In [2]:
```python
import numpy as np
import pandas as pd
```

In [3]:
```python
df = pd.read_csv('laptops.csv')
```

In [4]:
```python
df.head()
```

Out[4]:

	Laptop	Status	Brand	Model	CPU	RAM	Storage	Storage type	GPU	Screen	Touch	Final Price
0	ASUS ExpertBook B1 B1502CBA-EJ0436X Intel Core...	New	Asus	ExpertBook	Intel Core i5	8	512	SSD	NaN	15.6	No	1009.00
1	Alurin Go Start Intel Celeron N4020/8GB/256GB ...	New	Alurin	Go	Intel Celeron	8	256	SSD	NaN	15.6	No	299.00
2	ASUS ExpertBook B1 B1502CBA-EJ0424X Intel Core...	New	Asus	ExpertBook	Intel Core i3	8	256	SSD	NaN	15.6	No	789.00
3	MSI Katana GF66 12UC-082XES Intel Core i7-1270...	New	MSI	Katana	Intel Core i7	16	1000	SSD	RTX 3050	15.6	No	1199.00
4	HP 15S-FQ5085NS Intel Core i5-1235U/16GB/512GB...	New	HP	15S	Intel Core i5	16	512	SSD	NaN	15.6	No	669.01

In [5]:
```python
df.shape
```

Out[5]:
```
(2160, 12)
```

In [6]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2160 entries, 0 to 2159
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Laptop        2160 non-null   object 
 1   Status        2160 non-null   object 
 2   Brand         2160 non-null   object 
 3   Model         2160 non-null   object 
 4   CPU           2160 non-null   object 
 5   RAM           2160 non-null   int64  
 6   Storage       2160 non-null   int64  
 7   Storage type  2118 non-null   object 
 8   GPU           789 non-null    object 
 9   Screen        2156 non-null   float64
 10  Touch         2160 non-null   object 
 11  Final Price   2160 non-null   float64
dtypes: float64(2), int64(2), object(8)
memory usage: 202.6+ KB
```

In [7]:
```python
df.duplicated().sum()
```

Out[7]:
```
0
```

In [8]:
```python
df.isnull().sum()
```

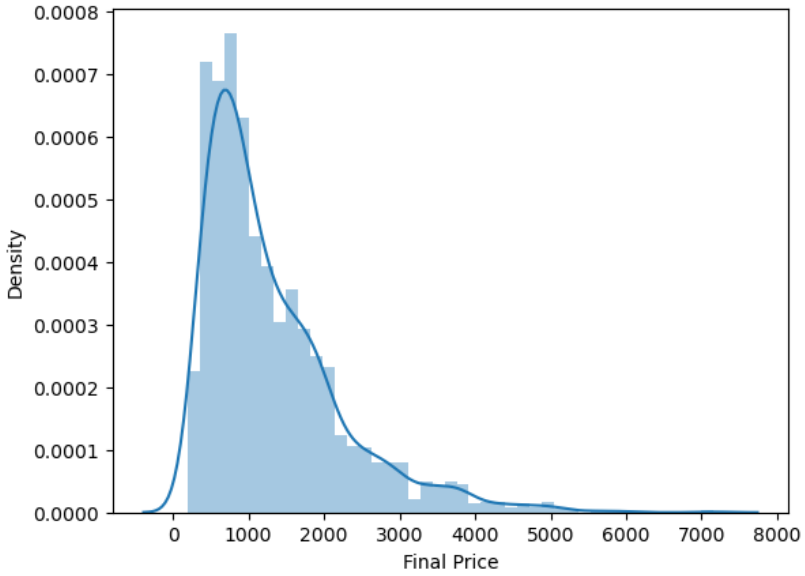
Out[8]:
```
Laptop             0
Status             0
Brand              0
Model              0
CPU                0
RAM                0
Storage            0
Storage type      42
GPU             1371
Screen             4
Touch              0
Final Price        0
dtype: int64
```

In [9]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [10]:
```
sns.distplot(df['Final Price'])
```

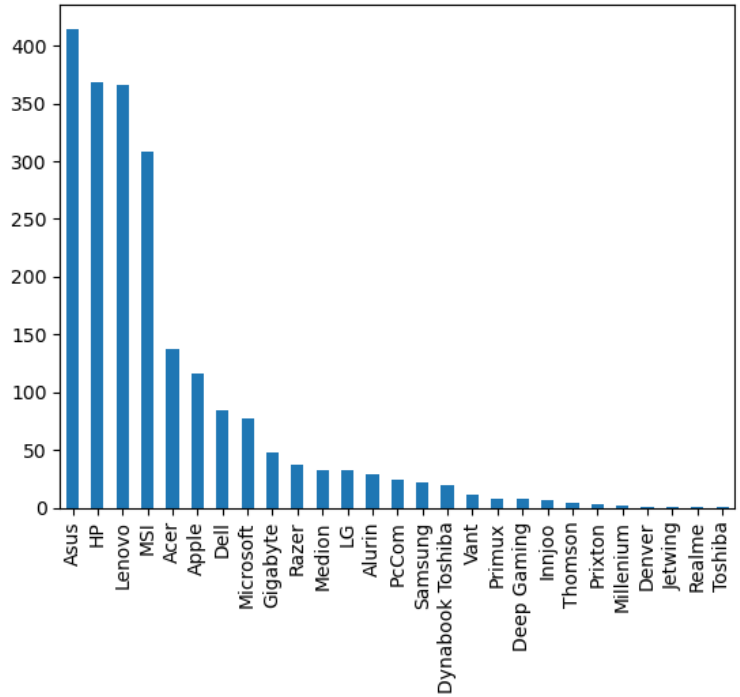
```
C:\Users\Red Devil\AppData\Local\Temp\ipykernel_8300\2393136421.py:1: UserWarning:

`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['Final Price'])
```

Out[10]:
```
<Axes: xlabel='Final Price', ylabel='Density'>
```

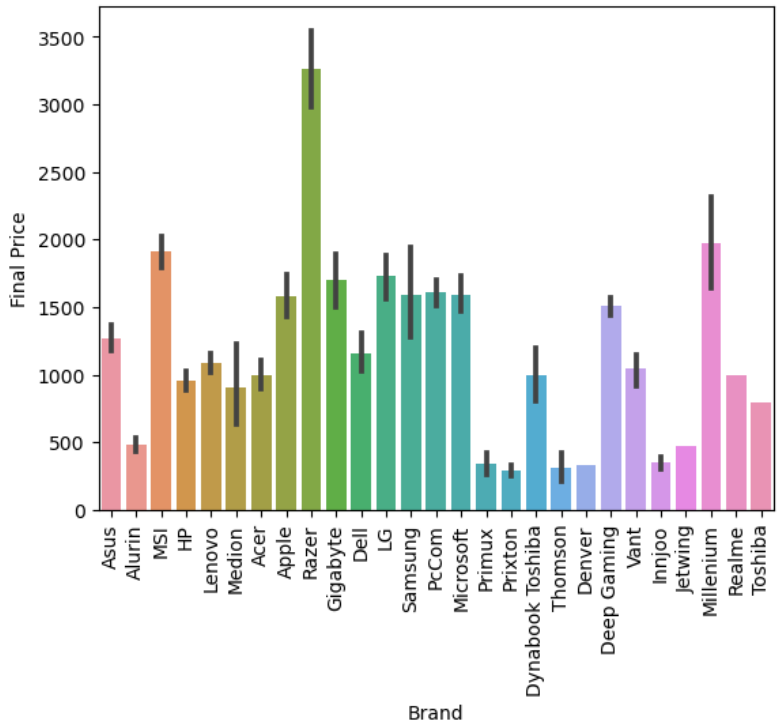


In [11]:
```
df['Brand'].value_counts().plot(kind='bar')
```

Out[11]:
```
<Axes: >
```

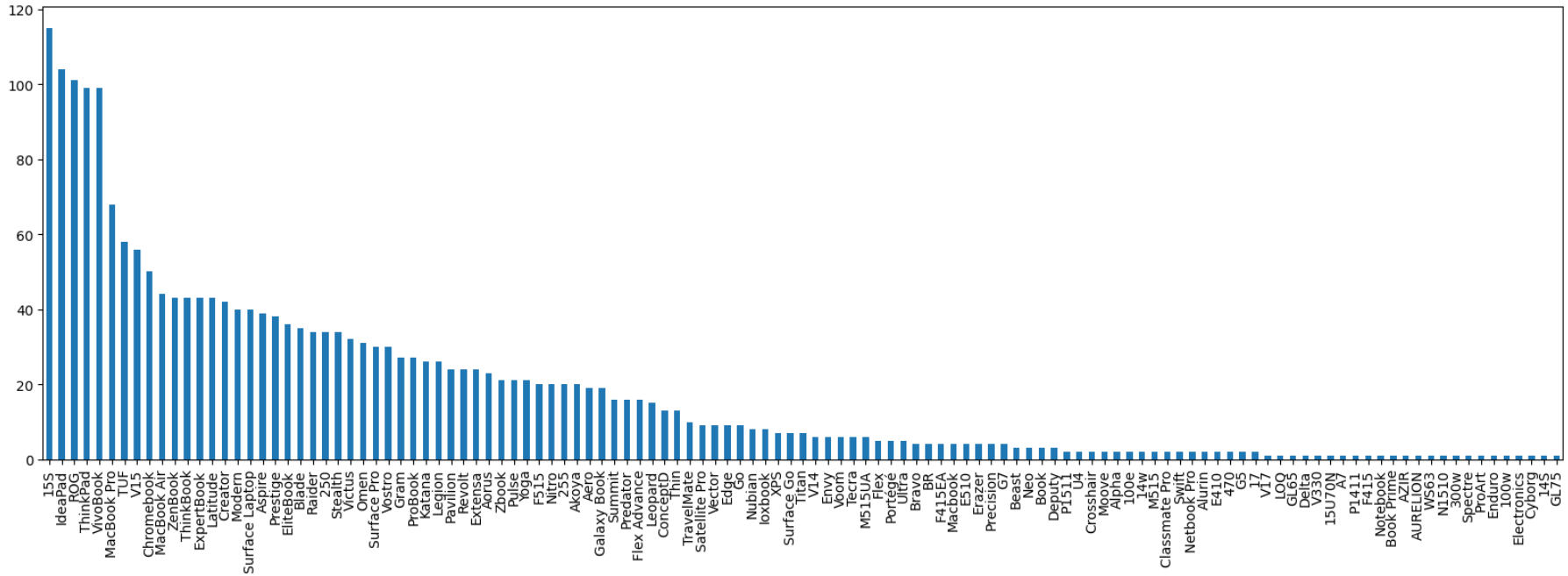


In [12]:
```
sns.barplot(x= df['Brand'], y=df['Final Price'])
plt.xticks(rotation = 90)
plt.show()
```



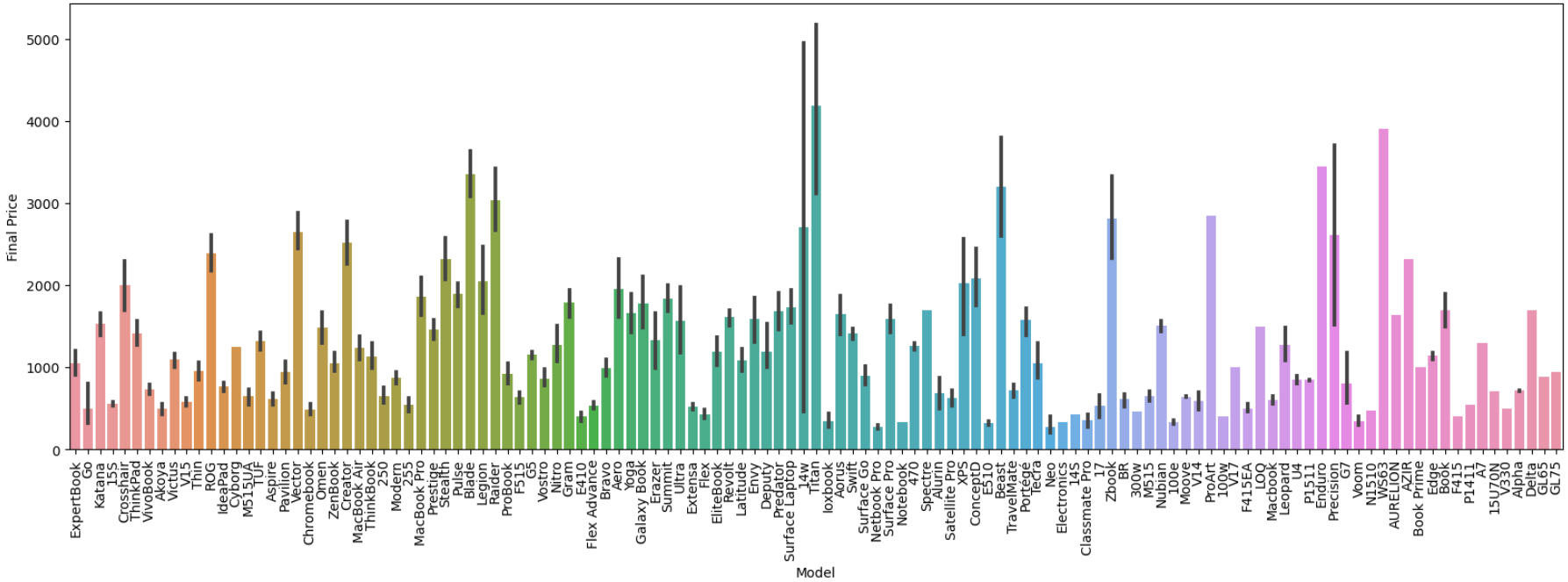
In [13]:
```
plt.figure(figsize=(20,6))
df['Model'].value_counts().plot(kind = 'bar')
```

Out[13]:
```
<Axes: >
```



In [14]:

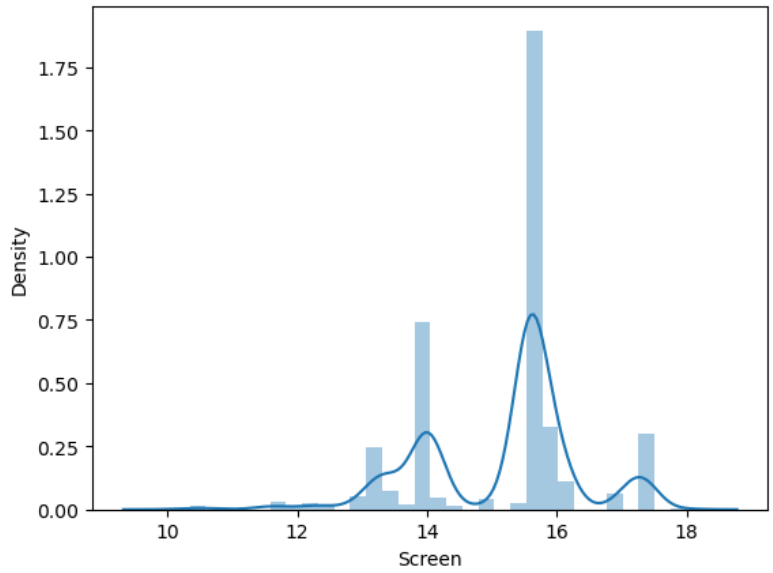
```
plt.figure(figsize = (20,6))
sns.barplot(x =df['Model'], y = df['Final Price'])
plt.xticks(rotation = 90)
plt.show()
```



In [15]:
```
sns.distplot(df['Screen'])
```

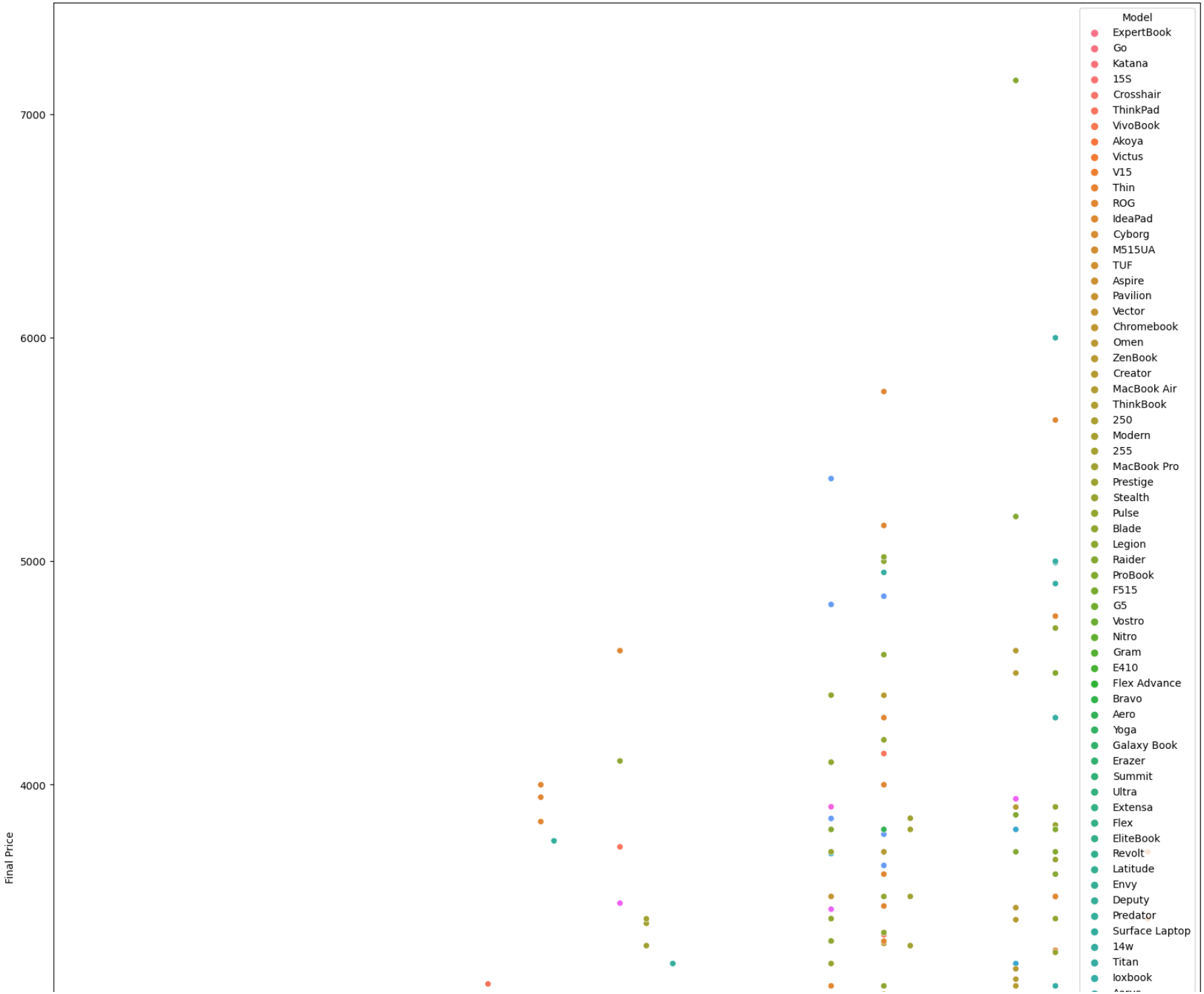
```
C:\Users\Red Devil\AppData\Local\Temp\ipykernel_8300\2896879728.py:1: UserWarning:

`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['Screen'])
```

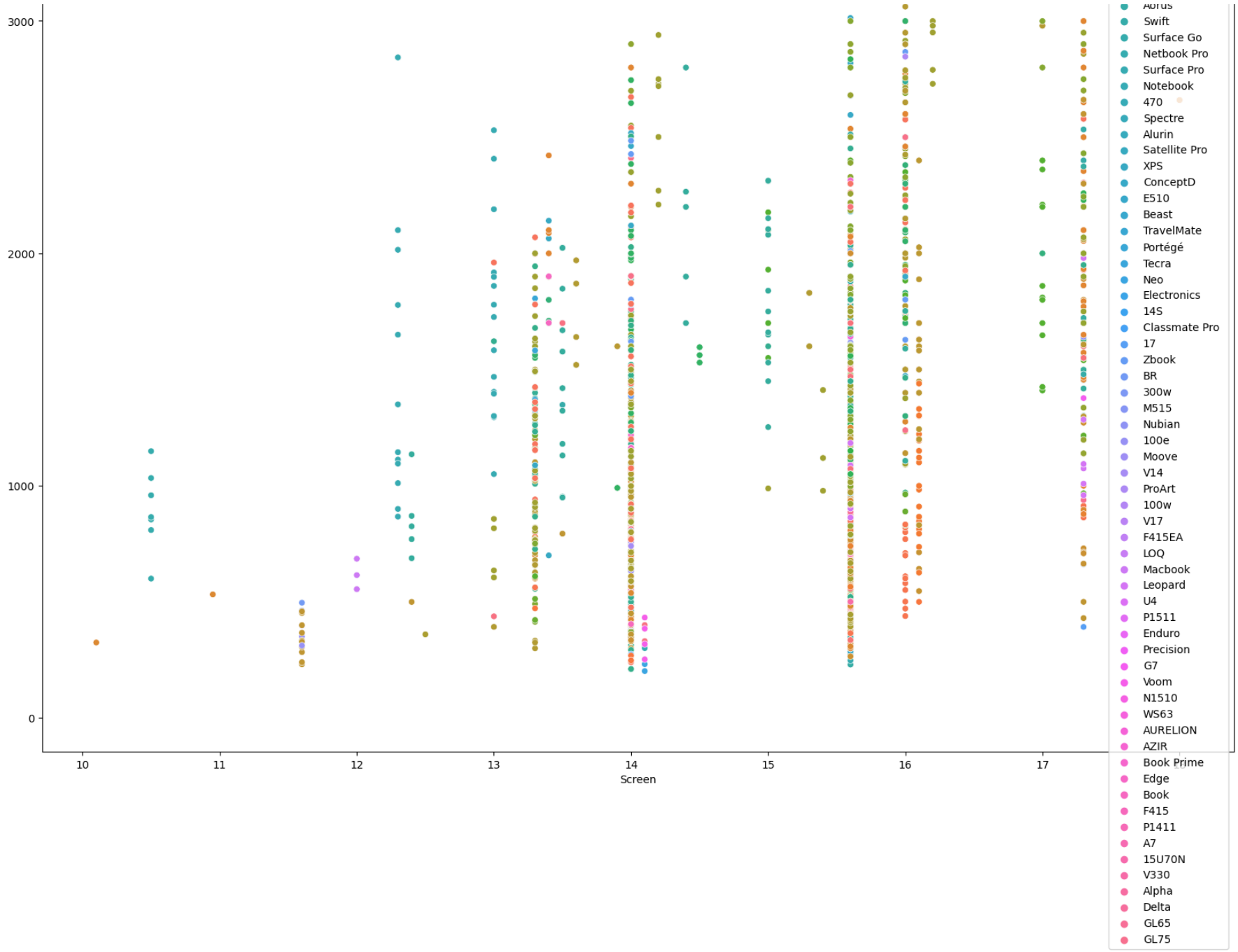
Out[15]:
```
<Axes: xlabel='Screen', ylabel='Density'>
```



In [16]:
```
plt.figure(figsize = (20,30))
sns.scatterplot(data  =df ,x='Screen', y = 'Final Price', hue = 'Model')
```

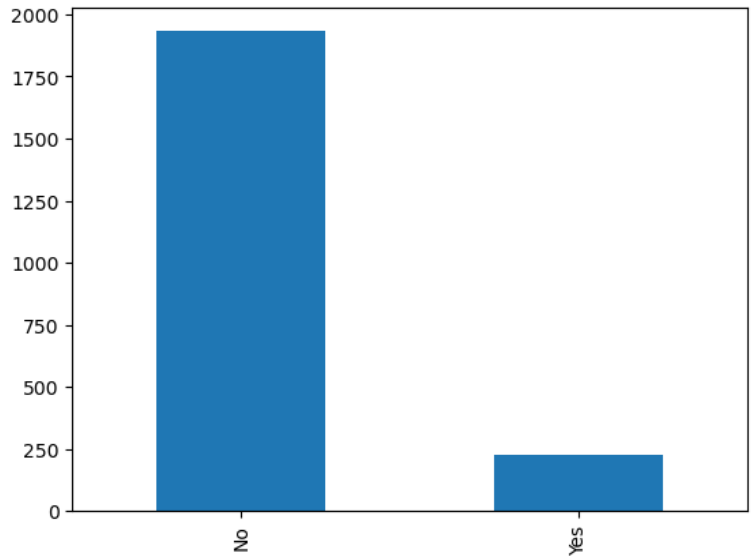
Out[16]:
```
<Axes: xlabel='Screen', ylabel='Final Price'>
```


Final Price
7000
6000
5000
4000
Model
ExpertBook
Go
Katana
15S
Crosshair
ThinkPad
VivoBook
Akoya
Victus
V15
Thin
ROG
IdeaPad
Cyborg
M515UA
TUF
Aspire
Pavilion
Vector
Chromebook
Omen
ZenBook
Creator
MacBook Air
ThinkBook
250
Modern
255
MacBook Pro
Prestige
Stealth
Pulse
Blade
Legion
Raider
ProBook
F515
G5
Vostro
Nitro
Gram
E410
Flex Advance
Bravo
Aero
Yoga
Galaxy Book
Erazer
Summit
Ultra
Extensa
Flex
EliteBook
Revolt
Latitude
Envy
Deputy
Predator
Surface Laptop
14w
Titan
loxbook



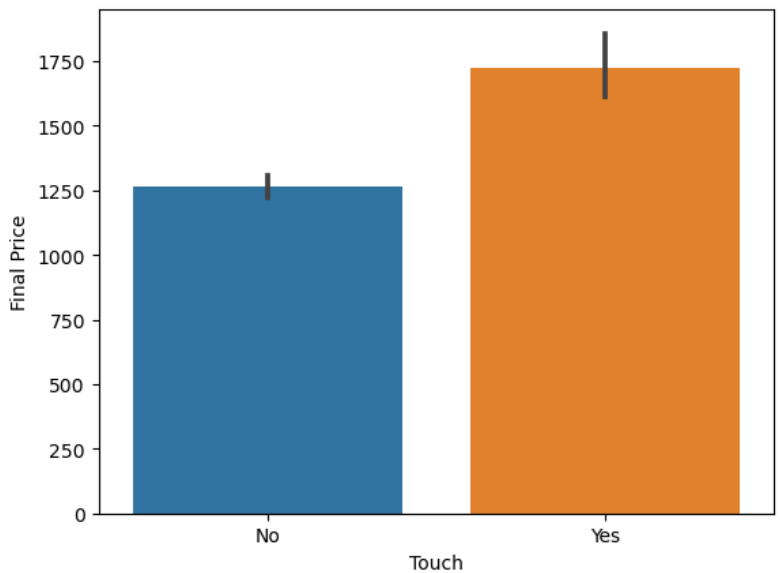
In [17]:
```
df['Touch'].value_counts().plot(kind = 'bar')
```

Out[17]:
```
<Axes: >
```



In [18]:
```
sns.barplot(data = df, x= 'Touch', y=  'Final Price')
```

Out[18]:
```
<Axes: xlabel='Touch', ylabel='Final Price'>
```



In [19]:
```
df.corr()
```

```
C:\Users\Red Devil\AppData\Local\Temp\ipykernel_8300\1134722465.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will de
fault to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.corr()
```

Out[19]:

	RAM	Storage	Screen	Final Price
RAM	1.000000	0.751297	0.361404	0.724946
Storage	0.751297	1.000000	0.398025	0.695631
Screen	0.361404	0.398025	1.000000	0.268359
Final Price	0.724946	0.695631	0.268359	1.000000

In [20]:
```
df['CPU'].value_counts()
```

Out[20]:
```
Intel Core i7            710
Intel Core i5            535
AMD Ryzen 7              156
Intel Core i3            130
AMD Ryzen 5              127
Intel Core i9             94
Intel Celeron             94
Intel Evo Core i7         82
AMD Ryzen 9               44
AMD Ryzen 3               44
Intel Evo Core i5         30
Apple M2                  28
AMD 3020e                 13
Apple M2 Pro              13
Apple M1                  11
AMD Athlon                10
Intel Pentium             10
Apple M1 Pro               7
Intel Core M3              5
AMD 3015e                  3
Microsoft SQ1              3
Qualcomm Snapdragon 7      3
AMD Radeon 9               2
Qualcomm Snapdragon 8      2
Intel Evo Core i9          1
Mediatek MT8183            1
AMD 3015Ce                 1
AMD Radeon 5               1
Name: CPU, dtype: int64
```

In [21]:
```
df['CPU Name'] = df['CPU'].apply(lambda x : " ".join(x.split()[0:3]))
```

In [22]:
```
df.head()
```

Out[22]:

	Laptop	Status	Brand	Model	CPU	RAM	Storage	Storage type	GPU	Screen	Touch	Final Price	CPU Name
0	ASUS ExpertBook B1 B1502CBA-EJ0436X Intel Core...	New	Asus	ExpertBook	Intel Core i5	8	512	SSD	NaN	15.6	No	1009.00	Intel Core i5
1	Alurin Go Start Intel Celeron N4020/8GB/256GB ...	New	Alurin	Go	Intel Celeron	8	256	SSD	NaN	15.6	No	299.00	Intel Celeron
2	ASUS ExpertBook B1 B1502CBA-EJ0424X Intel Core...	New	Asus	ExpertBook	Intel Core i3	8	256	SSD	NaN	15.6	No	789.00	Intel Core i3
3	MSI Katana GF66 12UC-082XES Intel Core i7-1270...	New	MSI	Katana	Intel Core i7	16	1000	SSD	RTX 3050	15.6	No	1199.00	Intel Core i7
4	HP 15S-FQ5085NS Intel Core i5-1235U/16GB/512GB...	New	HP	15S	Intel Core i5	16	512	SSD	NaN	15.6	No	669.01	Intel Core i5

In [23]:
```
def fetchprocessor(text):
    if text == 'Intel Core i7' or text == ' Intel Core i5' or text == 'Intel Core i3':
        return text
    else:
        if text == text.split()[0] == 'Intel':
            return 'other intel processor'
        else:
            return 'AMD processor'
```

In [24]:
```
df['CPU Brand']= df['CPU Name'].apply(fetchprocessor)
```

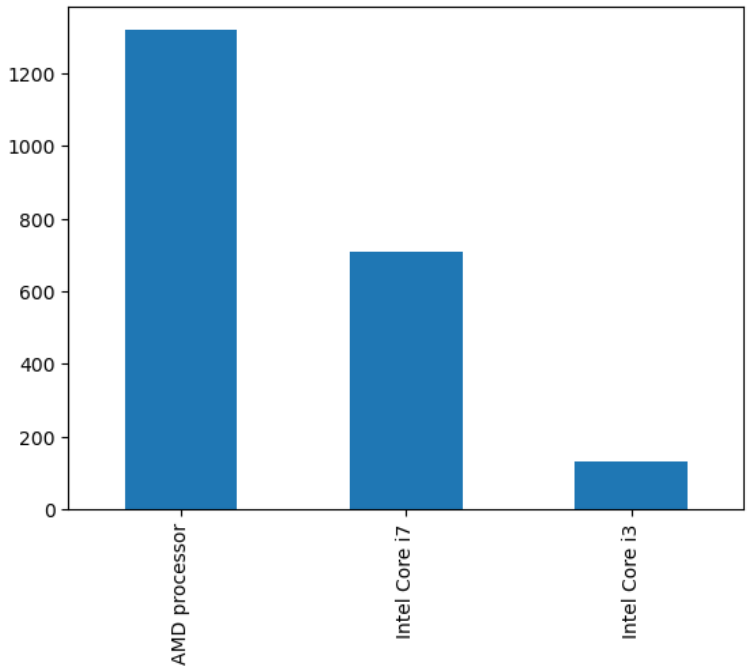
In [25]:
```
df.sample()
```

Out[25]:

	Laptop	Status	Brand	Model	CPU	RAM	Storage	Storage type	GPU	Screen	Touch	Final Price	CPU Name	CPU Brand
142	MSI Stealth 15 A13VF-028XES Intel Core i7-1362...	New	MSI	Stealth	Intel Core i7	32	1000	SSD	RTX 4060	15.6	No	1899.0	Intel Core i7	Intel Core i7

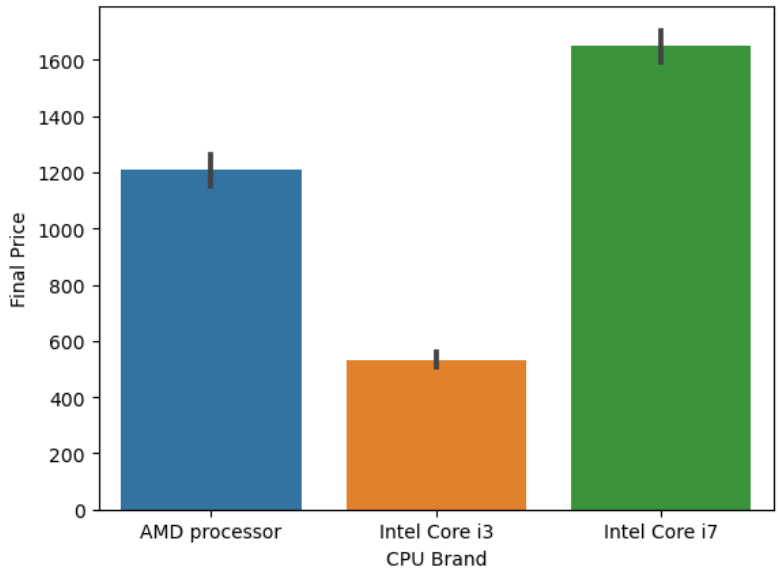
In [26]:
```
df['CPU Brand'].value_counts().plot(kind = 'bar')
```

Out[26]:
```
<Axes: >
```



In [27]:
```
sns.barplot(data = df, x = 'CPU Brand', y = 'Final Price')
```

Out[27]:
```
<Axes: xlabel='CPU Brand', ylabel='Final Price'>
```



In [28]:
```
df.drop(columns = 'CPU', axis =1 ,inplace = True)
df.drop(columns = 'CPU Name', axis =1 ,inplace = True)
```

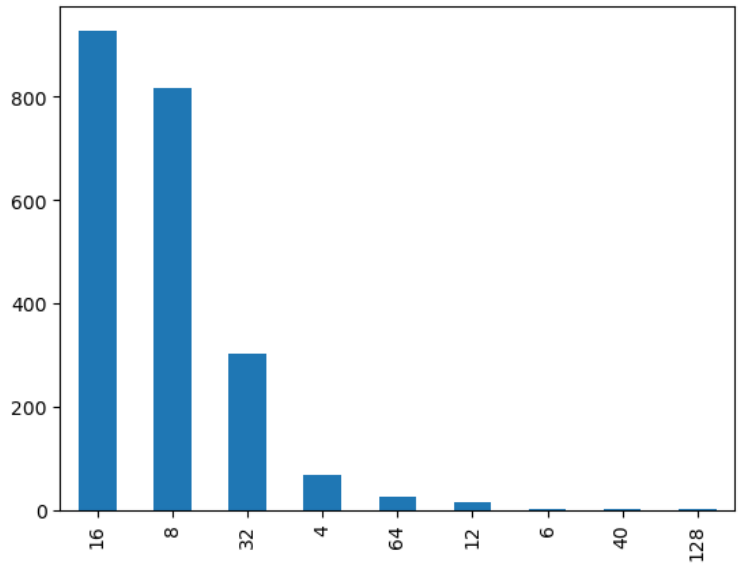
In [29]:
```
df.sample()
```

Out[29]:

	Laptop	Status	Brand	Model	RAM	Storage	Storage type	GPU	Screen	Touch	Final Price	CPU Brand
28	HP Pavilion 15-eh1004ns AMD Ryzen 5 5500U/16GB...	New	HP	Pavilion	16	512	SSD	NaN	15.6	No	799.0	AMD processor

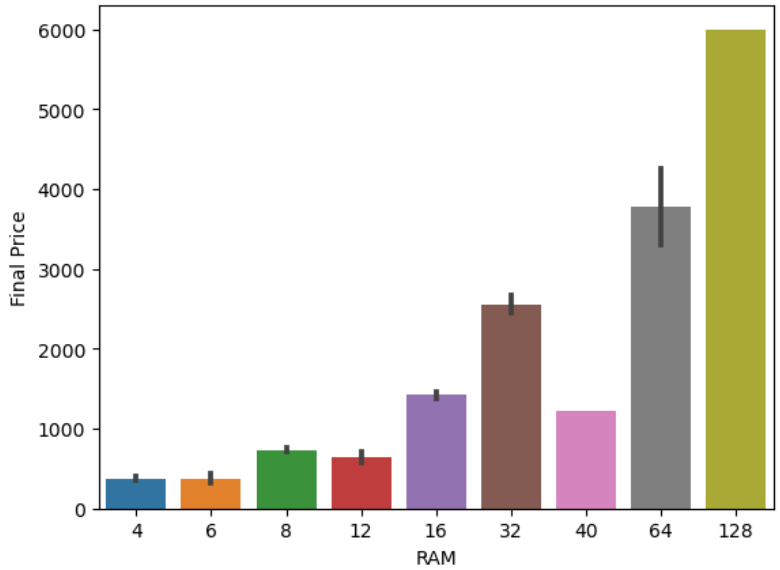
In [30]:
```
df['RAM'].value_counts().plot(kind = 'bar')
```

Out[30]:
```
<Axes: >
```



In [31]:
```
sns.barplot(data = df , x = 'RAM' , y= 'Final Price')
```

Out[31]:
```
<Axes: xlabel='RAM', ylabel='Final Price'>
```



In [32]:
```
df["GPU"].value_counts()
```

Out[32]:

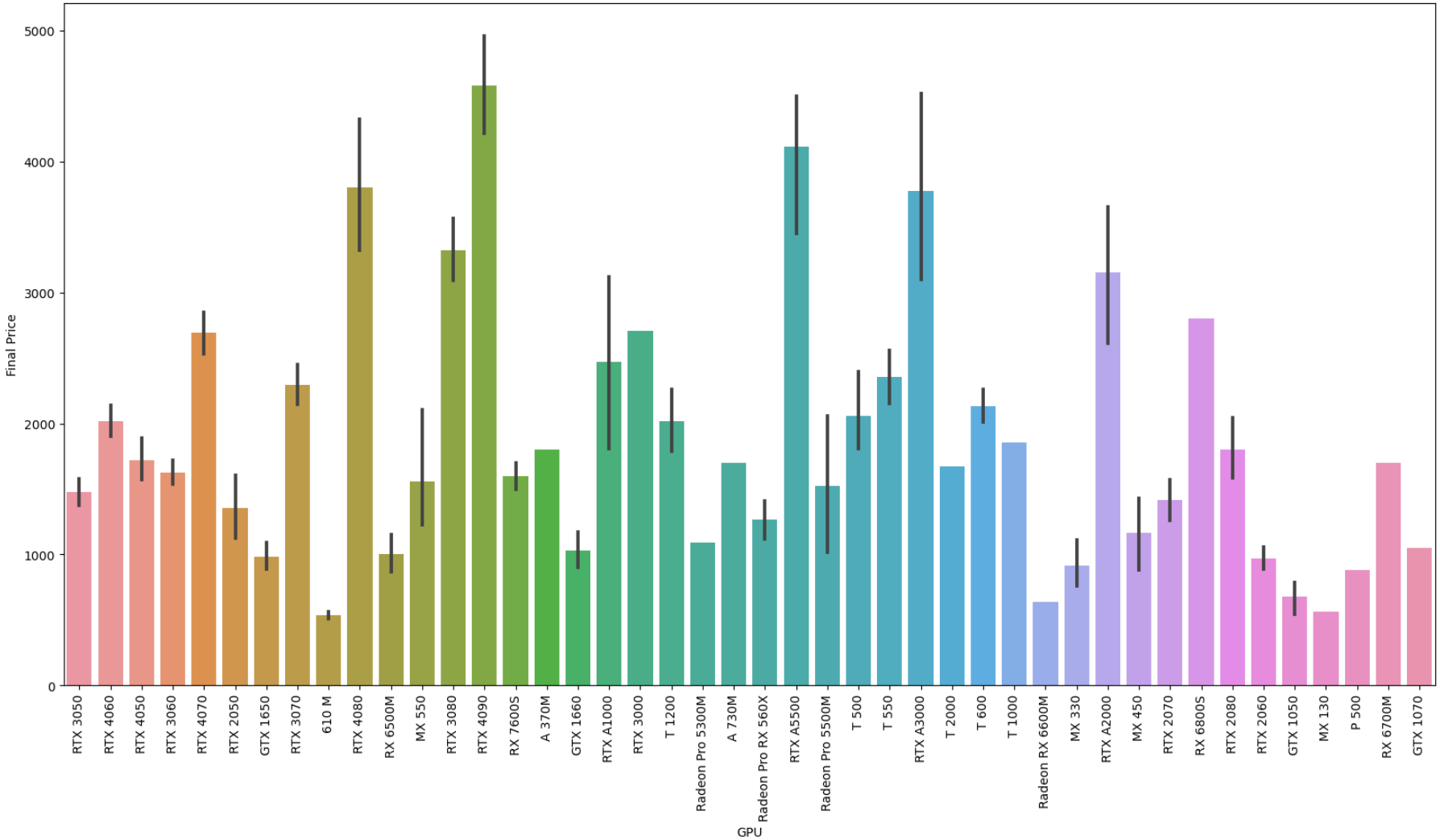
```
RTX 3050              129
RTX 3060              122
RTX 3070               97
RTX 4060               62
RTX 3080               51
GTX 1650               50
RTX 4070               40
RTX 4050               33
RTX 2070               25
RTX 2060               20
RTX 4080               18
RTX 4090               17
GTX 1660               14
RTX 2050               11
Radeon Pro 5500M        9
RTX A2000               8
RTX 2080                7
MX 450                  7
RTX A3000               7
RTX A1000               6
T 1200                  6
MX 550                  5
MX 330                  5
T 500                   5
T 550                   4
GTX 1050                3
Radeon Pro RX 560X      3
RTX A5500               3
RX 6500M                3
RX 7600S                2
T 600                   2
Radeon Pro 5300M        2
610 M                   2
RX 6700M                1
P 500                   1
MX 130                  1
A 370M                  1
RX 6800S                1
T 2000                  1
RTX 3000                1
A 730M                  1
Radeon RX 6600M         1
T 1000                  1
GTX 1070                1
Name: GPU, dtype: int64
```

In [33]:

```
plt.figure(figsize = (20,10))
sns.barplot(data = df, x= 'GPU', y= 'Final Price')
plt.xticks(rotation = 90)
```

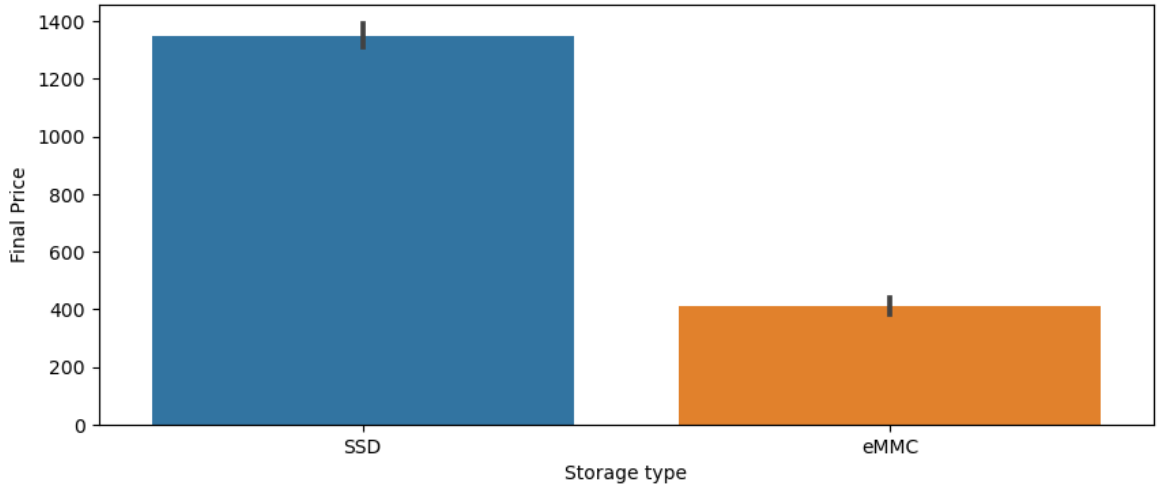
Out[33]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43]),
 [Text(0, 0, 'RTX 3050'),
  Text(1, 0, 'RTX 4060'),
  Text(2, 0, 'RTX 4050'),
  Text(3, 0, 'RTX 3060'),
  Text(4, 0, 'RTX 4070'),
  Text(5, 0, 'RTX 2050'),
  Text(6, 0, 'GTX 1650'),
  Text(7, 0, 'RTX 3070'),
  Text(8, 0, '610 M'),
  Text(9, 0, 'RTX 4080'),
  Text(10, 0, 'RX 6500M'),
  Text(11, 0, 'MX 550'),
  Text(12, 0, 'RTX 3080'),
  Text(13, 0, 'RTX 4090'),
  Text(14, 0, 'RX 7600S'),
  Text(15, 0, 'A 370M'),
  Text(16, 0, 'GTX 1660'),
  Text(17, 0, 'RTX A1000'),
  Text(18, 0, 'RTX 3000'),
  Text(19, 0, 'T 1200'),
  Text(20, 0, 'Radeon Pro 5300M'),
  Text(21, 0, 'A 730M'),
  Text(22, 0, 'Radeon Pro RX 560X'),
  Text(23, 0, 'RTX A5500'),
  Text(24, 0, 'Radeon Pro 5500M'),
  Text(25, 0, 'T 500'),
  Text(26, 0, 'T 550'),
  Text(27, 0, 'RTX A3000'),
  Text(28, 0, 'T 2000'),
  Text(29, 0, 'T 600'),
  Text(30, 0, 'T 1000'),
  Text(31, 0, 'Radeon RX 6600M'),
  Text(32, 0, 'MX 330'),
  Text(33, 0, 'RTX A2000'),
  Text(34, 0, 'MX 450'),
  Text(35, 0, 'RTX 2070'),
  Text(36, 0, 'RX 6800S'),
  Text(37, 0, 'RTX 2080'),
  Text(38, 0, 'RTX 2060'),
  Text(39, 0, 'GTX 1050'),
  Text(40, 0, 'MX 130'),
  Text(41, 0, 'P 500'),
  Text(42, 0, 'RX 6700M'),
  Text(43, 0, 'GTX 1070')])
```



In [34]:
```
plt.figure(figsize = (10,4))
sns.barplot(data = df, x= 'Storage type', y = 'Final Price')
```

Out[34]:
```
<Axes: xlabel='Storage type', ylabel='Final Price'>
```



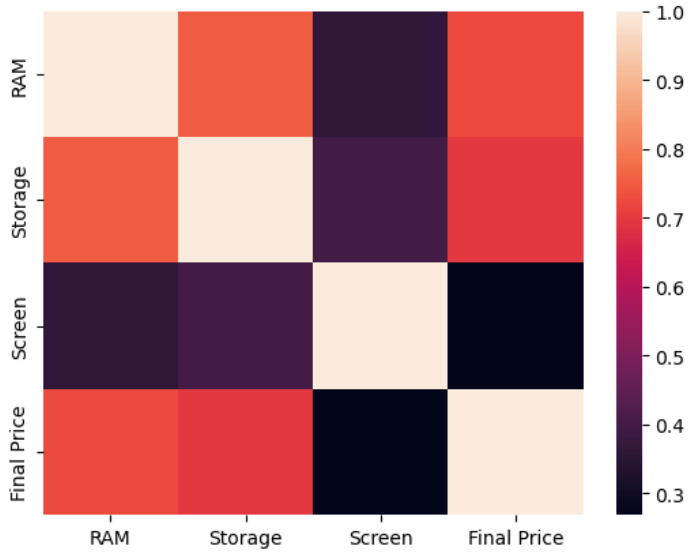
In [35]:
```
df.columns
```

Out[35]:
```
Index(['Laptop', 'Status', 'Brand', 'Model', 'RAM', 'Storage', 'Storage type',
       'GPU', 'Screen', 'Touch', 'Final Price', 'CPU Brand'],
      dtype='object')
```

In [36]:
```
sns.heatmap(df.corr())
```

```
C:\Users\Red Devil\AppData\Local\Temp\ipykernel_8300\58359773.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will defa
ult to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr())
```

Out[36]:
```
<Axes: >
```

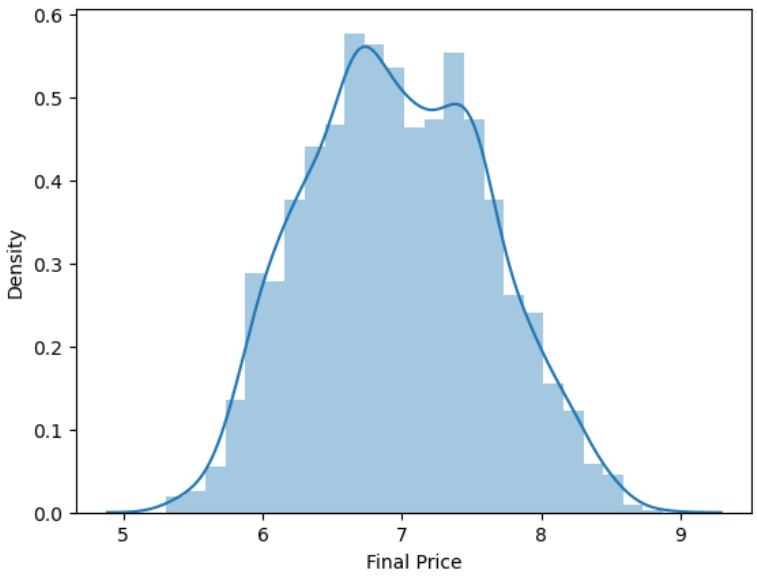


In [37]:
```
sns.distplot(np.log(df['Final Price']))
```

```
C:\Users\Red Devil\AppData\Local\Temp\ipykernel_8300\3897261032.py:1: UserWarning:

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

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

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

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

Out[37]:

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



In [38]:

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

In [39]:

```
x
```

Out[39]:

	Laptop	Status	Brand	Model	RAM	Storage	Storage type	GPU	Screen	Touch	CPU Brand
0	ASUS ExpertBook B1 B1502CBA-EJ0436X Intel Core...	New	Asus	ExpertBook	8	512	SSD	NaN	15.6	No	AMD processor
1	Alurin Go Start Intel Celeron N4020/8GB/256GB ...	New	Alurin	Go	8	256	SSD	NaN	15.6	No	AMD processor
2	ASUS ExpertBook B1 B1502CBA-EJ0424X Intel Core...	New	Asus	ExpertBook	8	256	SSD	NaN	15.6	No	Intel Core i3
3	MSI Katana GF66 12UC-082XES Intel Core i7-1270...	New	MSI	Katana	16	1000	SSD	RTX 3050	15.6	No	Intel Core i7
4	HP 15S-FQ5085NS Intel Core i5-1235U/16GB/512GB...	New	HP	15S	16	512	SSD	NaN	15.6	No	AMD processor
...	...	...	...	...	...	...	...	...	...	...	...
2155	Razer Blade 17 FHD 360Hz Intel Core i7-11800H/...	Refurbished	Razer	Blade	16	1000	SSD	RTX 3060	17.3	No	Intel Core i7
2156	Razer Blade 17 FHD 360Hz Intel Core i7-11800H/...	Refurbished	Razer	Blade	16	1000	SSD	RTX 3070	17.3	No	Intel Core i7
2157	Razer Blade 17 FHD 360Hz Intel Core i7-11800H/...	Refurbished	Razer	Blade	32	1000	SSD	RTX 3080	17.3	No	Intel Core i7
2158	Razer Book 13 Intel Evo Core i7-1165G7/16GB/1T...	Refurbished	Razer	Book	16	1000	SSD	NaN	13.4	Yes	AMD processor
2159	Razer Book FHD+ Intel Evo Core i7-1165G7/16GB/...	Refurbished	Razer	Book	16	256	SSD	NaN	13.4	Yes	AMD processor

2160 rows × 11 columns

In [40]:
```
y
```

Out[40]:
```
0       6.916715
1       5.700444
2       6.670766
3       7.089243
4       6.505799
          ...
2155    7.901003
2156    7.972463
2157    8.131528
2158    7.549604
2159    7.438378
Name: Final Price, Length: 2160, dtype: float64
```

In [41]:
```
from sklearn.model_selection import train_test_split
```

In [42]:
```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
```

In [43]:
```
from sklearn.cluster import KMeans
```

In [44]:
```
df.shape
df = pd.get_dummies(df)
df.fillna(value = 0 , inplace = True)
```

In [45]:
```
train = df[0:1650]
test = df[1651:]
```

In [46]:
```
test.head()
```

Out[46]:

	RAM	Storage	Screen	Final Price	Laptop_ASUS ROG Zephyrus M16 GU604VI-93D47PB1 Intel Core i9-13900H/32GB/1TB SSD/RTX 4070/16" (PT)	Laptop_ASUS BR1100FKA-BP1185XA Intel Celeron N4500/4GB/128GB SSD/11.6" Táctil	Laptop_ASUS Chromebook C433TA-AJ0336 Intel Core m3-8100Y/8GB/64GB eMMC/14" Táctil	Laptop_ASUS Chromebook CR1 CR1100CKA-GJ0132 Intel Celeron N4500/4GB/32GB/11.6"	Laptop_ASUS Chromebook CR1100FKA-BP0024 Intel Celeron N4500/4GB/32GB eMMC/11.6" Táctil	Laptop_ASUS Chromebook CX1400CKA-EK0138 Intel Celeron N4500/8GB/64GB eMMC/14"	...	GPU_T 1200	GPU_T 2000	GPU_T 500	GPU_T 550	GPU_T 600	Touch_No	Tou
1651	8	128	13.0	1582.00	0	0	0	0	0	0	...	0	0	0	0	0	0	
1652	16	256	13.0	1467.58	0	0	0	0	0	0	...	0	0	0	0	0	0	
1653	16	512	13.0	2406.35	0	0	0	0	0	0	...	0	0	0	0	0	0	
1654	16	500	15.6	1640.98	0	0	0	0	0	0	...	0	0	0	0	0	1	
1655	16	500	15.6	2312.71	0	0	0	0	0	0	...	0	0	0	0	0	1	

5 rows × 2365 columns

In [47]:
```
x_train = train.drop('Final Price', axis =1)
y_train =train['Final Price']
x_test = test.drop('Final Price',axis= 1)
y_test = test['Final Price']
```

In [48]:
```
true_p = test['Final Price']
```

In [49]:
```
lreg = LinearRegression()
```

In [50]:
```
lreg.fit(x_train,y_train)
```

Out[50]:
```
▾ LinearRegression
LinearRegression()
```

In [51]:
```
x_train = pd.get_dummies(x_train)
```

In [52]:
```
x_train.shape
```

Out[52]:
```
(1650, 2364)
```

In [53]:
```
x_test =pd.get_dummies(x_test)
```

In [54]:
```
x_test.shape
```

Out[54]:
```
(509, 2364)
```

In [55]:
```
x_train.fillna(0,inplace = True)
```

In [56]:
```
x_test.fillna(0, inplace = True)
```

In [57]:
```
pred = lreg.predict(x_test)
```

In [58]:
```
lreg.score(x_test, true_p)
```

Out[58]:
```
0.7389815907949435
```

In [59]:
```
lreg.score(x_train, y_train)
```

Out[59]: 1.0

In [60]:
```
print(pred)
```

```
[1500.46200765 1765.96391425 1934.05831902 1282.84851596 1216.8445166
 1122.79167871  571.26718579  326.33884441 3150.12656024  937.98364552
  759.64530598  358.42478079  805.38817133 1031.5265909  1359.83597524
 1359.83597524 1031.5265909  1904.20008785 1904.20008785  956.369283
 1031.5265909  1031.5265909   619.7249373   397.48299315  218.55087851
  392.606016    392.606016    147.67767461  392.606016    649.2180281
  392.606016    347.37896794  681.91411213 1912.7427048  1282.93585651
 2859.73908659 2058.82064186 2376.30749351 2376.30749351 2376.30749351
 5562.94628047 2026.8093399  2859.73908659 3523.88992631 1175.92924289
  794.96516792 1646.00726116  512.62147369  163.07236471  512.62147369
  726.87944414  726.87944414  395.16468137   52.2957221   150.23633999
  485.89203348  485.89203348  317.7976287   470.32198928  150.23633999
  738.73742666  288.20246057  470.32198928  538.42889752  217.75390178
 1268.49287115  378.98291468  779.99641623  430.44730724  855.15372413
  888.9676344   964.1249423   964.1249423  1345.08901727 1619.42986298
 1377.10031924  993.41797961 1370.11233027 1370.11233027 2050.50859033
 1740.55380973  954.65898753  954.65898753  954.65898753  947.44572174
 1031.49292413  947.44572174  947.44572174 1031.49292413 1031.49292413
 1199.58732891  784.99277245  587.82220044 2105.29261586 2142.95297111
 2142.95297111 1784.86265675 2468.20202427 2088.345456   1418.3663879
 1334.31918551 1334.31918551 1418.3663879  1418.3663879  1418.3663879
 1586.46079268 1586.46079268 2088.345456   2088.345456   1767.91549689
 1418.3663879  1418.3663879  1586.46079268 1586.46079268 1714.52177302
 1354.690781   1354.690781   2163.5027639  1037.14197742 3089.4352918
   46.49622433 -186.94887514  852.4679703   363.94360641  543.50890761
  467.7633239   375.41450283 3169.11597576 1663.89993149 1663.89993149
 1663.89993149 1159.61246917 2859.73908659 2376.30749351 1464.26631325
 1464.26631325 1440.76083554 1737.66624459 1874.96313865 4204.98792589
 1546.04642764 1480.04242828 1981.38678314 2344.29619155 2376.30749351
 1249.12194624 1874.96313865 1799.80583075 3617.56920537 1874.96313865
 1799.80583075 1440.76083554 1404.88512038 1159.61246917 1480.04242828
 1191.62377113  930.65659168 1251.08655079  916.91443178 1175.92924289
 1493.41609454 1207.94054485 1081.58121676  761.15125765  761.15125765
  596.48447267  916.91443178 1006.42390886  982.91843114  301.03848529
   86.2384281   497.05142949  331.16676949  331.16676949  557.65049739
  587.77878159  301.03848529  557.65049739  802.03675205 1064.50159975
  726.87944414  545.42473994  802.03675205  802.03675205  718.54918668
  718.54918668  718.54918668  708.04328638  664.63026849  900.00389089
  718.54918668  483.62604626  440.31663772  608.4110425   651.72045104
 1336.15502867 1590.88402301  450.64695967  618.74136445  640.88910209
 1400.63524536 1126.88547801 1447.31543712  867.8706669  1113.14331811
  867.8706669  1764.80228877 2127.71169718  867.8706669   867.8706669
 3232.14812404  899.88196886  946.48779763  946.48779763  946.48779763
  946.48779763 1205.50260874 1525.93256785 1885.8988688  1843.4194195
 1843.4194195  2753.78302491 1917.91017076 1525.93256785 1012.49179699
   98.47410579  266.56851057  266.56851057  266.56851057   21.64016918
  365.21291446  266.56851057  305.95286891  104.27402217  940.15223851
  653.59111729  653.59111729 1078.29753417  741.92954953 1091.47865851
  696.90052582  696.90052582 1121.60694271 1004.39439724  272.36842695
  664.57890546  926.64910134 1512.56585291 1270.23630916 1633.14571757
  814.11492629 1396.72711961 1973.22878237 1001.80640924 1016.15857841
 1503.1507641  1185.66391245  715.74962779  454.20053532  800.32202451
  710.81254743  641.46353345  414.65716231  415.06619623  428.24732057
  458.37560477  458.37560477  807.92471375  883.08202165  732.62285637
  841.59407454 1852.87641888 1544.36150689 3137.98448397 2491.1867627
  927.03867335  794.57949482  975.85502393 1972.29693644  351.50995247
  321.38166827  321.38166827  183.41554769  532.96465667  351.50995247
  876.53740196  183.41554769  106.58161108  274.67601586  608.12196458
  426.66726037  502.83637247  747.34225612  427.25660683  672.18494822
 1112.40942689  697.63144165  351.50995247  258.57285559 1306.09588384
 1306.09588384 1626.52584296 1122.23838064 2382.07941092 1709.21522191
 2829.89706911 1020.14797522 1211.74785771  465.52264481  891.9056904
  465.52264481  891.9056904   572.48486561  572.48486561  495.65092901
  961.43651192  898.62769924 1480.51337173 1142.89121612  780.1609828
  868.49941504  823.47039134  991.56479612  898.62769924 1142.89121612
 1142.89121612  793.34210714 1142.89121612  793.34210714  868.49941504
```

```
 868.49941504 1218.04852402  823.47039134 1036.59381982 1454.18957452
1568.85180397 1480.51337173  328.71284621  328.71284621  111.03281327
 460.58192226  535.73923016   34.19887667  160.10159592 1784.97181805
1281.20413697 1308.53364817 1509.325846    774.62649998  843.90632085
1040.84478502 1943.44068842 1592.63740847 1624.64871044 1557.32008287
3224.78271799 1859.18429063 1496.27488222 2209.98757058 3834.75469982
4369.02777249 3643.20895567 1890.61210053 1164.33627996 2252.94667879
2511.9614899  2829.44834155 2754.29103365 2829.44834155 3344.89123659
3344.89123659 2284.95798075 3376.90253855 2861.45964351  719.5837201
 719.5837201   630.07424303  813.93174624  696.07824239  662.08554499
 921.1003561   919.06362875 1180.99431093  794.56035246 1114.99031157
1271.50488982 1147.00161353 1147.00161353 1147.00161353  860.56435182
1374.00512268 1374.00512268 2054.40138274 1147.00161353 1723.50327629
1723.50327629 1723.50327629 1042.44247811 1984.79665674 3609.56378598
2817.72640343 1757.79314759 2414.41191626 1596.63499099 2979.07257649
1257.98983824 1318.52395411  816.63929079 1315.58084665 1318.52395411
 998.093995   1030.10529696 1435.44665757 1072.53724916  550.39318286
 625.55049076  945.98044987  945.98044987  625.55049076  550.39318286
 559.08130922  238.65135011  921.99071763  368.93847865  580.52146706
 976.10873407  589.20959342  952.11900183  952.11900183  399.06676285
1592.63740847 1264.31567752 1264.31567752  943.8857184   943.8857184
1373.28689569  854.37624133  974.01400261 1294.44396172 1657.35337013
1657.35337013  974.01400261 1657.35337013 1766.3245883  1425.00503382
1425.00503382 1742.49188547 1742.49188547 1819.92574419 1457.01633578
1457.01633578 1457.01633578 1774.50318743 1774.50318743 2511.9614899
2829.44834155 3269.73392869 2829.44834155 2861.45964351 3376.90253855
4168.7399211  4684.18281614 3376.90253855 4168.7399211  4684.18281614
1428.87643658 1362.87243722 1984.79665674 1653.89855029 2817.72640343
2817.72640343 2849.73770539 1799.55973547 1900.36731912 1097.14515369
1930.49560333 3480.41693134 3995.85982638 2201.873309   1870.97520255
2158.72730306 2749.32750601 -237.83022819 1301.65847736 1622.08843647
1133.56407258 1156.88926237 1240.93646476 1836.3875829  1836.3875829
 963.5101683  1419.63155117 1769.12970478  867.8706669  3970.3816681
3092.02936465 3435.60210993 2774.542513   3092.02936465 3970.3816681
3970.3816681  2454.11255388 2771.59940554 3927.23566216 2806.55381496
3124.04066661 4002.39297006 2092.77401004 1604.24964615]
```

In [61]:
```
rsme = np.sqrt(np.power((np.array(true_p) - np.array(pred)), 2 ))
```

In [62]:
```
rsme_train = np.sqrt(np.power((np.array(y_train) - np.array(lreg.predict(x_train))), 2 ))
```

In [63]:
```
print(rsme)
```

```
[8.15379923e+01 2.98383914e+02 4.72291681e+02 3.58131484e+02
 1.09586548e+03 9.28016787e+01 1.32807186e+02 1.99411556e+01
 5.01465602e+01 6.10163545e+01 2.60915306e+02 7.81352192e+01
 6.38817133e+00 9.14734091e+01 1.97835975e+02 1.97835975e+02
 9.14734091e+01 6.87190088e+02 6.87190088e+02 3.76307170e+01
 9.14734091e+01 9.14734091e+01 1.22750627e+01 1.51517007e+02
 2.80449121e+02 1.27383984e+02 2.06393984e+02 3.31322325e+02
 2.06393984e+02 9.97819719e+01 2.66393984e+02 2.21621032e+02
 3.29141121e+01 2.08625730e+03 3.33935857e+02 2.39260913e+02
 4.01793581e+01 4.22692506e+02 2.30674935e+01 6.22692506e+02
 1.56394628e+03 4.26829340e+02 9.70799087e+02 4.75110074e+02
 6.23070757e+02 4.04034832e+02 1.52992739e+02 2.36378526e+02
 4.35927635e+02 1.86378526e+02 3.72120556e+02 1.17839444e+02
 3.03835319e+02 2.11844278e+02 1.66703660e+02 8.61203348e+00
 1.12402033e+02 7.34423713e+01 5.50801072e+00 4.18763660e+02
 7.22525733e+01 4.60375394e+01 1.20558011e+02 8.94288975e+01
 1.81246098e+02 6.30507129e+02 4.45970853e+01 1.09466416e+02
 4.71426928e+01 1.79913724e+02 2.22277634e+02 7.81849423e+01
 2.34875058e+02 2.53910983e+02 4.04799863e+02 4.11060319e+02
 5.05582020e+02 4.55472330e+02 2.71822330e+02 7.05818590e+02
 1.05361903e+01 2.44658988e+02 2.27288988e+02 1.87268988e+02
 3.21535722e+02 1.57892924e+02 2.02295722e+02 2.89195722e+02
 6.65870759e+01 2.76492924e+02 1.79577329e+02 4.60992772e+02
 2.62577996e+01 3.94717384e+02 8.36047029e+02 8.07047029e+02
 7.15147343e+02 3.18079757e+01 8.93454560e+01 3.88606388e+02
 4.27819186e+02 4.27819186e+02 6.53696388e+02 4.91716388e+02
 6.53696388e+02 7.82000793e+02 5.58960793e+02 5.97015456e+02
 5.97015456e+02 4.68165497e+02 6.01966388e+02 2.63356388e+02
 5.23120793e+02 2.86710793e+02 5.96271773e+02 5.59592190e+01
 5.59592190e+01 5.31002764e+02 4.81858023e+02 3.78035292e+02
 3.52503776e+02 4.85948875e+02 1.53467970e+02 1.05056394e+02
 1.40278908e+02 2.31236676e+02 1.73585497e+02 1.07011598e+03
 6.43059931e+02 4.80219931e+02 3.64899931e+02 5.32724692e+01
 7.88449087e+02 4.44667494e+02 6.92476313e+02 6.12176313e+02
 4.17608355e+01 1.06133376e+03 8.20731387e+01 7.05977926e+02
 6.65856428e+02 6.23802428e+02 4.82386783e+02 9.46756192e+02
 3.77307494e+02 4.33681946e+02 4.09173139e+02 2.95658307e+01
 7.46279205e+02 3.03223139e+02 6.15341693e+01 5.86170836e+02
 3.13795120e+02 9.23224692e+01 2.08952428e+02 4.57066229e+02
 1.21834083e+01 4.20696551e+02 1.59384432e+02 7.89242889e-01
 5.05583905e+02 6.22894551e+01 4.29391217e+02 1.55061258e+02
 3.35887424e+01 1.69605527e+02 6.32244318e+01 3.59033909e+02
 1.97228431e+02 1.94701515e+02 2.77591572e+02 3.01948571e+02
 2.18823231e+02 3.67833231e+02 2.41349503e+02 2.11221218e+02
 3.97961515e+02 4.41349503e+02 4.69632480e+01 3.54884002e+01
 7.21205559e+01 2.61152601e+01 1.47963248e+02 4.69632480e+01
 1.39349187e+02 2.29549187e+02 2.19549187e+02 2.09043286e+02
 1.75630268e+02 3.71003891e+02 1.99539187e+02 1.33913954e+02
 1.95766377e+01 9.68210425e+01 1.01790451e+02 6.37155029e+02
 5.04004023e+02 2.30569597e+01 2.11251364e+02 2.48110898e+02
 1.04745245e+02 2.06804522e+02 2.64035437e+02 2.60669333e+02
 2.00496682e+02 2.43069333e+02 1.34197711e+02 5.70021697e+02
 7.37319333e+02 3.96119333e+02 5.66851876e+02 1.35795803e+03
 1.37997798e+02 1.96522024e+01 9.94522024e+01 1.07977976e+01
 1.62262609e+02 7.30674321e+01 5.84958869e+02 1.44419420e+02
 5.38629420e+02 3.45216975e+02 4.39520171e+02 1.73067432e+02
 1.77251797e+02 3.50525894e+02 1.41121489e+02 2.88921489e+02
 1.54701489e+02 3.12899831e+02 2.43727086e+02 2.09421489e+02
 8.67371311e+01 3.06415978e+02 7.13897761e+02 7.22988827e+01
 1.02442888e+03 8.65212466e+02 2.03929550e+02 5.97581341e+02
 1.76900526e+02 3.77159474e+02 7.67643057e+02 1.38394397e+02
 3.26631573e+02 1.20111095e+02 6.32091013e+01 2.70175853e+02
 2.28763691e+02 4.34505718e+02 1.52150737e+01 9.96871196e+01
 3.66788782e+02 8.99664092e+01 3.39318578e+02 3.03160764e+02
 2.73223912e+02 9.42096278e+01 9.11794647e+01 1.59182025e+02
 1.12745257e+00 2.19164665e+01 1.50882838e+02 3.34923804e+02
```

```
9.06952679e+02 1.11614395e+02 5.73514395e+02 4.90225286e+02
7.20017978e+02 1.07842856e+02 1.05754075e+02 2.33416419e+02
1.13346849e+03 7.10025516e+02 2.31446324e+03 2.22798673e+02
1.85539495e+02 2.74575024e+02 5.48106936e+02 2.17530048e+02
3.77618332e+02 3.27618332e+02 1.15584452e+02 4.77305343e+02
3.62320048e+02 1.55474020e+01 2.97534452e+02 2.00508389e+02
4.45113984e+02 2.58738035e+02 5.07572740e+02 4.96163628e+02
1.51657744e+02 1.09933393e+02 1.26815052e+02 1.36590573e+02
2.01368558e+02 4.51100048e+02 2.43417144e+02 1.56055884e+02
7.09588384e+00 6.30485843e+02 3.53271619e+02 5.83079411e+02
6.17564778e+02 5.07792931e+02 1.44607975e+02 3.50957858e+02
2.54267355e+02 1.57094310e+02 1.19767355e+02 8.42743096e+01
1.53694866e+02 3.97995134e+02 6.92609290e+01 7.88765119e+01
1.62707699e+02 3.28673372e+02 6.88312161e+01 2.51299017e+02
3.30500585e+02 2.30896087e+01 3.39964796e+02 2.09976992e+01
2.96368784e+02 4.04891216e+02 4.58567893e+02 1.06208878e+03
3.18342107e+02 1.00749415e+02 2.13794150e+01 3.19208524e+02
2.14703913e+01 4.49273820e+02 7.73890425e+02 5.69851804e+02
2.98486628e+02 4.10897154e+02 3.47237154e+02 3.34757187e+02
4.54488078e+02 2.50400770e+02 5.29721123e+02 3.39388404e+02
2.14028182e+02 2.91734137e+02 5.21693648e+02 4.89674154e+02
5.34865000e+01 1.31966321e+02 1.08155215e+02 5.29400688e+02
2.52591525e-01 4.34841290e+02 1.58320083e+02 2.24217282e+02
2.89815709e+02 5.02725118e+02 4.89012429e+02 1.35754700e+02
2.29972228e+02 2.55791044e+02 4.08387899e+02 5.34663720e+02
5.06616679e+02 1.22853149e+03 1.46310834e+03 9.33641034e+02
8.69208342e+02 1.22970124e+03 1.45560124e+03 5.67777981e+02
9.47262539e+02 6.18279644e+02 1.69062799e+01 7.28537201e+01
1.09305757e+02 7.44174624e+00 1.10201758e+02 3.36604455e+02
4.40103561e+01 3.32236287e+01 1.37054311e+02 6.74296475e+01
2.13900312e+02 3.34024890e+02 7.36616135e+01 1.39411614e+02
1.88961614e+02 3.21965648e+02 2.87365123e+02 3.51215123e+02
1.05401383e+02 5.45216135e+01 4.39963276e+02 2.55496724e+02
2.75496724e+02 3.87737522e+02 3.25556657e+02 9.29683786e+02
5.61696403e+02 5.61313148e+02 3.46571916e+02 1.24455009e+02
6.05932576e+02 3.41049838e+02 3.80476046e+02 6.52360709e+02
2.44140847e+02 4.30476046e+02 5.00906005e+02 5.18894703e+02
5.94606658e+02 1.72547249e+02 3.78968171e+01 2.34495092e+01
2.34040450e+02 3.01590450e+02 3.73449509e+02 2.48606817e+02
8.89918691e+02 4.38138650e+02 7.57492824e+01 2.73451521e+02
5.06585329e+01 3.10368734e+02 1.23780407e+02 4.46880998e+02
1.71170998e+02 4.51232371e+01 1.71552592e+02 1.39525678e+02
3.34684322e+02 1.00745718e+02 4.05114282e+02 1.25713104e+02
2.94623759e+02 1.84824003e+02 2.72053962e+02 6.42813370e+02
6.43163370e+02 5.54985997e+02 6.16813370e+02 1.67324588e+02
1.73994966e+02 3.73994966e+02 1.56508115e+02 3.56508115e+02
6.81695744e+02 1.22326336e+02 2.41983664e+02 4.41983664e+02
2.24496813e+02 4.24496813e+02 4.12961490e+02 3.04483415e+01
1.29266071e+02 1.69551658e+02 1.12459644e+02 3.22097461e+02
3.69739921e+02 1.48171839e+01 4.22097461e+02 5.69739921e+02
1.85182816e+02 6.03236437e+02 4.41832437e+02 2.35796657e+02
9.51014497e+01 3.81273597e+02 8.81273597e+02 3.99262295e+02
1.00559735e+02 3.17527319e+02 1.37184846e+02 6.10405603e+02
8.18583069e+02 1.00314017e+03 2.97126691e+02 4.28024797e+02
4.40272697e+02 1.49672494e+02 4.84720228e+02 4.02648477e+02
1.54701564e+02 3.95340726e+01 1.07889262e+02 5.80635352e+01
3.64875829e+01 1.13512417e+02 8.54898317e+01 2.93684488e+01
1.79870295e+02 1.81119333e+02 2.12224167e+03 1.92039365e+02
5.69032110e+02 2.74562513e+02 1.92039365e+02 6.70391668e+02
5.70391668e+02 1.22137255e+03 1.18820941e+03 7.72754338e+02
1.06563815e+02 2.24050667e+02 6.02402970e+02 1.92784010e+02
9.57403538e+01]
```

In [64]:
```
print(rsme_train)
```

```
[2.18153673e-09 2.81715984e-10 2.22144081e-10 ... 5.59339242e-11
 2.38060238e-10 1.42335921e-10]
```

In [65]:
```
from sklearn.cluster import KMeans
```

In [66]:
```
kmeans = KMeans(n_clusters =2)
```

In [67]:
```
kmeans.fit(df)
```

```
C:\Users\Red Devil\anaconda3.4\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n
_init` explicitly to suppress the warning
  warnings.warn(
```

Out[67]:
```
▾ KMeans
KMeans(n_clusters=2)
```

In [68]:
```
pred = kmeans.predict(df)
```

In [69]:
```
pred
```

Out[69]:
```
array([0, 0, 0, ..., 1, 1, 0])
```

In [70]:
```
pd.Series(pred).value_counts()
```

Out[70]:
```
0    1610
1     550
dtype: int64
```

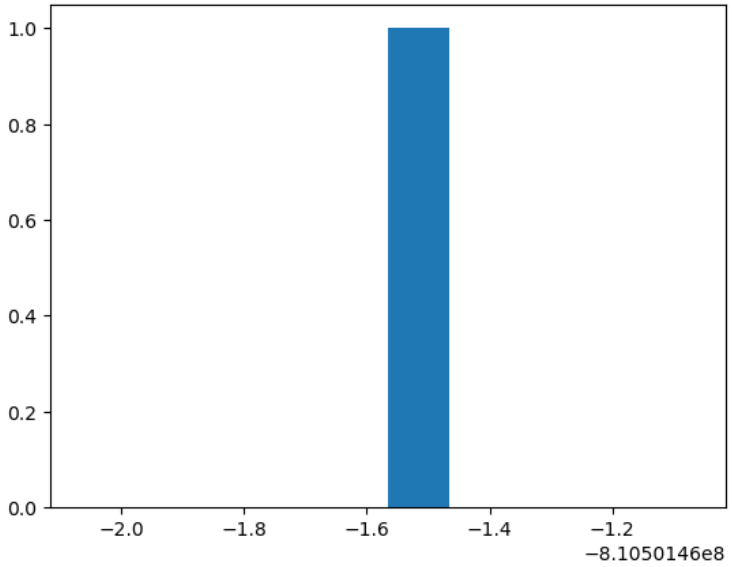
In [71]:
```
kmeans.inertia_
```

Out[71]:
```
810501461.5657579
```

In [77]:
```
k=kmeans.score(df)
```

In [80]:
```
plt.hist(k)
```

Out[80]:
```
(array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.]),
 array([-8.10501462e+08, -8.10501462e+08, -8.10501462e+08, -8.10501462e+08,
        -8.10501462e+08, -8.10501462e+08, -8.10501461e+08, -8.10501461e+08,
        -8.10501461e+08, -8.10501461e+08, -8.10501461e+08]),
 <BarContainer object of 10 artists>)
```



In [ ]: