Final Project: ETL Pikobar Data using Airflow

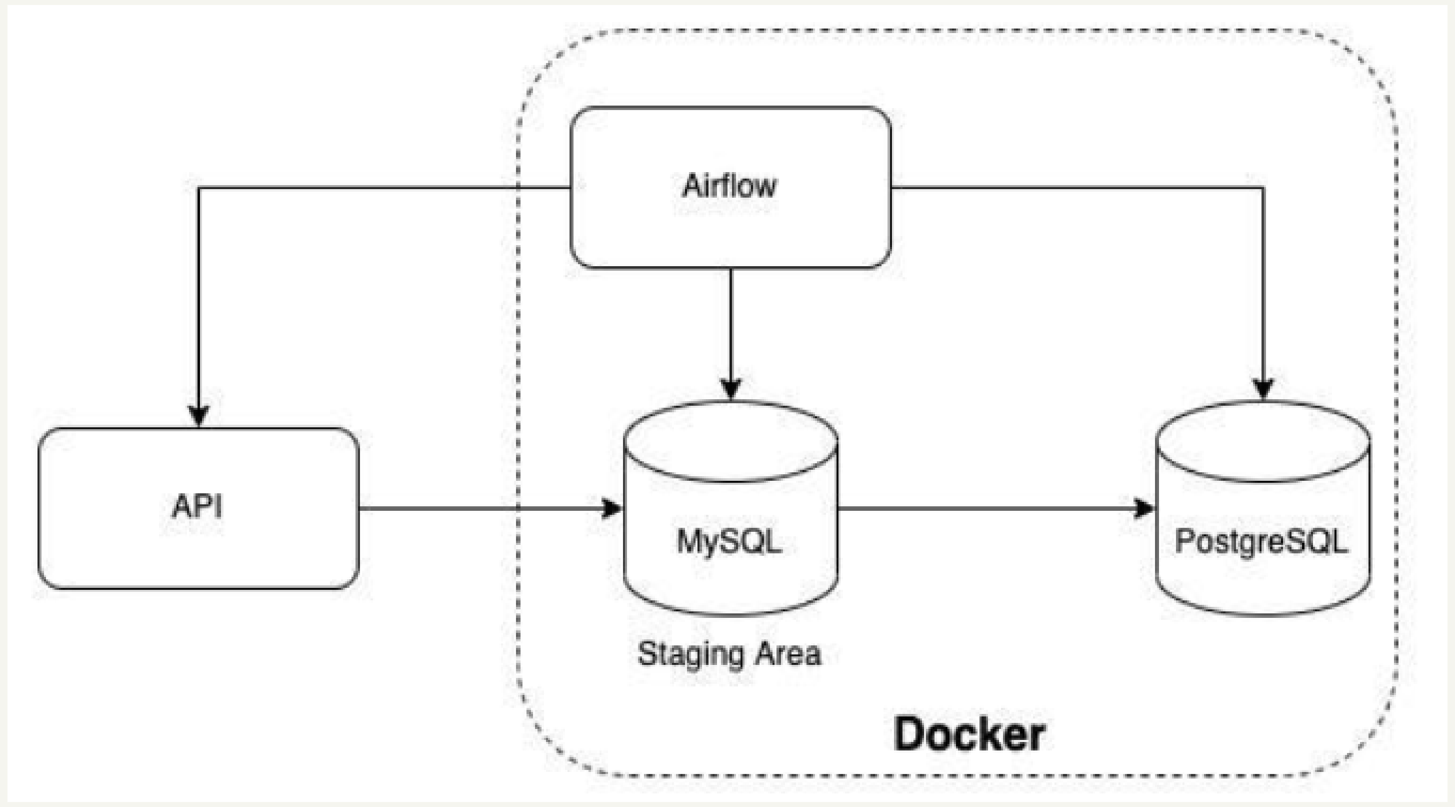
Oleh: Bernard Evan Kanigara

Daftar Isi

ETL Architecture

Deskripsi project

Dalam final project ini dibuat sebuah end-to-end Extract Transform Load (ETL) pipeline menggunakan Airflow. Data yang digunakan berupa data kasus covid dari Pusat Informasi dan Koordinasi COVID-19 Jawa Barat (PIKOBAR). Data dari PIKOBAR disimpan di MySQL (staging area) lalu diagregasi dan disimpan di PostgreSQL



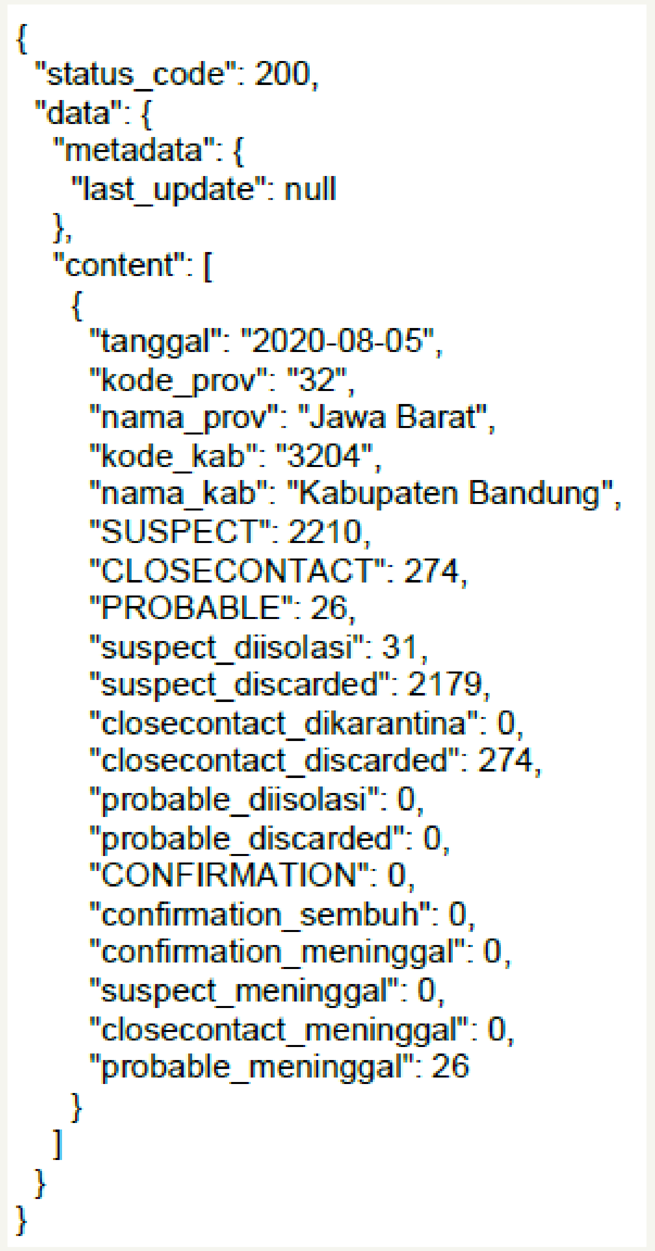
Diagram

Database PostgreSQL, MySQL, dan Airflow dibuat dalam docker di virtual machine

ssh -i <your-path>/user01.pem user01@34.69.56.212

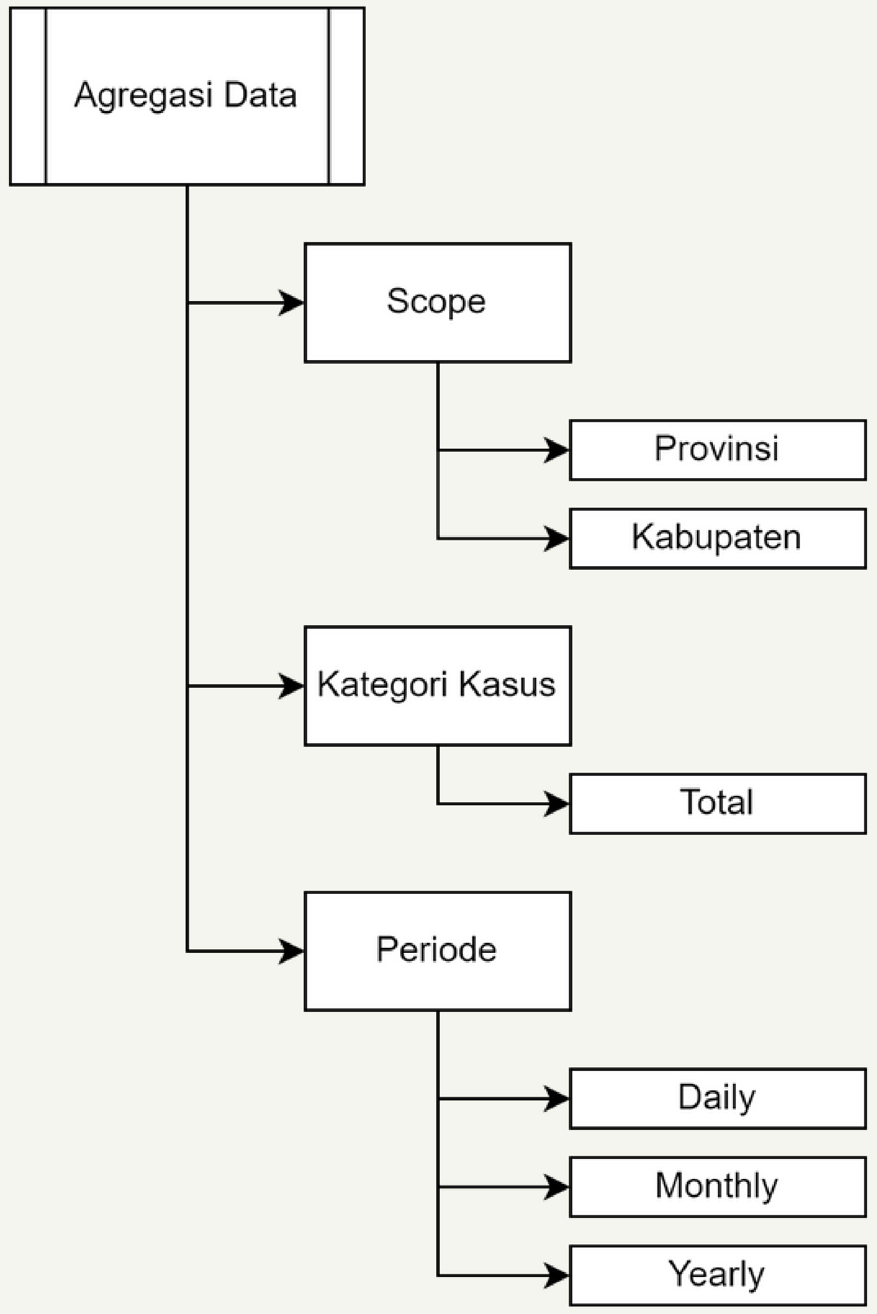
Project Data

PIKOBAR Data



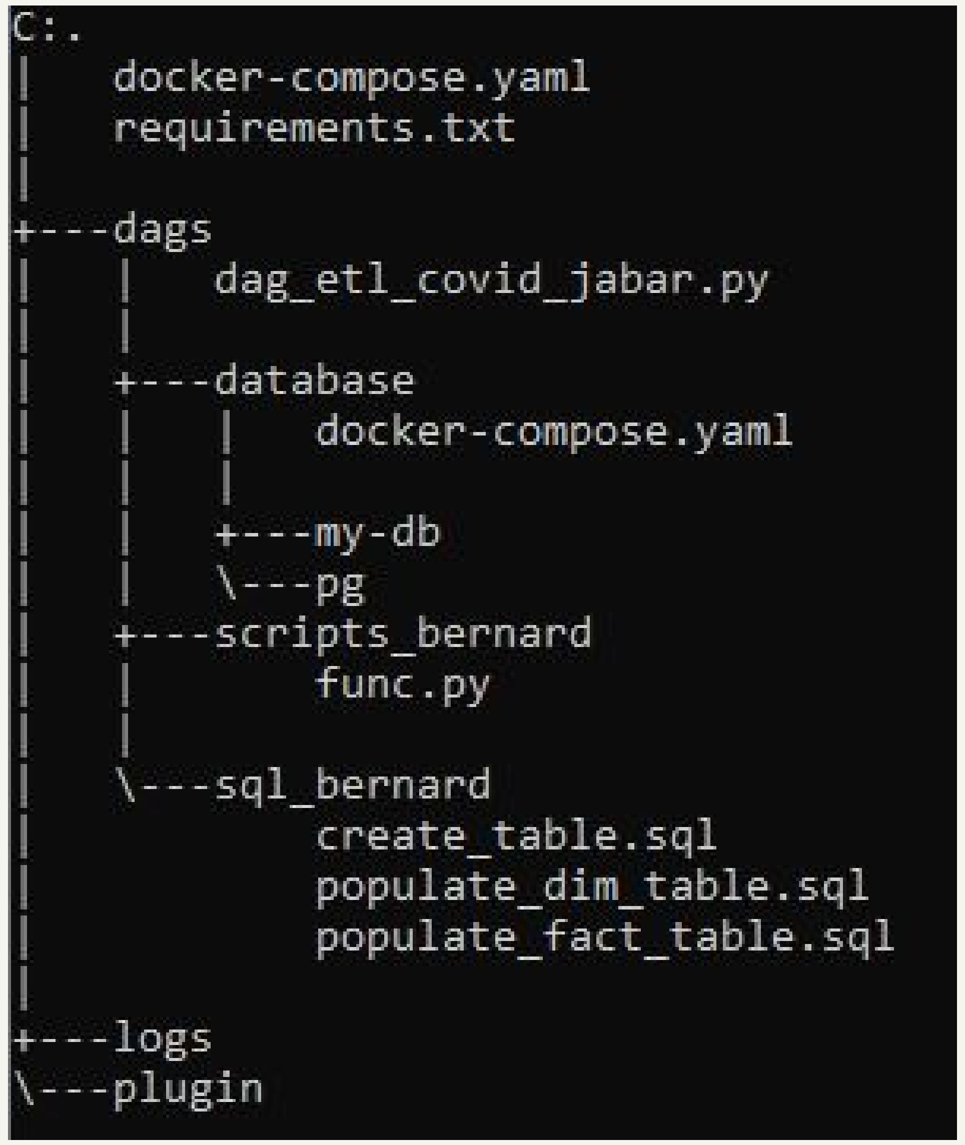
```
{
 "status_code": 200,
 "data": {
  "metadata": {
   "last_update": null
  },
  "content": [
   {
    "tanggal": "2020-08-05",
    "kode_prov": "32",
    "nama_prov": "Jawa Barat",
    "kode_kab": "3204",
    "nama_kab": "Kabupaten Bandung",
    "SUSPECT": 2210,
    "CLOSECONTACT": 274,
    "PROBABLE": 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
   }
  ]
 }
}
```

Agregasi akan dilakukan untuk mengetahui jumlah kasus COVID-19 dengan pembagian berdasrkan:
- Scope
- Kategori kasus
- Periode



Project File Structure

File Structure



```
C:.
|   docker-compose.yaml
|   requirements.txt
|
+---dags
|   |   dag_etl_covid_jabar.py
|   |
|   +---database
|   |   |   docker-compose.yaml
|   |   |
|   |   +---my-db
|   |   \---pg
|   +---scripts_bernard
|   |       func.py
|   |
|   \---sql_bernard
|           create_table.sql
|           populate_dim_table.sql
|           populate_fact_table.sql
|
+---logs
\---plugin
```

Dalam project ini terdapat file structure sebagai berikut:

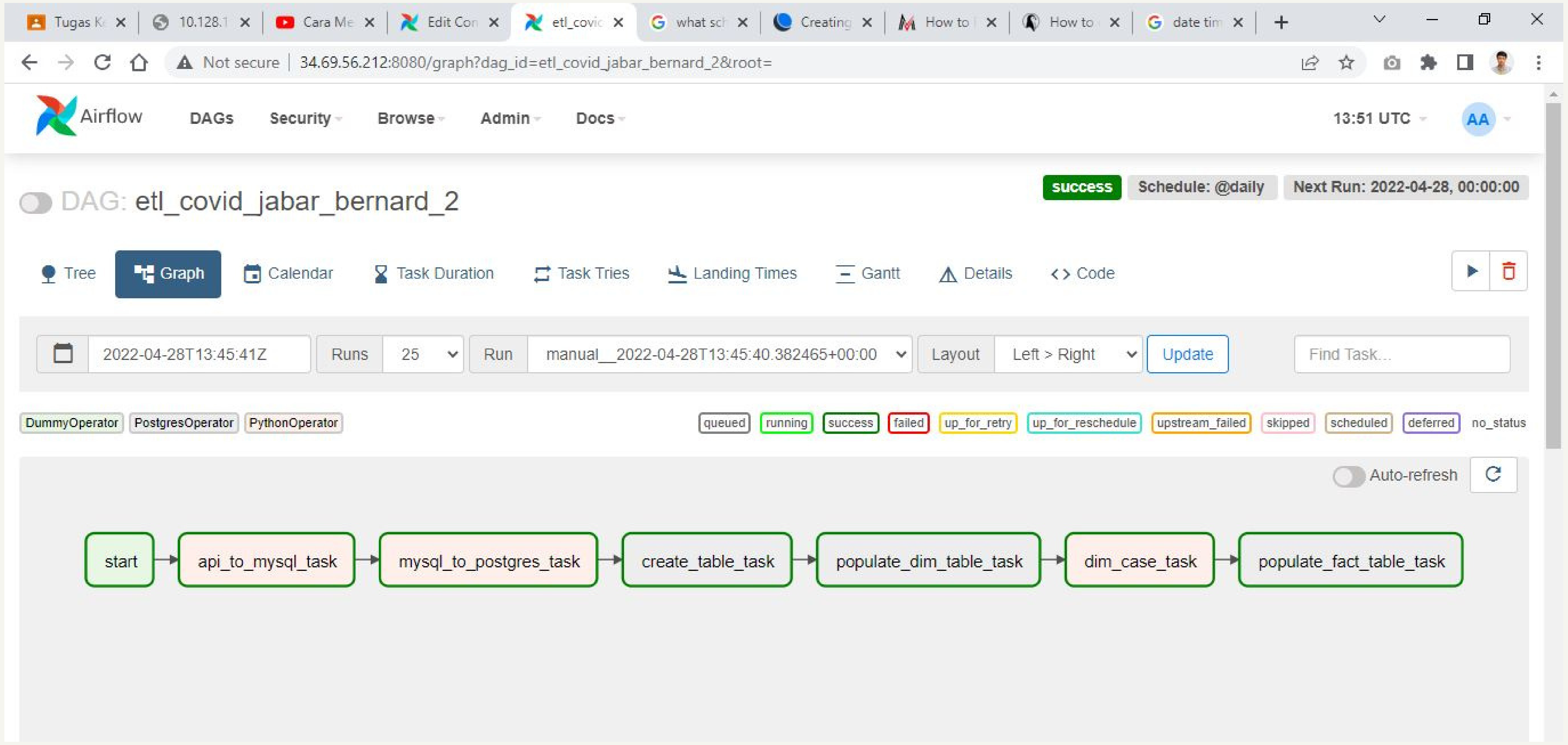
- dag_etl_covid_jabar akan befungsi sebagai file DAG di Airflow
- File DAG akan menjalankan task berupa ETL dari API menuju PostgreSQL
- Task akan menggunakan *python_operator* dalam func.py dan *postgres_operator* dalam file sql_bernard

Untuk database yang digunakkan:

- Mysql:5.7(Port: 3366)
- Postgres:14.1(Port: 5433)

Airflow DAG

DAG Structure


34.69.56.212:8080/graph?dag_id=etl_covid_jabar_bernard_2&root=

DAG: etl_covid_jabar_bernard_2

success | Schedule: @daily | Next Run: 2022-04-28, 00:00:00

2022-04-28T13:45:41Z | Runs 25 | Run manual__2022-04-28T13:45:40.382465+00:00 | Layout Left > Right | Update

DummyOperator | PostgresOperator | PythonOperator

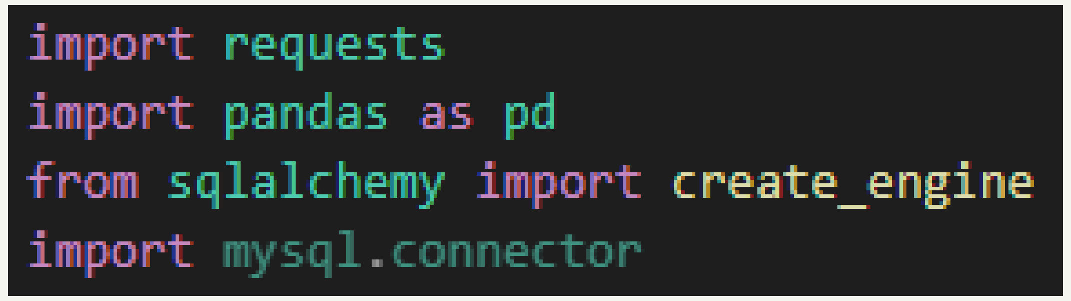
Dalam DAG digunakan dua operator yaitu:

- Dummy Operator
- Python Operator
- Postgre Operator

Project Step

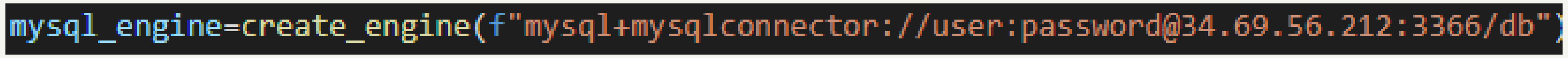
API to MySQL

1. Import Library



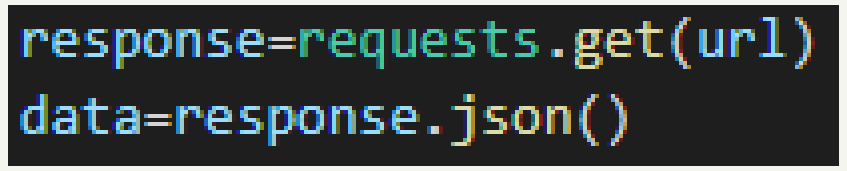
```
import requests
import pandas as pd
from sqlalchemy import create_engine
import mysql.connector
```

2. Create a MySQL Engine



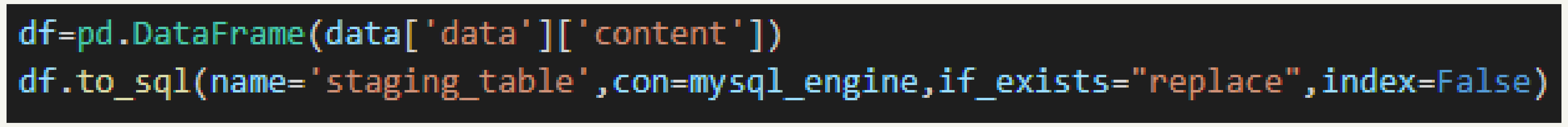
```
mysql_engine=create_engine(f"mysql+mysqlconnector://user:password@34.69.56.212:3366/db")
```

3. Get the API data



```
response=requests.get(url)
data=response.json()
```

4. Store the data in the MySQL Database



```
df=pd.DataFrame(data['data']['content'])
df.to_sql(name='staging_table',con=mysql_engine,if_exists="replace",index=False)
```

DigitalSkola

Project Step

MySQL to PostgreSQL

1. Creating MySQL and PostgreSQL engine

```
postgres_engine=create_engine(f"postgresql+psycopg2://postgres:postgres@34.69.56.212:5433/postgres")
mysql_engine=create_engine(f"mysql+mysqlconnector://user:password@34.69.56.212:3366/db")
```

2. Membaca data dari staging table di MySQL dan disimpan dalam dataframe

```
df=pd.read_sql(sql='staging_table',con=mysql_engine)
```

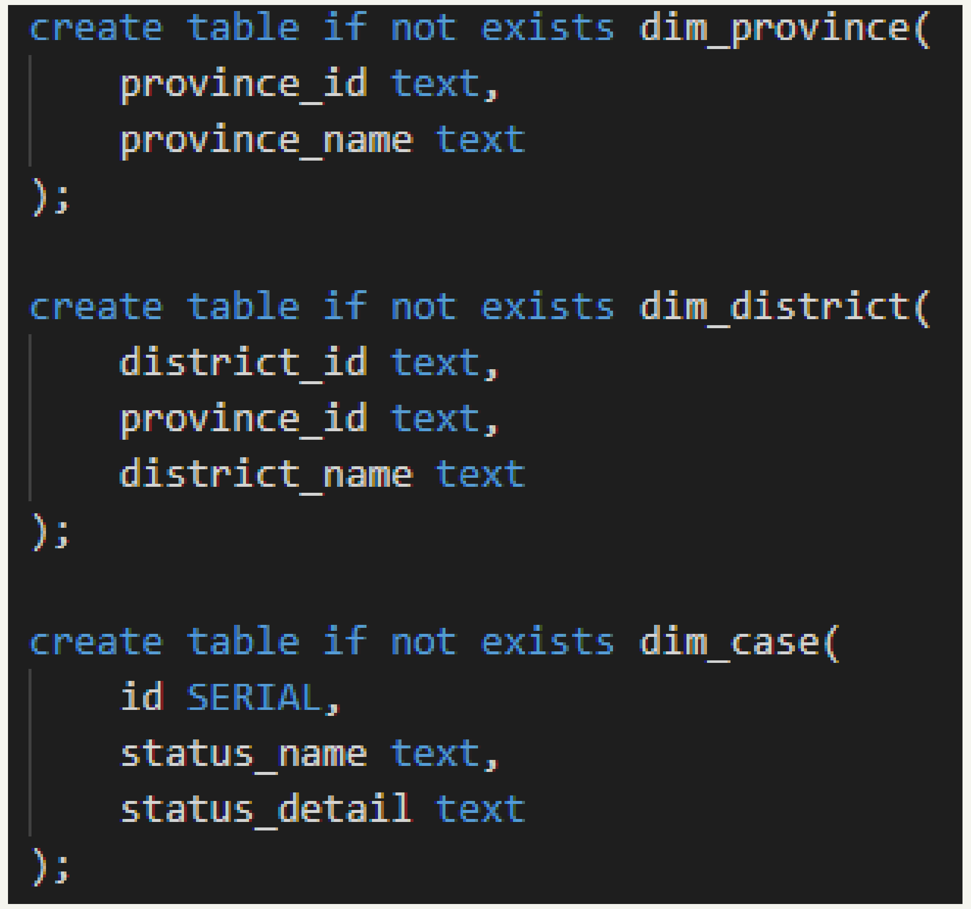
3. Menyimpan dataframe di PostgreSQL

```
df.to_sql(name='warehouse_table',con=postgres_engine,if_exists="replace",index=False)
```

Project Step

Create Table Task

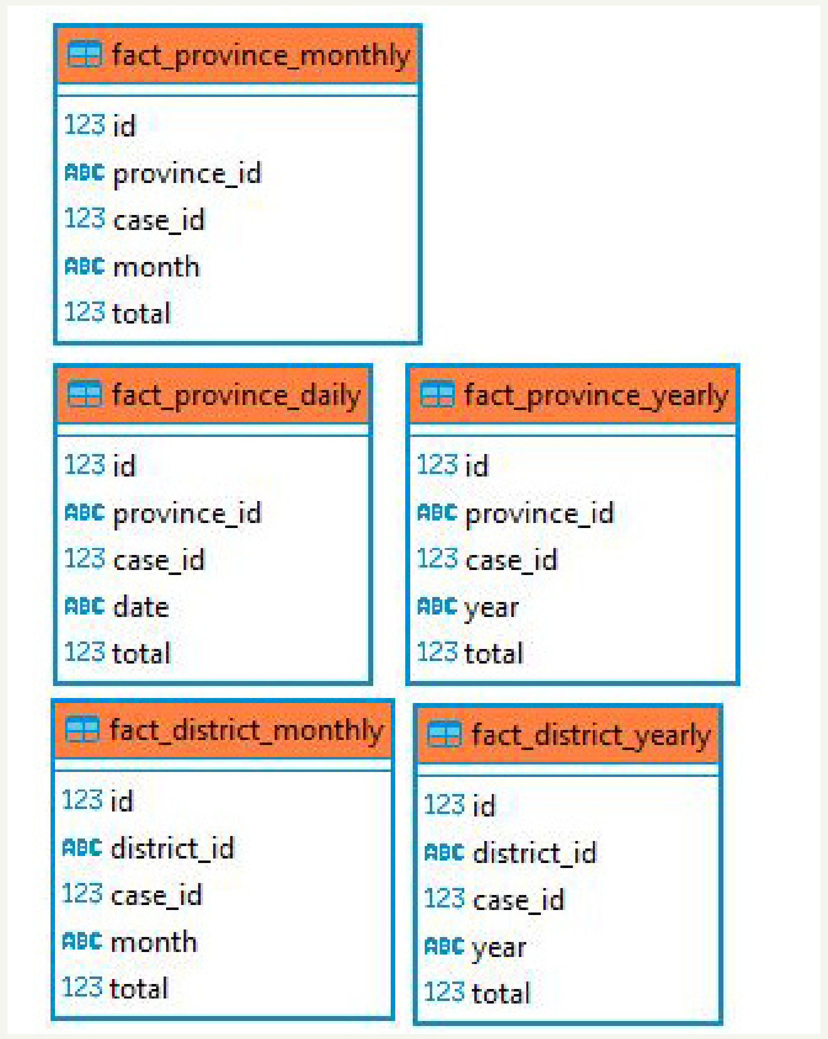
1. Membuat table menggunakan SQL



```
create table if not exists dim_province(
    province_id text,
    province_name text
);

create table if not exists dim_district(
    district_id text,
    province_id text,
    district_name text
);

create table if not exists dim_case(
    id SERIAL,
    status_name text,
    status_detail text
);
```

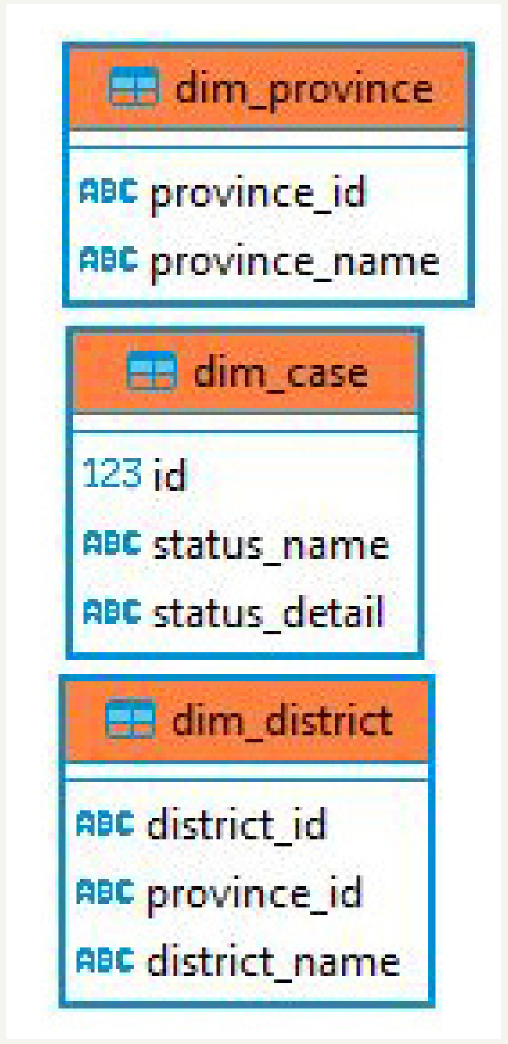
Fact & Dim Table

2. Fact Table



```
fact_province_monthly
123 id
ABC province_id
123 case_id
ABC month
123 total

fact_province_daily
123 id
ABC province_id
123 case_id
ABC date
123 total

fact_province_yearly
123 id
ABC province_id
123 case_id
ABC year
123 total

fact_district_monthly
123 id
ABC district_id
123 case_id
ABC month
123 total

fact_district_yearly
123 id
ABC district_id
123 case_id
ABC year
123 total
```

3. Dim Table

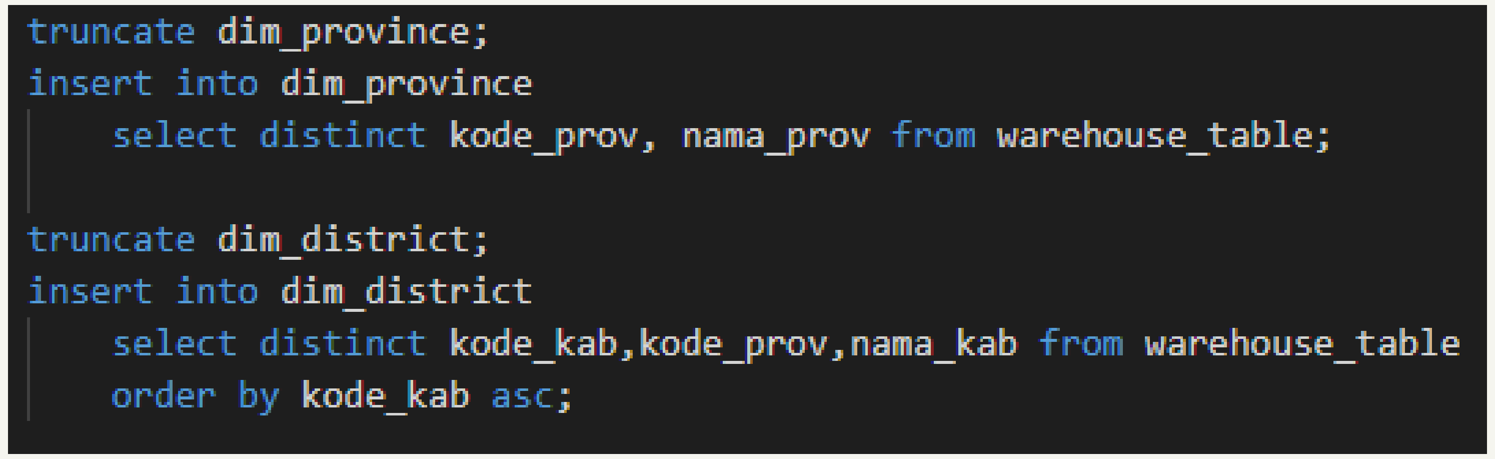


```
dim_province
ABC province_id
ABC province_name

dim_case
123 id
ABC status_name
ABC status_detail

dim_district
ABC district_id
ABC province_id
ABC district_name
```

Project Step

Populate Dim Table

1. Mengisi dim table sesuai dengan table yang sudah dibuat

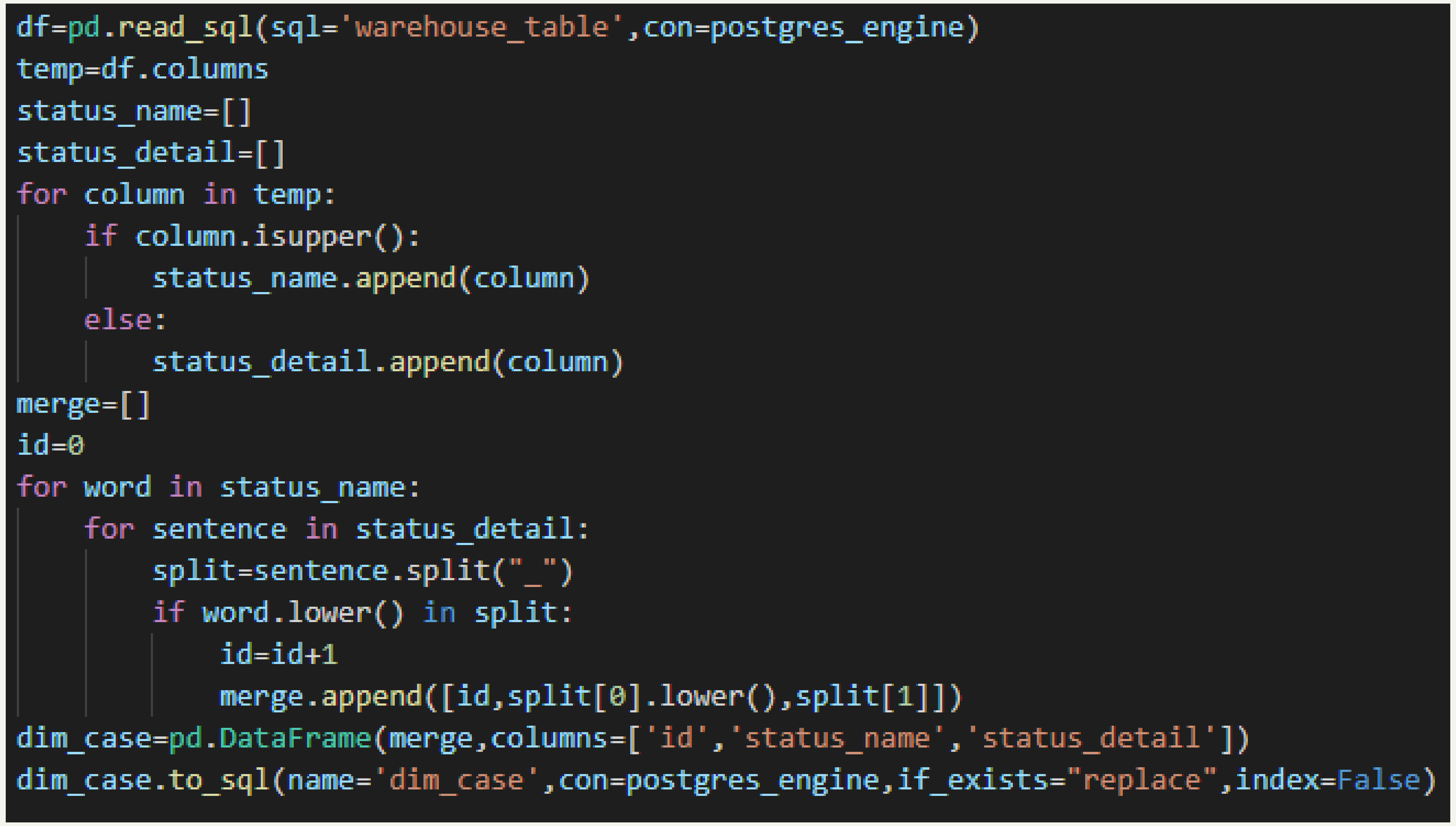


```sql
truncate dim_province;
insert into dim_province
    select distinct kode_prov, nama_prov from warehouse_table;

truncate dim_district;
insert into dim_district
    select distinct kode_kab,kode_prov,nama_kab from warehouse_table
    order by kode_kab asc;
```

Project Step

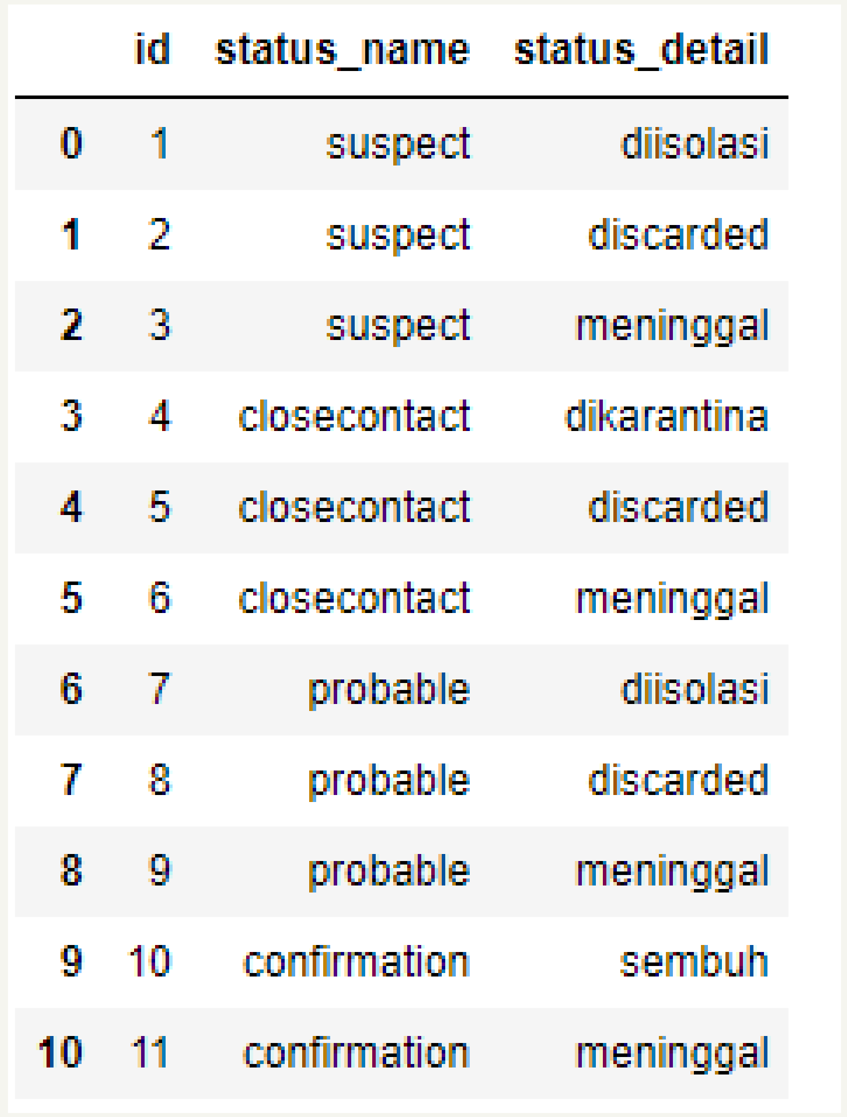
Dim Case Task

1. Membuat tabel untuk kategorisasi status covid



```python
df=pd.read_sql(sql='warehouse_table',con=postgres_engine)
temp=df.columns
status_name=[]
status_detail=[]
for column in temp:
    if column.isupper():
        status_name.append(column)
    else:
        status_detail.append(column)
merge=[]
id=0
for word in status_name:
    for sentence in status_detail:
        split=sentence.split("_")
        if word.lower() in split:
            id=id+1
            merge.append([id,split[0].lower(),split[1]])
dim_case=pd.DataFrame(merge,columns=['id','status_name','status_detail'])
dim_case.to_sql(name='dim_case',con=postgres_engine,if_exists="replace",index=False)
```

2. Hasil kategorisasi status name dan detail



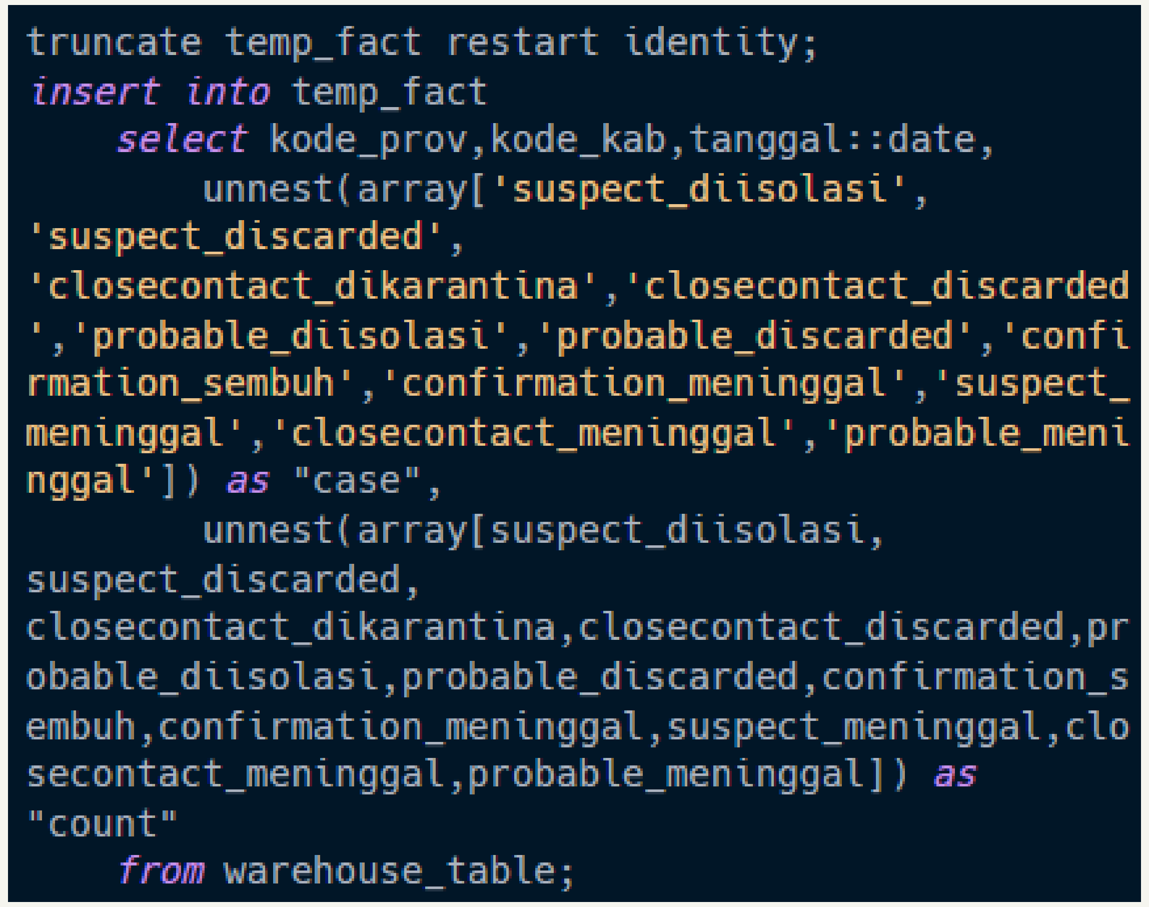
	id	status_name	status_detail
0	1	suspect	diisolasi
1	2	suspect	discarded
2	3	suspect	meninggal
3	4	closecontact	dikarantina
4	5	closecontact	discarded
5	6	closecontact	meninggal
6	7	probable	diisolasi
7	8	probable	discarded
8	9	probable	meninggal
9	10	confirmation	sembuh
10	11	confirmation	meninggal

DigitalSkola

Project Step

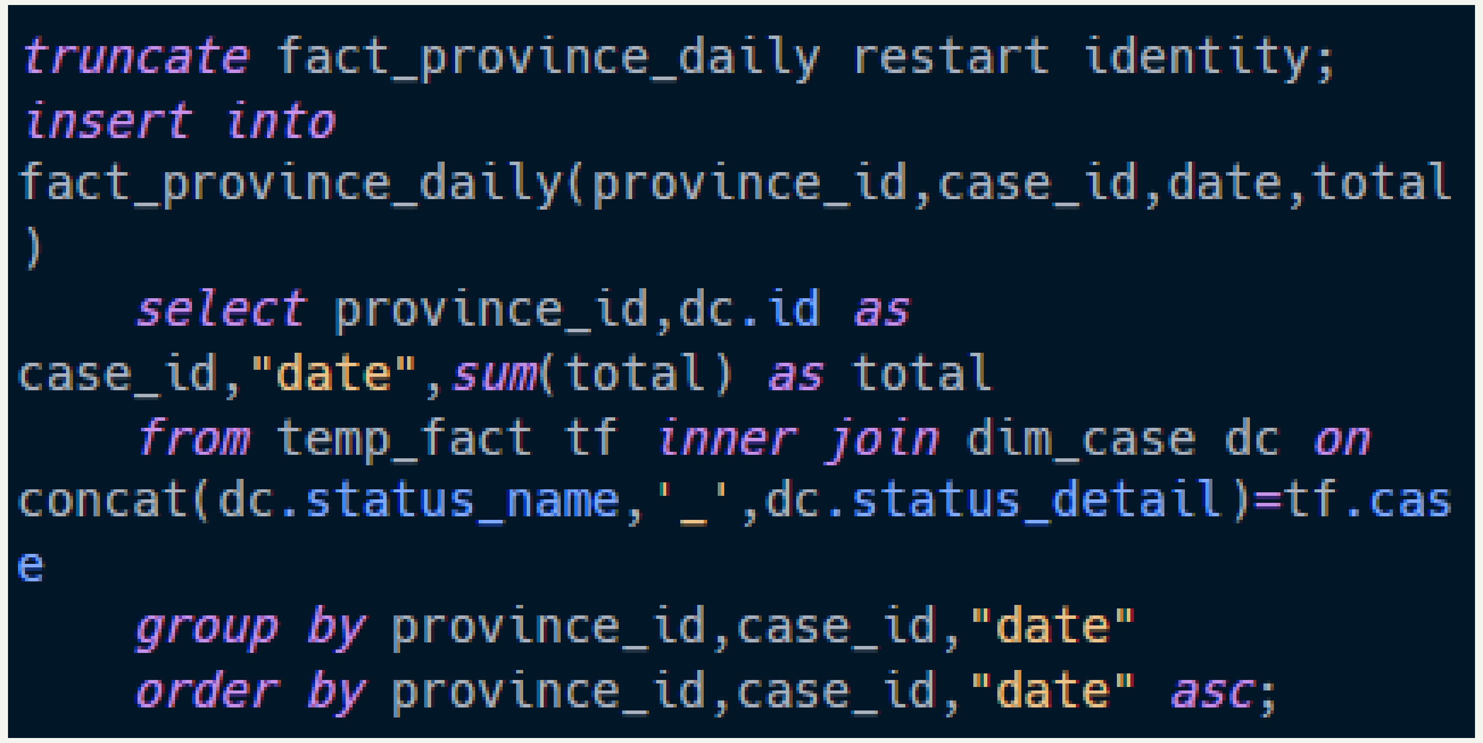
Populate Fact Table

1. Membuat temp_fact table untuk membantu agregasi



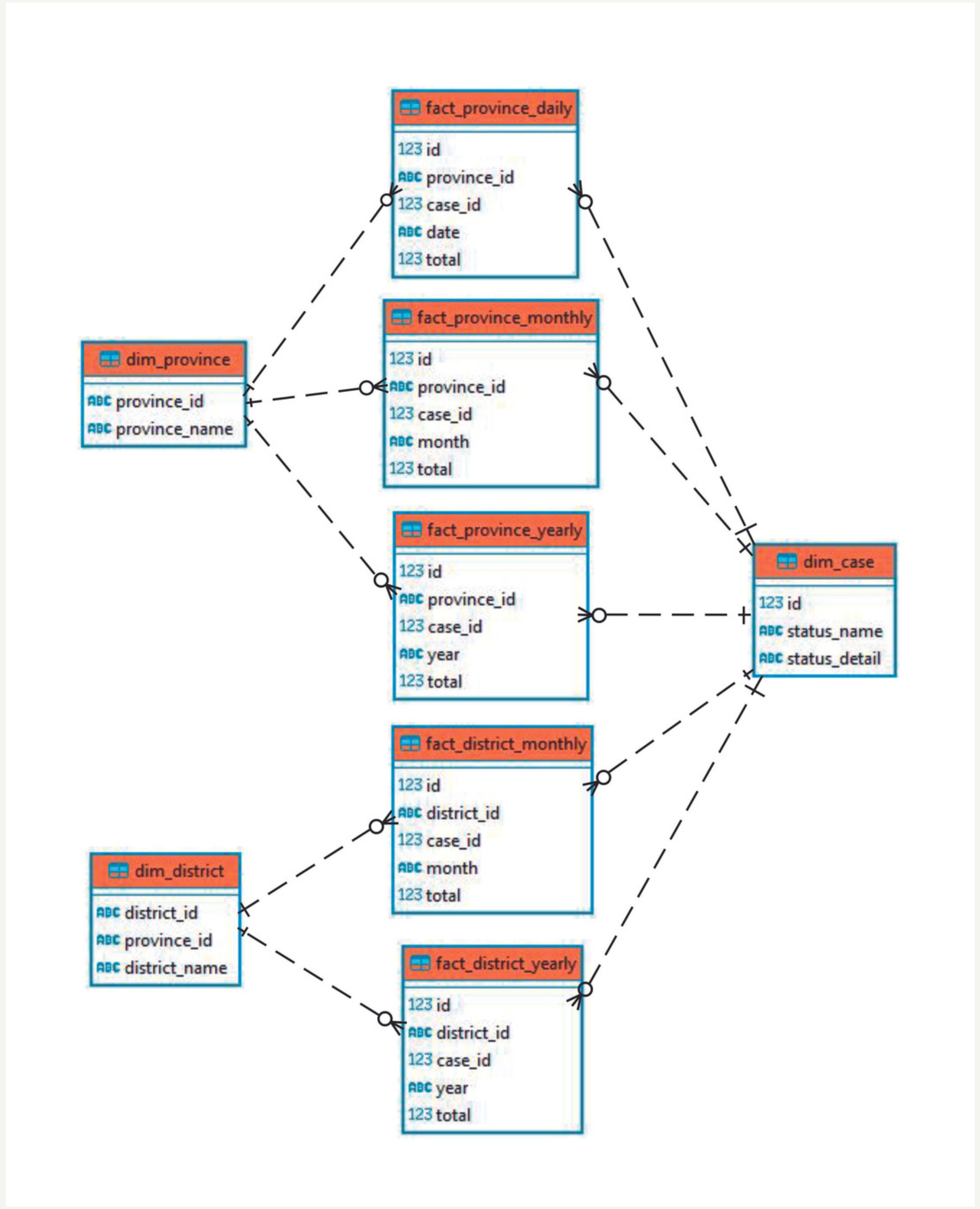
```
truncate temp_fact restart identity;
insert into temp_fact
    select kode_prov,kode_kab,tanggal::date,
        unnest(array['suspect_diisolasi',
'suspect_discarded',
'closecontact_dikarantina','closecontact_discarded
','probable_diisolasi','probable_discarded','confi
rmation_sembuh','confirmation_meninggal','suspect_
meninggal','closecontact_meninggal','probable_meni
nggal']) as "case",
        unnest(array[suspect_diisolasi,
suspect_discarded,
closecontact_dikarantina,closecontact_discarded,pr
obable_diisolasi,probable_discarded,confirmation_s
embuh,confirmation_meninggal,suspect_meninggal,clo
secontact_meninggal,probable_meninggal]) as
"count"
    from warehouse_table;
```

2. Membuat agregasi sesuai dengan fact table yang dibuat.
3. Memodifikasi jenis id dan waktu sesuai dengan kebutuhan



```
truncate fact_province_daily restart identity;
insert into
fact_province_daily(province_id,case_id,date,total
)
    select province_id,dc.id as
case_id,"date",sum(total) as total
    from temp_fact tf inner join dim_case dc on
concat(dc.status_name,'_',dc.status_detail)=tf.cas
e
    group by province_id,case_id,"date"
    order by province_id,case_id,"date" asc;
```

ERD



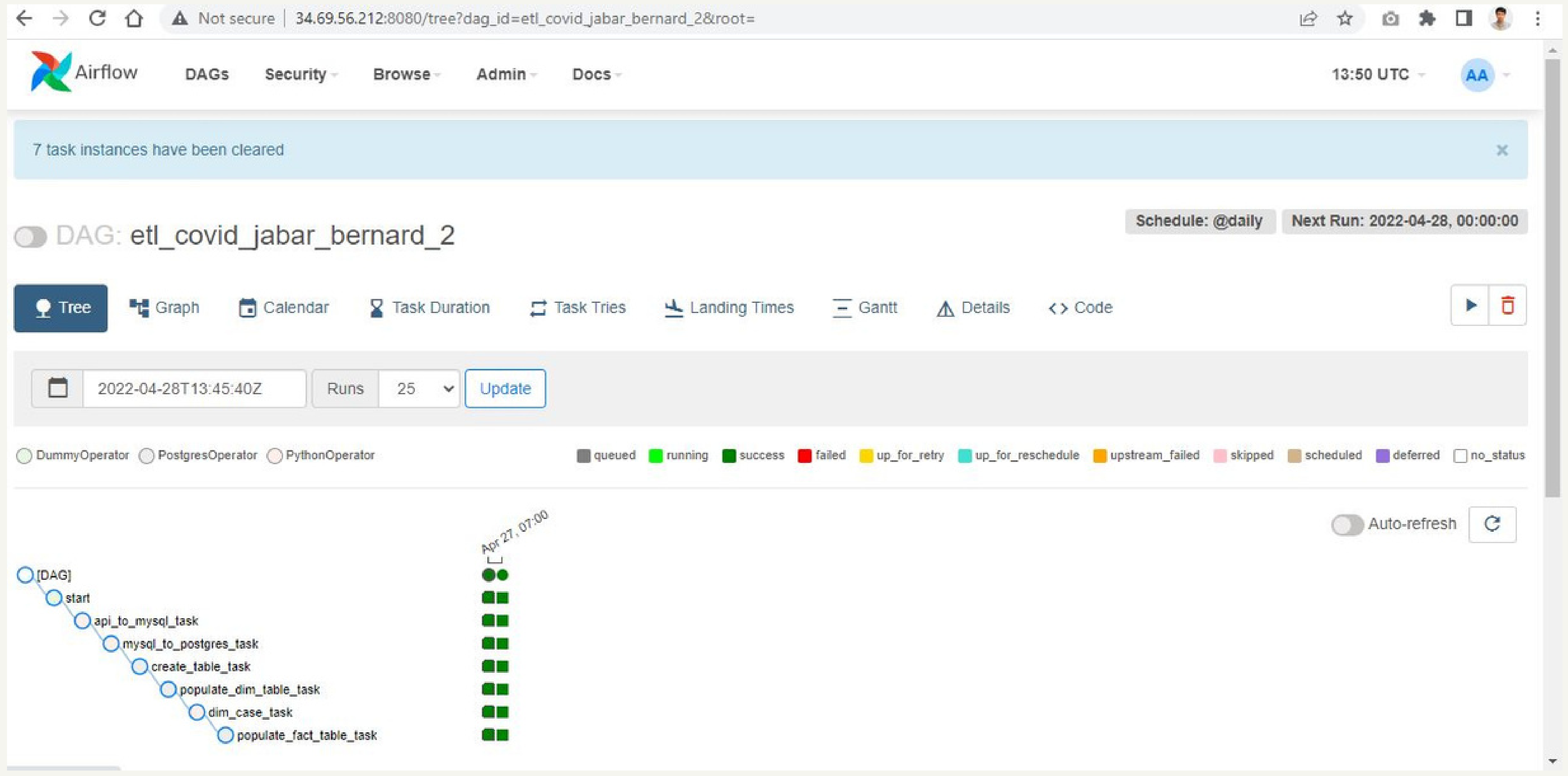
Fact Table (Province):
- Fact province daily
- Fact province monthly
- Fact province daily

Fact Table (District)
- Fact district monthly
- Fact district daily

Dimension Table
- Dim Province
- Dim District
- Dim Case

AIRFLOW

Tree View


34.69.56.212:8080/tree?dag_id=etl_covid_jabar_bernard_2&root=

7 task instances have been cleared

DAG: etl_covid_jabar_bernard_2

Schedule: @daily Next Run: 2022-04-28, 00:00:00

2022-04-28T13:45:40Z Runs 25 Update

DummyOperator PostgresOperator PythonOperator

queued running success failed up_for_retry up_for_reschedule upstream_failed skipped scheduled deferred no_status

Apr 27, 07:00

[DAG]
start
api_to_mysql_task
mysql_to_postgres_task
create_table_task
populate_dim_table_task
dim_case_task
populate_fact_table_task

Details

Owner: Bernard

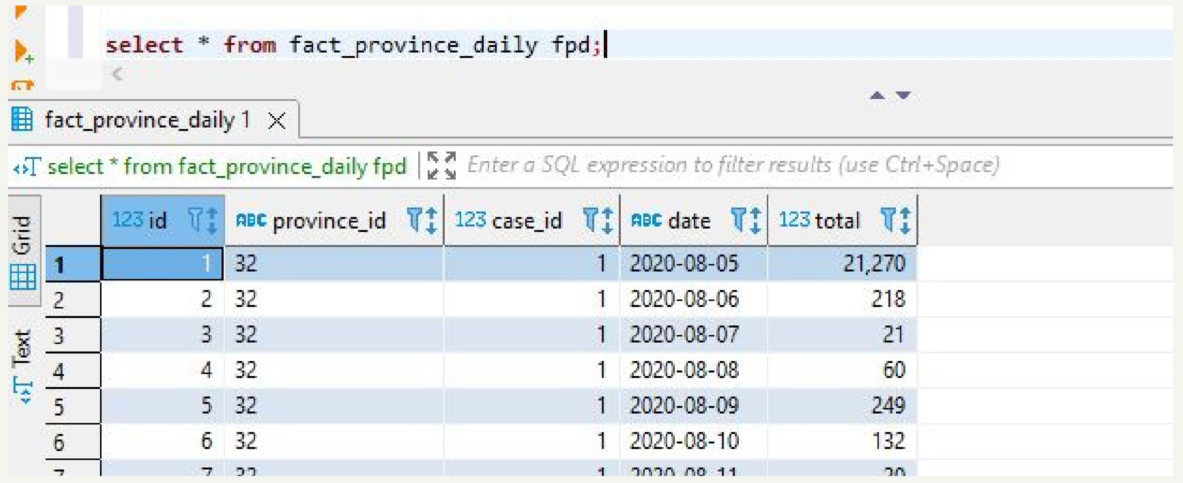
Email: bernardevankanigara@gmail.com

Start date: 2022, 5, 1

Schedule interval: Daily

PostgreSQL

Table



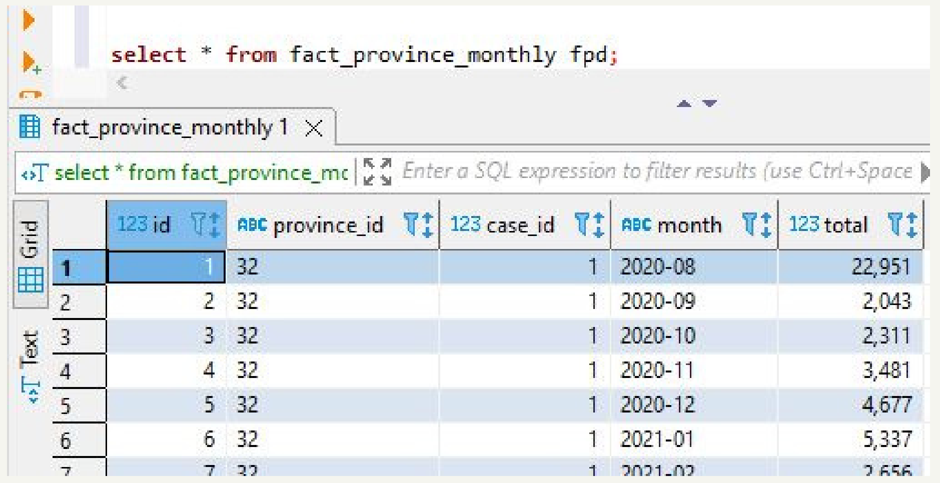
```
select * from fact_province_daily fpd;
```

fact_province_daily 1

select * from fact_province_daily fpd | Enter a SQL expression to filter results (use Ctrl+Space)

	id	province_id	case_id	date	total
1	1	32	1	2020-08-05	21,270
2	2	32	1	2020-08-06	218
3	3	32	1	2020-08-07	21
4	4	32	1	2020-08-08	60
5	5	32	1	2020-08-09	249
6	6	32	1	2020-08-10	132

Fact province daily



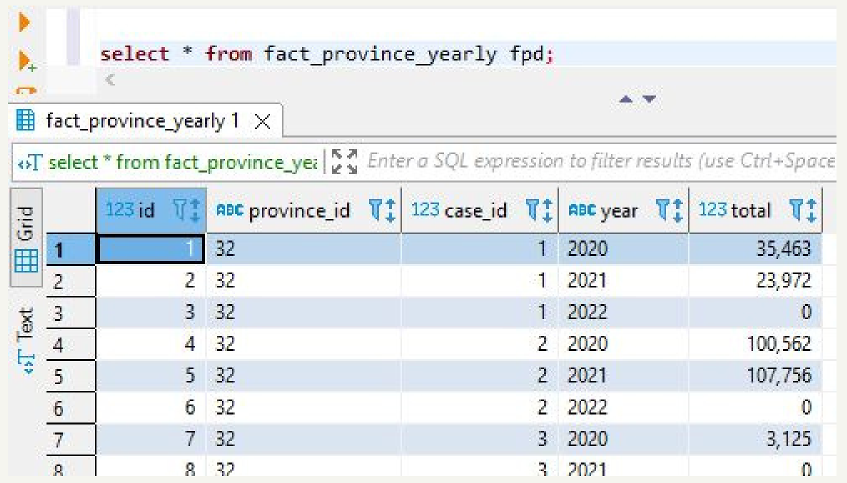
```
select * from fact_province_monthly fpd;
```

fact_province_monthly 1

select * from fact_province_mc | Enter a SQL expression to filter results (use Ctrl+Space

	id	province_id	case_id	month	total
1	1	32	1	2020-08	22,951
2	2	32	1	2020-09	2,043
3	3	32	1	2020-10	2,311
4	4	32	1	2020-11	3,481
5	5	32	1	2020-12	4,677
6	6	32	1	2021-01	5,337

Fact province monthly



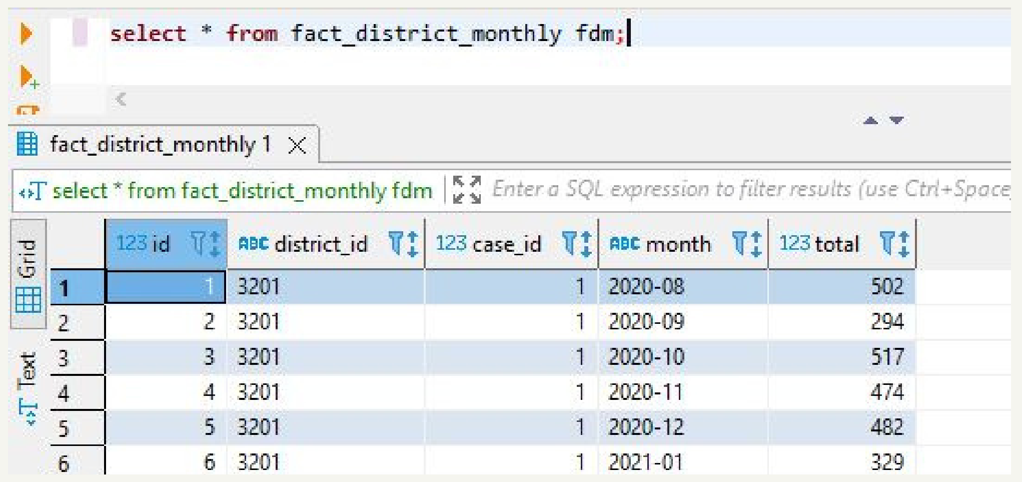
```
select * from fact_province_yearly fpd;
```

fact_province_yearly 1

select * from fact_province_yea | Enter a SQL expression to filter results (use Ctrl+Space

	id	province_id	case_id	year	total
1	1	32	1	2020	35,463
2	2	32	1	2021	23,972
3	3	32	1	2022	0
4	4	32	2	2020	100,562
5	5	32	2	2021	107,756
6	6	32	2	2022	0
7	7	32	3	2020	3,125

Fact province yearly



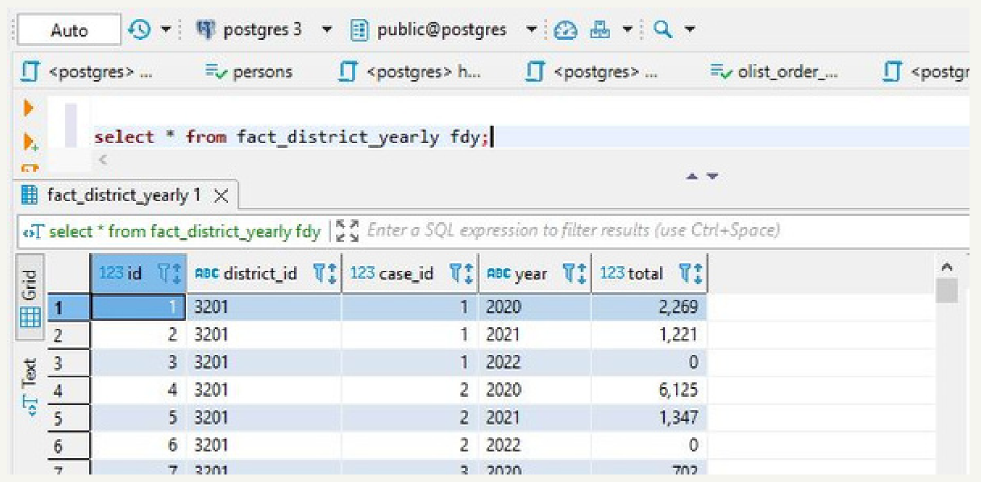
```
select * from fact_district_monthly fdm;
```

fact_district_monthly 1

select * from fact_district_monthly fdm | Enter a SQL expression to filter results (use Ctrl+Space

	id	district_id	case_id	month	total
1	1	3201	1	2020-08	502
2	2	3201	1	2020-09	294
3	3	3201	1	2020-10	517
4	4	3201	1	2020-11	474
5	5	3201	1	2020-12	482
6	6	3201	1	2021-01	329

Fact district monthly



```
select * from fact_district_yearly fdy;
```

fact_district_yearly 1

select * from fact_district_yearly fdy | Enter a SQL expression to filter results (use Ctrl+Space)

	id	district_id	case_id	year	total
1	1	3201	1	2020	2,269
2	2	3201	1	2021	1,221
3	3	3201	1	2022	0
4	4	3201	2	2020	6,125
5	5	3201	2	2021	1,347
6	6	3201	2	2022	0

Fact district yearly

Terima kasih