```
# Load the Drive module:
from google.colab import drive

drive.mount("/content/drive")
! ls "/content/drive/MyDrive"
```

```
Mounted at /content/drive
'Build AI for a Better Society'          'Learning Insights (Bachelor)'
'Build Your Dream Team '                 'Mission Identification (Bachelor)'
'Calibration Phase - Onboarding Time! '  'Orientation Group Challenge - 2023'
 cars_dataset.csv                         pairplot.pdf
'Colab Notebooks'                         pairplot.png
 correlation_heatmap.pdf                 'Strategy & Global Markets'
'Foundations for Tech Impact'            'Sustainability Foundations (Bx)'
'Introduction to Coding and AI (BX)'     'Technology Revolutions'
```

```
import os
print(os.getcwd())
print(os.listdir())
print(os.listdir("drive/MyDrive"))
```

```
/content
['.config', 'drive', 'sample_data']
['Learning Insights (Bachelor)', 'Mission Identification (Bachelor)', 'Build Your Dream Team ', 'Introduction to Coding and AI (BX)
```

```
#Milestone 3 Code
import pandas as pd


df = pd.read_csv("/content/drive/MyDrive/cars_dataset.csv")
df.head()
```

	uuid	Make	Model	Vehicle Class	Engine Size(L)	Cylinders	Transmis
0	eb24f214-f18b-11ec-a33e-acde48001122	CHEVROLET	Suburban 4WD	SUV - STANDARD	5.3	8	
1	eb2525fe-f18b-11ec-a33e-acde48001122	CHEVROLET	CAMARO	SUBCOMPACT	3.6	6	
2	eb252d2e-f18b-11ec-	LEXUS	GS 350	MID-SIZE	3.5	6	

```
df = pd.read_csv("/content/drive/MyDrive/cars_dataset.csv")

#Classification of datatypes
df.dtypes
```

```
uuid                                 object
Make                                 object
Model                                object
Vehicle Class                        object
Engine Size(L)                      float64
Cylinders                             int64
Transmission                         object
Fuel Type                            object
Fuel Consumption City (L/100 km)    float64
Fuel Consumption Hwy (L/100 km)     float64
Fuel Consumption Comb (L/100 km)    float64
Fuel Consumption Comb (mpg)           int64
CO2 Emissions(g/km)                   int64
dtype: object
```

```
df = pd.read_csv("/content/drive/MyDrive/cars_dataset.csv")

# Count the number of columns
num_columns = df.shape[1]

# Print the result
print("Number of columns:", num_columns)
```

```
Number of columns: 13
```

```
df = pd.read_csv("/content/drive/MyDrive/cars_dataset.csv")

# Get the dimensions (number of rows and columns)
dimensions = df.shape

# Print the result
print("Dimensions:", dimensions)
```

```
Dimensions: (7385, 13)
```

```
df = pd.read_csv("/content/drive/MyDrive/cars_dataset.csv")

df["Make"]
```

```
0           CHEVROLET
1           CHEVROLET
2               LEXUS
3               LEXUS
4             PORSCHE
              ...
7380            HONDA
7381    MERCEDES-BENZ
7382          HYUNDAI
7383          PORSCHE
7384             FORD
Name: Make, Length: 7385, dtype: object
```

```
df = pd.read_csv("/content/drive/MyDrive/cars_dataset.csv")

df['Make'].unique()
```

```
array(['CHEVROLET', 'LEXUS', 'PORSCHE', 'BMW', 'AUDI', 'NISSAN',
       'LINCOLN', 'VOLKSWAGEN', 'TOYOTA', 'ROLLS-ROYCE', 'MERCEDES-BENZ',
       'HONDA', 'HYUNDAI', 'KIA', 'GMC', 'ACURA', 'FORD', 'ASTON MARTIN',
       'JEEP', 'JAGUAR', 'DODGE', 'MAZDA', 'VOLVO', 'SUBARU', 'RAM',
       'CADILLAC', 'LAMBORGHINI', 'MINI', 'MITSUBISHI', 'MASERATI',
       'SCION', 'CHRYSLER', 'LAND ROVER', 'BENTLEY', 'INFINITI',
       'GENESIS', 'FIAT', 'BUICK', 'BUGATTI', 'ALFA ROMEO', 'SRT',
       'SMART'], dtype=object)
```

```
df = pd.read_csv("/content/drive/MyDrive/cars_dataset.csv")

df['Make'].value_counts()
```

```
FORD             628
CHEVROLET        588
BMW              527
MERCEDES-BENZ    419
PORSCHE          376
TOYOTA           330
GMC              328
AUDI             286
NISSAN           259
JEEP             251
DODGE            246
KIA              231
HONDA            214
HYUNDAI          210
MINI             204
VOLKSWAGEN       197
MAZDA            180
LEXUS            178
JAGUAR           160
CADILLAC         158
SUBARU           140
VOLVO            124
INFINITI         108
BUICK            103
RAM               97
LINCOLN           96
MITSUBISHI        95
CHRYSLER          88
LAND ROVER        85
FIAT              73
ACURA             72
MASERATI          61
ROLLS-ROYCE       50
ASTON MARTIN      47
BENTLEY           46
LAMBORGHINI       41
ALFA ROMEO        30
GENESIS           25
SCION             22
SMART              7
BUGATTI            3
```

```
SRT                2
Name: Make, dtype: int64
```

```
df = pd.read_csv("/content/drive/MyDrive/cars_dataset.csv")

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7385 entries, 0 to 7384
Data columns (total 13 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   uuid                              7385 non-null   object 
 1   Make                              7385 non-null   object 
 2   Model                             7385 non-null   object 
 3   Vehicle Class                     7385 non-null   object 
 4   Engine Size(L)                    7385 non-null   float64
 5   Cylinders                         7385 non-null   int64  
 6   Transmission                      7385 non-null   object 
 7   Fuel Type                         7385 non-null   object 
 8   Fuel Consumption City (L/100 km)  7385 non-null   float64
 9   Fuel Consumption Hwy (L/100 km)   7385 non-null   float64
 10  Fuel Consumption Comb (L/100 km)  7385 non-null   float64
 11  Fuel Consumption Comb (mpg)       7385 non-null   int64  
 12  CO2 Emissions(g/km)               7385 non-null   int64  
dtypes: float64(4), int64(3), object(6)
memory usage: 750.2+ KB
```

```
df = pd.read_csv("/content/drive/MyDrive/cars_dataset.csv")

df.corr(numeric_only=True)
```

	Engine Size(L)	Cylinders	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	F Consumpt Comb (m
Engine Size(L)	1.000000	0.927653	0.831379	0.761526	0.817060	-0.757
Cylinders	0.927653	1.000000	0.800702	0.715252	0.780534	-0.719
Fuel Consumption City (L/100 km)	0.831379	0.800702	1.000000	0.948180	0.993810	-0.927
Fuel Consumption	0.761526	0.715252	0.948180	1.000000	0.977299	-0.800

```
df = pd.read_csv("/content/drive/MyDrive/cars_dataset.csv")

df.describe()
```

	Engine Size(L)	Cylinders	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fuel Consumption Comb (mpg)	E
count	7385.000000	7385.000000	7385.000000	7385.000000	7385.000000	7385.000000	
mean	3.160068	5.615030	12.556534	9.041706	10.975071	27.481652	
std	1.354170	1.828307	3.500274	2.224456	2.892506	7.231879	
min	0.900000	3.000000	4.200000	4.000000	4.100000	11.000000	
25%	2.000000	4.000000	10.100000	7.500000	8.900000	22.000000	
50%	3.000000	6.000000	12.100000	8.700000	10.600000	27.000000	

```
df = pd.read_csv("/content/drive/MyDrive/cars_dataset.csv")

# Compute the distribution for all columns
distributions = {}
for column in df.columns:
    distribution = df[column].value_counts()
    distributions[column] = distribution

# Print the distributions
for column, distribution in distributions.items():
    print("Distribution of", column)
    print(distribution)
    print()
```

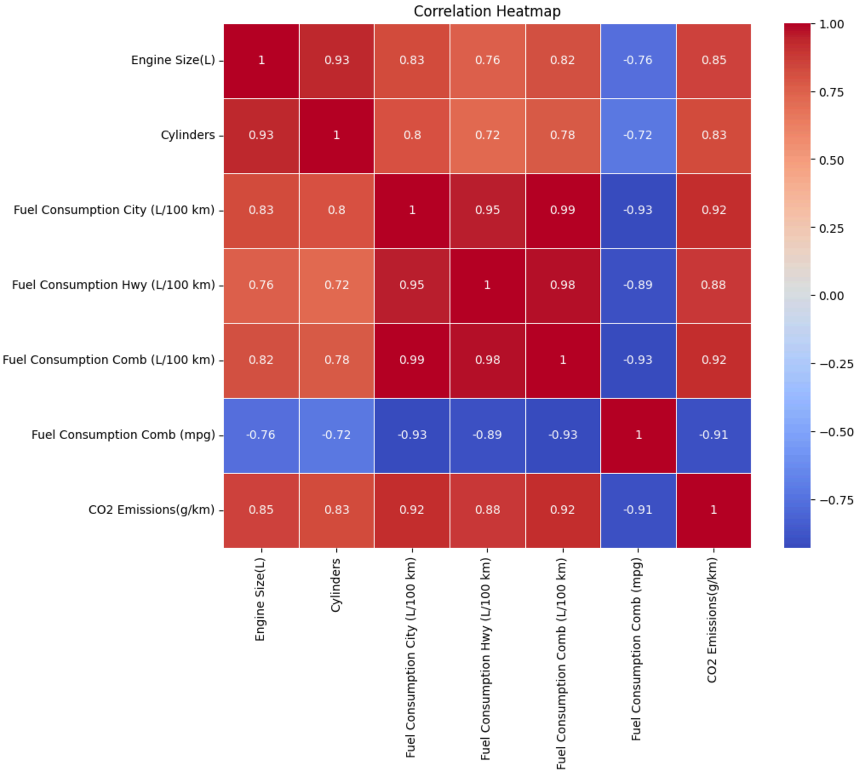
```
Distribution of uuid
eb24f214-f18b-11ec-a33e-acde48001122    1
```

```
eb733a46-f18b-11ec-a33e-acde48001122    1
eb73335c-f18b-11ec-a33e-acde48001122    1
eb732fec-f18b-11ec-a33e-acde48001122    1
eb732c72-f18b-11ec-a33e-acde48001122    1
                                       ..
eb4e8e3a-f18b-11ec-a33e-acde48001122    1
eb4e8a8e-f18b-11ec-a33e-acde48001122    1
eb4e8638-f18b-11ec-a33e-acde48001122    1
eb4e8228-f18b-11ec-a33e-acde48001122    1
eb98f772-f18b-11ec-a33e-acde48001122    1
Name: uuid, Length: 7385, dtype: int64

Distribution of Make
FORD             628
CHEVROLET        588
BMW              527
MERCEDES-BENZ    419
PORSCHE          376
TOYOTA           330
GMC              328
AUDI             286
NISSAN           259
JEEP             251
DODGE            246
KIA              231
HONDA            214
HYUNDAI          210
MINI             204
VOLKSWAGEN       197
MAZDA            180
LEXUS            178
JAGUAR           160
CADILLAC         158
SUBARU           140
VOLVO            124
INFINITI         108
BUICK            103
RAM               97
LINCOLN           96
MITSUBISHI        95
CHRYSLER          88
LAND ROVER        85
FIAT              73
ACURA             72
MASERATI          61
ROLLS-ROYCE       50
ASTON MARTIN      47
BENTLEY           46
LAMBORGHINI       41
ALFA ROMEO        30
GENESIS           25
SCION             22
SMART              7
BUGATTI            3
SRT                2
Name: Make, dtype: int64
```

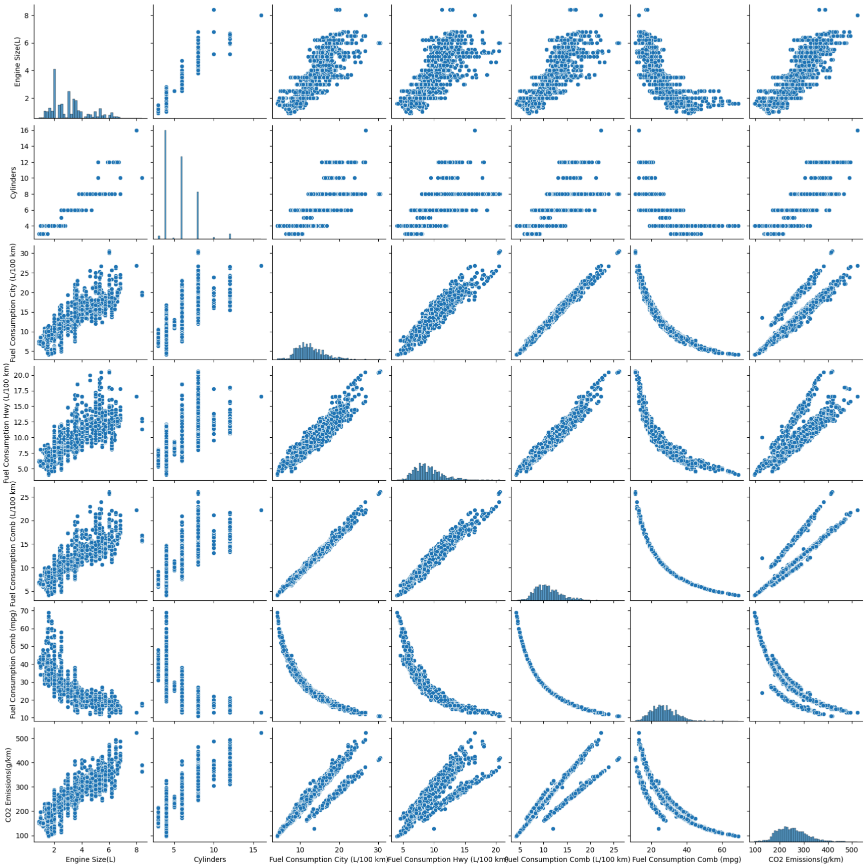
```
!pip install pandas
!pip install seaborn
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.22.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.22.4)
Requirement already satisfied: pandas>=0.25 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.1
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (8.
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seabor
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.25->seaborn) (2022.7.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=
import seaborn as sns
import matplotlib.pyplot as plt

# Read the CSV file into a pandas DataFrame
file_path = '/content/drive/MyDrive/cars_dataset.csv'
df = pd.read_csv(file_path)

# Calculate the correlation matrix
corr_matrix = df.corr()

# Create a heatmap using seaborn
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')

# Save the heatmap as a PDF file
plt.savefig('/content/drive/MyDrive/correlation_heatmap.pdf', format='pdf')
```

```
<ipython-input-16-e05a3341e700>:9: FutureWarning: The default value of numeric_only i
  corr_matrix = df.corr()
```



```
import pandas as pd
import seaborn as sns


sns.pairplot(df)

# Save the pair plot as a PNG file
plt.savefig('/content/drive/MyDrive/pairplot.png', dpi=300)
```


Engine Size(L)
Cylinders
Fuel Consumption City (L/100 km)
Fuel Consumption Hwy (L/100 km)
Fuel Consumption Comb (L/100 km)
Fuel Consumption Comb (mpg)
CO2 Emissions(g/km)
2
4
6
8
5
10
15
20
30
40
60
100
200
300
400
500

```
#Milestone 4 Code

import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np


df = pd.read_csv("/content/drive/MyDrive/cars_dataset.csv")


# --------------
# Import the data
# --------------
df = pd.read_csv('/content/drive/MyDrive/cars_dataset.csv')
print(df.info())

# Get Cylinders count
dMean_cylinders = df['Cylinders'].mean()
print('// complete ........ data model cylinders mean: ', dMean_cylinders)

# Generate a small image for slides ;-)
fig, ax = plt.subplots()
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.yaxis.set_ticks_position('left')
ax.xaxis.set_ticks_position('bottom')
sns.scatterplot(x='Fuel Consumption Comb (L/100 km)',
                y='CO2 Emissions(g/km)',
                data=df, alpha=0.5, color='grey')# palette='Grays')
plt.savefig('fig_scatter_Fuel_CO2.pdf')
plt.close()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7385 entries, 0 to 7384
Data columns (total 13 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   uuid                              7385 non-null   object
 1   Make                              7385 non-null   object
 2   Model                             7385 non-null   object
 3   Vehicle Class                     7385 non-null   object
 4   Engine Size(L)                    7385 non-null   float64
 5   Cylinders                         7385 non-null   int64
 6   Transmission                      7385 non-null   object
 7   Fuel Type                         7385 non-null   object
 8   Fuel Consumption City (L/100 km)  7385 non-null   float64
 9   Fuel Consumption Hwy (L/100 km)   7385 non-null   float64
 10  Fuel Consumption Comb (L/100 km)  7385 non-null   float64
 11  Fuel Consumption Comb (mpg)       7385 non-null   int64
 12  CO2 Emissions(g/km)               7385 non-null   int64
dtypes: float64(4), int64(3), object(6)
memory usage: 750.2+ KB
None
// complete ........ data model cylinders mean:  5.615030467163169
```

```
# --------------
# Import the data
# --------------
df = pd.read_csv('/content/drive/MyDrive/cars_dataset.csv')
print(df.info())

# Get Cylinders count
dMean_cylinders = df['Cylinders'].mean()
print('// complete ........ data model cylinders mean: ', dMean_cylinders)

# Generate a small image for slides ;-)
fig, ax = plt.subplots()
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.yaxis.set_ticks_position('left')
ax.xaxis.set_ticks_position('bottom')
sns.scatterplot(x='CO2 Emissions(g/km)',
                y='Cylinders',
                data=df, alpha=0.5, color='grey')# palette='Grays')
plt.savefig('fig_scatter_Cylinders_Fuel_CO2.pdf')
plt.close()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7385 entries, 0 to 7384
Data columns (total 13 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   uuid                              7385 non-null   object
 1   Make                              7385 non-null   object
 2   Model                             7385 non-null   object
 3   Vehicle Class                     7385 non-null   object
 4   Engine Size(L)                    7385 non-null   float64
 5   Cylinders                         7385 non-null   int64
 6   Transmission                      7385 non-null   object
 7   Fuel Type                         7385 non-null   object
 8   Fuel Consumption City (L/100 km)  7385 non-null   float64
 9   Fuel Consumption Hwy (L/100 km)   7385 non-null   float64
 10  Fuel Consumption Comb (L/100 km)  7385 non-null   float64
 11  Fuel Consumption Comb (mpg)       7385 non-null   int64
 12  CO2 Emissions(g/km)               7385 non-null   int64
dtypes: float64(4), int64(3), object(6)
memory usage: 750.2+ KB
None
// complete ........ data model cylinders mean:  5.615030467163169
```

```
# Selecting the desired columns
columns_of_interest = ['Cylinders', 'Fuel Consumption Comb (L/100 km)', 'CO2 Emissions(g/km)']
subset_df = df[columns_of_interest]

# Applying the describe() method on the subset DataFrame
description = subset_df.describe()

# Printing the description
print(description)
```

```
         Cylinders  Fuel Consumption Comb (L/100 km)  CO2 Emissions(g/km)
count  7385.000000                       7385.000000          7385.000000
mean      5.615030                         10.975071           250.584699
std       1.828307                          2.892506            58.512679
min       3.000000                          4.100000            96.000000
25%       4.000000                          8.900000           208.000000
50%       6.000000                         10.600000           246.000000
75%       6.000000                         12.600000           288.000000
max      16.000000                         26.100000           522.000000
```

```
from sklearn.linear_model import LinearRegression

# Import the data
df = pd.read_csv('/content/drive/MyDrive/cars_dataset.csv')
print(df.info())

# Get Cylinders count
dMean_cylinders = df['Cylinders'].mean()
print('// complete ........ data model cylinders mean: ', dMean_cylinders)

# Generate a scatter plot
fig, ax = plt.subplots()
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.yaxis.set_ticks_position('left')
ax.xaxis.set_ticks_position('bottom')
sns.scatterplot(x='CO2 Emissions(g/km)', y='Cylinders', data=df, alpha=0.5, color='grey')

# Perform linear regression
regression_model = LinearRegression()
X = df[['CO2 Emissions(g/km)']]  # Input feature
y = df['Cylinders']  # Target variable
regression_model.fit(X, y)

# Plot the linear regression line
plt.plot(X, regression_model.predict(X), color='red')

plt.savefig('fig_scatter_Linear_Regression_2_Cylinders_Fuel_CO2.pdf')
plt.close()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7385 entries, 0 to 7384
Data columns (total 13 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   uuid                              7385 non-null   object
 1   Make                              7385 non-null   object
 2   Model                             7385 non-null   object
 3   Vehicle Class                     7385 non-null   object
 4   Engine Size(L)                    7385 non-null   float64
 5   Cylinders                         7385 non-null   int64
 6   Transmission                      7385 non-null   object
 7   Fuel Type                         7385 non-null   object
 8   Fuel Consumption City (L/100 km)  7385 non-null   float64
 9   Fuel Consumption Hwy (L/100 km)   7385 non-null   float64
 10  Fuel Consumption Comb (L/100 km)  7385 non-null   float64
 11  Fuel Consumption Comb (mpg)       7385 non-null   int64
 12  CO2 Emissions(g/km)               7385 non-null   int64
dtypes: float64(4), int64(3), object(6)
memory usage: 750.2+ KB
None
// complete ........ data model cylinders mean:  5.615030467163169
```

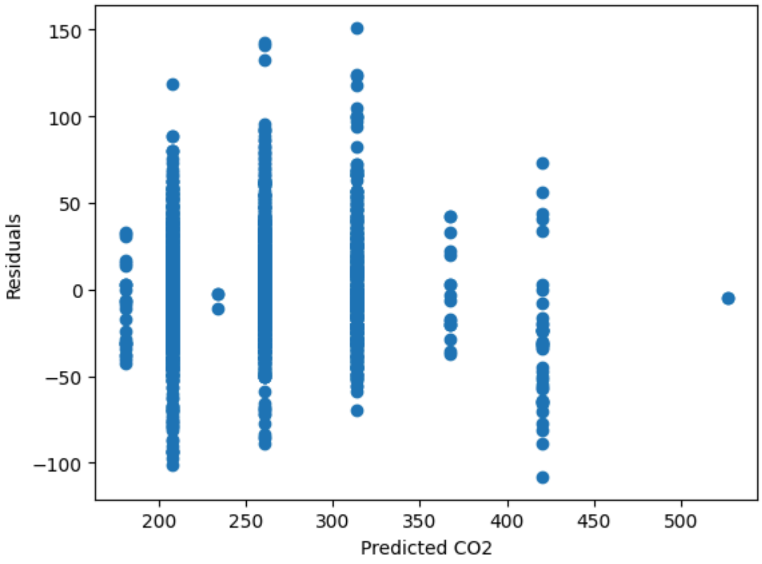
```
# Import the data
df = pd.read_csv('/content/drive/MyDrive/cars_dataset.csv')
print(df.info())

# Get Cylinders count
dMean_cylinders = df['Cylinders'].mean()
print('// complete ........ data model cylinders mean: ', dMean_cylinders)

# Generate a scatter plot
fig, ax = plt.subplots()
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.yaxis.set_ticks_position('left')
ax.xaxis.set_ticks_position('bottom')
sns.scatterplot(x='Fuel Consumption Comb (L/100 km)', y='CO2 Emissions(g/km)', data=df, alpha=0.5, color='grey')

# Perform linear regression
regression_model = LinearRegression()
X = df[['Fuel Consumption Comb (L/100 km)']]  # Input feature
y = df['CO2 Emissions(g/km)']  # Target variable
regression_model.fit(X, y)

# Plot the linear regression line
plt.plot(X, regression_model.predict(X), color='red')

plt.savefig('fig_scatter_Linear_Regression_2CO2_Emissions_Fuel_CO2.pdf')
plt.close()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7385 entries, 0 to 7384
```

```
Data columns (total 13 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   uuid                               7385 non-null   object 
 1   Make                               7385 non-null   object 
 2   Model                              7385 non-null   object 
 3   Vehicle Class                      7385 non-null   object 
 4   Engine Size(L)                     7385 non-null   float64
 5   Cylinders                          7385 non-null   int64  
 6   Transmission                       7385 non-null   object 
 7   Fuel Type                          7385 non-null   object 
 8   Fuel Consumption City (L/100 km)  7385 non-null   float64
 9   Fuel Consumption Hwy (L/100 km)   7385 non-null   float64
 10  Fuel Consumption Comb (L/100 km)  7385 non-null   float64
 11  Fuel Consumption Comb (mpg)       7385 non-null   int64  
 12  CO2 Emissions(g/km)               7385 non-null   int64  
dtypes: float64(4), int64(3), object(6)
memory usage: 750.2+ KB
None
// complete ........ data model cylinders mean:  5.615030467163169
```

```
#Residual Plot, r-squared and RMSE for Evaluation Metrics (Capstone)
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

df = pd.read_csv('/content/drive/MyDrive/cars_dataset.csv')

# Load data
X = df[['Cylinders']].values
y = df['CO2 Emissions(g/km)'].values

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Model evaluation metrics
r_squared = model.score(X_test, y_test)
print(f"R-Squared: {r_squared}")

y_pred = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"RMSE: {rmse}")

# Residual plot
residuals = y_test - y_pred
plt.scatter(y_pred, residuals)
plt.xlabel("Predicted CO2")
plt.ylabel("Residuals")

# Feature importance
print("Slope", model.coef_)
print("Intercept", model.intercept_)
```

```
R-Squared: 0.7061506959702248
RMSE: 31.655771694980565
Slope [26.57554309]
Intercept 101.26577344379822
```



```
#Milestone 6 Code
# ------------------------------------------
# Pre-Processing
# ------------------------------------------
# Data pre-processing
# Let's check the dataset for missing values.
# @code: 0, or ‘index’: Drop rows which contain missing values.
# Let's see where the Null values are.
# Let's see the data shape and NaN values.
# This will give number of NaN values in every column.
df_null_values = df.isnull().sum()
print('NANs?', df_null_values)

# Show missing values in a figure
# plt.figure(figsize=(15,5))
# sns.heatmap(df.isnull(), cbar=False, yticklabels=False, cmap='Greys')
# plt.xticks(rotation=45, fontsize=6)
# plt.tight_layout()
# plt.savefig('fig_MissingValues.pdf')
# plt.close()

# Drop all rows with NaN.
df = df.dropna(axis=0)
df_null_values = df.isnull().sum()
print('NANs_After_Update?', df_null_values)
print('// complete ........ Pre-Processing')
```

```
NANs? uuid                                0
Make                                0
Model                               0
Vehicle Class                       0
Engine Size(L)                      0
Cylinders                           0
Transmission                        0
Fuel Type                           0
Fuel Consumption City (L/100 km)    0
Fuel Consumption Hwy (L/100 km)     0
Fuel Consumption Comb (L/100 km)    0
Fuel Consumption Comb (mpg)         0
CO2 Emissions(g/km)                 0
dtype: int64
NANs_After_Update? uuid                                0
Make                                0
Model                               0
Vehicle Class                       0
Engine Size(L)                      0
Cylinders                           0
Transmission                        0
Fuel Type                           0
Fuel Consumption City (L/100 km)    0
Fuel Consumption Hwy (L/100 km)     0
Fuel Consumption Comb (L/100 km)    0
Fuel Consumption Comb (mpg)         0
CO2 Emissions(g/km)                 0
dtype: int64
// complete ........ Pre-Processing
```

```
# Check for missing values
missing_values = df.isnull().sum()
print("Missing values:\n", missing_values)

# Check datatypes
datatypes = df.dtypes
print("Datatypes:\n", datatypes)
```

```
Missing values:
 uuid                                0
Make                                0
```