After making docker up

compose up -d postgres

PS C:\fraud\_bd> docker compose exec spark bash -lc "spark-submit --version"

Welcome to

\_\_\_\_ \_\_

/ \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_

\_\ \/ \_ \/ \_ `/ \_\_/ '\_/

/\_\_\_/ .\_\_/\\_,\_/\_/ /\_/\\_\ version 3.4.0

/\_/

Using Scala version 2.12.17, OpenJDK 64-Bit Server VM, 17.0.7

Branch HEAD

Compiled by user xinrong.meng on 2023-04-07T02:18:01Z

Revision 87a5442f7ed96b11051d8a9333476d080054e5a0

Url https://github.com/apache/spark

Type --help for more information.

PS C:\fraud\_bd> docker compose run --rm app

With command:

PS C:\fraud\_bd> docker-compose exec spark python work/generate\_data.py

Database tables created successfully.

Users and payment methods populated.

5000 transaction records were inserted successfully.

PS C:\fraud\_bd> docker-compose exec postgres\_fraud psql -U root -d fraud\_detect

service "postgres\_fraud" is not running

PS C:\fraud\_bd> docker-compose exec postgres psql -U root -d fraud\_detect

psql (13.22 (Debian 13.22-1.pgdg13+1))

Type "help" for help.

fraud\_detect=# \dt

List of relations

Schema | Name | Type | Owner

--------+-----------------+-------+-------

public | payment\_methods | table | root

public | transactions | table | root

public | users | table | root

(3 rows)

fraud\_detect=# SELECT count(\*) FROM transactions;

count

-------

5000

**Data Pipeline Output Locations**

1. **PostgreSQL Database:** The generate\_data.py script populates your local PostgreSQL database with raw data. You've already confirmed this step works.
2. **Local Filesystem (Bronze & Silver Layers):**
   * The export\_data.py script takes data from PostgreSQL and writes it as **CSV files** to your local directory at app/output. These are your **Bronze** layer files.
   * The transform\_data.py script reads those Bronze CSVs, processes them, and saves the output as a **Parquet file** in a new directory, app/output/silver/flights\_silver. This is your **Silver** layer.
3. **Snowflake Database (Gold Layer):**
   * The load\_to\_snowflake.py script reads the Silver Parquet file and loads it into a table in Snowflake called **DRONE\_FLIGHTS\_GOLD**.
   * The create\_kpis.py script reads from the DRONE\_FLIGHTS\_GOLD table and writes the final KPI data to a new table in Snowflake called **PILOT\_PERFORMANCE\_KPIS**.

(1 row)

* If you only **added a volume**, you *can sometimes* restart just the affected service:

docker compose stop app

docker compose rm -f app

docker compose up -d app

1. docker compose down
2. docker compose up -d
3. docker ps
4. docker compose exec spark bash -lc "python -V; pyspark --version; ls -la /home/jovyan/work/jars; env | grep SNOWFLAKE"

or

1. docker compose exec spark bash -lc "python -V; pyspark --version; ls -la /home/jovyan/work/jars; env | grep -E '^SNOWFLAKE\_' || true"
2. docker compose exec spark python /home/jovyan/work/generate\_data.py
3. docker compose exec spark spark-submit --jars "/home/jovyan/work/jars/spark-snowflake\_2.12-2.14.0-spark\_3.4.jar,/home/jovyan/work/jars/snowflake-jdbc-3.14.4.jar" /home/jovyan/work/ingest\_to\_snowflake.py
4. docker compose exec spark spark-submit --jars "/home/jovyan/work/jars/spark-snowflake\_2.12-2.14.0-spark\_3.4.jar,/home/jovyan/work/jars/snowflake-jdbc-3.14.4.jar" /home/jovyan/work/transform\_data.py
5. docker compose exec spark spark-submit --jars "/home/jovyan/work/jars/spark-snowflake\_2.12-2.14.0-spark\_3.4.jar,/home/jovyan/work/jars/snowflake-jdbc-3.14.4.jar" /home/jovyan/work/create\_kpis.py