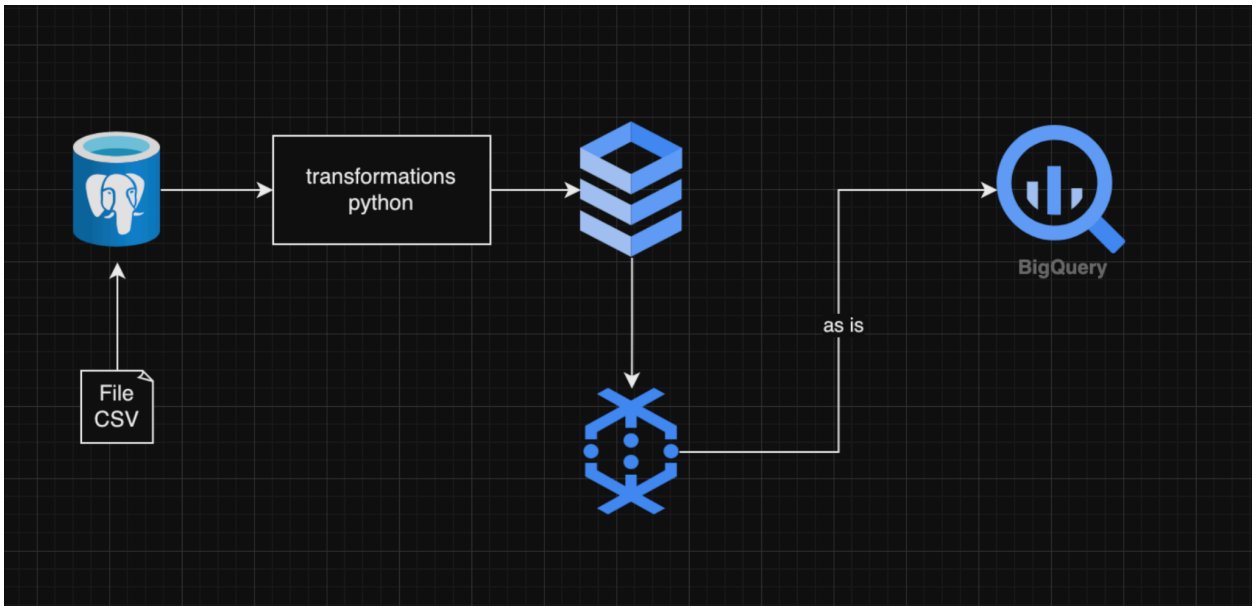
Data Engineer – Data Fellowship 12 IYKRA

Assignment 5

Alya Mutiara Firdausyi

Instruction

Task



1. Buatlah flow seperti pada diagram diatas.
2. Alur flow adalah sebagai berikut:
    a. Data dari csv pada folder weekly assignment di load pada postgresQL local machine.
    b. Kemudian lakukan transformasi, filtering, enriching data, dan berikan alasanya mengapa datanya harus dilakukan hal tersebut.
    c. Setelah datanya bersih, atau sesuai yang dikehendaki, lakukan load ke Cloud SQL di google cloud platform dengan nama table yang sama.
    d. Setelah datanyanya landing pada Cloud SQL, offload ke BigQuery secara as is atau apa adannya.
3. Di sini bebas akan membuat skenario seperti apa, mau pipelinenya di scheduling boleh tidak juga boleh.
4. Data yang digunakan adalah:
    a. Dataset banksim artificial → ada di folder

Output

Untuk di week 5 tugasnya sama seperti yang sudah temen temen kerjakan, namun semuanya didokumentasikan melalui github repo transformasi sampai settingan databasenya, data catalog nya, data modelnya. Kemudian invite dan buatkan akun sebagai viewer di gcp temen temen untuk melihat pipeline yang sudah temen temen buat, emailnya adalah sebagai berikut sasongkobgn@gmail.com

Challenge

Goal:

The task involves creating a data pipeline that ingests data from CSV files in a local folder, transforms, filters, and enriches the data, loads the processed data into Cloud SQL, and then offloads the data to BigQuery. The data pipeline can be scheduled or triggered manually.

Step:

1. Data Ingestion
   a. Raw data is collected from Data Fellowship LMS in a CSV format.
   b. A dockerized PostgreSQL is prepared for ingesting the first layer of the database. The latest version from the official docker hub image is used with details as follows:
      i. Host: localhost
      ii. Port: 5432
      iii. Root user: postgres
      iv. Password: postgres
   c. Pull the PostgreSQL image using this command.

      ```
      docker pull postgres:latest
      ```

      Then run the image.

      ```
      docker run --name dockerized-psql -v /Users/alya/Documents/GitHub/DataFellowship12-iykra/:/app -p 5432:5432 -e POSTGRES_PASSWORD=postgres -d postgres
      ```

      ```
      (myenv) alya@192 ~/D/G/DataFellowship12-iykra (main)> docker run --name local-psql -v /Users/alya/Documents/GitHub/DataFellowship12-iykra/:/app -p 5432:5432 -e POSTGRES_PASSWORD=postgres -d postgres
      58e8747575822941d04c918bd3f3070a98b32a9e18e44e960934ffae50e420ba
      (myenv) alya@192 ~/D/G/DataFellowship12-iykra (main)> docker ps
      CONTAINER ID   IMAGE      COMMAND                  CREATED         STATUS         PORTS                    NAMES
      58e874757582   postgres   "docker-entrypoint.s…"   6 seconds ago   Up 5 seconds   0.0.0.0:5432->5432/tcp   local-psql
      ```

      To check whether the image is running correctly, run this:

```
docker exec -it dockerized-psql psql -U postgres
```

```
(myenv) alya@192 ~/D/G/DataFellowship12-iykra (main)> docker exec -it local-psql psql -U postgres
psql (16.2 (Debian 16.2-1.pgdg120+2))
Type "help" for help.

postgres=# \l
                                                  List of databases
   Name    |  Owner   | Encoding | Locale Provider |  Collate   |   Ctype    | ICU Locale | ICU Rules |   Access privileges
-----------+----------+----------+-----------------+------------+------------+------------+-----------+-----------------------
 postgres  | postgres | UTF8     | libc            | en_US.utf8 | en_US.utf8 |            |           |
 template0 | postgres | UTF8     | libc            | en_US.utf8 | en_US.utf8 |            |           | =c/postgres          +
           |          |          |                 |            |            |            |           | postgres=CTc/postgres
 template1 | postgres | UTF8     | libc            | en_US.utf8 | en_US.utf8 |            |           | =c/postgres          +
           |          |          |                 |            |            |            |           | postgres=CTc/postgres
(3 rows)

postgres=#
```

Create a new database for storing the data.

```
docker exec -it dockerized-psql psql -U postgres -c "CREATE DATABASE banksim_db;"
```

Create a user and grant all access to the user.

```
docker exec -it dockerized-psql psql -U postgres -c "CREATE USER banksim WITH ENCRYPTED PASSWORD 'banksim';"
```

```
docker exec -it dockerized-psql psql -U postgres -c "GRANT ALL PRIVILEGES ON DATABASE banksim_db TO banksim;"
```

```
(myenv) alya@192 ~/D/G/D/W/Assignment_1 (main)> docker exec -it local-psql psql -U postgres -c "CREATE USER banksim WITH ENCRYPTED PASSWORD 'banksim';"
CREATE ROLE
(myenv) alya@192 ~/D/G/D/W/Assignment_1 (main)> docker exec -it local-psql psql -U postgres -c "GRANT ALL PRIVILEGES ON DATABASE banksim_db TO banksim;"
GRANT
(myenv) alya@192 ~/D/G/D/W/Assignment_1 (main)> docker exec -it local-psql psql -U postgres -c "\l"
                                                  List of databases
    Name    |  Owner   | Encoding | Locale Provider |  Collate   |   Ctype    | ICU Locale | ICU Rules |   Access privileges
------------+----------+----------+-----------------+------------+------------+------------+-----------+-----------------------
 banksim_db | postgres | UTF8     | libc            | en_US.utf8 | en_US.utf8 |            |           | =Tc/postgres         +
            |          |          |                 |            |            |            |           | postgres=CTc/postgres+
            |          |          |                 |            |            |            |           | banksim=CTc/postgres
 postgres   | postgres | UTF8     | libc            | en_US.utf8 | en_US.utf8 |            |           |
 template0  | postgres | UTF8     | libc            | en_US.utf8 | en_US.utf8 |            |           | =c/postgres          +
            |          |          |                 |            |            |            |           | postgres=CTc/postgres
 template1  | postgres | UTF8     | libc            | en_US.utf8 | en_US.utf8 |            |           | =c/postgres          +
            |          |          |                 |            |            |            |           | postgres=CTc/postgres
(4 rows)
```

d.

e.

2. Data Transformation and Enrichment

   Data transformation was done in Jupyter Notebook using Python's libraries such as pandas and psycopg2.

   Please see the jupyter notebook file.

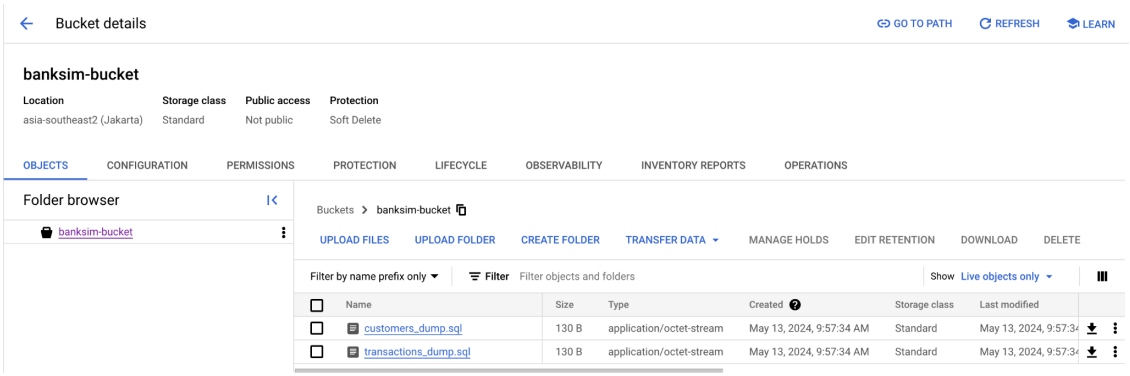
3. Data Loading to Cloud SQL
   a. In the PostgreSQL server, create an SQL dump file to be uploaded into the Cloud Storage using this command.

```
docker exec -it dockerized-psql -U banksim -d banksim_db -t customers --no-owner > customers_dump.sql
```

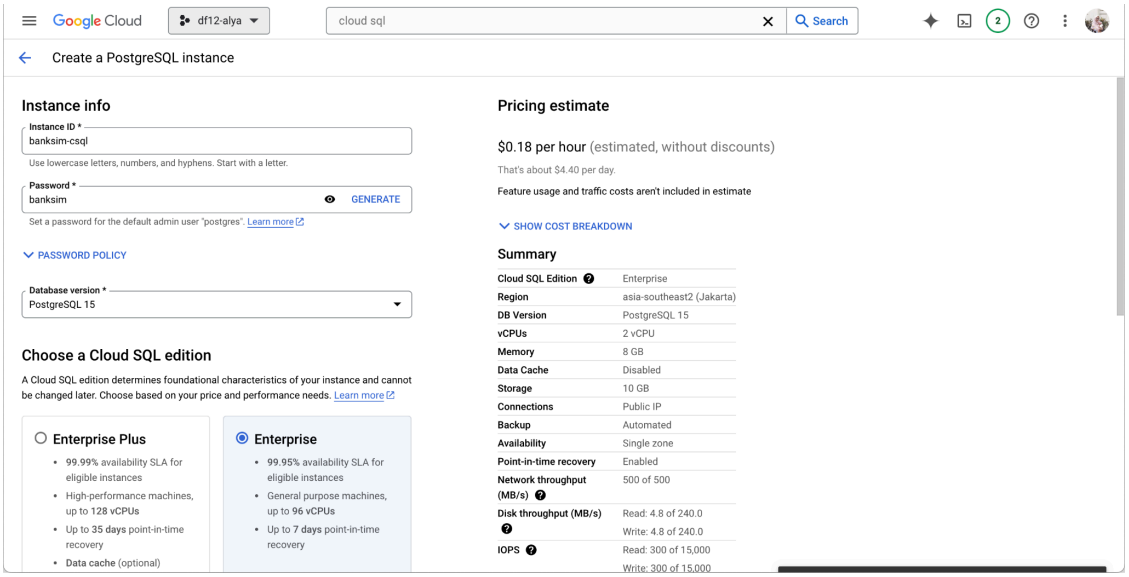
```
docker exec -it dockerized-psql -U banksim -d banksim_db -t transactions --no-owner > transactions_dump.sql
```

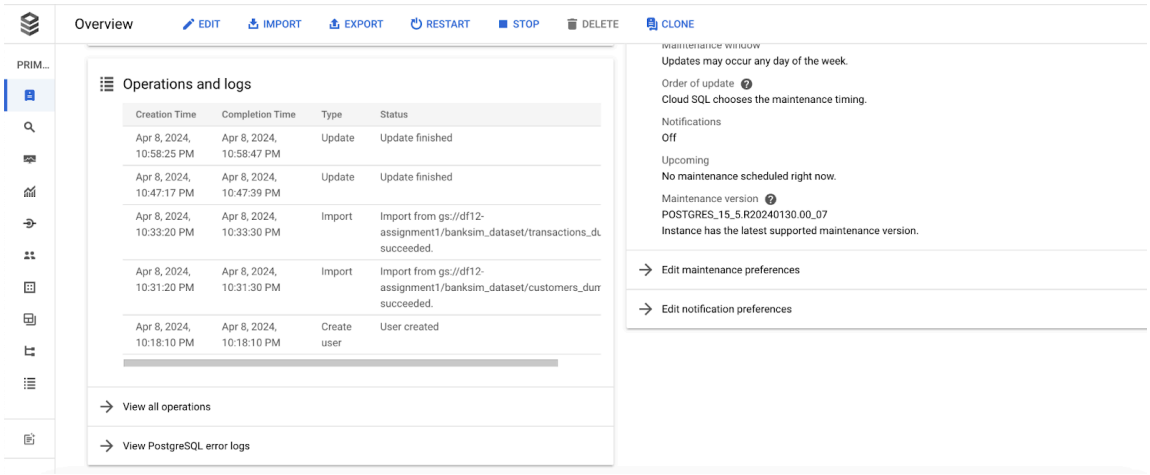
```
[alya@aaraitum ~/D/G/D/Assignment_5 (main)> docker exec -it dockerized-psql -U banksim -d banksim_db -t customers --no-owner > customers_dump.sql

What's next:
  Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug dockerized-psql
  Learn more at https://docs.docker.com/go/debug-cli/
[alya@aaraitum ~/D/G/D/Assignment_5 (main) [126]> docker exec -it dockerized-psql -U banksim -d banksim_db -t transactions --no-owner > transactions_dump.sql

What's next:
  Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug dockerized-psql
  Learn more at https://docs.docker.com/go/debug-cli/
[alya@aaraitum ~/D/G/D/Assignment_5 (main) [126]> ls
bs140513_032310.csv     bsNET140513_032310.csv  customers_dump.sql      transactions_dump.sql
alya@aaraitum ~/D/G/D/Assignment_5 (main)>
```

b. Then upload the dump into a Cloud Storage bucket.

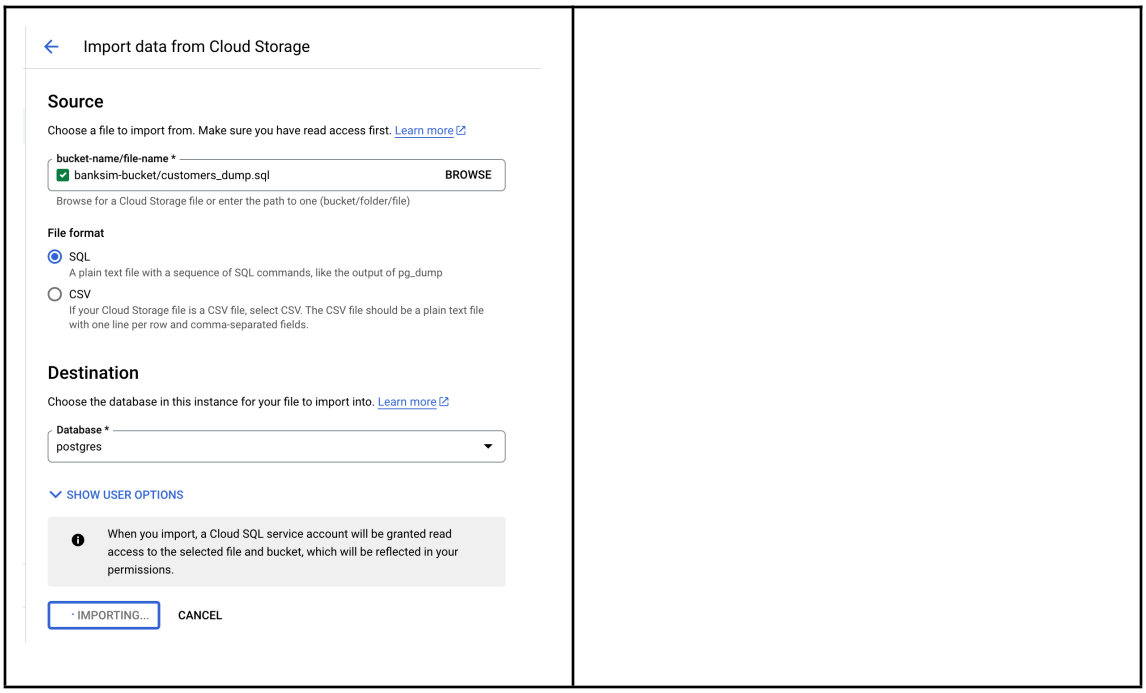


c. Create a Cloud SQL instance.



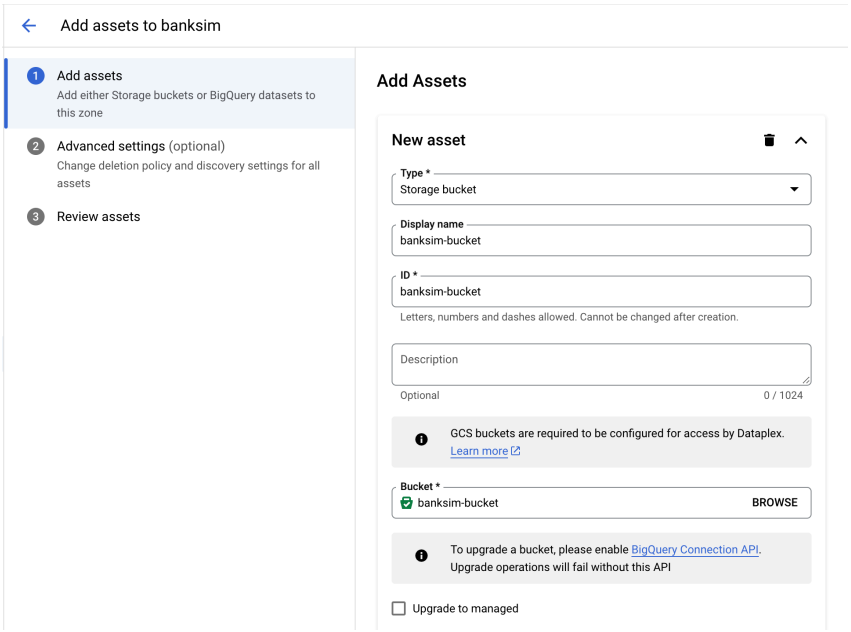


d. Import data from Cloud Storage to Cloud SQL.

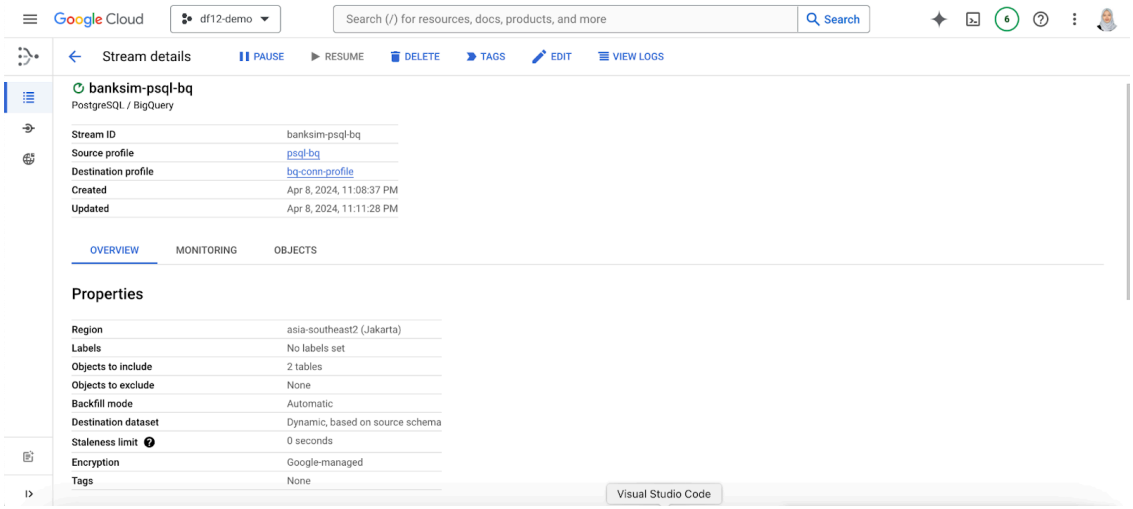


e.

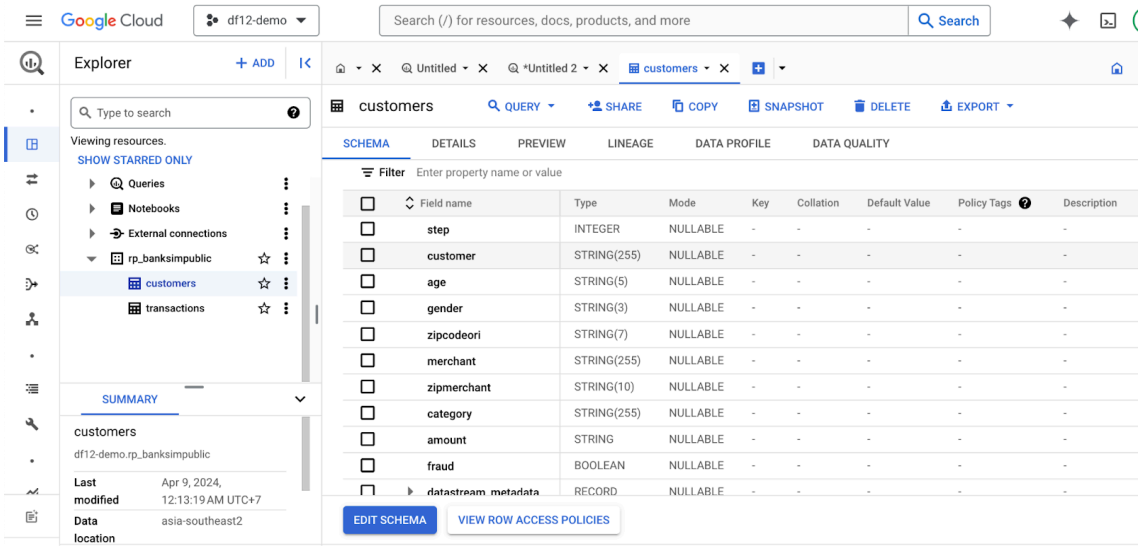
4. Data Lake



5. Data Offloading to BigQuery
   a. Create a stream configuration in Datastream and do the prerequisite in the Cloud SQL Studio to create the stream replication and stream publication. This way, Datastream will automatically ingest data to the BigQuery whenever they detect changes in the Cloud SQL database.



   b. After the stream is up and running, ensure that the dataset has appeared in BigQuery.



c. Finally, the data can be queried to do some analytical things.

6. Documentation and Sharing

The documentation of this task is placed in [this GitHub repository](#).