Assignment : `FDS LAB 3`

Name : `Ayush Panchal`

Roll No : `P24DS013`

Importing necessary libraries

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Importing Dataset

In [ ]:

```
data = pd.read_csv("dataset/data.csv")
data.head()
```

Out[ ]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Nu D
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	

Removing non-numerical columns from the dataset

In [ ]:

```
data = data.select_dtypes(include='number')

# Removing The Year Column also
data = data.drop(["Year"], axis = 1)
```

```
data.head()
```

Out[ ]:

	Engine HP	Engine Cylinders	Number of Doors	highway MPG	city mpg	Popularity	MSRP
0	335.0	6.0	2.0	26	19	3916	46135
1	300.0	6.0	2.0	28	19	3916	40650
2	300.0	6.0	2.0	28	20	3916	36350
3	230.0	6.0	2.0	28	18	3916	29450
4	230.0	6.0	2.0	28	18	3916	34500

1. Use built-in functions of python libraries, to perform all arithmetic and statistical operations performed in assignment

2. Display the results

In [ ]:

```
data.columns
```

Out[ ]:

```
Index(['Engine HP', 'Engine Cylinders', 'Number of Doors', 'highway MPG',
       'city mpg', 'Popularity', 'MSRP'],
      dtype='object')
```

In [ ]:

```
for idx, column in enumerate(data.columns):
    mean = data[column].mean()
    median = data[column].median()
    mode = data[column].mode()
    min = data[column].min()
    max = data[column].max()
    var = data[column].var()
    sd = var ** (1/2)

    print(f"Mean of Column '{column}' = {mean}")
    print(f"Midean of Column '{column}' = {median}")
    print(f"Mode of Column '{column}' = {mode[0]}")
    print(f"Minimum of Column '{column}' = {min}")
    print(f"Maximum of Column '{column}' = {max}")
    print(f"Variance of Column '{column}' = {var}")
    print(f"Standard Deviation of Column '{column}' = {sd}")
    print()
```

```
Mean of Column 'Engine HP' = 249.38607007176023
Midean of Column 'Engine HP' = 227.0
Mode of Column 'Engine HP' = 200.0
Minimum of Column 'Engine HP' = 55.0
Maximum of Column 'Engine HP' = 1001.0
Variance of Column 'Engine HP' = 11922.864530695864
Standard Deviation of Column 'Engine HP' = 109.19187025917206

Mean of Column 'Engine Cylinders' = 5.628828677213059
Midean of Column 'Engine Cylinders' = 6.0
Mode of Column 'Engine Cylinders' = 4.0
Minimum of Column 'Engine Cylinders' = 0.0
Maximum of Column 'Engine Cylinders' = 16.0
Variance of Column 'Engine Cylinders' = 3.170391592627014
Standard Deviation of Column 'Engine Cylinders' = 1.780559348246223

Mean of Column 'Number of Doors' = 3.4360933825999327
Midean of Column 'Number of Doors' = 4.0
Mode of Column 'Number of Doors' = 4.0
Minimum of Column 'Number of Doors' = 2.0
Maximum of Column 'Number of Doors' = 4.0
Variance of Column 'Number of Doors' = 0.7767168106289198
Standard Deviation of Column 'Number of Doors' = 0.8813153865835543

Mean of Column 'highway MPG' = 26.637485311398354
Midean of Column 'highway MPG' = 26.0
Mode of Column 'highway MPG' = 24
Minimum of Column 'highway MPG' = 12
Maximum of Column 'highway MPG' = 354
Variance of Column 'highway MPG' = 78.552782595478
Standard Deviation of Column 'highway MPG' = 8.863000766979432

Mean of Column 'city mpg' = 19.73325499412456
Midean of Column 'city mpg' = 18.0
Mode of Column 'city mpg' = 17
Minimum of Column 'city mpg' = 7
Maximum of Column 'city mpg' = 137
Variance of Column 'city mpg' = 80.7805157702785
Standard Deviation of Column 'city mpg' = 8.987798160299246

Mean of Column 'Popularity' = 1554.9111969111968
Midean of Column 'Popularity' = 1385.0
Mode of Column 'Popularity' = 1385
Minimum of Column 'Popularity' = 2
Maximum of Column 'Popularity' = 5657
Variance of Column 'Popularity' = 2078946.8405981981
Standard Deviation of Column 'Popularity' = 1441.8553466274618

Mean of Column 'MSRP' = 40594.737032063116
Midean of Column 'MSRP' = 29995.0
Mode of Column 'MSRP' = 2000
Minimum of Column 'MSRP' = 2000
Maximum of Column 'MSRP' = 2065902
Variance of Column 'MSRP' = 3613104336.034846
Standard Deviation of Column 'MSRP' = 60109.103603654294
```

3. Use any two different mean formulas and display both mean values for first (25%, 50% and 75%) and 100% data. And compare both the mean values for all subsets

In [ ]:

```
# Function to calculate harmonic mean
def harmonic_mean(data):
    return len(data) / np.sum(1.0 / data)

# Function to calculate and compare means for different data subsets
def compare_means(column):
    data_length = len(column)

    # Subsets: 25%, 50%, 75%, 100% of the data
    subsets = {
        '25%': column.iloc[:int(data_length * 0.25)],
        '50%': column.iloc[:int(data_length * 0.50)],
        '75%': column.iloc[:int(data_length * 0.75)],
        '100%': column
    }

    comparison_results = {}

    for subset_name, subset in subsets.items():
        # Arithmetic mean
        arithmetic_mean = subset.mean()

        # Harmonic mean
        harmonic_mean_value = harmonic_mean(subset)

        # Store the results
        comparison_results[subset_name] = {
            'Arithmetic Mean': arithmetic_mean,
            'Harmonic Mean': harmonic_mean_value
        }

    return comparison_results

for idx, column in enumerate(data.columns):
    # Calculate and compare means for the 'column_name' column
    comparison_results = compare_means(data[column])

    # Display the results
    for subset_name, means in comparison_results.items():
        print(f"{subset_name} data:")
        print(f"Arithmetic Mean: {means['Arithmetic Mean']}")
        print(f"Harmonic Mean: {means['Harmonic Mean']}")
        print()
```

```
25% data:
Arithmetic Mean: 248.69808145405588
Harmonic Mean: 208.07718313897806

50% data:
Arithmetic Mean: 250.9924089068826
Harmonic Mean: 211.55583697402764

75% data:
Arithmetic Mean: 247.76440085672417
Harmonic Mean: 209.9321360830685

100% data:
Arithmetic Mean: 249.38607007176023
Harmonic Mean: 210.84784938003503

25% data:
Arithmetic Mean: 5.455645161290323
Harmonic Mean: 0.0

50% data:
Arithmetic Mean: 5.54808338937458
Harmonic Mean: 0.0

75% data:
Arithmetic Mean: 5.5544076361594605
Harmonic Mean: 0.0

100% data:
Arithmetic Mean: 5.628828677213059
Harmonic Mean: 0.0

25% data:
Arithmetic Mean: 3.151443922095366
Harmonic Mean: 2.812750885478158

50% data:
Arithmetic Mean: 3.3396574882471457
Harmonic Mean: 3.016457084986075

75% data:
Arithmetic Mean: 3.3944450666368016
Harmonic Mean: 3.0820086809048837

100% data:
Arithmetic Mean: 3.4360933825999327
Harmonic Mean: 3.1353319151735786

25% data:
Arithmetic Mean: 27.988918737407655
Harmonic Mean: 26.19355479099412

50% data:
Arithmetic Mean: 27.358569749874096
Harmonic Mean: 25.532455783521034

75% data:
Arithmetic Mean: 27.22216004476777
Harmonic Mean: 25.30826975155105
```

```
100% data:
Arithmetic Mean: 26.637485311398354
Harmonic Mean: 24.842286105065263

25% data:
Arithmetic Mean: 20.21188717259906
Harmonic Mean: 18.591233720443803

50% data:
Arithmetic Mean: 20.010407923451403
Harmonic Mean: 18.28022834529847

75% data:
Arithmetic Mean: 20.114605484051484
Harmonic Mean: 18.217011096986358

100% data:
Arithmetic Mean: 19.73325499412456
Harmonic Mean: 17.97832397387949

25% data:
Arithmetic Mean: 1599.5003357958362
Harmonic Mean: 441.8822607736223

50% data:
Arithmetic Mean: 1748.2471042471043
Harmonic Mean: 421.1205543557981

75% data:
Arithmetic Mean: 1571.077112479015
Harmonic Mean: 401.1811857679068

100% data:
Arithmetic Mean: 1554.9111969111968
Harmonic Mean: 436.71877734825364

25% data:
Arithmetic Mean: 43974.78676964406
Harmonic Mean: 9458.716601716491

50% data:
Arithmetic Mean: 42355.817357730404
Harmonic Mean: 11553.6463893341

75% data:
Arithmetic Mean: 41422.66435366536
Harmonic Mean: 10607.369840770223

100% data:
Arithmetic Mean: 40594.737032063116
Harmonic Mean: 11170.957367181609
```

Perform following:

- A. Display all fields of dataset and prepare new sub set of datasets as per user’s choice

- `i. One column`
- `ii. Two columns`
- B. Display total number of rows of dataset
- `i. First row, Last row`
- `ii. Prepare new subset of datasetsAs per user’s choice: N rows from the beginning, N rows from the end and N random rows`
- C. Prepare new sub set of datasets with N rows and M columns as per the user’s choice

In [ ]:

```
data.shape
```

Out[ ]:

```
(11914, 7)
```

In [ ]:

```
data.head()
```

Out[ ]:

	Engine HP	Engine Cylinders	Number of Doors	highway MPG	city mpg	Popularity	MSRP
0	335.0	6.0	2.0	26	19	3916	46135
1	300.0	6.0	2.0	28	19	3916	40650
2	300.0	6.0	2.0	28	20	3916	36350
3	230.0	6.0	2.0	28	18	3916	29450
4	230.0	6.0	2.0	28	18	3916	34500

One column

In [ ]:

```
one_subset_columns = ["Engine Cylinders"]

subset_data = data[one_subset_columns]
subset_data.head()
```

Out[ ]:

	Engine Cylinders
0	6.0
1	6.0
2	6.0
3	6.0
4	6.0

Two columns

In [ ]:

```python
two_subset_columns = ["Number of Doors", "Popularity"]
subset_data = data[two_subset_columns]
subset_data.head()
```

Out[ ]:

	Number of Doors	Popularity
0	2.0	3916
1	2.0	3916
2	2.0	3916
3	2.0	3916
4	2.0	3916

Number of rows

In [ ]:

```python
data.shape[0]
```

Out[ ]:

```
11914
```

First Row

In [ ]:

```python
data.head(1)
```

Out[ ]:

	Engine HP	Engine Cylinders	Number of Doors	highway MPG	city mpg	Popularity	MSRP
0	335.0	6.0	2.0	26	19	3916	46135

Last Row

In [ ]:

```python
data.tail(1)
```

Out[ ]:

	Engine HP	Engine Cylinders	Number of Doors	highway MPG	city mpg	Popularity	MSRP
11913	221.0	6.0	4.0	26	17	61	28995

N rows from the beginning, N rows from the end and N random rows

In [ ]:

```python
n_rows = 10

n_rows_beginning_data = data.head(n_rows)
n_rows_ending_data = data.tail(n_rows)
n_rows_random_data = data.sample(n_rows)

n_rows_beginning_data
```

Out[ ]:

	Engine HP	Engine Cylinders	Number of Doors	highway MPG	city mpg	Popularity	MSRP
0	335.0	6.0	2.0	26	19	3916	46135
1	300.0	6.0	2.0	28	19	3916	40650
2	300.0	6.0	2.0	28	20	3916	36350
3	230.0	6.0	2.0	28	18	3916	29450
4	230.0	6.0	2.0	28	18	3916	34500
5	230.0	6.0	2.0	28	18	3916	31200
6	300.0	6.0	2.0	26	17	3916	44100
7	300.0	6.0	2.0	28	20	3916	39300
8	230.0	6.0	2.0	28	18	3916	36900
9	230.0	6.0	2.0	27	18	3916	37200

In [ ]:

```
n_rows_ending_data
```

Out[ ]:

	Engine HP	Engine Cylinders	Number of Doors	highway MPG	city mpg	Popularity	MSRP
11904	394.0	8.0	2.0	19	12	3916	130000
11905	394.0	8.0	2.0	19	12	3916	131500
11906	300.0	6.0	4.0	23	16	204	46020
11907	300.0	6.0	4.0	23	16	204	56570
11908	300.0	6.0	4.0	23	16	204	50520
11909	300.0	6.0	4.0	23	16	204	46120
11910	300.0	6.0	4.0	23	16	204	56670
11911	300.0	6.0	4.0	23	16	204	50620
11912	300.0	6.0	4.0	23	16	204	50920
11913	221.0	6.0	4.0	26	17	61	28995

In [ ]:

```
n_rows_random_data
```

Out[ ]:

	Engine HP	Engine Cylinders	Number of Doors	highway MPG	city mpg	Popularity	MSRP
11321	155.0	6.0	4.0	25	17	481	17199
1586	200.0	4.0	4.0	39	40	2031	37800
3076	130.0	4.0	2.0	39	36	2202	24140
6387	NaN	0.0	4.0	101	126	2009	28980
4178	420.0	8.0	4.0	22	15	1624	73395
5074	325.0	6.0	2.0	24	16	190	50200
7292	110.0	4.0	2.0	34	24	2009	2000
2147	290.0	6.0	4.0	28	20	1720	38990
4918	NaN	6.0	4.0	21	16	5657	29030
3060	185.0	4.0	4.0	31	25	2202	25845

In [ ]:

```
col_subset = ["Engine HP", "Number of Doors", "city mpg"]
row_subset = 15

data_subset = data[col_subset].sample(row_subset)
data_subset
```

Out[ ]:

	Engine HP	Number of Doors	city mpg
4609	220.0	2.0	12
9580	285.0	4.0	17
10170	159.0	4.0	19
3815	170.0	4.0	17
8604	192.0	4.0	18
3115	278.0	4.0	19
3266	335.0	4.0	19
6379	74.0	2.0	24
5070	536.0	4.0	12
7259	290.0	2.0	16
4247	122.0	4.0	23
5122	145.0	4.0	21
11767	304.0	4.0	18
9468	295.0	4.0	14
3491	160.0	4.0	28

4. Using built-in functions like describe(), generate the summary for the numeric columns in the dataset.

In [ ]:

```
data.describe()
```

Out[ ]:

	Engine HP	Engine Cylinders	Number of Doors	highway MPG	city mpg	Popular
count	11845.00000	11884.000000	11908.000000	11914.000000	11914.000000	11914.0000
mean	249.38607	5.628829	3.436093	26.637485	19.733255	1554.9111
std	109.19187	1.780559	0.881315	8.863001	8.987798	1441.8553
min	55.00000	0.000000	2.000000	12.000000	7.000000	2.0000
25%	170.00000	4.000000	2.000000	22.000000	16.000000	549.0000
50%	227.00000	6.000000	4.000000	26.000000	18.000000	1385.0000
75%	300.00000	6.000000	4.000000	30.000000	22.000000	2009.0000
max	1001.00000	16.000000	4.000000	354.000000	137.000000	5657.0000

5. Without using built-in functions, perform all the operations displayed in the above question.

In [ ]:

```
def describe_manual(df):
    # Selecting only numerical columns
    numeric_df = df.select_dtypes(include=[np.number])

    # Initialize an empty dictionary to store the results
    description = {}

    # Calculate the descriptive statistics
    description['count'] = numeric_df.count()
    description['mean'] = numeric_df.mean()
    description['std'] = numeric_df.std()
    description['min'] = numeric_df.min()
    description['25%'] = numeric_df.quantile(0.25)
    description['50%'] = numeric_df.median()
    description['75%'] = numeric_df.quantile(0.75)
    description['max'] = numeric_df.max()

    # Convert the dictionary to a DataFrame for a similar format to pd.DataFrame
    description_df = pd.DataFrame(description)

    return description_df.T

describe_manual(data)
```

Out[ ]:

	Engine HP	Engine Cylinders	Number of Doors	highway MPG	city mpg	Popular
count	11845.00000	11884.000000	11908.000000	11914.000000	11914.000000	11914.0000
mean	249.38607	5.628829	3.436093	26.637485	19.733255	1554.9111
std	109.19187	1.780559	0.881315	8.863001	8.987798	1441.8553
min	55.00000	0.000000	2.000000	12.000000	7.000000	2.0000
25%	170.00000	4.000000	2.000000	22.000000	16.000000	549.0000
50%	227.00000	6.000000	4.000000	26.000000	18.000000	1385.0000
75%	300.00000	6.000000	4.000000	30.000000	22.000000	2009.0000
max	1001.00000	16.000000	4.000000	354.000000	137.000000	5657.0000