Data Visualisation - Graded Questions

`Note` - *This stub file doesn't contain the conceptual questions asked on the platform*

I) Marks Analysis

In the **'Marks.csv'** file, you can find the scores obtained by 200 students in 4 subjects of a standardised test. The different columns - `Score A`, `Score B`, `Score C` and `Score D` indicate the score obtained by a particular student in the respective subjects A, B, C and D.

Load the dataset to your notebook and answer the following questions

```
#Load the necessary Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
#Load the dataset
df1 = pd.read_csv('Marks.csv')
df1.head()
```

	Score A	Score B	Score C	Score D
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

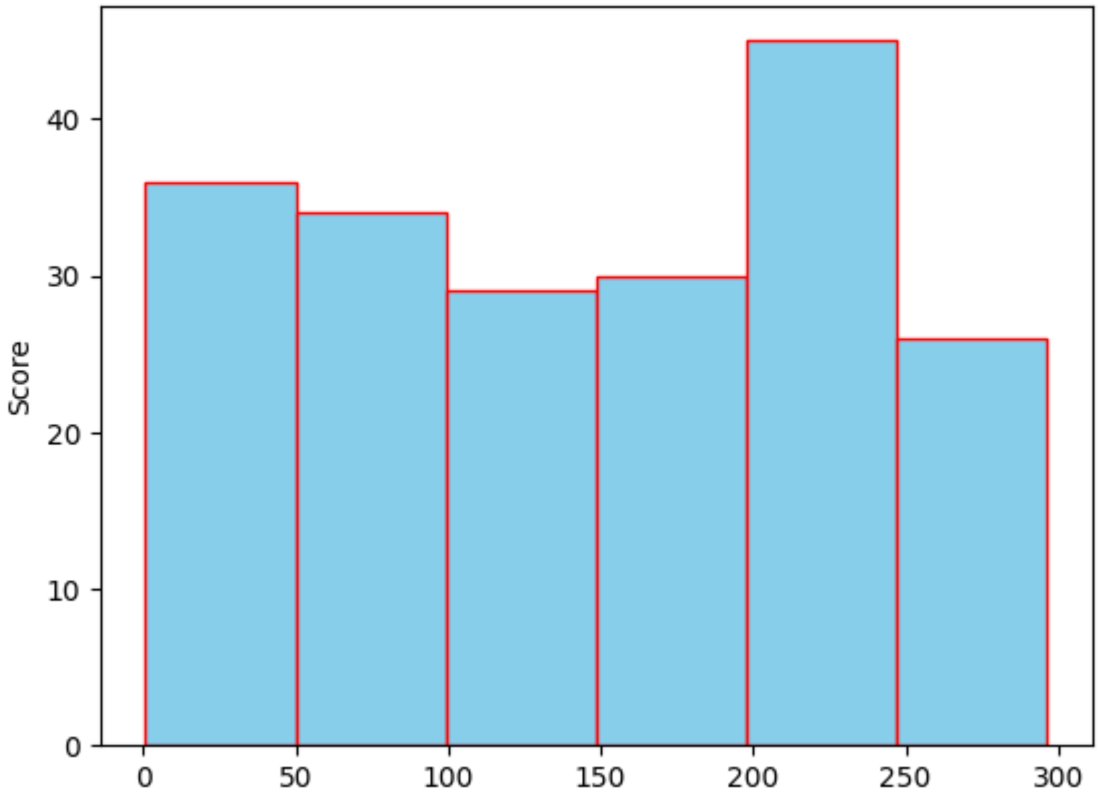
Q1) Load the dataset and plot a histogram for the `Score A` column by keeping the `number of bins to 6`. Which bin range among the following has the highest frequency?

(`Note` - *The bin ranges mentioned in the options are approximate values for the bin ranges that you'll actually get when you plot the histogram*)

```
a)0-50
b)50-100
c)150-200
d)200-250 - Correct
```

```
#Your code here
plt.ylabel('Score')
plt.hist(df1['Score A'], bins=6, color='#87ceeb', edgecolor='red')
```

```
(array([36., 34., 29., 30., 45., 26.]),
 array([  0.7       ,  49.98333333,  99.26666667, 148.55      ,
        197.83333333, 247.11666667, 296.4       ]),
 <BarContainer object of 6 artists>)
```

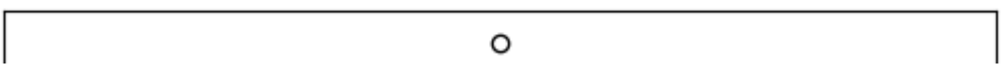


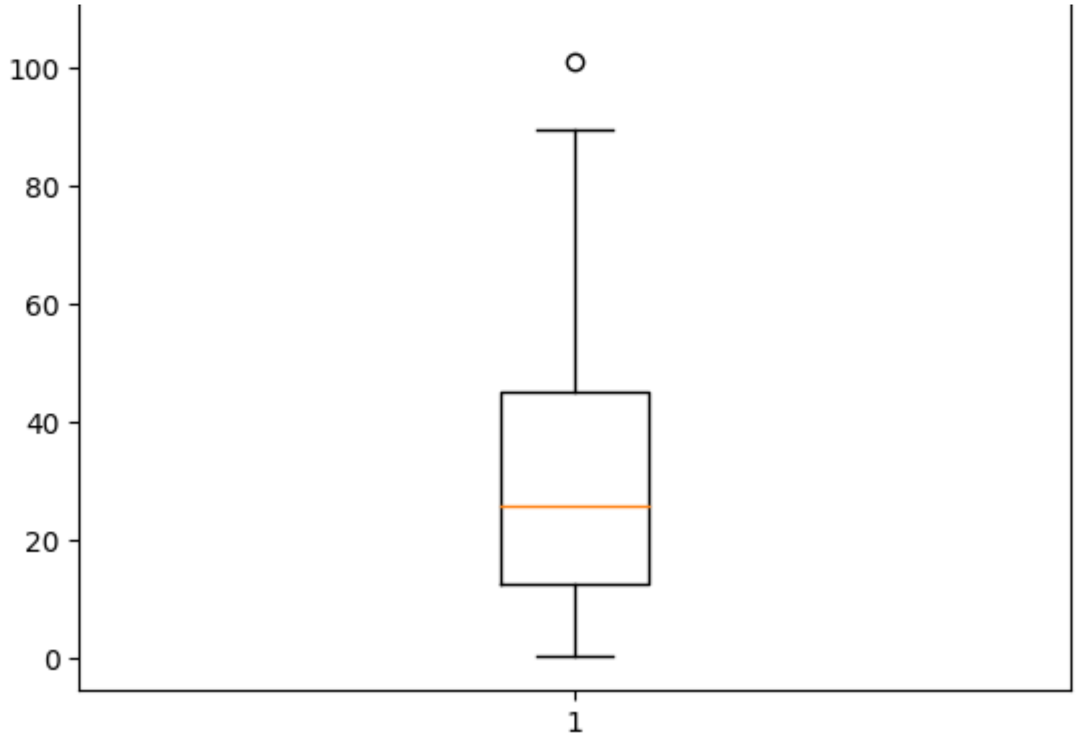
Q2) Plot a box plot for the column `Score C` and choose the correct option.

```
A - The 25th percentile lies between 20 and 40
B - The 75th percentile lies between 40 and 60
C - The 25th percentile lies between 0 and 20
D - Both B and C - Correct
```

```
#Your code here
plt.boxplot(df1['Score C'])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7e0bb2d6a920>,
  <matplotlib.lines.Line2D at 0x7e0bb2d68730>],
 'caps': [<matplotlib.lines.Line2D at 0x7e0bb2d6a110>,
  <matplotlib.lines.Line2D at 0x7e0bb2d686a0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7e0bb2d6a5c0>],
 'medians': [<matplotlib.lines.Line2D at 0x7e0bb2c77940>],
 'fliers': [<matplotlib.lines.Line2D at 0x7e0bb2c75f00>],
 'means': []}
```





II) Superstore Data

In the `superstore.csv` file, you have the details of orders purchased in an American online retail store. Load the dataset, observe and analyse the different columns and answer the following questions.

```
#Load the dataset
df2 = pd.read_csv('superstore.csv')
df2.head()
```

	Order ID	Ship Mode	Segment	Region	Product ID	Sales	Quant
0	CA-2016-152156	Second Class	Consumer	South	FUR-BO-10001798	261.9600	
1	CA-2016-152156	Second Class	Consumer	South	FUR-CH-10000454	731.9400	
2	CA-2016-138688	Second Class	Corporate	West	OFF-LA-10000240	14.6200	
3	US-2015-108966	Standard Class	Consumer	South	FUR-TA-10000577	957.5775	
4	US-2015-108966	Standard Class	Consumer	South	OFF-ST-10000760	22.3680	

Q4) Plot a pie-chart to find the Ship Mode through which most of the orders are being delivered.

a) Standard Class - correct

b) First Class

c) Second Class

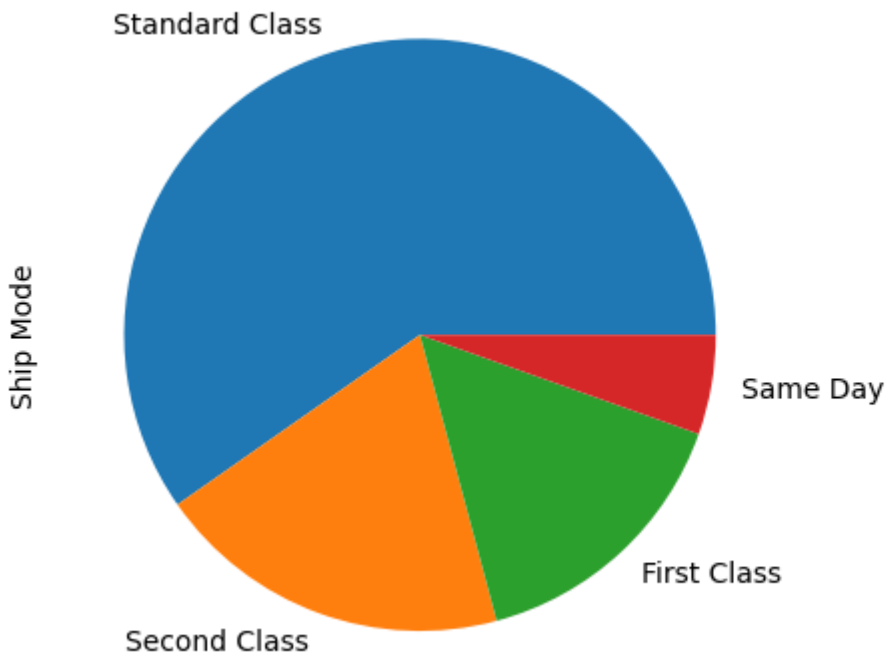
d) Same Day

```
df2['Ship Mode'].value_counts()
```

```
Standard Class    5968
Second Class      1945
First Class       1538
Same Day           543
Name: Ship Mode, dtype: int64
```

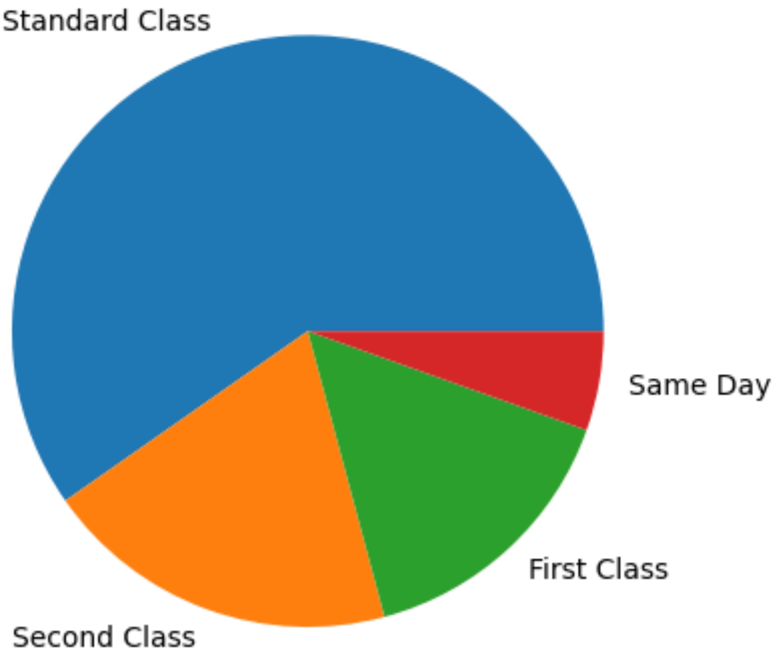
```
#Your code here
df2['Ship Mode'].value_counts().plot.pie()
```

```
<Axes: ylabel='Ship Mode'>
```




```
valcounts = df2['Ship Mode'].value_counts()
labels = valcounts.index
plt.pie(valcounts, labels=labels)
```

```
([<matplotlib.patches.Wedge at 0x7e0bb2cb1690>,
  <matplotlib.patches.Wedge at 0x7e0bb2cb39d0>,
  <matplotlib.patches.Wedge at 0x7e0bb2cb2470>,
  <matplotlib.patches.Wedge at 0x7e0bb2cb3c40>],
 [Text(-0.33056573952035373, 1.0491550370919267, 'Standard Class'),
  Text(-0.37607764230951635, -1.0337144707098356, 'Second Class'),
  Text(0.7465348771572817, -0.8078896441889587, 'First Class'),
  Text(1.0840144265772789, -0.18684946607452133, 'Same Day')])
```



Q5) Plot a bar chart comparing the average `Discount` across all the `Regions` and report back the `Region` getting the highest average discount

Note - You need to clean the `Discount` column first

```
a)Central - correct
b)South
c)West
d)East
```

```
df2.head()
```

	Order ID	Ship Mode	Segment	Region	Product ID	Sales	Quant:
0	CA-2016-152156	Second Class	Consumer	South	FUR-BO-10001798	261.9600	
1	CA-2016-152156	Second Class	Consumer	South	FUR-CH-10000454	731.9400	
2	CA-2016-138688	Second Class	Corporate	West	OFF-LA-10000240	14.6200	
3	US-2015-108966	Standard Class	Consumer	South	FUR-TA-10000577	957.5775	
4	US-2015-108966	Standard Class	Consumer	South	OFF-ST-10000760	22.3680	

```
df2['Discount'].value_counts()
```

```
0%       4798
0.20%    3657
0.70%     418
0.80%     300
0.30%     227
0.40%     206
0.60%     138
0.10%      94
0.50%      66
0.15%      52
0.32%      27
0.45%      11
Name: Discount, dtype: int64
```

```
df2['Discount values'] = df2['Discount'].str.strip('%').astype('float')
df2['Discount values'].value_counts()
```

```
0.00    4798
0.20    3657
0.70     418
0.80     300
0.30     227
0.40     206
0.60     138
0.10      94
0.50      66
0.15      52
0.32      27
0.45      11
Name: Discount values, dtype: int64
```

```
df2.groupby('Region').mean()
```

```
<ipython-input-70-625368922aac>:1: FutureWarning: The default value of nume
  df2.groupby('Region').mean()
```

	Sales	Quantity	Profit	Discount values
Region				
Central	215.772661	3.779595	17.092709	0.240353
East	238.336110	3.728230	32.135808	0.145365
South	241.803645	3.832716	28.857673	0.147253
West	226.493233	3.829535	33.849032	0.109335

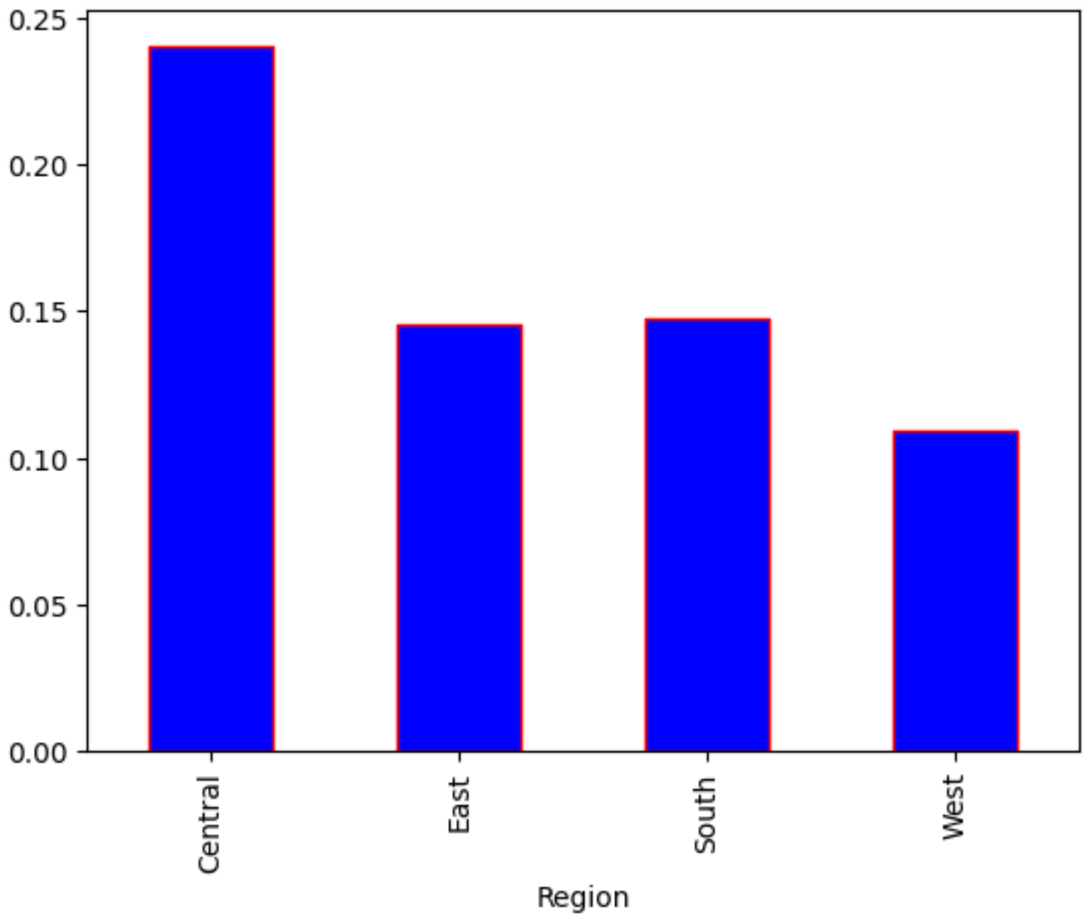
```
region_wise_avg_discount = df2.groupby('Region')['Discount values'].mean()
region_wise_avg_discount
```

```
Region
Central    0.240353
East       0.145365
```

```
South      0.147253
West       0.109335
Name: Discount values, dtype: float64
```

```
# Plot a bar chart comparing the average Discount across all the Regions and
# report back the Region getting the highest average discount
region_wise_avg_discount.plot.bar(color='blue', edgecolor='red')
```

```
<Axes: xlabel='Region'>
```



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

When to use scatter plots?

• Scatter plots are used to convey the relationship between two numerical variables

• Scatter plots are sometimes called correlation plots because they show how two variables are correlated

```
# Read Toyota.csv, Handle "??","????" at the time of reading
cars_data = pd.read_csv('Toyota.csv', na_values=['??', '????'], index_col=0)
```

```
cars_data
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Door
0	13500	23.0	46986.0	Diesel	90.0	1.0	0	2000	thre
1	13750	23.0	72937.0	Diesel	90.0	1.0	0	2000	
2	13950	24.0	41711.0	Diesel	90.0	NaN	0	2000	
3	14950	26.0	48000.0	Diesel	90.0	0.0	0	2000	
4	13750	30.0	38500.0	Diesel	90.0	0.0	0	2000	
...	...	...	...	...	...	...	...	...	
1431	7500	NaN	20544.0	Petrol	86.0	1.0	0	1300	
1432	10845	72.0	NaN	Petrol	86.0	0.0	0	1300	
1433	8500	NaN	17016.0	Petrol	86.0	0.0	0	1300	
1434	7250	70.0	NaN	NaN	86.0	1.0	0	1300	
1435	6950	76.0	1.0	Petrol	110.0	0.0	0	1600	

1436 rows × 10 columns

```
cars_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1436 entries, 0 to 1435
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Price      1436 non-null   int64
 1   Age        1336 non-null   float64
 2   KM         1421 non-null   float64
 3   FuelType   1336 non-null   object
```

```
 4   HP         1430 non-null   float64
 5   MetColor   1286 non-null   float64
 6   Automatic  1436 non-null   int64
 7   CC         1436 non-null   int64
 8   Doors      1436 non-null   object
 9   Weight     1436 non-null   int64
dtypes: float64(4), int64(4), object(2)
memory usage: 123.4+ KB
```

```
cars_data.isnull().sum()
```

```
Price          0
Age          100
KM            15
FuelType     100
HP             6
MetColor     150
Automatic      0
CC             0
Doors          0
Weight         0
dtype: int64
```

```
## Drop null values
cars_data.dropna(inplace=True)
```

```
cars_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1096 entries, 0 to 1435
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Price      1096 non-null   int64
 1   Age        1096 non-null   float64
 2   KM         1096 non-null   float64
 3   FuelType   1096 non-null   object
 4   HP         1096 non-null   float64
 5   MetColor   1096 non-null   float64
 6   Automatic  1096 non-null   int64
 7   CC         1096 non-null   int64
 8   Doors      1096 non-null   object
 9   Weight     1096 non-null   int64
dtypes: float64(4), int64(4), object(2)
memory usage: 94.2+ KB
```

matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, , *edgecolors=None, plotnonfinite=False, data=None,* *kwargs)

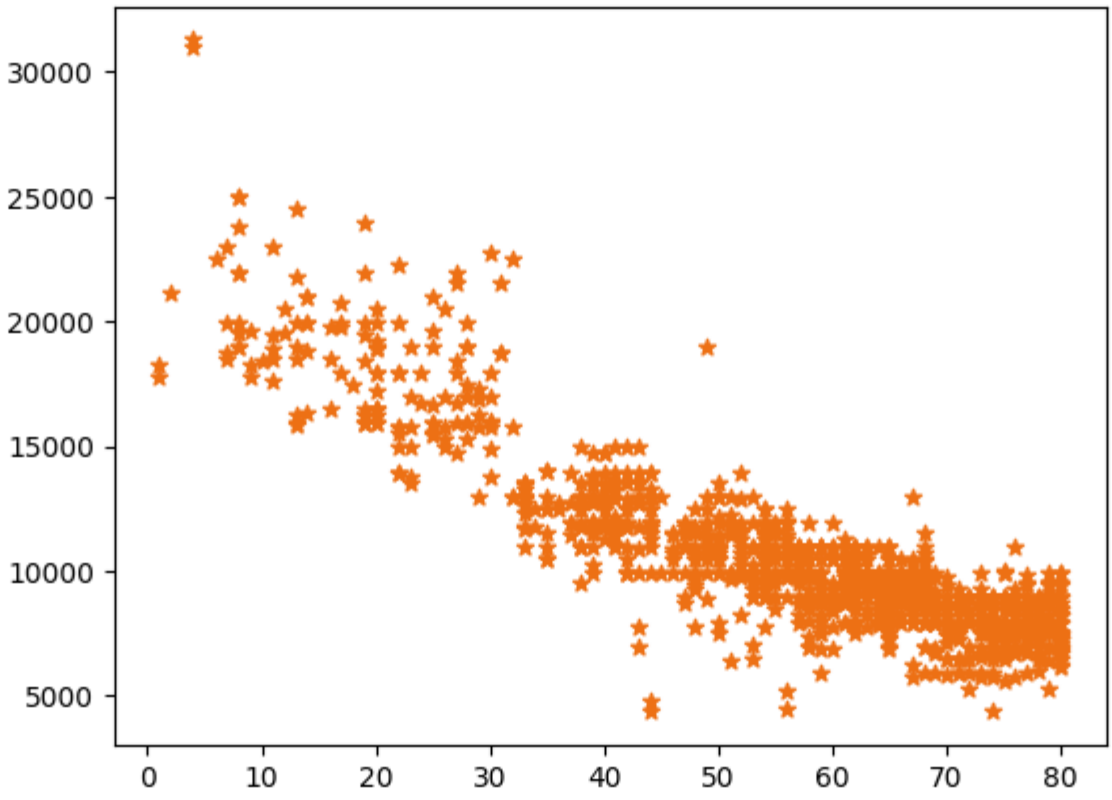
```
cars_data.isnull().sum()
```

```
Price        0
```

```
Age          0
KM           0
FuelType     0
HP           0
MetColor     0
Automatic    0
CC           0
Doors        0
Weight       0
dtype: int64
```

```
#Scatter plot of Age Vs Price
plt.scatter(cars_data['Age'], cars_data['Price'], marker='*', color='#ed7014')
```

```
<matplotlib.collections.PathCollection at 0x7c86dd3e9ae0>
```



write the analysis observed from plot

Histogram

What is a histogram?

• It is a graphical representation of data using bars of different heights

• Histogram groups numbers into ranges and the height of each bar depicts the frequency of each range or bin

When to use histograms?

When to use histograms?

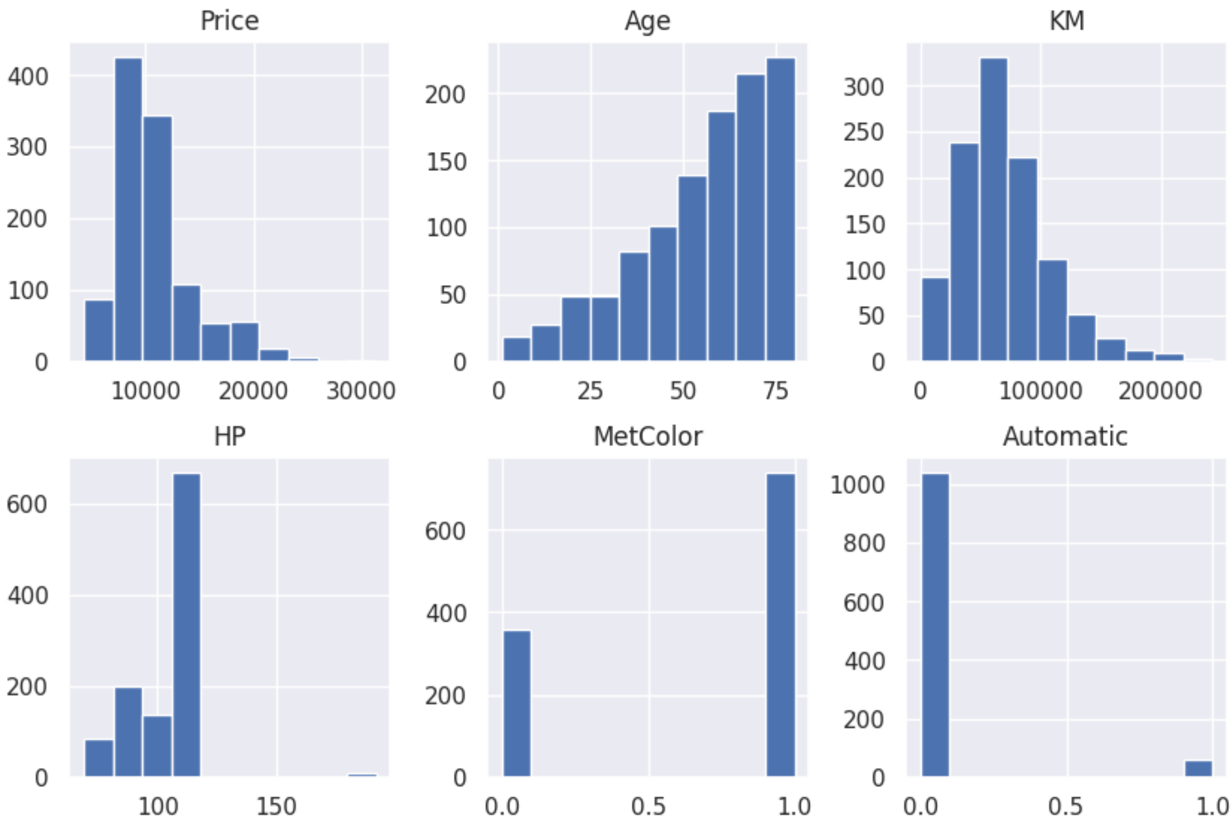
- To represent the frequency distribution of numerical variables

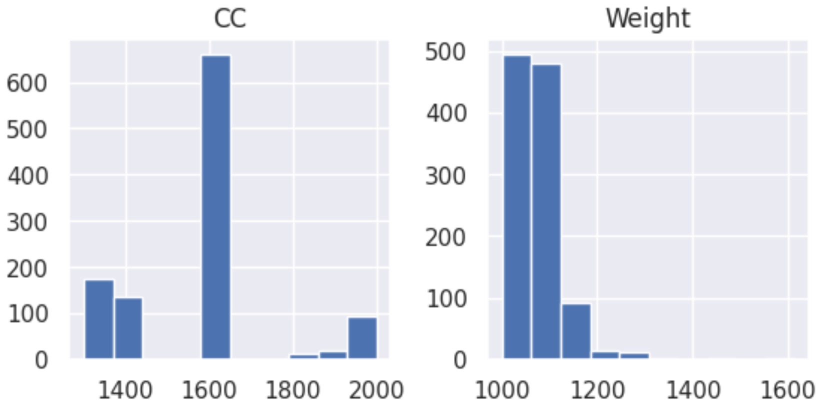
```
cars_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1096 entries, 0 to 1435
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Price      1096 non-null   int64
 1   Age        1096 non-null   float64
 2   KM         1096 non-null   float64
 3   FuelType   1096 non-null   object
 4   HP         1096 non-null   float64
 5   MetColor   1096 non-null   float64
 6   Automatic  1096 non-null   int64
 7   CC         1096 non-null   int64
 8   Doors      1096 non-null   object
 9   Weight     1096 non-null   int64
dtypes: float64(4), int64(4), object(2)
memory usage: 94.2+ KB
```

```
cars_data.hist(figsize=(10, 10))
```

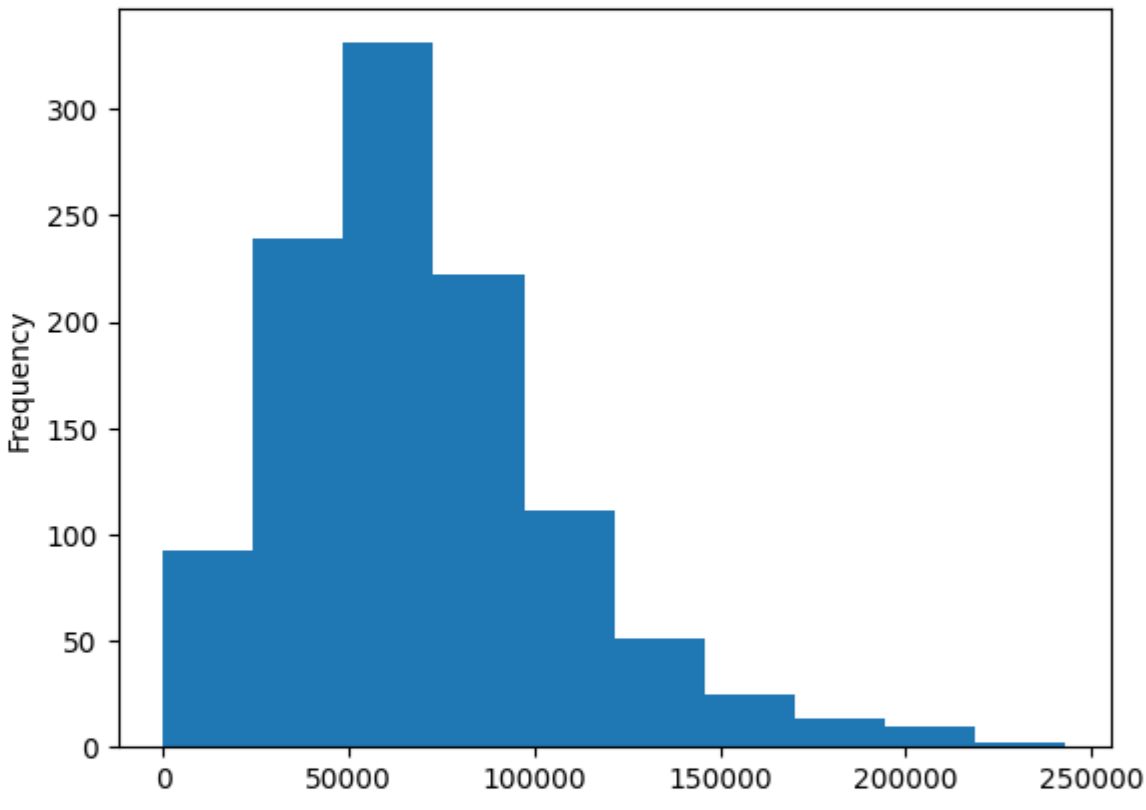
```
array([[<Axes: title={'center': 'Price'}>,
        <Axes: title={'center': 'Age'}>, <Axes: title={'center': 'KM'}>],
       [<Axes: title={'center': 'HP'}>,
        <Axes: title={'center': 'MetColor'}>,
        <Axes: title={'center': 'Automatic'}>],
       [<Axes: title={'center': 'CC'}>,
        <Axes: title={'center': 'Weight'}>, <Axes: >]], dtype=object)
```






```
# do some formatting
cars_data['KM'].plot.hist()
```

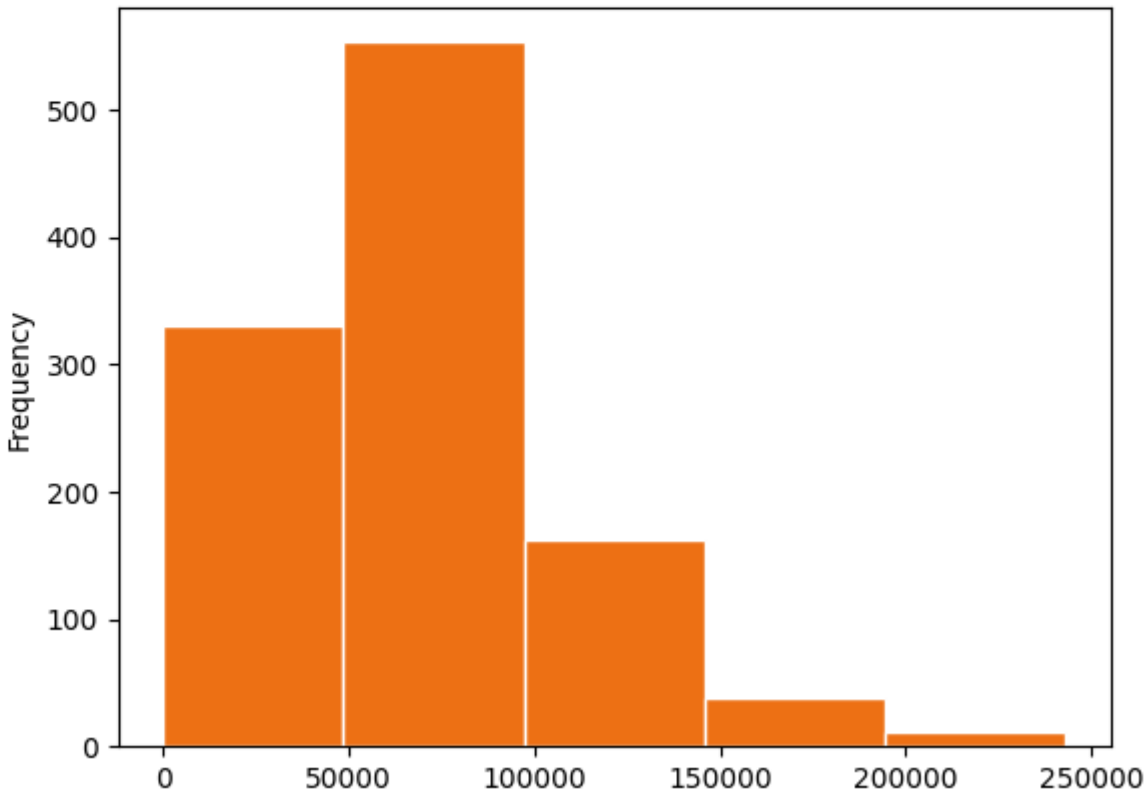
<Axes: ylabel='Frequency'>




```
cars_data['KM'].plot.hist(color='#ed7014', bins=5, edgecolor='white')
```

<Axes: ylabel='Frequency'>

```
<Axes: ylabel='Frequency'>
```



Bar plot

What is a bar plot?

• A bar plot is a plot that presents categorical data with rectangular bars with lengths proportional to the counts that they represent

When to use bar plot?

• To represent the frequency distribution of categorical variables • A bar diagram makes it easy to compare sets of data between different groups

```
cars_data['FuelType'].unique()
```

```
array(['Diesel', 'Petrol', 'CNG'], dtype=object)
```

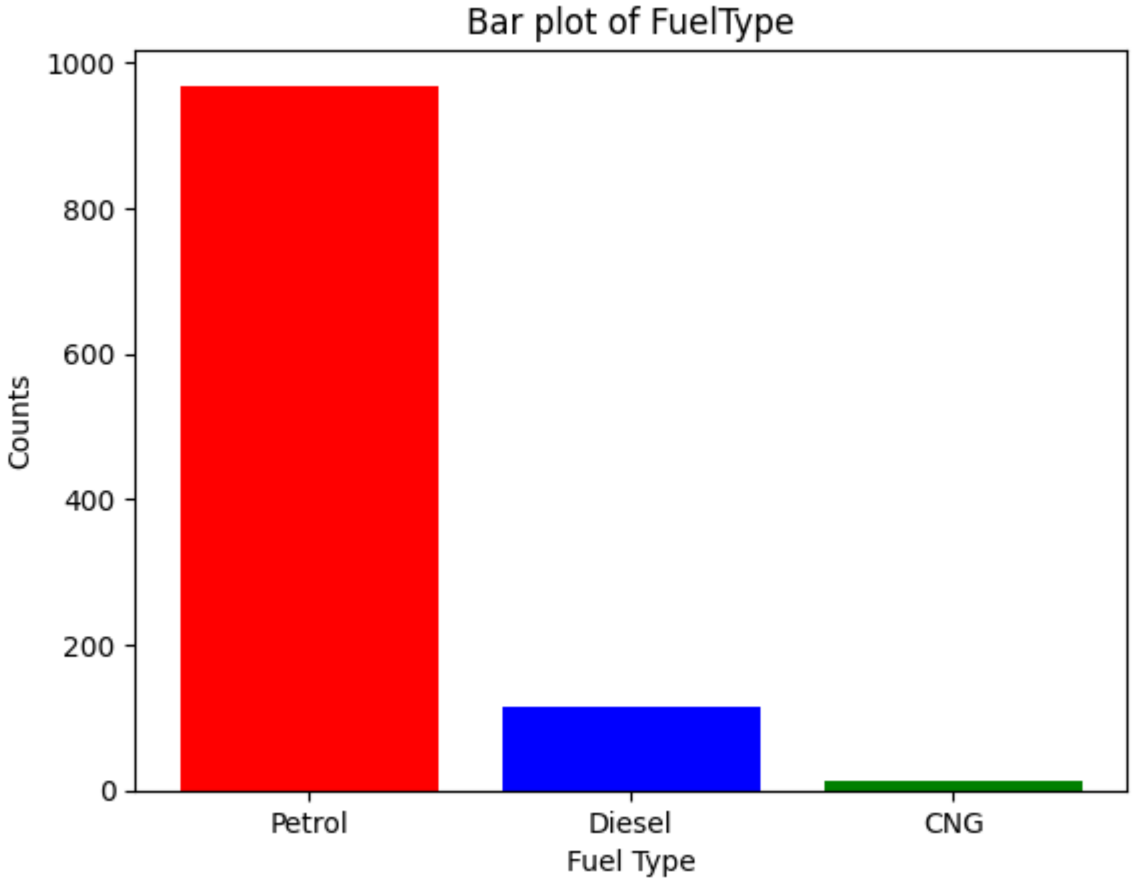
```
cars_data['FuelType'].value_counts()
```

```
Petrol    968
Diesel    116
CNG        12
Name: FuelType, dtype: int64
```

```
# Bar plot of FuelType match the given style
fuels = cars_data['FuelType'].value_counts()
plt.xlabel('Fuel Type')
plt.ylabel('Counts')
```

```
plt.ylabel('Counts')
plt.title('Bar plot of FuelType')
plt.bar(fuels.index, fuels, color=['red', 'blue','green'])
```

```
<BarContainer object of 3 artists>
```

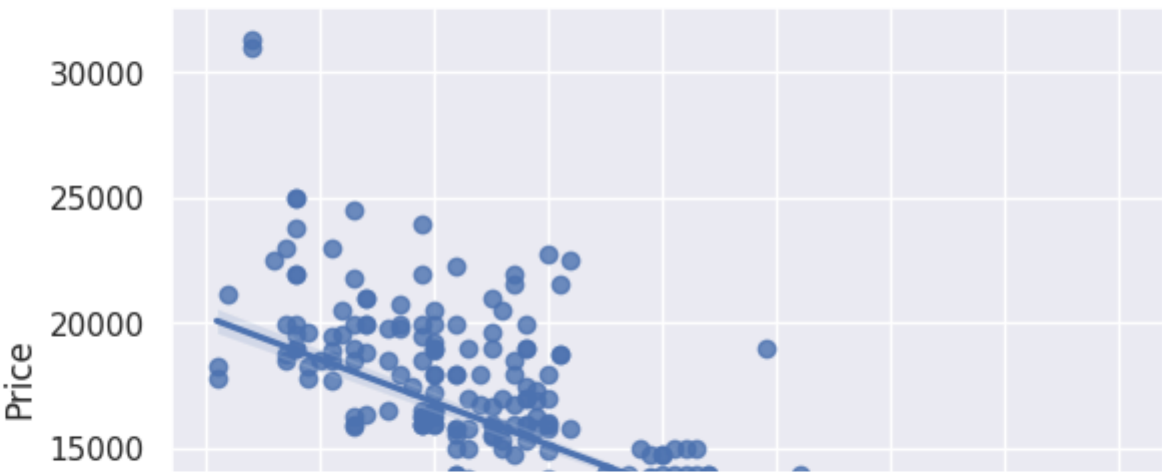


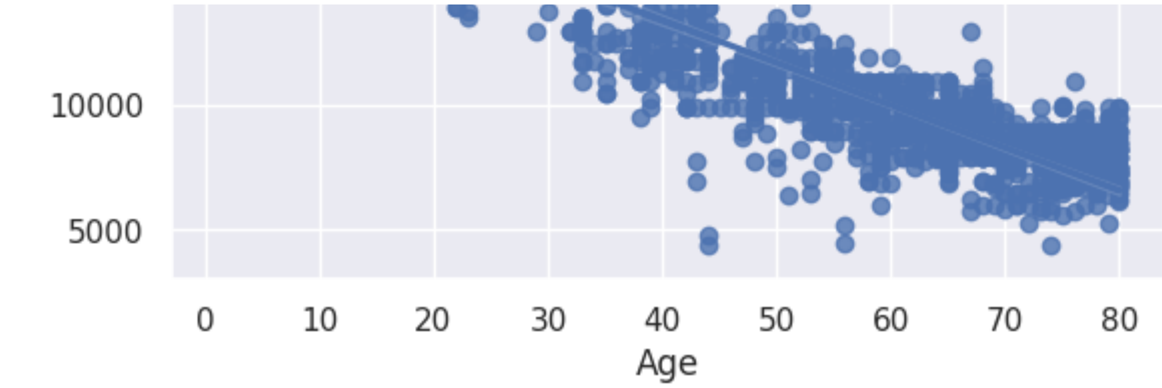
Seaborn

- Seaborn is a Python data visualization library based on matplotlib
- It provides a high-level interface for drawing attractive and informative statistical graphics

```
sns.set(style='darkgrid')
sns.regplot(x=cars_data['Age'],y=cars_data['Price'])
```

```
<Axes: xlabel='Age', ylabel='Price'>
```





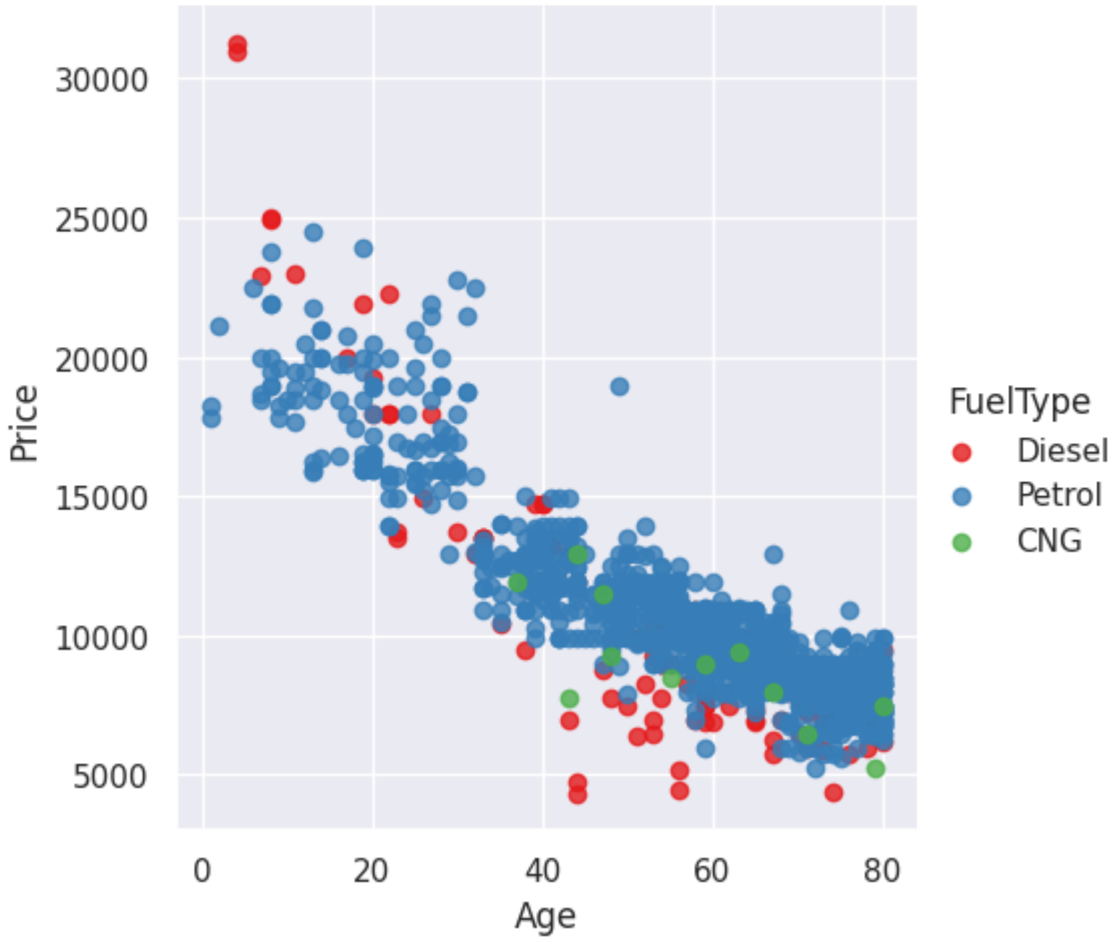
By default, fit_reg = True

o It estimates and plots a regression model relating the x and y variables

Using hue parameter, including another variable to show the fuel types categories with different colors

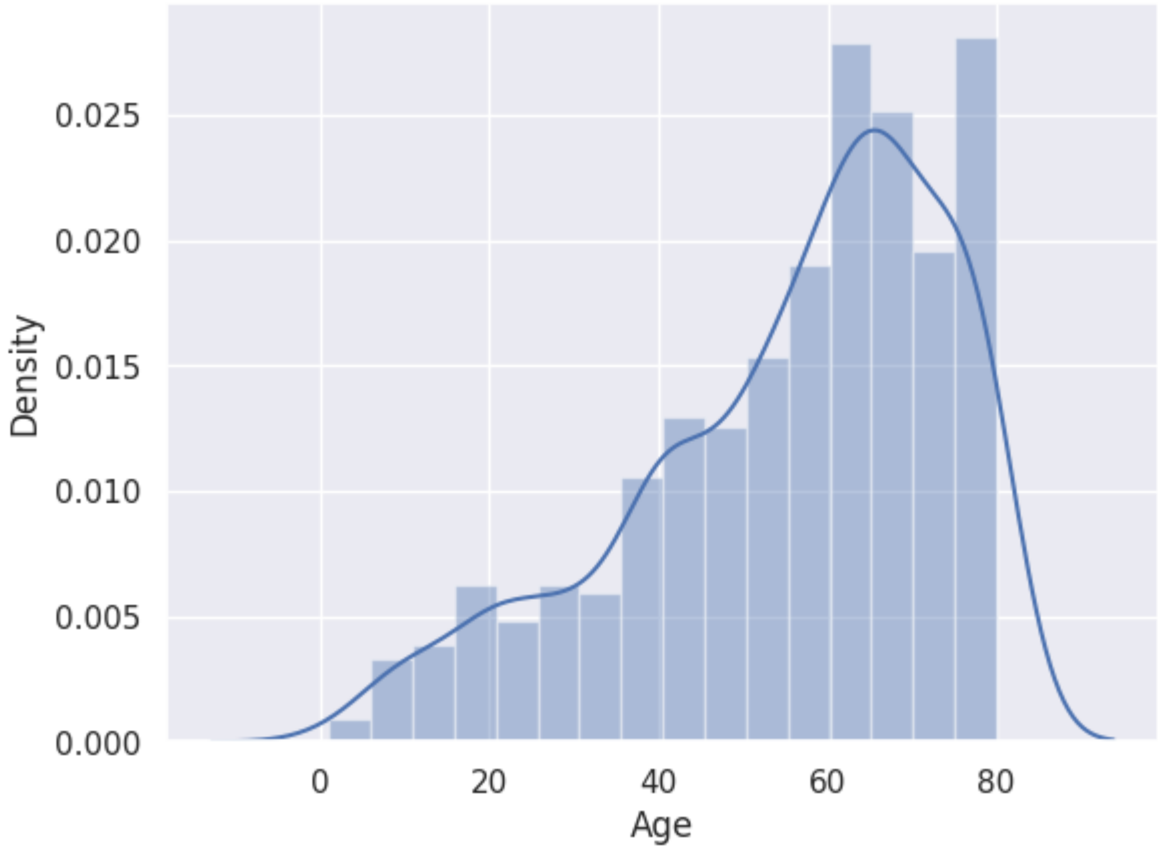
```
sns.lmplot(x='Age',y='Price',data=cars_data,hue='FuelType',legend=True,palette='
```

<seaborn.axisgrid.FacetGrid at 0x7c86d8fef490>



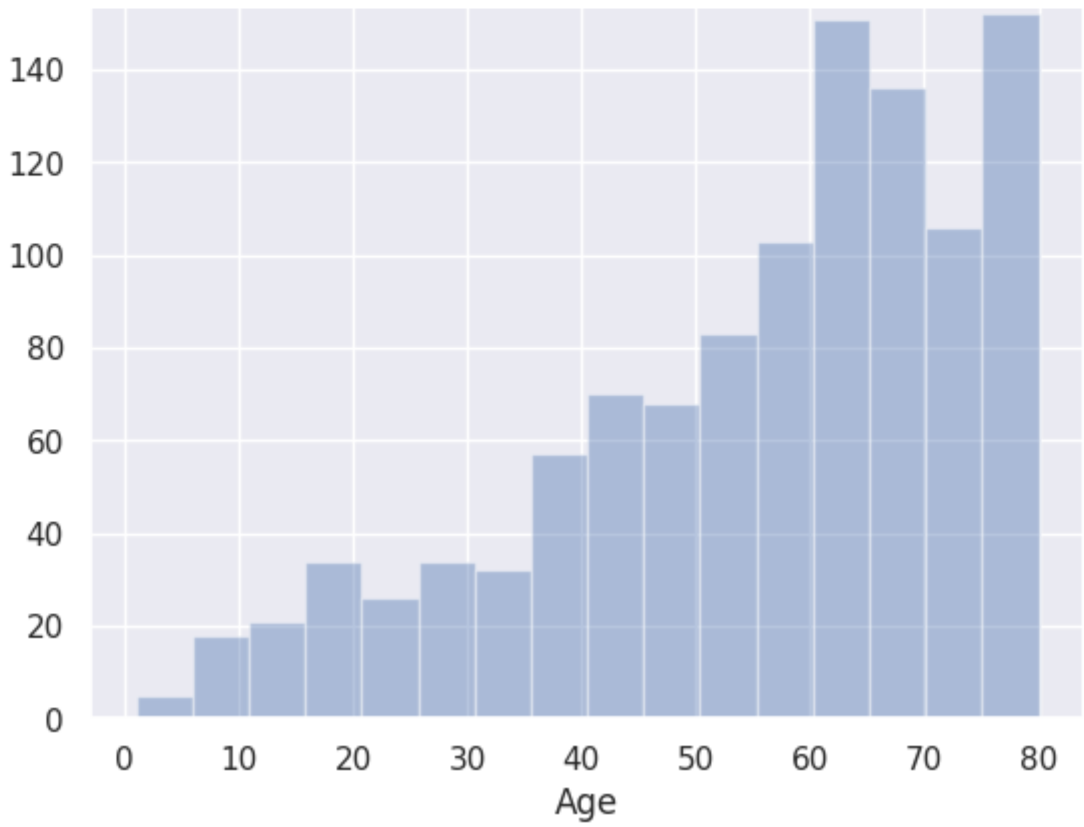
```
sns.distplot(cars_data['Age'])
```

```
<ipython-input-68-67ef1d320a1e>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0

Please adapt your code to use either `displot` (a figure-level function wi
similar flexibility) or `histplot` (an axes-level function for histograms)

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(cars_data['Age'])
<Axes: xlabel='Age', ylabel='Density'>
```



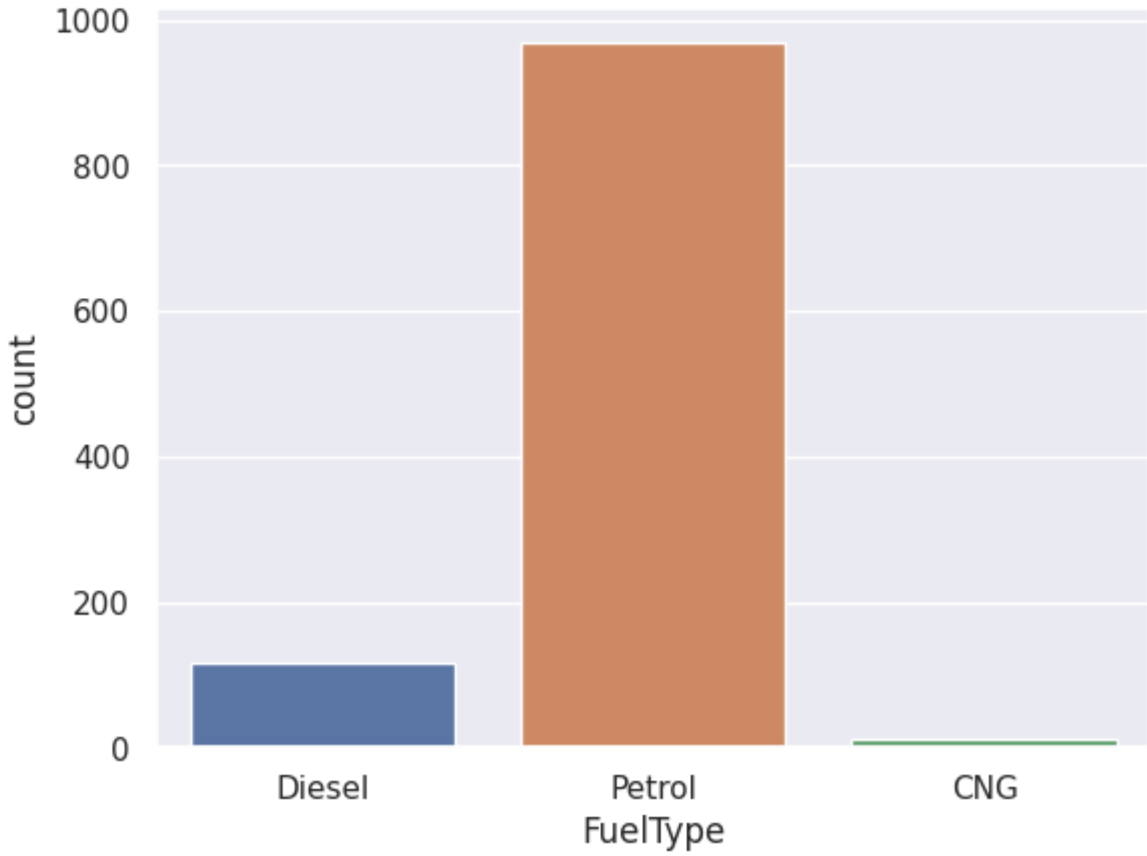

```
sns.distplot(cars_data['Age'],kde=False)
```

```
<ipython-input-69-0f8bc2d269a0>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0

Please adapt your code to use either `displot` (a figure-level function wi
similar flexibility) or `histplot` (an axes-level function for histograms)

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(cars_data['Age'],kde=False)
<Axes: xlabel='Age'>
```




```
sns.countplot(x='FuelType',data=cars_data)
```

<Axes: xlabel='FuelType', ylabel='count'>

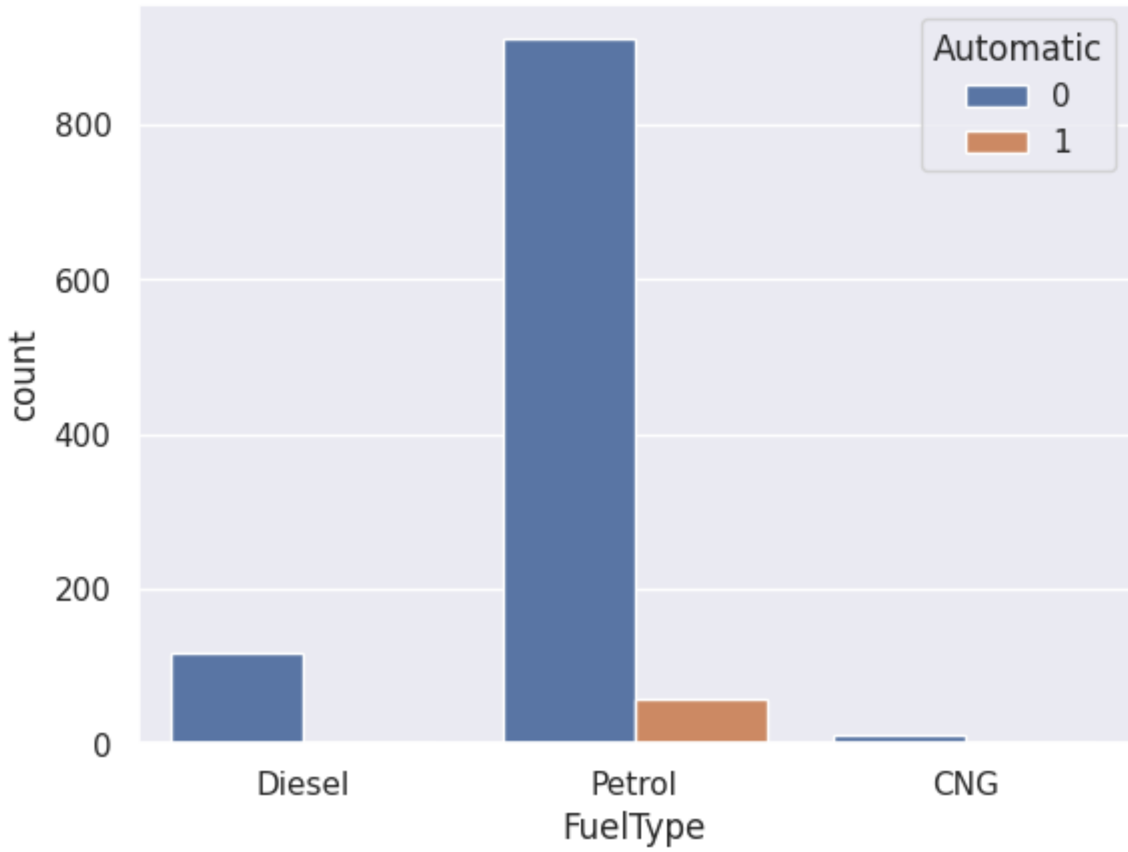



```
pd.crosstab(index=cars_data['Automatic'],columns=cars_data['FuelType'],dropna=Tr
```

FuelType	CNG	Diesel	Petrol
Automatic			
0	12	116	910
1	0	0	58

```
sns.countplot(x='FuelType',data=cars_data,hue='Automatic')
```

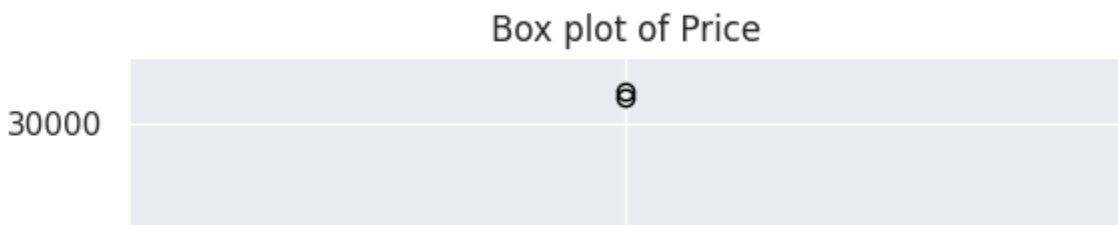
<Axes: xlabel='FuelType', ylabel='count'>

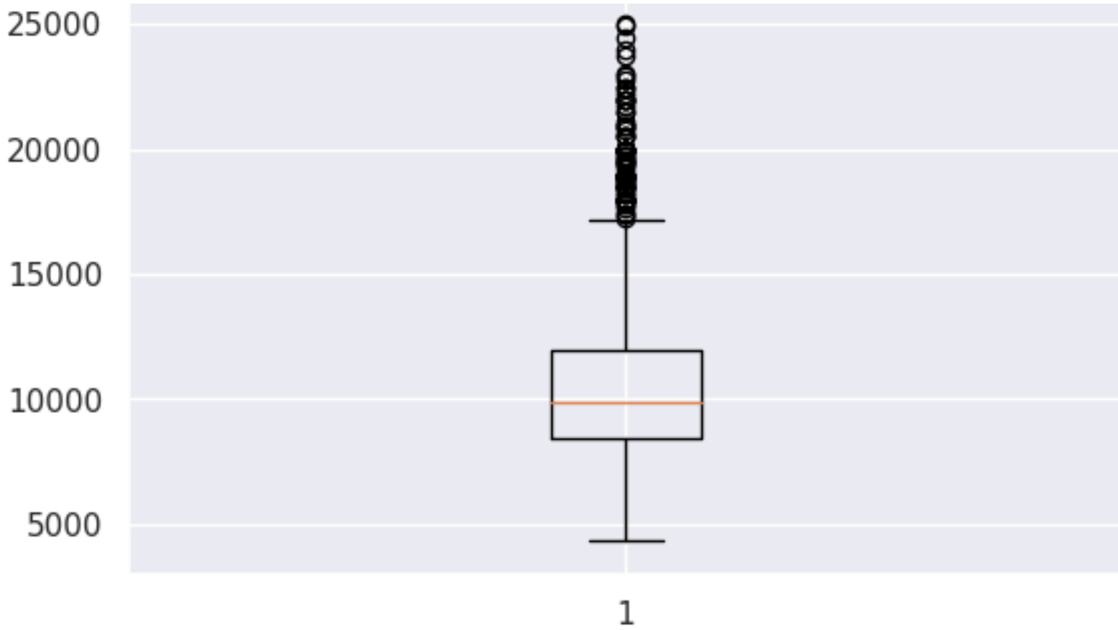


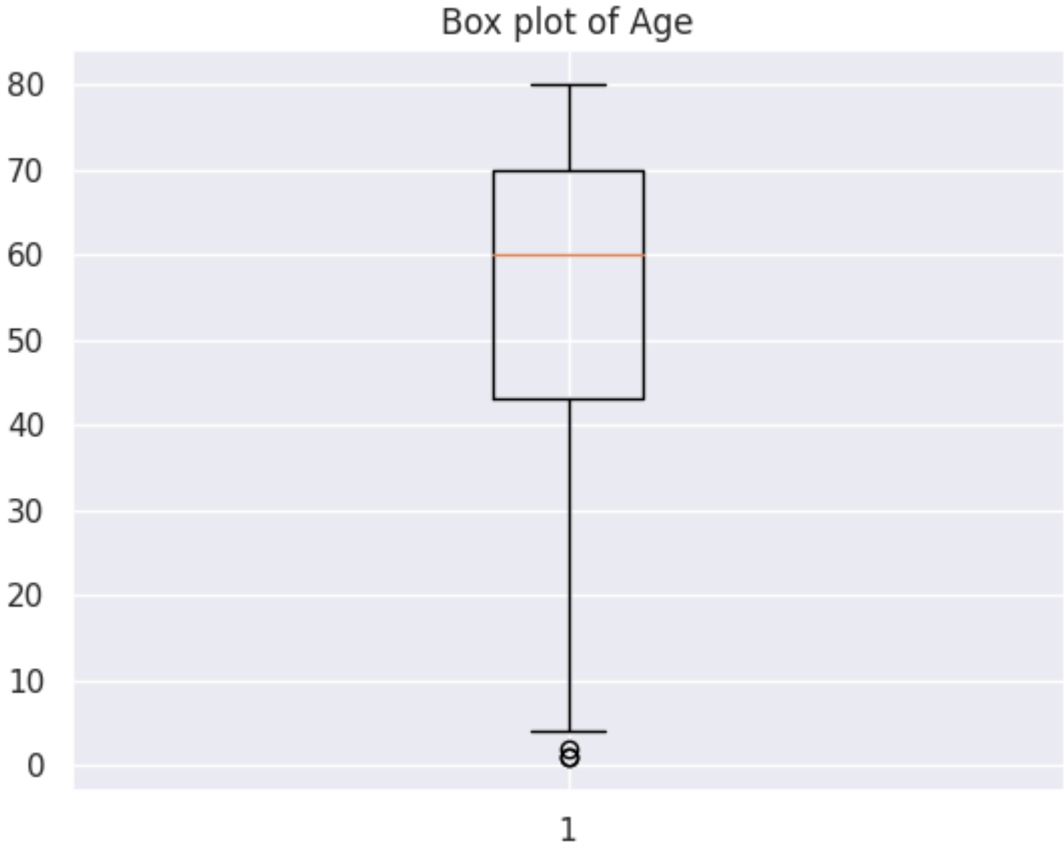
Box and whiskers plot – numerical variable

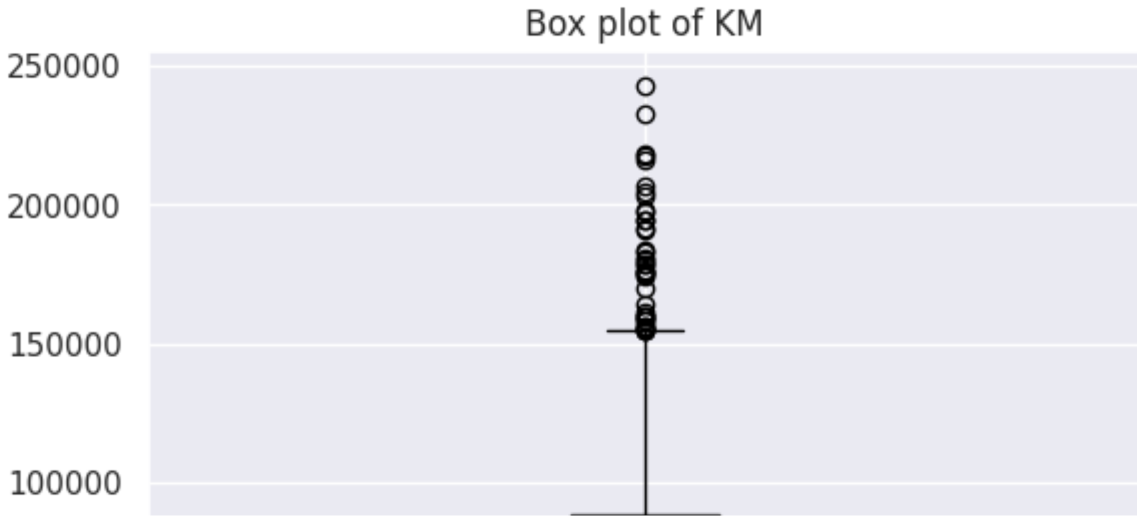
Box and whiskers plot of Price to visually interpret the five-number summary

```
# box plot of all numerical columns
# verify the stastical info
for col in cars_data.select_dtypes(include=['float64', 'int64']).columns:
  plt.boxplot(cars_data[col])
  plt.title(f"Box plot of {col}")
  plt.show()
```

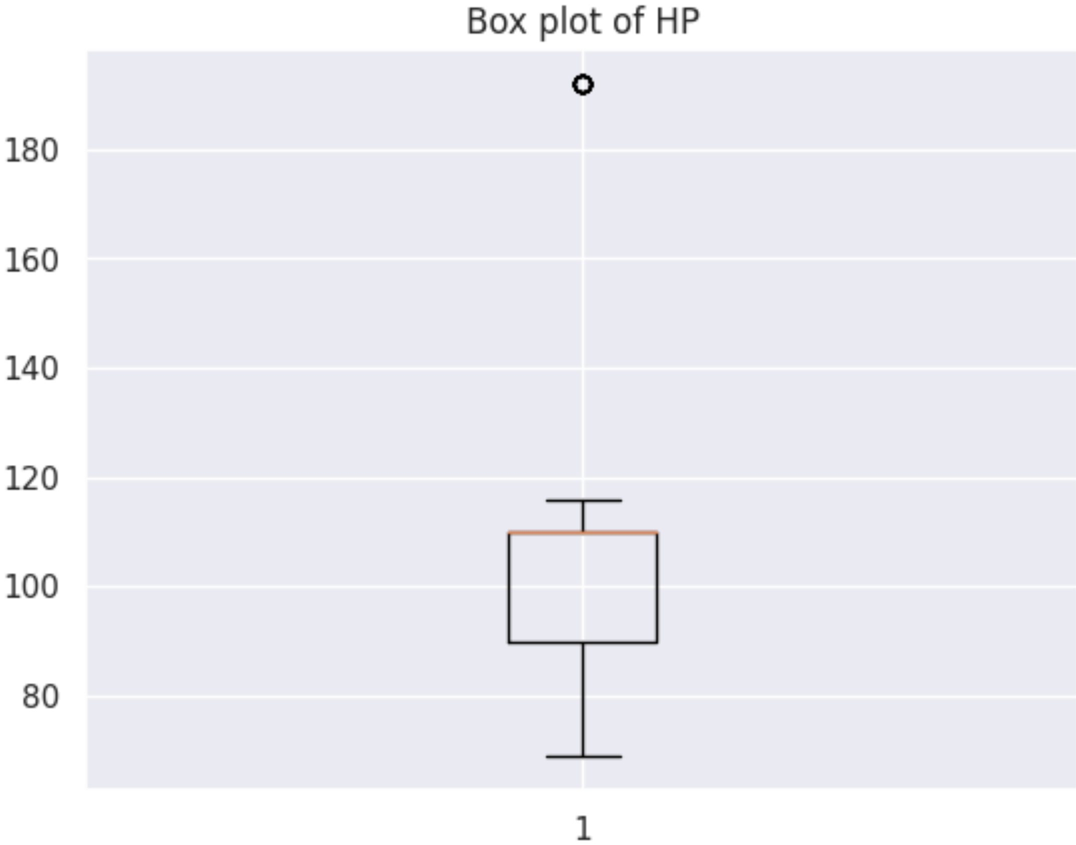


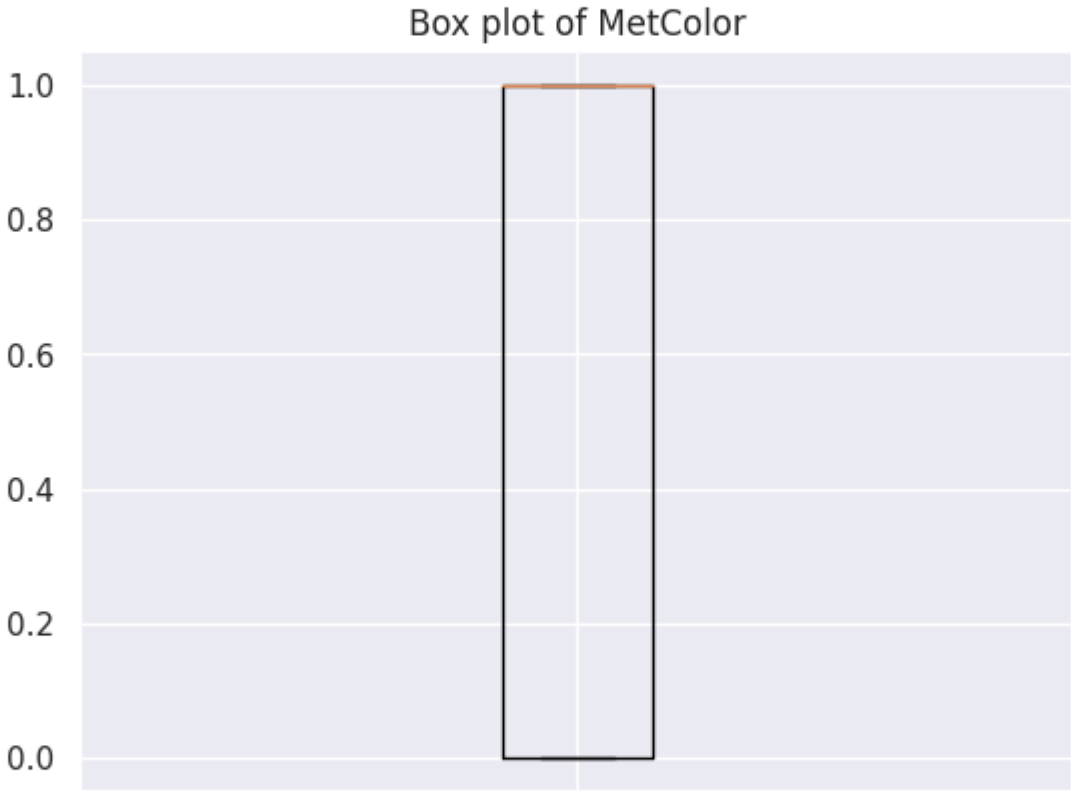

25000
20000
15000
10000
5000
1


Box plot of Age
80
70
60
50
40
30
20
10
0
1

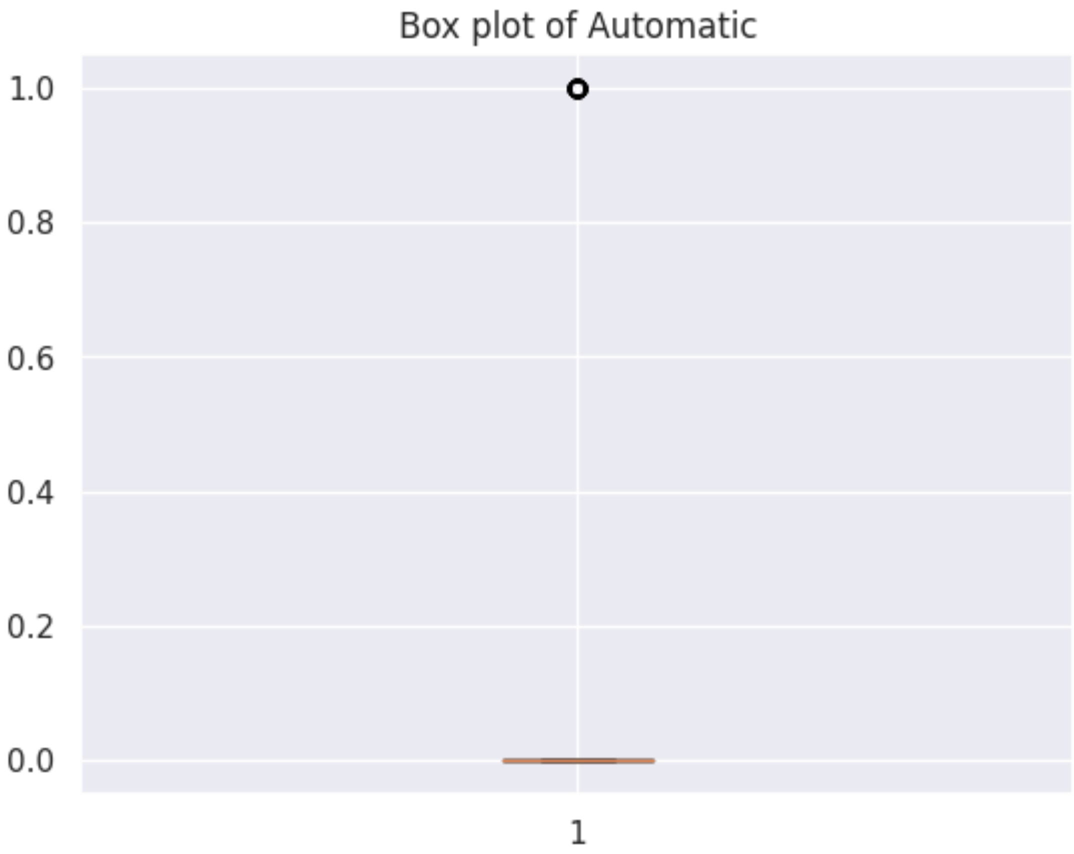

Box plot of KM
250000
200000
150000
100000

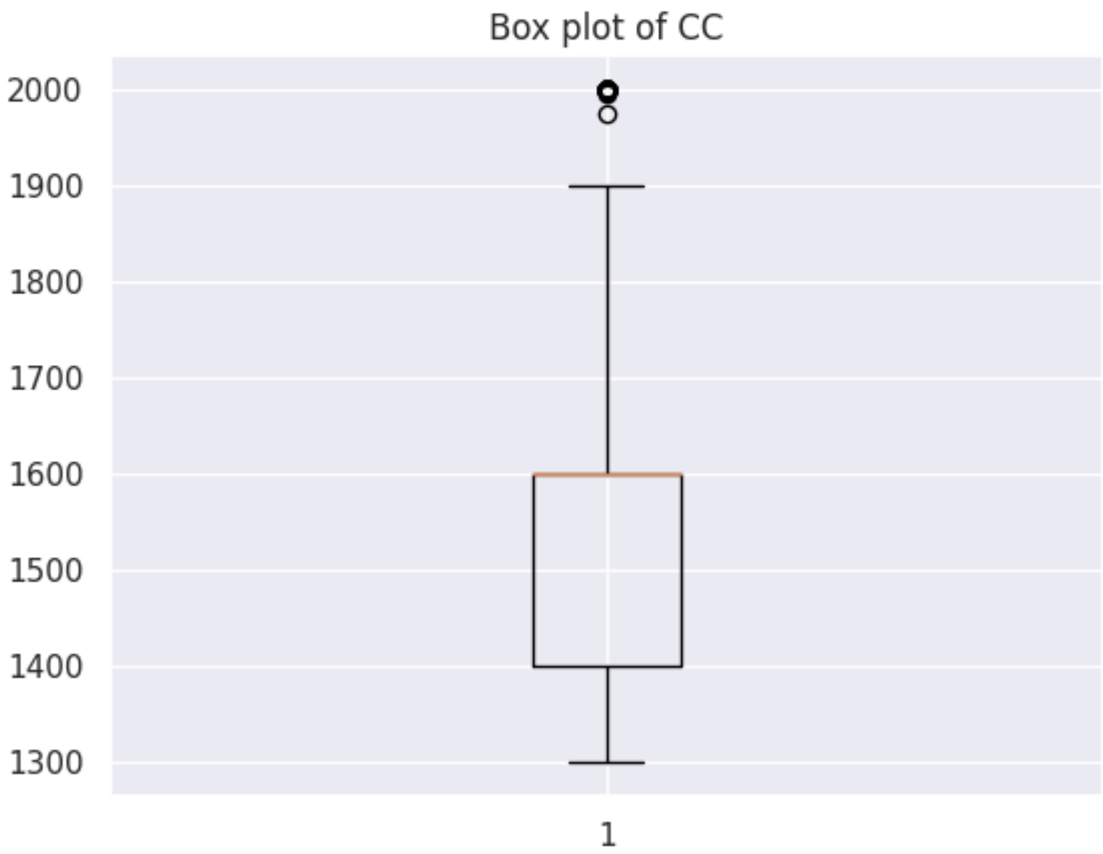

50000
0
1

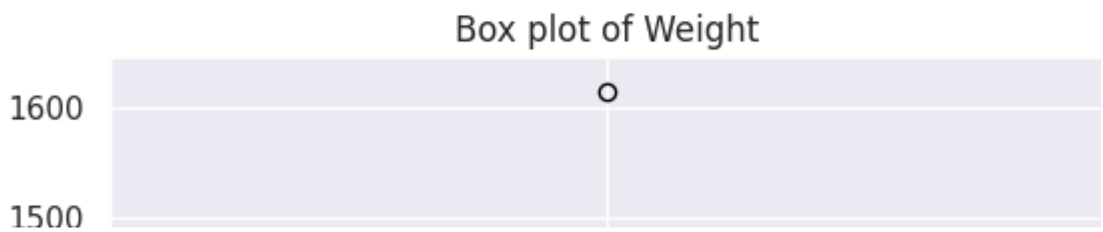

Box plot of HP
180
160
140
120
100
80
1

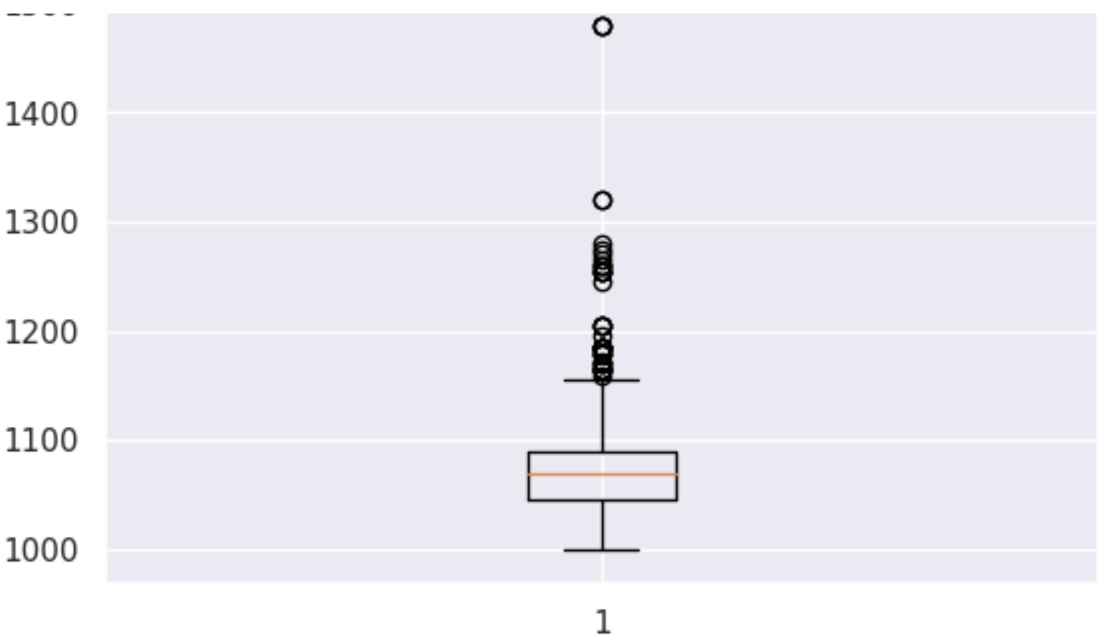

Box plot of MetColor
1.0
0.8
0.6
0.4
0.2
0.0

1


Box plot of Automatic
1.0
0.8
0.6
0.4
0.2
0.0
1


Box plot of CC
2000
1900
1800
1700
1600
1500
1400
1300
1


Box plot of Weight
1600
1500

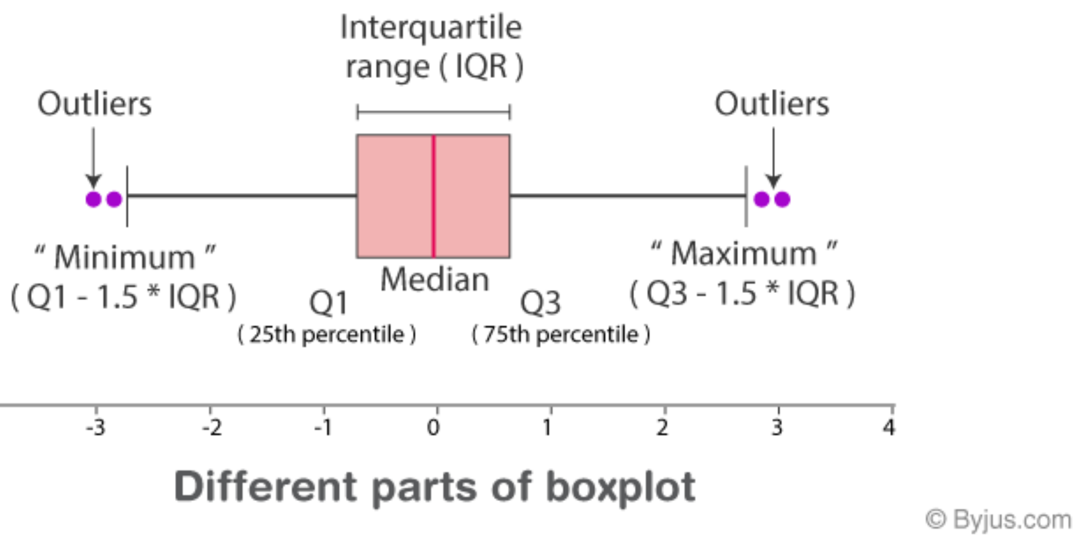


A box plot (aka box and whisker plot) uses boxes and lines to depict the distributions of one or more groups of numeric data.

Box limits indicate the range of the central 50% of the data, with a central line marking the median value.

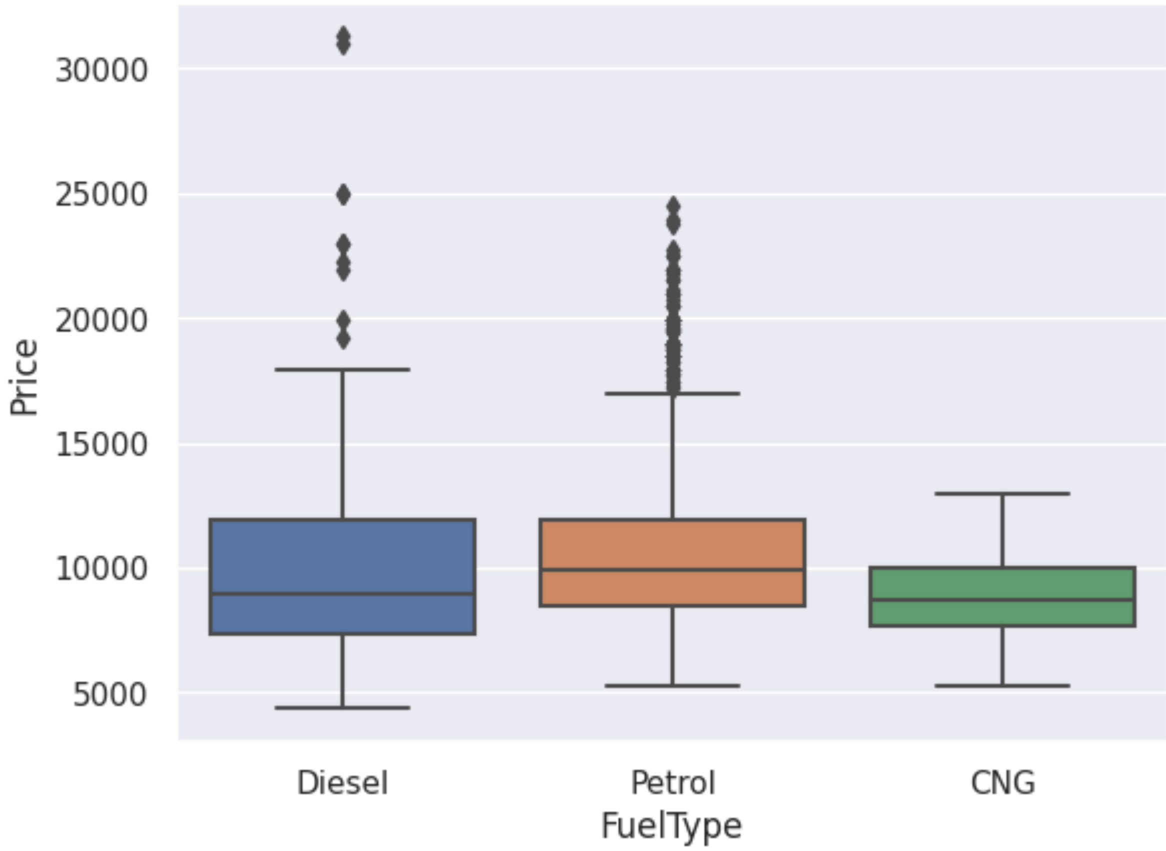
Lines extend from each box to capture the range of the remaining data,

with dots placed past the line edges to indicate outliers.



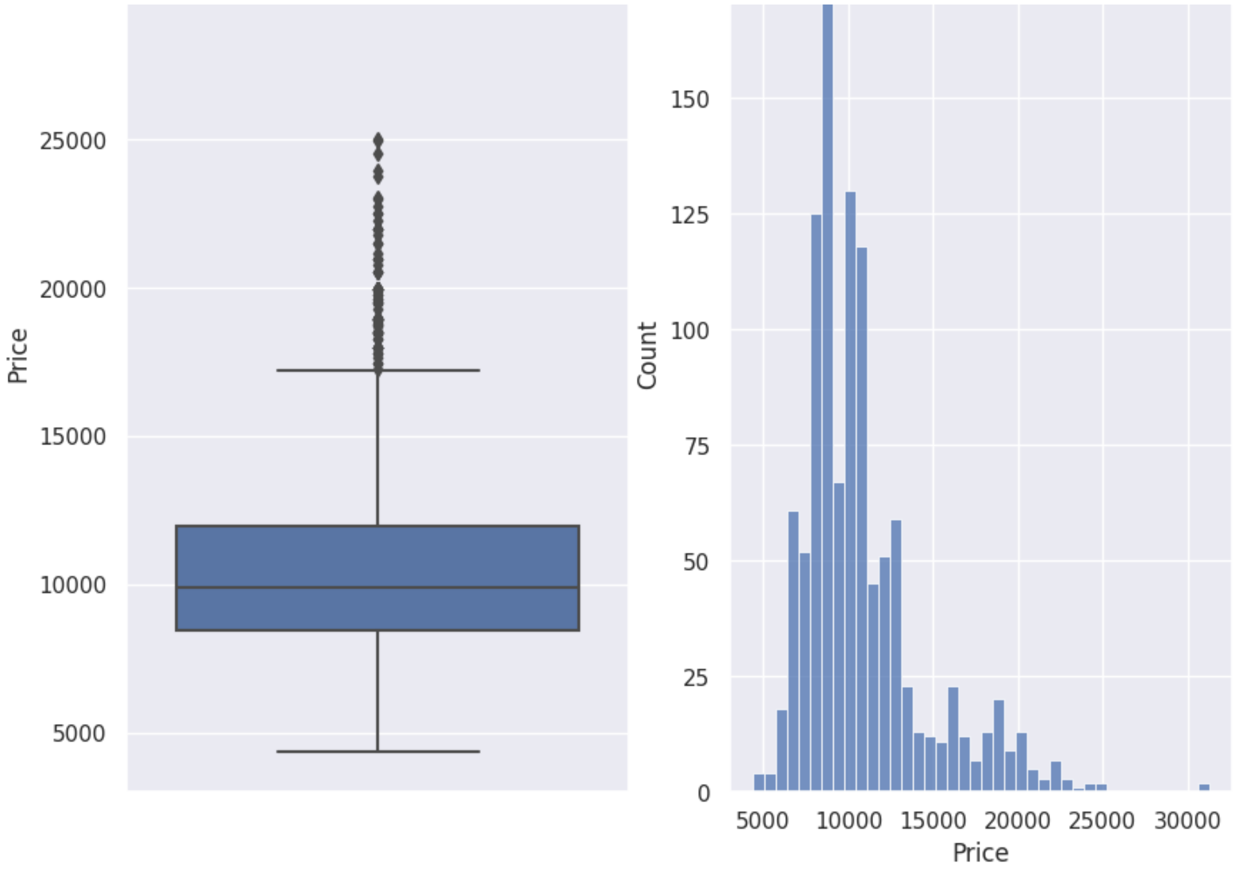
```
# box plot of numerical columns vs categorical column
# verify the stastical info
```

```
# box plot of numerical columns vs categorical column
sns.boxplot(x=cars_data['FuelType'],y=cars_data['Price'])
plt.show()

# verify the stastical info
cars_data.groupby('FuelType')['Price'].describe()
```



	count	mean	std	min	25%	50%	75%
FuelType							
CNG	12.0	8950.833333	2269.262829	5250.0	7677.5	8725.0	9962.5
Diesel	116.0	10550.956897	5233.610064	4350.0	7325.0	8950.0	11950.0
Petrol	968.0	10780.233471	3408.043316	5250.0	8500.0	9940.0	11950.0

```
#create two subplots and plot Box bolt and histogram plot of the same numerical
fig,ax=plt.subplots(1,2,figsize=(10,8))
sns.boxplot(y=cars_data['Price'],ax=ax[0])
sns.histplot(cars_data['Price'],ax=ax[1],kde=False)
plt.show()
```


Price
25000
20000
15000
10000
5000
Count
150
125
100
75
50
25
0
5000
10000
15000
20000
25000
30000
Price