AMAZON_MUSICAL_INSTRUMENTS_PIPELINE

Development:

Step 1. Create Virtual environment:

```
python3 -m venv de_venv
de_venv\Scripts\activate.bat
```

Step 2. Create a docker-compose.yml file with postgres as service. Then run docker compose up command as:

>>docker compose -f docker-compose.yml up -d

Now since PostgreSQL is not a HTTP server - it doesn't serve web pages over HTTP. So we will be using **psql CLI**

Step 3. Source DB (Postgres) setup:

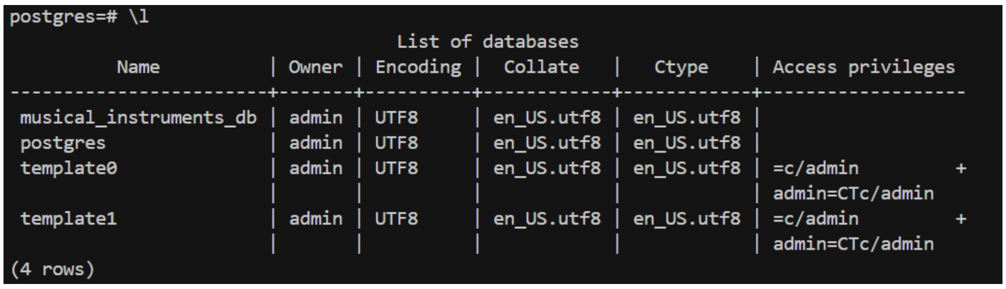
Do >> docker ps # To find out the container_id

docker exec -it <*your_container_id*> /bin/sh

Now we are in docker container

>> psql -U *admin* -d postgres [Since I used admin as postgres_user in my yml file, use accordingly]

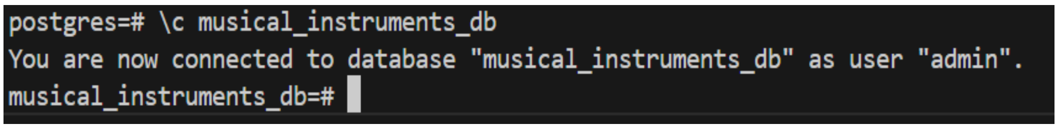
To get list of databases:



```
postgres=# \l
                                     List of databases
         Name          | Owner | Encoding |  Collate   |   Ctype    | Access privileges
-----------------------+-------+----------+------------+------------+-------------------
 musical_instruments_db | admin | UTF8     | en_US.utf8 | en_US.utf8 |
 postgres              | admin | UTF8     | en_US.utf8 | en_US.utf8 |
 template0             | admin | UTF8     | en_US.utf8 | en_US.utf8 | =c/admin         +
                       |       |          |            |            | admin=CTc/admin
 template1             | admin | UTF8     | en_US.utf8 | en_US.utf8 | =c/admin         +
                       |       |          |            |            | admin=CTc/admin
(4 rows)
```

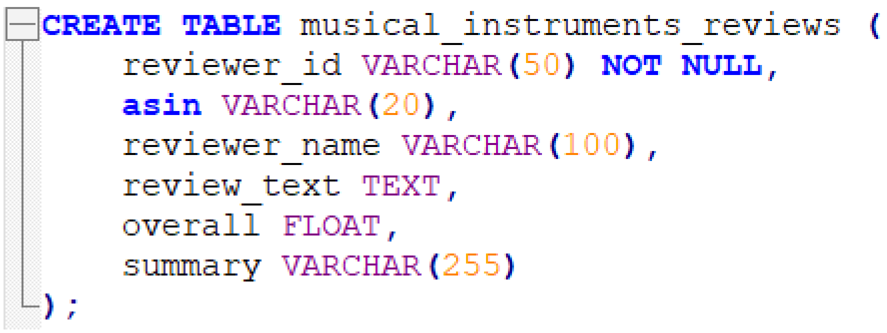
To get list of tables use: \dt

Use command: \c <your_db_name> to connect to database



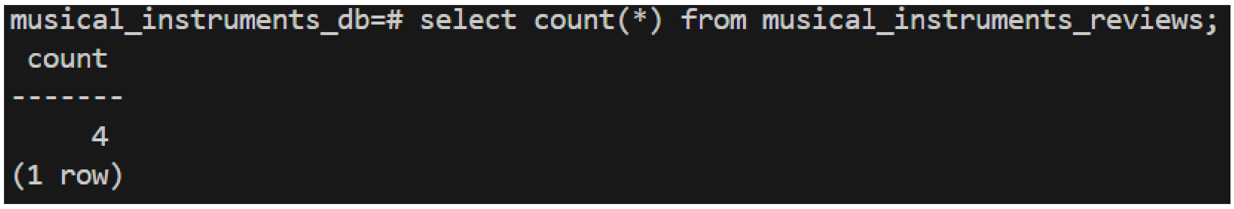
```
postgres=# \c musical_instruments_db
You are now connected to database "musical_instruments_db" as user "admin".
musical_instruments_db=#
```

Create the table (Best practice: Always use lowercase unquoted names when creating tables to avoid any issue) & insert some sample records (so that we can perform incremental load later)



```
CREATE TABLE musical_instruments_reviews (
    reviewer_id VARCHAR(50) NOT NULL,
    asin VARCHAR(20),
    reviewer_name VARCHAR(100),
    review_text TEXT,
    overall FLOAT,
    summary VARCHAR(255)
);
```

We inserted 4 sample rows as of now into the table.



```
musical_instruments_db=# select count(*) from musical_instruments_reviews;
 count
-------
     4
(1 row)
```

Now we have data set up in our source table.

Step 4. Target Setup(Snowflake)

a. Create DWH, DB, ROLE, SCHEMA as per the requirement.[SQL File - musical_instruments_reviews.sql]. Now we can see the target DB in snowflake.
b. We have not yet created any table in target, the connector will check if table exists or not , if not it will create and then start ingesting the data.

Step 5. Data Ingestion in staging.
Now how to ingest data from postgres to snowflake?
Answer is we will use **Airbyte, which is an open-source data integration platform that helps you move data from source to destination (called "connectors"). We can use any other solution as well.**

✅ **Think of Airbyte as:**

A ready-made pipeline builder that syncs data from PostgreSQL (source) to Snowflake (destination), without needing to write custom code.

🔌 **Key Features:**

Supports **hundreds of sources** (PostgreSQL, MySQL, APIs, etc.)
Works with **many destinations** (Snowflake, BigQuery, Redshift, etc.)
Handles **schema mapping**, **incremental sync**, **full refresh**, and even **Change Data Capture (CDC)**
Open-source + also available as a **cloud SaaS** product
Can run locally (Docker) or in cloud (Airbyte Cloud)

🔷 **Why you need Airbyte here?**

Snowflake **doesn't have a native built-in service like we have AWS Glue** to pull directly from source.

🔽 So our architecture will look like:

PostgreSQL (Source DB) → Airbyte (Connector Tool) → Snowflake (Target DB)

Step 6: Install Airbyte using the instruction given in : Link or using Link. Login using the credentials. You can check for credentials by running below command:

```
>> abctl local credentials
```

Step 7: Open airbyte on localhost:8000(default) , create a new source connection as postgres with all the details.

Host will be 'host.docker.internal' & port : 5433 for my case since running on docker.

Similarly setup destination (here snowflake).

Setup connection using created source & destination. (**We alter the table created and added PRIMARY KEY constraint that missed earlier**)

Step 8: Sync the pipeline & data will be loaded

```
-- DATA CHECK AFTER LOAD

SELECT * FROM AIRBYTE_MUSICAL_INSTRUMENTS_REVIEWS;

-- YES WE HAVE :)
```

Thanks for following this document.