Sephora E-Commerce Review Analysis

Self Overview

- Hello my name is Nanda Muhammad .I am a final-year diploma student in Electronics Engineering at Politeknik Negeri Malang with a GPA of 3.52/4.00. I have a strong interest in the data field, particularly data engineering, data science, and data analysis. Currently, I am undertaking relevant courses and a bootcamp to acquire foundational skills, with a score of 89/100. I have developing knowledge of data modeling, data quality, and building data pipelines. I am eager to deepen my expertise and learn new things. I am adaptable, a quick learner, and committed to achieving the best results in the data industry. In the past, I worked as an Embedded Engineer Intern, where I tackled engineering problems in micro edge devices.

Overview Project

E-Commerce Data Pipeline GCP

I make a e-commerce data pipeline that using the product and the customer data. The data source is from kaggle, then I make the pipeline using airflow as orchestrator with docker containerization and then for the rest I am using GCP it includes the services like cloud storage, dataproc, bigquery, and looker studio

Background

Background

E-commerce needs to manage their data effectively, so we need to create a robust pipeline to get some insight of the data.

Purpose

- Building reliable pipeline to support a lot of e-commerce data.
- Ensuring Data Driven Insight for E-Commerce Bussiness

Objective

Project Objective

- Product Performance Analysis, Identify Top Product
- Review Customer Analysis, Customer Segmentation
- Visualize the distribution

Expected Output

- Robust Data Pipeline
- Dashboard from Product and Review analysis
- Getting some actionable insight

Dataset

Using 1.110.000 records data

- For more details use this link to the kaggle

Timeline

Data Extraction

- Extract data from kaggle using kaggle API

Data Storage

- Save the data into Data Storage

Data Preprocessing and Modeling

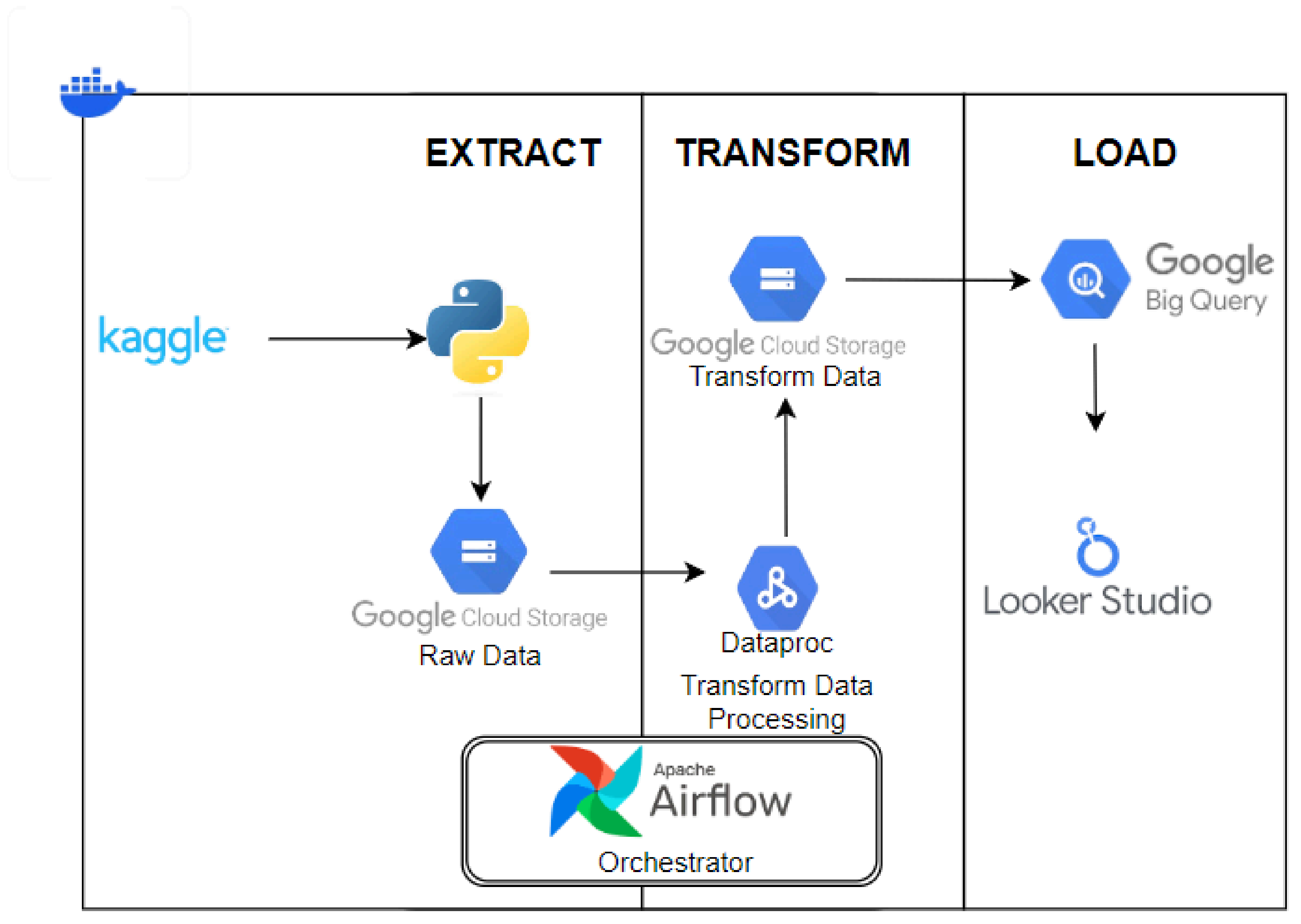
- Cleaning the data, null values, duplicate values
- Utilise spark to transform the data into star schema

Data Loading

- Load the data into Data Warehouse Bigquery

Data Visualization

- Visualize the data using Looker Studio



Architecture

Extract

- Extract the data using Python and kaggle API
- Save the raw data to data storage

Transform

- Clean the data using pyspark, duplicate, missing values, etc.

Load

- Load the data to Data Warehouse, Bigquery
- Get visualization using looker studio

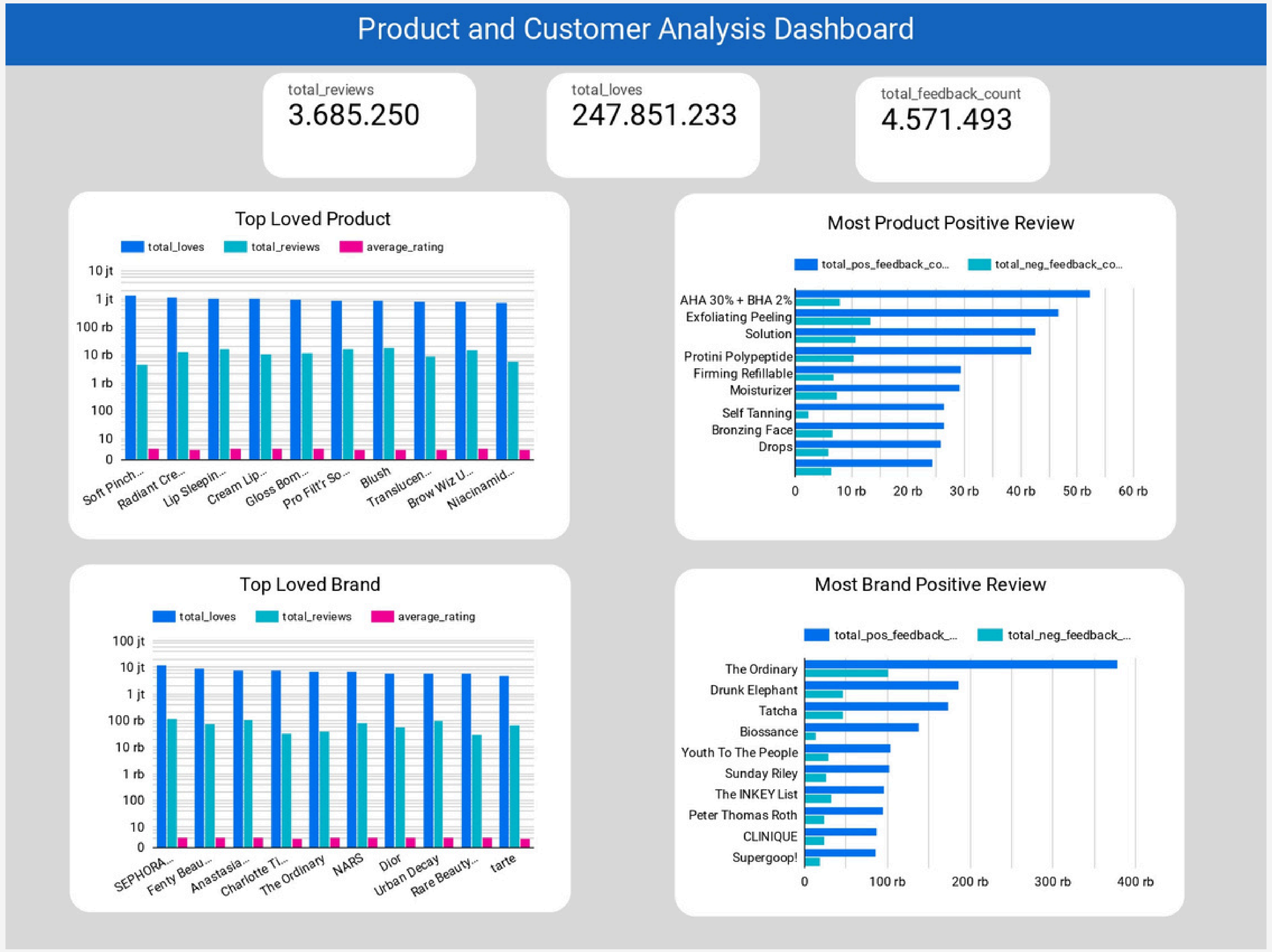
Data Orchestration

- Using Airflow for scheduling and monitoring data pipeline

Containerization

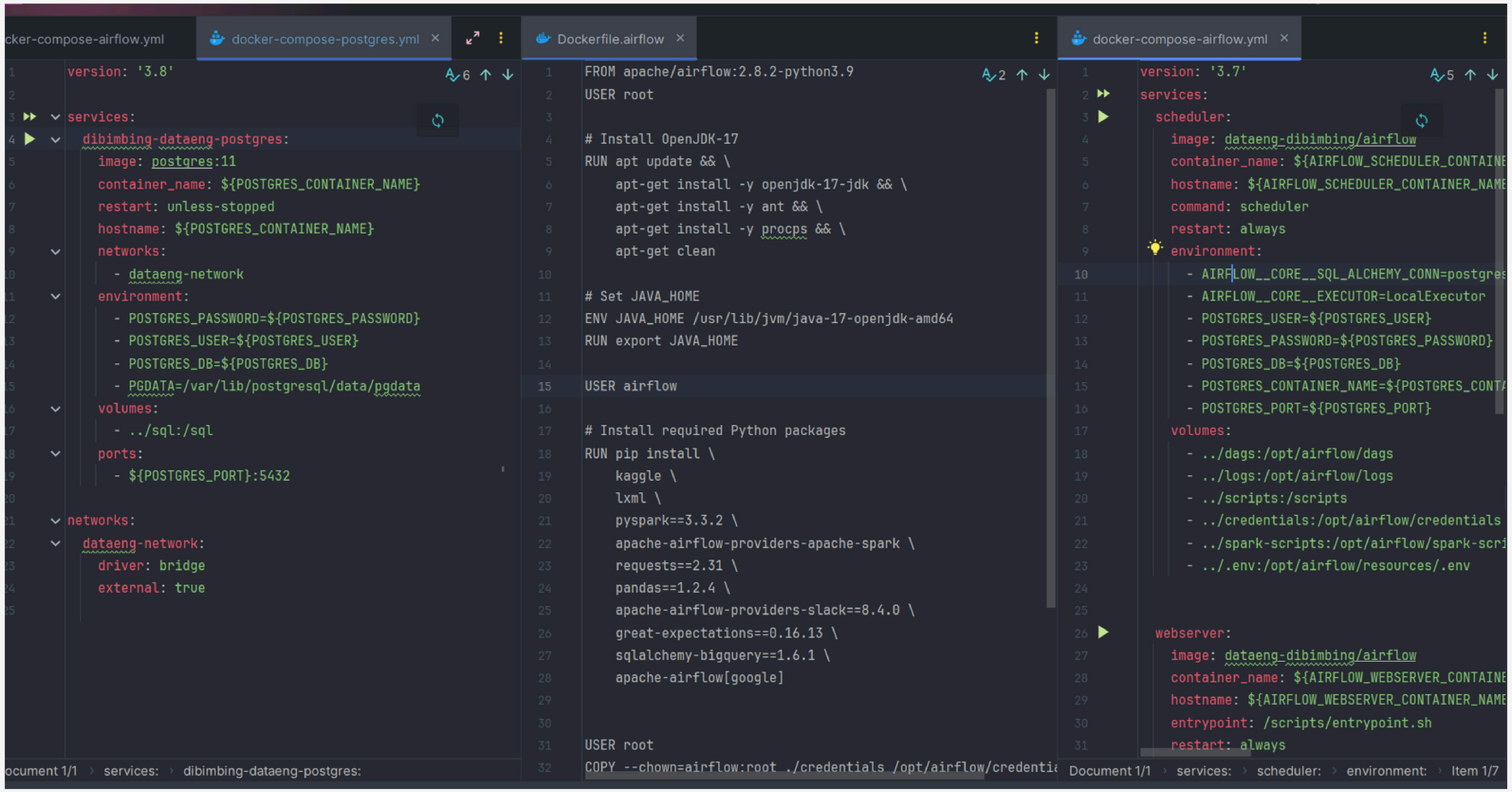
- Using Docker for seamless Containerization

VISUALIZATION


Product and Customer Analysis Dashboard

total_reviews	total_loves	total_feedback_count
3.685.250	247.851.233	4.571.493

Code Overview



```yaml
version: '3.8'

services:
  dibimbing-dataeng-postgres:
    image: postgres:11
    container_name: ${POSTGRES_CONTAINER_NAME}
    restart: unless-stopped
    hostname: ${POSTGRES_CONTAINER_NAME}
    networks:
      - dataeng-network
    environment:
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_DB=${POSTGRES_DB}
      - PGDATA=/var/lib/postgresql/data/pgdata
    volumes:
      - ../sql:/sql
    ports:
      - ${POSTGRES_PORT}:5432

networks:
  dataeng-network:
    driver: bridge
    external: true
```

```
ocument 1/1 › services: › dibimbing-dataeng-postgres:
```

```dockerfile
FROM apache/airflow:2.8.2-python3.9
USER root

# Install OpenJDK-17
RUN apt update && \
    apt-get install -y openjdk-17-jdk && \
    apt-get install -y ant && \
    apt-get install -y procps && \
    apt-get clean

# Set JAVA_HOME
ENV JAVA_HOME /usr/lib/jvm/java-17-openjdk-amd64
RUN export JAVA_HOME

USER airflow

# Install required Python packages
RUN pip install \
    kaggle \
    lxml \
    pyspark==3.3.2 \
    apache-airflow-providers-apache-spark \
    requests==2.31 \
    pandas==1.2.4 \
    apache-airflow-providers-slack==8.4.0 \
    great-expectations==0.16.13 \
    sqlalchemy-bigquery==1.6.1 \
    apache-airflow[google]


USER root
COPY --chown=airflow:root ./credentials /opt/airflow/credentia
```

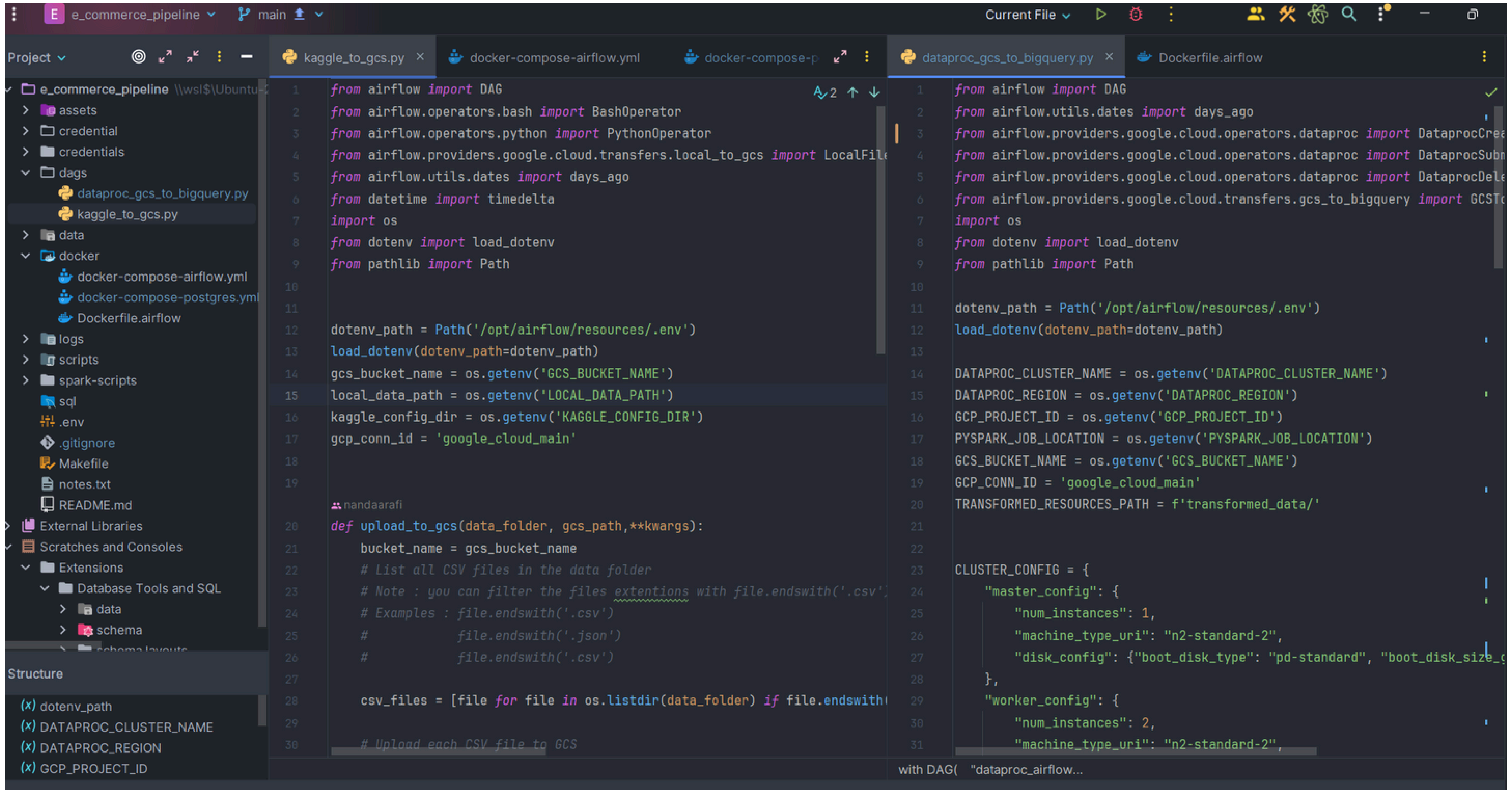
```yaml
version: '3.7'
services:
  scheduler:
    image: dataeng-dibimbing/airflow
    container_name: ${AIRFLOW_SCHEDULER_CONTAINE
    hostname: ${AIRFLOW_SCHEDULER_CONTAINER_NAME
    command: scheduler
    restart: always
    environment:
      - AIRFLOW__CORE__SQL_ALCHEMY_CONN=postgres
      - AIRFLOW__CORE__EXECUTOR=LocalExecutor
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
      - POSTGRES_DB=${POSTGRES_DB}
      - POSTGRES_CONTAINER_NAME=${POSTGRES_CONT
      - POSTGRES_PORT=${POSTGRES_PORT}
    volumes:
      - ../dags:/opt/airflow/dags
      - ../logs:/opt/airflow/logs
      - ../scripts:/scripts
      - ../credentials:/opt/airflow/credentials
      - ../spark-scripts:/opt/airflow/spark-scri
      - ../.env:/opt/airflow/resources/.env


  webserver:
    image: dataeng-dibimbing/airflow
    container_name: ${AIRFLOW_WEBSERVER_CONTAINE
    hostname: ${AIRFLOW_WEBSERVER_CONTAINER_NAME
    entrypoint: /scripts/entrypoint.sh
    restart: always
```

```
Document 1/1 › services: › scheduler: › environment: › Item 1/7
```

Usage

- Using Docker Airflow and Postgres for containerization
- Dockerfile airflow, for jdk installation, python installation and python package installation.

Code Overview



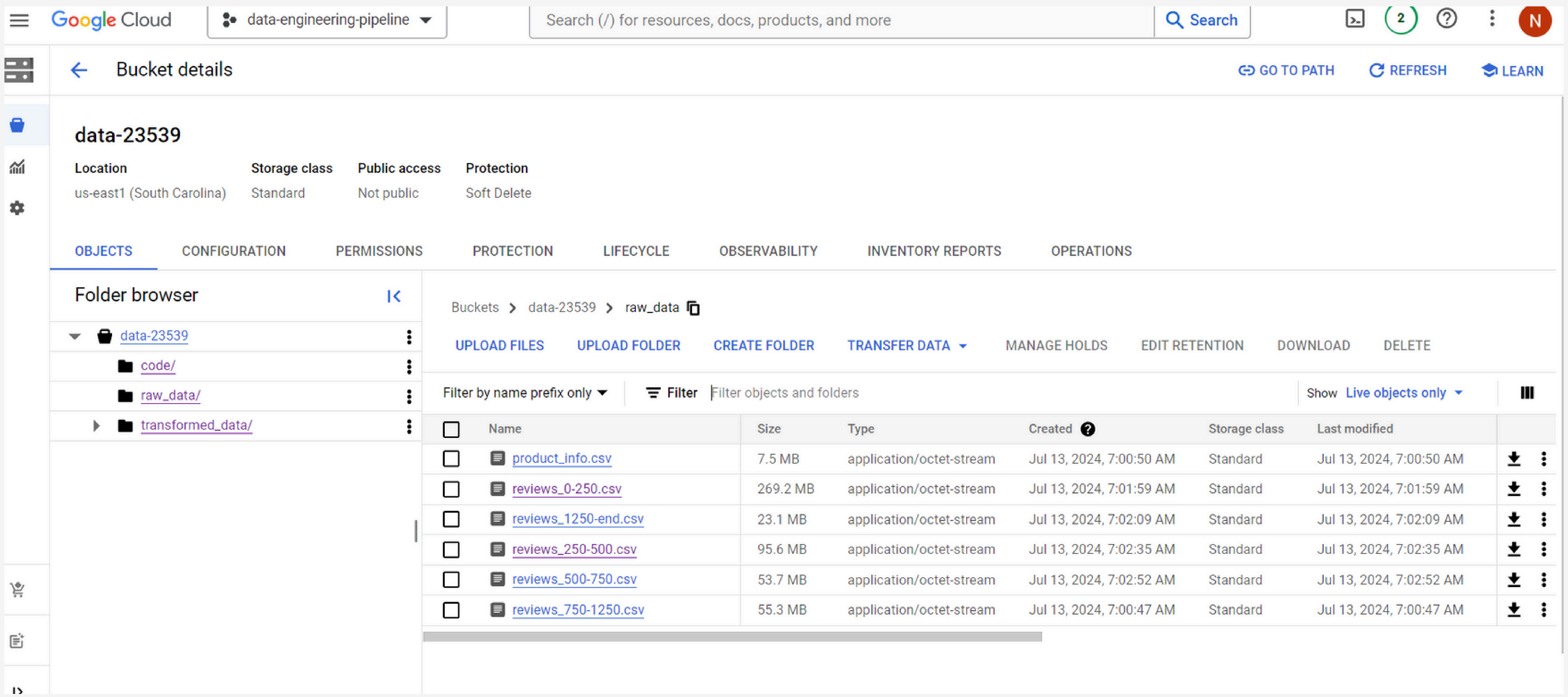
```python
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.python import PythonOperator
from airflow.providers.google.cloud.transfers.local_to_gcs import LocalFile
from airflow.utils.dates import days_ago
from datetime import timedelta
import os
from dotenv import load_dotenv
from pathlib import Path


dotenv_path = Path('/opt/airflow/resources/.env')
load_dotenv(dotenv_path=dotenv_path)
gcs_bucket_name = os.getenv('GCS_BUCKET_NAME')
local_data_path = os.getenv('LOCAL_DATA_PATH')
kaggle_config_dir = os.getenv('KAGGLE_CONFIG_DIR')
gcp_conn_id = 'google_cloud_main'


def upload_to_gcs(data_folder, gcs_path,**kwargs):
    bucket_name = gcs_bucket_name
    # List all CSV files in the data folder
    # Note : you can filter the files extentions with file.endswith('.csv')
    # Examples : file.endswith('.csv')
    #            file.endswith('.json')
    #            file.endswith('.csv')

    csv_files = [file for file in os.listdir(data_folder) if file.endswith(

    # Upload each CSV file to GCS
```

```python
from airflow import DAG
from airflow.utils.dates import days_ago
from airflow.providers.google.cloud.operators.dataproc import DataprocCrea
from airflow.providers.google.cloud.operators.dataproc import DataprocSubm
from airflow.providers.google.cloud.operators.dataproc import DataprocDele
from airflow.providers.google.cloud.transfers.gcs_to_bigquery import GCSTo
import os
from dotenv import load_dotenv
from pathlib import Path

dotenv_path = Path('/opt/airflow/resources/.env')
load_dotenv(dotenv_path=dotenv_path)

DATAPROC_CLUSTER_NAME = os.getenv('DATAPROC_CLUSTER_NAME')
DATAPROC_REGION = os.getenv('DATAPROC_REGION')
GCP_PROJECT_ID = os.getenv('GCP_PROJECT_ID')
PYSPARK_JOB_LOCATION = os.getenv('PYSPARK_JOB_LOCATION')
GCS_BUCKET_NAME = os.getenv('GCS_BUCKET_NAME')
GCP_CONN_ID = 'google_cloud_main'
TRANSFORMED_RESOURCES_PATH = f'transformed_data/'


CLUSTER_CONFIG = {
    "master_config": {
        "num_instances": 1,
        "machine_type_uri": "n2-standard-2",
        "disk_config": {"boot_disk_type": "pd-standard", "boot_disk_size_g
    },
    "worker_config": {
        "num_instances": 2,
        "machine_type_uri": "n2-standard-2",
```

Airflow DAG

- kaggle_to_gcs, for dag kaggle api to gcs storage
- dataproc_gcs_to_bigquery
  - creating dataproc cluster
  - submit spark job to transform data to store it on gcs
  - store it to google bigquery

GCP Overview

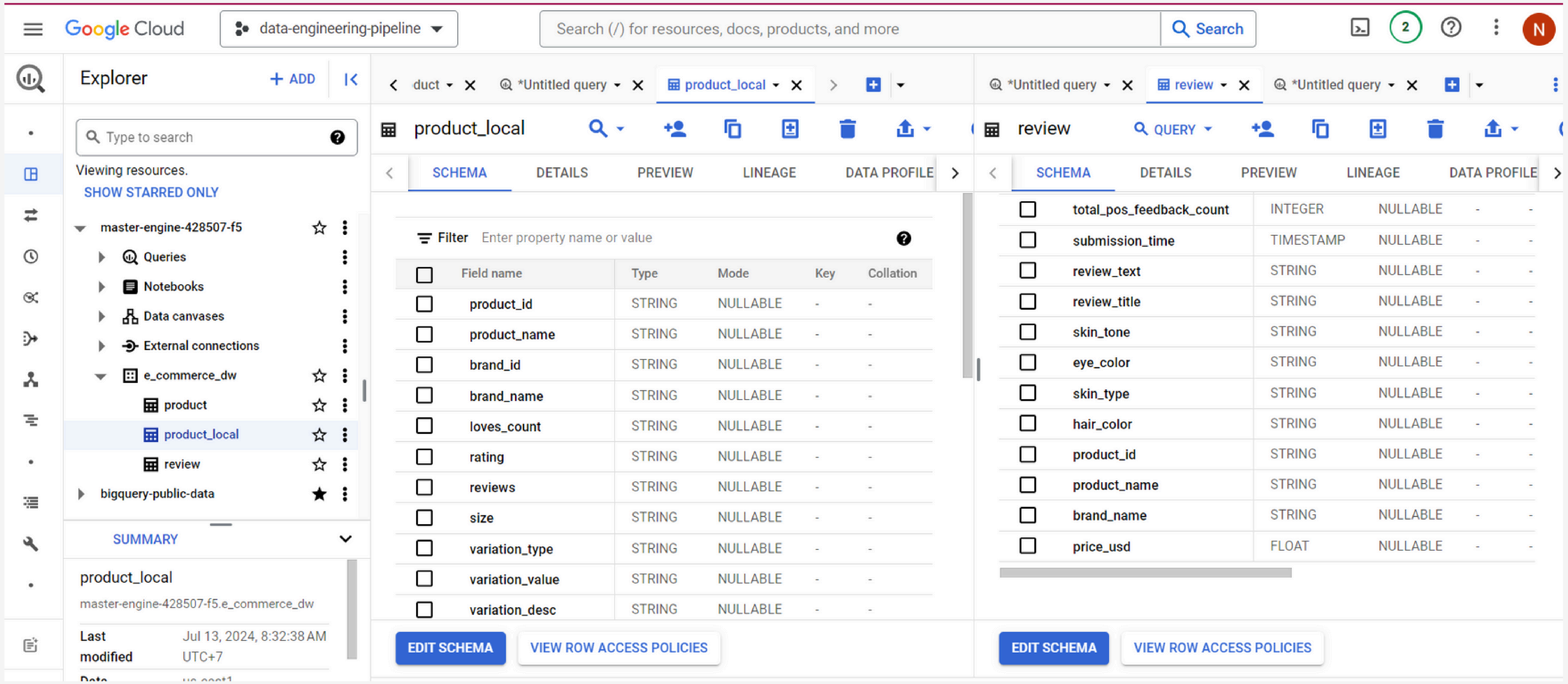


data-23539

Location	Storage class	Public access	Protection
us-east1 (South Carolina)	Standard	Not public	Soft Delete

Name	Size	Type	Created	Storage class	Last modified
product_info.csv	7.5 MB	application/octet-stream	Jul 13, 2024, 7:00:50 AM	Standard	Jul 13, 2024, 7:00:50 AM
reviews_0-250.csv	269.2 MB	application/octet-stream	Jul 13, 2024, 7:01:59 AM	Standard	Jul 13, 2024, 7:01:59 AM
reviews_1250-end.csv	23.1 MB	application/octet-stream	Jul 13, 2024, 7:02:09 AM	Standard	Jul 13, 2024, 7:02:09 AM
reviews_250-500.csv	95.6 MB	application/octet-stream	Jul 13, 2024, 7:02:35 AM	Standard	Jul 13, 2024, 7:02:35 AM
reviews_500-750.csv	53.7 MB	application/octet-stream	Jul 13, 2024, 7:02:52 AM	Standard	Jul 13, 2024, 7:02:52 AM
reviews_750-1250.csv	55.3 MB	application/octet-stream	Jul 13, 2024, 7:00:47 AM	Standard	Jul 13, 2024, 7:00:47 AM

GCP Overview



product_local

master-engine-428507-f5.e_commerce_dw

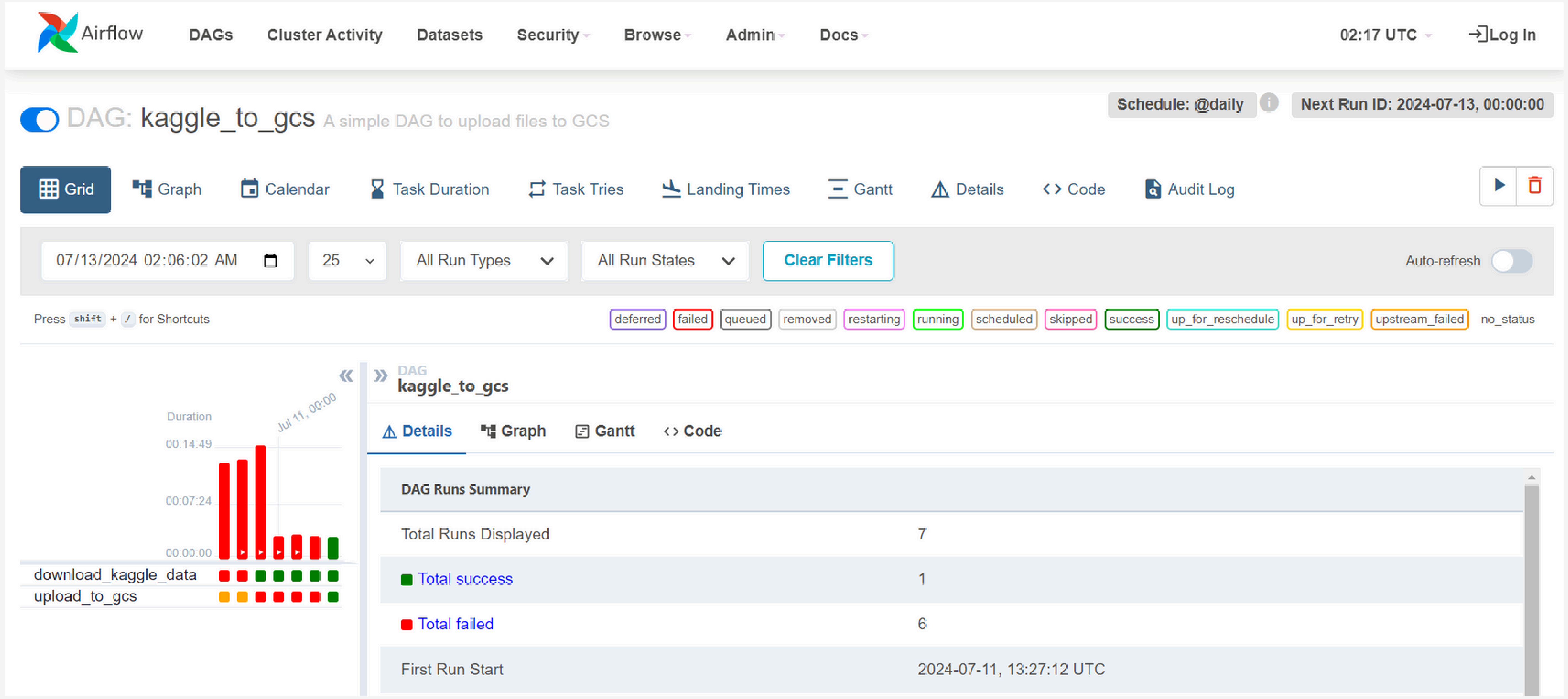
Last modified: Jul 13, 2024, 8:32:38 AM UTC+7

Field name	Type	Mode	Key	Collation
product_id	STRING	NULLABLE	-	-
product_name	STRING	NULLABLE	-	-
brand_id	STRING	NULLABLE	-	-
brand_name	STRING	NULLABLE	-	-
loves_count	STRING	NULLABLE	-	-
rating	STRING	NULLABLE	-	-
reviews	STRING	NULLABLE	-	-
size	STRING	NULLABLE	-	-
variation_type	STRING	NULLABLE	-	-
variation_value	STRING	NULLABLE	-	-
variation_desc	STRING	NULLABLE	-	-

review

total_pos_feedback_count	INTEGER	NULLABLE	-	-
submission_time	TIMESTAMP	NULLABLE	-	-
review_text	STRING	NULLABLE	-	-
review_title	STRING	NULLABLE	-	-
skin_tone	STRING	NULLABLE	-	-
eye_color	STRING	NULLABLE	-	-
skin_type	STRING	NULLABLE	-	-
hair_color	STRING	NULLABLE	-	-
product_id	STRING	NULLABLE	-	-
product_name	STRING	NULLABLE	-	-
brand_name	STRING	NULLABLE	-	-
price_usd	FLOAT	NULLABLE	-	-

Airflow Overview



Airflow | DAGs | Cluster Activity | Datasets | Security | Browse | Admin | Docs | 02:17 UTC | Log In

DAG: kaggle_to_gcs A simple DAG to upload files to GCS

Schedule: @daily | Next Run ID: 2024-07-13, 00:00:00

Grid | Graph | Calendar | Task Duration | Task Tries | Landing Times | Gantt | Details | Code | Audit Log

07/13/2024 02:06:02 AM | 25 | All Run Types | All Run States | Clear Filters | Auto-refresh

Press shift + / for Shortcuts

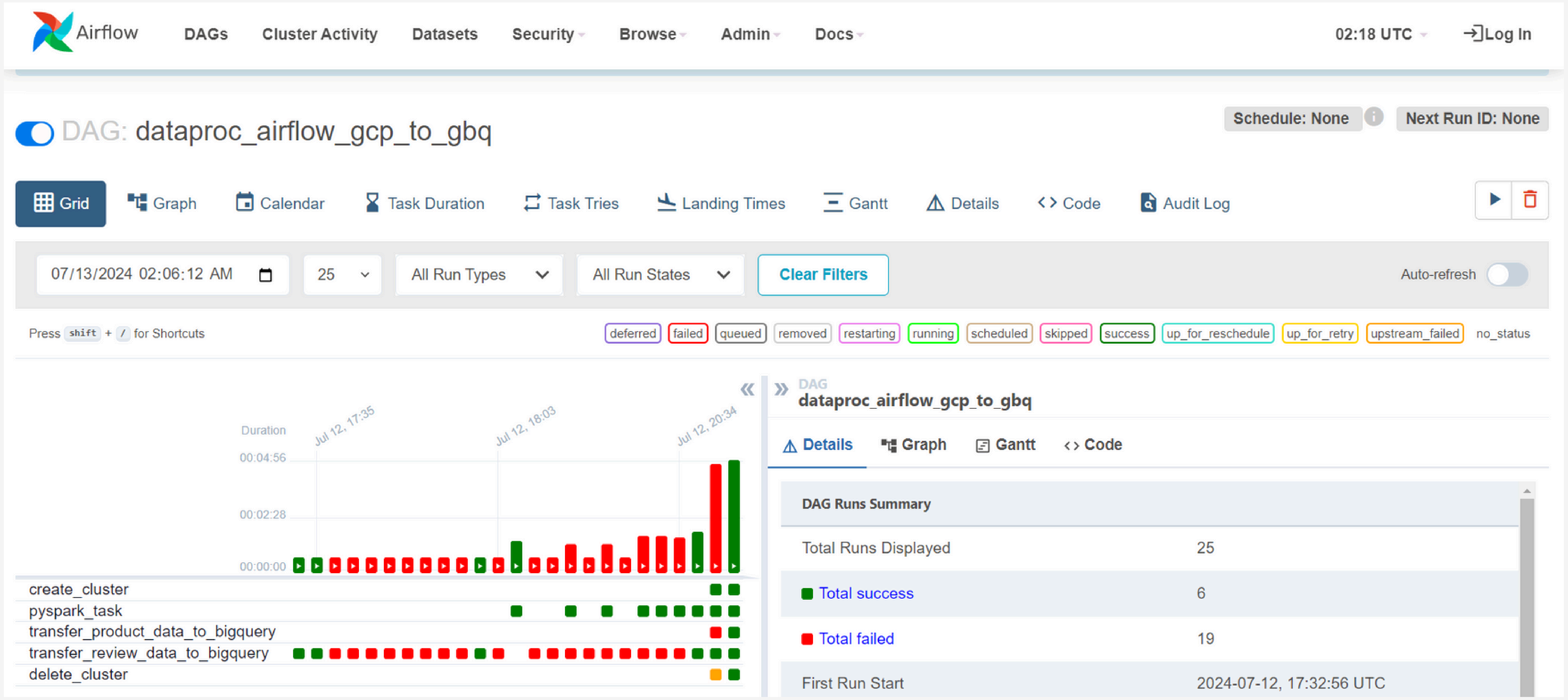
deferred | failed | queued | removed | restarting | running | scheduled | skipped | success | up_for_reschedule | up_for_retry | upstream_failed | no_status

DAG
kaggle_to_gcs

Details | Graph | Gantt | Code

DAG Runs Summary	
Total Runs Displayed	7
🟩 Total success	1
🟥 Total failed	6
First Run Start	2024-07-11, 13:27:12 UTC

Airflow Overview



DAG: dataproc_airflow_gcp_to_gbq

DAG
dataproc_airflow_gcp_to_gbq

DAG Runs Summary	
Total Runs Displayed	25
🟩 Total success	6
🟥 Total failed	19
First Run Start	2024-07-12, 17:32:56 UTC

Limitation and Improvement

Limitation

- Using Denormalized schema that have low cost query but have more cost on storage and data redundancy
- Bad visualization and performance metrics
- No machine learning implementation.

Improvement

- Data Modeling, For improvement we can use normalized schema (star-schema or snowflake schema), or for more advanced or more big data use One Big Table or nested schema. Please see this article Here
- Data Quality, check some quality of the data using tools like Dataplex from GCP or using open source tool like Great Expectation
- More advanced visualization, including some more advanced performance metrics
- Some sentiment analysis or some machine learning model for predicting the product

Thank You

Problem Overview

- The problem found out when I try to upload the data from GCS to BigQuery, result in bad record because of csv format data
- Solved with change the transformed data to GCS with a parquet then upload it to BigQuery using parquet format which is parquet include the data schema.