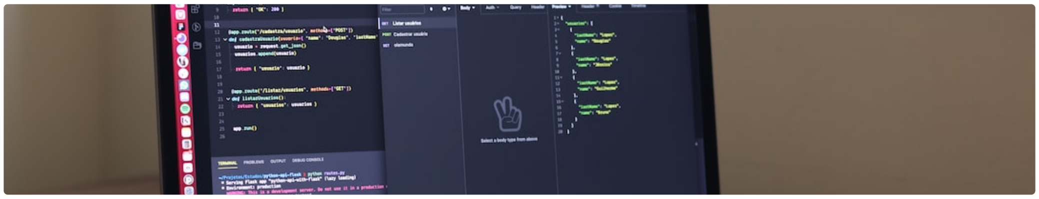


Python Extract Transform and Load Project

Introduction

In this project, I analyse fuel economy between different car manufacturers for an Automobile research company, that wants to launch new research based on the given data.

ETL process

Extract:

- Downloaded raw data files (CSV or XLSX) from the data source (Download Fuel Economy Data)
- I used Python to read these files and extract the data into a Pandas data frame.

Transform:

- I performed data cleaning operations, including missing values, renaming columns, and dropping duplicates.
- I calculated additional metrics such as checksums for data integrity.
- I visualized trends, distributions, and relationships in the data, which can be considered a form of transformation for analysis purposes.

Load:

- I cleaned and transformed data into a new CSV (called ‘cleaned_data.csv’) for the main analysis.
- I aggregated data to answer specific questions.
- I stored the results of all ten visualisations for analysis.

Python - Data cleaning process

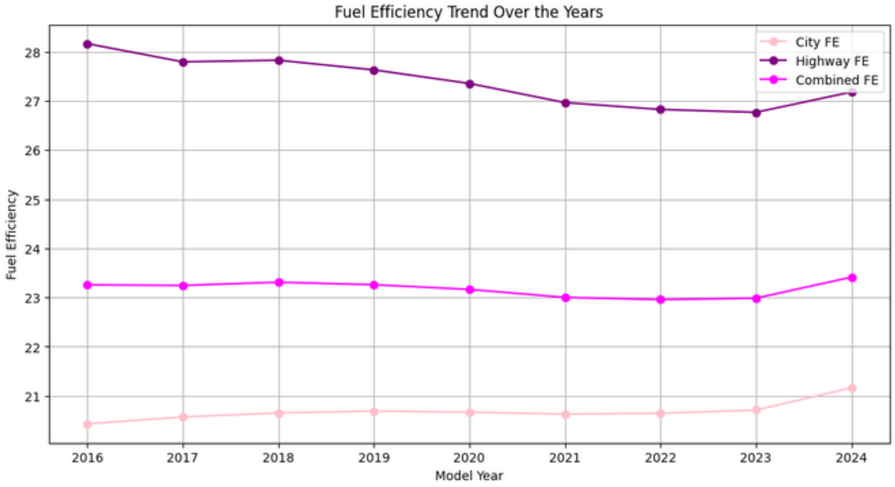
Data cleaning.py

```
import os
import pandas as pd
import glob

def clean_and_concatenate_data(folder_path):
    # Step 1: Read Excel files using the specified folder path
    excel_files = glob.glob(os.path.join(folder_path, '*.xlsx'))

    # Initialize an empty list to store DataFrames
    dfs = []

    # Read each Excel file and append to the list of DataFrames
    for file in excel_files:
        try:
            df = pd.read_excel(file)

            # Append the DataFrame to the list
```

```python
            dfs.append(df)

        except pd.errors.ParserError as e:
            print(f"Error reading file {file}: {e}")

    # Step 2: Concatenate all DataFrames into a single DataFrame
    concatenated_df = pd.concat(dfs, ignore_index=True)

    # Step 3: Clean the DataFrame
    # Rename columns, drop duplicates, handle missing values, etc.
    # Define columns to be renamed
    columns_to_rename = {
        'Eng Displ': 'Engine Displacement',
        '# Cyl': '# Cylinders',
        'City FE (Guide) - Conventional Fuel': 'City FE',
        'Hwy FE (Guide) - Conventional Fuel': 'Highway FE',
        'Comb FE (Guide) - Conventional Fuel': 'Combined FE',
        'Air Aspiration Method Desc': 'Air Aspiration Method',
        'Trans Desc': 'Transmission Description',
        'City CO2 Rounded Adjusted': 'City CO2',
        'Hwy CO2 Rounded Adjusted': 'Highway CO2',
        'Comb CO2 Rounded Adjusted (as shown on FE Label)': 'Combined CO2'
    }

    # Columns to retain
    columns_to_retain = [
        'Model Year', 'Mfr Name', 'Division', 'Carline', 'Engine Displacement', '# Cylinders', 'Transmission
        'City FE', 'Highway FE', 'Combined FE', 'Air Aspiration Method', 'Transmission Description', '# Gears
        'Drive Desc', 'Carline Class Desc', 'Release Date', 'City CO2', 'Highway CO2', 'Combined CO2'
    ]

    # Rename columns
    concatenated_df.rename(columns=columns_to_rename, inplace=True)

    # Retain specified columns and drop any other columns
    concatenated_df = concatenated_df[columns_to_retain]

    # Replace null values with None
    concatenated_df = concatenated_df.where(pd.notna(concatenated_df), None)

    # Remove duplicate columns
    concatenated_df = concatenated_df.loc[:, ~concatenated_df.columns.duplicated()]

    # Step 4: Define the output file path
    output_file_path = os.path.join(folder_path, 'cleaned_data.csv')

    # Step 5: Save the cleaned DataFrame to a new CSV file
    concatenated_df.to_csv(output_file_path, index=False, encoding='utf-8')

    # Step 6: Print a message indicating successful completion
    print(f"Data cleaning and concatenation completed. Cleaned data saved to: {output_file_path}")

    # Step 7: Return the cleaned DataFrame
    return concatenated_df

# Call the function with the specified folder path and store the result
folder_path = folder_path = "C:\\Users\\Esther Oluwaseye\\Desktop\\Git_assessment_local\\python-data-assessme
cleaned_data = clean_and_concatenate_data(folder_path)
```

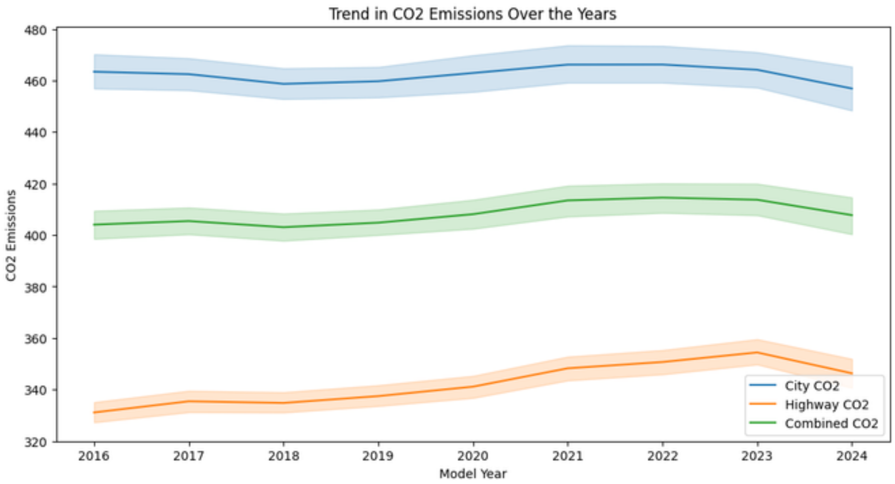
Python data analysis questions

`Question 1: How has fuel efficiency changed over the years`



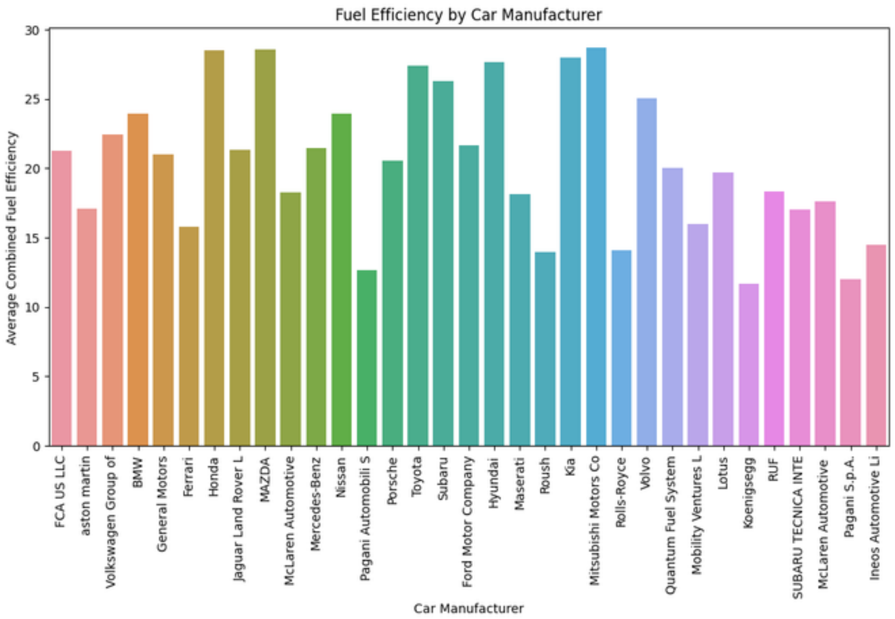
```python
#Question 1: How has fuel efficiency changed over the years

import pandas as pd
import matplotlib.pyplot as plt

# Load cleaned data frame
cleaned_data = pd.read_csv("cleaned_data.csv")

# Group the data by 'Model Year' and calculate the mean for each year
fuel_efficiency_trend = cleaned_data.groupby('Model Year')[['City FE', 'Highway FE', 'Combined FE']].mean()

# Plotting the trend with pink and purple colors
plt.figure(figsize=(12, 6))
plt.plot(fuel_efficiency_trend.index, fuel_efficiency_trend['City FE'], label='City FE', marker='o', color='pink
plt.plot(fuel_efficiency_trend.index, fuel_efficiency_trend['Highway FE'], label='Highway FE', marker='o', color
plt.plot(fuel_efficiency_trend.index, fuel_efficiency_trend['Combined FE'], label='Combined FE', marker='o', col

# Adding labels and title
plt.xlabel('Model Year')
plt.ylabel('Fuel Efficiency')
plt.title('Fuel Efficiency Trend Over the Years')
plt.legend()
plt.grid(True)
plt.show()

```

`Question 2: Look at the trend of CO2 emissions over the years`



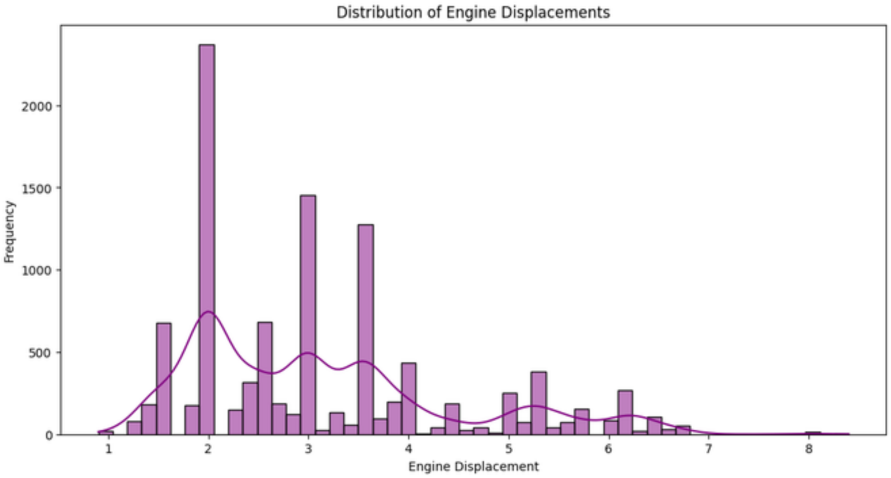
```python
# Q2:Look at the trend of CO2 emissions over the years

import matplotlib.pyplot as plt
import seaborn as sns

# Set the size of the plot
plt.figure(figsize=(12, 6))

# Plot the trend in CO2 emissions over the years for City, Highway, and Combined
sns.lineplot(x='Model Year', y='City CO2', data=cleaned_data, label='City CO2')
sns.lineplot(x='Model Year', y='Highway CO2', data=cleaned_data, label='Highway CO2')
sns.lineplot(x='Model Year', y='Combined CO2', data=cleaned_data, label='Combined CO2')

# Label the x-axis
plt.xlabel('Model Year')

# Label the y-axis
plt.ylabel('CO2 Emissions')

# Add a title to the plot
plt.title('Trend in CO2 Emissions Over the Years')

# Display legend to distinguish between City, Highway, and Combined CO2
plt.legend()

# Show the plot
plt.show()

```

`Q3: Which car manufacturers produce the most fuel-efficient vehicles?`



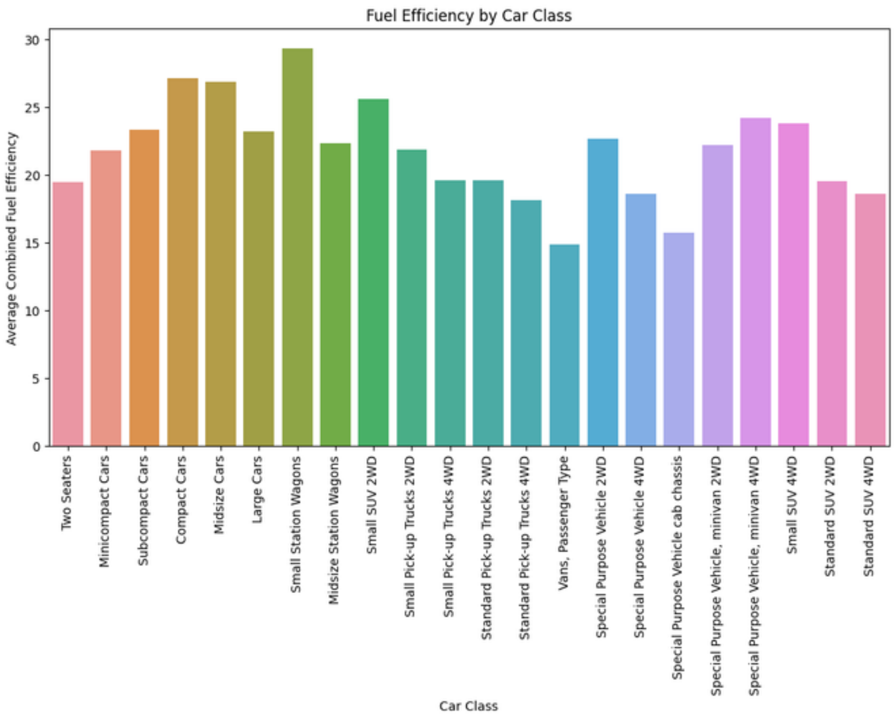
```python
# Q3: Which car manufacturers produce the most fuel-efficient vehicles?

import matplotlib.pyplot as plt
import seaborn as sns

# Set the size of the plot
plt.figure(figsize=(12, 6))

# Create a bar plot for average combined fuel efficiency by car manufacturer
sns.barplot(x='Mfr Name', y='Combined FE', data=cleaned_data, errorbar=None, estimator='mean')

# Rotate x-axis labels for better visibility
plt.xticks(rotation=90)

# Label the x-axis
plt.xlabel('Car Manufacturer')

# Label the y-axis
plt.ylabel('Average Combined Fuel Efficiency')

# Add a title to the plot
plt.title('Fuel Efficiency by Car Manufacturer')

# Show the plot
plt.show()

```

Q4: What is the distribution of engine displacements among different car models



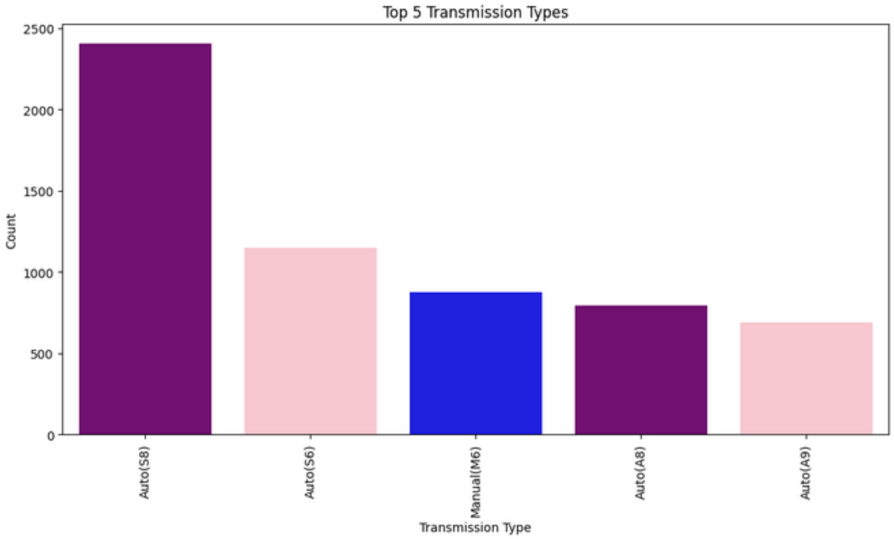
```python
# Q4:What is the distribution of engine displacements among different car models?

import matplotlib.pyplot as plt
import seaborn as sns

# Set the size of the plot
plt.figure(figsize=(12, 6))

# Create a histogram plot with KDE for the distribution of engine displacements
sns.histplot(cleaned_data['Engine Displacement'], kde=True, color='purple')

# Label the x-axis
plt.xlabel('Engine Displacement')

# Label the y-axis
plt.ylabel('Frequency')

# Add a title to the plot
plt.title('Distribution of Engine Displacements')

# Show the plot
plt.show()


```

`Q5: How do different car classes compare in terms of fuel efficiency?`



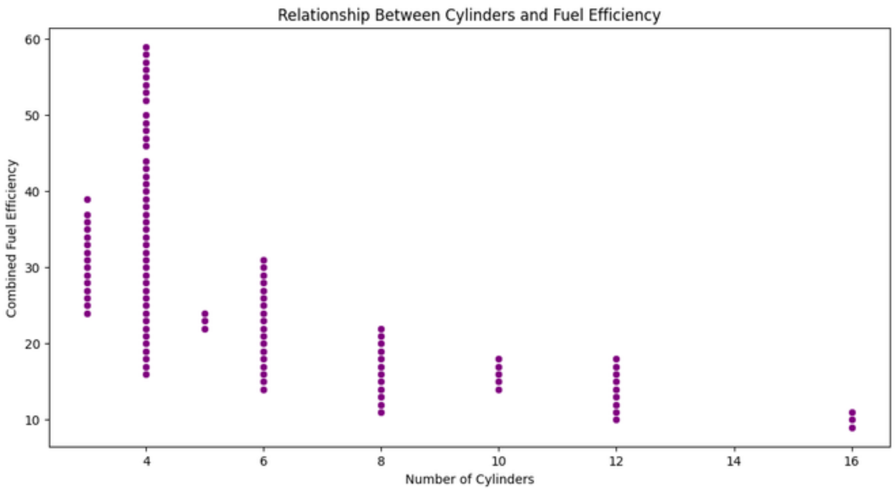
```python
# Q5:How do different car classes compare in terms of fuel efficiency?

import matplotlib.pyplot as plt
import seaborn as sns

# Set the size of the plot
plt.figure(figsize=(12, 6))

# Create a bar plot for the average combined fuel efficiency across different car classes
sns.barplot(x='Carline Class Desc', y='Combined FE', data=cleaned_data, errorbar=None, estimator='mean')

# Rotate x-axis labels for better readability
plt.xticks(rotation=90)

# Label the x-axis
plt.xlabel('Car Class')

# Label the y-axis
plt.ylabel('Average Combined Fuel Efficiency')

# Add a title to the plot
plt.title('Fuel Efficiency by Car Class')

# Show the plot
plt.show()

```

`Q6: Are there any notable trends in the transmission types used in vehicles?`



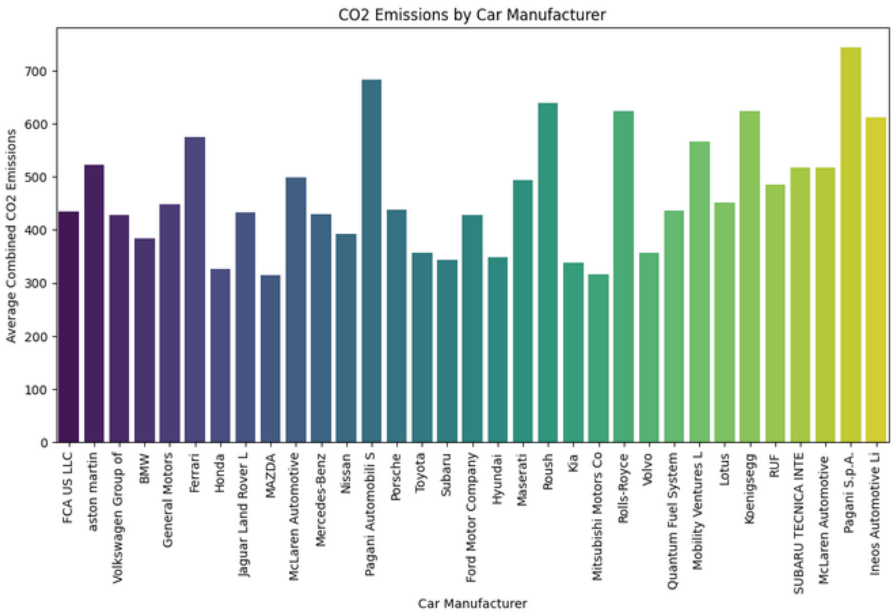
```python
# Q6: Are there any notable trends in the transmission types used in vehicles?
import matplotlib.pyplot as plt
import seaborn as sns

# Get the top 5 transmission types
top_transmissions = cleaned_data['Transmission'].value_counts().nlargest(5).index

# Filter the DataFrame for the top 5 transmission types
filtered_data = cleaned_data[cleaned_data['Transmission'].isin(top_transmissions)]

# Set the size of the plot
plt.figure(figsize=(12, 6))

# Create a count plot to visualize the distribution of transmission types
sns.countplot(x='Transmission', data=filtered_data, order=top_transmissions, palette=['purple', 'pink','blue'])

# Rotate x-axis labels for better readability
plt.xticks(rotation=90)

# Label the x-axis
plt.xlabel('Transmission Type')

# Label the y-axis
plt.ylabel('Count')

# Add a title to the plot
plt.title('Top 5 Transmission Types')

# Show the plot
plt.show()


```

`Q7: What is the relationship between the number of cylinders and fuel efficiency?`



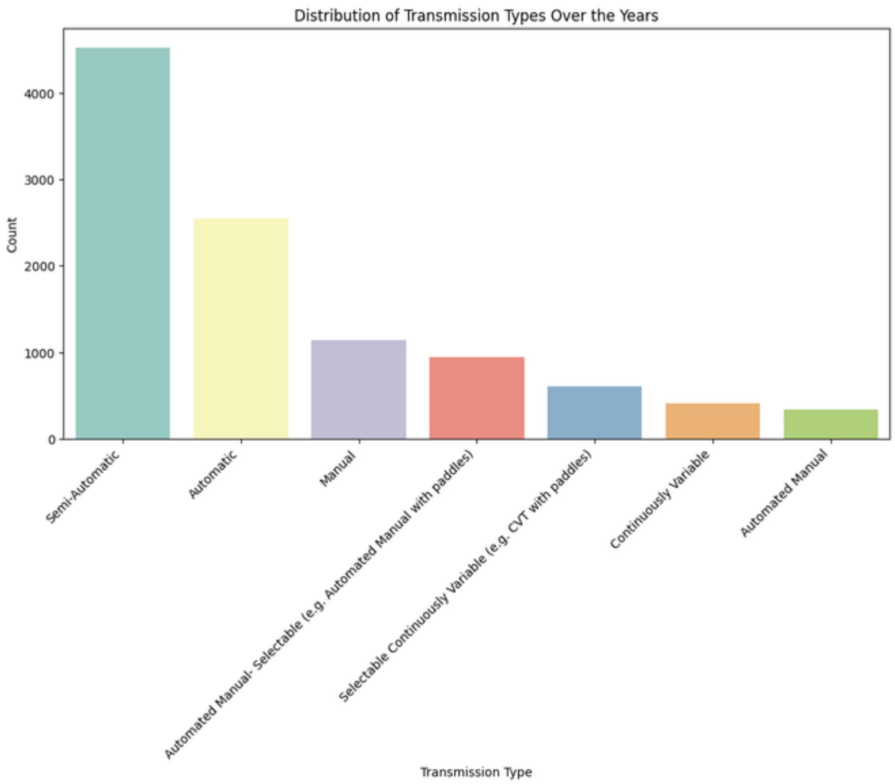
```python
import matplotlib.pyplot as plt
import seaborn as sns

# Set the size of the plot
plt.figure(figsize=(12, 6))

# Create a scatter plot to visualize the relationship
sns.scatterplot(x='# Cylinders', y='Combined FE', data=cleaned_data, color='purple')

# Label the x-axis
plt.xlabel('Number of Cylinders')

# Label the y-axis
plt.ylabel('Combined Fuel Efficiency')

# Add a title to the plot
plt.title('Relationship Between Cylinders and Fuel Efficiency')

# Show the plot
plt.show()

```

`Q8: Which car manufacturers produce the lowest CO2 emissions?`



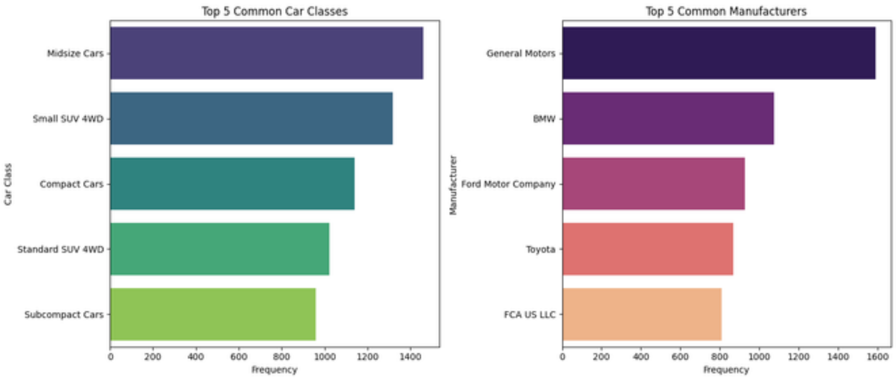
```python
# Q8: Which car manufacturers produce the lowest CO2 emissions?

import matplotlib.pyplot as plt
import seaborn as sns

# Set the size of the plot
plt.figure(figsize=(12, 6))

# Create a bar plot for the average combined CO2 emissions by car manufacturer
sns.barplot(x='Mfr Name', y='Combined CO2', data=cleaned_data, errorbar=None, estimator='mean', palette='viridis

# Rotate x-axis labels for better readability
plt.xticks(rotation=90)

# Label the x-axis
plt.xlabel('Car Manufacturer')

# Label the y-axis
plt.ylabel('Average Combined CO2 Emissions')

# Add a title to the plot
plt.title('CO2 Emissions by Car Manufacturer')

# Show the plot
plt.show()


```

`Q9: What are the top distribution of transmission types`



```python
import matplotlib.pyplot as plt
import seaborn as sns

# Set the size of the plot
plt.figure(figsize=(12, 6))

# Create a count plot for the distribution of transmission types over the years
sns.countplot(x='Transmission Description', data=cleaned_data, palette='Set3', order=cleaned_data['Transmission

# Add a title to the plot
plt.title('Distribution of Transmission Types Over the Years')

# Label the x-axis
plt.xlabel('Transmission Type')

# Label the y-axis
plt.ylabel('Count')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45, ha='right')

# Show the plot
plt.show()

```

`Q10: What are the top 5 common car classes and manufacturers?`



```python
# Q10: What are the top 5 common car classes and manufacturers?

import matplotlib.pyplot as plt
import seaborn as sns

# Top 5 common car classes
top_car_classes = cleaned_data['Carline Class Desc'].value_counts().head(5)

# Top 5 common manufacturers
top_manufacturers = cleaned_data['Mfr Name'].value_counts().head(5)

# Plotting the bar plots
plt.figure(figsize=(14, 6))

# Plot for car classes
plt.subplot(1, 2, 1)
sns.barplot(x=top_car_classes.values, y=top_car_classes.index, palette='viridis')
plt.title('Top 5 Common Car Classes')
plt.xlabel('Frequency')
plt.ylabel('Car Class')

# Plot for manufacturers
plt.subplot(1, 2, 2)
sns.barplot(x=top_manufacturers.values, y=top_manufacturers.index, palette='magma')
plt.title('Top 5 Common Manufacturers')
plt.xlabel('Frequency')
plt.ylabel('Manufacturer')

plt.tight_layout()
plt.show()


```