Exploring execution times

This topic explains how to examine the past performance of queries and tasks. This information helps identify candidates for performance optimizations and allows you to see whether your optimization strategies are having the desired effect.

You can explore historical performance using Snowsight or by writing queries against views in the ACCOUNT_USAGE schema. A user without access to the ACCOUNT_USAGE schema can query similar data using the Information Schema.

View execution times and load

You can use Snowsight to gain visual insights into the performance of queries and tasks as well as the load of a warehouse.

Queries

1. Sign in to Snowsight.
2. Select **Monitoring** » **Query History**.
3. Use the **Duration** column to understand how long it took a query to execute. You can sort the column to find the queries that ran the longest.
4. If you want to focus on a particular user's queries, use the **User** drop-down to select the user.
5. If you want to focus on the queries that ran on a particular warehouse, select **Filters** » **Warehouse**, and then select the warehouse.

Warehouses

1. Sign in to Snowsight.
2. Switch to a role that has privileges for the warehouse.
3. Select **Admin** » **Warehouses**.
4. Select a warehouse.
5. Use the **Warehouse Activity** chart to visualize the load of the warehouse, including whether queries were queued.

Tasks

1. Sign in to Snowsight.
2. Select **Monitoring** » **Task History** to view how long it took to execute a task's SQL code.

Drill down into execution times

The Query Profile allows you to examine which parts of a query are taking the longest to execute. It includes a **Most Expensive Nodes** pane that identifies the operator nodes that are

taking the longest to execute. You can drill down even further by viewing what percentage of a node's execution time was spent in a particular category of query processing.

To access the Query Profile for a query:

1. Sign in to Snowsight.
2. Select **Monitoring** » **Query History**.
3. Select the query ID of a query.
4. Select the **Query Profile** tab.

Tip
You can programmatically access the performance statistics of the Query Profile by executing the GET_QUERY_OPERATOR_STATS function.

Write queries to explore execution times

The Account Usage schema contains views related to the execution times of queries and tasks. It also contains a view related to the load of a warehouse as it executes queries. You can write queries against these views to drill down into performance data and create custom reports and dashboards.

By default, only the account administrator (i.e. user with the ACCOUNTADMIN role) can access views in the ACCOUNT_USAGE schema. To allow other users to access these views, refer to Enabling the SNOWFLAKE Database Usage for Other Roles.

Users without access to the ACCOUNT_USAGE schema (e.g. a user who ran a query or a warehouse administrator) can still return recent execution times and other query metadata using the QUERY_HISTORY table functions of the Information Schema.

Be aware that the ACCOUNT_USAGE views are not updated immediately after running a query or task. If you want to check the execution time of a query right after running it, use Snowsight to view its performance. The Information Schema is also updated quicker than the ACCOUNT_USAGE views.

ACCOUNT_USAGE View	Description	Latency

QUERY_HISTORY	Used to analyze the Snowflake query history by various dimensions (time range, execution time, session, user, warehouse, etc.) within the last 365 days (1 year).	Up to 45 minutes
WAREHOUSE_LOAD_HISTORY	Used to analyze the workload on a warehouse within a specified date range.	Up to 3 hours
TASK_HISTORY	Used to retrieve the history of task usage within the last 365 days (1 year).	Up to 45 minutes

Example queries

The following queries against the ACCOUNT_USAGE schema provide insight into the past performance of queries, warehouses, and tasks. Click the name of a query to see the full SQL example.

Query Performance

- Query: Top n longest-running queries
- Query: Queries organized by execution time over past month
- Query: Find long running repeated queries
- Query: Track the average performance of a query over time

Warehouse Load

- Query: Total warehouse load

Task Performance

- Query: Longest running tasks

Query performance

Query: Top n longest-running queries

This query provides a listing of the top n (50 in the example below) longest-running queries in the last day. You can adjust the `DATEADD` function to focus on a shorter or longer period of time. Replace `my_warehouse` with the name of a warehouse.

```
SELECT query_id,
  ROW_NUMBER() OVER(ORDER BY partitions_scanned DESC) AS
query_id_int,
  query_text,
  total_elapsed_time/1000 AS query_execution_time_seconds,
  partitions_scanned,
  partitions_total,
FROM snowflake.account_usage.query_history Q
WHERE warehouse_name = 'my_warehouse' AND TO_DATE(Q.start_time) >
DATEADD(day,-1,TO_DATE(CURRENT_TIMESTAMP()))
  AND total_elapsed_time > 0 --only get queries that actually used
compute
  AND error_code IS NULL
  AND partitions_scanned IS NOT NULL
ORDER BY total_elapsed_time desc
LIMIT 50;
```

Query: Queries organized by execution time over past month

This query groups queries for a given warehouse by buckets for execution time over the last month. These trends in query completion time can help inform decisions to resize warehouses or separate out some queries to another warehouse. Replace `MY_WAREHOUSE` with the name of a warehouse.

```
SELECT
  CASE
    WHEN Q.total_elapsed_time <= 60000 THEN 'Less than 60 seconds'
    WHEN Q.total_elapsed_time <= 300000 THEN '60 seconds to 5
minutes'
    WHEN Q.total_elapsed_time <= 1800000 THEN '5 minutes to 30
minutes'
    ELSE 'more than 30 minutes'
  END AS BUCKETS,
  COUNT(query_id) AS number_of_queries
FROM snowflake.account_usage.query_history Q
WHERE  TO_DATE(Q.START_TIME) >
DATEADD(month,-1,TO_DATE(CURRENT_TIMESTAMP()))
  AND total_elapsed_time > 0
  AND warehouse_name = 'my_warehouse'
GROUP BY 1;
```

Query: Find long running repeated queries

You can use the query hash (the value of the `query_hash` column in the ACCOUNT_USAGE QUERY_HISTORY view) to find patterns in query performance that might not be obvious. For example, although a query might not be excessively expensive during any single execution, a frequently repeated query could lead to high costs, based on the number of times the query runs.

You can use the query hash to identify the queries that you should focus on optimizing first. For example, the following query uses the value in the `query_hash` column to identify the query IDs for the 100 longest-running queries:

```
SELECT
    query_hash,
    COUNT(*),
    SUM(total_elapsed_time),
    ANY_VALUE(query_id)
  FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
  WHERE warehouse_name = 'MY_WAREHOUSE'
    AND DATE_TRUNC('day', start_time) >= CURRENT_DATE() - 7
  GROUP BY query_hash
  ORDER BY SUM(total_elapsed_time) DESC
  LIMIT 100;
```

Query: Track the average performance of a query over time

The following statement computes the daily average total elapsed time for all queries that have a specific parameterized query hash (`cbd58379a88c37ed6cc0ecfebb053b03`).

```
SELECT
    DATE_TRUNC('day', start_time),
    SUM(total_elapsed_time),
    ANY_VALUE(query_id)
  FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
  WHERE query_parameterized_hash = 'cbd58379a88c37ed6cc0ecfebb053b03'
    AND DATE_TRUNC('day', start_time) >= CURRENT_DATE() - 30
  GROUP BY DATE_TRUNC('day', start_time);
```

Warehouse load

Query: Total warehouse load

This query provides insight into the total load of a warehouse for executed and queued queries. These load values represent the ratio of the total execution time (in seconds) of all queries in a specific state in an interval by the total time (in seconds) for that interval.

For example, if 276 seconds was the total time for 4 queries in a 5 minute (300 second) interval, then the query load value is 276 / 300 = 0.92.

```
SELECT TO_DATE(start_time) AS date,
  warehouse_name,
  SUM(avg_running) AS sum_running,
  SUM(avg_queued_load) AS sum_queued
FROM snowflake.account_usage.warehouse_load_history
WHERE TO_DATE(start_time) >= DATEADD(month,-1,CURRENT_TIMESTAMP())
GROUP BY 1,2
HAVING SUM(avg_queued_load) >0;
```

Task performance

Query: Longest running tasks

This query lists the longest running tasks in the last day, which can indicate an opportunity to optimize the SQL being executed by the task.

```
SELECT DATEDIFF(seconds, query_start_time,completed_time) AS
duration_seconds,*
FROM snowflake.account_usage.task_history
WHERE state = 'SUCCEEDED'
  AND query_start_time >= DATEADD (day, -1, CURRENT_TIMESTAMP())
ORDER BY duration_seconds DESC;
```

Was this page helpful?