

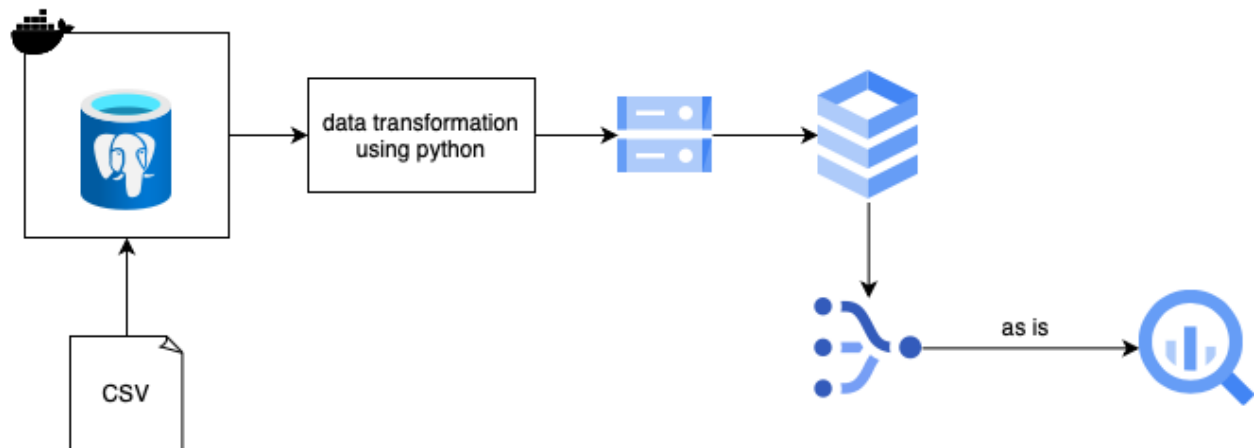
# Data Fellowship 12 - Assignment 1

Alya Mutiara Firdausyi

Flow:

1. Load CSV files from local computer to PostgreSQL using Docker.
2. Do data transformation, filtering, enriching and give reasons why the steps must be done.
3. After the data has been cleared out, load it to the Cloud SQL on the Google Cloud Platform with the same table name.
4. After the data exist in the Cloud SQL, pipe the data to BigQuery as it is.

Infrastructure:



Pipeline:

1. Raw data is collected from Data Fellowship LMS in a CSV format.
2. A dockerized PostgreSQL is prepared for ingesting the first layer of the database. The latest version from official docker hub image is used with detail as follows:
  - a. Host: localhost
  - b. Port: 5432
  - c. Root user: postgres
  - d. Password: postgres

Pull 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

```

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"

```

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)
```

3. Data transformation was done in Jupyter Notebook using Python's library such as pandas and pycpg2.

*Please see the jupyter notebook file.*

4. In PostgreSQL server, create 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
```

- Then upload the dump into a Cloud Storage bucket.

The screenshot shows the 'Bucket details' page for 'df12-demo-bucket'. The location is 'asia-southeast2 (Jakarta)', storage class is 'Standard', public access is 'Public to internet', and protection is 'None'. The 'OBJECTS' tab is selected, showing a 'Folder browser' on the left and a list of objects on the right. The objects list includes 'Customers.csv', 'Orders.csv', 'ProductCategory.csv', and 'Products.csv'.

Name	Size	Type	Created	Storage class	Last modified	Public a
Customers.csv	272.5 KB	text/csv	Apr 2, 2024, 2:19:42 PM	Standard	Apr 2, 2024, 2:19:42 PM	Public
Orders.csv	85.1 KB	text/csv	Apr 2, 2024, 2:20:27 PM	Standard	Apr 2, 2024, 2:20:27 PM	Public
ProductCategory.csv	174 B	text/csv	Apr 2, 2024, 2:20:54 PM	Standard	Apr 2, 2024, 2:20:54 PM	Public
Products.csv	2.3 KB	text/csv	Apr 2, 2024, 2:20:17 PM	Standard	Apr 2, 2024, 2:20:17 PM	Public

- Load the data into Cloud SQL by creating database and user beforehand.

The screenshot shows the 'Overview' page for a Cloud SQL instance named 'PRIM...'. The 'Operations and logs' section is expanded, showing a table of operations. The operations include updates and imports, all of which were successful. The 'Maintenance window' section on the right indicates that updates may occur any day of the week, and the instance has the latest supported maintenance version.

Creation Time	Completion Time	Type	Status
Apr 8, 2024, 10:58:25 PM	Apr 8, 2024, 10:58:47 PM	Update	Update finished
Apr 8, 2024, 10:47:17 PM	Apr 8, 2024, 10:47:39 PM	Update	Update finished
Apr 8, 2024, 10:33:20 PM	Apr 8, 2024, 10:33:30 PM	Import	Import from gs://df12-assignment1/banksim_dataset/transactions_dk succeeded.
Apr 8, 2024, 10:31:20 PM	Apr 8, 2024, 10:31:30 PM	Import	Import from gs://df12-assignment1/banksim_dataset/customers_durr succeeded.
Apr 8, 2024, 10:18:10 PM	Apr 8, 2024, 10:18:10 PM	Create user	User created

- 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.

Google Cloud df12-demo Search (/) for resources, docs, products, and more

Stream details PAUSE RESUME DELETE TAGS EDIT VIEW LOGS

**banksim-psql-bq**  
PostgreSQL / BigQuery

Stream ID banksim-psql-bq  
Source profile [psql-bq](#)  
Destination profile [bq-conn-profile](#)  
Created Apr 8, 2024, 11:08:37 PM  
Updated Apr 8, 2024, 11:11:28 PM

OVERVIEW MONITORING OBJECTS

**Properties**

Region asia-southeast2 (Jakarta)  
Labels No labels set  
Objects to include 2 tables  
Objects to exclude None  
Backfill mode Automatic  
Destination dataset Dynamic, based on source schema  
Staleness limit 0 seconds  
Encryption Google-managed  
Tags None

Visual Studio Code

8. After the stream is up and running, ensure that the dataset had appeared in BigQuery.

Google Cloud df12-demo Search (/) for resources, docs, products, and more

Explorer + ADD

Viewing resources. SHOW STARRED ONLY

- Queries
- Notebooks
- External connections
- rp\_banksimpbublic
  - customers
  - transactions

SUMMARY

customers  
df12-demo\_rp\_banksimpbublic  
Last modified Apr 9, 2024, 12:13:19 AM UTC+7  
Data location asia-southeast2

customers QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW LINEAGE DATA PROFILE DATA QUALITY

Filter Enter property name or value

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
step	INTEGER	NULLABLE	-	-	-	-	-
customer	STRING(255)	NULLABLE	-	-	-	-	-
age	STRING(5)	NULLABLE	-	-	-	-	-
gender	STRING(3)	NULLABLE	-	-	-	-	-
zipcodeori	STRING(7)	NULLABLE	-	-	-	-	-
merchant	STRING(255)	NULLABLE	-	-	-	-	-
zipmerchant	STRING(10)	NULLABLE	-	-	-	-	-
category	STRING(255)	NULLABLE	-	-	-	-	-
amount	STRING	NULLABLE	-	-	-	-	-
fraud	BOOLEAN	NULLABLE	-	-	-	-	-
datastream_metadata	RECORD	NULLABLE	-	-	-	-	-

EDIT SCHEMA VIEW ROW ACCESS POLICIES

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