In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv('laptop_data (1).csv')
```

In [3]:
```
df.head()
```

Out[3]:

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

In [4]:
```
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
```

In [5]:
```
df.describe()
```

Out[5]:

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

In [6]:
```
df.shape
```

Out[6]:
```
(1303, 12)
```

In [7]:
```
df['Company'].unique()
```

Out[7]:
```
array(['Apple', 'HP', 'Acer', 'Asus', 'Dell', 'Lenovo', 'Chuwi', 'MSI',
       'Microsoft', 'Toshiba', 'Huawei', 'Xiaomi', 'Vero', 'Razer',
       'Mediacom', 'Samsung', 'Google', 'Fujitsu', 'LG'], dtype=object)
```

In [8]:
```
df['TypeName'].unique()
```

Out[8]:
```
array(['Ultrabook', 'Notebook', 'Netbook', 'Gaming', '2 in 1 Convertible',
       'Workstation'], dtype=object)
```

In [9]:
```
df['Memory'].unique() # needs separation
```

Out[9]:
```
array(['128GB SSD', '128GB Flash Storage', '256GB SSD', '512GB SSD',
       '500GB HDD', '256GB Flash Storage', '1TB HDD',
       '32GB Flash Storage', '128GB SSD +  1TB HDD',
       '256GB SSD +  256GB SSD', '64GB Flash Storage',
       '256GB SSD +  1TB HDD', '256GB SSD +  2TB HDD', '32GB SSD',
       '2TB HDD', '64GB SSD', '1.0TB Hybrid', '512GB SSD +  1TB HDD',
       '1TB SSD', '256GB SSD +  500GB HDD', '128GB SSD +  2TB HDD',
       '512GB SSD +  512GB SSD', '16GB SSD', '16GB Flash Storage',
       '512GB SSD +  256GB SSD', '512GB SSD +  2TB HDD',
       '64GB Flash Storage +  1TB HDD', '180GB SSD', '1TB HDD +  1TB HDD',
       '32GB HDD', '1TB SSD +  1TB HDD', '512GB Flash Storage',
       '128GB HDD', '240GB SSD', '8GB SSD', '508GB Hybrid', '1.0TB HDD',
       '512GB SSD +  1.0TB Hybrid', '256GB SSD +  1.0TB Hybrid'],
      dtype=object)
```

In [10]:
```
df.duplicated().sum()
```

Out[10]:
```
0
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
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 [12]:
```
df.drop(columns=['Unnamed: 0'],inplace=True)
```

In [13]:
```
df.head()
```

Out[13]:

	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
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg
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
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
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

In [14]:
```
# Ram has got an int value but is a srtring with the usage of GB, that has to be separ
df['Ram']=df['Ram'].str.replace('GB','')
df['Weight']=df['Weight'].str.replace('kg','')
```

In [15]:
```
df.head()
```

Out[15]:

	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	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37

In [16]:
```
df['Ram']=df['Ram'].astype('int32')
df['Weight']=df['Weight'].astype('float32')
```

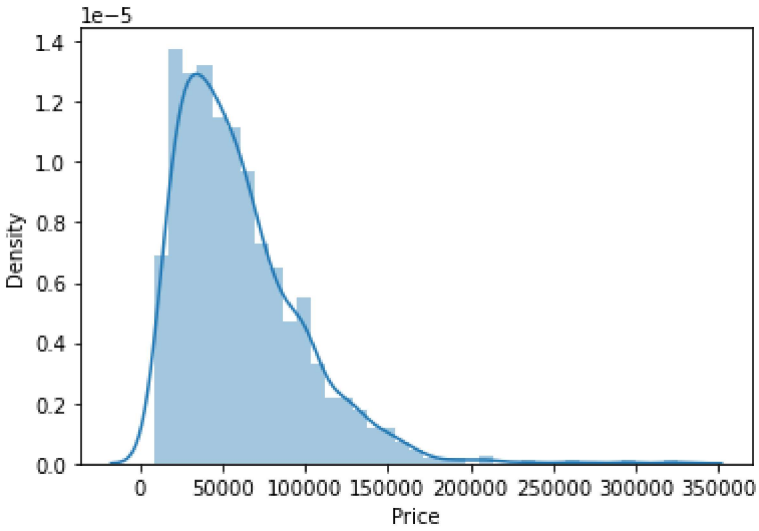
In [17]:
```
df.info()
```

```
<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
```

In [18]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.distplot(df['Price']) # left skewed a little
```

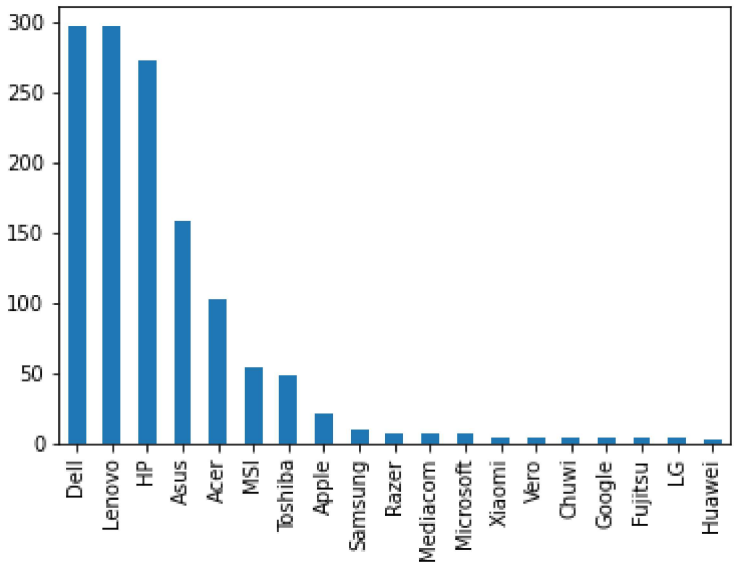
```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[18]: <AxesSubplot:xlabel='Price', ylabel='Density'>



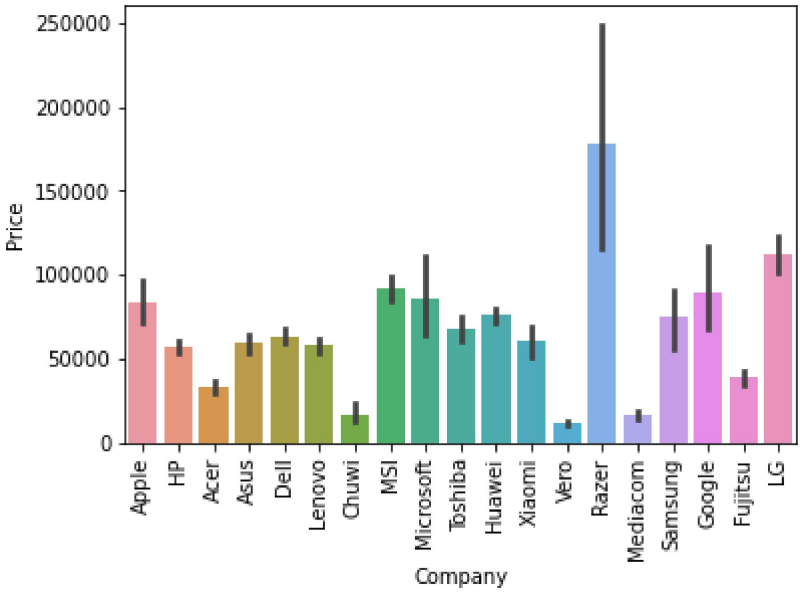
In [19]:
```
df['Company'].value_counts().plot(kind='bar')
```

Out[19]: <AxesSubplot:>



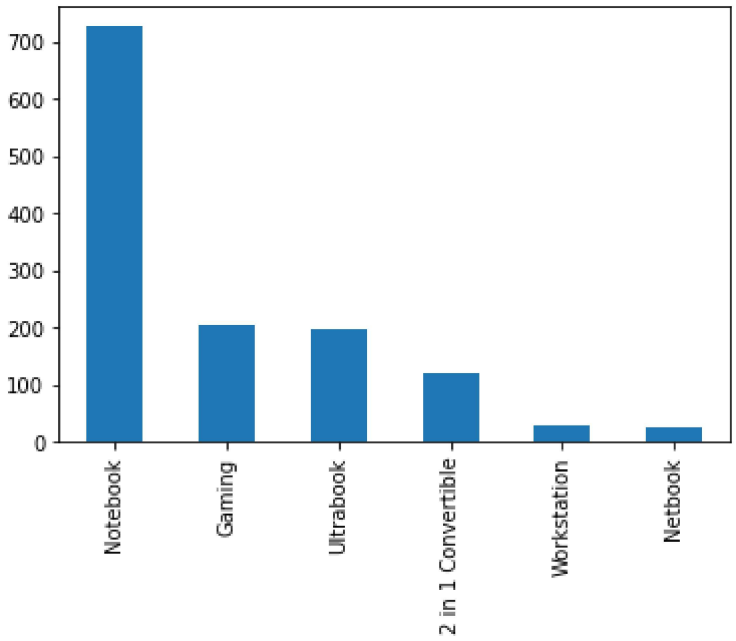
In [20]:
```
sns.barplot(df['Company'],df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



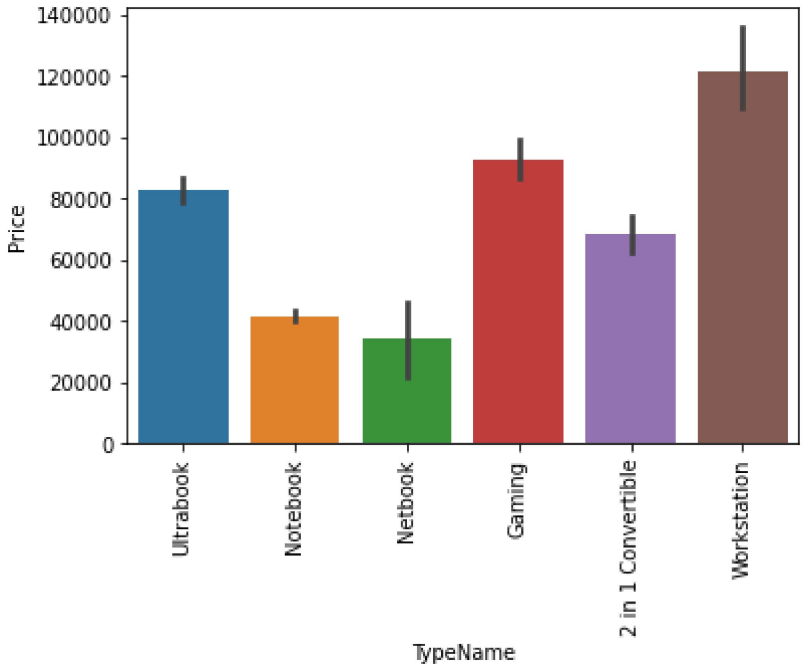
In [21]:
```
df['TypeName'].value_counts().plot(kind='bar')
```

Out[21]:
```
<AxesSubplot:>
```



In [22]:
```
sns.barplot(df['TypeName'],df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```

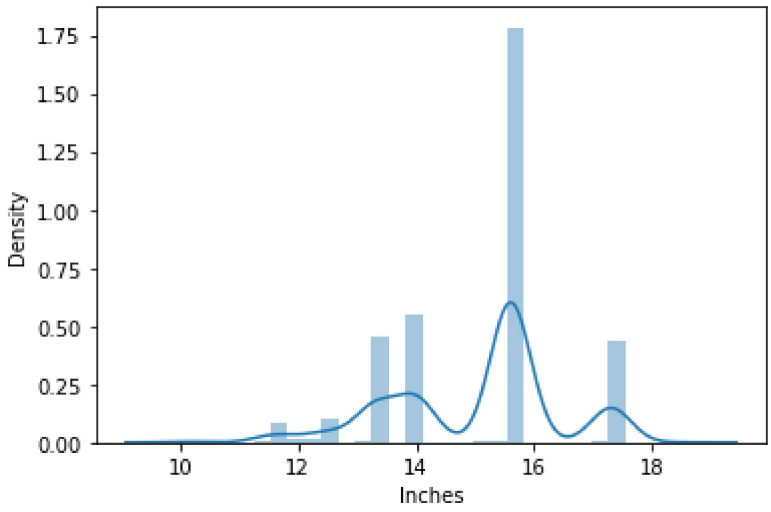
```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



In [23]:
```
sns.distplot(df['Inches'])
```

```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

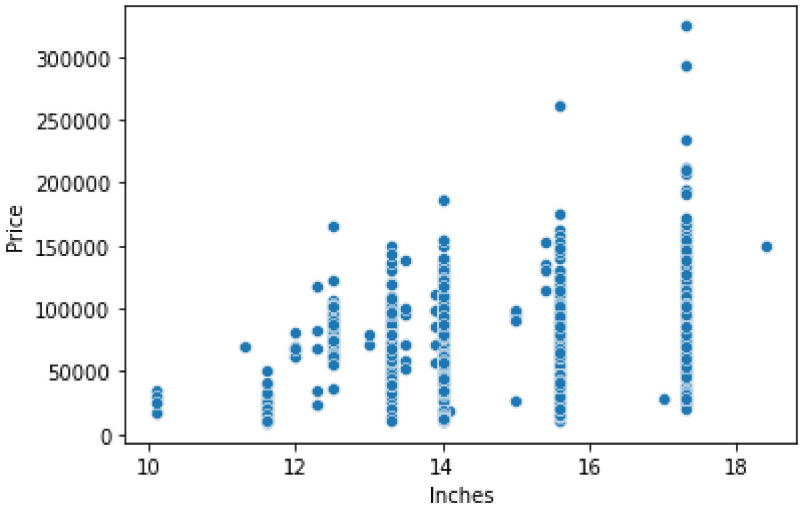
Out[23]:
```
<AxesSubplot:xlabel='Inches', ylabel='Density'>
```



In [24]:
```
sns.scatterplot(df['Inches'],df['Price']) # # not a very strong co relation
```

```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[24]:
```
<AxesSubplot:xlabel='Inches', ylabel='Price'>
```



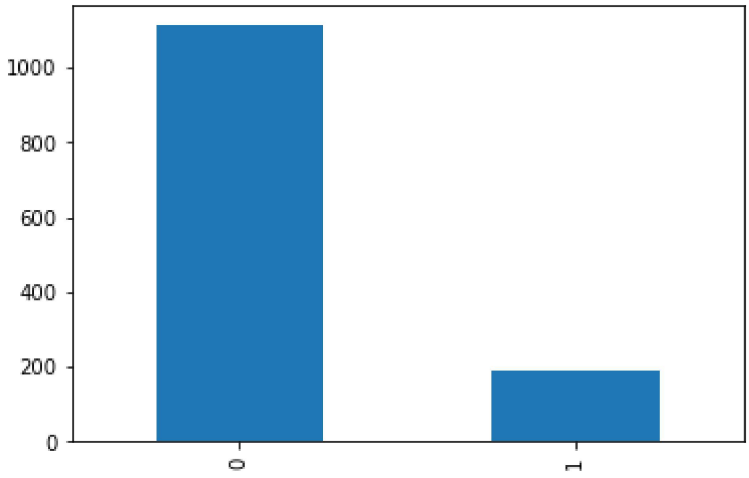
In [25]:
```
df['ScreenResolution'].value_counts()
```

Out[25]:
```
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 [26]:
```
# mostly IPS panel, Touch screen ? this must be analysed
df['Touchscreen']=df['ScreenResolution'].apply(lambda x:1 if 'Touchscreen' in x else 0
```

In [27]:
```
df['Touchscreen'].value_counts().plot(kind='bar')
```

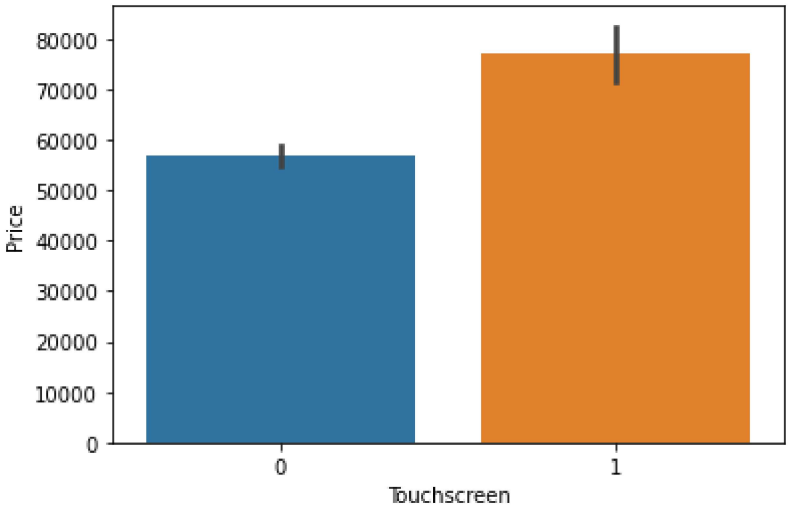
Out[27]:
```
<AxesSubplot:>
```



In [28]:
```
sns.barplot(df['Touchscreen'],df['Price']) #shows touch screeen is costly
```

```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[28]:
```
<AxesSubplot:xlabel='Touchscreen', ylabel='Price'>
```

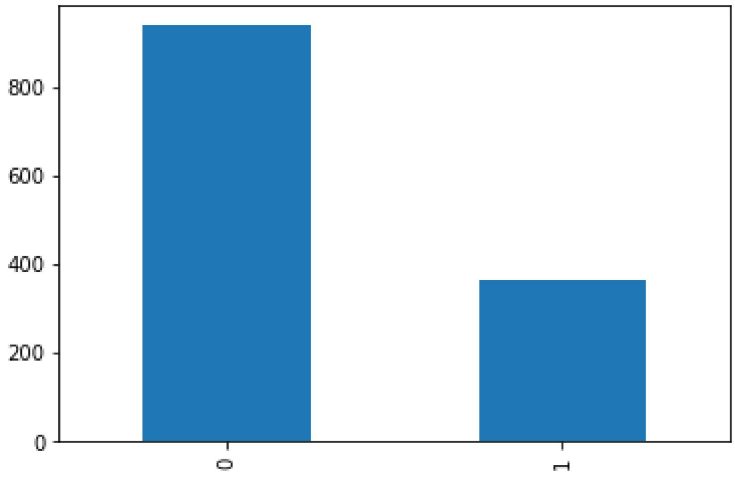


In [29]:
```
df['Ips']=df['ScreenResolution'].apply(lambda x:1 if 'IPS' in x else 0)
df.head()
```

Out[29]:

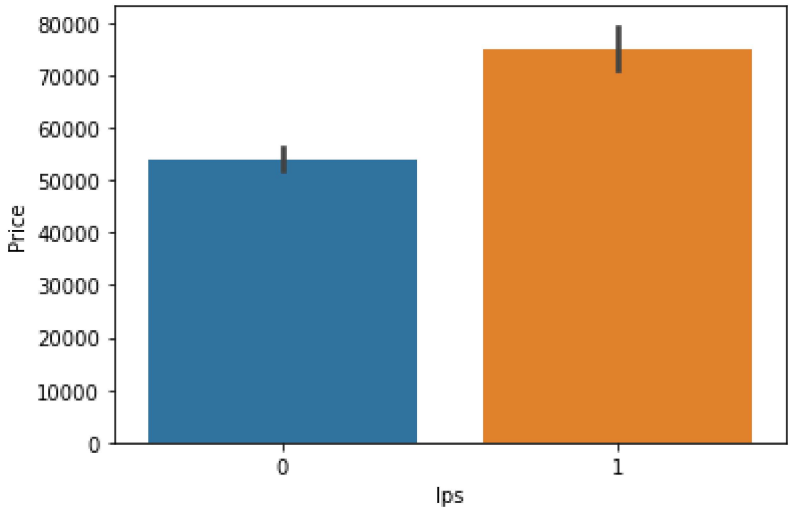
	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000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

In [30]:
```
df['Ips'].value_counts().plot(kind='bar')
plt.show()
sns.barplot(df['Ips'],df['Price']) # most of them are not IPS, if IPS they are costly
```




```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[30]:
```
<AxesSubplot:xlabel='Ips', ylabel='Price'>
```



In [31]:
```
# X and Y resoulution  Eg 3200x1800 - this needs to be extracted

new=df['ScreenResolution'].str.split('x',n=1,expand=True)
df['X_res']=new[0] # but we need just number here what can be done
df['Y_res']=new[1]
```

In [32]:
```
df.head()
```

Out[32]:

	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	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37

In [33]:
```
df['X_res']=df['X_res'].str.replace(',','').str.findall(r'(\d+\.?\d+)').apply(lambda x
```

In [34]:
```
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   object 
 14  Y_res             1303 non-null   object 
dtypes: float32(1), float64(2), int32(1), int64(2), object(9)
memory usage: 142.6+ KB
```

In [35]:
```
df['X_res']=df['X_res'].astype('int32')
df['Y_res']=df['Y_res'].astype('int32')
```

In [36]:
```
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
```

In [37]:
```
df.corr()['Price']
```

Out[37]:
```
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 [38]:
```
# PPI :
df['ppi']=((((df['X_res']**2) + (df['Y_res']**2))**0.5)/df['Inches']).astype('float')
```

In [39]:
```
df.corr()['Price'] # as PPI has a great co relation
```

Out[39]:
```
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 [40]:
```
df.drop(columns=['ScreenResolution','X_res','Y_res'],inplace= True)
```

In [41]:
```
df.head()
```

Out[41]:

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

In [42]:
```
df['Cpu'].value_counts() # at most are intel , so lets divide into categories
```

Out[42]:
```
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 [43]:
```
df['Cpu Name']=df['Cpu'].apply(lambda x:" ".join(x.split()[0:3])) # first 3 works of p
```

In [44]:
```
df.head()
```

Out[44]:

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

In [45]:
```
def fetch_processor(text):
    if text == 'Intel Core i5' or text == "Intel Core i7" or text == 'Intel Core i3':
        return text
    else:
        if text.split()[0]=='Intel':
            return "Other Inter Processor"
        else:
            return "AMD Processor"
```

In [46]:
```
df['Cpu brand']= df['Cpu Name'].apply(fetch_processor)
```

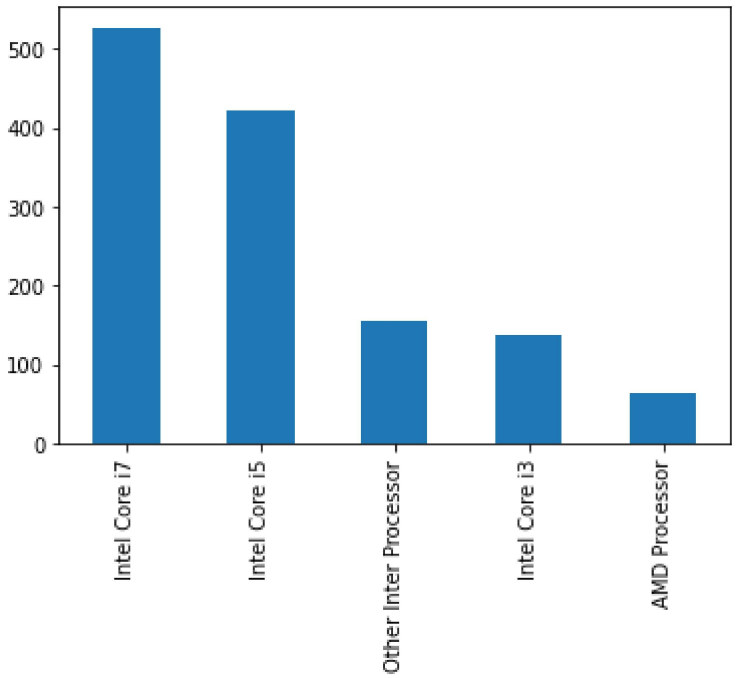
In [47]:
```
df.head()
```

Out[47]:

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

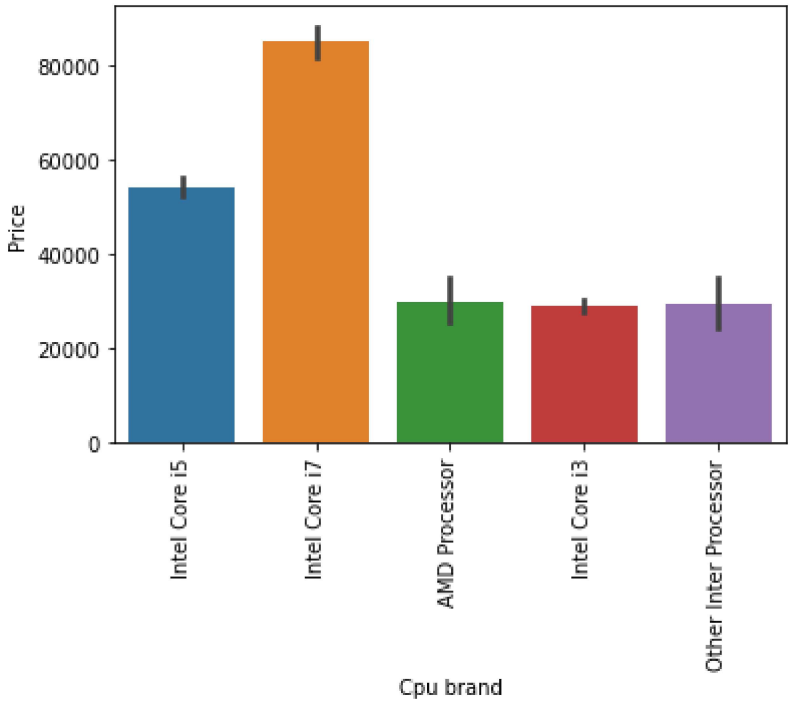
In [48]:
```
df['Cpu brand'].value_counts().plot(kind='bar')
```

Out[48]:
```
<AxesSubplot:>
```



In [49]:
```
sns.barplot(df['Cpu brand'],df['Price'])
plt.xticks(rotation='vertical')
plt.show() # i7 has the most price
```

```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



In [50]:
```
df.drop(columns=['Cpu','Cpu Name'],inplace=True)
```

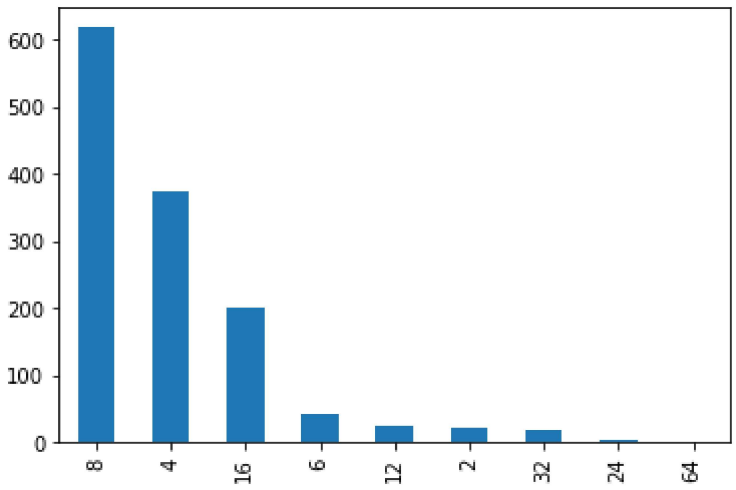
In [51]:
```
df.head()
```

Out[51]:

	Company	TypeName	Inches	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen
0	Apple	Ultrabook	13.3	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0
1	Apple	Ultrabook	13.3	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0
2	HP	Notebook	15.6	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0
3	Apple	Ultrabook	15.4	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0
4	Apple	Ultrabook	13.3	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0

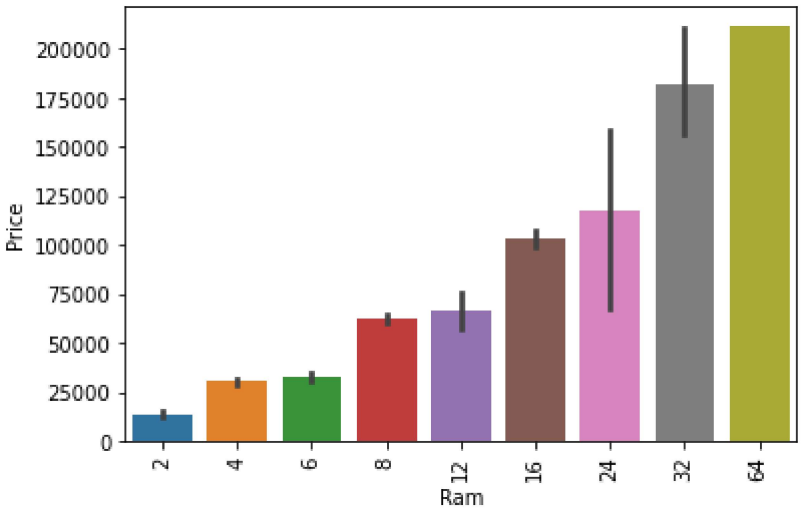
In [52]:
```
df['Ram'].value_counts().plot(kind='bar') # its shows people prefer budget friendly
```

Out[52]: <AxesSubplot:>



In [53]:
```
sns.barplot(df['Ram'],df['Price'])
plt.xticks(rotation='vertical')
plt.show()  # price increases with ram
```

```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [54]:
```
df['Memory'].value_counts() # we can make 4 coumns oiut of these
```

Out[54]:
```
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
```

In [55]:
```
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

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 els

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["Layer2Flas

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

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 els

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["Layer2Flas

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

```
C:\Users\Sindhu\AppData\Local\Temp\ipykernel_1196\211918721.py:16: 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\Sindhu\AppData\Local\Temp\ipykernel_1196\211918721.py:25: FutureWarning: The
default value of regex will change from True to False in a future version.
  df['second'] = df['second'].str.replace(r'\D', '')
C:\Users\Sindhu\AppData\Local\Temp\ipykernel_1196\211918721.py:50: 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\Sindhu\AppData\Local\Temp\ipykernel_1196\211918721.py:59: FutureWarning: The
default value of regex will change from True to False in a future version.
  df['second'] = df['second'].str.replace(r'\D', '')
```

In [56]:
```
df.head(5)
```

Out[56]:

	Company	TypeName	Inches	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen
0	Apple	Ultrabook	13.3	8	128 SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0
1	Apple	Ultrabook	13.3	8	128 Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0
2	HP	Notebook	15.6	8	256 SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0
3	Apple	Ultrabook	15.4	16	512 SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0
4	Apple	Ultrabook	13.3	8	256 SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0

In [57]:
```
df.drop(columns=['Memory'],inplace=True)
df.head()
```

Out[57]:

	Company	TypeName	Inches	Ram	Gpu	OpSys	Weight	Price	Touchscreen	Ips	
0	Apple	Ultrabook	13.3	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.
1	Apple	Ultrabook	13.3	8	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.
2	HP	Notebook	15.6	8	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.
3	Apple	Ultrabook	15.4	16	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.
4	Apple	Ultrabook	13.3	8	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.

In [58]:
```
df.corr()['Price']
```

Out[58]:
```
Inches           0.068197
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 [59]:
```
df.drop(columns=['Hybrid','Flash_Storage'],inplace=True)
```

In [60]:
```
df.head()
```

Out[60]:

	Company	TypeName	Inches	Ram	Gpu	OpSys	Weight	Price	Touchscreen	Ips	
0	Apple	Ultrabook	13.3	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.
1	Apple	Ultrabook	13.3	8	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.
2	HP	Notebook	15.6	8	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.
3	Apple	Ultrabook	15.4	16	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.
4	Apple	Ultrabook	13.3	8	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.

In [61]:
```
# GPU
df['Gpu'].value_counts()
```

Out[61]:
```
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 [62]:
```
df['Gpu brand'] = df['Gpu'].apply(lambda x: x.split()[0])
```

In [63]:
```
df['Gpu brand'].value_counts()
```

Out[63]:
```
Intel     722
Nvidia    400
AMD       180
ARM         1
Name: Gpu brand, dtype: int64
```

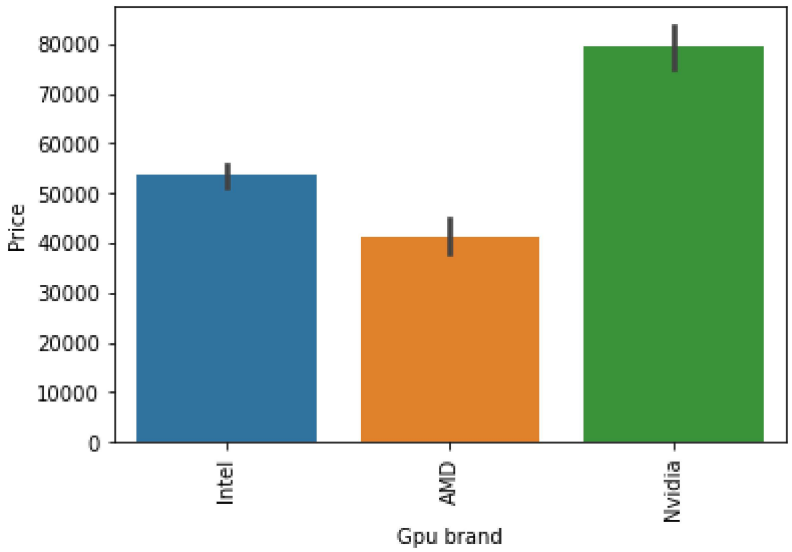
In [64]:
```
df=df[df['Gpu brand']!='ARM']
```

In [65]:
```
df['Gpu brand'].value_counts()
```

Out[65]:
```
Intel     722
Nvidia    400
AMD       180
Name: Gpu brand, dtype: int64
```

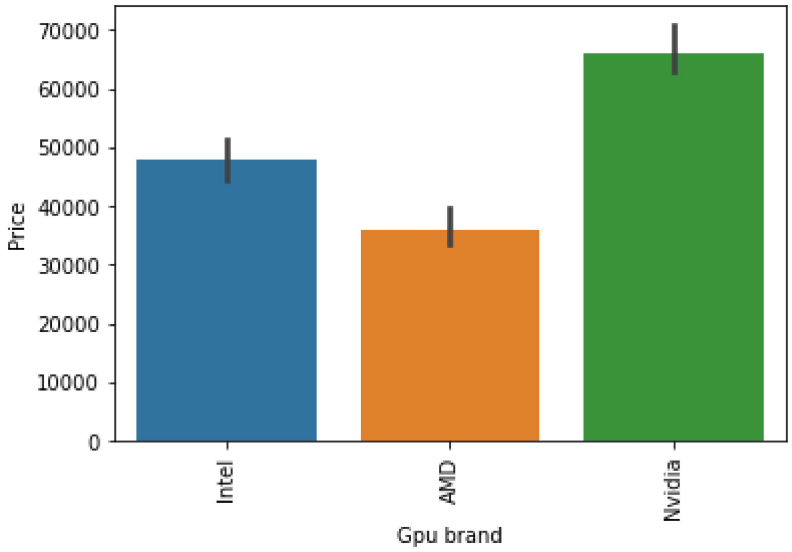
In [66]:
```
sns.barplot(df['Gpu brand'],df['Price'])
plt.xticks(rotation='vertical')
plt.show()  # price increases with ram
```

```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



In [67]:
```
sns.barplot(df['Gpu brand'],df['Price'],estimator=np.median)
plt.xticks(rotation='vertical')
plt.show()  # price increases with ram
```

```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



In [68]:
```
df.drop(columns=['Gpu'],inplace=True)
df.head()
```

Out[68]:

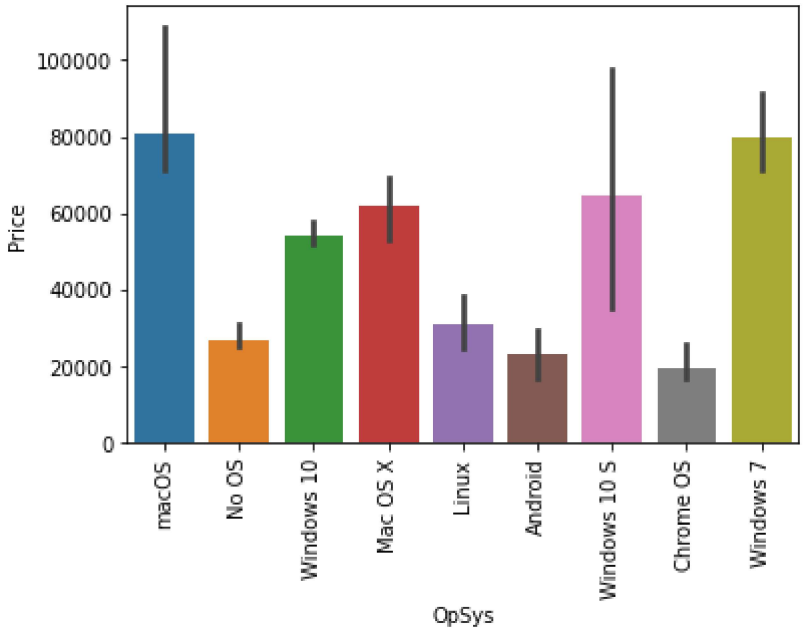
	Company	TypeName	Inches	Ram	OpSys	Weight	Price	Touchscreen	Ips	ppi	b
0	Apple	Ultrabook	13.3	8	macOS	1.37	71378.6832	0	1	226.983005	
1	Apple	Ultrabook	13.3	8	macOS	1.34	47895.5232	0	0	127.677940	
2	HP	Notebook	15.6	8	No OS	1.86	30636.0000	0	0	141.211998	
3	Apple	Ultrabook	15.4	16	macOS	1.83	135195.3360	0	1	220.534624	
4	Apple	Ultrabook	13.3	8	macOS	1.37	96095.8080	0	1	226.983005	

In [69]:
```
df['OpSys'].value_counts()
```

Out[69]:
```
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 [70]:
```
sns.barplot(df['OpSys'],df['Price'],estimator=np.median)
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



In [71]:
```
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)
```

In [72]:
```
df.head()
```

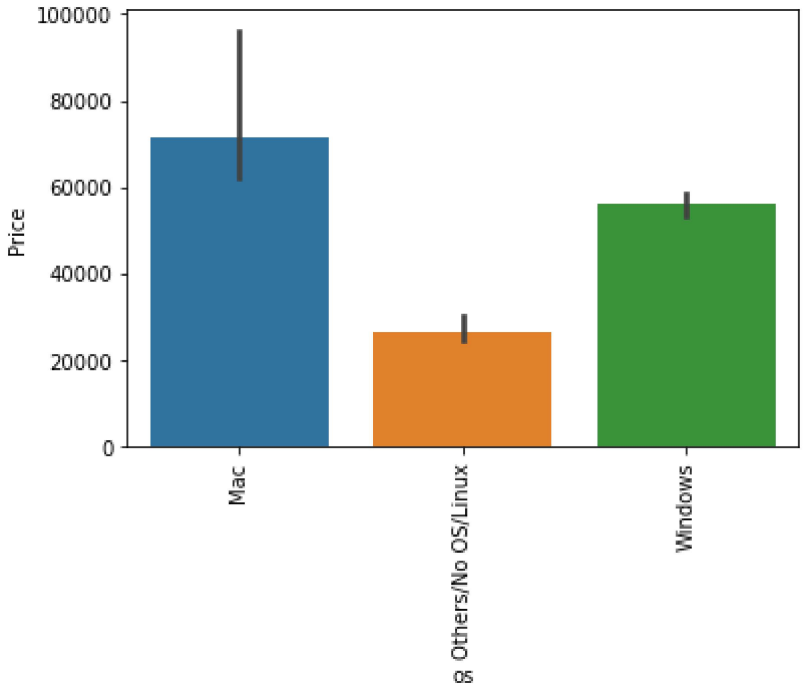
Out[72]:

	Company	TypeName	Inches	Ram	OpSys	Weight	Price	Touchscreen	Ips	ppi	b
0	Apple	Ultrabook	13.3	8	macOS	1.37	71378.6832	0	1	226.983005	
1	Apple	Ultrabook	13.3	8	macOS	1.34	47895.5232	0	0	127.677940	
2	HP	Notebook	15.6	8	No OS	1.86	30636.0000	0	0	141.211998	
3	Apple	Ultrabook	15.4	16	macOS	1.83	135195.3360	0	1	220.534624	
4	Apple	Ultrabook	13.3	8	macOS	1.37	96095.8080	0	1	226.983005	

In [73]:
```
df.drop(columns=['OpSys'],inplace=True)
```

In [74]:
```
sns.barplot(df['os'],df['Price'],estimator=np.median)
plt.xticks(rotation='vertical')
plt.show()
```

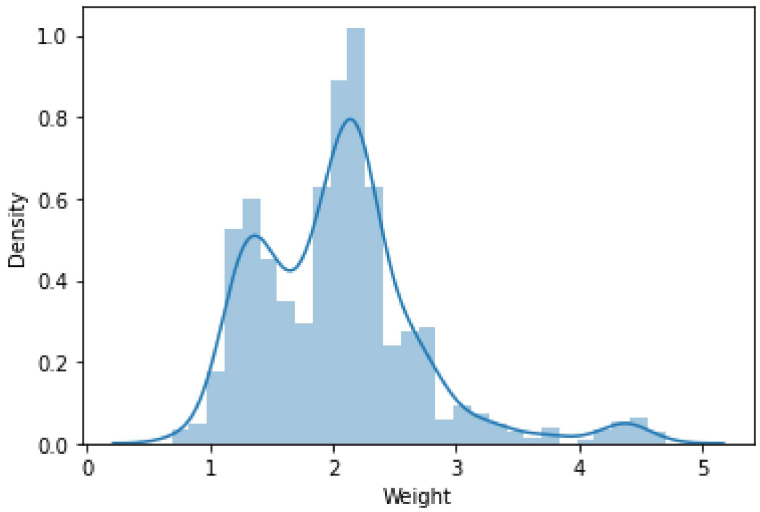
```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



In [75]:
```
sns.distplot(df['Weight'])
```

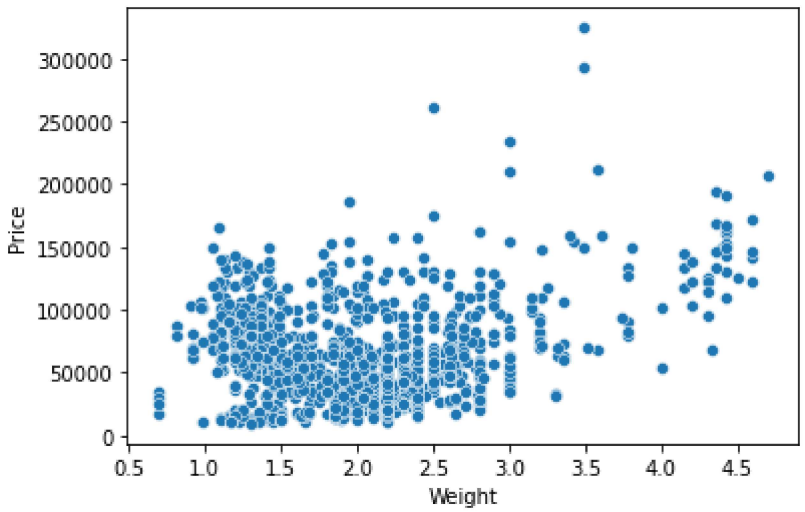
```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[75]:
```
<AxesSubplot:xlabel='Weight', ylabel='Density'>
```



In [76]:
```
sns.scatterplot(x=df['Weight'],y=df['Price'])
```

Out[76]:
```
<AxesSubplot:xlabel='Weight', ylabel='Price'>
```

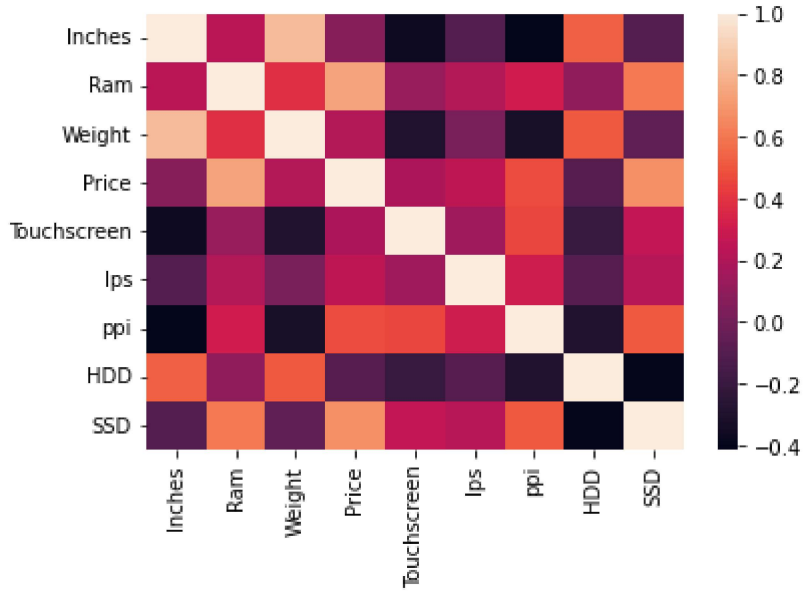


In [77]:
```
df.corr()['Price']
```

Out[77]:
```
Inches         0.067329
Ram            0.742905
Weight         0.209867
Price          1.000000
Touchscreen    0.192917
Ips            0.253320
ppi            0.475368
HDD           -0.096891
SSD            0.670660
Name: Price, dtype: float64
```

In [78]:
```
sns.heatmap(df.corr())
```

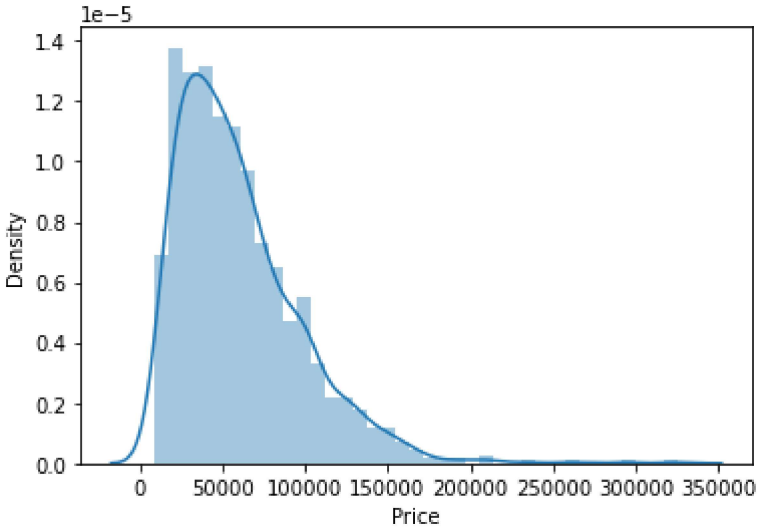
Out[78]:
```
<AxesSubplot:>
```



In [79]:
```
# skewed target column (can apply log)
sns.distplot(df['Price'])
```

```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

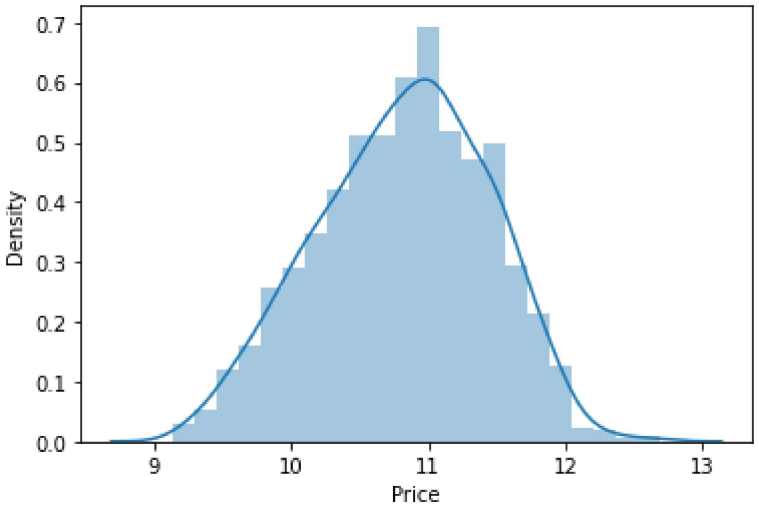
Out[79]:
```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```



In [80]:
```
# skewed target column (can apply log)
sns.distplot(np.log(df['Price']))
```

```
C:\Users\Sindhu\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[80]:
```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```



In [81]:
```
X=df.drop(columns=['Price'])
```

In [82]:
```
y=np.log(df['Price'])
```

In [83]:
```
X
```

Out[83]:

	Company	TypeName	Inches	Ram	Weight	Touchscreen	Ips	ppi	Cpu brand	HDD	SS
0	Apple	Ultrabook	13.3	8	1.37	0	1	226.983005	Intel Core i5	0	12
1	Apple	Ultrabook	13.3	8	1.34	0	0	127.677940	Intel Core i5	0	
2	HP	Notebook	15.6	8	1.86	0	0	141.211998	Intel Core i5	0	25
3	Apple	Ultrabook	15.4	16	1.83	0	1	220.534624	Intel Core i7	0	51
4	Apple	Ultrabook	13.3	8	1.37	0	1	226.983005	Intel Core i5	0	25
...	...	...	...	...	...	...	...	...	...	...	
1298	Lenovo	2 in 1 Convertible	14.0	4	1.80	1	1	157.350512	Intel Core i7	0	12
1299	Lenovo	2 in 1 Convertible	13.3	16	1.30	1	1	276.053530	Intel Core i7	0	51
1300	Lenovo	Notebook	14.0	2	1.50	0	0	111.935204	Other Inter Processor	0	
1301	HP	Notebook	15.6	6	2.19	0	0	100.454670	Intel Core i7	1000	
1302	Asus	Notebook	15.6	4	2.20	0	0	100.454670	Other Inter Processor	500	

1302 rows × 13 columns

In [84]:
```
y
```

Out[84]:
```
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 [85]:
```
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 [86]:
```
X_train
```

Out[86]:

	Company	TypeName	Inches	Ram	Weight	Touchscreen	Ips	ppi	Cpu brand	HDD	SS
183	Toshiba	Notebook	15.6	8	2.00	0	0	100.454670	Intel Core i5	0	12
1141	MSI	Gaming	15.6	8	2.40	0	0	141.211998	Intel Core i7	1000	12
1049	Asus	Netbook	11.6	4	1.20	0	0	135.094211	Other Inter Processor	0	
1020	Dell	2 in 1 Convertible	15.6	4	2.08	1	1	141.211998	Intel Core i3	1000	
878	Dell	Notebook	15.6	4	2.18	0	0	141.211998	Intel Core i5	1000	12
...	...	...	...	...	...	...	...	...	...	...	
466	Acer	Notebook	15.6	4	2.20	0	0	100.454670	Intel Core i3	500	
299	Asus	Ultrabook	15.6	16	1.63	0	0	141.211998	Intel Core i7	0	51
493	Acer	Notebook	15.6	8	2.20	0	0	100.454670	AMD Processor	1000	
527	Lenovo	Notebook	15.6	8	2.20	0	0	100.454670	Intel Core i3	2000	
1193	Apple	Ultrabook	12.0	8	0.92	0	1	226.415547	Other Inter Processor	0	

1106 rows × 13 columns

In [87]:
```
# Column transformer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import r2_score,mean_absolute_error
```

In [88]:
```
from sklearn.linear_model import LinearRegression,Ridge,Lasso
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor,AdaBoostR
from sklearn.svm import SVR
from xgboost import XGBRegressor
```

In [89]:
```
pip install --user xgboost
```

```
Requirement already satisfied: xgboost in c:\users\sindhu\appdata\roaming\python\pyth
on39\site-packages (2.1.2)
Requirement already satisfied: numpy in c:\users\sindhu\appdata\roaming\python\python
39\site-packages (from xgboost) (1.22.4)
Requirement already satisfied: scipy in c:\users\sindhu\anaconda3\lib\site-packages
(from xgboost) (1.7.3)
Note: you may need to restart the kernel to use updated packages.
WARNING: Ignoring invalid distribution -umpy (c:\users\sindhu\anaconda3\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -umpy (c:\users\sindhu\anaconda3\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -umpy (c:\users\sindhu\anaconda3\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -umpy (c:\users\sindhu\anaconda3\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -umpy (c:\users\sindhu\anaconda3\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -umpy (c:\users\sindhu\anaconda3\lib\site-pack
ages)
```

In [ ]:

In [91]:
```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

# Assuming 'Price' is the target variable
X = df.drop(columns='Price')
y = np.log(df['Price'])

# Splitting the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

# Preprocessing
categorical_features = ['Company', 'TypeName', 'Cpu brand', 'Gpu brand', 'os']
numerical_features = ['Inches', 'Ram', 'Weight', 'Touchscreen', 'Ips', 'ppi', 'HDD', '

# Column Transformer for preprocessing
preprocessor = ColumnTransformer([
    ('num', StandardScaler(), numerical_features),
    ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
])

# Pipeline
pipe = Pipeline([
    ('preprocessor', preprocessor),
    ('model', LinearRegression())
])

# Fit the pipeline
pipe.fit(X_train, y_train)

# Predict and evaluate
y_pred = pipe.predict(X_test)
print('R2 score:', r2_score(y_test, y_pred))
```

```
R2 score: 0.8086493879244961
```

In [92]:
```
print('MAE',mean_absolute_error(y_test,y_pred))
```

```
MAE 0.21366485760424406
```

In [93]:
```
np.exp(0.21)
```

Out[93]:
```
1.2336780599567432
```

In [94]:
```
categorical_features = ['Company', 'TypeName', 'Cpu brand', 'Gpu brand', 'os']
numerical_features = ['Inches', 'Ram', 'Weight', 'Touchscreen', 'Ips', 'ppi', 'HDD', '

step1 = ColumnTransformer(
    transformers=[
        ('col_tnf', OneHotEncoder(sparse=False, drop='first'), categorical_features)
    ],
    remainder='passthrough'  # Keep numerical columns
)
```

In [95]:
```
X_transformed = step1.fit_transform(X_train)
print(X_transformed)
```

```
[[   0.            1.            0.         ...  141.21199808
  1000.            0.        ]
 [   0.            0.            0.         ...  141.21199808
  1000.            0.        ]
 [   0.            0.            0.         ...  111.93520356
   500.            0.        ]
 ...
 [   0.            0.            0.         ...  100.45466986
  1000.            0.        ]
 [   0.            0.            0.         ...  141.21199808
  1000.            0.        ]
 [   0.            0.            0.         ...  224.17380908
     0.            0.        ]]
```

In [96]:
```
pipe = Pipeline([
    ('preprocessor', step1),
    ('model', Ridge(alpha=10))
])
```

In [97]:
```
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: 0.8018593615383057
MAE: 0.21947245686467293
```

In [98]:
```
# try lassso, decisopn and svm
```

In [99]:
```
# Random Forest
```

In [100…
```
pipe = Pipeline([
    ('preprocessor', preprocessor),
    ('model', RandomForestRegressor(
        n_estimators=100,
        random_state=3,
```

```
        max_samples=0.5,
        max_features=0.75,
        max_depth=15
    ))
])


# Fit the pipeline
pipe.fit(X_train, y_train)

# Predict and evaluate
y_pred = pipe.predict(X_test)
print('R2 score:', r2_score(y_test, y_pred))
```

```
R2 score: 0.8637178664140408
```

In [101…

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

```
MAE: 0.17335443840013795
```

In [102…

```
# GradientBoostingRegressor
pipe = Pipeline([
    ('preprocessor', preprocessor),
    ('model', GradientBoostingRegressor(n_estimators=500)
    )
])


# Fit the pipeline
pipe.fit(X_train, y_train)

# Predict and evaluate
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: 0.8780903165272979
MAE: 0.16471407339113897
```

In [ ]:

```
# best GradientBoostingRegressor model
```

In [105…

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