PIM Training Program

SQL

Query Optimization

Learning Objective

- At the end of the module, you will be able to write efficient queries that will reduce run-time and cost by applying best practices

Agenda

- Optimization Best Practices
  - Adding partition columns
  - Replacing distinct with Group BY
  - Temp Table creation

➢Optimization Best Practices

Adding Partition Columns

Restrict Spectrum Scan by adding filter on Partition Columns

Always have a filter on partition keys (like REGION_ID / MARKETPLACE_ID / SNAPSHOT_DAY) to avoid full table scan. Adding filters on those columns will boost the query performance as well

Temp Table Creation

Create a temp table with only required columns for each external table and distribute it on the joining column (like DISTKEY(ASIN)). We observed that it reduces the scan size and performs faster. Also with data distribution on joining column, joins happen faster compare to non distribution. It will also eliminate the need to hit that table multiple times incase it’s used more than once in the same sql.

Temp Table Creation

Query to get Scan Size and Cost of a SQL

In order to get the scanned size and spectrum cost of a particular sql, we can use the below. Query id will be available when we run a job through Datanet.

```
SELECT QUERY,
      ROUND(ELAPSED :: FLOAT / 1000 / 1000, 3) AS
ELAPSED_SEC,
      S3_SCANNED_ROWS,
      S3QUERY_RETURNED_ROWS,
      S3_SCANNED_BYTES,
      S3QUERY_RETURNED_BYTES,
      MAX_RETRIES,
      S3_SCANNED_BYTES,
      ROUND(CASE WHEN S3_SCANNED_BYTES < 10000000
                  THEN 0.00005 /* 10*(5/1000/1000) */
                  ELSE S3_SCANNED_BYTES*(5/1024^4) END,3) AS
COST
FROM   SVL_S3QUERY_SUMMARY
WHERE  QUERY = xxxx
```

For more information about Redshift Spectrum best practices, please refer here

Everyone can create a Datanet job on their own with this query to check the Scan cost and performance of their SQL

END