Laptop_Price_Predictor_Model

August 31, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

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

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

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

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

[3]:
```
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
```

[4]:
```
df.isnull().sum()
```

[4]:
```
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
```

[5]:
```
df.drop(columns=['Unnamed: 0'],inplace=True)
df['Ram'] = df['Ram'].str.replace('GB','')
df['Weight'] = df['Weight'].str.replace('kg','')
df['Ram'] = df['Ram'].astype('int32')
df['Weight'] = df['Weight'].astype('float32')
```

[6]:
```
df.head()
```

[6]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232
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	30636.0000
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	135195.3360
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	96095.8080

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

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

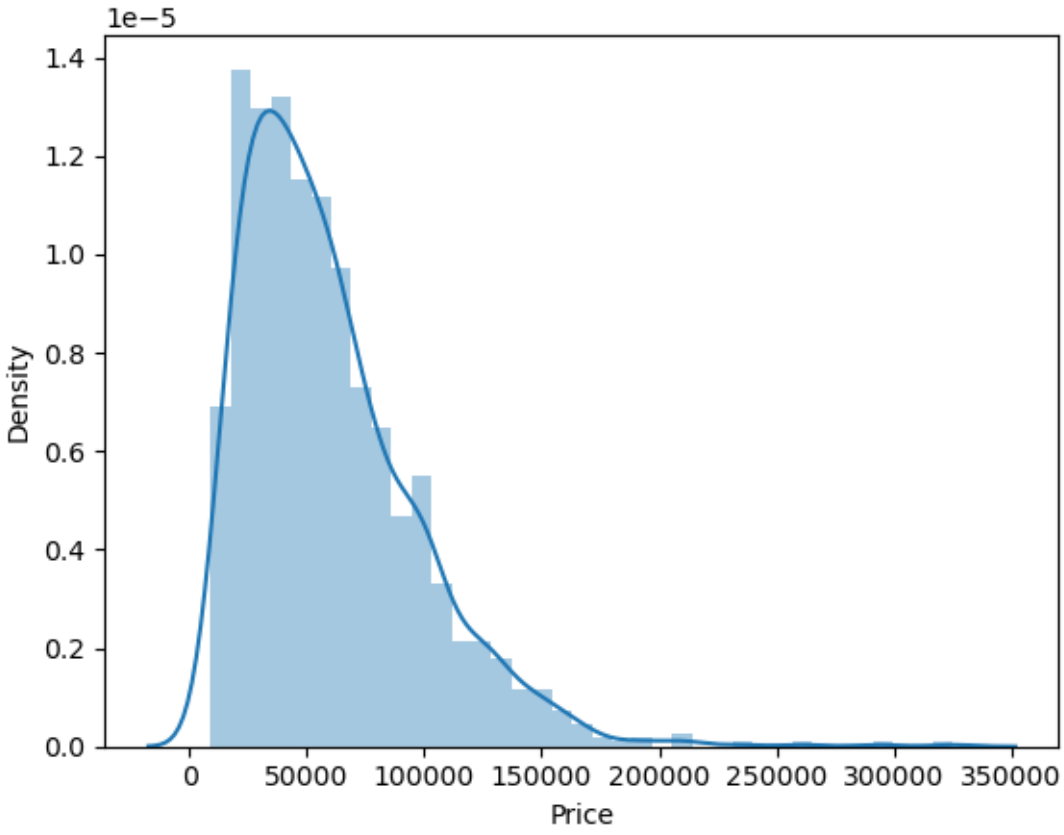
[7]:
```
df.info()
```

```
<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
```

[8]:
```
sns.distplot(df['Price'])
```

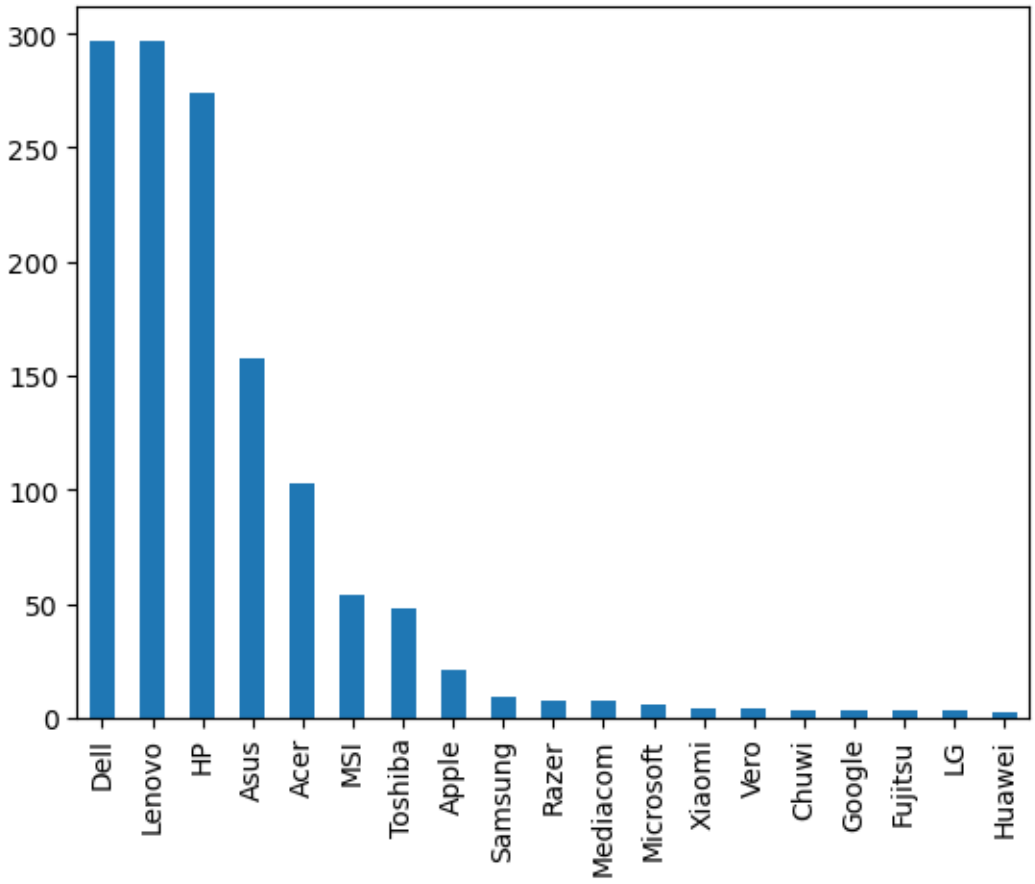
```
C:\Users\hp.pc\AppData\Local\Temp\ipykernel_6820\834922981.py:1: UserWarning:

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

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

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

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

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

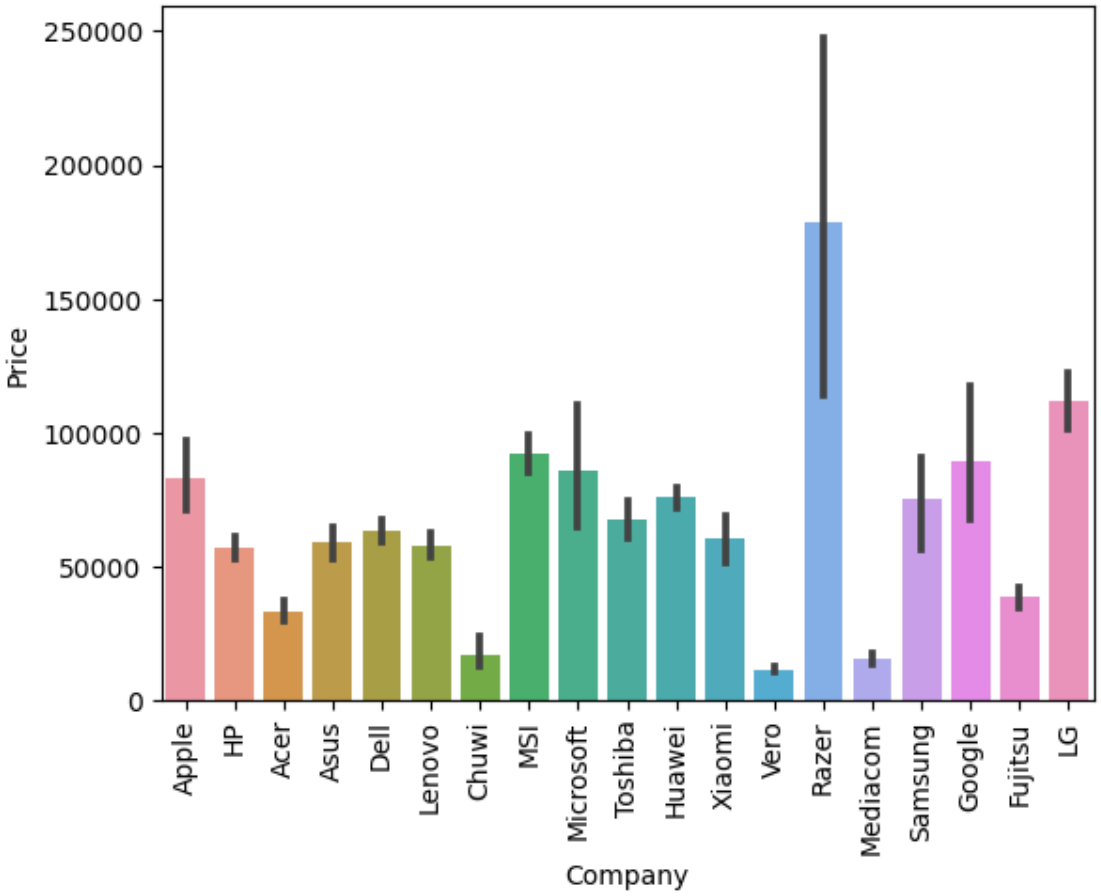


[9]:
```
df['Company'].value_counts().plot(kind='bar')
```

[9]: <Axes: >

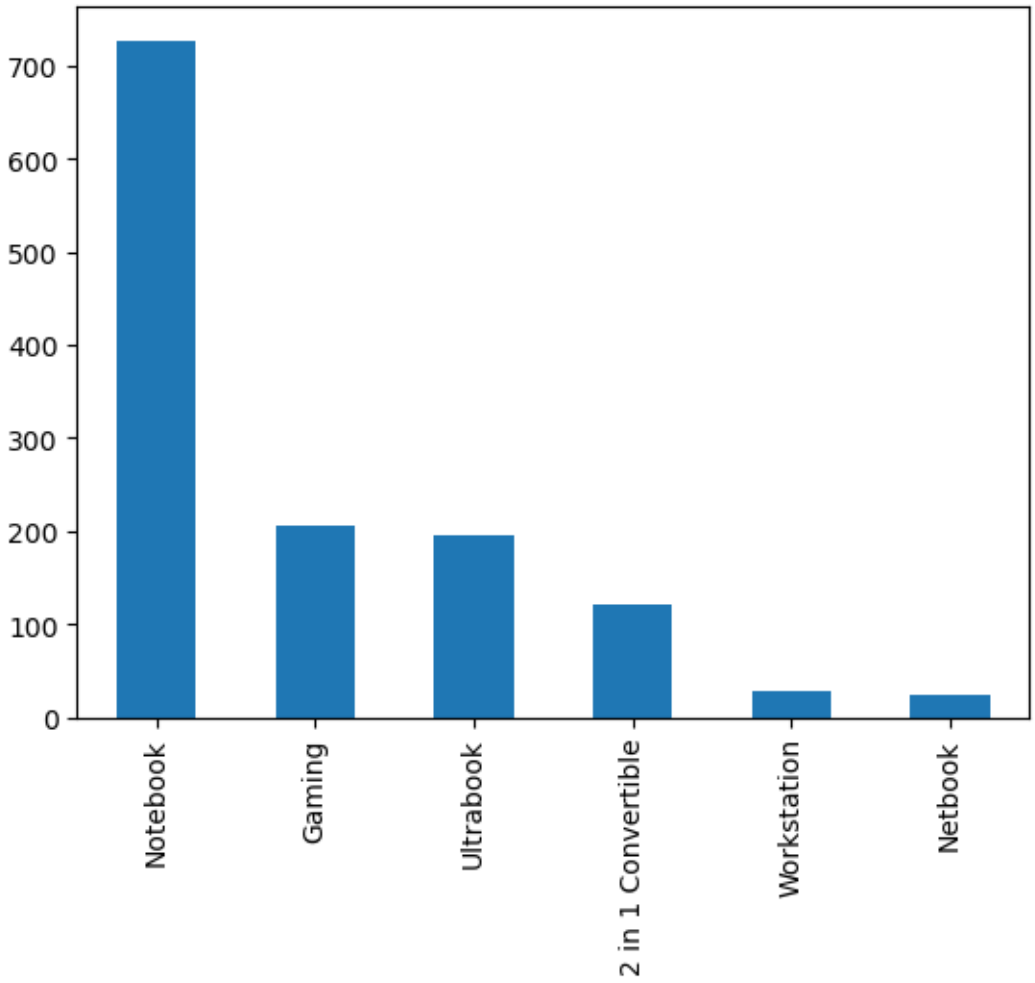


[10]:
```
sns.barplot(x=df['Company'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```

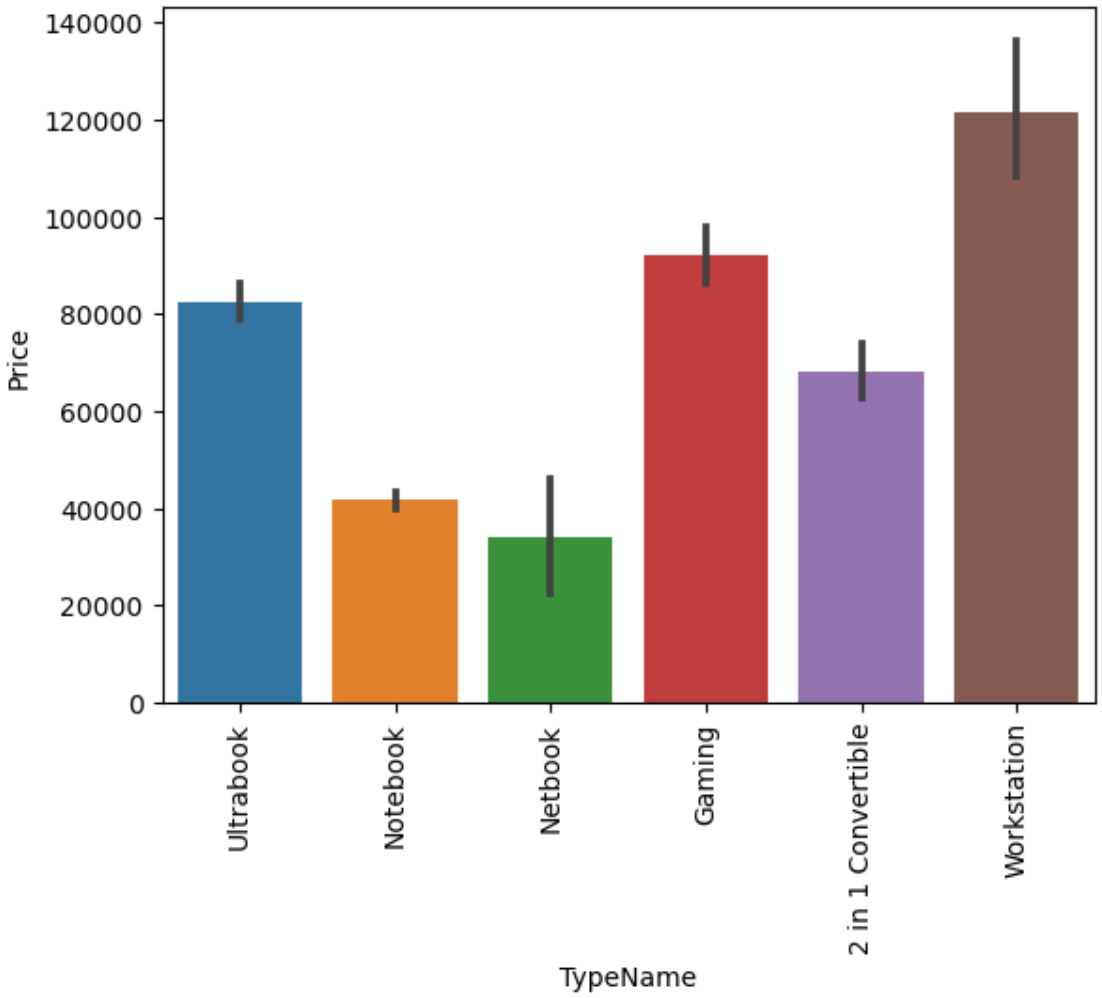


[11]:
```
df['TypeName'].value_counts().plot(kind='bar')
```

[11]: <Axes: >



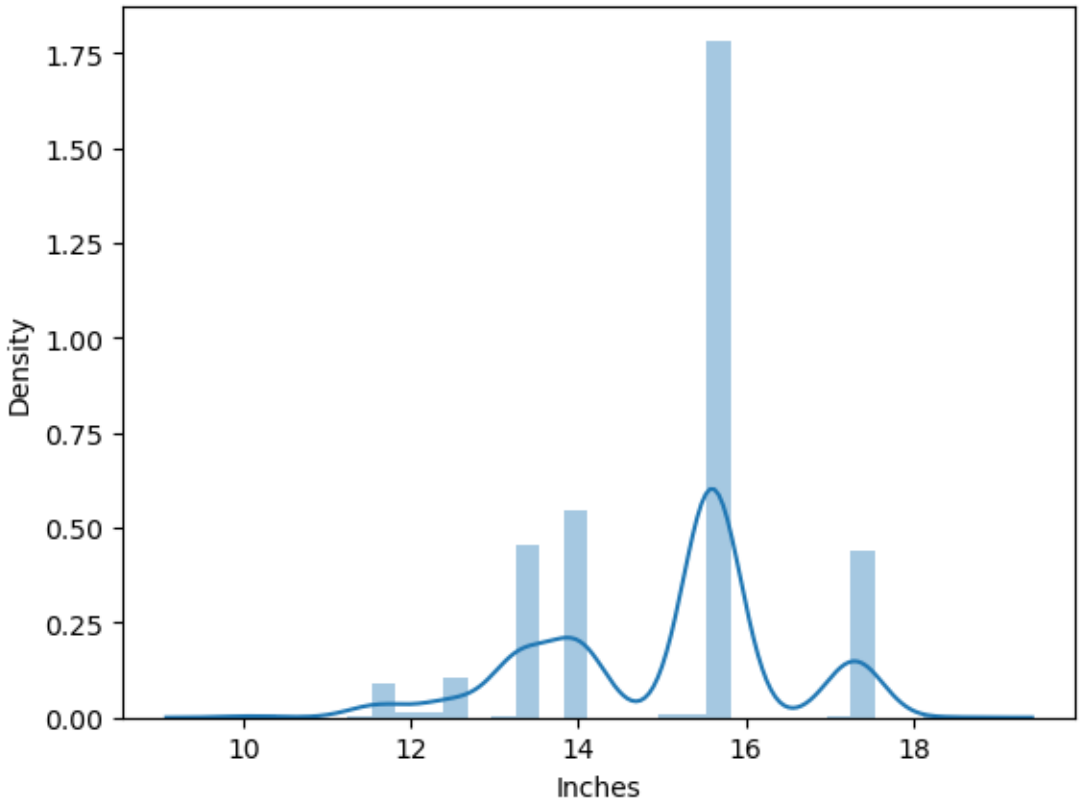
[12]:
```
sns.barplot(x=df['TypeName'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



[13]:
```
sns.distplot(df['Inches'])
```

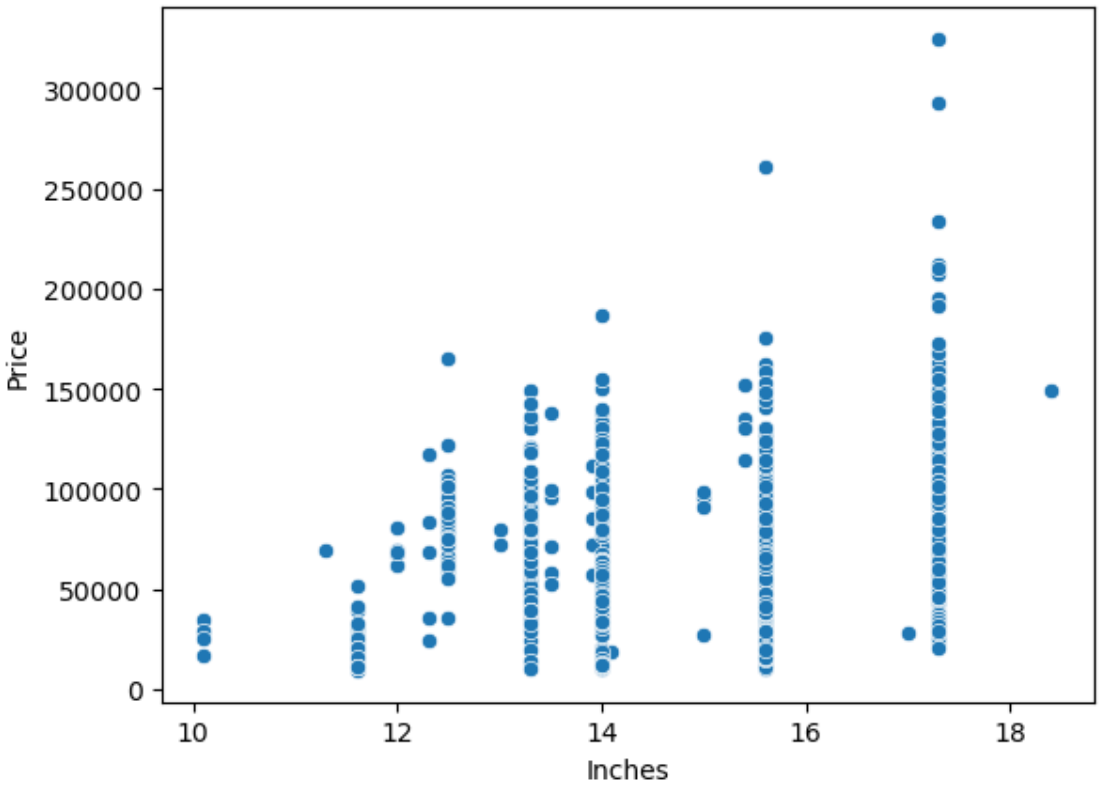
```
C:\Users\hp.pc\AppData\Local\Temp\ipykernel_6820\1439577752.py:1: UserWarning:

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

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

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

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

[13]:
```
<Axes: xlabel='Inches', ylabel='Density'>
```



[14]:
```
sns.scatterplot(x=df['Inches'],y=df['Price'])
```

[14]:
```
<Axes: xlabel='Inches', ylabel='Price'>
```



[15]:
```
df['ScreenResolution'].value_counts()
```

[15]:
```
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
```

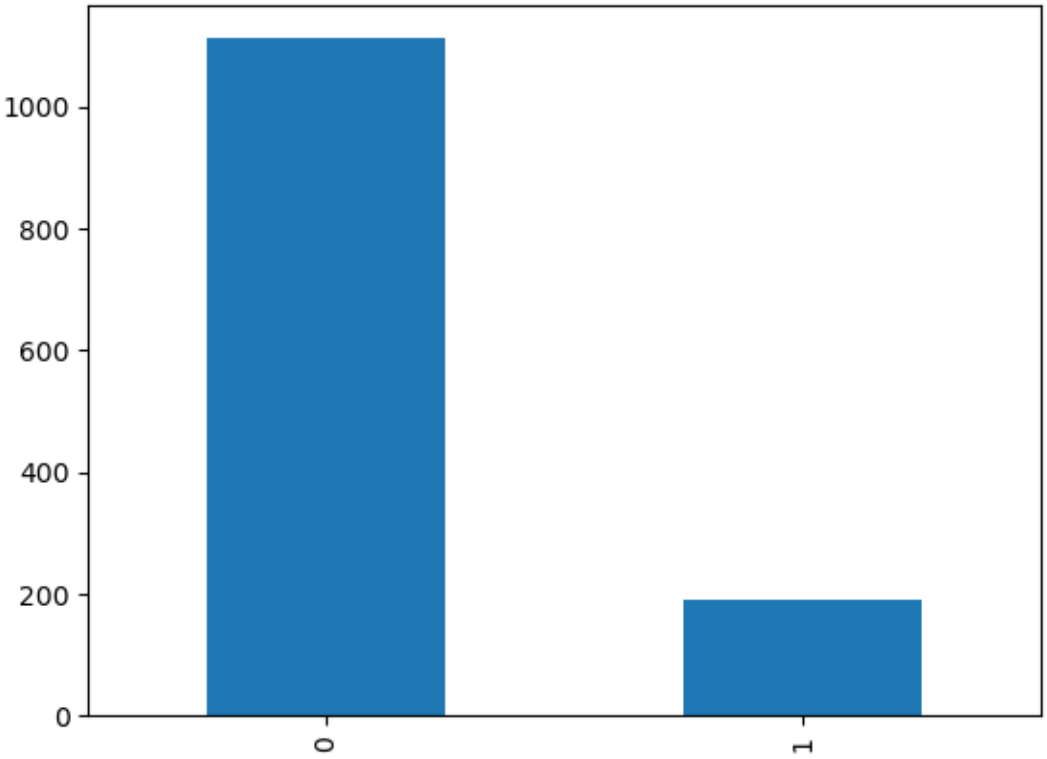
[16]:
```
df['Touchscreen'] = df['ScreenResolution'].apply(lambda x:1 if 'Touchscreen' in␣
 ↪x else 0)
df.sample(5)
```

[16]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen
835	Dell	Gaming	15.6	Full HD 1920x1080	Intel Core i7 7700HQ 2.8GHz	16	256GB SSD + 1TB HDD	Nvidia GeForce GTX 1070	Windows 10	3.21	147832.2864	0
914	Acer	Notebook	15.6	Full HD 1920x1080	Intel Core i3 7100U 2.4GHz	4	1TB HDD	Intel HD Graphics 620	Windows 10	2.40	26586.7200	0
856	Asus	Notebook	15.6	1366x768	Intel Pentium Quad Core N4200 1.1GHz	4	1TB HDD	Intel HD Graphics 505	Windows 10	2.00	23922.7200	0
1066	Asus	Gaming	17.3	IPS Panel Full HD 1920x1080	Intel Core i7 6820HK 2.7GHz	64	1TB SSD	Nvidia GeForce GTX 980	Windows 10	3.58	211788.0000	0
702	Lenovo	Notebook	15.6	1366x768	AMD A12-Series 9720P 3.6GHz	8	1TB HDD	AMD Radeon R7	Windows 10	2.20	22857.1200	0

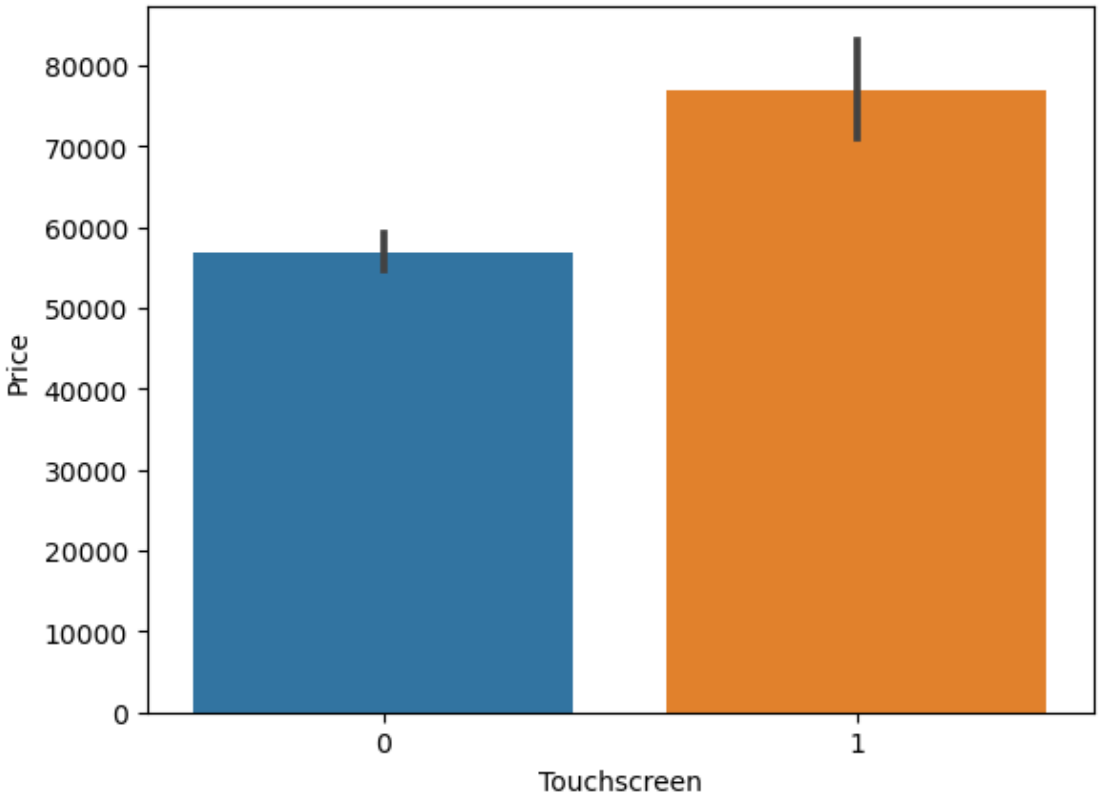
[17]:
```
df['Touchscreen'].value_counts().plot(kind='bar')
```

[17]: <Axes: >



[18]:
```
sns.barplot(x=df['Touchscreen'],y=df['Price'])
```

[18]: <Axes: xlabel='Touchscreen', ylabel='Price'>



[19]:
```
df['Ips'] = df['ScreenResolution'].apply(lambda x:1 if 'IPS' in x else 0)
df.sample(5)
```

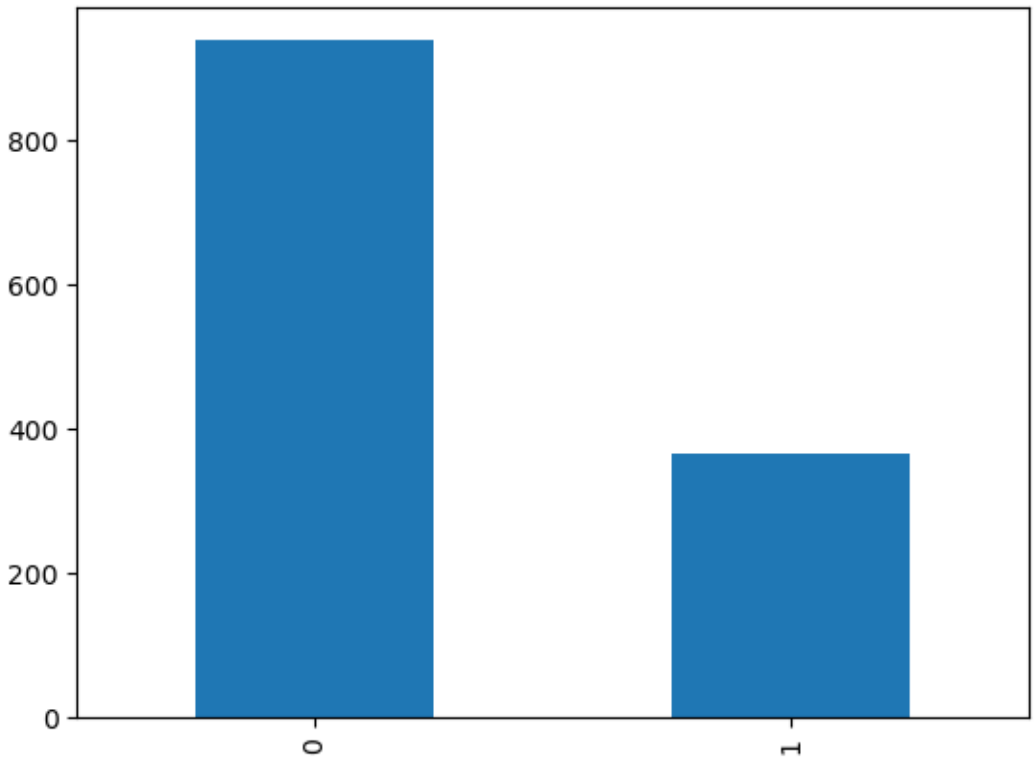
[19]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips
1145	HP	Workstation	15.6	Full HD 1920x1080	Intel Core i7 6700HQ 2.6GHz	8	256GB SSD	Nvidia Quadro M1000M	Windows 7	2.00	101178.7200	0	0
1168	HP	Notebook	14.0	1366x768	Intel Celeron Dual Core N3060 1.6GHz	2	32GB Flash Storage	Intel HD Graphics 400	Windows 10	1.44	13266.7200	0	0
1002	Dell	Notebook	15.6	1366x768	Intel Core i3 7100U 2.4GHz	4	128GB SSD	Intel HD Graphics 620	Windows 10	2.18	29144.1600	0	0
1266	HP	Notebook	15.6	Full HD 1920x1080	AMD A9-Series 9410 2.9GHz	6	1.0TB Hybrid						
12	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.8GHz	16	256GB SSD						

```
1266      AMD Radeon R7 M440  Windows 10    2.04   29303.4672            0    0
12        AMD Radeon Pro 555       macOS    1.83  130001.6016            0    1
```

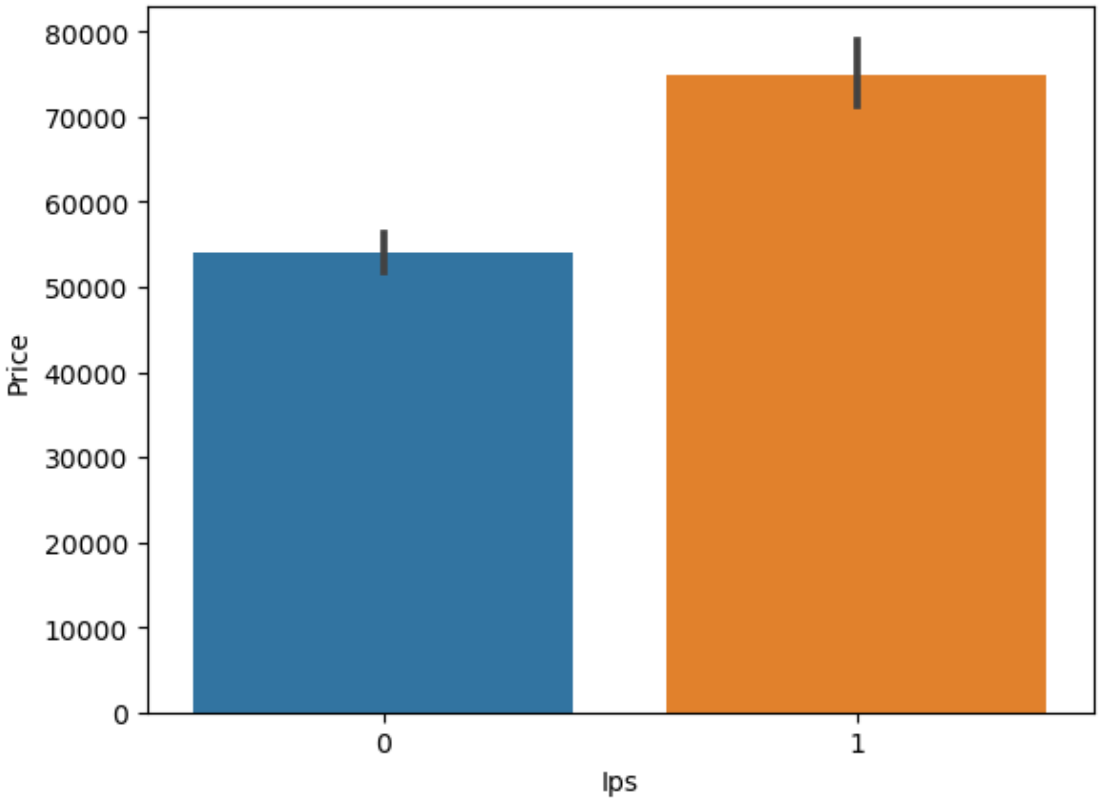
[20]:
```
df['Ips'].value_counts().plot(kind='bar')
```

[20]: <Axes: >



[21]:
```
sns.barplot(x=df['Ips'],y=df['Price'])
```

[21]: <Axes: xlabel='Ips', ylabel='Price'>



[22]:
```
new = df['ScreenResolution'].str.split('x',n=1,expand=True)
df['X_res'] = new[0]
df['Y_res'] = new[1]
df.sample(5)
```

[22]:

	Company	TypeName	Inches	ScreenResolution
23	Dell	2 in 1 Convertible	13.3	Full HD / Touchscreen 1920x1080
923	Toshiba	Notebook	15.6	IPS Panel Full HD 1920x1080
1127	HP	Ultrabook	12.5	1366x768
761	Dell	Ultrabook	12.5	Full HD 1920x1080
260	Dell	Notebook	17.3	Full HD 1920x1080

	Cpu	Ram	Memory
23	Intel Core i5 8250U 1.6GHz	8	256GB SSD
923	Intel Core i7 6600U 2.6GHz	16	256GB SSD
1127	Intel Core i5 6300U 2.4GHz	8	256GB SSD
761	Intel Core i7 7600U 2.8GHz	16	256GB SSD
260	Intel Core i7 8550U 1.8GHz	8	128GB SSD + 1TB HDD

	Gpu	OpSys	Weight	Price	Touchscreen	Ips
23	Intel UHD Graphics 620	Windows 10	1.62	43636.32	1	0

```
923      Nvidia GeForce 930M  Windows 10    2.40  105228.00           0    1
1127   Intel HD Graphics 520   Windows 7    1.26  100965.60           0    0
761    Intel HD Graphics 620  Windows 10    1.18   99047.52           0    0
260            AMD Radeon 530  Windows 10    2.80   60845.76           0    0

                           X_res Y_res
23    Full HD / Touchscreen 1920  1080
923       IPS Panel Full HD 1920  1080
1127                        1366   768
761                 Full HD 1920  1080
260                 Full HD 1920  1080
```

[23]:
```
df['X_res'] = df['X_res'].str.replace(',','').str.findall(r'(\d+\.?\d+)').
  ↪apply(lambda x:x[0])
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
```

[24]:
```
df.head()
```

[24]:

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

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

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

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

[25]:
```
df['X_res'] = df['X_res'].astype('int')
df['Y_res'] = df['Y_res'].astype('int')
df.info()
```

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

[26]:
```
df.corr()['Price']
```

```
C:\Users\hp.pc\AppData\Local\Temp\ipykernel_6820\815546952.py:1: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  df.corr()['Price']
```

[26]:
```
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
```

[27]:
```
df['ppi'] = (((df['X_res']**2) + (df['Y_res']**2))**0.5/df['Inches']).
 ↪astype('float')
df.corr()['Price']
```

```
C:\Users\hp.pc\AppData\Local\Temp\ipykernel_6820\4248563634.py:2: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  df.corr()['Price']
```

[27]:
```
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
```

[28]:
```
df.drop(columns=['ScreenResolution'],inplace=True)
df.head()
```

[28]:

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

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

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

[29]:
```
df.drop(columns=['Inches','X_res','Y_res'],inplace=True)
df.head()
```

[29]:
```
  Company   TypeName                         Cpu  Ram               Memory  \
0   Apple  Ultrabook        Intel Core i5 2.3GHz    8            128GB SSD
1   Apple  Ultrabook        Intel Core i5 1.8GHz    8  128GB Flash Storage
2      HP   Notebook  Intel Core i5 7200U 2.5GHz    8            256GB SSD
3   Apple  Ultrabook        Intel Core i7 2.7GHz   16            512GB SSD
4   Apple  Ultrabook        Intel Core i5 3.1GHz    8            256GB SSD

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

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

[30]:
```
df['Cpu'].value_counts()
```

[30]:
```
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
```

[31]:
```
df['Cpu Name'] = df['Cpu'].apply(lambda x:" ".join(x.split()[0:3]))
df.head()
```

[31]:

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

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

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

[32]:
```
def fetch_processor(text):
    if text == 'Intel Core i7' or text == 'Intel Core i5' or text == 'Intel␣
 ↪Core i3':
        return text
    else:
        if text.split()[0] == 'Intel':
            return 'Other Intel Processor'
        else:
            return 'AMD Processor'
```

[33]:
```
df['Cpu brand'] = df['Cpu Name'].apply(fetch_processor)
df.head()
```

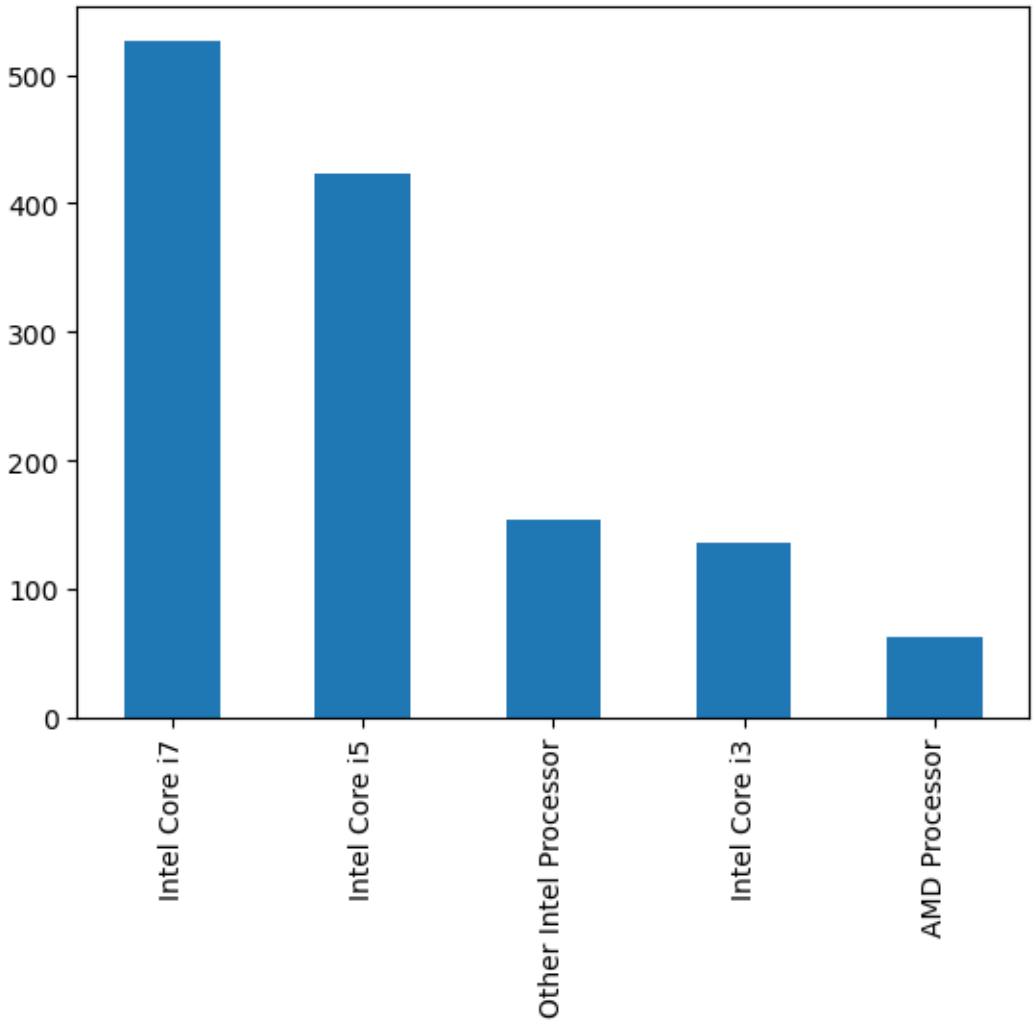
[33]:

	Company	TypeName	Cpu	Ram	Memory	\
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	128GB Flash Storage	

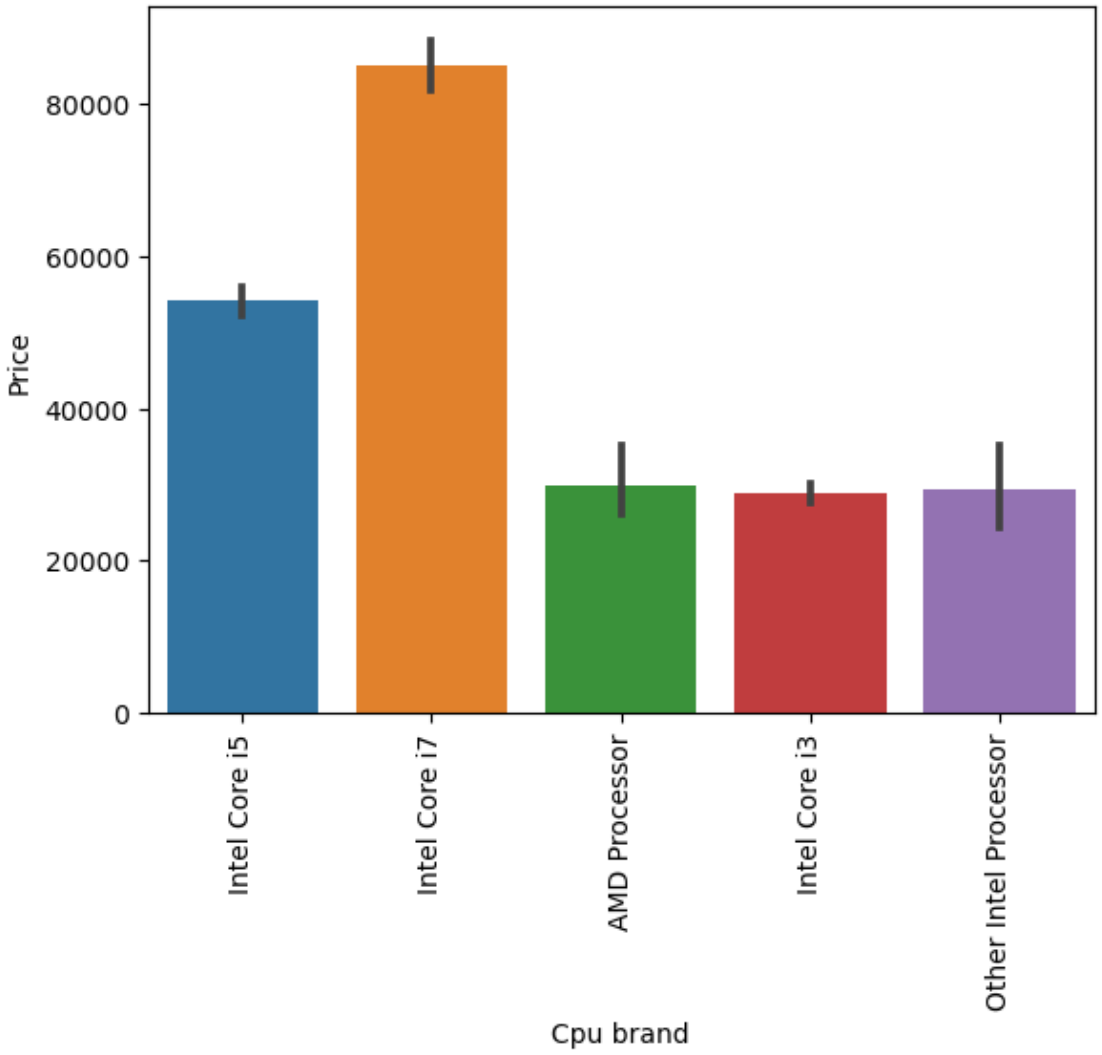
```
2      HP   Notebook  Intel Core i5 7200U 2.5GHz    8           256GB SSD
3   Apple  Ultrabook        Intel Core i7 2.7GHz   16           512GB SSD
4   Apple  Ultrabook        Intel Core i5 3.1GHz    8           256GB SSD

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

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

[34]:
```
df['Cpu brand'].value_counts().plot(kind='bar')
```

[34]: <Axes: >



[35]:
```
sns.barplot(x=df['Cpu brand'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



[36]:
```
df.drop(columns=['Cpu','Cpu Name'],inplace=True)
df.head()
```

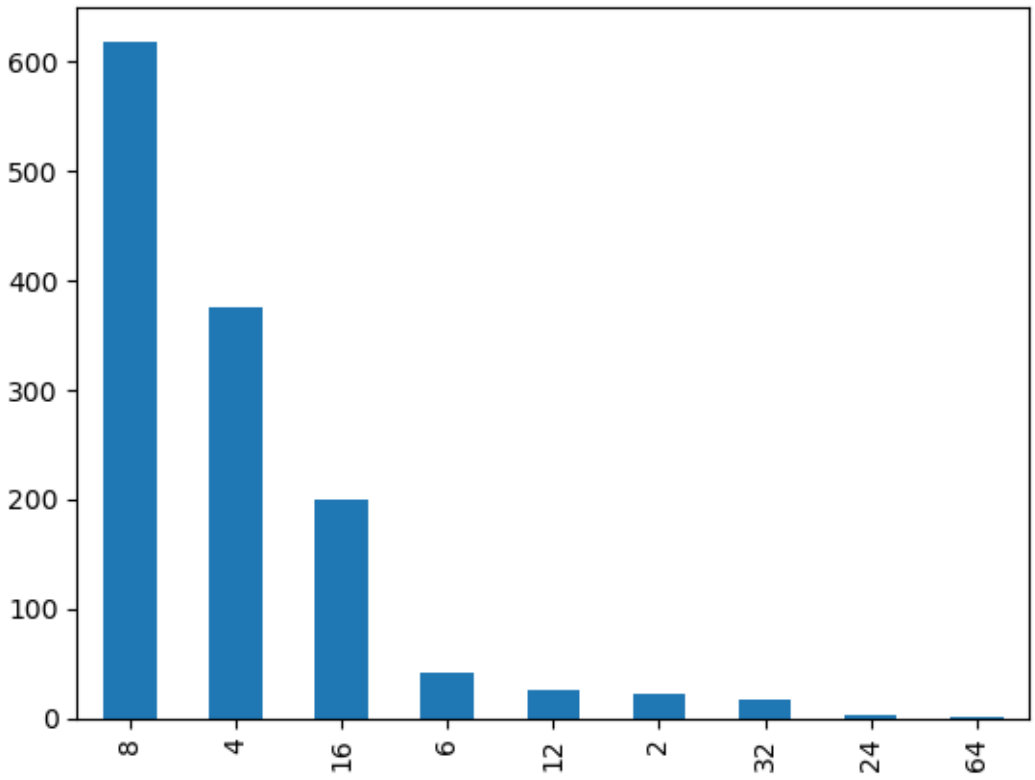
[36]:

	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	Touchscreen	Ips	ppi	Cpu brand
0	Apple	Ultrabook	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5
1	Apple	Ultrabook	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5
2	HP	Notebook	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5
3	Apple	Ultrabook	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7
4	Apple	Ultrabook	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5

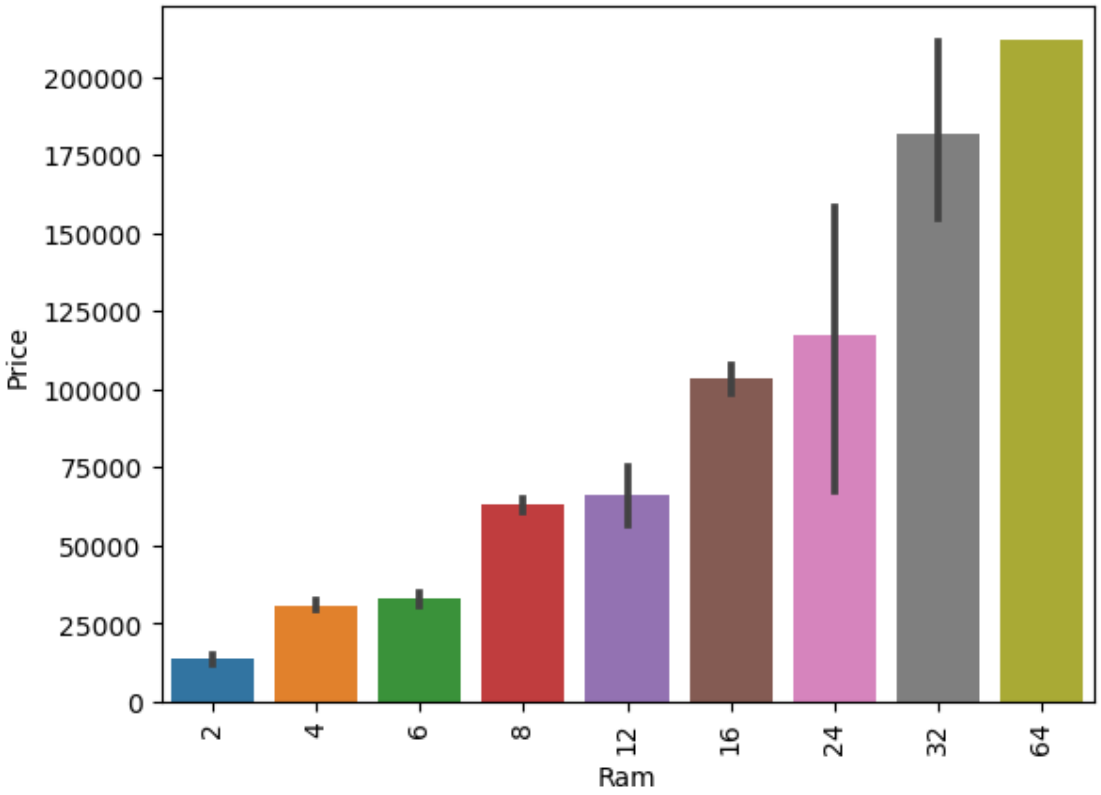
```
3   macOS    1.83  135195.3360            0    1  220.534624  Intel Core i7
4   macOS    1.37   96095.8080            0    1  226.983005  Intel Core i5
```

[37]:
```
df['Ram'].value_counts().plot(kind='bar')
```

[37]:
```
<Axes: >
```



[38]:
```
sns.barplot(x=df['Ram'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



[39]:
```
df['Memory'].value_counts()
```

[39]:
```
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
```

[40]:
```
df['Memory'] = df['Memory'].astype(str).replace('\.0', '', regex=True)
df["Memory"] = df["Memory"].str.replace('GB', '')
df["Memory"] = df["Memory"].str.replace('TB', '000')
new = df["Memory"].str.split("+", n = 1, expand = True)

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

df["second"]= new[1]

df["Layer1HDD"] = df["first"].apply(lambda x: 1 if "HDD" in x else 0)
df["Layer1SSD"] = df["first"].apply(lambda x: 1 if "SSD" in x else 0)
df["Layer1Hybrid"] = df["first"].apply(lambda x: 1 if "Hybrid" in x else 0)
df["Layer1Flash_Storage"] = df["first"].apply(lambda x: 1 if "Flash Storage" in␣
 ↪x else 0)

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

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

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

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

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

df["HDD"]=(df["first"]*df["Layer1HDD"]+df["second"]*df["Layer2HDD"])
df["SSD"]=(df["first"]*df["Layer1SSD"]+df["second"]*df["Layer2SSD"])
df["Hybrid"]=(df["first"]*df["Layer1Hybrid"]+df["second"]*df["Layer2Hybrid"])
df["Flash_Storage"]=(df["first"]*df["Layer1Flash_Storage"]+df["second"]*df["Layer2Flash_Storag

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

```
C:\Users\hp.pc\AppData\Local\Temp\ipykernel_6820\4023190604.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\hp.pc\AppData\Local\Temp\ipykernel_6820\4023190604.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', '')
```

[41]:
```
df.sample(5)
```

[41]:

	Company	TypeName	Ram	Memory	\
319	Acer	Notebook	4	128 Flash Storage	
187	Lenovo	Gaming	8	256 SSD	
953	Asus	2 in 1 Convertible	8	64 Flash Storage	
554	HP	Notebook	8	1000 HDD	
229	HP	Notebook	8	1000 HDD	

	Gpu	OpSys	Weight	Price	Touchscreen	Ips	\
319	Intel HD Graphics 405	Windows 10	1.40	25840.80	0	0	
187	Nvidia GeForce GTX 1050	No OS	2.40	41505.12	0	1	
953	Intel HD Graphics 515	Chrome OS	1.20	61751.52	0	0	
554	Nvidia GeForce 930MX	Windows 10	2.63	68198.40	0	0	
229	AMD FirePro W4190M	Windows 10	1.90	67612.32	0	0	

	ppi	Cpu brand	HDD	SSD	Hybrid	Flash_Storage
319	135.094211	Other Intel Processor	0	0	0	128
187	141.211998	Intel Core i5	0	256	0	0
953	176.232574	Other Intel Processor	0	0	0	64
554	127.335675	Intel Core i7	1000	0	0	0
229	141.211998	Intel Core i7	1000	0	0	0

[42]:
```
df.drop(columns=['Memory'],inplace=True)
df.head()
```

[42]:

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	Touchscreen	Ips	ppi	Cpu brand	HDD	SSD	Hybrid	Flash_Storage
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1	226.983005	Intel Core i5	0	128	0	0
1	Apple	Ultrabook	8	Intel HD Graphics 6000	macOS	1.34	47895.5232	0	0	127.677940	Intel Core i5	0	0	0	128
2	HP	Notebook	8	Intel HD Graphics 620	No OS	1.86	30636.0000	0	0	141.211998	Intel Core i5	0	256	0	0
3	Apple	Ultrabook	16	AMD Radeon Pro 455	macOS	1.83	135195.3360	0	1	220.534624	Intel Core i7	0	512	0	0
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	96095.8080	0	1	226.983005	Intel Core i5	0	256	0	0

[43]:
```
df.corr()['Price']
```

```
C:\Users\hp.pc\AppData\Local\Temp\ipykernel_6820\815546952.py:1: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  df.corr()['Price']
```

[43]:
```
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
```

[44]:
```
df.drop(columns=['Hybrid','Flash_Storage'],inplace=True)
df.head()
```

[44]:

	Company	TypeName	Ram	Gpu	OpSys	Weight	\
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	
1	Apple	Ultrabook	8	Intel HD Graphics 6000	macOS	1.34	
2	HP	Notebook	8	Intel HD Graphics 620	No OS	1.86	
3	Apple	Ultrabook	16	AMD Radeon Pro 455	macOS	1.83	
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	

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

[45]:
```
df['Gpu'].value_counts()
```

[45]:
```
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
```

[46]:
```
df['Gpu brand'] = df['Gpu'].apply(lambda x:x.split()[0])
df.head()
```

[46]:

	Company	TypeName	Ram	Gpu	OpSys	Weight	\
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	
1	Apple	Ultrabook	8	Intel HD Graphics 6000	macOS	1.34	
2	HP	Notebook	8	Intel HD Graphics 620	No OS	1.86	
3	Apple	Ultrabook	16	AMD Radeon Pro 455	macOS	1.83	
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	

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

	Gpu brand

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

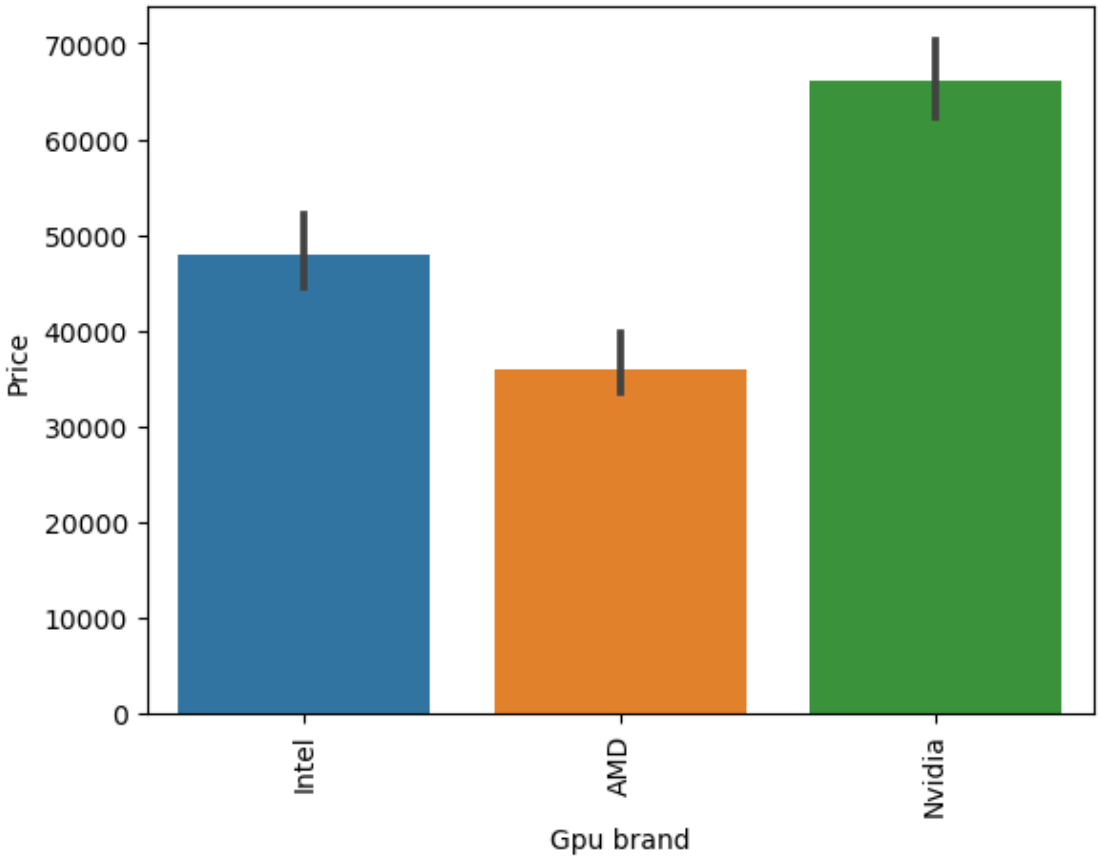
[47]:
```
df['Gpu brand'].value_counts()
```

[47]:
```
Intel     722
Nvidia    400
AMD       180
ARM         1
Name: Gpu brand, dtype: int64
```

[48]:
```
df = df[df['Gpu brand'] != 'ARM']
df['Gpu brand'].value_counts()
```

[48]:
```
Intel     722
Nvidia    400
AMD       180
Name: Gpu brand, dtype: int64
```

[49]:
```
sns.barplot(x=df['Gpu brand'],y=df['Price'],estimator=np.median)
plt.xticks(rotation='vertical')
plt.show()
```



[50]:
```
df.drop(columns=['Gpu'],inplace=True)
df.head()
```

[50]:

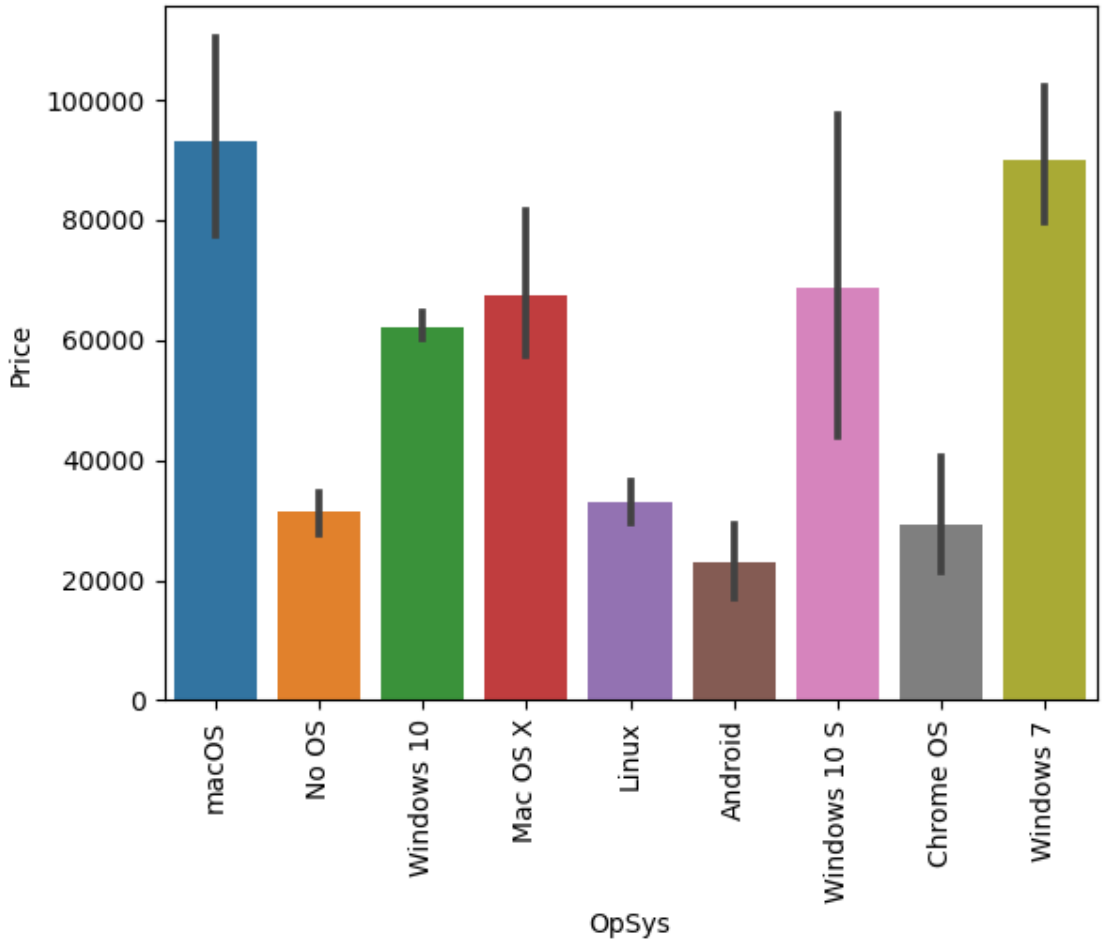
	Company	TypeName	Ram	OpSys	Weight	Price	Touchscreen	Ips
0	Apple	Ultrabook	8	macOS	1.37	71378.6832	0	1
1	Apple	Ultrabook	8	macOS	1.34	47895.5232	0	0
2	HP	Notebook	8	No OS	1.86	30636.0000	0	0
3	Apple	Ultrabook	16	macOS	1.83	135195.3360	0	1
4	Apple	Ultrabook	8	macOS	1.37	96095.8080	0	1

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

[51]:
```
df['OpSys'].value_counts()
```

[51]:
```
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
```

[52]:
```
sns.barplot(x=df['OpSys'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



[53]:
```
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'
```

[54]:
```
df['os'] = df['OpSys'].apply(cat_os)
df.head()
```

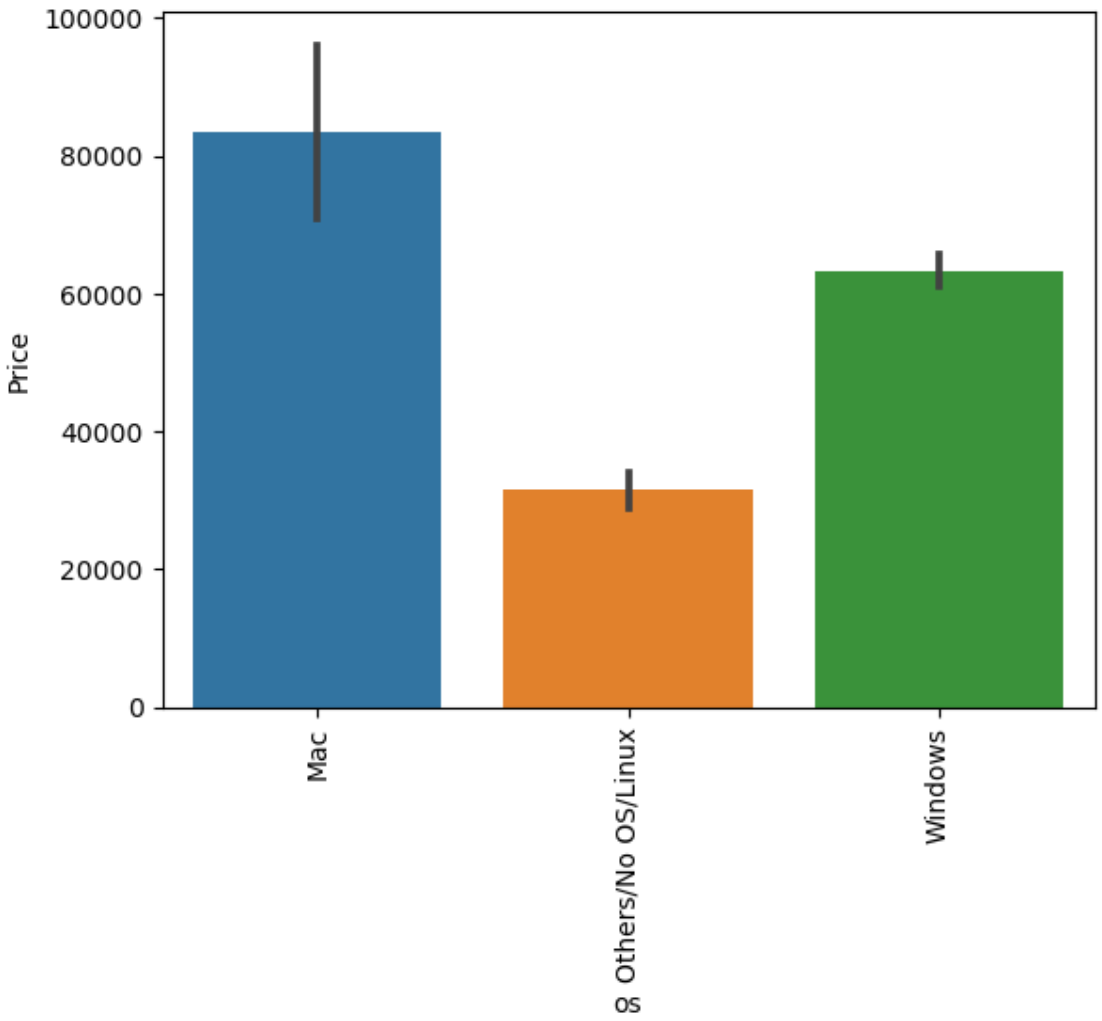
[54]:

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

	ppi	Cpu brand	HDD	SSD	Gpu brand	os
0	226.983005	Intel Core i5	0	128	Intel	Mac
1	127.677940	Intel Core i5	0	0	Intel	Mac
2	141.211998	Intel Core i5	0	256	Intel	Others/No OS/Linux
3	220.534624	Intel Core i7	0	512	AMD	Mac
4	226.983005	Intel Core i5	0	256	Intel	Mac

[55]:
```
df.drop(columns=['OpSys'],inplace=True)
```

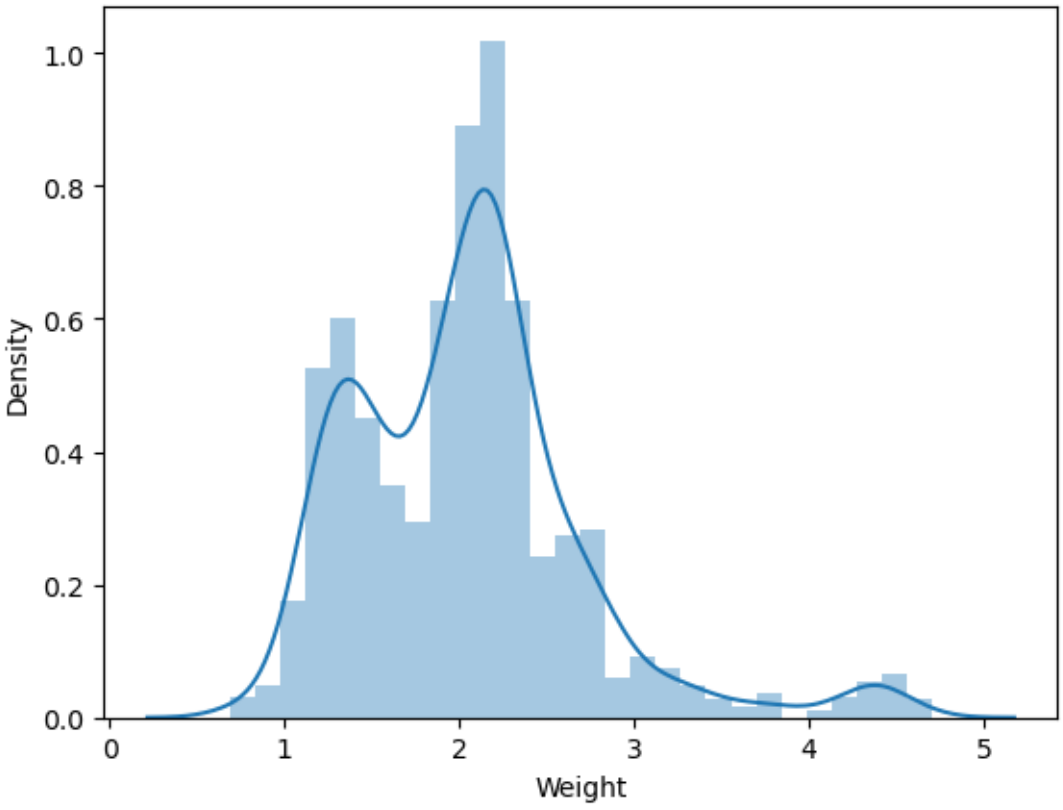
[56]:
```
sns.barplot(x=df['os'],y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



[57]:
```
sns.distplot(df['Weight'])
```

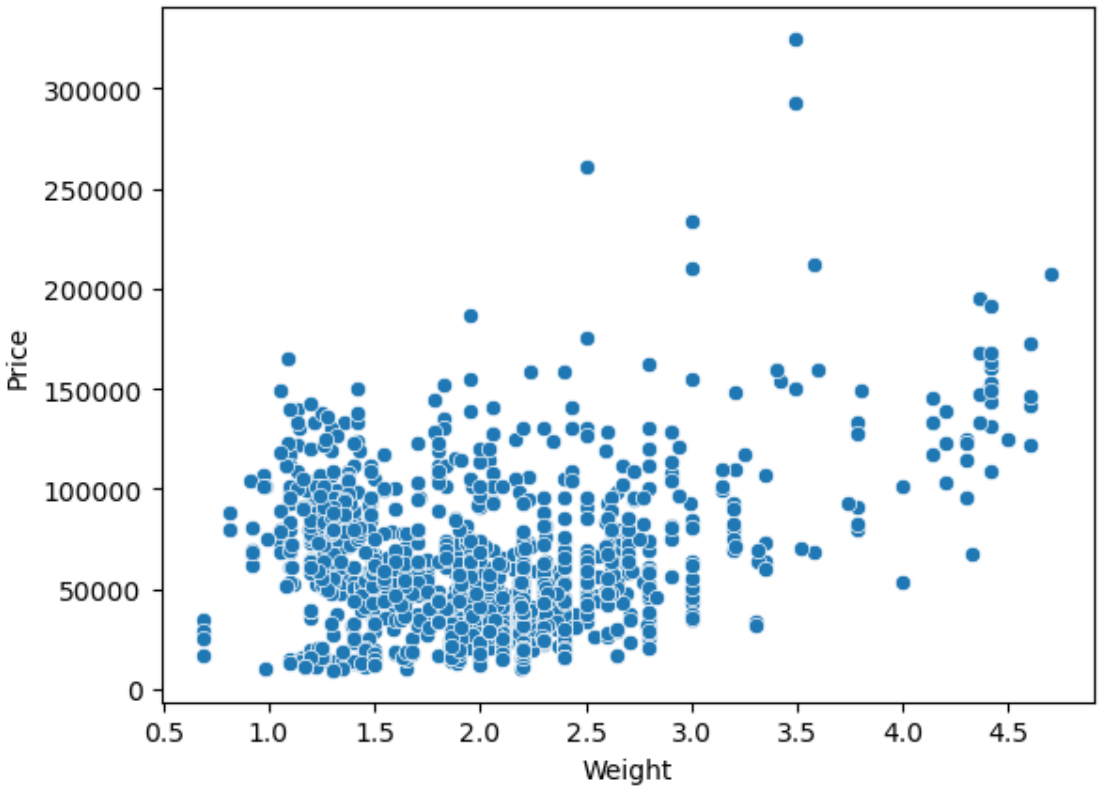
```
C:\Users\hp.pc\AppData\Local\Temp\ipykernel_6820\1125578356.py:1: UserWarning:

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

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

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

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

[57]:
```
<Axes: xlabel='Weight', ylabel='Density'>
```



[58]:
```
sns.scatterplot(x=df['Weight'],y=df['Price'])
```

[58]: <Axes: xlabel='Weight', ylabel='Price'>



[59]:
```
df.corr()['Price']
```

```
C:\Users\hp.pc\AppData\Local\Temp\ipykernel_6820\815546952.py:1: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  df.corr()['Price']
```

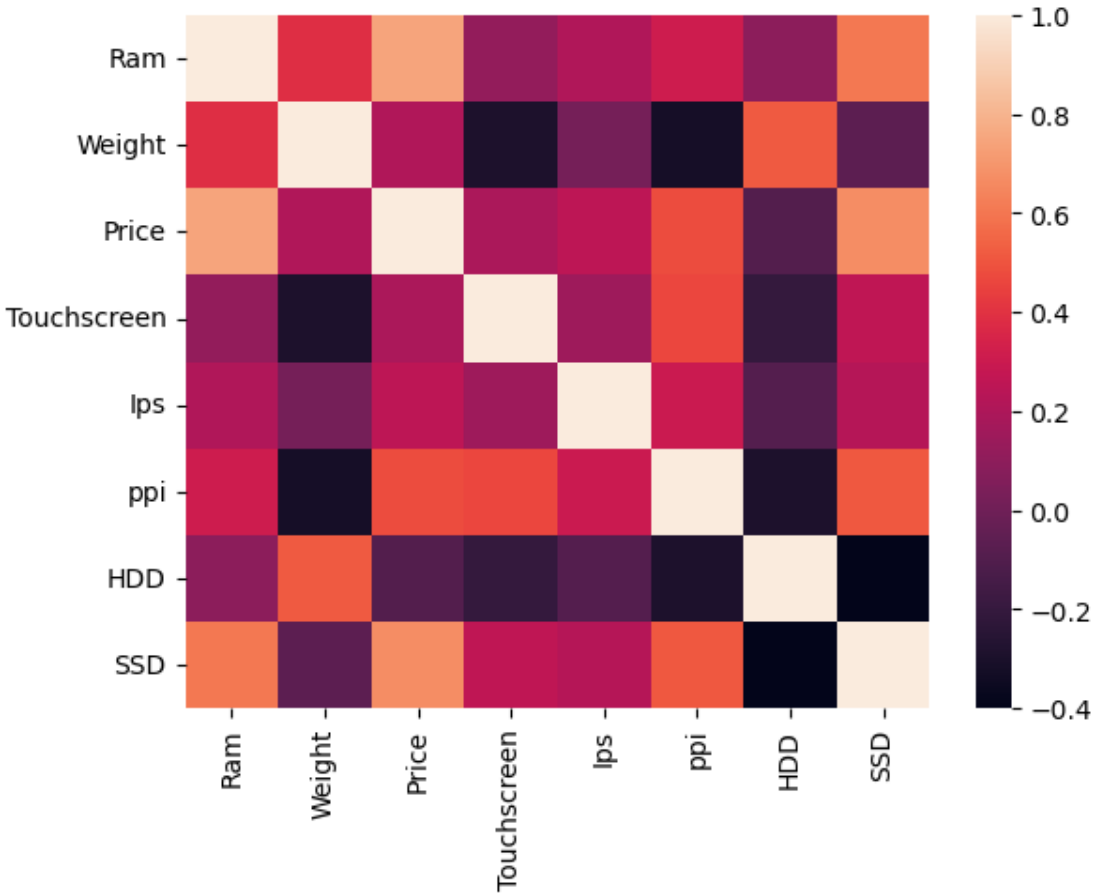
[59]:
```
Ram            0.742905
Weight         0.209867
Price          1.000000
Touchscreen    0.192917
Ips            0.253320
ppi            0.475368
HDD           -0.096891
SSD            0.670660
Name: Price, dtype: float64
```

[60]:
```
sns.heatmap(df.corr())
```

```
C:\Users\hp.pc\AppData\Local\Temp\ipykernel_6820\58359773.py:1: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
```

```
value of numeric_only to silence this warning.
  sns.heatmap(df.corr())
```

[60]: <Axes: >



[61]:
```
X = df.drop(columns=['Price'])
y = df['Price']
```

[62]:
```
X
```

[62]:

	Company	TypeName	Ram	Weight	Touchscreen	Ips	ppi
0	Apple	Ultrabook	8	1.37	0	1	226.983005
1	Apple	Ultrabook	8	1.34	0	0	127.677940
2	HP	Notebook	8	1.86	0	0	141.211998
3	Apple	Ultrabook	16	1.83	0	1	220.534624
4	Apple	Ultrabook	8	1.37	0	1	226.983005
…	…	…	…	…	…	…	…
1298	Lenovo	2 in 1 Convertible	4	1.80	1	1	157.350512
1299	Lenovo	2 in 1 Convertible	16	1.30	1	1	276.053530

```
1300  Lenovo            Notebook    2    1.50            0    0  111.935204
1301      HP            Notebook    6    2.19            0    0  100.454670
1302    Asus            Notebook    4    2.20            0    0  100.454670

                  Cpu brand   HDD  SSD Gpu brand                 os
0             Intel Core i5     0  128     Intel                Mac
1             Intel Core i5     0    0     Intel                Mac
2             Intel Core i5     0  256     Intel  Others/No OS/Linux
3             Intel Core i7     0  512       AMD                Mac
4             Intel Core i5     0  256     Intel                Mac
…                       …   …  …       …                  …
1298          Intel Core i7     0  128     Intel            Windows
1299          Intel Core i7     0  512     Intel            Windows
1300  Other Intel Processor     0    0     Intel            Windows
1301          Intel Core i7  1000    0       AMD            Windows
1302  Other Intel Processor   500    0     Intel            Windows

[1302 rows x 12 columns]
```

[63]:
```
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)
```

[64]:
```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import r2_score, mean_absolute_error
```

[65]:
```
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
```

1 Linear regression

[66]:
```
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 0.7177391544744516
MAE 12535.796993929072
```

```
C:\Users\hp.pc\anaconda3\Lib\site-
packages\sklearn\preprocessing\_encoders.py:972: FutureWarning: `sparse` was
renamed to `sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

2 KNN Classifier

[67]:
```
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\hp.pc\anaconda3\Lib\site-
packages\sklearn\preprocessing\_encoders.py:972: FutureWarning: `sparse` was
renamed to `sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

```
R2 score 0.6641149464793218
MAE 11521.127657142857
```

3 Decision Tree

[68]:
```
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))
```

```
R2 score 0.7513189588769726
MAE 10924.190206594045
```

```
C:\Users\hp.pc\anaconda3\Lib\site-
packages\sklearn\preprocessing\_encoders.py:972: FutureWarning: `sparse` was
renamed to `sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

4 SVM

[69]:
```
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\hp.pc\anaconda3\Lib\site-
packages\sklearn\preprocessing\_encoders.py:972: FutureWarning: `sparse` was
renamed to `sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

```
R2 score 0.5394366345493855
MAE 15429.141206065116
```

5 Random Forest

[70]:
```
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\hp.pc\anaconda3\Lib\site-
packages\sklearn\preprocessing\_encoders.py:972: FutureWarning: `sparse` was
renamed to `sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

```
R2 score 0.8196209205661376
MAE 9296.072297265368
```

[71]:
```
# Replace with your new laptop's features and predict the price
new_laptop_features = np.array([['Apple', 'Notebook', 8, 1.30, 1, 0, 200.
↪234543, 'Intel Core i7', 0, 500, 'Intel', 'Mac']])
predicted_price = pipe.predict(new_laptop_features)
print(f'Predicted Price: Rs {predicted_price[0]:.2f}')
```

```
Predicted Price: Rs 90182.38
```

```
C:\Users\hp.pc\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X
does not have valid feature names, but OneHotEncoder was fitted with feature
names
  warnings.warn(
```

[ ]:

[ ]: