```
import numpy as np
import pandas as pd

df = pd.read_csv('laptop_data.csv')

df.head()
```

```
   Unnamed: 0 Company   TypeName  Inches
ScreenResolution  \
0           0   Apple  Ultrabook    13.3  IPS Panel Retina Display
2560x1600
1           1   Apple  Ultrabook    13.3
1440x900
2           2      HP   Notebook    15.6                    Full HD
1920x1080
3           3   Apple  Ultrabook    15.4  IPS Panel Retina Display
2880x1800
4           4   Apple  Ultrabook    13.3  IPS Panel Retina Display
2560x1600

                          Cpu   Ram               Memory  \
0        Intel Core i5 2.3GHz   8GB            128GB SSD
1        Intel Core i5 1.8GHz   8GB  128GB Flash Storage
2  Intel Core i5 7200U 2.5GHz   8GB            256GB SSD
3        Intel Core i7 2.7GHz  16GB            512GB SSD
4        Intel Core i5 3.1GHz   8GB            256GB SSD

                            Gpu  OpSys  Weight        Price
0  Intel Iris Plus Graphics 640  macOS  1.37kg   71378.6832
1        Intel HD Graphics 6000  macOS  1.34kg   47895.5232
2         Intel HD Graphics 620  No OS  1.86kg   30636.0000
3            AMD Radeon Pro 455  macOS  1.83kg  135195.3360
4  Intel Iris Plus Graphics 650  macOS  1.37kg   96095.8080
```

```
df.shape
```

```
(1303, 12)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        1303 non-null   int64
 1   Company           1303 non-null   object
 2   TypeName          1303 non-null   object
 3   Inches            1303 non-null   float64
 4   ScreenResolution  1303 non-null   object
 5   Cpu               1303 non-null   object
```

```
 6   Ram               1303 non-null   object
 7   Memory            1303 non-null   object
 8   Gpu               1303 non-null   object
 9   OpSys             1303 non-null   object
 10  Weight            1303 non-null   object
 11  Price             1303 non-null   float64
dtypes: float64(2), int64(1), object(9)
memory usage: 122.3+ KB
```

To find the duplicated rows, missing values and eliminate them .

```
print('The number of duplicated rows are -->',df.duplicated().sum())
```

```
The number of duplicated rows are --> 0
```

```
print('Is there any missing values :--> \n ',df.isnull().sum())
```

```
Is there any missing values :-->
  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
```

```
df.drop(columns= ["Unnamed: 0"],inplace=True) # removing the column
"Unnamed: 0"  it is of no use
```

```
df.head()
```

```
  Company   TypeName  Inches                    ScreenResolution  \
0   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600
1   Apple  Ultrabook    13.3                            1440x900
2      HP   Notebook    15.6                   Full HD 1920x1080
3   Apple  Ultrabook    15.4  IPS Panel Retina Display 2880x1800
4   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600

                          Cpu   Ram               Memory  \
0        Intel Core i5 2.3GHz   8GB            128GB SSD
1        Intel Core i5 1.8GHz   8GB  128GB Flash Storage
```

```
2  Intel Core i5 7200U 2.5GHz   8GB            256GB SSD
3         Intel Core i7 2.7GHz  16GB            512GB SSD
4         Intel Core i5 3.1GHz   8GB            256GB SSD

                             Gpu  OpSys  Weight        Price
0  Intel Iris Plus Graphics 640  macOS  1.37kg   71378.6832
1        Intel HD Graphics 6000  macOS  1.34kg   47895.5232
2         Intel HD Graphics 620  No OS  1.86kg   30636.0000
3            AMD Radeon Pro 455  macOS  1.83kg  135195.3360
4  Intel Iris Plus Graphics 650  macOS  1.37kg   96095.8080
```

```
df["Ram"]=df["Ram"].str.replace('GB','') # removing "GB "  from RAM
df["Weight"]=df["Weight"].str.replace('kg','') # removing "kg "  from
Weight

df.head()
```

```
  Company   TypeName  Inches                    ScreenResolution  \
0   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600
1   Apple  Ultrabook    13.3                        1440x900
2      HP   Notebook    15.6                  Full HD 1920x1080
3   Apple  Ultrabook    15.4  IPS Panel Retina Display 2880x1800
4   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600

                          Cpu Ram               Memory  \
0        Intel Core i5 2.3GHz   8            128GB SSD
1        Intel Core i5 1.8GHz   8  128GB Flash Storage
2  Intel Core i5 7200U 2.5GHz   8            256GB SSD
3        Intel Core i7 2.7GHz  16            512GB SSD
4        Intel Core i5 3.1GHz   8            256GB SSD

                             Gpu  OpSys Weight        Price
0  Intel Iris Plus Graphics 640  macOS   1.37   71378.6832
1        Intel HD Graphics 6000  macOS   1.34   47895.5232
2         Intel HD Graphics 620  No OS   1.86   30636.0000
3            AMD Radeon Pro 455  macOS   1.83  135195.3360
4  Intel Iris Plus Graphics 650  macOS   1.37   96095.8080
```

```
df['Ram']= df['Ram'].astype('int32')
df['Weight']=df['Weight'].astype('float32')

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
```

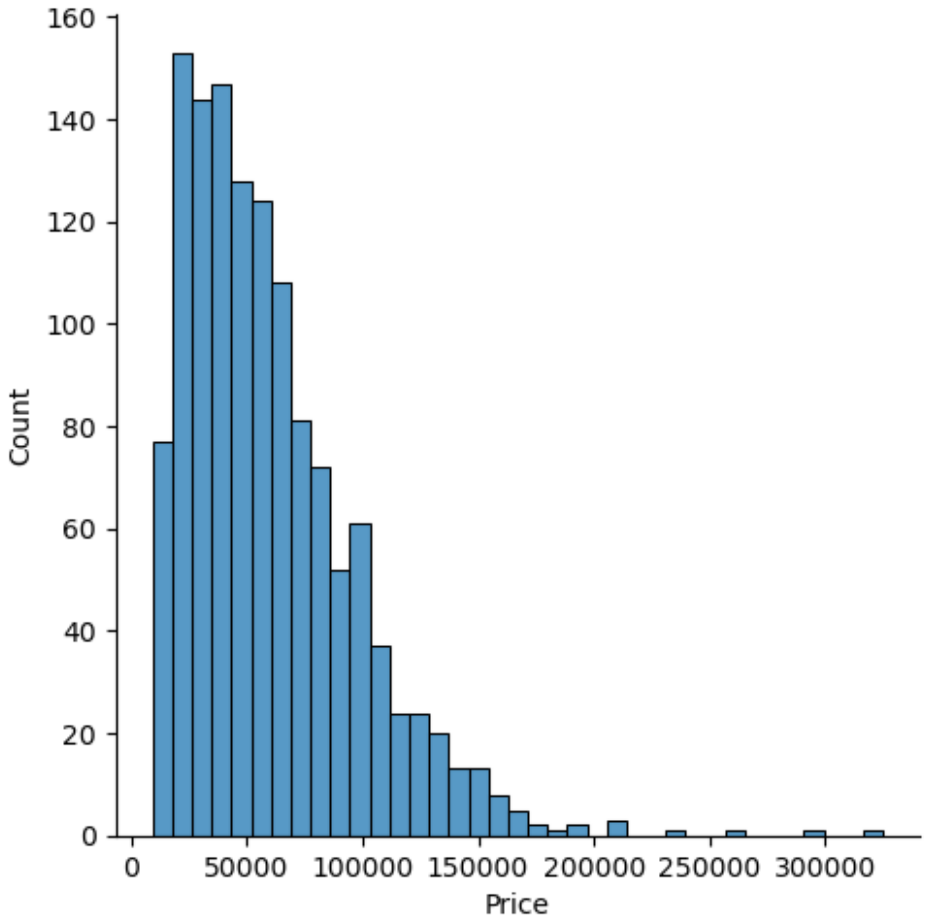
```
 3   ScreenResolution  1303 non-null   object
 4   Cpu               1303 non-null   object
 5   Ram               1303 non-null   int32
 6   Memory            1303 non-null   object
 7   Gpu               1303 non-null   object
 8   OpSys             1303 non-null   object
 9   Weight            1303 non-null   float32
 10  Price             1303 non-null   float64
dtypes: float32(1), float64(2), int32(1), object(7)
memory usage: 101.9+ KB
```

DATA ANALYSIS

Univariate Analysis

```
import seaborn as sns
import matplotlib.pyplot as plt


sns.displot(df['Price'])
```

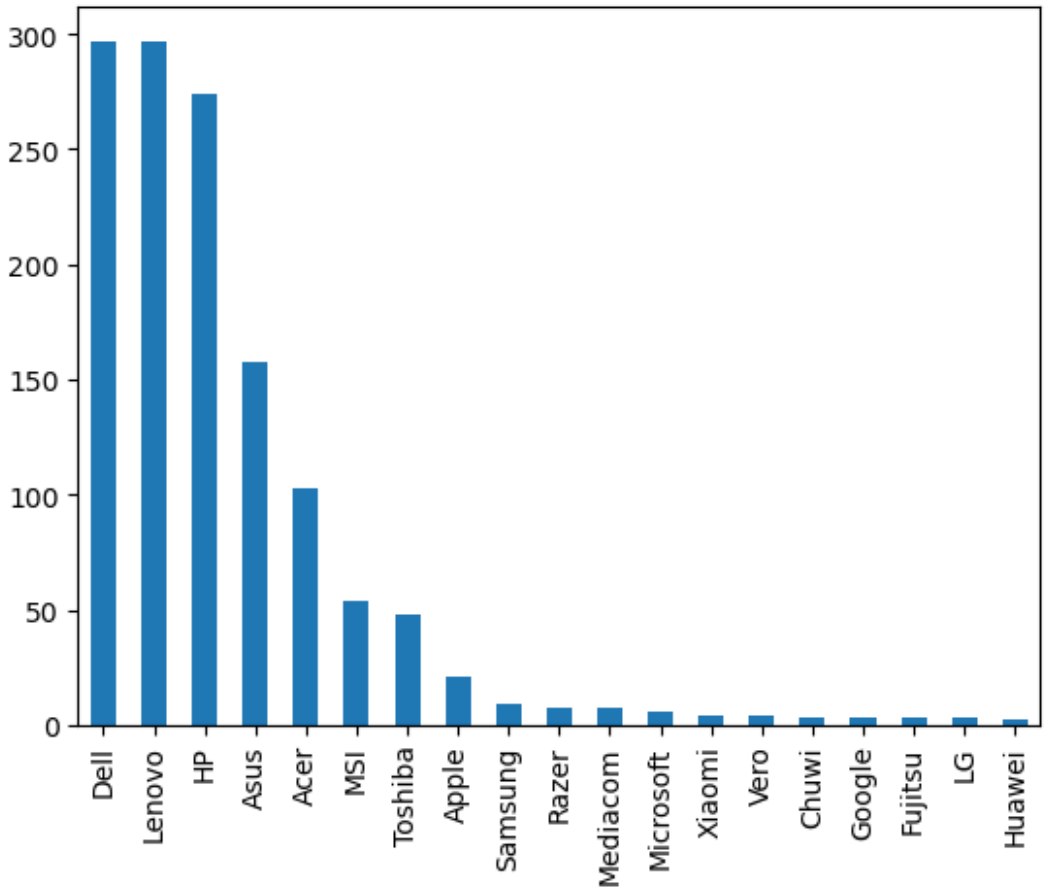
```
<seaborn.axisgrid.FacetGrid at 0x2b567d7df90>
```



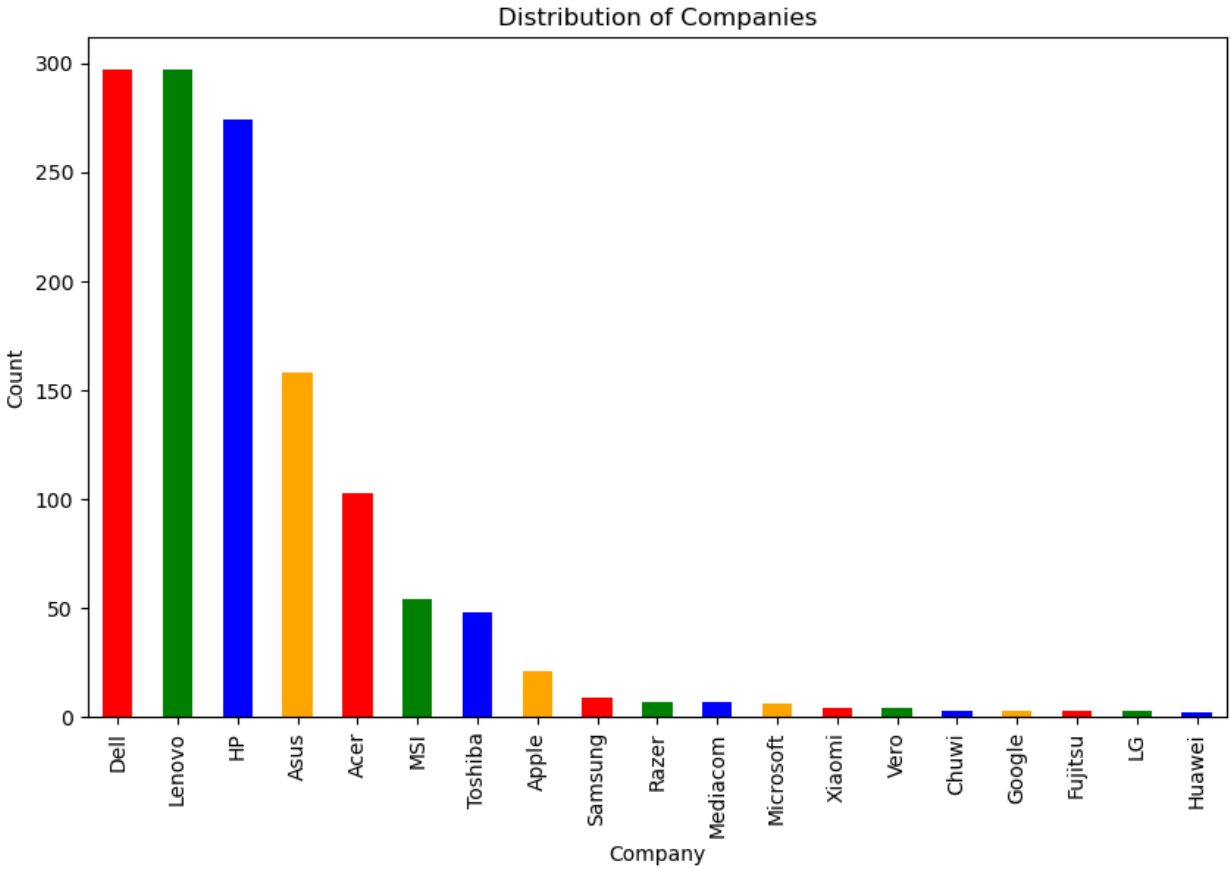
Here we can see that many laptop are of less price and few are expensive .

```
df['Company'].value_counts().plot(kind= 'bar')
```

```
<Axes: >
```



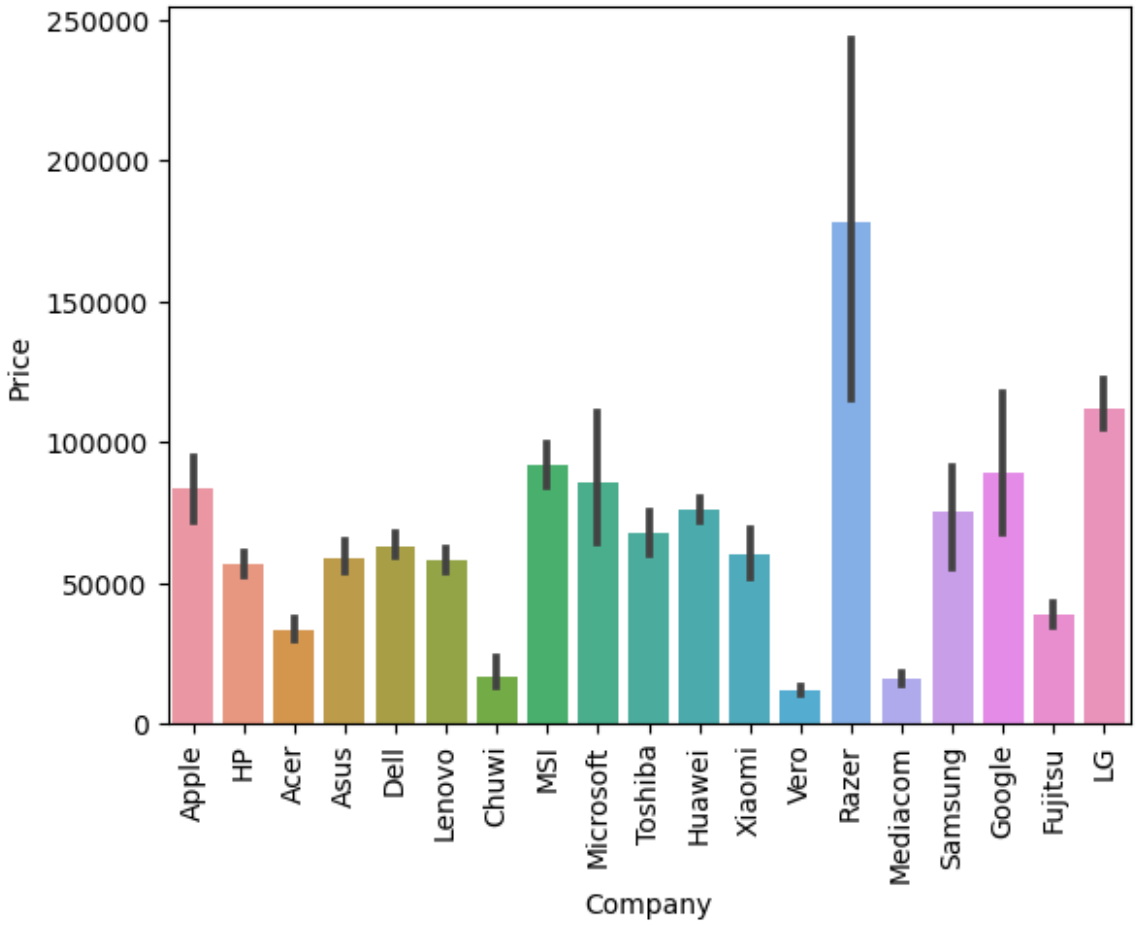

```
plt.figure(figsize=(10, 6))

df['Company'].value_counts().plot(kind='bar', color=['red', 'green',
'blue', 'orange'])
plt.title('Distribution of Companies')

plt.xlabel('Company')
plt.ylabel('Count')
plt.show()
```



Here there are many laptop avaliable of 'Dell' ,'Lenevo' ,'Hp' ,'Acer' ,'Msi' , rest laptop brand not so variety of laptop is available

Average price of each of the brand

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

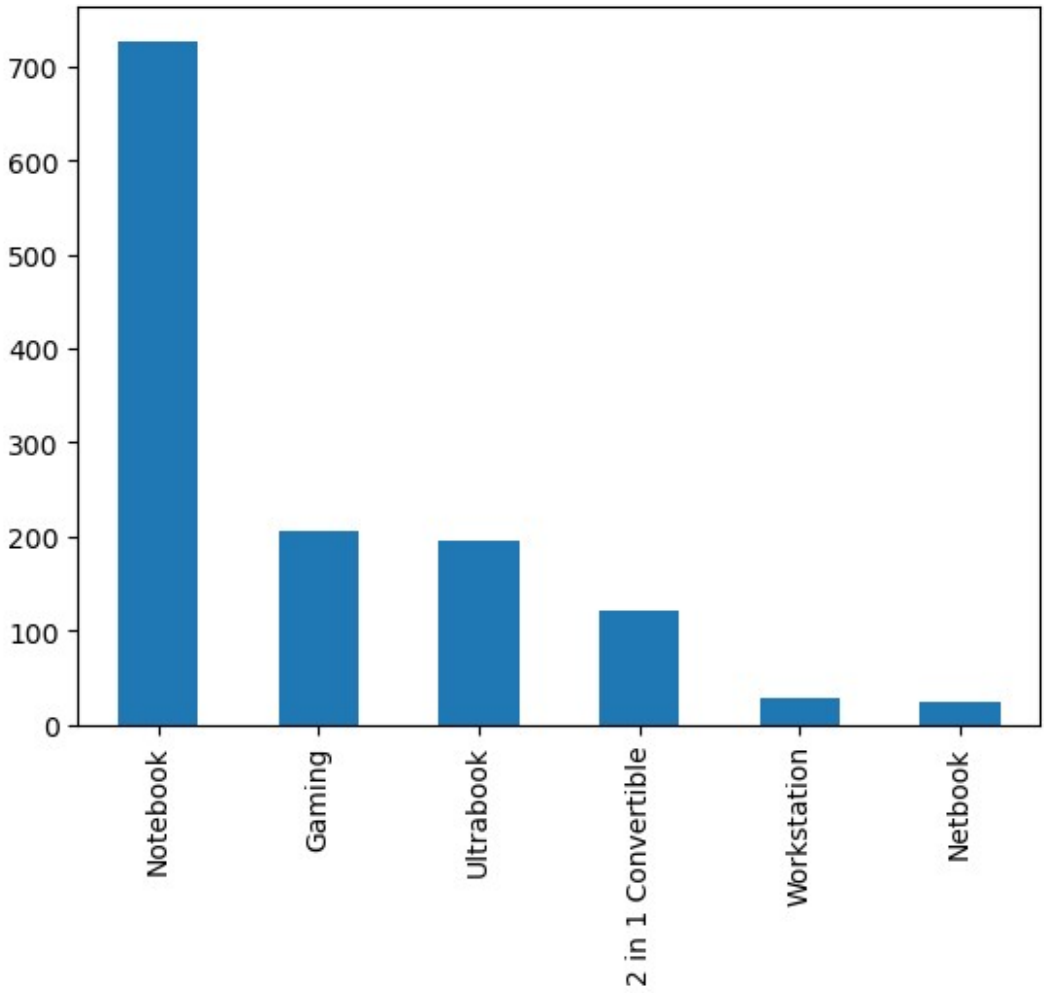


Here we can see that price of the laptop varies according to the brand of the laptop. Like the most expensive laptop is of 'Razor', than 'LG' like wise we can predict the price by seeing the brand as well.

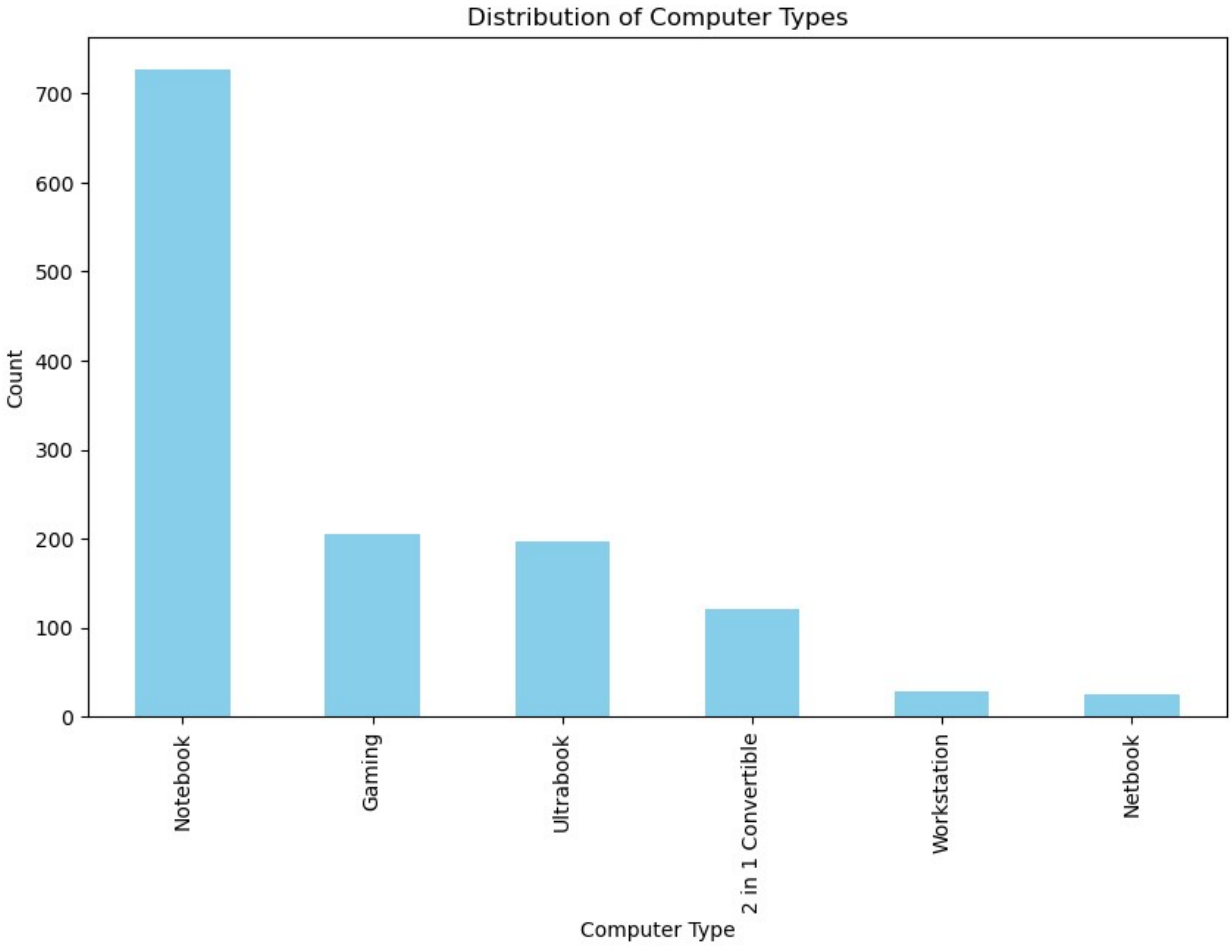
Let see how many types of laptop are available .

```
df['TypeName'].value_counts().plot(kind='bar')
```

```
<Axes: >
```



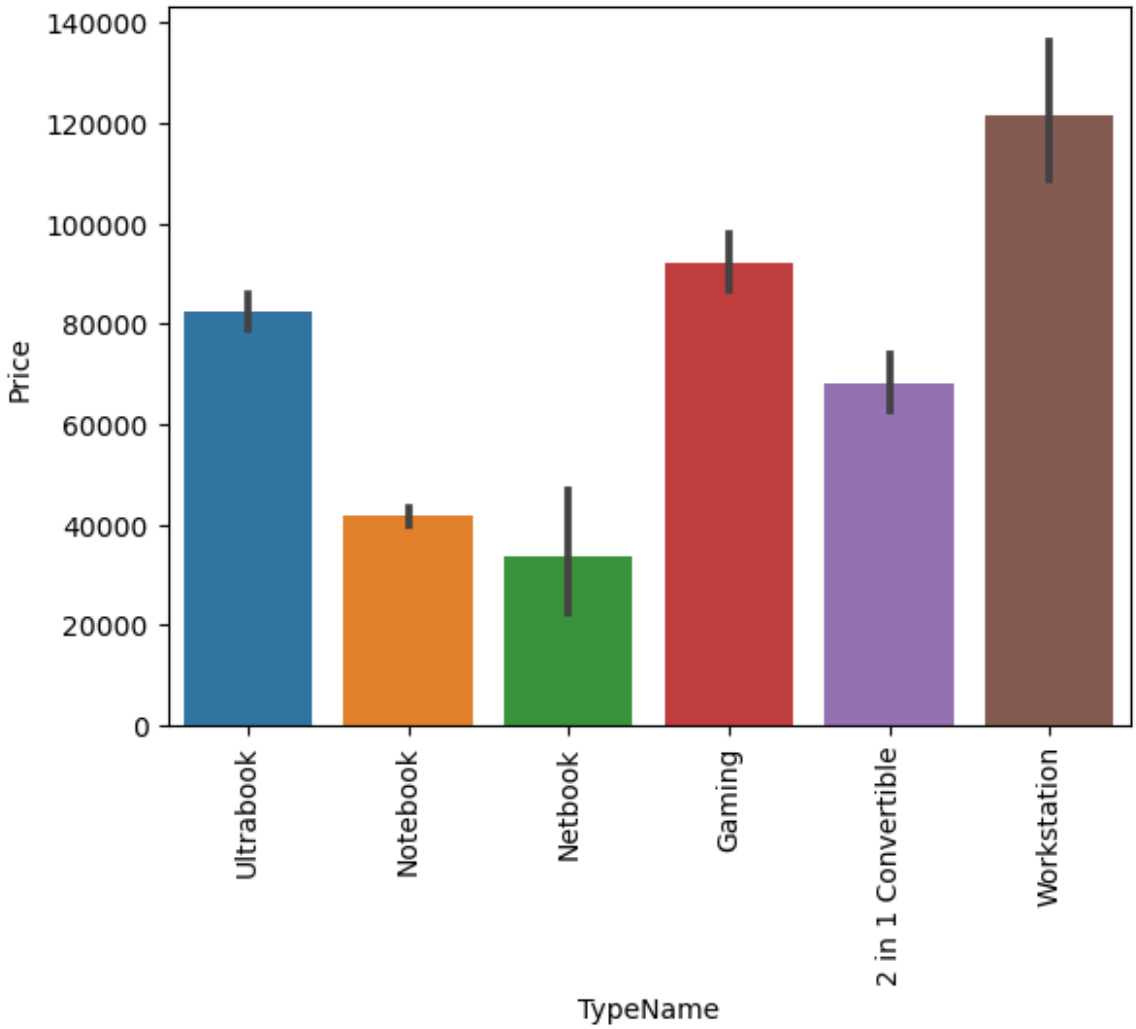

```
plt.figure(figsize=(10, 6))
df['TypeName'].value_counts().plot(kind='bar', color='skyblue')
plt.title('Distribution of Computer Types')
plt.xlabel('Computer Type')
plt.ylabel('Count')
plt.show()
```



Here it can be seen that the notebook type of laptop is mainly produce as the demand is high and netbook type has the least demand

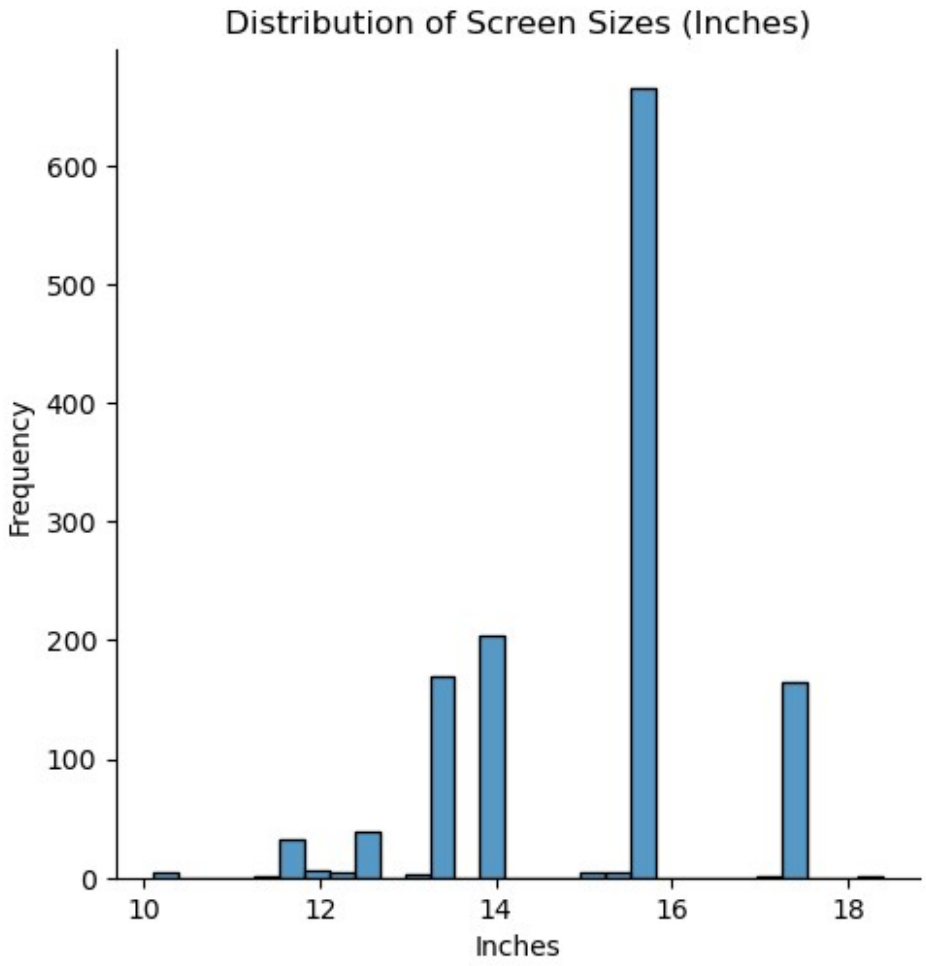
Let see the price of each type of laptop .

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



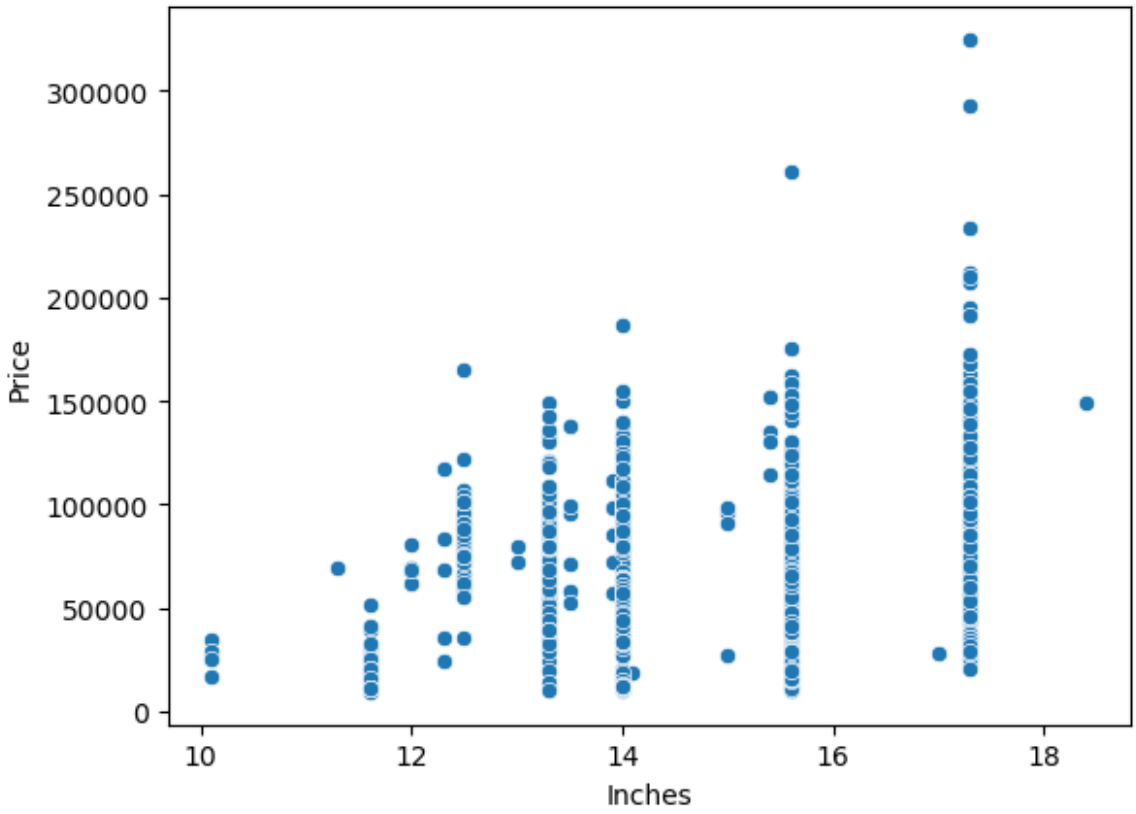
Size

```
sns.displot(df['Inches'])
plt.title('Distribution of Screen Sizes (Inches)')
plt.xlabel('Inches')
plt.ylabel('Frequency')
plt.show()
```




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

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



we can see that for 17.6 inches the price is increasing than any other size of laptop

Now , looking for screen resolution :--

```
df['ScreenResolution'].value_counts()
```

```
Full HD 1920x1080                                   507
1366x768                                            281
IPS Panel Full HD 1920x1080                         230
IPS Panel Full HD / Touchscreen 1920x1080            53
Full HD / Touchscreen 1920x1080                      47
1600x900                                             23
Touchscreen 1366x768                                 16
Quad HD+ / Touchscreen 3200x1800                     15
IPS Panel 4K Ultra HD 3840x2160                      12
IPS Panel 4K Ultra HD / Touchscreen 3840x2160        11
4K Ultra HD / Touchscreen 3840x2160                  10
4K Ultra HD 3840x2160                                 7
Touchscreen 2560x1440                                 7
```

```
IPS Panel 1366x768                                    7
IPS Panel Quad HD+ / Touchscreen 3200x1800            6
IPS Panel Retina Display 2560x1600                    6
IPS Panel Retina Display 2304x1440                    6
Touchscreen 2256x1504                                 6
IPS Panel Touchscreen 2560x1440                       5
IPS Panel Retina Display 2880x1800                    4
IPS Panel Touchscreen 1920x1200                       4
1440x900                                              4
IPS Panel 2560x1440                                   4
IPS Panel Quad HD+ 2560x1440                          3
Quad HD+ 3200x1800                                    3
1920x1080                                             3
Touchscreen 2400x1600                                 3
2560x1440                                             3
IPS Panel Touchscreen 1366x768                        3
IPS Panel Touchscreen / 4K Ultra HD 3840x2160         2
IPS Panel Full HD 2160x1440                           2
IPS Panel Quad HD+ 3200x1800                          2
IPS Panel Retina Display 2736x1824                    1
IPS Panel Full HD 1920x1200                           1
IPS Panel Full HD 2560x1440                           1
IPS Panel Full HD 1366x768                            1
Touchscreen / Full HD 1920x1080                       1
Touchscreen / Quad HD+ 3200x1800                      1
Touchscreen / 4K Ultra HD 3840x2160                   1
IPS Panel Touchscreen 2400x1600                       1
Name: ScreenResolution, dtype: int64
```

Here, we are getting the resolution , IPS panel information , Touch screen information

```
df['Touchscreen'] = df['ScreenResolution'].apply(lambda  x:1 if
'Touchscreen' in x else 0 )

df.sample(5)
```

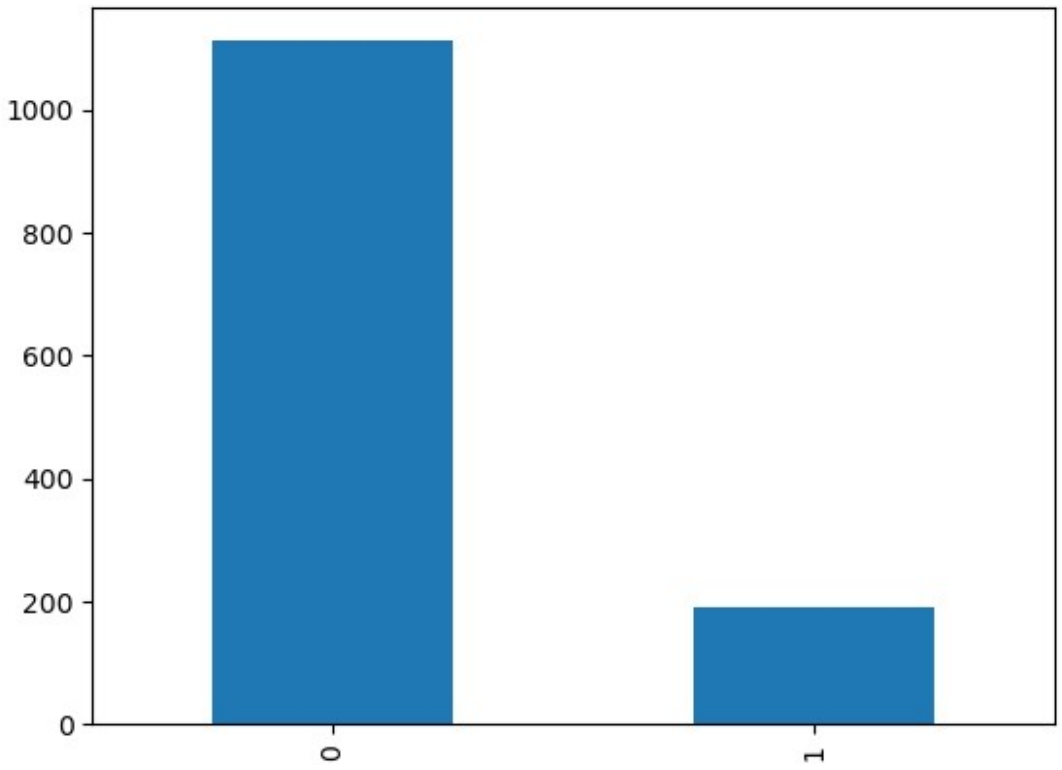
```
     Company  TypeName  Inches             ScreenResolution  \
1177  Lenovo    Gaming    15.6  IPS Panel Full HD 1920x1080
228     Asus  Notebook    17.3            Full HD 1920x1080
525   Lenovo  Notebook    14.0            Full HD 1920x1080
1269    Asus  Notebook    15.6                    1366x768
601       HP  Notebook    15.6                    1366x768

                                Cpu  Ram                Memory  \
1177  Intel Core i7 6700HQ 2.6GHz   16             512GB SSD
228    Intel Core i5 7200U 2.5GHz    8  256GB SSD +  500GB HDD
```

```
525    Intel Core i5 6200U 2.3GHz    8                256GB SSD
1269   Intel Core i7 6500U 2.5GHz    4                500GB HDD
601    Intel Core i3 7100U 2.4GHz    4                500GB HDD

                            Gpu       OpSys  Weight       Price
Touchscreen
1177   Nvidia GeForce GTX 960  Windows 10    3.31  69530.4000
0
228   Nvidia GeForce GTX 950M  Windows 10    2.69  47472.4800
0
525     Intel HD Graphics 520   Windows 7    2.02  71395.2000
0
1269      Nvidia GeForce 920M  Windows 10    2.20  38378.6496
0
601     Intel HD Graphics 620  Windows 10    2.10  35616.6144
0
```

A seperate column Touchscreen is added to differentiate the screen resolution

```
df['Touchscreen'].value_counts().plot(kind = 'bar')
```

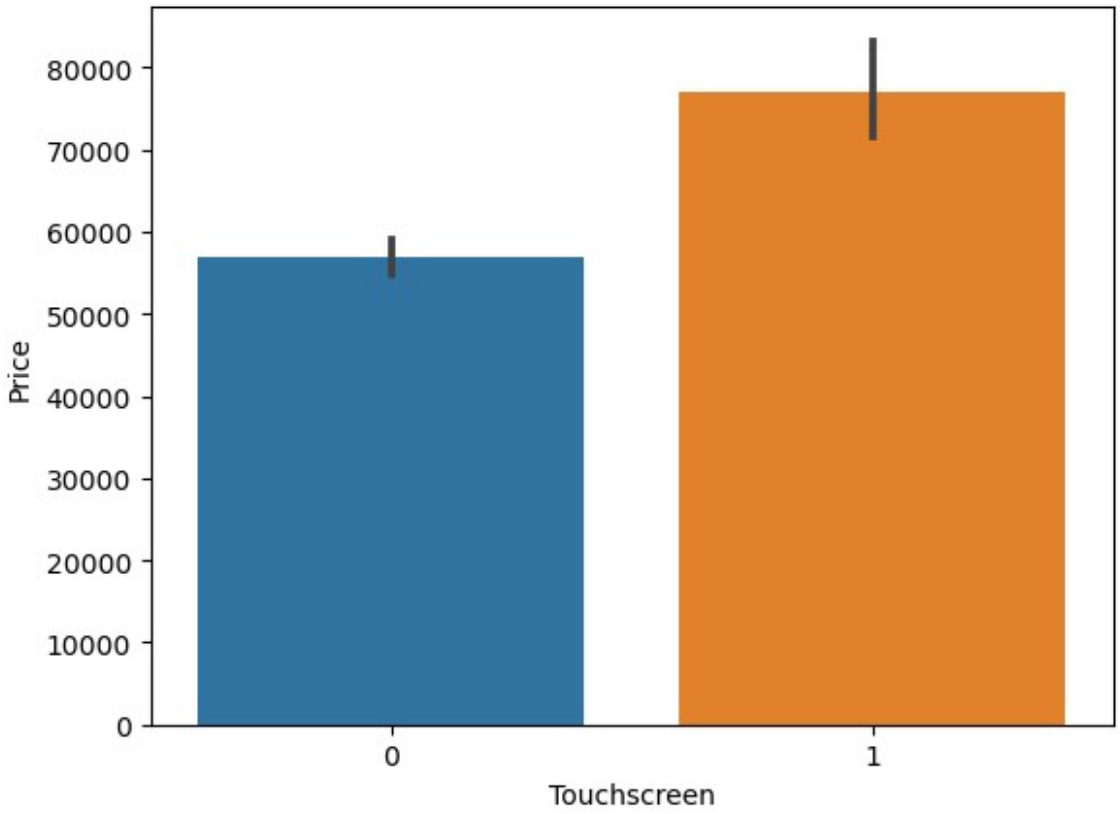
```
<Axes: >
```



Here we are seeing that the count of non - touchscreen laptop are more than the touchscreen laptop .

```
sns.barplot(x= df['Touchscreen'], y = df['Price'])
```

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



Here it is seen that the price of touchscreen laptop is more than the laptop without touchscreen.

```
df['Ips'] = df['ScreenResolution'].apply(lambda  x:1 if 'IPS' in x
else 0 )

df.sample(6)
```

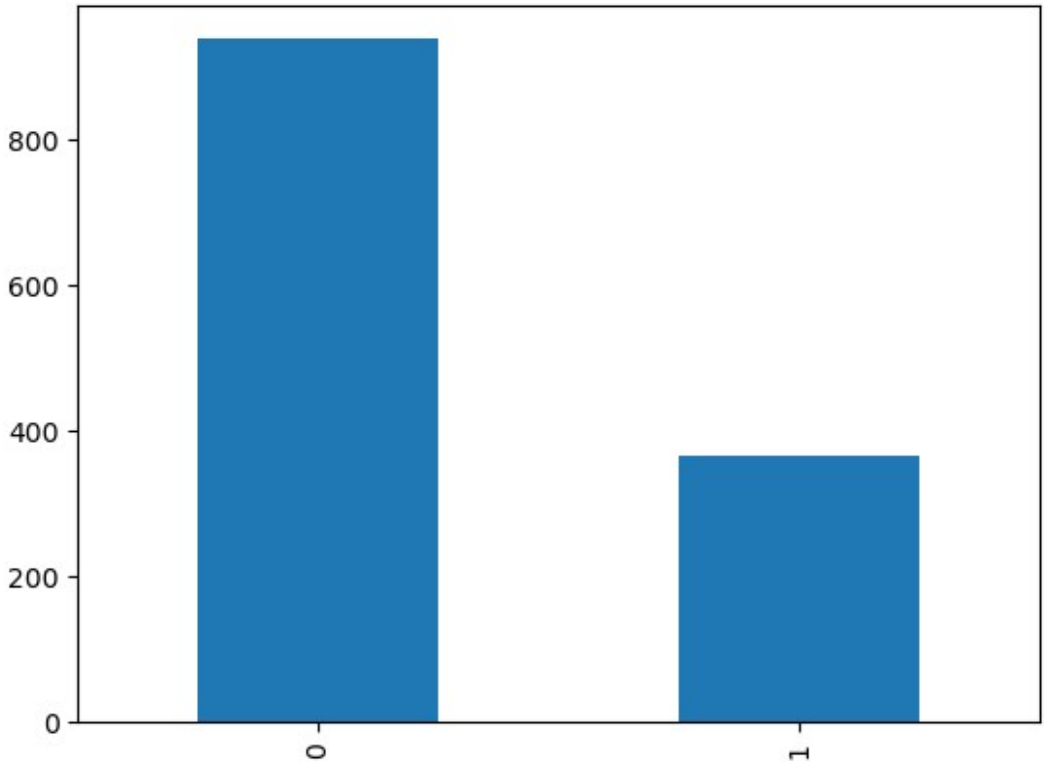
```
     Company            TypeName  Inches
ScreenResolution  \
```

```
283   Lenovo            Notebook    15.6                    Full HD
1920x1080
703   Lenovo            Notebook    15.6                    Full HD
1920x1080
1138     MSI              Gaming    17.3                    Full HD
1920x1080
12     Apple           Ultrabook    15.4  IPS Panel Retina Display
2880x1800
793   Lenovo  2 in 1 Convertible    15.6     Full HD / Touchscreen
1920x1080
687     Dell            Notebook    15.6
1366x768

                                Cpu  Ram               Memory  \
283    Intel Core i5 7200U 2.5GHz    6             256GB SSD
703    Intel Core i5 7200U 2.5GHz    4    1TB HDD +  1TB HDD
1138  Intel Core i7 6820HQ 2.7GHz   16  256GB SSD +  1TB HDD
12           Intel Core i7 2.8GHz   16             256GB SSD
793    Intel Core i5 7200U 2.5GHz    8             256GB SSD
687      Intel Core i3 6006U 2GHz    4               1TB HDD

                           Gpu       OpSys  Weight        Price
Touchscreen  \
283     Intel HD Graphics 620  Windows 10    2.20   30849.1200
0
703     Intel HD Graphics 620  Windows 10    2.10   33110.8560
0
1138  Nvidia GeForce GTX 980M  Windows 10    3.78  127818.7200
0
12         AMD Radeon Pro 555       macOS    1.83  130001.6016
0
793        AMD Radeon R7 M460  Windows 10    2.08   51095.5200
1
687        AMD Radeon R5 M430  Windows 10    2.20   29073.2976
0

      Ips
283     0
703     0
1138    0
12      1
793     0
687     0
```

We excluded the IPS from the screen resolution , here we can see which laptop has IPS or not

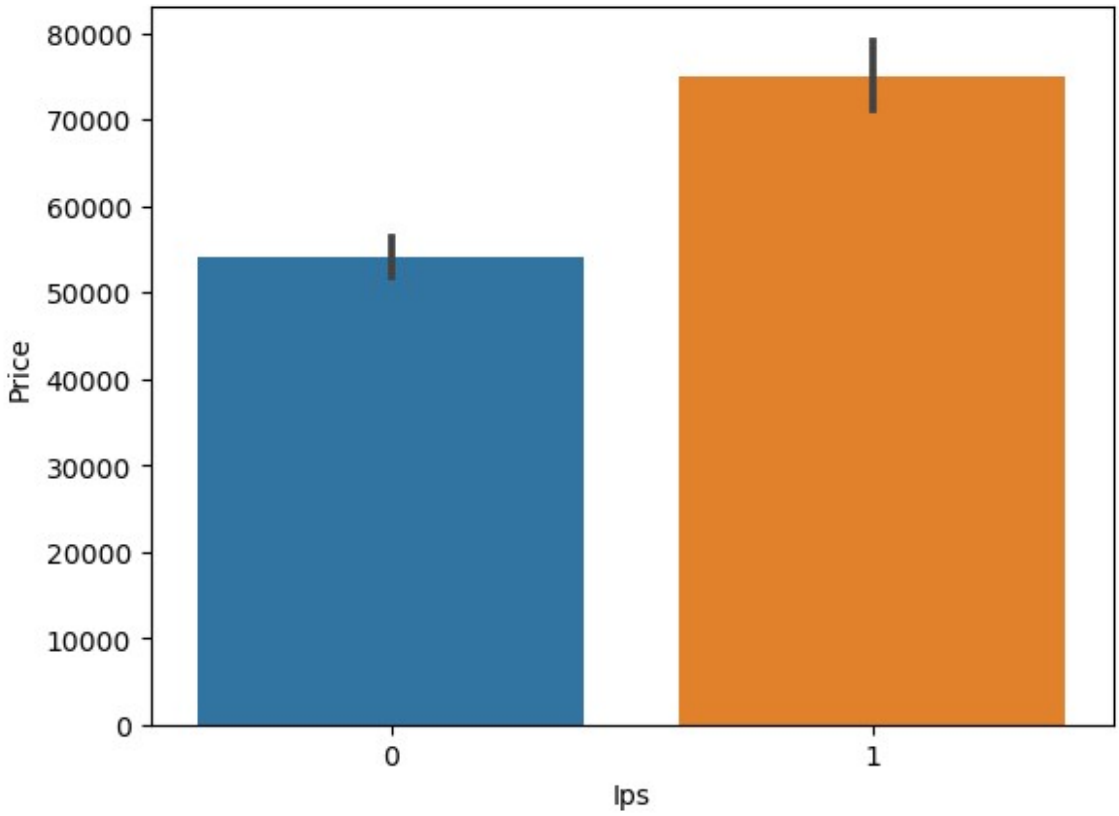
```
df['Ips'].value_counts().plot(kind = 'bar')
```

```
<Axes: >
```




```
sns.barplot(x= df['Ips'], y = df['Price'])
```

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



Now dividing the resolution

```
df['ScreenResolution'].str.split('x',n=1,expand=True)
```

```
                                          0     1
0              IPS Panel Retina Display 2560  1600
1                                       1440   900
2                                 Full HD 1920  1080
3              IPS Panel Retina Display 2880  1800
4              IPS Panel Retina Display 2560  1600
...                                     ...   ...
1298   IPS Panel Full HD / Touchscreen 1920  1080
1299  IPS Panel Quad HD+ / Touchscreen 3200  1800
1300                                    1366   768
1301                                    1366   768
1302                                    1366   768

[1303 rows x 2 columns]
```

```
new = df['ScreenResolution'].str.split('x',n=1,expand=True)
df['X_res']= new[0]
df['Y_res'] = new[1]

df.head()
```

```
  Company   TypeName  Inches                    ScreenResolution  \
0   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600
1   Apple  Ultrabook    13.3                            1440x900
2      HP   Notebook    15.6                   Full HD 1920x1080
3   Apple  Ultrabook    15.4  IPS Panel Retina Display 2880x1800
4   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600

                          Cpu  Ram               Memory  \
0        Intel Core i5 2.3GHz    8            128GB SSD
1        Intel Core i5 1.8GHz    8  128GB Flash Storage
2  Intel Core i5 7200U 2.5GHz    8            256GB SSD
3        Intel Core i7 2.7GHz   16            512GB SSD
4        Intel Core i5 3.1GHz    8            256GB SSD

                            Gpu  OpSys  Weight        Price
Touchscreen  Ips  \
0  Intel Iris Plus Graphics 640  macOS    1.37   71378.6832
0    1
1        Intel HD Graphics 6000  macOS    1.34   47895.5232
0    0
2         Intel HD Graphics 620  No OS    1.86   30636.0000
0    0
3            AMD Radeon Pro 455  macOS    1.83  135195.3360
0    1
4  Intel Iris Plus Graphics 650  macOS    1.37   96095.8080
0    1

                           X_res Y_res
0  IPS Panel Retina Display 2560  1600
1                           1440   900
2                    Full HD 1920  1080
3  IPS Panel Retina Display 2880  1800
4  IPS Panel Retina Display 2560  1600
```

```
df['X_res'] = df['X_res'].str.replace(',','').str.findall(r'(\d+\.?\
d+)').apply(lambda x:x[0])

df.head()
```

```
  Company   TypeName  Inches                    ScreenResolution  \
0   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600
1   Apple  Ultrabook    13.3                            1440x900
2      HP   Notebook    15.6                   Full HD 1920x1080
3   Apple  Ultrabook    15.4  IPS Panel Retina Display 2880x1800
4   Apple  Ultrabook    13.3  IPS Panel Retina Display 2560x1600

                          Cpu  Ram               Memory  \
0        Intel Core i5 2.3GHz    8            128GB SSD
1        Intel Core i5 1.8GHz    8  128GB Flash Storage
2  Intel Core i5 7200U 2.5GHz    8            256GB SSD
```

```
3        Intel Core i7 2.7GHz   16            512GB SSD
4        Intel Core i5 3.1GHz    8            256GB SSD

                            Gpu  OpSys  Weight        Price
Touchscreen  Ips  \
0  Intel Iris Plus Graphics 640  macOS    1.37   71378.6832
0    1
1        Intel HD Graphics 6000  macOS    1.34   47895.5232
0    0
2         Intel HD Graphics 620  No OS    1.86   30636.0000
0    0
3            AMD Radeon Pro 455  macOS    1.83  135195.3360
0    1
4  Intel Iris Plus Graphics 650  macOS    1.37   96095.8080
0    1

  X_res Y_res
0  2560  1600
1  1440   900
2  1920  1080
3  2880  1800
4  2560  1600
```

```
df['X_res'] = df['X_res'].astype('int')
df['Y_res'] = df['Y_res'].astype('int')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 15 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   int32
 6   Memory            1303 non-null   object
 7   Gpu               1303 non-null   object
 8   OpSys             1303 non-null   object
 9   Weight            1303 non-null   float32
 10  Price             1303 non-null   float64
 11  Touchscreen       1303 non-null   int64
 12  Ips               1303 non-null   int64
 13  X_res             1303 non-null   int32
 14  Y_res             1303 non-null   int32
dtypes: float32(1), float64(2), int32(3), int64(2), object(7)
memory usage: 132.5+ KB
```

```
df.corr()['Price']
```

```
C:\Users\sanya\AppData\Local\Temp\ipykernel_14776\815546952.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only
valid columns or specify the value of numeric_only to silence this
warning.
  df.corr()['Price']
```

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

The corelation of price with other variable , RAM , X_res, Y_res are highly corelated to price

```
#PPI = Pixel Per Inches

df['ppi'] =   (((df['X_res']**2) +
(df['Y_res']**2))**0.5/df['Inches']).astype('float')

df.corr()['Price']
```

```
C:\Users\sanya\AppData\Local\Temp\ipykernel_14776\815546952.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only
valid columns or specify the value of numeric_only to silence this
warning.
  df.corr()['Price']
```

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

As the column Screen Resolution is divided into valuable columns - X-res, Y-res, ppi . So now drop the Screen Resolution column as it is not of major use .

```
df.drop(columns=['ScreenResolution'],inplace=True)

df.head()
```

```
  Company   TypeName  Inches                         Cpu  Ram  \
0   Apple  Ultrabook    13.3        Intel Core i5 2.3GHz    8
1   Apple  Ultrabook    13.3        Intel Core i5 1.8GHz    8
2      HP   Notebook    15.6  Intel Core i5 7200U 2.5GHz    8
3   Apple  Ultrabook    15.4        Intel Core i7 2.7GHz   16
4   Apple  Ultrabook    13.3        Intel Core i5 3.1GHz    8

                Memory                           Gpu  OpSys  Weight  \
0            128GB SSD  Intel Iris Plus Graphics 640  macOS    1.37
1  128GB Flash Storage        Intel HD Graphics 6000  macOS    1.34
2            256GB SSD         Intel HD Graphics 620  No OS    1.86
3            512GB SSD            AMD Radeon Pro 455  macOS    1.83
4            256GB SSD  Intel Iris Plus Graphics 650  macOS    1.37

         Price  Touchscreen  Ips  X_res  Y_res         ppi
0   71378.6832            0    1   2560   1600  226.983005
1   47895.5232            0    0   1440    900  127.677940
2   30636.0000            0    0   1920   1080  141.211998
3  135195.3360            0    1   2880   1800  220.534624
4   96095.8080            0    1   2560   1600  226.983005
```

Now we have the ppi , so there is no use of screen resolution , x-res , y--res , inches

```
df.drop(columns=['Inches','X_res','Y_res'],inplace=True)

df.head()
```

```
  Company   TypeName                         Cpu  Ram
Memory  \
0   Apple  Ultrabook        Intel Core i5 2.3GHz    8            128GB
SSD
1   Apple  Ultrabook        Intel Core i5 1.8GHz    8  128GB Flash
Storage
2      HP   Notebook  Intel Core i5 7200U 2.5GHz    8            256GB
```

```
SSD
3   Apple  Ultrabook        Intel Core i7 2.7GHz   16            512GB
SSD
4   Apple  Ultrabook        Intel Core i5 3.1GHz    8            256GB
SSD

                            Gpu  OpSys  Weight        Price
Touchscreen  Ips  \
0  Intel Iris Plus Graphics 640  macOS    1.37   71378.6832
0    1
1        Intel HD Graphics 6000  macOS    1.34   47895.5232
0    0
2         Intel HD Graphics 620  No OS    1.86   30636.0000
0    0
3            AMD Radeon Pro 455  macOS    1.83  135195.3360
0    1
4  Intel Iris Plus Graphics 650  macOS    1.37   96095.8080
0    1

           ppi
0  226.983005
1  127.677940
2  141.211998
3  220.534624
4  226.983005
```

Now concentrating on CPU column, here also we have to divide the data with respect to processor and GHz

```
df['Cpu'].value_counts()

Intel Core i5 7200U 2.5GHz       190
Intel Core i7 7700HQ 2.8GHz      146
Intel Core i7 7500U 2.7GHz       134
Intel Core i7 8550U 1.8GHz        73
Intel Core i5 8250U 1.6GHz        72
                                ...
Intel Core M M3-6Y30 0.9GHz        1
AMD A9-Series 9420 2.9GHz          1
Intel Core i3 6006U 2.2GHz         1
AMD A6-Series 7310 2GHz            1
Intel Xeon E3-1535M v6 3.1GHz      1
Name: Cpu, Length: 118, dtype: int64

df['Cpu Name'] =df['Cpu'].apply(lambda x:" ".join(x.split()[0:3]))
```

```
df.head()
```

```
  Company   TypeName                           Cpu  Ram
Memory  \
0   Apple  Ultrabook        Intel Core i5 2.3GHz    8            128GB
SSD
1   Apple  Ultrabook        Intel Core i5 1.8GHz    8  128GB Flash
Storage
2      HP   Notebook  Intel Core i5 7200U 2.5GHz    8            256GB
SSD
3   Apple  Ultrabook        Intel Core i7 2.7GHz   16            512GB
SSD
4   Apple  Ultrabook        Intel Core i5 3.1GHz    8            256GB
SSD

                             Gpu  OpSys  Weight        Price
Touchscreen  Ips  \
0  Intel Iris Plus Graphics 640  macOS    1.37   71378.6832
0    1
1        Intel HD Graphics 6000  macOS    1.34   47895.5232
0    0
2         Intel HD Graphics 620  No OS    1.86   30636.0000
0    0
3            AMD Radeon Pro 455  macOS    1.83  135195.3360
0    1
4  Intel Iris Plus Graphics 650  macOS    1.37   96095.8080
0    1

          ppi       Cpu Name
0  226.983005  Intel Core i5
1  127.677940  Intel Core i5
2  141.211998  Intel Core i5
3  220.534624  Intel Core i7
4  226.983005  Intel Core i5
```

Here we divided the processer from cpu to cpu name .

```
def fetch_processor(text):
    if text == 'Intel Core i3'  or text == 'Intel Core i5' or text ==
'Intel Core i7':
        return text
    else:
        if text.split()[0] == 'Intel':
            return 'Other Intel Processor'
        else:
            return 'AMD Processor '
```

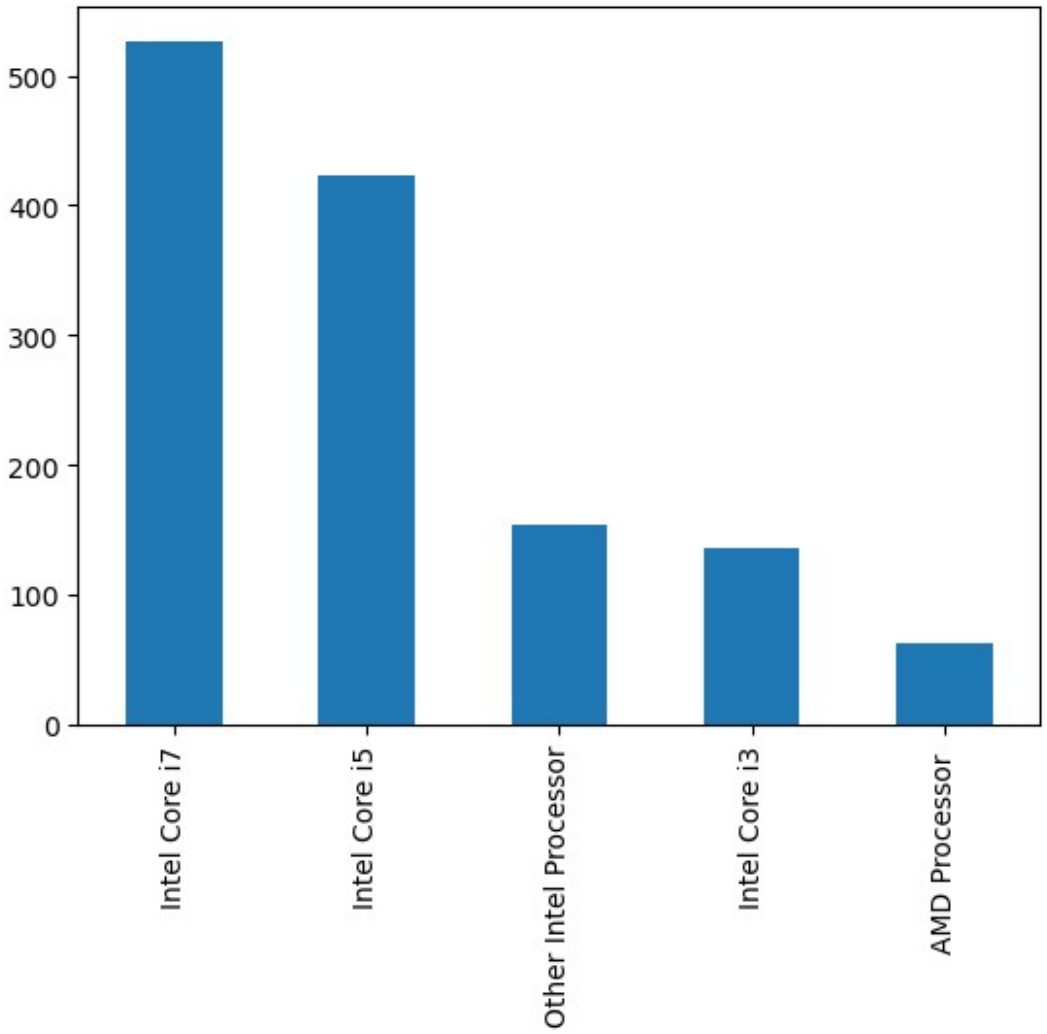
```
df['Cpu brand'] = df['Cpu Name'].apply(fetch_processor)

df.head()
```

```
  Company   TypeName                          Cpu  Ram
Memory  \
0   Apple  Ultrabook        Intel Core i5 2.3GHz    8            128GB
SSD
1   Apple  Ultrabook        Intel Core i5 1.8GHz    8  128GB Flash
Storage
2      HP   Notebook  Intel Core i5 7200U 2.5GHz    8            256GB
SSD
3   Apple  Ultrabook        Intel Core i7 2.7GHz   16            512GB
SSD
4   Apple  Ultrabook        Intel Core i5 3.1GHz    8            256GB
SSD

                            Gpu  OpSys  Weight        Price
Touchscreen  Ips  \
0  Intel Iris Plus Graphics 640  macOS    1.37   71378.6832
0    1
1        Intel HD Graphics 6000  macOS    1.34   47895.5232
0    0
2         Intel HD Graphics 620  No OS    1.86   30636.0000
0    0
3            AMD Radeon Pro 455  macOS    1.83  135195.3360
0    1
4  Intel Iris Plus Graphics 650  macOS    1.37   96095.8080
0    1

           ppi       Cpu Name      Cpu brand
0  226.983005  Intel Core i5  Intel Core i5
1  127.677940  Intel Core i5  Intel Core i5
2  141.211998  Intel Core i5  Intel Core i5
3  220.534624  Intel Core i7  Intel Core i7
4  226.983005  Intel Core i5  Intel Core i5
```

Here we have divided the processor from cpu name column

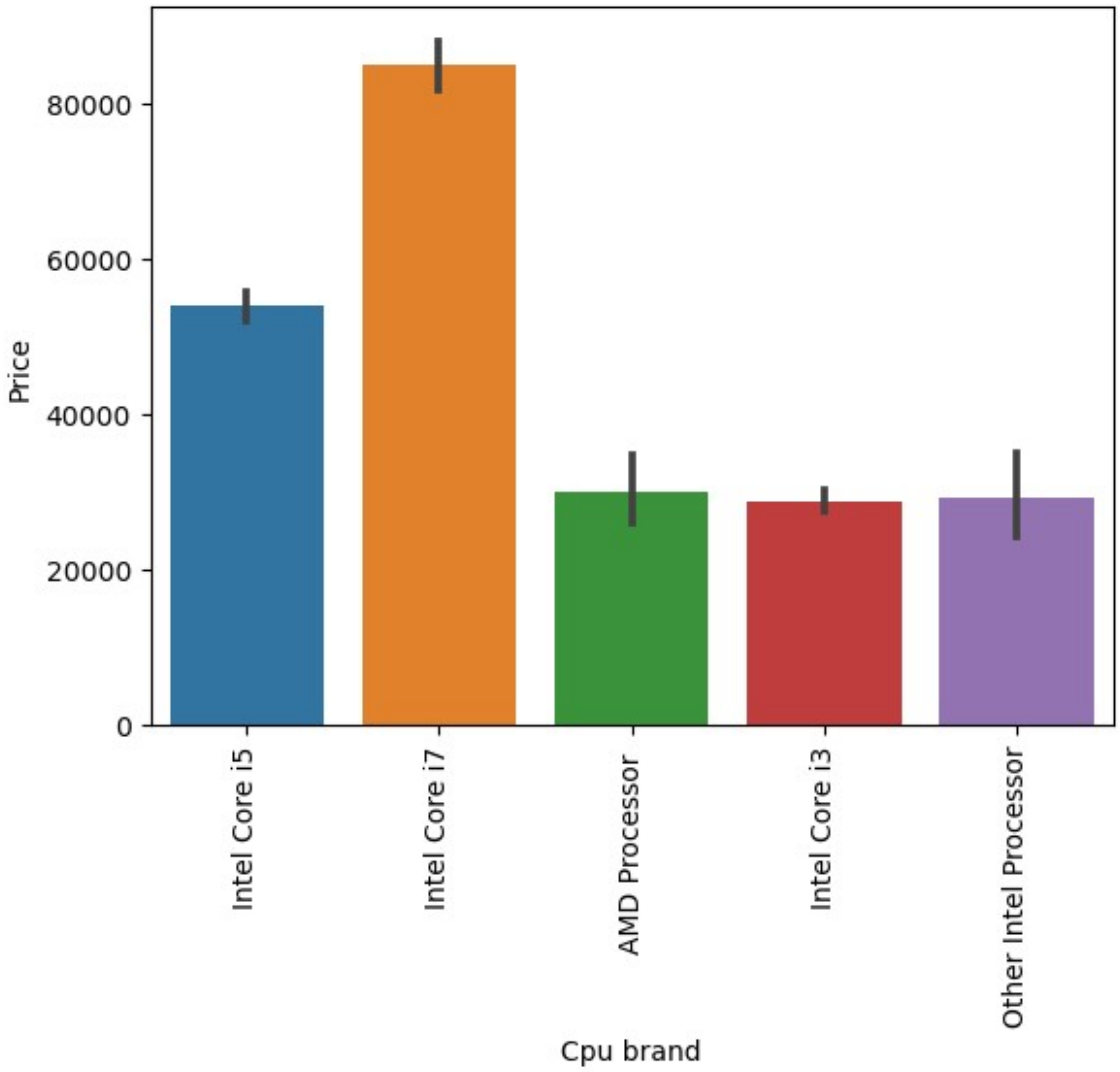
```
df['Cpu brand'].value_counts().plot(kind ='bar')
```

```
<Axes: >
```



Here we can see that the 'i7' is more produced than ' i5' and the least variety of processor in this data is of ' AMD'

```
sns.barplot(x= df['Cpu brand'], y = df['Price'])
plt.xticks(rotation ='vertical')
plt.show()
```



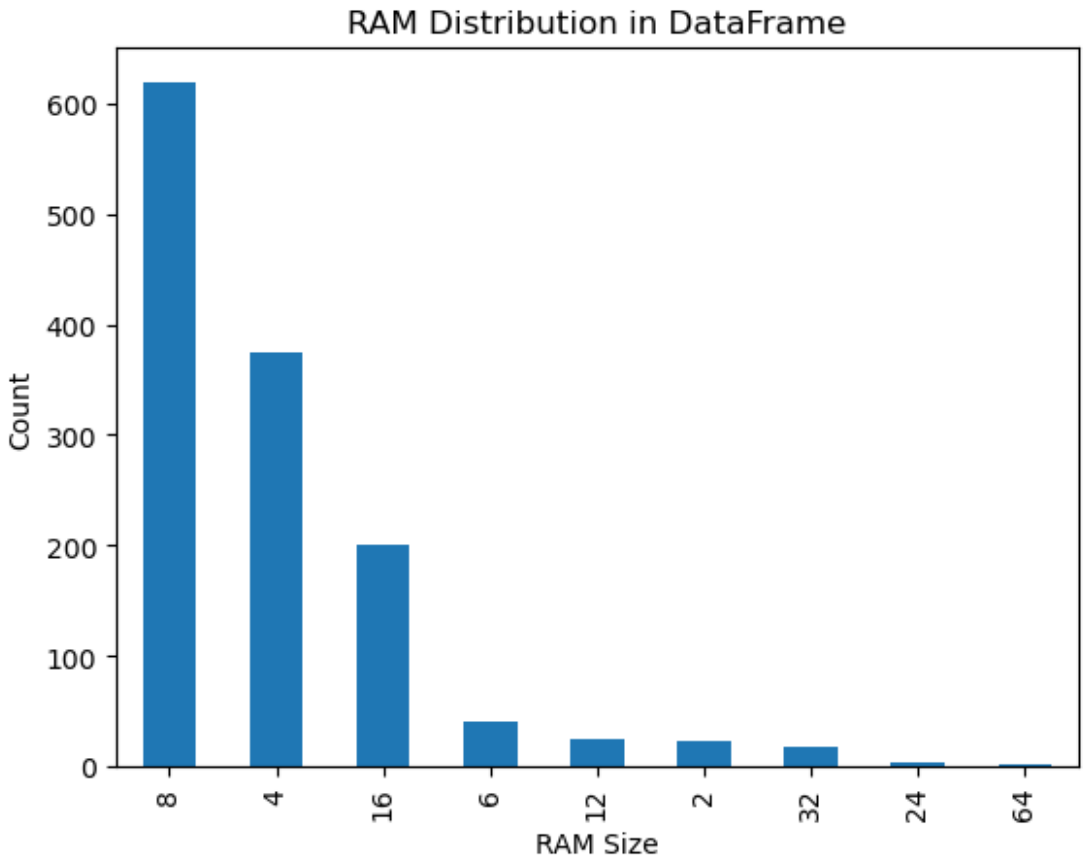
Here we can see that the 'Intel Core i7' is the most expensive processor with a range of above 80,000 and 'Intel Core i5' is in range of 50,000 , where the other processor are in range of 30,000 . So yes we can say that processor is a important factor for laptop price prediction .

```
df.drop(columns=['Cpu','Cpu Name'], inplace=True)

df.head()
```

```
  Company   TypeName  Ram                Memory
Gpu  \
0   Apple  Ultrabook    8            128GB SSD  Intel Iris Plus
Graphics 640
1   Apple  Ultrabook    8  128GB Flash Storage        Intel HD
Graphics 6000
2      HP   Notebook    8            256GB SSD         Intel HD
Graphics 620
3   Apple  Ultrabook   16            512GB SSD            AMD Radeon
Pro 455
4   Apple  Ultrabook    8            256GB SSD  Intel Iris Plus
Graphics 650

   OpSys  Weight        Price  Touchscreen  Ips         ppi      Cpu
brand
0  macOS    1.37   71378.6832            0    1  226.983005  Intel
Core i5
1  macOS    1.34   47895.5232            0    0  127.677940  Intel
Core i5
2  No OS    1.86   30636.0000            0    0  141.211998  Intel
Core i5
3  macOS    1.83  135195.3360            0    1  220.534624  Intel
Core i7
4  macOS    1.37   96095.8080            0    1  226.983005  Intel
Core i5
```

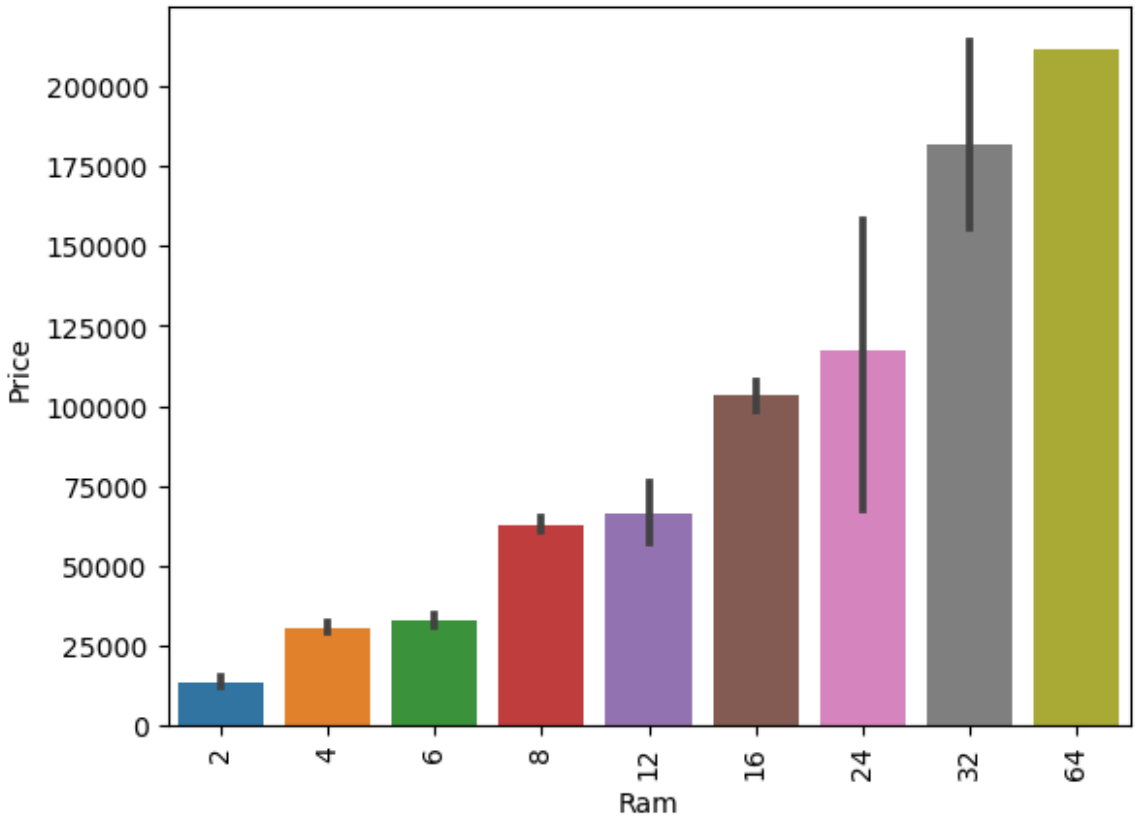
Now , let's focus on RAM

```
df['Ram'].value_counts().plot(kind='bar')
plt.title('RAM Distribution in DataFrame')
plt.xlabel('RAM Size')
plt.ylabel('Count')
plt.show()
```



Here, we can see that the most demanding Ram is of 8GB, suprisely after that is 4GB, than 16GB, so yes RAM is the cruical factor for determining the price of the laptop

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



As the memory size is increasing than the Price for Ram is also increasing .

Let look into the MEMORY impact on Price.

```
df['Memory'].value_counts()
```

```
256GB SSD                        412
1TB HDD                          223
500GB HDD                        132
512GB SSD                        118
128GB SSD +  1TB HDD              94
128GB SSD                         76
256GB SSD +  1TB HDD              73
32GB Flash Storage                38
2TB HDD                           16
64GB Flash Storage                15
512GB SSD +  1TB HDD              14
1TB SSD                           14
256GB SSD +  2TB HDD              10
```

```
1.0TB Hybrid                        9
256GB Flash Storage                 8
16GB Flash Storage                  7
32GB SSD                            6
180GB SSD                           5
128GB Flash Storage                 4
512GB SSD +  2TB HDD                3
16GB SSD                            3
512GB Flash Storage                 2
1TB SSD +  1TB HDD                  2
256GB SSD +  500GB HDD              2
128GB SSD +  2TB HDD                2
256GB SSD +  256GB SSD              2
512GB SSD +  256GB SSD              1
512GB SSD +  512GB SSD              1
64GB Flash Storage +  1TB HDD       1
1TB HDD +  1TB HDD                  1
32GB HDD                            1
64GB SSD                            1
128GB HDD                           1
240GB SSD                           1
8GB SSD                             1
508GB Hybrid                        1
1.0TB HDD                           1
512GB SSD +  1.0TB Hybrid           1
256GB SSD +  1.0TB Hybrid           1
Name: Memory, dtype: int64
```

we can see here that byfurgating memory is bit complex , so we will make 4 new columns for 1- SSD, 2- HDD, 3- Flash Memory , 4-Hybrid

```
df['Memory'] =df['Memory'].astype(str).replace('\.0','',regex=True)
df['Memory'] =df['Memory'].str.replace('GB','')
df['Memory'] =df['Memory'].str.replace('TB','000')
new =df['Memory'].str.split('+',n=1,expand= True)


df['first']=new[0]
df['first']=df['first'].str.strip()


df['second'] =new[1]

df['Layer1HDD']= df['first'].apply(lambda  x: 1 if 'HDD' in x else 0)
df['Layer1SSD']= df['first'].apply(lambda  x:1 if 'SSD' in x else 0)
```

```
df['Layer1Hybrid']= df['first'].apply(lambda  x:1 if 'Hybrid' in x
else 0)
df['Layer1Flash_Storage']= df['first'].apply(lambda  x:1 if 'Flash
Storage' in x else 0)

df['first'] =df['first'].str.replace(r'\D','')

df['second'].fillna('0',inplace = True)

df['Layer2HDD']= df['second'].apply(lambda  x: 1 if 'HDD' in x else 0)
df['Layer2SSD']= df['second'].apply(lambda  x:1 if 'SSD' in x else 0)
df['Layer2Hybrid']= df['second'].apply(lambda  x:1 if 'Hybrid' in x
else 0)
df['Layer2Flash_Storage']= df['second'].apply(lambda  x:1 if 'Flash
Storage' in x else 0)

df['second'] =df['second'].str.replace(r'\D','')

df['first'] =df['first'].astype(int)
df['second'] =df['second'].astype(int)

df['HDD']= (df['first']*df['Layer1HDD']+df['second']*df['Layer2HDD'])
df['SSD']= (df['first']*df['Layer1SSD']+df['second']*df['Layer2SSD'])
df['Hybrid']= (df['first']*df['Layer1Hybrid']
+df['second']*df['Layer2Hybrid'])
df['Flash_Storage']= (df['first']*df['Layer1Flash_Storage']
+df['second']*df['Layer2Flash_Storage'])
```

```
C:\Users\sanya\AppData\Local\Temp\ipykernel_14776\55096629.py:18:
FutureWarning: The default value of regex will change from True to
False in a future version.
  df['first'] =df['first'].str.replace(r'\D','')
C:\Users\sanya\AppData\Local\Temp\ipykernel_14776\55096629.py:27:
FutureWarning: The default value of regex will change from True to
False in a future version.
  df['second'] =df['second'].str.replace(r'\D','')
```

```
df.head()
```

```
  Company   TypeName  Ram              Memory
Gpu  \
0   Apple  Ultrabook    8            128 SSD  Intel Iris Plus Graphics
640
1   Apple  Ultrabook    8  128 Flash Storage        Intel HD Graphics
6000
2      HP   Notebook    8            256 SSD         Intel HD Graphics
620
3   Apple  Ultrabook   16            512 SSD            AMD Radeon Pro
```

```
455
4   Apple  Ultrabook    8           256 SSD  Intel Iris Plus Graphics
650

   OpSys  Weight        Price  Touchscreen  Ips  ...  Layer1Hybrid  \
0  macOS    1.37   71378.6832            0    1  ...             0
1  macOS    1.34   47895.5232            0    0  ...             0
2  No OS    1.86   30636.0000            0    0  ...             0
3  macOS    1.83  135195.3360            0    1  ...             0
4  macOS    1.37   96095.8080            0    1  ...             0

  Layer1Flash_Storage  Layer2HDD  Layer2SSD  Layer2Hybrid  \
0                   0          0          0             0
1                   1          0          0             0
2                   0          0          0             0
3                   0          0          0             0
4                   0          0          0             0

   Layer2Flash_Storage  HDD  SSD  Hybrid  Flash_Storage
0                    0    0  128       0              0
1                    0    0    0       0            128
2                    0    0  256       0              0
3                    0    0  512       0              0
4                    0    0  256       0              0

[5 rows x 26 columns]
```

```
df.drop(columns=['first','second','Layer1HDD','Layer1SSD','Layer1Hybrid','Layer1Flash_Storage','Layer2HDD','Layer2Hybrid','Layer2SSD','Layer2Flash_Storage'],inplace=True)
```

```
df.sample(6)
```

```
     Company   TypeName  Ram               Memory
Gpu  \
1122      HP  Notebook    8            256 SSD   Intel HD Graphics
520
461     Acer   Netbook    4            128 SSD   Intel HD Graphics
400
184   Xiaomi  Notebook    8            256 SSD    Nvidia GeForce
MX150
1171      HP  Notebook   16            512 SSD  Intel UHD Graphics
620
402   Lenovo  Notebook    8  256 SSD +  1000 HDD       AMD Radeon RX
550
268       HP  Notebook    8           1000 HDD   Nvidia GeForce
930MX

           OpSys  Weight     Price  Touchscreen  Ips         ppi  \
1122   Windows 7    1.43  80612.64            0    0  157.350512
```

```
461   Windows 10    1.40  23176.80            0    0  135.094211
184        No OS    1.95  63882.72            0    1  141.211998
1171  Windows 10    2.10  61751.52            0    0  141.211998
402   Windows 10    2.10  60978.96            0    1  141.211998
268   Windows 10    2.50  54239.04            0    0  127.335675

                    Cpu brand   HDD  SSD  Hybrid  Flash_Storage
1122          Intel Core i5     0  256       0              0
461   Other Intel Processor     0  128       0              0
184           Intel Core i5     0  256       0              0
1171          Intel Core i7     0  512       0              0
402           Intel Core i5  1000  256       0              0
268           Intel Core i7  1000    0       0              0
```

Now , we made 4 seperate column for HDD, SSD ,Hybrid , Flash Storage , where we can see the value of each memory .

```
df.drop(columns='Memory',inplace=True)

df.head()
```

```
  Company   TypeName  Ram                           Gpu  OpSys  Weight
\
0   Apple  Ultrabook    8  Intel Iris Plus Graphics 640  macOS    1.37

1   Apple  Ultrabook    8        Intel HD Graphics 6000  macOS    1.34

2      HP   Notebook    8         Intel HD Graphics 620  No OS    1.86

3   Apple  Ultrabook   16            AMD Radeon Pro 455  macOS    1.83

4   Apple  Ultrabook    8  Intel Iris Plus Graphics 650  macOS    1.37


         Price  Touchscreen  Ips         ppi      Cpu brand  HDD  SSD
Hybrid  \
0   71378.6832            0    1  226.983005  Intel Core i5    0  128
0
1   47895.5232            0    0  127.677940  Intel Core i5    0    0
0
2   30636.0000            0    0  141.211998  Intel Core i5    0  256
0
3  135195.3360            0    1  220.534624  Intel Core i7    0  512
0
4   96095.8080            0    1  226.983005  Intel Core i5    0  256
```

```
0

   Flash_Storage
0              0
1            128
2              0
3              0
4              0
```

```
df.corr()['Price']
```

```
C:\Users\sanya\AppData\Local\Temp\ipykernel_14776\815546952.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only
valid columns or specify the value of numeric_only to silence this
warning.
  df.corr()['Price']

Ram               0.743007
Weight            0.210370
Price             1.000000
Touchscreen       0.191226
Ips               0.252208
ppi               0.473487
HDD              -0.096441
SSD               0.670799
Hybrid            0.007989
Flash_Storage    -0.040511
Name: Price, dtype: float64
```

With respect to memory we can see that -->

1- HDD has negitive corelation means on increasing the HDD the price is decreasing .

2- SSD has positive corealtion , on increasing the value of SSD the price is also increasing .

3- Hybrid and Flash Storage are not corelated so just droping both the columns.

```
df.drop(columns= ['Hybrid', 'Flash_Storage'],inplace=True)
```

```
df.head()
```

```
  Company   TypeName  Ram                           Gpu  OpSys  Weight
\
0   Apple  Ultrabook    8  Intel Iris Plus Graphics 640  macOS    1.37

1   Apple  Ultrabook    8        Intel HD Graphics 6000  macOS    1.34

2      HP   Notebook    8         Intel HD Graphics 620  No OS    1.86

3   Apple  Ultrabook   16            AMD Radeon Pro 455  macOS    1.83
```

```
4   Apple  Ultrabook    8  Intel Iris Plus Graphics 650  macOS    1.37


         Price  Touchscreen  Ips         ppi      Cpu brand  HDD  SSD
0   71378.6832            0    1  226.983005  Intel Core i5    0  128
1   47895.5232            0    0  127.677940  Intel Core i5    0    0
2   30636.0000            0    0  141.211998  Intel Core i5    0  256
3  135195.3360            0    1  220.534624  Intel Core i7    0  512
4   96095.8080            0    1  226.983005  Intel Core i5    0  256
```

Now , let's look into GPU(Graphical Processing Unit)

```
df['Gpu'].value_counts()
```

```
Intel HD Graphics 620      281
Intel HD Graphics 520      185
Intel UHD Graphics 620      68
Nvidia GeForce GTX 1050     66
Nvidia GeForce GTX 1060     48
                          ...
AMD Radeon R5 520             1
AMD Radeon R7                 1
Intel HD Graphics 540         1
AMD Radeon 540                1
ARM Mali T860 MP4             1
Name: Gpu, Length: 110, dtype: int64
```

Now we are spliting Gpu by brand names

```
df['Gpu brand']=df['Gpu'].apply(lambda x :x.split()[0])

df.head()
```

```
  Company   TypeName  Ram                           Gpu  OpSys  Weight
\
0   Apple  Ultrabook    8  Intel Iris Plus Graphics 640  macOS    1.37

1   Apple  Ultrabook    8        Intel HD Graphics 6000  macOS    1.34
```

```
2      HP   Notebook    8         Intel HD Graphics 620  No OS    1.86
3   Apple  Ultrabook   16            AMD Radeon Pro 455  macOS    1.83
4   Apple  Ultrabook    8  Intel Iris Plus Graphics 650  macOS    1.37

         Price  Touchscreen  Ips         ppi      Cpu brand  HDD  SSD
\
0   71378.6832            0    1  226.983005  Intel Core i5    0  128
1   47895.5232            0    0  127.677940  Intel Core i5    0    0
2   30636.0000            0    0  141.211998  Intel Core i5    0  256
3  135195.3360            0    1  220.534624  Intel Core i7    0  512
4   96095.8080            0    1  226.983005  Intel Core i5    0  256

  Gpu brand
0     Intel
1     Intel
2     Intel
3       AMD
4     Intel
```

```
df['Gpu brand'].value_counts()
```

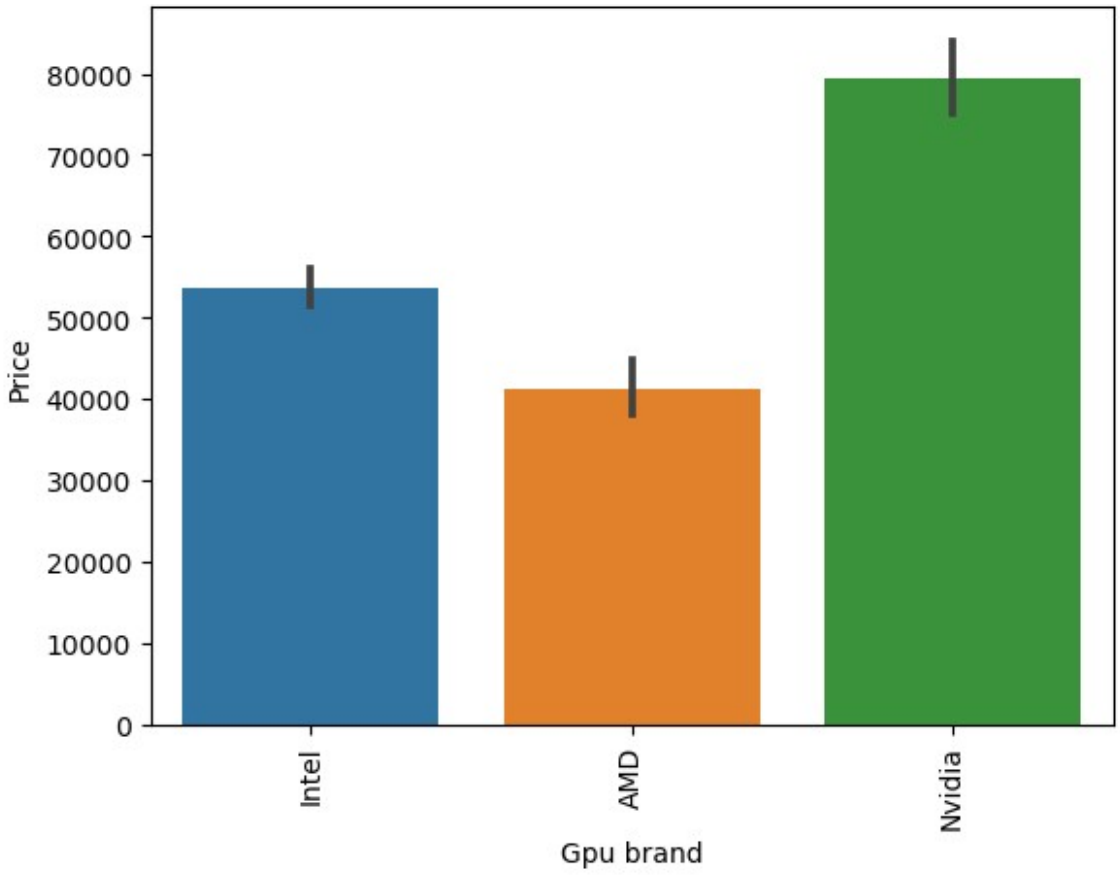
```
Intel     722
Nvidia    400
AMD       180
ARM         1
Name: Gpu brand, dtype: int64
```

One row that is of ARM , as it will not effect much so removing it .

```
df[df['Gpu brand'] == 'ARM']
```

```
      Company            TypeName  Ram                Gpu      OpSys
Weight  \
1191  Samsung  2 in 1 Convertible    4  ARM Mali T860 MP4  Chrome OS
1.15

         Price  Touchscreen  Ips       ppi       Cpu brand  HDD  SSD
Gpu brand
```

```
1191  35111.52            1    1  234.5074  AMD Processor     0    0
ARM

df = df[df['Gpu brand'] != 'ARM']

df['Gpu brand'].value_counts()

Intel     722
Nvidia    400
AMD       180
Name: Gpu brand, dtype: int64

sns.barplot(x= df['Gpu brand'], y = df['Price'])
plt.xticks(rotation ='vertical')
plt.show()
```



Here we can see that 'Nvidia ' is more expensive than 'Intel' and 'Amd'. so yes on predicting the price of laptop Gpu brand is the major factor .

```
df.drop(columns=['Gpu'],inplace=True)

df.head()
```

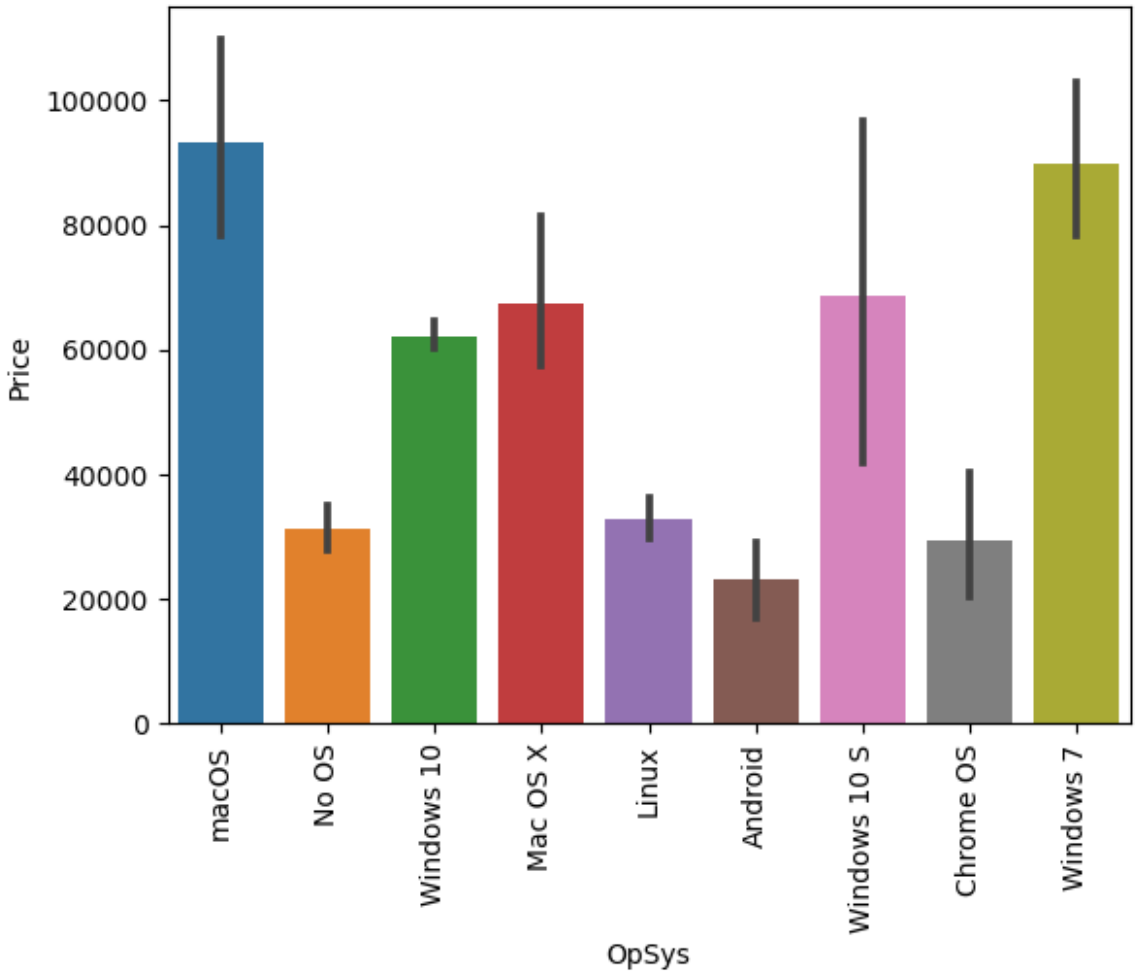
```
  Company   TypeName  Ram  OpSys  Weight        Price  Touchscreen
Ips  \
0   Apple  Ultrabook    8  macOS    1.37   71378.6832            0
1
1   Apple  Ultrabook    8  macOS    1.34   47895.5232            0
0
2      HP   Notebook    8  No OS    1.86   30636.0000            0
0
3   Apple  Ultrabook   16  macOS    1.83  135195.3360            0
1
4   Apple  Ultrabook    8  macOS    1.37   96095.8080            0
1

           ppi      Cpu brand  HDD  SSD Gpu brand
0  226.983005  Intel Core i5    0  128     Intel
1  127.677940  Intel Core i5    0    0     Intel
2  141.211998  Intel Core i5    0  256     Intel
3  220.534624  Intel Core i7    0  512       AMD
4  226.983005  Intel Core i5    0  256     Intel
```

Now look into operating systems impact on laptop price .

```
df['OpSys'].value_counts()
```

```
Windows 10      1072
No OS             66
Linux             62
Windows 7         45
Chrome OS         26
macOS             13
Mac OS X           8
Windows 10 S       8
Android            2
Name: OpSys, dtype: int64
```

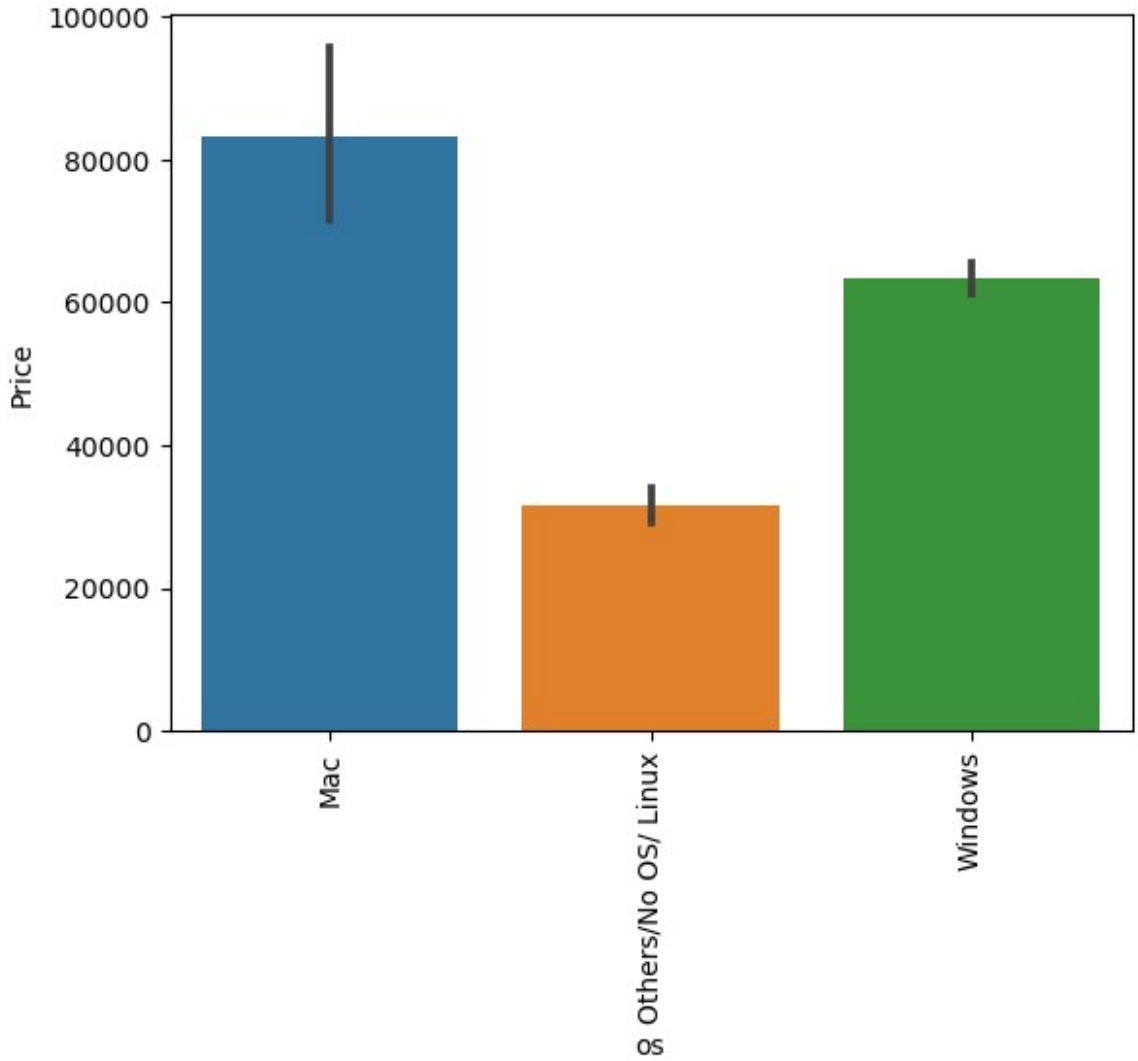
```
sns.barplot(x= df['OpSys'], y = df['Price'])
plt.xticks(rotation ='vertical')
plt.show()
```




```
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.head()
```

```
  Company   TypeName  Ram  OpSys  Weight        Price  Touchscreen
Ips  \
```

```
0   Apple  Ultrabook    8  macOS    1.37   71378.6832             0
1
1   Apple  Ultrabook    8  macOS    1.34   47895.5232             0
0
2      HP   Notebook    8  No OS    1.86   30636.0000             0
0
3   Apple  Ultrabook   16  macOS    1.83  135195.3360             0
1
4   Apple  Ultrabook    8  macOS    1.37   96095.8080             0
1

          ppi      Cpu brand  HDD  SSD Gpu brand                    os

0  226.983005  Intel Core i5    0  128     Intel                   Mac

1  127.677940  Intel Core i5    0    0     Intel                   Mac

2  141.211998  Intel Core i5    0  256     Intel  Others/No OS/ Linux

3  220.534624  Intel Core i7    0  512       AMD                   Mac

4  226.983005  Intel Core i5    0  256     Intel                   Mac


df.drop(columns=['OpSys'], inplace= True)

df.head()

  Company   TypeName  Ram  Weight        Price  Touchscreen  Ips
ppi  \
0   Apple  Ultrabook    8    1.37   71378.6832            0    1
226.983005
1   Apple  Ultrabook    8    1.34   47895.5232            0    0
127.677940
2      HP   Notebook    8    1.86   30636.0000            0    0
141.211998
3   Apple  Ultrabook   16    1.83  135195.3360            0    1
220.534624
4   Apple  Ultrabook    8    1.37   96095.8080            0    1
226.983005

       Cpu brand  HDD  SSD Gpu brand                    os
0  Intel Core i5    0  128     Intel                   Mac
1  Intel Core i5    0    0     Intel                   Mac
2  Intel Core i5    0  256     Intel  Others/No OS/ Linux
3  Intel Core i7    0  512       AMD                   Mac
4  Intel Core i5    0  256     Intel                   Mac

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



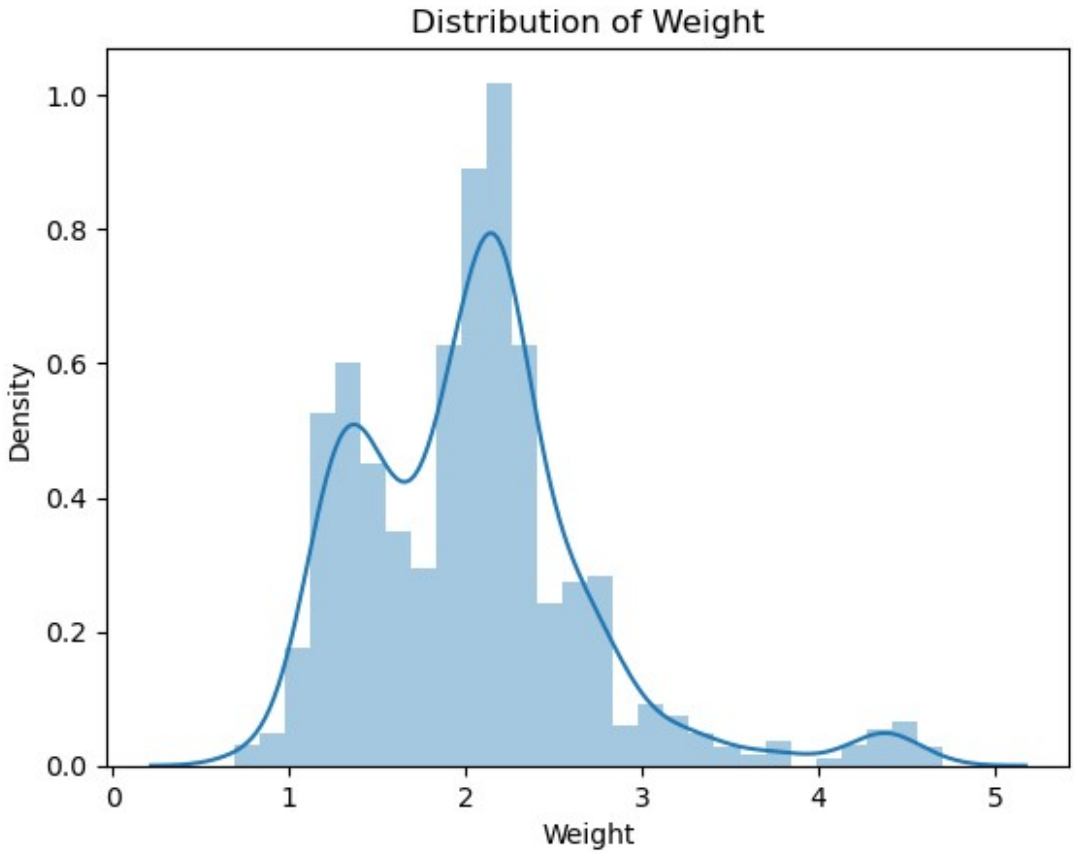
Here , we can estimate that Mac is most expensive , than Windows and Other laptop . So , yes brand also helps in price prediction of laptop .

Now , look into weight .

```
sns.distplot(df['Weight'])
plt.title('Distribution of Weight')
plt.xlabel('Weight')
```

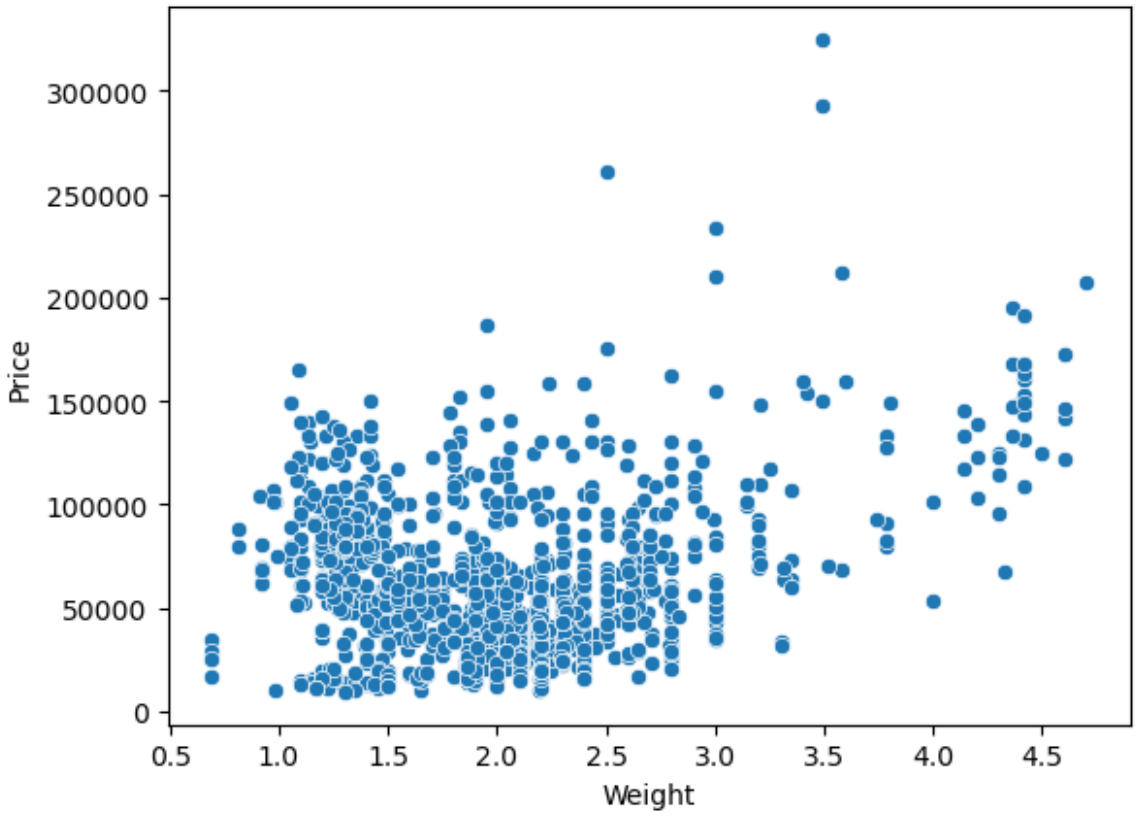
```
plt.ylabel('Density')
plt.show()
```

```
C:\Users\sanya\AppData\Local\Temp\ipykernel_14776\2769249511.py: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 with
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(df['Weight'])
```




```
sns.scatterplot(x= df['Weight'], y= df['Price'])
```

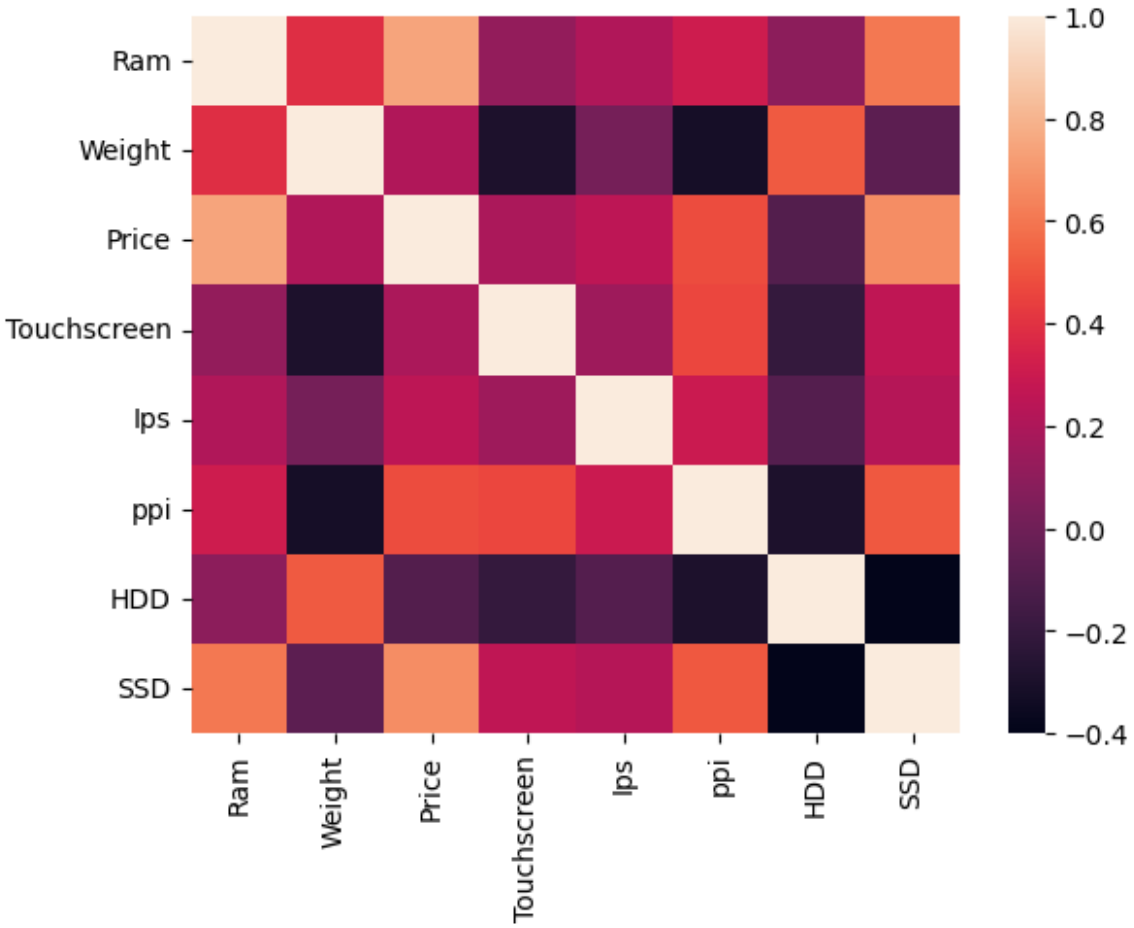
```
<Axes: xlabel='Weight', ylabel='Price'>
```



here , we can predict but a weak prediction that on increasing the weight , price is also increasing . but Weight is not so corelated with price .

```
sns.heatmap(df.corr())
```

```
C:\Users\sanya\AppData\Local\Temp\ipykernel_14776\58359773.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only
valid columns or specify the value of numeric_only to silence this
warning.
  sns.heatmap(df.corr())

<Axes: >
```

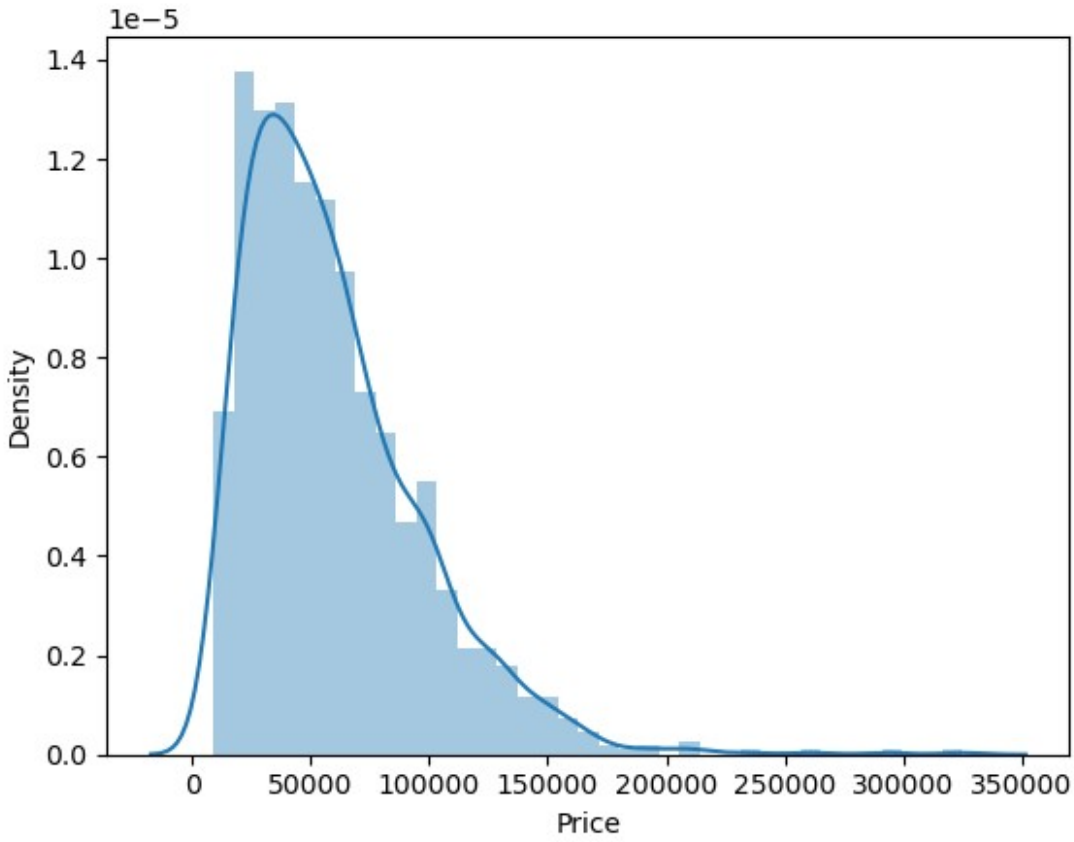


Here in the heatmap , we are seeing that is there any corelation between the independent variables , if it is so than we have to remove one , but we can see as there is not so strong corelation is there as per as the heatmap . So, non of the colum will be eleminated.

```
sns.distplot(df['Price'])
```

```
C:\Users\sanya\AppData\Local\Temp\ipykernel_14776\834922981.py:1:
UserWarning:

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

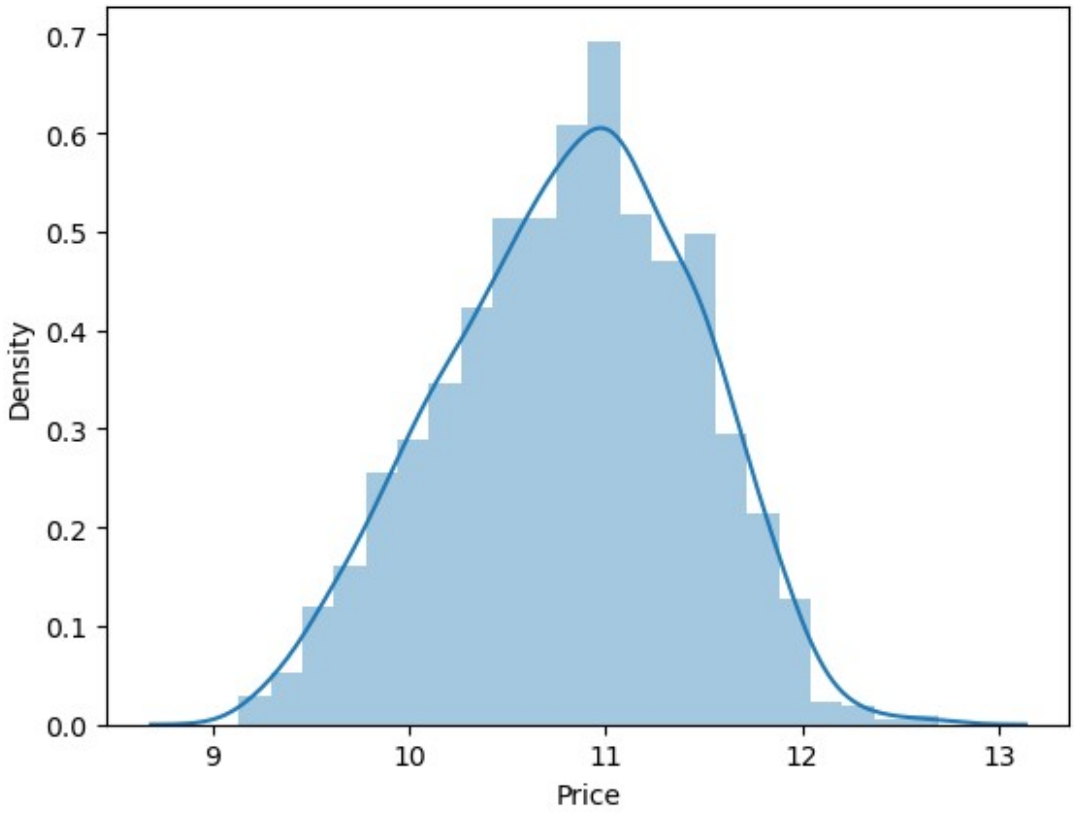
```
Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

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

  sns.distplot(df['Price'])

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



Here the graph is skewed we have to convert it normally distributed .

```
sns.distplot(np.log(df['Price']))

C:\Users\sanya\AppData\Local\Temp\ipykernel_14776\3556049916.py: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 with
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(np.log(df['Price']))

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



Modelling

```
X= df.drop(columns=['Price'])
y = np.log(df['Price'])

X.head()
```

	Company	TypeName	Ram	Weight	Touchscreen	Ips	ppi	\
0	Apple	Ultrabook	8	1.37	0	1	226.983005	

```
1   Apple  Ultrabook    8    1.34            0    0  127.677940
2      HP   Notebook    8    1.86            0    0  141.211998
3   Apple  Ultrabook   16    1.83            0    1  220.534624
4   Apple  Ultrabook    8    1.37            0    1  226.983005

       Cpu brand  HDD  SSD Gpu brand                   os
0  Intel Core i5    0  128     Intel                  Mac
1  Intel Core i5    0    0     Intel                  Mac
2  Intel Core i5    0  256     Intel  Others/No OS/ Linux
3  Intel Core i7    0  512       AMD                  Mac
4  Intel Core i5    0  256     Intel                  Mac

y.head()

0    11.175755
1    10.776777
2    10.329931
3    11.814476
4    11.473101
Name: Price, dtype: float64

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size=0.15,random_state=2)

X_train

      Company            TypeName  Ram  Weight  Touchscreen  Ips
ppi  \
183   Toshiba            Notebook    8    2.00            0    0
100.454670
1141      MSI              Gaming    8    2.40            0    0
141.211998
1049     Asus             Netbook    4    1.20            0    0
135.094211
1020     Dell  2 in 1 Convertible    4    2.08            1    1
141.211998
878      Dell            Notebook    4    2.18            0    0
141.211998
...       ...                 ...  ...     ...          ...  ...
...
466      Acer            Notebook    4    2.20            0    0
100.454670
299      Asus           Ultrabook   16    1.63            0    0
141.211998
493      Acer            Notebook    8    2.20            0    0
100.454670
527    Lenovo            Notebook    8    2.20            0    0
100.454670
1193    Apple           Ultrabook    8    0.92            0    1
```

```
226.415547
```

	Cpu brand	HDD	SSD	Gpu brand	os
183	Intel Core i5	0	128	Intel	Windows
1141	Intel Core i7	1000	128	Nvidia	Windows
1049	Other Intel Processor	0	0	Intel	Others/No OS/ Linux
1020	Intel Core i3	1000	0	Intel	Windows
878	Intel Core i5	1000	128	Nvidia	Windows
...	...	...	...	...	...
466	Intel Core i3	500	0	Nvidia	Windows
299	Intel Core i7	0	512	Nvidia	Windows
493	AMD Processor	1000	0	AMD	Windows
527	Intel Core i3	2000	0	Nvidia	Others/No OS/ Linux
1193	Other Intel Processor	0	0	Intel	Mac

```
[1106 rows x 12 columns]
```

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

from sklearn.linear_model import LinearRegression,Lasso,Ridge
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor, AdaBoostRegressor, ExtraTreesRegressor,
BaggingRegressor,HistGradientBoostingRegressor,StackingRegressor
from sklearn.svm  import  SVR
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import r2_score, mean_absolute_error
```

Linear Regression

```
step1= ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,7,1,10,11])

],remainder = 'passthrough')
```

```
step2= LinearRegression()

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score ',r2_score(y_test,y_pred))
print('Mean absolute error ',mean_absolute_error(y_test,y_pred))
```

```
R2 score  0.8073277448418574
Mean absolute error  0.21017827976429299

C:\Users\sanya\anaconda3\Lib\site-packages\sklearn\preprocessing\
_encoders.py:975: FutureWarning: `sparse` was renamed to
`sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default
value.
  warnings.warn(
```

Ridge Regression

```
step1= ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,7,1,10,11])

],remainder = 'passthrough')

step2= Ridge(alpha=10)

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score ',r2_score(y_test,y_pred))
print('Mean absolute error ',mean_absolute_error(y_test,y_pred))
```

```
R2 score  0.81273310313118
Mean absolute error  0.20926802242583048
```

```
C:\Users\sanya\anaconda3\Lib\site-packages\sklearn\preprocessing\
_encoders.py:975: FutureWarning: `sparse` was renamed to
`sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default
value.
  warnings.warn(
```

Lasso Regression

```
step1= ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,7,1,10,11])

],remainder = 'passthrough')

step2= Lasso(alpha=0.001)

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score ',r2_score(y_test,y_pred))
print('Mean absolute error ',mean_absolute_error(y_test,y_pred))
```

```
R2 score  0.8071853123382866
Mean absolute error  0.21114343357087614

C:\Users\sanya\anaconda3\Lib\site-packages\sklearn\preprocessing\
_encoders.py:975: FutureWarning: `sparse` was renamed to
`sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default
value.
  warnings.warn(
```

KNN

```
step1= ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,7,1,10,11])

],remainder = 'passthrough')

step2= KNeighborsRegressor(n_neighbors=3)
```

```
pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score ',r2_score(y_test,y_pred))
print('Mean absolute error ',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\sanya\anaconda3\Lib\site-packages\sklearn\preprocessing\
_encoders.py:975: FutureWarning: `sparse` was renamed to
`sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default
value.
  warnings.warn(

R2 score  0.803148868705085
Mean absolute error  0.19264883332948868
```

Decision Tree

```
step1= ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,7,1,10,11])

],remainder = 'passthrough')

step2= DecisionTreeRegressor(max_depth=8)

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score ',r2_score(y_test,y_pred))
print('Mean absolute error ',mean_absolute_error(y_test,y_pred))
```

```
R2 score  0.8462969667947657
Mean absolute error  0.179434641824465

C:\Users\sanya\anaconda3\Lib\site-packages\sklearn\preprocessing\
_encoders.py:975: FutureWarning: `sparse` was renamed to
`sparse_output` in version 1.2 and will be removed in 1.4.
```

```
`sparse_output` is ignored unless you leave `sparse` to its default
value.
  warnings.warn(
```

SVM

```
step1= ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,7,1,10,11])

],remainder = 'passthrough')

step2= SVR(kernel='rbf', C= 10000, epsilon=0.1)

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score ',r2_score(y_test,y_pred))
print('Mean absolute error ',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\sanya\anaconda3\Lib\site-packages\sklearn\preprocessing\
_encoders.py:975: FutureWarning: `sparse` was renamed to
`sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default
value.
  warnings.warn(

R2 score  0.808318090228966
Mean absolute error  0.20239059427193437
```

Random Forest

```
step1= ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,7,1,10,11])

],remainder = 'passthrough')

step2= RandomForestRegressor(n_estimators=100,
                             random_state=3,
                             max_samples=0.5,
                             max_features=0.75,
                             max_depth= 15
```

```
                              )

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score ',r2_score(y_test,y_pred))
print('Mean absolute error ',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\sanya\anaconda3\Lib\site-packages\sklearn\preprocessing\
_encoders.py:975: FutureWarning: `sparse` was renamed to
`sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default
value.
  warnings.warn(

R2 score  0.8872275616843143
Mean absolute error  0.1598510712113854
```

ExtraTrees

```
step1= ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,7,1,10,11])

],remainder = 'passthrough')

step2= ExtraTreesRegressor(n_estimators=100,
                           random_state=3,
                           max_samples=None,
                           max_features=0.75,
                           max_depth= 15
                           )

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score ',r2_score(y_test,y_pred))
print('Mean absolute error ',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\sanya\anaconda3\Lib\site-packages\sklearn\preprocessing\
_encoders.py:975: FutureWarning: `sparse` was renamed to
`sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default
value.
  warnings.warn(

R2 score  0.8779014490286797
Mean absolute error  0.1590822333498326
```

AdaBoost

```
step1= ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,7,1,10,11])

],remainder = 'passthrough')

step2= AdaBoostRegressor(n_estimators=15,learning_rate=1.0)
pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score ',r2_score(y_test,y_pred))
print('Mean absolute error ',mean_absolute_error(y_test,y_pred))

R2 score  0.792105899607572
Mean absolute error  0.23012312417719016

C:\Users\sanya\anaconda3\Lib\site-packages\sklearn\preprocessing\
_encoders.py:975: FutureWarning: `sparse` was renamed to
`sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default
value.
  warnings.warn(
```

Gradient Boost

```
step1= ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,7,1,10,11])

],remainder = 'passthrough')
```

```
step2= GradientBoostingRegressor(n_estimators=500)

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score ',r2_score(y_test,y_pred))
print('Mean absolute error ',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\sanya\anaconda3\Lib\site-packages\sklearn\preprocessing\
_encoders.py:975: FutureWarning: `sparse` was renamed to
`sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default
value.
  warnings.warn(

R2 score  0.8826180407219418
Mean absolute error  0.15904347241564465
```

We are getting the best result from RANDOM FOREST , so now exporting the model to make a website out of it .

Exporting the model

Now , as comparision to all model , we got Random Forest as the best model , by refering it we will make our website .

```
import pickle

pickle.dump(df,open('df.pkl','wb'))
pickle.dump(pipe,open('pipe.pkl','wb'))

df
```

```
     Company            TypeName  Ram  Weight        Price
Touchscreen  Ips  \
0      Apple          Ultrabook    8    1.37   71378.6832
0    1
```

```
1      Apple           Ultrabook    8    1.34   47895.5232
0    0
2         HP            Notebook    8    1.86   30636.0000
0    0
3      Apple           Ultrabook   16    1.83  135195.3360
0    1
4      Apple           Ultrabook    8    1.37   96095.8080
0    1
...      ...                 ...  ...     ...          ...           ..
.  ...
1298  Lenovo  2 in 1 Convertible    4    1.80   33992.6400
1    1
1299  Lenovo  2 in 1 Convertible   16    1.30   79866.7200
1    1
1300  Lenovo            Notebook    2    1.50   12201.1200
0    0
1301      HP            Notebook    6    2.19   40705.9200
0    0
1302    Asus            Notebook    4    2.20   19660.3200
0    0

              ppi              Cpu brand   HDD  SSD Gpu brand  \
0     226.983005          Intel Core i5     0  128     Intel
1     127.677940          Intel Core i5     0    0     Intel
2     141.211998          Intel Core i5     0  256     Intel
3     220.534624          Intel Core i7     0  512       AMD
4     226.983005          Intel Core i5     0  256     Intel
...          ...                   ...   ...  ...       ...
1298  157.350512          Intel Core i7     0  128     Intel
1299  276.053530          Intel Core i7     0  512     Intel
1300  111.935204  Other Intel Processor     0    0     Intel
1301  100.454670          Intel Core i7  1000    0       AMD
1302  100.454670  Other Intel Processor   500    0     Intel

                        os
0                      Mac
1                      Mac
2     Others/No OS/ Linux
3                      Mac
4                      Mac
...                    ...
1298               Windows
1299               Windows
1300               Windows
1301               Windows
1302               Windows

[1302 rows x 13 columns]
```

```
import streamlit as st
import pickle
import pandas as pd
import  numpy as np

# Import the
model
pipe = pickle.load(open('pipe.pkl', 'rb'))
#df = pickle.load(open('df.pkl', 'rb'))
df
= pd.read_pickle('df.pkl')

st.title('Laptop Price Predictor')

# Brand
company =
st.selectbox('Brand', df['Company'].unique())

# Type of laptop
laptop_type =
st.selectbox('Type', df['TypeName'].unique())

# RAM
ram = st.selectbox('RAM (inGB)', [2, 4,
6, 8, 12, 16, 24, 32, 64])

# Weight
weight = st.number_input('Weight of the laptop')

#
Touchscreen
touchscreen = st.selectbox('Touchscreen', ['NO', 'YES'])

# IPS
ips =
st.selectbox('IPS', ['NO', 'YES'])

# Screensize
screensize = st.number_input('Screen
Size')

# Resolution
resolution = st.selectbox('Screen Resolution', ['1920*1080',
'1366*768', '1600*900', '3840*2160', '3200*1800', '2880*1800', '2560*1600', '2560*1440',
'2304*1440'])

# CPU
cpu = st.selectbox('CPU', df['Cpu brand'].unique())

# HDD
hdd =
st.selectbox('HDD (IN GB)', [0, 128, 256, 512, 1024, 2048])

# SSD
ssd = st.selectbox('SSD
(IN GB)', [0, 8, 128, 256, 512, 1024])

# GPU
gpu = st.selectbox('GPU', df['Gpu
brand'].unique())

# OS
os = st.selectbox('OS', df['os'].unique())

if st.button('Predict
Price'):
    #query
    ppi=None
    if  touchscreen=='Yes':
        touchscreen=1

else:
        touchscreen=0
```

```
    if ips =='Yes':
        ips=1
    else:
        ips= 0

   X_res= int(resolution.split('*')[0])
    Y_res= int(resolution.split('*')[1])

ppi=((X_res**2)+(Y_res**2))**0.5/screensize


query=np.array([company,laptop_type,ram,weight,touchscreen,ips,ppi,cpu,hdd,ssd,gpu,os])

query=query.reshape(1,12)
    st.title("The predicted price of this configuration is :
Rs. "+str(int(np.exp(pipe.predict(query)[0]))))
```