# 00-working

July 27, 2024

## 1 Car Sales Project

### 1.1 Working

I quickly reviewed the task, the time required for completion, and thought about the tools I would need to complete the task.

I knew the job role required MySQL and Python, and after seeing the `CSV` files contained tabular data, and looking at the task questions, I decided to use these as my main tools.

As the task needed to be done in 2 hours, I decided the quickest workflow would have been using `Tableau`, but as I only have a free account with `Tableau Public`, I decided on the quickest workflow using Free and Open Source tools:

- `Jupyter` notebooks to document my working and do exploratory data analysis (EDA)
- `pandas` for data importing and wrangling the `CSV` files
- `pandasql` to run SQL queries on the `pandas` DataFrame, which is easier syntax for basic queries
- `pygwalker` to create quick interactive data visualisations (EDA) for the stakeholders
- `streamlit` to make the visualisations accessible for non-tehnical stakeholders

In the end, I used `Quarto` instead of `streamlit`, because I wanted to try creating a presentation/report which could be version controlled. I wanted to see if `pygwalker` would work inside a `Quarto` document.

### 1.2 Steps

1. Create project folder structure
2. Initialise `git` repository
3. Create `conda` environment with Python 3.10 and key packages
4. Start inspecting data
5. EDA cycles to visualise, inspect, and make any transformations to the data.
6. Drafted code then used GPT-4o as code assistant when I forgot/got stuff wrong.
7. Decide on appropriate visualisations relevant to answering the questions.
8. Write and publish the report/presentation as Quarto documents.

### 1.3 Discoveries

#### 1.3.1 Tables

I know straight away that to answer the questions, I will need to do some SQL joins, because there is more than one table.

There are two tables (DataFrames) of car sales data: `purchase_data` and `vehicle_data`

`customer_id` is the primary key of the `purchase_data` table

`vehicle_id` is the primary key of the `vehicle_data` table

So, I will use `vehicle_id` as the key to join the tables on.

#### 1.3.2 `purchase_data`

This table contains:

- Information about car purchases per customer
- `9` columns
- `2_000_000` purchases (rows)

#### 1.3.3 `test_vehicle_data`

- Information about vehicles
- `19` columns
- `978` vehicles (rows)

#### 1.3.4 Answering the Questions

Next, because there is a lot of data in the tables, I read the questions, to know which variables are needed to answer them.

I listed the questions, highlighted the variables, and then clarified which columns refer to which variables in the tables.

I need to do this to check if there are any issues with data quality, missing values, or outliers, only for the relevant variables. It will take too long to look at all of the columns, due to the number of columns.

### 1.4 TODO

**DONE** - Tables are already indexed using the `customer_id` (`purchase_data`) and `vehicle_id` (`vehicle_data`) columns, so I need to import the `CSV` to specify the index (and not add another index).

## 2 EDA

See the `01-eda.ipynb` for my working.

## 3 Overall Evaluation

Strengths:

- Automated workflow which means I could make reporting into a pipeline
- I used Free and Open Source tools, so avoided paying subscriptions
- Version control with `Quarto` is brilliant and possible compared to `Jupyter` notebooks.

Weaknesses:

- Coding is slower than using `Tableau` GUI
- Choosing between SQL and pandas for queries took a bit of time
- `duckdb` would have been much faster than `pandasql`, but for some reason, it didn't work
- Folium/Leafleat could not display map (`for` loop took too long), so went with `PyGWalker` (which sadly did not work with Quarto; I've made an issue on their GitHub), and then settled on datashader/holoviews/bokeh (efficient, but the map looked a bit crap, so I took a screenshot of the `PyGWalker` map)

Ideas for next time:

- `Tableau` for quicker workflow
- `PyGWalker` for entire project as interactive visualisations
- `Streamlit` for the app
- `duckdb` to speed up SQL queries