```
import numpy as np
import matplotlib as mt
import matplotlib.pyplot as plt
import pandas as pdd
```

1- Data Gathering

a- Loding Data

```
#load Data Tables without column header
df = pd.read_csv("/data_science.csv", sep=",", header=None)

#load Data Tables as file with name "header"
df = pd.read_csv("/data_science.csv", sep=",", \
    names=["symboling", "normalized-losses", "make", "fuel-type", "aspiration", "num-of-doors", "body-style", "drive-wheels", "engine-location", "wheel-base", "length", "width", "height", "curb-weight", "engine-type", "num-of-cylinders", "engine-size", "fuel-system", "bore", "stroke", "compression-ratio", "horsepower", "peak-rpm", "city-mpg", "highway-mpg", "price"])

#Display the of columns named in the "header"
df.columns, df.shape
# printing
df.head()
```

- `sep=","` : Removes any leading and trailing whitespace characters from the string ("a", "x")
- `sep=r"\s+"` : Splits the string into a list of substrings based on the specified delimiter ( )

b- Data Preview

- quick look at data

```
df.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height	curb-weight	engine-type	num-of-cylinders	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
0	3	?	alfa-romeo	gas	std	two	convertible	rwd	front	88.6	168	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romeo	gas	std	two	convertible	rwd	front	88.6	168	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romeo	gas	std	two	hatchback	rwd	front	94.5	171	65.5	52	2823	ohcv	six	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	176	66.2	54.3	2337	ohc	four	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	176	66.4	54.3	2824	ohc	five	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

5 rows × 26 columns

```
df.tail()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height	curb-weight	engine-type	num-of-cylinders	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	188.8	68.9	55.5	2952	ohc	four	141	mpfi	3.78	3.15	9.5	114	5400	23	28	16845
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3049	ohc	four	141	mpfi	3.78	3.15	8.7	160	5300	19	25	19045
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3012	ohcv	six	173	mpfi	3.58	2.87	8.8	134	5500	18	23	21485
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3086	ohc	four	145	idi	3.01	3.40	23.0	106	4800	26	27	22470
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3084	ohcv	six	141	mpfi	3.78	3.15	9.5	114	5400	19	25	22625

5 rows × 26 columns

- column heads has need to be clear
- **normalized-losses** has "?" instead of nulls

2- Data Inspect

```
df.shape
```

```
(205, 26)
```

```
df.describe()
```

symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count 205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean 0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.179512	30.686829
std 1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.423220	6.815150
min -2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25% 0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50% 1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75% 2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max 3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

3- Data Backup

```
# Make a backup of my original DataFrame
df_backup = df.copy()
```

4- Quality Issues

- Handle Column Heads
- Missing Data
- Dublicate Data
- Inconsistent Data Types
- Inconsistent Data
- or Outliers and Anomalies

4-i Handle Column Heads

i-a Define

- Column Heads has extra characters

i-b Code

```
# remove any unwanted characters from the column names
df.columns = df.columns.str.strip().str.replace("/", "_").str.replace(r'[\s+]', '_', regex=False).str.replace(r'[^\w_]', '', regex=False).str.lower()
```

i-c Test

```
df.columns
```

```
Index(['symboling', 'normalized_losses', 'make', 'fuel_type', 'aspiration',
       'num_of_doors', 'body_style', 'drive_wheels', 'engine_location',
       'wheel_base', 'length', 'width', 'height', 'curb_weight',
       'engine_type', 'num_of_cylinders', 'engine_size', 'fuel_system',
       'bore', 'stroke', 'compression_ratio', 'horsepower', 'peak_rpm',
       'city_mpg', 'highway_mpg', 'price'],
      dtype='object')
```

4-ii Missing Data

ii-a Define

- there are null values replaced by "?"

ii-b Code

```
# replace '?' to Nan so we can know where they are and count how many nulls and how it will affect my insights.
df.replace('?', np.nan, inplace=True)
```

ii-c Test

```
# Check missing values
df.isnull().sum()
```

	0
symboling	0
normalized_losses	41
make	0
fuel_type	0
aspiration	0
num_of_doors	2
body_style	0
drive_wheels	0
engine_location	0
wheel_base	0
length	0
width	0
height	0
curb_weight	0
engine_type	0
num_of_cylinders	0
engine_size	0
fuel_system	0
bore	4
stroke	4
compression_ratio	0
horsepower	2
peak_rpm	2
city_mpg	0
highway_mpg	0
price	4

dtype: int64

- missing values issue
- **normalized-losses** has 41 nulls
- **num-of-doors**, **horsepower**, and **peak-rpm** has only 2 nulls
- **bore**, **stroke** and **price** all of them has 4 nulls

Missing Values will not affect my insights so I will ignore all nulls

4-iii Dublicate Data

```
# check dublicated rows
df.duplicated().sum()
```

```
0
```

- has no dublicated rows

4-iv Inconsistant Data Type

```
df.dtypes
```

	0
symboling	int64
normalized_losses	object
make	object
fuel_type	object
aspiration	object
num_of_doors	object
body_style	object
drive_wheels	object
engine_location	object
wheel_base	float64
length	float64
width	float64
height	float64
curb_weight	int64
engine_type	object
num_of_cylinders	object
engine_size	int64
fuel_system	object
bore	object
stroke	object
compression_ratio	float64
horsepower	object
peak_rpm	object
city_mpg	int64
highway_mpg	int64
price	object

dtype: object

iv-a Define

- columns Data Type Issue
- 'price', 'bore', 'peak-rpm' object need of float
- 'horsepower' object need of int64
- 'num-of-cylinders', 'num-of-doors' numbers in letters

iv-b Code

convert to float

```
# Convert price to numeric, coerce errors to NaN
df['price'] = pd.to_numeric(df['price'], errors='coerce')
```

convert to int

```
# Convert horsepower to numeric, coerce errors to NaN
df['horsepower'] = pd.to_numeric(df['horsepower'], errors='coerce')
```

```
# Convert peak-rpm to numeric, coerce errors to NaN
df['peak-rpm'] = pd.to_numeric(df['peak-rpm'], errors='coerce')
```

```
# Convert the columns to int, handling errors as NaN
df['horsepower'] = df['horsepower'].astype('Int64')
df['peak-rpm'] = df['peak-rpm'].astype('Int64')
```

convert letters to numbers

```
# Convert bore and stroke to numeric, coerce errors to NaN
df['bore'] = pd.to_numeric(df['bore'], errors='coerce')
df['stroke'] = pd.to_numeric(df['stroke'], errors='coerce')
```

```
# Define the mapping dictionary for the values with their corresponding integer values
df['num-of-cylinders'] = df['num-of-cylinders'].map(cylinder_dict)
df['num-of-doors'].unique()
```

	num-of-cylinders
0	4
1	6
2	5
3	3
4	4
...	...
200	4
201	4
202	6
203	6
204	4

205 rows × 1 columns

dtype: int64

iv-c Test

```
df.dtypes
```

	0
symboling	int64
normalized_losses	object
make	object
fuel_type	object
aspiration	object
num_of_doors	object
body_style	object
drive_wheels	object
engine_location	object
wheel_base	float64
length	float64
width	float64
height	float64
curb_weight	int64
engine_type	object
num_of_cylinders	int64
engine_size	int64
fuel_system	object
bore	float64
stroke	float64
compression_ratio	float64
horsepower	Int64
peak_rpm	Int64
city_mpg	int64
highway_mpg	int64
price	float64

dtype: object

iv-d Inconsistant Values

- Has No Inconsistant Values

4-vi Outliers and Anomalies

- Has No Outliers and Anomalies

5- Tidiness issue

- Each variable forms a column
- Each observation forms a row
- Each type of observational unit forms a table

```
df
```

	symboling	normalized_losses	make	fuel_type	aspiration	num_of_doors	body_style	drive_wheels	engine_location	wheel_base	length	width	height	curb_weight	engine_type	num_of_cylinders	engine_size	fuel_system	bore	stroke	compression_ratio	horsepower	peak_rpm	city_mpg	highway_mpg	price
0	3	?	alfa-romeo	gas	std	two	convertible	rwd	front	88.6	168	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68	9.0	111.0	5000.0	21	27	13495.0
1	3	?	alfa-romeo	gas	std	two	convertible	rwd	front	88.6	168	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68	9.0	111.0	5000.0	21	27	16500.0
2	1	?	alfa-romeo	gas	std	two	hatchback	rwd	front	94.5	171	65.5	52	2823	ohcv	six	152	mpfi	2.68	3.47	9.0	154.0	5000.0	19	26	16500.0
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	176	66.2	54.3	2337	ohc	four	109	mpfi	3.19	3.40	10.0	102.0	5500.0	24	30	13950.0
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	176	66.4	54.3	2824	ohc	five	136	mpfi	3.19	3.40	8.0	115.0	5500.0	18	22	17450.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	188.8	68.9	55.5	2952	ohc	four	141	mpfi	3.78	3.15	9.5	114.0	5400.0	23	28	16845.0
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3049	ohc	four	141	mpfi	3.78	3.15	8.7	160.0	5300.0	19	25	19045.0
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3012	ohcv	six	173	mpfi	3.58	2.87	8.8	134.0	5500.0	18	23	21485.0
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3086	ohc	four	145	idi	3.01	3.40	23.0	106.0	4800.0	26	27	22470.0
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3084	ohcv	six	141	mpfi	3.78	3.15	9.5	114.0	5400.0	19	25	22625.0

205 rows × 26 columns

- Data hasn't any Tidiness issues

6- Export as CSV file

```
df.to_csv("/data.csv", index=False)
```

7- Visualization

- Distribution of Car Prices
- Horsepower vs. Price
- Fuel Type Distribution

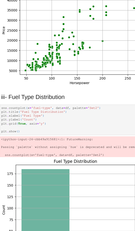
i- Distribution of Car Prices

```
sns.histplot(df['price'], bins=15, kde=True, color='blue')
plt.title('Distribution of Car Prices')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

ii- Horsepower vs. Price

```
sns.scatterplot(df['horsepower'], df['price'], data=df, color='green', edgecolor='r')
plt.title('Horsepower vs. Price')
plt.xlabel('Horsepower')
plt.ylabel('Price')
plt.grid(True)

plt.show()
```



iii- Fuel Type Distribution

```
sns.countplot(df['fuel_type'], palette='Set1')
plt.title('Fuel Type Distribution')
plt.xlabel('Fuel Type')
plt.ylabel('Count')
plt.grid(True, axis='y')

plt.show()
```

```
/tmp/ipykernel_1124/3.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(df['fuel_type'], palette='Set1')
```