Final Project – Building Data Pipeline

Rizky Fajar Aditya – JCDEOL001

February 2025

Github: https://github.com/rizkyjarr/final-project-purwadhika

Study Case: Go Hailing Inc. faces growing challenges in Data Access and Quality, hindering decision-making and business agility.

Background and Study Case:

Go-Hailing is a ride-hailing company that provides on-demand transportation services .

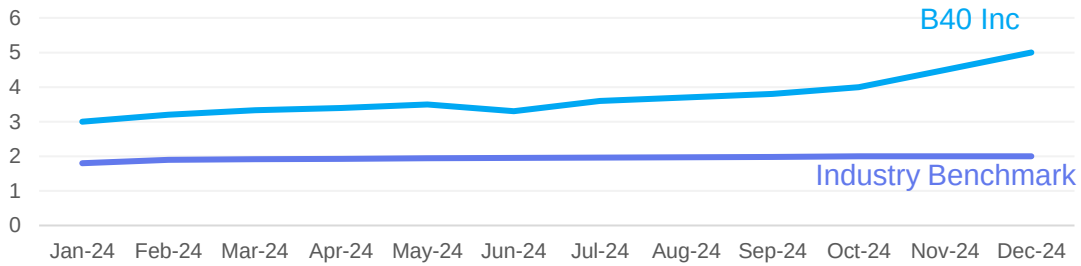
The Sales and Operation Team at Go Hailing. is facing challenges in gathering data for their reports and insights, which is impacting their decision-making processes. To address this, they have enlisted the help of Data Engineering team to **design and implement a robust and scalable data pipeline** that will streamline data collection and processing, enabling faster reporting and more informed decision-making.

Please note: The case study and data presented in this slide are hypothetical and created for illustrative purposes only

Several challenges that makes Go-Hailing Inc is on the verge to develop a robust data pipeline:

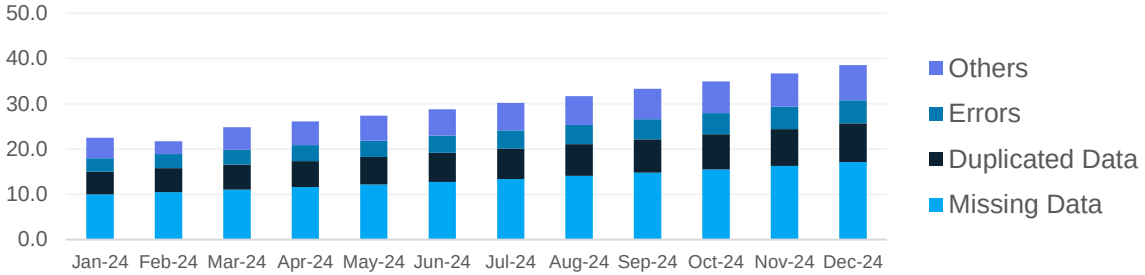
1. The time required to access critical data increases by 4% each month, as the duration from data collection to reporting grows, which delays their ability to gain insights and make decisions

Time needed teed to collect data to reporting, in hours

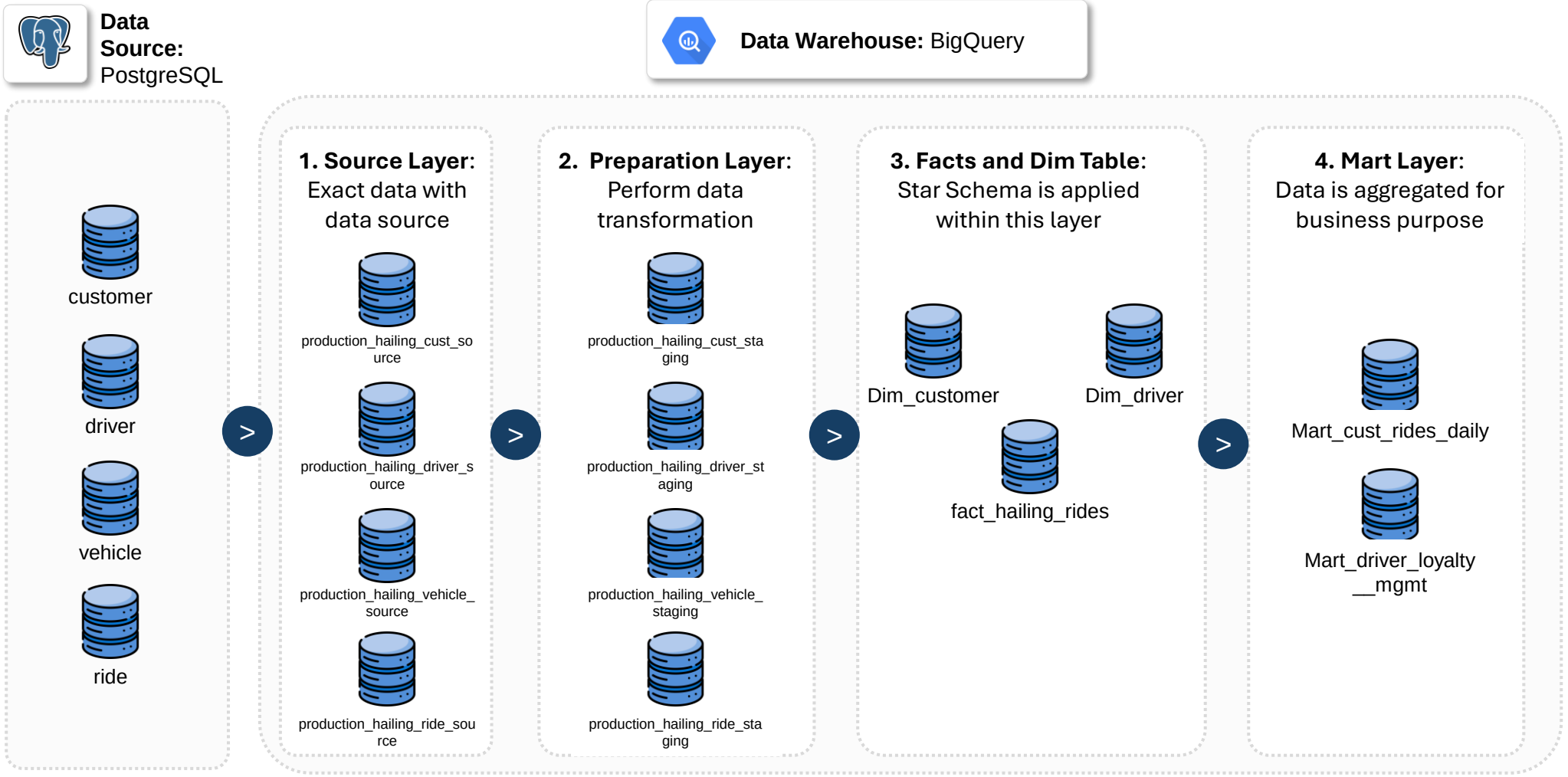


2. Inconsistent data quality makes it challenging to perform comprehensive data analysis

Types of data qualities issues, in #



The 1st stage to resolve the issue, I developed the end-to-end data pipeline as a framework reference.



Remarks:

1. Data pipeline is orchestrated using **Airflow**
2. **DBT (Data Build Tool)** is used for data modelling in Data Warehouse
3. All apps are **containerized** using **Docker.**
4. If any tasks failed or retry it will be notified through **Discord WebHok**
5. **ELT (Extract Transform and Load)** is used in this data pipeline
6. Emphasizing **batching** data pipeline

The 2nd stage is to develop the codes based on the data pipeline architecture design

Code	Code Objective	File Name	Main Logic/Function
Docker	• Build Airflow • Build PostgreSQL • Build DBT	• docker-compose.yaml (Airflow and dbt) << click here >> • docker-compose.yaml (postgresQL) << click here >>	• **Airflow** – build postgre for airflow metadata >> build scheduler for orchestrating tasks >> build webserver to access and trigger DAGS interactively • **PostgreSQL** – build postgre in different container to store dummy data • **DBT (Data Build Tool)** – Build DBT in different container to run models for data modelling within data warehouse. Airflow can send command to DBT to run task using BashOperator (docker exec –it dbt run models). Credentials is also mounted in the volume so it can access BigQuery.
DAG 1	• Generate dummy tables • Load to PostgreSQL	• DAG1_load_to_postgre.py **as main DAG** << click here >> • Generate_data.py **as helper** << click here >> • Postgres_helper.py **as helper** << click here >>	• Execute .sql files to create schema in PostgreSQL >> generate and insert data to PostgresSQL
DAG 2	• Data Ingestion to BigQuery	• DAG2_load_to_bigquery **as main DAG** << click here >> • Bigquery_helper.py **as helper** << click here >>	• Ensure if table exists (creates if not exists) >> fetch data from postgresql >> return to dataframe in pandas >> map data types befor upsert in BigQuery >> perform upsert in bigquery, partitioned by created_at • Table that has no created_at/partitioned columed can also be load into bigquery – emphasizing for modularity
DAG	• Perform data transformation after ingestion	• **DAG3 as main DAG** << click here >> • Models/staging as models in preparation layer << click here >> • Models/facts/ as models in fact and dim table << click here >> • Models/marts/ as sql models in marts layer << click here >>	• Perform data transformation (unify phone number) from source layer to preparation layer >> build star schema model in facts and dim layer >> build aggregated data to gain insights from rides activities, and driver idleness.

Project Results

Successfully generate data, load data, and build relationship between tables in PostgresSQL– all orchestrated using Airflow

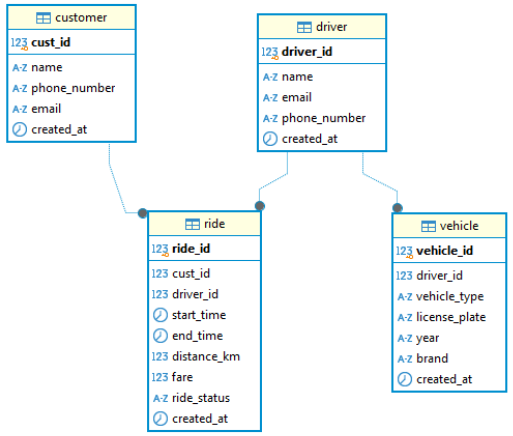


Fig 1. ERD in postgreSQL

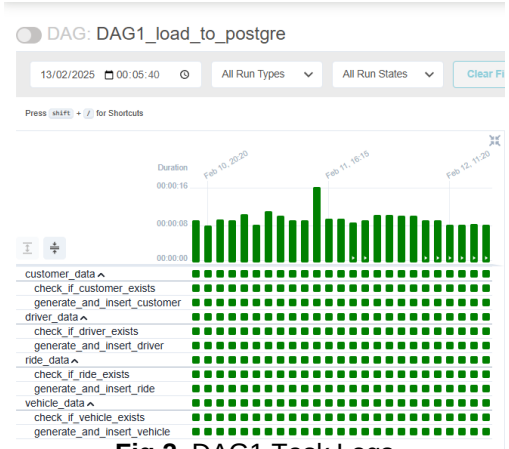


Fig 2. DAG1 Task Logs

- Successfully performed ELT ingesting data from postgre to bigquery incrementally
- Performed data transformation from source table to mart-layer using DBT
- Created STAR SCHEMA in facts tables

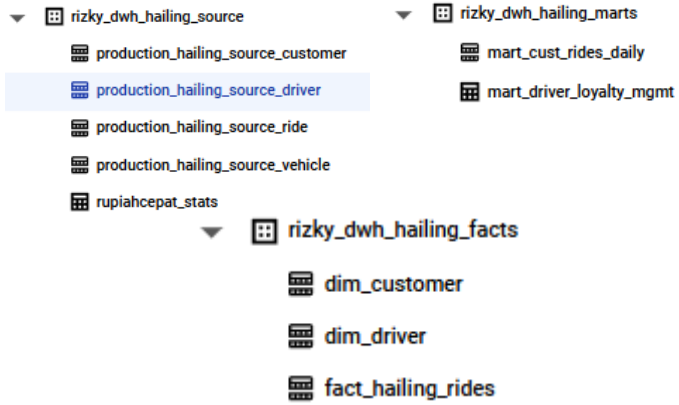


Fig 3. Datasets in BigQuery

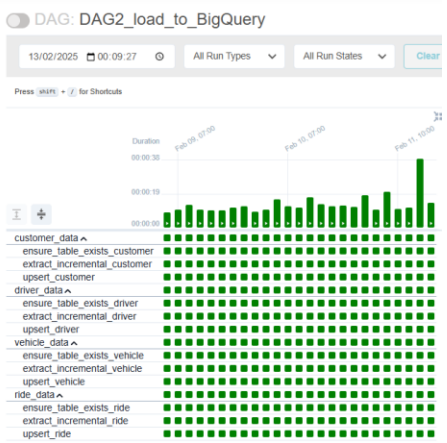


Fig 4. DAG2 Task Logs

Developed a function to send notification through Discord Webhook if a task faile or attempts to retry

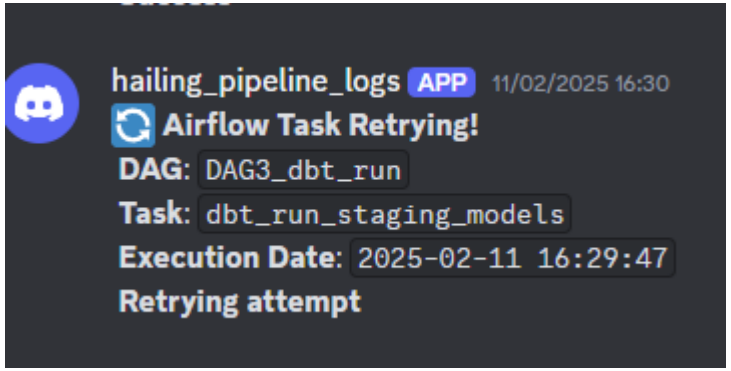


Fig 5. Retry task notification

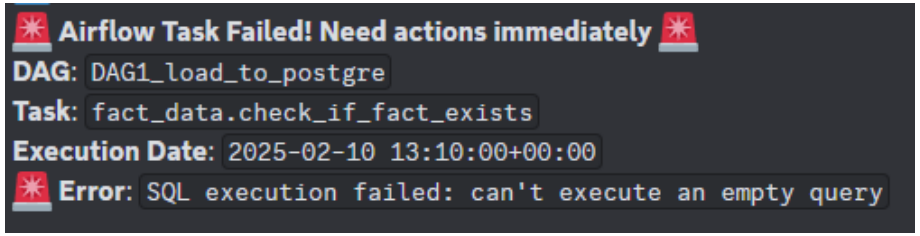
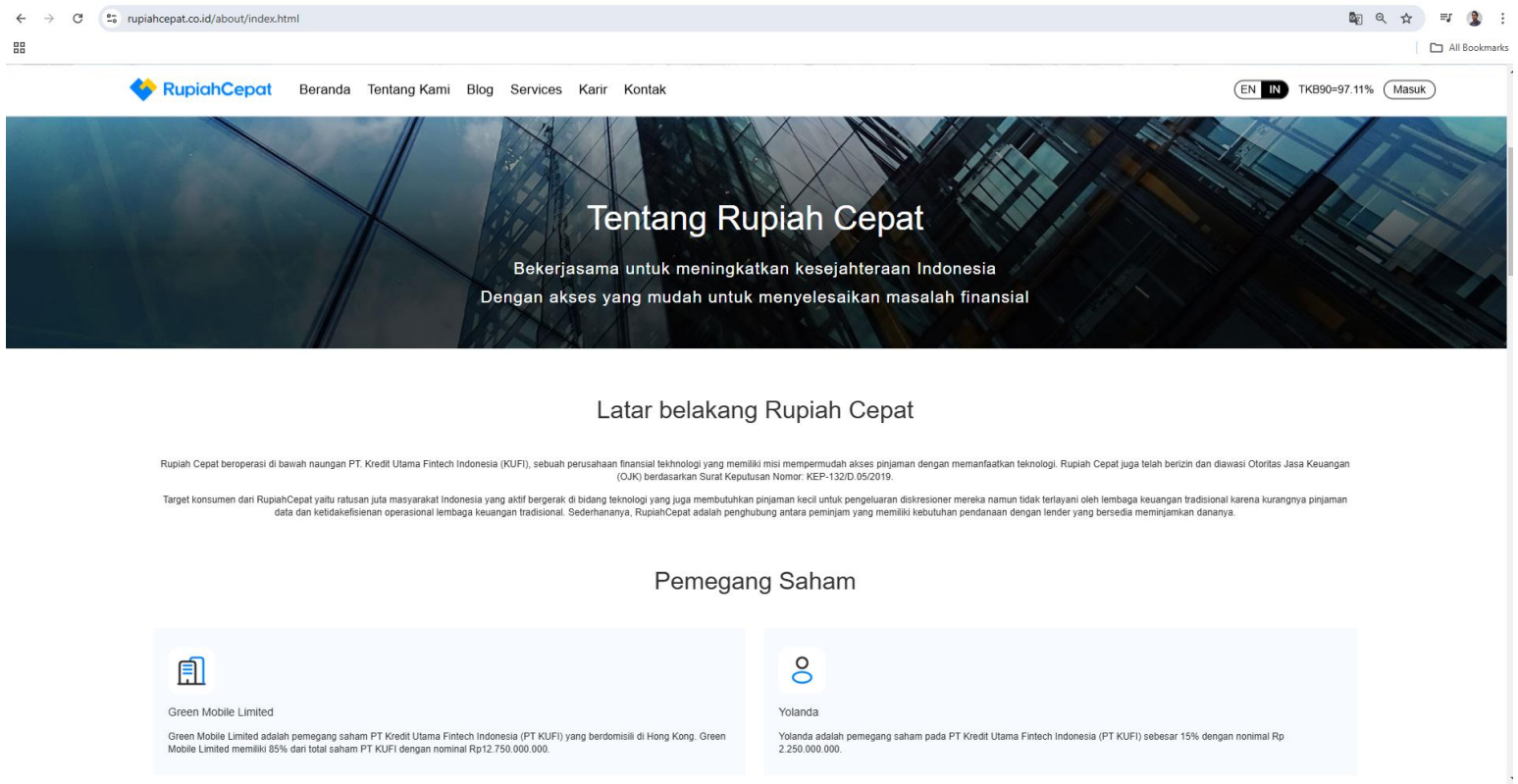


Fig 6. Failed task notification

Final Project - Web Scraping Using Selenium

Github: https://github.com/rizkyjarr/final-project-purwadhika

Study Case: Web Scraping dynamic website of Rupiah Cepat



Project Objective: Perform web scraping from one of P2P lending website (RupiahCepat) and load to Data Warehouse (BigQuery)

Project;s Result:

No	Activity	Code Development	Main Logic/Function
1	**Build Airflow in Docker**	• << click here >>	• Build Airflow using requirements.txt that includes Selenium and WebDriver
2	**Build Selenium**	• << click here >>	• I built the selenium and chrome driver in same docker-compose.yml file as Airflow. This makes it simpler to running the apps
3	**Create DAG4 to automate data scraping and load to bigquery**	• << click here >>	• Check if table exists in BigQuery >> Scrape data using remote server (this allows airflow to command selenium container to scrape the data >> find element (in my case in <div //div[@data-v-35084c97])[1]“ >> map to column/headers >> return df >> load to BigQuery using upsert (any changes within the data will be inserted as new row and set the old data inactive)

Project Results

- Successfully scraped the data from Rupiah cepat
  - Load to BigQuery using merge update
  - Successfully automated using Airflow

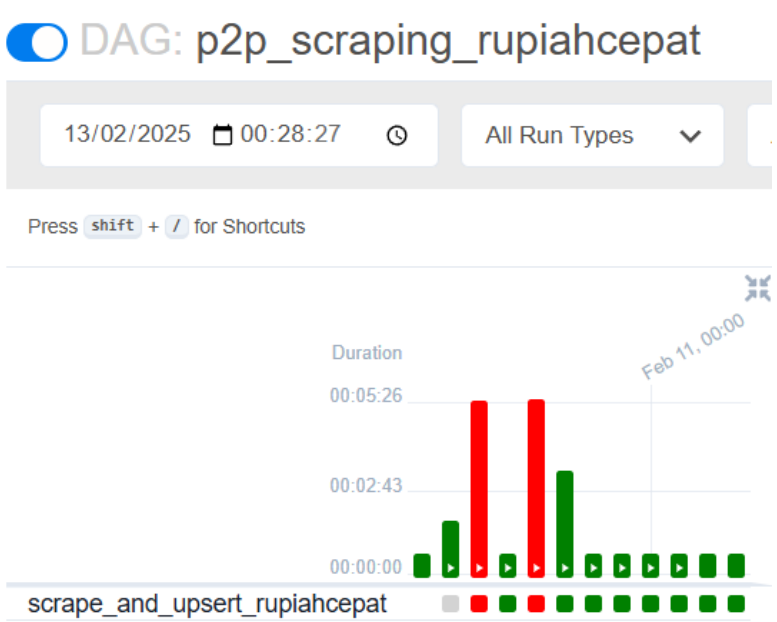


Fig 1. DAG4 Historical Task Logs

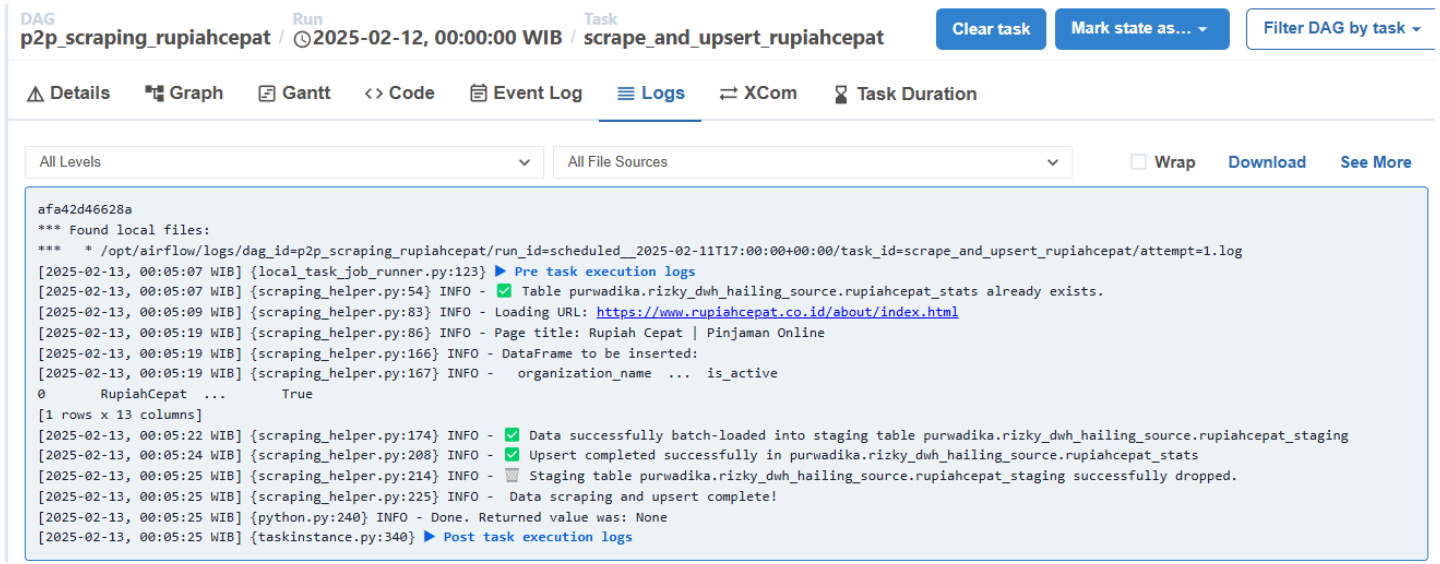


Fig 2. DAG4 Detailed Task Logs

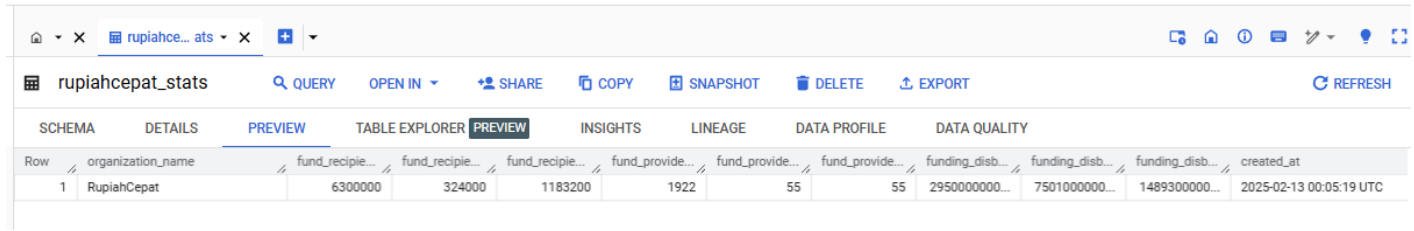
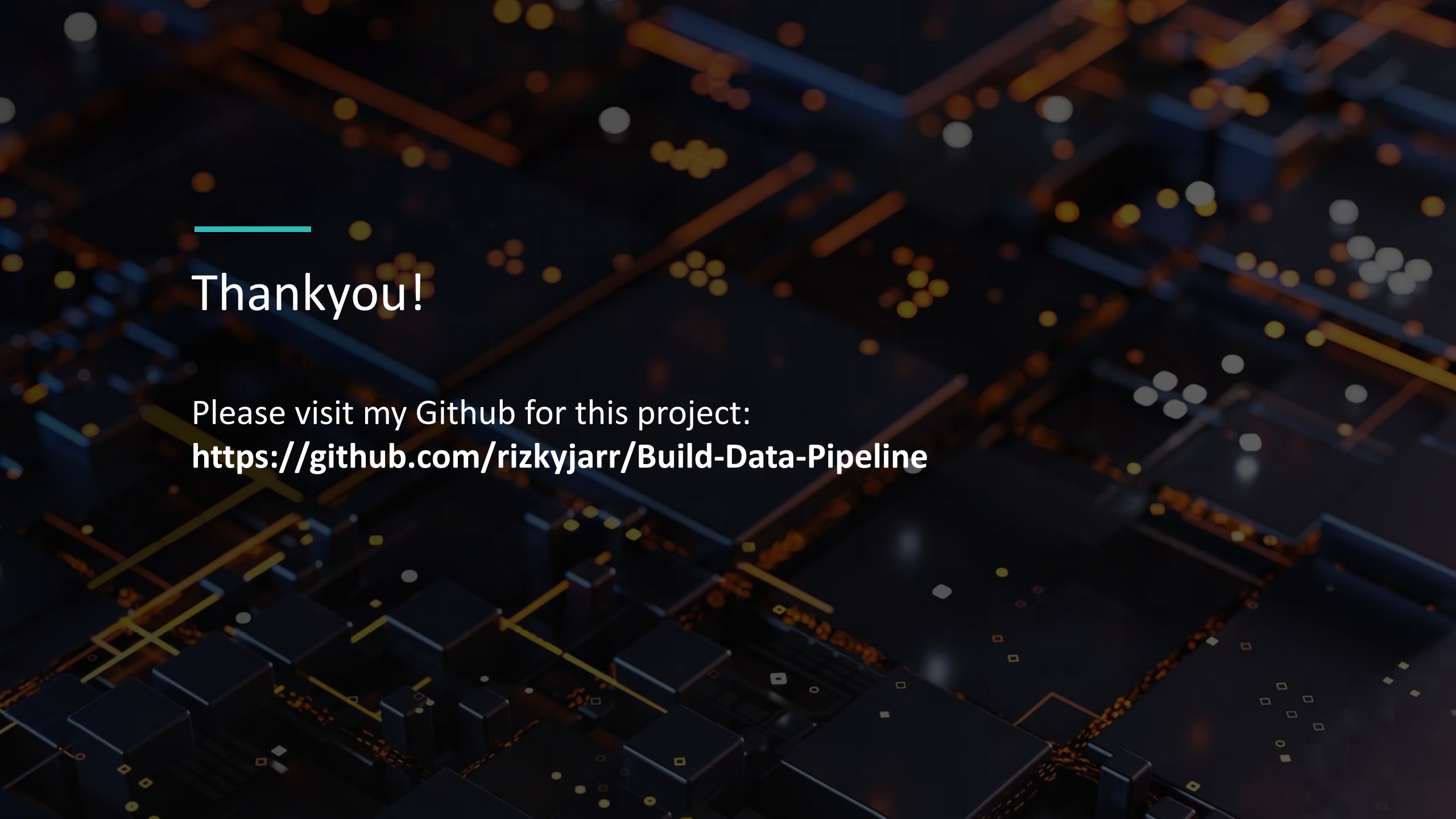


Fig 3. Data scraped and ingested in BigQuery


Thankyou!
Please visit my Github for this project:
https://github.com/rizkyjarr/Build-Data-Pipeline