

# **STEP BY STEP PROJECT 3**



**Disusun Oleh:**

**Spiderweb Team**



# TEAM MEMBER

K E L O M P O K   5   F T D E - 0 1



Hafidha  
Harfiana



Humairoh



Hermawan



I Gusti Agung  
Krishna



I Putu Ferry  
Wstika

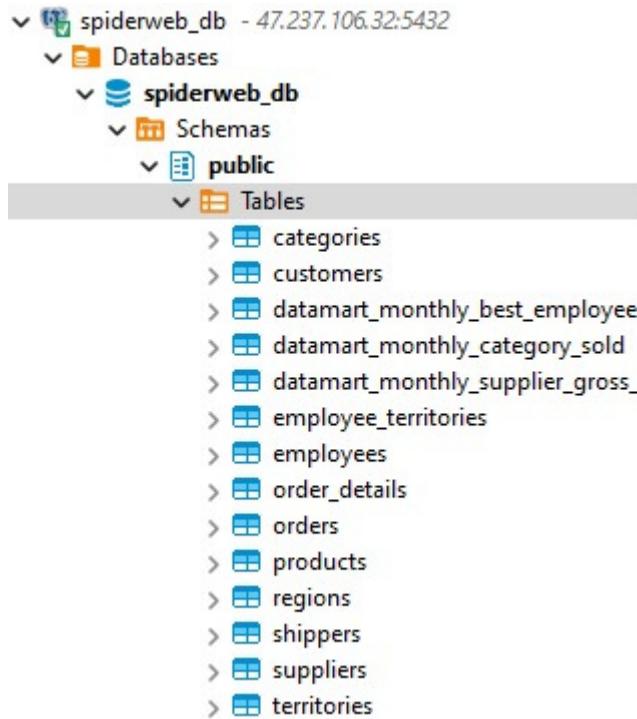


Fenny  
Inriana

FTDE-01 DigitalSkola



1. database sudah tersedia di postgre dengan rincian :



```
spiderweb_db - 47.237.106.32:5432
  Databases
    spiderweb_db
      Schemas
        public
          Tables
            categories
            customers
            datamart_monthly_best_employee
            datamart_monthly_category_sold
            datamart_monthly_supplier_gross_
            employee_territories
            employees
            order_details
            orders
            products
            regions
            shippers
            suppliers
            territories
```

2. Menginstal lib yang diperlukan menggunakan !pip di google colab

```
!pip install psycopg2-binary pandas-gbq google-cloud-bigquery
```

3. Lalu kami memasukkan informasi koneksi ke PostgreSQL ( password dirahasikan untuk membuat dokumen ini )

```
import psycopg2
import pandas as pd

# Informasi koneksi
conn = psycopg2.connect(
    host="47.237.106.32",
    database="spiderweb_db",
    user="spiderweb_p3",
    password=""

)

# Membuat cursor
cursor = conn.cursor()

# Mendapatkan nama-nama tabel
cursor.execute("""SELECT table_name FROM information_schema.tables
                 WHERE table_schema = 'public'""")
tables = cursor.fetchall()

# Tampilkan daftar tabel
for table in tables:
    print(table[0])
```

## 4. Lalu membaca setiap tabel

```
▶ def read_table_to_df(table_name):
    query = f"SELECT * FROM {table_name}"
    df = pd.read_sql_query(query, conn)
    return df

# Contoh membaca tabel pertama
table_name = tables[0][0] # nama tabel pertama
df = read_table_to_df(table_name)
print(df.head())
```

→ <ipython-input-12-e65708bb4e71>:3: UserWarning: pandas only supports SQLAlchemy connection
df = pd.read\_sql\_query(query, conn)

	customerid	companyname	contactname	\
0	ALFKI	Alfreds Futterkiste	Maria Anders	
1	ANATR	Aña Trujillo Emparedados y helados	Ana Trujillo	
2	ANTON	Antonio Moreno Taquería	Antonio Moreno	
3	AROUT	Around the Horn	Thomas Hardy	
4	BERGS	Berglunds snabbköp	Christina Berglund	

	contacttitle	address	city	region	\
0	Sales Representative	Obere Str. 57	Berlin	NULL	
1	Owner	Avda. de la Constitución 2222	México D.F.	NULL	
2	Owner	Mataderos 2312	México D.F.	NULL	
3	Sales Representative	120 Hanover Sq.	London	NULL	
4	Order Administrator	Berguvsvägen 8	Luleå	NULL	

	postalcode	country	phone	fax	
0	12209	Germany	030-0074321	030-0076545	
1	05021	Mexico	(5) 555-4729	(5) 555-3745	
2	05023	Mexico	(5) 555-3932	NULL	
3	WA1 1DP	UK	(171) 555-7788	(171) 555-6750	
4	S-958 22	Sweden	0921-12 34 65	0921-12 34 67	

## 5. Lalu mengunggah dataframe ke bigquery dengan ‘google-cloud-bigquery’

```

from google.cloud import bigquery

def upload_to_bigquery(df, table_id):
    job_config = bigquery.LoadJobConfig(
        write_disposition=bigquery.WriteDisposition.WRITE_TRUNCATE,
    )
    job = client.load_table_from_dataframe(df, table_id, job_config=job_config)
    job.result() # Tunggu hingga pekerjaan selesai
    print(f"Data berhasil diunggah ke {table_id}!")

# Nama project dan dataset di BigQuery
project_id = 'spiderweb3'
dataset_id = 'spiderweb_p3'

# Loop untuk mengunggah setiap tabel ke BigQuery
for table in tables:
    table_name = table[0]
    print(f"Mengunggah tabel {table_name}...")
    df = read_table_to_df(table_name)
    table_id = f'{project_id}.{dataset_id}.{table_name}'
    upload_to_bigquery(df, table_id)

→ Mengunggah tabel customers...
<ipython-input-12-e65708bb4e71>:3: UserWarning: pandas only supports SQLAlchemy conn
  df = pd.read_sql_query(query, conn)
Data berhasil diunggah ke spiderweb3.spiderweb_p3.customers!
Mengunggah tabel employee_territories...
<ipython-input-12-e65708bb4e71>:3: UserWarning: pandas only supports SQLAlchemy conn
  df = pd.read_sql_query(query, conn)
Data berhasil diunggah ke spiderweb3.spiderweb_p3.employee_territories!
Mengunggah tabel employees...
<ipython-input-12-e65708bb4e71>:3: UserWarning: pandas only supports SQLAlchemy conn
  df = pd.read_sql_query(query, conn)

```

6. Kode sukses dijalankan dan data sudah tampil di bigquery. Kini kami sudah bisa melihat schema, details, dan preview dari masing-masing tabel.

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
categoryid	INTEGER	NULLABLE	-	-	-	-	-
categoryname	STRING	NULLABLE	-	-	-	-	-
description	STRING	NULLABLE	-	-	-	-	-
picture	BYTES	NULLABLE	-	-	-	-	-

## 7. Pada git bash, saya membuat direktori baru dan pindah ke direktori tersebut. direktori ini bernama “airflow\_docker”.

```

user@DESKTOP-39Q6M5E MINGW64 ~ (master)
$ cd airflow_docker

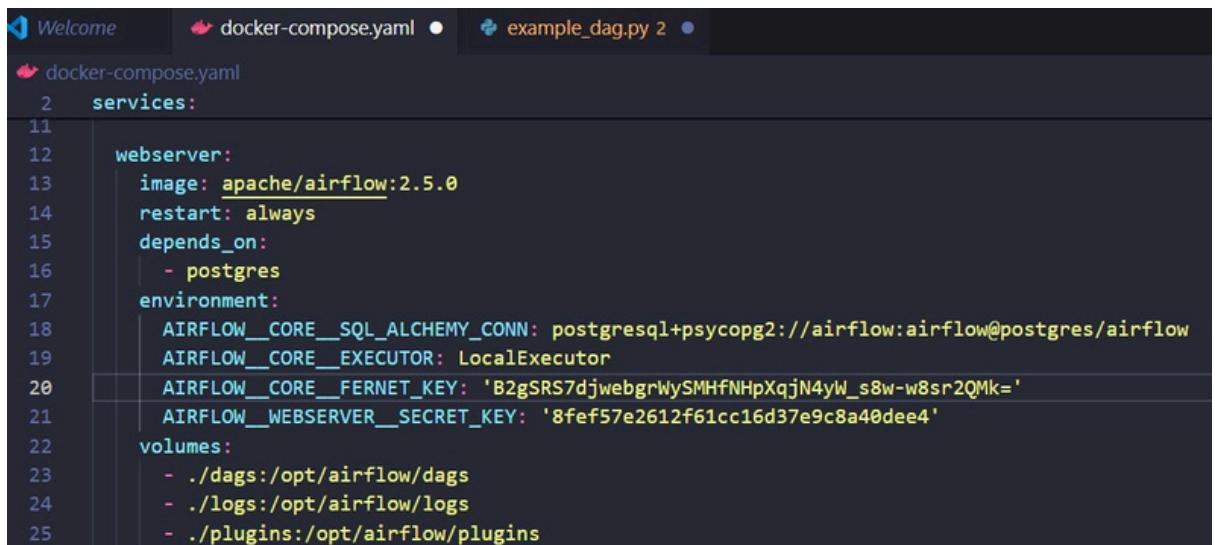
user@DESKTOP-39Q6M5E MINGW64 ~/airflow_docker (master)
$
```

8. Kami membuka VSCode dan memastikan bahwa folder direktori sudah terbuka untuk membuat file docker-compose.yaml, namun sebelum itu saya mendefinisikan fernet key dan webserver secret key di git bash

```
user@DESKTOP-39Q6M5E MINGW64 ~/airflow_docker (master)
$ python -c "from cryptography.fernet import Fernet; print(Fernet(b'').decode())"
B2gSRS7djwebgrWySMHfNHpXqjN4yw_s8w-w8sr2QMk=

user@DESKTOP-39Q6M5E MINGW64 ~/airflow_docker (master)
$ python -c "import os; print(os.urandom(16).hex())"
8fef57e2612f61cc16d37e9c8a40dee4
```

9. Kami memasukkan unci fernet dan secret key webserver tersebut ke file docker-compose.yaml



```
Welcome docker-compose.yaml example_dag.py 2

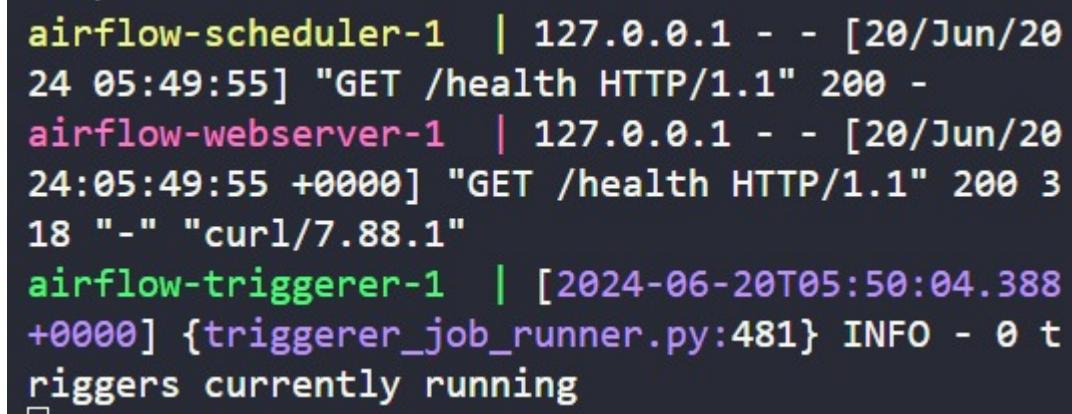
docker-compose.yaml
  2   services:
 11
 12     webserver:
 13       image: apache/airflow:2.5.0
 14       restart: always
 15       depends_on:
 16         - postgres
 17       environment:
 18         AIRFLOW__CORE__SQLALCHEMY_CONN: postgresql+psycopg2://airflow:airflow@postgres/airflow
 19         AIRFLOW__CORE__EXECUTOR: LocalExecutor
 20         AIRFLOW__CORE__FERNET_KEY: 'B2gSRS7djwebgrWySMHfNHpXqjN4yw_s8w-w8sr2QMk='
 21         AIRFLOW__WEBSERVER__SECRET_KEY: '8fef57e2612f61cc16d37e9c8a40dee4'
 22       volumes:
 23         - ./dags:/opt/airflow/dags
 24         - ./logs:/opt/airflow/logs
 25         - ./plugins:/opt/airflow/plugins
```

10. Kami membuat file dag di dalam direktori dan memasukkan file example\_dag.py



```
dags
  example_dag.py 2
```

11. Kami menjalankan airflow



```
airflow-scheduler-1 | 127.0.0.1 - - [20/Jun/2024 05:49:55] "GET /health HTTP/1.1" 200 -
airflow-webserver-1 | 127.0.0.1 - - [20/Jun/2024:05:49:55 +0000] "GET /health HTTP/1.1" 200 3
18 "-" "curl/7.88.1"
airflow-triggerer-1 | [2024-06-20T05:50:04.388+0000] {triggerer_job_runner.py:481} INFO - 0 triggers currently running
```

12. Kami mengecek airflow apakah running healthy

PS C:\Users\user\airflow_docker> docker ps						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
1a85964ce0b5	apache/airflow:2.9.2	"/usr/bin/dumb-init ..."	4 minutes ago	Up 4 minutes (healthy)	8080/tcp	airfl
ow_docker-airflow-triggerer-1						
aced74419773	apache/airflow:2.9.2	"/usr/bin/dumb-init ..."	4 minutes ago	Up 4 minutes (healthy)	8080/tcp	airfl
ow_docker-airflow-scheduler-1						
29c9ba69a301	apache/airflow:2.9.2	"/usr/bin/dumb-init ..."	4 minutes ago	Up 4 minutes (healthy)	0.0.0.0:8080->8080/tcp	airfl
ow_docker-airflow-webserver-1						
f84b87015107	apache/airflow:2.9.2	"/usr/bin/dumb-init ..."	4 minutes ago	Up 4 minutes (healthy)	8080/tcp	airfl
ow_docker-airflow-worker-1						
b0632a3c933a	redis:7.2-bookworm	"docker-entrypoint.s..."	6 minutes ago	Up 6 minutes (healthy)	6379/tcp	airfl
ow_docker-redis-1						
9d1993a2906a	postgres:13	"docker-entrypoint.s..."	6 minutes ago	Up 6 minutes (healthy)	5432/tcp	airfl
ow_docker-postgres-1						

13.Kami saya akses airflow WEB UI

The screenshot shows the Airflow interface for managing Data Pipelines (DAGs). At the top, there are navigation links for Airflow, Dags, Cluster Activity, Datasets, Security, Browse, Admin, and Docs. The timestamp 05:52 UTC is displayed on the right, along with a blue circular icon with a white 'AA' symbol. Below the header, the title 'DAGs' is centered. A toolbar below the title includes buttons for All (selected), Active (1), Paused (6), Running (1), Failed (1), a Filter DAGs by tag input field, a Search DAGs input field, an Auto-refresh checkbox, and a refresh icon. The main content area displays a table of DAGs. Each row represents a DAG with the following columns: DAG name, Owner (airflow), Runs (5), Schedule (Dataset or 0 \* \* \* 1), Last Run (2024-06-19, 01:00:00), Next Run (2024-06-19, 01:00:00), Recent Tasks (empty), Actions (button with a play icon), and Links (three dots). The DAG names listed are: conditional\_dataset\_and\_time\_based\_timetable, consume\_1\_and\_2\_with\_dataset\_expressions, consume\_1\_or\_2\_with\_dataset\_expressions, consume\_1\_or\_both\_2\_and\_3\_with\_dataset\_expressions, dataset\_consumes\_1, and dataset\_consumes\_1\_and\_2.

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
conditional_dataset_and_time_based_timetable dataset time-based timetable	airflow	5	Dataset or 0 * * * 1	2024-06-19, 01:00:00	2024-06-19, 01:00:00			
consume_1_and_2_with_dataset_expressions	airflow	5	Dataset		0 of 2 datasets updated			
consume_1_or_2_with_dataset_expressions	airflow	5	Dataset		0 of 2 datasets updated			
consume_1_or_both_2_and_3_with_dataset_expressions	airflow	5	Dataset		0 of 3 datasets updated			
dataset_consumes_1 consumes dataset scheduled	airflow	5	Dataset		On s3://dag1/output_1.txt			
dataset_consumes_1_and_2 consumes dataset scheduled	airflow	5	Dataset		0 of 2 datasets updated			

14. lalu kami mengaktifkan example\_dag dan sudah memastikan task hello\_task berjalan dengan sukses

DAG: example\_dag

Schedule: @daily | Next Run ID: 2023-01-16, 00:00:00 UTC | Auto-refresh | 25

All Run Types | All Run States | Clear Filters

Press ⌘H + / for Shortcuts

Duration

Jan 14, 2024

Jan 15, 2024

hello\_task

« » DAG example\_dag

Details | Graph | Gantt | Code | Audit Log | Run Duration | Calendar

DAG Runs Summary

Total Runs Displayed	16
Total success	16
First Run Start	2024-06-20, 06:50:03 UTC
Last Run Start	2024-06-20, 06:50:07 UTC

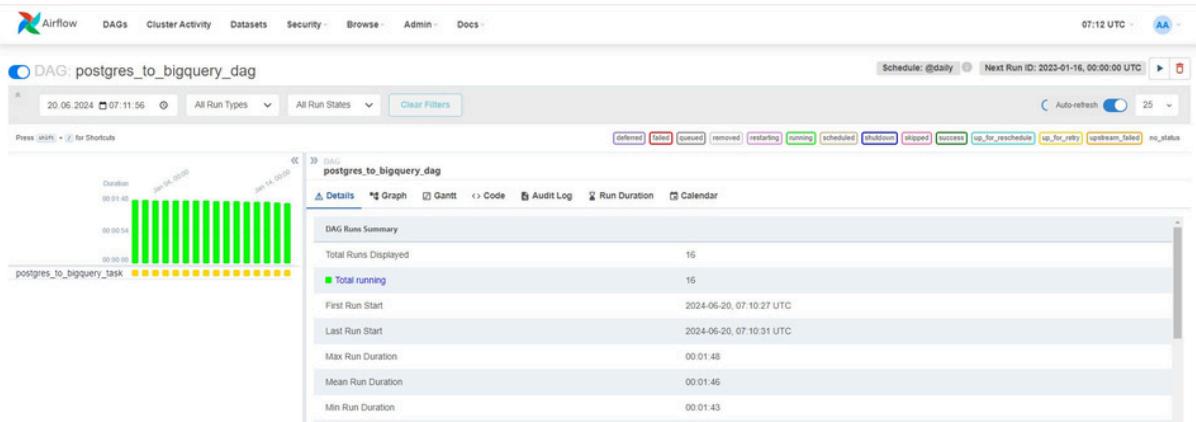
Refined Run States: [refined, failed, queued, removed, restarting, running, scheduled, shutdown, skipped, success, up\_for\_reschedule, up\_for\_retry, upstream\_failed, no\_status]

15.lalu kami akan membuat DAG baru untuk mengotomatiskan proses ekstraksi data dari PostgreSQL dan mengunggahnya ke BigQuery. Misalnya, buat file `postgres_to_bigquery_dag.py` di direktori `dags`

```
from airflow import DAG
from airflow.operators.python import PythonOperator
from datetime import datetime
import pandas as pd
import psycopg2
from google.cloud import bigquery
from google.oauth2 import service_account

def extract_postgres_to_df(table_name):
    conn = psycopg2.connect(
        host="47.237.106.32",
        database="spiderweb_db",
        user="spiderweb_p3",
        password="spiderweb_p3"
    )
    query = f"SELECT * FROM {table_name}"
    df = pd.read_sql_query(query, conn)
    return df
```

16.refresh airflow. lalu mengaktifkan postgres\_to\_biqury\_dag dan sudah memastikan task task berjalan dengan sukses



namun kami masih harus mengedit task untuk menampilkan semua tabel  
17.Kami menggunakan kode python sebagai berikut :

```
# C:/Users/user/Downloads/spiderweb3-7a16d1d8cbf2.json

from airflow import DAG
from airflow.operators.python import PythonOperator
from datetime import datetime
import pandas as pd
import psycopg2
from google.cloud import bigquery
from google.oauth2 import service_account

# Definisi fungsi extract_postgres_to_df dan load_df_to_bigquery seperti sebelumnya

def postgres_to_bigquery_task(table_name, table_id, host, database, user, password, key_path):
    try:
        df = extract_postgres_to_df(host, database, user, password, table_name)
        load_df_to_bigquery(df, table_id, key_path)
        print(f"Task for table {table_name} completed successfully.")
    except Exception as e:
        print(f"Error processing table {table_name}: {str(e)}")
        raise e

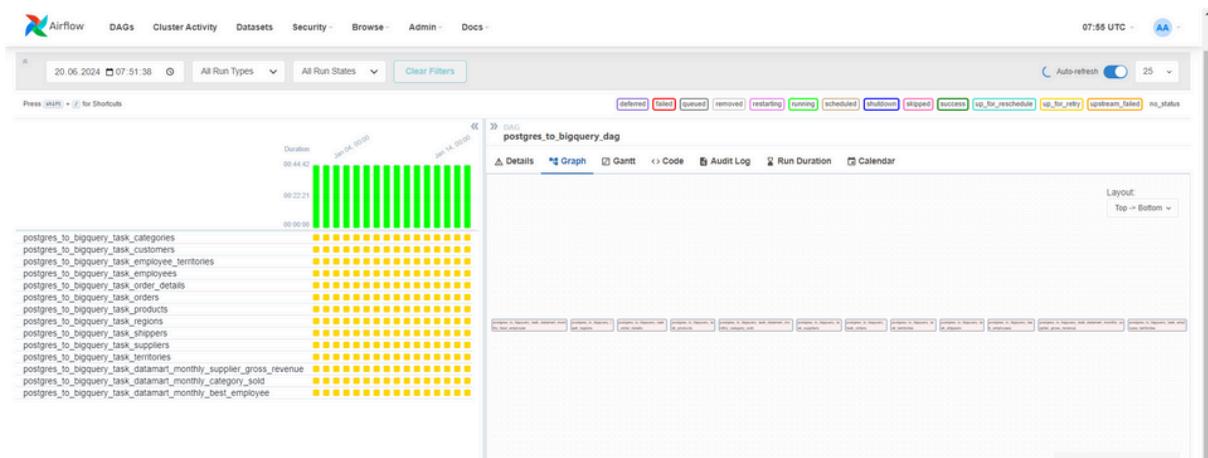
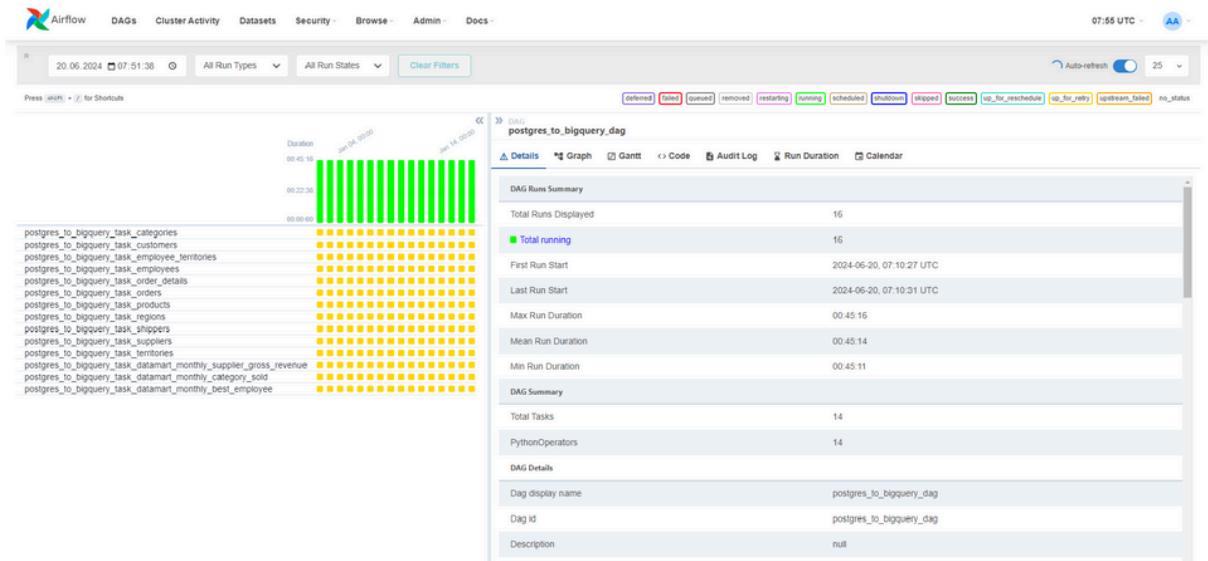
# Definisi default arguments untuk DAG
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2023, 1, 1),
    'retries': 1,
}

# Buat DAG
with DAG(
    'postgres_to_bigquery_dag',
    default_args=default_args,
    schedule_interval='@daily',
) as dag:

    # Daftar tabel-tabel yang ingin diekstrak dan dimuat ke BigQuery
    tables_to_process = [
        {'table_name': 'categories', 'table_id': 'spiderweb3.spiderweb_p3.categories'},
        {'table_name': 'customers', 'table_id': 'spiderweb3.spiderweb_p3.customers'},
        {'table_name': 'employee_territories', 'table_id': 'spiderweb3.spiderweb_p3.employee_territories'},
        {'table_name': 'employees', 'table_id': 'spiderweb3.spiderweb_p3.employees'},
        {'table_name': 'order_details', 'table_id': 'spiderweb3.spiderweb_p3.order_details'},
        {'table_name': 'orders', 'table_id': 'spiderweb3.spiderweb_p3.orders'},
        {'table_name': 'products', 'table_id': 'spiderweb3.spiderweb_p3.products'},
        {'table_name': 'regions', 'table_id': 'spiderweb3.spiderweb_p3.regions'},
        {'table_name': 'shippers', 'table_id': 'spiderweb3.spiderweb_p3.shippers'},
        {'table_name': 'suppliers', 'table_id': 'spiderweb3.spiderweb_p3.suppliers'},
        {'table_name': 'territories', 'table_id': 'spiderweb3.spiderweb_p3.territories'},
        {'table_name': 'datamart_monthly_supplier_gross_revenue', 'table_id': 'spiderweb3.spiderweb_p3.datamart_monthly_supplier_gross_revenue'},
        {'table_name': 'datamart_monthly_category_sold', 'table_id': 'spiderweb3.spiderweb_p3.datamart_monthly_category_sold'},
        {'table_name': 'datamart_monthly_best_employee', 'table_id': 'spiderweb3.spiderweb_p3.datamart_monthly_best_employee'}
        # Tambahkan tabel-tabel lainnya sesuai kebutuhan
    ]

    # Definisi task untuk setiap tabel
    for table_info in tables_to_process:
        task_id = f'postgres_to_bigquery_task_{table_info["table_name"]}'
        PythonOperator(
            task_id=task_id,
            python_callable=postgres_to_bigquery_task,
            op_args=[table_info['table_name'], table_info['table_id']],
            '47.237.106.32', 'spiderweb_db', 'spiderweb_p3', 'spiderweb_p3',
            'C:/Users/user/Downloads/spiderweb3-7a16d1d8cbf2.json', # Ganti dengan lokasi kunci Anda
        )
```

18.lalu saya jalankan di airflow dan mendapat hasil :



DAG  
postgres to bigquery dag

Details Graph Gantt Code Audit Log Run Duration Calendar

Parsed at: 2024-06-20, 07:51:26 UTC

```

1 # C:/Users/user/Downloads/spiderweb3-7a16d1d8cbf2.json
2
3 from airflow import DAG
4 from airflow.operators.python import PythonOperator
5 from datetime import datetime
6 import pandas as pd
7 import psycopg2
8 from google.cloud import bigquery
9 from google.oauth2 import service_account
10
11 # Definisi fungsi extract_postgres_to_df dan load_df_to_bigquery seperti sebelumnya
12
13 def postgres_to_bigquery_task(table_name, table_id, host, database, user, password, key_path):
14     try:
15         df = extract_postgres_to_df(host, database, user, password, table_name)
16         load_df_to_bigquery(df, table_id, key_path)
17         print(f"Task for table {table_name} completed successfully.")
18     except Exception as e:
19         print(f"Error processing table {table_name}: {str(e)}")
20         raise e
21
22 # Definisi default arguments untuk DAG
23 default_args = {
24     'owner': 'airflow',
25     'depends_on_past': False,
26     'start_date': datetime(2023, 1, 1),
27     'retries': 1,
28 }
29
30 # Buat DAG
31 with DAG(
32     'postgres_to_bigquery_dag',
33     default_args=default_args,
34     schedule_interval='@daily',
35 ) as dag:
36
37     # Daftar tabel-tabel yang ingin diekstrak dan dimuat ke BigQuery
38     tables_to_process = [
39         ('table_name': 'categories', 'table_id': 'spiderweb3.spiderweb_p3.categories'),
40         ('table_name': 'customers', 'table_id': 'spiderweb3.spiderweb_p3.customers'),
41         ('table_name': 'employee_territories', 'table_id': 'spiderweb3.spiderweb_p3.employee_territories'),
42         ('table_name': 'employees', 'table_id': 'spiderweb3.spiderweb_p3.employees'),
43         ('table_name': 'order_details', 'table_id': 'spiderweb3.spiderweb_p3.order_details'),
44         ('table_name': 'orders', 'table_id': 'spiderweb3.spiderweb_p3.orders'),
45         ('table_name': 'products', 'table_id': 'spiderweb3.spiderweb_p3.products'),
46         ('table_name': 'regions', 'table_id': 'spiderweb3.spiderweb_p3.regions'),
47         ('table_name': 'shippers', 'table_id': 'spiderweb3.spiderweb_p3.shippers'),
48         ('table_name': 'suppliers', 'table_id': 'spiderweb3.spiderweb_p3.suppliers'),
49         ('table_name': 'territories', 'table_id': 'spiderweb3.spiderweb_p3.territories'),
50         ('table_name': 'datamart_monthly_supplier_gross_revenue', 'table_id': 'spiderweb3.spiderweb_p3.datamart_monthly_supplier_gross_revenue'),
51         ('table_name': 'datamart_monthly_category_sold', 'table_id': 'spiderweb3.spiderweb_p3.datamart_monthly_category_sold'),
52         ('table_name': 'datamart_monthly_best_employee', 'table_id': 'spiderweb3.spiderweb_p3.datamart_monthly_best_employee')
53     # Tambahkan tabel-tabel lainnya sesuai kebutuhan
54     ]
55
56     # Definisi task untuk setup tabel
57     for table_info in tables_to_process:
58         task_id = f'postgres_to_bigquery_task_{table_info["table_name"]}'
59         PythonOperator(
60             task_id=task_id,
61             python_callable=postgres_to_bigquery_task,
62             op_args=[table_info['table_name'], table_info['table_id']],
63             '47.237.106.32', 'spiderweb_db', 'spiderweb_p3', 'spiderweb_p3',
64             'C:/Users/user/Downloads/spiderweb3-7a16d1d8cbf2.json', # Ganti dengan Lokasi kunci Anda
65         )

```