Exploratory Data Analysis

Problem Statement:

We have used Cars dataset from kaggle with features including make, model, year, engine, and other properties of the car used to predict its price.

TO DOWNLOAD DATASET USED IN VIDEOS : https://drive.google.com/drive/folders/15UNxHTINnphfk43m36ujfw6epMG pDWp?usp=sharing (https://drive.google.com/drive/folders/15UNxHTINnphfk43m36ujfw6epMG pDWp?usp=sharing)

FULL PLAYLIST OF VIDEOS : Use this if videos given below show any kind of error.

https://youtube.com/playlist?list=PLsR_0x6BuM-EBpLRJ8vNpiBuHzu27jtI3 (https://youtube.com/playlist?list=PLsR_0x6BuM-EBpLRJ8vNpiBuHzu27jtI3)

1. Importing the necessary libraries

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)
from scipy import stats
import warnings
warnings.filterwarnings("ignore")
```

2. Download the dataset and load into dataframe

5 points

Please download the dataset from here (https://www.kaggle.com/CooperUnion/cardataset) and extract the csv file. Load the csv file as pandas dataframe.

In [2]:
```
## Load the csv file
df = pd.read_csv('G:\\ClodyML\\Data\\car_data.csv')
```

Now we observe the each features present in the dataset.

`Make:` The Make feature is the company name of the Car.
`Model:` The Model feature is the model or different version of Car models.
`Year:` The year describes the model has been launched.
`Engine Fuel Type:` It defines the Fuel type of the car model.
`Engine HP:` It's say the Horsepower that refers to the power an engine produces.
`Engine Cylinders:` It define the nos of cylinders in present in the engine.
`Transmission Type:` It is the type of feature that describe about the car transmission type i.e Mannual or automatic.
`Driven_Wheels:` The type of wheel drive.
`No of doors:` It defined nos of doors present in the car.
`Market Category:` This features tells about the type of car or which category the car belongs.
`Vehicle Size:` It's say about the about car size.
`Vehicle Style:` The feature is all about the style that belongs to car.
`highway MPG:` The average a car will get while driving on an open stretch of road without stopping or starting, typically at a higher speed.
`city mpg:` City MPG refers to driving with occasional stopping and braking.
`Popularity:` It can refered to rating of that car or popularity of car.
`MSRP:` The price of that car.

In [3]:

```
## print the head of the dataframe
df.head(10)

```

Out[3]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Mar
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Tune
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	
5	BMW	1 Series	2012	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury
6	BMW	1 Series	2012	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury
7	BMW	1 Series	2012	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	
8	BMW	1 Series	2012	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	
9	BMW	1 Series	2013	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	

3. Check the datatypes

2 points

In [24]:
```
# Get the datatypes of each columns number of records in each column.
df.info()

```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10827 entries, 0 to 11913
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Make          10827 non-null  object 
 1   Model         10827 non-null  object 
 2   Year          10827 non-null  int64  
 3   HP            10827 non-null  float64
 4   Cylinders     10827 non-null  float64
 5   Transmission  10827 non-null  object 
 6   Drive Mode    10827 non-null  object 
 7   MPG_H         10827 non-null  int64  
 8   MPG-C         10827 non-null  int64  
 9   Price         10827 non-null  int64  
dtypes: float64(2), int64(4), object(4)
memory usage: 930.4+ KB
```

4. Dropping irrevalent columns

Reference video below

If we consider all columns present in the dataset then unneccessary columns will impact on the model's accuracy.
Not all the columns are important to us in the given dataframe, and hence we would drop the columns that are irrevalent to us. It would reflect our model's accucary so we need to drop them. Otherwise it will affect our model.

The list cols_to_drop contains the names of the cols that are irrevalent, drop all these cols from the dataframe.

```
cols_to_drop = ["Engine Fuel Type", "Market Category", "Vehicle Style", "Popularity", "Number of Doors", "Vehicle Size"]
```

These features are not neccessary to obtain the model's accucary. It does not contain any relevant information in the dataset.

In [6]:
```
# initialise cols_to_drop
cols_to_drop=["Engine Fuel Type", "Market Category", "Vehicle Style","Popula

```

In [7]:

```
# drop the irrevalent cols and print the head of the dataframe
df=df.drop(cols_to_drop,axis=1)

# print df head
df.head(10)

```

Out[7]:

	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	BMW	1 Series	2012	230.0	6.0	MANUAL	rear wheel drive	28	18	31200
6	BMW	1 Series	2012	300.0	6.0	MANUAL	rear wheel drive	26	17	44100
7	BMW	1 Series	2012	300.0	6.0	MANUAL	rear wheel drive	28	20	39300
8	BMW	1 Series	2012	230.0	6.0	MANUAL	rear wheel drive	28	18	36900
9	BMW	1 Series	2013	230.0	6.0	MANUAL	rear wheel drive	27	18	37200

5. Renaming the columns

5 points

Now, Its time for renaming the feature to useful feature name. It will help to use them in model training purpose.

We have already dropped the unneccesary columns, and now we are left with useful columns. One extra thing that we would do is to rename the columns such that the name clearly represents the essence of the column.

The given dict represents (in key value pair) the previous name, and the new name for the dataframe columns

In [8]:
```
# rename cols
rename_cols ={'Engine HP':'HP','Engine Cylinders':'Cylinders','Transmission
                                  'Driven_Wheels':'Drive Mode','highway MPG':

```

In [9]:
```
df=df.rename(columns=rename_cols)
```

In [10]:
```
df.head()

```

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

6. Dropping the duplicate rows

`Reference video below`

There are many rows in the dataframe which are duplicate, and hence they are just repeating the information. Its better if we remove these rows as they don't add any value to the dataframe.

For given data, we would like to see how many rows were duplicates. For this, we will count the number of rows, remove the dublicated rows, and again count the number of rows.

`Documentation Link` : Must go through this - [https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html)](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html)

In [12]:

```
# number of rows before removing duplicated rows

df.count()
```

Out[12]:

```
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
```

In [13]:

```
# drop the duplicated rows
df.drop_duplicates(inplace=True)

# print head of df
df.head()

```

Out[13]:

	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 [14]:

```
# Count Number of rows after deleting duplicated rows

df.count()
```

Out[14]:

```
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 null or missing values

10 points

Missing values are usually represented in the form of Nan or null or None in the dataset.

Finding whether we have null values in the data is by using the isnull() function.

There are many values which are missing, in pandas dataframe these values are reffered to as np.nan. We want to deal with these values beause we can't use nan values to train models. Either we can remove them to apply some strategy to replace them with other values.

To keep things simple we will be dropping nan values

`Documentation Link` : Must go through this - [https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html)](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html)

In [15]:

```
# check for nan values in each columns
df.isnull().sum()

```

Out[15]:

```
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
```

As we can see that the HP and Cylinders have null values of 69 and 30. As these null values will impact on models' accuracy. So to avoid the impact we will drop the these values. As these values are small camparing with dataset that will not impact any major affect on model accuracy so we will drop the values.

In [16]:

```
# drop missing values
df.dropna(inplace=True)

```

In [17]:

```
# Make sure that missing values are removed
# check number of nan values in each col again

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

Out[17]:

```
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
```

In [18]:

```
#Describe statistics of df
df.describe()

```

Out[18]:

	Year	HP	Cylinders	MPG_H	MPG-C	Price
count	10827.000000	10827.000000	10827.000000	10827.000000	10827.000000	1.082700e+04
mean	2010.896370	254.553062	5.691604	26.308119	19.327607	4.249325e+04
std	7.029534	109.841537	1.768551	7.504652	6.643567	6.229451e+04
min	1990.000000	55.000000	0.000000	12.000000	7.000000	2.000000e+03
25%	2007.000000	173.000000	4.000000	22.000000	16.000000	2.197250e+04
50%	2015.000000	240.000000	6.000000	25.000000	18.000000	3.084500e+04
75%	2016.000000	303.000000	6.000000	30.000000	22.000000	4.330000e+04
max	2017.000000	1001.000000	16.000000	354.000000	137.000000	2.065902e+06

8. Removing outliers

`Reference video below`

Sometimes a dataset can contain extreme values that are outside the range of what is expected and unlike the other data. These are called outliers and often machine learning modeling and model skill in general can be improved by understanding and even removing these outlier values.

Detecting outliers

There many techiniques to detect outliers. Let us first see the simplest form of visualizing outliers.

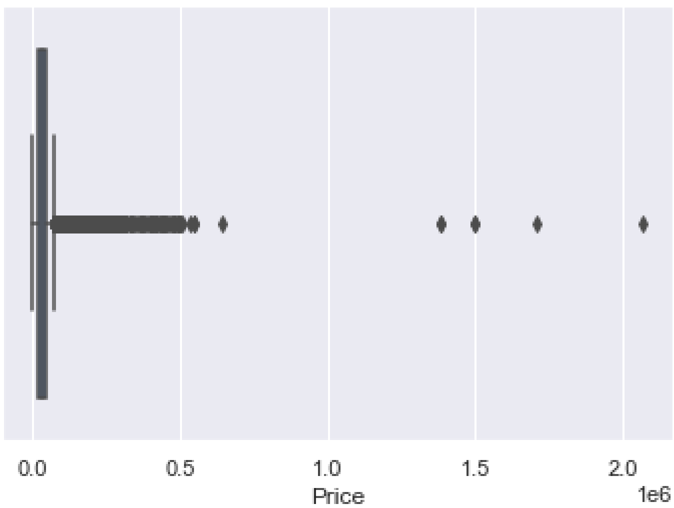
Box plots are a graphical depiction of numerical data through their quantiles. It is a very simple but effective way to visualize outliers. Think about the lower and upper whiskers as the boundaries of the data distribution. Any data points that show above or below the whiskers, can be considered outliers or anomalous.

`Documentation Link` : Must go through this - [https://seaborn.pydata.org/generated/seaborn.boxplot.html (https://seaborn.pydata.org/generated/seaborn.boxplot.html)](https://seaborn.pydata.org/generated/seaborn.boxplot.html)

15 points

In [20]:

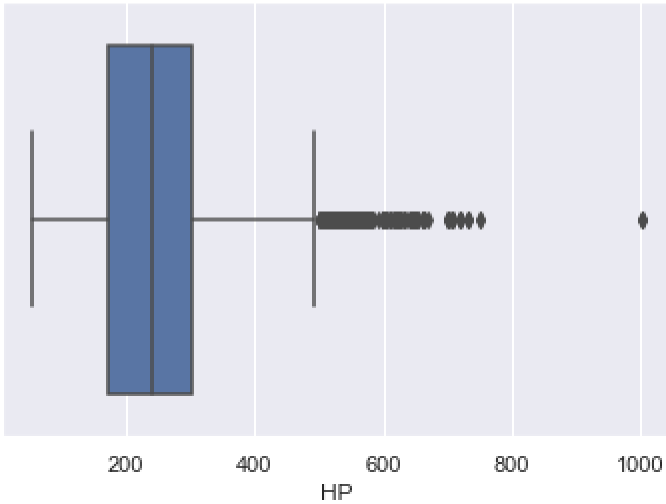
```
## Plot a boxplot for 'Price' column in dataset.
sns.boxplot(x=df["Price"])
plt.show()
```



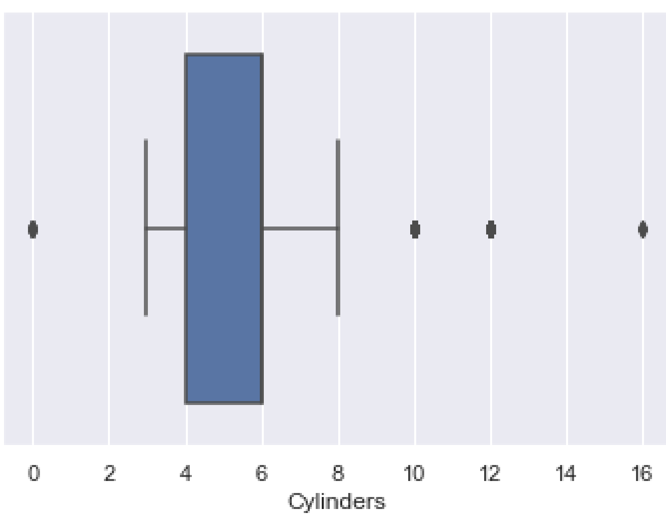
Observation:

Here as you see that we got some values near to 1.5 and 2.0 . So these values are called outliers. Because there are away from the normal values. Now we have detect the outliers of the feature of Price. Similarly we will checking of anothers features.

In [21]:
```
## PLot a boxplot for 'HP' columns in dataset
sns.boxplot(x=df['HP'])
plt.show()
```



In [22]:
```
sns.boxplot(x=df['Cylinders'])
plt.show()
```



Observation:

Here boxplots show the proper distribution of of 25 percentile and 75 percentile of the feature of HP.

print all the columns which are of int or float datatype in df.

Hint: Use loc with condition

In [25]:
```
# print all the columns which are of int or float datatype in df.

df.loc[:,df.dtypes!=object]
```

Out[25]:

	Year	HP	Cylinders	MPG_H	MPG-C	Price
0	2011	335.0	6.0	26	19	46135
1	2011	300.0	6.0	28	19	40650
2	2011	300.0	6.0	28	20	36350
3	2011	230.0	6.0	28	18	29450
4	2011	230.0	6.0	28	18	34500
...	...	...	...	...	...	...
11909	2012	300.0	6.0	23	16	46120
11910	2012	300.0	6.0	23	16	56670
11911	2012	300.0	6.0	23	16	50620
11912	2013	300.0	6.0	23	16	50920
11913	2006	221.0	6.0	26	17	28995

10827 rows × 6 columns

Save the column names of the above output in variable list named 'l'

In [27]:
```
l=list(df.loc[:,df.dtypes != object].columns)
```

Outliers removal techniques

1. **Using IQR Technique**

Here comes cool Fact for you!

IQR is the first quartile subtracted from the third quartile; these quartiles can be clearly seen on a box plot on the data.

The anatomy of boxplot is given below.

image.png

- Calculate IQR and give a suitable threshold to remove the outliers and save this new dataframe into df2.

Let us help you to decide threshold: Outliers in this case are defined as the observations that are below (Q1 − 1.5x IQR) or above (Q3 + 1.5x IQR)

In [28]:

```
## define Q1 and Q2
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)

# define IQR (interquantile range)
IQR = Q3-Q1


# define df2 after removing outliers
df2 = df[~((df < (Q1-1.5*IQR)) | (df > (Q3+1.5*IQR))).any(axis=1)]

```

In [29]:

```
print(df.shape)
print(df2.shape)
```

```
(10827, 10)
(9191, 10)
```

2. **Outlier removal using Z-score function**

    - The intuition behind Z-score is to describe any data point by finding their relationship with the Standard Deviation and Mean of the group of data points.

We will use Z-score function defined in scipy library to detect the outliers in dataframe df having columns which are in variable 'l'

In [34]:

```
# use stats.zscore on list l from above code and take abs value
z = np.abs(stats.zscore(df[l]))

print(z)

```

```
[[0.01474274 0.73242469 0.17438565 0.04105891 0.04931418 0.05846284]
 [0.01474274 0.41376913 0.17438565 0.22545477 0.04931418 0.02959072]
 [0.01474274 0.41376913 0.17438565 0.22545477 0.10121432 0.09862087]
 ...
 [0.15700625 0.41376913 0.17438565 0.44082944 0.50089968 0.13046289]
 [0.29926976 0.41376913 0.17438565 0.44082944 0.50089968 0.13527894]
 [0.69657482 0.30548199 0.17438565 0.04105891 0.35037118 0.21669452]]
```

Hey buddy! do you understand the above output? Difficult right? let’s try and define a threshold to identify an outlier so that we get a clear picture of whats going on.

We will not spare you without a good fact! ;)

In most of the cases a threshold of 3 or -3 is used i.e if the Z-score value is greater than or less than 3 or -3 respectively, that data point will be identified as outliers.

In [35]:
```
# print the values in dataframe which are less than the threshold and save t
threshold = 3
df3 = df[(z < threshold).all(axis=1)]

df3

```

Out[35]:

	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
...	...	...	...	...	...	...	...	...	...	...
11909	Acura	ZDX	2012	300.0	6.0	AUTOMATIC	all wheel drive	23	16	46120
11910	Acura	ZDX	2012	300.0	6.0	AUTOMATIC	all wheel drive	23	16	56670
11911	Acura	ZDX	2012	300.0	6.0	AUTOMATIC	all wheel drive	23	16	50620
11912	Acura	ZDX	2013	300.0	6.0	AUTOMATIC	all wheel drive	23	16	50920
11913	Lincoln	Zephyr	2006	221.0	6.0	AUTOMATIC	front wheel drive	26	17	28995

10338 rows × 10 columns

print the shape difference of df df2 and df3.

In [36]:
```
# print the shape difference of df df2 and df3.
print(df.shape)
print(df2.shape)
print(df3.shape)
```

```
(10827, 10)
(9191, 10)
(10338, 10)
```

Interesting right? Bam! you have removed 489 rows from the dataframe which was detected as outlier by Z-score technique. and removed 1636 rows from the dataframe which was detected as outlier by IQR technique.

By the way there are many other techniques by which you can remove outliers. You can explore on more interesting techniques available.

We know you must be having many questions in you mind like:

- Which technique we should use and why?
- Is it neccessary that whatever detected as outlier are really outliers?

Dont't worry these delimma is faced my many data analyst. We provide you with good references below for you to explore further on this

- [https://www.theanalysisfactor.com/outliers-to-drop-or-not-to-drop/ (https://www.theanalysisfactor.com/outliers-to-drop-or-not-to-drop/)](https://www.theanalysisfactor.com/outliers-to-drop-or-not-to-drop/)
- [https://www.researchgate.net/post/Which-is-the-best-method-for-removing-outliers-in-a-data-set (https://www.researchgate.net/post/Which-is-the-best-method-for-removing-outliers-in-a-data-set)](https://www.researchgate.net/post/Which-is-the-best-method-for-removing-outliers-in-a-data-set)

Lets find unique values and there counts in each column in df using value counts function.

In [ ]:

```
YouTubeVideo('4x-l7t7QNzI',width=700, height=400)
```

In [37]:

```
# find unique values and there counts in each column in df using value count
for i in df.columns:
    print ("--------------- %s ----------------" % i)
    print(df[i].value_counts())
```

```
--------------- Make ----------------
Chevrolet        1043
Ford              798
Toyota            651
Volkswagen        563
Nissan            540
Dodge             513
GMC               475
Honda             429
Cadillac          396
Mazda             392
Mercedes-Benz     340
Suzuki            338
Infiniti          326
BMW               324
Audi              320
Hyundai           254
Acura             246
Volvo             241
```

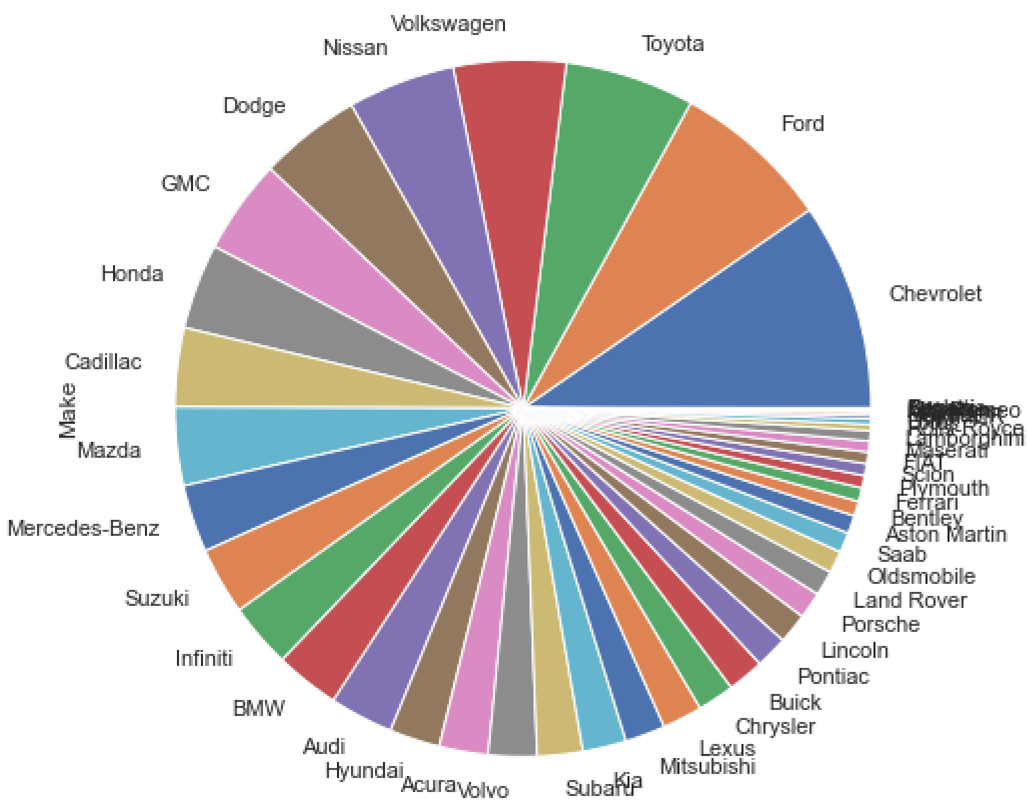
Visualising Univariate Distributions

In [ ]:
```
YouTubeVideo('TxycTY1tAek',width=700, height=400)
```

We will use seaborn library to visualize eye catchy univariate plots.

Do you know? you have just now already explored one univariate plot. guess which one? Yeah its box plot.

In [38]:
```
plt.figure(figsize=(8,8))
vc=df['Make'].value_counts()
vc.plot(kind='pie')
plt.show()
```



1 . Histogram & Density Plots

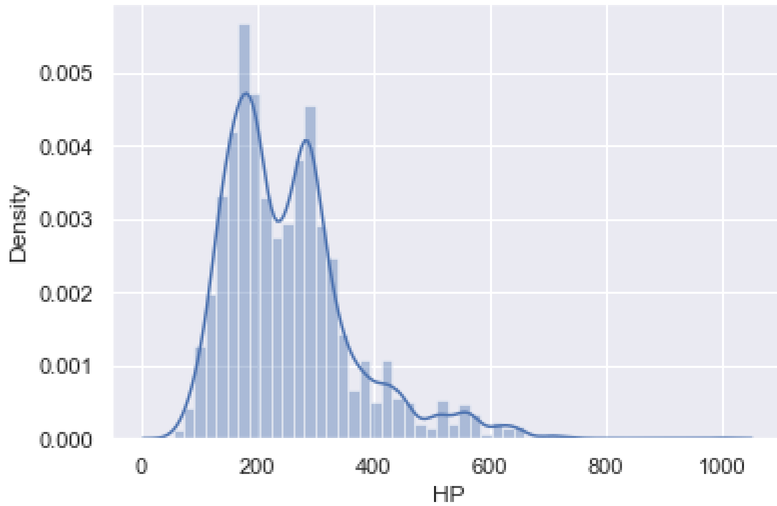
15 points

Histograms and density plots show the frequency of a numeric variable along the y-axis, and the value along the x-axis. The `sns.distplot()` function plots a density curve. Notice that this is aesthetically better than vanilla `matplotlib` .

`Documentation Link` : Must go through this - [https://seaborn.pydata.org/generated/seaborn.displot.html (https://seaborn.pydata.org/generated/seaborn.displot.html)](https://seaborn.pydata.org/generated/seaborn.displot.html)

In [39]:
```
#ploting distplot for variable HP

sns.distplot(df['HP'])
plt.show()
```



Observation:

We plot the Histogram of feature HP with help of distplot in seaborn.
In this graph we can see that there is max values near at 200. similary we have also the 2nd highest value near 400 and so on.
It represents the overall distribution of continuous data variables.

Since seaborn uses matplotlib behind the scenes, the usual matplotlib functions work well with seaborn. For example, you can use subplots to plot multiple univariate distributions.

- Hint: use matplotlib subplot function

In [ ]:
```
YouTubeVideo('sWj1t7By178',width=700, height=400)
```

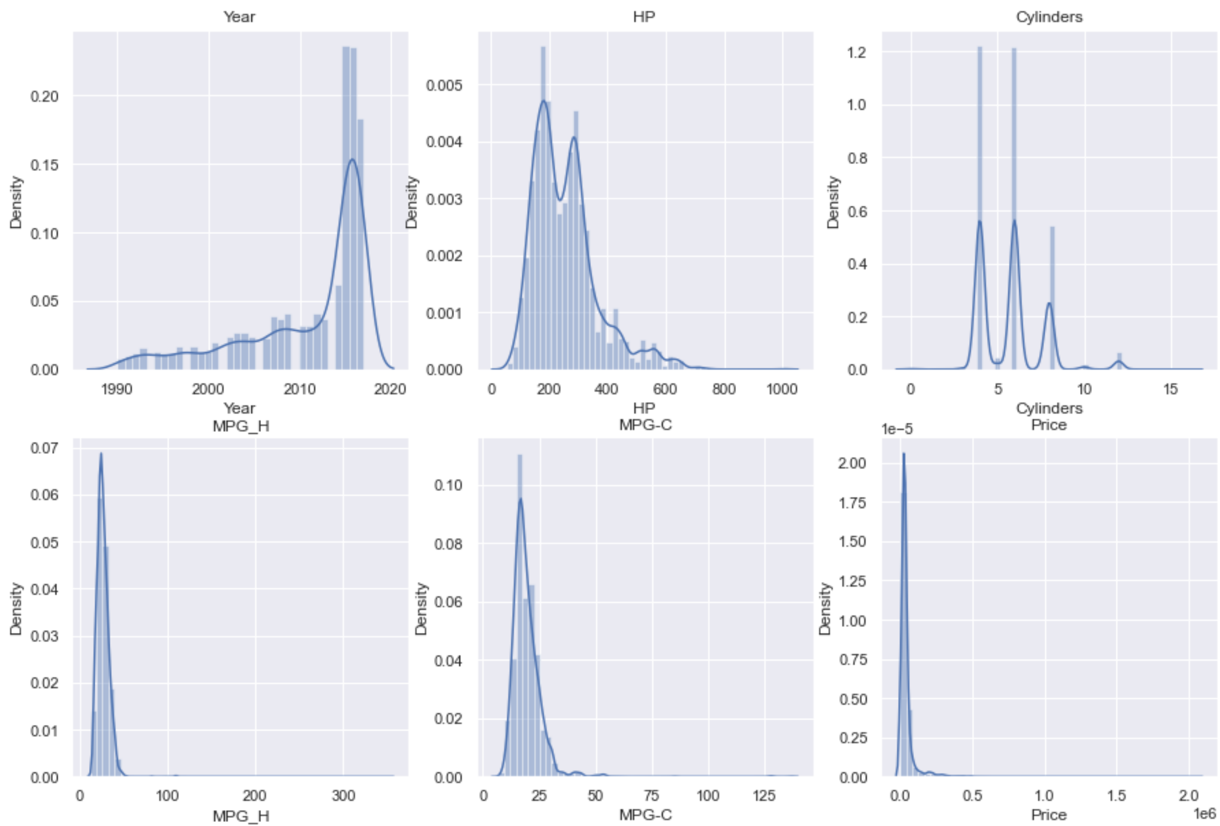
CHECK THIS FOR SUBPLOT : https://matplotlib.org/stable/gallery/subplots_axes_and_figures/subplots_demo.html (https://matplotlib.org/stable/gallery/subplots_axes_and_figures/subplots_demo.html)

In [40]:

```python
# plot all the columns present in list l together using subplot of dimention
c=0
plt.figure(figsize=(15,10))

for i in l:
    c=c+1
    plt.subplot(2,3,c)
    plt.title(i)
    sns.distplot(df[i])

plt.show()
```



2. Bar plots

10 points

Plot a histogram depicting the make in X axis and number of cars in y axis.

In [ ]:

```
YouTubeVideo('SrJLnIHkXIY',width=700, height=400)
```

BAR PLOT LINK USING KIND PARAMETER : [https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.html (https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.html)](https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.html)

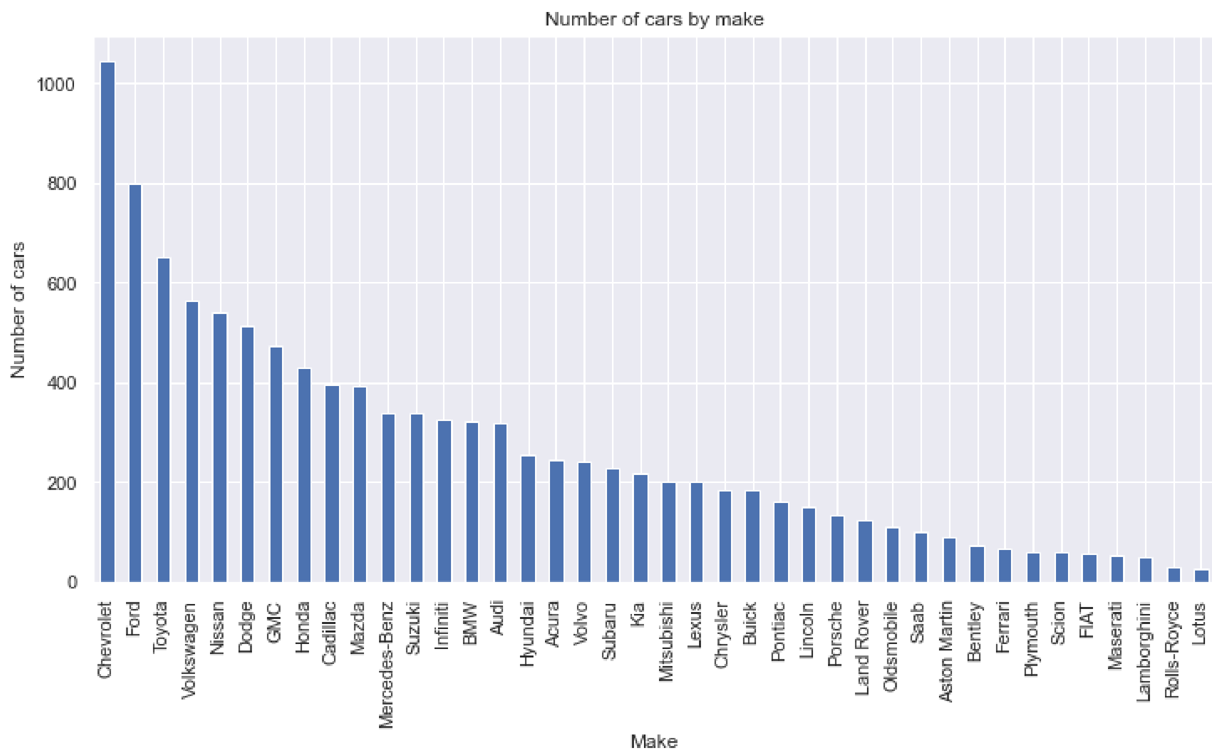
In [43]:

```
plt.figure(figsize = (12,8))

# use nlargest and then .plot to get bar plot like below output
df.Make.value_counts().nlargest(40).plot(kind="bar",figsize=(12,6))

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

Out[43]: Text(0.5, 0, 'Make')



Observation:

In this plot we can see that we have plot the bar plot with the cars model and nos. of cars.

3. Count Plot

10 points

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable.

In [ ]:
```
YouTubeVideo('2VbWiE8IfiA',width=700, height=400)
```

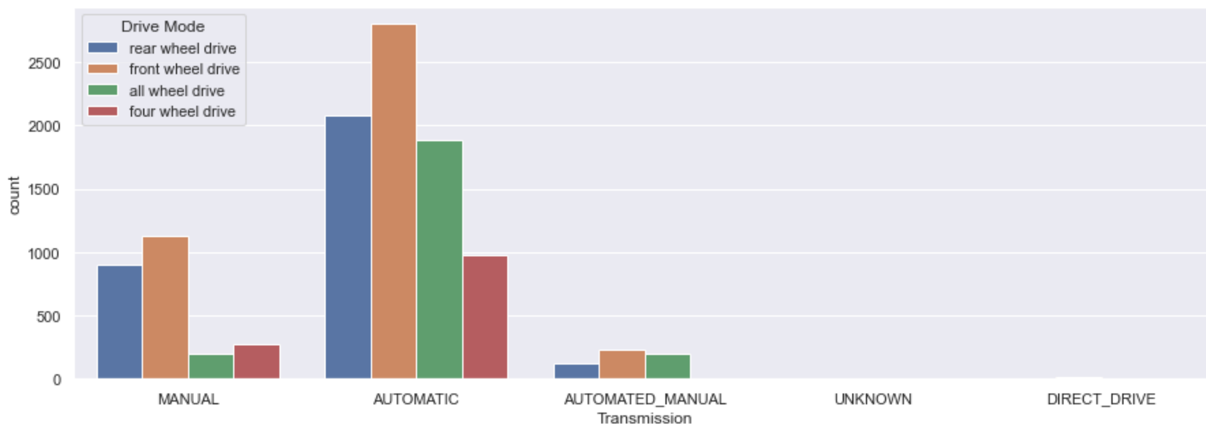
Plot a countplot for a variable Transmission vertically with hue as Drive mode

`COUNTPLOT LINK` : [https://seaborn.pydata.org/generated/seaborn.countplot.html (https://seaborn.pydata.org/generated/seaborn.countplot.html)](https://seaborn.pydata.org/generated/seaborn.countplot.html)

In [46]:
```
plt.figure(figsize=(15,5))

# plot countplot on transmission and drive mode
sns.countplot(x="Transmission",hue='Drive Mode',data=df)
plt.show()


# 'Cylinders', y='Price'
```



Observation:

In this count plot, We have plot the feature of Transmission with help of hue.
We can see that the the nos of count and the transmission type and automated manual is plotted.
Drive mode as been given with help of hue.

Visualising Bivariate Distributions

Bivariate distributions are simply two univariate distributions plotted on x and y axes respectively. They help you observe the relationship between the two variables.

1. Scatterplots

Scatterplots are used to find the correlation between two continuos variables.

10 points

Using scatterplot find the correlation between 'HP' and 'Price' column of the data.

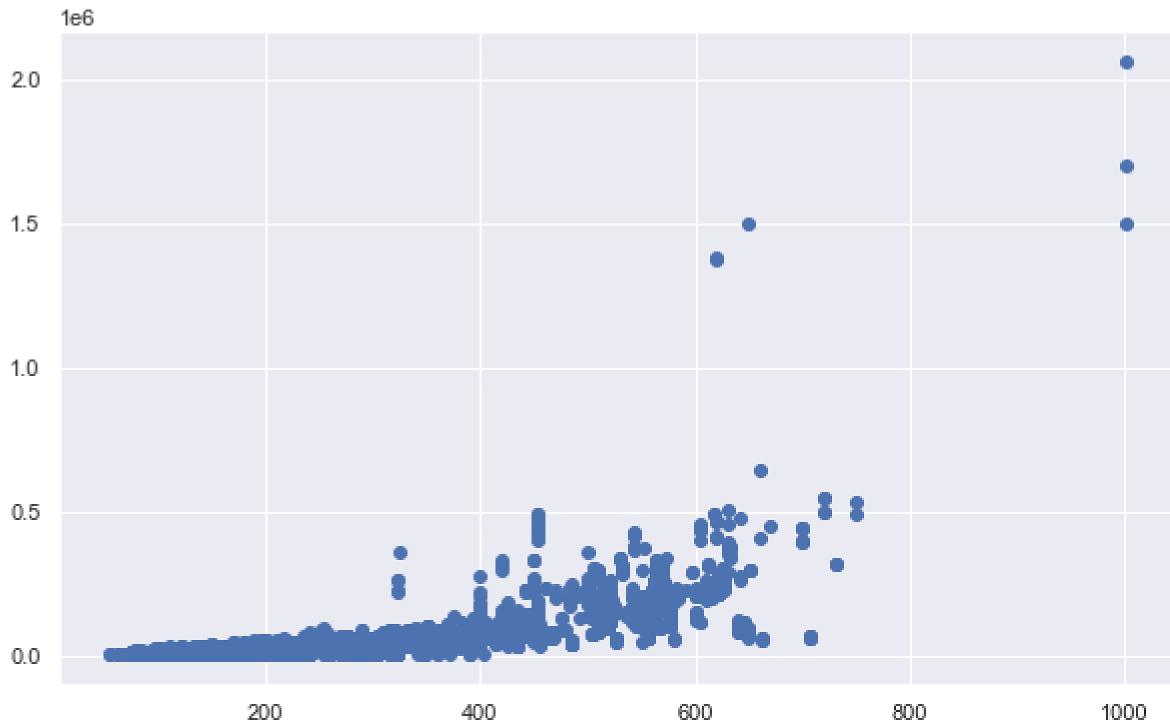
In [ ]:
```
YouTubeVideo('cu0nHeXyO4M',width=700, height=400)
```

CHECK THIS SCATTERPLOT METHOD ON STACKOVERFLOW :
https://stackoverflow.com/questions/57435771/scatter-plot-with-subplot-in-seaborn
(https://stackoverflow.com/questions/57435771/scatter-plot-with-subplot-in-seaborn)

In [47]:
```
## Your code here -
fig, ax = plt.subplots(figsize=(10,6))

# plot scatterplot on hp and price
ax.scatter(df['HP'],df['Price'])
ax.xlabel=('HP')
ax.ylabel=('Price')
plt.show()

```



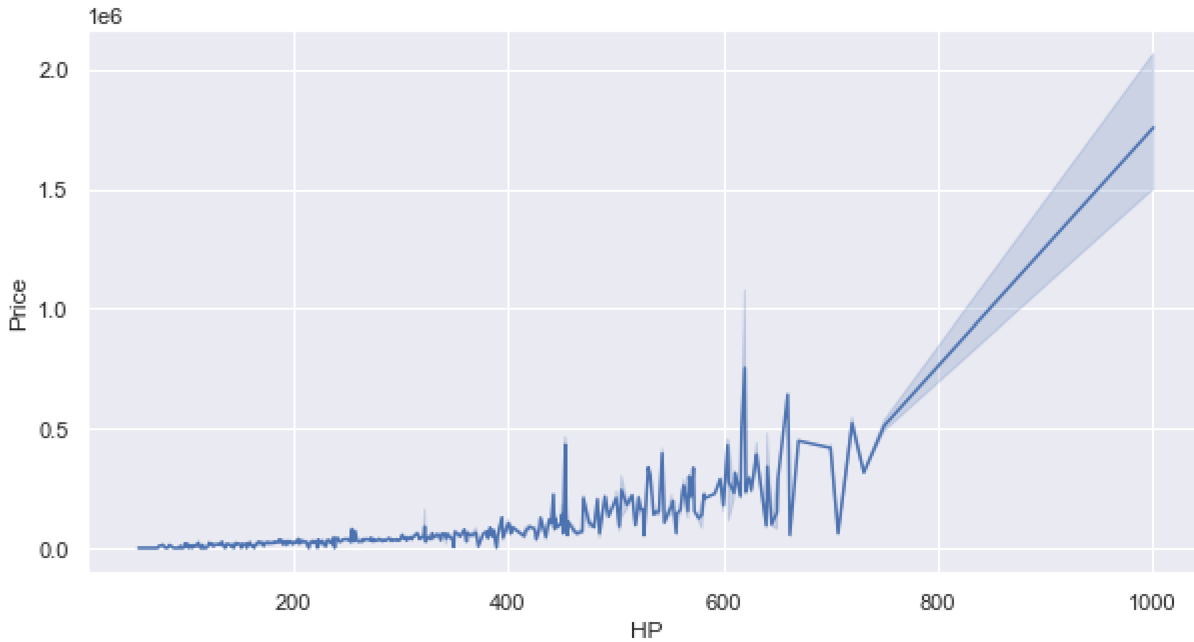
Observation:

It is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.
We have plot the scatter plot with x axis as HP and y axis as Price.
The data points between the features should be same either wise it give errors.

2.Line Plot

In [48]:
```
plt.figure(figsize=(10,5))
sns.lineplot(x='HP',y='Price',data=df)
plt.show()

```



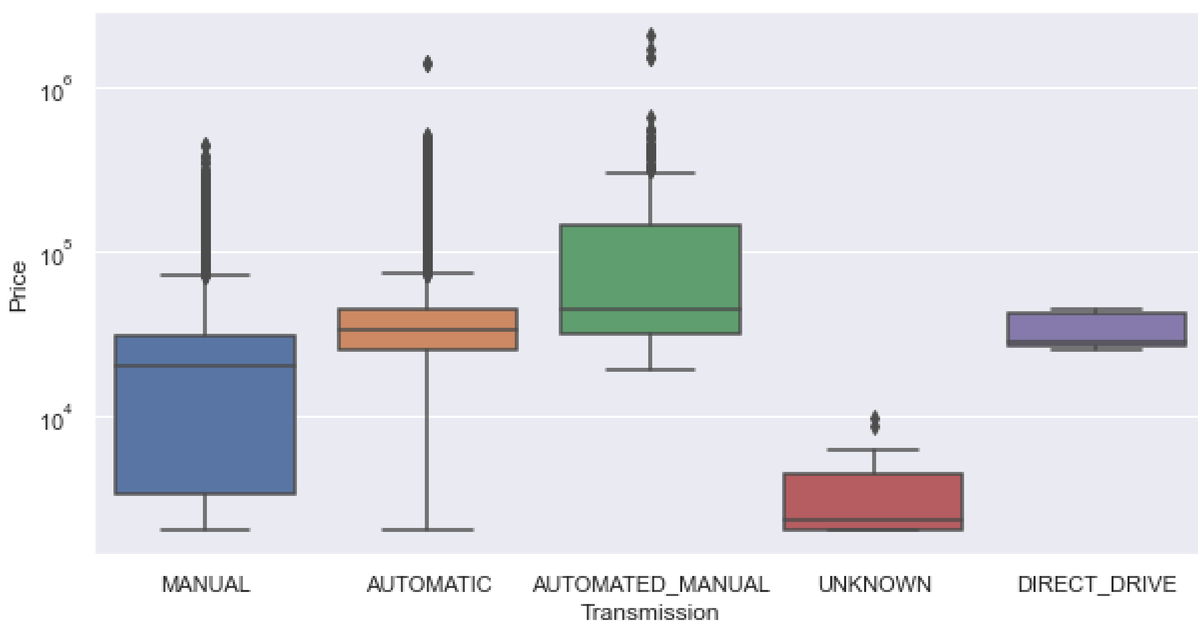
3.Box Plot w.r.t various variables

In [49]:

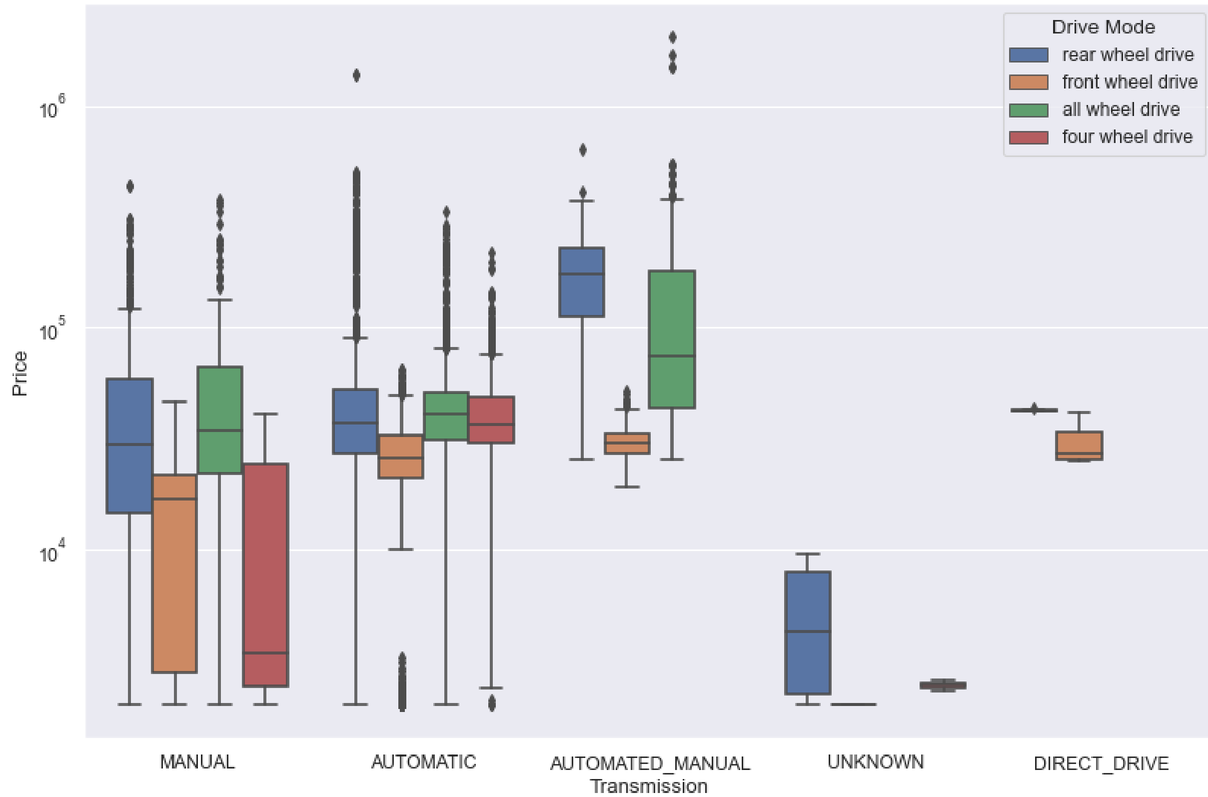
```
plt.figure(figsize=(10,5))
sns.boxplot(x='Drive Mode',y='Price',data=df)
plt.yscale('log')
plt.show()
```

In [50]:

```
plt.figure(figsize=(10,5))
sns.boxplot(x='Transmission',y='Price',data=df)
plt.yscale('log')
plt.show()
```



In [51]:
```
plt.figure(num=None,figsize=(12,8),dpi=80,facecolor='w',edgecolor='k')
sns.boxplot(x='Transmission',y='Price',hue='Drive Mode',data=df)
plt.yscale('log')
plt.show()
```



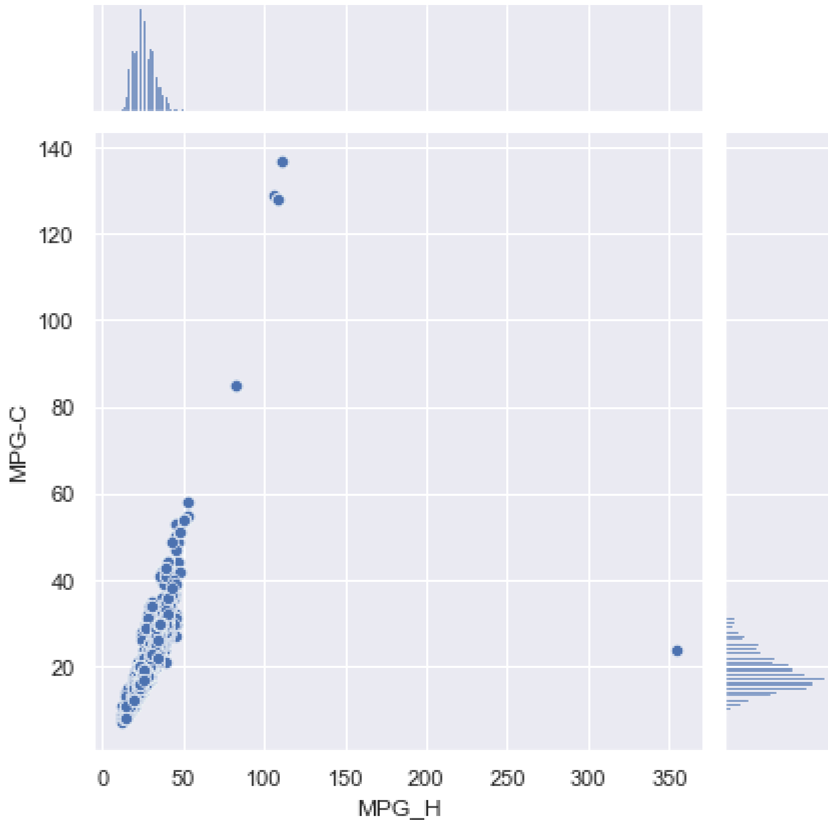
4. joint distributions

Seaborn's jointplot displays a relationship between 2 variables (bivariate) as well as 1D profiles (univariate) in the margins. This plot is a convenience class that wraps JointGrid

In [ ]:
```
YouTubeVideo('pz9bme5NIWU',width=700, height=400)
```

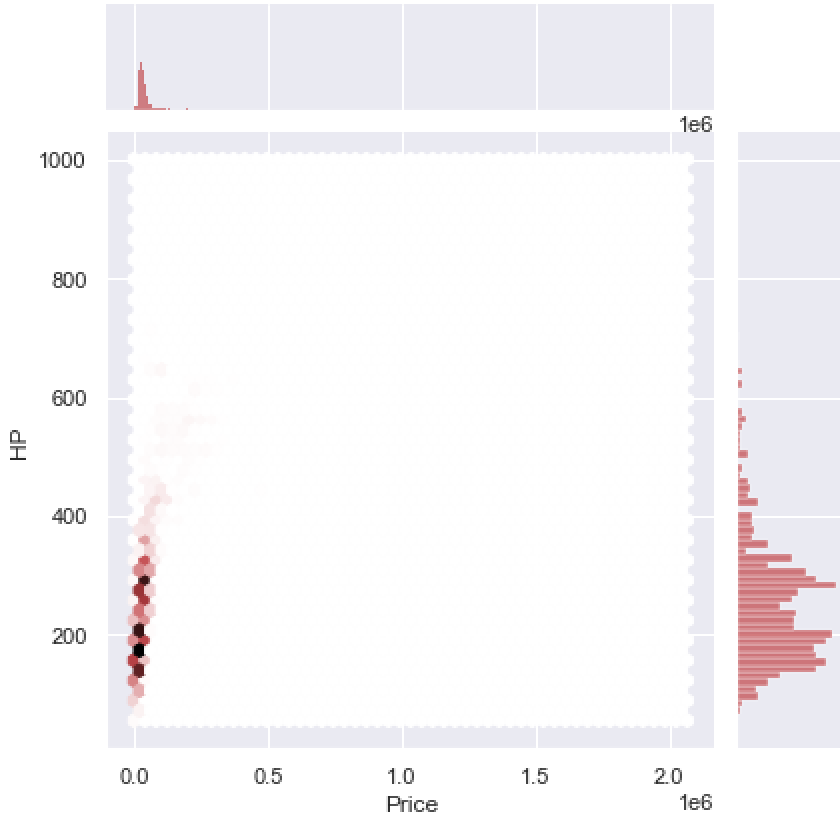
`CHECK TYPE OF JOINTPLOT` : [https://seaborn.pydata.org/generated/seaborn.jointplot.html (https://seaborn.pydata.org/generated/seaborn.jointplot.html)](https://seaborn.pydata.org/generated/seaborn.jointplot.html)

In [52]:
```
# joint plots of MPG_H and MPG-C
sns.jointplot('MPG_H','MPG-C',df)
plt.show()
```



In [53]:
```
sns.jointplot('Price','Year',df)
plt.show()
```

In [55]:
```
sns.jointplot('Price','HP',df,kind="hex",color="r")
plt.show()
```



Observations:

Jointplot is library specific and can be used to quickly visualize and analyze the relationship between two variables and describe their individual distributions on the same plot.
In this plot we can see the relationship of MPG-C abd MPG_H.

You can adjust the arguments of the jointplot() to make the plot more readable.

5. Plotting Aggregated Values across Categories

Bar Plots - Mean, Median and Count Plots

30 points

Bar plots are used to **display aggregated values** of a variable, rather than entire distributions. This is especially useful when you have a lot of data which is difficult to visualise in a single figure.

For example, say you want to visualise and *compare the Price across Cylinders*. The `sns.barplot()` function can be used to do that.

In [ ]:

```
YouTubeVideo('SrJLnIHkXIY',width=700, height=400)
```

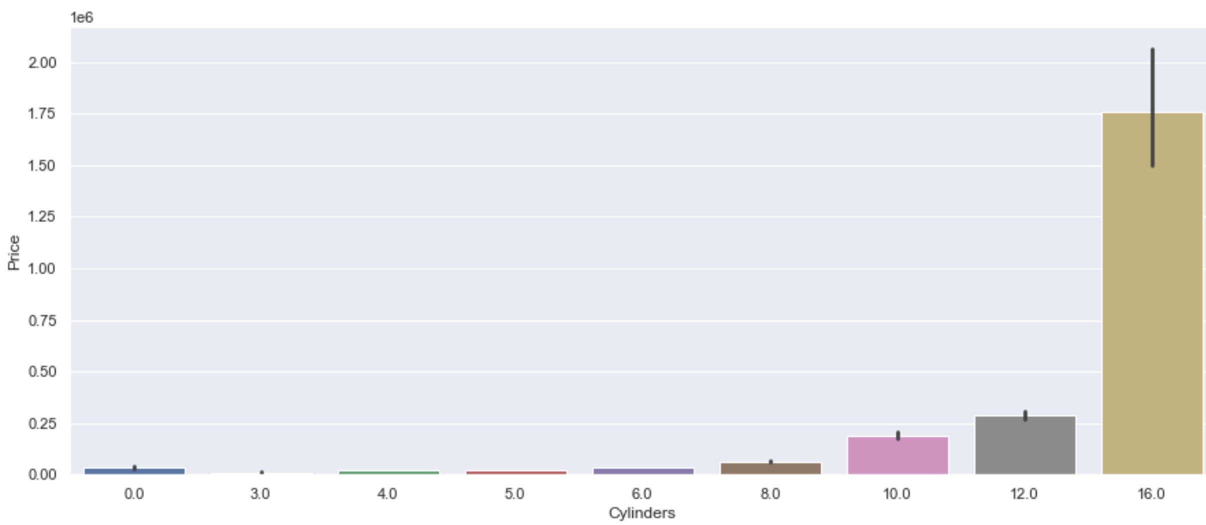
`BARPLOT USING SEABORN` : https://seaborn.pydata.org/generated/seaborn.barplot.html (https://seaborn.pydata.org/generated/seaborn.barplot.html)

In [57]:

```
# bar plot with default statistic=mean between Cylinder and Price

plt.figure(figsize=(15,6))
sns.barplot(x="Cylinders",y="Price",data=df)
plt.show()


```

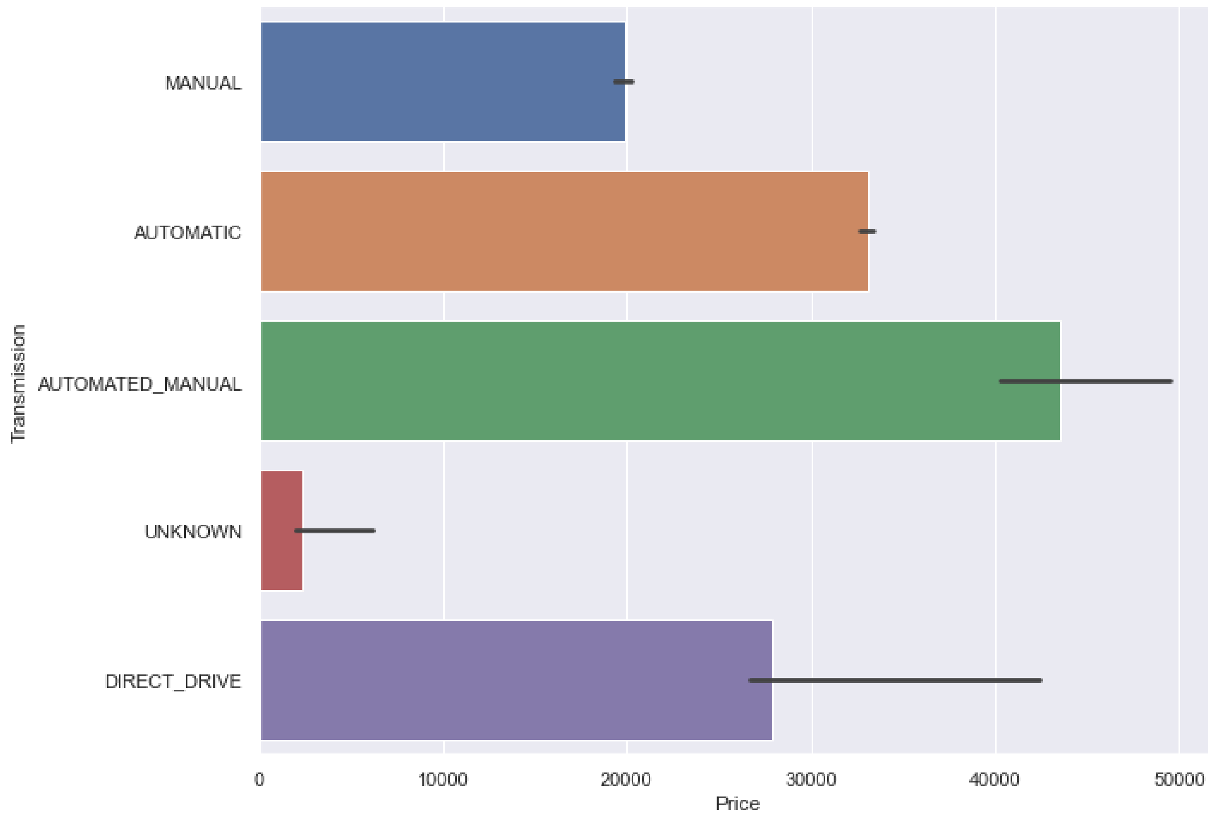


Observation:

By default, seaborn plots the mean value across categories, though you can plot the count, median, sum etc.
Also, barplot computes and shows the confidence interval of the mean as well.

When you want to visualise having a large number of categories, it is helpful to plot the categories across the y-axis. Let's now *drill down into Transmission sub categories*.

In [58]:
```
# Plotting categorical variable Transmission across the y-axis
plt.figure(figsize=(10,8))
sns.barplot(x="Price",y="Transmission",data=df,estimator=np.median)
plt.show()
```

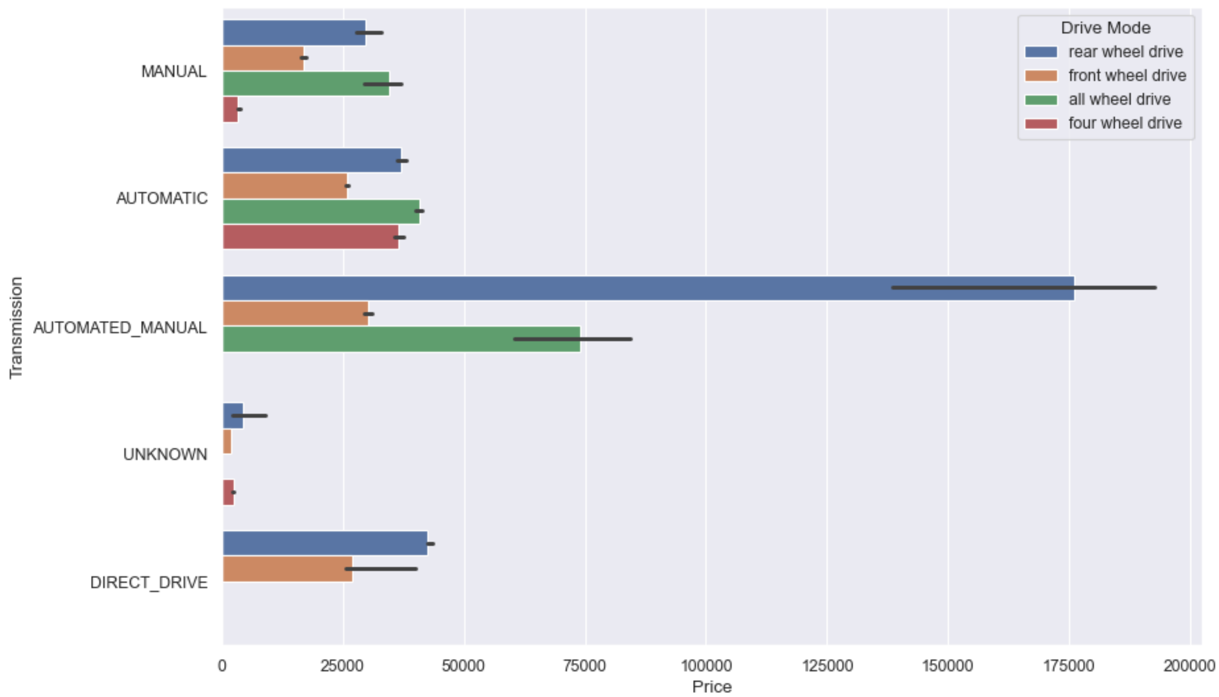


Plot bar plot for Price and Transmission with hue="Drive Mode"

In [59]:
```
plt.figure(num=None, figsize=(12, 8), dpi=80, facecolor='w', edgecolor='k')

# Plot bar plot for Price and Transmission , specify hue="Drive Mode"
sns.barplot(x="Price",y="Transmission",hue="Drive Mode",data=df,estimator=np
plt.show()

```



These plots looks beutiful isn't it? In Data Analyst life such charts are there unavoidable friend.:)

Multivariate Plots

1. Pairplot

10 points

Plot a pairplot for the dataframe df.

In [ ]:

```
YouTubeVideo('JblXpKS4cg8',width=700, height=400)
```

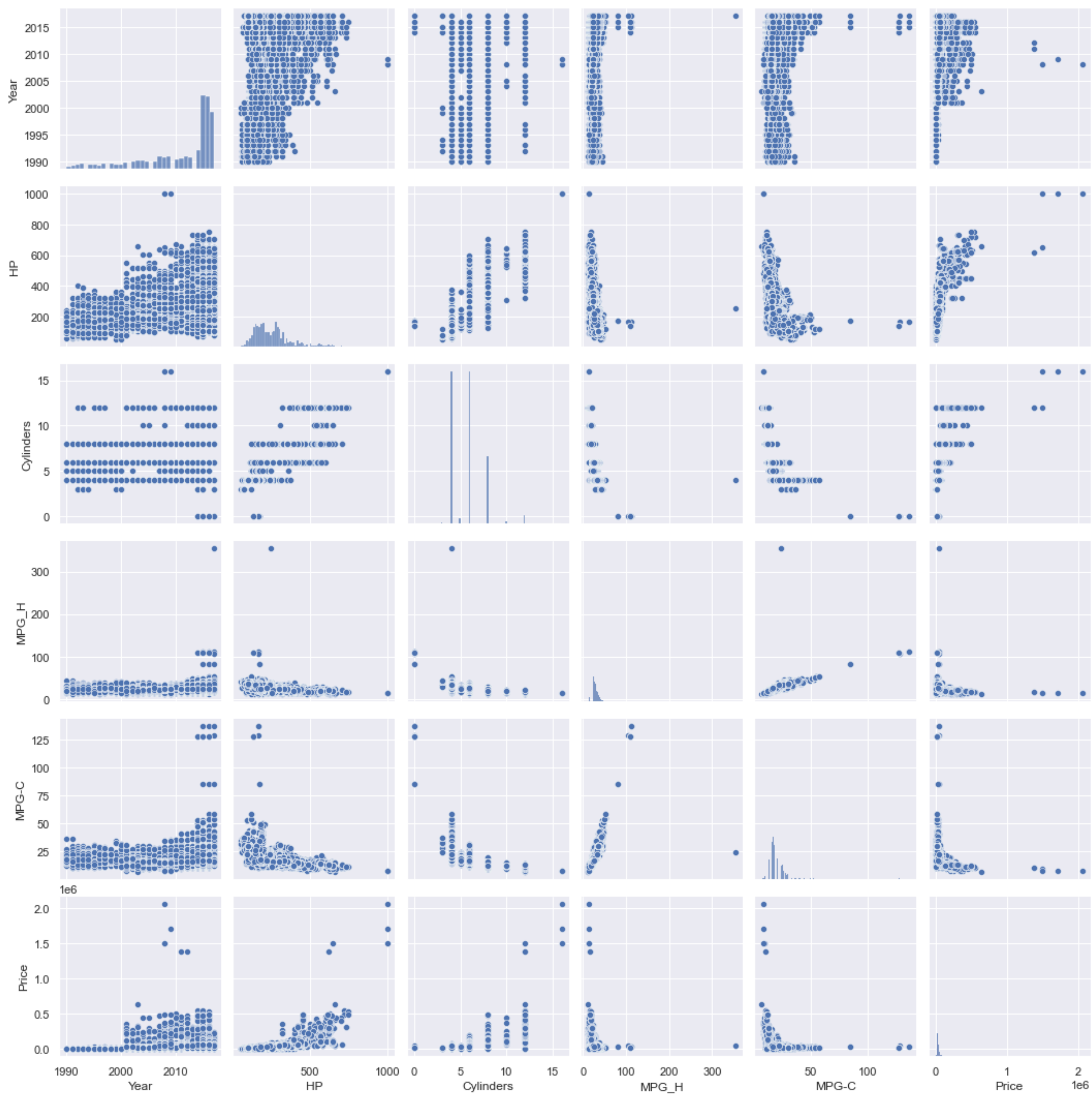
`SEABORN PAIRPLOT` : [https://seaborn.pydata.org/generated/seaborn.pairplot.html (https://seaborn.pydata.org/generated/seaborn.pairplot.html)](https://seaborn.pydata.org/generated/seaborn.pairplot.html)

In [60]:

```
# plot pairplot on df

sns.pairplot(df)
plt.show
```

Out[60]: <function matplotlib.pyplot.show(close=None, block=None)>



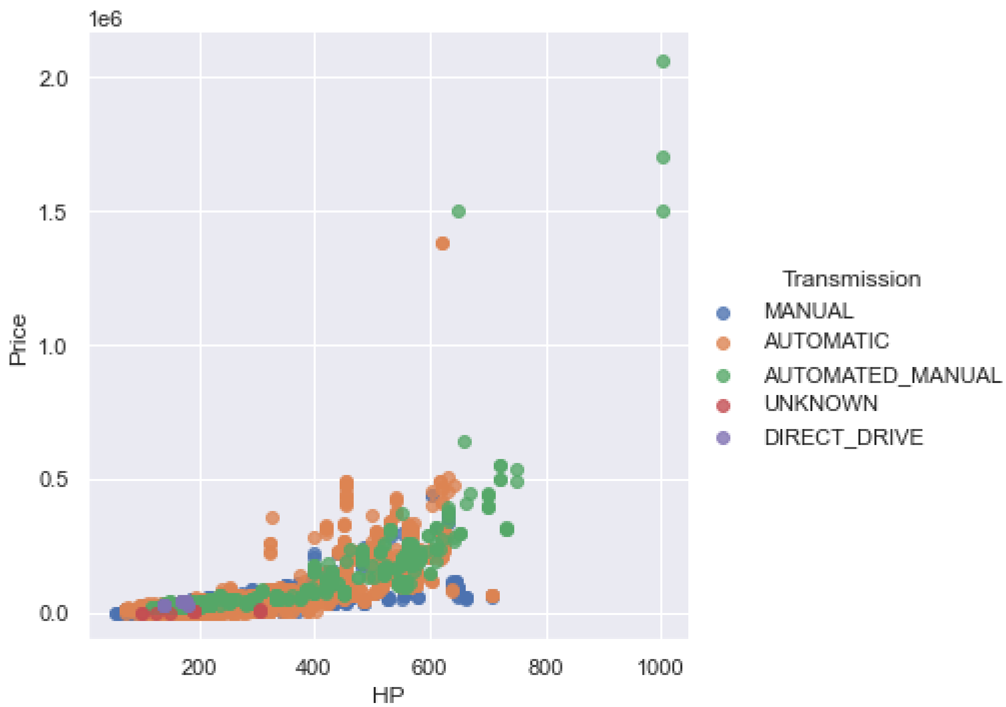
Observation:

To plot multiple pairwise bivariate distributions in a dataset, you can use the pairplot() function. This shows the relationship for (n, 2) combination of variable in a DataFrame as a matrix of plots and the diagonal plots are the univariate plots.

2.Multivariate Scatter Plot

In [61]:
```
plt.figure(figsize=(15,5))
sns.lmplot(x="HP",y="Price",hue="Transmission",data=df,fit_reg=False)
plt.show()
```

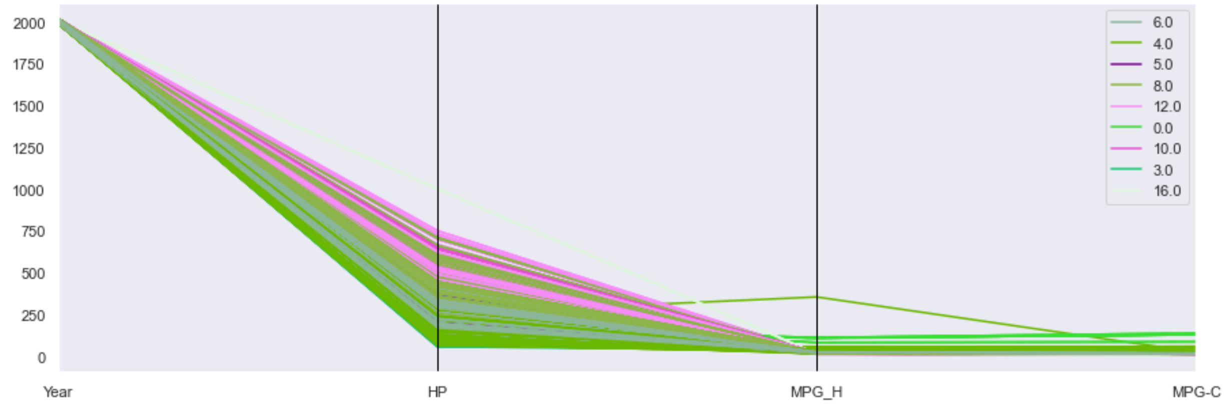
```
<Figure size 1080x360 with 0 Axes>
```



Parallel Co-ordinates

In [63]:
```
l1=l.copy()
l1.remove('Price')
```

In [64]:
```
from pandas.plotting import parallel_coordinates
plt.figure(figsize=(15,5))
parallel_coordinates(df[l1],'Cylinders')
plt.show()
```



3. Heatmaps

A heat map is a two-dimensional representation of information with the help of colors. Heat maps can help the user visualize simple or complex information

In [ ]:
```
YouTubeVideo('BXq5XhA-fb4',width=700, height=400)
```

20 points

Using heatmaps plot the correlation between the features present in the dataset.

`SEABORN HEATMAP` : [https://seaborn.pydata.org/generated/seaborn.pairplot.html (https://seaborn.pydata.org/generated/seaborn.pairplot.html)](https://seaborn.pydata.org/generated/seaborn.pairplot.html)

In [65]:

```
#find the correlation of features of the data
corr = df.corr()
corr
# print corr

```

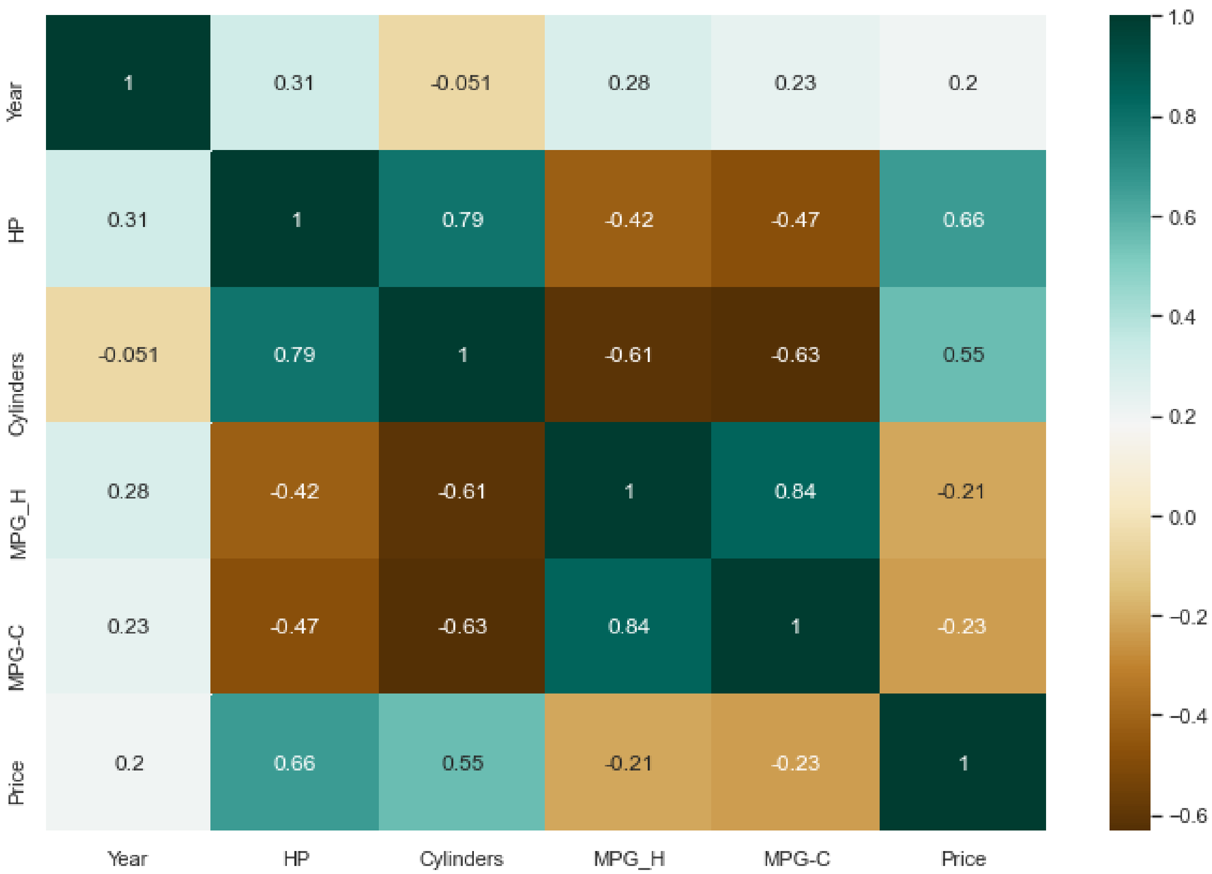
Out[65]:

	Year	HP	Cylinders	MPG_H	MPG-C	Price
Year	1.000000	0.314971	-0.050598	0.284237	0.234135	0.196789
HP	0.314971	1.000000	0.788007	-0.420281	-0.473551	0.659835
Cylinders	-0.050598	0.788007	1.000000	-0.611576	-0.632407	0.554740
MPG_H	0.284237	-0.420281	-0.611576	1.000000	0.841229	-0.209150
MPG-C	0.234135	-0.473551	-0.632407	0.841229	1.000000	-0.234050
Price	0.196789	0.659835	0.554740	-0.209150	-0.234050	1.000000

In [68]:

```
# Using the correlated df, plot the heatmap
# set cmap = 'BrBG', annot = True - to get the same graph as shown below
# set size of graph = (12,8)

plt.figure(figsize=(12,8))
sns.heatmap(corr,cmap='BrBG',annot=True)
plt.show()
```



Observation:

A heatmap contains values representing various shades of the same colour for each value to be plotted. Usually the darker shades of the chart represent higher values than the lighter shade. For a very different value a completely different colour can also be used.

The above heatmap plot shows correlation between various variables in the colored scale of -1 to 1.

Amazing work done ! you have really made eye catchy visualization plots so far. Did you felt its complicate to understand the above plot?. Hey smarty don't worry, in near assignments you will have enough practise to analyse and prepare insights from such plots that you will become pro in this field.

Then soon you will be like below meme
image.png

Have a sweet cookie:) Congratulations! you have completed the 6th milestone challenge too.

FeedBack

We hope you've enjoyed this course so far. We're committed to helping you use AIforAll course to its full potential so you can grow with us. And that's why we need your help in form of a feedback here

We appreciate your time for your thoughtful comment.

https://forms.gle/SedkKUD2TNPCnafj8 (https://forms.gle/SedkKUD2TNPCnafj8)