# **QUERY ANALYSER**

## **Objective**

Analyzing the long running queries in all aspects to identify pain points & scope for improving the query performance & runtime of the queries.

## **High Level Proposal**

1. Capture the required performance KPIs of long running queries from QUERY\_HISTORY table
2. Get the Top 3 most time consuming steps from GET\_QUERY\_OPERATOR\_STATS function
3. Apply different types of filters on each KPI as per the requirements in order to fetch the long running queries, below are the KPIs where we applied filters

i) TOTAL\_ELAPSED\_TIME

ii) QUERY\_LOAD\_PERCENT

iii) PARTITIONS\_SCANNED

iv) BYTES\_SPILLED\_TO\_REMOTE\_STORAGE

v) TRANSACTION\_BLOCKED\_TIME

vi) QUEUED\_OVERLOAD\_TIME

vii) ERROR\_MESSAGE

1. Based on each filter / condition, the recommendations will be given, by which the queries can be modified and improve the query load.

## **How to use / run the tool**

-> The code is written in the form of Stored Procedure using Python script, where we have used data frames in order to view the data and to apply filters.

## **Technical Implementation**

-> Compares the data and filter out the long running queries from the Query history table, and suggest some recommendations in order to improve the query load.

## **Prerequisites**

1. Fetching the required columns from QUERY\_HISTORY table for one query id.
2. Capturing data(top running queries) from get\_query\_operator\_stats.
3. Creating a temporary table and inserting the values for QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION.
4. Applying filters on the below columns, to attain the certain results

i) TOTAL\_ELAPSED\_TIME

ii) QUERY\_LOAD\_PERCENT

iii) PARTITIONS\_SCANNED

iv) BYTES\_SPILLED\_TO\_REMOTE\_STORAGE

v) TRANSACTION\_BLOCKED\_TIME

vi) QUEUED\_OVERLOAD\_TIME

vii) ERROR\_MESSAGE

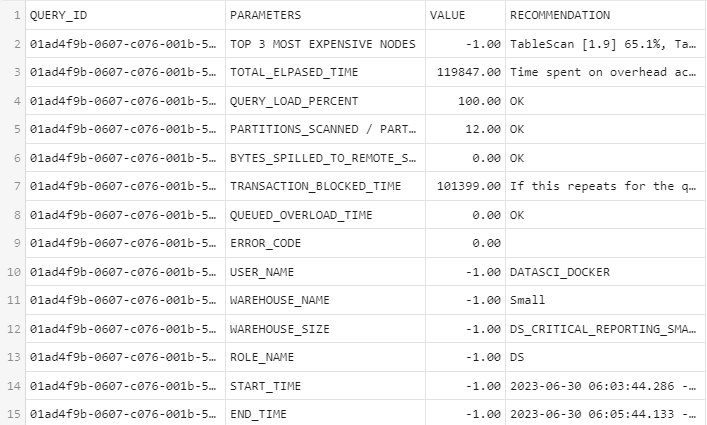
1. Creating an other procedure for QUERY\_ANALYSER\_REPORT on top of QUERY\_ANALYZER procedure.
2. Initially Capturing the LAST\_QUERY\_START\_TIME and INSERT\_TIME into new table CONTROL\_TABLE.
3. Fetching query\_id and start\_time which are older than 14 days from BAD\_QUERY table of long running queries.
4. Calling QUERY\_ANALYZER procedure for all the query ids and execution has to be happened continuously / loop (immediately right after the one statement).
5. Creating new table and capturing the values of QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION values into QUERY\_PERFORMANCE\_KPI.
6. Updating LAST\_TIMESTAMP as START\_TIME which will be helpful in Step 6.

## **Source of data**

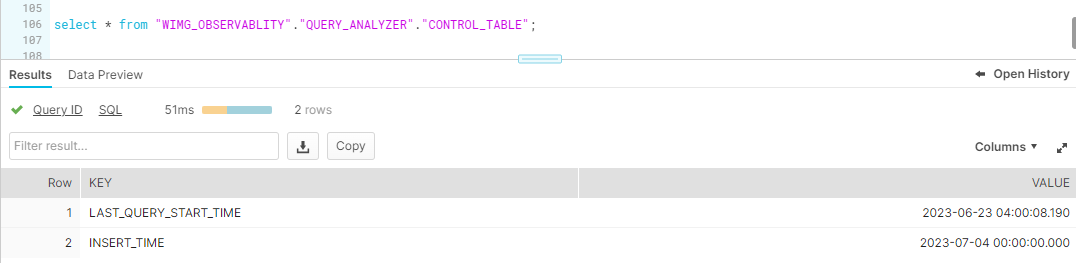
1. SNOWFLAKE.ACCOUNT\_USAGE.QUERY\_HISTORY
2. Get\_query\_operator\_stats
3. WIMG\_OBSERVABLITY.PERFORMANCE\_MONITOR\_SCHEMA.BAD\_QUERY

## **Output Format & Details**

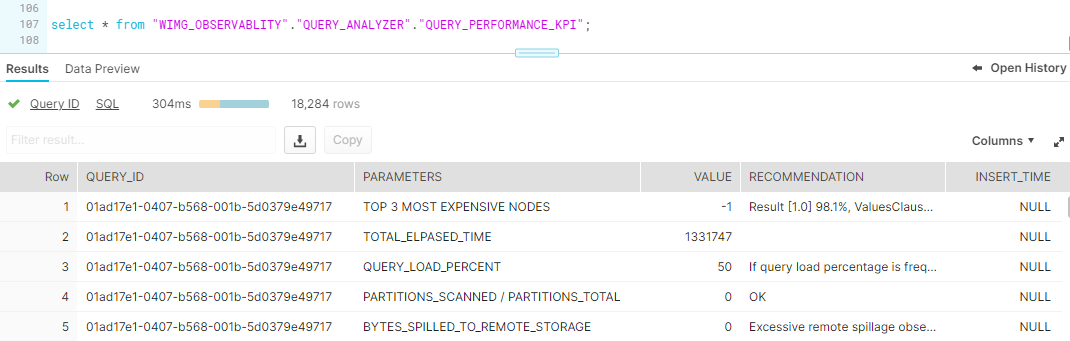
1. QUERY\_ANALYZER Output : Basically it captured the below 14 keys



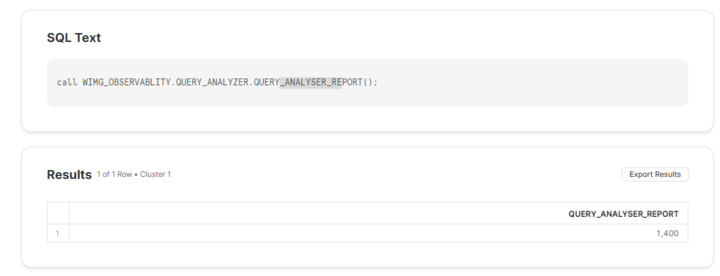
1. Control\_Table Output :



1. QUERY\_PERFORMANCE\_KPI Output :



1. QUERY\_ANALYSER\_REPORT Output :



## Appendix

### Source Code