Senior DE test tasks

Dataset: bigquery-public-data.new_york_taxi_trips

Languages: SQL, Python

Orchestration: Cloud Composer

Data storage: BigQuery

Expected Deliverables:

- A detailed design document outlining your solution architecture, choice of GCP components, and rationale for these choices.
- Code repository (e.g., GitHub) containing all scripts and code used in the implementation, including clear documentation on how to run the solution.

Task 1

Create a pipeline for Yellow taxi with the following requirements:

General requirements:

- The pipeline is build using metadata to generate the operations.
- Pipeline must be resilient to Cloud Composer transient fails.
- Code must be modular.
- Code the query should be generated from the metadata.
- Source/destination tables are defined in metadata.
- Columns that require transfromation are passed using metadata.
- Each transformation can have any number of columns.
- Transfromation are appending the columns at the end of the dataset, original columns are not affected.
- Lookup tables are passed in metadata.
- After transformation data is appened to destination table.
- Pipeline executes daily and gets data for the previous day by ride end date column. The watermark column is passed in metadata.

Operations:

- Transformation:
  - Convert ride start and end time to UTC timestamp.
  - Add names for pick and dropoff location.
- Validation:
  - Check that there are no anomalies in number of records from previous day compared to records for the last month. All anomalies are logged.

Task 2

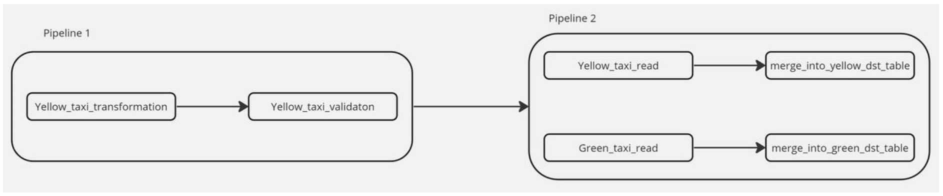
Create a pipeline with following requirements:

General Requirements:

- The pipeline is build using metadata to generate the operations.
- Pipeline must be resilient to Cloud Composer transient fails.
- Code must be modular.
- Code the query should be generated from the metadata.
- The tables for processing are passed in metadata. Any number of tables can be passed.
- All tables have the same write behaviour.

Pipeline requirements:

- Green and Yellow taxi data is used. Yellow taxi data is taken after transformations from task 1.
- Pipeline is executed immediately after pipeline from task 1 is successfully completed.
- Green and Yellow taxi data processed in paralles, written in corresponding destination table which are partitioned by month_year.
- The data is written to destination table by overwriting the latest partition.
- For Yellow taxi dst table should contain per vendor per hour count of the passangers.
- For Green taxi dst table should contain per vendor per day tip amount.
- Make sure that long running BQ query are not blocking new Apache Airflow tasks.
- The destination tables must be protected against user manipulation.
- The only specific users (different) can read data from green and yellow taxi destination tables.


Pipeline 1
Yellow_taxi_transformation
Yellow_taxi_validaton
Pipeline 2
Yellow_taxi_read
merge_into_yellow_dst_table
Green_taxi_read
merge_into_green_dst_table

Task 3

Language: SQL

Description:

Table *transcations* has information about start and end dates of the transaction, duration of the transaction in month (values - any number [0..12]) and subscription billing day.

For monthly recognized revenue calculation it’s necessary to create a database augnostinc sql query that will split transactions into monthly transactions and store them in *rr_transactions* table in such way that:

- All monthly transactions except first monthly transaction have duration equal to 1 month and total sum of all durations from monthly transactions should be equal to duration of the original transaction.
- End date of the monthly transactions should be in line with subscription billing day, end date of the last monthly transaction is equal to end date of the original transaction.
- Start date of the first monthly transactions should be equal to start date of the original transaction. For the rest it should be equal to end date of previous monthly transaction.

Example:

transcations table

id	**subscription_billing_day**	**start_date**	**end_date**	**duration_month**
1	29	2024-01-29	2024-05-29	3
2	12	2024-01-12	2024-07-12	6

rr_transcations table

id	**rr_start_date**	**rr_end_date**	**duration_rr**
1	2024-01-29	2024-02-29	1
1	2024-03-29	2024-04-29	1
1	2024-04-29	2024-05-29	1
2	2024-01-12	2024-02-12	1
2	2024-02-12	2024-03-12	1
2	2024-03-12	2024-04-12	1
2	2024-04-12	2024-05-12	1
2	2024-05-12	2024-06-12	1
2	2024-06-12	2024-07-12	1

Source Table DDL:

```
CREATE TABLE transcations (
id int,
subscription_billing_day int,
start_date date,
end_date date,
duration_month float)
```

Result table DDL:

```
CREATE TABLE rr_transcations (
id int,
rr_start_date date,
rr_end_date date,
duration_rr float)
```