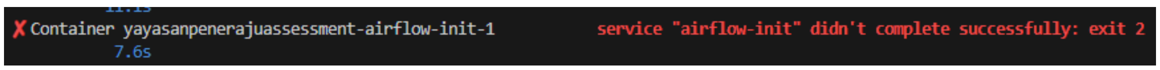
Summary:

For this ETL pipeline project, I encountered an issue during the setup of Airflow, specifically with the airflow-init container, which repeatedly exits and is marked as unhealthy. Despite attempting multiple solutions, I was unable to resolve this issue. The root cause seems to be related to the failure in mounting the Google Cloud Storage (GCS) key, which I believe is preventing the Airflow container from initializing correctly.

Given this, I made the decision to proceed with the project to demonstrate the next steps as if the issue were resolved. To provide a comprehensive view of the transformation process, I have included the data_transformation.py script in the data folder. This script showcases the transformation logic that I would have implemented had there been no issues with the Airflow setup.

While the full ETL pipeline setup is incomplete due to the technical challenges with Airflow, the transformation logic is available for testing and validation.

The image below shows that the airflow-init failed to run.



```
✗ Container yayasanpenerajuassessment-airflow-init-1          service "airflow-init" didn't complete successfully: exit 2
          7.6s
```

Setup and Run:

Follow these steps to set up and run the ETL pipeline:

1. Clone the repository:
   - git clone https://github.com/mirzahazimm/yayasanpeneraju_test.git
   - cd your-repo
2. Place your GCS key:
   - mkdir -p keys
   - cp ~/Downloads/etl-gcs.json keys/etl-gcs.json
3. Create a `.env` file in the project root with the following content:
   - AIRFLOW_UID=50000
   - AIRFLOW_IMAGE_NAME=yayasanpeneraju_test
   - AIRFLOW_WWW_USER_USERNAME=yayasanpeneraju
   - AIRFLOW_WWW_USER_PASSWORD=yayasanpeneraju
   -
   - POSTGRES_USER=yayasanpeneraju
   - POSTGRES_PASSWORD=yayasanpeneraju
   - POSTGRES_DB=yayasanpeneraju
   -
   - REDIS_PASSWORD=yayasanpeneraju
4. Start the Docker stack:
   - docker-compose up --build -d
5. Wait for the `airflow-init` container to complete migrations:
   - docker-compose logs -f airflow-init
6. Open the Airflow UI in your browser to see the etl_pipeline DAG:
   - http://localhost:8080

Explanation of Key Design Choices:

1. **GCS instead of AWS S3****: Google Cloud Storage was chosen because AWS S3 would have required a 24-hour activation delay. I already have an active GCS account, which enabled immediate testing and implementation.

2. **Error Handling**: Each function includes try-except blocks to catch any exceptions that occur during the downloading, transforming, or loading of data, ensuring that any issues are logged for debugging and handled gracefully.

3. **Environment Variables**: The PostgreSQL connection details (user, password, database) are obtained from environment variables (os.getenv()), making the setup flexible and secure.

4. **Data Quality Check**: A basic data quality check is implemented via the row_count_validation() function. It ensures that the data loaded into PostgreSQL is valid by checking if the table is empty, raising an error if no data is present.

5. **Scheduling**: The DAG is set to run daily (schedule_interval='@daily'), ensuring that the ETL process happens automatically each day without manual intervention.

Assumptions:

- The Google Cloud Storage (GCS) key file (gcs_key.json) is mounted correctly to the Airflow container for authentication.

- The PostgreSQL database is correctly set up and running as a Docker container or external service.

- The required libraries such as pandas, sqlalchemy, and google-cloud-storage are installed and available in the Airflow environment.