Use Case Data Engineer
Amarildo Xhyra

Brief Explanation

After downloading **car.csv** and **consumer.csv**, I noticed data formatting issues:
Issues Identified

- **car.csv**: Headers contained **special characters and extra values** (e.g., ('Make', 0)).
- **consumer.csv**: Header misalignment – six columns of data but only five headers. The third column was labeled **"Year"**, but it actually contained **"Type"** data.

Fixes Implemented

- **Cleaned Headers**: Removed special characters and corrected column names.
- **Shifted Headers**: Adjusted misaligned headers and renamed **"Year"** → **"Type"**.
- **Handled Missing Values**: If Sales Volume was missing, it was set to None to prevent database errors.
- **Converted Data**: Fixed formatting issues that prevented direct manipulation.

Optimization & Execution

- Used **batch insertion (executemany)** for faster database writes.
- When Excel editing wasn't possible, converted the file to **JSON**, which resolved formatting errors.

Files Overview

- **insert.py** → Cleans & inserts data into PostgreSQL.
- **queries.sql** → Contains SQL queries for analysis.
- **generate_graphs.py** → Generates data visualizations.

Detailed Explanation

The automotive industry has a very large model base that keeps growing, and we at Beev want to try and get a better understanding of the market shares in a few key countries. As a Beev Data Engineer, you are asked by the Sales team to aggregate their numbers into a robust environment where they can easily scale and enrich the existing data, they provide you with two csv files. In the following exercise you are asked to:

1. Create a database that can handle the sample data. You are provided with a simple docker file that allows you to run a virtual database on your pc. You need to install docker desktop first, then run the docker-compose commands "up-d" and "down" to start and shut down the database. You can try to connect to the database with the pgAdmin tool or DBeaver.

Steps to Implement the Database:

1. First we download docker:***https://www.docker.com/products/docker-desktop/***
2. In the terminal CMD:
We navigate to the folder where **docker-compose.yml** file is located:
cd path\desktop\amarildo\docker-compose-file
3.To start the database (and download the files needed):
docker compose up –d
4.Test: ***docker compose version***
5. TEST:database container is running:***docker ps***

Use Case Data Engineer
Amarildo Xhyra

Connect to the Database Using DBeaver

DBeaver (For Developers/Data Analysts: more powerful and flexible).
1.Downolad DBeaver:***https://dbeaver.io/download/***
2.New Database Connection → PostgreSQL.
Details(docker-compose.yml file):

a. **Host**: localhost
b. **Port**: 5432
c. **Database**: test_db
d. **Username**: admin
e. **Password**: admin

2.Once the database is up and running we need a python script that will be able to read the provided csv files and insert the data into the database. The data model is up to you to decide and should be flexible enough to easily understand, scale & enrich the data.
Python script: insert_data.py,
In the CMD:To run the python script we have created:

```
py -m pip install pandas
py -m pip install psycopg2
py insert_data.py
```

3. Once the data is in the database, we want to run a few sql queries
to check some basic data quality:(write the answers in a .sql file)
a. We want to find the total number of cars by model by country.
b. We want to know which country has the most of each model
and how many they have.
c. We want to know if any model was sold in the USA but not in
France.
d. We want to know how much the average car costs in every
country by engine type.
e. We want to know the average ratings of electric cars vs
thermal cars
Solution: queries.sql

4. Bonus: Write a script that can read data from the database and
return a graph that would show the volume of electric vs thermal
cars sold per year, another graph that shows the value of electric vs thermal cars sold per year.
Script: generate_graphs.py
In the CMD:To run the python script we have created:

```
py generate_graphs.py
```