FINAL PROJECT DATA ENGINEER
BUILDING DOCKERIZE ETL PIPELINE USING AIRFLOW

NAME: HENDRI ANGKASA

ETL Architecture Diagram and Integration Design Diagram

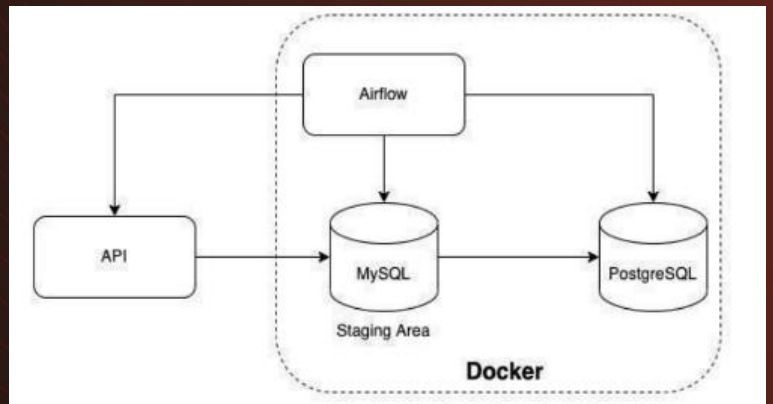


Figure 1. ETL Architecture Diagram

- Get covid19 data from API
- Load data to MySQL as data lake
- Transform and load data to PostgreSQL as data warehouse
- ETL processes were done using Airflow
- Airflow, MySQL and PostgreSQL were run in Docker

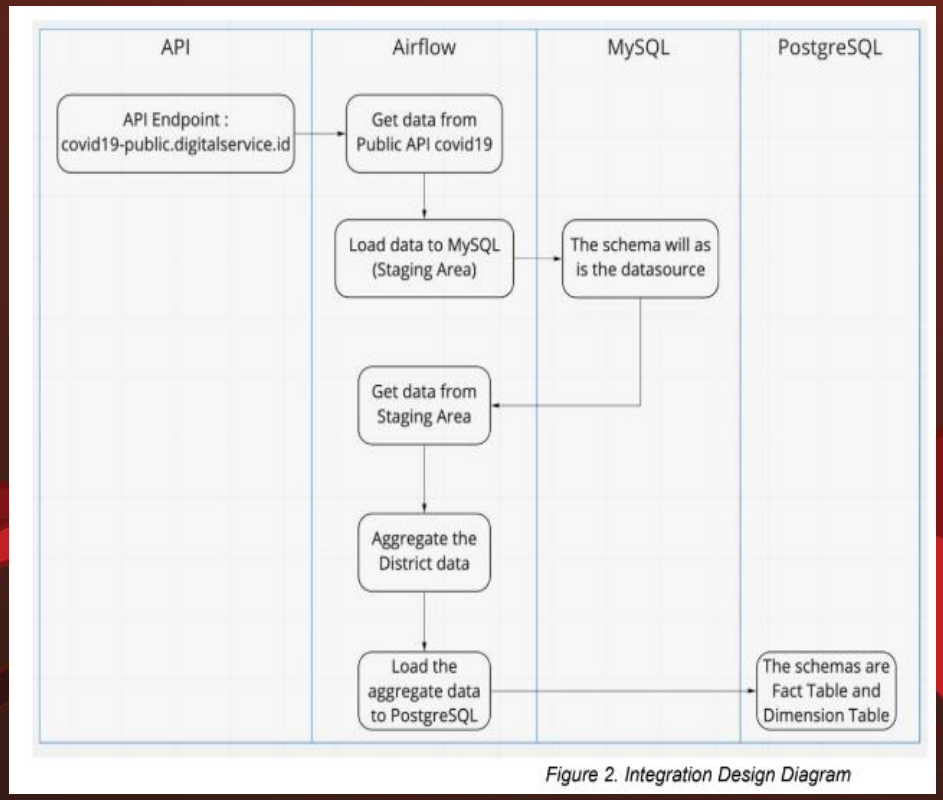


Figure 2. Integration Design Diagram

API Response Example


```
{
   "status_code":
200, "data": {
         "metadata": {
   "last_update": null
      },
      "content": [
         {
            "tanggal": "2020-08-05",
            "kode_prov": "32",
     "nama_prov": "Jawa Barat",
     "kode_kab": "3204",
            "nama_kab": "Kabupaten
     Bandung", "SUSPECT": 2210,
            "CLOSECONTACT": 274,
            "PROBABLE": 26,
            "suspect_diisolasi": 31,
            "suspect_discarded": 2179,
            "closecontact_dikarantina": 0,
            "closecontact_discarded": 274,
            "probable_diisolasi": 0,
            "probable_discarded": 0,
            "CONFIRMATION": 0,
            "confirmation_sembuh": 0,
            "confirmation_meninggal": 0,
            "suspect_meninggal": 0,
            "closecontact_meninggal": 0,
            "probable_meninggal": 26
         }
      ]
   }
}
```

Table Specification

Table Specification

Dimension table

1. Province table
    a. province_id
    b. province_name
2. District table
    a. district_id
    b. province_id
    c. district_name
3. Case table
    a. Id
    b. Status name (suspect, closecontact, probable, confirmation)
    c. Status detail

Fact table

1. Province Daily Table
    a. Id (auto generate)
    b. province_id
    c. case_id
    d. date
    e. total
2. District Daily Table
    a. Id (auto generate)
    b. district_id
    c. case_id
    d. date
    e. total

Create Docker (MySQL, PostgreSQL and Airflow) in Local

docker-compose.yaml

```yaml
version: '2'
services:
  postgres-db:
    image: postgres:15
    environment:
      PGDATA: /var/lib/postgresql/data
      POSTGRES_PASSWORD: postgres
      POSTGRES_USER: postgres
      POSTGRES_DB: dwh
    volumes:
    - postgres_data:/var/lib/postgresql/data
    ports:
    - 5435:5432/tcp
  mysql-db:
    image: mysql:8.0
    environment:
      - MYSQL_DATABASE=mysql
      - MYSQL_USER=mysql
      - MYSQL_PASSWORD=mysql
      - MYSQL_ROOT_PASSWORD=mysql
    ports:
      - 3307:3306/tcp
    volumes:
      - mysql_data:/var/lib/mysql
volumes:
  mysql_data: # docker volume create mysql_data
    external: true
  postgres_data: # docker volume create postgres_data
    external: true
networks:
  final_project:
    driver: bridge
```

airflow-docker-compose.yml

```yaml
---
version: '3.4'

x-common:
  &common
  image: apache/airflow:2.3.0
  user: "${AIRFLOW_UID}:0"
  env_file:
    - .env
  volumes:
    - ./dags:/opt/airflow/dags
    - ./logs:/opt/airflow/logs
    - ./plugins:/opt/airflow/plugins
    - /var/run/docker.sock:/var/run/docker.sock

x-depends-on:
  &depends-on
  depends_on:
    postgres:
      condition: service_healthy
    airflow-init:
      condition: service_completed_successfully

services:
  postgres:
    image: postgres:13
    container_name: postgres
    ports:
      - "5434:5432"
    healthcheck:
      test: ["CMD", "pg_isready", "-U", "airflow"]
      interval: 5s
      retries: 5
    env_file:
      - .env
    volumes:
      - postgres_airflow:/var/lib/postgresql/data

  scheduler:
    <<: [*common, *depends-on]
    container_name: airflow-scheduler
    command: scheduler
    restart: on-failure
    ports:
      - "8793:8793"

  webserver:
    <<: [*common, *depends-on]
    container_name: airflow-webserver
    restart: always
    command: webserver
    ports:
      - "8080:8080"
    healthcheck:
      test: ["CMD", "curl", "--fail", "http://localhost:8080/health"]
      interval: 30s
      timeout: 30s
      retries: 5

  airflow-init:
    <<: *common
    container_name: airflow-init
    entrypoint: /bin/bash
    command:
      - -c
      - |
        mkdir -p /sources/logs /sources/dags /sources/plugins
        chown -R "${AIRFLOW_UID}:0" /sources/{logs,dags,plugins}
        exec /entrypoint airflow version
volumes:
  postgres_airflow:
    external: true
networks:
  final_project:
    driver: bridge
```

Setting .env for Airflow and create docker volume

.env

```
# Meta-Database
POSTGRES_USER=airflow
POSTGRES_PASSWORD=airflow
POSTGRES_DB=airflow

# Airflow Core
AIRFLOW__CORE__FERNET_KEY=UKMzEm3yIuFYEq1y3-2FxPNWSVwRASpahmQ9kQfEr8E=
AIRFLOW__CORE__EXECUTOR=LocalExecutor
AIRFLOW__CORE__DAGS_ARE_PAUSED_AT_CREATION=True
AIRFLOW__CORE__LOAD_EXAMPLES=False
AIRFLOW_UID=0

# Backend DB
AIRFLOW__DATABASE__SQL_ALCHEMY_CONN=postgresql+psycopg2://airflow:airflow@postgres/airflow
AIRFLOW__DATABASE__LOAD_DEFAULT_CONNECTIONS=False

# Airflow Init
_AIRFLOW_DB_UPGRADE=True
_AIRFLOW_WWW_USER_CREATE=True
_AIRFLOW_WWW_USER_USERNAME=airflow
_AIRFLOW_WWW_USER_PASSWORD=airflow
```

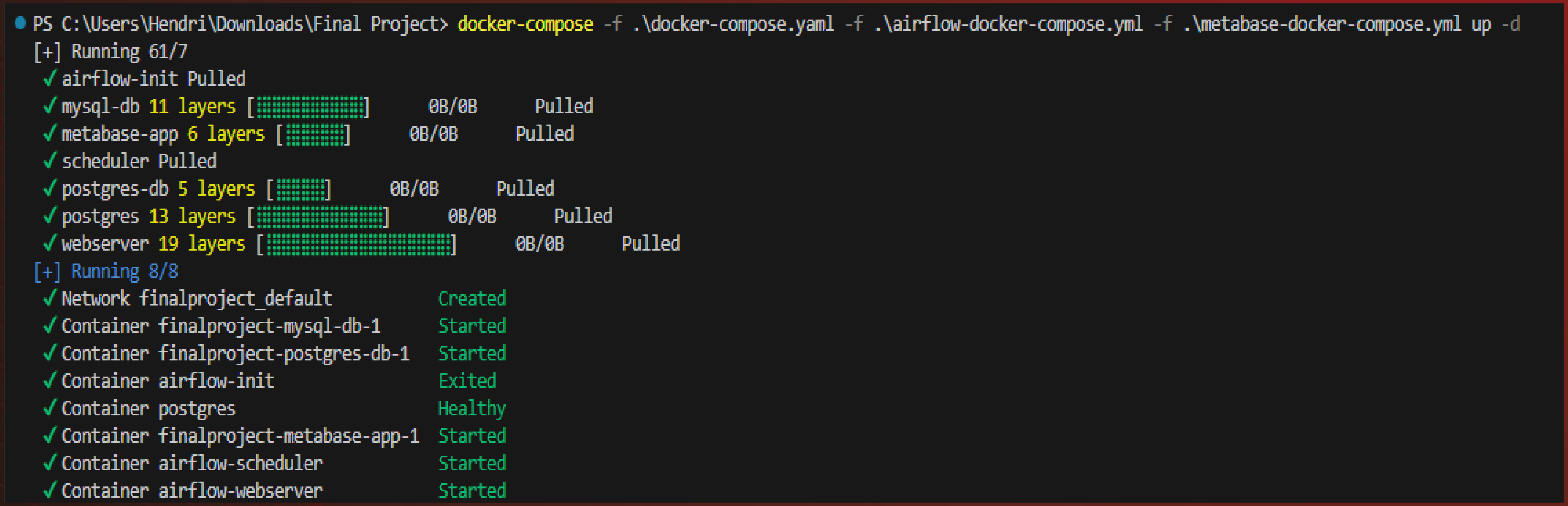
```
docker volume create mysql_data
docker volume create postgres_data
docker volume create postgres_airflow
```

Create Docker (Metabase) in Local

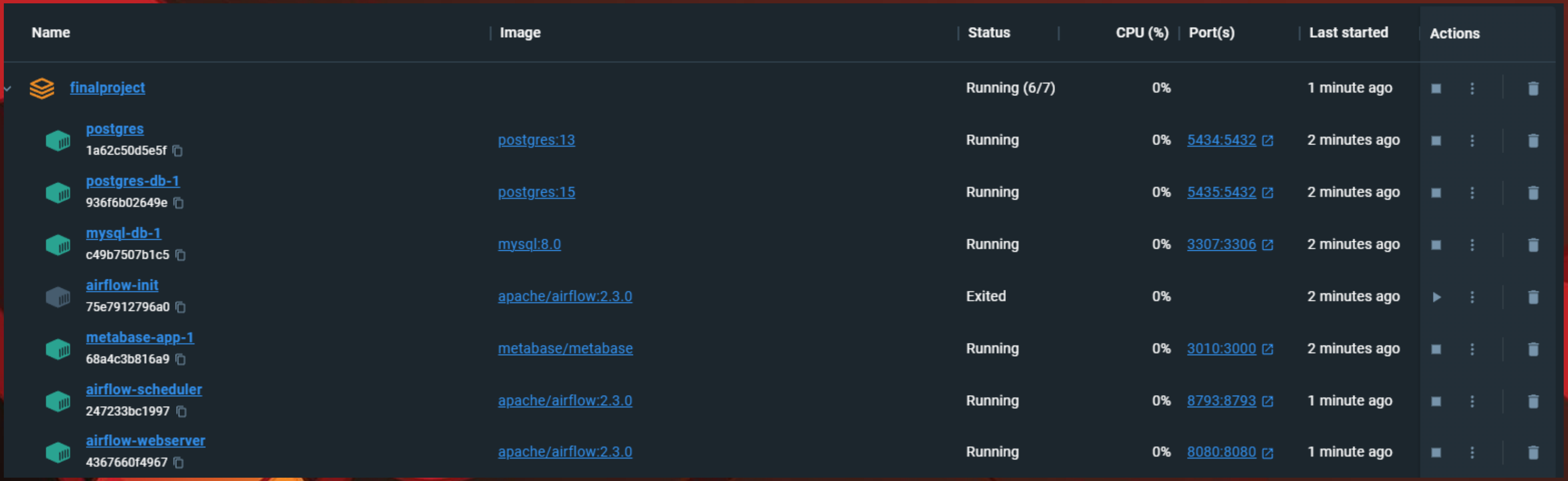
metabase-docker-compose.yml

```yaml
version: '2'
services:
  postgres-db:
    image: postgres:15
    environment:
      PGDATA: /var/lib/postgresql/data
      POSTGRES_PASSWORD: postgres
      POSTGRES_USER: postgres
      POSTGRES_DB: dwh
    volumes:
    - postgres_data:/var/lib/postgresql/data
    ports:
    - 5435:5432/tcp
  metabase-app:
    image: metabase/metabase
    environment:
      MB_DB_DBNAME: dwh
      MB_DB_HOST: postgres-db
      MB_DB_PASS: postgres
      MB_DB_PORT: 5432
      MB_DB_TYPE: postgres
      MB_DB_USER: postgres
    volumes:
    - metabase:/metabase-data
    links:
    - postgres-db:postgres-db
    ports:
    - 3010:3000/tcp
volumes:
  metabase: # docker volume create metabase
    external: true
  postgres_data: # docker volume create postgres_data
    external: true
networks:
  final_project:
    driver: bridge
```

Running Docker Compose on Images

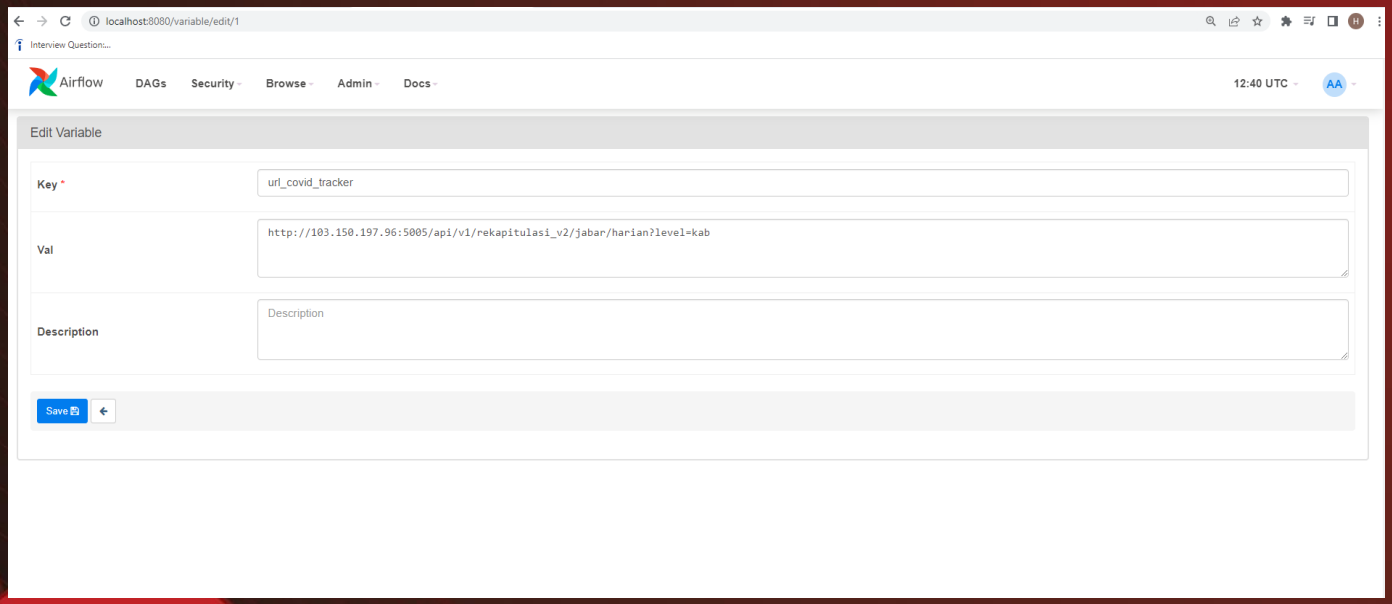


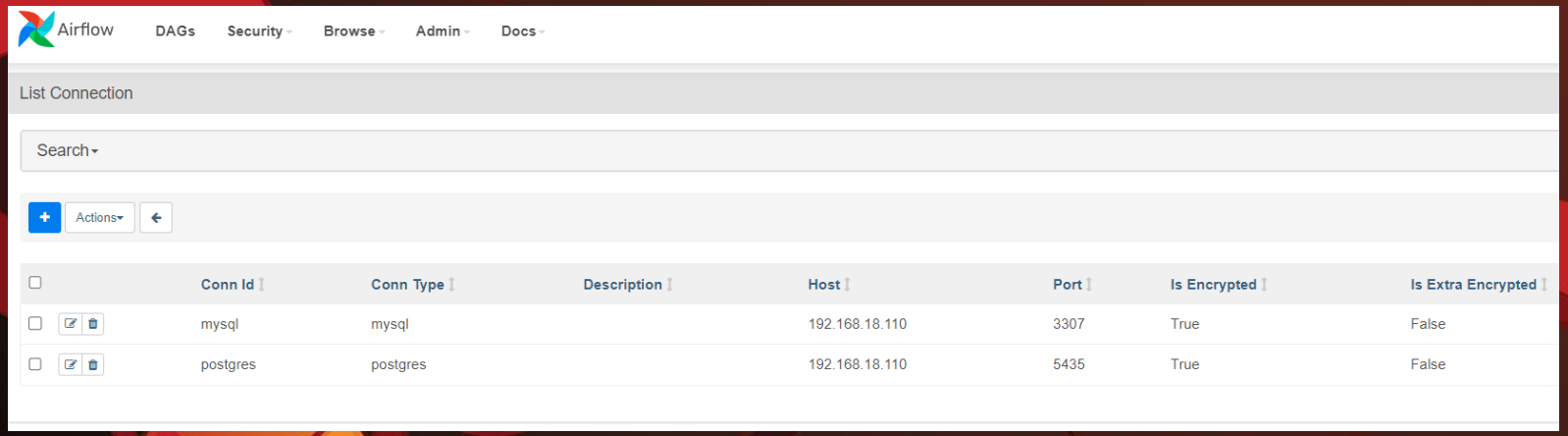
```
PS C:\Users\Hendri\Downloads\Final Project> docker-compose -f .\docker-compose.yaml -f .\airflow-docker-compose.yml -f .\metabase-docker-compose.yml up -d
[+] Running 61/7
 ✓ airflow-init Pulled
 ✓ mysql-db 11 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled
 ✓ metabase-app 6 layers [⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled
 ✓ scheduler Pulled
 ✓ postgres-db 5 layers [⣿⣿⣿⣿⣿]      0B/0B      Pulled
 ✓ postgres 13 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled
 ✓ webserver 19 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled
[+] Running 8/8
 ✓ Network finalproject_default             Created
 ✓ Container finalproject-mysql-db-1        Started
 ✓ Container finalproject-postgres-db-1     Started
 ✓ Container airflow-init                   Exited
 ✓ Container postgres                       Healthy
 ✓ Container finalproject-metabase-app-1    Started
 ✓ Container airflow-scheduler              Started
 ✓ Container airflow-webserver              Started
```



Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
finalproject		Running (6/7)	0%		1 minute ago	■ ⋮ 🗑
postgres 1a62c50d5e5f	postgres:13	Running	0%	5434:5432	2 minutes ago	■ ⋮ 🗑
postgres-db-1 936f6b02649e	postgres:15	Running	0%	5435:5432	2 minutes ago	■ ⋮ 🗑
mysql-db-1 c49b7507b1c5	mysql:8.0	Running	0%	3307:3306	2 minutes ago	■ ⋮ 🗑
airflow-init 75e7912796a0	apache/airflow:2.3.0	Exited	0%		2 minutes ago	▶ ⋮ 🗑
metabase-app-1 68a4c3b816a9	metabase/metabase	Running	0%	3010:3000	2 minutes ago	■ ⋮ 🗑
airflow-scheduler 247233bc1997	apache/airflow:2.3.0	Running	0%	8793:8793	1 minute ago	■ ⋮ 🗑
airflow-webserver 4367660f4967	apache/airflow:2.3.0	Running	0%	8080:8080	1 minute ago	■ ⋮ 🗑

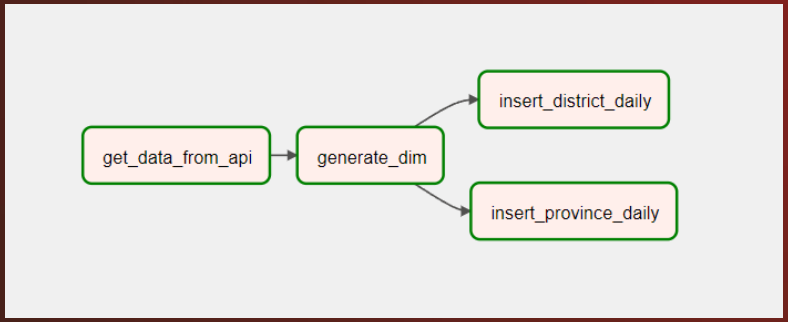
Input Variable and Connections in Airflow UI





Conn Id	Conn Type	Description	Host	Port	Is Encrypted	Is Extra Encrypted
mysql	mysql		192.168.18.110	3307	True	False
postgres	postgres		192.168.18.110	5435	True	False

Defining DAG Tasks and Functions for ETL



Functions needed:

- Scrap/get data from API
- Connect to MySQL and PostgreSQL
- Transform data into fact and dimension table

Functions for DAG file

dags > modules > covid_scraper.py > CovidScraper > __init__

```python
import requests
import pandas as pd
import logging


class CovidScraper():
    def __init__(self, url):
        self.url = url

    def get_data(self):
        response = requests.get(self.url)
        result = response.json()['data']['content']
        logging.info("GET DATA FROM API COMPLETED")
        df = pd.json_normalize(result)
        logging.info("DATA FROM API TO DATAFRAME READY")
        return df

```

dags > modules > connector.py > Connector > connect_postgres

```python
from sqlalchemy import create_engine


class Connector():
    def __init__(self):
        pass

    def connect_mysql(self, user, password, host, db, port):
        engine = create_engine('mysql+mysqlconnector://{}:{}@{}:{}/{}'.format(
            user, password, host, port, db
        ))
        return engine

    def connect_postgres(self, user, password, host, db, port):
        engine = create_engine('postgresql+psycopg2://{}:{}@{}:{}/{}'.format(
            user, password, host, port, db
        ))
        return engine
```

Functions for DAG file

dags > modules > transformer.py > Transformer > create_dimension_province

```python
import pandas as pd
from sqlalchemy.exc import SQLAlchemyError
import numpy as np


class Transformer():
    def __init__(self, engine_sql, engine_postgres):
        self.engine_sql = engine_sql
        self.engine_postgres = engine_postgres

    def get_data_from_mysql(self):
        sql = 'SELECT * FROM covid_jabar'
        df = pd.read_sql(sql, con = self.engine_sql)
        print("GET DATA FROM MYSQL SUCCESS")
        return df

    def create_dimension_province(self):
        df = self.get_data_from_mysql()
        df_province = df[['kode_prov', 'nama_prov']]
        df_province = df_province.rename(columns={'kode_prov': 'province_id', 'nama_prov': 'province_name'})
        df_province = df_province.drop_duplicates()

        try:
            drop = 'DROP TABLE IF EXISTS dim_province'
            self.engine_postgres.execute(drop)
        except SQLAlchemyError as e:
            print(e)

        #    to postgres
        df_province.to_sql(con=self.engine_postgres, name='dim_province', index=False)

        print('INSERTED TO POSTGRES SUCCESSFULLY')

    def create_dimension_district(self):
        df = self.get_data_from_mysql()
        df_district = df[['kode_kab', 'kode_prov', 'nama_kab']]
        df_district = df_district.rename(columns={'kode_kab': 'district_id', 'kode_prov': 'province_id', 'nama_kab': 'district_name'})
        df_district = df_district.drop_duplicates()

        try:
            drop = 'DROP TABLE IF EXISTS dim_district'
            self.engine_postgres.execute(drop)
        except SQLAlchemyError as e:
            print(e)

        # insert to postgres
        df_district.to_sql(con=self.engine_postgres, name='dim_district', index=False)

        print('INSERTED TO POSTGRES SUCCESSFULLY')
```

```python
    def create_dimension_case(self):
        df = self.get_data_from_mysql()
        column_start = ['suspect_diisolasi', 'suspect_discarded', 'closecontact_dikarantina', 'closecontact_discarded', 'probable_diisolasi' ,
                        'probable_discarded', 'confirmation_sembuh', 'confirmation_meninggal', 'suspect_meninggal', 'closecontact_meninggal', 'probable_meninggal']
        column_end = ['id', 'status_name', 'status_detail', 'status']

        df = df[column_start]
        df = df[:1]
        df = df.melt(var_name='status', value_name='total')
        df = df.drop_duplicates('status').sort_values('status')

        df['id'] = np.arange(1, df.shape[0]+1)
        df[['status_name', 'status_detail']] = df['status'].str.split('_', n=1, expand=True)

        df = df[column_end]

        try:
            drop = 'DROP TABLE IF EXISTS dim_case'
            self.engine_postgres.execute(drop)
        except SQLAlchemyError as e:
            print(e)

        # insert to postgres
        df.to_sql(con=self.engine_postgres, name='dim_case', index=False)

        print('INSERTED TO POSTGRES SUCCESSFULLY')

        return df
```

Functions for DAG file

```python
def create_province_daily(self):
    df = self.get_data_from_mysql()
    df_case_dim = self.create_dimension_case()

    column_start = ['tanggal', 'kode_prov', 'suspect_diisolasi', 'suspect_discarded', 'closecontact_dikarantina', 'closecontact_discarded', 'probable_diisolasi',
                    'probable_discarded', 'confirmation_sembuh', 'confirmation_meninggal', 'suspect_meninggal', 'closecontact_meninggal', 'probable_meninggal']
    column_end = ['date', 'province_id', 'status', 'total']

    data = df[column_start]
    data = data.melt(id_vars = ['tanggal', 'kode_prov'], var_name='status', value_name='total').sort_values(['tanggal', 'kode_prov', 'status'])
    data = data.groupby(by=['tanggal', 'kode_prov', 'status']).sum()
    data = data.reset_index()

    data.columns = column_end
    data['id'] = np.arange(1, data.shape[0]+1)
    df_case_dim = df_case_dim.rename({'id': 'case_id'}, axis=1)

    data = pd.merge(data, df_case_dim, how='inner', on='status')
    data = data[['id', 'province_id', 'case_id', 'date', 'total']]

    try:
        drop = 'DROP TABLE IF EXISTS province_daily'
        self.engine_postgres.execute(drop)
    except SQLAlchemyError as e:
        print(e)

    # insert to postgres
    data.to_sql(con=self.engine_postgres, name='province_daily', index=False)
```

```python
def create_district_daily(self):
    df = self.get_data_from_mysql()
    df_case_dim = self.create_dimension_case()

    column_start = ['tanggal', 'kode_kab', 'suspect_diisolasi', 'suspect_discarded', 'closecontact_dikarantina', 'closecontact_discarded', 'probable_diisolasi',
                    'probable_discarded', 'confirmation_sembuh', 'confirmation_meninggal', 'suspect_meninggal', 'closecontact_meninggal', 'probable_meninggal']
    column_end = ['date', 'district_id', 'status', 'total']

    data = df[column_start]
    data = data.melt(id_vars = ['tanggal', 'kode_kab'], var_name='status', value_name='total').sort_values(['tanggal', 'kode_kab', 'status'])
    data = data.groupby(by=['tanggal', 'kode_kab', 'status']).sum()
    data = data.reset_index()

    data.columns = column_end
    data['id'] = np.arange(1, data.shape[0]+1)
    df_case_dim = df_case_dim.rename({'id': 'case_id'}, axis=1)

    data = pd.merge(data, df_case_dim, how='inner', on='status')
    data = data[['id', 'district_id', 'case_id', 'date', 'total']]

    try:
        drop = 'DROP TABLE IF EXISTS district_daily'
        self.engine_postgres.execute(drop)
    except SQLAlchemyError as e:
        print(e)

    # insert to postgres
    data.to_sql(con=self.engine_postgres, name='district_daily', index=False)
```

Creating DAG file

dags > final_project.py > ...

```python
from datetime import datetime
import logging

from airflow import DAG
from airflow.models import Variable, Connection
from airflow.operators.python import PythonOperator

from modules.covid_scraper import CovidScraper
from modules.transformer import Transformer
from modules.connector import Connector


def fun_get_data_from_api(**kwargs):
    # get data
    scraper = CovidScraper(Variable.get('url_covid_tracker'))
    data = scraper.get_data()
    print(data.info())

    # create connector
    get_conn = Connection.get_connection_from_secrets("mysql")
    connector = Connector()
    engine_sql = connector.connect_mysql(
        host = get_conn.host,
        user = get_conn.login,
        password = get_conn.password,
        db = get_conn.schema,
        port = get_conn.port
    )

    # drop table if exists
    try:
        drop = "DROP TABLE IF EXISTS covid_jabar"
        engine_sql.execute(drop)
    except Exception as e:
        logging.error(e)

    # insert to mysql
    data.to_sql(con=engine_sql, name='covid_jabar', index=False)
    logging.info("DATA INSERTED SUCCESSFULLY TO MYSQL")
```

```python

def fun_generate_dim(**kwargs):
    # create connector
    get_conn_mysql = Connection.get_connection_from_secrets("mysql")
    get_conn_postgres = Connection.get_connection_from_secrets("postgres")
    connector = Connector()
    engine_sql = connector.connect_mysql(
        host = get_conn_mysql.host,
        user = get_conn_mysql.login,
        password = get_conn_mysql.password,
        db = get_conn_mysql.schema,
        port = get_conn_mysql.port
    )
    engine_postgres = connector.connect_postgres(
        host = get_conn_postgres.host,
        user = get_conn_postgres.login,
        password = get_conn_postgres.password,
        db = get_conn_postgres.schema,
        port = get_conn_postgres.port
    )

    # insert data
    transformer = Transformer(engine_sql, engine_postgres)
    transformer.create_dimension_case()
    transformer.create_dimension_district()
    transformer.create_dimension_province()

```

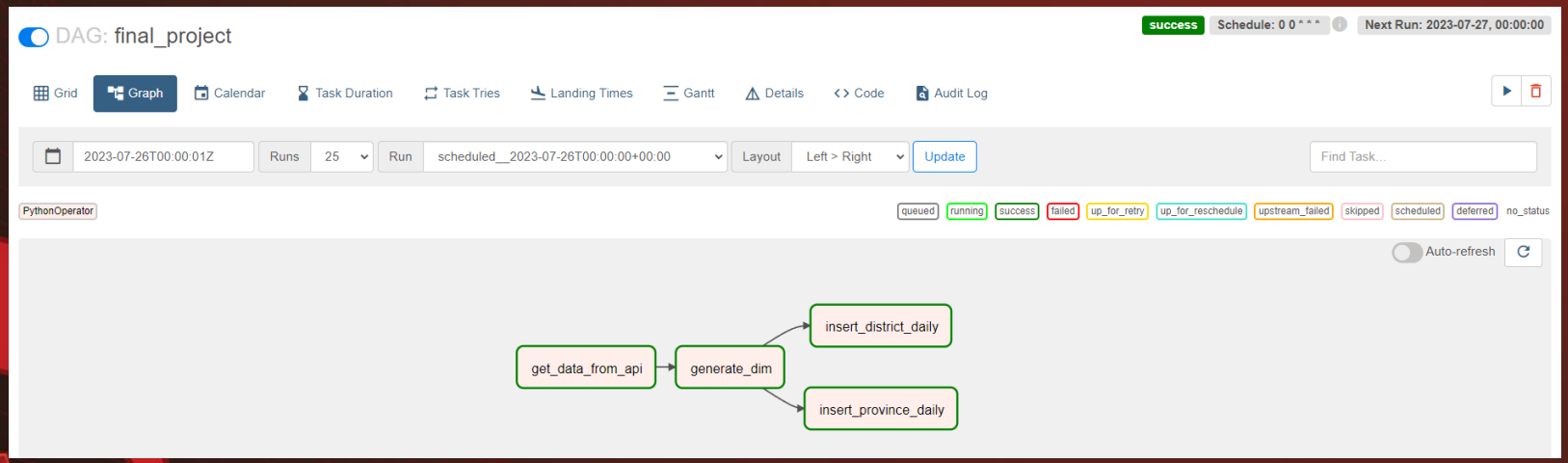
Creating DAG file

```python
def fun_insert_province_daily(**kwargs):
    # create connector
    get_conn_mysql = Connection.get_connection_from_secrets("mysql")
    get_conn_postgres = Connection.get_connection_from_secrets("postgres")
    connector = Connector()
    engine_sql = connector.connect_mysql(
        host = get_conn_mysql.host,
        user = get_conn_mysql.login,
        password = get_conn_mysql.password,
        db = get_conn_mysql.schema,
        port = get_conn_mysql.port
    )
    engine_postgres = connector.connect_postgres(
        host = get_conn_postgres.host,
        user = get_conn_postgres.login,
        password = get_conn_postgres.password,
        db = get_conn_postgres.schema,
        port = get_conn_postgres.port
    )

    # insert data
    transformer = Transformer(engine_sql, engine_postgres)
    transformer.create_province_daily()
```

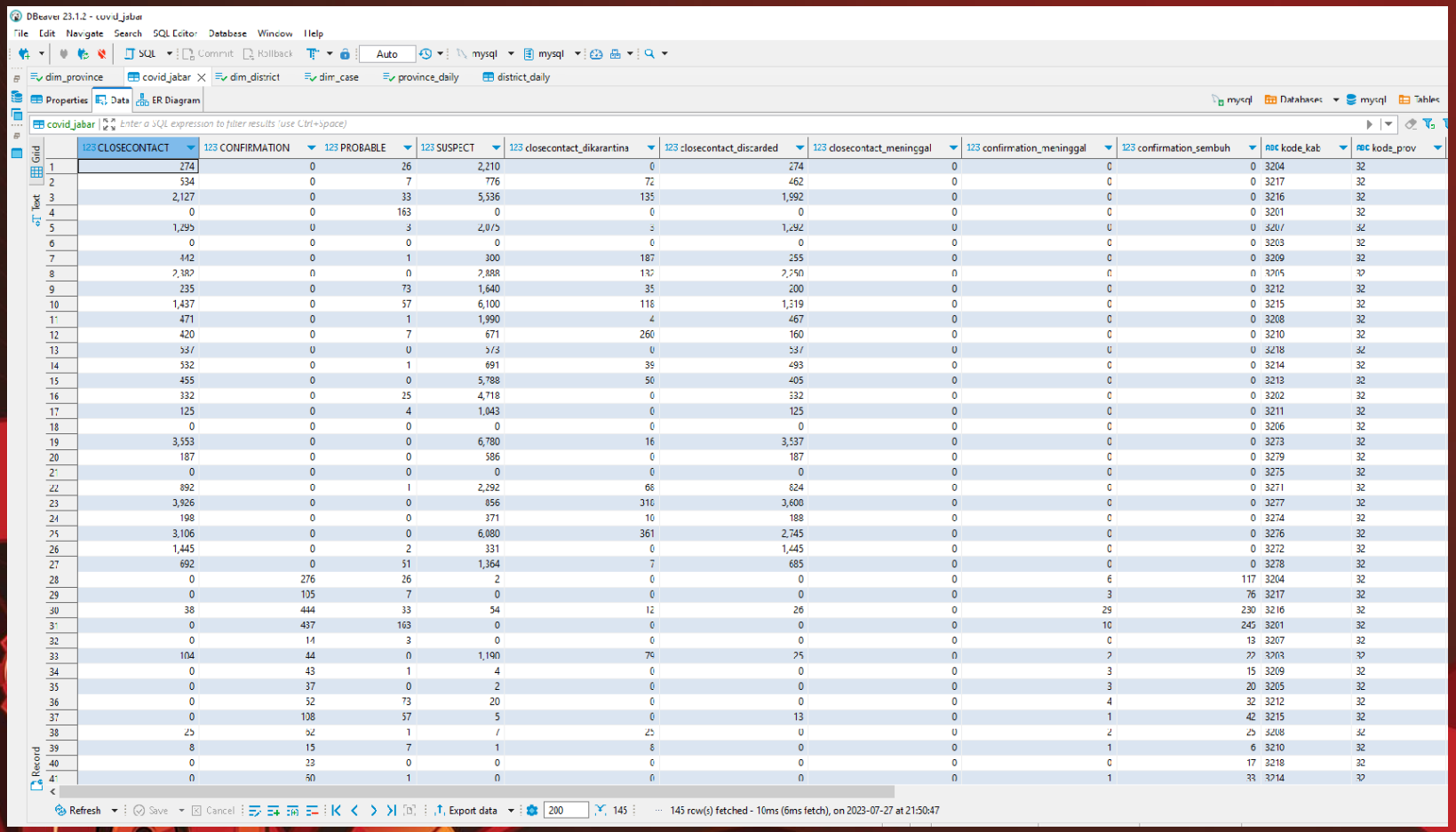
```python
def fun_insert_district_daily(**kwargs):
    # create connector
    get_conn_mysql = Connection.get_connection_from_secrets("mysql")
    get_conn_postgres = Connection.get_connection_from_secrets("postgres")
    connector = Connector()
    engine_sql = connector.connect_mysql(
        host = get_conn_mysql.host,
        user = get_conn_mysql.login,
        password = get_conn_mysql.password,
        db = get_conn_mysql.schema,
        port = get_conn_mysql.port
    )
    engine_postgres = connector.connect_postgres(
        host = get_conn_postgres.host,
        user = get_conn_postgres.login,
        password = get_conn_postgres.password,
        db = get_conn_postgres.schema,
        port = get_conn_postgres.port
    )

    # insert data
    transformer = Transformer(engine_sql, engine_postgres)
    transformer.create_district_daily()
```

```python
with DAG(
    dag_id = 'final_project',
    start_date = datetime(2022, 7, 20),
    schedule_interval = '0 0 * * *',
    catchup = False
) as dag:

    op_get_data_from_api = PythonOperator(
        task_id = 'get_data_from_api',
        python_callable = fun_get_data_from_api
    )

    op_generate_dim = PythonOperator(
        task_id = 'generate_dim',
        python_callable = fun_generate_dim
    )

    op_insert_province_daily = PythonOperator(
        task_id = 'insert_province_daily',
        python_callable = fun_insert_province_daily
    )

    op_insert_district_daily = PythonOperator(
        task_id = 'insert_district_daily',
        python_callable = fun_insert_district_daily
    )

op_get_data_from_api >> op_generate_dim
op_generate_dim >> op_insert_province_daily
op_generate_dim >> op_insert_district_daily
```

Execute or Start DAG



Checking Data on DBeaver (MySQL)



	CLOSECONTACT	CONFIRMATION	PROBABLE	SUSPECT	closecontact_dikarantina	closecontact_discarded	closecontact_meninggal	confirmation_meninggal	confirmation_sembuh	kode_kab	kode_prov
1	274	0	26	2,210	0	274	0	0	0	3204	32
2	534	0	7	776	72	462	0	0	0	3217	32
3	2,127	0	33	5,536	135	1,992	0	0	0	3216	32
4	0	0	163	0	0	0	0	0	0	3201	32
5	1,295	0	3	2,075	3	1,292	0	0	0	3207	32
6	0	0	0	0	0	0	0	0	0	3203	32
7	442	0	1	300	187	255	0	0	0	3209	32
8	2,382	0	0	2,888	132	2,250	0	0	0	3205	32
9	235	0	73	1,640	35	200	0	0	0	3212	32
10	1,437	0	57	6,100	118	1,319	0	0	0	3215	32
11	471	0	1	1,990	4	467	0	0	0	3208	32
12	420	0	7	671	260	160	0	0	0	3210	32
13	537	0	0	573	0	537	0	0	0	3218	32
14	532	0	1	691	39	493	0	0	0	3214	32
15	455	0	0	5,788	50	405	0	0	0	3213	32
16	332	0	25	4,718	0	332	0	0	0	3202	32
17	125	0	4	1,043	0	125	0	0	0	3211	32
18	0	0	0	0	0	0	0	0	0	3206	32
19	3,553	0	0	6,780	16	3,537	0	0	0	3273	32
20	187	0	0	586	0	187	0	0	0	3279	32
21	0	0	0	0	0	0	0	0	0	3275	32
22	892	0	1	2,292	68	824	0	0	0	3271	32
23	3,926	0	0	856	316	3,600	0	0	0	3277	32
24	198	0	0	371	10	188	0	0	0	3274	32
25	3,106	0	0	6,080	361	2,745	0	0	0	3276	32
26	1,445	0	2	331	0	1,445	0	0	0	3272	32
27	692	0	51	1,364	7	685	0	0	0	3278	32
28	0	276	26	2	0	0	0	6	117	3204	32
29	0	105	7	0	0	0	0	3	76	3217	32
30	38	444	33	54	12	26	0	29	230	3216	32
31	0	437	163	0	0	0	0	10	245	3201	32
32	0	14	3	0	0	0	0	0	13	3207	32
33	104	44	0	1,190	79	25	0	2	22	3203	32
34	0	43	1	4	0	0	0	3	15	3209	32
35	0	37	0	2	0	0	0	3	20	3205	32
36	0	52	73	20	0	0	0	4	32	3212	32
37	0	108	57	5	0	13	0	1	42	3215	32
38	25	62	1	7	25	0	0	2	25	3208	32
39	8	15	7	1	8	0	0	1	6	3210	32
40	0	23	0	0	0	0	0	0	17	3218	32
41	0	60	1	0	0	0	0	1	33	3214	32

Checking Data on DBeaver (MySQL) - Continue

nama_kab	nama_prov	probable_diisolasi	probable_discarded	probable_meninggal	suspect_diisolasi	suspect_discarded	suspect_meninggal	tanggal
Kabupaten Bandung	Jawa Barat	0	0	26	31	2,179	0	2020-08-05
Kabupaten Bandung Barat	Jawa Barat	0	0	7	3	773	0	2020-08-05
Kabupaten Bekasi	Jawa Barat	1	0	32	4,001	1,535	0	2020-08-05
Kabupaten Bogor	Jawa Barat	0	0	163	0	0	0	2020-08-05
Kabupaten Ciamis	Jawa Barat	0	3	0	2,075	0	0	2020-08-05
Kabupaten Cianjur	Jawa Barat	0	0	0	0	0	0	2020-08-05
Kabupaten Cirebon	Jawa Barat	0	0	1	5	295	0	2020-08-05
Kabupaten Garut	Jawa Barat	0	0	0	2,644	244	0	2020-08-05
Kabupaten Indramayu	Jawa Barat	0	0	73	347	1,293	0	2020-08-05
Kabupaten Karawang	Jawa Barat	0	0	57	25	6,075	0	2020-08-05
Kabupaten Kuningan	Jawa Barat	0	0	1	1,683	307	0	2020-08-05
Kabupaten Majalengka	Jawa Barat	0	0	7	11	660	0	2020-08-05
Kabupaten Pangandaran	Jawa Barat	0	0	0	0	573	0	2020-08-05
Kabupaten Purwakarta	Jawa Barat	0	0	1	5	686	0	2020-08-05
Kabupaten Subang	Jawa Barat	0	0	0	1	5,787	0	2020-08-05
Kabupaten Sukabumi	Jawa Barat	0	0	25	4	4,714	0	2020-08-05
Kabupaten Sumedang	Jawa Barat	0	0	4	991	52	0	2020-08-05
Kabupaten Tasikmalaya	Jawa Barat	0	0	0	0	0	0	2020-08-05
Kota Bandung	Jawa Barat	0	0	0	6,780	0	0	2020-08-05
Kota Banjar	Jawa Barat	0	0	0	475	111	0	2020-08-05
Kota Bekasi	Jawa Barat	0	0	0	0	0	0	2020-08-05
Kota Bogor	Jawa Barat	0	0	1	61	2,231	0	2020-08-05
Kota Cimahi	Jawa Barat	0	0	0	330	526	0	2020-08-05
Kota Cirebon	Jawa Barat	0	0	0	4	367	0	2020-08-05
Kota Depok	Jawa Barat	0	0	0	428	5,652	0	2020-08-05
Kota Sukabumi	Jawa Barat	0	0	2	2	329	0	2020-08-05
Kota Tasikmalaya	Jawa Barat	0	46	5	1,364	0	0	2020-08-05
Kabupaten Bandung	Jawa Barat	0	0	26	0	7	117	2020-08-06
Kabupaten Bandung Barat	Jawa Barat	0	0	7	0	0	52	2020-08-06
Kabupaten Bekasi	Jawa Barat	1	0	32	0	3,947	249	2020-08-06
Kabupaten Bogor	Jawa Barat	0	0	163	0	0	486	2020-08-06
Kabupaten Ciamis	Jawa Barat	0	3	0	0	2,075	20	2020-08-06
Kabupaten Cianjur	Jawa Barat	0	0	0	170	1,020	89	2020-08-06
Kabupaten Cirebon	Jawa Barat	0	0	1	2	2	54	2020-08-06
Kabupaten Garut	Jawa Barat	0	0	0	0	2,626	6	2020-08-06
Kabupaten Indramayu	Jawa Barat	0	0	73	20	0	64	2020-08-06
Kabupaten Karawang	Jawa Barat	0	0	57	0	6	62	2020-08-06
Kabupaten Kuningan	Jawa Barat	0	0	1	0	1,656	28	2020-08-06
Kabupaten Majalengka	Jawa Barat	0	0	7	1	0	17	2020-08-06
Kabupaten Pangandaran	Jawa Barat	0	0	0	0	0	10	2020-08-06
Kabupaten Purwakarta	Jawa Barat	0	0	1	0	0	33	2020-08-06

Checking Data on DBeaver (PostgreSQL) – Dim Table

dim_province | covid_jabar | dim_district | dim

Properties | Data | ER Diagram

dim_province | Enter a SQL expression to filter results (use Ctrl+S

	province_id	province_name
1	32	Jawa Barat

dim_province | covid_jabar | dim_district | dim_case | prov

Properties | Data | ER Diagram

dim_district | Enter a SQL expression to filter results (use Ctrl+Space)

	district_id	province_id	district_name
1	3204	32	Kabupaten Bandung
2	3217	32	Kabupaten Bandung Barat
3	3216	32	Kabupaten Bekasi
4	3201	32	Kabupaten Bogor
5	3207	32	Kabupaten Ciamis
6	3203	32	Kabupaten Cianjur
7	3209	32	Kabupaten Cirebon
8	3205	32	Kabupaten Garut
9	3212	32	Kabupaten Indramayu
10	3215	32	Kabupaten Karawang
11	3208	32	Kabupaten Kuningan
12	3210	32	Kabupaten Majalengka
13	3218	32	Kabupaten Pangandaran
14	3214	32	Kabupaten Purwakarta
15	3213	32	Kabupaten Subang
16	3202	32	Kabupaten Sukabumi
17	3211	32	Kabupaten Sumedang
18	3206	32	Kabupaten Tasikmalaya
19	3273	32	Kota Bandung
20	3279	32	Kota Banjar
21	3275	32	Kota Bekasi
22	3271	32	Kota Bogor
23	3277	32	Kota Cimahi
24	3274	32	Kota Cirebon
25	3276	32	Kota Depok
26	3272	32	Kota Sukabumi
27	3278	32	Kota Tasikmalaya

Checking Data on DBeaver (PostgreSQL) - Dim Table

dim_case | Enter a SQL expression to filter results (use Ctrl+Space)

Grid | Text

	123 id	ABC status_name	ABC status_detail	ABC status
1	1	closecontact	dikarantina	closecontact_dikarantina
2	2	closecontact	discarded	closecontact_discarded
3	3	closecontact	meninggal	closecontact_meninggal
4	4	confirmation	meninggal	confirmation_meninggal
5	5	confirmation	sembuh	confirmation_sembuh
6	6	probable	diisolasi	probable_diisolasi
7	7	probable	discarded	probable_discarded
8	8	probable	meninggal	probable_meninggal
9	9	suspect	diisolasi	suspect_diisolasi
10	10	suspect	discarded	suspect_discarded
11	11	suspect	meninggal	suspect_meninggal

Checking Data on DBeaver (PostgreSQL) - Fact Table

province_daily | Enter a SQL expression to filter results (use Ctrl+Space)

	id	province_id	case_id	date	total
1	1	32	1	2020-08-05	1,815
2	12	32	1	2020-08-06	197
3	23	32	1	2020-08-07	37
4	34	32	1	2020-08-08	224
5	45	32	1	2020-08-09	120
6	56	32	1	2020-08-10	114
7	2	32	2	2020-08-05	23,782
8	13	32	2	2020-08-06	144
9	24	32	2	2020-08-07	8
10	35	32	2	2020-08-08	83
11	46	32	2	2020-08-09	36
12	57	32	2	2020-08-10	49
13	3	32	3	2020-08-05	0
14	14	32	3	2020-08-06	0
15	25	32	3	2020-08-07	0
16	36	32	3	2020-08-08	0
17	47	32	3	2020-08-09	0
18	58	32	3	2020-08-10	0
19	4	32	4	2020-08-05	0
20	15	32	4	2020-08-06	211
21	26	32	4	2020-08-07	4
22	37	32	4	2020-08-08	3
23	48	32	4	2020-08-09	4
24	59	32	4	2020-08-10	0
25	5	32	5	2020-08-05	0
26	16	32	5	2020-08-06	3,281
27	27	32	5	2020-08-07	0
28	38	32	5	2020-08-08	0
29	49	32	5	2020-08-09	0
30	60	32	5	2020-08-10	0
31	6	32	6	2020-08-05	1
32	17	32	6	2020-08-06	35
33	28	32	6	2020-08-07	36
34	39	32	6	2020-08-08	36
35	50	32	6	2020-08-09	37
36	61	32	6	2020-08-10	1
37	7	32	7	2020-08-05	49
38	18	32	7	2020-08-06	63
39	29	32	7	2020-08-07	63
40	40	32	7	2020-08-08	213
41	51	32	7	2020-08-09	215
42	62	32	7	2020-08-10	6

Refresh | Save | Cancel | Export data | 200 | 66 | 66 row(s) fetched - 2ms (1ms fetch),

Checking Data on DBeaver (PostgreSQL) - Fact Table

district_daily | Enter a SQL expression to filter results (use Ctrl+Space)

	id	district_id	case_id	date	total
1	1	3201	1	2020-08-05	0
2	12	3202	1	2020-08-05	0
3	23	3203	1	2020-08-05	0
4	34	3204	1	2020-08-05	0
5	45	3205	1	2020-08-05	132
6	56	3206	1	2020-08-05	0
7	67	3207	1	2020-08-05	3
8	78	3208	1	2020-08-05	4
9	89	3209	1	2020-08-05	187
10	100	3210	1	2020-08-05	260
11	111	3211	1	2020-08-05	0
12	122	3212	1	2020-08-05	35
13	133	3213	1	2020-08-05	50
14	144	3214	1	2020-08-05	39
15	155	3215	1	2020-08-05	118
16	166	3216	1	2020-08-05	135
17	177	3217	1	2020-08-05	72
18	188	3218	1	2020-08-05	0
19	199	3271	1	2020-08-05	68
20	210	3272	1	2020-08-05	0
21	221	3273	1	2020-08-05	16
22	232	3274	1	2020-08-05	10
23	243	3275	1	2020-08-05	0
24	254	3276	1	2020-08-05	361
25	265	3277	1	2020-08-05	318
26	276	3278	1	2020-08-05	7
27	287	3279	1	2020-08-05	0
28	298	3201	1	2020-08-06	0
29	309	3202	1	2020-08-06	0
30	320	3203	1	2020-08-06	79
31	331	3204	1	2020-08-06	0
32	342	3205	1	2020-08-06	0
33	353	3206	1	2020-08-06	0
34	364	3207	1	2020-08-06	0
35	375	3208	1	2020-08-06	25
36	386	3209	1	2020-08-06	0
37	397	3210	1	2020-08-06	8
38	408	3211	1	2020-08-06	0
39	419	3212	1	2020-08-06	0
40	430	3213	1	2020-08-06	0
41	441	3214	1	2020-08-06	0
42	452	3215	1	2020-08-06	0

Refresh | Save | Cancel | Export data | 200 | 200+ | 200 row(s) fetched -

Dashboard link:
http:///localhost:3010/public/dashboard/44821939-af6e-409c-b692-4e1bb98ceb32

Thank You!