Name: SIMRAN ANAND

Registration Number: 19BCD7243

Slot: L21+L22

Lab 1: Visualizing data using R or Python with different types of graphs and charts. Report consisting the answers to the given questions along with visualizations and Exploratory Data Analysis done as shown below.

Exploratory Data Analysis in Python.

What is Exploratory Data Analysis? What is its importance?

Exploratory Data Analysis or (EDA) is understanding the data sets by summarizing their main characteristics often plotting them visually. This step is very important especially when we arrive at modeling the data in order to apply Machine learning. Plotting in EDA consists of Histograms, Box plot, Scatter plot and many more. It often takes much time to explore the data. Through the process of EDA, we can ask to define the problem statement or definition on our data set which is very important.

What data are we exploring for this lab experiment?

The data-set is downloaded from here. The data set contains more of 10, 000 rows and more than 10 columns which contains features of the car such as Engine Fuel Type, Engine HP, Transmission Type, highway MPG, city MPG and many more.

What are various graphs/charts used for Exploratory Data Analysis? CountPlot, Pie Chart, Histogram, Distplot, Scatter Plot, Multivariate analysis with scatter plot, Bar Plot.

What are the applications of various graphs/charts?

Reading and analyzing bar graphs and pie charts is useful not just in the math classroom but also in real life, both personal and business. Bar graphs and pie charts provide information that is readily understood with a quick glance. Businesses use both bar graphs and pie charts to present information, such as sales information, to customers as well as to employees and other businesses. People can also use bar graphs and pie charts for personal reasons, such as keeping track of finances.

Being able to read and analyze these graphs is a good skill to have because these graphs and charts are seen all the time in the real world. Sometimes, they are even shown in commercials.These graphs and charts can be particularly useful when shown to potential investors before they decide to invest in a company.There are countless other graphs in science, engineering or everyday life: The links between atoms in molecules and crystal grids form a graph. The spread of diseases and epidemics can be modelled using a network. In Biology, the evolutionary trees that show the ancestry of species form a graph.Charts and graphs also make it easier to notice important information within the data.

Difference between histogram and Barplot.

Histogram refers to a graphical representation; that displays data by way of bars to show the frequency of numerical data. A bar graph is a pictorial representation of data that uses bars to compare different categories of data.A histogram represents the frequency distribution of continuous variables. Conversely, a bar graph is a diagrammatic comparison of discrete variables.Histogram presents numerical data whereas bar graph shows categorical data.The width of rectangular blocks in a histogram may or may not be same while the width of the bars in a bar graph is always same.

bars in a bar graph is always same.

What is the need for Boxplot?

A boxplot is a graph that gives you a good indication of how the values in the data are spread out. Although boxplots may seem primitive in comparison to a histogram or density plot, they have the advantage of taking up less space, which is useful when comparing distributions between many groups or datasets.Box plots divide the data into sections that each contain approximately 25% of the data in that set. Box plots are useful as they provide a visual summary of the data enabling researchers to quickly identify mean values, the dispersion of the data set, and signs of skewness.A box plot is constructed from five values: the minimum value, the first quartile, the median, the third quartile, and the maximum value.It is especially useful when you want to see if a distribution is skewed and whether there are potential unusual data values (outliers) in a given dataset. These plots are also widely used for comparing two data sets.

Perform Outlier detection using Boxplot.

Done in my ipynb file notebook. (Section 8)

What is legend? How to create a legend in a graph?

With a chart,a legend is an area of a chart describing each of the parts of the chart.The legend of a graph reflects the data displayed in the graph's Y-axis, also called the graph series.This is the data that comes from the columns of the corresponding grid report,and usually represents metrics.A graph legend generally appears as a box to the right or left of the graph.To create a legend in Excel, start by clicking on the chart or graph. Click on a plus sign on the right.Click on this, and a Chart Elements menu pops up.Check the box next to "Legend". A chart or graph legend will immediately appear.

---

1. Importing the required libraries for EDA

Below are the libraries that are used in order to perform EDA (Exploratory data analysis) for my dataset.

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns                          #visualisation
import matplotlib.pyplot as plt                #visualisation
%matplotlib inline
sns.set(color_codes=True)
```

---

2. Loading the data into the data frame.

In [2]:

```
df = pd.read_csv("/content/data.csv")
# To display the top 5 rows
df.head(5)
```

Out[2]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	C
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	C
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	C

In [3]:

```
df.tail(5)                          # To display the botton 5 rows
```

Out[3]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market C
11909	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchbacl
11910	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchbacl
11911	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchbacl
11912	Acura	ZDX	2013	premium unleaded (recommended)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchbacl
11913	Lincoln	Zephyr	2006	regular unleaded	221.0	6.0	AUTOMATIC	front wheel drive	4.0	

3. Checking the types of data

Here we check for the datatypes because sometimes the MSRP or the price of the car would be stored as a string, if in that case, we have to convert that string to the integer data only then we can plot the data via a graph. Here, in this case, the data is already in integer format so nothing to worry.

In [4]:

```
df.dtypes
```

Out[4]:

```
Make                    object
Model                   object
Year                     int64
Engine Fuel Type        object
Engine HP              float64
Engine Cylinders       float64
Transmission Type       object
Driven_Wheels           object
Number of Doors        float64
```

```
Market Category        object
Vehicle Size           object
Vehicle Style          object
highway MPG             int64
city mpg                int64
Popularity              int64
MSRP                    int64
dtype: object
```

4. Dropping irrelevant columns

This step is certainly needed in every EDA because sometimes there would be many columns that we never use in such cases dropping is the only solution. In this case, the columns such as Engine Fuel Type, Market Category, Vehicle style, Popularity, Number of doors, Vehicle Size doesn't make any sense to me so I just dropped for this instance.

In [5]:

```
df = df.drop(['Engine Fuel Type', 'Market Category', 'Vehicle Style', 'Popularity', 'Num
ber of Doors', 'Vehicle Size'], axis=1)
df.head(5)
```

Out[5]:

	Make	Model	Year	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	highway MPG	city mpg	MSRP
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

5. Renaming the columns

In this instance, most of the column names are very confusing to read, so I just tweaked their column names. This is a good approach it improves the readability of the data set.

In [6]:

```
df = df.rename(columns={"Engine HP": "HP", "Engine Cylinders": "Cylinders", "Transmissio
n Type": "Transmission", "Driven_Wheels": "Drive Mode","highway MPG": "MPG-H", "city mpg
": "MPG-C", "MSRP": "Price" })
df.head(5)
```

Out[6]:

	Make	Model	Year	HP	Cylinders	Transmission	Drive Mode	MPG-H	MPG-C	Price
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

6. Dropping the duplicate rows

In [7]:

```
df.shape
```

Out[7]:

```
(11914, 10)
```

In [8]:

```
duplicate_rows_df = df[df.duplicated()]
print("number of duplicate rows: ", duplicate_rows_df.shape)
```

```
number of duplicate rows:  (989, 10)
```

Now let us remove the duplicate data because it's ok to remove them.

In [9]:

```
df.count()      # Used to count the number of rows
```

Out[9]:

```
Make            11914
Model           11914
Year            11914
HP              11845
Cylinders       11884
Transmission    11914
Drive Mode      11914
MPG-H           11914
MPG-C           11914
Price           11914
dtype: int64
```

So seen above there are 11914 rows and we are removing 989 rows of duplicate data.

In [10]:

```
df = df.drop_duplicates()
df.head(5)
```

Out[10]:

	Make	Model	Year	HP	Cylinders	Transmission	Drive Mode	MPG-H	MPG-C	Price
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

In [11]:

```
df.count()
```

Out[11]:

```
Make            10925
Model           10925
Year            10925
```

```
HP              10856
Cylinders       10895
Transmission    10925
Drive Mode      10925
MPG-H           10925
MPG-C           10925
Price           10925
dtype: int64
```

7. Dropping the missing or null values.

This is mostly similar to the previous step but in here all the missing values are detected and are dropped later.

In [12]:

```
print(df.isnull().sum())
```

```
Make             0
Model            0
Year             0
HP              69
Cylinders       30
Transmission     0
Drive Mode       0
MPG-H            0
MPG-C            0
Price            0
dtype: int64
```

This is the reason in the above step while counting both Cylinders and Horsepower (HP) had 10856 and 10895 over 10925 rows.

In [13]:

```
df = df.dropna()    # Dropping the missing values.
df.count()
```

Out[13]:

```
Make            10827
Model           10827
Year            10827
HP              10827
Cylinders       10827
Transmission    10827
Drive Mode      10827
MPG-H           10827
MPG-C           10827
Price           10827
dtype: int64
```

Now we have removed all the rows which contain the Null or N/A values (Cylinders and Horsepower (HP)).

In [14]:

```
print(df.isnull().sum())   # After dropping the values
```

```
Make            0
Model           0
Year            0
HP              0
Cylinders       0
Transmission    0
Drive Mode      0
```

```
MPG-H           0
MPG-C           0
Price           0
dtype: int64
```

8. Detecting Outliers (Using Boxplot)

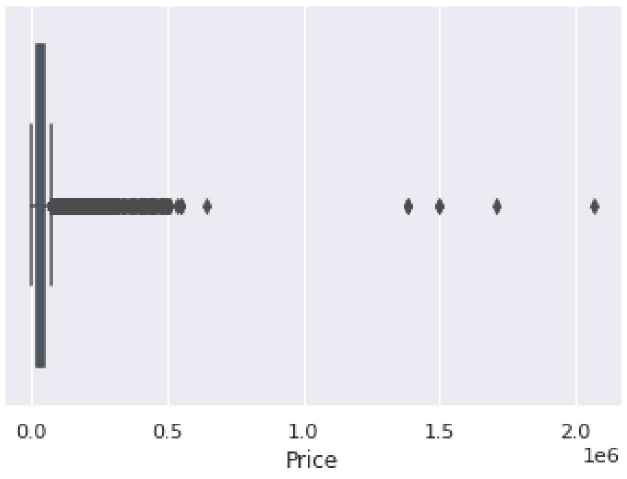
An outlier is a point or set of points that are different from other points. Sometimes they can be very high or very low. It's often a good idea to detect and remove the outliers. Because outliers are one of the primary reasons for resulting in a less accurate model. Hence it's a good idea to remove them. The outlier detection and removing that I am going to perform is called IQR score technique. Often outliers can be seen with visualizations using a box plot. Shown below are the box plot of MSRP, Cylinders, Horsepower and EngineSize. Herein all the plots, you can find some points are outside the box they are none other than outliers.

In [15]:

```
sns.boxplot(x=df['Price'])
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f28eecc74d0>
```

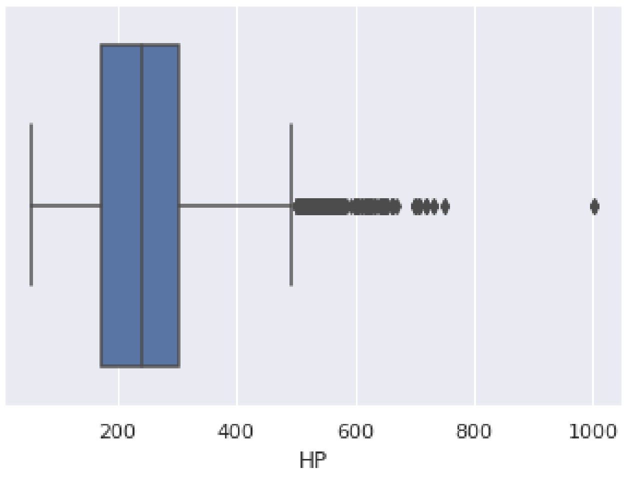


In [16]:

```
sns.boxplot(x=df['HP'])
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f28eebcf7d0>
```

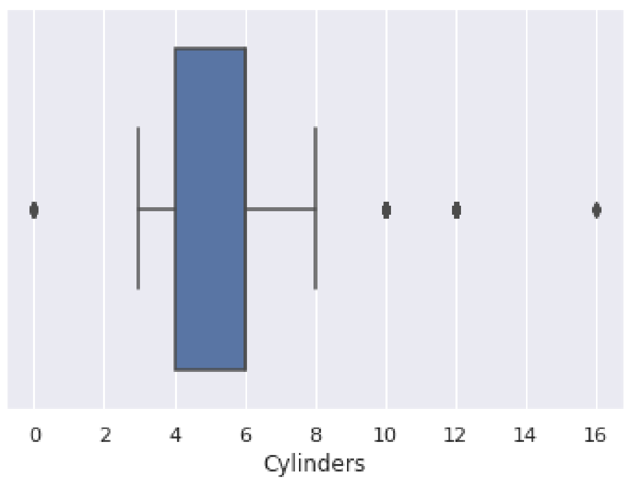


In [17]:

```
sns.boxplot(x=df['Cylinders'])
sns.boxplot(x=df['Cylinders'])
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f28ee6eaf90>
```



In [18]:

```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
Year              9.0
HP              130.0
Cylinders         2.0
MPG-H             8.0
MPG-C             6.0
Price         21327.5
dtype: float64
```

In [19]:

```
df = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
df.shape
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Automatic
reindexing on DataFrame vs Series comparisons is deprecated and will raise ValueError in
a future version.  Do `left, right = left.align(right, axis=1, copy=False)` before e.g. `
left == right`
  """Entry point for launching an IPython kernel.
```

Out[19]:

```
(9191, 10)
```

9. Plot different features against one another (scatter), against frequency (histogram)

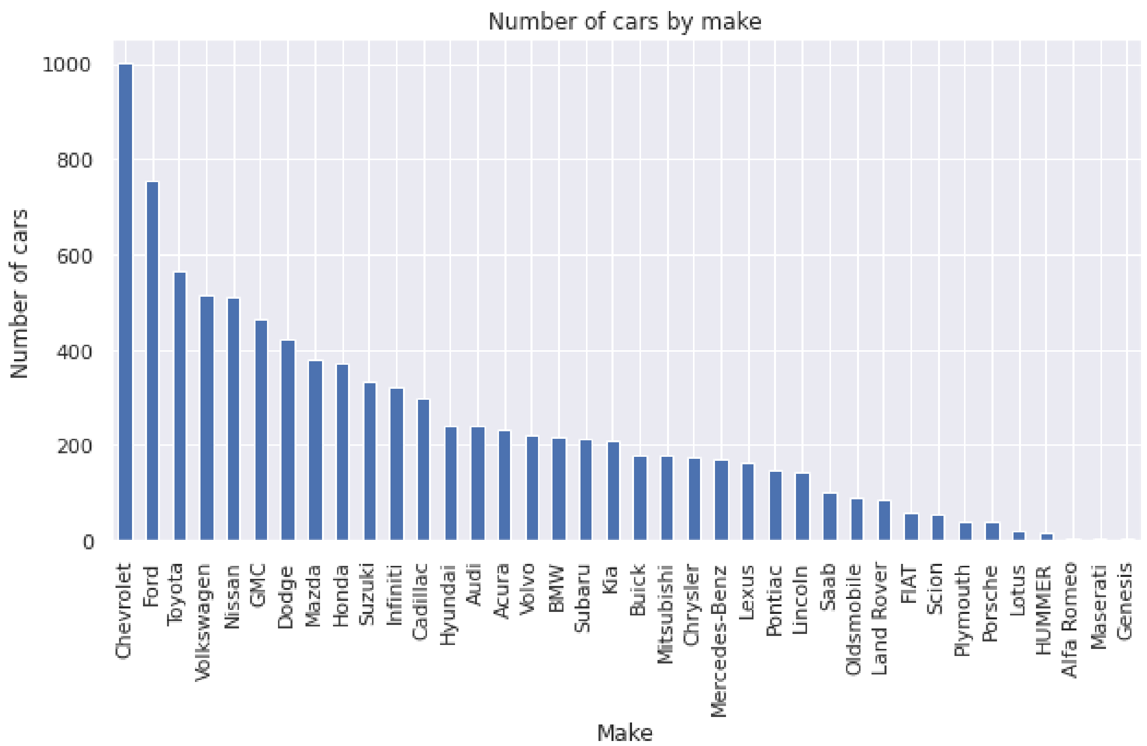
Histogram

Histogram refers to the frequency of occurrence of variables in an interval. In this case, there are mainly 10 different types of car manufacturing companies, but it is often important to know who has the most number of cars. To do this histogram is one of the trivial solutions which lets us know the total number of car manufactured by a different company.

In [20]:

```
df.Make.value_counts().nlargest(40).plot(kind='bar', figsize=(10,5))
```

```
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make');
```



Heat Maps

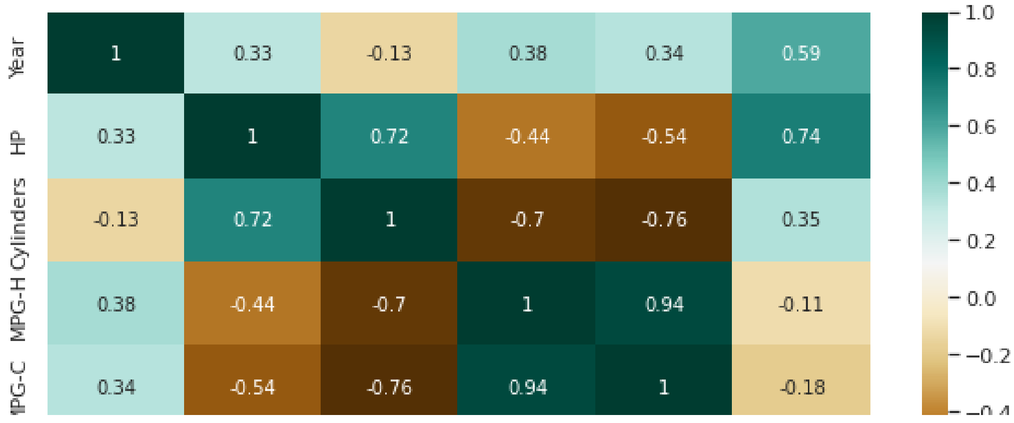
Heat Maps is a type of plot which is necessary when we need to find the dependent variables. One of the best way to find the relationship between the features can be done using heat maps. In the below heat map we know that the price feature depends mainly on the Engine Size, Horsepower, and Cylinders.

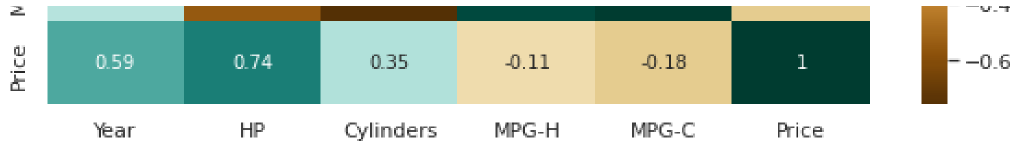
In [21]:

```
plt.figure(figsize=(10,5))
c= df.corr()
sns.heatmap(c,cmap="BrBG",annot=True)
c
```

Out[21]:

	Year	HP	Cylinders	MPG-H	MPG-C	Price
Year	1.000000	0.326726	-0.133920	0.378479	0.338145	0.592983
HP	0.326726	1.000000	0.715237	-0.443807	-0.544551	0.739042
Cylinders	-0.133920	0.715237	1.000000	-0.703856	-0.755540	0.354013
MPG-H	0.378479	-0.443807	-0.703856	1.000000	0.939141	-0.106320
MPG-C	0.338145	-0.544551	-0.755540	0.939141	1.000000	-0.180515
Price	0.592983	0.739042	0.354013	-0.106320	-0.180515	1.000000

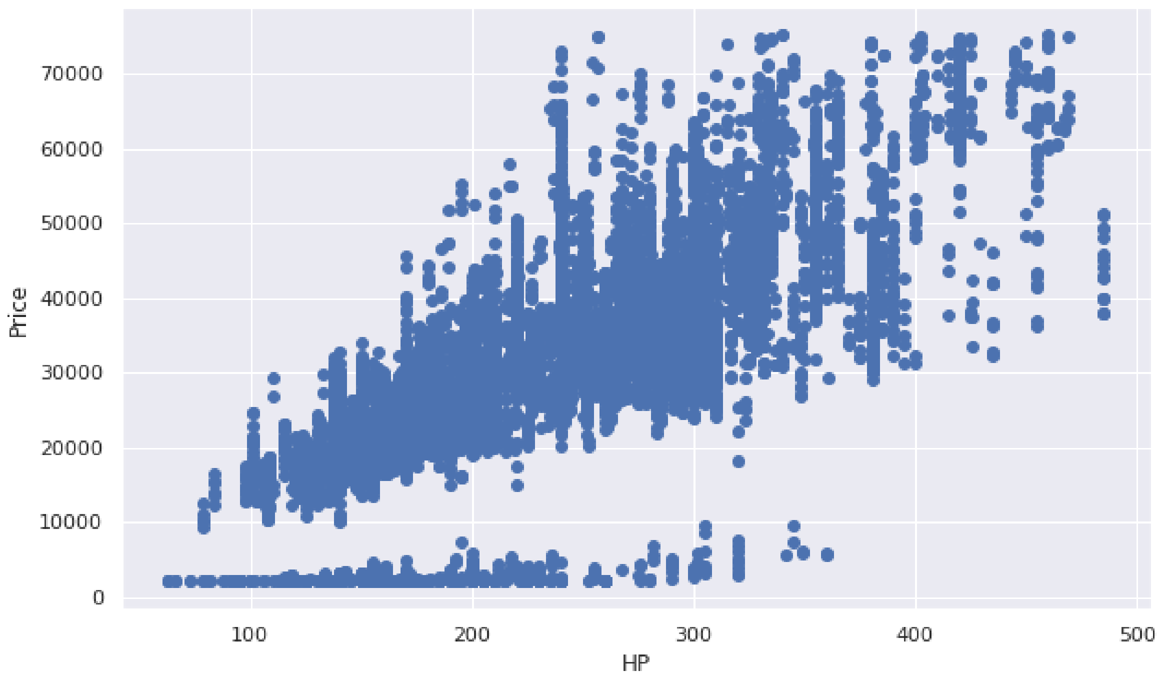




Scatterplot

We generally use scatter plots to find the correlation between two variables. Here the scatter plots are plotted between Horsepower and Price and we can see the plot below. With the plot given below, we can easily draw a trend line. These features provide a good scattering of points.

In [22]:
```
fig, ax = plt.subplots(figsize=(10,6))
ax.scatter(df['HP'], df['Price'])
ax.set_xlabel('HP')
ax.set_ylabel('Price')
plt.show()
```



Thank you.

SIMRAN ANAND

19BCD7243