

---

# Building Batching end-to-end Data Pipeline

**Rizky Fajar Aditya – JCDEOL001**  
**December 2024**

# Table of Contents:

<b>1</b>	<b>Background and Study Case</b>
<b>2</b>	<b>Development Cycle and Process</b>
<b>3</b>	<b>Project Results</b>

# Study Case: B40 Inc. faces growing challenges in Data Access and Quality, hindering decision-making and business agility.

## Background and Study Case:



**B40 Inc** is a major energy company in Indonesia that distributes Biodiesel product to Mining Companies in every corner in Indonesia.

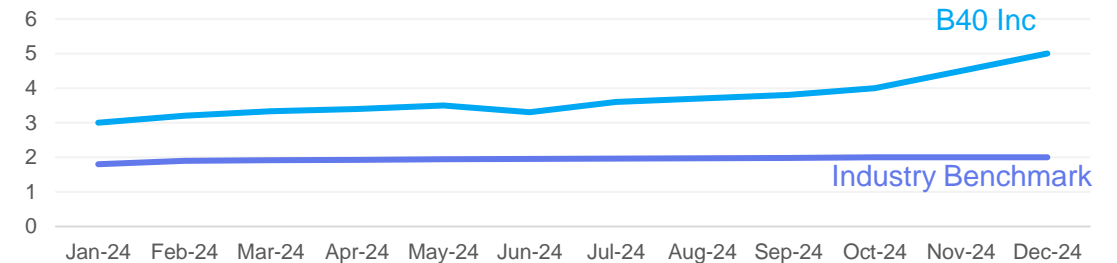
The Marketing and Sales Team at **B40 Inc.** is facing challenges in gathering data for their reports and insights, which is impacting their decision-making processes. To address this, they have enlisted the help of Data Engineering team to **design and implement a robust data pipeline** that will streamline 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 B40 Inc is on the verge to develop a robust data pipeline:

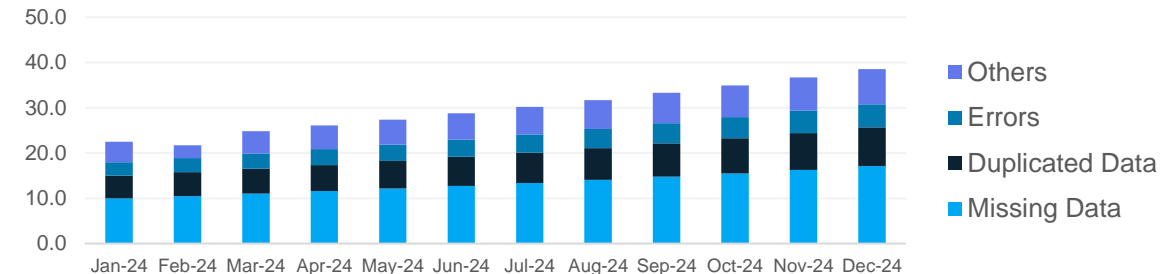
1. The time required to access critical data increases by 4% each month, as the duration from data collection to reporting grows, which delays their ability to respond to market changes

Time needed teed to collect data to reporting, in hours

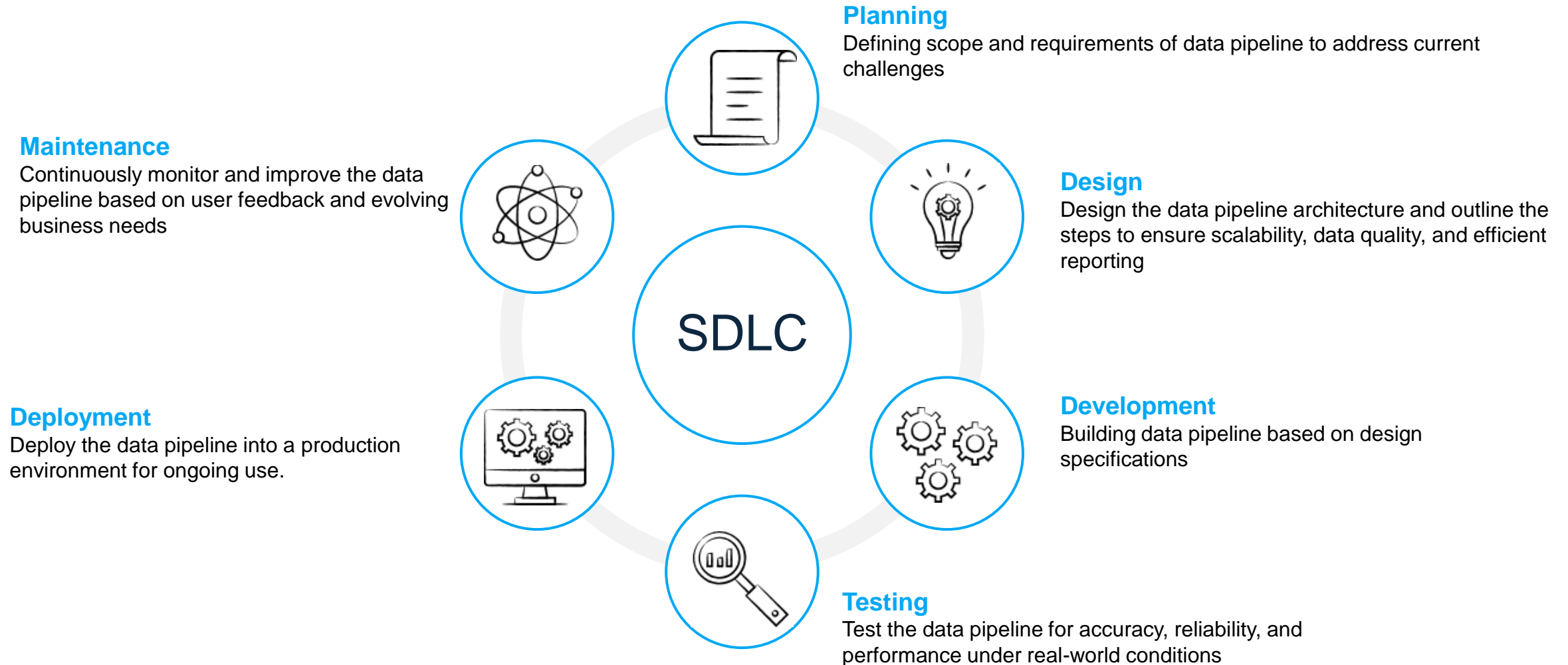


2. Inconsistent data quality makes it challenging to perform comprehensive data analysis

Types of data qualities issues, in #



# To address the issue, The SDLC cycle is applied to develop a robust data pipeline tailored to meet B40 Inc.'s needs.

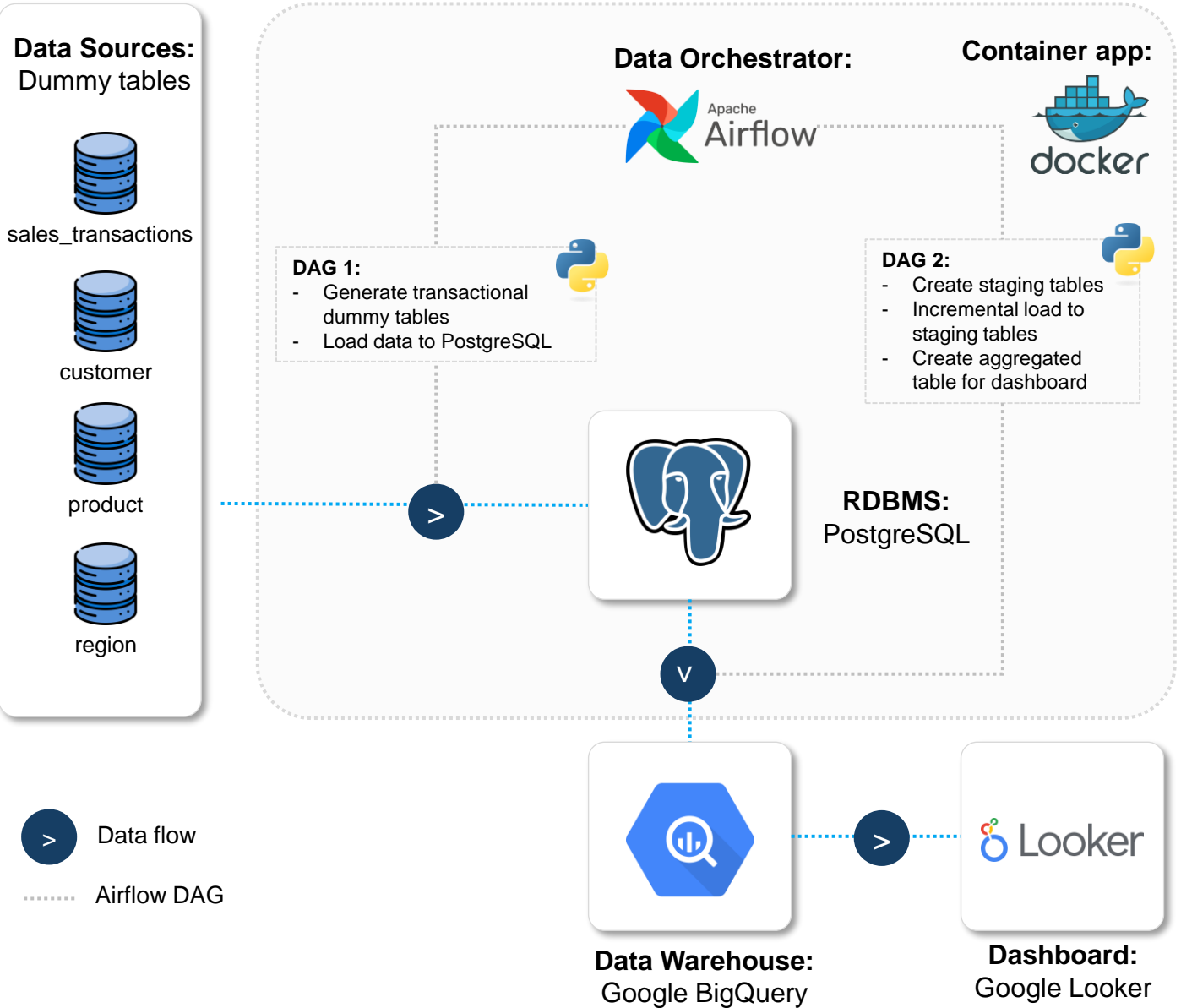


# Stage 1: Planning and gathering design requirements

Defining scope and requirements of data pipeline to address current challenges:

Key Requirement	Description
Key Stakeholders	Marketing and Sales Team
Data Sources / Schema	Generate 4 dummy tables: 1. sales_transactions (fact) 2. product (dim) 3. customers (dim) 4. region (dim)
Transactional data storage	PostgreSQL Server
Data Warehouse	BigQuery
Batching Method	Data will be updated in daily basis at 10 AM
Incremental Load	Load only new or updated data to reduce processing time
Data Granularity	Daily
Reporting	Data and report can be accessed via Dashboard (Google Looker)

# Stage 2: Data Pipeline Architecture Design

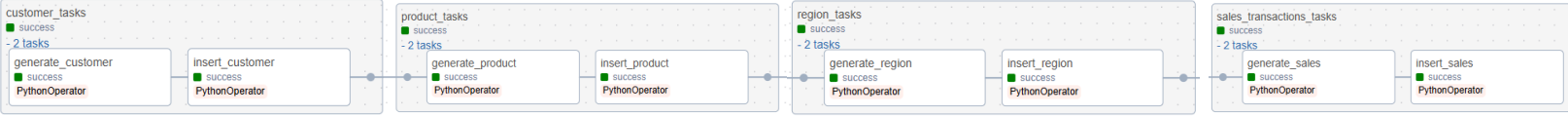
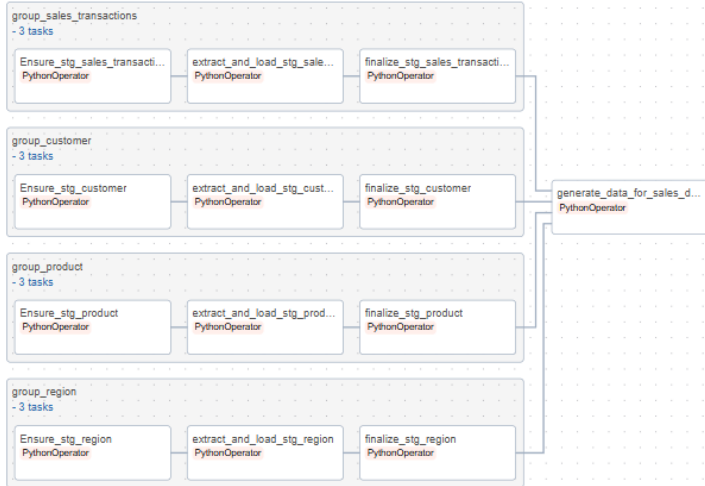


Tools used in this project:

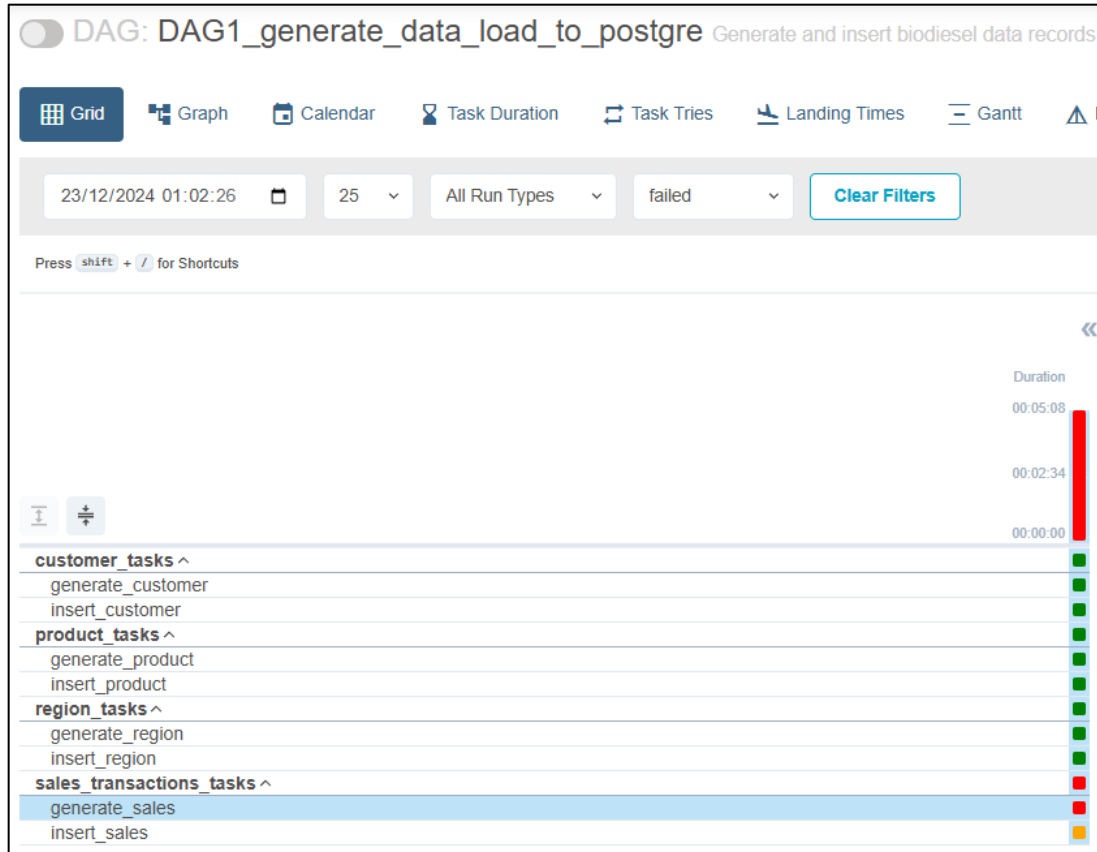
Function	Tools	Explanation
Data Orchestrator	Apache Airflow	<ul style="list-style-type: none"><li>• Able to define task dependency</li><li>• Suitable for batching pipeline</li></ul>
Program Script	Python	<ul style="list-style-type: none"><li>• Versatile programming language for building data pipeline and processing</li></ul>
Containerization	Docker	<ul style="list-style-type: none"><li>• Enables all apps run consistently for development</li></ul>
RDBMS	PostgreSQL	<ul style="list-style-type: none"><li>• The data sources are structured data, suitable for storing data with relationships.</li></ul>
Data Warehouse	Google BigQuery	<ul style="list-style-type: none"><li>• Partitioning enables faster querying, and improving performance</li><li>• Good performance warehouse that's scalable and efficient</li></ul>
Dashboard	Google Looker	<ul style="list-style-type: none"><li>• Seamlessly integrates with Google BigQuery, allowing real-time data updates</li></ul>



# Stage 3: Code Development

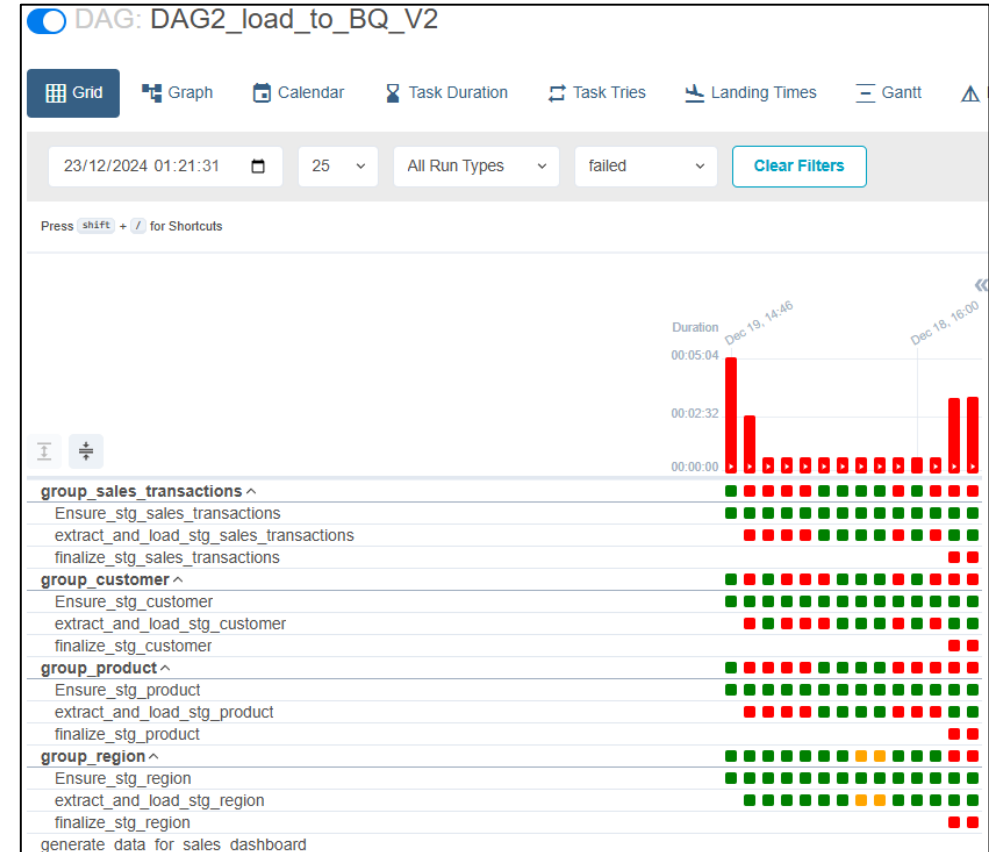
Code	Code Objective	File Name	Logical Flow/Task
DAG 1	<ul style="list-style-type: none"><li>Generate dummy tables</li><li>Load to PostgreSQL</li></ul>	<ul style="list-style-type: none"><li>DAG1_load_to_postgre.py as main DAG</li><li>Generate_dummy.py as helper</li><li>Insert_data.py as helper</li></ul> <div>&lt;&lt; click here &gt;&gt;</div>	
DAG 2	<ul style="list-style-type: none"><li>Ensure staging table</li><li>Fetch data from PostgreSQL and incrementally load to staging tables in BQ</li><li>Create final data using replication</li><li>Create aggregated data for dashboard</li></ul>	<ul style="list-style-type: none"><li>DAG2V2_load_to_BQ.py</li></ul> <div>&lt;&lt; click here &gt;&gt;</div>	
Docker	<ul style="list-style-type: none"><li>Build Airflow</li><li>Build PostgreSQL</li></ul>	<ul style="list-style-type: none"><li>docker-compose.yaml (Airflow)</li><li>docker-compose.yaml (postgresQL)</li></ul> <div>&lt;&lt; click here &gt;&gt;</div>	N/A

# Stage 4: Testing



## DAG1 Testing Log

During testing, **only one** error was recorded. The task failed because the function to connect to PostgreSQL was **not available** in the code. As a result, the code **was updated and successfully worked** at this stage.



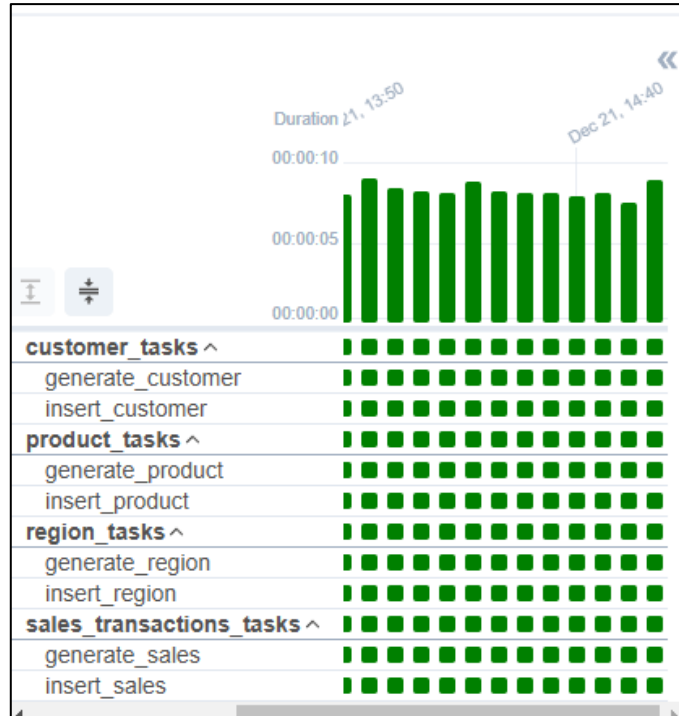
## DAG2 Testing Log

During testing, **13 errors** was recorded. The task failed due to various reasons: failed fetching, failed ensuring staging tables, etc. Thus, every error was thoroughly learned and revised. As a result, the code **was updated and successfully worked** at this stage.



# Stage 5 & 6: Deployment and Maintenance

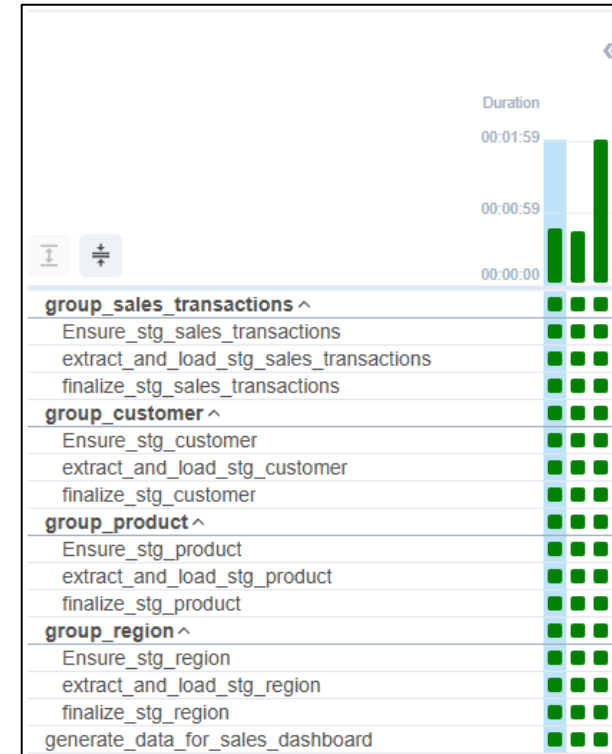
## DAG: DAG1\_generate\_data\_load\_to\_postgre



### DAG1 Testing Log

DAG1 was deployed with a **scheduled run** on December 19, 2024. The task will execute automatically every **5 minutes** to generate data and load to PostgreSQL. The scheduled tasks have been successfully performed and are running smoothly. Monitoring and maintenance will be closely overseen.

## DAG: DAG2\_load\_to\_BQ\_V2

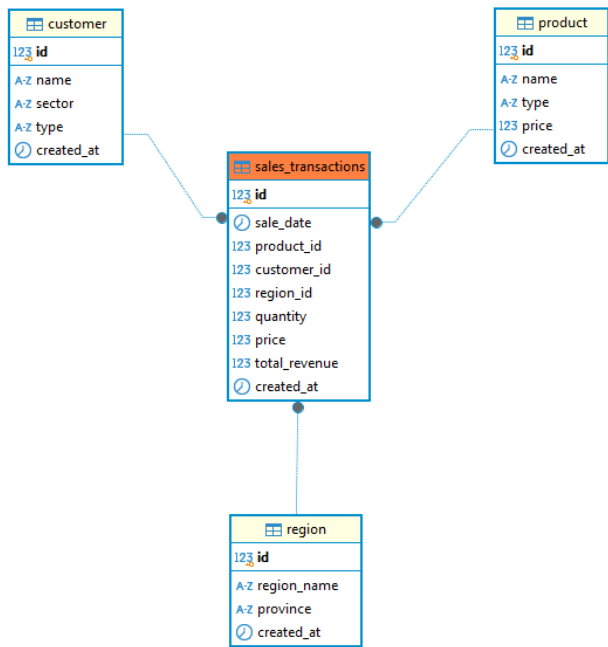


### DAG2 Testing Log

DAG2 were deployed using **scheduled run** in 20 December 2024. The task will run automatically **daily at 10.00 AM** local Jakarta time. Three tasks were scheduled and running well. Monitoring and maintenance will be tightly monitored.

# Project Results

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



Successfully fetched data from Postgre, incrementally load to staging stables in BigQuery, create sales\_dashboard table as source for dashboard

rizky\_biodiesel\_capstone3

customer

product

region

sales\_dashboard

sales\_transactions

stg\_customer

stg\_product

stg\_region

stg\_sales\_transactio...

sales\_dashboard

QUERY

OPEN IN

SCHEMA

DETAILS

PREVIEW

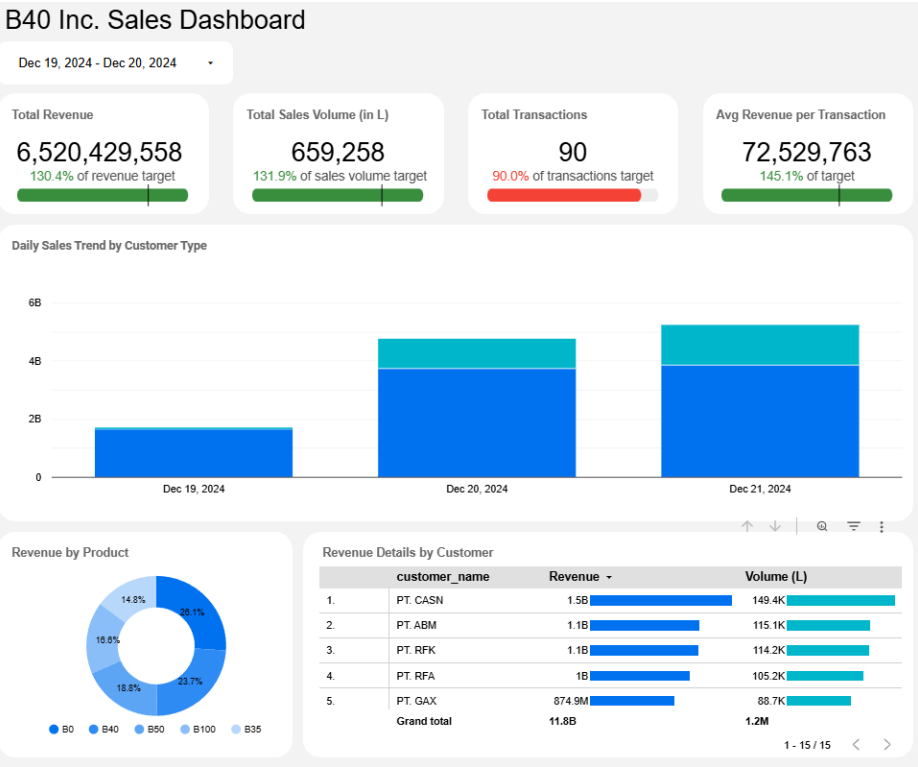
TABLE EXPLOR

Filter

Enter property name or value

	Field name	Type	Mode
<input type="checkbox"/>	sale_date	DATE	NULLABLE
<input type="checkbox"/>	customer_name	STRING	NULLABLE
<input type="checkbox"/>	customer_type	STRING	NULLABLE
<input type="checkbox"/>	region_name	STRING	NULLABLE
<input type="checkbox"/>	product_name	STRING	NULLABLE
<input type="checkbox"/>	transaction_count	INTEGER	NULLABLE
<input type="checkbox"/>	total_quantity	INTEGER	NULLABLE
<input type="checkbox"/>	total_revenue	NUMERIC	NULLABLE

Created dashboard to generate business insights and informations: reducing reporting time through automation, increase data quality, and enhancing decision-making capabilities



<< Dashboard link click here >>



—  
Thankyou!

Please visit my Github for this project:  
[\*\*https://github.com/rizkyjarr/Build-Data-Pipeline\*\*](https://github.com/rizkyjarr/Build-Data-Pipeline)