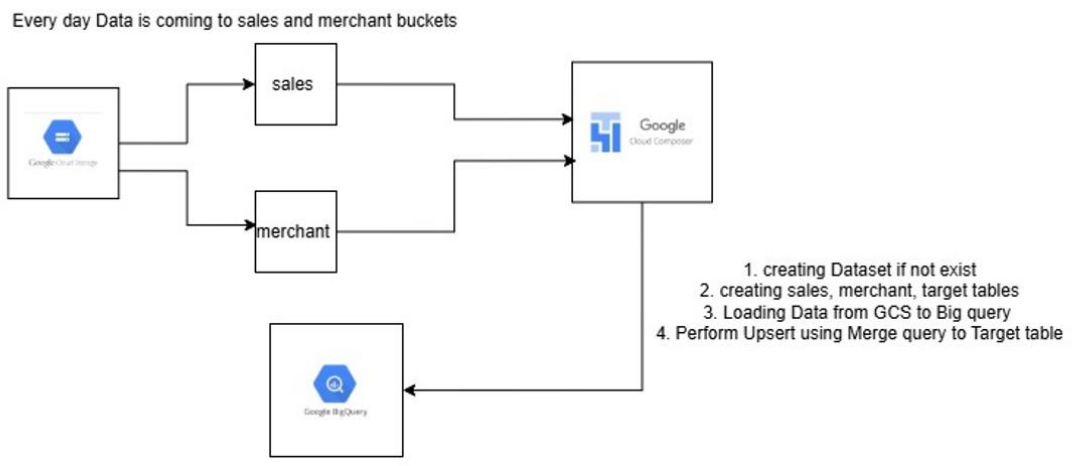
Walmart Data Ingestion From GCS to BigQuery

Project Architecture


Every day Data is coming to sales and merchant buckets
sales
merchant
Google Cloud Composer
1. creating Dataset if not exist
2. creating sales, merchant, target tables
3. Loading Data from GCS to Big query
4. Perform Upsert using Merge query to Target table

Requirements

1. Create a bucket in GCS for merchant and sales.
2. Create Cloud Composer for Airflow so that you can upload the airflow code in Dag Section.

Airflow Code Explanation

1. Import the required dependencies like BigQueryCreateEmptyDatasetOperator, BigQueryCreateEmptyTableOperator, BigQueryExecuteQueryOperator

   BigQueryCreateEmptyDatasetOperator: This operator creates a new dataset in Google BigQuery .
   BigQueryCreateEmptyTableOperator: This operator creates a new, empty table in an existing BigQuery dataset.
   BigQueryExecuteQueryOperator :This operator allows you to execute SQL queries against a BigQuery dataset, making it useful for performing operations such as data transformation, table updates, and querying.

2. Start with Default Arguments
3. Task 1 is creating BQ Dataset if not exist
4. Task 2 is for creating Sales, Merchant, Target tables respectively
5. Task 3 is for Loading Data from GCS to BigQuery
6. Task 4 is for Joining two tables data into arget Table.

Observations :

Clear and Structured ETL Pipeline:

The code defines a clear and structured ETL (Extract, Transform, Load) pipeline for Walmart sales data. It uses appropriate task operators for each stage of the pipeline:

- **Extract**: Data is extracted from Google Cloud Storage (GCS).
- **Transform**: A MERGE query is used for upserting data into the final BigQuery table.
- **Load**: The data is loaded into BigQuery tables in different stages (raw, staging, and final).

Task Grouping for Data Loading:

- The code makes use of **TaskGroup** to group the data-loading tasks (gcs_to_bq_merchants and gcs_to_bq_walmart_sales). Task grouping helps to logically organize related tasks in a visually hierarchical manner, making it easier to manage and monitor tasks.
- Task grouping ensures that both GCS to BigQuery load operations are executed together and grouped in the DAG visualization.

Separation of Staging and Target Tables:

- The use of **two separate BigQuery tables**—walmart_sales_stage and walmart_sales_tgt—shows a well-thought-out design for handling data before and after transformation.
- **Staging Table (walmart_sales_stage)**: Temporary storage for raw data before it is merged.

- **Target Table (walmart_sales_tgt)**: The final storage for transformed, cleansed, and up-to-date data.

Use of MERGE SQL for UPSERT:

- The pipeline uses a **MERGE SQL query** in the merge_walmart_sales task to upsert the data into the target table.
- This approach ensures that the target table remains synchronized with the latest available data without duplicating records or losing information.

Use of Google Cloud Operators:

- **BigQueryCreateEmptyDatasetOperator**: Ensures that the dataset is created if it doesn't exist.
- **BigQueryCreateEmptyTableOperator**: Creates the required BigQuery tables with specific schemas.
- **GCSToBigQueryOperator**: Handles the data transfer from GCS to BigQuery efficiently.
- **BigQueryExecuteQueryOperator**: Executes SQL queries (in this case, the MERGE operation) on BigQuery.

Error Handling and Retries:

- The **default_args** dictionary defines basic retry logic ('retries': 1), ensuring that tasks are retried once in case of failure.

Scheduling and Catchup:

- The **schedule_interval='@daily'** ensures the DAG runs every day.
- **catchup=False** means that if the DAG is paused or hasn't run for a specific time, it will only run for the most recent date, and it won't backfill the missed schedules. This avoids potential issues with large volumes of historical data and ensures that only the latest data is processed.