Olist E-Commerce Data Engineering Project Functional Documentation

Jaykumar Dhola
Jiaying Liang
Lydia Avikoke Timona
Oluwasegun Ajetunmobi
Naayi Tei-Dornoo

1. Introduction

This end-to-end project includes all the essential steps in a data analytical process, consisting of collecting data, storing data, validating data, processing data and creating analytical report out of it to gain insights. This project uses the olist e-commerce dataset from Kaggle. Terraform is used to set up infrastructure on Google Cloud Platform. The python library Great Expectations is used for data validation. Airflow is used to orchestrate the data pipelines. All the pipelines are hosted in a Docker compose container. At last, Google Looker Studio is used to create the analytical report.

2. Architecture

The first pipeline downloads olist e-commerce data from Kaggle using Kaggle API and uploads it directly onto Google Storage Bucket, which serves as the data lake. The three data tables which are relevant for the report are validated with Great Expectations in the second pipeline. Data that pass the validation then get ingested in a staging dataset which serves as the staging layer on BigQuery. The third pipeline takes data from the staging layer and performs aggregations, joins them into one denormalized table that is ready for analytical use, and ingests the table into the warehouse dataset which serves as the warehouse layer on BigQuery. At last, the data in the warehouse layer is connected to Google Looker Studio and an analytical report is created with it. Figure1 illustrates the whole process.

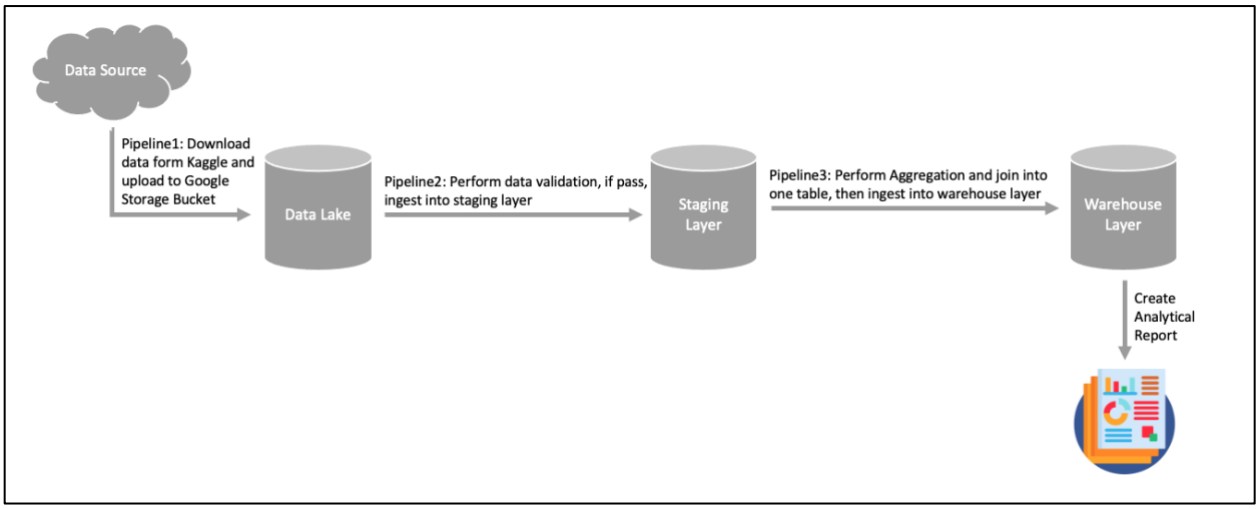

Figure1: Architecture

3. Infrastructure

A Google Storage Bucket called “de_data_lake_the-data-engineering-project” serving as the data lake is created and two datasets, one called “de_dataset_staging” serving as the staging layer and one called “de_dataset_warehouse” serving as the data warehouse layer, are created on BigQuery with Terraform. The google authentication JSON needs to be put in the terraform folder.

4. Pipeline1: data_to_bucket_dag

The first pipeline downloads the olist e-commerce dataset from Kaggle (https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce) using BashOperator and Kaggle API. The Kaggle API key JSON should be placed under the .kaggle folder in the airflow folder. The zip file gets unzipped and the 9 csv files are converted into parquet format and uploaded onto Google Storage Bucket. The google authentication JSON needs to be put under the .google folder in the airflow folder. Figure2 shows the pipeline DAG.

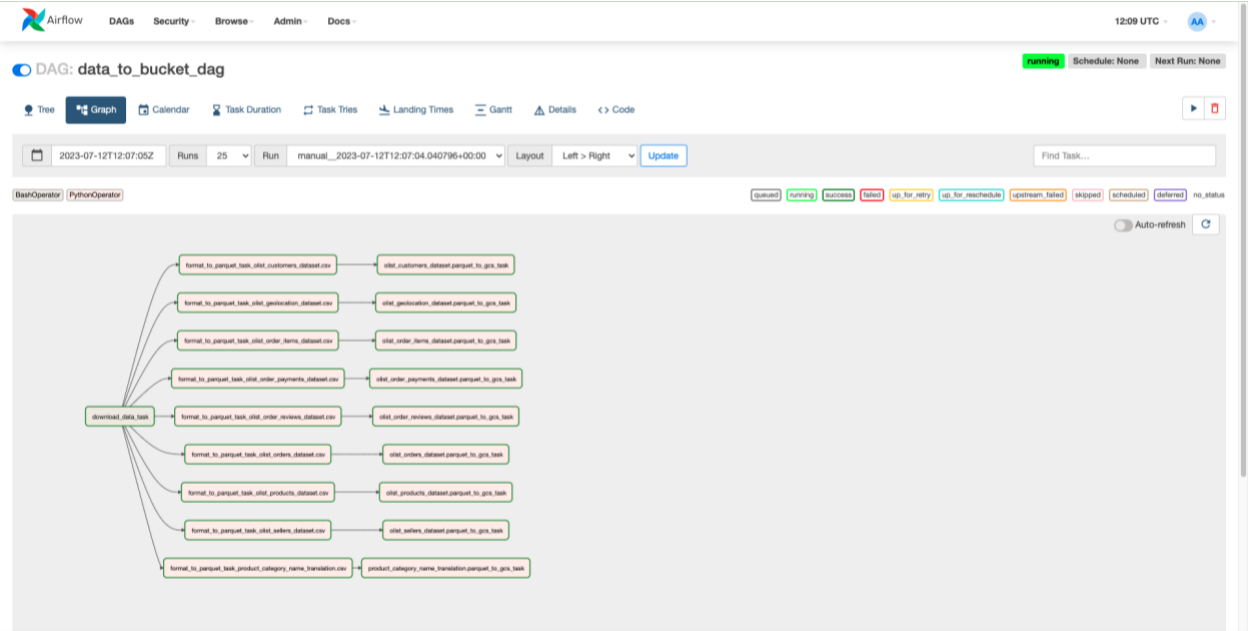

Figure2: Pipeline1

5. Pipeline2: data_quality_check_with_ge_dag

The second pipeline creates a testing dataset on BigQuery, transfers the 3 out of 9 parquet files, which are needed for the report, into 3 tables in the testing dataset. They are the orders table, customers table and the payments table. For each table, one expectation suite JSON and one checkpoint yml are defined. All the files can be found under airflow/config/ge. After validation, there is a task checking the validation result using xcom.pull function. If the validation is successful, the testing table gets copied into a new table in the staging dataset, if not, there is a dummy task shows “validation fails” and there will be no further operation for this table. When all the upstream tasks are done, the testing dataset gets deleted. Figure3 shows the second DAG. Figure4 shows the validation results in great expectations docs.

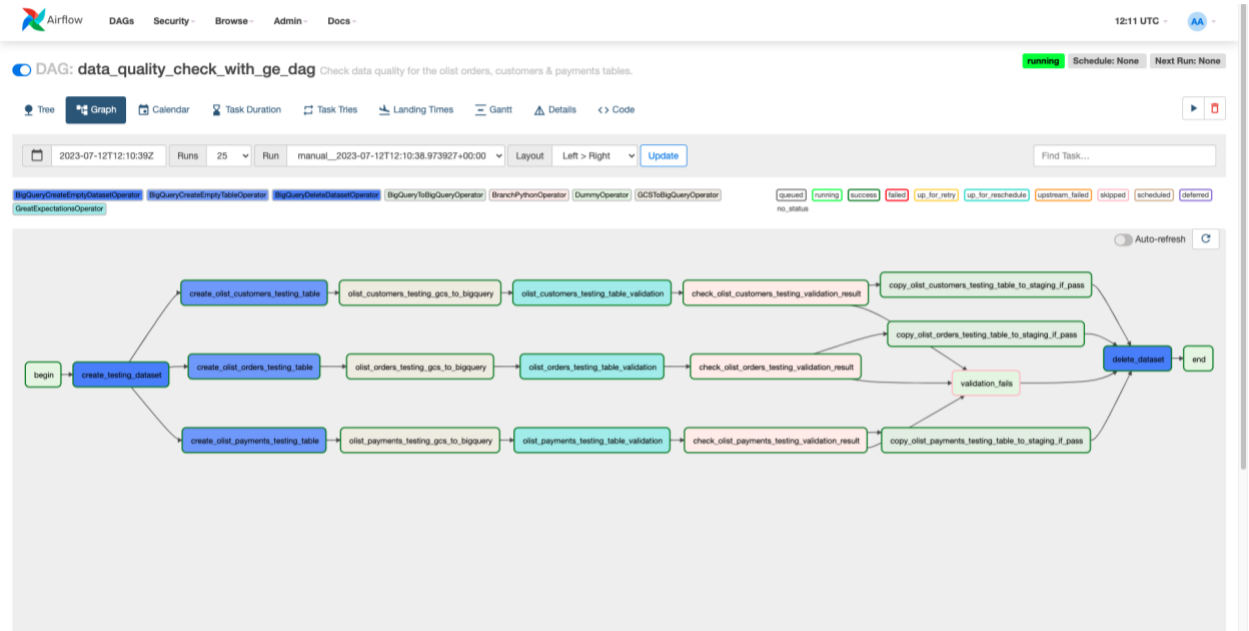

Figure3: Pipeline2

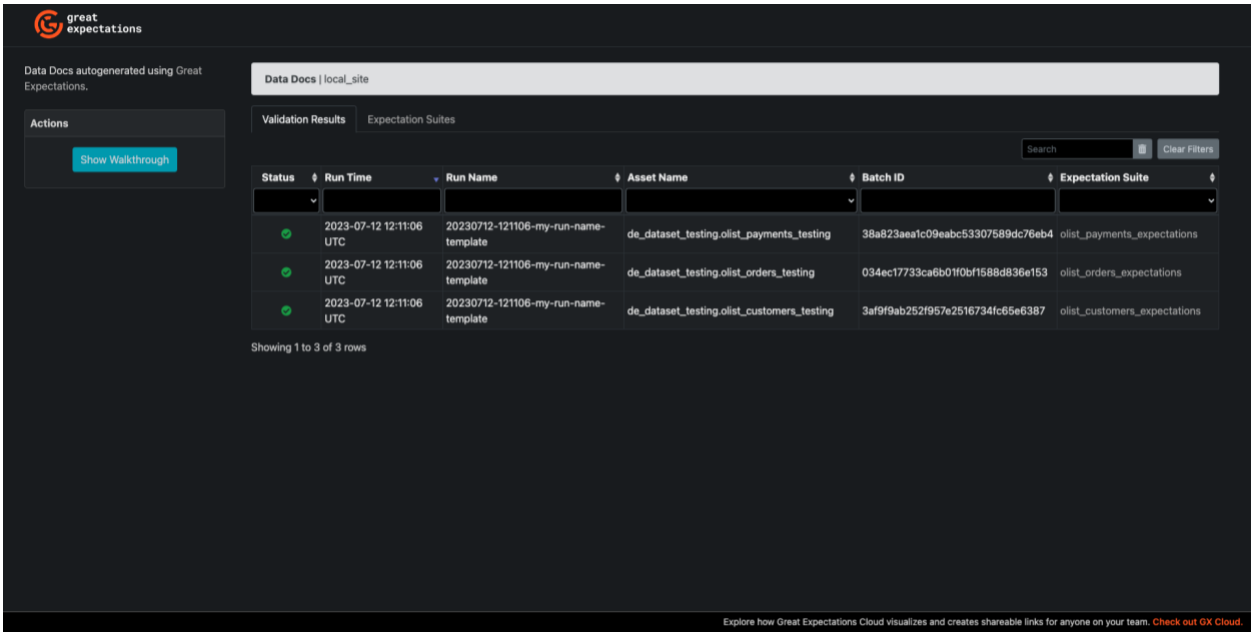

Figure4: Great Expectations Validation Results

6. Pipeline3: data_transformation_dag

BigQueryExecuteQueryOperator is used to perform SQL transformation and joins for the 3 tables in the staging layer. For the payments table, customers can use multiple payments method for one order, so payment_value is aggregated for the same order_id to get the total value for each order. Orders table is joined with customers table using customer id to get customer info only for delivered orders, and then the resulting table is joined to the aggregated payments table to get the total order value. Order year and

month is extracted from the order purchase timestamp to form a new column. And the table is partitioned by year based on the order_purchase_timestamp field. At last, this joined table called "order_customer_value_table" is written into the warehouse layer, ready for the analytical use. Figure5 shows the third DAG.

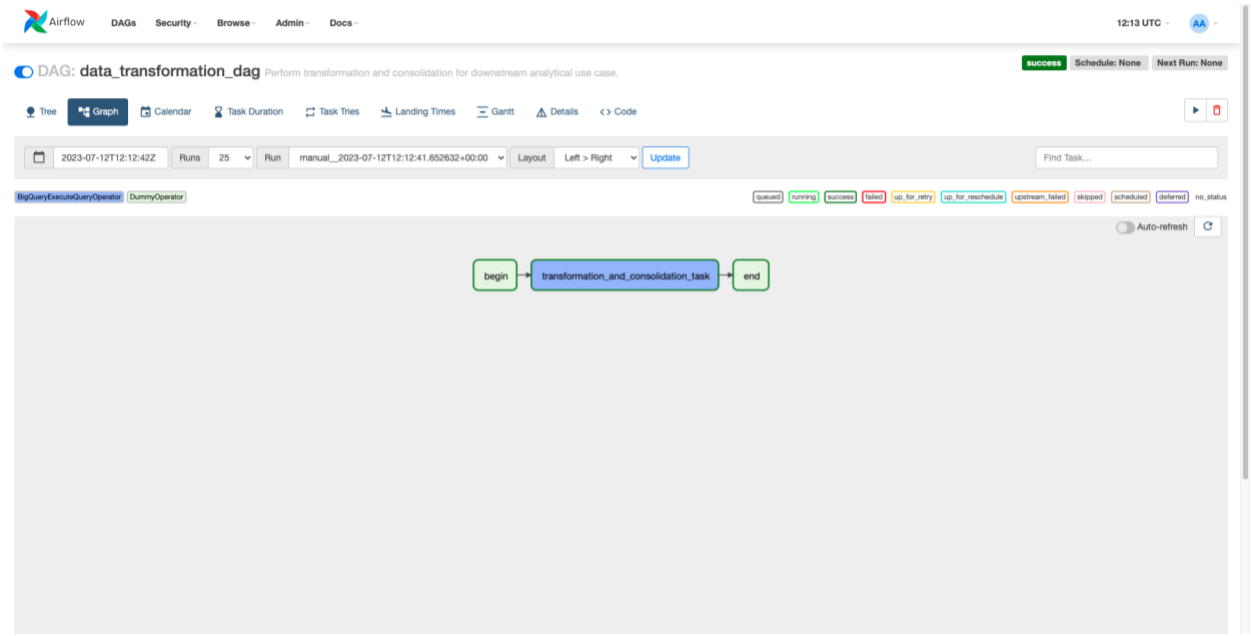

Figure5: Pipeline3

7. Analytical Report: olist e-commerce 2017 sales report

The data in the warehouse is connected to Google Looker Studio and an analytical report is created. The report contains 3 KPIs and two graphs: The first one is for the top 10 states with highest sales and number of orders. And the second one is for monthly sales in 2017. All the graphs apply date filter to show only 2017 data. The report is to be found under: https://lookerstudio.google.com/reporting/aad507fa-648b-4a2b-920c-bfb3a784e234