the `pipeline start` commands:

```
lat_client pipeline start
```

#### Example responses:

```
10.0.0.1:8443: Running
10.0.0.2:8443: Running
```

#### Step 7: Confirm that Loading is Operating Correctly

With your pipeline in place and running, data will immediately begin loading from the S3 file groups that you defined. If there were many files per file group, the LAT would first sort the files, then partition them for the fastest loading based on the sorting criteria you provided.

#### Observing Loading Progress

With the pipeline running, data immediately begins to load into Ocint. To observe this progress, you can use the `pipeline status` command from the LAT Client or monitor the LAT metrics endpoint of the Loader Nodes.

You can check the status with this command by using the `--list-files` flag to include a summary of the files included in the load.

```
lat_client pipeline status --list-files
```

#### Example responses:

```
10.0.0.1:8443: Running
10.0.0.2:8443: Running
```

```
Pipeline Files Processed: 0
Pipeline Error Count: 0
Pipeline Files Remaining: 2
```

#### orders

```
Files Processed: 0
Error Count: 0
Files Remaining: 1
```

#### products

```
Files Processed: 0
Error Count: 0
Files Remaining: 1
```

#### orders

```
Status      Filename
processing  metabase_samples/csv/orders.csv
```

#### In Process Files:

```
processing  metabase_samples/csv/orders.csv
```

#### products

```
Status      Filename
processing  metabase_samples/csv/products.csv
```

#### In Process Files:

```
processing  metabase_samples/csv/products.csv
```

You can monitor the LAT metrics endpoint manually at the command line. Or, you can use a tool like Grafana® to retrieve metrics. For this example, run the `curl` command against the endpoint and review the result. For details on metrics, see the [LAT Metrics](#) Documentation.

#### Command:

```
curl https://127.0.0.1:8443/v2/metrics/lat?type=pipeline
```

If your LAT is running without TLS configured, replace the port number of your LAT Hosts with 8080 and the protocol with `http://`.

#### Example response:

```

{
  "request": {
    "mbean": "lat:type=pipeline",
    "type": "read"
  },
  "value": {
    "partitions": [
      {
        "offsets_durable": 1,
        "pushes_errors": 0,
        "pushes_attempts": 1,
        "rows_pushed": 1,
        "offsets_written": 18759,
        "records_buffered": 0,
        "records_errors_column": 0,
        "records_errors_deserialization": 0,
        "source_bytes_buffered": 0,
        "records_errors_transformation": 0,
        "offsets_processed": 18759,
        "partition": "table_orders-0",
        "records_filter_accepted": 1,
        "records_errors_row": 0,
        "records_filter_rejected": 0,
        "records_errors_generic": 0,
        "producer_send_attempts": 0,
        "offsets_pushed": 18759,
        "pushes_unacknowledged": 0,
        "invalid_state": 0,
        "bytes_pushed": 88,
        "records_errors_total": 0,
        "offsets_buffered": 18759,
        "complete": 0,
        "offsets_end": 1,
        "producer_send_errors": 0
      },
      {
        "offsets_durable": 1,
        "pushes_errors": 0,
        "pushes_attempts": 1,
        "rows_pushed": 1,
        "offsets_written": 199,
        "records_buffered": 0,
        "records_errors_column": 0,
        "records_errors_deserialization": 0,
        "source_bytes_buffered": 0,
        "records_errors_transformation": 0,
        "offsets_processed": 199,
        "partition": "table_products-0",
        "records_filter_accepted": 1,
        "records_errors_row": 0,
        "records_filter_rejected": 0,
        "records_errors_generic": 0,
        "producer_send_attempts": 0,
        "offsets_pushed": 199,
        "pushes_unacknowledged": 0,

```

```

    "invalid_state": 0,
    "bytes_pushed": 145,
    "records_errors_total": 0,
    "offsets_buffered": 199,
    "complete": 0,
    "offsets_end": 1,
    "producer_send_errors": 0
  }
],
"paused": 1,
"bytes_buffered": 0,
"workers": 20
},
"timestamp": 1626970368,
"status": 200,
}

```

### Check Row Counts in Tables

To confirm that you are seeing results in the target tables, you can also run some simple queries to check row counts. Depending on the streamloader role settings, the time for records to become queryable can vary from a few seconds to minutes:

#### Example Queries:

```
Ocient> SELECT count(*) FROM public.orders;
COUNT(*)
```

```
-----
18760
```

```
Ocient> SELECT count() FROM public.products;
COUNT()
```

```
-----
200
```

Now you can explore the data in these four tables with any Ocient SQL queries.

#### Related Links

[LAT Overview](#)

[LAT Data Types in Loading](#)

[LAT Advanced Topics](#)

### 12.1.1.3 LAT Load JSON Data from S3

Data Pipelines are now the preferred method for loading data into the Ocient System. For details, see [Load Data](#).

A common setup for batch loading files into Ocient® is to load from a bucket on Amazon® Web Services<sup>SM</sup> (AWS<sup>SM</sup>) S3 with time partitioned data. In many instances, a batch load is performed on a recurring basis to load new files. The LAT transforms each document into rows in one or more different tables. Ocient's Loading and Transformation capabilities use a simple SQL-like syntax for transforming data. This tutorial will guide users through a simple example load using a small set of data in JSONL (newline delimited JSON) format. The data in this example is created from a test set for the Metabase<sup>SM</sup> Business Intelligence tool.

#### Prerequisites

This tutorial assumes that:

1. The Ocient System has network access to S3 from the Loader Nodes.
2. An Ocient System is installed and configured with an active storage cluster (See the [Ocient Application Configuration](#) guide).
3. Loading and Transformation is installed on the Loader Nodes.
4. A default “sink” for the Ocient Loader Nodes is configured on the system.
5. The [LAT Client Command Line Interface](#) is installed.

#### Step 1: Create a New Database

To begin, you are going to load two example tables in a database. First, connect to a SQL Node using the [Commands Supported by the Ocient JDBC CLI Program](#). Then run the following DDL command:

```
CREATE DATABASE metabase;
```

#### Step 2: Create Tables

To create tables in the new database, first connect to that database (e.g., `connect to jdbc:ociant://sql-node:4050/metabase`), then run the following DDL commands:

```
CREATE TABLE public.orders (
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  user_id INT NOT NULL,
  product_id INT NOT NULL,
  subtotal DOUBLE,
  tax DOUBLE,
  total DOUBLE,
  discount DOUBLE,
  quantity INT,
  CLUSTERING INDEX idx01 (user_id, product_id)
);
```

```
CREATE TABLE public.products(
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  ean VARCHAR(255),
  title VARCHAR(255),
  category VARCHAR(255) COMPRESSION GDC(2) NOT NULL,
  vendor VARCHAR(255),
  price DOUBLE,
  rating DOUBLE,
  CLUSTERING INDEX idx01 (category)
);
```

Now, the database tables are created and you can begin loading data.

#### Step 3: Create a Data Pipeline

Data pipelines are created using a simple loading configuration that is submitted to the Transformation Nodes to start loading. File Groups designate a batch of files to load. Each File Group is routed to one or more Ocient tables, and each column is the result of a transformation applied to the source document.

First, let’s inspect the data you plan to load. Each document has a format similar to the following example:

```
/* orders */
{"id": 1, "user_id": 1, "product_id": 14, "subtotal": 37.65, "tax": 2.07, "total": 39.72, "discount": null}
{"id": 2, "user_id": 1, "product_id": 123, "subtotal": 110.93, "tax": 6.1, "total": 117.03, "discount": null}
...

/* products */
```

```
{
  "id": 1, "ean": "1018947080336", "title": "Rustic Paper Wallet", "category": "Gizmo", "vendor": "Swaniaws",
  "id": 2, "ean": "7663515285824", "title": "Small Marble Shoes", "category": "Doohickey", "vendor": "Balis",
  "id": 3, "ean": "4966277046676", "title": "Synergistic Granite Chair", "category": "Doohickey", "vendor": "Balis",
  ...
}
```

As you can see, this is similar to your target schema, but will require some transformation. Most transformations are identical to functions already in Ociant's SQL dialect. To route data to your tables, you need to create a pipeline.json file that has the following structure:

```
{
  "version": 2,
  "workers": 4,
  "source": {
    "type": "s3",
    "endpoint": "https://s3.us-east-1.amazonaws.com",
    "bucket": "ocient-examples",
    "compression": "none",
    "file_groups": {
      "orders": {
        "prefix": "metabase_samples/jsonl",
        "file_matcher_syntax": "glob",
        "file_matcher_pattern": "**orders*.jsonl",
        "sort_type": "lexicographic"
      },
      "products": {
        "prefix": "metabase_samples/jsonl",
        "file_matcher_syntax": "glob",
        "file_matcher_pattern": "**products*.jsonl",
        "sort_type": "lexicographic"
      }
    }
  },
  "transform": {
    "file_groups": {
      "orders": {
        "tables": {
          "metabase.public.orders": {
            "columns": {
              "id": "id",
              "user_id": "user_id",
              "product_id": "product_id",
              "subtotal": "subtotal",
              "tax": "tax",
              "total": "total",
              "discount": "discount",
              "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\'T\\'HH:mm:ss[.SSS]X')",
              "quantity": "quantity"
            }
          }
        }
      },
      "products": {
        "tables": {
          "metabase.public.products": {
            "columns": {
              "id": "id",

```



```

        "ean": "ean",
        "title": "title",
        "category": "category",
        "vendor": "vendor",
        "price": "price",
        "rating": "rating",
        "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\'T\\'HH:mm:ss[.SSS]X')"
```

The most interesting part of this pipeline.json file is the way it defines the file groups. Note that each sets the S3 endpoint, a bucket, a prefix used for filtering the considered files, and then a file matcher. In this case you only have a single file, but if there were many files matching the pattern **\*\*orders\*.jsonl** then they would all be part of the file group.

The final parameter that you supplied is the sort type for the file load. This informs the LAT how you would like data to be ordered when loading. The ideal sort is in time order according to the defined TimeKey®. This makes more efficient segments and is much faster to load. In this case, you used the lexicographic sort which orders according to the characters in the file name. Other sort types are available to use file modified time or to extract the timestamp for sorting from the file path or file name.

#### Step 4: Using the Loading and Transformation CLI

With a pipeline.json file ready to go, you can test this pipeline. To test, use the LAT CLI. For these examples, assume that two LATs are configured and set using an environment variable.

First configure the LAT CLI to use the hosts of the Ocient Loading and Transformation service. You can add these to every CLI command as a flag, but for simplicity you can also set them as environment variables. From a command line, run the following command replacing the IP addresses with the IP addresses of your LAT processes:

```
export LAT_HOSTS="https://10.0.0.1:8443,https://10.0.0.2:8443"
```

If your LAT is running without TLS configured, replace the port number of your LAT Hosts with 8080 and the protocol with **http://**.

Next, check on the status of the LAT:

```
lat_client pipeline status
```

#### Example response:

```
10.0.0.1:8443: Stopped
10.0.0.2:8443: Stopped
```

This confirms that you can reach the LAT from your CLI. If the status is “Running” it means a pipeline is already executing a pipeline. Next, you are going to update and start your new pipeline.

This example uses secure connections. If you receive an SSL Error when testing, your service might not be configured to use TLS or you might need to use the **--no-verify** flag if the certificate validation fails.

#### Step 5: Test the Transformation

The CLI supports previewing a transformation with an example document and the pipeline file. This makes it easy to test your transformations.

First, save an example document to your file system to use for this test. For this demo, you can download an example file from [https://ocient-examples.s3.amazonaws.com/metabase\\_samples/jsonl/orders.jsonl](https://ocient-examples.s3.amazonaws.com/metabase_samples/jsonl/orders.jsonl) and save it to `~/orders.jsonl`.

Next, make sure the `pipeline.json` file that you created is stored at `~/pipeline.json`.

Now that both files are available, you can run the CLI to preview the results. Pass the preview command the topic name, the pipeline file, and the sample record file. The response contains the transformed data tied to the destination table and a list of any error records.

Similar to how you can preview records on a Apache® Kafka® topic for file loads, you can supply any one of the topics you created as file groups to preview the transformations.

```
lat_client preview --topic orders --pipeline ~/pipeline.json --records ~/orders.jsonl
```

#### Example response:

```
{
  "tableRecords": {
    "metabase.public.orders": [
      {
        "id": 1,
        "user_id": 1,
        "product_id": 14,
        "subtotal": 37.65,
        "tax": 2.07,
        "total": 39.72,
        "discount": null,
        "created_at": 1549921227892000000,
        "quantity": 2
      },
      {
        "id": 2,
        "user_id": 1,
        "product_id": 123,
        "subtotal": 110.93,
        "tax": 6.1,
        "total": 117.03,
        "discount": null,
        "created_at": 1526371444580000000,
        "quantity": 3
      },
      {
        "id": 3,
        "user_id": 1,
        "product_id": 105,
        "subtotal": 52.72,
        "tax": 2.9,
        "total": 49.2,
        "discount": 6.42,
        "created_at": 1575670968544000000,
        "quantity": 2
      }
    ]
  },
  "recordErrors": []
}
```

You can see that the data is transformed and the columns to which each transformed value will be mapped. If there are issues in the values, these will appear in the **recordErrors** object. You can quickly update your pipeline.json file and preview again. Now, you can inspect different documents to confirm that various states of data cleanliness like missing columns, null values, and special characters are well handled by your transformations.

#### Step 6: Configure and Start the Data Pipeline

With a tested transformation, the next step is to setup and start the data pipeline.

First, you must configure the pipeline using the **pipeline create** command. This validates and creates the pipeline, but will not take effect until you start the pipeline:

```
lat_client pipeline create --pipeline ~/pipeline.json
```

#### Example response:

```
10.0.0.1:8443: Created
10.0.0.2:8443: Created
```

In cases where there is an existing pipeline operating, it is necessary to stop the pipeline and remove the original pipeline before creating and starting the new pipeline.

Now that the pipeline has been created on all LAT Nodes, you can start the LAT by running the **pipeline start** commands:

```
lat_client pipeline start
```

Example responses:

```
10.0.0.1:8443: Running
10.0.0.2:8443: Running
```

#### Step 7: Confirm that Loading is Operating Correctly

With your pipeline in place and running, data will immediately begin loading from the S3 file groups you defined. If there were many files per file group, the LAT would first sort the files, then partition them for the fastest loading based on the sorting criteria you provided.

#### Observing Loading Progress

With the pipeline running, data immediately begins to load into Ocient. To observe this progress, you can use the **pipeline status** command from the LAT Client or monitor the LAT metrics endpoint of the Loader Nodes.

You can check the status with this command by using the **--list-files** flag to include a summary of the files included in the load.

```
lat_client pipeline status --list-files
```

#### Example responses:

```
10.0.0.1:8443: Running
10.0.0.2:8443: Running
```

```
Pipeline Files Processed: 0
Pipeline Error Count: 0
Pipeline Files Remaining: 2
```

#### orders

```
Files Processed: 0
Error Count: 0
Files Remaining: 1
```

```
products
  Files Processed: 0
  Error Count: 0
  Files Remaining: 1

orders

Status      Filename
processing  metabase_samples/jsonl/orders.jsonl

In Process Files:
processing  metabase_samples/jsonl/orders.jsonl

products

Status      Filename
processing  metabase_samples/jsonl/products.jsonl

In Process Files:
processing  metabase_samples/jsonl/products.jsonl
```

**Command:**

```
curl https://127.0.0.1:8443/v2/metrics/lat:type=pipeline
```

If your LAT is running without TLS configured, replace the port number of your LAT Hosts with 8080 and the protocol with http://.

**Example response:**

```
{
  "request": {
    "mbean": "lat:type=pipeline",
    "type": "read"
  },
  "value": {
    "partitions": [
      {
        "offsets_durable": 1,
        "pushes_errors": 0,
        "pushes_attempts": 1,
        "rows_pushed": 1,
        "offsets_written": 18759,
        "records_buffered": 0,
        "records_errors_column": 0,
        "records_errors_deserialization": 0,
        "source_bytes_buffered": 0,
        "records_errors_transformation": 0,
        "offsets_processed": 18759,
        "partition": "table_orders-0",
        "records_filter_accepted": 1,
        "records_errors_row": 0,
        "records_filter_rejected": 0,

```

```

        "records_errors_generic": 0,
        "producer_send_attempts": 0,
        "offsets_pushed": 18759,
        "pushes_unacknowledged": 0,
        "invalid_state": 0,
        "bytes_pushed": 88,
        "records_errors_total": 0,
        "offsets_buffered": 18759,
        "complete": 0,
        "offsets_end": 1,
        "producer_send_errors": 0
    },
    {
        "offsets_durable": 1,
        "pushes_errors": 0,
        "pushes_attempts": 1,
        "rows_pushed": 1,
        "offsets_written": 199,
        "records_buffered": 0,
        "records_errors_column": 0,
        "records_errors_deserialization": 0,
        "source_bytes_buffered": 0,
        "records_errors_transformation": 0,
        "offsets_processed": 199,
        "partition": "table_products-0",
        "records_filter_accepted": 1,
        "records_errors_row": 0,
        "records_filter_rejected": 0,
        "records_errors_generic": 0,
        "producer_send_attempts": 0,
        "offsets_pushed": 199,
        "pushes_unacknowledged": 0,
        "invalid_state": 0,
        "bytes_pushed": 145,
        "records_errors_total": 0,
        "offsets_buffered": 199,
        "complete": 0,
        "offsets_end": 1,
        "producer_send_errors": 0
    }
],
"paused": 1,
"bytes_buffered": 0,
"workers": 20
},
"timestamp": 1626970368,
"status": 200
}

```

### Check Row Counts in Tables

To confirm that you are seeing results in the target tables, you can also run some simple queries to check row counts. Depending on the streamloader role settings, the time for records to become queryable can vary from a few seconds to minutes:

### Example Queries:

```
Ocient> SELECT count(*) FROM public.orders;
COUNT(*)
-----
18760
```

```
Ocient> SELECT count() FROM public.products;
COUNT()
-----
200
```

Now you can explore the data in these four tables with any Ocient SQL queries.

Related Links

[LAT Overview](#)

[LAT Data Types in Loading](#)

[LAT Advanced Topics](#)

### 12.1.2 LAT Advanced Loading and Transformations

Data Pipelines are now the preferred method for loading data into the Ocient System. For details, see [Load Data](#).

This guide presents examples of some more advanced transformation topics such as exploding data, loading multiple tables from a single topic, working with arrays, and accessing nested data.

#### Loading Multiple Tables from a Topic

The [LAT Load JSON Data from Kafka](#) guide shows an example of loading each topic into its own table. However, each topic can route data to multiple tables and many Apache® Kafka® topics can stream load simultaneously to one or more tables in a single pipeline.

Here is an example `pipeline.json` file that will load data into two separate tables:

```
{
  "version": 2,
  "pipeline_id": "pipeline-metabase",
  "source": {
    "type": "kafka",
    "kafka": {
      "bootstrap.servers": "127.0.0.1:9092",
    }
  },
  "transform": {
    "topics": {
      "products": {
        "tables": {
          "metabase.public.products": {
            "columns": {
              "id": "id",
              "ean": "ean",
              "title": "title",
              "category": "category",
              "vendor": "vendor",
              "price": "price",
              "rating": "rating",
            }
          }
        }
      }
    }
  }
}
```

```

        "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\'T\\'HH:mm:ss[.SSS]X')",
      },
    },
    "metabase.public.product_ratings": {
      "columns": {
        "created_at": "now()",
        "product_id": "id",
        "rating": "rating",
      }
    }
  }
}

```

A few key points here.

1. You are loading data from a Kafka topic called **products**.
2. That topic is loading into a table called **database.schema.table**.
3. Most columns are simply mapped from the source JSON document into, however some are transformed from the original data.
  1. **created\_at** is parsed from a string using the **to\_timestamp** function
  2. **created\_at** for **product\_ratings** is set dynamically to the value of the **now()** function

### Loading Nested JSON Data

As you can see from the transformation, the syntax is similar to SQL, but the inner language used to traverse the JSON document and manipulate arrays can be unfamiliar. The approach used to select a property from the source JSON is to follow the tree with dot notation. For example:

Given a series of two documents like:

```

{
  "country" : "US",
  "states" : ["AR", "VA", "MD", "IL"],
  "languages" : [
    {
      "name" : "English",
      "locale" : "en-US",
      "dialects" : ["Northern American English", "Southeast Super-Regional English", "Western American English"],
    },
    {
      "name" : "Spanish",
      "locale" : "es-US",
      "dialects" : ["Mexican Spanish", "Caribbean Spanish", "Central American Spanish", "South American Spanish"],
    }
  ],
  "capital" : {
    "city" : "Washington",
    "region" : "District of Columbia"
  },
  "founded_in" : 1776
}

{
  "country" : "CA",

```

```

"provinces" : ["BC", "ON", "QC", "MB"],
"languages" : [
  {
    "name" : "English",
    "locale" : "en-CA"
  },
  {
    "name" : "French",
    "locale" : "fr-CA"
  }
],
"capital" : {
  "city" : "Ottawa",
  "region" : "Ontario"
},
"founded_in" : 1867
}

```

You can access a property like the country simply by referencing the property:

```

/* Ocient transformations are in this format */
{
  "column_name" : "transformation"
}

/* to apply the country property to the country_code column */
{
  "country_code" : "country"
}

```

You can access nested properties easily as well:

```

/* sets the column capital_city to values like "Ottawa" */
{
  "capital_city" : "capital.city"
}

```

If you want to join the capital sub-properties together, you can easily do so:

```

/* sets the column capital_city to values like "Ottawa, Ontario" */
{
  "capital_city" : "concat(capital.city, ',', capital.region)"
}

```

Note that these two documents are not fully cleansed, and you want to capture the states or provinces array and assign it to a common destination column, regions. You can do this with a coalesce operator ||:

```

/* sets the column regions to ["AR", "VA", "MD", "IL"] for the US and ["BC", "ON", "QC", "MB"] for Canada */
{
  "regions" : "states || provinces"
}

```

## Loading Array Data

Where arrays are encountered, special array notation is used. The benefit of this syntax is that properties in objects in arrays can also be accessed to return an array.

Another useful capability is the flattening of arrays. In the event that the values returned from a transformation includes nested arrays, you can flatten it to a single dimension array by appending [] to the end of the transformation





```

    "dialects" : NULL,
    "locale" : "en-CA"
  }
  {
    "country" : "CA",
    "language_name" : "French",
    "dialects" : NULL,
    "locale" : "fr-CA"
  }
}

```

If multiple columns in the transformation include an **EXPLODE**, they results will be “zipped” together. If the arrays are different lengths, the number of rows will match the longest array. The shorter array will populate the end of its rows with NULL. This is sometimes referred to as an “Outer Explode.”

### Transforming and Joining String Data

One common requirement is to split a string on a token to make an array. This is accomplished using transformation functions in the LAT.

Assuming a data input with a space separated string:

```
{ "states": "AL AK AZ AR CA CO CT DE" }
```

You can extract an array of state values in a few ways using the **tokenize** function. Tokenize has a number of flags to control how the pattern should be interpreted. For example 'q' indicates the pattern is a literal and 'i' indicates case insensitive regular expression.

```

/* You can use a literal space to split and load ["AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE"] */
{
  "states_col" : "tokenize(states, ' ', 'q')"
}

/* You can use a regular expression for whitespace to split and load ["AL", "AK", "AZ", "AR", "CA", "CO",
{
  "states_col" : "tokenize(states, '\\s', 'i')"
}

```

For more complex situations, you can split on any regular expression such as cases where values are wrapped with double quotes and might contain spaces. For this, you can use the expression `(?=((['"])*[\\"]){2})*[\\"]*$)` that leverages a zero-width positive lookahead assertion. Note that because this regex will be presented in a JSON document, certain characters must be escaped within the JSON string. For example, in JSON:

```

{
  "my_column" : "tokenize(states, ' (?=((['"])*[\\"]){2})*[\\"]*$)', 'i')"
}

```

This would actually pass the following regular expression to the **tokenize** function when the backslash escape sequences are processed:

```
(?=((['"])*[\\"]){2})*[\\"]*$)
```

Similarly, note that double backslashes are required to provide escape characters in the regular expression. For example, '\\s' would actually pass the regular expression `\\s` to the **tokenize** function.

Similar to splitting on a token, the LAT can easily join an array into a string with the **join** function. With source data that includes an array like the following:

```
{ "state_names": ["Alabama", "Alaska", "Arizona", "Arkansas", "California"] }
```

You can use the **join** function to produce this string:

```
/* join takes a string delimiter and will produce "Alabama, Alaska, Arizona, Arkansas, California"*/
{
  "state_names_col": "join(', ', state_names)"
}
```

## Related Links

[LAT Overview](#)

[LAT Data Types in Loading](#)

[LAT Advanced Topics](#)

## 12.2 LAT Overview

Data Pipelines are now the preferred method for loading data into the Ocient System. For details, see [Load Data](#).

The Ocient® Loading and Transformation (LAT) is a service used for transforming and streaming data into Ocient. The LAT transforms records from a configured Source using the [JMESPath](#) transformation language and streams records into Ocient tables.

### LAT Benefits

- Exactly-once processing from a Source
- Selecting data and arrays from nested JSON data
- Transforming records and routing into one or more tables
- Scalar data transformation (e.g., concatenate strings, uppercasing, rounding)
- Dynamic schema change handling

### LAT Data Flow

The LAT consumes JSON records, transforms them, and loads them into one or more Ocient tables. This process is referred to as a “pipeline”, with a single pipeline running per LAT instance. The pipeline is described by a JSON-based configuration that instructs the LAT how to consume records from a Source, how to transform the records, and which Ocient table to route the records to. As records are consumed, the transformations described in the configuration are applied to each record before forwarding the resulting records to Ocient.

Each pipeline configuration can map a record to 1 or more Ocient Tables, each table with its own transformation configuration. LAT can optionally be configured to write records which were unable to be processed to an Apache® Kafka® error topic.

Multiple LAT instances can be used in tandem to further parallelize a pipeline. The LAT makes use of the Kafka Consumer library to ensure that each Kafka Partition is only assigned to a single instance. To use multiple instances, configure each LAT instance with the same pipeline configuration. The LAT instances will automatically coordinate and distribute the load across the instances.

The LAT is controlled by an HTTP endpoint and a command line interface (CLI) is supplied to perform common actions such as starting, stopping, or updating the LAT pipelines.

## Related Links

[Connect to Ocient](#)

[Ingest Data with Legacy LAT Reference](#)

## Related Links

At the Whiteboard with Ocient: Loading and Transformation

## 12.3 LAT Data Types in Loading

Data Pipelines are now the preferred method for loading data into the Ocient System. For details, see [Load Data](#).

After extracting from source files or topics, data can be transformed and mapped to target columns in Ocient® tables.

LAT automatically binds many primitive source data types to Ocient column data types and automatically converts some source types to the target column type. Different source data types automatically bind to target columns using different conventions.

For example, a string in JSON such as '123' that loads into an INT column properly binds to the corresponding integer value 123. Similarly, an integer like 456 can automatically bind to an INT column as 456, a FLOAT column as 456.0, and a VARCHAR column as '456'. Finally, a string including Well-Known Text (WKT) representation of a POINT automatically binds into an ST\_POINT column.

### Empty String Handling

When you attempt to load an empty string into a target column with LAT, most data types raise an error and the record does not load into the target table. The exceptions are **varchar**, **decimal**, **binary**, and **array** column types. In those cases, the data warehouse loads these values:

- **varchar**: An empty string
- **decimal**: 0.0 (with the appropriate precision)
- **binary**: An empty binary blob
- **array**: An empty array

When you extract data from text-based formats like delimited data, the data warehouse loads empty data as an empty string unless you use the **empty\_as\_null** setting.

### NULL Handling

If you do not provide a value for a column in the target table, this table explains what value the data warehouse stores in the column. For the purposes of default value handling, these are equivalent:

- Pipeline is missing a column.
- Pipeline includes a column but the value in the JSON or the source data evaluates to **null**.
- Pipeline includes a column but the field is missing on the source record (e.g., **a.b** does not exist in a JSON record).

To transform a NULL or missing value into a default in the pipeline, use the transformation functions. For example: `not_null(a.b, 'my_default')`

| Ocient Column Definition | Database Behavior When Column is Missing or Column Value Evaluates to NULL |
|--------------------------|----------------------------------------------------------------------------|
| NOT NULL with default    | Default                                                                    |
| NOT NULL with no default | Record fails                                                               |
| Nullable with default    | Default                                                                    |
| Nullable with no default | NULL                                                                       |

### Data Type Binding

Refer to the [Data Types](#) for detailed information about each data type. These tables describe the automatic binding conventions of each data type.

**BIGINT**

|                               |                                                                                                                                                                                                                                                                                |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | Number, String                                                                                                                                                                                                                                                                 |
| <b>Note:</b>                  | Number and String can be either integer or floating point values and are converted to 64-bit signed integers. Decimals are dropped. '+', '-' can be used at the beginning of a string to indicate sign. String overflows result in errors, while numeric overflows are silent. |
| <b>Example:</b>               | <pre>{   "number_bigint": 1.0,   "string_bigint": "1" }</pre> <p>-&gt; 1, 1  </p>                                                                                                                                                                                              |

**BINARY**

|                               |                                                                                                               |
|-------------------------------|---------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | String                                                                                                        |
| <b>Note:</b>                  | Converted as UTF-8 to binary: each character in the string taking its binary representation.                  |
| <b>Example:</b>               | <pre>{   "string_binary": "hello" }</pre> <p>-&gt; BINARY(01101000 01100101 01101100 01101100 01101111)  </p> |

**BOOLEAN**

|                               |                                                                                                                                                                                     |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | Boolean, String                                                                                                                                                                     |
| <b>Note:</b>                  | For a value of true, a string must be one of (case insensitive): true, t, yes, on, y, 1. For a value of false, a string must be one of (case insensitive): false, f, no, off, n, 0. |
| <b>Example:</b>               | <pre>{   "boolean": true,   "string_0": "yes",   "string_1": "FALSE" }</pre> <p>-&gt; true, true, false  </p>                                                                       |

**DATE**

|                               |                                                                                                                                                                                         |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | String, Number                                                                                                                                                                          |
| <b>Note:</b>                  | If String, must be in format "YYYY-MM-DD". If Number must be an EPOCH_DAY. Use the built-in TO_DATE function to convert a string with a different date format to an EPOCH_DAY and bind. |
| <b>Example:</b>               | <pre>{   "string_date": "2021-11-04",   "epoch_date": 0 }</pre> <p>-&gt; DATE(2021-11-04), DATE(1970-01-01)  </p>                                                                       |

## DECIMAL

|                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | Number, String                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Valid Logical Types:</b>   | Decimal                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>Note:</b>                  | Number and String can be either integer or floating point values. Precision and scale for the loaded data are defined by the database table definition. The target column must support enough digits both before and after the decimal point to accommodate decimal values extracted from Fixed Width Binary or other record formats that support decimal data. If insufficient precision or scale exists in the target column, the data will fail to load. |
| <b>Example:</b>               | {                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

```
"number_decimal": 1.0,
"string_decimal": "1.1"
}
-> DECIMAL(1.0), DECIMAL(1.1) |
```

## DOUBLE, DOUBLE PRECISION

|                               |                                                                   |
|-------------------------------|-------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | Number, String                                                    |
| <b>Note:</b>                  | Number and String can be either integer or floating point values. |
| <b>Example:</b>               | {                                                                 |

```
"number_double": 1.0,
"string_double": "1"
}
-> 1.0, 1.0 |
```

## HASH

|                               |                                                                                                                  |
|-------------------------------|------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | String                                                                                                           |
| <b>Note:</b>                  | Converted as hexadecimal to binary: each character in the string must be a valid hexadecimal digit /[0-9a-f]+/i. |
| <b>Example:</b>               | {                                                                                                                |

```
"hex_string": "0a1b"
}
-> HASH(0000 1010 0001 1011) |
```

## INT

|                               |                                                                                                                                                                                                                                                                                |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | Number, String                                                                                                                                                                                                                                                                 |
| <b>Note:</b>                  | Number and String can be either integer or floating point values and are converted to 32-bit signed integers. Decimals are dropped. '+', '-' can be used at the beginning of a string to indicate sign. String overflows result in errors, while numeric overflows are silent. |

|                               |                |
|-------------------------------|----------------|
| <b>Valid JSON Bind Types:</b> | Number, String |
| <b>Example:</b>               | {              |

```
"number_int": 1.0,
"string_int": "1"
}
-> 1, 1 |
```

**IPV4**

|                               |                                                                     |
|-------------------------------|---------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | String                                                              |
| <b>Note:</b>                  | Source data must be in the form:<br>[0-255].[0-255].[0-255].[0-255] |
| <b>Example:</b>               | {                                                                   |

```
"string_ipv4": "192.168.0.1"
}
-> IPV4(192.168.0.1) |
```

**IP**

|                               |                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | String                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>Note:</b>                  | Source string can be in IPV4 or IPV6 format. IPV6 can appear in full form as shown in this example or in condensed form. Condensed form can use :: notation (e.g., 2001:cdba::3257:9652) and leading zeros in IPV6 hextets can also be omitted (e.g., 2001:0db8:0000:0000:0000:3257:9652 → 2001:db8:0:0:0:3257:9652). IPV6 can also appear as a string of 32 hex characters without colons (e.g., 2001cdba0000000000000000000032579652). |
| <b>Example:</b>               | {                                                                                                                                                                                                                                                                                                                                                                                                                                        |

```
"string_ip_ipv4": "192.168.0.1",
"string_ip_ipv6": "2001:cdba:0000:0000:0000:0000:3257:9652"
}
-> IP(192.168.0.1), IP(2001:cdba:0000:0000:0000:0000:3257:9652) |
```

**LINESTRING, ST\_LINESTRING**

|                               |                                                                              |
|-------------------------------|------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | String                                                                       |
| <b>Note:</b>                  | Must be in the WKT format “LINESTRING(LONG LAT, ... )” or “LINESTRING EMPTY” |
| <b>Example:</b>               | {                                                                            |

```
"string_st_linestring": "LINESTRING(-71.064544 42.28787, -90.444444 89.562993)"
}
-> ST_LINESTRING(-71.064544 42.28787, -90.444444 89.562993) |
```

**MATRIX**

|                               |                                                                                                                                                                                                                                |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | Array <a href="#">Array</a> , String                                                                                                                                                                                           |
| <b>Note:</b>                  | Arrays must be two-dimensional with the same rows and columns as specified in the table. String must be in JSON two-dimensional array format. Each element will be cast according to the element type's rules in this section. |
| <b>Example:</b>               | {                                                                                                                                                                                                                              |

```
"int_matrix": [[0, 1, 2],[3, 4, 5],[6, 7, 8]],
"string_matrix": "[[0,1],[2,3]]"
}
-> MATRIX[[0,1,2],[3,4,5],[6,7,8]], MATRIX[[0,1],[2,3]] |
```

**POINT, ST\_POINT**

|                               |                                         |
|-------------------------------|-----------------------------------------|
| <b>Valid JSON Bind Types:</b> | String                                  |
| <b>Note:</b>                  | Must be in the format "POINT(LONG LAT)" |
| <b>Example:</b>               | {                                       |

```
"string_st_point": "POINT(-71.064544 42.28787)"
}
-> ST_POINT(-71.064544, 42.28787) |
```

**POLYGON, ST\_POLYGON**

Ensure that polygons are oriented in your intended way when you load them as polygon orientation matters in Ocint semantics. If you intend all polygons to be oriented counter-clockwise, use the [LAT Transformation Functions](#) transform function to enforce counter-clockwise semantics.

|                               |                                                                                                                           |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | String                                                                                                                    |
| <b>Note:</b>                  | Must be in the WKT format "POLYGON((LONG LAT, ... ))" or "POLYGON((LONG LAT, ... ), (LONG LAT, ... ))" or "POLYGON EMPTY" |
| <b>Example:</b>               | {                                                                                                                         |

```
"string_st_polygon": "POLYGON((-87.62 41.87, -89.40 43.07, -87.906 43.04, -87.62 41.87))"
}
-> ST_POLYGON((-87.62 41.87, -89.40 43.07, -87.906 43.04, -87.62 41.87)) |
```

**REAL, FLOAT, SINGLE PRECISION**

|                               |                                                                                                                                |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | Number, String                                                                                                                 |
| <b>Note:</b>                  | Number and String can be either integer or floating point values. Precision and scale are inherited from the table definition. |
| <b>Example:</b>               | {                                                                                                                              |

```
"number_float": 1.0,
"string_float": "1"
}
-> 1.0, 1.0 |
```



**SMALLINT**

|                               |                                                                                                                                                                                                                                                                               |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | Number, String                                                                                                                                                                                                                                                                |
| <b>Note:</b>                  | Number and String can be either integer or floating point values and are converted to 16-bit signed integers. Decimals are dropped. '+','-' can be used at the beginning of a string to indicate sign. String overflows result in errors, while numeric overflows are silent. |
| <b>Example:</b>               | {<br><br>"number_smallint": 1.0,<br>"string_smallint": "1"<br>}                                                                                                                                                                                                               |
|                               | -> 1, 1                                                                                                                                                                                                                                                                       |

**TIME**

|                               |                                                                                                                                                                                                              |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | Number, String                                                                                                                                                                                               |
| <b>Note:</b>                  | String must be in the format: HH:MM:SS[.SSSSSSSSS]. Number must represent nanoseconds since midnight. Use the TO_TIME transformation function to convert an arbitrary string format to nanoseconds and bind. |
| <b>Example:</b>               | {<br><br>"number_time": 1000000000,<br>"string_time": "07:33:10.0001"<br>}                                                                                                                                   |
|                               | -> TIME(00:00:01.0000), TIME(07:33:10.0001)                                                                                                                                                                  |

**TIMESTAMP**

|                               |                                                                                                                                                                                                                           |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | Number, String                                                                                                                                                                                                            |
| <b>Note:</b>                  | String must be in the format: YYYY-MM-DD HH:MM:SS[.SSSSSSSSS]. Number must represent nanoseconds since epoch. Use the TO_TIMESTAMP transformation function to convert an arbitrary string format to nanoseconds and bind. |
| <b>Example:</b>               | {<br><br>"number_timestamp": 1000000000,<br>"string_timestamp": "2021-11-04 07:33:10.0001"<br>}                                                                                                                           |
|                               | -> TIMESTAMP(1970-01-01 00:00:01.0000), TIMESTAMP(2021-11-04 07:33:10.0001)                                                                                                                                               |

**TINYINT, BYTE**

|                               |                                                                                                                                                                                                                                                                               |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | Number, String                                                                                                                                                                                                                                                                |
| <b>Note:</b>                  | Number and String can be either integer or floating point values and are converted to 8-bit signed integers. Decimals are dropped. '+', '-' can be used at the beginning of a string to indicate sign. String overflows result in errors, while numeric overflows are silent. |
| <b>Example:</b>               | {                                                                                                                                                                                                                                                                             |

```
"number_smallint": 1.0,
"string_smallint": "1"
}
-> 1, 1 |
```

#### TUPLE

|                               |                                                                                                                                                                                                                  |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | Array, String                                                                                                                                                                                                    |
| <b>Note:</b>                  | String must be in JSON array format "[e0,e1,e2]". Each element will be cast to the tuple element type at that element's index from the Ocint table according to the rules for that element type in this section. |
| <b>Example:</b>               | {                                                                                                                                                                                                                |

```
"double_int": [1.0, 1],
"string_array": ["Ocint", [0, 1]],
"string_tuple": "[1.0, 1]",
"tuple_tuple": [[1.0, 1], ["Ocint", 2]]
}
-> TUPLE(DOUBLE, INT)(1.0, 1), TUPLE(STRING, INT[])( "Ocint", INT[0, 1]), TUPLE(DOUBLE, INT)(1.0, 1), TU
```

#### TYPE[]

|                               |                                                                                                                                                                                          |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | Array, String                                                                                                                                                                            |
| <b>Note:</b>                  | String must be in JSON array format "[e0,e1,e2]". Each element will be cast to the array element type from the Ocint table according to the rules for that element type in this section. |
| <b>Example:</b>               | {                                                                                                                                                                                        |

```
"int_array": [0,1,2],
"2d_array": [[0, 1, 2],[3, 4]],
"string_array": "[0,1,2]",
"tuple_array": [[0, 1.0, "Ocint"],[1, 2.0, "rocks"]]
}
-> INT[0,1,2], INT[] [INT[0,1,2],INT[3,4]], INT[0,1,2] TUPLE(INT, FLOAT, VARCHAR)[(0, 1.0, "Ocint"), (1,
```

#### UUID

|                               |                                                    |
|-------------------------------|----------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | String                                             |
| <b>Note:</b>                  | Source data must be a valid hyphen-separated UUID. |
| <b>Example:</b>               | {                                                  |

```
"string_uuid": "40e6215d-b5c6-4896-987c-f30f3678f608"
}
-> UUID(40e6215d-b5c6-4896-987c-f30f3678f608) |
```

#### VARCHAR, CHAR

|                               |                                                                         |
|-------------------------------|-------------------------------------------------------------------------|
| <b>Valid JSON Bind Types:</b> | String, Number, Boolean                                                 |
| <b>Note:</b>                  | Number and Boolean types are converted to their string representations. |
| <b>Example:</b>               | {                                                                       |

```
"number_string": 1000,
"boolean_string": true,
"string_string": "string"
}
-> "1000", "true", "string" |
```

### Load Geospatial Data

Geospatial data can be loaded from a variety of formats into Ocint. The Ocint System requires special considerations when you load geospatial data into Ocint including:

- Coordinate System
- Supported Geospatial Loading Formats
- Geospatial Type Conversion
- POINT Normalization
- POLYGON Normalization

#### Coordinate System

Ocint uses Spatial Reference System ID (SRID) 4326 for all internal geospatial data types. SRID 4326 is defined in the [WGS84 Standard](#). Before you load the source data, all data should have the SRID 4326 data type. The loading operation ignores any source data that contains the coordinate system SRID information.

#### Supported Geospatial Loading Formats

##### Well-Known Text

The primary data format for loading geospatial data in Ocint is the Well-Known Text (WKT) format. This string format is a portable format that can represent points, lines, and polygons.

String data in WKT format can automatically load into POINT, LINESTRING, and POLYGON Ocint column types. In addition, arrays of strings in WKT format can automatically load into array columns of a suitable geospatial data type. The Ocint System assumes raw strings are in WKT format when the system loads these strings into geospatial column types.

Each LINESTRING or POLYGON value can be up to a maximum of 512 MB in size. This means a LINESTRING or POLYGON can contain approximately 32 million point values.

- WKT strings are whitespace insensitive, so POINT(10 20) is equivalent to POINT (10 20).
- WKT strings are case insensitive.
- Extended Well-Known Text (EWKT) is not supported. To load EWKT data, load the substring after the EWKT semicolon as the WKT format.

## Well-Known Binary and Extended Well-Known Binary

The other data formats that can load into Ocint are Well-Known Binary (WKB) and Extended Well-Known Binary (EWKB) formats. The Ocint System requires an explicit transformation function for both WKB and EWKB formats when you load data and convert it into the target column type.

You can use the function `st_geomfromewkb` to transform WKB and EWKB data.

In both cases, the Ocint System represents the WKB or EWKB string data as hexadecimal string data. The hexadecimal string can be one of these formats:

- Only hexadecimal digits: 000000000140000000000000004010000000000000
- Prefixed with 0x: 0x000000000140000000000000004010000000000000
- Prefixed with \x: \x000000000140000000000000004010000000000000

## Geometry Conversion

Ocint provides the automatic conversion of geometries from simpler types to more complex types. The conversion allows the load of a LINESTRING column with a POINT, or a POLYGON column with a POINT or LINESTRING. This capability also allows the load of arrays of POLYGON data from the source arrays of a combination of POINT, LINESTRING, and POLYGON data, which represents a geometry collection.

## POINT Normalization

During loading, Ocint automatically performs normalization of POINT data into a regular format used within Ocint. The Ocint System performs the following operations on POINT data during the load:

- Constrain longitude to [-180, 180) and latitude to [-90, 90]; wrap around invalid coordinates using correct geographical handling.
- Snap points near the pole to the pole.
- Set longitude of points on the pole to 0
- Remove signed zeros from coordinates, so -0 becomes 0.

## POLYGON Normalization

Ocint follows a counterclockwise rotation convention to indicate the outer ring of a POLYGON. The inner ring follows a clockwise rotation. If the Ocint System loads a POLYGON with a clockwise outer ring, the system indicates that the POLYGON is outside of the given ring.

In some cases, the POLYGON load can lead to unexpected results when the source data follows a different polygon rotation convention.

To account for this, the LAT provides the `st_forcepolygonccw` function that forces a counterclockwise rotation of the outer polygon ring and a clockwise rotation of the inner polygon ring when applicable.

## Geospatial Transformation Functions

The LAT has a select set of transformation functions to construct geospatial types and manipulate them during the load. You can find the supported geospatial transformation functions in [LAT Transformation Functions](#).

## Related Links

[Ingest Data with Legacy LAT Reference](#)

[Understanding Data Types](#)

[Geospatial Functions](#)

## 12.4 LAT Pipeline Configuration

Data Pipelines are now the preferred method for loading data into the Ocient System. For details, see [Load Data](#).

A pipeline configuration is a JSON file that fully describes the necessary elements to run a pipeline, including

- **Source** - the source location from which a pipeline should read records to process
- **Extract** - how a pipeline should extract data from the source
- **Transform** - how a pipeline should transform incoming records
- **Sink** - the destination where a pipeline should write transformed rows

The JSON file is a list of keys with (possibly nested) values. The available key-value pairs are documented in these sections.

### Pipeline

Top level configuration for a pipeline.

Required keys:

- Pipeline Configuration
- Pipeline Configuration
- Pipeline Configuration
- Pipeline Configuration

The following is an example of the structure of a pipeline configuration.

```
{
  "version": 2,
  "source": {
    "type": "kafka"
    // kafka source configuration
  },
  "sink": {
    "type": "ocient"
    // ocient sink configuration
  },
  "extract": {
    // extract configuration
  },
  "transform": {
    "topics": {
      "my-topic": {
        "tables": {
          "my-table": {
            "columns": {
              "my-col1": "record_field_1",
              "my-col2": "record_field_2"
            }
          }
        }
      }
    }
  }
}
```

### Configuration

**version**

The pipeline's version. Required value is 2.

|           |            |
|-----------|------------|
| Type:     | <b>int</b> |
| Required: | Yes        |
| Default:  |            |

**pipeline\_id**

A unique identifier for this pipeline. Allowed characters are **a-z**, **A-Z**, **0-9**, **\_**, and **-**.

The **pipeline\_id** is used to uniquely identify a pipeline. It is used for a few purposes:

Deduplication scope. See [LAT Advanced Topics](#).

1. For Apache® Kafka® loads, the consumer **group.id** is set to **ocient-lat-[pipeline\_id]**.

For most loads from File Sources, it is advisable to leave the **pipeline\_id** unset when creating a pipeline using the LAT client. The client will assign the pipeline a random UUID.

|           |                                     |
|-----------|-------------------------------------|
| Type:     | <b>string</b>                       |
| Required: | No                                  |
| Default:  | LAT client randomly generated UUID. |

**workers**

The number of workers this pipeline should use for processing records.

|           |                                                     |
|-----------|-----------------------------------------------------|
| Type:     | <b>int</b>                                          |
| Required: | No                                                  |
| Default:  | <b>DEFAULT_NUM_WORKERS</b> in service configuration |

**log\_original\_records**

Add original records to the error log when errors occur. When this setting is set to **true**, LAT writes data extracted from the source to the error log and in some error messages. By default, this setting is **false** and source data is not written to the error log nor included in error messages.

In order to enable this setting, the [LAT Service Configuration](#) service configuration must also be enabled.

This configuration only affects pipelines that do not use an **Pipeline Configuration**.

**seek\_on\_rebalance**

Whether to seek a newly assigned partition to the latest known durable record prior to resuming processing. Disabling this behavior should typically be reserved for test scenarios and is only supported for Kafka loading.

|           |                |
|-----------|----------------|
| Type:     | <b>boolean</b> |
| Required: | No             |
| Default:  | <b>true</b>    |

**continue\_on\_unrecoverable\_error**

Whether to allow workers to continue processing when they encounter an ordinarily unrecoverable error.

|           |                |
|-----------|----------------|
| Type:     | <b>boolean</b> |
| Required: | No             |
| Default:  | <b>false</b>   |

**single\_file\_mode**

Enable the single file mode. This mode is designed for a specific use case where there are few files but each file size is large. When using this mode, only a single file is processed at a time, so [LAT Source Configuration](#) must be equal to 1 and only one [LAT Source Configuration](#) can be defined. The single file is processed in parallel by the number of workers defined by the pipeline **workers** setting. There is no need to enable this in common use cases.

|           |                |
|-----------|----------------|
| Type:     | <b>boolean</b> |
| Required: | No             |
| Default:  | <b>false</b>   |

A known limitation exists with the LAT metrics when you use the single file mode. Metrics returned from the `lat_client pipeline status` command might not display the expected count of files processed, processing, and so on. However, the load still displays the correct **PROCESSING** and **COMPLETED** statuses.

**error\_topic**

A Kafka Topic to write records which cannot be processed. If absent, error records will be logged to the error log file without additional processing.

This configuration is only available if a Kafka Source is configured for the pipeline. The configuration for that source will apply to the Kafka Producer for this topic.

|           |               |
|-----------|---------------|
| Type:     | <b>string</b> |
| Required: | No            |
| Default:  | <b>null</b>   |

**polling\_duration**

Maximum duration to block while polling for new records from a Source, in milliseconds.

|           |             |
|-----------|-------------|
| Type:     | <b>int</b>  |
| Required: | No          |
| Default:  | <b>1000</b> |

**source**

Source configuration section. See [LAT Source Configuration](#) for nested configuration.

|           |        |
|-----------|--------|
| Type:     | object |
| Required: | Yes    |
| Default:  |        |

**sink**

Sink configuration section. See [LAT Sink Configuration](#) for inline configuration details. **sink** cannot be set if **sink\_name** is set. **sink** can be omitted if a default sink is defined as an [LAT Sink Configuration](#).

|           |        |
|-----------|--------|
| Type:     | object |
| Required: | No     |
| Default:  | null   |

**sink\_name**

The name of an externally configured sink. **sink\_name** cannot be set if **Pipeline Configuration** is set. See [LAT Sink Configuration](#). **sink\_name** can be omitted if a default sink is defined as an [LAT Sink Configuration](#).

|           |        |
|-----------|--------|
| Type:     | string |
| Required: | No     |
| Default:  | null   |

**extract**

Extract configuration section. See [LAT Extract Configuration](#) for nested configuration.

|           |                                                    |
|-----------|----------------------------------------------------|
| Type:     | object                                             |
| Required: | No                                                 |
| Default:  | Defaults to JSON record type with default settings |

**transform**

Transform configuration section. See [LAT Transform Configuration](#) for nested configuration.

|           |        |
|-----------|--------|
| Type:     | object |
| Required: | Yes    |
| Default:  |        |

**Related Links**

[Ingest Data with Legacy LAT Reference](#)

**12.4.1 LAT Source Configuration**

Data Pipelines are now the preferred method for loading data into the Ocident System. For details, see [Load Data](#).



**source**

A source configuration object.

Required keys:

- [Source Configuration](#)

**source.type**

Type of source to use in the pipeline.

|                  |        |
|------------------|--------|
| <b>Type:</b>     | string |
| <b>Required:</b> | Yes    |
| <b>Default:</b>  |        |

Allowed values:

- **kafka**: see [Source Configuration](#) for additional configuration.
- **s3**: see [Source Configuration](#) and [Source Configuration](#) for additional configuration.
- **local**: see [Source Configuration](#) for additional configuration.

**Load From a File Source**

Currently, LAT supports loading files from an Amazon® Web Services<sup>SM</sup> (AWS<sup>SM</sup>) S3 instance (like AWS S3 or IBM® Cloud® Object Storage) or from a local file system. Common configuration for all File Sources will be listed in [Source Configuration](#), followed by sections describing source-specific configurations.

File Sources are defined by creating “file groups” that represent a logical set of files to be loaded. Each file group is given a **file\_group\_name** that corresponds to the file group name in the transform section of the pipeline configuration. Each file group has settings to select the specific files that should be loaded.

The LAT supports loading from individual files with extensions such as **.jsonl** and **.csv**. GZIP compression is supported. However, files using the TAR archive format, LZOP compression, or ZIP compression are not supported.

If you load files with unsupported archive or compression formats, the load might stop or produce unexpected results.

The performance of loading into Ocient® is greatly improved when records are presented in a well ordered time sequence. This sequence allows more efficient creation of segments and sorting of records into TimeKey® buckets. For this reason, the LAT has options to define how files in a file group should be sorted prior to loading.

**File Group Filtering and Sorting Example**

If you want to load all the files under a certain directory, the file group configuration is basic:

```
"my_file_group": {
  "prefix": "/dir1",
  "file_matcher_syntax": "glob",
  "file_matcher_pattern": "**",
  "sort_type": "lexicographic"
}
```

This file group configuration will select all files under **/dir1** (including those under its subdirectories) and sort them lexicographically.

However, if a user needs to selectively choose, filter, and sort the files selected, this process provides flexible options for doing so.

For an individual file group in a file type load, there are multiple steps to select, rename, filter, and sort the files retrieved. The files selected occur through this sequence:

1. Prefix filtering
2. File matching
3. File renaming
4. Timestamp extract pattern matching (only `extract_timestamp`)
5. Range matching
6. Final list sorting

For example, given these files from a local file system:

```
/dir1/file.json
/dir1/files1/year=2021/month=01/day=01/files2/10-00-00.csv
/dir1/files1/year=2021/month=01/day=01/file.json
/dir1/files1/year=2021/month=01/day=01/files3/11-00-00.json
/dir1/files1/year=2021/month=01/day=01/files2/12-00-00.json
/dir1/files1/year=2021/month=01/day=01/files2/14-00-00.json
/dir1/files1/year=2021/month=01/day=01/files2/13-00-00.json
```

And a `file_group` configuration:

```
"my_group": {
  "prefix": "/dir1/files1/",
  "file_matcher_syntax": "regex",
  "file_matcher_pattern": "/dir1/files1/year=(\\d{4})/month=(\\d{2})/day=(\\d{2})/files\\d/(\\d{2})-(\\d{2})-(\\d{2})",
  "rename_format": "{1}-{2}-{3}-{4}-{5}-{6}",
  "sort_type": "extract_timestamp",
  "path_timestamp_pattern": "yyyy-MM-dd-HH-mm-ss",
  "start_time": "2021-01-01T13:00:00",
  "stop_time": "2021-01-01T14:00:01"
}
```

### Step 1: Prefix Filtering

Prefix filtering occurs first and includes only files that are in the matching prefix paths. The prefix is the path part following the bucket for S3 types, and it is the path to files in a local file type load.

```
"prefix": "/dir1/files1/"
```

Result: `/dir1/file.json` is filtered out because it is not in the `prefix`.

### Step 2: File Matching

Next, file matching uses a pattern to match files that should be included. File matcher patterns apply to the full path of the file including any prefix defined in the prior step.

```
"file_matcher_syntax": "regex"
"file_matcher_pattern": "/dir1/files1/year=(\\d{4})/month=(\\d{2})/day=(\\d{2})/files\\d/(\\d{2})-(\\d{2})-(\\d{2})"
```

Result: `/dir1/files1/year=2021/month=01/day=01/files2/10-00-00.csv` is filtered out because it does not end in `.json`. `/dir1/files1/year=2021/month=01/day=01/file.json` is filtered out because it is not in the subdirectory `files1`.

### Step 3: File Renaming

File renaming is a step that can be useful in a case where the selected files have disparate file names that would make it difficult to extract timestamps from. Files are only renamed internally to the LAT to facilitate the file selection process; they are *not* actually renamed locally or on S3.

By unifying each file name into a consistent format, this step should make it easier to use the `extract_timestamp` sort type, or even to lexicographically sort on some parameter within the file name itself.

```
"file_matcher_pattern": "/dir1/files1/year=(\\d{4})/month=(\\d{2})/day=(\\d{2})/files\\d/(\\d{2})-(\\d{2})-(\\d{2})"
"rename_format": "{1}-{2}-{3}-{4}-{5}-{6}",
```

Result: The files that match the `file_matcher_pattern` regular expression are renamed using the `rename_format`.

Current Files at this Step:

```
/dir1/files1/year=2021/month=01/day=01/files3/11-00-00.json
/dir1/files1/year=2021/month=01/day=01/files2/12-00-00.json
/dir1/files1/year=2021/month=01/day=01/files2/14-00-00.json
/dir1/files1/year=2021/month=01/day=01/files2/13-00-00.json
```

Renamed file list that enters the next step:

```
2021-01-01-11-00-00
2021-01-01-12-00-00
2021-01-01-14-00-00
2021-01-01-13-00-00
```

#### Step 4: Timestamp Extract Pattern Matching

A filter is applied when the `sort_type` is `extract_timestamp`. The `extract_timestamp` sort type extracts timestamps from the file's filepath. Either a `path_timestamp_pattern` or a `file_timestamp_pattern` must be set. If a filename does not match the set pattern, it will be filtered out.

```
"sort_type": "extract_timestamp"
"path_timestamp_pattern": "yyyy-MM-dd-HH-mm-ss",
```

Result: In this example, a `path_timestamp_pattern` is set, which is a `DateTimeFormatter` pattern that will extract a timestamp from the starting from the beginning of the filename, used for the next step. No files are filtered here because the rename step above was able to unify filename formats in a way that the `path_timestamp_pattern` could match on all the files.

Internal to the LAT, the files are associated with their datetimes, which will be used in the next step:

```
filename -> datetime extracted in ISO 8601 format
2021-01-01-11-00-00 -> 2021-01-01T11:00:00Z
2021-01-01-12-00-00 -> 2021-01-01T12:00:00Z
2021-01-01-14-00-00 -> 2021-01-01T14:00:00Z
2021-01-01-13-00-00 -> 2021-01-01T13:00:00Z
```

There are potential pitfalls with using the `extract_timestamp` `sort_type`, described in the [LAT Source Configuration](#) section.

#### Step 5: Range Matching

Range matching occurs next based on the chosen `sort_type` algorithm.

- For `extract_timestamp` or `metadata`, a timestamp is associated with each file for sorting. A start and end time can be optionally provided to limit files to select.
- For `lexicographic`, the filename itself is used for sorting. A start and end file name can be provided to limit the files that are selected.

```
"sort_type": "extract_timestamp"
"path_timestamp_pattern": "yyyy-MM-dd-HH-mm-ss",
"start_time": "2021-01-01T13:00:00"
"stop_time": "2021-01-01T14:00:01"t
```

Result: The file 2021-01-01-12-00-00 is filtered out because it is not in the start and end time ranges.

#### Step 6: Final List Sorting

Finally, the fully filtered file list contains two files from your original list. These are then sorted according to the `sort_type` algorithm and partitioned across the workers for loading.

Remaining files before sorting:

```
2021-01-01-14-00-00
2021-01-01-13-00-00
```

Final list - sorted, renamed files:

```
2021-01-01-13-00-00
2021-01-01-14-00-00
```

(original file names):

```
/dir1/files1/year=2021/month=01/day=01/files2/13-00-00.json
/dir1/files1/year=2021/month=01/day=01/files2/14-00-00.json
```

#### Example S3 Source Configuration

The names defined for `file_groups` (“my\_file\_group” in this example) should match the `file_groups` used in the transform configuration section.

```
{
  "source": {
    "type": "s3",
    "endpoint": "http://some.endpoint/",
    "bucket": "my_bucket",
    "file_groups": {
      "my_file_group": {
        "prefix": "some/prefix/",
        "file_matcher_syntax": "glob",
        "file_matcher_pattern": "**.json",
        "sort_type": "extract_timestamp",
        "path_timestamp_pattern": "'dir1/files1/'yyyy'-'MM'-'dd'T'HH':'mm':'ss",
      }
    },
    "compression": "gzip"
  },
  "transform": {
    "file_groups": {
      "my_file_group": {
        "tables": {
          ...
        }
      }
    }
  }
}
```

#### Example Local File Source Configuration

The names defined for `file_groups` (“my\_file\_group” in this example) should match the `file_groups` used in the transform configuration section.

```
{
  "source": {
    "type": "local",
    "file_groups": {
      "my_file_group": {
        "prefix": "/path/to/data/directory/",

```

```

        "file_matcher_syntax": "glob",
        "file_matcher_pattern": "**.json",
        "sort_type": "extract_timestamp",
        "path_timestamp_pattern": "'dir1/files1/'yyyy'-'MM'-'dd'T'HH':'mm':'ss",
    },
    },
    "compression": "gzip"
},
"transform" : {
    "file_groups" : {
        "my_file_group" : {
            "tables" : {
                ...
            }
        }
    }
}
}
}
}

```

### Common File Source Configuration

#### source.file\_groups

An object that maps file group names to their corresponding configuration objects.

|           |        |
|-----------|--------|
| Type:     | object |
| Required: | Yes    |
| Default:  |        |

#### source.file\_groups.<file\_group\_name>

The configuration object for a file group.

|           |        |
|-----------|--------|
| Type:     | object |
| Required: | Yes    |
| Default:  |        |

#### source.file\_groups.<file\_group\_name>.prefix

For an S3 Source, the **prefix** is used to get a subset of S3 objects from the bucket. A more specific prefix can improve time required to list files in S3 sources.

For a Local File Source, **prefix** is an absolute or relative path from the working directory of the LAT. In either case, the path can be a file or a directory. If the path is a directory, LAT tries to load all files recursively under this path. When looking for files, LAT follows the symbol links. LAT only loads regular files that are readable and not hidden. Note that shell-specific expansions like `~` are not supported.

You can include multiple prefixes for a single file group if you specify this property as an array of strings. In that case, the LAT loads all files that are under any of the prefixes in that array. The system loads files that match more than one prefix within a single file group only once.

For a Local File Source, the path used for prefix is the path on the server where the LAT is running, which might not be the same machine where you are running the LAT Client Command Line Interface to create your pipeline.

|           |                        |
|-----------|------------------------|
| Type:     | string or string array |
| Required: | No                     |
| Default:  | ""                     |

`source.file_groups.<file_group_name>.file_matcher_syntax`

The `file_matcher_syntax` defines the type of syntax used by the `file_matcher_pattern` setting, which includes matching files in the file group. The pattern is applied to the fully qualified file name in the list of files found under the `prefix`.

See the `file_matcher_pattern` description for examples and complete syntax details for `glob` and `regex` options.

|           |                                                                                                                                    |
|-----------|------------------------------------------------------------------------------------------------------------------------------------|
| Type:     | string                                                                                                                             |
| Required: | No, although it is not valid to have only one of <code>file_matcher_syntax</code> or <code>file_matcher_pattern</code> be provided |
| Default:  | <code>glob</code>                                                                                                                  |

Allowed values:

- `glob` - a simplified pattern matching system based on wildcards
- `regex` - a regular expression syntax

`source.file_groups.<file_group_name>.file_matcher_pattern`

The `pattern` used to select files from the `prefix`-filtered list. Files that match the pattern are included in the file group. The **fully qualified file name** (path and filename under the `prefix`) are matched. Files that do not match are excluded from the file group.

This pattern is also used in the case of LAT **file renaming** - if a `rename_format` is provided, the LAT will attempt to internally rename files using capture groups in the `file_matcher_pattern`. If capture groups are provided in the `file_matcher_pattern`, they will be extracted and placed into the `rename_format` in order to generate a renamed file. Capture groups do *not* need to be named; the captured values are assigned to the `rename_format` sequentially, i.e. the first capture group in the `file_matcher_pattern` will be placed in `{1}` in the `rename_format`.

Files are only renamed internally to the LAT to facilitate the file selection process; they are *not* actually renamed locally or on S3.

This matched file list can be further filtered using an `extract_timestamp's` pattern and the start/end ranges of any file group type algorithm. See an expanded example in the [Source Configuration](#) section.

A double backslash within the regular expression is necessary to generate valid escape characters for the `pipeline.json`. For example, the capture group `(\d{4})` needs to be escaped as `(\\d{4})`.

|           |                                                                                                                                    |
|-----------|------------------------------------------------------------------------------------------------------------------------------------|
| Type:     | string                                                                                                                             |
| Required: | No, although it is not valid to have only one of <code>file_matcher_syntax</code> or <code>file_matcher_pattern</code> be provided |
| Default:  | <code>**</code>                                                                                                                    |

## Example

With a `file_matcher_syntax` of `glob` and `file_matcher_pattern` of `**/*.json`, all files (after prefix-filtering) that end in `.json` will be selected.

A `**` in the glob based pattern ensures that any pattern including directory boundaries are matched. A pattern `*` does not match the directory character `/`.

### Example

With a `file_matcher_syntax` of `regex`, a `file_matcher_pattern` of `auctions.*-2021.json` would use a regular expression to ensure that files such as `auctions-12-01-2021.json` and `auctions-12-02-2021.json` would be selected.

See [getPathMatcher\(\)](#) for complete syntax details of the `glob` and `regex` options.

Common Matcher Patterns:

| Matcher Syntax | Pattern | Meaning                                                                                 |
|----------------|---------|-----------------------------------------------------------------------------------------|
| glob           | *       | Matches zero or more characters without crossing directory boundaries                   |
| glob           | **      | Matches zero or more characters crossing directory boundaries                           |
| glob           | ?       | Matches exactly one character of a name component                                       |
| glob           | []      | Matches any of the characters in the bracket expression (e.g., [abc]). Supports ranges. |
| regex          | .       | Matches any character                                                                   |
| regex          | \d      | Matches any digit                                                                       |
| regex          | \D      | Matches any non-digit                                                                   |
| regex          | []      | Matches any of the characters in the bracket expression (e.g., [abc]). Supports ranges. |
| regex          | *       | Matches the preceding character zero or more times                                      |
| regex          | +       | Matches the preceding character one or more times                                       |
| regex          | ?       | Matches the preceding character once or not at all                                      |

```
source.file_groups.<file_group_name>.rename_format
```

A Java® [MessageFormat](#) string. Captured groups from the `file_matcher_pattern` will be placed into the `rename_format` string's format elements corresponding to the order they were captured. The renamed file is used for subsequent sorting of files. When using `lexicographic` sort, the renamed file is used instead of the original filename. For `extract_timestamp` sorting, the `path_timestamp_pattern` is applied against the renamed file. This is best elucidated with an example:

```
filename: 20210102.json
file_matcher_pattern: (\\d{4})(\\d{2})(\\d{2}).*
rename_format: "file-{1}-{3}-{2}"
"sort_type": "extract_timestamp"
"path_timestamp_pattern": "'file-'yyyy'-'MM'-'dd"
```

Renames the file to: `"file-2021-02-01"`

The `"path_timestamp_pattern"` extracts an ISO-8601 datetime of `2021-02-01T00:00:00Z` from the renamed file

Files are only renamed internally to the LAT to facilitate the file selection process; they are *not* actually renamed locally or on S3.

|           |                                                                                             |
|-----------|---------------------------------------------------------------------------------------------|
| Type:     | string                                                                                      |
| Required: | No                                                                                          |
| Default:  | None - if <code>rename_format</code> is not provided, files will not undergo a rename step. |

`source.file_groups.<file_group_name>.sort_type`

Defines the sorting algorithm used for sorting the files selected for loading in this file group by time. Files should be sorted in time order for best loading performance and the creation of efficient Ocident segments. `sort_type` can be `extract_timestamp`, `metadata`, or `lexicographic`, and different settings apply to each of these selections.

|           |                   |
|-----------|-------------------|
| Type:     | string            |
| Required: | Yes               |
| Default:  | <code>none</code> |

Allowed values:

- `extract_timestamp`: extract the timestamp from the file name or file path information. Either `path_timestamp_pattern` or `file_timestamp_pattern` is required when choosing this `sort_type`.
- `metadata`: extract the timestamp from the source file's metadata. This option uses the file last modified time when sorting the list.
- `lexicographic`: sort files based on the alphanumeric sort of files in the file group using the full path and filename as the sorting key.

The system breaks the tie among files in a lexicographic way by using their fully qualified original file names.

`source.file_groups.<file_group_name>.path_timestamp_pattern`

A Format string with `DateTimeFormatter` patterns used to extract a datetime from the path portion of a file's fully-qualified filename. This timestamp is used with the `extract_timestamp` sort type to order the selected files in the file group prior to loading. It is attempted on every file's fully-qualified filename under the `prefix`; files that do not match the pattern will be skipped and the remaining subset will be used for sorting.

|           |                                                                                                                                                          |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Type:     | string                                                                                                                                                   |
| Required: | One of <code>path_timestamp_pattern</code> or <code>file_timestamp_pattern</code> is required for an <code>extract_timestamp</code> sort type file group |
| Default:  | <code>none</code>                                                                                                                                        |

`source.file_groups.<file_group_name>.file_timestamp_pattern`

A Format string with `DateTimeFormatter` patterns used to extract a datetime from the base filename of a file's fully-qualified filename. This timestamp is used with the `extract_timestamp` sort type to order the selected files in the file group prior to loading. It is attempted on every file's base filename under the `prefix`; files that do not match the pattern will be skipped and the remaining subset will be used for sorting.



|           |                                                                                                                                                                |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Type:     | string                                                                                                                                                         |
| Required: | One<br>of <code>path_timestamp_pattern</code> or <code>file_timestamp_pattern</code> is<br>required for an <code>extract_timestamp</code> sort type file group |
| Default:  | none                                                                                                                                                           |

`source.file_groups.<file_group_name>.start_time`

An ISO-8601 compliant Date or Datetime, used as the lower bound to filter for files in a `extract_timestamp` or `metadata` sort type file group. Datetimes are UTC unless a timezone is provided. Inclusive.

|           |                          |
|-----------|--------------------------|
| Type:     | string                   |
| Required: | No                       |
| Default:  | <code>Instant.MIN</code> |

`source.file_groups.<file_group_name>.stop_time`

An ISO-8601 compliant Date or Datetime, used as the upper bound to filter for files in a `extract_timestamp` or `metadata` sort type file group. Datetimes are UTC unless a timezone is provided. Exclusive.

|           |                          |
|-----------|--------------------------|
| Type:     | string                   |
| Required: | No                       |
| Default:  | <code>Instant.MAX</code> |

#### *Extract Timestamp Default Values*

With an `extract_timestamp sort_type`, if a certain time unit is not provided, they will default to their respective value in '1970-01-01T00:00:00.000000'.

#### Example:

```
Filename: `July03.json`
`path_timestamp_pattern` or `file_timestamp_pattern`: `MMMMdd'.json`
```

The full timestamp extracted from this filename will be 1970-07-03T00:00:00.000000 UTC.

#### Potential Pitfalls

Pitfall 1: This default value should be considered when a `start_time` or `stop_time` is provided but a certain time unit is not present within the filename.

#### Example:

Capture files ranging from July 1st through July 3rd:

```
Filenames: July01.json, July02.json, July03.json
"file_timestamp_pattern": "MMMMdd'.json"
"start_time": "1970-07-01"
"stop_time": "1970-07-04"
```

Note that 1970 was used as the year in the `start_time` and `stop_time`, but the system could not extract a year from the filenames. An alternative strategy is to use the File Renaming step to insert missing dates or times.

Pitfall 2: Only applicable to using the `extract_timestamp` sort\_type with `start_time` and `stop_time`: Consider the time unit granularities between your pattern and start/stop times.

**Example:**

```

FileNames: dir1/2021/01/01/00:00:00.json, dir1/2021/01/01/01:00:00.json, dir1/2021/01/01/05:00:00.json
"path_timestamp_pattern": "'dir1/'yyyy'/'MM'/'dd'/'"
"start_time": "2021-01-01T00:00:00"
"stop_time": "2021-01-01T04:00:00"

```

Given this configuration, one might expect that the files that are hour 4 and later would be filtered out. However, because the pattern does not extract the times from the filenames, each file's extracted datetime in ISO 8601 format is 2021-01-01T:00:00:00UTC, so no files are outside the range and filtered. Make sure to extract as much information about the datetime as necessary.

`source.file_groups.<file_group_name>.start_file`

A string, used as the lower bound to filter for files in a `lexicographic` sort type file group. Inclusive.

|           |                                                                                                                     |
|-----------|---------------------------------------------------------------------------------------------------------------------|
| Type:     | string                                                                                                              |
| Required: | No                                                                                                                  |
| Default:  | none, but <code>allexicographicsort</code> type file group will not check for a lower bound if this is not present. |

`source.file_groups.<file_group_name>.stop_file`

A string, used as the upper bound to filter for files in a `lexicographic` sort type file group. Inclusive.

|           |                                                                                                                      |
|-----------|----------------------------------------------------------------------------------------------------------------------|
| Type:     | string                                                                                                               |
| Required: | No                                                                                                                   |
| Default:  | none, but <code>allexicographicsort</code> type file group will not check for an upper bound if this is not present. |

`source.file_groups.<file_group_name>.compression`

The compression method for the files in this file group. If this value is set, it will override `source.compression`; if not set, it will inherit `source.compression`. See `source_compression` for available options.

|           |        |
|-----------|--------|
| Type:     | string |
| Required: | No     |
| Default:  | null   |

`source.file_groups.<file_group_name>.bucket`

This setting is available only for S3 sources.

If specified, this setting overrides the source-wide bucket for this file group. This allows loading from multiple buckets at once within the same pipeline.

If a `bucket` is specified for every file group, you can omit the `source.bucket`.

|           |        |
|-----------|--------|
| Type:     | string |
| Required: | No     |
| Default:  | null   |

**source.partitions**

The number of partitions for each file group. For example, if there are 2 file groups and **partitions** is set to 32, then a total of 64 partitions will exist, with 32 for each file group. The partition index is 0-based for each file group. In the above example, each file group will have partitions [0, 31].

When **partitions** is specified for a load, its value must not change for the lifetime of this load, otherwise records cannot load correctly.

When using the LAT Client, **partitions** will be automatically assigned by the client. If it is set in the source configuration provided by the user, it will be overwritten.

|           |                       |
|-----------|-----------------------|
| Type:     | int                   |
| Required: | No                    |
| Default:  | Set by the LAT client |

**source.partitions\_assigned**

A 2-element array indicating the minimum and maximum partition indices that this LAT Node owns, inclusive on both ends. For example, [0, 15] means that the node owns Partitions 0, 1, 2, ..., 15. The first element should be greater than or equal to 0. The second element should be greater than or equal to the first element, and less than the number of partitions.

When using the LAT Client, **partitions\_assigned** will be automatically assigned by the client. If it is set in the source configuration provided by the user, it will be overwritten.

|           |                       |
|-----------|-----------------------|
| Type:     | int array             |
| Required: | No                    |
| Default:  | Set by the LAT client |

**source.compression**

The compression method for the files. Currently, only **none** and **gzip** are supported. **none** means that the files are not compressed. Note that this value sets the default compression method for all the files from this source, and it can be overridden by **source\_file\_groups\_compression** per file group.

|           |             |
|-----------|-------------|
| Type:     | string      |
| Required: | No          |
| Default:  | <b>none</b> |

**source.chunk\_size**

The size of a chunk when fetching data. For example, if you have a 40 MB file and the chunk size is 16 MB, you can issue 3 requests sequentially to get the file. The unit is MB. The value must be greater than 0.

The chunk size should be larger than or equal to the maximum record size.

|           |     |
|-----------|-----|
| Type:     | int |
| Required: | No  |
| Default:  | 16  |

**source.buffer\_size**

The total buffer size for *all* assigned partitions. This value will be divided by the total number of assigned partitions to calculate each partition's buffer size. For example, if this value is 4096, **source.partitions** is 16, **source.partitions\_assigned** is [0, 3], and you have 2 file groups, then you have a total of  $(3-0+1)*2=8$  assigned partitions, and each partition will get  $4096/8=512$  MB as the buffer size. The unit is MB. The value must be greater than 0.

The buffer size *per partition* must be at least twice the value of **source.chunk\_size**.

|           |      |
|-----------|------|
| Type:     | int  |
| Required: | No   |
| Default:  | 4096 |

**source.max\_fetch\_concurrency**

The maximum concurrency when fetching files for each partition. Setting this configuration too high can result in thread contention. The value must be greater than 0.

|           |     |
|-----------|-----|
| Type:     | int |
| Required: | No  |
| Default:  | 2   |

**S3 Source-Specific Configuration****source.endpoint**

The endpoint for the S3 instance, usually starting with **http** or **https**. It can be an IP address or a domain. For example, for AWS S3 in the **us-east-2** region, endpoint would be **https://s3.us-east-2.amazonaws.com**. The S3 Source supports Virtual-hosted-style access and Path-style access. The **s3://** protocol based access is not supported. For more details, see [Methods for Accessing an S3 Bucket](#)

|           |        |
|-----------|--------|
| Type:     | string |
| Required: | Yes    |
| Default:  |        |

**source.region**

The region for the S3 instance. This is mostly used for AWS S3 instances.

|           |                  |
|-----------|------------------|
| Type:     | string           |
| Required: | No               |
| Default:  | <b>us-east-2</b> |

**source.bucket**

The bucket from which to get S3 objects.

You can override the bucket on a per-file group basis by setting the `source.file_groups.<file_group_name>.bucket`.

This setting is required unless it is specified for every file group.

|           |        |
|-----------|--------|
| Type:     | string |
| Required: | No     |
| Default:  |        |

#### `source.path_style_access`

Whether to force path style access for the S3 instance. If set to **false**, LAT will try to use the virtual-hosted style (using DNS subdomains), and falls back to the path style access.

AWS S3 is deprecating path style access according to [this post](#).

|           |              |
|-----------|--------------|
| Type:     | boolean      |
| Required: | No           |
| Default:  | <b>false</b> |

#### `source.access_key_id`

The access key for the S3 instance. This should be used together with the `source.secret_access_key` setting. If either of them is absent, LAT will default to the next item in the credentials hierarchy. See [Source Configuration](#).

|           |             |
|-----------|-------------|
| Type:     | string      |
| Required: | No          |
| Default:  | <b>null</b> |

#### `source.secret_access_key`

The access secret for the S3 instance. This should be used together with the `source.access_key_id` setting. If either of them is absent, LAT will default to the next item in the credentials hierarchy. See [Source Configuration](#).

|           |             |
|-----------|-------------|
| Type:     | string      |
| Required: | No          |
| Default:  | <b>null</b> |

#### `source.session_token`

Temporary credentials can be made with a combination with the existing secret ID and secret key, and an additional session token. Note that this is only used when both `source.access_key_id` and `source.secret_access_key` are specified.

|           |             |
|-----------|-------------|
| Type:     | string      |
| Required: | No          |
| Default:  | <b>null</b> |

#### `source.retries`

Configures the number of retries that will be attempted when download from S3 source fails.

|           |                                        |
|-----------|----------------------------------------|
| Type:     | int                                    |
| Required: | No                                     |
| Default:  | Default attempts specified in AWS SDK. |

`source.backoff_strategy_base_delay_seconds`

Configures the base delay for the backoff strategy of the S3 client in seconds.

|           |     |
|-----------|-----|
| Type:     | int |
| Required: | No  |
| Default:  | 1   |

`source.backoff_strategy_max_backoff_seconds`

Configures the maximum backoff delay for the backoff strategy of the S3 client in seconds.

|           |                                                       |
|-----------|-------------------------------------------------------|
| Type:     | int                                                   |
| Required: | No                                                    |
| Default:  | For the default delay, see the AWS SDK documentation. |

`source.max_pending_connection_acquires`

Configures the maximum number of pending acquires allowed by the Netty client.

|           |                                                                    |
|-----------|--------------------------------------------------------------------|
| Type:     | int                                                                |
| Required: | No                                                                 |
| Default:  | For the default number of acquires, see the AWS SDK documentation. |

`source.netty_read_timeout_seconds`

Configures the read timeout, in seconds, of the Netty client. When you set this value to zero, the system disables the read timeout. The LAT configures read timeouts to be tried again by the S3 Client.

|           |                                                               |
|-----------|---------------------------------------------------------------|
| Type:     | int                                                           |
| Required: | No                                                            |
| Default:  | For the default timeout value, see the AWS SDK documentation. |

`source.connection_timeout_seconds`

Configures the amount of time in seconds for the Netty client to wait when initially establishing a connection before giving up and timing out. The LAT configures connection timeouts to be retryable by the S3 Client.

|           |                                                               |
|-----------|---------------------------------------------------------------|
| Type:     | int                                                           |
| Required: | No                                                            |
| Default:  | For the default timeout value, see the AWS SDK documentation. |

**source.connection\_acquisition\_timeout\_seconds**

Configures the amount of time in seconds for the Netty client to wait when acquiring a connection from the pool before giving up and timing out. The LAT configures connection acquisition timeouts to be retryable by the S3 Client.

|           |                                                               |
|-----------|---------------------------------------------------------------|
| Type:     | int                                                           |
| Required: | No                                                            |
| Default:  | For the default timeout value, see the AWS SDK documentation. |

**source.requester\_pays**

Configures whether or not the requester (i.e. the LAT user) should be charged for downloading data from the S3 Requester Pays buckets. This configuration should be set to **true** whenever you request from a Requester Pays bucket, or else the request fails and the bucket owner is charged for the request.

|           |              |
|-----------|--------------|
| Type:     | boolean      |
| Required: | No           |
| Default:  | <b>false</b> |

**S3 Credentials Hierarchy**

The S3 source configuration supports the following hierarchy to obtain S3 credentials. If the LAT does not obtain the credentials at a level, the LAT tries the lower level.

- Level 1: Pipeline configuration
- Level 2: [AWS SDK Default Credential Provider Chain](#)
- Level 3: (Default) Anonymous access

You can choose the level to store the credentials where Level 1 is the highest. Higher levels take precedence for credential storage.

**Load from a Kafka Source**

The Apache® Kafka® Source allows LAT to connect to a Kafka cluster and process records from one or more topics. You can reference each Kafka topic in the [LAT Transform Configuration](#) and can load into one or more tables.

In addition to the LAT specific configuration, this Source supports configuration passthrough to the underlying [Kafka library](#) which this Source uses. There are some library configurations which are not allowed, and will be noted in this table.

**Example Kafka Source Configuration**

```
{
  "source": {
    "type": "kafka",
    "kafka": {
      "bootstrap.servers": "127.0.0.1:9092",
      "auto.offset.reset": "earliest"
    }
  }
}
```

## Kafka Configuration

`source.end_offsets_polling_duration`

Frequency with which to poll for end offsets for lag calculation, in milliseconds.

|           |       |
|-----------|-------|
| Type:     | int   |
| Required: | No    |
| Default:  | 30000 |

`source.kafka`

Kafka consumer configuration object. See [ConsumerConfig](#) for details.

|           |        |
|-----------|--------|
| Type:     | object |
| Required: | Yes    |
| Default:  |        |

`source.kafka`

Required Keys:

- `kafka.bootstrap.servers`
  - This key is required for connection information to the Kafka servers.

Common Keys:

- `kafka.auto.offset.reset`
  - This key is commonly set to `earliest` to consume all of the data in an existing Kafka topic.

## Disallowed Kafka Configuration

The following Kafka consumer configuration options are not allowed:

- `kafka.enable.auto.commit`
  - LAT will always internally set this configuration to `false`; it is critical for correct operation to do so.
- `kafka.key.deserializer`
  - LAT will always internally configure deserialization
- `kafka.value.deserializer`
  - LAT will always internally configure deserialization

If any of the disallowed Kafka library configurations are set, a warning will be logged and their configurations will be ignored.

## Defaulted Kafka Configuration

- `kafka.group.id`
  - If unset. LAT will configure `group.id` internally using the value of `ocient-lat-[$pipeline_id]`. Most users should leave this unset unless they want to explicitly control the `group.id` that is being used.

## Related Links

[Ingest Data with Legacy LAT Reference](#)

## 12.4.2 LAT Sink Configuration

Data Pipelines are now the preferred method for loading data into the Ocident System. For details, see [Load Data](#).

The Sink Configuration controls the destination of data in the LAT Pipeline.



**sink**

A sink configuration object.

Required keys:

- [Sink Configuration](#)

**sink.type**

Type of sink to use in the pipeline.

|                  |        |
|------------------|--------|
| <b>Type:</b>     | string |
| <b>Required:</b> | Yes    |
| <b>Default:</b>  |        |

Allowed values:

- **ocient**: see [Sink Configuration](#) for additional configuration.
- **file**: see [Sink Configuration](#) for additional configuration.

**Ocient Sink**

The Ociant® Sink allows LAT to connect to an Ociant cluster to write rows to one or more tables.

Required keys:

- [Sink Configuration](#)

**sink.remotes**

Array of one or more Ociant Loader Nodes, in `host:port,...` format

|                  |          |
|------------------|----------|
| <b>Type:</b>     | string[] |
| <b>Required:</b> | Yes      |
| <b>Default:</b>  |          |

**sink.batch\_records**

Number of records to buffer per partition before flushing records to Ociant

|                  |      |
|------------------|------|
| <b>Type:</b>     | int  |
| <b>Required:</b> | No   |
| <b>Default:</b>  | 1000 |

**sink.batch\_duration**

Time based flushing parameter, in milliseconds. Records will flush to Ociant after this duration has elapsed with no new activity, even if fewer than `batch_records` records have been processed.

|                  |       |
|------------------|-------|
| <b>Type:</b>     | int   |
| <b>Required:</b> | Yes   |
| <b>Default:</b>  | 30000 |

**sink.idle\_partition\_polling\_period**

Time based polling parameter, in milliseconds. This Sink will periodically poll the remote for progress on write durability for idle partitions.

|                  |       |
|------------------|-------|
| <b>Type:</b>     | int   |
| <b>Required:</b> | No    |
| <b>Default:</b>  | 60000 |

#### `sink.request_timeout`

Request timeout when communicating with Ocient remotes, in milliseconds.

|                  |        |
|------------------|--------|
| <b>Type:</b>     | int    |
| <b>Required:</b> | No     |
| <b>Default:</b>  | 300000 |

#### `sink.request_backoff`

Duration to delay after a failed request to an Ocient remote prior retrying, in milliseconds.

|                  |      |
|------------------|------|
| <b>Type:</b>     | int  |
| <b>Required:</b> | No   |
| <b>Default:</b>  | 1000 |

#### `sink.request_jitter`

Additional duration to delay after a failed request to an Ocient remote prior retrying, in milliseconds. The total delay incurred prior to a given retry is `request_backoff + rand(0, request_jitter)`.

|                  |      |
|------------------|------|
| <b>Type:</b>     | int  |
| <b>Required:</b> | No   |
| <b>Default:</b>  | 5000 |

#### `sink.high_watermark`

High watermark memory point, in bytes. The LAT will stop pushing new rows to memory buffers. It will not resume pushing rows into the memory buffers until `low_watermark` is reached.

|                  |            |
|------------------|------------|
| <b>Type:</b>     | int        |
| <b>Required:</b> | No         |
| <b>Default:</b>  | 1000000000 |

#### `sink.low_watermark`

Low watermark memory point, in bytes. After reaching `high_watermark`, the LAT will begin pushing rows to memory buffers again when this memory level is reached.

|                  |           |
|------------------|-----------|
| <b>Type:</b>     | int       |
| <b>Required:</b> | No        |
| <b>Default:</b>  | 500000000 |

**sink.storage\_scope\_id**

UUID of the storage scope that rows will be associated with. The scope with the given UUID must already exist in the target cluster.

|                  |        |
|------------------|--------|
| <b>Type:</b>     | string |
| <b>Required:</b> | No     |
| <b>Default:</b>  | null   |

**sink.skip\_page\_replication**

A Boolean value to determine whether to omit page replicas for the specified storage scope. This is ignored if **sink.storage\_scope\_id** is not specified or has already been seen by the remotes.

|                  |         |
|------------------|---------|
| <b>Type:</b>     | boolean |
| <b>Required:</b> | No      |
| <b>Default:</b>  | false   |

**sink.netty\_event\_loop\_group\_threads**

The number of threads in the Netty event loop group used to communicate with remotes.

|                  |     |
|------------------|-----|
| <b>Type:</b>     | int |
| <b>Required:</b> | No  |
| <b>Default:</b>  | 1   |

## Example Ociant Sink Configuration

```
{
  "sink": {
    "type": "ocient",
    "remotes": ["loader0:5050", "loader1:5050", "loader2:5050"]
  }
}
```

**File Sink**

A Sink type for **testing** LAT pipelines that writes the transformed data to local [JSONL](#) files.

Required keys:

- **Sink Configuration**

**sink.location**

An absolute or relative path to the location that the sink should write files to.

|                  |        |
|------------------|--------|
| <b>Type:</b>     | String |
| <b>Required:</b> | Yes    |
| <b>Default:</b>  |        |

## Example File Sink Configuration

```
{
  "sink": {
    "type": "file",
    "location": "out/"
  }
}
```

### External Sink Configuration

Rather than including a sink directly within the pipeline, it is also possible to configure a pipeline to use a sink that is specified externally. Sinks can be managed (created, deleted, and more) using the [LAT Client Command Line Interface](#). A sink must exist before a pipeline can use it.

There are three ways to configure a pipeline to use a sink.

1. If a sink is included directly within a pipeline (using the [LAT Sink Configuration](#)), it will be used.
2. If a sink is not specified within the pipeline, you can specify a [LAT Pipeline Configuration](#) that corresponds to a sink previously created using the LAT Client.
3. If neither `sink` nor `sink_name` is specified in a pipeline, the default sink will be used. If a default sink has not been created using the LAT Client, a pipeline must specify either a `sink` or a `sink_name`.

### Related Links

[Ingest Data with Legacy LAT Reference](#)

#### 12.4.3 LAT Extract Configuration

Data Pipelines are now the preferred method for loading data into the Ocient System. For details, see [Load Data](#).

An extract configuration object. This object defines the way that data is extracted from the source records. The extract definition is applied to a specific topic or file group as defined in the source. Keys are file group names or topic names and values are [Extract Configuration](#).

For file based loads, this should match a `file_group_name` defined in the `source.file_groups` section. For Apache® Kafka® based loads this should match a `topic` defined in Kafka.

Required keys:

- None - Extract defaults to JSON for all topics or file\_groups unless provided

### Example Extract

In the following example, two file groups are defined. The file groups defined in the source section are referenced in the extract section to define how data is extracted from the file group. For Kafka based loads, the topic names are used in the extract section to define extraction if required.

```
{
  ...,
  "source": {
    "type": "s3",
    "endpoint": "http://some.endpoint/",
    "bucket": "my_bucket",
    "file_groups": {
      "delimited_user_data": {
        "prefix": "crm/users/"
      }
    }
  }
}
```

```

        "file_matcher_syntax": "glob",
        "file_matcher_pattern": "**.csv",
        "sort_type": "lexicographic"
    },
    "delimited_order_data": {
        "prefix": "crm/orders/",
        "file_matcher_syntax": "glob",
        "file_matcher_pattern": "**.json",
        "sort_type": "lexicographic"
    }
},
"extract": {
    "delimited_user_data": {
        "record_type": "delimited",
        "headers": ["id", "username", "last_login_at", null, "first_name", "last_name"],
        "field_delimiter": "|",
        "trim_whitespace": true,
        "skip_blank_lines": true,
        "field_optionally_enclosed_by": "\"",
        "null_strings": ["NULL", "N/A"]
    },
    "delimited_order_data": {
        "record_type": "delimited",
        "num_header_lines": 1,
        "exclude_columns": ["user_id", "order_total"],
        "field_delimiter": ","
    }
},
"transform": {
    ...
}
}

```

## Extract Settings

`extract.<file_group_name>.record_type`

Similar to Source and Sink config, `record_type` instructs the Pipeline to instantiate a specific extraction implementation for a given type of source record such as Delimited Records (e.g., CSV, TSV), JSON Records, or Fixed Width Binary Records.

|           |        |
|-----------|--------|
| Type:     | string |
| Required: | No     |
| Default:  | json   |

Allowed values:

- `json`: see [Extract Configuration](#) for additional configuration.
- `delimited`: see [Extract Configuration](#) for additional configuration.
- `fixed_width_binary`: see [Extract Configuration](#) for additional configuration.

## Shared Extract Configuration

`extract.<file_group_name>.null_strings`

**null\_strings** instructs the extraction process to convert a string literal such as *NA* to a **null** value in the transformation layer. This allows different null string values to be provided. **null\_strings** must be an array of strings. If **trim\_whitespace** is set to true and a string value is trimmed to become a **null\_string**, that value converts to **null**.

|           |          |
|-----------|----------|
| Type:     | string[] |
| Required: | No       |
| Default:  |          |

**extract.<file\_group\_name>.empty\_as\_null**

The **empty\_as\_null** setting specifies whether empty fields should be treated as null. An empty field is defined as a string field containing no characters (its length is 0). When set to false, empty fields will be treated literally as empty strings. When set to true, empty fields will be treated as **null** in the transformation and loading process. If **trim\_whitespace** is set to true and **empty\_as\_null** is set to true, a field that consists of only whitespace will also be converted to **null**.

|           |                                                                             |
|-----------|-----------------------------------------------------------------------------|
| Type:     | boolean                                                                     |
| Required: | No                                                                          |
| Default:  | True for the Delimited and FWB extract types, False for a JSON extract type |

**extract.<file\_group\_name>.trim\_whitespace**

The **trim\_whitespace** setting specifies whether whitespace should be trimmed from the beginning and end of each string field as it is extracted.

|           |         |
|-----------|---------|
| Type:     | boolean |
| Required: | No      |
| Default:  | False   |

**extract.<file\_group\_name>.encoding**

The **encoding** setting specifies the file encoding to use. Encodings should use standard Java® encodings defined in the Java Internationalization Guide.

|           |        |
|-----------|--------|
| Type:     | string |
| Required: | No     |
| Default:  | UTF8   |

**extract.<file\_group\_name>.replace\_invalid\_characters**

**replace\_invalid\_characters** defines whether invalid characters based on the encoding type should be replaced with a replacement character (i.e. U+FFFD). If this setting is false and invalid characters are encountered, the record will not be loaded and a record extraction error will be logged.

|           |         |
|-----------|---------|
| Type:     | boolean |
| Required: | No      |
| Default:  | False   |

## JSON Extract

The JSON Extract type allows LAT to extract JSON data from the source records. There are no additional configuration settings available for JSON beyond the common Extract settings.

### Example JSON Extract Configuration

In this example JSON extract configuration, the `replace_invalid_characters` setting is used to convert any invalid UTF-8 characters with the UTF-8 replacement character. The file group name assigned to this file group was defined as “`json_user_data`” in the `source` section of the pipelines. When transforming this data, you can reference columns by the JSON object keys.

```
{
  "extract": {
    "json_user_data" : {
      "record_type": "json",
      "replace_invalid_characters": true
    }
  },
  ...
}
```

## Delimited Extract

The Delimited Extract type allows the LAT to extract delimited data from the source records. The delimiter character is configurable to support CSV, TSV, or other delimited formats like pipe and semicolon. Fixed width files are a separate extraction type.

### Notes on Delimited Files

- Delimited files can have additional comment rows that are prefixed by a `#` character. These lines are not processed and will be skipped during loading.
- A maximum number of 1024 columns can be loaded in a single delimited file.
- You can enclose fields with an optional enclosure character. In these cases, the enclosure character is not retained in the loaded data.
- You can skip a configurable number of header lines during loading to prevent column headers from appearing as loaded data.

The system parses headers from the beginning of each file if you configure [Extract Configuration](#) and do not configure [Extract Configuration](#).

The system automatically names columns if you do not configure [Extract Configuration](#) and [Extract Configuration](#). The columns are named “`$0`”, “`$1`”, and so on consecutively instead.

- You can define multiple file groups. This allows different delimited extraction settings on each file group.
- The default record delimiter is a `\n` newline character. If source data differs, this should be overridden.

Records with improperly escaped “quote characters” or malformed “quoting” of column data will not parse. See the [Extract Configuration](#) setting for more details.

- Control characters such as the NULL byte `0x0` are automatically processed as whitespace when parsing delimited records.

### Delimited Configuration

`extract.<file_group_name>.field_delimiter`

The `field_delimiter` specifies the string that should function as a column delimiter. The `field_delimiter` character applies to any unquoted and unescaped text when extracting records from the source file. This can be a single character or a multi-character string delimiter. Only one `field_delimiter` string is allowed per extract.

|           |        |
|-----------|--------|
| Type:     | string |
| Required: | No     |
| Default:  | ,      |

`extract.<file_group_name>.record_delimiter`

The `record_delimiter` specifies the string that indicates the end of a record. The `record_delimiter` character applies to any unquoted and unescaped text when extracting records from the source file. This can be a single character or a two-character string delimiter. Only one `record_delimiter` string is allowed per extract.

The default is a standard newline character `\n`. Depending on the file, typical settings for this delimiter are a newline `\n`, a carriage return `\r` or a carriage return and newline `\r\n`.

|           |                              |
|-----------|------------------------------|
| Type:     | string; maximum 2 characters |
| Required: | No                           |
| Default:  | <code>\n</code>              |

`extract.<file_group_name>.field_optionally_enclosed_by`

Also referred to as the “quote character,” the `field_optionally_enclosed_by` settings specifies the character that is optionally used to enclose a field or column. The data between enclosure characters can contain delimiter characters. The enclosure character itself must be escaped inside of an enclosed field. Common values are `"` to enclose fields by double quotes. Because this setting defines an optional enclosure, not all fields in the record need to be enclosed. The `escape` character used to escape the enclosure character is separately configured and defaults to `"` per the RFC-4180 standard.

|           |                             |
|-----------|-----------------------------|
| Type:     | string; maximum 1 character |
| Required: | No                          |
| Default:  | <code>"</code>              |

### Example:

In the following pipe delimited record, the double quote character can be specified so that the final column will be `this field has a | in it` when loaded into the database. If no enclosure character is specified, this record would yield four columns instead of three.

```
a|b|"this field has a | in it"
```

`extract.<file_group_name>.escape`

The `escape` settings specifies the character that is used as an escape occurrences of the enclosing character used to value so that the following character is escaped. This is typically used in enclosed columns to escape the enclosure character defined as `field_optionally_enclosed_by`.

|           |                             |
|-----------|-----------------------------|
| Type:     | string; maximum 1 character |
| Required: | No                          |
| Default:  | <code>"</code>              |

`extract.<file_group_name>.num_header_lines`

The `num_header_lines` setting defines the number of header lines in the source. The system reads these header lines but does not load them as data.



If you do not specify **Extract Configuration**, the system uses the final line within this section to name the columns in each file. Then, the system uses these names in the Transform section to refer to column values. The system names columns for each file, so the names available in the Transform section might change if the file header changes.

This setting only applies to file-based source types.

|           |     |
|-----------|-----|
| Type:     | int |
| Required: | No  |
| Default:  | 0   |

#### `extract.<file_group_name>.headers`

The **headers** setting defines the header labels associated with each column in a delimited file. This array of values is associated with the columns in order from left to right. The strings assigned to the column names are used in the Transform section to refer to column values. Any explicitly supplied headers that are an empty string "" or a null literal will be skipped during extraction. This can have performance benefits if all columns are not needed.

|           |          |
|-----------|----------|
| Type:     | string[] |
| Required: | No       |
| Default:  | null     |

#### `extract.<file_group_name>.include_columns`

The **include\_columns** setting restricts extraction to the columns with the specified names. If you do not need all columns during transformation, you might improve performance when you use the **include\_columns** setting.

If there are no headers and you specify this setting, the system names columns based on their original positions before other columns are filtered out. For example, if there are four columns and \$0 and \$3 are included, the system names these columns \$0 and \$3 during transformation, not \$0 and \$1.

If you specify the **exclude\_columns** setting, you cannot specify this setting.

|           |          |
|-----------|----------|
| Type:     | string[] |
| Required: | No       |
| Default:  | null     |

#### `extract.<file_group_name>.exclude_columns`

The **exclude\_columns** setting causes the LAT to not extract any of the specified columns. If you do not need all columns during transformation, you might improve performance when you use the **exclude\_columns** setting.

If there are no headers and you specify this setting, the system names columns based on their original positions before other columns are filtered out. For example, if there are four columns and \$1 and \$2 are excluded, the system names these columns \$0 and \$3 during transformation, not \$0 and \$1.

If you specify the **include\_columns** setting, you cannot specify this setting.

|           |          |
|-----------|----------|
| Type:     | string[] |
| Required: | No       |
| Default:  | null     |

#### `extract.<file_group_name>.skip_blank_lines`

The `skip_blank_lines` setting specifies whether lines with only whitespace should be skipped or processed.

|           |         |
|-----------|---------|
| Type:     | boolean |
| Required: | No      |
| Default:  | False   |

### Example Delimited Extract Configuration

In this example extract configuration, a pipe delimited file type with five column names is provided. The file group name assigned to this file group was defined as “`delimited_user_data`” in the `source` section of the pipelines. Whitespace will be trimmed from each field and blank lines will be skipped instead of reporting errors. Fields with special characters can be escaped by `"`. Any literal values of `NULL` or `N/A` are converted to `null`. When transforming this data, columns can be referred to by `id`, `username`, `last_login_at`, `first_name`, and `last_name`. The fourth column is listed as `null` in the header, so it will be skipped during extraction. Because `num_header_lines` is not specified, the files are assumed to have no headers.

```
{
  "extract": {
    "delimited_user_data": {
      "record_type": "delimited",
      "headers": ["id", "username", "last_login_at", null, "first_name", "last_name"],
      "field_delimiter": "|",
      "trim_whitespace": true,
      "skip_blank_lines": true,
      "include_columns": ["id", "username", "last_name"],
      "field_optionally_enclosed_by": "\"",
      "null_strings": ["NULL", "N/A"]
    },
    ...
  },
  ...
}
```

### Fixed Width Binary Extract

The Fixed Width Binary (FWB) Extract type allows the LAT to extract data from binary source records that have a fixed number of bytes per column. FWB source records typically come from sources like mainframe systems like the IBM® Z® series that use Cobol Copybooks to produce character, integer, floating point, and decimal data in various byte formats. The FWB Extract type requires users to specify the byte offset, width in bytes, and the type of extraction to use on the indicated bytes for each column. The FWB type also allows users to specify a record length in bytes rather than a character record delimiter. FWB files are typically extracted using `encodings` such as `cp500` and `cp037`.

#### Fixed Width Binary Data Types

The following data types are supported when extracting Fixed Width Binary data. All data types require specification of `offset` and `width` to locate the bytes to extract in each record.

| Data Type | Description                                                                                                                                                  | Supported Settings                |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| string    | Represents bytes of character data following the encoding defined in the extract settings. Bytes are extracted into a string for transformation and loading. | <code>encoding</code> - inherited |

| Data Type        | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Supported Settings                                                                                                                                                                                                                                                                                 |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| integer          | Represents a signed integer. Uses the <b>endianness</b> setting to determine how bytes should be interpreted. The <b>integer</b> type can be used for 1, 2, 4, or 8 bytes of data. These correspond to mainframe concepts of a Byte, Half-word, Full-word, and Double-word.                                                                                                                                                                                                      | <b>endianness</b> - inherited                                                                                                                                                                                                                                                                      |
| unsigned_integer | Represents a unsigned integer. Uses the <b>endianness</b> setting to determine how bytes should be interpreted. Can be used for 1, 2, 4, or 8 bytes of data. The largest integer data type in the {{Ocint}} System is a signed BIGINT, so the maximum value for an <b>unsigned_integer</b> is $2^{63}-1$ .                                                                                                                                                                       | <b>endianness</b> - inherited                                                                                                                                                                                                                                                                      |
| float            | Represents a floating point number as defined by the {{IEEE}}-754 floating point specification. Can be used for 4 or 8 bytes of data. 4 bytes represents a single precision floating point number, and 8 bytes represents a double precision floating point number.                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                    |
| packed_decimal   | A Packed Binary Coded Decimal format for decimal precision that packs two integral values in each byte and encodes the sign in the final byte. The number of decimal points represented by the data is supplied with a separate extraction setting. The sign is encoded in the 4 least significant bits of the final byte. A value of 0xD or 0xB indicates a negative sign. Any other value including the common values of 0xF, 0xE, 0xC or 0xA indicating positive or unsigned. | <b>endianness</b> - inherited<br><b>decimals</b> - number of digits to place the right of the decimal point<br><b>sign_format</b> - defaults to ‘trailing’ byte format indicating that the sign is encoded in the most significant four bits of the final byte; provided for future extensibility. |
| zoned_decimal    | A Zoned Binary Coded Decimal format for decimal precision that packs two integral values in each byte and encodes the sign in the final byte. The number of decimal points represented by the data is supplied with a separate extraction setting. The sign is encoded in the 4 most significant bits of the final byte. A value of 0xD, 0xB, 0x7 indicates a negative sign. Values of 0xF, 0xE, 0xC or 0xA indicate positive or unsigned.                                       | <b>endianness</b> - inherited<br><b>decimals</b> - number of digits to place the right of the decimal point<br><b>sign_format</b> - defaults to ‘trailing’ byte format indicating that the sign is encoded in the most significant four bits of the final byte; provided for future extensibility. |

## Fixed Width Binary Configuration

`extract.<file_group_name>.record_width`

The `record_width` setting specifies the total number of bytes in each record. Files are read serially, and the next record is assumed to start on the following byte. The `record_width` can be greater than the number of bytes used in extracting the individual columns.

|           |         |
|-----------|---------|
| Type:     | integer |
| Required: | Yes     |
| Default:  |         |

`extract.<file_group_name>.padding_character`

The `padding_character` setting specifies a character in a fixed width file type that should be removed from extracted character data. After extracting a character field, any leading or trailing occurrences of this character will be stripped from the ends of the character data. Typical examples are the space character or an underscore.

|           |                                 |
|-----------|---------------------------------|
| Type:     | string                          |
| Required: | No                              |
| Default:  | " " (in the specified encoding) |

`extract.<file_group_name>.endianness`

The `endianness` setting specifies whether the extraction of numeric fields should use Big Endian or Little Endian byte ordering to process extracted bytes. `endianness` impacts integer, packed\_decimal, and zoned\_decimal column types.

|           |        |
|-----------|--------|
| Type:     | string |
| Required: | No     |
| Default:  | big    |

Allowed values:

- `big`: Big Endian
- `little`: Little Endian

`extract.<file_group_name>.columns`

A `extract columns` configuration object. A collection of columns and their associated extraction configuration. Keys are column names and values are [Extract Configuration](#).

Required keys:

- [Extract Configuration](#)

## Column Extracts

Column extract configurations for Fixed Width Binary records.

`extract.<file_group_name>.columns.<column>`

A column extract configuration object keyed by a column name. A column's value is defined as an extract definition. The extract definition will indicate the offset and width to locate the bytes to extract, the data type to use in the extraction, and any additional parameters about the bytes to extract.

Required keys:

- `offset`

- width
- data\_type

Optional keys:

- decimals
- sign\_format

`extract.<file_group_name>.columns.<column>.offset`

Number of bytes to skip from the beginning of the record before beginning to extract bytes for this column. Note that `offset` is relative to the beginning of the record, so arbitrary offsets can be used including overlapping sets of bytes with other columns.

|           |         |
|-----------|---------|
| Type:     | integer |
| Required: | Yes     |
| Default:  |         |

`extract.<file_group_name>.columns.<column>.width`

Number of bytes to extract from the `offset` location.

|           |         |
|-----------|---------|
| Type:     | integer |
| Required: | Yes     |
| Default:  |         |

`extract.<file_group_name>.columns.<column>.data_type`

Data type to extract from the bytes that are specified by `offset` and `width`. See [Extract Configuration](#) for reference on available data types.

|           |        |
|-----------|--------|
| Type:     | string |
| Required: | Yes    |
| Default:  | string |

Allowed values:

- string
- integer
- unsigned\_integer
- float
- packed\_decimal
- zoned\_decimal

Note that each `data_type` has specific settings that are allowed to control the extraction to that `data_type`. Some of these are inherited from the overall extract configuration, while others are defined on each column.

`extract.<file_group_name>.columns.<column>.decimals`

The number of numeric values that are placed to the right of the decimal point in a `zoned_decimal` or `packed_decimal` data type. For example, if `decimals` is set to 2` and the data in a `zoned_decimal` represents the numerals 12345, this would create the decimal 123.45` in the extracted value.

`decimals` configures the **scale** of the extracted decimal and the total number of digits present in the source data determines the precision. If insufficient scale or precision exists in the target Ocint column, data will fail to load.

|           |         |
|-----------|---------|
| Type:     | integer |
| Required: | No      |
| Default:  | 0       |

Applies to:

- `packed_decimal`
- `zoned_decimal`

`extract.<file_group_name>.columns.<column>.sign_format`

In decimal types, the sign for the number can be encoded in different ways. Currently, only the **trailing** format is supported which requires the sign to be encoded in 4 bits of the last byte in the column data. This setting is provided for future extensibility. The values that indicate a positive or negative sign are listed in [Extract Configuration](#).

|           |          |
|-----------|----------|
| Type:     | string   |
| Required: | No       |
| Default:  | trailing |

Allowed values:

- **trailing** - the sign is encoded in the most significant 4 bits of the last byte in the binary coded decimal.

Applies to:

- `packed_decimal`
- `zoned_decimal`

### Fixed Width Binary Extract Example

The following example illustrates extracting fixed width binary data from a file group that was given the name “order\_files” in the `source` section of the pipeline. The file has 3100 bytes per record and is only extracting a few columns.

```
{
  "extract": {
    "order_files": {
      "record_type": "fixed_width_binary",
      "record_width": 3100,
      "endianness": "big",
      "columns": {
        "order_id": {
          "offset": 0,
          "width": 4,
          "data_type": "integer"
        },
        "username": {
          "offset": 4,
          "width": 20,
          "data_type": "string"
        },
        "order_total": {
          "offset": 24,
          "width": 10,
          "data_type": "packed_decimal",
          "decimals": 2
        }
      }
    }
  }
}
```

```

    },
    "sales_tax_total": {
      "offset": 34,
      "width": 8,
      "data_type": "zoned_decimal",
      "decimals": 3
    }
  }
}
}
}
}

```

## Related Links

[Ingest Data with Legacy LAT Reference](#)

### 12.4.4 LAT Transform Configuration

Data Pipelines are now the preferred method for loading data into the Ocident System. For details, see [Load Data](#).

#### **transform**

A transform configuration object.

Required keys:

- Transform Configuration source only: [Transform Configuration](#)
- File sources only: [Transform Configuration](#)

#### **transform.topics**

A collection of topics and their associated configuration. Keys are topic names and values are [Transform Configuration](#). Each <topic> set as a key in this object represents an Apache® Kafka® topic defined in Kafka.

When loading from an s3 or local File Source type, `transform.file_groups` should be used instead of `transform.topics`. Each key in the `file_group` must match a `file_group_name` defined in the Source section of the pipeline configuration.

|                  |        |
|------------------|--------|
| <b>Type:</b>     | object |
| <b>Required:</b> | Yes    |
| <b>Default:</b>  |        |

#### **transform.database**

A database to be used to fully qualify any table names that are not fully qualified.

For example, if `transform.database` is set to `myDatabase`, a table name of the form `schema.table` will become `myDatabase.schema.table`.

|                  |        |
|------------------|--------|
| <b>Type:</b>     | string |
| <b>Required:</b> | No     |
| <b>Default:</b>  | null   |

**transform.schema**

A schema to be used to fully qualify any table names that are not fully qualified. Specifying this property requires that **Transform Configuration** is specified.

For example, if **transform.database** is set to **myDatabase** and **transform.schema** is set to **mySchema**:

- A table name of the form **table** will become **myDatabase.mySchema.table**
- A table name of the form **schema.table** will become **myDatabase.schema.table**
- A table name of the form **database.schema.table** will stay as **database.schema.table**

|                  |        |
|------------------|--------|
| <b>Type:</b>     | string |
| <b>Required:</b> | No     |
| <b>Default:</b>  | null   |

## Kafka Load Transform Example

```
{
  ...
  "transform": {
    "topics": {
      "topic_1": {
        "tables": {
          ...
        }
      },
      "topic_2": {
        "tables": {
          ...
        }
      }
    }
  }
}
```

## File Based Load Transform Example

Unlike Kafka, File loads define file groups in the source section of the pipeline configuration. The “file\_groups” defined in the source and transform sections must match.

```
{
  "source": {
    "type": "s3",
    ...
    "file_groups": {
      "file_group_1": {
        ...
      },
      "file_group_2": {
        ...
      },
    }
  }
  "transform": {
    "file_groups": {
      "file_group_1": {
        "tables": {

```



```
    ...  
  }  
},  
"file_group_2": {  
  "tables": {  
    ...  
  }  
}  
}  
}
```

Topics

Topic configuration objects.

Required keys:

- Transform Configuration

For file based loads, `topics` are replaced by `file_groups`, but all other settings are equivalent.

`transform.topics.<topic>.filter`

A record filter to apply at the topic level. See [LAT Record Filtering](#) for details.

|           |        |
|-----------|--------|
| Type:     | string |
| Required: | Yes    |
| Default:  |        |

`transform.topics.<topic>.tables`

A collection of tables and their associated configuration. Keys are table names and values are [Transform Configuration](#).

|           |        |
|-----------|--------|
| Type:     | object |
| Required: | Yes    |
| Default:  |        |

Tables

Table configuration objects.

Required keys:

- Transform Configuration

|                                        |           |        |          |
|----------------------------------------|-----------|--------|----------|
| <code>transform.topics..tables.</code> | Type:     | string |          |
| <code> — — </code>                     | Required: | Yes    | Default: |
| <code>transform.topics..tables.</code> | Type:     | string |          |
| <code> — — </code>                     | Required: | Yes    | Default: |

Complex Transform Example

```

{
  "transform": {
    "database": "adtechdb",
    "schema": "adtech",
    "topics" : {
      "dsp.Auctions" : {
        "tables" : {
          "auction" : {
            "columns" : {
              "auctionid" : "auctionId",
              "created" : "created",
              "bidder_nodeid" : "bidder.nodeId",
              "deals_dealid" : "deals[].dealId"
            }
          },
          "trafficSource" : {
            "columns" : {
              "auctionid" : "auctionId",
              "created" : "created",
              "sourcesellerid" : "trafficSource.sourceSellerId",
              "domainname" : "trafficSource.domainName",
              "contextualdataset_externalentryid" : "trafficSource.siteContextualProfile.com",
              "dimensions_hw" : "join('x', [dimensions.height, dimensions.width])"
            }
          },
          "feedback" : {
            "columns" : {
              "auctionid" : "auctionId",
              "created" : "created",
              "received" : "received",
              "campaignid" : "EXPLODE(auctionResponseFeedbacks[].campaignId)",
              "won" : "EXPLODE(auctionResponseFeedbacks[].won)",
              "winningbid" : "EXPLODE(auctionResponseFeedbacks[].winningBid)",
              "sourcelossreasoncode" : "EXPLODE(auctionResponseFeedbacks[].sourceLossReasonC"
            }
          }
        }
      }
    }
  }
}

```

## Related Links

[Ingest Data with Legacy LAT Reference](#)

## 12.5 LAT Data Transformation with JMESPath

Data Pipelines are now the preferred method for loading data into the Ocident System. For details, see [Load Data](#).

Within a transformation each column is configured with a JMESPath expression describing where to extract the column's value and any transformations to apply to that value.

For information on common JMESPath functions and filtering tools, see the [LAT Transformation Functions](#) section.

For a list of functions to modify data during loading, see the [LAT Transformation Functions](#) page.

Information on customizing your own transformation functions can be found on the [LAT User-Defined Transformations](#) page.

For expressions to select elements from an array matching certain criteria, see the [LAT Record Filtering](#) page.

## Notes/Warnings

### JSON Keys with Numbers and Special Characters

You must escape JSON keys that begin with a number or special character in the pipeline configuration.

- For a JSON record: `{ "0_123" : "value" }` transform this record as follows to write “value” to “my\_col” — `{ "my_col" : "\"0_123\"" }`
- For a JSON record: `{ "a-col" : 123 }` transform this record as follows to write “value” to “my\_col” — `{ "my_col" : "\"a-col\"" }`

### Case Sensitivity

- The Ocient® System normalizes database objects (e.g., table names, column names, schemas) in LAT Transformations according to the same rules as the database unless explicitly escaped with quotes.
  - The Ocient System normalizes all names to lowercase unless explicitly escaped. For example, the transformation: `{ "myCol" : "value" }` normalizes to `mycol` and matches a database column `mycol` that was not escaped at creation time.
  - If you create the database column as an escaped literal such as `"myCol"` then the LAT transformation must be specified as: `{ "\"myCol\"" : "value" }` to correctly map to the column.
- Transformation functions are case sensitive, so represent them by using all lower case letters (e.g., `st_geomfromewkb(my_column)`).

### Literals

- You must escape literal values with a backtick. This applies to numbers, booleans, and empty arrays (otherwise the Ocient System interprets the value as a JSON transformation operator).

Right:

```
{
  "column_1": "`0`",
  "column_2": "`1`",
  "column_3": "`true`",
  "column_4": "`[]`"
}
```

Wrong:

```
{
  "column_1": "0",
  "column_2": "1",
  "column_3": "true",
  "column_4": "[]"
}
```

- You can write string literals with a single quote.

```
{ "column_1": "'example_value'" }
```

### Loading Numbers in Exponential Notation

When you need to load numbers from exponential notation (e.g. `1.20E-05`), you might require a conversion to load them into the target Ocient column or to perform transformations on the data. Use the `to_number` transformation

to load this into a column of type `decimal` or to convert to a numeric value for further transformations like multiplication.

#### Example:

A source record in a CSV file might have a number like 7.00E-05 that appears as a string in the field named `source_field`.

```
{ "column_1": "to_number(source_field)" }
```

This JSON code converts the string in exponential notation into the double 0.00007 that is now suitable for loading into decimal, double, or float columns, or for use in mathematical transformation functions.

#### Flatten, Array Access, and Pipes

For general syntax help for arrays, projections, and pipe expressions, see [JMESPath website](#). Flattening arrays, accessing entire arrays, and using pipe expressions are three important operations when you work with nested data in LAT transformations.

- The flatten operator (e.g., `[]`) flattens an array one level. This also removes NULL values from the array.

This can cause unintended consequences if multiple arrays need to remain aligned for data processing.

- When you work with nested arrays, you can maintain array hierarchy. For example, you can replace an array index such as `*` with `[*]` to maintain the hierarchy.
- You can use the pipe operator (e.g., `a.b[*] | [0]`) to stop the projection of an expression into the prior expression.

#### Related Links

[Ingest Data with Legacy LAT Reference](#)

#### 12.5.1 LAT Transformation Functions

Data Pipelines are now the preferred method for loading data into the Ocient System. For details, see [Load Data](#).

LAT supports many transformation functions to modify data during loading. The functions include standard JMESPath Functions and Ocient@-specific transformation functions.

Transformation functions in LAT are case sensitive. The standard syntax for transformation function calls is all lower case characters.

#### Standard JMESPath Functions

The following standard JMESPath Functions are supported in LAT Transformation expressions. For details about using these functions, see [JMESPath Functions](#).

| JMESPath Function     | Function Signature                                                      | Example                                |
|-----------------------|-------------------------------------------------------------------------|----------------------------------------|
| <code>abs</code>      | number <code>abs(number \$value)</code>                                 | <code>abs(num_1)</code>                |
| <code>avg</code>      | number <code>avg(array[number]<br/>\$elements)</code>                   | <code>avg(array_1)</code>              |
| <code>contains</code> | boolean <code>contains(array string<br/>\$subject, any \$search)</code> | <code>contains(array_1, 'test')</code> |
| <code>ceil</code>     | number <code>ceil(number \$value)</code>                                | <code>ceil(num_1)</code>               |

| JMESPath Function | Function Signature                                                | Example                       |
|-------------------|-------------------------------------------------------------------|-------------------------------|
| ends_with         | boolean ends_with(string \$subject, string \$prefix)              | ends_with(str_1, 'ing')       |
| floor             | number floor(number \$value)                                      | floor(num_1)                  |
| join              | string join(string \$glue, array[string] \$stringsarray)          | join(',', array_1)            |
| keys              | array keys(object \$obj)                                          | keys(obj_1)                   |
| length            | number length(string array object \$subject)                      | length(array_1)               |
| map               | array[any]<br>map(expression→any→any expr, array[any] elements)   | map(&to_string(@), array_1)   |
| max               | number<br>max(array[number] array[string] \$collection)           | max(array_1)                  |
| max_by            | max_by(array elements, expression→number expression→string expr)  | max_by(array_1, &age)         |
| merge             | object merge([object *argument, [object \$. ..]])                 | merge(obj_1, obj_2, obj_3)    |
| min               | number<br>min(array[number] array[string] \$collection)           | min(array_1)                  |
| min_by            | min_by(array elements, expression→number expression→string expr)  | min_by(array_1, &age)         |
| not_null          | any not_null([any \$argument [, any \$. ..]])                     | not_null(val_1, val_2, val_3) |
| reverse           | array reverse(string array \$argument)                            | reverse(str_1)                |
| sort              | array<br>sort(array[number] array[string] \$list)                 | sort(array_1)                 |
| sort_by           | sort_by(array elements, expression→number expression→string expr) | sort_by(array_1, &age)        |
| starts_with       | boolean starts_with(string \$subject, string \$prefix)            | starts_with(str_1, 'chi')     |
| sum               | number sum(array[number] \$collection)                            | sum(array_1)                  |
| to_array          | array to_array(any \$arg)                                         | to_array(val_1)               |
| to_string         | string to_string(any \$arg)                                       | to_string(val_1)              |
| type              | string<br>type(array object string number boolean null \$subject) | type(val_1)                   |
| values            | array values(object \$obj)                                        | values(obj_1)                 |

### Other Transformation Functions

The following transformation functions are also available in LAT transformation expressions.

#### add

Returns the sum of the two arguments.

|                   |                                       |
|-------------------|---------------------------------------|
| <b>Signature:</b> | number add(number \$a, number \$b)    |
| <b>Arguments:</b> | -a: first argument-b: second argument |
| <b>Example:</b>   | add(int_col_0, int_col_1)             |

**array\_cap**

Caps the length of an array to a maximum number of elements.

|                   |                                                                                 |
|-------------------|---------------------------------------------------------------------------------|
| <b>Signature:</b> | array array_cap(Array[any] \$array_1, number \$max_elements)                    |
| <b>Arguments:</b> | -array_1: the array to cap-max_elements: number of elements to cap the array at |
| <b>Example:</b>   | array_cap(array_1,1000)                                                         |

**array\_cat**

Concatenate two arrays, returning an array. Accepts an array or null value, returning an empty array if both parameters are null.

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| <b>Signature:</b> | array array_cat(array[any] \$array_1, array[any] \$array_2)       |
| <b>Arguments:</b> | -array_1: first array-array_2: second array concatenated to first |
| <b>Example:</b>   | array_cat(array_1, array_2)                                       |

**concat**

Concatenates two strings or converts arguments into strings and concatenates them into a string.

|                   |                                                                                                                                                                             |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | string concat(any \$a, any \$b)                                                                                                                                             |
| <b>Arguments:</b> | -a: first argument (automatically converted to string when not specified as a string)-b: second argument (automatically converted to string when not specified as a string) |
| <b>Example:</b>   | concat("ocient",10)                                                                                                                                                         |

**divide**

Returns the floating-point division of the two arguments.

|                   |                                       |
|-------------------|---------------------------------------|
| <b>Signature:</b> | number divide(number \$a, number \$b) |
| <b>Arguments:</b> | -a: first argument-b: second argument |
| <b>Example:</b>   | divide(int_col_0, int_col_1)          |

**hash\_code**

Returns an integer hash code for a string, number, or Boolean.

|                   |                                                                                                                                                                                                                 |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>number hash_code(string number boolean \$val)</code>                                                                                                                                                      |
| <b>Arguments:</b> | <code>val</code> : the value to hash                                                                                                                                                                            |
| <b>Example:</b>   | <pre>hash_code(null) → null `hash_code("Chicago") → -1884315574 `hash_code("test123") → -1422501792 `hash_code(12345) → 12345 `hash_code(123.45) → -1936584703 `hash_code(true) → 3 `hash_code(false) → 1</pre> |

### if

Returns one of the other arguments based on the truth value of the first. When the first argument is true, return the second argument. When false, return the third.

A NULL input into the first argument will be evaluated as false. Both branches are eagerly evaluated.

|                   |                                                                                                                                                                                          |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>any if(boolean \$b, any \$if_true, any \$if_false)</code>                                                                                                                          |
| <b>Arguments:</b> | <code>-b</code> : boolean value that determines which of the options to return- <code>if_true</code> : value returned if b is true- <code>if_false</code> : value returned if b is false |
| <b>Example:</b>   | <code>if(value &gt;5, "North America", "Europe")</code>                                                                                                                                  |

### json

Parses a string into a JSON value for use in further JMESPath operations. All values other than string are returned unmodified. Strings containing null, boolean, number, array, and object types are supported.

|                   |                                        |
|-------------------|----------------------------------------|
| <b>Signature:</b> | <code>object json(string \$val)</code> |
| <b>Arguments:</b> | <code>val</code> : string to parse     |
| <b>Example:</b>   | <code>json("{ \"key\": 10 }")</code>   |

### lazy\_if

Returns one of the other arguments based on the truth value of the first. When the first argument is true, return the second argument. When false, return the third. The second and third arguments are both expressions. Branches are lazily evaluated.

|                   |                                                                                                                                                                                                                                                                                          |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>any lazy_if(boolean \$b, expression \$if_true, expression \$if_false, any data)</code>                                                                                                                                                                                             |
| <b>Arguments:</b> | <code>-b</code> : boolean value that determines which of the options to return- <code>if_true</code> : expression evaluated and returned if true- <code>if_false</code> : expression evaluated and returned if false- <code>data</code> : information used to evaluate either expression |
| <b>Example:</b>   | <code>if(value &gt; 5, &amp;"North America", &amp;abs(@), 1.0)</code>                                                                                                                                                                                                                    |

### left

Returns the `num_chars` leftmost characters of the string column. If null, null is returned. If the index is greater than the length of the string, the entire string is returned.

|                   |                                                                                       |
|-------------------|---------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>string left(string \$val, number \$num_chars)</code>                            |
| <b>Arguments:</b> | -a: string column-num_chars: number of chars to use from the left of a string column. |
| <b>Example:</b>   | <code>left("ocient",3)</code>                                                         |

`lower_case`

Returns lower case version of a string.

|                   |                                            |
|-------------------|--------------------------------------------|
| <b>Signature:</b> | <code>string lower_case(string \$a)</code> |
| <b>Arguments:</b> | a: string to lower case                    |
| <b>Example:</b>   | <code>lower_case("OCIENT")</code>          |

`lpad`

Pads an input string to a specified length with a padding string added to the left side.

|                   |                                                                                                                                                                                                                                                                                     |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>string lpad(string \$val, number \$length, string \$padding)</code>                                                                                                                                                                                                           |
| <b>Arguments:</b> | -val: the input string. This cannot be null.-length: the length you want after padding. If the input string is longer than this length, it is truncated to length characters-padding: the padding string. If no padding string is provided, the input string is padded with spaces. |
| <b>Example:</b>   | <code>lpad(string_1,3, '_')</code>                                                                                                                                                                                                                                                  |

`ltrim`

Removes leading contiguous instances of a set of characters in a given string.

|                   |                                                                                                                                                                                        |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>string ltrim(string \$val, string \$strip_chars)</code>                                                                                                                          |
| <b>Arguments:</b> | -val: the input string to trim-strip_chars: set of characters that should be removed (default is ' ' if the value is null or not specified)                                            |
| <b>Example:</b>   | <code>ltrim(null, ' ') → null`ltrim(' Chicago ') → 'Chicago '`ltrim(' Chicago ', ' ') → 'Chicago '`ltrim(' Chicago ', null) → 'Chicago '`ltrim('123test123', '321') → 'test123'</code> |

`map`

An override of JmesPath's built in map which maps a given expression into elements of an array. This override makes the small adjustment that if the array itself is null, then a null is returned.

|                   |                                                                                                              |
|-------------------|--------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>array map(expression \$expr, array[any] array_1)</code>                                                |
| <b>Arguments:</b> | -expr: expression the expression to map-array_1: array[any] the array to which the expression will be mapped |



|                   |                                                                                                |
|-------------------|------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>array map(expression \$expr, array[any] array_1)</code>                                  |
| <b>Example:</b>   | <code>map(&amp;if(@ == 1, true, false), [1,2,1,1,2]) → [true, false, true, true, false]</code> |

### matches

Returns Boolean for whether this input matches the pattern.

Flags can be:

- `s` → `Pattern.DOTALL`
- `m` → `Pattern.MULTILINE`
- `i` → `Pattern.CASE_INSENSITIVE`
- `x` → `Pattern.COMMENTS`
- `q` → `Pattern.LITERAL`

|                   |                                                                              |
|-------------------|------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>boolean matches(string \$val, string \$pattern, string \$flags)</code> |
| <b>Arguments:</b> | -val: value to match-pattern: regex to match<br>against-args: regex flags    |
| <b>Example:</b>   | <code>matches(col_1, ".*mytest.*", 'i')</code>                               |

### millis\_to\_timestamp

Converts an integer number of milliseconds since the Epoch to an Ocient timestamp column.

|                   |                                                          |
|-------------------|----------------------------------------------------------|
| <b>Signature:</b> | <code>number millis_to_timestamp(number \$t)</code>      |
| <b>Arguments:</b> | <code>t</code> : value to convert to an Ocient timestamp |
| <b>Example:</b>   | <code>millis_to_timestamp(my_millis)</code>              |

### multiply

Returns the floating-point multiplication of the two arguments.

|                   |                                                      |
|-------------------|------------------------------------------------------|
| <b>Signature:</b> | <code>number multiply(number \$a, number \$b)</code> |
| <b>Arguments:</b> | -a: first argument-b: second argument                |
| <b>Example:</b>   | <code>multiply(int_col_0, int_col_1)</code>          |

### nanos\_to\_timestamp

Converts an integer number of nanoseconds since the Epoch to an Ocient timestamp column.

|                   |                                                    |
|-------------------|----------------------------------------------------|
| <b>Signature:</b> | <code>number nanos_to_timestamp(number \$t)</code> |
| <b>Arguments:</b> | -t: value to convert to an Ocient timestamp        |
| <b>Example:</b>   | <code>nanos_to_timestamp(my_nanosecs)</code>       |

### normalize\_space

Removes leading and trailing whitespace and replaces connected whitespace with a single space.

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <b>Signature:</b> | <code>string normalize_space(string \$val)</code> |
| <b>Arguments:</b> | <code>val</code> : string to normalize            |
| <b>Example:</b>   | <code>normalize_space(col_1)</code>               |

`now, current_timestamp`

Returns the current timestamp in nanos. This will correctly load into an Ocident timestamp column

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <b>Signature:</b> | <code>number now()   number current_timestamp()</code> |
| <b>Arguments:</b> | None                                                   |
| <b>Example:</b>   | <code>now(), current_timestamp()</code>                |

`null_if, nullif`

Returns null if the first argument matches any of the following arguments. Can be applied to multiple columns or to compare a column with a constant. Can also be used with more than two arguments in which case returns null if a matches any of the subsequent arguments. Returns the value the first argument if it does not match any other arguments.

|                   |                                                                                                                                              |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>any null_if(any \$val, any \$c0, [, any \$...])</code>                                                                                 |
| <b>Arguments:</b> | - <code>val</code> : value to compare- <code>c0</code> : required value to compare against-... : optional multiple values to compare against |
| <b>Example:</b>   | <code>null_if(col_a, "NULL", "NONE")</code>                                                                                                  |

`parse_array`

Given a string, parses it into an array of strings. Handles nested arrays as well.

|                   |                                                                                                                                                                                                                                                       |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>array parse_array(string \$array_string, string \$bracket_char, string \$delim_char, int \$dimensions)</code>                                                                                                                                   |
| <b>Arguments:</b> | - <code>array_string</code> : the string of the array to be parsed- <code>bracket_char</code> : the opening bracket character to use- <code>delim_char</code> : the delimiter character to use- <code>dimensions</code> : the dimension of this array |
| <b>Example:</b>   | <code>parse_array("[ 1, 2, 3]", "[", ",", 1) ≤ [1, 2, 3]`parse_array("[[1,2],[3], []]", '[', ',', 2) ≤ [[1,2], [3], []]`parse_array("{ \"cat\", \"hat\" }", "{", ",", 1) ≤ ["cat", "hat"]</code>                                                      |

`record_uuid`

Returns string that represents a unique identifier for the record in a given pipeline for a given file\_group or topic. This return value will be consistent over multiple calls in the pipeline's transformation configuration for the same source record. Can be used as a surrogate key when loading multiple tables from the same source record in a pipeline to facilitate joins when no unique identifier exists on the source record.

|                   |                                   |
|-------------------|-----------------------------------|
| <b>Signature:</b> | <code>string record_uuid()</code> |
| <b>Arguments:</b> | None                              |

|                   |                                                                                         |
|-------------------|-----------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>string record_uuid()</code>                                                       |
| <b>Example:</b>   | <code>record_uuid() -&gt;</code><br><code>"6064fc40-9961-4f5e-b74d-090458e4a609"</code> |

### replace

Returns string with text matching the regex pattern replaced by the replacement text.

Flags can be:

- `s` → `Pattern.DOTALL`
- `m` → `Pattern.MULTILINE`
- `i` → `Pattern.CASE_INSENSITIVE`
- `x` → `Pattern.COMMENTS`
- `q` → `Pattern.LITERAL`

|                   |                                                                                                                                                                                      |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>string replace(string \$val, string \$pattern, string \$replacement_text, string \$flags)</code>                                                                               |
| <b>Arguments:</b> | - <code>val</code> : value to search for replacement- <code>pattern</code> : section to replace- <code>replacement_text</code> : replacement string- <code>args</code> : regex flags |
| <b>Example:</b>   | <code>replace(col_1, '[o0]ld', 'new', 'q')</code>                                                                                                                                    |

### right

Returns the `num_chars` rightmost characters of the string column. If null, null is returned. If the index is greater than the length of the string, the entire string is returned.

|                   |                                                                                                                      |
|-------------------|----------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>string right(string \$a, number \$num_chars)</code>                                                            |
| <b>Arguments:</b> | - <code>a</code> : string column- <code>num_chars</code> : number of chars to use from the right of a string column. |
| <b>Example:</b>   | <code>right("ocient",3)</code>                                                                                       |

### rpadd

Pads an input string to a specified length with a padding string added to the right side.

|                   |                                                                                                                                                                                                                                                                                                                                  |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>string rpadd(string \$val, number \$length, string \$padding)</code>                                                                                                                                                                                                                                                       |
| <b>Arguments:</b> | - <code>val</code> : the input string. This cannot be null.- <code>length</code> : the length you want after padding. If the input string is longer than this length, it is truncated to length characters- <code>padding</code> : the padding string. If no padding string is provided, the input string is padded with spaces. |
| <b>Example:</b>   | <code>rpadd(string_1,3, '_')</code>                                                                                                                                                                                                                                                                                              |

### rtrim

Removes trailing contiguous instances of a set of characters in a given string.

|                   |                                                                                                                                                                                           |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>string rtrim(string \$val, string \$strip_chars)</code>                                                                                                                             |
| <b>Arguments:</b> | - <code>val</code> : the input string to trim- <code>strip_chars</code> : set of characters that should be removed (default is ' ' if the value is null or not specified)                 |
| <b>Example:</b>   | <code>rtrim(null, ' ') → null`rtrim(' Chicago ') → ' Chicago'``rtrim(' Chicago ', ' ') → ' Chicago'``rtrim(' Chicago ', null) → ' Chicago'``rtrim('123test123', '321') → '123test'</code> |

### sample

Returns true or false based on the given sample rate. The sample rate must be a number. Commonly used inside of IF statements. An exception will be thrown if the sample rate is outside of [0.0, 1.0].

|                   |                                                                      |
|-------------------|----------------------------------------------------------------------|
| <b>Signature:</b> | <code>boolean sample(number \$rate)</code>                           |
| <b>Arguments:</b> | <code>rate</code> : rate to return true. Must be between [0.0, 1.0]. |
| <b>Example:</b>   | <code>sample(0.5)</code>                                             |

### st\_forcepolygonccw

Forces a polygon to have a counterclockwise rotation of the outer polygon ring and a clockwise rotation of the inner polygon ring. You can use this function to convert polygons extracted from systems that interpret polygon rotation differently.

|                   |                                                                                                                                                        |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>geometry st_forcepolygonccw(string \$wkt_geometry)</code>                                                                                        |
| <b>Arguments:</b> | <code>wkt_geometry</code> : WKT string representing the polygon to rotate. Also accepts WKT of POINT or LINESTRING and returns those types unmodified. |
| <b>Example:</b>   | <code>st_forcepolygonccw('POLYGON(...)' )</code>                                                                                                       |

### st\_geomfromewkb

Returns a geometry from an Extended Well-Known Binary (EWKB) and Well-Known Binary (WKB) representations of a geometry. You can load the result of this function into POINT, LINESTRING, and POLYGON column types.

|                   |                                                                                                                                                                                       |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>geometry st_geomfromewkb(string ewkb)</code>                                                                                                                                    |
| <b>Arguments:</b> | <code>wkb</code> : hex encoded string Extended WKB or WKB representation of a geometry. EWKB or WKB can be standalone hexadecimal characters or can be prefixed with either 0x or \x. |
| <b>Example:</b>   | <code>st_geomfromewkb('0103...')``st_geomfromewkb('0x0103...')``st_geomfromewkb('\x0103...')</code>                                                                                   |

### st\_point

Constructs an POINT from numeric coordinates that represent the longitude and latitude of the point.

|                   |                                                                                                                                  |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | geometry st_point(float longitude, float latitude)                                                                               |
| <b>Arguments:</b> | - longitude: floating point longitude of the point to construct<br>- latitude: floating point latitude of the point to construct |
| <b>Example:</b>   | st_point(float_col_1, float_col_2)                                                                                               |

**substring**

Returns the substring beginning at `start_index` (with indexes beginning at 1) to the end of the string, or to `num_chars` characters.

|                   |                                                                                                                                                       |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | string substring(string \$val, number \$start_index, [number \$num_chars])                                                                            |
| <b>Arguments:</b> | val: string to substring<br>start_index: index to start at. First character is index 1.<br>num_chars: optional. The maximum number of chars to return |
| <b>Example:</b>   | substring(str_col,10)``substring(str_col,10,5)``substring('Chicago Bears',3) → 'icago Bears'``substring('Chicago Bears',3,4) → 'icag'                 |

**substring\_after**

Returns the string after the first occurrence of the needle.

|                   |                                                                                                             |
|-------------------|-------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | string substring_after(string \$val, string \$needle)                                                       |
| <b>Arguments:</b> | - val: string to substring<br>- needle: delimiter to use to designate where to return subsequent substring. |
| <b>Example:</b>   | substring_after('my_text', '_') → 'text'                                                                    |

**substring\_before**

Returns the string before the first occurrence of the needle. Both parameters are literal strings.

|                   |                                                                                                        |
|-------------------|--------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | string substring_before(string \$val, string \$needle)                                                 |
| <b>Arguments:</b> | - val: string to substring<br>- needle: delimiter to use to designate where to return prior substring. |
| <b>Example:</b>   | substring_before('my_text', '_') → 'my'                                                                |

**subtract**

Returns the difference of the two arguments.

|                   |                                             |
|-------------------|---------------------------------------------|
| <b>Signature:</b> | number subtract(number \$a, number \$b)     |
| <b>Arguments:</b> | - a: first argument<br>- b: second argument |
| <b>Example:</b>   | subtract(int_col_0, int_col_1)              |

**to\_array\_length**

Turns any JSON object into an array with N copies of that object.

|                   |                                                                                                 |
|-------------------|-------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>array[any] to_array_length(any \$val, number \$n)</code>                                  |
| <b>Arguments:</b> | - <code>val</code> : value to copy into an array<br>- <code>n</code> : number of copies to make |
| <b>Example:</b>   | <code>to_array_length(value_col,3)</code>                                                       |

#### `to_binary`

Converts a string to binary data for loading into an Ocint binary or hash column.

If `mode` is 'hex', `data` is parsed as a sequence of hexadecimal digits. Note that the sequence of digits must not begin with 0x.

Otherwise, `mode` must be the name of a character encodings listed [here](#), and `data` is encoded in that encoding.

|                   |                                                                                                                                          |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>binary to_binary(string \$data, string \$mode)</code>                                                                              |
| <b>Arguments:</b> | - <code>data</code> : data to convert to binary<br>- <code>mode</code> : either 'hex' or the name of a standard Java® character encoding |
| <b>Example:</b>   | <code>to_binary(str_col, 'utf-16')</code>                                                                                                |

#### `to_date`

Converts a string to a date for loading into an Ocint date column. Format string follows these [formatting rules](#).

|                   |                                                                                                                    |
|-------------------|--------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>number to_date(string \$date, string \$format_string)</code>                                                 |
| <b>Arguments:</b> | - <code>date</code> : date string to convert<br>- <code>format_string</code> : format string to use for conversion |
| <b>Example:</b>   | <code>to_date(str_col, 'yyyy-MM-dd')</code>                                                                        |

#### `to_time`

Converts a string into a time of day for loading into an Ocint time column. Format string follows these [formatting rules](#).

|                   |                                                                                                                    |
|-------------------|--------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>number to_time(string \$time, string \$format_string)</code>                                                 |
| <b>Arguments:</b> | - <code>time</code> : time string to convert<br>- <code>format_string</code> : format string to use for conversion |
| <b>Example:</b>   | <code>to_time(str_col, 'HH:mm:ss.nnnnnnnnn')</code>                                                                |

#### `to_timestamp`

Converts a string to timestamp for loading into an Ocint timestamp column. Format string follows these [formatting rules](#).

|                   |                                                                                                   |
|-------------------|---------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | number to_timestamp(string \$timestamp, string \$format_string)                                   |
| <b>Arguments:</b> | - timestamp: timestamp string to<br>convert-format_string: format string to use for<br>conversion |
| <b>Example:</b>   | to_timestamp(str_col, 'yyyy-MM-dd<br>HH:mm:ss.SSS')                                               |

### to\_tuple

Works on tuples that get tokenized at the first nesting level by a given delimiter. Nested tuples, arrays or objects are considered entirely. For each tokenized tuple element, a transformation function can be specified that is applied to the tuple element. Tokenized elements are always treated as strings, requiring additional transformation when being treated as numbers.

|                   |                                                                                                                                   |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | array to_tuple(string \$tuple_string, string \$bracket_char, string \$delim_char [,<br>expression \$transformation_function ...]) |
| <b>Arguments:</b> | - tuple_string: the string of the tuple to be parsed.                                                                             |

Must be encapsulated in the given bracket character and its closing peer. Empty tuple values are treated as empty strings. If the whole argument is NULL, the function returns NULL ignoring all given expressions in transformation.

**Example:** | to\_tuple((-1,2), '(', ',', &abs(to\_number(@)), &@ → '(1.0,2)' | to\_tuple(('8',2),  
'(', ',', &if(@ != null, @, 'missing'), &if(length(@) > 0, @, 'missing'), &to\_number(@) ) →  
('missing','8',2) |

### tokenize

Returns array of strings after splitting on the regex pattern.

Flags can be:

- s → Pattern.DOTALL
- m → Pattern.MULTILINE
- i → Pattern.CASE\_INSENSITIVE
- x → Pattern.COMMENTS
- q → Pattern.LITERAL

|                   |                                                                              |
|-------------------|------------------------------------------------------------------------------|
| <b>Signature:</b> | array[string] tokenize(string \$val, string \$pattern, string \$flags)       |
| <b>Arguments:</b> | - val: string to tokenize-pattern: token regex<br>pattern-flags: regex flags |
| <b>Example:</b>   | tokenize(col_1, '\$', 'i')                                                   |

### translate

Replaces characters according to a map of characters. If the to\_chars string is shorter than the from\_chars string, the function removes the characters that are not mapped.

|                   |                                                                                        |
|-------------------|----------------------------------------------------------------------------------------|
| <b>Signature:</b> | string translate(string \$val, string \$from_chars, string \$to_chars)                 |
| <b>Arguments:</b> | - val: string to translate-from_char: from character<br>map-to_chars: to character map |

|                   |                                                                                     |
|-------------------|-------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>string translate(string \$val, string \$from_chars, string \$to_chars)</code> |
| <b>Example:</b>   | <code>translate('Lorem Ipsum', 'mnop', '1234') → 'L3re1 I4su1'</code>               |

**trim**

Removes leading and trailing contiguous instances of a set of characters in a given string.

|                   |                                                                                                                                                                                 |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>string trim(string \$val, string \$strip_chars)</code>                                                                                                                    |
| <b>Arguments:</b> | - <code>val</code> : the input string to trim- <code>strip_chars</code> : set of characters that should be removed (default is ' ' if the value is null or not specified)       |
| <b>Example:</b>   | <code>trim(null, ' ') → null``trim(' Chicago ') → 'Chicago'``trim(' Chicago ', ' ') → 'Chicago'``trim(' Chicago ', null) → 'Chicago'``trim('123test123', '321') → 'test'</code> |

**truncate**

Truncates a decimal number to have a specified number of digits after its decimal point.

If `decimal` is null, this function returns null.

During the transformation stage, the LAT stores numbers as integers or floating point numbers by default. However, floating point numbers are inexact. For this reason, this function might return unexpected results. For example, `truncate(`0.29`, `2`)` evaluates to 0.28, not 0.29 (because 0.29 is really 0.28999999...).

|                   |                                                                                                                                     |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>number truncate(number \$decimal, number \$decimal_places)</code>                                                             |
| <b>Arguments:</b> | - <code>decimal</code> : the number to truncate- <code>decimal_places</code> : the number of digits to keep after the decimal point |
| <b>Example:</b>   | <code>truncate(decimal_col,2)</code>                                                                                                |

**tuple\_element\_transformation**

Works on an element of a given array and applies the specified expression to that element. The function returns an array or null if specified.

|                                                                                                                                                                                                                                                                                                             |                                                                                                                                   |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b>                                                                                                                                                                                                                                                                                           | <code>array tuple_element_transformation(array \$array_values, number \$array_index, expression \$transformation_function)</code> |
| <b>Arguments:</b>                                                                                                                                                                                                                                                                                           | - <code>array_values</code> : an array of values.                                                                                 |
| If the whole argument is NULL, the function returns NULL.- <code>array_index</code> : a number pointing to an element in the array. The first index of the array is 0.- <code>transformation_function</code> : a transformation functions to apply to array element specified by <code>array_index</code> . |                                                                                                                                   |
| <b>Example:</b>                                                                                                                                                                                                                                                                                             | <code>tuple_element_transformation([-1.0,2,3, '0', abs(@)) → [1.0,2,3] ]</code>                                                   |



**unique**

Return all unique values in an array.

|                   |                                                                                  |
|-------------------|----------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>array[any] unique(array[any] \$vals)</code>                                |
| <b>Arguments:</b> | <code>vals</code> : array to return unique values from, removing all duplicates. |
| <b>Example:</b>   | <code>unique(vals_col)</code>                                                    |

**upper\_case**

Returns upper case version of string.

|                   |                                            |
|-------------------|--------------------------------------------|
| <b>Signature:</b> | <code>string upper_case(string \$a)</code> |
| <b>Arguments:</b> | <code>a</code> : string to upper case      |
| <b>Example:</b>   | <code>upper_case("ocient")</code>          |

**width\_bucket**

Returns the bucket number of the queried value in a histogram starting at min, ending at max, and consisting of num\_buckets count of buckets.

- The values are inclusive on the lower bound and exclusive of the upper bound so that all buckets are the same width.
- Also note that requesting 100 buckets will actually get you 102 possible buckets. Bucket 0 through 101. Where 0 captures all values below the minimum and 101 captures all values at the maximum and beyond.

|                   |                                                                                                                                                                                                                            |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>number width_bucket(number \$val, number \$min, number max, number \$num_buckets)</code>                                                                                                                             |
| <b>Arguments:</b> | - <code>val</code> : value to determine the bucket number for<br>- <code>min</code> : min value of histogram<br>- <code>max</code> : max value of histogram<br>- <code>num_buckets</code> : number of buckets in histogram |
| <b>Example:</b>   | <code>width_bucket(my_col,0,100,20)</code>                                                                                                                                                                                 |

**zip**

Zips N one-dimensional arrays into a single two-dimensional array with each inner array containing N elements.

The resulting outer array will have a length equal to the max length of any input array. For input arrays smaller than the max length, null elements will be placed in the inner arrays.

|                   |                                                                                                                   |
|-------------------|-------------------------------------------------------------------------------------------------------------------|
| <b>Signature:</b> | <code>array[any] zip(array[any] \$a0, array[any] \$a1, [, array[any] \$...])</code>                               |
| <b>Arguments:</b> | - <code>a0</code> : array to zip<br>- <code>a1</code> : array to zip<br>- ... : Optional additional arrays to zip |
| <b>Example:</b>   | <code>zip(array_col0, array_col1, array_col2)</code>                                                              |

## Special Functions

### EXPLODE

Performs an explosion on all indicated columns, producing one row per value in the exploded array. This can be applied to multiple columns in the same table. Non-exploded columns are held constant and exploded columns are zipped together in array order. If multiple exploded columns have different array lengths, the record is exploded into N records where N is the max length. Columns with array lengths less than N will have null values after each of their values are exploded.

You can explode a maximum of 8,192 array elements for each source record.

#### Usage

EXPLODE(array[any] arr) can be used like any other JMESPath function on a JSON Array type with a few limitations:

- EXPLODE can only be used as the outermost function in a JMESPath expression
- EXPLODE only works on a single dimension. Multidimensional explode is not supported.

#### Examples

##### Single-Column EXPLODE

Transform Configuration:

```
{
  "transform": {
    "topics" : {
      "topic0" : {
        "tables" : {
          "table0" : {
            "columns" {
              "col0": "EXPLODE(array0)",
              "col1": "integer0"
            }
          }
        }
      }
    }
  }
}
```

Input:

```
{
  "array0": [0, 1, 2, 3],
  "integer0": 4
}
```

Output:

```
[
  {
    "col0": 0,
    "col1": 4
  },
  {
    "col0": 1,
    "col1": 4
  },
  {
    "col0": 2,
    "col1": 4
  },
  {
    "col0": 3,
    "col1": 4
  }
]
```

```

    "col0": 2,
    "col1": 4
  },
  {
    "col0": 3,
    "col1": 4
  }
]

```

### Multi-Column EXPLODE

Transform Configuration:

```

{
  "transform": {
    "topics" : {
      "topic0" : {
        "tables" : {
          "table0" : {
            "columns" : {
              "col0": "EXPLODE(array0)",
              "col1": "EXPLODE(array1)",
              "col2": "integer0"
            }
          }
        }
      }
    }
  }
}

```

Input:

```

{
  "array0": [0, 1, 2, 3],
  "array1": [0, 1, 2],
  "integer0": 4
}

```

Output:

```

[
  {
    "col0": 0,
    "col1": 0,
    "col2": 4
  },
  {
    "col0": 1,
    "col1": 1,
    "col2": 4
  },
  {
    "col0": 2,
    "col1": 2,
    "col2": 4
  },
  {

```

```

        "col10": 3,
        "col11": null,
        "col12": 4
    }
]

```

Note that in the 4th record `col11` has a `null` value because `array1` only has 3 values while `array0` has 4.

## Related Links

[Ingest Data with Legacy LAT Reference](#)

### 12.5.2 LAT User-Defined Transformations

Data Pipelines are now the preferred method for loading data into the Ocient System. For details, see [Load Data](#).

In addition to built-in transformations, users can extend LAT by writing custom user-defined transformations (UDTs).

To bootstrap UDT development, contact Ocient® Support.

## UDT Interface

A UDT is implemented as a [JMESPath Java Function](#). A Function is comprised of three elements:

1. **name** - the function's name determines how it can be called in a user pipeline e.g., `negate`
2. **arguments** - what inputs the function expects to be called with
3. **body** - the function's implementation, including return value

UDT names must be globally unique. LAT validates UDT names on startup to ensure that no two UDT names collide with each other and that no UDT name collides with an LAT builtin transformation.

This example function demonstrates these concepts:

```

// negate(number) - takes an input number and changes its sign
// function name - derived from the class name, NegateFunction -> negate
public class NegateFunction extends BaseFunction {
    public NegateFunction() {
        super(
            // function arguments; this function takes exactly one argument of type number
            ArgumentConstraints.typeOf(JmesPathType.NUMBER));
    }

    // function body
    protected <T> T callFunction(Adapter<T> runtime, List<FunctionArgument<T>> arguments) {
        double original = runtime.toNumber(arguments.get(0).value()).doubleValue();
        return runtime.createNumber(original * -1);
    }
}

```

## UDT Packaging

One or more UDTs can be packaged in a jar file for consumption by LAT. In addition to the UDT implementations themselves, the jar file must contain a services file containing a mapping of all the UDTs the jar should expose for

use by LAT. This service file must be located at `META-INF/services/io.burt.jmespath.function.Function` in the resulting jar, and its contents should be as follows:

```
com.ocient.myudt.NegateFunction
```

The file at the path `META-INF/services/io.burt.jmespath.function.Function` should include the fully qualified name of one or more UDTs contained within this jar to be used by LAT. Each UDT should be specified on its own line in the file.

## Dependencies

If the UDT has dependencies on third party libraries, those libraries must be packaged in the same jar file. This is commonly referred to as a “fat jar”, “uber jar”, or “jar with dependencies.” Instructions for building such a jar file with Maven™ can be found [here](#).

## UDT Deployment

LAT can be configured to search for UDT jars in a particular directory. See [LAT Source Configuration](#) for details. LAT can read multiple UDTs across multiple jar files.

On startup, LAT will read all available UDTs into its configuration. To update the set of UDTs available to LAT, such as to add an additional UDT jar, a process restart is required.

## Related Links

[Ingest Data with Legacy LAT Reference](#)

### 12.5.3 LAT Record Filtering

Data Pipelines are now the preferred method for loading data into the Ocient System. For details, see [Load Data](#).

Incoming records can optionally be filtered by a JMESPath expression added with an additional filter key in a pipeline configuration.

```
{
  "transform": {
    "topics": {
      "my_topic": {
        "filter": "<jmespath-expression>",
        "tables": {
          "my_table": {
            "filter": "<jmespath-expression>",
            "columns": {}
          }
        }
      }
    }
  }
}
```

Records that pass through the filter (the filter expression evaluates to true) are processed as normal, while records that fail the filter (the filter expression evaluates to false) are dropped. Note that filters can be applied at the topic or table levels. A record needs to pass through all configured filters in order to be loaded to a table.

The filter JMESPath expression must evaluate to a `boolean` value. Examples of this are JMESPath comparators or functions that return a `boolean` like `contains`, `starts_with`, and `ends_with`.

**Example**

Pipeline:

```
{
  "transform": {
    "topics": {
      "topic0": {
        "filter": "integer0 == `2`",
        "tables": {
          "table0": {
            "filter": "integer1 == `3`",
            "columns": {
              "col0": "integer0",
              "col1": "integer1"
            }
          }
        }
      }
    }
  }
}
```

Input:

```
[
  {
    "integer0": 0,
    "integer1": 3
  },
  {
    "integer0": 2,
    "integer1": 1
  },
  {
    "integer0": 2,
    "integer1": 3
  }
]
```

Output:

```
[
  {
    "integer0": 2,
    "integer1": 3
  }
]
```

**Related Links**[Ingest Data with Legacy LAT Reference](#)**12.6 LAT Service Configuration**

Data Pipelines are now the preferred method for loading data into the Ocident System. For details, see [Load Data](#).

The LAT service is configurable using Service Configuration.

This configuration is derived from three sources, with the following precedence order:

1. Environment Variables
2. Properties File
3. Code Defaults

### Configuration File Backup

When you install a new LAT instance or upgrade the existing instance, LAT creates a backup file of the `/etc/lat/lat.conf` as `/etc/lat/override.conf` and creates a new version of the `/etc/lat/lat.conf` file. The `/etc/lat/override.conf` file does not control the LAT configuration. If you made prior changes to `lat.conf`, you must review `override.conf` and add any custom values into the revised `lat.conf`.

### Environment Variables

#### LAT\_DEFAULT\_WORKERS

Default number of workers used by a newly configured pipeline.

|                              |                     |
|------------------------------|---------------------|
| <b>Environment Variable:</b> | LAT_DEFAULT_WORKERS |
| <b>Property:</b>             | lat.default.workers |
| <b>Default:</b>              | Server core count   |

#### LAT\_SHUTDOWN\_DURATION

Duration, in milliseconds, to wait prior to termination on shutdown.

|                              |                       |
|------------------------------|-----------------------|
| <b>Environment Variable:</b> | LAT_SHUTDOWN_DURATION |
| <b>Property:</b>             | lat.shutdown.duration |
| <b>Default:</b>              | 120000                |

#### LAT\_UDT\_PATH

Directory containing zero or more user defined transformation (UDT) jars. Ensure that the specified path can be accessed by the LAT (e.g., CentOS® does not allow the LAT to access the home directory of an administrative user).

|                              |              |
|------------------------------|--------------|
| <b>Environment Variable:</b> | LAT_UDT_PATH |
| <b>Property:</b>             | lat.udt.path |
| <b>Default:</b>              | null         |

#### LAT\_API\_PORT

REST API and Metrics port.

|                              |              |
|------------------------------|--------------|
| <b>Environment Variable:</b> | LAT_API_PORT |
| <b>Property:</b>             | lat.api.port |
| <b>Default:</b>              | 8080         |

**LAT\_DATA\_PATH**

Path to the LAT data directory. This directory is used to store persistent data related to pipelines (**pipeline/**) and sinks (**sink/**). In almost all cases, using the [LAT Client Command Line Interface](#) is preferred over editing the files in this directory manually. Ensure that the specified path can be accessed by the LAT (e.g., CentOS does not allow the LAT to access the home directory of an administrative user).

|                              |               |
|------------------------------|---------------|
| <b>Environment Variable:</b> | LAT_DATA_PATH |
| <b>Property:</b>             | lat.data.path |
| <b>Default:</b>              | .lat-data     |

**LAT\_AUTO\_START**

On LAT startup, automatically starts running a Pipeline if one is present. This only supports pipelines with Apache® Kafka® sources; File sources will not automatically start even if this configuration is set to **true**.

|                              |                |
|------------------------------|----------------|
| <b>Environment Variable:</b> | LAT_AUTO_START |
| <b>Property:</b>             | lat.auto.start |
| <b>Default:</b>              | false          |

**LAT\_ALLOW\_LOG\_ORIGINAL\_RECORDS**

Allow the original records that are being loaded to be logged when errors occur. This setting must be enabled for **log\_original\_records** to be enabled.

This setting only affects pipelines that log errors to the error log file.

|                              |                                |
|------------------------------|--------------------------------|
| <b>Environment Variable:</b> | LAT_ALLOW_LOG_ORIGINAL_RECORDS |
| <b>Property:</b>             | lat.allow.log.original.records |
| <b>Default:</b>              | false                          |

**LAT\_MAX\_RETRIEVABLE\_ERRORS**

A limit on the number of errors that you can retrieve using the **pipeline errors** command of the LAT client.

|                              |                             |
|------------------------------|-----------------------------|
| <b>Environment Variable:</b> | LAT_MAX_RETRIEVABLE_ERRORS  |
| <b>Property:</b>             | lat.max.retrieveable.errors |
| <b>Default:</b>              | 1000                        |

**LAT\_API\_PROXY\_HOST**

API proxy host for OKTA® authorization requests.

|                              |                    |
|------------------------------|--------------------|
| <b>Environment Variable:</b> | LAT_API_PROXY_HOST |
| <b>Property:</b>             | lat.api.proxy.host |
| <b>Default:</b>              | null               |

**LAT\_API\_PROXY\_PORT**

API proxy port for Okta authorization requests.



|                              |                    |
|------------------------------|--------------------|
| <b>Environment Variable:</b> | LAT_API_PROXY_PORT |
| <b>Property:</b>             | lat.api.proxy.PORT |
| <b>Default:</b>              | null               |

**LAT\_API\_OAUTH\_DOMAIN**

Okta authorization domain.

|                              |                      |
|------------------------------|----------------------|
| <b>Environment Variable:</b> | LAT_API_OAUTH_DOMAIN |
| <b>Property:</b>             | lat.api.oauth.domain |
| <b>Default:</b>              | null                 |

**LAT\_API\_OAUTH\_ENDPOINTS**

A list of LAT API endpoints that Okta should apply authentication on. To override, provide a comma-delimited list, e.g., /endpoint1,/endpoint2,/endpoint3.

|                              |                         |
|------------------------------|-------------------------|
| <b>Environment Variable:</b> | LAT_API_OAUTH_ENDPOINTS |
| <b>Property:</b>             | lat.api.oauth.endpoints |
| <b>Default:</b>              | null                    |

**LAT\_API\_OAUTH\_SERVER**

Okta authorization server.

|                              |                      |
|------------------------------|----------------------|
| <b>Environment Variable:</b> | LAT_API_OAUTH_SERVER |
| <b>Property:</b>             | lat.api.oauth.server |
| <b>Default:</b>              | null                 |

**LAT\_API\_CERT\_PATH**

API cert file path. Ensure that the specified path can be accessed by the LAT (e.g., CentOS® does not allow the LAT to access the home directory of an administrative user).

|                              |                   |
|------------------------------|-------------------|
| <b>Environment Variable:</b> | LAT_API_CERT_PATH |
| <b>Property:</b>             | lat.api.cert.path |
| <b>Default:</b>              | null              |

**LAT\_DNS\_CACHE\_TTL**

The time-to-live (TTL) in seconds that the LAT should use for its DNS cache. This configuration is only supported as an Environment Variable or as Default.

|                              |                   |
|------------------------------|-------------------|
| <b>Environment Variable:</b> | LAT_DNS_CACHE_TTL |
| <b>Property:</b>             | N/A               |
| <b>Default:</b>              | 30                |

**LAT\_MEMORY**

Maximum JVM® heap memory. This configuration is only supported as an Environment Variable or as Default when running the LAT through the provided package service file or through the provided `lat.sh` script. If directly running the Jar JVM heap memory should be set through command line arguments.

|                              |            |
|------------------------------|------------|
| <b>Environment Variable:</b> | LAT_MEMORY |
| <b>Property:</b>             | N/A        |
| <b>Default:</b>              | 32G        |

**LAT\_MEMORY\_DIRECT**

Maximum JVM direct memory. This configuration is only supported as an Environment Variable or as Default when running the LAT through the provided package service file or through the provided `lat.sh` script. If directly running the Jar JVM heap memory should be set through command line arguments.

|                              |                   |
|------------------------------|-------------------|
| <b>Environment Variable:</b> | LAT_MEMORY_DIRECT |
| <b>Property:</b>             | N/A               |
| <b>Default:</b>              | 32G               |

**Override Environment Variables**

The recommended way to manage the production service configuration when LAT is running as a daemon is using environment variables or the `/etc/systemd/system/lat.service.d/override.conf` file.

To override LAT environment variables as a `systemd` service, there are two recommended commands that edit the `etc/systemd/system/lat.service.d/override.conf` file.

1. Run `sudo vim etc/systemd/system/lat.service.d/override.conf` or `sudo systemctl edit lat`.
2. Insert the environment variables to override in the file. Save and exit the editor.

**Configuration File Example:**

```
[Service]
Environment=LAT_MEMORY_DIRECT=200G
Environment=LAT_MEMORY=200G
```

**Related Links**

[Ingest Data with Legacy LAT Reference](#)

[LAT Packaging and Installation](#)

**12.7 LAT Metrics**

Data Pipelines are now the preferred method for loading data into the Ocident System. For details, see [Load Data](#).

**Configuration**

LAT metrics are exposed using the LAT REST API.

**Endpoints**

LAT exposes two metrics endpoints; a *partitions* endpoint, and a *pipeline* endpoint.

## Partition Metrics

The partitions endpoint exposes individual metrics for every partition that is participating in the current LAT instance. For Apache® Kafka® loads, partitions correspond to Kafka's partitions. For File Source based loads, partitions correspond to the independent file sets created within each File Group for parallel processing.

The partitions metrics endpoint can be accessed by issuing a curl request as follows:

```
curl http://127.0.0.1:8080/v2/metrics/read/lat:type=partitions
```

A response for this request will look as follows, with one entry per partition in the partitions array:

```
{
  "request": {
    "mbean": "lat:type=partitions",
    "type": "read"
  },
  "value": {
    "partitions": [
      {
        "offsets_durable": 0,
        "pushes_errors": 0,
        "pushes_attempts": 0,
        "rows_pushed": 0,
        "offsets_written": 0,
        "records_buffered": 0,
        "records_errors_column": 0,
        "records_errors_deserialization": 0,
        "records_errors_transformation": 0,
        "offsets_processed": 0,
        "lag": 0,
        "partition": "topic-0",
        "records_filter_accepted": 0,
        "records_errors_row": 0,
        "records_filter_rejected": 0,
        "records_errors_generic": 0,
        "producer_send_attempts": 0,
        "offsets_pushed": 0,
        "pushes_unacknowledged": 0,
        "invalid_state": 0,
        "bytes_pushed": 0,
        "errors_partition": 0,
        "records_errors_total": 0,
        "offsets_buffered": 0,
        "complete": 0,
        "offsets_end": 0,
        "producer_send_errors": 0
      }
    ]
  },
  "timestamp": 1642497992,
  "status": 200
}
```

## Partition Metrics Definitions

Metrics definitions are as follows:

| Metric                         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| partition                      | Topic partition pair (e.g.,mytopic-0)                                                                                                                                                                                                                                                                                                                                                                                                      |
| records_errors_deserialization | Total number of processed records which failed to deserialize                                                                                                                                                                                                                                                                                                                                                                              |
| records_errors_transformation  | Total number of processed records which failed to transform                                                                                                                                                                                                                                                                                                                                                                                |
| records_errors_column          | Total number of processed records which failed to bind transformed values to columns                                                                                                                                                                                                                                                                                                                                                       |
| records_errors_row             | Total number of processed records which failed during builder row advancement/other builder errors                                                                                                                                                                                                                                                                                                                                         |
| records_errors_generic         | Total number of processed records which failed for any other reason                                                                                                                                                                                                                                                                                                                                                                        |
| records_errors_total           | Sum of all record errors for this topic                                                                                                                                                                                                                                                                                                                                                                                                    |
| records_buffered               | Total number of processed records which have not yet been pushed. Should always be modulo configured buffer size                                                                                                                                                                                                                                                                                                                           |
| records_filter_rejected        | Number of records rejected by the topic filter.                                                                                                                                                                                                                                                                                                                                                                                            |
| records_filter_accepted        | Number of records accepted by the topic filter.                                                                                                                                                                                                                                                                                                                                                                                            |
| offsets_processed              | For Kafka loading, records are processed if their offsets are strictly less than the value ofoffsets_processed.For file loading, this represents the index of the most recently processed file.If there is an error during the processing of a record/file, this metric is still updated according to that record/file's offset. When a file load is complete, this offset becomes equal to offsets_end. Can decrease due to reprocessing. |
| offsets_written                | For Kafka loading, records are written if their offsets are strictly less than the value of offsets_written.For file loading, this represents the index of the most recently written file.This metric is only updated when the record/file is processed and the write is successful. Note that a successful write does not mean the record/file is durable. Can decrease due to reprocessing.                                              |
| offsets_buffered               | For Kafka loading, this represents the most recently processed record which was processed to completion and is waiting to be pushed into the {{OcientHyperscaleDataWarehouse}}.For file loading, this represents the index of the most recently processed file which was processed to completion and is waiting to be pushed into the {{Ocient}} data warehouse.                                                                           |
| offsets_pushed                 | For Kafka loading, this represents the highest offset of the batch of rows most recently pushed to the Ocient data warehouse.For file loading, this represents the highest file index of the batch of rows most recently pushed to the Ocient data warehouse.Can decrease due to reprocessing.                                                                                                                                             |
| offsets_end                    | For Kafka loading, this represents the end offset of the partition.For file loading, this represents the file count of the partition.Should never decrease.                                                                                                                                                                                                                                                                                |
| offsets_durable                | For Kafka loading, records are made durable if their offsets are strictly less than the value of offsets_durable.For file loading, this represents the most durable file index.Should never decrease.                                                                                                                                                                                                                                      |
| producer_send_attempts         | Number of Kafka error topic producer send attempts                                                                                                                                                                                                                                                                                                                                                                                         |
| producer_send_errors           | Number of Kafka error topic producer send errors                                                                                                                                                                                                                                                                                                                                                                                           |

| Metric                | Description                                                                                                                                                                                                                        |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bytes_pushed          | Number of bytes pushed into the Ocient data warehouse                                                                                                                                                                              |
| rows_pushed           | Number of rows pushed into the Ocient data warehouse                                                                                                                                                                               |
| pushes_attempts       | Number of attempts to push record batches into the Ocient data warehouse for this partition                                                                                                                                        |
| pushes_errors         | Number of attempts to push record batches into the Ocient data warehouse which resulted in error                                                                                                                                   |
| pushes_unacknowledged | Number of attempts to push record batches into the Ocient data warehouse for which no response has yet been received                                                                                                               |
| invalid_state         | Number of times a code path was reached in LAT which is erroneous                                                                                                                                                                  |
| complete              | Whether or not the partition has any records left to process at the moment; this status can change often in a Kafka load but will likely not change from complete to incomplete in a file load                                     |
| errors_partition      | The number of times the LAT failed to fetch records for a particular partition                                                                                                                                                     |
| lag                   | Calculated as offsets_end - offsets_durable; In a Kafka load, lag represents the number of unprocessed records. In a file load, lag represents the number of unprocessed files. Lag becomes zero when the complete status is true. |

### Pipeline Metrics

The pipeline endpoint exposes aggregate metrics for the LAT instance. There are three categories of pipeline metrics:

| Pipeline Metric Category | Description                                                                                                                                                                                                                                       |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| "pipeline" aggregate     | Monotonically-increasing metrics that act as lifetime counters for the running pipeline. They are independent of the current set of partitions. For instance, <code>rows_pushed</code> is the total number of rows ever pushed for this pipeline. |
| "partitions" aggregate   | Metrics that depend on the current set of active partitions for the running pipeline. For instance, <code>total_lag</code> is the summed lag across all currently active partitions for the pipeline.                                             |
| pipeline-specific        | Pipeline only metrics that are not aggregates.                                                                                                                                                                                                    |

The Pipeline Metrics endpoint can be accessed by issuing a curl request as follows:

```
curl http://127.0.0.1:8080/v2/metrics/read/lat:type=pipeline
```

A response for this request will look as follows:

```
{
  "request": {
    "mbean": "lat:type=pipeline",
    "type": "read"
  },
  "value": {
    "pipeline": {
      "pushes_errors": 0,
      "pushes_attempts": 0,
```

```

    "rows_pushed": 0,
    "records_errors_column": 0,
    "records_errors_deserialization": 0,
    "records_errors_transformation": 0,
    "bytes_pushed": 0,
    "errors_partition": 0,
    "records_filter_accepted": 0,
    "records_errors_total": 0,
    "records_errors_row": 0,
    "records_filter_rejected": 0,
    "records_errors_generic": 0,
    "producer_send_attempts": 0,
    "producer_send_errors": 0
  },
  "partitions": {
    "max_lag": 0,
    "total_offsets_buffered": 0,
    "avg_lag": 0.0,
    "total_pushes_unacknowledged": 0,
    "min_lag": 0,
    "total_lag": 0,
    "total_invalid_state": 0,
    "total_offsets_pushed": 0,
    "total_records_buffered": 0,
    "total_complete": 0,
    "total_offsets_processed": 0
  },
  "paused": 0,
  "bytes_buffered": 0,
  "complete": 0,
  "workers": 32,
  "lat_version": "3.0.0"
},
"timestamp": 1642497956,
"status": 200
}

```

Individual metrics such as `lag`, `offsets_buffered`, `pushes_errors`, etc. are defined within the **Metrics** section. However, metrics specific *only* to the pipeline endpoint are defined as follows:

| Metric                      | Description                                                             |
|-----------------------------|-------------------------------------------------------------------------|
| <code>paused</code>         | 0 if processing is active across all workers, 1 otherwise.              |
| <code>bytes_buffered</code> | Global allocated memory in bytes.                                       |
| <code>complete</code>       | 1 if the pipeline has finished the entire loading process, 0 otherwise. |
| <code>workers</code>        | The number of active workers.                                           |
| <code>lat_version</code>    | The version of the running LAT server.                                  |

## Related Links

[Ingest Data with Legacy LAT Reference](#)

[System Information REST Endpoints](#)

## 12.8 LAT Log4J Configuration

Data Pipelines are now the preferred method for loading data into the Ocident System. For details, see [Load Data](#).

The LAT uses Apache® Log4J 2™ for logging. Users can configure their logging setup however they would like. The LAT packages come with example `log4j2.xml` files that can be used directly, or modified by the user. This XML code in the `log4j2.xml` file shows an example configuration.

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" strict="true" monitorInterval="30" packages="biz.paluch.logging.gelf.log4j2">
  <Appenders>
    <RollingRandomAccessFile name="HttpLog" append="true" fileName="logs/http.log" filePattern="logs/http.%d{ISO8601}-%i.log" >
      <PatternLayout pattern="%m%n" />
      <Policies>
        <SizeBasedTriggeringPolicy size="100 MB" />
      </Policies>
      <DefaultRolloverStrategy>
        <Delete basePath="logs">
          <IfFileName glob="http.*.log.gz" />
          <IfAccumulatedFileSize exceeds="5 GB" />
        </Delete>
      </DefaultRolloverStrategy>
    </RollingRandomAccessFile>
    <RollingRandomAccessFile name="LATLog" append="true" fileName="logs/lat.log" filePattern="logs/lat.%d{ISO8601}-%i.log" >
      <PatternLayout pattern="%d{ISO8601} %-5p [%t] - %m%n" />
      <Policies>
        <SizeBasedTriggeringPolicy size="100 MB" />
      </Policies>
      <DefaultRolloverStrategy>
        <Delete basePath="logs">
          <IfFileName glob="lat.*.log.gz" />
          <IfAccumulatedFileSize exceeds="20 GB" />
        </Delete>
      </DefaultRolloverStrategy>
    </RollingRandomAccessFile>
    <RollingRandomAccessFile name="ErrorLog" append="true" fileName="logs/error.log" filePattern="logs/error.%d{ISO8601}-%i.log" >
      <PatternLayout pattern="%m%n" />
      <Policies>
        <SizeBasedTriggeringPolicy size="100 MB" />
      </Policies>
      <DefaultRolloverStrategy fileIndex="nomax">
        <Delete basePath="logs">
          <IfFileName glob="error.*.log.gz" />
          <IfAccumulatedFileSize exceeds="20 GB" />
        </Delete>
      </DefaultRolloverStrategy>
    </RollingRandomAccessFile>
  </Appenders>
  <!--
  <Gelf name="Graylog" host="udp:graylog-dev" port="12201" version="1.1" extractStackTrace="true">
    <Field name="process_name" literal="lat" />
    <Field name="log_level" pattern="%-5p" />
    <Field name="thread" pattern="%t" />
  </Gelf>
  -->
```

```

</Appenders>
<Loggers>
  <Logger name="org.apache.kafka" level="info" />
  <Logger name="com.ocient.lat.sink.ocient.Binders.Binder" level="info" />
  <!-- HttpChannel at debug so that jetty preserves the stacktrace when logging uncaught exceptions -->
  <Logger name="org.eclipse.jetty.server.HttpChannel" level="debug" />
  <Logger name="org.eclipse.jetty.server.RequestLog">
    <AppenderRef ref="HttpLog" />
  </Logger>
  <Logger name="com.ocient.lat.sink.file.FileErrorSink" level="info" additivity="false">
    <AppenderRef ref="ErrorLog" />
    <!--
    <AppenderRef ref="Graylog" />
    -->
  </Logger>
  <Logger name="software.amazon.awssdk" level="info" />
  <Logger name="io.netty" level="info" />
  <Root level="info">
    <AppenderRef ref="LATLog" />
    <!--
    <AppenderRef ref="Graylog" />
    -->
  </Root>
</Loggers>
</Configuration>

```

Relevant Loggers:

- `org.apache.kafka`: Apache® Kafka® logger.
- `com.ocient.lat.sink.ocient.Binders.Binder`: LAT binder logger. Can be verbose, recommended to keep at info level.
- `org.eclipse.jetty.server.HttpChannel` and `org.eclipse.jetty.server.RequestLog`: Jetty logs for the HTTP endpoint.
- `com.ocient.lat.Worker`: Main worker logger containing important pipeline logs.

Valid log levels are (in order of least to most verbose):

- error
- warn
- info
- debug
- trace

Log4j can be configured to log different packages at different levels as shown in the example. Here, the example logs the `com.ocient.streaming.client` package at the trace level, but the remainder of the code is logged at info level.

## Related Links

[Ingest Data with Legacy LAT Reference](#)

## 12.9 LAT Packaging and Installation

Data Pipelines are now the preferred method for loading data into the Ocient System. For details, see [Load Data](#).

The Ocient® LAT supports multiple types of packages for deployment and installation:



- Debian® .deb package
- CentOS® .rpm package
- Raw .tar.gz tar folder
- Direct .jar
- Docker® container

Contact Ocint Support for the latest package for a deployment method.

### Prerequisites

- Java® 17 or greater
  - Can either be system-level Java, or installed somewhere accessible by the user.

The LAT must run on Ocint Loader Nodes where you installed the Ocint RPM to provide access to numerical libraries, ensuring the correct loading of geospatial data types. If this condition is not met, some geospatial objects might load inaccurately.

## Debian/CentOS Packages

### 1. Install/Upgrade

Debian: `sudo dpkg -i lat_${VERSION}_all.deb`

CentOS: `sudo rpm -i lat-${VERSION}-1.noarch.rpm`

### 2. Setup Java

If the default system Java is version 17 or greater, no action is required. If not, edit the service file to contain a `JAVA_HOME` environment variable pointing to a Java version 17 or greater installation.

1. Run: `systemctl edit lat`
2. Insert the following

```
[Service]
Environment=JAVA_HOME="/PATH_TO_JAVA_HOME"
```

### 3. Configure

Modify the [LAT Source Configuration](#) and [LAT Log4J Configuration](#) (see [LAT Packaging and Installation](#) for file locations).

It is recommended to use the .conf file for most LAT configuration. However, `LAT_MEMORY` and `LAT_MEMORY_DIRECT` can only be set by using environment variable. For these use the same process as setting `JAVA_HOME` as detailed in step 2.

### 4. Start/Stop

Start: `systemctl start lat`

Stop: `systemctl stop lat`

### Package Contents

- Jar: `/opt/lat/lat-${VERSION}.jar`
- Service File: `/usr/lib/systemd/system/lat.service`
- Configuration: `/etc/lat/`
  - Server Configuration: `lat.conf`
  - Logging Configuration: `log4j2.xml`
- Logs: `/var/log/lat/lat.log`
- LAT data directory: `/opt/lat/.lat-data`

## Tar Folder

### 1. Extract file in your preferred location

Run: `tar -xzf lat-$VERSION.tar.gz`

### 2. Setup Java

If the default system Java is version 17 or greater, no action is required. If not, set the `$JAVA_HOME` environment variable to a Java installation with version 17 or greater:

```
export JAVA_HOME="PATH_TO_JAVA_HOME"
```

### 3. Configure

Modify the [LAT Source Configuration](#) and [LAT Log4J Configuration](#) (see [LAT Packaging and Installation](#) for file locations).

It is recommended to use the `.conf` file for most LAT configuration. However, `LAT_MEMORY` and `LAT_MEMORY_DIRECT` can only be set using environment variable. For these use the same process as setting `JAVA_HOME` as detailed in step 2.

### 4. Start/Stop

Start: `./lat.sh [path]`, where `path` is an optional parameter for the path to a server configuration file (e.g., `lat.conf`). If `path` is not present, it will default to `config/lat.conf`.

Stop: Stop the running `lat.sh` script with `ctrl-c`.

## Folder Contents

- Jar: `lat-$VERSION.jar`
- Configuration: `config/`
  - Server Configuration: `lat.conf`
  - Logging Configuration: `log4j2.xml`
- Start Script: `lat.sh`
- LAT data directory: `.lat-data`

## Jar Direct

The LAT is a Java jar that can be deployed on any machine running Java 17+.

To run the LAT use the following command line statement:

```
java [options] -Dlog4j.configurationFile=<log4j-config-path> -jar <lat-jar-path> [lat-config-path]
```

- `log4j-config-path`: REQUIRED : path to the log4j configuration file
- `lat-jar-path`: REQUIRED : path to the LAT jar
- `lat-config-path`: OPTIONAL : path to a LAT service configuration file

## Recommended Java Command Line Options

See the [Java command line documentation](#) for options.

## Max Direct Memory Size

|              |                             |
|--------------|-----------------------------|
| Option:      | -XX:MaxDirectMemorySize     |
| Recommended: | 32G                         |
| Example:     | -XX:MaxDirectMemorySize=32G |

## Minimum Heap Size

|              |          |
|--------------|----------|
| Option:      | -Xms     |
| Recommended: | 32G      |
| Example:     | -Xms=32G |

## Maximum Heap Size

|              |          |
|--------------|----------|
| Option:      | -Xmx     |
| Recommended: | 32G      |
| Example:     | -Xmx=32G |

## Example Java Command Line

```
java \
  -Dlog4j.configurationFile=config/log4j2.xml \
  -XX:MaxDirectMemorySize=32G \
  -Xms32G \
  -Xmx32G \
  -jar lat-[VERSION NUMBER].jar \
  config/lat.conf
```

## Docker Container

Ocient provides a Docker container for the LAT for testing workloads.

The container is not recommended for production workloads and should be used for testing only.

## Configuration

The only mode of configuring the LAT within a Docker container is by using the service configuration environment variables. A `lat.conf` file is not supported.

See [LAT Service Configuration](#) for the list of supported environment variables.

It is not recommended to override the following environment variables.

- LAT\_DATA\_PATH
- LAT\_API\_PORT

## Logging

Within the Docker container the LAT logs `stdout`. The user can use the `docker logs ...` command to view the LAT logs.

The following environment variables are available to modify the log level of various loggers. See [LAT Log4J Configuration](#) for descriptions of the loggers and log levels.

## ROOT\_LOG\_LEVEL

|          |       |
|----------|-------|
| Logger:  | root  |
| Default: | debug |

**BINDERS\_LOG\_LEVEL**

|          |                                           |
|----------|-------------------------------------------|
| Logger:  | com.ocient.lat.sink.ocient.Binders.Binder |
| Default: | info                                      |

**HTTP\_LOG\_LEVEL**

|          |                                      |
|----------|--------------------------------------|
| Logger:  | org.eclipse.jetty.server.HttpChannel |
| Default: | debug                                |

**KAFKA\_LOG\_LEVEL**

|          |                  |
|----------|------------------|
| Logger:  | org.apache.kafka |
| Default: | info             |

**Java**

The `JAVA_OPTS` environment variable is available for controlling JVM® settings. Container defaults to default JVM settings.

See the [Java command line documentation](#) for detail on the command line options.

Example:

```
JAVA_OPTS="-XX:MaxDirectMemorySize=32G -Xms32G -Xmx32G"
```

**Data Volumes**

The LAT in the docker container writes persistent data to `/lat/data`. It is recommended to mount a volume to this path so that the data can outlive the container instance. The [LAT Advanced Topics](#) is also written to this path if configured. If viewing this error file is necessary, a volume must be used.

Do not use the `LAT_DATA_PATH` configuration to override the data path inside the docker container. It will cause the LAT to write data and error logs into different locations and will require two volumes.

If using the Local File Source, the user must mount the data directory as a bind mount inside the container. It is recommended to mount to a location under the `/lat/` directory. When configuring the local file source the bind mount path should be configured with a file group.

**Ports**

By default the LAT API listens on port 8080. It is not recommended to override the `LAT_API_PORT` environment variable and instead use Docker's port mapping functionality.

**Examples****Docker**

```
docker run --rm --name lat \
  --env ROOT_LOG_LEVEL=info \
  --env JAVA_OPTS="-XX:MaxDirectMemorySize=32G -Xms32G -Xmx32G" \
  -p 8080:8080 \
  --mount source=lat-data,target=/lat/data \
  lat:2.0.0
```

## Docker Compose

```
---
version: '3.9'
services:
  lat:
    image: lat:2.0.0
    ports:
      - "8080:8080"
    restart: on-failure
    volumes:
      - lat-data:/lat/data
    environment:
      JAVA_OPTS: "-XX:MaxDirectMemorySize=32G -Xms32G -Xmx32G"
      ROOT_LOG_LEVEL: info
volumes:
  lat-data:
```

## Related Links

[Install an Ocient System](#)

[Ingest Data with Legacy LAT Reference](#)

## 12.10 LAT Advanced Topics

Data Pipelines are now the preferred method for loading data into the Ocient System. For details, see [Load Data](#).

### Addition of an LAT Instance to the Loading Path

Use these steps to add an LAT instance to the loading path:

1. Stop the LAT process on all Loader Nodes using `sudo systemctl stop lat`.
2. Copy `/opt/lat/.lat-data/pipeline/pipeline.json` from one of the Loader Nodes to the new node.
3. On the new node, execute this command: `sudo chown lat:lat /opt/lat/.lat-data/pipeline/pipeline.json`
4. On the new node, execute this command: `sudo chmod 644 /opt/lat/.lat-data/pipeline/pipeline.json`
5. Start the LAT process on all Loader Nodes using `sudo systemctl start lat`.

### Removal of an LAT Instance from the Loading Path

Use these steps to remove an LAT instance from the loading path:

1. Stop the LAT process on all Loader Nodes using `sudo systemctl stop lat`.
2. Delete the `/opt/lat/.lat-data/pipeline/pipeline.json` from the chosen node.
3. Start the LAT process on all Loader Nodes using `sudo systemctl start lat`.
4. For file loading, rebalance the load across the remaining nodes. For details, see the `pipeline rebalance` command in [LAT Client Command Line Interface](#).

For a load using Apache® Kafka®, shut down the service on the chosen node and delete the configuration.

### Dynamic Schema Changes

The loading system in Ocient® is designed to support dynamic schema changes. The LAT allows data to load continuously even while database tables are altered (e.g., ADD COLUMN, DROP COLUMN). In this way dynamic changes can occur on the database and the pipeline can be updated later to begin streaming the revised data through the pipeline into the revised tables. The typical flow for updating a table is to first execute a DDL Command such

as “ALTER TABLE ... ADD COLUMN ...” and then when it is completed to execute the update on the LAT Pipeline to include (or remove) the altered column in the transformation.

## Error Sink

The LAT provides 2 ways to view loading, transformation, or binding errors that occur during a pipeline.

### Kafka Error Topic

If an Apache® Kafka® source is used, failed records can optionally be routed to a configurable [LAT Pipeline Configuration](#). This topic will contain the records that fail and their failure reason/exception. The topic can be used to gain insight into why a given record failed to load. Each entry in the topic will contain the following:

- Value[byte[][]]: A byte array containing the record itself.
- Headers:
  - topic: The topic from which the record originated.
  - partition: The partition from which the record originated.
  - offset: The partition offset from which the record originated.
  - state: The state of the record (the location where the record encountered an error).
  - exception: The exception associated with the error
  - exception\_message: The exception message, if it exists.

### Error Log File

If you use a File source, or if the Error Topic configuration is not set, the LAT sends errors to a dedicated error log file. If you use the default log4j2 configuration, this error log can be found alongside the rest of the LAT logs in `error.log` and accessed using the LAT Client with the [LAT Client Command Line Interface](#).

If you use a custom log4j2 configuration, your appender configuration should look similar to this code.

```
<RollingRandomAccessFile name="ErrorLog" append="true" fileName="logs/error.log" filePattern="logs/error.%d{MM-dd-yyyy}-%i.log.gz" />
  <PatternLayout pattern="%m%n" />
  <Policies>
    <SizeBasedTriggeringPolicy size="100 MB" />
  </Policies>
  <DefaultRolloverStrategy fileIndex="nomax">
    <Delete basePath="logs">
      <IfFileName glob="error.*.log.gz" />
      <IfAccumulatedFileSize exceeds="20 GB" />
    </Delete>
  </DefaultRolloverStrategy>
</RollingRandomAccessFile>
```

If the [LAT Service Configuration](#) service configuration and the [LAT Pipeline Configuration](#) pipeline configuration are both set to `true`, the error log file includes a JSON representation of the record that caused an error.

If you use a custom log4j2 configuration, you must still use a `RollingRandomAccessFileAppender` named `ErrorLog`. Also, `PatternLayout` must still be `%m%n`, because the errors API can retrieve errors. If you do not use `RollingRandomAccessFileAppender`, the LAT does not start. If `PatternLayout` is not correct, the errors API does not work.

## Understanding Deduplication

The LAT loads rows from data sources in an exactly-once fashion. This is made possible by row level deduplication for LAT pipelines and the ability to replay records from a source. In short, if the same records are replayed through the LAT, there are specific scenarios that guarantee that no duplicate records will be persisted into the Ocient tables.

For a deeper understanding this works, a few key concepts are explained here:

- Partitioning Data
- The Durability Horizon
- Deduplication Scope

## Partitioning Data

To deliver high throughput loading, the LAT partitions the data source into independent sets of data. These are then loaded in parallel across all LAT instances. Each partition is considered a well-ordered sequence of rows that is replayable.

Some sources like Kafka natively support the concept of partitions and have a native record ID as part of their protocol.

Others, like a batch of files from Amazon® Web Services<sup>SM</sup> (AWS<sup>SM</sup>) S3, require LAT to partition the data on its own and assign a record ID. For file loading, LAT establishes a record ID based on the sorted list of files in the file group and the row of each record within the files.

Altering the list of files in the target directory on the source system can change the record ID. This will impair the ability of the LAT to properly deduplicate rows if a file loading pipeline is stopped and restarted.

## The Durability Horizon

Within a partition, each record is assigned a unique record ID. This ID is monotonically increasing within a partition. As data is loaded through the LAT and into Ocident's page stores, this data is said to become "Durable" meaning that in the event of a node shutdown, the data would be preserved on non-volatile storage. At this point the record is no longer in memory, but stored on disk in a redundant fashion. The "Durability Horizon" is the largest record ID that has become Durable on each partition of data. If a previously loaded record were replayed it is recognized as a duplicate of the original record and ignored. New records are loaded and the Durability Horizon is increased.

## Deduplication Scope

Deduplication is constrained based on a few settings in the LAT pipeline configuration. Each pipeline ID is considered an independent loading task. Additionally, each topic or file group is considered an independent data set. As a result, deduplication does not apply between different pipelines or different topics and file groups even if they are loading the same underlying files.

Records are deduplicated when all of the following are true:

1. The pipeline ID matches
2. The topic name or file group name matches

When this is true, any record with a record ID less than or equal to the current Durability Horizon will be considered a duplicate and ignored. When the record ID has progressed higher than the Durability Horizon, then new data will begin loading into the database.

If no `pipeline_id` is set, the LAT Client will automatically assign a randomly generated ID, resulting in no deduplication across different pipelines. In the event that you want deduplication between multiple pipelines, the `pipeline_id` should be copied from the previous pipeline and explicitly set. The topic/file groups must also be the same to ensure deduplication.

When updating an existing file load pipeline to select a different group of files by altering start/stop time or another filter, be sure to use a new pipeline ID (best accomplished by avoiding setting an explicit pipeline ID when creating the pipeline with the client). Otherwise, unexpected results can occur such as rows in a new file being considered a duplicate.

## V1 → V2 Migration

LAT V2 modified the way the deduplication scope is calculated. In V1 only the topic name was used for scope calculation. Therefore, to maintain deduplication in a pipeline that is being upgraded from V1 to V2 **must be explicitly set to (the empty string)** upon creation.

## Related Links

[Ingest Data with Legacy LAT Reference](#)

## 12.11 LAT Client Command Line Interface

Data Pipelines are now the preferred method for loading data into the Ocint System. For details, see [Load Data](#).

The LAT Client can be used to interact with a running LAT instance. It supports subcommands for interacting with pipelines, and previewing transformations.

The LAT Client is distributed in the form of a Python® wheel file. Contact Ocint® Support for the wheel that corresponds to the LAT version.

## Prerequisites

- Python >= 3.8
- pip3 >= 20.2.3
  - If `pip install` fails, try upgrading `pip`.
- Wheel Python package
  - `pip install wheel`

## Install

It is recommended to install the wheel in a Python virtual environment to avoid conflicts with globally installed Python packages. For the install command, replace `$VERSION` with the latest version of the LAT client.

Steps:

1. Create the virtual environment. `python3 -m venv venv`
2. Activate the virtual environment. `source venv/bin/activate`
3. Install the wheel. `pip install lat_client-$VERSION-py3-none-any.whl`
4. Run commands with `lat_client COMMAND ARGS`
5. When a new terminal is opened, repeat step 2 to activate the environment and gain access to the `lat_client` command.

## Usage

Get help on the command line:

```
lat_client --help
lat_client <subcommand> --help
```

## Common Arguments

Some arguments are available on all subcommands. For convenience, most of them can also be set using an environment variable.

`--no-verify`

Skip certificate validation when connecting to LAT. Ignored if using http scheme



---



---

Example: `--no-verify`


---



---

**--hosts (LAT\_HOSTS)**

One or more LAT hosts to orchestrate. Valid domain names or IP addresses can be used.

---

|              |                                                                               |
|--------------|-------------------------------------------------------------------------------|
| Example:     | <code>--hosts http://192.168.0.1:8080<br/>http://192.168.0.2:8081</code>      |
| Environment: | <code>export<br/>LAT_HOSTS="http://10.4.0.1:8080,http://10.4.0.2:8081"</code> |

---

**--oauth-domain (LAT\_OAUTH\_DOMAIN)**

OKTA® OAuth domain to use for token acquisition.

---

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Example:     | <code>--oauth-domain https://dev-12345678.okta.com</code>                |
| Environment: | <code>export<br/>LAT_OAUTH_DOMAIN="https://dev-12345678.okta.com"</code> |

---

**--oauth-server (LAT\_OAUTH\_SERVER)**

Okta OAuth authorization server to use for token acquisition.

---

|              |                                                            |
|--------------|------------------------------------------------------------|
| Example:     | <code>--oauth-server abcdef000ghijklm111</code>            |
| Environment: | <code>export LAT_OAUTH_SERVER="abcdef000ghijklm111"</code> |

---

**--client-id (LAT\_CLIENT\_ID)**

Okta client id to use for token acquisition.

---

|              |                                              |
|--------------|----------------------------------------------|
| Example:     | <code>--client-id 12345678</code>            |
| Environment: | <code>export LAT_CLIENT_ID="12345678"</code> |

---

**--client-secret (LAT\_CLIENT\_SECRET)**

Okta client secret to use for token acquisition.

---

|              |                                                |
|--------------|------------------------------------------------|
| Example:     | <code>--client-secret abc123</code>            |
| Environment: | <code>export LAT_CLIENT_SECRET="abc123"</code> |

---

**--oauth-http-proxy (LAT\_OAUTH\_HTTP\_PROXY)**

HTTP proxy URL to use for token acquisition. Authentication credentials can be passed in proxy URL.

---

|              |                                                                                |
|--------------|--------------------------------------------------------------------------------|
| Example:     | <code>--oauth-http-proxy<br/>http://user:pass@some.proxy.com</code>            |
| Environment: | <code>export<br/>LAT_OAUTH_HTTP_PROXY="http://user:pass@some.proxy.com"</code> |

---

## Subcommands

### pipeline create

Create a new pipeline.

For most use cases, it is advisable to leave the `pipeline_id` unset when creating a pipeline. The client will set it to a random UUID to prevent deduplication across different pipelines. In the event that you want deduplication between pipelines, the `pipeline_id` should be copied from the previous pipeline and included in the new pipeline. The [LAT Transform Configuration](#) must also be the same to ensure deduplication.

When the client is used to create a pipeline with a file source, the client will make adjustments to the source configuration such that partitions are assigned evenly across nodes.

First, the client will get the number of workers from the pipeline. If one is not set, it will use the minimum configured `lat.default.workers` instead. Then, it will set `partitions = workers * num_nodes`. Finally, it will set `partitions_assigned = [workers * node_index, workers * (node_index + 1) - 1]` for each node.

|                   |                                                                                    |
|-------------------|------------------------------------------------------------------------------------|
| <b>Arguments:</b> | <code>--pipeline:</code> path to the pipeline configuration .json file             |
| <b>Example:</b>   | <code>lat_client pipeline create --pipeline /home/user/my_new_pipeline.json</code> |

### pipeline get

Get the configuration for an existing pipeline.

If the configured pipelines are identical, print the pipeline, otherwise an explanation of the inconsistency will be provided.

For pipelines with a file source, `partitions_assigned` is ignored when checking if pipelines are identical. Additionally, the client will validate that all partitions are assigned, and that no partition is assigned more than once.

|                                               |
|-----------------------------------------------|
| Example: <code>lat_client pipeline get</code> |
|-----------------------------------------------|

### pipeline update

Update the configuration for an existing pipeline.

The new pipeline can only make changes to subfields in `transform`, *except* for any `topic` / `file_group` names. All other subfields of `transform` are allowed to change, including to the `table` and `column` fields.

For pipelines with a file source, partition assignments will be copied from the existing pipeline.

If the pipeline was running prior to the update, successful completion of this command will automatically restart the pipeline.

|                   |                                                                                    |
|-------------------|------------------------------------------------------------------------------------|
| <b>Arguments:</b> | <code>--pipeline:</code> path to the pipeline configuration .json file             |
| <b>Example:</b>   | <code>lat_client pipeline update --pipeline /home/user/my_new_pipeline.json</code> |

### pipeline delete

Delete an existing pipeline.

Unless the `--force` flag is used, a pipeline must be stopped, or deletion will fail.

|                   |                                                                                                                                                           |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Arguments:</b> | <b>---force:</b> Force a pipeline to delete regardless of running status<br><b>--skip-validation:</b> Delete a pipeline regardless of cluster consistency |
| <b>Example:</b>   | <code>lat_client pipeline delete --skip-validation</code>                                                                                                 |

**pipeline start**

Start the configured pipeline.

Prior to starting the pipeline, the pipeline start subcommand will validate that all specified hosts are configured with an identical pipeline, except for pipelines with a file source, which must have different `partitions_assigned` such that each partition is assigned exactly once across all hosts.

---

Example: `lat_client pipeline start`

---

**pipeline stop**

Stop the configured pipeline.

---

Example: `lat_client pipeline stop`

---

**pipeline status**

Retrieve status of the pipeline. The valid pipeline statuses are **STOPPED**, **RUNNING**, **COMPLETED**, and **FAILED**. When the pipeline is **FAILED**, the file statuses will remain in processing.

|                   |                                                                                                                                                                                                                                                    |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Arguments:</b> | <b>--list-files:</b> Lists selected files in their sorted order for each file group, along with file statuses ( <code>completed,processing,not_started</code> ). Output is summarized to be human readable. The system truncates large file lists. |
| <b>Arguments:</b> | <b>--list-all-files:</b> Lists all selected files in their sorted order for each file group, along with file statuses ( <code>completed,processing,not_started</code> ).                                                                           |
| <b>Example:</b>   | <code>lat_client pipeline status</code>                                                                                                                                                                                                            |

**pipeline errors**

Retrieve errors that occur while the current pipeline runs.

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                   | ---json: output errors as lines of JSON rather than in the default human-readable format<br>---max-errorsMAX_ERRORS: an upper limit on the number of errors to retrieve (default is 100)<br>---only-records: only show records (not error messages or other information)<br>---only-error-messages: only show error messages (not records or other information)<br>---no-records: show all information except records |
| <b>Arguments:</b> |                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Example:</b>   | <pre>lat_client pipeline errors --no-records --max-errors 10</pre>                                                                                                                                                                                                                                                                                                                                                    |

### pipeline rebalance

Rebalances partitions evenly to all provided LAT Nodes.

This subcommand only applies to pipelines running file sources.

This command is meant to be used in the case of a node outage during a file load. LAT file loading does not support automatic partition re-balancing and manual intervention is required. The flow is as follows:

1. LAT Node goes offline.
2. LAT operator rebalances the partitions from the offline node onto the online nodes using the client. The operator should use the rebalance command and omit the offline node from the hosts argument.
3. LAT Node comes back online.
4. LAT operator rebalances partitions using the client to include all nodes including the newly online node. The operator should use the rebalance command and include all online nodes in the hosts argument.

---

Example: `lat_client pipeline rebalance`

---

### sink create

Create a new sink configuration.

The sink configuration file for this subcommand should match the same format as the [LAT Sink Configuration](#). For example:

```
{ "type": "ocient", "remotes": ["1.2.3.4:5050"] }
```

|                   |                                                                                                                                       |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------|
|                   | ---sink: path to the sink configuration .json file<br>---name: Name of the sink to create<br>---default: set this as the default sink |
| <b>Arguments:</b> |                                                                                                                                       |
| <b>Example:</b>   | <pre>lat_client sink create --sink /home/user/my_sink_config.json --name my-sink-name-1 --default</pre>                               |

### sink delete

Delete a sink configuration. The sink configuration must not be part of a created or running pipeline.

---

|                   |                                                           |
|-------------------|-----------------------------------------------------------|
| <b>Arguments:</b> | <code>--name</code> : Name of the sink to delete          |
| <b>Example:</b>   | <code>lat_client sink delete --name my-sink-name-1</code> |

---

**sink list**

List all configured sinks.

---

Example: `lat_client sink list`

---

**sink get**

Get a sink configuration by id.

---

|                   |                                                                  |
|-------------------|------------------------------------------------------------------|
| <b>Arguments:</b> | <code>--name</code> : Name of the sink to get configuration for. |
| <b>Example:</b>   | <code>lat_client sink get --name my-sink-name-1</code>           |

---

**preview**

Preview a transformation.

At most one of `--transform` or `--pipeline` can be provided. If neither is provided, the host will attempt to use the transformation configured in its pipeline.

You can specify the `--extract` or `--pipeline` option. If you specify none of these options, the host uses the JSON Extractor by default. For details about record and extractor types, see the [LAT Extract Configuration](#).

You must specify the `--topic` or `--file-group` option, which should match the `topic` or `file_group` key in the specified `transform` section.

---

|                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                  | <code>---topic</code> name of the topic the records are associated with.<br><code>---file-group</code> name of the file group the records are associated with.<br><code>---records</code> path to a file of records to transform. Record formats can be of Delimited Records (e.g., CSV, TSV), JSON Records, or Fixed Width Binary Records.<br><code>---extract</code> <a href="#">Optional</a> path to a .json file containing the extract section of a pipeline definition to use for extraction.<br><code>---transform</code> <a href="#">Optional</a> path to a .json file containing the transform section of a pipeline definition to use for transformation.<br><code>---pipeline</code> <a href="#">Optional</a> path to a .json file containing a pipeline to use for transformation and extraction, if present. |
| <b>Arguments:</b>                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Example without pipeline:</b> | <code>lat_client preview --topic test_topic</code><br><code>--records ./data/my_records --extract</code><br><code>/home/user/my_extract.json --transform</code><br><code>/home/user/my_transform.json</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Example using pipeline:</b>   | <code>lat_client preview --topic test_topic</code><br><code>--records ./data/my_records --pipeline</code><br><code>/home/user/my_pipeline.json</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

---

**Common Workflows**

**Check on Status of LAT Pipelines**

```
$ lat_client pipeline status
10.4.0.1:8080: STOPPED
10.4.0.2:8080: STOPPED
10.4.0.3:8080: STOPPED

$ lat_client pipeline status --hosts https://10.4.0.1:8080
10.4.0.1:8080: STOPPED
```

**Updating an Existing LAT Pipeline**

```
lat_client pipeline update --pipeline /home/user/my_new_pipeline.json
```

If the pipeline was running prior to the update, successful completion of this command will automatically restart the pipeline.

If unsuccessful, the CLI will report an error with an explanation of what is wrong with the command. Common issues are invalid JSON, missing a required column. An unsuccessful update of the pipeline config does not impact actively running pipelines.

**Restart the Pipeline**

```
# First, stop the running pipelines. To send custom hosts, use --hosts IP1:port IP2:port
$ lat_client pipeline stop
Stopped: 10.4.0.1:8080, 10.4.0.2:8080, 10.4.0.3:8080

# Then, start the pipelines.
$ lat_client pipeline start
Started:
10.4.0.1:8080, 10.4.0.2:8080, 10.4.0.3:8080
```

**Check Multiple LAT Nodes to See If the Pipeline Configurations Are Compatible**

```
# hosts with compatible pipelines
$ lat_client pipeline get --hosts https://10.4.0.1:8080 https://10.4.0.2:8080 https://10.4.0.3:8080
{
  "pipeline_id": "91326229-5fc6-4542-99ce-87bdc00a978",
  "version": 2,
  "source": { ... },
  "sink": { ... },
  "transform": { ... }
}

# hosts with incompatible pipeline
$ lat_client pipeline get --hosts https://10.4.0.1:8080 https://10.4.0.2:8080 https://10.4.0.3:8080
Hosts responded with incompatible pipelines:
  b5a6e8647cd3a28d4a5bd07cd1c491ce: 10.4.0.1:8080, 10.4.0.2:8080
  160f270e39d7a51a431c841613e4dc4e: 10.4.0.3:8080
```

The CLI will compare the MD5 hash of the pipeline configurations on all nodes and respond that all pipelines match or are inconsistent.

**LAT Client Command Line Interface Troubleshooting**

If the certificate authorities on the system running the Python client (LAT Client) need to be updated, an error can occur. This type of error message might appear.

```
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get lo
```

You can run this command to resolve the error.

```
pip install --upgrade certifi
```

The root cause of this error can be a connection to either the LAT Server over SSL or an authentication to Okta to obtain an access token over SSL. You can run the same command in both cases to resolve the issue.

When the SSL certificate is self-signed on the LAT Server, you can use the `--no-verify` flag when you connect to the LAT Server without verifying the SSL Certificate.

## Related Links

[Ingest Data with Legacy LAT Reference](#)

[Install an Ocient System](#)

### 12.11.1 Install LAT Client in a Disconnected Network

Data Pipelines are now the preferred method for loading data into the Ocient System. For details, see [Load Data](#).

Operate the LAT Client in a disconnected network using these steps to install the LAT Client.

## Requirements

On the Red Hat® Enterprise Linux® (RHEL®) version 8.0, the Python® version 3.6.8 specifies the default `python3` requirement and the LAT Client requires Python version 3.8 or greater.

Contact Ocient® Support for the LAT Client Python wheel file.

Install the LAT Client Python wheel by first gathering the RPMs for Python 3.11 on a system with access to the appropriate software repository `python3.11` and its dependencies:

- `mpdecimal`
- `python3.11-libs`
- `python3.11-pip-wheel`
- `python3.11-setuptools-wheel`

Use your regular path for the Python packages.

Gather the `lat_client` and its dependencies (`.whl` files). Execute this download command on a machine with Python 3.11 and `pip` installed.

```
$ pip download lat_client-3.1.2-py3-none-any.whl
```

This command downloads these dependencies.

```
lat_client
  aiohttp==3.9.3
    aiosignal==1.3.1
      frozenlist==1.4.1
    attrs==23.2.0
    frozenlist==1.4.1
    multidict==6.0.5
    yarl==1.9.4
      idna==3.6
      multidict==6.0.5
    more-itertools==10.2.0
```

```

pydantic==2.6.4
  annotated-types==0.6.0
  pydantic_core==2.16.3
    typing_extensions==4.10.0
  typing_extensions==4.10.0
rich==13.7.1
  markdown-it-py==3.0.0
    mdurl==0.1.2
  Pygments==2.17.2
wheel==0.43.0

```

Copy the RPMs and wheel files to the machine in the disconnected network.

### Install the LAT Client

Install Python 3.11 on the machine in the disconnected network where `<python3.11 RPM>` is the path to the Python 3.11 RPM. `<mpdecimal rpm>` is the path to a dependency of Python 3.11. You can list more dependencies in this way using spaces between each one.

```
dnf install <python3.11 RPM> <mpdecimal rpm>
```

After Python 3.11 is installed, create a `venv` to install the LAT Client and its dependencies.

```
$ /usr/bin/python3.11 -m venv venv
```

Enter the `venv`.

```
$ source venv/bin/activate
```

Install `lat_client` and its dependencies in the `venv`. Specify wheel files `<wheels>` as paths to wheel files separated by spaces.

```
(venv)$ pip install lat_client-3.1.2-py3-none-any.whl <wheels>
```

Check that the LAT Client runs.

```
(venv)$ lat_client --version
3.1.2
```

### Related Links

[LAT Client Command Line Interface](#)



## 13 Version Compatibility

### Recommended JDBC Driver Versions

This table describes the recommended versions of the JDBC driver and Java® for each Ocient® System release.

| Ocient System Version      | JDBC Driver Version | Java Version |
|----------------------------|---------------------|--------------|
| 25.2                       | 3.5.1               | 8 or later   |
| 25.1                       | 3.4.1               | 8 or later   |
| 25.0                       | 3.3.1               | 8 or later   |
| 24.3 and 24.4              | 3.2.x               | 11 or later  |
| 23.1, 24.0, 24.1, and 24.2 | 3.1.7               | 8 or later   |
| 23.0                       | 3.0                 | 8 or later   |

JDBC versions 3.2.x are not compatible with Java 8. Version 3.3.0 is compatible.

Ocient versions 22.0 and earlier might experience tail latency after submitting `FETCH_DATA` requests using JDBC driver version 3.0.0.

To reduce this latency, upgrade to Ocient version 23.0 or later.

### JDBC API Compatibility

This table describes the specifications for each version of the JDBC API and whether it is supported by the latest Ocient JDBC driver.

| JDBC API Version | Requirement                                                                                                                       | Support as of JDBC 2.93 |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| 4.3              | Enhanced DatabaseMetaData to determine if Sharding is supported                                                                   | Supported               |
| 4.3              | Added the method drivers to DriverManager to return a Stream of the currently loaded and available JDBC drivers                   | Supported               |
| 4.3              | Added support to Statement for enquoting literals and simple identifiers                                                          | Supported               |
| 4.3              | Clarified the Java SE version that methods were deprecated                                                                        | Supported               |
| 4.3              | Added Sharding support                                                                                                            | Not Supported           |
| 4.3              | Enhanced Connection to be able to provide hints to the driver that a request, an independent unit of work, is beginning or ending | Not Supported           |
| 4.2              | Added JDBCType enum and SQLType interface                                                                                         | Supported               |
| 4.2              | Added support for large update counts                                                                                             | Supported               |
| 4.2              | DatabaseMetaData methods to return maximum Logical LOB size and if Ref Cursors are supported                                      | Not Supported           |
| 4.2              | Support for REF CURSORS in CallableStatement                                                                                      | Not Supported           |

| JDBC API Version | Requirement                                                                                                                                                                               | Support as of JDBC 2.93 |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| 4.1              | Allow Connection, ResultSet and Statement objects to be used with the try-with-resources statement                                                                                        | Supported               |
| 4.1              | Support added to CallableStatement and ResultSet to specify the Java type to convert to using the getObject method                                                                        | Supported               |
| 4.1              | Added support to Connection to specify a database schema, abort and timeout a physical connection.                                                                                        | Supported               |
| 4.1              | DatabaseMetaData methods to return PseudoColumns and if a generated key is always returned                                                                                                | Not Supported           |
| 4.1              | Added support to close a Statement object when its dependent objects have been closed                                                                                                     | Not Supported           |
| 4.1              | Support for obtaining the parent logger for a Driver, DataSource, ConnectionPoolDataSource and XADataSource                                                                               | Not Supported           |
| 4.0              | auto java.sql.Driver discovery – no longer need to load a java.sql.Driver class using Class.forName                                                                                       | Not Supported           |
| 4.0              | National Character Set support added                                                                                                                                                      | Not Supported           |
| 4.0              | Support added for the SQL:2003 XML data type                                                                                                                                              | Not Supported           |
| 4.0              | SQLException enhancements – Added support for cause chaining; New SQLExceptions added for common SQLState class value codes                                                               | Not Supported           |
| 4.0              | Enhanced Blob/Clob functionality – Support provided to create and free a Blob/Clob instance as well as additional methods added to improve accessibility                                  | Not Supported           |
| 4.0              | Support added for accessing a SQL ROWID                                                                                                                                                   | Not Supported           |
| 4.0              | Support added to allow a JDBC application to access an instance of a JDBC resource that has been wrapped by a vendor, usually in an application server or connection pooling environment. | Not Supported           |
| 4.0              | Availability to be notified when a PreparedStatement that is associated with a PooledConnection has been closed or the driver determines is invalid                                       | Not Supported           |
| 3.0              | Pooled statements – reuse of statements associated with a pooled connection                                                                                                               | Supported               |

| JDBC API Version | Requirement                                                                                                                                                                                                             | Support as of JDBC 2.93 |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| 3.0              | Savepoints – allow a transaction to be rolled back to a designated savepoint                                                                                                                                            | Not Supported           |
| 3.0              | Properties defined for ConnectionPoolDataSource – specify how connections are to be pooled                                                                                                                              | Not Supported           |
| 3.0              | Metadata for parameters of a PreparedStatement object                                                                                                                                                                   | Not Supported           |
| 3.0              | Ability to retrieve values from automatically generated columns                                                                                                                                                         | Not Supported           |
| 3.0              | Ability to have multiple ResultSet objects returned from CallableStatement objects open at the same time                                                                                                                | Not Supported           |
| 3.0              | Ability to identify parameters to CallableStatement objects by name as well as by index                                                                                                                                 | Not Supported           |
| 3.0              | ResultSet holdability – ability to specify whether cursors should be held open or closed at the end of a transaction                                                                                                    | Not Supported           |
| 3.0              | Ability to retrieve and update the SQL structured type instance that a Ref object references                                                                                                                            | Not Supported           |
| 3.0              | Ability to programmatically update BLOB, CLOB, ARRAY, and REF values.                                                                                                                                                   | Not Supported           |
| 3.0              | Addition of the java.sql.Types.DATALINK data type – allows JDBC drivers access to objects stored outside a data source                                                                                                  | Not Supported           |
| 3.0              | Addition of metadata for retrieving SQL type hierarchies                                                                                                                                                                | Not Supported           |
| 2.1              | Scrollable result sets—using new methods in the ResultSet interface that allow the cursor to be moved to a particular row or to a position relative to its current position                                             | Not Supported           |
| 2.1              | Batch updates                                                                                                                                                                                                           | Not Supported           |
| 2.1              | Programmatic updates—using ResultSet updater methods                                                                                                                                                                    | Not Supported           |
| 2.1              | New data types—interfaces mapping the SQL3 data types                                                                                                                                                                   | Not Supported           |
| 2.1              | Custom mapping of user-defined types (UDTs)                                                                                                                                                                             | Not Supported           |
| 2.1              | Miscellaneous features, including performance hints, the use of character streams, full precision for java.math.BigDecimal values, additional security, and support for time zones in date, time, and timestamp values. | Not Supported           |
| 2.0              | Pooled connections – allowing connections to be used and reused                                                                                                                                                         | Supported               |

| JDBC API Version | Requirement                                                                                                                                                            | Support as of JDBC 2.93 |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| 2.0              | Distributed transactions – allowing a transaction to span diverse DBMS servers                                                                                         | Supported               |
| 2.0              | The DataSource interface as a means of making a connection. The {{JNDI}} is used for registering a DataSource object with a naming service and also for retrieving it. | Not Supported           |
| 2.0              | RowSet technology – providing a convenient means of handling and passing data                                                                                          | Not Supported           |

### Recommended LAT Versions

This table describes the recommended Ocint Loading and Transformation (LAT) versions for each Ocint System release.

| Ocint System Version | Minimum Version | Recommended Version |
|----------------------|-----------------|---------------------|
| 25.2                 | 3.0             | 3.4.3               |

LAT Version 3.0.0 and greater is only compatible with Version 22.0 and greater of the Ocint System due to a change in the calculations of empty bounding boxes on geospatial data types.

### Related Links

[Release Notes](#)

---

## 14 Release Notes

Release notes are available for the Ocient Hyperscale Data Warehouse™ and the associated clients, JDBC and pyocient. For the corresponding release notes, see these links.

| Products                         | Release Notes                                  |
|----------------------------------|------------------------------------------------|
| Ocient Hyperscale Data Warehouse | Ocient Hyperscale Data Warehouse Release Notes |
| JDBC                             | JDBC Release Notes                             |
| pyocient                         | pyocient Release Notes                         |

### 14.1 Ocient Hyperscale Data Warehouse Release Notes

#### 25.2

- Data Pipeline Changes —
  - Added the **WHERE** filter keyword to the **SELECT** SQL statement in the **CREATE PIPELINE** statement. For details, see [Data Pipelines](#).
  - Added new **HEADERS** Amazon® Web Services<sup>SM</sup> (AWS<sup>SM</sup>) S3 source option for the **CREATE PIPELINE** SQL statement to define the headers to send with every request.
  - Added the **MODE** option to the **PREVIEW PIPELINE** SQL statement. This option indicates whether to perform a validation of the **PREVIEW PIPELINE** statement. For details, see [Data Pipelines](#).
  - Added the `streamloader.extractorEngineParameters.configurationOption.filesystem.access.directories` configuration setting to denote where you can load server file system data. For details, see [Configuration Settings for Data Pipelines](#).
  - Updated the **METADATA** function a new metadata key and syntax. For details, see [Load Metadata Values in Data Pipelines](#):
    - \* The `source_record_id` metadata key indicates a unique source row identifier that is generated by the Ocient System during loading.
    - \* New `METADATA('hive_partition',search_string)` syntax that uses a search string as a key for searching the filename metadata to return the value from the key-value pair as a string.
  - Added these transformation functions. For details, see [Transform Data in Data Pipelines](#):
    - \* **IF** — Evaluate an expression for **true** or **false**.
    - \* **CASE WHEN** — Return a result value based on whether an expression is **true**.
    - \* **ELEMENT\_AT** — Returns the element value of the array or tuple at the specified index.
    - \* **MAP\_KEYS** — Returns the keys in the specified JSON string.
    - \* **MAP\_VALUES** — Returns the values in the specified JSON string.
    - \* **REDUCE** — Applies a merge function to a starting value and all elements in the array, and then reduces the array to a single value.
- System Catalog Table Changes —
  - Added Boolean parameter `scopes_on_refresh` to the `sys.oidc_integrations` system catalog table to include or not include scopes when submitting a refresh token request.

#### 25.1

- New data pipeline transformation function **ARRAY\_CONTAINS** determines whether an array contains the specified value, including NULLs.
- System Catalog Table Changes —
  - The `record_number` column of the `sys.pipeline_errors` system catalog table is now the index, which starts at 1, of the processed record relative to the source file (or Apache® Kafka® partition) where the record was extracted.
  - The `record_offset` column of the `sys.pipeline_errors` is now the offset of 1 for the processed record relative to the start of the source file for files with contiguous records, -1 for files with non-contiguous records (i.e., Apache® Parquet™), and the Kafka offset for Kafka loads.

## 25.0

## Release Highlights

- Load Data into the Ociant® System — The new way to load data into the Ociant System is to use Data Pipeline functionality. This functionality supports loading data from multiple formats such as CSV, JSON, and Apache® Parquet™. Data pipelines support data transformation during the load. You can load multiple data formats, including geospatial data. This functionality supports these new syntaxes:
  - CREATE PIPELINE SQL statement to create a data pipeline.
  - DROP PIPELINE statement to remove a data pipeline.
  - PREVIEW PIPELINE statement to view the results of a data load before creating the data pipeline.
  - START PIPELINE statement to start the execution of the data load.
  - STOP PIPELINE statement to stop the execution of the data load.
  - ALTER PIPELINE RENAME statement to rename a data pipeline.
  - EXPORT PIPELINE statement to return the CREATE PIPELINE SQL statement used to create the specified data pipeline.
  - CREATE OR REPLACE PIPELINE FUNCTION statement to define your own data pipeline function for loading data.
  - DROP PIPELINE FUNCTION statement to remove the data pipeline function.
  - For details, see [Load Data](#) and [Data Pipelines](#).

## Features

- Java® Runtime Environment — The Ociant System installation requires Java® 21, where the `openjdk-21-jre-headless` package is recommended.
- Persistent Data — Data persists in the system storage space of the Ociant System in system catalog tables such as the `sys.completed_queries` table. For details, see [Core Elements of an Ociant System](#).
- Default Compression Scheme — The default compression scheme for variable-length columns is `NONE`.
- Set more granular privileges for database and system objects. For details, see [Object-Type Level Privileges Management](#) and [Data Control Language \(DCL\) Statement Reference](#).
- The `SHOW SYSTEM TABLES` SQL statement displays the system catalog tables.
- System Catalog Table Changes —
  - `information_schema.tables` has a column `table_type` with two new types: `SYSTEM TABLE` for system catalog tables and `SYSTEM VIEW` for information schema tables. `BASE TABLE` and `VIEW` still exist for user-defined tables and views.
  - `sys.system_tables` has new columns: `schema`, `type`, and `description`.
  - `sys.segment_groups` has the new `depth` and `visibility` columns.
  - `sys.stored_segments` has the new `abnormal_placement` column.
  - `sys.completed_queries` contains persistent data. The system experiences a small loading delay between completing the execution of a SQL query and when the corresponding results appear in the table.
  - The definition for the `sys.queries` table has been updated to remove these columns:
    - \* `time_start`
    - \* `time_optimization_start`
    - \* `time_execution_start`
    - \* `time_first_byte_sent`
  - The definition for the `sys.queries` table has been updated to change the data type of these columns from `LONG` to `TIMESTAMP`:
    - \* `timestamp_start`
    - \* `timestamp_optimization_start`
    - \* `timestamp_execution_start`
    - \* `timestamp_first_byte_sent`
  - The definition for the `sys.locks` table has been updated for these columns:
    - \* Renamed `createTime` to `create_time`.
    - \* Renamed `lastRefreshTime` to `last_refresh_time`.
    - \* Renamed `priorityId` to `priority_id`.
  - Updated column descriptions for these tables:
    - \* `sys.segment_directories`

- \* `sys.addendum_directories`
- \* `sys.segment_groups`
- \* `sys.segment_parts`
- \* `sys.segments`
- \* `sys.stored_segments`
- \* `sys.service_role_status`
- Updated the data type for the `segment_group_ids` column in the `sys.segment_group_transfers` table from `ARRAY(LONG)` to `ARRAY(CHAR)` because segment group identifiers are unsigned.
- Renamed the `service_roles_id` column in the `sys.service_role_channel_endpoint` table to `service_role_id` to align with the naming in the `sys.service_role_status` table.

## Version Compatibility

- `sys.built_in_views` has been removed. The built-in views have been removed from `sys.built_in_views`, `sys.views`, and `information_schema.views`. The view information now appears in the `sys.system_tables` system catalog table. `information_schema.views` includes user-defined views and only information schema views.
- `sys.views` no longer has a `view_type` column.

## 24.0

### Release Highlights

- Better SQL Errors: Improved error messages for poorly constructed SQL queries.
- Enhance Build Metadata: Updated Ocident® package file names and build information for clarity.
- Improved Tracing: Enabled using the `TRACE` keyword in SQL queries to profile query performance. For details, see the [General SQL Syntax](#) keyword.
- Large Blobstore Support: Enabled more than 2TiB of data to spill per disk for high-density drive situations.
- Large Drive Support: Added support for drives up to 15.36TB in size per node.
- Rebalance System: Added the `REBALANCE` task, which enables optimization of query efficiency by transferring data around the system until nodes are roughly balanced in terms of data volume per node. For details, see [Expand and Rebalance System](#).
- System Storage Space: Added support for multiple storage spaces and enabled the creation of a system storage space for internal Ocident System data.
- Workload Management Usability:
  - Added support for assigning service classes to queries based on query text. For details, see the [Users, Groups, and Service Classes](#) SQL statement.
  - Added support for changing the priority for queries. For details, see the [Users, Groups, and Service Classes](#) SQL statement.

### Features

- [DB-19266]: Network Configuration —
  - All nodes must now belong to a connectivity pool. Manage connectivity pools using these new SQL statements. For details, see [Cluster and Node Management](#).
    - \* `CREATE CONNECTIVITY_POOL` to create a connectivity pool.
    - \* `DROP CONNECTIVITY_POOL` to drop a connectivity pool.
    - \* `ALTER CONNECTIVITY_POOL SET` to set the metadata of a connectivity pool.
    - \* `ALTER CONNECTIVITY_POOL RENAME TO` to rename a connectivity pool.
    - \* `ALTER CONNECTIVITY_POOL ADD PARTICIPANTS` to add nodes to a connectivity pool.
    - \* `ALTER CONNECTIVITY_POOL DROP PARTICIPANTS` to remove nodes from a connectivity pool.
  - The `ALTER NODE SET ADDRESS` SQL statement changes the internal IP address for a node. For details, see [Cluster and Node Management](#).
  - You can now configure the network of an Ocident System. For details, see [Manage the Network Configuration of an Ocident System](#).
  - Redirects now occur only within connectivity pools. When you upgrade an Ocident System, you must first configure a connectivity pool.
- [DB-28986]: System Catalog Table Updates —

- Renamed `client_version` to `protocol_version` in the `sys.queries` and `sys.completed_queries` system catalog tables.
- Added `driver_version` column to the `sys.queries` and `sys.completed_queries` tables.
- [DB-29788]: Regular Expression Functions —
  - Added new functions that use regular expression search patterns. The new functions are:
    - \* [Character and Binary Functions](#)
    - \* [Character and Binary Functions](#)
    - \* [Character and Binary Functions](#)
    - \* [Character and Binary Functions](#)

### Version Compatibility

- The `bootstrap.conf` no longer supports `highspeedAddress` as an advanced system configuration option. For other options, see [Node Bootstrapping Reference](#).
- The default behavior for the `UNNEST` function no longer uses the `NULL_INPUT` clause for a multi-item `SELECT` list. If you would like to utilize the default behavior from version 23.0 and prior, you may do so by changing `ALTER SYSTEM ALTER CONFIG SET sql.unnestLegacySelectListBehavior = 'true'`.

### Feature Removal

- Removed `COMPRESSION LZ4` from the compression options. This compression scheme remains in use as part of `COMPRESSION DYNAMIC` for variable-length columns.
- Removed the table-valued function `REPLACEMENT_JOIN`. For creating compressed lookup tables, see [Global Dictionary Compression](#).
- Particle swarm optimization functionality has been removed from the Ocint System.
- ODBC connection has been removed from the Ocint System.

## 23.0

### Release Highlights

- All machine learning functionality is available to use. For details, see [Machine Learning Model Functions](#) and [Machine Learning in Ocint](#) to get started.
- Delete Syntax: Enabled the deletion of individual rows in the database.
- Integrations: Added drivers and support for the following third-party applications:
  - DBeaver
  - Tableau®

### Features

- [DB-13607]: Delete Syntax — Added the SQL `DELETE` statement syntax that enables the deletion of individual rows in the database. For details, see [Database, Tables, Views, and Indexes](#).
- [DB-18020]: Large Geospatial Types — Increased the size of `LINESTRING` and `POLYGON` geospatial data types to 512 MB. For details, see [LAT Data Types in Loading](#).
- [DB-19048]: Geospatial Index — Added the `SPATIAL` index type for indexing geospatial data. For details, see [Secondary Indexes](#).
- [DB-18280]: Connectors Refresh — Added integration with DBeaver and Tableau®. For details, see [DBeaver Integration](#) and [Tableau Integration](#).
- [DB-20412]: Multi-Cluster Loading and Cluster of Clusters — Added support for loading and working with multiple clusters. For details, see [Multiple Storage Clusters for Loading Data](#).
- [DB-21609]: Machine Learning Model Updates —
  - Added the [Clustering and Dimension Reduction Models](#).
  - Added the [Other Models](#).
  - Added the [Clustering and Dimension Reduction Models](#).

### Version Compatibility

- Large Geospatial Types are not backwards compatible with earlier releases. For details, see [Version Compatibility](#).



- The database data control language (DCL) denotes user role privileges to remove data using the `DELETE` keyword instead of `TRUNCATE`.

## 22.0

### Release Highlights

- HyperLogLog (HLL): Added HLL sketch functionality.
- Information Schema: Added the `information_schema` schema that shows system metadata.
- Integrations: Added drivers and support for the following third-party applications:
  - Metabase<sup>SM</sup>
  - SQLAlchemy
  - Apache® Superset®

### Features

- [DB-13603]: Information Schema — Added the `information_schema` schema that shows system metadata in an accessible format.
- [DB-20484]: Superset Integration — Merged sqlalchemy-ocient driver into Superset repository, allowing Superset to support Ocient® database connections.
- [DB-20484]: SQLAlchemy Integration — Published sqlalchemy-ocient driver to PyPI.
- [DB-21011]: EXCEPT Clause — Added EXCEPT clause so that `SELECT *` queries can explicitly omit columns from results.
- [DB-21769]: Metabase Integration — Added Ocient as a Metabase partner driver, allowing Metabase to access Ocient databases out-of-the-box.
- [DB-23030]: JDBC Packaging — Removed OpenJump dependency from ocient-jdbc4.
- [DB-23175]: Time Zone Adjustment Support — Added various improvements to time zone functionality, including:
  - Support for daylight savings adjustment based on time zone.
  - Added time zone functions `CONVERT_UTC_TIMESTAMP_TO_LOCAL` and `CONVERT_LOCAL_TIMESTAMP_TO_UTC`. For more information, see [Time Zone Functions](#).
  - Enhanced performance for time zone conversion.
- [DB-23177]: Push-Down Aggregation to the I/O Layer — Under certain conditions, the system pushes aggregation to the I/O operator for better efficiency and performance.
- [DB-23299]: HLL Sketch Functionality — Added support for variable log2k HLL sketch algorithm and associated functions. For details, see the [HyperLogLog Functions](#).
- [DB-23745]: Implement Evacuate Node — Evacuate node is a tool to move all segments off of a node in a system that is overprovisioned to the other nodes in the cluster. This tool is useful when you replace drives or a node.
- [DB-19888]: Machine Learning Model Updates —
  - The Ocient System scopes machine learning models to schemas. The system assigns the `pre_v22_mlmodel` schema to any model you created prior to version 22.0.
  - The `sys.multiple_linear_regression_slopes` system catalog table has been removed.
  - Rename machine learning models using `ALTER MLMODEL`.
  - New DDL commands:
    - \* `CREATE OR REPLACE`
    - \* `REFRESH`
    - \* `EXPORT`
  - For details about the new syntaxes, see [Machine Learning Model Functions](#).

### LAT Features

- [LAT-1469]: Manual Configuration of LAT Endpoints — Enabled manual configuration of LAT endpoints for OAuth with OKTA®.
- [LAT-1475]: Enablement of Stopping Load Processing During Error Condition — Added default behavior to stop processing during file loading in the event of an unrecoverable error when the system extracts records from a file. For details, see [LAT Pipeline Configuration](#).

- [LAT-1476]: Enablement of LAT Service in Installation — Enabled LAT Service in `systemd` by default upon installation completion. This update reflects a change in the default behavior during installation.
- [LAT-1477]: LAT Version for Metrics — Exposed `lat_version` in the metrics.
- [LAT-1557]: Support for Loading Multiple S3 Buckets — Added LAT functionality to load data from multiple S3 buckets simultaneously within the same pipeline.

### Version Compatibility

- Information Schema — Views created prior to Version 22.0 do not have column data appearing in the `information_schema`. You can drop and recreate these views to populate column data.
- LAT — Version 3.0.0 and greater is only compatible with Version 22.0 and greater of the Ocident system. For details, see [Version Compatibility](#).

## 21.0

### Release Highlights

The Ocident System now supports the following operating systems:

- Ubuntu® 20.04
- Debian 11
- RHEL 8

Other highlights include:

- Whole column compression: Added Zstandard (ZSTD) compression for fixed and variable length columns.
- Check system configuration: Added `precheck` and `postcheck` commands to check system configuration before and after installation.
- Workload management dynamic priority: Enabled the adjustment of the query priority dynamically at the session, service class, and query levels.
- Ability to quiesce node: Added process for graceful node shutdown.

### Features

- [DB-18636]: ZSTD Compression - Added a new whole-column compression scheme (ZSTD) that can be enabled for fixed and variable length columns.
- [DB-18990]: Improved Stats Storage And Usage - Various improvements have been added to speed up the fetching of statistics by the optimizer and ensure it gets up-to-date statistics. These changes primarily center around probability density functions being stored as pre-aggregated stats files instead of on a per-segment basis.
- [DB-20190]: Distributed Tasks - Added `check_disk` task type and new vtables `sys.subtasks`, `sys.tasks`, and `sys.rebuild_tasks` for monitoring tasks. Remove `CHECK DATA` command.
- [DB-19117]: Metadata - Added `participating_nodes` to the `sys.queries` and `sys.completed_queries` virtual tables.
- [DB-18633]: Graceful Node Shutdown - Added quiesce process for graceful node shutdown.
- [DB-18061]: LCK Deprecation - Added new disk data format that is smaller and also improves performance of some index based queries.
- [DB-19414]: Range Query Improvement - Improved performance of range queries by utilizing the inverted secondary index.
- [DB-20168]: Geospatial Function Expansion - Added these geospatial scalar functions.
  - Measurement Functions
    - \* `ST_ANGLE`
    - \* `ST_DISTANCESPHERE`
    - \* `ST_DISTANCESPHEROID`
    - \* `ST_LENGTH2D`
    - \* `ST_HAUSDORFFDISTANCE`
  - Analytic and Property Functions
    - \* `ST_DIMENSION`
    - \* `ST_GEOHASH`
    - \* `ST_SRID`

- \* ST\_ISPOLYGONCW
- \* ST\_ISPOLYGONCCW
- To String and Binary Functions
  - \* ST\_ASWKT
  - \* ST\_ASWKB
  - \* ST\_ASEWKT
- Geography Simplification Function
  - \* ST\_SIMPLIFY
- Constructor Functions
  - \* ST\_POINTFROMGEOHASH
  - \* ST\_GEOPOINT
  - \* ST\_MAKEPOLYGONORIENTED
  - \* ST\_POINT\_FROMEWKT
  - \* ST\_LINESTRING\_FROMEWKT
  - \* ST\_POLYGON\_FROMEWKT
  - \* ST\_MAKEENVELOPE
- Additionally, you can construct ST\_POLYGON types directly from a POINT[] without going through an intermediate ST\_LINESTRING.

### Keywords\* \*

Added these new keywords as reserved words in the Ocident system.

- ANALYSIS
- AUTOREGRESSION
- BAYES
- CANCEL
- COMPONENT
- DECISION
- DISABLE
- DISABLE\_STATS\_FILE\_UPDATES
- ENABLE
- FEEDFORWARD
- INSERT
- KMEANS
- KNN
- LOGISTIC
- MACHINE
- MOVE
- NAIVE
- NETWORK
- NONLINEAR
- PRINCIPAL
- REPLACE
- SOURCE
- SUPPORT
- TREE
- VECTOR
- ZSTD

## 20.0

### Release Highlights

- CREATE TABLE AS SELECT SQL Statement: Extract, load, and transform (ELT) workflow functionality to extract data and load it into a new database table by using the query results from a SELECT SQL statement. The tables you create using the CREATE TABLE AS SELECT SQL statement have some indexing limitations

in version 20.0. For details, see the “About Create Table As Select (CTAS)” section of the Ociant user documentation.

- INSERT INTO SQL Statement: ELT workflow functionality to extract data and insert it into an existing database table using the INSERT INTO SQL statement.
- N-gram Indexes: Full index on VARCHAR, VARCHAR arrays, and VARCHAR tuple components for efficient queries using the LIKE SQL statement.
- Large VARCHAR [DB-16142]: Support VARCHAR columns up to 1GB in size.
- Ociant Simulator: An instance of the Ociant system for data loading and functional testing.
- Single Sign-On (SSO): Authenticate access to Ociant through an external SSO server and assign SSO users to groups in Ociant.

### Feature Removal

The ALTER ROLE DDL command has been removed. You can make all changes using the ALTER CONFIG SQL statement. To alter a role, prefix the key with the role name followed by a dot.

The following system tables have been added:

- average\_bb\_sizes
- linear\_combination\_regression\_models
- node\_config
- node\_status
- sso\_connections
- storage\_device\_status

The following system tables have been removed:

- hugepage\_configurations
- memory\_module\_models
- node\_memory\_modules
- oidc\_integrations
- oidc\_sessions
- polynomial\_regression\_models
- security\_integrations
- sessions

## 19.0

### Features

- [DB-14527] - Adaptive Water Mark Feature - Indexer Node dynamically increase and reduce batch size without manual tuning
- [DB-14656] - Added a rest endpoint to expose a node's configuration parameters (:9090/v1/configparams)
- [DB-15123] - Expose cluster total storage space and storage usage through virtual tables
- [DB-15515] - Add support for expr::dtype cast notation
- [DB-16289] - Remove the web ui and YAML service role configuration
- [DB-16904] - Allow any predicate type to be used in conjunction with the values in arrays
- [DB-17889] - Improve ability to continue data loading when a foundation node is down
- [DB-18393] - Leverage hyperthreading in query execution
- In v19 the service role configuration previously set through the web UI has been replaced by the ALTER ... ALTER ROLE/CONFIG ... DDL command. The web UI is still available in v19, but will be removed in a subsequent release. The ALTER ... ALTER ROLE/CONFIG ... command should be used to change system configuration, rather than the web UI. Please reference the Upgrade Ociant Software section of the user documentation for details.
- [DB-12747] - Add support for lateral joins.
- [DB-13924] - Add support for multi-column subqueries.
- [DB-14990] - Add support for native right joins.
- [DB-15996] - Add support for array\_to\_string function.

- [DB-16231] - Improve GIS function performance and introduce expanded support for GIS functions. Please refer to the User Documentation for details.
- [DB-17037] - Add new scalar functions and operators added for GIS types (POINT, LINESTRING, and POLYGON). Please refer to the User Documentation for details.
- [DB-17892] - Add support for right lateral joins.
- [DB-16061] - Secondary indexes can now be created on VARCHAR and VARCHAR[] columns. Please refer to the User Documentation for details.

## 18.0

### Features

- [DB-17635] - Remove query log properties timestamp\_optimizationcomplete and time\_optimizationcomplete and add new properties timestamp\_optimizationstart and time\_optimizationstart
- [DB-16567] - Make error messages more clear for queries with GroupBy missing
- [DB-17316] - Change array\_length(empty array) to return 0
- [DB-16417] - Allow for integral types for integer field in GIS functions
- [DB-16200] - Make Explains more convenient for the user.
- [DB-16092] - Distributed Result Set Caching
- [DB-15623] - Add support for rebuilding individual nodes via DDL
- [DB-15375] - ALTER CLUSTER ADD PARTICIPANTS DDL
- [DB-14720] - Provide a way to kill long running optimizations
- [DB-14017] - Support for CLI command history across sessions

## 17.0

### Features

- [DB-12888] - Add support for array values larger than 128 KB. The new maximum value of an array is 512 MB.

## 16.0

### Features

- [DB-10329] - Add support for full disk encryption of Opal drives. Disk encryption will be automatically enabled when Opal support is detected.

## 14.0

### Features

- [DB-14159] - Default hex values for binary or varbinary columns must contain a leading 0x

## 13.0

### Features

- [DB-14330] - Remove last dependencies on PostgreSQL from the database

## 12.0

### Features

- [DB-13334] - Add support for zip unnest, which unnests multiple arrays in parallel

## 11.0

### Features

- [DB-12887] - Add support for the array of tuples. Users can create array columns containing tuple SQL types. Please refer to the User Documentation for the latest information on supported data types.
- [DB-12885] - Add support for `unnest()`, which expands array elements from input array columns out to individual output rows

## 10.0

### Features

- [DB-12394] - Added support for running on CentOS 8.

## 9.0

### Features

- [DB-13162] - Added support for Tasks to the System Catalog
- [DB-10332] - Implemented access controls on system and database-level objects. Improved users, groups, and added new roles within Ocident.
- [DB-12829] - Optionally enforce encrypted connections for JDBC and ODBC.

## 8.0

### Features

- [DB-10330] - External Network Security. SSL/TLS support in ODBC and JDBC, SSL support for the web interface.

## 7.0

### Features

- [DB-10921] - Adds support for multi-dimensional arrays and the ability to do joins, windows, sorts and aggregations that involve arrays.
- [DB-10927] - Adds support for global dictionary compression (GDC) on VARCHAR array columns and the ability to do replacement joins.
- [DB-10282] - Adds support for DROP COLUMN DDL to remove columns from a table.
- [DB-11472] - Adds support for skipping failed rows for CSV loading up to some specified threshold.

## 6.0

### Features

- [DB-9707] - Scriptable Bulk Load Essentials: Allows users to create translations and launch bulk load tasks via DDL
- [DB-10479] - Adds support for Tableau through Ocident's JDBC Custom Connector. Users can find Ocident's connector and the installation instructions on Tableau's extension gallery. Please refer to Tableau for more information.

## 5.0

### Features

- [DB-9477] - Adds support for the array data type. Users can create single-dimensional array columns from any other supported data type. Please refer to the User Documentation for the latest information on supported data types.

- [DB-9656] - Add column support. The engine now supports the add column DDL statement with the ability to add columns to an existing table. Existing data that was loaded without the new column uses the configured default values when queried. Please refer to User Documentation for information on the DDL syntax and default values.

## 4.0

### Features

- [DB-6386] - Availability of the storage engine, allowing queries to run with a node or drive failure
- [DB-6588] - Bulk loading of CSV files from HDFS or an S3 endpoint
- [DB-7221] - Delta compression in the TKT engine for timestamp columns
- [DB-7623] - Virtual tables to retrieve information from the storage cluster state
- [DB-7247] - OS Upgrade functionality
- [DB-6125] - AWS initial support
- [DB-6383] - Data Definition Language (DDL) operations
- [DB-6940] - All system configuration in the System Catalog
- [DB-6362] - Stats Virtual Tables
- [DB-7098] - External Window Operator support
- [DB-7497] - List Running Tasks Page
- [DB-7097] - Segment Group Deletion
- [DB-7139] - Cancel Query and Cancel Task support
- Numerous stability and performance improvements

## 14.2 JDBC Release Notes

All Ocient® JDBC drivers are located the Ocient Maven repository.

### 3.5.1 (2025-06-16)

- Disabled asynchronous updates.

### 3.5.0 (2025-06-07)

- Added support for asynchronous updates, enabling you to retrieve the query identifier immediately after executing the SQL statement instead of when it finishes on the server. Now, you can use Ctrl-C to cancel the update at the command line.
- Added the custom GraphX library for Apache® Spark™.
- Added the `DISTRIBUTE_WORK` connection parameter, which distributes queries or commands between all endpoints in a round-robin way, instead of relying on server-side load balancing.

### 3.4.1 (2025-05-14)

- New Features
  - Changed the default value for the `path-style-access` extract option to `true`.

### 3.4.0 (2025-04-30)

- New Features
  - Removed quotes added during the extract around `VARCHAR` data types inside arrays by default. Added the `QUOTE_ARRAY_VARCHARS` option for the extract that, when you set the value to `true`, adds those quotes back.
  - Removed the leading forward slash from IP addresses extracted from the database.
  - Added periodic polling of the `.ociant-cli-configuration` file, so the system applies changes to the settings in the file at runtime. The default interval is 5 minutes. You can change the default value by setting the `com.ociant.cli.config-poll-interval` system property to the chosen number of seconds.
  - Fixed the CLI result set output from formatted to a raw string.

**3.3.1 (2025-03-30)**

Internal updates only.

**3.3.0 (2025-03-06)**

- New Features
  - Added the specification of the identity provider in the connection string when connecting to the database.
  - Added support for the bzip2 and xz compression.
  - Added support for multi-line Apache® Groovy™ definitions in SQL statements.
  - Changed the default SSO callback address to `http://localhost:7050/ocient/oauth2/v1/callback`.
  - Added the `ssoNumericAddress` parameter to force the client to use the IP address 127.0.0.1 instead of `localhost` when you set this parameter to `true`.
  - Enabled switching the metrics target directory without having to restart the CLI session.
  - Added support for line and block comments in multi-line SQL statements.

**3.2.4 (2025-01-03)**

Internal updates only.

**3.2.3 (2024-12-13)**

Internal updates only.

**3.2.2 (2024-11-08)**

Internal updates only.

**3.2.1 (2024-09-30)**

- New Features
  - Changed the default escape character for quotes in VARCHAR arrays from `\` to `"` for extracts.
  - Added connection property to redact usernames and SQL text in JDBC logs.
  - Increased security for passwords in the command-line interface and history file.

**3.1.7 (2024-06-03)**

Internal updates only.

**3.1.6 (2024-06-03)**

Internal updates only.

**3.1.5 (2024-06-03)****Bugs**

- [DB-28722]: Fixed the issue where the client closed the result set before sending the KILL\_QUERY request.
- [DB-28325]: Fixed an underlying SLF4JLogger error that prevented the driver from importing.
- [DB-24984]: Fixed HTTP client configuration to accept standard cookie specifications.

**3.1.1 (2024-03-08)**

- *New Features*
  - Enabled support of fetching cached queries using the JDBC 3.1.1 driver with version 22.1 and older versions of the Ocint Hyperscale Data Warehouse™.
  - Fixed connection failures by adding sequence numbers to metadata responses on the client side for legacy databases without sequence number support.



**3.1.0 (2024-01-31)**

- *New Features*
  - Improved maximum throughput between client and server, specifically for transferring large and sorted result sets.
  - Deprecated `com.ocient.jdbc.XGConnection#redirect`. The `redirect` method in the `com.ocient.jdbc.XGConnection` class is now deprecated and is slated for removal in future releases. For applications that require a connection to a specific SQL Node, Ocient recommends to use the `force=true` connection property.

**3.0.0 (2023-10-24)**

- *New Features*
  - Improved scalability for high-volume concurrent database connections for applications such as Metabase<sup>SM</sup> and Superset.
  - Improved latency for bulk data transfers.

**2.106 (2023-06-29)**

- *New Features*
  - Fixed requirement to restart the Java® application after cluster upgrade.

**2.105 (2023-03-22)**

- *New Features*
  - Improved performance by pooling database connections used for sideband channels.
  - Added `MAX_THREADS_PER_RESULTSET` connection property.

**2.100 (2023-03-10)**

- *New Features*
  - Added ability to periodically log metrics to a file.
  - Added `METRICS ON <alphanumeric> CLI` command.

**2.99 (2023-01-24)**

- *New Features*
  - Added `PERFORMANCE NETWORK CLI` command.
  - Added `PERFORMANCE DATABASE CLI` command.
  - Deprecate `PERFORMANCE ON CLI` command.

**2.98 (2023-01-17)**

- *New Features*
  - Removed OpenJump system dependency

**2.97 (2022-12-27)**

- *New Features*
  - Added support for `DELETE FROM`

**2.96 (2022-12-21)**

- *New Features*
  - Fixed `IndexOutOfBoundsException` in server redirect

**2.95 (2022-12-07)**

- *New Features*
  - Log stack trace of `XGStatement#sendAndReceive()` read errors

**2.94 (2022-12-07)**

- New Features
  - Allow extract to `/dev/null`

**2.93 (2022-11-28)**

- New Features
  - Strip trailing statement delimiter from queries

**2.92 (2022-11-22)**

- New Features
  - Include `jdbc.sql.Driver` service in META-INF

**2.91 (2022-11-22)**

- New Features
  - Add support for `REFRESH MLMODEL` command, which allows updating of a Machine Learning model without changing the ML model properties

**2.90 (2022-11-22)**

- [DB-22501] Never resubmit queries that have been sent to the server

**2.89 (2022-11-18)**

- Add support for `EXPORT MLMODEL` command

**2.88 (2022-11-08)**

- Load JDBC version from build properties file at runtime

**2.87 (2022-11-07)**

- Set Service Class

**2.86 (2022-10-24)**

- Allow connection property override

**2.85 (2022-10-24)**

- Address CVE-2022-33980 and CVE-2022-24823

**2.84 (2022-10-24)**

- Print hex instead of address of byte array in `XGTuple#toString`

**2.83 (2022-10-18)**

- Re-enabled ability to override a connection property
- Fix query timeouts which were not being honored
- Handle non-resolvable database DNS names received from the database

**2.82 (2022-10-14)**

- Made JDBC Connection properties case-insensitive using a new class `CaselessProperties`
- Converted the rest of the connection properties to the `ConnectionProperty` enum

**2.81 (2022-10-03)**

- Implement `XGPreparedStatement#setObject(Index, Timestamp, Calendar)`
- Add “statementPooling” driver property. Set to “OFF” to disable statement caching

**2.80 (2022-10-03)**

- Add `GET JDBC COMMIT INFO` command
- Add `/version`, `/commit`, `/info`, and `/pid` metrics endpoints

**2.79 (2022-10-03)**

- Ensure backwards compatibility with JRE8

**2.78 (2022-09-19)**

- Add CLI support for ‘EXPORT PIPELINE’

**2.77 (2022-09-19)**

- Add `START PIPELINE` and `STOP PIPELINE` for updates

**2.76 (2022-09-02)**

- Made `SHOW` command work as a `SELECT` outside of just the CLI

**2.75 (2022-09-02)**

- Made `SHOW` command a select command rather than an update command

**2.74 (2022-08-29)**

- `BENCHMARK` command consumes result set for consistency w/ manual input

**2.73 (2022-08-26)**

- New Features
  - Add support for `SHOW` command, which allows easy querying of the information schema.

**2.72 (2022-08-23)**

- New Features
  - [DB-20912] Add support for one-time password (OTP) based SSO flows
  - [DB-20912] Add OAuth2 Authorization Code w/ PKCE Single Sign-On Flow
  - [DB-20912] Add OAuth2 Device Authorization Grant Single Sign-On Flow for input constrained environments

**2.71 (2022-08-23)**

- New Features
  - Add `BENCHMARK NSAMPLES COMMAND` which will run `COMMAND NSAMPLES` times and produce a timing report in the CLI output.
  - Add `PRINTUUID [ON | OFF]` which will print the query UUID to the CLI output.

**2.70 (2022-08-11)**

- New Features
  - [DB-21078] Add `s3_upload_part_size` and `s3_upload_part_parallelism` extract configuration properties.

**2.69 (2022-08-11)**

- New Features
  - [DB-21078] Add REST endpoint to expose query and data extract performance metrics
  - [DB-21078] Add ms precision to JDBC log entries

**2.68 (2022-08-09)**

- New Features
  - [DB-21251] Allow extract property override via “JDBC\_EXTRACT\_” prefixed environment variable

**2.67 (2022-06-30)**

- New Features
  - Allow aliases on nested queries in extract SQL statements (DB-21075)

**2.66 (2022-06-27)**

- New Features
  - Escape escape characters in quoted field for non-comma delimited extracts

**2.65**

- New Features
  - Add support for using `CANCEL TASK` to cancel a task instead of treating it like a `CANCEL` command

**2.64 (2022-06-21)**

- New Features
  - Resolve CVE-2022-24823; update netty-common version (4.1.77.Final)
  - Resolve CVE-2021-43797; update netty-codec-http version (4.1.71.Final)

**2.63 (2022-06-01)**

- New Features
  - Replace `skip_header` with `header_mode` in CSV extract options.
  - Add CLI configuration file, `~/.ocient-cli-configuration`, and idle timeout option `cliIdleTimeoutMinutes`

**2.62 (2022-06-01)**

- New Features
  - Add `escape_unquoted_values`, `input_escaped`, and `quote_all_fields` in CSV extract options.
  - Add a new custom TSV writer that handles all non-comma delimiter cases.

**2.61 (2022-05-31)**

- New Features
  - Add `compression_block_size`, `compression_level`, and `num_compression_threads` Data extract properties.

**2.60 (2022-05-24)**

- New Features
  - Allow non-query commands to print results when performance mode is turned on for easier debugging.

**2.59 (2022-05-17)**

- New Features
  - Add UTF-8 character translation to Data extract tool.

**2.58 (2022-4-20)**

- New Features
  - Prevent shell expansion of ! when reading user input from CLI

**2.57 (2022-3-17)**

- New Features
  - Bumped protocol version for quiesce
  - Modified length checks to check if higher order bit is set, signalling a message from the server

**2.56 (2022-3-15)**

- New Features
  - Added SQL states for EXECUTE\_PLAN\_AUTH\_FAILURE and EXECUTE\_INLINE\_PLAN\_AUTH\_FAILURE

**2.55 (2022-3-11)**

- New Features
  - Reconnect and rerun if writing into socket fails when sending request
  - Reconnect and rerun if when sending request, shows -1 length
  - Added checks for -1 length on response for server signalling a quiescing connection close
  - Copy secondary interfaces and secondary index when returning a cached connection

**2.54 (2022-3-9)**

- New Features
  - Added CLI support for SET ADJUSTFACTOR, ADJUSTTIME
  - Added new columns in system queries: sys.queries: initial\_priority, initial\_effective\_priority, effective\_priority, priority\_adjust\_factor, priority\_adjust\_time sys.completed\_queries: initial\_priority, initial\_effective\_priority, final\_effective\_priority, priority\_adjust\_factor, priority\_adjust\_time

**2.53 (2022-2-8)**

- New Features
  - Added redirect support for execute export, explain pipeline, check data

**2.52 (2022-1-18)**

- New Features
  - Secondary result set threads will start fetching immediately after the first result set thread gets a ping.

**2.51 (2022-1-14)**

- New Features
  - Make force external consistent with other connection parameters

**2.50 (2022-1-13)**

- New Features
  - Fix invalid argument message to not refer to ODBC client

**2.49 (2022-1-6)**

- New Features
  - Add more logging to multithreaded result set fetch.

**2.48 (2022-1-3)**

- New Features
  - Insert statements supported

**2.47 (2021-12-16)**

- New Features
  - Data extract tool released.
  - Deactive connection caching for calls to createConnection

**2.46 (2021-12-8)**

- New Features
  - Fix connection pooling handling with regards to setting schema incorrectly.

**2.45 (2021-12-2)**

- New Features
  - Swallow runtime error as well when trying to set socket options.

**2.44 (2021-12-1)**

- New Features
  - Wrap invalid socket options in try catch block and swallow exceptions. For compatability with Java versions that are older.

**2.43 (2021-11-22)**

- New Features
  - Change driver to not rely on the pom/MANIFEST for the driver version. Move the version into JDBCDriver

**2.42 (2021-11-18)**

- New Features
  - Changed connections to utilize keep alive probes in order to detect dead connections.

**2.41 (2021-11-07)**

- New Features
  - Upgrade Jline version to 3.21

**2.40 (2021-11-03)**

- New Features
  - Improve result set fetching performance

**2.39 (2021-10-29)**

- New Features
  - Fix handling of expired token being used to start a session

**2.38 (2021-10-27)**

- New Features
  - Updated auth related SQLState error codes
  - Fix handling of result set close during result set caching

**2.37 (2021-10-25)**

- New Features
  - Force redirect support for testing purposes.
  - Removed duplicate code from CLI

**2.36 (2021-10-14)**

- New Features
  - Implemented refresh sessions support.
  - Implement get server session ID.

**2.35 (2021-10-11)**

- New Features
  - ExecutePlan and ExecuteInlinePlan bugfixes

**2.34 (2021-9-9)**

- New Features
  - Add token signature and issuer fingerprint fields to sso token handshake.
  - Add explicit SSO flags to handshake GCM.
  - Pass networkTimeout to new connection upon copy.

**2.33 (2021-9-7)**

- New Features
  - Add support for explicit SSO handshakes
  - Revert executeQuery redirection change

**2.32 (2021-9-1)**

- New Features
  - CLI source command stops on error
  - Support “output next query append”
  - Add statement command “get jdbc version”

**2.31 (2021-8-30)**

- New Features
  - Add sso handshake support
  - Refactor duplicated code in handshakes for saving secondary interfaces

**2.30 (2021-8-26)**

- New Features
  - Add validation on the driver side for set parameter settings.
  - Pass up invalid set parameter command errors.

**2.29 (2021-8-20)**

- New Features
  - Fix the way connection resets parameters when reset is true.
  - Fix an issue where redirect was not properly using mapped secondary interfaces

**2.28 (2021-8-16)**

- New Features
  - Fix sending parameters on reset. maxrow, maxtempdisk, maxtime, parallelism, priority now gets reset properly.
  - Map list all queries to select \* from sys.queries
  - Improve logging
  - Statements with closed connections will not get returned to the cache

**2.27 (2021-8-10)**

- New Features
  - Fixed poorly structured if statement in CLI source command.
  - Remove more stack trace printing and add more logs.

**2.26 (2021-7-30)**

- New Features
  - Move the duplicated regex code in CLI.java and XGStatement.java into a new file.
  - Implement a generic regex for syntax checking the set family of sql commands.
  - Fix a bug in resetting commands using lower cases. “set maxrow reset;” does not work. Needs to be capitalized.

**2.25 (2021-7-27)**

- New Features
  - Default clientVersion to 0.00 if the driver provides a null clientVersion to XGConnection.

**2.24 (2021-7-12)**

- New Features
  - Have setSchema handle unquoted caps sql. “SET SCHEMA MADISON” will set schema to “madison”.

**2.23 (2021-7-6)**

- New Features
  - Change SET CONCURRENCY command to SET PARALLELISM
  - Change “concurrency” driver property to “parallelism”

**2.22 (2021-7-1)**

- New Features
  - Add openJump extensions to jdbc jar.
  - Fix source command for plan execute inline.

**2.21 (2021-6-30)**

- New Features
  - Fix new hashcode for null elements.

**2.20 (2021-6-28)**

- New Features
  - Update list all queries metadata to match that of select \* from sys.queries
  - Add more connection level settings to hashCode.



**2.19 (2021-6-26)**

- New Features
  - Improve source command capabilities and performance

**2.18 (2021-6-26)**

- New Features
  - Save command history across sessions

**2.17 (2021-6-25)**

- New Features
  - Include the currently set schema to the connection hashcode.

**2.16 (2021-6-23)**

- New Features
  - Fix another issue with timeoutMillis. The timeout will now work correctly with zero ping buffers.

**2.15 (2021-6-21)**

- New Features
  - For dBeaver compatability, not setting a user in driver properties will cause the driver to default to empty string.

**2.14 (2021-6-17)**

- New Features
  - Fix timeoutMillis by correctly inheriting timeoutMillis from properties.
  - Clear warnings before running executeQuery and executeUpdate.

**2.13 (2021-6-09)**

- New Features
  - Added command to SET PSO SEED for the random number generator used in PSO
  - Remove chatty number of rows log.
  - Add remoteIp and service class to list all queries.
  - Add sleep\_in\_optimizer command for testing.

**2.12 (2021-5-26)**

- New Features
  - Fix the parsing for execute() to properly route to executeStatement()

**2.11 (2021-5-20)**

- New Features
  - Improve cache and statement pooling support for redirection.
  - Fix timestamp and time with negative nanos.

**2.10 (2021-5-19)**

- New Features
  - Added 10 different colors for GIS types in KML
  - Added non-GIS types to description of each GIS object in KML
  - Moved KML feature from CLI to driver

**2.09 (2021-5-18)**

- New Features
  - Use daemon threads for background caching tasks. This allows the program to exit without finishing these tasks.

**2.08 (2021-5-10)**

- New Features
  - Add versions packaging to pom.xml.
  - Updated dependencies to latest version using versions plugin.
  - Change TLS to unverified by default.

**2.07 (2021-5-05)**

- New Features
  - Add OBJECT\_NOT\_FOUND\_WARN warning.
  - Add OBJECT\_ALREADY\_EXISTS\_WARN warning.
  - Fix some misleading log messages.

**2.06 (2021-4-29)**

- New Features
  - Add a log for if reconnect() fails to close a socket.
  - Driver sends over unique session ID to server.
  - Fix a misleading log in client handshake.

**2.05 (2021-4-28)**

- New Features
  - Fix spotbugs and narrow spotbugs filter.

**2.04 (2021-4-22)**

- New Features
  - Added getters to GIS types

**2.03 (2021-4-22)**

- New Features
  - Output next query also prints all gis types into a kml file

**2.01 (2021-4-21)**

- New Features
  - Fix empty point for STPoint class

**2.01 (2021-4-20)**

- New Features
  - Better build integration with xgsrc
  - Upgraded protobuf to version 3.14

**2.00 (2021-4-17)**

- New Features
  - Handling for cache limit warning.

**1.99 (2021-4-13)**

- New Features
  - Add support for EXPLAIN DEBUG, and make EXPLAIN format JSON by default

**1.98 (2021-4-05)**

- New Features
  - Add support for QUARANTINE
  - Fix CLI command force external on

**1.97 (2021-4-02)**

- New Features
  - Queries that exceed the row limit set by XGConnection::setMaxRows now silently omit excess rows. The previous behavior would result in query failure.

**1.96 (2021-4-01)**

- New Features
  - Add support for POINT EMPTY
  - Fix CLI performance on/off affecting timing setting.

**1.95 (2021-3-22)**

- New Features
  - Add support for CHECK DATA to CLI
  - Fix bug where defaultSchema was not working.

**1.94 (2021-3-18)**

- New Features
  - Add major and minor version to client handshake.
  - Add logging to set param functions.

**1.93 (2021-3-11)**

- New Features
  - Improve performance of writing query results to a file.

**1.92 (2021-3-5)**

- New Features
  - Add driver method to cancel all cache return threads.
  - Add CLI functionality to limit max history size.

**1.91 (2021-3-4)**

- New Features
  - Add support for st\_linestring and st\_polygon

**1.90 (2021-3-2)**

- New Features
  - Enable spotbugs analysis
  - Increase tracing to millisecond granularity

**1.89 (2021-2-24)**

- New Features
  - Switch to using AES/GCM/NoPadding encryption

**1.88 (2021-2-24)**

- New Features
  - Fix some broken custom xgMetadata calls.

**1.87 (2021-2-24)**

- New Features
  - Added support for TUPLE columns as a SQL Struct type

**1.86 (2021-2-19)**

- New Features
  - Fix more OWASP bugs

**1.85 (2021-2-18)**

- New Features
  - Added CLI support for EXPORT VIEW

**1.84 (2021-2-16)**

- New Features
  - Update the request type for EXPLAIN PIPELINE

**1.83 (2021-2-15)**

- New Features
  - Fixed a bunch of OWASP bugs.

**1.82 (2021-2-12)**

- New Features
  - Add CLI support for SET MAXROWS, PRIORITY, CONCURRENCY, MAXTIME, and MAXTEMPDISK

**1.81 (2021-2-12)**

- New Features
  - Fix a compiler error with CLUSTER\_NOT\_FOUND error

**1.80 (2021-2-12)**

- New Features
  - Add CLUSTER\_NOT\_FOUND error

**1.79 (2021-2-11)**

- New Features
  - Add spotbugs
  - Add support for EXPLAIN PIPELINE to CLI

**1.78 (2021-2-5)**

- New Features
  - Add SqlStates for LUP decomposition
  - Add additional cache logging.
  - Fix incorrect schema generated when first statements are created.

**1.77 (2021-2-5)**

- New Features
  - Remove unnecessary calls to fetchServerVersion
  - Fix schema for pooled connections and statement.

**1.76 (2021-2-2)**

- New Features
  - Pooling improvements

**1.75 (2021-2-1)**

- New Features
  - Allow changing session variable defaults via connection properties and make session variable overrides local to statement objects.

**1.74 (2021-1-31)**

- New Features
  - Performance optimizations and restructuring

**1.73 (2021-1-30)**

- New Features
  - The JDBC driver now does connection and statement pooling that is automatic and transparent to the caller.

**1.72 (2021-1-27)**

- New Features
  - Add support for clearBatch()

**1.71 (2021-1-14)**

- New Features
  - Add support for session overrides of service class limits
  - SET MAXROWS {N}
  - SET MAXTIME {N}
  - SET PRIORITY {N}
  - SET CONCURRENCY {N}

**1.70 (2021-1-12)**

- New Features
  - Fix race condition between adding fetch threads to array and iterating. Also removed some duplicated code.

**1.69 (2021-1-07)**

- New Features
  - Remove deprecated plan proto

**1.68 (2020-12-24)**

- New Features
  - Fix describe view truncate in CLI.

**1.67 (2020-12-14)**

- New Features
  - Fixed cols2Types mapping in RS returned by LIST ALL COMPLETED QUERIES.

**1.66 (2020-12-10)**

- New Features
  - Add spaces to syntax parsing.

**1.65 (2020-12-9)**

- New Features
  - Completed queries update.

**1.64 (2020-12-7)**

- New Features
  - Properly handle nullability for the LIST ALL COMPLETED QUERIES command.

**1.63 (2020-12-1)**

- New Features
  - Don't print stack trace when select queries complete exceptionally.

**1.62 (2020-11-30)**

- New Features
  - Added support for LIST ALL COMPLETED QUERIES command.

**1.61 (2020-11-15)**

- New Features
  - Fix reconnect and resending logic.

**1.60 (2020-11-10)**

- New Features
  - DESCRIBE TABLE now consistently uses SMALLINT and BIGINT to describe inner types of arrays

**1.59 (2020-11-4)**

- New Features
  - remove another unnecessary driver version fetch.
  - add jar version to log

**1.58 (2020-11-4)**

- New Features
  - fix fetch version redirect loop.

**1.57 (2020-11-4)**

- New Features
  - make SQLException codes unique.

**1.56 (2020-11-2)**

- New Features
  - kill and cancel now throws syntax errors.

**1.55 (2020-10-28)**

- New Features
  - CLI now recognizes a DDL statement for INVALIDATE STATS.

**1.54 (2020-10-21)**

- New Features
  - fix another spot where all the slq nodes can be brought down.

**1.53 (2020-10-20)**

- New Features
  - exportTranslation will close the result set.

**1.52 (2020-10-19)**

- New Features
  - executeQuery will not rerun query after reconnect.

**1.51 (2020-10-11)**

- New Features
  - Add token fields and username to localQueries protobuf

**1.50 (2020-10-11)**

- New Features
  - Multiple result set threads support is complete

**1.49 (2020-10-06)**

- New Features
  - Experimental support for multiple result set threads

**1.48 (2020-09-28)**

- New Features
  - Minor fix of SQLStates
  - setParms() does no work when there are no parameters

**1.47 (2020-09-26)**

- New Features
  - Improve performance for large SQL statements

**1.46 (2020-09-21)**

- New Features
  - Bug fix for driver not reconnecting when a previous result set is still open.

**1.45 (2020-09-16)**

- New Features
  - Move export translation into result set.

**1.44 (2020-09-15)**

- New Features
  - Support granting and revoking both privileges and role membership.

**1.43 (2020-09-14)**

- New Features
  - Explain, listTables, listViews, getSchema, describeTable, describeView, explainPlan, listAllQueries, and exportTable.

**1.42 (2020-09-2)**

- New Features
  - Move some custom functionalities into the driver including: get/set schema, list (system) tables, list views, describe table/views, list indexes, execute/explain plan, cancel/kill query, list all queries, export table, set max rows, set pso.

**1.41 (2020-08-20)**

- New Features
  - Ability to set max output rows from the CLI

**1.40 (2020-08-14)**

- New Features
  - SSL support.

**1.37 (2020-07-26)**

- New Features
  - Timestamp and Time now have nanosecond precision

**1.36 (2020-07-25)**

- New Features
  - JDBC driver can now handle hostnames sent back in interface list

**1.35 (2020-07-23)**

- New Features
  - Capture initial connection IP and use as last resort for reconnect()

**1.34 (2020-07-14)**

- New Features
  - Implement new time/date/timestamp subclasses to circumvent bugs in the standard versions

**1.33 (2020-07-04)**

- New Features
  - Add support for load balancing with secondary SQL interfaces

**1.32 (2020-06-26)**

- New Features
  - Add support for Export Translation to CLI



**1.31 (2020-06-23)**

- New Features
  - Fix misbehavior with ‘plan execute inline’ using the CLI

**1.30 (2020-06-22)**

- New Features
  - Deprecated the HDFS connection table, which was consolidated into the external connection table

**1.29 (2020-06-18)**

- New Features
  - Creating an MLModel on no data is now an error, not a warning

**1.28 (2020-06-11)**

- New Features
  - Allow one connection to be shared by multiple threads, as long as each thread uses its own Statement object

**1.27 (2020-06-11)**

- New Features
  - Improve trace log formatting, including adding thread id

**1.26 (2020-06-09)**

- New Features
  - Add support for weeks interval type

**1.25 (2020-06-05)**

- New Features
  - Support for getTableTypes()

**1.24 (2020-06-05)**

- New Features
  - Protobuf message changes for Add Column feature

**1.23 (2020-06-02)**

- New Features
  - Add new error code: Value too large, which indicates that a column value is larger than internal limits

**1.22 (2020-05-31)**

- New Features
  - Add calendar support
- Bug Fixes
  - Properly handle case-insensitivity of column names on result sets

**1.21 (2020-05-26)**

- New Features
  - Add tracing support

**1.20 (2020-05-19)**

- Bug Fixes
  - DB-11119 - JDBC multi-host support was not implemented correctly

**1.19 (2020-05-13)****1.18 (2020-05-11)****1.16 (2020-04-15)**

- Bug Fixes
  - DB-10687 - Fix for Kill/Cancel query

**1.15 (2020-04-14)**

- Bug Fixes
  - DB-9928 - Fixes for SQL Array type

**1.14 (2020-04-11)**

- New Features
  - Protocol support for the NULLS FIRST in the ORDER BY clause

**1.13 (2020-03-20)**

- New Features
  - Binary support for the serialization of the following data types: ST\_POINT, UUID, IP and IPV4
  - Implementation of the Array SQL type

**1.12 (2020-03-01)**

- Bug Fixes
  - DB-10155 - Broadcast kill query and cancel query requests to all sql nodes.
- New Features
  - DB-10137 - Implement JDBC cancel query. Now it is possible to kill query in dbeaver.
  - DB-10120 - JDBC driver support list of sql nodes to connect to
  - DB-10119 - Support for multiple IPs under the same DNS address

**14.3 pyocient Release Notes****3.4.0**

- Added support for the PREVIEW DDL SQL statement.
- Support cryptography versions greater than 42.0.0.
- Fixed an issue that returned -1 rows modified.
- Fixed the handling of an invalid DSN on the pyocient CLI.
- Formatted Unicode characters in JSON output.
- Fixed an issue where not all warnings were reported.
- Added the messages attribute to the cursor.
- Improved parameter expansion.
- Supported loglevel and logfile parameters on DSNs.
- Preserved the Connection object for reconnects.

**3.3.2**

- Fixed ending SSO sessions when you close the connection.

### 3.3.0

- Pinned the cryptography package to a version earlier than 42.0.0 for stability purposes.
- Added OIDC single sign on (SSO).
- Added the `performance mode` command.
- Enabled the display of nulls as NULL when using the table format.
- Set the command-line default fetch size to 30000.
- Fixed the implementation of the `set serviceclass` command.
- Changed `pyocient.Warning` to be a subclass of the Python® `Warning` class.
- Added the `set printuuid on` command.
- Added the `--version` option at the command line.
- Set the TCP `keepalive` options on connections.
- Added a valid error message when the system rejects a connection without SSL encryption.

### 3.0.3

- The `TypeCodes` class has been changed to a Python `IntEnum`.
- GIS types `_STPoint`, `_STLinestring`, and `_STPolygon` have been renamed to `STPoint`, `STLinestring`, and `STPolygon`, respectively.
- The `user` and `password` fields have been removed from the `Connection` object. This object no longer stores these fields.
- You can pass the `security_token` from another connection to the `connect` function.
- The `Session` object has been removed.
- The `custom_type_to_json` function has been moved into the pyocient API to allow the consistent conversion of the result set to the JSON format.
- The `patch` portion of the pyocient version is sent to and logged in the database.
- The `--nooutput` command-line parameter does not throw a Python exception.
- The local variable `'return_code'` referenced before assignment error has been fixed.
- Added the `--rows` argument to the command-line program to load the result set in batches.

### 2.0.0

- Packaging has been significantly refactored to match Python® packaging standards.
- Python context manager support has been added to connections and cursors.
- Python type annotations have been added to the pyocient API.
- Fixed the command-line interface (CLI) to return a non-zero code for statements that fail.
- CLI output has been standardized to either go to the output file or standard output (stdout).
- The option to echo statements has been added as a command-line option.
- The `pyocient.SyntaxError` exception, which conflicted with the built-in Python `SyntaxException`, has been removed.
- `pyocient.version` has been removed in favor of the more standard `pyocient.__version__`.

### 1.0.15

- The dependency on `protobuf` has been upgraded from `>=3.12.0, <=3.19.1` to `>=3.20.0, <=4.22.0`.
- Fixed an error that occurs when you do not specify the host parameter in the `connect()` API.

### 1.0.14

- Updated pyocient dependencies to eliminate some unused dependencies and tighten the dependencies on `protobuf` and `dsnparse`.

### 1.0.13

- Fixed bug that left `Cursor.rowcount` unset after Data Manipulation Language (DML) query execution.
- The `cryptography` package requirement of version 36 or lower is no longer required.

**1.0.12**

- Added `--nohistory` command-line option. The history file does not store commands from the session. Added `ignorespace` option to command-line history. The history file omits lines that begin with a white space character.

**1.0.11**

- Added a link to the release notes to the package description.

**1.0.10**

- Fixed error in the parameter substitution docstring.
- Added support for `SERVICECLASS`, `ADJUSTFACTOR`, `ADJUSTTIME`, and `PSO` query settings.
- Added support for the `SHOW` command.
- Added support for the `CANCEL TASK` command.
- Removed support for `PRIORITY_ADJUSTMENT_FACTOR`, `PRIORITY_ADJUSTMENT_TIME`, `MIN_PRIORITY`, and `MAX_PRIORITY` query settings.
- Deprecate support for the `CHECK DATA` command.
- Fixed output of binary types at the command line.

**1.0.9**

- Added CSV output support
- Updated cryptography package dependency from `<3.4` to `<=36.0`
- Updated cffi and google-auth dependencies
- Improved decoding performance by caching the decoders
- Added formal `STPoint`, `STLinestring`, and `STPolygon` classes rather than using tuples and lists
- Added support for `PRIORITY_ADJUSTMENT_FACTOR`, `PRIORITY_ADJUSTMENT_TIME`, `MIN_PRIORITY`, and `MAX_PRIORITY` query settings.
- Enabled the handling of invalid DNS names received from the database

**1.0.7**

- Fixed crash on invalid UTF-8 characters received from the database
- Fixed a bug related to specifying the user through the connect API
- Standardized Python(R) code formatting

**1.0.6**

- Updated handling of the `force` option on connections for accuracy

**1.0.5**

- Added additional debug logging to the redirection of connection

**1.0.4**

- Updated the redirect logic for accuracy

**1.0.3**

- Added multiple host support to the DSN
- Removed `-tables`, `-systables`, and `-views` command-line options
- Added support for the `source`, `connect to`, and `set` commands
- Added support for starting without a DSN

---

## 15 Open Source Acknowledgements

The Ocident System uses some third-party software to perform certain operations. Here, you can find acknowledgments for this software.

### asn1bean

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

#### TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

##### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including

the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of

the Derivative Works; and

- (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

- 5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
- 6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
- 7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
- 8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a

result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

#### END OF TERMS AND CONDITIONS

#### APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

#### antlr

Copyright (c) 2012-2022 The ANTLR Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.



2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither name of copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## aws-sdk-cpp

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

### TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

#### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

- 
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
- (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
  - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or

agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "{}" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright {yyyy} {name of copyright owner}

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS,

---

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
See the License for the specific language governing permissions and  
limitations under the License.

## boost

Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## cityhash

```
// Copyright (c) 2011 Google, Inc.  
//  
// Permission is hereby granted, free of charge, to any person obtaining a copy  
// of this software and associated documentation files (the "Software"), to deal  
// in the Software without restriction, including without limitation the rights  
// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
// copies of the Software, and to permit persons to whom the Software is  
// furnished to do so, subject to the following conditions:  
//  
// The above copyright notice and this permission notice shall be included in  
// all copies or substantial portions of the Software.  
//  
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
// FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE  
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,  
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
// THE SOFTWARE.
```

## cjose

The MIT License (MIT)

---

Copyright (c) 2014-2016 Cisco Systems, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## protobuf

Copyright 2008 Google Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Code generated by the Protocol Buffer compiler is owned by the owner of the input file used when generating it. This code is not standalone and requires a support library to be linked with it. This support library is itself covered by the above license.

---

## cpp\_httplib

The MIT License (MIT)

Copyright (c) 2017 yhirose

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## cpr

This license applies to everything except the contents of the "test" directory and its subdirectories.

MIT License

Copyright (c) 2017-2021 Huu Nguyen

Copyright (c) 2022 libcpr and many other contributors

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## curl

## COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1996 - 2025, Daniel Stenberg, <daniel@haxx.se>, and many contributors, see the THANKS file.

All rights reserved.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

## date

```
// The MIT License (MIT)
//
// Copyright (c) 2015, 2016, 2017 Howard Hinnant
// Copyright (c) 2016 Adrian Colomitchi
// Copyright (c) 2017 Florian Dang
// Copyright (c) 2017 Paul Thompson
// Copyright (c) 2018, 2019 Tomasz Kamiński
// Copyright (c) 2019 Jiangang Zhuang
//
// Permission is hereby granted, free of charge, to any person obtaining a copy
// of this software and associated documentation files (the "Software"), to deal
// in the Software without restriction, including without limitation the rights
// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
// copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included in all
// copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
// SOFTWARE.
//
// Our apologies.  When the previous paragraph was written, lowercase had not yet
```

## dpdk



The DPDK uses the Open Source BSD-3-Clause license for the core libraries and drivers. The kernel components are naturally GPL-2.0 licensed.

Including big blocks of License headers in all files blows up the source code with mostly redundant information. An additional problem is that even the same licenses are referred to by a number of slightly varying text blocks (full, abbreviated, different indentation, line wrapping and/or white space, with obsolete address information, ...) which makes validation and automatic processing a nightmare.

To make this easier, DPDK uses a single line reference to Unique License Identifiers in source files as defined by the Linux Foundation's SPDX project (<https://spdx.org/>).

Adding license information in this fashion, rather than adding full license text, can be more efficient for developers; decreases errors; and improves automated detection of licenses. The current set of valid, predefined SPDX identifiers is set forth on the SPDX License List at <https://spdx.org/licenses/>.

DPDK uses first line of the file to be SPDX tag. In case of `#!/*` scripts, SPDX tag can be placed in 2nd line of the file.

For example, to label a file as subject to the BSD-3-Clause license, the following text would be used:

```
SPDX-License-Identifier: BSD-3-Clause
```

To label a file as GPL-2.0 (e.g., for code that runs in the kernel), the following text would be used:

```
SPDX-License-Identifier: GPL-2.0
```

To label a file as dual-licensed with BSD-3-Clause and GPL-2.0 (e.g., for code that is shared between the kernel and userspace), the following text would be used:

```
SPDX-License-Identifier: (BSD-3-Clause OR GPL-2.0)
```

To label a file as dual-licensed with BSD-3-Clause and LGPL-2.1 (e.g., for code that is shared between the kernel and userspace), the following text would be used:

```
SPDX-License-Identifier: (BSD-3-Clause OR LGPL-2.1)
```

Any new file contributions in DPDK shall adhere to the above scheme. It is also being recommended to replace the existing license text in the code with SPDX-License-Identifiers.

Any exception to the DPDK IP policies shall be approved by DPDK Tech Board and DPDK Governing Board. Steps for any exception approval:

1. Mention the appropriate license identifier from SPDX. If the license is not listed in SPDX Licenses. It is the submitters responsibility to get it first listed.
2. Get the required approval from the DPDK Technical Board. Technical Board may advise the author to check alternate means first. If no other alternative

---

are found and the merit of the contributions are important for DPDK's mission, it may decide on such exception with two-thirds vote of the members.

3. Technical Board then approach Governing Board for such limited approval for the given contribution only.

Any approvals shall be documented in "license/exceptions.txt" with record dates.

DPDK project supported licenses are:

1. BSD 3-clause "New" or "Revised" License  
SPDX-License-Identifier: BSD-3-Clause  
URL: <http://spdx.org/licenses/BSD-3-Clause#licenseText>  
DPDK License text: license/bsd-3-clause.txt
2. GNU General Public License v2.0 only  
SPDX-License-Identifier: GPL-2.0  
URL: <http://spdx.org/licenses/GPL-2.0.html#licenseText>  
DPDK License text: license/gpl-2.0.txt
3. GNU Lesser General Public License v2.1  
SPDX-License-Identifier: LGPL-2.1  
URL: <http://spdx.org/licenses/LGPL-2.1.html#licenseText>  
DPDK License text: license/lgpl-2.1.txt

## dpdk

Redistribution and use in source and binary forms, with or without modification, are permitted provided that

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and t
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or pr

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WAR

## icu4c

UNICODE, INC. LICENSE AGREEMENT - DATA FILES AND SOFTWARE

See Terms of Use <<https://www.unicode.org/copyright.html>>  
for definitions of Unicode Inc.'s Data Files and Software.

NOTICE TO USER: Carefully read the following legal agreement.  
BY DOWNLOADING, INSTALLING, COPYING OR OTHERWISE USING UNICODE INC.'S  
DATA FILES ("DATA FILES"), AND/OR SOFTWARE ("SOFTWARE"),  
YOU UNEQUIVOCALLY ACCEPT, AND AGREE TO BE BOUND BY, ALL OF THE  
TERMS AND CONDITIONS OF THIS AGREEMENT.  
IF YOU DO NOT AGREE, DO NOT DOWNLOAD, INSTALL, COPY, DISTRIBUTE OR USE  
THE DATA FILES OR SOFTWARE.

## COPYRIGHT AND PERMISSION NOTICE

Copyright © 1991-2022 Unicode, Inc. All rights reserved.  
Distributed under the Terms of Use in <https://www.unicode.org/copyright.html>.

Permission is hereby granted, free of charge, to any person obtaining

---

a copy of the Unicode data files and any associated documentation (the "Data Files") or Unicode software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that either

- (a) this copyright and permission notice appear with all copies of the Data Files or Software, or
- (b) this copyright and permission notice appear in associated Documentation.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS.

IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.

---

#### Third-Party Software Licenses

This section contains third-party software notices and/or additional terms for licensed third-party software components included within ICU libraries.

---

ICU License - ICU 1.8.1 to ICU 57.1

#### COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1995-2016 International Business Machines Corporation and others  
All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

All trademarks and registered trademarks mentioned herein are the property of their respective owners.

-----  
Chinese/Japanese Word Break Dictionary Data (cjdict.txt)

```
# The Google Chrome software developed by Google is licensed under
# the BSD license. Other software included in this distribution is
# provided under other licenses, as set forth below.
#
# The BSD License
# http://opensource.org/licenses/bsd-license.php
# Copyright (C) 2006-2008, Google Inc.
#
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are met:
#
# Redistributions of source code must retain the above copyright notice,
# this list of conditions and the following disclaimer.
# Redistributions in binary form must reproduce the above
# copyright notice, this list of conditions and the following
# disclaimer in the documentation and/or other materials provided with
# the distribution.
# Neither the name of Google Inc. nor the names of its
# contributors may be used to endorse or promote products derived from
# this software without specific prior written permission.
#
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
# CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
# INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
# MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
# LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
# CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
# SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
# BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
```

```

# LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
# NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
# SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
#
# The word list in cjdict.txt are generated by combining three word lists
# listed below with further processing for compound word breaking. The
# frequency is generated with an iterative training against Google web
# corpora.
#
# * Libtabe (Chinese)
#   - https://sourceforge.net/project/?group\_id=1519
#   - Its license terms and conditions are shown below.
#
# * IPADIC (Japanese)
#   - http://chasen.aist-nara.ac.jp/chasen/distribution.html
#   - Its license terms and conditions are shown below.
#
# -----COPYING.libtabe ---- BEGIN-----
#
# /*
#  * Copyright (c) 1999 TaBE Project.
#  * Copyright (c) 1999 Pai-Hsiang Hsiao.
#  * All rights reserved.
#  *
#  * Redistribution and use in source and binary forms, with or without
#  * modification, are permitted provided that the following conditions
#  * are met:
#  *
#  * . Redistributions of source code must retain the above copyright
#  *   notice, this list of conditions and the following disclaimer.
#  * . Redistributions in binary form must reproduce the above copyright
#  *   notice, this list of conditions and the following disclaimer in
#  *   the documentation and/or other materials provided with the
#  *   distribution.
#  * . Neither the name of the TaBE Project nor the names of its
#  *   contributors may be used to endorse or promote products derived
#  *   from this software without specific prior written permission.
#  *
#  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
#  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
#  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
#  * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
#  * REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
#  * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#  * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
#  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
#  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
#  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
#  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
#  * OF THE POSSIBILITY OF SUCH DAMAGE.
#  */
#
# /*
#  * Copyright (c) 1999 Computer Systems and Communication Lab,

```

```

# *           Institute of Information Science, Academia
# *           Sinica. All rights reserved.
# *
# * Redistribution and use in source and binary forms, with or without
# * modification, are permitted provided that the following conditions
# * are met:
# *
# * . Redistributions of source code must retain the above copyright
# * notice, this list of conditions and the following disclaimer.
# * . Redistributions in binary form must reproduce the above copyright
# * notice, this list of conditions and the following disclaimer in
# * the documentation and/or other materials provided with the
# * distribution.
# * . Neither the name of the Computer Systems and Communication Lab
# * nor the names of its contributors may be used to endorse or
# * promote products derived from this software without specific
# * prior written permission.
# *
# * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
# * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
# * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
# * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
# * REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
# * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
# * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
# * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
# * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
# * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
# * OF THE POSSIBILITY OF SUCH DAMAGE.
# */
#
# Copyright 1996 Chih-Hao Tsai @ Beckman Institute,
# University of Illinois
# c-tsai4@uiuc.edu http://casper.beckman.uiuc.edu/~c-tsai4
#
# -----COPYING.libtabe-----END-----
#
# -----COPYING.ipadic-----BEGIN-----
#
# Copyright 2000, 2001, 2002, 2003 Nara Institute of Science
# and Technology. All Rights Reserved.
#
# Use, reproduction, and distribution of this software is permitted.
# Any copy of this software, whether in its original form or modified,
# must include both the above copyright notice and the following
# paragraphs.
#
# Nara Institute of Science and Technology (NAIST),
# the copyright holders, disclaims all warranties with regard to this
# software, including all implied warranties of merchantability and
# fitness, in no event shall NAIST be liable for
# any special, indirect or consequential damages or any damages
# whatsoever resulting from loss of use, data or profits, whether in an

```

```
# action of contract, negligence or other tortuous action, arising out
# of or in connection with the use or performance of this software.
#
# A large portion of the dictionary entries
# originate from ICOT Free Software. The following conditions for ICOT
# Free Software applies to the current dictionary as well.
#
# Each User may also freely distribute the Program, whether in its
# original form or modified, to any third party or parties, PROVIDED
# that the provisions of Section 3 ("NO WARRANTY") will ALWAYS appear
# on, or be attached to, the Program, which is distributed substantially
# in the same form as set out herein and that such intended
# distribution, if actually made, will neither violate or otherwise
# contravene any of the laws and regulations of the countries having
# jurisdiction over the User or the intended distribution itself.
#
# NO WARRANTY
#
# The program was produced on an experimental basis in the course of the
# research and development conducted during the project and is provided
# to users as so produced on an experimental basis. Accordingly, the
# program is provided without any warranty whatsoever, whether express,
# implied, statutory or otherwise. The term "warranty" used herein
# includes, but is not limited to, any warranty of the quality,
# performance, merchantability and fitness for a particular purpose of
# the program and the nonexistence of any infringement or violation of
# any right of any third party.
#
# Each user of the program will agree and understand, and be deemed to
# have agreed and understood, that there is no warranty whatsoever for
# the program and, accordingly, the entire risk arising from or
# otherwise connected with the program is assumed by the user.
#
# Therefore, neither ICOT, the copyright holder, or any other
# organization that participated in or was otherwise related to the
# development of the program and their respective officials, directors,
# officers and other employees shall be held liable for any and all
# damages, including, without limitation, general, special, incidental
# and consequential damages, arising out of or otherwise in connection
# with the use or inability to use the program or any product, material
# or result produced or otherwise obtained by using the program,
# regardless of whether they have been advised of, or otherwise had
# knowledge of, the possibility of such damages at any time during the
# project or thereafter. Each user will be deemed to have agreed to the
# foregoing by his or her commencement of use of the program. The term
# "use" as used herein includes, but is not limited to, the use,
# modification, copying and distribution of the program and the
# production of secondary products from the program.
#
# In the case where the program, whether in its original form or
# modified, was distributed or delivered to or received by a user from
# any person, organization or entity other than ICOT, unless it makes or
# grants independently of ICOT any specific warranty to the user in
# writing, such person, organization or entity, will also be exempted
# from and not be held liable to the user for any such damages as noted
```

```

# above as far as the program is concerned.
#
# -----COPYING.ipadic-----END-----

-----

Lao Word Break Dictionary Data (laodict.txt)

# Copyright (C) 2016 and later: Unicode, Inc. and others.
# License & terms of use: http://www.unicode.org/copyright.html
# Copyright (c) 2015 International Business Machines Corporation
# and others. All Rights Reserved.
#
# Project: https://github.com/rober42539/lao-dictionary
# Dictionary: https://github.com/rober42539/lao-dictionary/laodict.txt
# License: https://github.com/rober42539/lao-dictionary/LICENSE.txt
#         (copied below)
#
# This file is derived from the above dictionary version of Nov 22, 2020
# -----
# Copyright (C) 2013 Brian Eugene Wilson, Robert Martin Campbell.
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are met:
#
# Redistributions of source code must retain the above copyright notice, this
# list of conditions and the following disclaimer. Redistributions in binary
# form must reproduce the above copyright notice, this list of conditions and
# the following disclaimer in the documentation and/or other materials
# provided with the distribution.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
# "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
# FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
# COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
# INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
# (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
# SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
# STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
# ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
# OF THE POSSIBILITY OF SUCH DAMAGE.
# -----

-----

Burmese Word Break Dictionary Data (burmesedict.txt)

# Copyright (c) 2014 International Business Machines Corporation
# and others. All Rights Reserved.
#
# This list is part of a project hosted at:
# github.com/kanyawtech/myanmar-karen-word-lists

```



```

#
# -----
# Copyright (c) 2013, LeRoy Benjamin Sharon
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions
# are met: Redistributions of source code must retain the above
# copyright notice, this list of conditions and the following
# disclaimer. Redistributions in binary form must reproduce the
# above copyright notice, this list of conditions and the following
# disclaimer in the documentation and/or other materials provided
# with the distribution.
#
#   Neither the name Myanmar Karen Word Lists, nor the names of its
#   contributors may be used to endorse or promote products derived
#   from this software without specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
# CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
# INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
# MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
# BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
# EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
# TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
# DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
# ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
# TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
# THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
# SUCH DAMAGE.
# -----

```

## Time Zone Database

ICU uses the public domain data and code derived from Time Zone Database for its time zone support. The ownership of the TZ database is explained in BCP 175: Procedure for Maintaining the Time Zone Database section 7.

### # 7. Database Ownership

```

#
#   The TZ database itself is not an IETF Contribution or an IETF
#   document. Rather it is a pre-existing and regularly updated work
#   that is in the public domain, and is intended to remain in the
#   public domain. Therefore, BCPs 78 [RFC5378] and 79 [RFC3979] do
#   not apply to the TZ Database or contributions that individuals make
#   to it. Should any claims be made and substantiated against the TZ
#   Database, the organization that is providing the IANA
#   Considerations defined in this RFC, under the memorandum of
#   understanding with the IETF, currently ICANN, may act in accordance
#   with all competent court orders. No ownership claims will be made
#   by ICANN or the IETF Trust on the database or the code. Any person

```

---

```
# making a contribution to the database or code waives all rights to
# future claims in that contribution or in the TZ Database.
```

-----

Google double-conversion

Copyright 2006-2011, the V8 project authors. All rights reserved.  
Redistribution and use in source and binary forms, with or without  
modification, are permitted provided that the following conditions are  
met:

- \* Redistributions of source code must retain the above copyright  
notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above  
copyright notice, this list of conditions and the following  
disclaimer in the documentation and/or other materials provided  
with the distribution.
- \* Neither the name of Google Inc. nor the names of its  
contributors may be used to endorse or promote products derived  
from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS  
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT  
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR  
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT  
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,  
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT  
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,  
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY  
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE  
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

-----

File: aclocal.m4 (only for ICU4C)  
Section: pkg.m4 - Macros to locate and utilise pkg-config.

Copyright © 2004 Scott James Remnant <scott@netsplit.com>.  
Copyright © 2012-2015 Dan Nicholson <dbn.lists@gmail.com>

This program is free software; you can redistribute it and/or modify  
it under the terms of the GNU General Public License as published by  
the Free Software Foundation; either version 2 of the License, or  
(at your option) any later version.

This program is distributed in the hope that it will be useful, but  
WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU  
General Public License for more details.

You should have received a copy of the GNU General Public License  
along with this program; if not, write to the Free Software

---

Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  
02111-1307, USA.

As a special exception to the GNU General Public License, if you distribute this file as part of a program that contains a configuration script generated by Autoconf, you may include it under the same distribution terms that you use for the rest of that program.

(The condition for the exception is fulfilled because ICU4C includes a configuration script generated by Autoconf, namely the `configure` script.)

-----  
File: config.guess (only for ICU4C)

This file is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, see <<https://www.gnu.org/licenses/>>.

As a special exception to the GNU General Public License, if you distribute this file as part of a program that contains a configuration script generated by Autoconf, you may include it under the same distribution terms that you use for the rest of that program. This Exception is an additional permission under section 7 of the GNU General Public License, version 3 ("GPLv3").

(The condition for the exception is fulfilled because ICU4C includes a configuration script generated by Autoconf, namely the `configure` script.)

-----  
File: install-sh (only for ICU4C)

Copyright 1991 by the Massachusetts Institute of Technology

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting

---

documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

#### isa-l

Copyright(c) 2011-2024 Intel Corporation All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

SPDX-License-Identifier: BSD-3-Clause

#### isa-l\_crypto

Copyright(c) 2011-2024 Intel Corporation All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

SPDX-License-Identifier: BSD-3-Clause

## jansson

Copyright (c) 2009-2020 Petri Lehtinen <petri@digip.org>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## jemalloc

Unless otherwise specified, files in the jemalloc source distribution are subject to the following license:

-----  
Copyright (C) 2002-present Jason Evans <jasone@canonware.com>.

All rights reserved.

Copyright (C) 2007-2012 Mozilla Foundation. All rights reserved.

Copyright (C) 2009-present Facebook, Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice(s), this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice(s), this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER(S) ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF

MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER(S) BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

-----

## jsoncpp

The JsonCpp library's source code, including accompanying documentation, tests and demonstration applications, are licensed under the following conditions...

Baptiste Lepilleur and The JsonCpp Authors explicitly disclaim copyright in all jurisdictions which recognize such a disclaimer. In such jurisdictions, this software is released into the Public Domain.

In jurisdictions which do not recognize Public Domain property (e.g. Germany as of 2010), this software is Copyright (c) 2007-2010 by Baptiste Lepilleur and The JsonCpp Authors, and is released under the terms of the MIT License (see below).

In jurisdictions which recognize Public Domain property, the user of this software may choose to accept it either as 1) Public Domain, 2) under the conditions of the MIT License (see below), or 3) under the terms of dual Public Domain/MIT License conditions described here, as they choose.

The MIT License is about as close to Public Domain as a license can get, and is described in clear, concise terms at:

[http://en.wikipedia.org/wiki/MIT\\_License](http://en.wikipedia.org/wiki/MIT_License)

The full text of the MIT License follows:

=====  
Copyright (c) 2007-2010 Baptiste Lepilleur and The JsonCpp Authors

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN

---

ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

=====  
(END LICENSE TEXT)

The MIT license is compatible with both the GPL and commercial software, affording one all of the rights of Public Domain with the minor nuisance of being required to keep the above copyright notice and license text in the source code. Note also that by accepting the Public Domain "license" you can re-license your copy using whatever license you like.

## krb5

Copyright (C) 1985-2024 by the Massachusetts Institute of Technology.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Downloading of this software may constitute an export of cryptographic software from the United States of America that is subject to the United States Export Administration Regulations (EAR), 15 CFR 730-774. Additional laws or regulations may apply. It is the responsibility of the person or entity contemplating export to comply with all applicable export laws and regulations, including obtaining any required license from the U.S. government.

The U.S. government prohibits export of encryption source code to certain countries and individuals, including, but not limited to, the countries of Cuba, Iran, North Korea, Sudan, Syria, and residents and nationals of those countries.

Documentation components of this software distribution are licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. (<https://creativecommons.org/licenses/by-sa/3.0/>)

Individual source code files are copyright MIT, Cygnus Support, Novell, OpenVision Technologies, Oracle, Red Hat, Sun Microsystems, FundsXpress, and others.

Project Athena, Athena, Athena MUSE, Discuss, Hesiod, Kerberos, Moira, and Zephyr are trademarks of the Massachusetts Institute of Technology (MIT). No commercial use of these trademarks may be made without prior written permission of MIT.

"Commercial use" means use of a name in a product or other for-profit manner. It does NOT prevent a commercial firm from referring to the MIT trademarks in order to convey information (although in doing so, recognition of their

trademark status should be given).

=====

The following copyright and permission notice applies to the OpenVision Kerberos Administration system located in “kadmin/create”, “kadmin/dbutil”, “kadmin/passwd”, “kadmin/server”, “lib/kadm5”, and portions of “lib/rpc”:

Copyright, OpenVision Technologies, Inc., 1993-1996, All Rights Reserved

WARNING: Retrieving the OpenVision Kerberos Administration system source code, as described below, indicates your acceptance of the following terms. If you do not agree to the following terms, do not retrieve the OpenVision Kerberos administration system.

You may freely use and distribute the Source Code and Object Code compiled from it, with or without modification, but this Source Code is provided to you “AS IS” EXCLUSIVE OF ANY WARRANTY, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, OR ANY OTHER WARRANTY, WHETHER EXPRESS OR IMPLIED. IN NO EVENT WILL OPENVISION HAVE ANY LIABILITY FOR ANY LOST PROFITS, LOSS OF DATA OR COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS AGREEMENT, INCLUDING, WITHOUT LIMITATION, THOSE RESULTING FROM THE USE OF THE SOURCE CODE, OR THE FAILURE OF THE SOURCE CODE TO PERFORM, OR FOR ANY OTHER REASON.

OpenVision retains all copyrights in the donated Source Code. OpenVision also retains copyright to derivative works of the Source Code, whether created by OpenVision or by a third party. The OpenVision copyright notice must be preserved if derivative works are made based on the donated Source Code.

OpenVision Technologies, Inc. has donated this Kerberos Administration system to MIT for inclusion in the standard Kerberos 5 distribution. This donation underscores our commitment to continuing Kerberos technology development and our gratitude for the valuable work which has been performed by MIT and the Kerberos community.

=====

Portions contributed by Matt Crawford “crawdad@fnal.gov” were work performed at Fermi National Accelerator Laboratory, which is operated by Universities Research Association, Inc., under contract DE-AC02-76CHO3000 with the U.S. Department of Energy.

=====

Portions of “src/lib/crypto” have the following copyright:

Copyright (C) 1998 by the FundsXpress, INC.

All rights reserved.

Export of this software from the United States of America may  
require a specific license from the United States Government.  
It is the responsibility of any person or organization  
contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of FundsXpress. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. FundsXpress makes no representations about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty.

THIS SOFTWARE IS PROVIDED “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

=====

The implementation of the AES encryption algorithm in “src/lib/crypto/builtin/aes” has the following copyright:



---

Copyright (C) 1998-2013, Brian Gladman, Worcester, UK. All rights reserved.

The redistribution and use of this software (with or without changes) is allowed without the payment of fees or royalties provided that:

source code distributions include the above copyright notice,  
this list of conditions and the following disclaimer;

binary distributions include the above copyright notice, this  
list of conditions and the following disclaimer in their  
documentation.

This software is provided 'as is' with no explicit or implied warranties in respect of its operation, including, but not limited to, correctness and fitness for purpose.

=====

Portions contributed by Red Hat, including the pre-authentication plug-in framework and the NSS crypto implementation, contain the following copyright:

Copyright (C) 2006 Red Hat, Inc.  
Portions copyright (C) 2006 Massachusetts Institute of Technology  
All Rights Reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Red Hat, Inc., nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====

The bundled verito source code is subject to the following license:

Copyright 2011 Red Hat, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PAR-

---

TICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

=====

The MS-KKDCP client implementation has the following copyright:

Copyright 2013,2014 Red Hat, Inc.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====

The implementations of GSSAPI mechglue in GSSAPI-SPNEGO in “src/lib/gssapi”, including the following files:

lib/gssapi/generic/gssapi\_err\_generic.et lib/gssapi/mechglue/g\_accept\_sec\_context.c lib/gssapi/mechglue/g\_acquire\_cred.c lib/gssapi/mechglue/g\_canon\_name.c lib/gssapi/mechglue/g\_compare\_name.c lib/gssapi/mechglue/g\_context\_time.c lib/gssapi/mechglue/g\_delete\_sec\_context.c lib/gssapi/mechglue/g\_dsp\_name.c lib/gssapi/mechglue/g\_dsp\_status.c lib/gssapi/mechglue/g\_dup\_name.c lib/gssapi/mechglue/g\_exp\_sec\_context.c lib/gssapi/mechglue/g\_export\_name.c lib/gssapi/mechglue/g\_glue.c lib/gssapi/mechglue/g\_imp\_name.c lib/gssapi/mechglue/g\_imp\_sec\_context.c lib/gssapi/mechglue/g\_init\_sec\_context.c lib/gssapi/mechglue/g\_initialize.c lib/gssapi/mechglue/g\_inquire\_context.c lib/gssapi/mechglue/g\_inquire\_cred.c lib/gssapi/mechglue/g\_inquire\_names.c lib/gssapi/mechglue/g\_process\_context.c lib/gssapi/mechglue/g\_rel\_buffer.c lib/gssapi/mechglue/g\_rel\_cred.c lib/gssapi/mechglue/g\_rel\_name.c lib/gssapi/mechglue/g\_rel\_oid\_set.c lib/gssapi/mechglue/g\_seal.c lib/gssapi/mechglue/g\_sign.c lib/gssapi/mechglue/g\_store\_cred.c lib/gssapi/mechglue/g\_unseal.c lib/gssapi/mechglue/g\_userok.c lib/gssapi/mechglue/g\_utils.c lib/gssapi/mechglue/g\_verify.c lib/gssapi/mechglue/gssd\_pname\_to\_uid.c lib/gssapi/mechglue/mglueP.h lib/gssapi/mechglue/oid\_ops.c lib/gssapi/spnego/gssapiP\_spnego.h lib/gssapi/spnego/spnego\_mech.c

and the initial implementation of incremental propagation, including the following new or changed files:

include/iprop\_hdr.h kadmin/server/ipropd\_svc.c lib/kdb/iprop.x lib/kdb/kdb\_convert.c lib/kdb/kdb\_log.c lib/kdb/kdb\_log.h lib/krb5/error\_tables/kdb5\_err.et kprop/kpropd\_rpc.c kprop/kproplog.c

are subject to the following license:

Copyright (C) 2004 Sun Microsystems, Inc.

---

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

=====

Kerberos V5 includes documentation and software developed at the University of California at Berkeley, which includes this copyright notice:

Copyright (C) 1983 Regents of the University of California.  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====

Portions contributed by Novell, Inc., including the LDAP database backend, are subject to the following license:

Copyright (C) 2004-2005, Novell, Inc.  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- The copyright holder's name is not used to endorse or promote products derived from this software without specific prior written permission.

---

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====

Portions funded by Sandia National Laboratory and developed by the University of Michigan's Center for Information Technology Integration, including the PKINIT implementation, are subject to the following license:

COPYRIGHT (C) 2006-2007  
THE REGENTS OF THE UNIVERSITY OF MICHIGAN  
ALL RIGHTS RESERVED

Permission is granted to use, copy, create derivative works and redistribute this software and such derivative works for any purpose, so long as the name of The University of Michigan is not used in any advertising or publicity pertaining to the use of distribution of this software without specific, written prior authorization. If the above copyright notice or any other identification of the University of Michigan is included in any copy of any portion of this software, then the disclaimer below must also be included.

THIS SOFTWARE IS PROVIDED AS IS, WITHOUT REPRESENTATION FROM THE UNIVERSITY OF MICHIGAN AS TO ITS FITNESS FOR ANY PURPOSE, AND WITHOUT WARRANTY BY THE UNIVERSITY OF MICHIGAN OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE REGENTS OF THE UNIVERSITY OF MICHIGAN SHALL NOT BE LIABLE FOR ANY DAMAGES, INCLUDING SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WITH RESPECT TO ANY CLAIM ARISING OUT OF OR IN CONNECTION WITH THE USE OF THE SOFTWARE, EVEN IF IT HAS BEEN OR IS HEREAFTER ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

=====

The pkcs11.h file included in the PKINIT code has the following license:

Copyright 2006 g10 Code GmbH  
Copyright 2006 Andreas Jellinghaus

This file is free software; as a special exception the author gives unlimited permission to copy and/or distribute it, with or without modifications, as long as this notice is preserved.

This file is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY, to the extent permitted by law; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

=====

Portions contributed by Apple Inc. are subject to the following license:

Copyright 2004-2008 Apple Inc. All Rights Reserved.

Export of this software from the United States of America may  
require a specific license from the United States Government.  
It is the responsibility of any person or organization  
contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the

---

name of Apple Inc. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Apple Inc. makes no representations about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty.

THIS SOFTWARE IS PROVIDED “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

=====

The implementations of UTF-8 string handling in src/util/support and src/lib/krb5/unicode are subject to the following copyright and permission notice:

The OpenLDAP Public License  
Version 2.8, 17 August 2003

Redistribution and use of this software and associated documentation (“Software”), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions in source form must retain copyright statements and notices,
2. Redistributions in binary form must reproduce applicable copyright statements and notices, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution, and
3. Redistributions must contain a verbatim copy of this document.

The OpenLDAP Foundation may revise this license from time to time. Each revision is distinguished by a version number. You may use this Software under terms of this license revision or under the terms of any subsequent revision of the license.

THIS SOFTWARE IS PROVIDED BY THE OPENLDAP FOUNDATION AND ITS CONTRIBUTORS “AS IS” AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OPENLDAP FOUNDATION, ITS CONTRIBUTORS, OR THE AUTHOR(S) OR OWNER(S) OF THE SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The names of the authors and copyright holders must not be used in advertising or otherwise to promote the sale, use or other dealing in this Software without specific, written prior permission. Title to copyright in this Software shall at all times remain with copyright holders.

OpenLDAP is a registered trademark of the OpenLDAP Foundation.

Copyright 1999-2003 The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved. Permission to copy and distribute verbatim copies of this document is granted.

=====

Marked test programs in src/lib/krb5/krb have the following copyright:

Copyright (C) 2006 Kungliga Tekniska Högskola  
(Royal Institute of Technology, Stockholm, Sweden).  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- 
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
  3. Neither the name of KTH nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY KTH AND ITS CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL KTH OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====

The KCM Mach RPC definition file used on macOS has the following copyright:

Copyright (C) 2009 Kungliga Tekniska Högskola  
(Royal Institute of Technology, Stockholm, Sweden).  
All rights reserved.

Portions Copyright (C) 2009 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====

Portions of the RPC implementation in src/lib/rpc and src/include/gssrpc have the following copyright and permission notice:

Copyright (C) 2010, Oracle America, Inc.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- 
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
  3. Neither the name of the “Oracle America, Inc.” nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====  
Copyright (C) 2006,2007,2009 NTT (Nippon Telegraph and Telephone Corporation). All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY NTT “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL NTT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====  
Copyright 2000 by Carnegie Mellon University

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Carnegie Mellon University not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

CARNEGIE MELLON UNIVERSITY DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL CARNEGIE MELLON UNIVERSITY BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

=====  
Copyright (C) 2002 Naval Research Laboratory (NRL/CCS)

---

Permission to use, copy, modify and distribute this software and its documentation is hereby granted, provided that both the copyright notice and this permission notice appear in all copies of the software, derivative works or modified versions, and any portions thereof.

NRL ALLOWS FREE USE OF THIS SOFTWARE IN ITS “AS IS” CONDITION AND DISCLAIMS ANY LIABILITY OF ANY KIND FOR ANY DAMAGES WHATSOEVER RESULTING FROM THE USE OF THIS SOFTWARE.

=====

Copyright (C) 2022 United States Government as represented by the Secretary of the Navy. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====

Copyright (C) 1991, 1992, 1994 by Cygnus Support.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Cygnus Support makes no representations about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty.

=====

Copyright (C) 2006 Secure Endpoints Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

=====

Portions of the implementation of the Fortuna-like PRNG are subject to the following notice:



---

Copyright (C) 2005 Marko Kreen  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (C) 1994 by the University of Southern California

EXPORT OF THIS SOFTWARE from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to copy, modify, and distribute this software and its documentation in source and binary forms is hereby granted, provided that any documentation or other materials related to such distribution or use acknowledge that the software was developed by the University of Southern California.

DISCLAIMER OF WARRANTY. THIS SOFTWARE IS PROVIDED "AS IS". The University of Southern California MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. By way of example, but not limitation, the University of Southern California MAKES NO REPRESENTATIONS OR WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. The University of Southern California shall not be held liable for any liability nor for any direct, indirect, or consequential damages with respect to any claim by the user or distributor of the ksu software.

=====  
Copyright (C) 1995  
The President and Fellows of Harvard University

This code is derived from software contributed to Harvard by Jeremy Rassen.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:  
This product includes software developed by the University of California, Berkeley and its contributors.
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

---

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====  
Copyright (C) 2008 by the Massachusetts Institute of Technology.  
Copyright 1995 by Richard P. Basch. All Rights Reserved.  
Copyright 1995 by Lehman Brothers, Inc. All Rights Reserved.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Richard P. Basch, Lehman Brothers and M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Richard P. Basch, Lehman Brothers and M.I.T. make no representations about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty.

=====  
The following notice applies to “src/lib/krb5/krb/strptime.c” and “src/include/k5-queue.h”.

Copyright (C) 1997, 1998 The NetBSD Foundation, Inc.  
All rights reserved.

This code was contributed to The NetBSD Foundation by Klaus Klein.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:  
This product includes software developed by the NetBSD Foundation, Inc. and its contributors.
4. Neither the name of The NetBSD Foundation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE NETBSD FOUNDATION, INC. AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FOUNDATION OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN

---

ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====

The following notice applies to Unicode library files in “src/lib/krb5/unicode”:

Copyright 1997, 1998, 1999 Computing Research Labs,  
New Mexico State University

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE COMPUTING RESEARCH LAB OR NEW MEXICO STATE UNIVERSITY BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

=====

The following notice applies to “src/util/support/strncpy.c”:

Copyright (C) 1998 Todd C. Miller “Todd.Miller@courtesan.com”

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED “AS IS” AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

=====

The following notice applies to “src/util/profile/argv\_parse.c” and “src/util/profile/argv\_parse.h”:

Copyright 1999 by Theodore Ts'o.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies. THE SOFTWARE IS PROVIDED “AS IS” AND THEODORE TS'O (THE AUTHOR) DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. (Isn't it sick that the U.S. culture of lawsuit-happy lawyers requires this kind of disclaimer?)

=====

The following notice applies to SWIG-generated code in “src/util/profile/profile\_tcl.c”:

Copyright (C) 1999-2000, The University of Chicago

This file may be freely redistributed without license or fee provided this copyright message remains intact.

=====

The following notice applies to portions of “src/lib/rpc” and “src/include/gssrpc”:

Copyright (C) 2000 The Regents of the University of Michigan. All rights reserved.

Copyright (C) 2000 Dug Song “dugsong@UMICH.EDU”. All rights reserved, all wrongs reversed.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====

Implementations of the MD4 algorithm are subject to the following notice:

Copyright (C) 1990, RSA Data Security, Inc. All rights reserved.

License to copy and use this software is granted provided that it is identified as the “RSA Data Security, Inc. MD4 Message Digest Algorithm” in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as “derived from the RSA Data Security, Inc. MD4 Message Digest Algorithm” in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided “as is” without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

=====

Implementations of the MD5 algorithm are subject to the following notice:

Copyright (C) 1990, RSA Data Security, Inc. All rights reserved.

License to copy and use this software is granted provided that it is identified as the “RSA Data Security, Inc. MD5 Message- Digest Algorithm” in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as “derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm” in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided “as is” without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

=====

---

The following notice applies to “src/lib/crypto/crypto\_tests/t\_mdmdriver.c”:

Copyright (C) 1990-2, RSA Data Security, Inc. Created 1990. All rights reserved.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided “as is” without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

=====

Portions of “src/lib/krb5” are subject to the following notice:

Copyright (C) 1994 CyberSAFE Corporation.  
Copyright 1990,1991,2007,2008 by the Massachusetts  
Institute of Technology.  
All Rights Reserved.

Export of this software from the United States of America may  
require a specific license from the United States Government. It  
is the responsibility of any person or organization  
contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Furthermore if you modify this software you must label your software as modified software and not distribute it in such a fashion that it might be confused with the original M.I.T. software. Neither M.I.T., the Open Computing Security Group, nor CyberSAFE Corporation make any representations about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty.

=====

Portions contributed by PADL Software are subject to the following license:

Copyright (c) 2011, PADL Software Pty Ltd. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of PADL Software nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY PADL SOFTWARE AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL PADL SOFTWARE OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====

The bundled libev source code is subject to the following license:

---

All files in libev are Copyright (C)2007,2008,2009 Marc Alexander Lehmann.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Alternatively, the contents of this package may be used under the terms of the GNU General Public License (“GPL”) version 2 or any later version, in which case the provisions of the GPL are applicable instead of the above. If you wish to allow the use of your version of this package only under the terms of the GPL and not to allow others to use your version of this file under the BSD license, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL in this and the other files of this package. If you do not delete the provisions above, a recipient may use your version of this file under either the BSD or the GPL.

=====

Files copied from the Intel AESNI Sample Library are subject to the following license:

Copyright (C) 2010, Intel Corporation All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

---

ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====

The following notice applies to “src/ccapi/common/win/OldCC/autolock.hxx”:

Copyright (C) 1998 by Danilo Almeida. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====

The following notice applies to portions of “src/plugins/preauth/spake/edwards25519.c” and “src/plugins/preauth/spake/edwards25519\_tables.h”:

The MIT License (MIT)

Copyright (c) 2015-2016 the fiat-crypto authors (see the AUTHORS file).

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

=====

The following notice applies to portions of “src/plugins/preauth/spake/edwards25519.c”:

Copyright (c) 2015-2016, Google Inc.

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED “AS IS” AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR

---

CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

## libaio

### GNU LESSER GENERAL PUBLIC LICENSE Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.  
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA  
Everyone is permitted to copy and distribute verbatim copies  
of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts  
as the successor of the GNU Library Public License, version 2, hence  
the version number 2.1.]

#### Preamble

The licenses for most software are designed to take away your  
freedom to share and change it. By contrast, the GNU General Public  
Licenses are intended to guarantee your freedom to share and change  
free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some  
specially designated software packages--typically libraries--of the  
Free Software Foundation and other authors who decide to use it. You  
can use it too, but we suggest you first think carefully about whether  
this license or the ordinary General Public License is the better  
strategy to use in any particular case, based on the explanations  
below.

When we speak of free software, we are referring to freedom of use,  
not price. Our General Public Licenses are designed to make sure that  
you have the freedom to distribute copies of free software (and charge  
for this service if you wish); that you receive source code or can get  
it if you want it; that you can change the software and use pieces of  
it in new free programs; and that you are informed that you can do  
these things.

To protect your rights, we need to make restrictions that forbid  
distributors to deny you these rights or to ask you to surrender these  
rights. These restrictions translate to certain responsibilities for  
you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis  
or for a fee, you must give the recipients all the rights that we gave  
you. You must make sure that they, too, receive or can get the source  
code. If you link other code with the library, you must provide  
complete object files to the recipients, so that they can relink them  
with the library after making changes to the library and recompiling  
it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the



---

library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

^L

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

^L

GNU LESSER GENERAL PUBLIC LICENSE  
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

---

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public

---

License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

^L

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may

distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

^L

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

- a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
- b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
- c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
- d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
- e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is

normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

^L

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.
- b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

^L

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or

otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

^L

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our

decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

#### NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

#### END OF TERMS AND CONDITIONS

^L

#### How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>

Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of



MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library `Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990  
Ty Coon, President of Vice

That's all there is to it!

## libfuse

The following files may be used under the terms of the GNU Lesser General Public License, version 2.1 ("LGPL"):

- All files in the include/ directory.
- All files in the lib/ directory.
- meson.build

The full terms of the LGPL can be found in the LGPL2.txt file.

All other files may be used only under the terms of the GNU General Public License, version 2 ("GPL"). The full text of this license can be found in the GPL2.txt file.

## librdkafka

### LICENSE

-----  
librdkafka - Apache Kafka C driver library

Copyright (c) 2012-2022, Magnus Edenhill  
2023, Confluent Inc.  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,

this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

LICENSE.cjson

-----  
For cJSON.c and cJSON.h:

Copyright (c) 2009-2017 Dave Gamble and cJSON contributors

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

LICENSE.crc32c

-----  
# For src/crc32c.c copied (with modifications) from  
# <http://stackoverflow.com/a/17646775/1821055>

/\* crc32c.c -- compute CRC-32C using the Intel crc32 instruction  
 \* Copyright (C) 2013 Mark Adler  
 \* Version 1.1 1 Aug 2013 Mark Adler  
 \*/

/\*

---

This software is provided 'as-is', without any express or implied warranty. In no event will the author be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Mark Adler  
madler@alumni.caltech.edu  
\*/

LICENSE.fnv1a

-----  
parts of src/rdfnv1a.c: [http://www.isthe.com/chongo/src/fnv/hash\\_32a.c](http://www.isthe.com/chongo/src/fnv/hash_32a.c)

Please do not copyright this code. This code is in the public domain.

LANDON CURT NOLL DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL LANDON CURT NOLL BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

By:  
chongo <Landon Curt Noll> /\oo\  
<http://www.isthe.com/chongo/>

Share and Enjoy! :-)

LICENSE.hdrhistogram

-----  
This license covers src/rdhdrhistogram.c which is a C port of  
Coda Hale's Golang HdrHistogram <https://github.com/codahale/hdrhistogram>  
at revision 3a0bb77429bd3a61596f5e8a3172445844342120

-----  
The MIT License (MIT)

Copyright (c) 2014 Coda Hale

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE

LICENSE.lz4

-----  
src/rdxxhash.[ch] src/lz4\*.[ch]: git@github.com:lz4/lz4.git 5ff839680134437dbf4678f3d0c7b371d84f4964

LZ4 Library

Copyright (c) 2011-2020, Yann Collet

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

LICENSE.murmur2

-----  
parts of src/rdmurmur2.c: git@github.com:abandoned/murmur2.git

MurMurHash2 Library

---

```
//-----  
// MurmurHash2 was written by Austin Appleby, and is placed in the public  
// domain. The author hereby disclaims copyright to this source code.
```

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

#### LICENSE.nanopb

-----  
For files in src/nanopb : <https://github.com/nanopb/nanopb/blob/8ef41e0ebd45daaf19459a011f67e66224b247cd/L>

Copyright (c) 2011 Petteri Aimonen <jpa at nanopb.mail.kapsi.fi>

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

#### LICENSE.opentelemetry

-----  
For files in src/opentelemetry: <https://github.com/open-telemetry/opentelemetry-proto/blob/81a296f9dba23e3>

Apache License  
Version 2.0, January 2004

## TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the

Licensors for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
  - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or

documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity,



or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

LICENSE.pycrc

-----  
The following license applies to the files rdcrc32.c and rdcrc32.h which have been generated by the pycrc tool.  
=====

Copyright (c) 2006-2012, Thomas Pircher <tehpeh@gmx.net>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

LICENSE.queue

-----  
For sys/queue.h:

```
* Copyright (c) 1991, 1993
* The Regents of the University of California. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in the
* documentation and/or other materials provided with the distribution.
* 4. Neither the name of the University nor the names of its contributors
* may be used to endorse or promote products derived from this software
* without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* @(#)queue.h 8.5 (Berkeley) 8/20/94
* $FreeBSD$
```

LICENSE.regexp

-----  
regexp.c and regexp.h from <https://github.com/ccxvii/minilibs> sha 875c33568b5a4aa4fb3dd0c52ea98f7f0e5ca684

"

These libraries are in the public domain (or the equivalent where that is not possible). You can do anything you want with them.

"

LICENSE.snappy

-----  
#####  
# LICENSE.snappy covers files: snappy.c, snappy.h, snappy\_compat.h #  
# originally retrieved from <http://github.com/andikleen/snappy-c> #

```
# git revision 8015f2d28739b9a6076ebaa6c53fe27bc238d219 #
#####
```

The snappy-c code is under the same license as the original snappy source

Copyright 2011 Intel Corporation All Rights Reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- \* Neither the name of Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

LICENSE.tinycthread

-----  
From <https://github.com/tinycthread/tinycthread/README.txt> c57166cd510ffb5022dd5f127489b131b61441b9

License

-----

Copyright (c) 2012 Marcus Geelnard  
2013-2014 Evan Nemerson

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software

in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

3. This notice may not be removed or altered from any source distribution.

LICENSE.wingetopt

-----  
For the files wingetopt.c wingetopt.h downloaded from <https://github.com/alex85k/wingetopt>

```
/*
 * Copyright (c) 2002 Todd C. Miller <Todd.Miller@courtesan.com>
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose with or without fee is hereby granted, provided that the above
 * copyright notice and this permission notice appear in all copies.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
 *
 * Sponsored in part by the Defense Advanced Research Projects
 * Agency (DARPA) and Air Force Research Laboratory, Air Force
 * Materiel Command, USAF, under agreement number F39502-99-1-0512.
 */
/*-
 * Copyright (c) 2000 The NetBSD Foundation, Inc.
 * All rights reserved.
 *
 * This code is derived from software contributed to The NetBSD Foundation
 * by Dieter Baron and Thomas Klausner.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE NETBSD FOUNDATION, INC. AND CONTRIBUTORS
 * ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
 * TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FOUNDATION OR CONTRIBUTORS
 * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
```

```
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
*/
```

## libuuid

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, and the entire permission notice in its entirety, including the disclaimer of warranties.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ALL OF WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF NOT ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## libuv

Copyright (c) 2015-present libuv project contributors.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

#### lz4

LZ4 Library  
Copyright (c) 2011-2020, Yann Collet  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

#### openssl

Apache License  
Version 2.0, January 2004  
<https://www.apache.org/licenses/>

##### TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

##### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work,

where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
  - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of



this License, without any additional terms or conditions.  
Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

## pcr2

### PCRE2 LICENCE

-----

PCRE2 is a library of functions to support regular expressions whose syntax and semantics are as close as possible to those of the Perl 5 language.

Releases 10.00 and above of PCRE2 are distributed under the terms of the "BSD" licence, as specified below, with one exemption for certain binary

---

redistributions. The documentation for PCRE2, supplied in the "doc" directory, is distributed under the same terms as the software itself. The data in the testdata directory is not copyrighted and is in the public domain.

The basic library functions are written in C and are freestanding. Also included in the distribution is a just-in-time compiler that can be used to optimize pattern matching. This is an optional feature that can be omitted when the library is built.

#### THE BASIC LIBRARY FUNCTIONS

-----

Written by: Philip Hazel  
Email local part: Philip.Hazel  
Email domain: gmail.com

Retired from University of Cambridge Computing Service,  
Cambridge, England.

Copyright (c) 1997-2024 University of Cambridge  
All rights reserved.

#### PCRE2 JUST-IN-TIME COMPILATION SUPPORT

-----

Written by: Zoltan Herczeg  
Email local part: hzmester  
Email domain: freemail.hu

Copyright(c) 2010-2024 Zoltan Herczeg  
All rights reserved.

#### STACK-LESS JUST-IN-TIME COMPILER

-----

Written by: Zoltan Herczeg  
Email local part: hzmester  
Email domain: freemail.hu

Copyright(c) 2009-2024 Zoltan Herczeg  
All rights reserved.

#### THE "BSD" LICENCE

-----

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notices, this list of conditions and the following disclaimer.

- \* Redistributions in binary form must reproduce the above copyright notices, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of the University of Cambridge nor the names of any contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

#### EXEMPTION FOR BINARY LIBRARY-LIKE PACKAGES

-----

The second condition in the BSD licence (covering binary redistributions) does not apply all the way down a chain of software. If binary package A includes PCRE2, it must respect the condition, but if package B is software that includes package A, the condition is not imposed on package B unless it uses PCRE2 independently.

End

#### pstreams

Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,

ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## scrypt

The included code and documentation ("scrypt") is distributed under the following terms:

Copyright 2005-2016 Colin Percival. All rights reserved.  
Copyright 2005-2016 Tarsnap Backup Inc. All rights reserved.  
Copyright 2014 Sean Kelly. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## spdk

### BSD LICENSE

Copyright (c) Intel Corporation.  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

strtk

[COPYRIGHT NOTICE]

Free use of the String Toolkit Library is permitted under the guidelines and in accordance with the most current version of the MIT License.

<http://www.opensource.org/licenses/MIT>

tbb

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

#### TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

##### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the

Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

- (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
- (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
- (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
- (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

- 5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
- 6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
- 7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each

Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.



---

See the License for the specific language governing permissions and limitations under the License.

## **tclap**

Copyright (c) 2003-2012 Michael E. Smoot  
Copyright (c) 2004-2016 Daniel Aarno  
Copyright (c) 2017-2021 Google LLC

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## **libudev, part of systemd**

```
/**
 * This file is part of systemd.
 *
 * Copyright 2008-2012 Kay Sievers <kay@vrfy.org>
 *
 * systemd is free software; you can redistribute it and/or modify it
 * under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * systemd is distributed in the hope that it will be useful, but
 * WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with systemd; If not, see <http://www.gnu.org/licenses/>.
 */
```

## **xxhash**

---

xxHash Library  
Copyright (c) 2012-2021 Yann Collet  
All rights reserved.

BSD 2-Clause License (<https://www.opensource.org/licenses/bsd-license.php>)

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## yaml-cpp

Copyright (c) 2008-2015 Jesse Beder.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## zlib

/\* zlib.h -- interface of the 'zlib' general purpose compression library  
version 1.3.1, January 22nd, 2024

Copyright (C) 1995-2024 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

|                  |                           |
|------------------|---------------------------|
| Jean-loup Gailly | Mark Adler                |
| jloup@gzip.org   | madler@alumni.caltech.edu |

The data format used by the zlib library is described by RFCs (Request for Comments) 1950 to 1952 in the files <http://tools.ietf.org/html/rfc1950> (zlib format), rfc1951 (deflate format) and rfc1952 (gzip format).

\*/

zstd

BSD License

For Zstandard software

Copyright (c) Meta Platforms, Inc. and affiliates. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name Facebook, nor Meta, nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

---

(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.