Experiment No. 1

Name : Sudhanshu Narsude

Class : TE COMPS

UID : 2019130044

Subject : Data Analytics

Aim : Performing Exploratory Data Analysis (EDA) on Laptop dataset such as number of data samples, number of features, number of classes, number of data samples per class, removing missing values, conversion to numbers, using seaborn library to plot different graphs.

Theory :

Exploratory Data Analysis, or EDA, is an important step in any Data Analysis or Data Science project. EDA is the process of investigating the dataset to discover patterns, and anomalies (outliers), and form hypotheses based on our understanding of the dataset.

EDA involves generating summary statistics for numerical data in the dataset and creating various graphical representations to understand the data better. In this article, we will understand EDA with the help of an example dataset. We will use Python language (Pandas library) for this purpose.

Implementation:

Step 1 : Importing all the required python libraries.

```
# Imported all the required Libraries
import numpy as np
import pandas as pd
import seaborn as sea
import matplotlib.pyplot as plt
from google.colab import files
data = files.upload()
```

Step 2: Loading data

```
df = pd.read_csv("laptop_data.csv")
df.head()
```

	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
0	0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	71378.6832
1	1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	47895.5232
2	2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	30636.0000
3	3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	135195.3360
4	4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	96095.8080

Step 3: Checking Size of our dataset

```
# Shows total rows and columns in a dataset
df.shape
```

```
(1303, 12)
```

Step 4: Checking whether there exists any null values in our dataset or not.

```
# Checks whether there is null values in our dataset (observation : We
have 0 null values hence we will conclude it is a clean data)
df.isnull().sum()
```

```
Unnamed: 0          0
Company             0
TypeName            0
Inches              0
ScreenResolution    0
Cpu                 0
Ram                 0
Memory              0
Gpu                 0
OpSys               0
Weight              0
Price               0
dtype: int64
```

Step 5: describe() function gives a better idea about the data.

```
df.describe()
```

	Unnamed: 0	Inches	Price
count	1303.00000	1303.000000	1303.000000
mean	651.00000	15.017191	59870.042910
std	376.28801	1.426304	37243.201786
min	0.00000	10.100000	9270.720000
25%	325.50000	14.000000	31914.720000
50%	651.00000	15.600000	52054.560000
75%	976.50000	15.600000	79274.246400
max	1302.00000	18.400000	324954.720000

Step 6 :

```
# Dropped a 'Unnamed: 0' column which is not required
df.drop(columns=['Unnamed: 0'],inplace=True)
df.head()
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	71378.6832
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	47895.5232
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	30636.0000
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	135195.3360
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	96095.8080

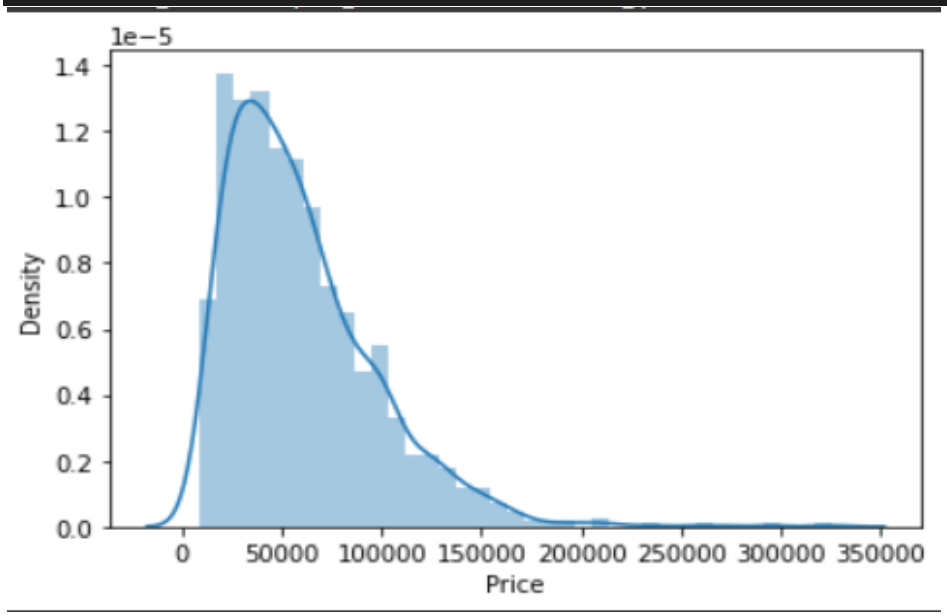
Step 7:

```
# Changes in Weight and Ram column (converted them to numeric values by
removing the unit written after value which is GB and kg)
df['Ram']=df['Ram'].str.replace("GB", "")
df['Weight']=df['Weight'].str.replace("kg", "")
df['Ram']=df['Ram'].astype('int64')
df['Weight']=df['Weight'].astype('float64')
df.info()
```

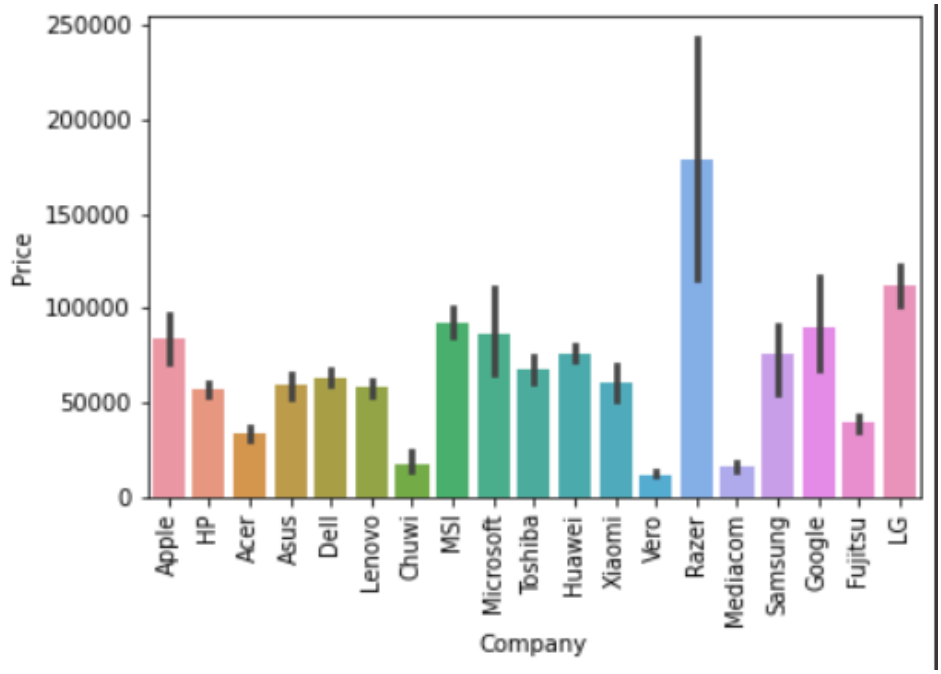
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Company           1303 non-null   object
 1   TypeName          1303 non-null   object
 2   Inches            1303 non-null   float64
 3   ScreenResolution  1303 non-null   object
 4   Cpu               1303 non-null   object
 5   Ram               1303 non-null   int64
 6   Memory            1303 non-null   object
 7   Gpu               1303 non-null   object
 8   OpSys             1303 non-null   object
 9   Weight            1303 non-null   float64
 10  Price             1303 non-null   float64
dtypes: float64(3), int64(1), object(7)
memory usage: 112.1+ KB
```

Step 8: Plotting graphs using seaborn to find relationships between them.

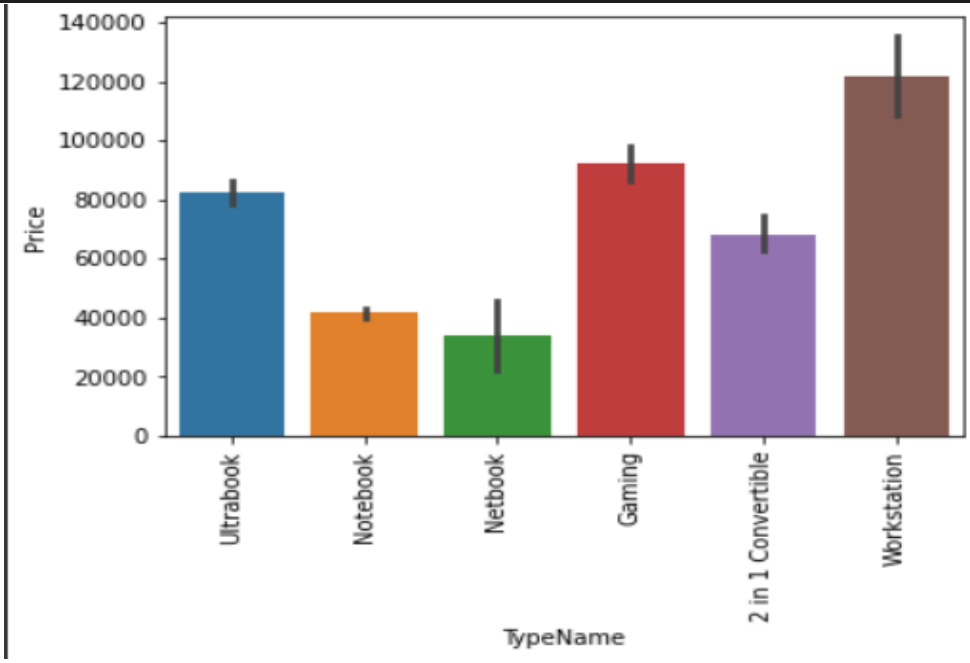
```
sea.distplot(df['Price'])
plt.show()
```



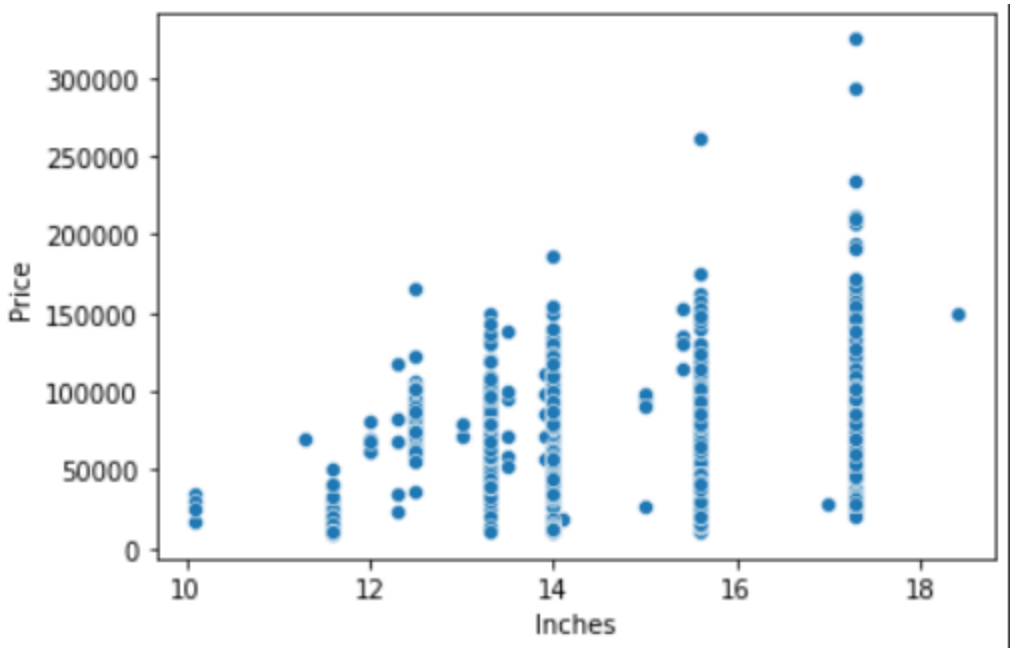

```
# Average Price of each brand

sea.barplot(x=df['Company'], y=df['Price'])

plt.xticks(rotation="vertical")

plt.show()
```



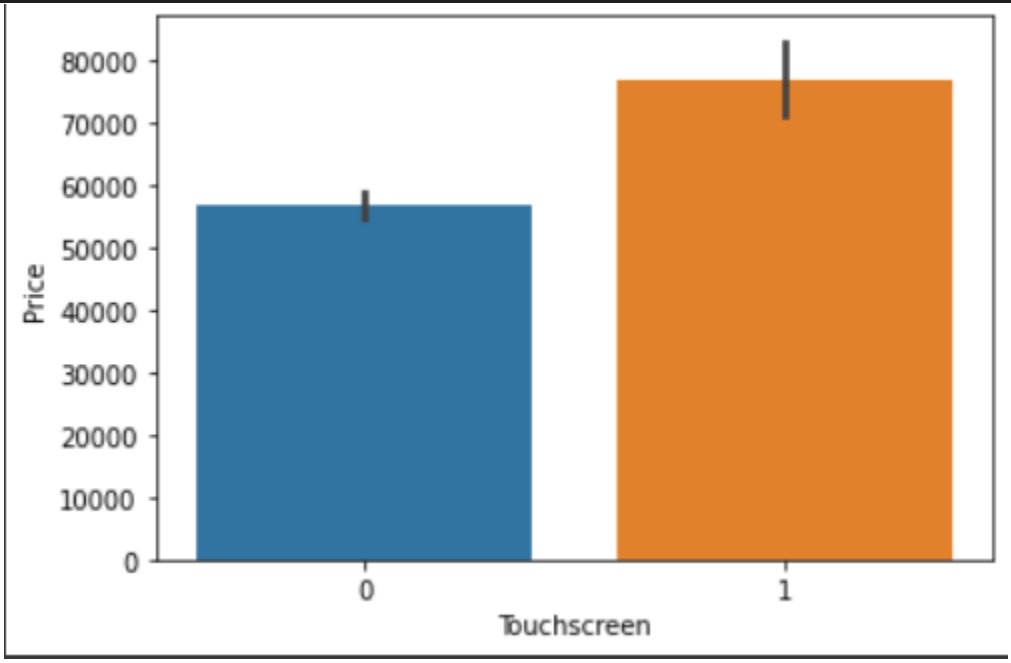

```
#data['TypeName'].value_counts().plot(kind='bar')  [Types of Laptops]

sea.barplot(x=df['TypeName'], y=df['Price'])
plt.xticks(rotation="vertical")
plt.show()
```




```
sea.scatterplot(x=df['Inches'],y=df['Price'])
```




```
df['Touchscreen'] = df['ScreenResolution'].apply(lambda x:1 if
'Touchscreen' in x else 0)
#how many laptops in data are touchscreen
sea.countplot(df['Touchscreen'])
#Plot against price
sea.barplot(x=df['Touchscreen'],y=df['Price'])
```

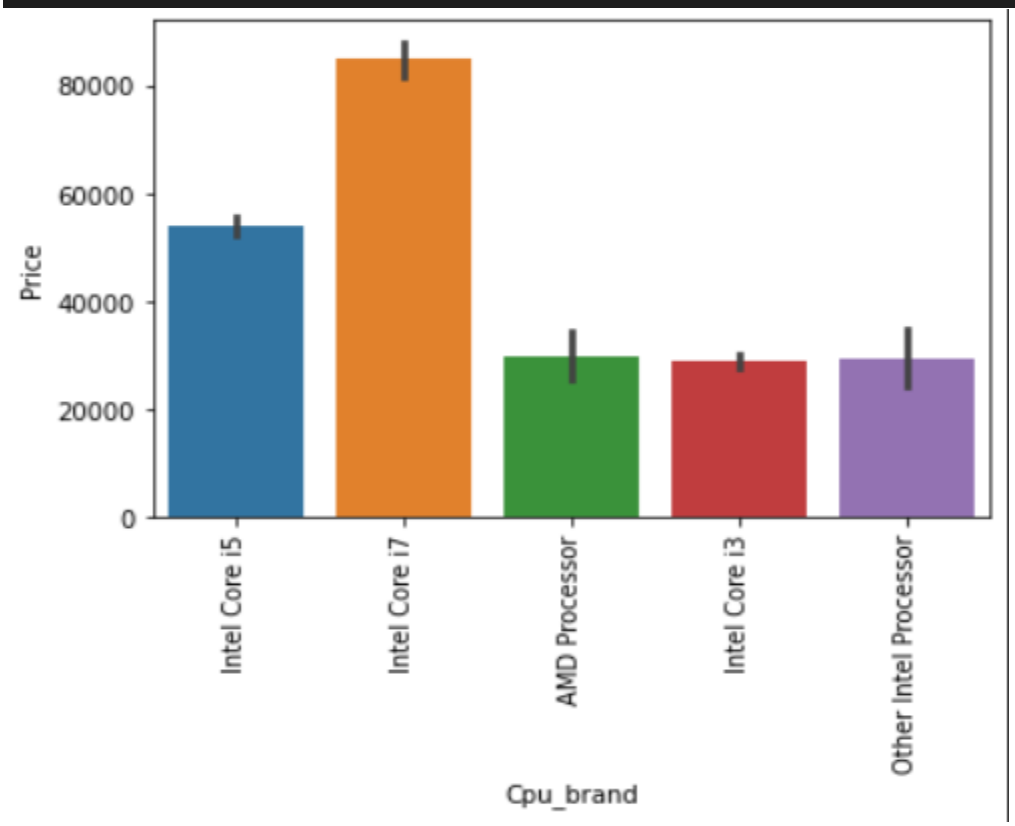


Step 9 : Checking Correlation

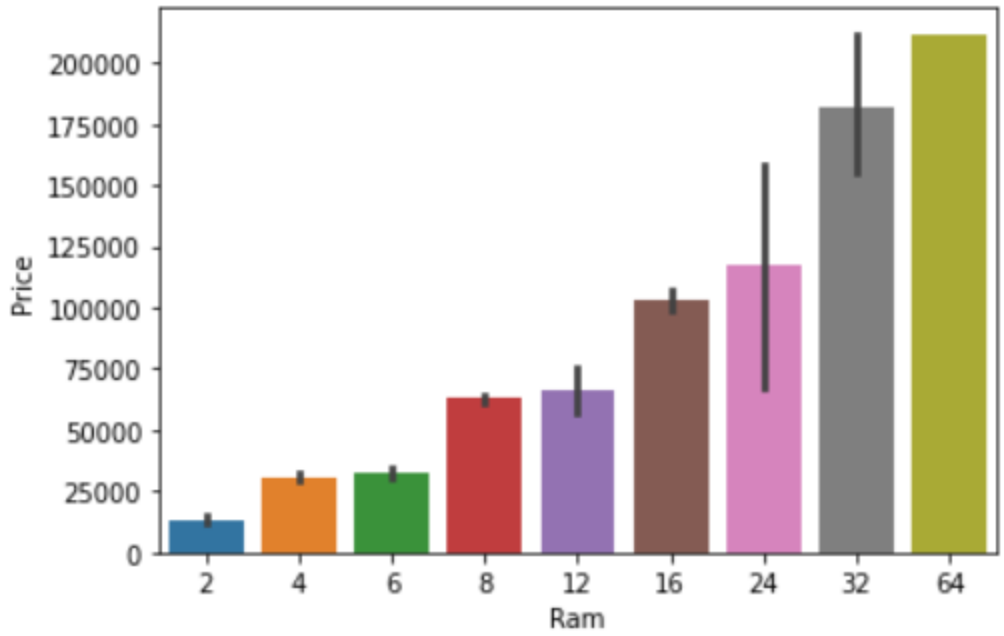
```
df['ppi'] = (((df['X_res']**2) +
(df['Y_res']**2))**0.5/df['Inches']).astype('float')
df.corr()['Price'].sort_values(ascending=False)
```

```
Price          1.000000
Ram            0.743007
X_res          0.556529
Y_res          0.552809
ppi            0.473487
Ips            0.252208
Weight         0.210370
Touchscreen    0.191226
Inches         0.068197
Name: Price, dtype: float64
```

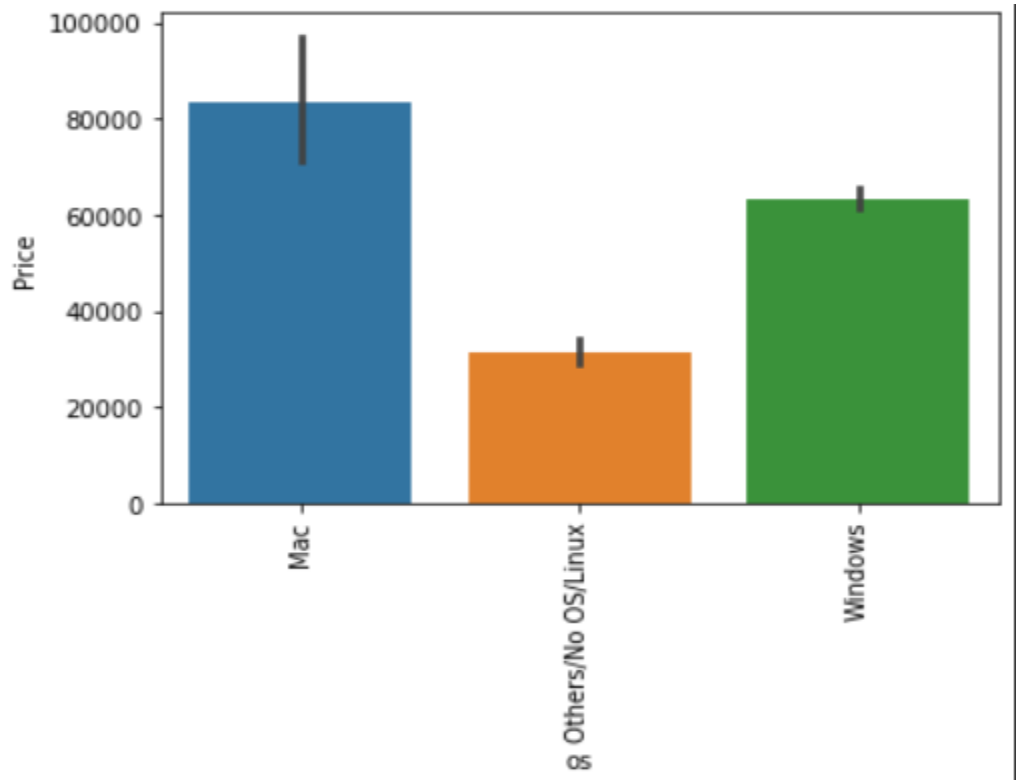
```
sea.barplot(x=df['Cpu_brand'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



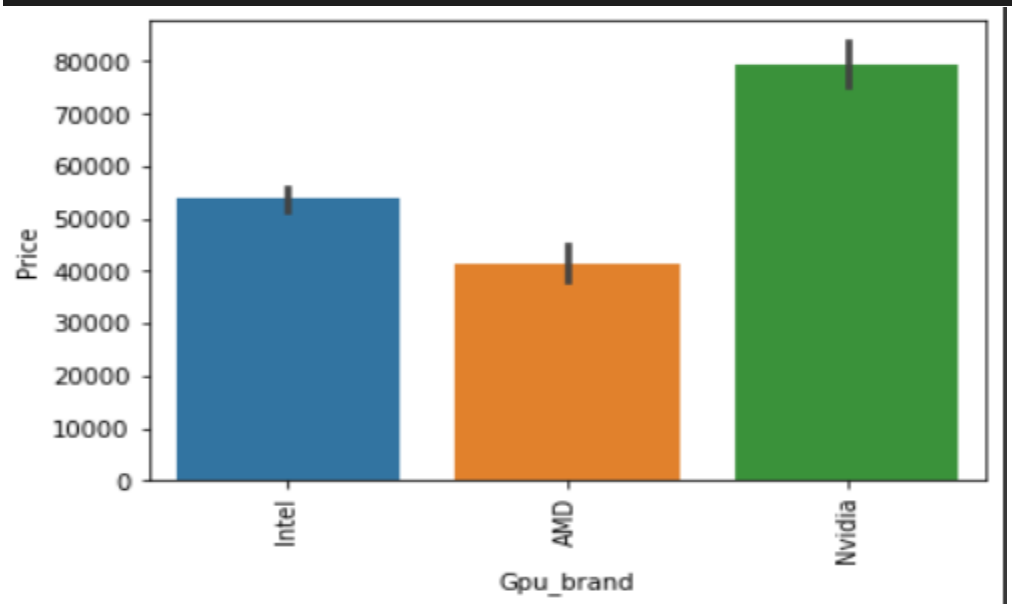

```
sea.barplot(df['Ram'], df['Price'])
plt.show()
```




```
#Get which OP sys
def cat_os(inp):
    if inp == 'Windows 10' or inp == 'Windows 7' or inp == 'Windows 10 S':
        return 'Windows'
    elif inp == 'macOS' or inp == 'Mac OS X':
        return 'Mac'
    else:
        return 'Others/No OS/Linux'
df['os'] = df['OpSys'].apply(cat_os)
df.drop(columns=['OpSys'],inplace=True)

sea.barplot(x=df['os'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```




```
sea.barplot(x=df['Gpu_brand'], y=df['Price'])

plt.xticks(rotation="vertical")

plt.show()
```



Conclusion :

In this Experiment I performed EDA on the selected dataset . I used google colab to perform my experiment and imported all python libraries which are required. First we need to understand what all information is given in the dataset . It consists of 1303 rows and 12 columns . In the dataset they have given information about the laptops with its company name , type name , Inches, CPU , RAM, Weight and Price. It is good that there are no NULL values. And we need little changes in weight and Ram column to convert them to numeric by removing the unit written after value. So we will perform data cleaning here to get the correct types of columns. EDA helps to prove our assumptions true or false. In other words, it helps to perform hypothesis testing. I used the seaborn library to plot the different graphs to get better visualization of data . The distribution of the target variable, which is Price of a laptop , is skewed and it is obvious that commodities with low prices are sold and purchased more than the branded ones. Razer, Apple, LG, Microsoft, Google, MSI laptops are expensive, and others are in the budget range. We can also use different graphs to have a better understanding of the dataset.

I also learned how to deal with the missing values in the dataset. In order to fill null values in a datasets, we use fillna(), replace() and interpolate() functions. These functions replace NaN values with some value of their own. All these functions help in filling null values in datasets of a DataFrame.