Query History

SITUATION: You manage weekly reports for your sales department, and one day, a report fails to run. Instead of troubleshooting in the dark, you check Query History and immediately see that the SQL query failed due to a typo. After correcting the error, you quickly rerun the query, and the report generates without issue. Query History not only saved you time but also pinpointed the exact problem.

Guiding Questions:

1. How does Query History help you identify and resolve slow-running queries?
2. In what ways can Query History help your organization optimize costs in Snowflake?

In Snowflake, Query History(opens in a new tab) acts as the system's memory, logging every query executed in your account. From investigating slow-running queries, monitoring resource usage, or reviewing user activity for compliance, Query History serves as your comprehensive record.

The Query History secure view acts as the system's log, containing every query executed in your account for the last year. For quickly finding recent queries, the Query History area in Snowsight displays queries that have run in the previous 14 days.

Key Concepts to Know:

1. Query Text - This stores the actual SQL code that was run.
2. Execution Status - It indicates whether the query succeeded, failed, or was canceled.
3. User Information - This shows which user executed the query.
4. Timing - It tracks when the query started and ended, and how long it took to complete.
5. Resource Usage - This displays the virtual warehouses and data scanned during query execution.
6. Micro-partitions - It tracks the total number of micro-partitions and the number of them scanned.
7. Percentage scanned from Cache - This displays the percentage of the query's data that came from the warehouse cache.
8. Spillage - It displays whether local and/or remote spillage occurred. Spillage(opens in a new tab) occurs when queries are too large to fit in memory.

Real-World Example

The IT department head might prefer Snowsight to monitor day-to-day activity quickly, viewing recent queries and their statuses at a glance. Meanwhile, a data engineer might use SQL commands in QUERY_HISTORY to dig deeper into long-running queries or analyze trends in query performance over time.

EXAMPLE SQL Query:
For example, to find all failed queries in the last 24 hours:

```
SELECT QUERY_ID, QUERY_TEXT, EXECUTION_STATUS, START_TIME
FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
WHERE EXECUTION_STATUS = 'FAILED'
AND START_TIME > DATEADD(HOUR, -24, CURRENT_TIMESTAMP());
```

Snowflake's Query History offers powerful features that allow you to filter, sort, and extend the time window of your query logs. These features become particularly useful for longer-term audits or troubleshooting specific problems.
Key Insights:

- **Filtering**: Filters allow you to target specific queries, such as by user, time, or status. For instance, you can quickly retrieve all queries that failed over a specific period.
- **Extended Time Windows**: By default, Query History only holds the last 14 days of logs, but the **ACCOUNT_USAGE** views allow you to access up to 365 days of data, making it useful for compliance reviews or historical performance analysis.

Why does performance monitoring matter?
By monitoring key performance metrics like query execution time and resource usage, you can pinpoint inefficiencies and optimize performance. Query History helps you identify bottlenecks or unnecessary resource consumption.
Key Insights:
- **Execution Time**: This metric helps you identify slow-running queries that could be optimized.
- **Bytes Scanned**: Understanding how much data each query processes allows you to recognize resource-heavy queries.
- **Partitions Scanned, Partitions Total, Rows Processed**: Analyze whether a query is scanning large amounts of unnecessary data, which can waste resources. To identify whether pruning
- (opens in a new tab)
- occurred, divide the partitions scanned by the partitions total.

Real-World Example
A marketing manager notices a regular campaign query is running slower than usual. Using Query History, the team finds that the query now scans more data than before, indicating the need to optimize the query or adjust the warehouse size for better performance.

Example Query for Performance Monitoring:
```
SELECT QUERY_ID, QUERY_TEXT, TOTAL_ELAPSED_TIME, BYTES_SCANNED
FROM QUERY_HISTORY
WHERE TOTAL_ELAPSED_TIME > 5000; -- Queries taking more than 5 seconds
```

Monitor query activity with Query History

To monitor query activity in your account, you can use:
- The **Query History** and **Grouped Query History** pages in Snowsight.
- The QUERY_HISTORY view and AGGREGATE_QUERY_HISTORY view in the ACCOUNT_USAGE schema of the SNOWFLAKE database.
- The QUERY_HISTORY family of table functions in INFORMATION_SCHEMA.

With the **Query History** page in Snowsight, you can do the following:
- Monitor individual or grouped queries that are executed by users in your account.
- View details about queries, including performance data. In some cases, query details are unavailable.
- Explore each step of an executed query in the query profile.

The Query History page lets you explore queries executed in your Snowflake account over the last 14 days. Within a worksheet, you can see the query history for queries that have been run in that worksheet. See View query history.

Review Query History in Snowsight

To access the **Query History** page in Snowsight, do the following:

1. Sign in to Snowsight.
2. Select **Monitoring** » **Query History**.
3. Go to **Individual Queries** or **Grouped Queries**. For more information about grouped queries, see Use the Grouped Query History view in Snowsight.
4. For **Individual Queries**, filter your view to see the most relevant and accurate results.
   If a **Load More** button appears at the top of the list, it means that there are more available results to load. You can fetch the next set of results by either selecting **Load More** or scrolling to the bottom of the list.

Privileges required to view Query History

You can always view history for queries that you have run.
To view history for other queries, your active role affects what else you can see in **Query History**:

- If your active role is the ACCOUNTADMIN role, you can view all query history for the account.
- If your active role has the MONITOR or OPERATE privilege granted on a warehouse, you can view queries run by other users that use that warehouse.
- If your active role is granted the GOVERNANCE_VIEWER database role for the SNOWFLAKE database, you can view the query history for all users in your account. See SNOWFLAKE database roles.
- If your active role is granted the READER_USAGE_VIEWER database role for the SNOWFLAKE database, you can view the query history for all users in reader accounts associated with your account. See SNOWFLAKE database roles.

Considerations for using Query History

When reviewing the **Query History** for your account, consider the following:

- Details for queries executed more than seven days ago do not include **User** information due to the data retention policy for sessions. You can use the user filter to retrieve queries run by individual users. See Filter Query History.
- For queries that failed due to syntax or parsing errors, you see `<redacted>` instead of the SQL statement that was executed. If you are granted a role with appropriate privileges, you can set the ENABLE_UNREDACTED_QUERY_SYNTAX_ERROR parameter to view the full query text.
- Filters and the **Started** and **End Time** columns use your current time zone. You can't change this setting. Setting the TIMEZONE parameter for the session doesn't change the time zone used.

Filter Query History

You can filter the Query History list by the following:

- Status of the query, for example to identify long-running queries, failed queries, and queued queries.
- User who performed the query, including:
  - **All**, to see all users for which you have access to view query history.
  - The user you are signed in as (default)
  - Individual Snowflake users in your account, if your role can view query history for other users.
- Time period during which the query was run, up to 14 days.
- Other filters, including the following:
  - **SQL Text**, for example, to view queries that use specific statements, such as GROUP BY.
  - **Query ID**, to view details for a specific query.
  - **Warehouse**, to view queries that were run using a specific warehouse.
  - **Statement Type**, to view queries that used a specific type of statement, such as DELETE, UPDATE, INSERT, or SELECT.

- **Duration**, for example, to identify especially long-running queries.
- **Session ID**, to view queries run during a specific Snowflake session.
- **Query Tag**, to view queries with a specific query tag set through the QUERY_TAG session parameter.
- **Parameterized Query Hash**, to display queries grouped according to the parameterized query hash ID specified in the filter. For more information, see Using the Hash of the Parameterized Query (query_parameterized_hash).
- **Client generated statements**, to view internal queries run by a client, driver, or library, including the web interface. For example, whenever a user navigates to the **Warehouses** page in Snowsight, Snowflake executes a SHOW WAREHOUSES statement in the background. That statement would be visible when this filter is enabled. Your account is not billed for client-generated statements.
- **Queries executed by user tasks**, to view SQL statements executed or stored procedures called by user tasks.
- **Show replication refresh history**, to view queries used to perform replication refresh tasks to remote regions and accounts.

If you want to see near-real-time results, enable **Auto Refresh**. When **Auto Refresh** is enabled, the table refreshes every ten seconds.
You can see the following columns in the **Queries** table by default:

- **SQL Text**, the text of the executed statement (always shown).
- **Query ID**, the ID of the query (always shown).
- **Status**, the status of the executed statement (always shown).
- **User**, to see the username that executed a statement.
- **Warehouse**, to see the warehouse used to execute a statement.
- **Duration**, to see the length of time it took to execute a statement.
- **Started**, to see the time a statement started running.

If you have more results, you cannot sort the table. If you select **Load More** at the top of the list after sorting the table, the new results will be appended to the end of the data and the sort order will no longer apply.
To view more specific information, you can select **Columns** to add or remove columns from the table, such as:

- **All** to display all columns.
- **User** to display the user who ran the statement.
- **Warehouse** to display the name of the warehouse used to run the statement.
- **Warehouse Size** to display the size of the warehouse used to run the statement.
- **Duration** to display the time it took for the statement to run.
- **Started** to display the start time of the statement.
- **End Time** to display the end time of the statement.
- **Session ID** to display the ID of the session that executed the statement.
- **Client Driver** to display the name and version of the client, driver, or library used to execute the statement. Statements run in Snowsight display `Go 1.1.5`.
- **Bytes Scanned** to display the number of bytes scanned during the processing of the query.
- **Rows** to display the number of rows returned by a statement.
- **Query Tag** to display the query tag set for a query.
- **Parameterized Query Hash** to display queries grouped according to the parameterized query hash ID specified in the filter. For more information, see Using the Hash of the Parameterized Query (query_parameterized_hash).
- **Incident** to display details for statements with an execution status of incident, used for troubleshooting or debugging purposes.

What is Caching in Snowflake?

❓ Guiding Questions:

1. What is the main benefit of caching in Snowflake?
2. How does auto-suspension affect warehouse cache?
3. Why is query profiling useful when managing cache performance?

Think of caching in Snowflake, like saving your favorite shows on a streaming platform. Once you've watched an episode, it's easier to rewatch it since it's already loaded. Similarly, caching temporarily stores query results so that if you ask for the same information again, Snowflake retrieves it faster instead of starting from scratch. This saves both time and resources and when Snowflake retrieves the stored data, you get your answer quicker.

Caching helps to hold information from previous queries. Whether it's storing the data used in solving queries, results sets or metadata information, all these can potentially be used to speed up later queries.

How Does Caching Help?

Faster Performance: Queries that can leverage either the Query Results Cache or Warehouse Cache will process faster than if the queries had to read data from micro-partitions in the storage layer.

Cost Savings: Faster performance means less compute resources for queries in later runs, saving credits.

Key Concepts in Caching

- **Auto-Suspension for warehouses:** Auto-suspension allows Snowflake warehouses to save costs by shutting down during periods of inactivity. However, when a warehouse is suspended, its warehouse cache is cleared, which could slow down subsequent queries. Auto suspension is a parameter that you can set depending on workloads and warehouse usage in your business.

Real-World Consideration: While useful for cost savings - an auto-suspend setting that is too short could prevent other queries from making use of the warehouse cache.

- **Query Profiling:**

Query profiling helps you understand if your queries benefit from cache usage. You can analyze whether Snowflake pulled the data from the query result cache, warehouse cache, or slower cloud storage.

Types of Caches in Snowflake

When data is not found in the query result or warehouse cache, Snowflake retrieves it from its cloud storage. Efficient caching strategies can reduce the need to scan micro-partitions(opens in a new tab) in the storage layer.(opens in a new tab)

Query Result Cache:

Real-World Example: A financial analyst running a stock performance report several times a day will benefit from result caching. Instead of recalculating the data every time, Snowflake fetches the preprocessed results, speeding up the workflow significantly, as long as the underlying data hasn't changed.

This stores the results of queries for up to 24 hours. If the same query is run again without underlying data changes, Snowflake uses the cached result instead of reprocessing the query. The counter is set back to zero each time the same query is run, thus, beginning a new 24-hour period. Once a query result cache is older than 31 days, it is purged from the system.

Pitfall to Watch: In order for the query result cache to be used, the underlying queries must be identical using the same case of letters. For example, the following queries will not share the query result cache due to the differences in the capitalization of the letters. They must be syntactically identical.

SELECT * FROM CUSTOMER;

select * from customer;

Select * from customer;
All three of these queries are NOT equivalent from the perspective of the query result cache.

Warehouse Cache:

Real-World Example: A data science team working with customer transaction data throughout the day will experience faster performance as the warehouse cache keeps its data readily available.

This local cache stores table data in the warehouse's SSD storage. Queries running on an active warehouse can benefit from cached data if they request similar datasets multiple times.

Important Consideration: The warehouse cache is purged if the warehouse is suspended, which means you may lose this benefit if the warehouse shuts down too quickly.

Caching Best Practices

To leverage caching effectively in Snowflake:

- **Use Consistent Queries**: Minor changes in query structure, even in formatting, can prevent query result cache reuse.
- **Optimize Auto-Suspension**: For environments that run frequent queries, such as analytics dashboards, setting longer warehouse auto-suspension intervals ensures the warehouse cache remains available.
- **Monitor and Analyze**: Use query profiling and history to track cache usage. Identify patterns where caching can be enhanced or where costs can be minimized through auto-suspension settings.

1. How can User A and User B access one another's result sets from the Query Result Cache?
   A. They use the same Warehouse.
   B. They run the exact same SQL query formatted identically.
   C. They sign in to the same session.
   D. They run within 59 minutes of one another.

2. If you view Snowsight's Query History area, how far back in time can you review queries?
   A. 365 days
   B. A calendar month
   C. 14 days
   D. 7 days
   E. 24 hours

3. A query you initiated is taking too long. You've gone into Snowsight's Query History area to check whether this query (which usually runs every hour) is supposed to take a long time. Select the four statements that are true.
   A. Information in the Query History area can be filtered to show a single Warehouse.
   B. Information in the Query History area can be filtered to show a single User.
   C. Information in the Query History area can be filtered to show a single Session.
   D. Information in the Query History History area can help you visually compare query times.
   E. If you decide to end the query, you must return to the worksheet to abort the query.

4. You have a dashboard that connects to Snowflake via JDBC. The dashboard is refreshed hundreds of times per day. The data is very stable, only changing once or twice per day. The query run by the dashboard connector never changes. How will Snowflake manage changing and non-changing data? Select the three true statements.
   A. Snowflake will show the most up-to-date data each time the dashboard is refreshed.
   B. Snowflake will spin up a warehouse each time the dashboard is refreshed.
   C. Snowflake will spin up a warehouse if the underlying data has changed.
   D. Snowflake will compile Query Result Cache data from all user results so no warehouse is needed.
   E. Snowflake will re-use data from the Query Result Cache as long as it is still the most up-to-date data available.

5. Which cache type runs on a 24 hour "clock"?
   A. Metadata
   B. Query Result Cache
   C. Warehouse Cache

6. What action causes a warehouse's warehouse cache to be purged?
   A. Resuming the warehouse.
   B. Suspending the warehouse.
   C. The passing of 24 hours.
   D. A change to the Metadata size.

7. What two types of cache are mentioned in the discussion? Select two.
   A. Storage Cache
   B. Warehouse Cache
   C. Query Result Cache
   D. History Cache

ANSWER KEYS

1. B
2. C
3. ABCD
4. ACE
5. B
6. B

HAPPY LEARNING
REGARDS
SARANSH JAIN