**TO run the project:**

1. Extract the project.
2. Create a virtual environment:

python3 -m venv .venv

3. Activate the virtual environment:

.venv\Scripts\activate

4. Install dependencies:

pip install -r requirements.txt

5. To run app:

flask run

-------------------------------------------------------------------------------------------------------------------------------

1. **Set Up Python and Libraries**:
   * **Python** was installed, and two important libraries were added:
     + **pandas**: This library is used for data handling, manipulation, and analysis. It allows us to load, explore, and clean data efficiently.
     + **matplotlib**: A powerful library used for creating visualizations, such as charts and graphs.
2. **Load and Preview the Dataset**:
   * We started with a dataset containing car sales information.
     + **Brand**: The car's brand (e.g., BMW, Toyota, etc.)
     + **Price**: The car’s selling price.
     + **Body Type**: The type of car (e.g., sedan, van, crossover).
     + **Mileage**: The distance the car has traveled.
     + **Engine Volume**: The volume of the car's engine in liters.
     + **Fuel Type**: The type of fuel (e.g., petrol, diesel).
     + **Year**: The manufacturing year of the car.
     + **Model**: The model of the car.
     + **Category**: A category for organizing the cars (e.g., luxury, sports).
   * We loaded this dataset into a pandas DataFrame and previewed it to understand its structure and contents.
3. **Performing the Analysis**:
   * Using **pandas**, we grouped the data based on different attributes to answer the questions:
     + **Count by Brand**: We counted how many cars were listed for each brand.
     + **Count by Body Type**: We counted how many cars there were in each body type category (e.g., sedan, van).
     + **Average Price by Brand**: We calculated the average price of cars for each brand.
     + **Average Engine Volume by Car Type**: We calculated the average engine volume for each body type.
4. **Data Visualization**:
   * We used **matplotlib** to create visualizations, which made the data easier to interpret:
     + **Bar Charts**: We created bar charts to show how many cars were available for each brand and body type.
     + **Pie Charts**: We used pie charts to visualize the distribution of car types (e.g., sedan, van).
   * These charts helped visualize the patterns and trends from the analysis, making the results more understandable.

### **HTML and Python Changes:**

#### **HTML Changes:**

1. **Results Page (results.html)**:
   * Displayed the results of the analysis, including:
     + Total Cars by Brand
     + Average Price by Brand
     + Cars by Category
   * Added a **"View Charts" button** that links to the charts.html page.
2. **Charts Page (charts.html)**:
   * Created a new page for displaying data visualizations (charts).
   * Added a **"Go to Results" button** to allow users to navigate back to the results page.

#### **Python (Flask) Changes:**

1. **New Route for /charts**:
   * Created a new route in app.py for the charts page, where the car sales data visualizations would be rendered.
   * This route handles the rendering of the charts.html page.
2. **Handling Form Data**:
   * The form data entered by the user (such as car brand, price, body type, etc.) is saved in a CSV file.
   * The app appends new car data to the existing CSV file when submitted.
3. **Analysis Function (perform\_analysis)**:
   * The perform\_analysis function groups and processes the data to calculate:
     + Total count by brand
     + Average price by brand
     + Count by category
   * This function is called when rendering the results page, passing the computed analysis results.
4. **Redirect Functionality**:
   * After the form submission, the app redirects the user to the results page.
   * The results page allows the user to view the analysis and has a button to navigate to the charts page.
5. **Chart Data**:
   * Although the visualization step is outlined, the full integration with libraries like **matplotlib** or **JavaScript charts** (e.g., using **Chart.js**) is still open for further development.
   * The charts will represent the data visually, such as showing the number of cars by brand and price distribution.

### **Key Terms Explained:**

* **Dataset**: A structured collection of data, usually in tabular format.
* **pandas**: A Python library used for data manipulation and analysis.
* **matplotlib**: A Python library used for creating static, animated, and interactive visualizations.
* **Grouping**: Combining data based on a common feature (e.g., grouping by brand or body type).
* **Visualization**: The process of converting data into graphical representations, like charts or graphs.