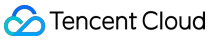
Data Lake Compute

FAQs

Product Documentation

Copyright Notice

Trademark Notice



Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

FAQs

FAQs on Permissions

Last updated：2025-07-30 15:10:53

Why Should I Enable Both Cloud Access Management (CAM) Permissions and Data Lake Compute (DLC) Permissions?

Type	Description
CAM Permissions	Grant the DLC access permissions to a sub-account.
DLC Permissions	After adding a CAM sub-account as a DLC user, you can configure and manage the permissions for databases, tables, and engines.

Why Am I Unable to Use Data Explore (Data Job) After Activating the Sub-account?

After account activation, DLC administrators should enable the role attributes, engine usage permissions, and database and table read-write permissions for the sub-account through the User and Permission Management feature in DLC.

FAQs on Engines

Last updated：2025-07-30 15:10:53

What Are the Differences Between Shared Engines and Private Engines?

Engine Type	Description	Billing Mode	Features
Shared Engine	A public engine used by all users in the current region.	Pay-as-you-go: Billing is based on the scan volume. No fee is incurred when not in use.	1. Configuration is not required. 2. Suitable for small-scale data or temporary data computing scenarios.
Private Engine	Engine resources exclusively allocated to the user.	Pay-as-you-go: Billing is based on compute unit (CU) usage. The cluster can be suspended when there is no task. No fees incur during cluster suspension.	1. Private resources with configurable resource scale and auto scaling. 2. Data computing scenarios with moderate workloads but irregular execution patterns.
		Monthly subscription: Billing is based on CU usage with on-demand cluster availability.	1. Private resources with configurable resource scale and auto scaling. 2. Data computing scenarios with a large and stable workloads.

How Many Tasks Can Run in Parallel in a Single Cluster? Can This Be Adjusted?

A single cluster supports up to 5 concurrent tasks by default. If adjustments are necessary, go to Standard Engine feature, select the target engine, and click Specification Configurations to make adjustments.

Why Is the Actual Number of CU Cores Used by A Task Less than the CU Cores Specified in the Cluster Configurations?

The following scenarios may result in fewer CU cores being utilized than configured:

1. Other tasks are running in the cluster.
2. The sum of driver and executor resources allocated for the batch job is less than the total configured cluster CU cores.

3. For pay-as-you-go clusters, resources are only allocated upon usage. When requesting large CU allocations, full capacity cannot be guaranteed.
4. The batch job is configured with enhanced networking to connect to another Virtual Private Cloud (VPC), but the target VPC lacks sufficient IP addresses to launch all executors.

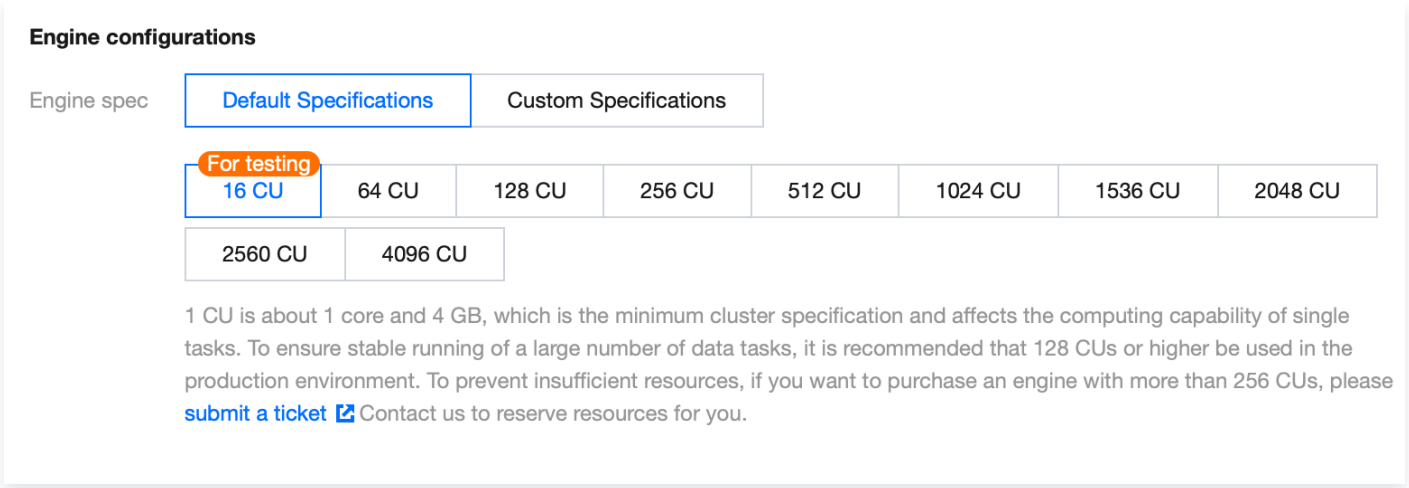
Can Data Lake Compute (DLC) Clusters Access IP Addresses/Services in Other VPCs Within the Same Region? Can They Access Public Networks?

DLC engines can access same-region VPCs by creating a network configuration in Network Connection Configuration to establish connectivity with the target VPC, then specifying this configuration in batch jobs. By default, DLC engines cannot access public networks. However, if enhanced networking is configured and the target VPC has routing rules for public network access, DLC can access public network through this setup.

Will Resizing a Cluster During Task Execution Cause Task Failure?

Below is the impact of resizing across engine types during task execution:

Engine Type	Purchase Method	Impact on Task Execution
SuperSQL-Spark job	Pay-as-you-go	No impact on tasks.
	Monthly subscription	When the cluster is resized: • Scaling out: No impact. • Scaling in: Waits for pod completion before isolating or terminating resized nodes. Long-running tasks may cause operation blocking and task restart.
SuperSQL-SparkSQL	Pay-as-you-go	Resizing the cluster will cause task restart.
	Monthly subscription	Resizing the cluster or reducing the number of clusters will cause task restart.
SuperSQL-Presto	Pay-as-you-go	Resizing the cluster will cause task restart.
	Monthly subscription	Resizing the cluster or reducing the number of clusters will cause task restart.



How to Troubleshoot Standard Engines Where Tasks Remain in Non-executing States for Extended Periods?

If your task remains in "Initializing," "Starting," or "Queued" status for an extended period, it may fail to execute normally. Follow these steps based on your task type to diagnose and resolve the issues.

1. Identifying the Task Subtype

Steps: Navigate to Historical Task Instances > Historical Task List > Task Type, Resource Group Name, and check the task type and resource group name to determine the subtype.

Task Subtype	Task Type	Resource Group Name
SQL Task	SQL	Has a value
Interactive SQL Task	SQL	Has no value (--)
Spark Batch/Streaming Task	Job	Has no value (--)

2. SQL Tasks Troubleshooting Procedure

2.1 Checking Engine and Gateway Status

1. Go to the Engine List and verify the engine status is "Running" or "Ready."
2. Check the gateway status on the management page to confirm it is "Running."
3. If the engine or gateway status is abnormal, it may indicate other processes are in progress. Please wait until the processes are complete.
4. If the status does not recover within 10 minutes, please submit a ticket to contact technical support.

2.2 Checking If Computing Resources Are in Cold Startup

1. If the task is submitted to a suspended resource group, the group may be starting (typically takes 3–5 minutes).
2. Go to the Resource Group page and confirm if the status is "Starting."
3. If startup exceeds 5 minutes, proceed with further troubleshooting.
4. If the status is neither "Running" nor "Starting," the resource group may be processing other operations. Please wait. If the problem persists for more than 10 minutes, please submit a ticket.

2.3 Verifying Resource Allocation for the Resource Group

1. If the resource group fails to start after 10 minutes, the engine may be at full capacity, preventing sufficient resources from being allocated to the group.
2. In the Resource Group List, check the resource requirements (for example, CU cores) and confirm the minimum resources required.
3. In the Engine List, click the engine name and go to "Cluster Monitoring" to review resource usage.
4. If "Occupied Cluster Capacity" reaches the total limit, insufficient remaining resources will prevent the resource group from starting.
5. Solution: Release resources by pausing or canceling other resource groups or batch/streaming job tasks.

2.4 Checking for Pending Dependent Tasks

1. By default, DLC executes multiple SQL tasks sequentially. If a preceding task is incomplete, subsequent tasks remain in "Initializing" status.
2. Confirm whether queuing is caused by unfinished dependent tasks.

2.5 Checking If the Concurrency Quantity of the Resource Group Has Been Reached

1. The default maximum concurrency quantity for a resource group is 5.
2. Tasks exceeding the concurrency quantity will remain in queued status.
3. View the concurrency quantity configuration on the Resource Group Details page, and make adjustments as needed.
4. Check the currently running task count on the Historical Task Instances page to confirm if the limit has been reached.

3. Interactive SQL Tasks (BatchSQL tasks) Troubleshooting Procedure

3.1 Checking Engine and Gateway Status

Refer to the operation steps in SQL Tasks Troubleshooting Procedure > Checking engine and gateway status.

3.2 Verifying Resource Availability

1. Check the resource usage in the "Cluster Monitoring" section of the Engine List.
2. If resources are insufficient, the task will fail to start.
3. Solution: Same as SQL tasks. Release other occupied resources.

3.3 Cold Startup Duration

- Each interactive SQL task submission requires new resource allocation, with a typical cold startup duration of 3–5 minutes.
- If startup exceeds 5 minutes, troubleshoot for potential issues.

4. Spark Batch/Streaming Jobs Troubleshooting Procedure

4.1 Checking Engine and Gateway Status

1. Refer to the operation steps in SQL Tasks Troubleshooting Procedure Checking Engine and Gateway Status.
2. Resource startup usually takes 3–5 minutes. If the job remains inactive beyond this period, proceed with further checks.

4.2 Verifying Engine Resource Availability

1. In the Historical Task Instances page, check the required resources (CU count for Driver and Executor).
2. Check the resource usage in the "Cluster Monitoring" section of the Engine List.
3. If resources are insufficient, the batch/streaming job will fail to start.
4. Solution: Same as SQL tasks. Release other occupied resources.

FAQs on Features

Last updated：2025-07-30 15:10:53

What Are the Differences Between External Tables and Native Tables?

- Native table: A table stored in DLC-managed storage (defaulting to Iceberg format). Using native tables eliminates the need to manage underlying Iceberg files and provides data optimization capabilities to help build your data lake.
- External table: A table resides in your own COS buckets or third-party storage. DLC can directly establish external tables to analyze without loading data.

Does DLC Support Data Scheduling?

If you have requirements for data scheduling, consult on and enable Tencent Cloud WeData.

Does DLC Support Self-Service Upload of JAR Packages to Define Custom Spark Functions?

Yes. It is supported.

Do Metadata-Accelerated Buckets Support CDN?

Accelerated buckets are not supported, but standard buckets are supported.

What Is the Difference Between Metadata-Accelerated Buckets and Standard COS Buckets?

The difference is that metadata-accelerated buckets deliver significant performance improvements for common file operations like RENAME and LIST in big data analysis scenarios.

Does Data Storage Require Binding Engine Permissions?

Metadata-accelerated buckets require binding engine permissions, while standard COS buckets do not.

Can a Single COS Buckets Be Bound to Multiple DLC Data Engines?

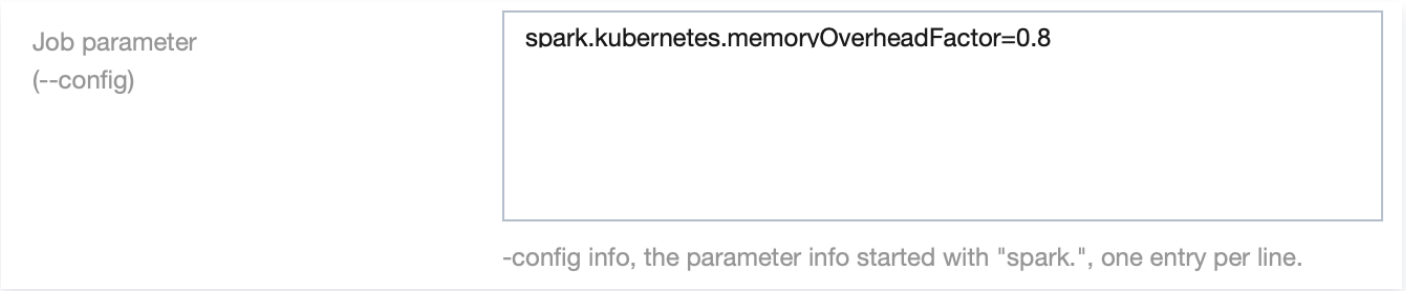
Yes.

FAQs on Spark Jobs

Last updated：2025-07-30 15:10:53

Does Data Skew in PySpark Tasks Cause OOMKilled Due to Python+JVM Memory Usage Exceeding Kubernetes Requests?

Issue description: During PySpark task execution, executor logs show "Kubernetes OOMKilled" with memory usage exceeding Kubernetes limits.
Cause analysis: The memory requested by Kubernetes is calculated based on Spark executor memory multiplied by memoryOverheadFactor. If the data processed by Python is skewed or a single data entry is too large, it may cause memory usage to exceed the memory allocated by Kubernetes.
Solution: Add a task parameter spark.kubernetes.memoryOverheadFactor=0.8 with the default value of 0.4.
Operation steps: Go to DLC console and Data Job (Spark Job) > Edit Jobs, then configure as follows:



How to Automatically Add a REPARTITION command After INSERT INTO/OVERWRITE for Data Partition to Reduce the Number of Small Files?

Solution: Enable auto repartitioning and configure the following parameters:

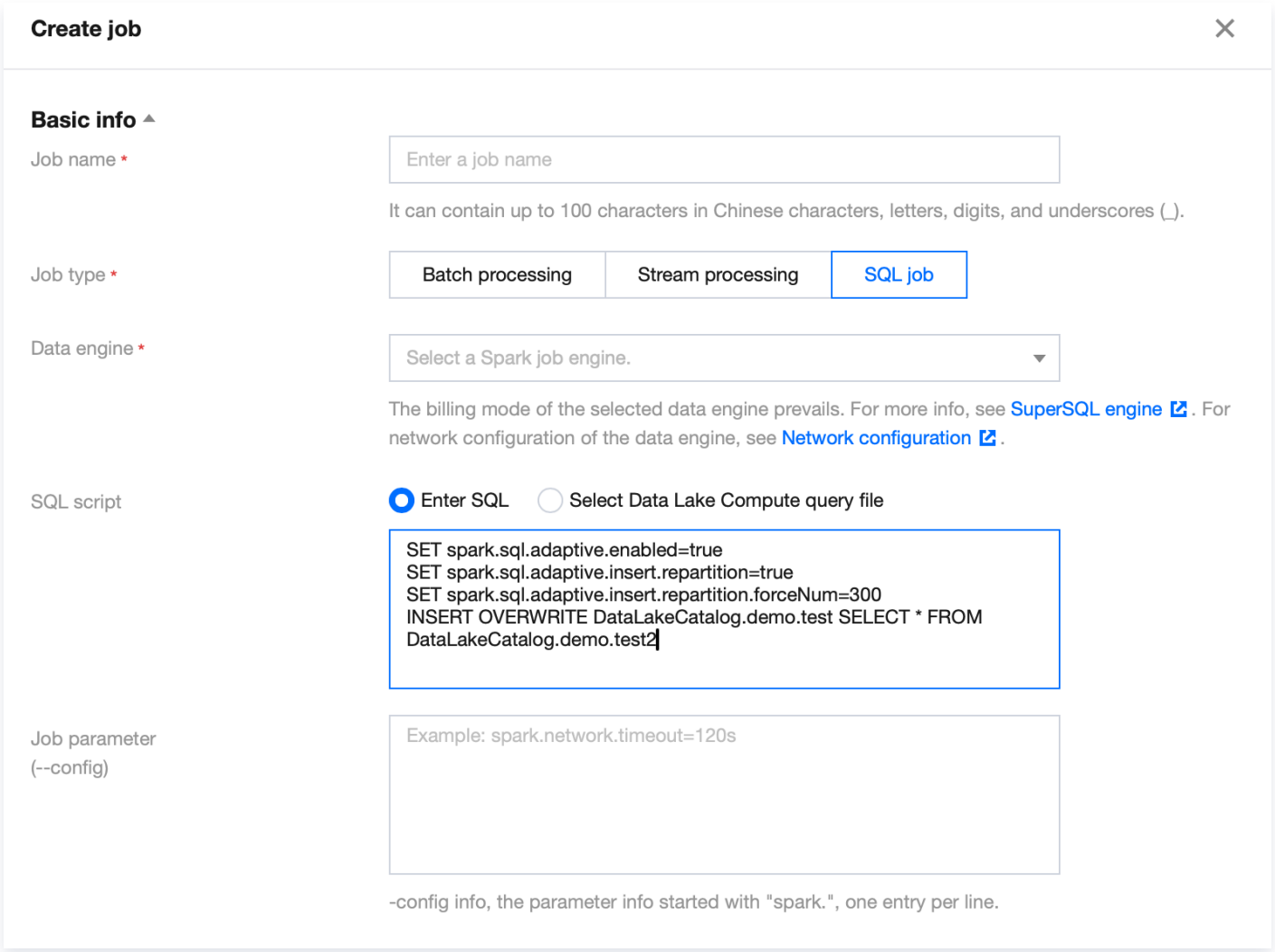
```
spark.sql.adaptive.enabled:true
spark.sql.adaptive.insert.repartition:true
spark.sql.adaptive.insert.repartition.forceNum: 300 (specifies partition
count)
```

Directions:

- Configure to SparkConf in the program:

```
val spark: SparkSession = args(0) match {
  case "1" =>
    SparkSession.builder().getOrCreate()
  case "2" =>
    SparkSession.builder().config("spark.sql.adaptive.insert.repartition", "true").getOrCreate()
  case "3" =>
    SparkSession.builder().config("spark.sql.adaptive.insert.repartition", "true")
      .config("spark.sql.adaptive.coalescePartitions.enabled", "true")
      .config("spark.sql.adaptive.coalescePartitions.initialPartitionNum", "300")
      .getOrCreate()
  case "4" =>
    SparkSession.builder().config("spark.sql.adaptive.insert.repartition.forceNum", "300").getOrCreate()
  case _ =>
    SparkSession.builder().getOrCreate()
}
```

- Set SQL SET in the program:



Why Do PySpark Tasks Return 503 Errors During High-Concurrency Writes to COS Buckets?

Issue description: During high-concurrency writes to COS buckets by PySpark tasks, executors report frequent 503 errors returned by COS.
Cause: The parallelism for Spark tasks writing to COS is determined by fs.cosn.trsf.fs.ofs.data.transfer.thread.count. For example, without tuning on 4096 cores, the default concurrency is 4096×32=131,072, creating COS bottlenecks.
Solution:

1. Create a metadata-accelerated bucket in COS to prevent rate limiting from frequent list and rename operations during Spark task writes.
2. Adjust bandwidth limits for metadata-accelerated buckets in COS.
3. Add the following parameters to tasks to reduce excessive access pressure on COS during high parallelism.

```
fs.cosn.trsf.fs.ofs.data.transfer.thread.count=8
fs.cosn.trsf.fs.ofs.block.max.file.cache.mb=0
spark.hadoop.fs.cosn.trsf.fs.ofs.data.transfer.thread.count=8
spark.hadoop.fs.cosn.trsf.fs.ofs.block.max.file.cache.mb=0
```

What Are Common Data Governance SQL Commands?

- Disable database governance SQL.

```
ALTER DATABASE DataLakeCatalog.demo_db
SET
  DBPROPERTIES (
    'dlc.ao.data.govern.inherit' = 'none',
    'dlc.ao.merge.data.enable' = 'disable',
    'dlc.ao.expired.snapshots.enable' = 'disable',
    'dlc.ao.remove.orphan.enable' = 'disable',
    'dlc.ao.merge.manifests.enable' = 'disable'
  )
```

- Enable database governance SQL.

```
ALTER DATABASE DataLakeCatalog.db_name
SET
  DBPROPERTIES (
    'dlc.ao.data.govern.inherit' = 'none',
    'dlc.ao.merge.data.enable' = 'enable',
    'dlc.ao.merge.data.engine' = 'bda-sinker',
```

```
    'dlc.ao.merge.data.min-input-files' = '10',
    'dlc.ao.merge.data.target-file-size-bytes' = '536870912',
    'dlc.ao.merge.data.interval-min' = '90',
    'dlc.ao.expired.snapshots.enable' = 'enable',
    'dlc.ao.expired.snapshots.engine' = 'bda-sinker',
    'dlc.ao.expired.snapshots.retain-last' = '5',
    'dlc.ao.expired.snapshots.before-days' = '2',
    'dlc.ao.expired.snapshots.max-concurrent-deletes' = '4',
    'dlc.ao.expired.snapshots.interval-min' = '150',
    'dlc.ao.remove.orphan.enable' = 'enable',
    'dlc.ao.remove.orphan.engine' = 'bda-sinker',
    'dlc.ao.remove.orphan.before-days' = '3',
    'dlc.ao.remove.orphan.max-concurrent-deletes' = '4',
    'dlc.ao.remove.orphan.interval-min' = '600',
    'dlc.ao.merge.manifests.enable' = 'enable',
    'dlc.ao.merge.manifests.engine' = 'bda-sinker',
    'dlc.ao.merge.manifests.interval-min' = '1440'
  )
```

- Disable table governance SQL.

```
  ALTER TABLE
  `DataLakeCatalog`.`db_name`.`tb_name`
SET
  TBLPROPERTIES (
    'dlc.ao.data.govern.inherit' = 'none',
    'dlc.ao.merge.data.enable' = 'disable',
    'dlc.ao.expired.snapshots.enable' = 'disable',
    'dlc.ao.remove.orphan.enable' = 'disable',
    'dlc.ao.merge.manifests.enable' = 'disable'
  )
```

- Enable inherited database governance SQL..

```
  ALTER TABLE `DataLakeCatalog`.`db_name`.`tb_name`
  SET TBLPROPERTIES ('dlc.ao.data.govern.inherit' = 'default')
```

- Enable table governance SQL.

```
  ALTER TABLE
  `DataLakeCatalog`.`db_name`.`tb_name`
SET
  TBLPROPERTIES (
    'dlc.ao.data.govern.inherit' = 'none',
    'dlc.ao.merge.data.enable' = 'enable',
    'dlc.ao.merge.data.engine' = 'bda-sinker',
    'dlc.ao.merge.data.min-input-files' = '10',
    'dlc.ao.merge.data.target-file-size-bytes' = '536870912',
    'dlc.ao.merge.data.interval-min' = '90',
    'dlc.ao.expired.snapshots.enable' = 'enable',
    'dlc.ao.expired.snapshots.engine' = 'bda-sinker',
    'dlc.ao.expired.snapshots.retain-last' = '5',
    'dlc.ao.expired.snapshots.before-days' = '2',
    'dlc.ao.expired.snapshots.max-concurrent-deletes' = '4',
    'dlc.ao.expired.snapshots.interval-min' = '150',
    'dlc.ao.remove.orphan.enable' = 'enable',
    'dlc.ao.remove.orphan.engine' = 'bda-sinker',
    'dlc.ao.remove.orphan.before-days' = '3',
    'dlc.ao.remove.orphan.max-concurrent-deletes' = '4',
    'dlc.ao.remove.orphan.interval-min' = '600',
    'dlc.ao.merge.manifests.enable' = 'enable',
    'dlc.ao.merge.manifests.engine' = 'bda-sinker',
    'dlc.ao.merge.manifests.interval-min' = '1440'
  )
```

- Full-table merge SQL (no WHERE clause).

```
  CALL `DataLakeCatalog`.`system`.`rewrite_data_files`(
  `table` => 'tb_name',
  `options` => map(
    'min-input-files',
    '10',
    'target-file-size-bytes',
    '536870912',
    'delete-file-threshold',
    '1',
    'max-concurrent-file-group-rewrites',
    '20'
```

```
  )
)
```

- Incremental merge SQL (with WHERE clause).

```
 CALL `DataLakeCatalog`.`system`.`rewrite_data_files`(
  `table` => 'tb_name',
  `options` => map(
    'min-input-files',
    '10',
    'target-file-size-bytes',
    '536870912',
    'delete-file-threshold',
    '1',
    'max-concurrent-file-group-rewrites',
    '20'
  ),
  `where` => 'field_date > "2022-01-01" and field_date <= "2023-01-
01"'
)
```

- Snapshot expiration SQL.

```
  CALL `DataLakeCatalog`.`system`.`expire_snapshots`(
  `table` => 'tb_name',
  older_than => TIMESTAMP '2023-02-28 16:06:35.000',
  retain_last => 1,
  max_concurrent_deletes => 4,
  stream_results => true
)
```

How to View SQL Execution Plans and Logs?

View SQL execution plans: Use the EXPLAIN keyword in Data Explore to examine physical execution plans. For detailed EXPLAIN usage, refer to EXPLAIN.

View SQL execution logs:

1. SQL execution logs display in the results when running SQL in Data Explore.
2. View SQL execution logs in DLC console > Data Ops > Execution History.

Does Data Write Failure Occur Due to CAST Operations Not Automatically Converting Precision?

Issue description: When migrating Hive SQL to Spark SQL, an error occurred: Cannot safely cast 'class_type': string to bigint.
Cause positioning: Starting from Spark 3.0.0, Spark SQL enforces 3 security policies when performing type conversions:

- ANSI: Prohibits Spark from performing certain unreasonable type conversions, for example, 'string' to 'timestamp'.
- LEGACY: Allows Spark to force type conversions as long as it's an effective CAST operation.
- STRICT: Prohibits Spark from performing any conversion that may damage precision. The default policy is ANSI.

Solution: Switch to the LEGACY policy by setting spark.sql.storeAssignmentPolicy=LEGACY.

An error of 'QUERY_PROGRESS_UPDATE_ERROR(code=3060): Failed to update statement progress'

Issue description: When submitting Spark SQL tasks in Data Explore, the system reports a "Failed to update statement progress" error during execution.
Cause positioning: When multiple Spark SQL tasks are submitted, continuous asynchronous tracking is required for each SQL execution progress. The async processing queue here has a capacity limit, with a default value of 100 (updated to 300 for versions after January 14, 2024). Therefore, if a submitted task remains incomplete while new tasks exceed the queue limit, this error occurs. Such errors typically indicate the SQL task might be a long-tail task. You should evaluate its resource impact on concurrent operations.
Solution: You can adjust the parameter livy.rsc.retained-statements to a value larger than its default in engine configurations. Note that the engine will restart after adjustments. The specific value can be set based on task concurrency. This parameter has minimal impact on the cluster. When concurrent SQL submissions reach 100-200/min, setting this parameter to 6000 was validated through production testing with negligible impact.