Laptop-price-Predictor

In [210]:
```
import numpy as np
import pandas as pd
```

In [211]:
```
laptop_data=pd.read_csv('laptop_data.csv')
```

In [212]:
```
laptop_data.head()
```

Out[212]:

	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	We
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	1.3
1	1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.3
2	2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.8
3	3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.8
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	1.3

In [213]:
```
laptop_data.shape
```

Out[213]:
```
(1303, 12)
```

In [214]:
```
laptop_data.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
```

In [215]:
```
laptop_data.duplicated().sum()
```

Out[215]:
```
0
```

In [216]:
```
laptop_data.isnull().sum()
```

Out[216]:
```
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
```

In [217]:
```
#dropping a column 'unnamed'
laptop_data.drop(columns='Unnamed: 0',axis=1,inplace=True)
```

In [218]:
```
laptop_data.head()
```

Out[218]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	
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
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	47895
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
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
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

In [219]:
```
laptop_data['Ram']=laptop_data['Ram'].str.replace('GB','')
laptop_data['Weight']=laptop_data['Weight'].str.replace('kg','')
```

In [220]:
```
#making Ram and Weight as int
laptop_data['Ram']=laptop_data['Ram'].astype('int32')
laptop_data['Weight']=laptop_data['Weight'].astype('float32')
```

In [221]:
```
laptop_data.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
```

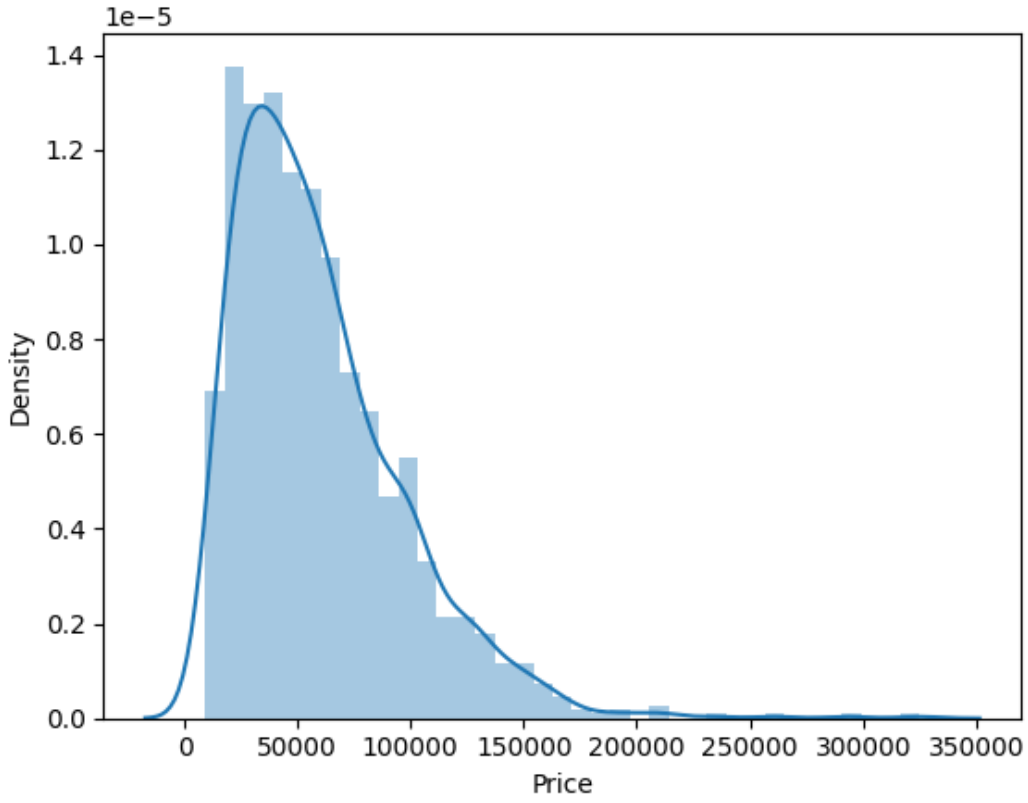
#Data Analysis

In [222]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [223]:
```
sns.distplot(laptop_data['Price'])
```

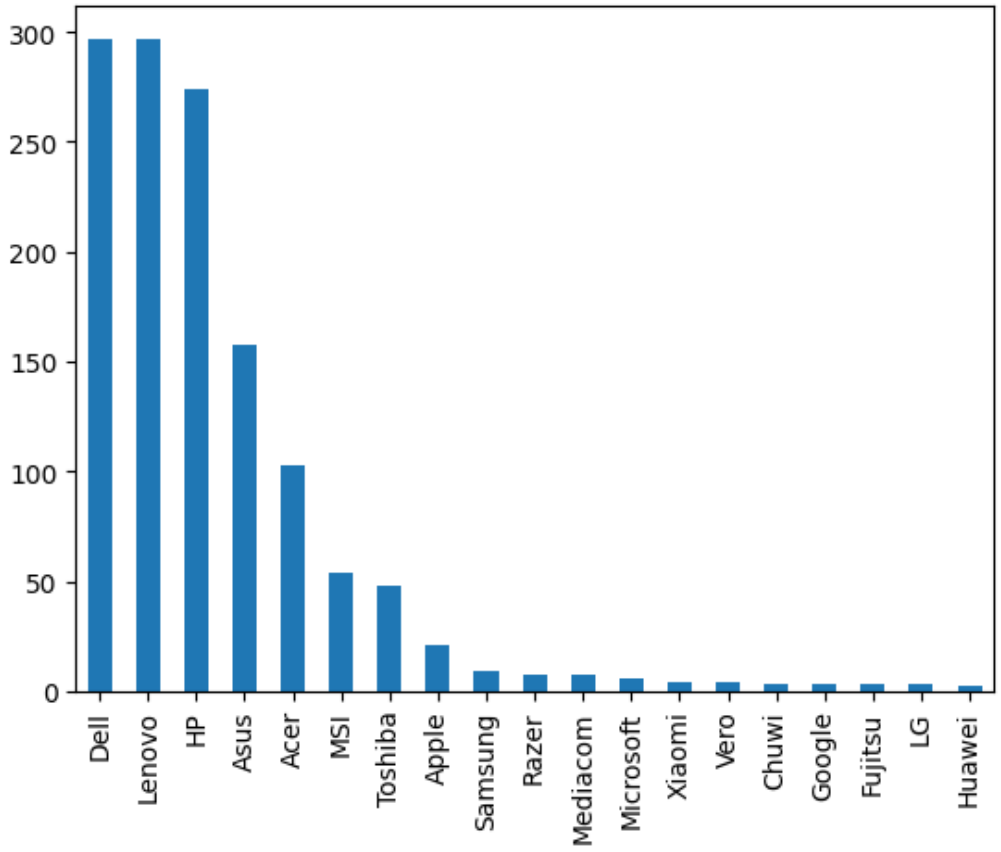
```
C:\Users\pcc\AppData\Local\Temp\ipykernel_9820\1627308308.py:1: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

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

Out[223]:
```
<Axes: xlabel='Price', ylabel='Density'>
```

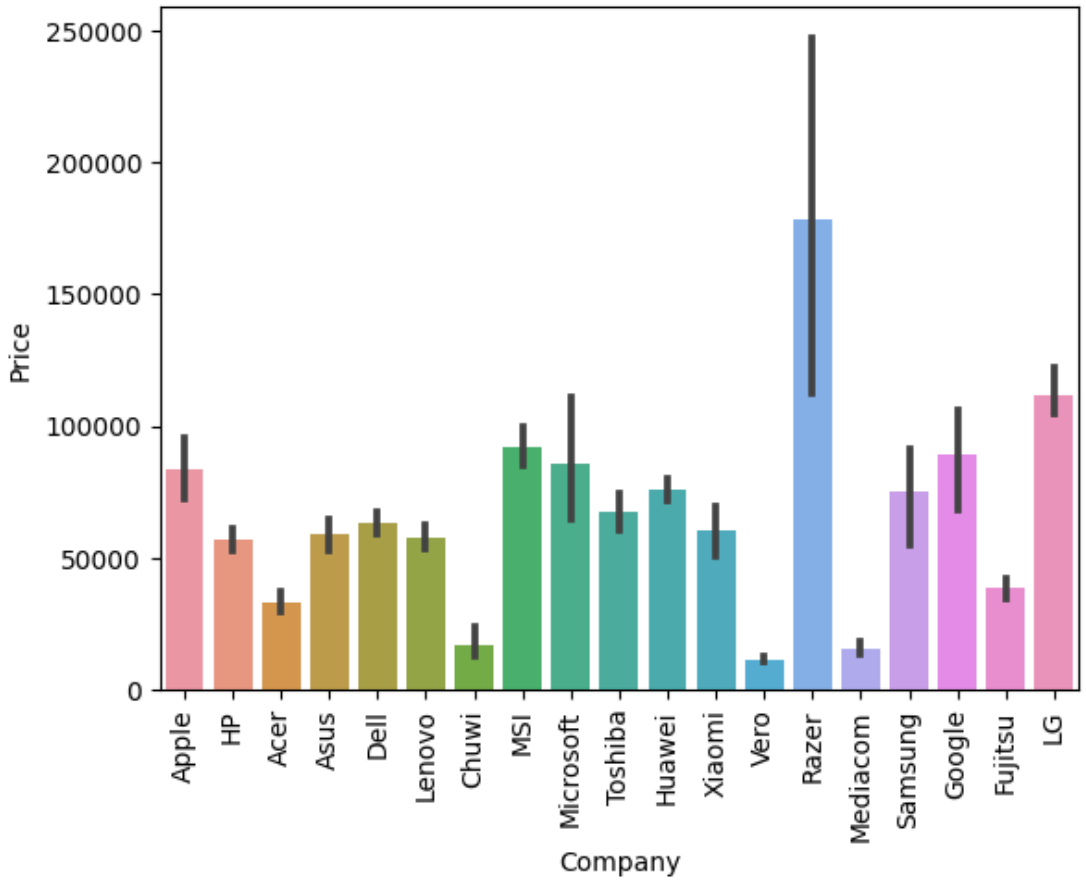


In [224]:
```
laptop_data['Company'].value_counts().plot(kind='bar')
```

Out[224]: <Axes: >

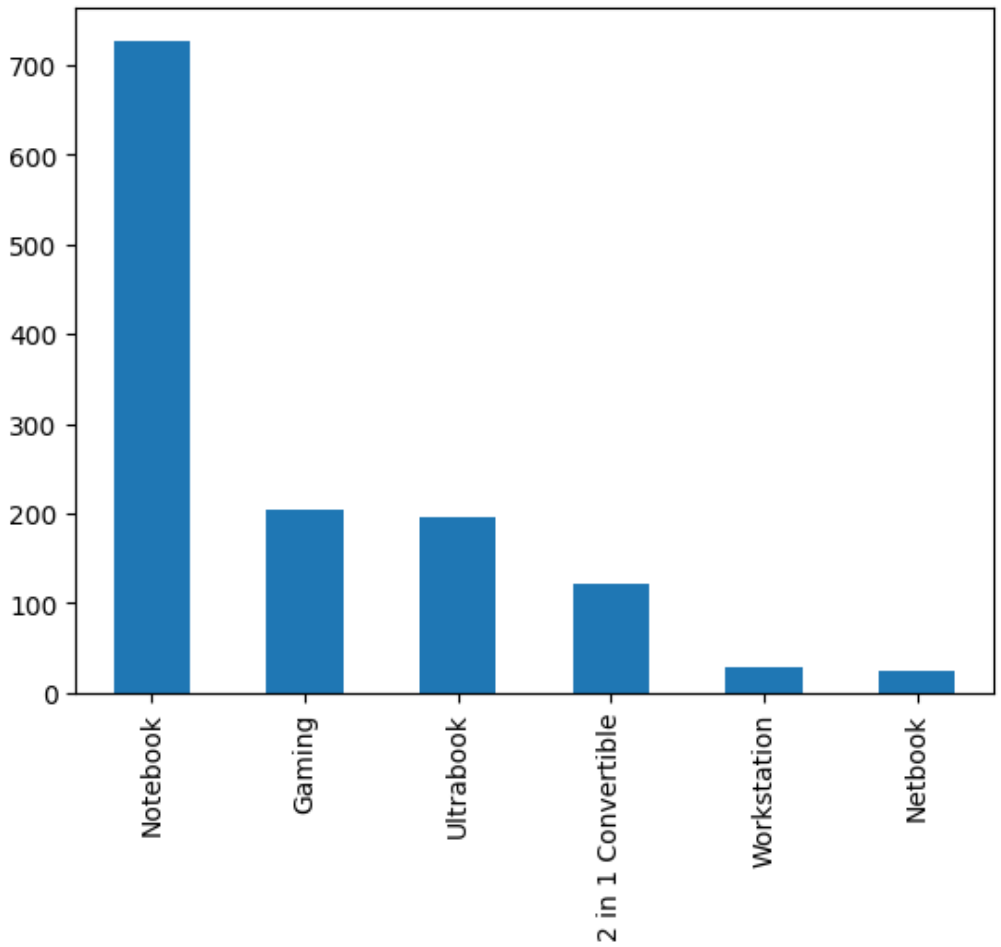


In [225]:
```
sns.barplot(x=laptop_data['Company'],y=laptop_data['Price'])
plt.xticks(rotation='vertical')
plt.show()
```

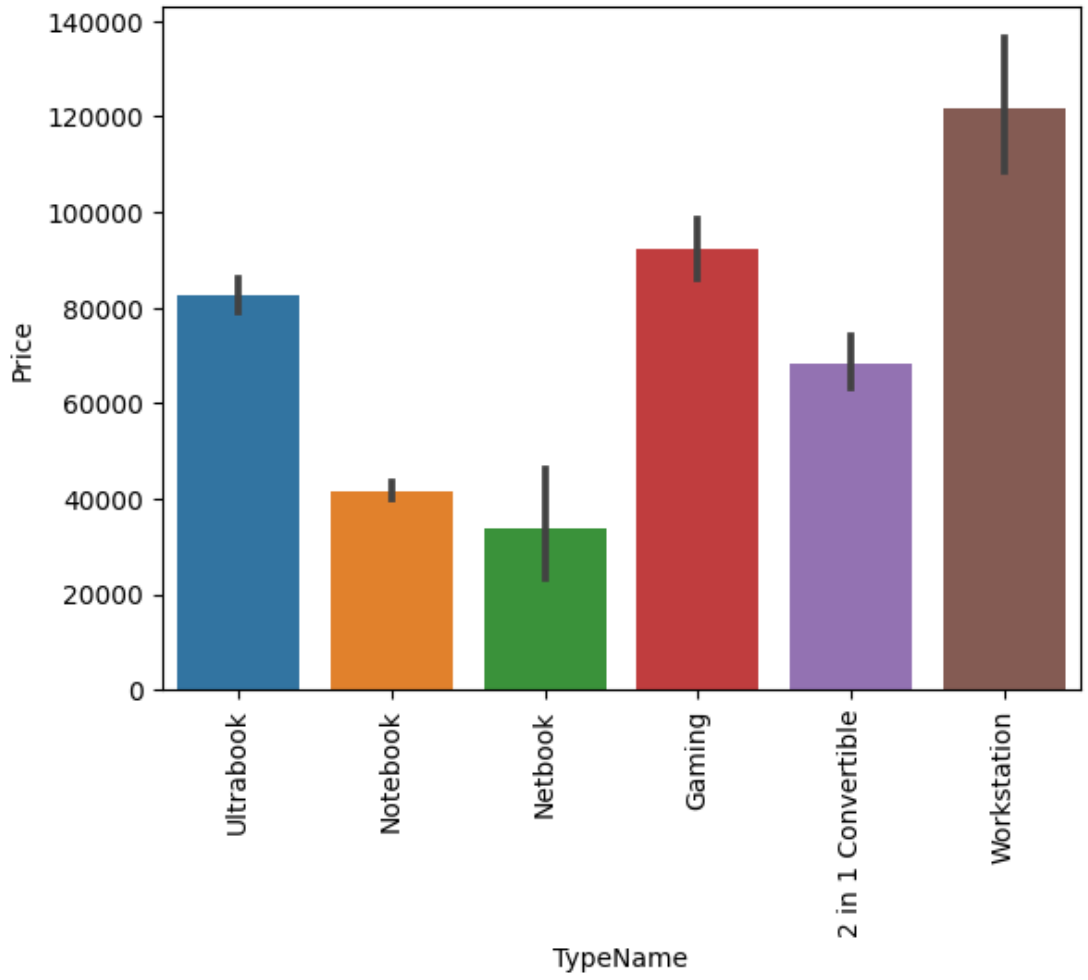


In [226]:
```
laptop_data['TypeName'].value_counts().plot(kind='bar')
```

Out[226]: <Axes: >



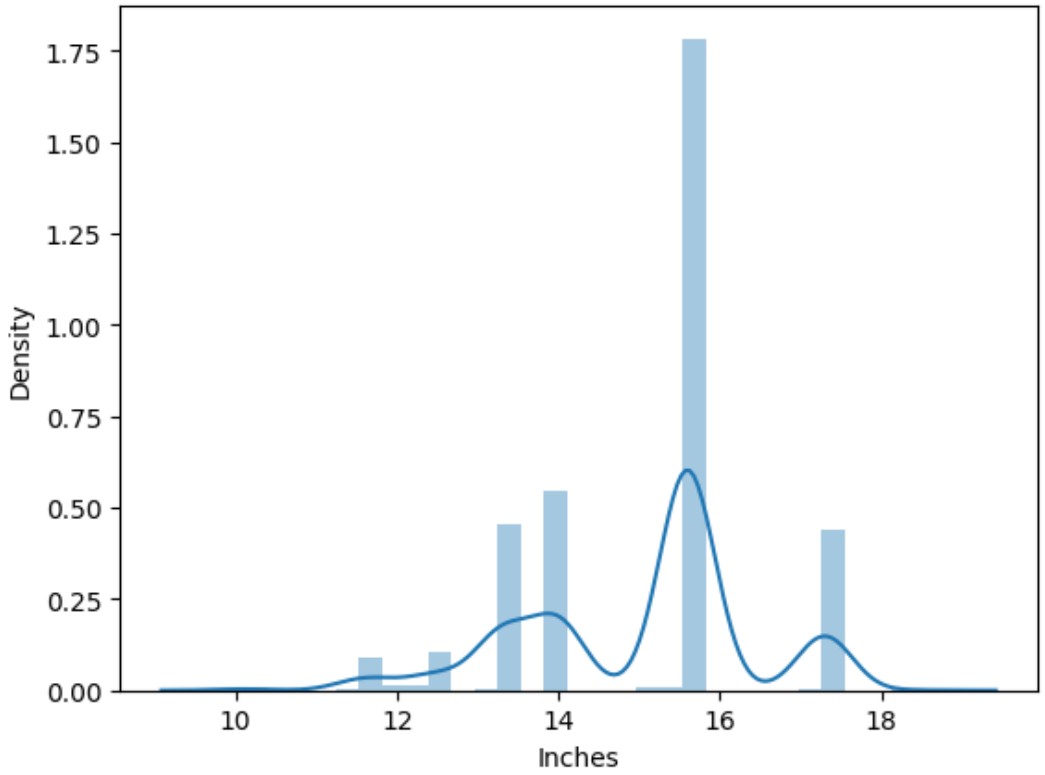
In [227]:
```
sns.barplot(x=laptop_data['TypeName'],y=laptop_data['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



In [228]:
```
sns.distplot(laptop_data['Inches'])
```

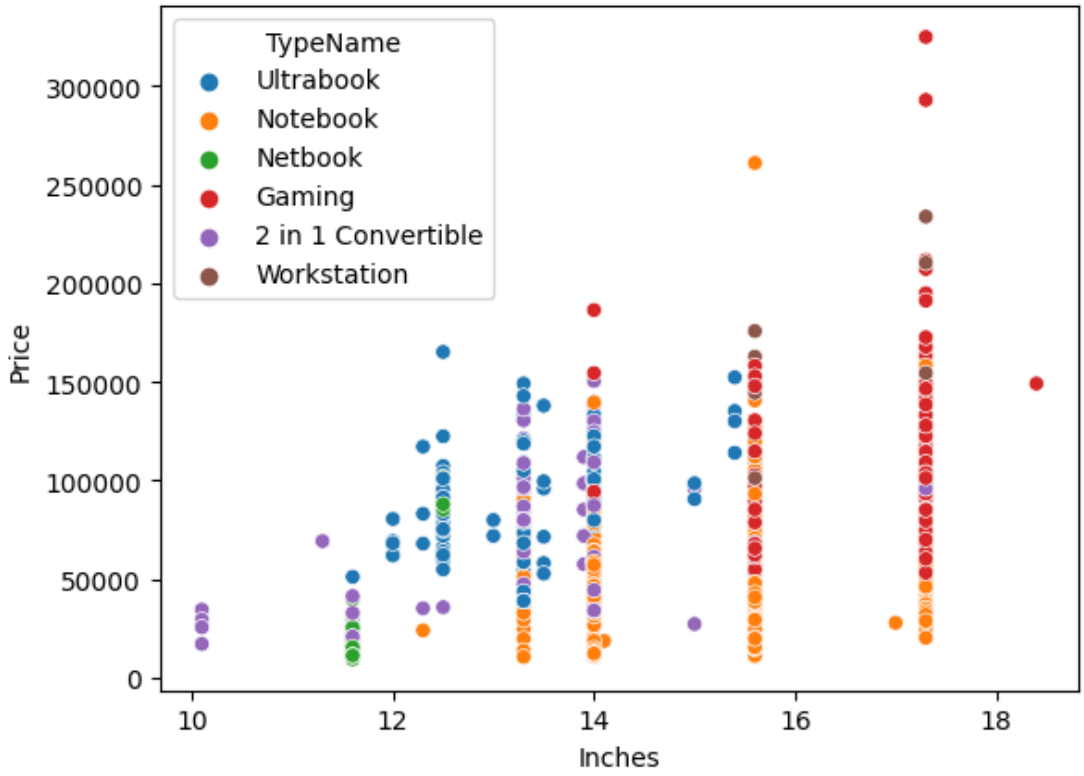
```
C:\Users\pcc\AppData\Local\Temp\ipykernel_9820\157593019.py:1: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(laptop_data['Inches'])
```

Out[228]: <Axes: xlabel='Inches', ylabel='Density'>



In [229]:
```
sns.scatterplot(data=laptop_data,x='Inches',y='Price',hue='TypeName')
```

Out[229]:
```
<Axes: xlabel='Inches', ylabel='Price'>
```



In [230]:
```
laptop_data['ScreenResolution'].value_counts()
```

Out[230]:
```
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
```

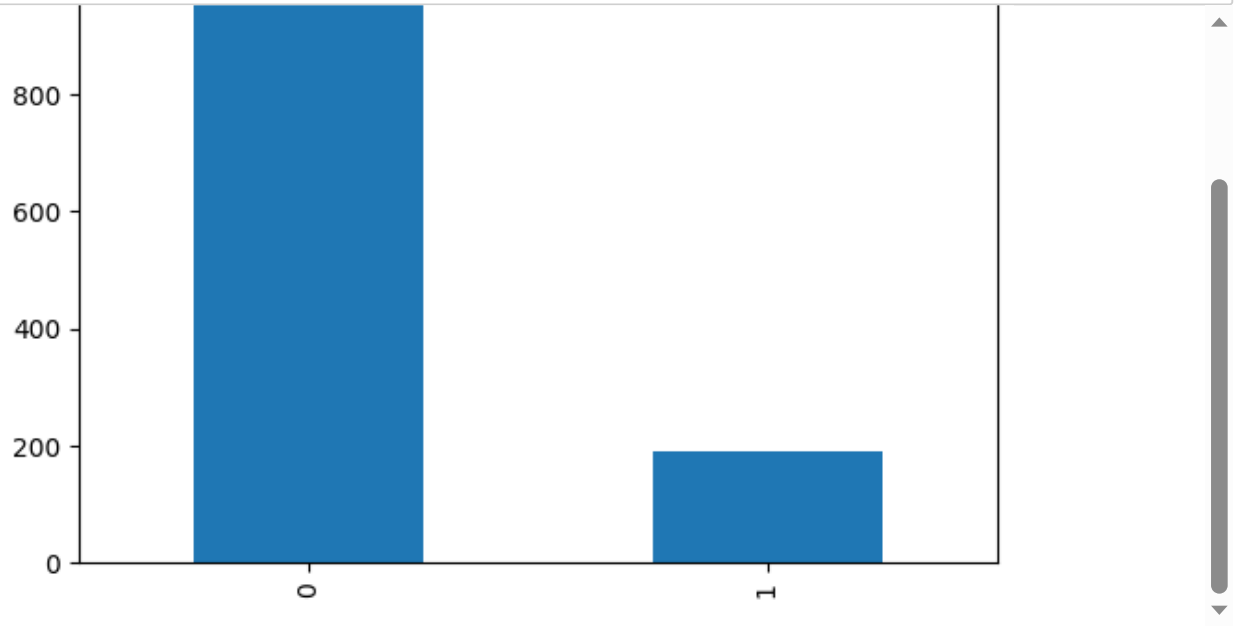
In [231]:
```
#making a new column 'touchscreen'
laptop_data['Touchscreen']=laptop_data['ScreenResolution'].apply(lambda x:1 if 'Touchscr
```

In [232]:
```
laptop_data.sample(5)
```

Out[232]:

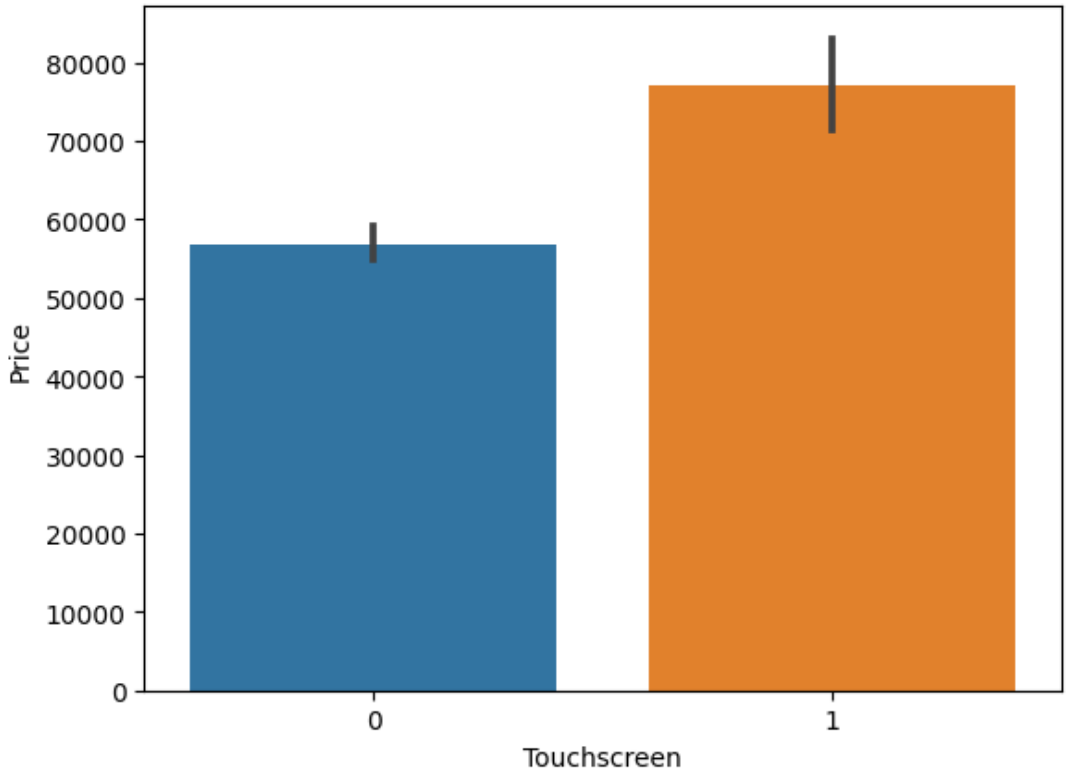
	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	
1133	HP	Ultrabook	15.6	Full HD 1920x1080	Intel Core i7 7500U 2.7GHz	8	256GB SSD	Intel HD Graphics 620	Windows 10	1.84	4
284	Acer	Notebook	15.6	IPS Panel Full HD 1920x1080	Intel Core i7 8550U 1.8GHz	8	256GB SSD	Nvidia GeForce MX150	Windows 10	3.00	5
259	Lenovo	2 in 1 Convertible	15.6	IPS Panel Full HD / Touchscreen 1920x1080	Intel Core i7 7700HQ 2.8GHz	8	512GB SSD	Nvidia GeForce GTX 1050M	Windows 10	2.00	9
541	Dell	Notebook	14.0	Full HD 1920x1080	Intel Core i3 6006U 2GHz	4	128GB SSD	Intel HD Graphics 520	Windows 10	1.60	3
492	Asus	Gaming	15.6	Full HD 1920x1080	Intel Core i7 7700HQ 2.8GHz	8	128GB SSD + 1TB HDD	Nvidia GeForce GTX 1050	Windows 10	1.99	7

In [233]:
```
laptop_data['Touchscreen'].value_counts().plot(kind='bar')
```



In [234]:
```
sns.barplot(data=laptop_data,x='Touchscreen',y='Price')
```

Out[234]:
```
<Axes: xlabel='Touchscreen', ylabel='Price'>
```



In [235]:
```
laptop_data['IPS']=laptop_data['ScreenResolution'].apply(lambda x:1 if 'IPS' in x else 0
```

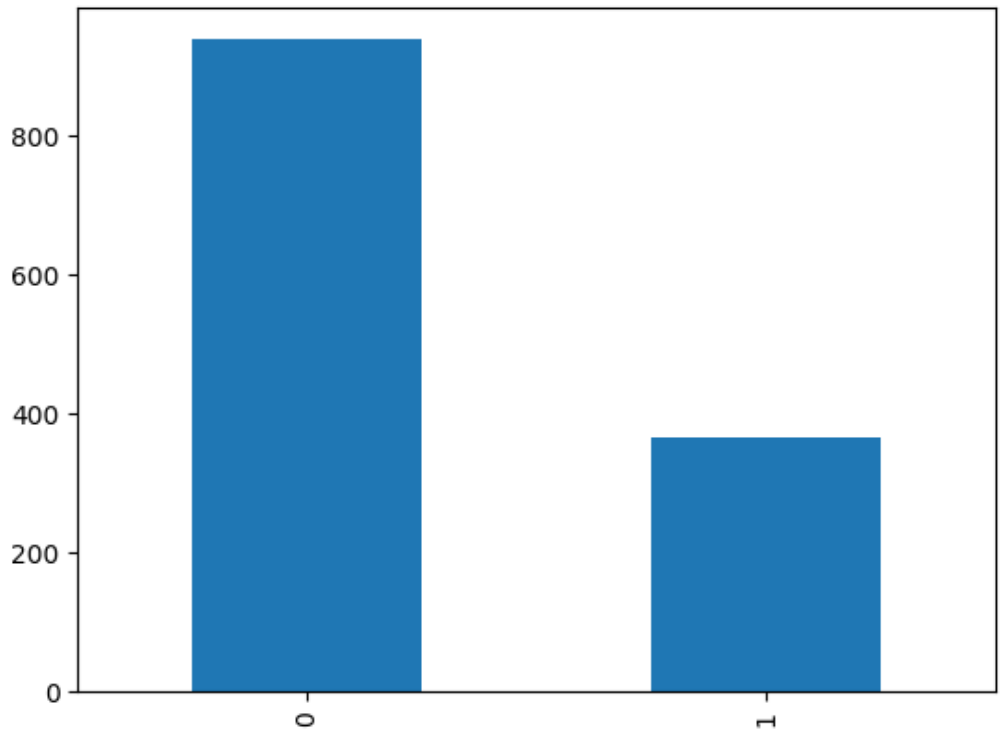
In [236]:
```
laptop_data.sample(5)
```

Out[236]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight
1191	Samsung	2 in 1 Convertible	12.3	IPS Panel Touchscreen 2400x1600	Samsung Cortex A72&A53 2.0GHz	4	32GB Flash Storage	ARM Mali T860 MP4	Chrome OS	1.15
176	Acer	Notebook	15.6	1366x768	Intel Core i3 6006U 2GHz	4	128GB SSD	Intel HD Graphics 520	Windows 10	2.10
414	Asus	2 in 1 Convertible	13.3	IPS Panel Full HD / Touchscreen 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	Windows 10	1.27
799	Dell	Notebook	15.6	Full HD 1920x1080	Intel Core i3 6006U 2GHz	8	256GB SSD	AMD Radeon R5 M420X	Windows 10	2.00
864	Dell	Ultrabook	13.3	Quad HD+ / Touchscreen 3200x1800	Intel Core i7 7660U 2.5GHz	16	512GB SSD	Intel Iris Plus Graphics 640	Windows 10	1.29

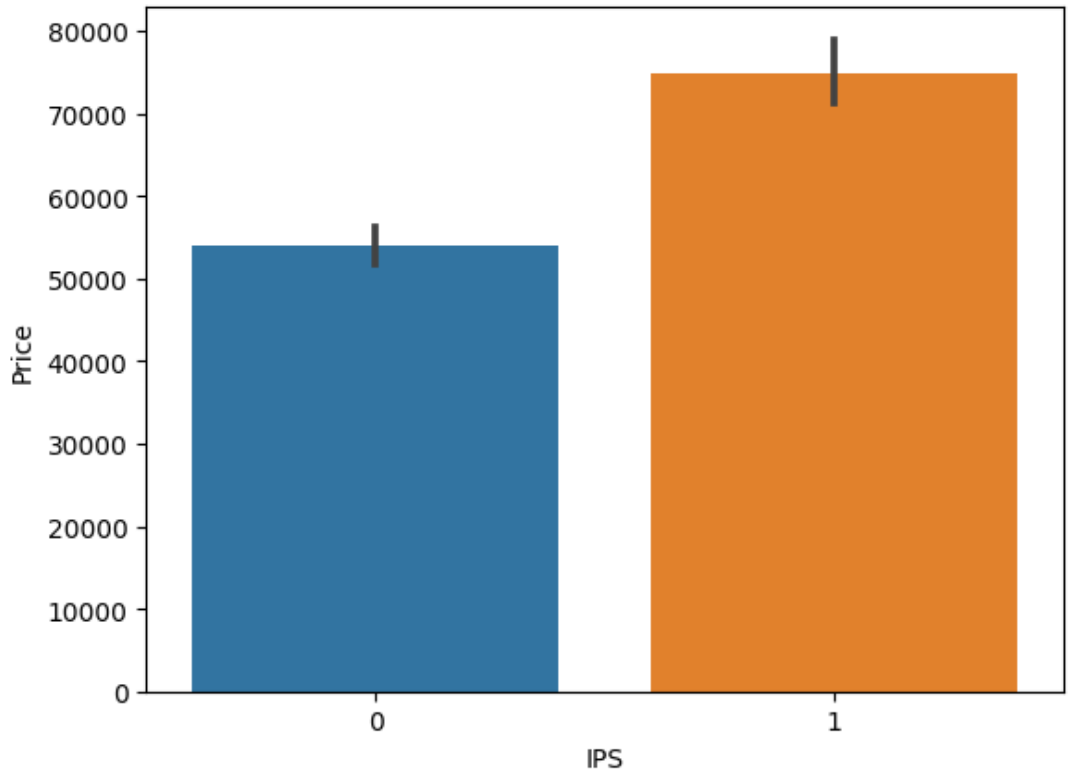
In [237]:
```
laptop_data['IPS'].value_counts().plot(kind='bar')
```

Out[237]: <Axes: >



In [238]:
```
sns.barplot(data=laptop_data,x='IPS',y='Price')
```

Out[238]: <Axes: xlabel='IPS', ylabel='Price'>



In [239]:
```
new=laptop_data['ScreenResolution'].str.split('x',n=1,expand=True)
laptop_data['x_res']=new[0]
laptop_data['y_res']=new[1]
```

In [240]:
```
laptop_data.sample()
```

Out[240]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	
788	Acer	Gaming	17.3	IPS Panel Full HD 1920x1080	Intel Core i7 7700HQ 2.8GHz	16	256GB SSD + 1TB HDD	Nvidia GeForce GTX 1060	Windows 10	4.2	12

In [241]:
```
laptop_data['x_res'].str.replace(',','').str.findall(r'(\d+\.?\d+)').apply(lambda x:x[0]
```

Out[241]:
```
0       2560
1       1440
2       1920
3       2880
4       2560
        ...
1298    1920
1299    3200
1300    1366
1301    1366
1302    1366
Name: x_res, Length: 1303, dtype: object
```

In [242]:
```
laptop_data['x_res']=laptop_data['x_res'].str.replace(',','').str.findall(r'(\d+\.?\d+)'
```

In [243]:
```
laptop_data.sample()
```

Out[243]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	
1206	HP	Notebook	15.6	1366x768	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	Windows 10	1.91	30

In [244]:
```
#changing data type
laptop_data['x_res']=laptop_data['x_res'].astype(int)
laptop_data['y_res']=laptop_data['y_res'].astype(int)
```

In [245]:
```
laptop_data.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
```

In [246]:
```
laptop_data.corr()['Price']
```

```
C:\Users\pcc\AppData\Local\Temp\ipykernel_9820\2420686662.py:1: FutureWarning: The defa
ult 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 sil
ence this warning.
  laptop_data.corr()['Price']
```

Out[246]:
```
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
```

In [247]:
```
laptop_data['ppi']=(((laptop_data['x_res']**2+laptop_data['y_res']**2))**0.5/laptop_data
```

In [248]:
```
laptop_data.corr()['Price']
```

```
C:\Users\pcc\AppData\Local\Temp\ipykernel_9820\2420686662.py:1: FutureWarning: The defa
ult 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 sil
ence this warning.
  laptop_data.corr()['Price']
```

Out[248]:
```
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
```

In [249]:
```
#now dropping screen resolution column
laptop_data.drop(columns='ScreenResolution',axis=1,inplace=True)
```

In [250]:
```
laptop_data.sample(5)
```

Out[250]:

	Company	TypeName	Inches	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchs
64	HP	Notebook	15.6	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	Windows 10	1.86	36763.2000	
907	Acer	Notebook	15.6	Intel Celeron Dual Core 3205U 1.5GHz	4	16GB Flash Storage	Intel HD Graphics	Chrome OS	2.20	19127.5200	
743	Lenovo	2 in 1 Convertible	14.0	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	Windows 10	1.70	53226.7200	
701	Lenovo	Notebook	15.6	AMD A9-Series 9420 2.9GHz	4	256GB SSD	AMD Radeon 530	Windows 10	2.20	21258.7200	
692	HP	Workstation	17.3	Intel Core i7 7700HQ 2.8GHz	8	500GB HDD	Nvidia Quadro M1200	Windows 10	3.14	101657.7072	

In [251]:
```
#now we have ppi column so we are dropping 'inches' 'x_res'and 'y_res'
laptop_data.drop(columns='Inches',axis=1,inplace=True)
laptop_data.drop(columns='x_res',axis=1,inplace=True)
laptop_data.drop(columns='y_res',axis=1,inplace=True)
```

In [252]:
```
laptop_data.sample()
```

Out[252]:

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	IPS
329	Dell	Notebook	Intel Core i7 7700HQ 2.8GHz	32	1TB SSD	Nvidia GeForce GTX 1050	Windows 10	2.06	140605.92	1	0

In [253]:
```
laptop_data['Cpu'].value_counts()
```

Out[253]:
```
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
```

In [254]:
```
laptop_data['Cpu name']=laptop_data['Cpu'].apply(lambda x:" ".join(x.split()[0:3]))
```

In [255]:
```
laptop_data.head()
```

Out[255]:

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	IPS	
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	2
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	1
2	HP	Notebook	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	1
3	Apple	Ultrabook	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	2
4	Apple	Ultrabook	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	2

In [256]:
```
#making a function
def fetch_processor(text):
    if text=='Intel Core i7' or text=='Intel Core i5' or text=='Intel Core i3':
        return text
    else:
        if text==text.split()[0]=='Intel':
            return 'other intel processor'
        else:
            return 'AMD processor'
```

In [257]:
```
laptop_data['Cpu brand'] = laptop_data['Cpu name'].apply(fetch_processor)
```

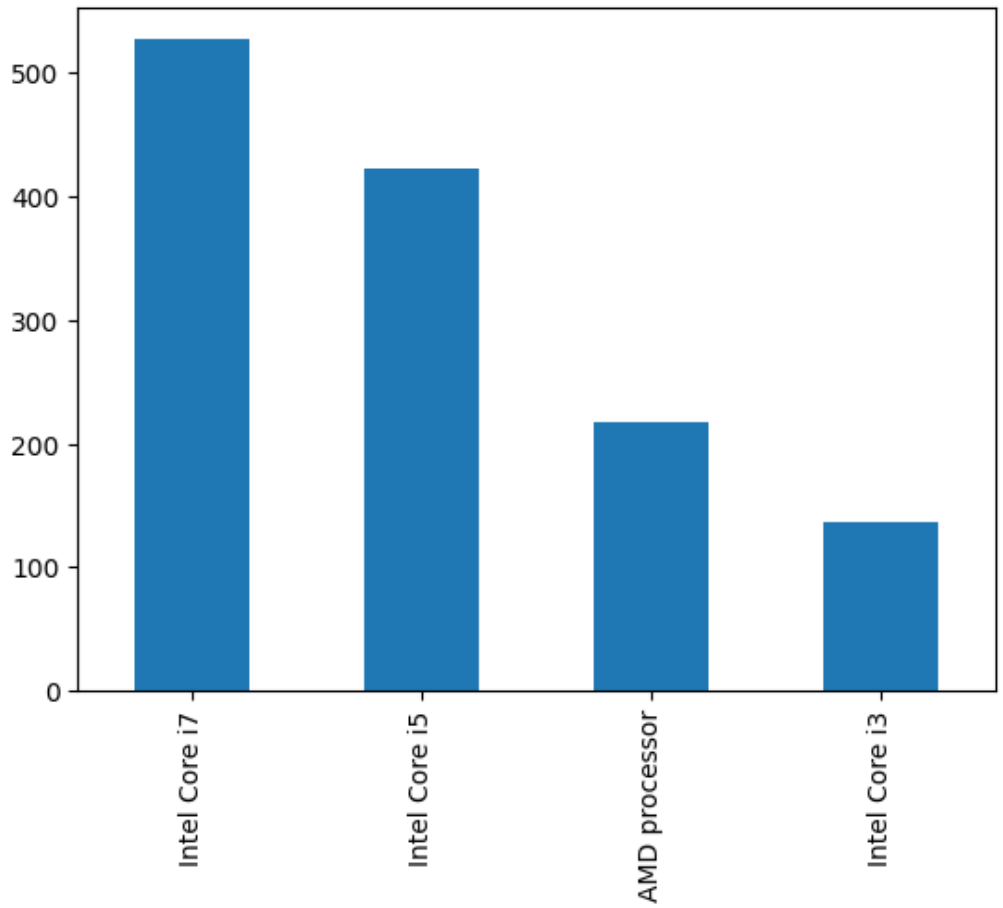
In [258]:
```
laptop_data.sample()
```

Out[258]:

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	IPS
1243	Dell	2 in 1 Convertible	Intel Core i7 7500U 2.7GHz	16	512GB SSD	Nvidia GeForce 940MX	Windows 10	2.77	95850.72	1	0

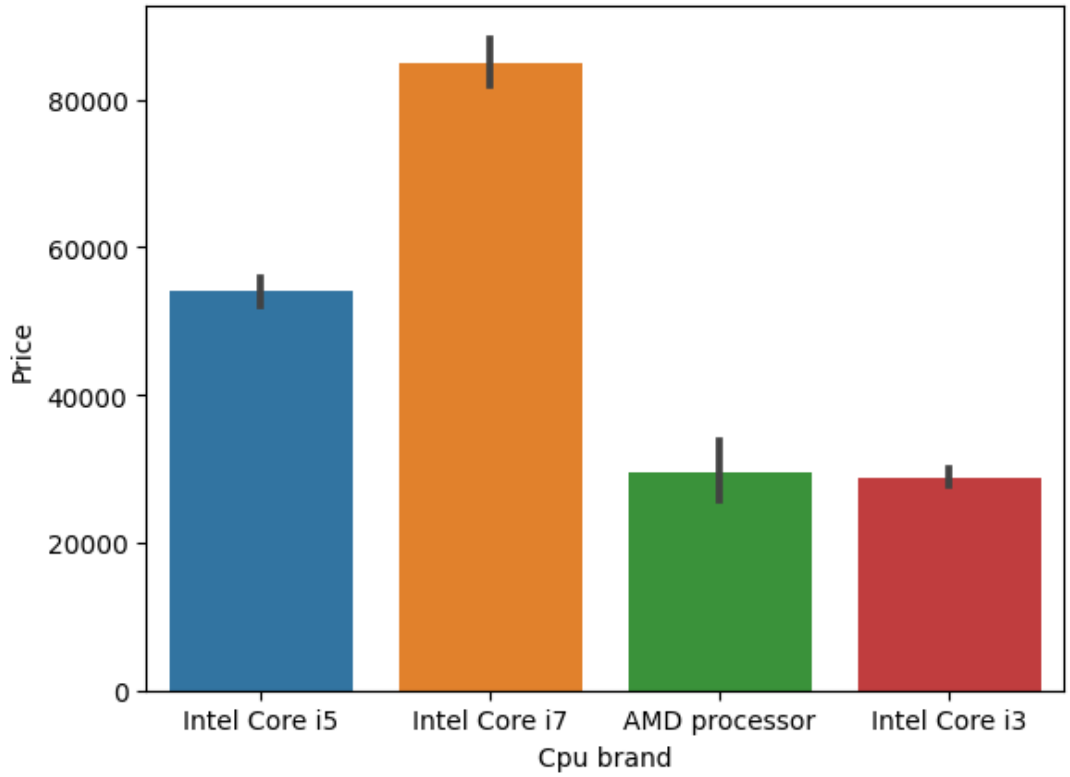
In [259]:
```
laptop_data['Cpu brand'].value_counts().plot(kind='bar')
```

Out[259]: <Axes: >



In [260]:
```
sns.barplot(data=laptop_data,x='Cpu brand',y='Price')
```

Out[260]:
```
<Axes: xlabel='Cpu brand', ylabel='Price'>
```



In [261]:
```
#dropping the columns 'cpu'and 'cpu name'
laptop_data.drop(columns='Cpu',axis=1,inplace=True)
laptop_data.drop(columns='Cpu name',axis=1,inplace=True)
```

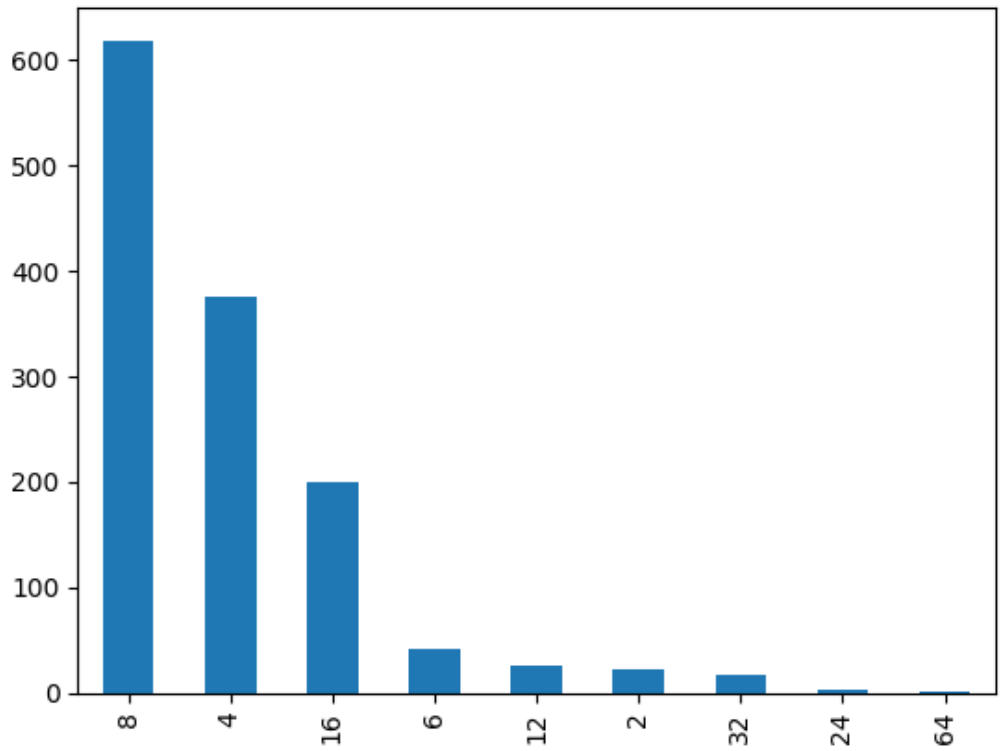
In [262]:
```
laptop_data.head()
```

Out[262]:

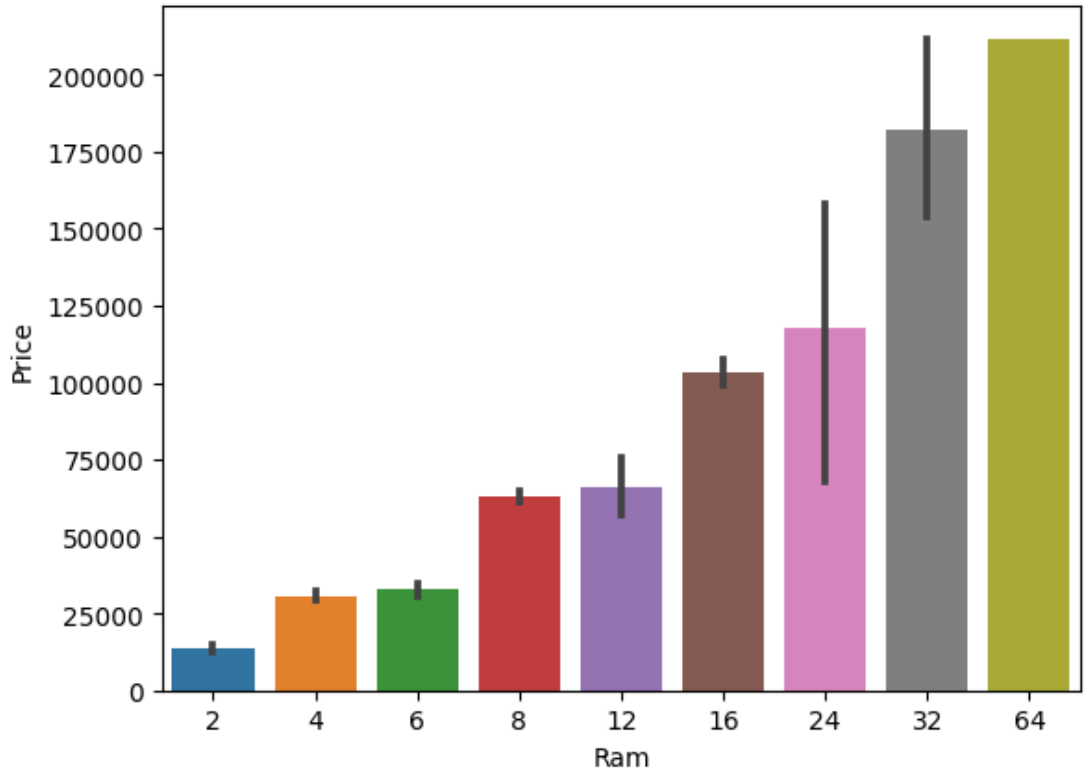
	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	IPS	pp
0	Apple	Ultrabook	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.98300
1	Apple	Ultrabook	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.67794
2	HP	Notebook	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.21199
3	Apple	Ultrabook	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.53462
4	Apple	Ultrabook	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.98300

In [263]:
```
laptop_data['Ram'].value_counts().plot(kind='bar')
```

Out[263]: <Axes: >



In [264]:
```
sns.barplot(data=laptop_data,x='Ram',y='Price')
plt.show()
```



In [265]:
```
#to short the name of dataframe changing name as 'df'
def change_dataset_name(df):
    df.rename(columns={"laptop_data": "df"}, inplace=True)
    return df

df = change_dataset_name(laptop_data.copy())
```

In [266]:
```
#making a seprate column for memory
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 (

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_S

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

```
C:\Users\pcc\AppData\Local\Temp\ipykernel_9820\859465599.py:17: FutureWarning: The defa
ult value of regex will change from True to False in a future version.
  df['first'] = df['first'].str.replace(r'\D', '')
C:\Users\pcc\AppData\Local\Temp\ipykernel_9820\859465599.py:26: FutureWarning: The defa
ult value of regex will change from True to False in a future version.
  df['second'] = df['second'].str.replace(r'\D', '')
```

In [267]:
```
df.head()
```

Out[267]:

	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	IPS	pp
0	Apple	Ultrabook	8	128 SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.98300
1	Apple	Ultrabook	8	128 Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.67794
2	HP	Notebook	8	256 SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.21199
3	Apple	Ultrabook	16	512 SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.53462
4	Apple	Ultrabook	8	256 SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.98300

In [268]:
```
#dropping memory column
df.drop(columns='Memory',axis=1,inplace=True)
```

In [269]:
```
df.corr()['Price']
```

```
C:\Users\pcc\AppData\Local\Temp\ipykernel_9820\815546952.py:1: FutureWarning: The defau
lt 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 sil
ence this warning.
  df.corr()['Price']
```

Out[269]:
```
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
```

In [270]:
```
#dropping hybrid and Flash_storage
df.drop(columns=['Hybrid','Flash_Storage'],axis=1,inplace=True)
```

In [271]:
```
df.head()
```

Out[271]:

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	Touchscreen	IPS	ppi	Cpu brand
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5
1	Apple	Ultrabook	8	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5
2	HP	Notebook	8	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5
3	Apple	Ultrabook	16	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5

In [272]:
```
df['Gpu'].value_counts()
```

Out[272]:
```
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
```

In [277]:
```
df['Brand']=df['Gpu'].apply(lambda x:x.split()[0])
```

In [279]:
```
df['Brand'].value_counts()
```

Out[279]:
```
Intel     722
Nvidia    400
AMD       180
ARM         1
Name: Brand, dtype: int64
```

In [280]:
```
#dropping a row ARM
df[df['Brand']=='ARM']
```

Out[280]:

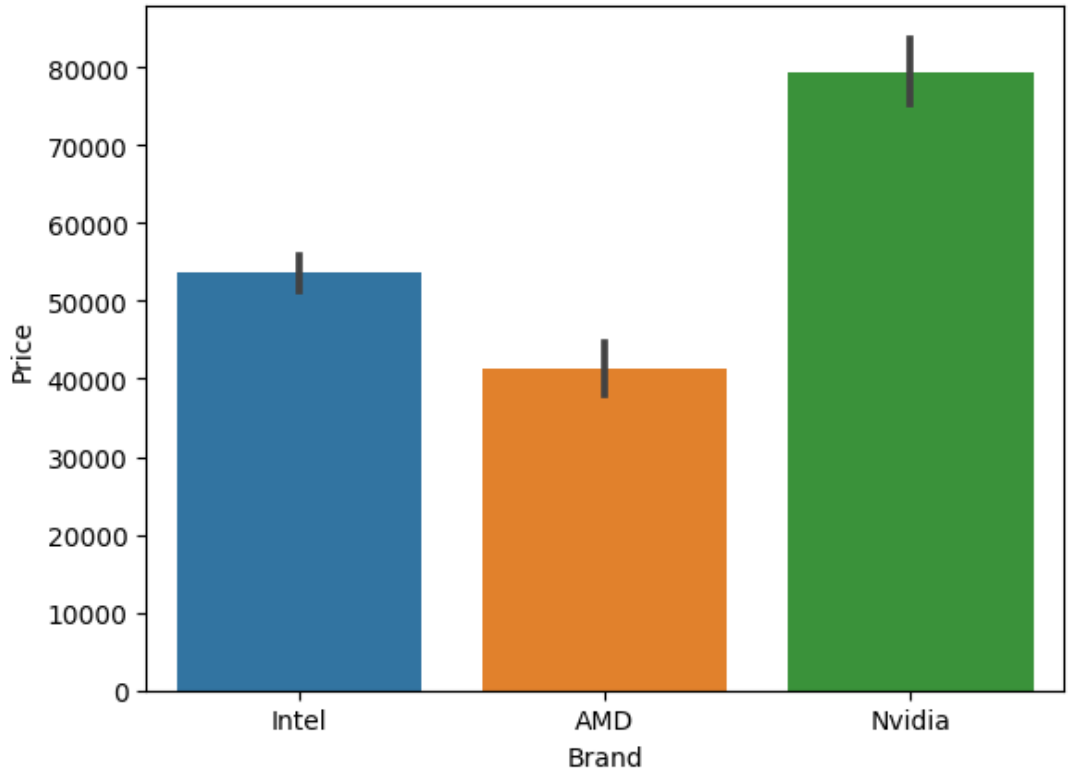
	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	Touchscreen	IPS	ppi	Cpu brand	H
1191	Samsung	2 in 1 Convertible	4	ARM Mali T860 MP4	Chrome OS	1.15	35111.52	1	1	234.5074	AMD processor	

In [281]:
```
df=df[df['Brand']!='ARM']
```

In [282]:
```
df['Brand'].value_counts()
```

Out[282]:
```
Intel     722
Nvidia    400
AMD       180
Name: Brand, dtype: int64
```

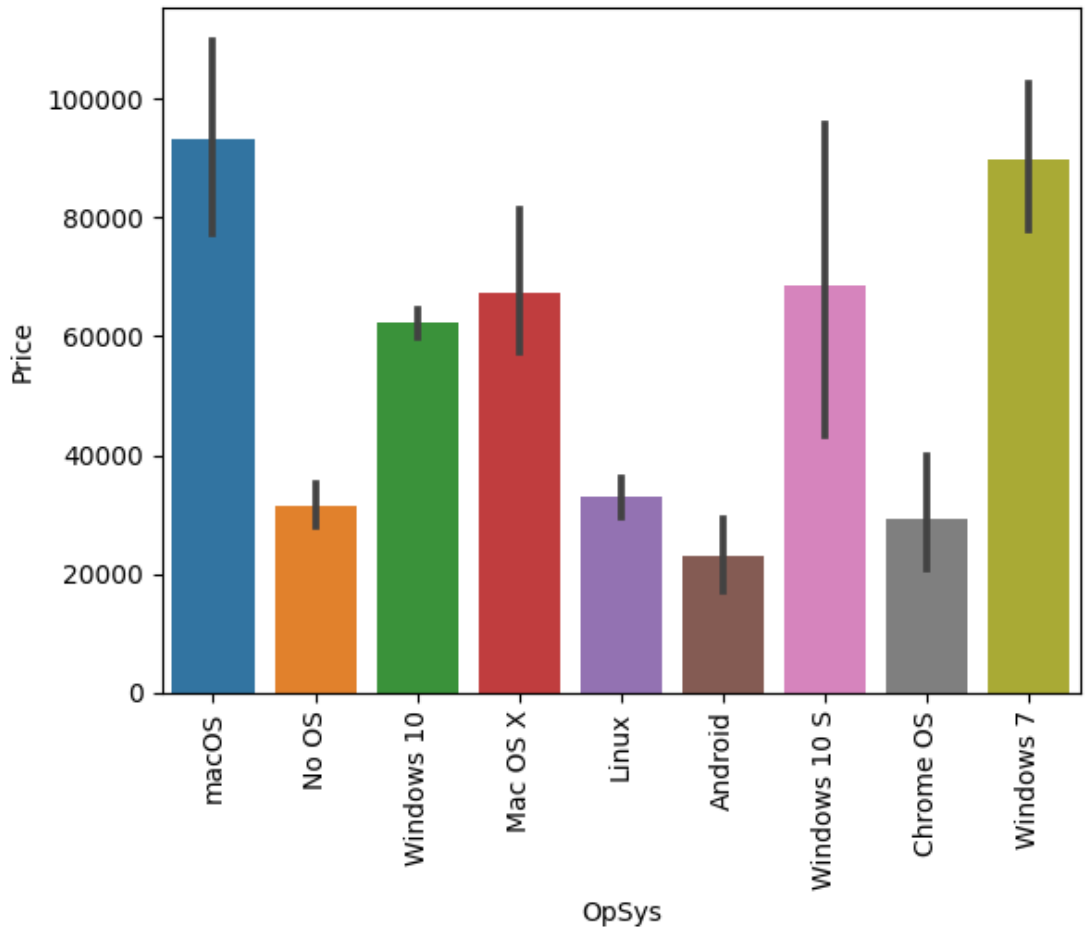
In [283]:
```
sns.barplot(data=df,x='Brand',y='Price')
plt.show()
```



In [286]:
```
df['OpSys'].value_counts()
```

Out[286]:
```
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
```

In [285]:
```
sns.barplot(data=df,x='OpSys',y='Price')
plt.xticks(rotation='vertical')
plt.show()
```



In [287]:
```
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 'other/no OS/Linux'
```

In [288]:
```
df['OS']=df['OpSys'].apply(cat_os)
```

```
C:\Users\pcc\AppData\Local\Temp\ipykernel_9820\1995948337.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['OS']=df['OpSys'].apply(cat_os)
```

In [289]:
```
df.sample()
```

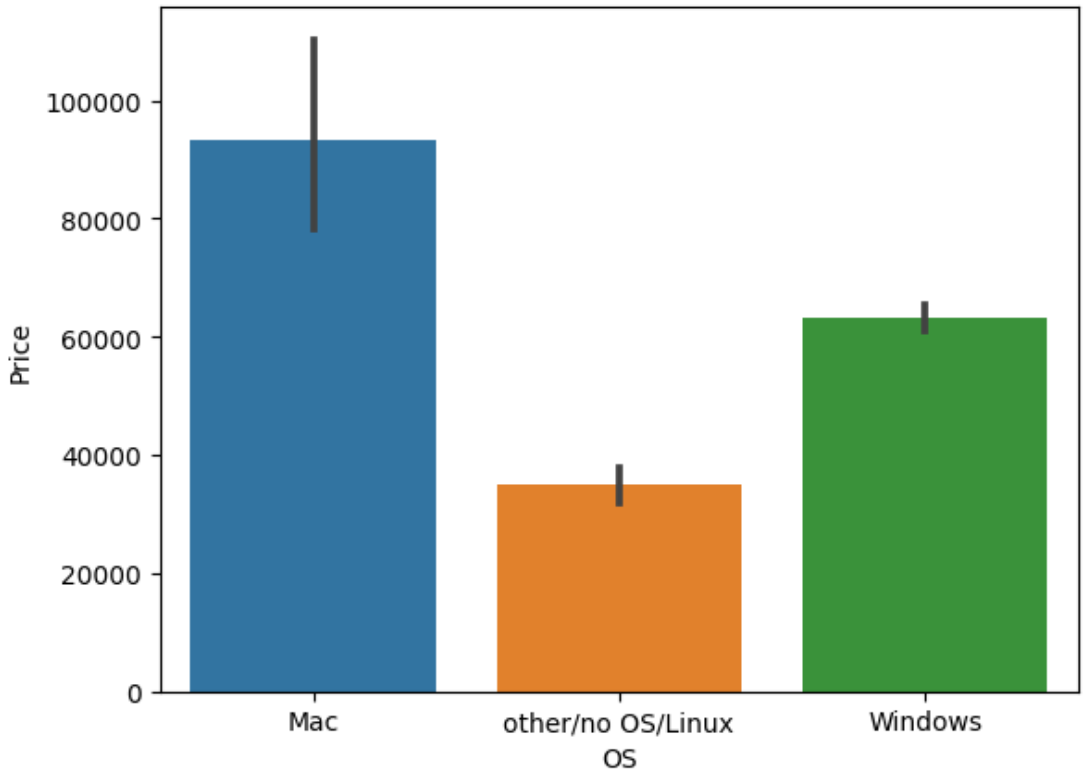
Out[289]:

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	Touchscreen	IPS	ppi	Cpu brand
1194	Dell	2 in 1 Convertible	8	Intel HD Graphics 620	Windows 10	1.6	63882.72	1	1	165.632118	Intel Core i5

In [290]:
```
#dropping ossys column
df.drop(columns="OpSys",axis=1,inplace=True)
```

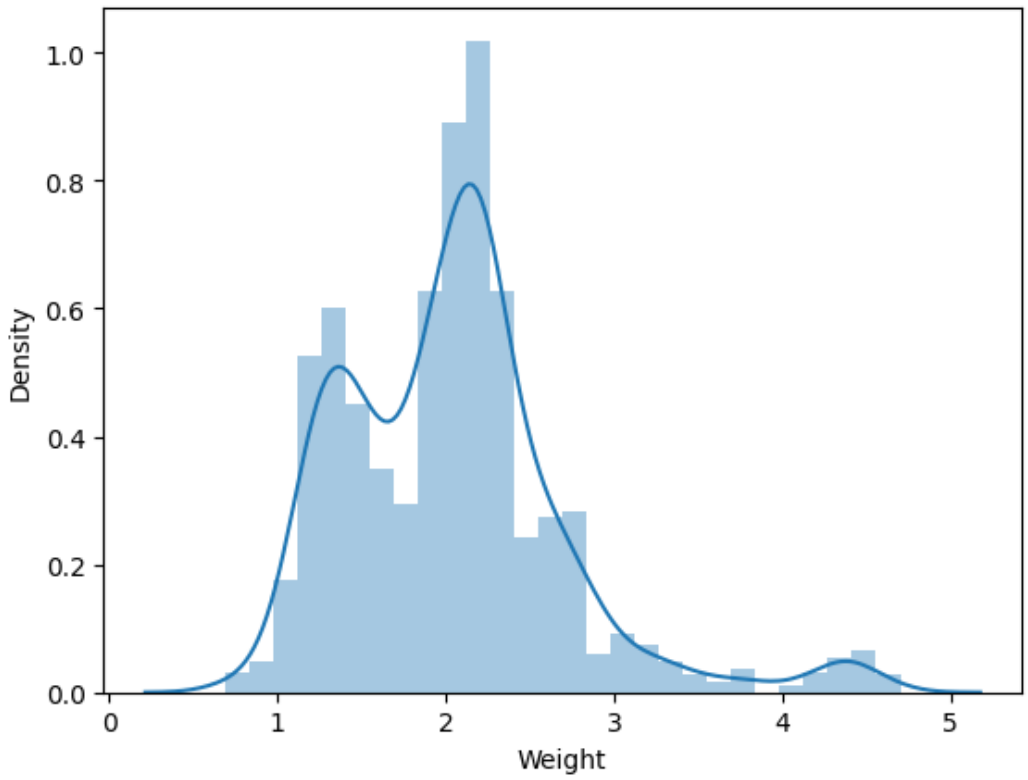
```
C:\Users\pcc\AppData\Local\Temp\ipykernel_9820\387713929.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.drop(columns="OpSys",axis=1,inplace=True)
```

In [291]:
```
sns.barplot(data=df,x='OS',y='Price')
plt.show()
```

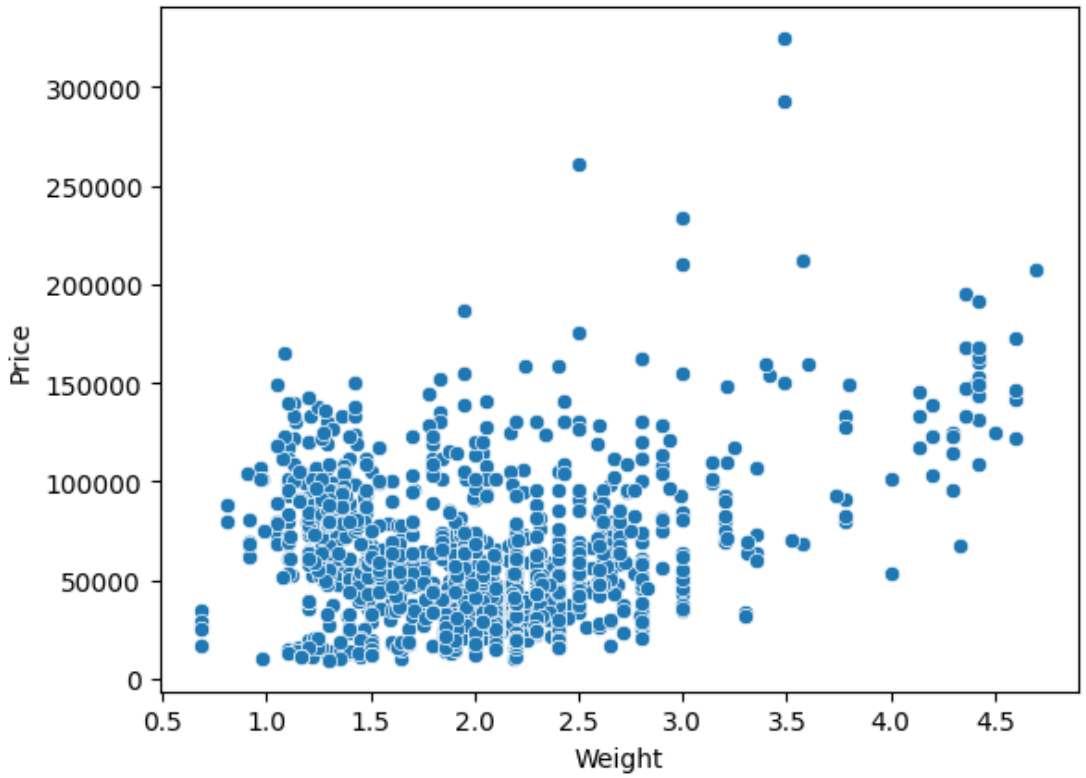


In [292]:
```
sns.distplot(df['Weight'])
plt.show()
```

```
C:\Users\pcc\AppData\Local\Temp\ipykernel_9820\2601973532.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 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

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

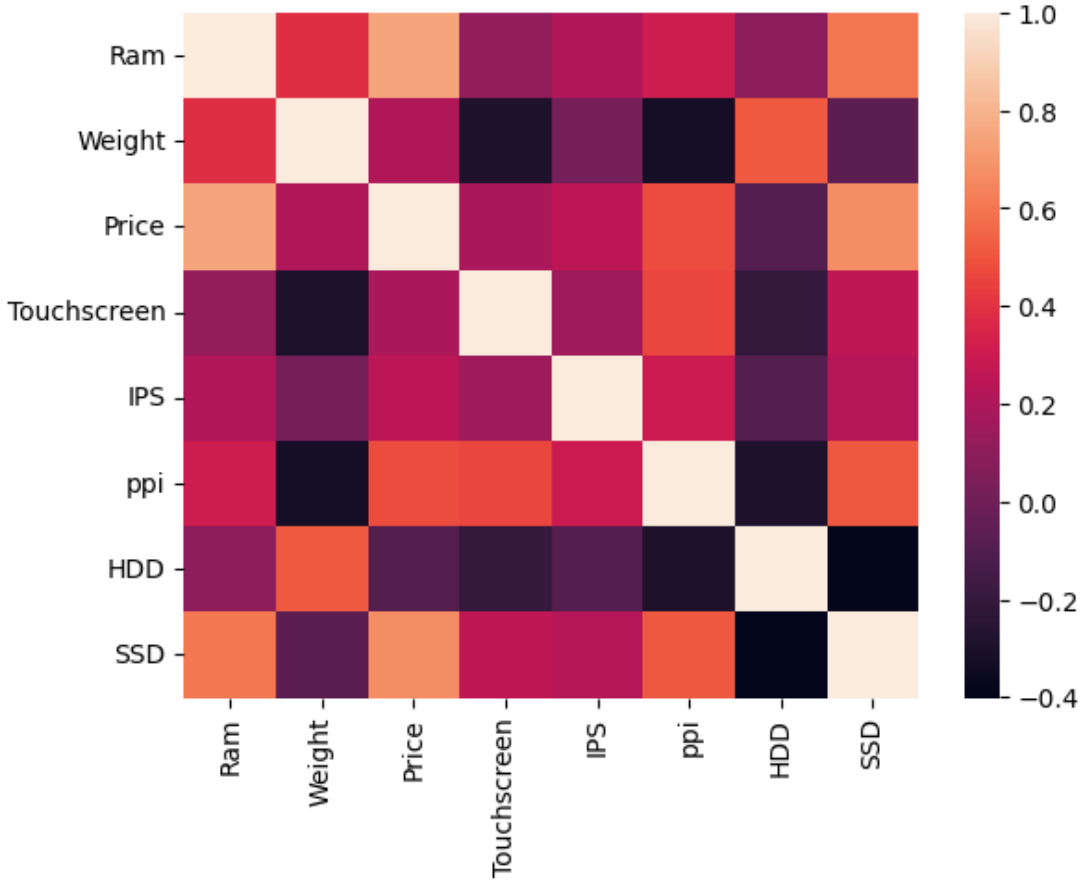


In [293]:
```
sns.scatterplot(data=df,x='Weight',y='Price')
plt.show()
```



In [294]:
```
sns.heatmap(df.corr())
plt.show()
```

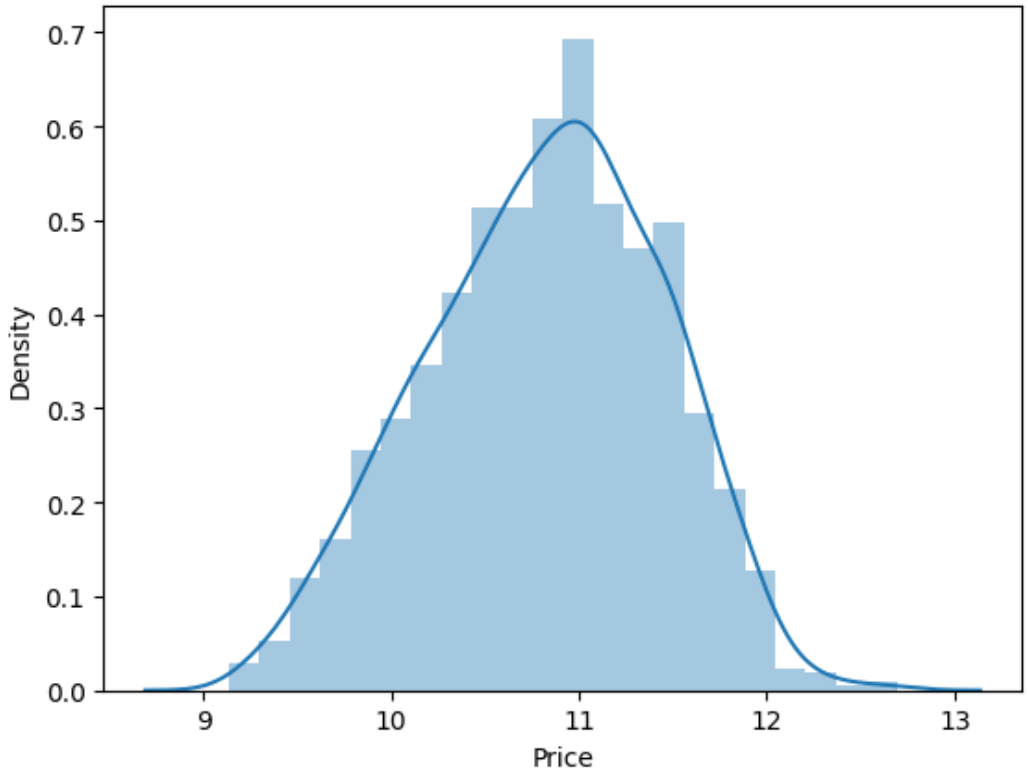
```
C:\Users\pcc\AppData\Local\Temp\ipykernel_9820\2975651719.py:1: FutureWarning: The defa
ult value of numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only to sil
ence this warning.
  sns.heatmap(df.corr())
```



In [295]:
```
sns.distplot(np.log(df['Price']))
```

```
C:\Users\pcc\AppData\Local\Temp\ipykernel_9820\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 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

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

Out[295]: <Axes: xlabel='Price', ylabel='Density'>



In [297]:
```
df.sample()
```

Out[297]:

	Company	TypeName	Ram	Gpu	Weight	Price	Touchscreen	IPS	ppi	Cpu brand	HDD	SSD
339	Dell	Notebook	8	Nvidia GeForce 940MX	2.0	60885.72	0	0	141.211998	Intel Core i5	0	256

In [298]:
```
df.drop(columns='Gpu',axis=1,inplace=True)
```

```
C:\Users\pcc\AppData\Local\Temp\ipykernel_9820\1941717793.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.drop(columns='Gpu',axis=1,inplace=True)
```

In [299]:
```
df
```

Out[299]:

	Company	TypeName	Ram	Weight	Price	Touchscreen	IPS	ppi	Cpu brand	HDD	SSD
0	Apple	Ultrabook	8	1.37	71378.6832	0	1	226.983005	Intel Core i5	0	128
1	Apple	Ultrabook	8	1.34	47895.5232	0	0	127.677940	Intel Core i5	0	0
2	HP	Notebook	8	1.86	30636.0000	0	0	141.211998	Intel Core i5	0	256
3	Apple	Ultrabook	16	1.83	135195.3360	0	1	220.534624	Intel Core i7	0	512
4	Apple	Ultrabook	8	1.37	96095.8080	0	1	226.983005	Intel Core i5	0	256
...	...	...	...	...	...	...	...	...	...	...	...
1298	Lenovo	2 in 1 Convertible	4	1.80	33992.6400	1	1	157.350512	Intel Core i7	0	128
1299	Lenovo	2 in 1 Convertible	16	1.30	79866.7200	1	1	276.053530	Intel Core i7	0	512
1300	Lenovo	Notebook	2	1.50	12201.1200	0	0	111.935204	AMD processor	0	0
1301	HP	Notebook	6	2.19	40705.9200	0	0	100.454670	Intel Core i7	1000	0
1302	Asus	Notebook	4	2.20	19660.3200	0	0	100.454670	AMD processor	500	0

1302 rows × 13 columns

In [335]:
```
X=df.drop(columns='Price')
y=np.log(df['Price'])
```

In [336]:
```
X
```

Out[336]:

	Company	TypeName	Ram	Weight	Touchscreen	IPS	ppi	Cpu brand	HDD	SSD	Brand	
0	Apple	Ultrabook	8	1.37	0	1	226.983005	Intel Core i5	0	128	Intel	
1	Apple	Ultrabook	8	1.34	0	0	127.677940	Intel Core i5	0	0	Intel	
2	HP	Notebook	8	1.86	0	0	141.211998	Intel Core i5	0	256	Intel	othe OS/L
3	Apple	Ultrabook	16	1.83	0	1	220.534624	Intel Core i7	0	512	AMD	
4	Apple	Ultrabook	8	1.37	0	1	226.983005	Intel Core i5	0	256	Intel	
...	...	...	...	...	...	...	...	...	...	...	...	
1298	Lenovo	2 in 1 Convertible	4	1.80	1	1	157.350512	Intel Core i7	0	128	Intel	Wind
1299	Lenovo	2 in 1 Convertible	16	1.30	1	1	276.053530	Intel Core i7	0	512	Intel	Wind
1300	Lenovo	Notebook	2	1.50	0	0	111.935204	AMD processor	0	0	Intel	Wind
1301	HP	Notebook	6	2.19	0	0	100.454670	Intel Core i7	1000	0	AMD	Wind
1302	Asus	Notebook	4	2.20	0	0	100.454670	AMD processor	500	0	Intel	Wind

1302 rows × 12 columns

In [337]:
```
y
```

Out[337]:
```
0       11.175755
1       10.776777
2       10.329931
3       11.814476
4       11.473101
           ...
1298    10.433899
1299    11.288115
1300     9.409283
1301    10.614129
1302     9.886358
Name: Price, Length: 1302, dtype: float64
```

In [338]:
```
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)
```

In [318]:
```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import r2_score,mean_absolute_error
```

```
from sklearn.linear_model import LinearRegression,Ridge,Lasso
from sklearn.neighbors import KNeighborsRegressor
```

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import
RandomForestRegressor,GradientBoostingRegressor,AdaBoostRegressor,ExtraTreesRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
```

```
LinearRegression
```

In [339]:
```
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],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('MAE',mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.8123578982819424
MAE 0.20576251545729152

C:\Users\pcc\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preproce
ssing\_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in versi
on 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
```

In [340]:
```
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],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('MAE',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\pcc\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preproce
ssing\_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in versi
on 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 0.81769309396726
MAE 0.2064526184836341
```

Lasso Regression

In [341]:
```
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],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('MAE',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\pcc\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preproce
ssing\_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in versi
on 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 0.8129782173452689
MAE 0.2087779158884776
```

KNN

In [342]:
```
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],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('MAE',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\pcc\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preproce
ssing\_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in versi
on 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 0.8044214081114575
MAE 0.19358319789713704
```

Decision Tree

In [343]:
```
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],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('MAE',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\pcc\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preproce
ssing\_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in versi
on 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 0.8410081494116903
MAE 0.18069737762826088
```

SVM

In [344]:
```
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],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('MAE',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\pcc\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preproce
ssing\_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in versi
on 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 0.8074275799094983
MAE 0.20198742683407786
```

```
Random Forest
```

In [345]:
```
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],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('MAE',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\pcc\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preproce
ssing\_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in versi
on 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 0.8898542259695731
MAE 0.15605695145182924
```

AdaBoost

In [347]:
```
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],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('MAE',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\pcc\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preproce
ssing\_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in versi
on 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 0.7683213752929258
MAE 0.23837526102445517
```

Gradient Boost

In [348]:
```
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],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('MAE',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\pcc\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preproce
ssing\_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in versi
on 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 0.8928907613814371
MAE 0.15242185134253308
```

XgBoost

In [349]:
```
step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],remainder='passthrough')

step2 = XGBRegressor(n_estimators=45,max_depth=5,learning_rate=0.5)

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('MAE',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\pcc\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preproce
ssing\_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in versi
on 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 0.8857545759386917
MAE 0.15748483630439358
```

Stacking

In [351]:
```
from sklearn.ensemble import VotingRegressor,StackingRegressor

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


estimators = [
    ('rf', RandomForestRegressor(n_estimators=350,random_state=3,max_samples=0.5,max_fea
    ('gbdt',GradientBoostingRegressor(n_estimators=100,max_features=0.5)),
    ('xgb', XGBRegressor(n_estimators=25,learning_rate=0.3,max_depth=5))
]

step2 = StackingRegressor(estimators=estimators, final_estimator=Ridge(alpha=100))

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('MAE',mean_absolute_error(y_test,y_pred))
```

```
C:\Users\pcc\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preproce
ssing\_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in versi
on 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 0.8839042094705434
MAE 0.16685472940695725
```

Exporting the Mode

In [352]:
```
import pickle

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

In [353]:
```
df
```

Out[353]:

	Company	TypeName	Ram	Weight	Price	Touchscreen	IPS	ppi	Cpu brand	HDD	SSD
0	Apple	Ultrabook	8	1.37	71378.6832	0	1	226.983005	Intel Core i5	0	128
1	Apple	Ultrabook	8	1.34	47895.5232	0	0	127.677940	Intel Core i5	0	0
2	HP	Notebook	8	1.86	30636.0000	0	0	141.211998	Intel Core i5	0	256
3	Apple	Ultrabook	16	1.83	135195.3360	0	1	220.534624	Intel Core i7	0	512
4	Apple	Ultrabook	8	1.37	96095.8080	0	1	226.983005	Intel Core i5	0	256
...	...	...	...	...	...	...	...	...	...	...	...
1298	Lenovo	2 in 1 Convertible	4	1.80	33992.6400	1	1	157.350512	Intel Core i7	0	128
1299	Lenovo	2 in 1 Convertible	16	1.30	79866.7200	1	1	276.053530	Intel Core i7	0	512
1300	Lenovo	Notebook	2	1.50	12201.1200	0	0	111.935204	AMD processor	0	0
1301	HP	Notebook	6	2.19	40705.9200	0	0	100.454670	Intel Core i7	1000	0
1302	Asus	Notebook	4	2.20	19660.3200	0	0	100.454670	AMD processor	500	0

1302 rows × 13 columns