# Project\_2\_customer\_orders\_report

## The business wants to know who their best customers are and how much they are spending.

I built a data pipeline that reads raw customer order data, loads it into a database, runs SQL queries to extract insights like total orders and spending per customer, and generates both CSV reports and bar charts — so that client can easily see who their best customers are.

STEP 1: Get the Customer Orders (CSV File)

* This is where all the customer purchase info is stored—like an Excel table.

STEP 2: Load That File into a Database (SQLite)

* Why? Because databases are easier and faster to query than Excel files.

STEP 3: Ask Questions Using SQL (SQL Queries)

* Now that the data is stored in a database, we use **SQL** (a simple query language) to ask

STEP 4: Run the SQL Queries Using Python

* Now we write a Python script

STEP 5: Show Visual Charts

* We use a Python tool called matplotlib for this.
* To help decision-makers, we also show a **bar chart**

STEP 6: Automate the Whole Thing

* Instead of running multiple scripts, we create one master file called:
* 📁 run\_pipeline.py

These were my questions, while I was working on this project. Upon checking and understanding each, I here write my understanding of each step I have followed.

1. Why should we save the CSV into a database (SQLite)? What's the purpose?

CSV file it's useful, but hard to search and slow for large amounts of data.

A **database** (like SQLite) is like a **well-organized digital filing cabinet. so, we can analyze it using SQL, which is a powerful and efficient way to work with data.**

2. Why did we create generate\_reports.py? Why not run the sql\_queries.sql file directly?

* sql\_queries.sql is just a **text file** containing SQL questions.
* Python is the **tool that connects to the database and runs those SQL queries**.
* We can’t run .sql files directly. So, we need Python to act like the brain that reads the questions from sql\_queries.sql, asks them to the database, and processes the results**.**

3. How did we connect the query file (sql\_queries.sql) to the Python file (generate\_reports.py)?

We "connect" them by reading the .sql file inside the Python script and sending those queries to the database using Python.

Ex: with open(sql\_file, "r") as file:

sql\_script = file.read()

for i, query in enumerate(queries):

df = pd.read\_sql\_query(query, conn)

This **sends each query to the database** and gets the answer

4. What’s the purpose of the load\_data.py file?

Take my raw data (CSV) and put it into a structured database (SQLite), so I can analyze it later using SQL

1. Reads the CSV (order data)
2. Connects to the SQLite database
3. Saves the data as a table inside the database

5. What’s the difference between load file and generate file?

* load\_data.py = **Buying ingredients and putting them in the fridge**
* generate\_reports.py = **Cooking and serving dishes from the ingredients**

Instead of running multiple files, I created single py file to run the code which is run\_pipeline.py

In my first project I just loaded the csv file , analysed and reported.

In this project I wanted to see if I can create charts (just had a thought). After researching found that we can use python tool to create a chart. So tried here.

The below is my understanding of each line of the code and what it does.

**Run\_pipleline.py**

**1. Import Tools (Libraries)**

import os

import pandas as pd

import sqlite3

import matplotlib.pyplot as plt

| **Line** | **Meaning** |
| --- | --- |
| Os | Helps find file or folder paths on your computer. |
| pandas | Handles data tables, CSV files, and SQL results. |
| sqlite3 | Connects to SQLite database and runs commands. |
| matplotlib.pyplot | Draws charts (like bar graphs). |

2. Set Up File and Folder Paths

script\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

csv\_file = os.path.abspath(os.path.join(script\_dir, "..", "data", "customer\_orders.csv"))

db\_file = os.path.abspath(os.path.join(script\_dir, "..", "db", "orders.db"))

sql\_file = os.path.abspath(os.path.join(script\_dir, "sql\_queries.sql"))

report\_dir = os.path.abspath(os.path.join(script\_dir, "..", "data", "reports"))

os.makedirs(report\_dir, exist\_ok=True)

Meaning: This block is just preparing your workspace: “Where is everything?”

3. Load CSV into SQLite Database

print("🔄 Loading data into SQLite database...")

df = pd.read\_csv(csv\_file)

conn = sqlite3.connect(db\_file)

df.to\_sql("orders", conn, if\_exists="replace", index=False)

print("✅ Data loaded into:", db\_file)

| **Line** | **Meaning** |
| --- | --- |
| pd.read\_csv() | Reads your Excel-like CSV file into Python. |
| sqlite3.connect() | Connects to the SQLite database file. |
| df.to\_sql() | Saves the data into a table called "orders" inside the database. |
| if\_exists="replace" | Replaces old data if the table already exists. |
| index=False | Don’t save row numbers as a separate column. |

Meaning: Upload the order list into the database so I can ask questions later.

4. Read SQL Queries from File

with open(sql\_file, "r") as file:

sql\_script = file.read()

queries = [q.strip() for q in sql\_script.split(";") if q.strip()]

| **Line** | **Meaning** |
| --- | --- |
| open(..., "r") | Opens the .sql file so we can read it. |
| file.read() | Reads all the text (SQL queries) into a string. |
| .split(";") | Splits the queries wherever there’s a ; (each ends with a semicolon). |
| .strip() | Removes extra spaces or new lines. |
| if q.strip() | Keeps only valid queries (ignores blank ones). |

Meaning:You’re now ready to **run each question (SQL query) one by one**

5. Run Each Query, Save Result, and Show Graphs

for i, query in enumerate(queries, start=1):

df\_result = pd.read\_sql\_query(query, conn)

print(df\_result)

report\_path = os.path.join(report\_dir, f"report\_{i}.csv")

df\_result.to\_csv(report\_path, index=False)

| **Line** | **Meaning** |
| --- | --- |
| for i, query in enumerate() | Loop through each SQL query one by one. |
| pd.read\_sql\_query() | Run the SQL query on the database and get results as a table (DataFrame). |
| print(df\_result) | Show the result in the terminal. |
| df\_result.to\_csv() | Save the result as a CSV file like report\_1.csv, report\_2.csv, etc. |

6. Show Charts for First Two Reports

| **Line** | **Meaning** |
| --- | --- |
| if i == 1: | If it’s the first report, make a bar chart of **total orders** per customer. |
| elif i == 2: | If it’s the second report, make a bar chart of **total amount spent**. |
| plt.bar(...) | Build the bar chart. |
| plt.show() | Show the chart in a window. |

7. Close the Database: Closes the connection to the database

conn.close()

8. Completed

print("\n🎉 All done! Reports and visualizations are ready.")