Import necessary libraries

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, A
from xgboost import XGBRegressor
```

Load the dataset

In [4]:
```
data = pd.read_csv('C:/Users/91700/OneDrive/Desktop/SARTHAK/Project UM/Datas/lap
```

Information Related to Data

In [6]:
```
data.head()
```

Out[6]:

	Company	Product	TypeName	Inches	Ram	OS	Weight	Price_euros	Screer
0	Apple	MacBook Pro	Ultrabook	13.3	8	macOS	1.37	1339.69	Standarc
1	Apple	Macbook Air	Ultrabook	13.3	8	macOS	1.34	898.94	Standarc
2	HP	250 G6	Notebook	15.6	8	No OS	1.86	575.00	Full HD
3	Apple	MacBook Pro	Ultrabook	15.4	16	macOS	1.83	2537.45	Standarc
4	Apple	MacBook Pro	Ultrabook	13.3	8	macOS	1.37	1803.60	Standarc

5 rows × 23 columns

In [7]:
```
data.shape
```

Out[7]:
```
(1275, 23)
```

In [8]:
```
data.describe()
```

Out[8]:

	Inches	Ram	Weight	Price_euros	ScreenW	ScreenH	1
count	1275.000000	1275.000000	1275.000000	1275.000000	1275.000000	1275.000000	
mean	15.022902	8.440784	2.040525	1134.969059	1900.043922	1073.904314	
std	1.429470	5.097809	0.669196	700.752504	493.346186	283.883940	
min	10.100000	2.000000	0.690000	174.000000	1366.000000	768.000000	
25%	14.000000	4.000000	1.500000	609.000000	1920.000000	1080.000000	
50%	15.600000	8.000000	2.040000	989.000000	1920.000000	1080.000000	
75%	15.600000	8.000000	2.310000	1496.500000	1920.000000	1080.000000	
max	18.400000	64.000000	4.700000	6099.000000	3840.000000	2160.000000	

In [9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1275 entries, 0 to 1274
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Company               1275 non-null   object 
 1   Product               1275 non-null   object 
 2   TypeName              1275 non-null   object 
 3   Inches                1275 non-null   float64
 4   Ram                   1275 non-null   int64  
 5   OS                    1275 non-null   object 
 6   Weight                1275 non-null   float64
 7   Price_euros           1275 non-null   float64
 8   Screen                1275 non-null   object 
 9   ScreenW               1275 non-null   int64  
 10  ScreenH               1275 non-null   int64  
 11  Touchscreen           1275 non-null   object 
 12  IPSpanel              1275 non-null   object 
 13  RetinaDisplay         1275 non-null   object 
 14  CPU_company           1275 non-null   object 
 15  CPU_freq              1275 non-null   float64
 16  CPU_model             1275 non-null   object 
 17  PrimaryStorage        1275 non-null   int64  
 18  SecondaryStorage      1275 non-null   int64  
 19  PrimaryStorageType    1275 non-null   object 
 20  SecondaryStorageType  1275 non-null   object 
 21  GPU_company           1275 non-null   object 
 22  GPU_model             1275 non-null   object 
dtypes: float64(4), int64(5), object(14)
memory usage: 229.2+ KB
```

In [10]:
```
data.isnull().sum()
```

Out[10]:
```
Company               0
Product               0
TypeName              0
Inches                0
Ram                   0
OS                    0
Weight                0
Price_euros           0
Screen                0
ScreenW               0
ScreenH               0
Touchscreen           0
IPSpanel              0
RetinaDisplay         0
CPU_company           0
CPU_freq              0
CPU_model             0
PrimaryStorage        0
SecondaryStorage      0
PrimaryStorageType    0
SecondaryStorageType  0
GPU_company           0
GPU_model             0
dtype: int64
```

In [11]:
```
data.corr(numeric_only = True)
```

Out[11]:

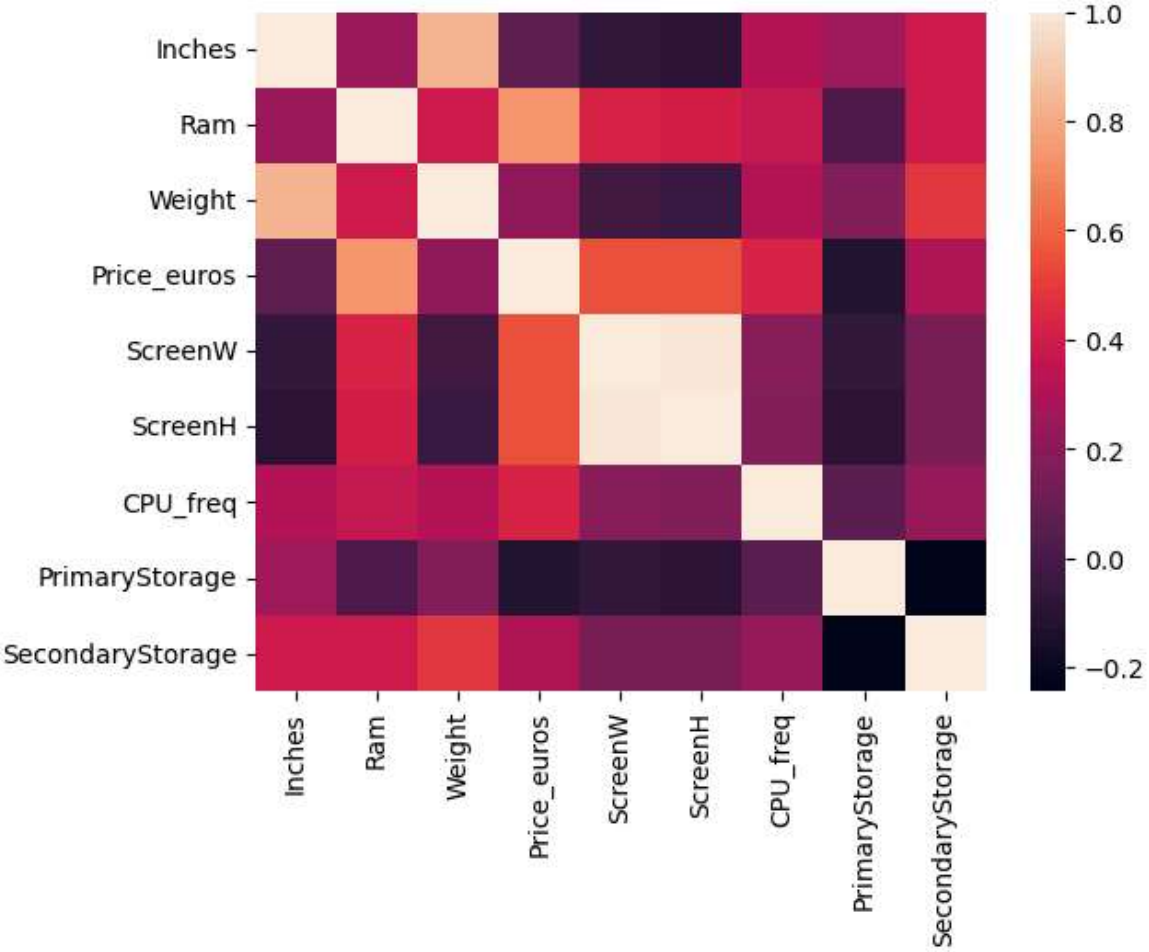
	Inches	Ram	Weight	Price_euros	ScreenW	ScreenH	CPU
Inches	1.000000	0.241078	0.826638	0.066608	-0.068223	-0.093062	0.3
Ram	0.241078	1.000000	0.389370	0.740287	0.424089	0.415241	0.3
Weight	0.826638	0.389370	1.000000	0.211883	-0.028605	-0.050106	0.3
Price_euros	0.066608	0.740287	0.211883	1.000000	0.552491	0.548529	0.4
ScreenW	-0.068223	0.424089	-0.028605	0.552491	1.000000	0.994069	0.1
ScreenH	-0.093062	0.415241	-0.050106	0.548529	0.994069	1.000000	0.1
CPU_freq	0.305037	0.366254	0.318649	0.428847	0.178659	0.164369	1.0
PrimaryStorage	0.264280	0.015365	0.175433	-0.124775	-0.072977	-0.080135	0.0
SecondaryStorage	0.389067	0.390939	0.481495	0.291207	0.146232	0.135293	0.2

In [12]:
```
data.corr(numeric_only = True)['Price_euros']
```

Out[12]:
```
Inches              0.066608
Ram                 0.740287
Weight              0.211883
Price_euros         1.000000
ScreenW             0.552491
ScreenH             0.548529
CPU_freq            0.428847
PrimaryStorage     -0.124775
SecondaryStorage    0.291207
Name: Price_euros, dtype: float64
```

In [13]:
```
sns.heatmap(data.corr(numeric_only = True))
```

Out[13]: <Axes: >

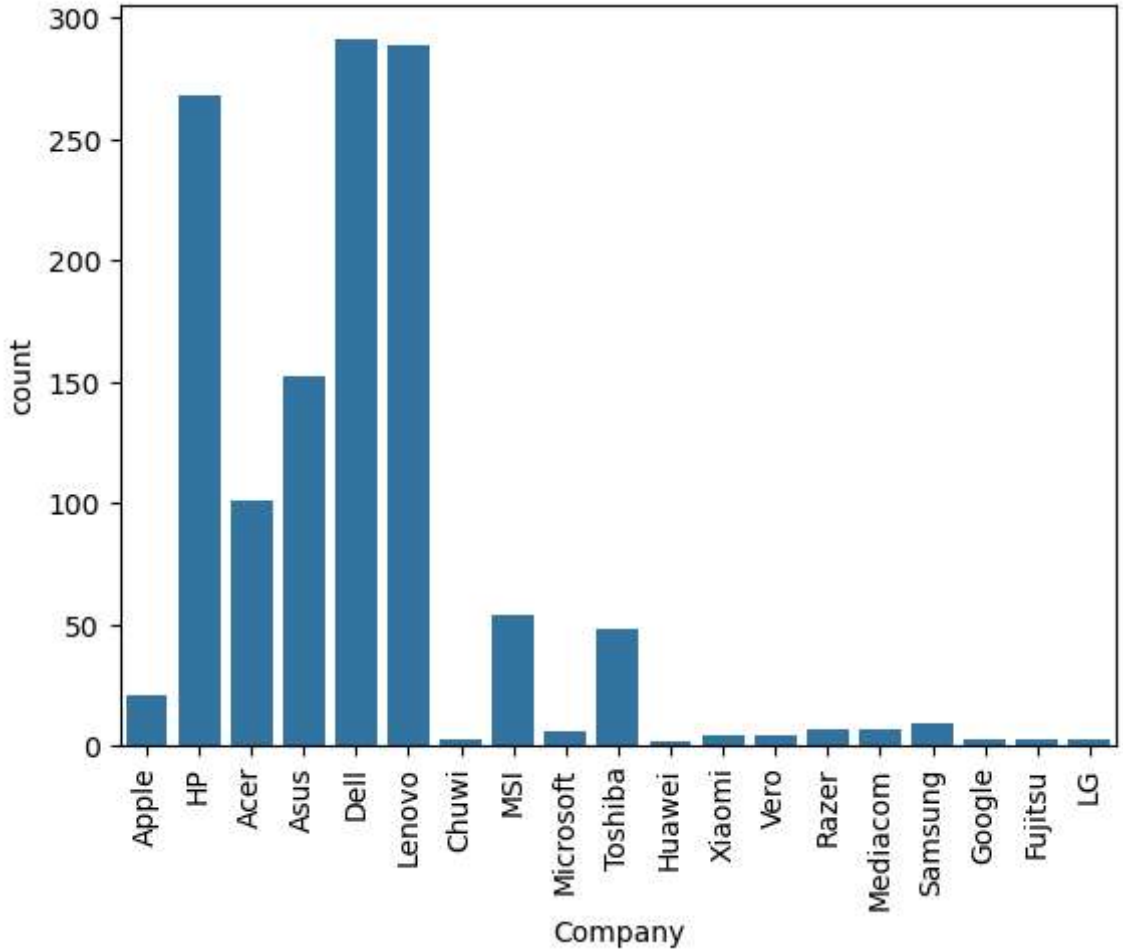


Exploratory Data Analysis

In [15]:
```
data['Company'].value_counts()
```

Out[15]:
```
Company
Dell          291
Lenovo        289
HP            268
Asus          152
Acer          101
MSI            54
Toshiba        48
Apple          21
Samsung         9
Razer           7
Mediacom        7
Microsoft       6
Xiaomi          4
Vero            4
Chuwi           3
Google          3
Fujitsu         3
LG              3
Huawei          2
Name: count, dtype: int64
```

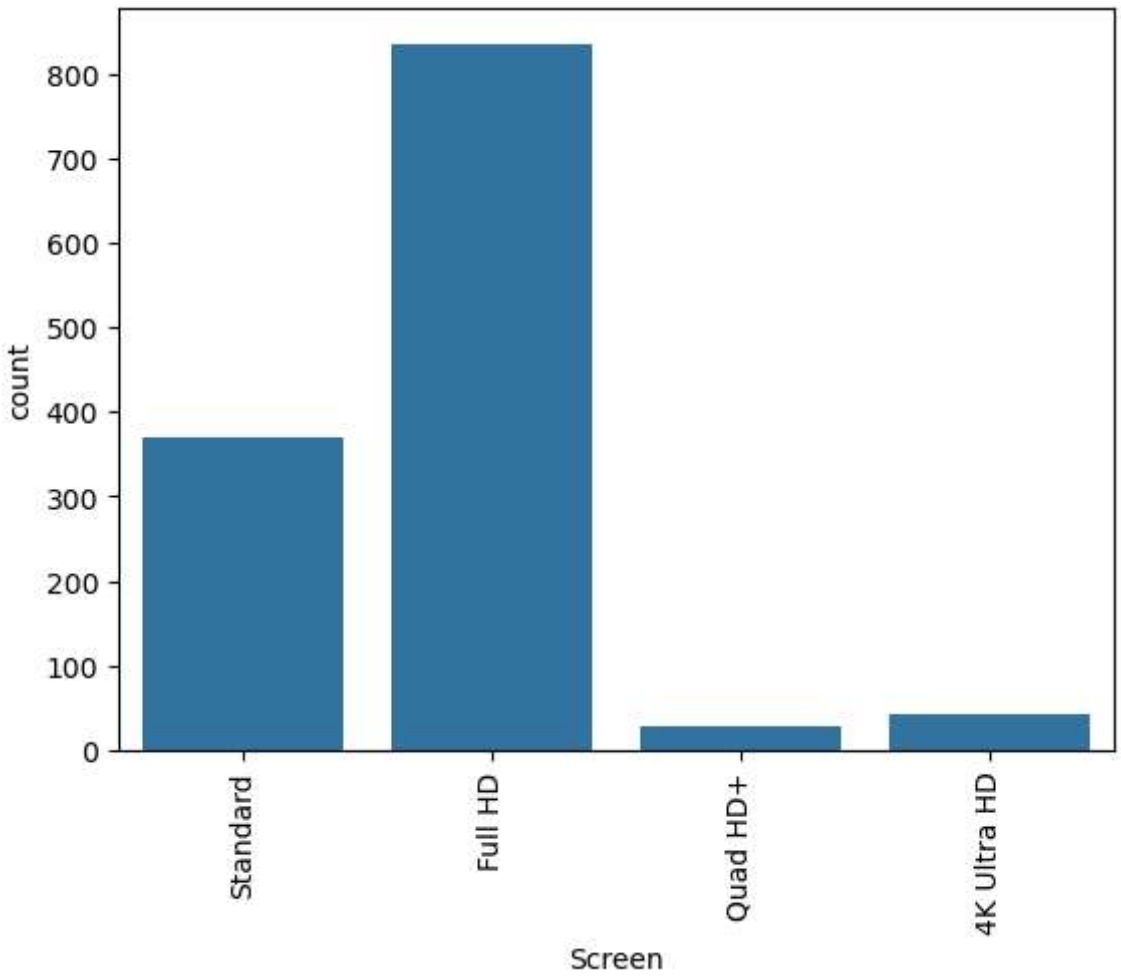
In [16]:
```
# data['Company'].value_counts().plot(kind='bar')

sns.countplot(x='Company', data=data, legend='auto')

# Rotate x-axis labels to be vertical
plt.xticks(rotation=90)

# Show the plot
plt.show()
```



In [17]:
```
data['Screen'].value_counts()
```

Out[17]:
```
Screen
Full HD        835
Standard       369
4K Ultra HD     43
Quad HD+        28
Name: count, dtype: int64
```

In [18]:
```
sns.countplot(x='Screen', data=data, legend='auto')

# Rotate x-axis labels to be vertical
plt.xticks(rotation=90)

# Show the plot
plt.show()
```



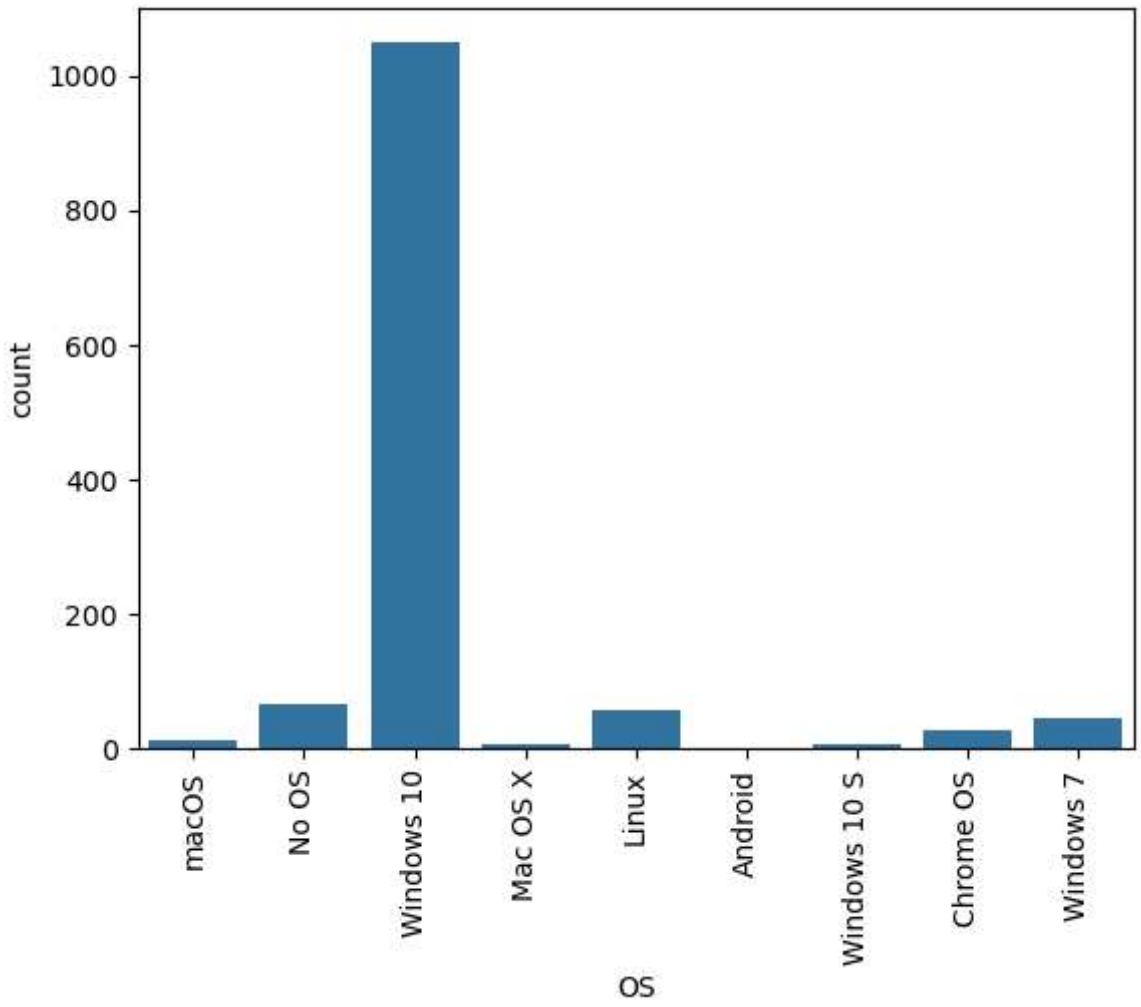
In [19]:
```
data['Product'].value_counts()
```

Out[19]:
```
Product
XPS 13                                  30
Inspiron 3567                           25
250 G6                                  21
Vostro 3568                             19
Legion Y520-15IKBN                      19
                                        ..
VivoBook E201NA                          1
Ideapad 520-15IKBR                       1
Thinkpad X260                            1
Rog G752VL-UH71T                         1
X553SA-XX031T (N3050/4GB/500GB/W10)      1
Name: count, Length: 618, dtype: int64
```

In [20]:
```
data['OS'].value_counts()
```

Out[20]:
```
OS
Windows 10      1048
No OS             66
Linux             58
Windows 7         45
Chrome OS         27
macOS             13
Mac OS X           8
Windows 10 S       8
Android            2
Name: count, dtype: int64
```

In [21]:
```
sns.countplot(x='OS', data=data, legend='auto')

# Rotate x-axis labels to be vertical
plt.xticks(rotation=90)

# Show the plot
plt.show()
```



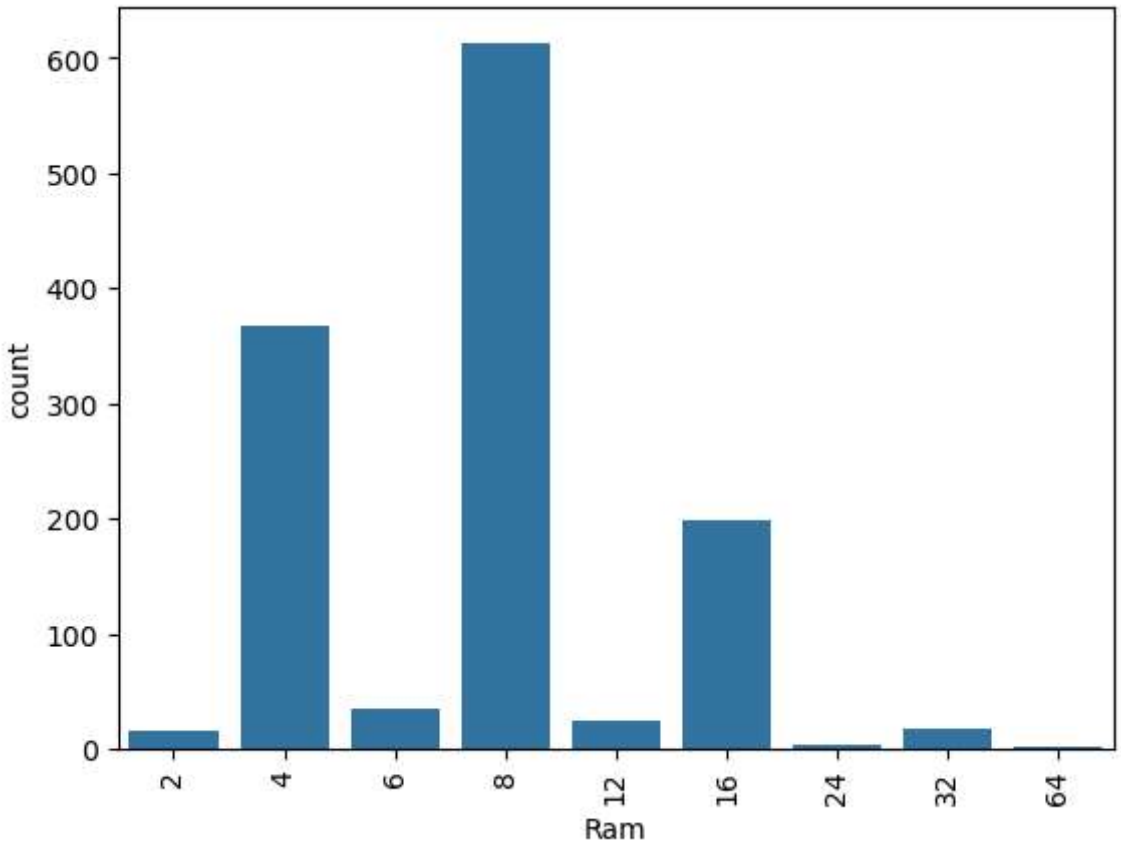
In [22]:
```
data['TypeName'].value_counts()
```

Out[22]:
```
TypeName
Notebook              707
Gaming                205
Ultrabook             194
2 in 1 Convertible    117
Workstation            29
Netbook                23
Name: count, dtype: int64
```

In [23]:
```
data['Ram'].value_counts()
```

Out[23]:
```
Ram
8     613
4     367
16    198
6      35
12     25
32     17
2      16
24      3
64      1
Name: count, dtype: int64
```

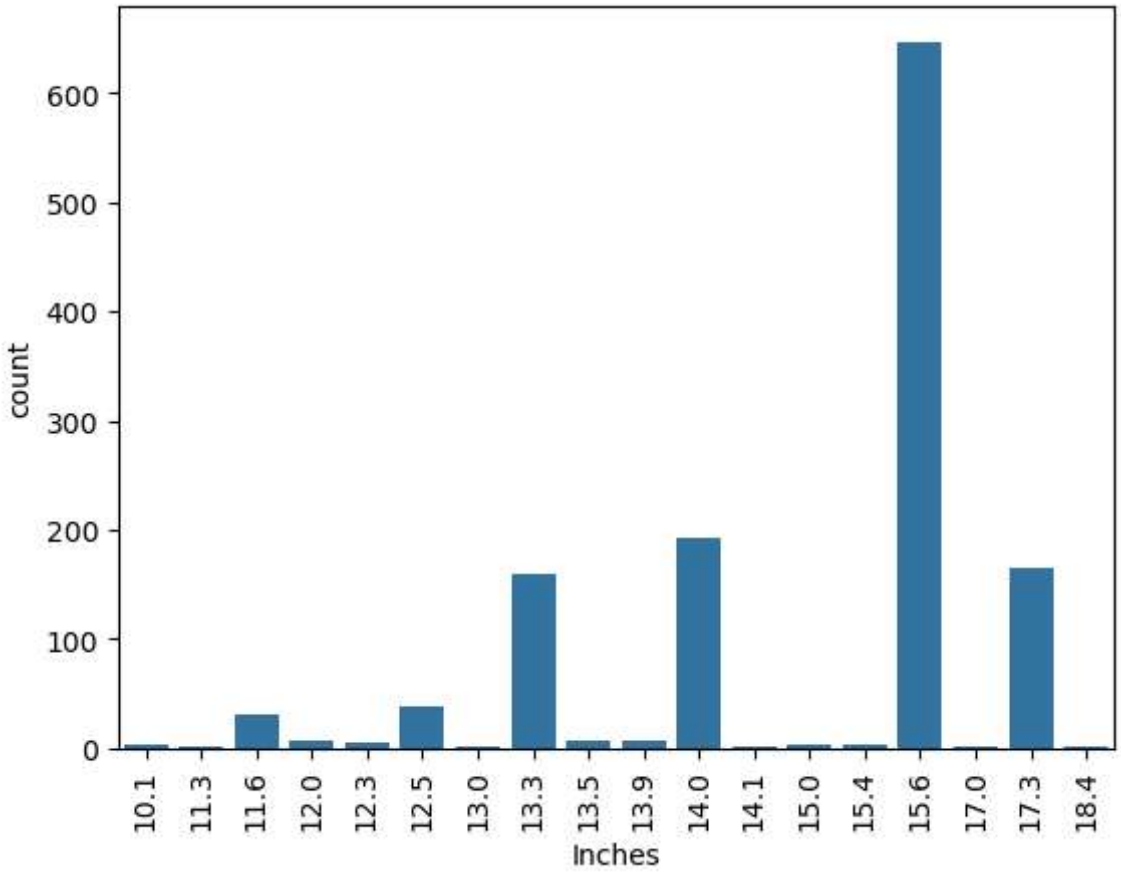
In [24]:
```
sns.countplot(x='Ram', data=data, legend='auto')

# Rotate x-axis labels to be vertical
plt.xticks(rotation=90)

# Show the plot
plt.show()
```



In [25]:
```
data['Inches'].value_counts()
```

Out[25]:
```
Inches
15.6    647
14.0    193
17.3    164
13.3    160
12.5     39
11.6     31
12.0      6
13.5      6
13.9      6
12.3      5
10.1      4
15.4      4
15.0      4
13.0      2
18.4      1
17.0      1
14.1      1
11.3      1
Name: count, dtype: int64
```

In [26]:
```
sns.countplot(x='Inches', data=data, legend='auto')

# Rotate x-axis labels to be vertical
plt.xticks(rotation=90)

# Show the plot
plt.show()
```



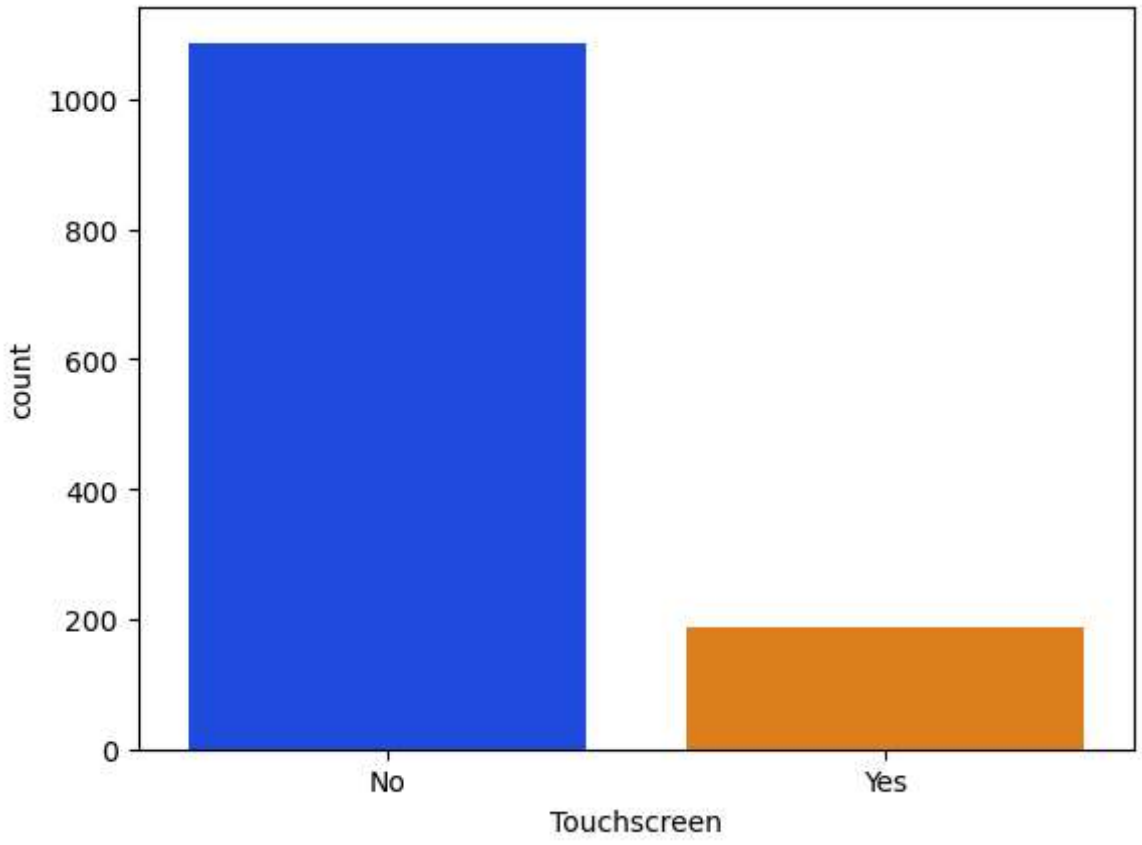
In [27]:
```
data['Touchscreen'].value_counts()
```

Out[27]:
```
Touchscreen
No      1087
Yes      188
Name: count, dtype: int64
```

In [28]:
```
sns.countplot(x='Touchscreen', data=data, legend='auto', palette='bright' )
```

```
C:\Users\91700\AppData\Local\Temp\ipykernel_15640\3302626064.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.countplot(x='Touchscreen', data=data, legend='auto', palette='bright' )
```

Out[28]: <Axes: xlabel='Touchscreen', ylabel='count'>



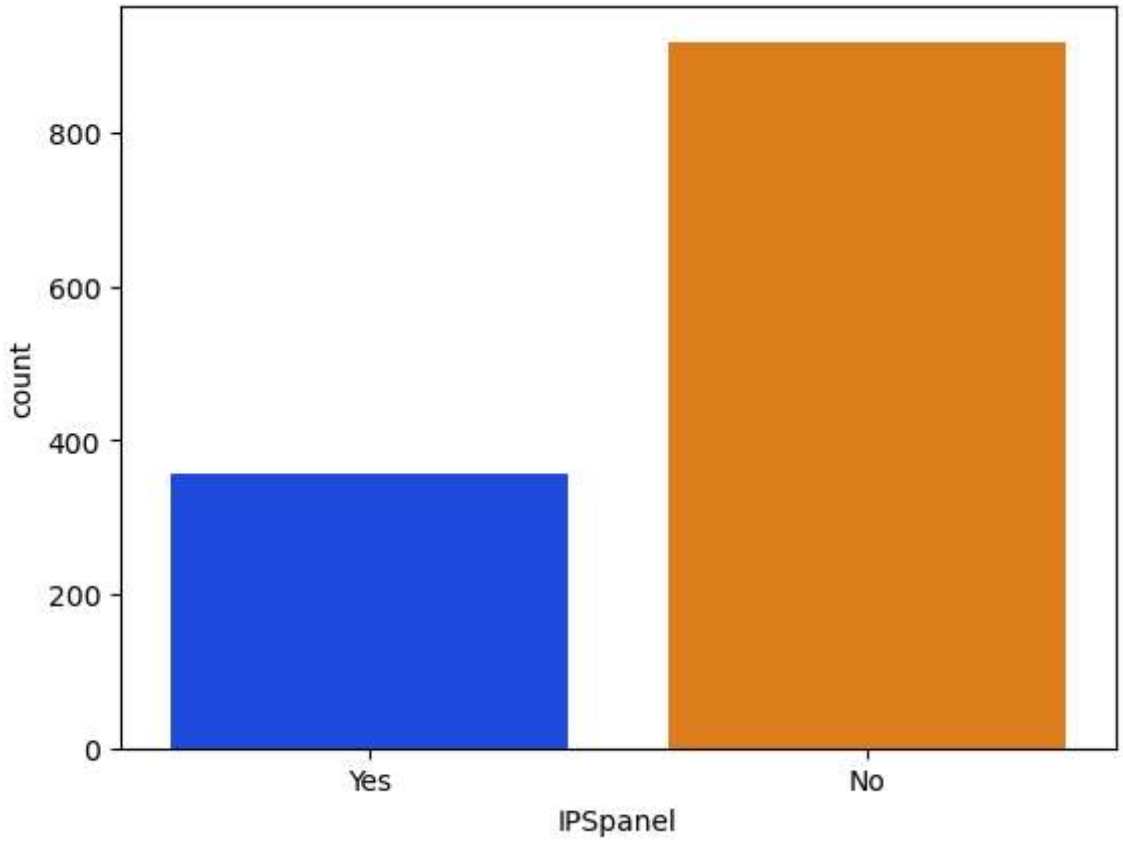
In [29]:
```
data['IPSpanel'].value_counts()
```

Out[29]:
```
IPSpanel
No     918
Yes    357
Name: count, dtype: int64
```

In [30]:
```
sns.countplot(x='IPSpanel', data=data, legend='auto', palette='bright')
```

```
C:\Users\91700\AppData\Local\Temp\ipykernel_15640\2524309836.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.countplot(x='IPSpanel', data=data, legend='auto', palette='bright')
```

Out[30]: <Axes: xlabel='IPSpanel', ylabel='count'>



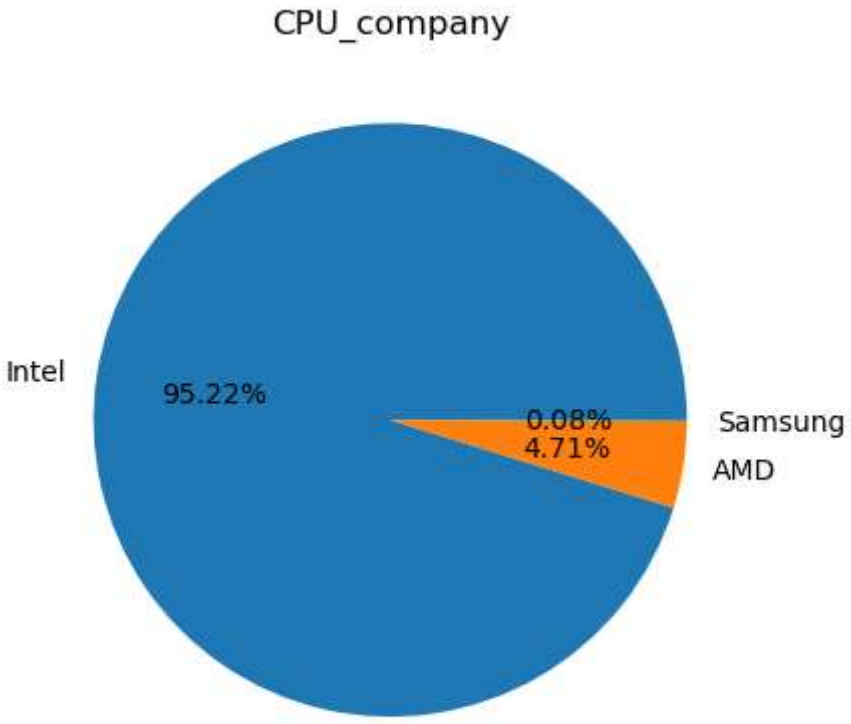
In [31]:
```
data['RetinaDisplay'].value_counts()
```

Out[31]:
```
RetinaDisplay
No     1258
Yes      17
Name: count, dtype: int64
```

In [32]:
```
data['CPU_company'].value_counts()
```

Out[32]:
```
CPU_company
Intel      1214
AMD          60
Samsung       1
Name: count, dtype: int64
```

In [33]:
```
# Count the occurrences of each CPU_company
CPU_company = data['CPU_company'].value_counts()

# Plot pie chart
plt.pie(CPU_company, labels=CPU_company.index, autopct='%.2f%%')
plt.title("CPU_company")
plt.show()
```



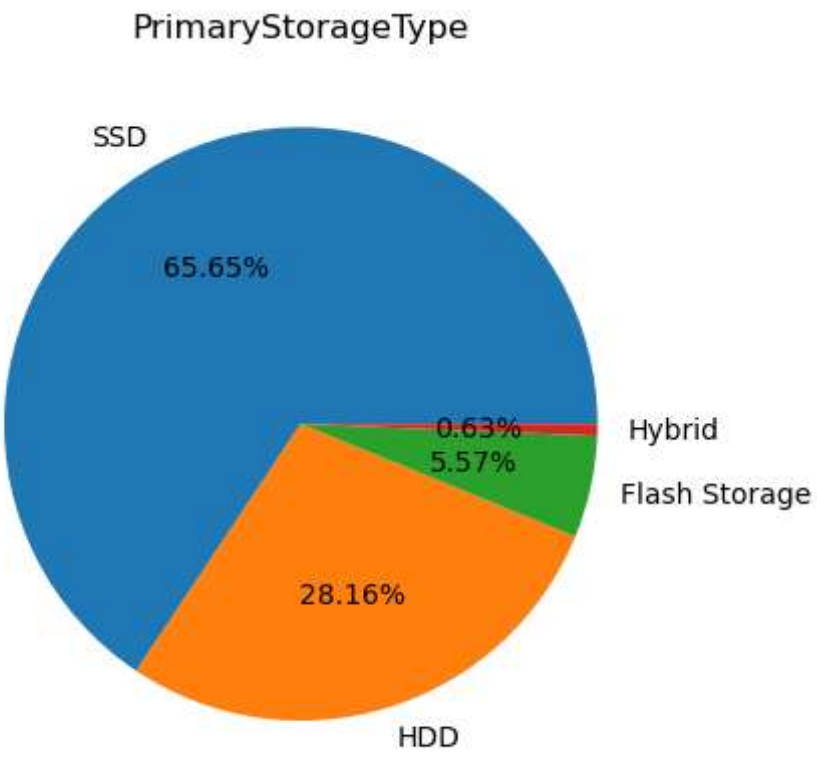
In [34]:
```
data['CPU_model'].value_counts()
```

Out[34]:
```
CPU_model
Core i5 7200U       193
Core i7 7700HQ      147
Core i7 7500U       133
Core i3 6006U        81
Core i7 8550U        73
                   ...
Core M m3             1
E-Series E2-9000      1
Core M M3-6Y30        1
A6-Series 7310        1
A9-Series 9410        1
Name: count, Length: 93, dtype: int64
```

In [35]:
```
data['PrimaryStorageType'].value_counts()
```

Out[35]:
```
PrimaryStorageType
SSD              837
HDD              359
Flash Storage     71
Hybrid             8
Name: count, dtype: int64
```

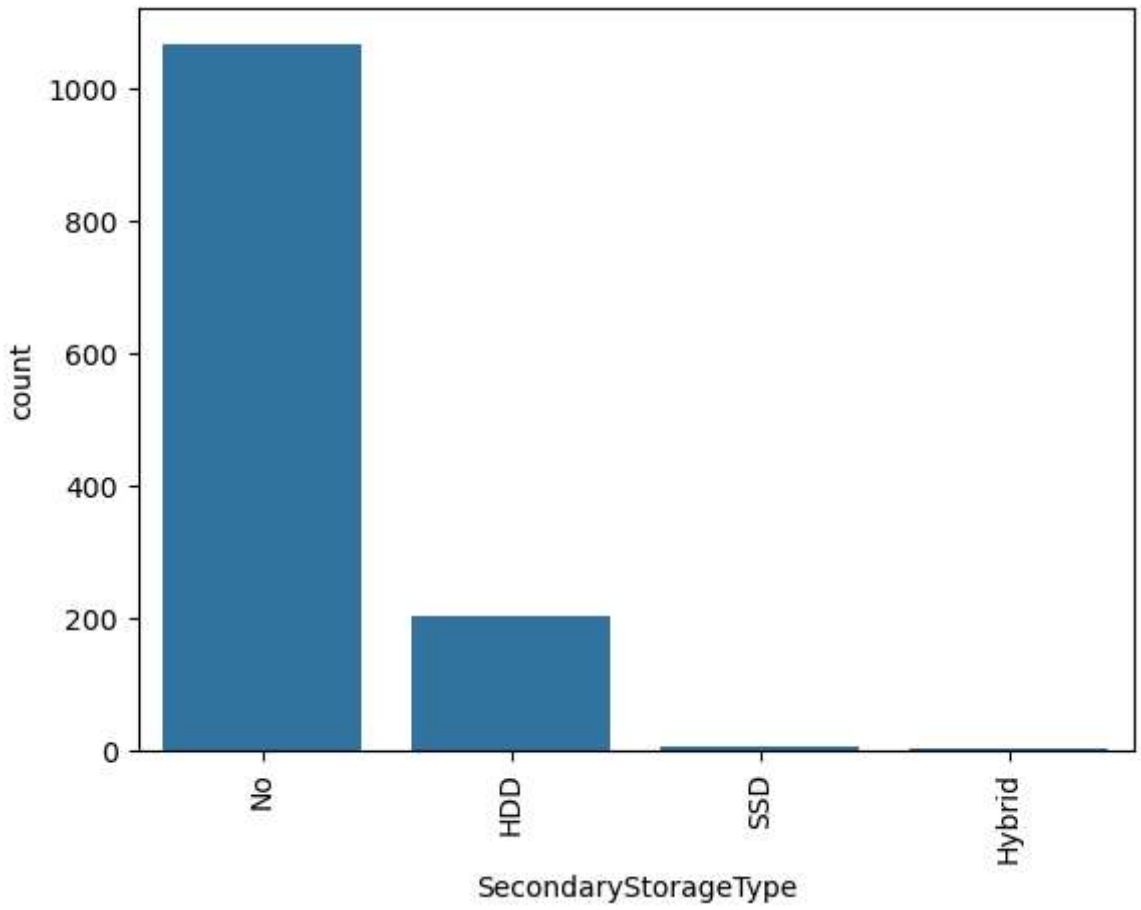
In [36]:
```
# Count the occurrences of each CPU_company
PrimaryStorageType = data['PrimaryStorageType'].value_counts()

# Plot pie chart
plt.pie(PrimaryStorageType, labels=PrimaryStorageType.index, autopct='%.2f%%')
plt.title("PrimaryStorageType")
plt.show()
```



In [37]:
```
data['SecondaryStorageType'].value_counts()
```

Out[37]:
```
SecondaryStorageType
No         1067
HDD         202
SSD           4
Hybrid        2
Name: count, dtype: int64
```

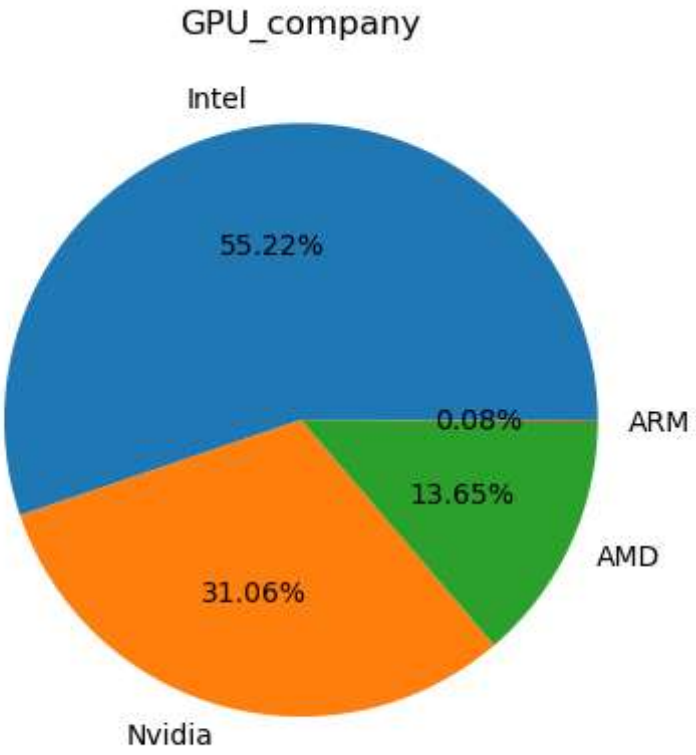
In [38]:
```
sns.countplot(x='SecondaryStorageType', data=data, legend='auto')

# Rotate x-axis labels to be vertical
plt.xticks(rotation=90)

# Show the plot
plt.show()
```



In [39]:
```
data['GPU_company'].value_counts()
```

Out[39]:
```
GPU_company
Intel     704
Nvidia    396
AMD       174
ARM         1
Name: count, dtype: int64
```

In [40]:
```
# Count the occurrences of each CPU_company
GPU_company = data['GPU_company'].value_counts()

# Plot pie chart
plt.pie(GPU_company, labels=GPU_company.index, autopct='%.2f%%')
plt.title("GPU_company")
plt.show()
```



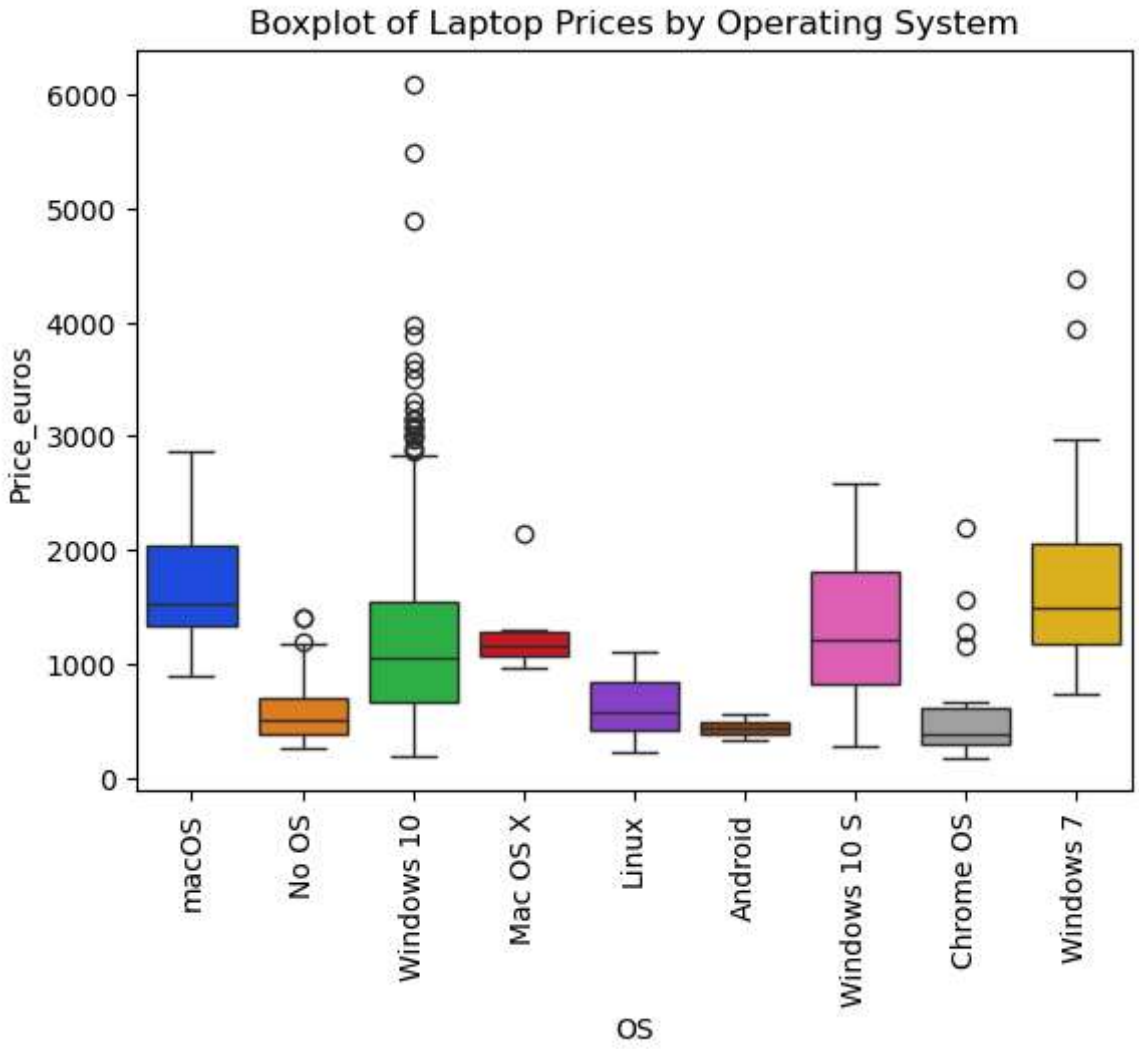
In [41]:
```
data['GPU_model'].value_counts()
```

Out[41]:
```
GPU_model
HD Graphics 620     279
HD Graphics 520     181
UHD Graphics 620     68
GeForce GTX 1050     66
GeForce GTX 1060     48
                   ...
Radeon R5 520          1
Radeon R7              1
HD Graphics 540        1
Radeon 540             1
Mali T860 MP4          1
Name: count, Length: 110, dtype: int64
```

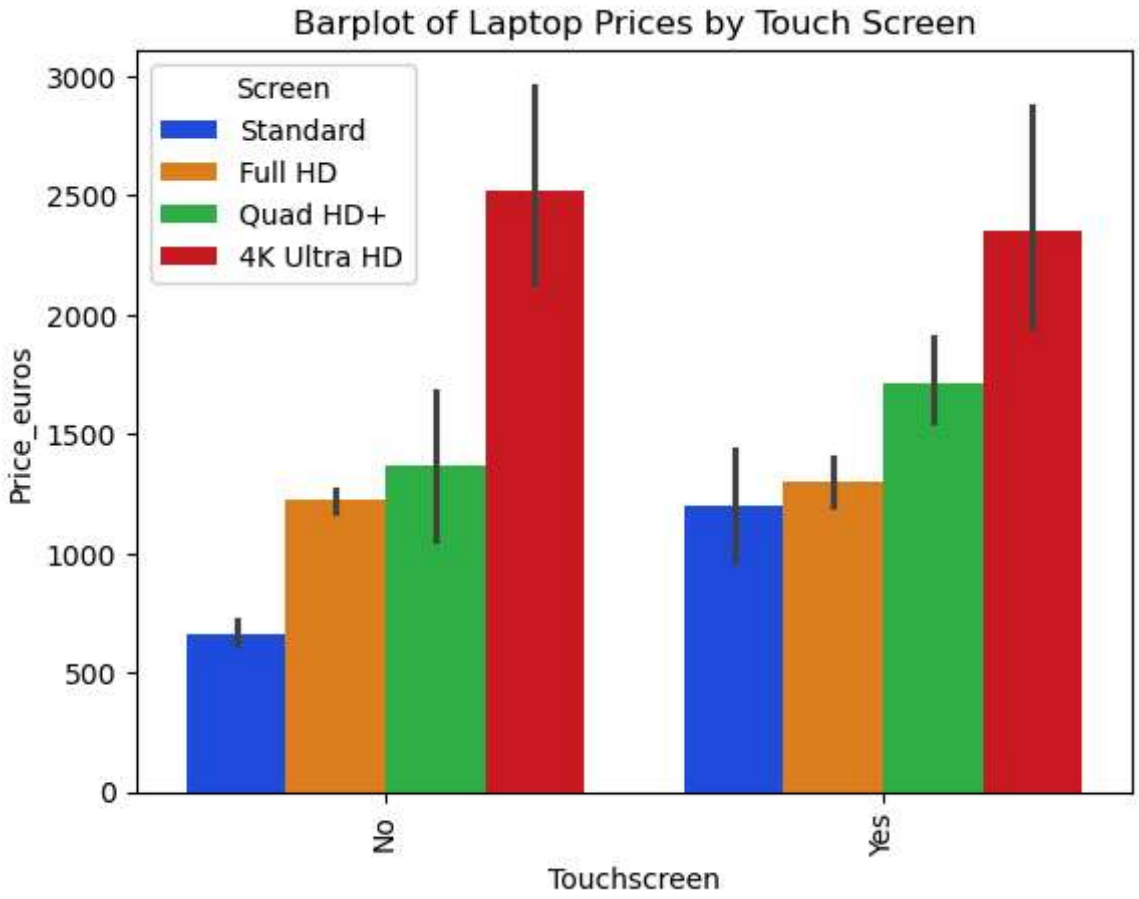
In [ ]:
```
```

In [43]:
```
# Plot boxplot
sns.boxplot(x="OS", y="Price_euros", data=data, palette="bright")
plt.title("Boxplot of Laptop Prices by Operating System")

# Rotate x-axis labels to be vertical
plt.xticks(rotation=90)

# Show the plot
plt.show()
```

```
C:\Users\91700\AppData\Local\Temp\ipykernel_15640\1734334298.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.boxplot(x="OS", y="Price_euros", data=data, palette="bright")
```

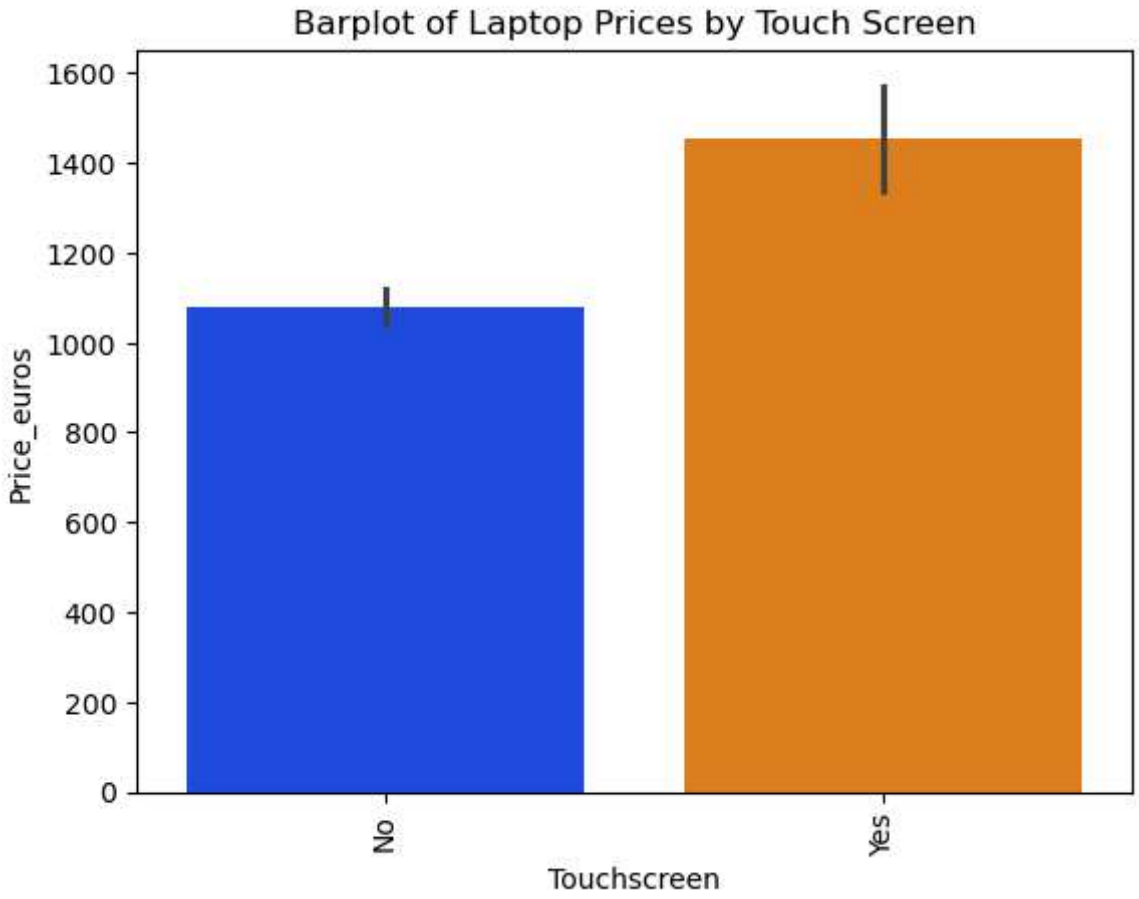


In [44]:
```
# Plot barplot
sns.barplot(x = 'Touchscreen', y = 'Price_euros', hue ='Screen', palette="bright
plt.title("Barplot of Laptop Prices by Touch Screen")

# Rotate x-axis labels to be vertical
plt.xticks(rotation=90)

# Show the plot
plt.show()
```

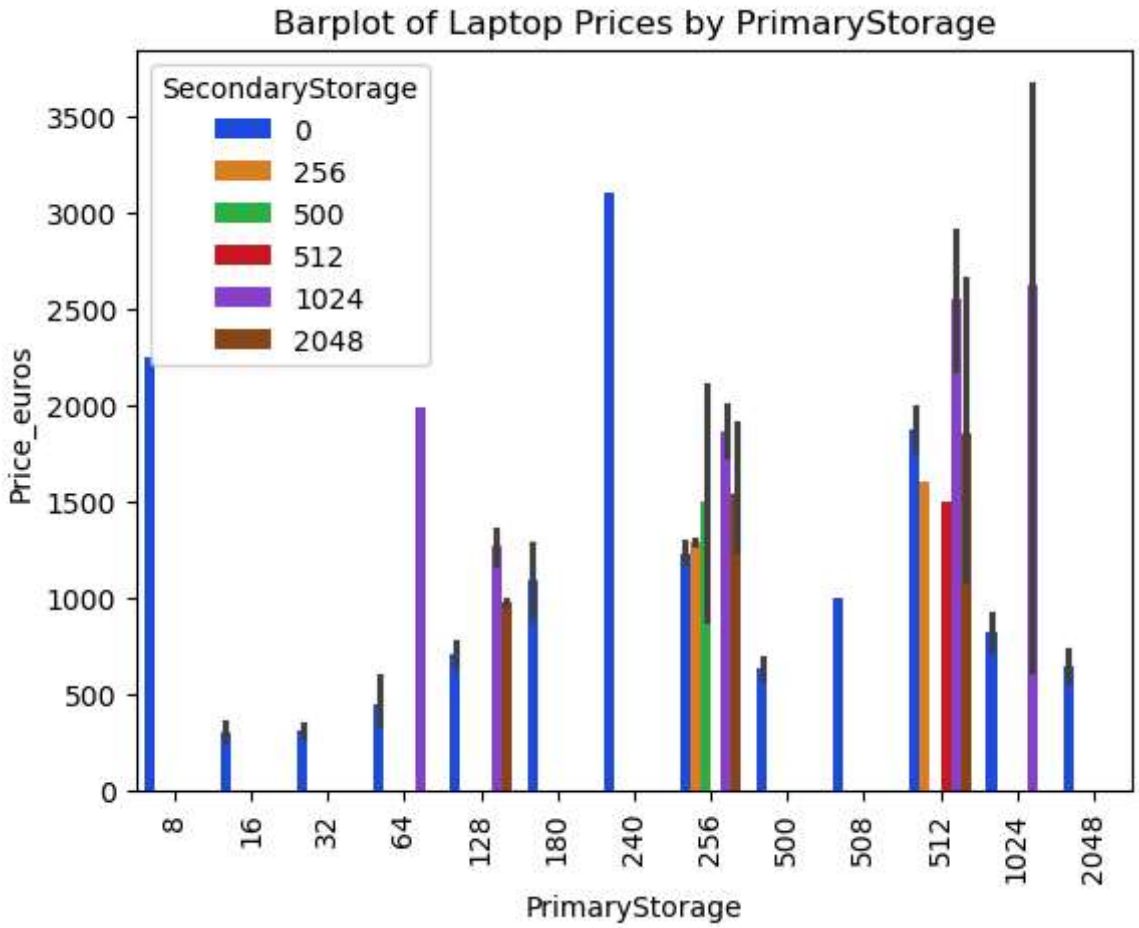


In [45]:
```
# Plot boxplot
sns.barplot(x = 'Touchscreen', y = 'Price_euros', palette="bright", data = data)
plt.title("Barplot of Laptop Prices by Touch Screen")

# Rotate x-axis labels to be vertical
plt.xticks(rotation=90)

# Show the plot
plt.show()
```

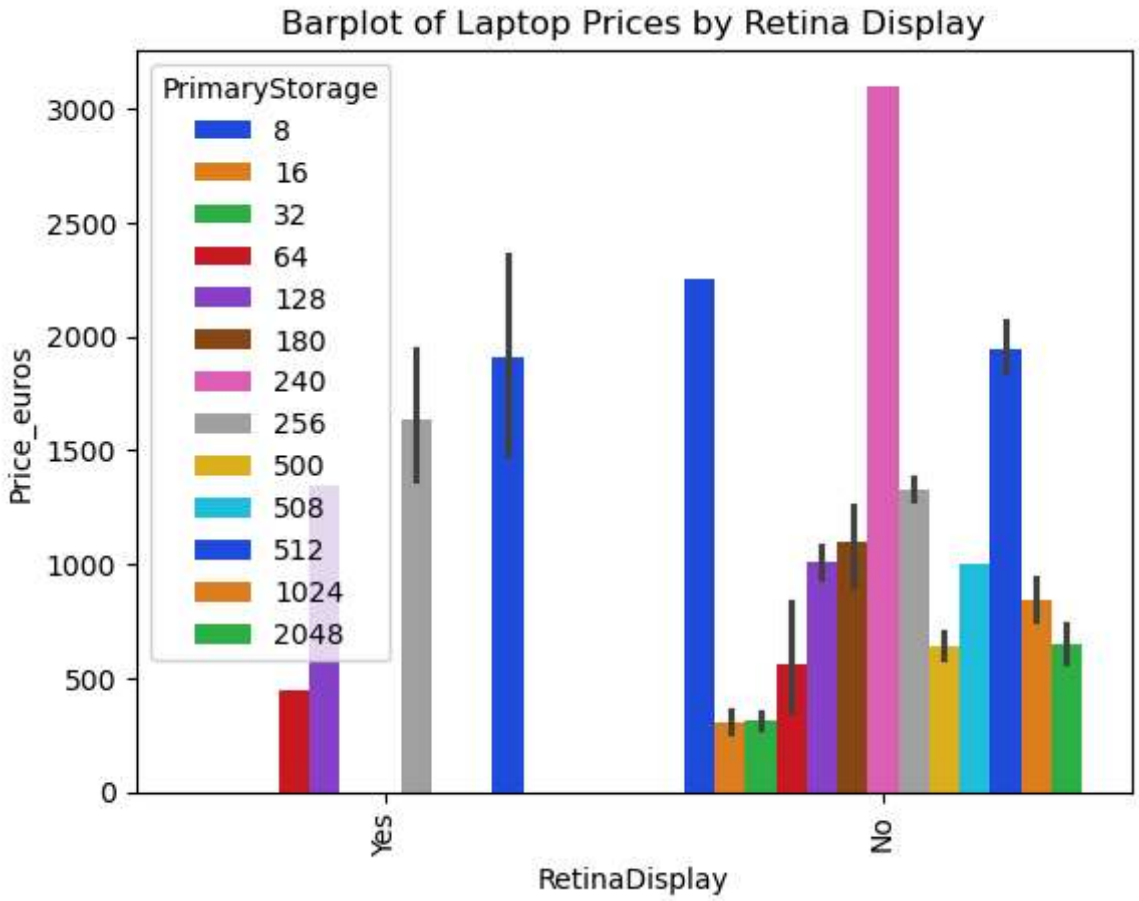
```
C:\Users\91700\AppData\Local\Temp\ipykernel_15640\3365394172.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x = 'Touchscreen', y = 'Price_euros', palette="bright", data = dat
a)
```



In [86]:
```
# Plot boxplot
sns.barplot(x="PrimaryStorage", y="Price_euros", data=data, palette="bright", hu
plt.title("Barplot of Laptop Prices by PrimaryStorage")

# Rotate x-axis labels to be vertical
plt.xticks(rotation=90)

# Show the plot
plt.show()
```

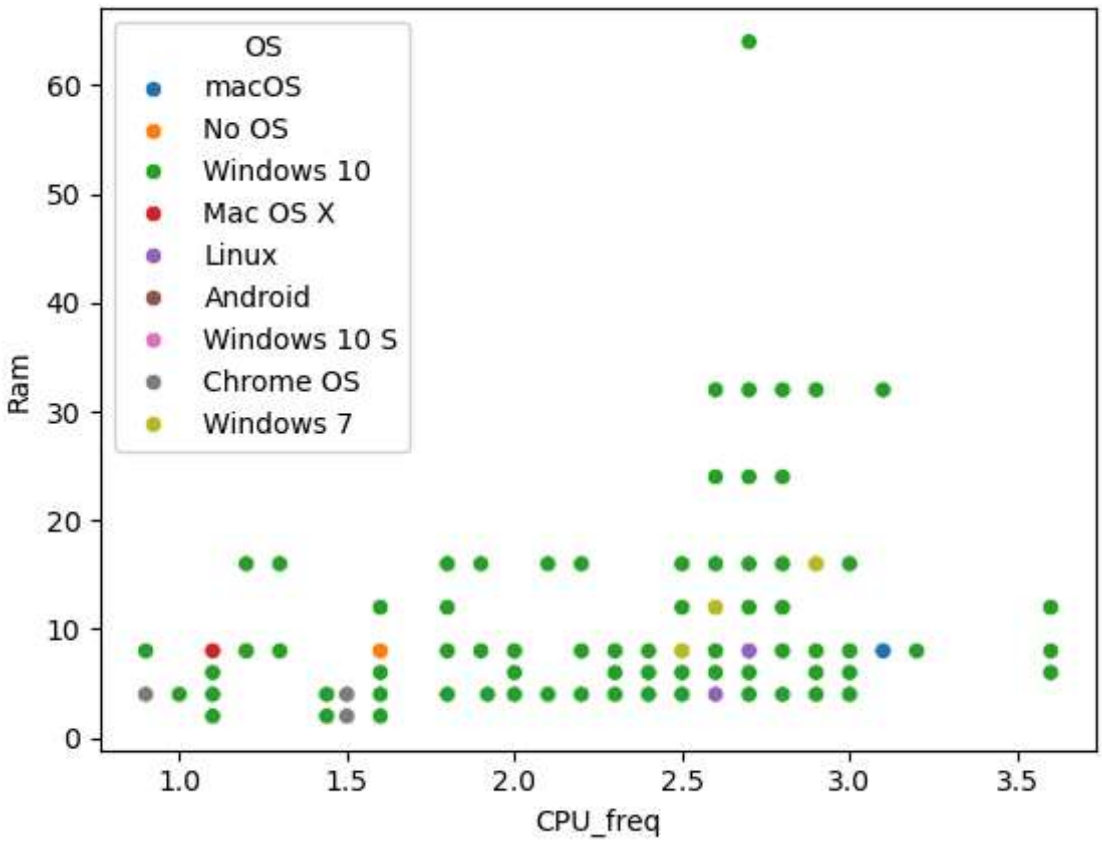


In [88]:
```
# Plot boxplot
sns.barplot(x="RetinaDisplay", y="Price_euros", data=data, palette="bright", hue
plt.title("Barplot of Laptop Prices by Retina Display")

# Rotate x-axis labels to be vertical
plt.xticks(rotation=90)

# Show the plot
plt.show()
```



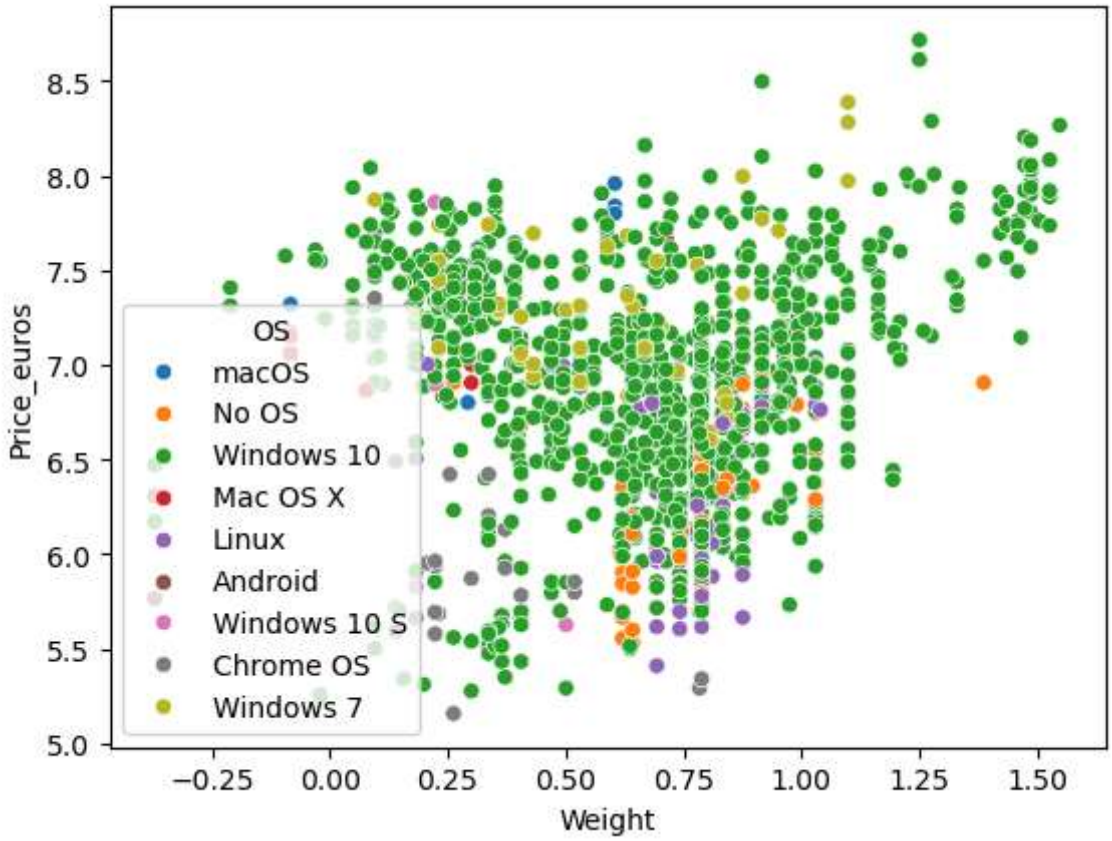
In [90]:
```
sns.scatterplot(data = data , x= 'CPU_freq', y = 'Ram', hue = 'OS')
```

Out[90]:
```
<Axes: xlabel='CPU_freq', ylabel='Ram'>
```



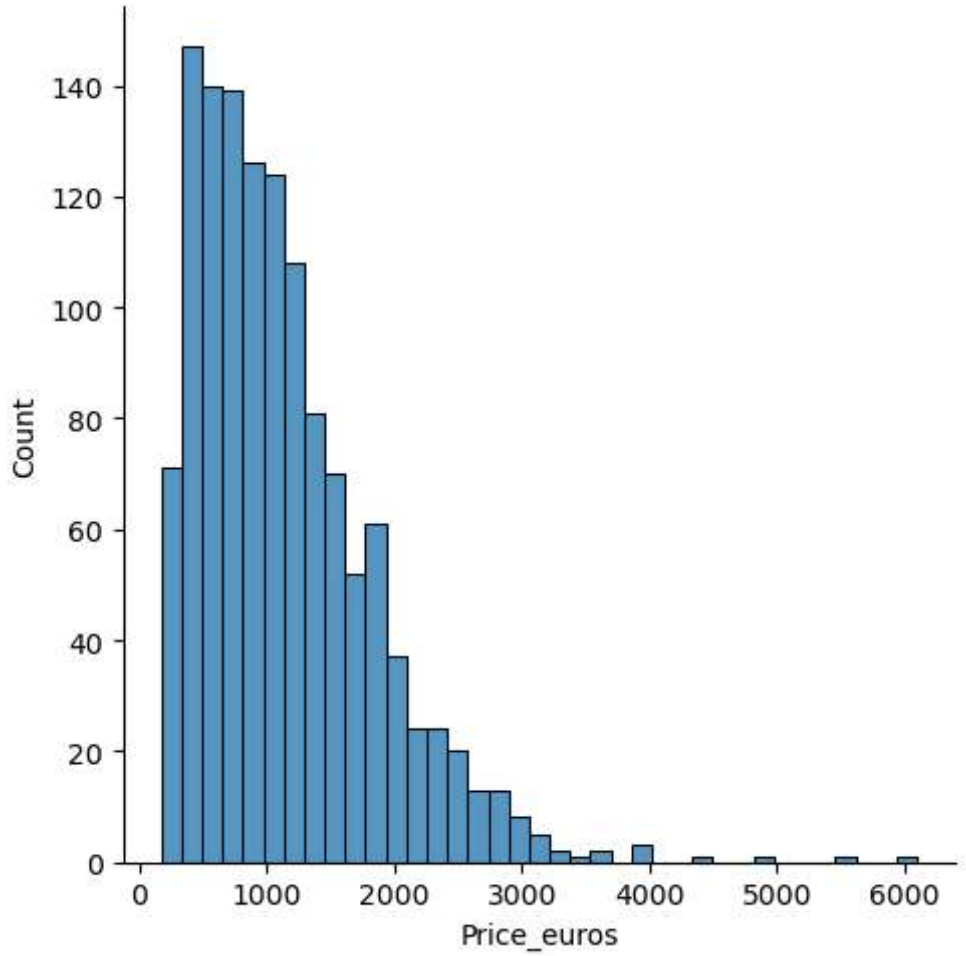
In [92]:
```
sns.scatterplot(data = data , x= np.log(data['Weight']), y = np.log(data['Price
```

Out[92]: <Axes: xlabel='Weight', ylabel='Price_euros'>



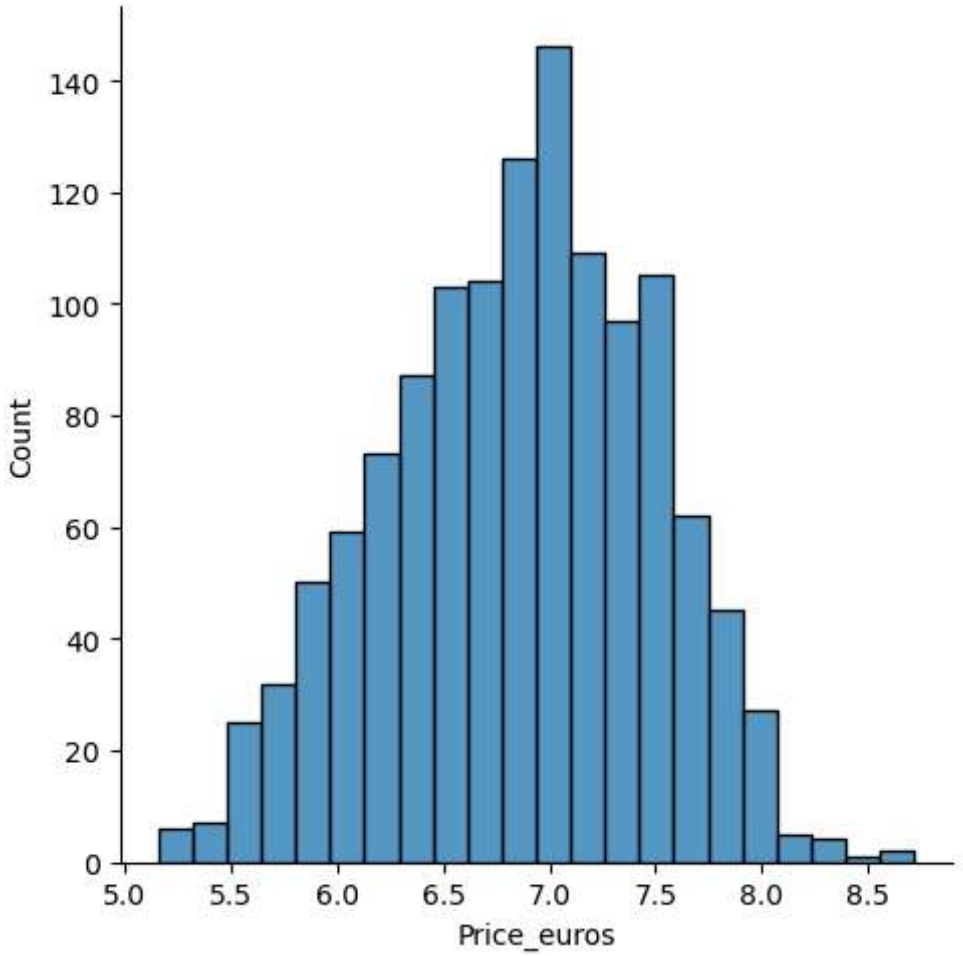
In [94]:
```
sns.displot(data['Price_euros'])
```

Out[94]: <seaborn.axisgrid.FacetGrid at 0x28a02a40950>



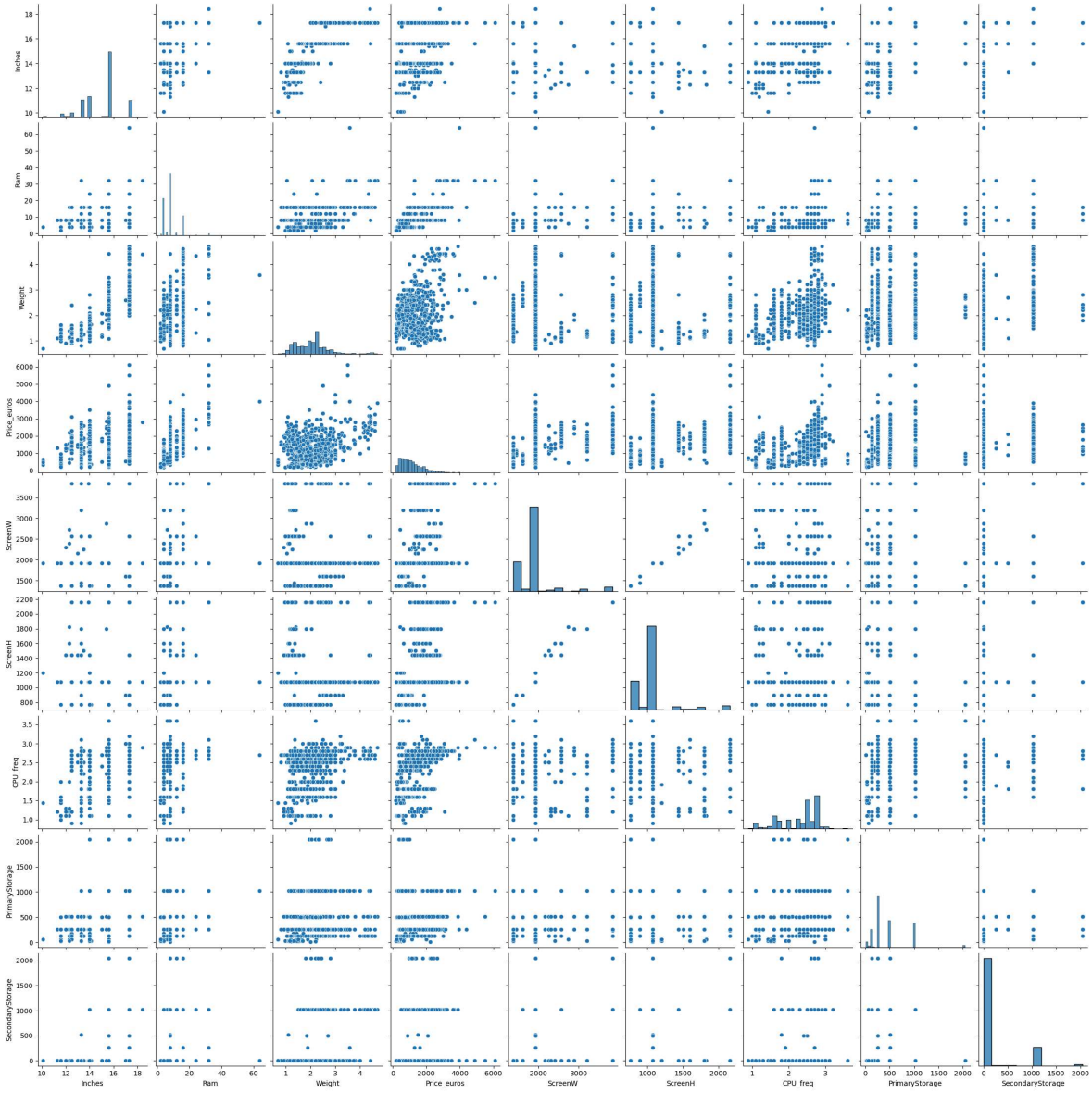
In [96]:
```
sns.displot(x= np.log(data['Price_euros']))
```

Out[96]:
```
<seaborn.axisgrid.FacetGrid at 0x28a094f6390>
```



In [100…
```
sns.pairplot(data)
```

Out[100…
```
<seaborn.axisgrid.PairGrid at 0x28a03838da0>
```



Define features and target

In [102…
```
X = data.drop('Price_euros', axis=1)
y = np.log(data['Price_euros'])
```

Identify categorical and numerical columns

In [104…
```
categorical_cols = X.select_dtypes(include=['object']).columns
categorical_cols
```

Out[104…
```
Index(['Company', 'Product', 'TypeName', 'OS', 'Screen', 'Touchscreen',
       'IPSpanel', 'RetinaDisplay', 'CPU_company', 'CPU_model',
       'PrimaryStorageType', 'SecondaryStorageType', 'GPU_company',
       'GPU_model'],
      dtype='object')
```

In [105…
```
numerical_cols = X.select_dtypes(include=['float64', 'int64', 'int32']).columns
numerical_cols
```

Out[105…
```
Index(['Inches', 'Ram', 'Weight', 'ScreenW', 'ScreenH', 'CPU_freq',
       'PrimaryStorage', 'SecondaryStorage'],
      dtype='object')
```

Split the data

In [107…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

Encoding for categorical features and Preprocessing pipelines

In [109…
```
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(drop = 'first', handle_unknown= 'ignore'), ['Compa
                                                                          'Touch
                                                                          'CPU_c
                                                                          'Secon
        # ('num', StandardScaler(), ['Inches', 'Ram', 'Weight', 'ScreenW', 'Scre
        #                            'SecondaryStorage'])
    ], remainder = 'passthrough')

regressor = LinearRegression()

pipe_reg = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', regressor)
])

# Train the Model
pipe_reg.fit(X_train, y_train)

y_pred_reg = pipe_reg.predict(X_test)

# Evaluate the test model
mse_test_reg = mean_squared_error(y_test, y_pred_reg)
r2_test_reg = r2_score(y_test, y_pred_reg)

print(f"Mean Squared Error Test data: {mse_test_reg}")
print(f"R-squared Test data: {r2_test_reg}")
```

```
Mean Squared Error Test data: 0.04296617417476588
R-squared Test data: 0.8749081962913069
```

```
D:\Anaconda\Lib\site-packages\sklearn\preprocessing\_encoders.py:242: UserWarnin
g: Found unknown categories in columns [1, 9, 13] during transform. These unknown
categories will be encoded as all zeros
  warnings.warn(
```

Ridge Regression

In [112…
```
rid_prepro = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(drop = 'first', handle_unknown= 'ignore'), ['Compa
                                                                          'Touch
                                                                          'CPU_c
                                                                          'Secon
```

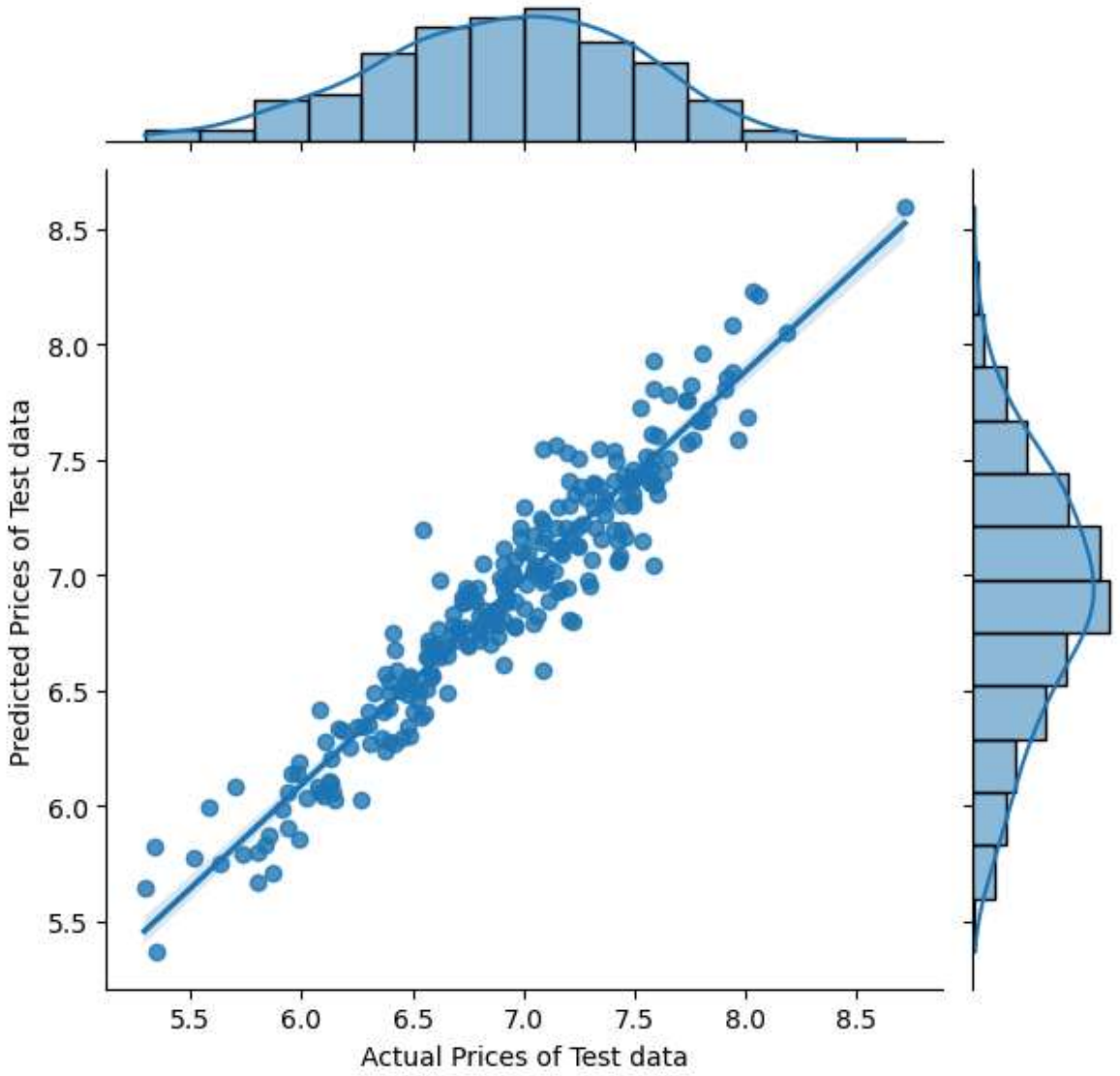
```
        ('num', StandardScaler(), ['Inches', 'Ram', 'Weight', 'ScreenW', 'Screen
                                   'SecondaryStorage'])
    ], remainder = 'passthrough')

rid_reg = Ridge(alpha=0.9)

pipe_rid = Pipeline([
    ('rid_prepro', rid_prepro),
    ('rid_reg', rid_reg)
])

# Train the Model
pipe_rid.fit(X_train, y_train)

y_pred_rid = pipe_rid.predict(X_test)

# Evaluate the test model
mse_test_rid = mean_squared_error(y_test, y_pred_rid)
r2_test_rid = r2_score(y_test, y_pred_rid)

print(f"Mean Squared Error Test data: {mse_test_rid}")
print(f"R-squared Test data: {r2_test_rid}")
```

```
Mean Squared Error Test data: 0.030172629390433903
R-squared Test data: 0.9121553476525265
```

```
D:\Anaconda\Lib\site-packages\sklearn\preprocessing\_encoders.py:242: UserWarnin
g: Found unknown categories in columns [1, 9, 13] during transform. These unknown
categories will be encoded as all zeros
  warnings.warn(
```

In [165…

```
# Visualize test results
sns.jointplot(x = y_test, y= y_pred_rid, kind="reg")
plt.xlabel("Actual Prices of Test data")
plt.ylabel("Predicted Prices of Test data")
plt.show()
```



Lasso Regression

In [115…

```
las_prepro = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), ['Inches', 'Ram', 'Weight', 'ScreenW', 'Screen
                                   'SecondaryStorage']),
        ('cat', OneHotEncoder(drop = 'first', handle_unknown= 'ignore'), ['Compa
                                                                          'Touch
                                                                          'CPU_c
                                                                          'Secon

    ], remainder = 'passthrough')

las_reg = Lasso(alpha=0.0001)

pipe_las = Pipeline([
    ('las_prepro', las_prepro),
    ('las_reg', las_reg)
])

# Train the Model
pipe_las.fit(X_train, y_train)

y_pred_las = pipe_las.predict(X_test)
```

```
# Evaluate the test model
mse_test_las = mean_squared_error(y_test, y_pred_las)
r2_test_las = r2_score(y_test, y_pred_las)

print(f"Mean Squared Error Test data: {mse_test_las}")
print(f"R-squared Test data: {r2_test_las}")
```

```
Mean Squared Error Test data: 0.0312409212885543
R-squared Test data: 0.9090451205264226
```

```
D:\Anaconda\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:658: Co
nvergenceWarning: Objective did not converge. You might want to increase the numb
er of iterations. Duality gap: 2.0202187520186685, tolerance: 0.0400768992666236
4
  model = cd_fast.sparse_enet_coordinate_descent(
D:\Anaconda\Lib\site-packages\sklearn\preprocessing\_encoders.py:242: UserWarnin
g: Found unknown categories in columns [1, 9, 13] during transform. These unknown
categories will be encoded as all zeros
  warnings.warn(
```

KNeighbors Regressor

In [118…

```
knn_prepro = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), ['Inches', 'Ram', 'Weight', 'ScreenW', 'Screen
                                   'SecondaryStorage']),
        ('cat', OneHotEncoder(drop = 'first', handle_unknown= 'ignore'), ['Compa
                                                                          'Touch
                                                                          'CPU_c
                                                                          'Secon

    ], remainder = 'passthrough')

knn = KNeighborsRegressor(n_neighbors=5)

pipe_knn = Pipeline([
    ('knn_prepro', knn_prepro),
    ('knn', knn)
])

# Train the Model
pipe_knn.fit(X_train, y_train)

y_pred_knn = pipe_knn.predict(X_test)

# Evaluate the test model
mse_test_knn = mean_squared_error(y_test, y_pred_knn)
r2_test_knn = r2_score(y_test, y_pred_knn)

print(f"Mean Squared Error Test data: {mse_test_knn}")
print(f"R-squared Test data: {r2_test_knn}")
```

```
Mean Squared Error Test data: 0.05207344800605449
R-squared Test data: 0.84839326140809
```

```
D:\Anaconda\Lib\site-packages\sklearn\preprocessing\_encoders.py:242: UserWarnin
g: Found unknown categories in columns [1, 9, 13] during transform. These unknown
categories will be encoded as all zeros
  warnings.warn(
```

Decision Tree Regressor

In [121…
```
dtr_prepro = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), ['Inches', 'Ram', 'Weight', 'ScreenW', 'Screen
                                   'SecondaryStorage']),
        ('cat', OneHotEncoder(drop = 'first', handle_unknown= 'ignore'), ['Compa
                                                                          'Touch
                                                                          'CPU_c
                                                                          'Secon
    ], remainder = 'passthrough')

dtr = DecisionTreeRegressor(max_depth=12, min_samples_split=12, max_features=0.8

pipe_dtr = Pipeline([
    ('dtr_prepro', dtr_prepro),
    ('dtr', dtr)
])

# Train the Model
pipe_dtr.fit(X_train, y_train)

y_pred_dtr = pipe_dtr.predict(X_test)

# Evaluate the test model
mse_test_dtr = mean_squared_error(y_test, y_pred_dtr)
r2_test_dtr = r2_score(y_test, y_pred_dtr)

print(f"Mean Squared Error Test data: {mse_test_dtr}")
print(f"R-squared Test data: {r2_test_dtr}")
```

```
Mean Squared Error Test data: 0.06291982229794918
R-squared Test data: 0.8168151060351233
```

```
D:\Anaconda\Lib\site-packages\sklearn\preprocessing\_encoders.py:242: UserWarnin
g: Found unknown categories in columns [1, 9, 13] during transform. These unknown
categories will be encoded as all zeros
  warnings.warn(
```

Random Forest Regressor

In [124…
```
rfr_prepro = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), ['Inches', 'Ram', 'Weight', 'ScreenW', 'Screen
                                   'SecondaryStorage']),
        ('cat', OneHotEncoder(drop = 'first', handle_unknown= 'ignore'), ['Compa
                                                                          'Touch
                                                                          'CPU_c
                                                                          'Secon
    ], remainder = 'passthrough')

rfr = RandomForestRegressor(n_estimators = 100 ,max_features=0.5, max_depth=15,

pipe_rfr = Pipeline([
    ('rfr_prepro', rfr_prepro),
    ('rfr', rfr)
])

# Train the Model
pipe_rfr.fit(X_train, y_train)
```

```
y_pred_rfr = pipe_rfr.predict(X_test)

# Evaluate the test model
mse_test_rfr = mean_squared_error(y_test, y_pred_rfr)
r2_test_rfr = r2_score(y_test, y_pred_rfr)

print(f"Mean Squared Error Test data: {mse_test_rfr}")
print(f"R-squared Test data: {r2_test_rfr}")
```

```
Mean Squared Error Test data: 0.03839887502774533
R-squared Test data: 0.8882054399801229
```

```
D:\Anaconda\Lib\site-packages\sklearn\preprocessing\_encoders.py:242: UserWarnin
g: Found unknown categories in columns [1, 9, 13] during transform. These unknown
categories will be encoded as all zeros
  warnings.warn(
```

AdaBoost Regressor

In [127…

```
ab_prepro = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), ['Inches', 'Ram', 'Weight', 'ScreenW', 'Screen
                                   'SecondaryStorage']),
        ('cat', OneHotEncoder(drop = 'first', handle_unknown= 'ignore'), ['Compa
                                                                          'Touch
                                                                          'CPU_c
                                                                          'Secon

    ], remainder = 'passthrough')

ab_regg = AdaBoostRegressor(n_estimators =500, learning_rate=0.9, random_state=3

pipe_ab = Pipeline([
    ('ab_prepro', ab_prepro),
    ('ab_regg', ab_regg)
])

# Train the Model
pipe_ab.fit(X_train, y_train)

y_pred_ab = pipe_ab.predict(X_test)

# Evaluate the test model
mse_test_ab = mean_squared_error(y_test, y_pred_ab)
r2_test_ab = r2_score(y_test, y_pred_ab)

print(f"Mean Squared Error Test data: {mse_test_ab}")
print(f"R-squared Test data: {r2_test_ab}")
```

```
Mean Squared Error Test data: 0.06575269858865244
R-squared Test data: 0.8085674644497624
```

```
D:\Anaconda\Lib\site-packages\sklearn\preprocessing\_encoders.py:242: UserWarnin
g: Found unknown categories in columns [1, 9, 13] during transform. These unknown
categories will be encoded as all zeros
  warnings.warn(
```

Gradient Boosting Regressor

In [130…
```
gbr_prepro = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), ['Inches', 'Ram', 'Weight', 'ScreenW', 'Screen
                                   'SecondaryStorage']),
        ('cat', OneHotEncoder(drop = 'first', handle_unknown= 'ignore'), ['Compa
                                                                          'Touch
                                                                          'CPU_c
                                                                          'Secon
    ], remainder = 'passthrough')

gb_regg = GradientBoostingRegressor(n_estimators = 1000, random_state=3, alpha=0

pipe_gb = Pipeline([
    ('gbr_prepro', gbr_prepro),
    ('gb_regg', gb_regg)
])

# Train the Model
pipe_gb.fit(X_train, y_train)

y_pred_gb = pipe_gb.predict(X_test)

# Evaluate the test model
mse_test_gb = mean_squared_error(y_test, y_pred_gb)
r2_test_gb = r2_score(y_test, y_pred_gb)

print(f"Mean Squared Error Test data: {mse_test_gb}")
print(f"R-squared Test data: {r2_test_gb}")
```

```
Mean Squared Error Test data: 0.03388778820575893
R-squared Test data: 0.901339026995655
```

```
D:\Anaconda\Lib\site-packages\sklearn\preprocessing\_encoders.py:242: UserWarnin
g: Found unknown categories in columns [1, 9, 13] during transform. These unknown
categories will be encoded as all zeros
  warnings.warn(
```

XGBoost

In [133…
```
xgb_prepro = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), ['Inches', 'Ram', 'Weight', 'ScreenW', 'Screen
                                   'SecondaryStorage']),
        ('cat', OneHotEncoder(drop = 'first', handle_unknown= 'ignore'), ['Compa
                                                                          'Touch
                                                                          'CPU_c
                                                                          'Secon
    ], remainder = 'passthrough')

xgb_regg = XGBRegressor(n_estimators = 100, learning_rate=0.9, max_depth=10, ran

pipe_xgb = Pipeline([
    ('xgb_prepro', xgb_prepro),
    ('xgb_regg', xgb_regg)
])

# Train the Model
pipe_xgb.fit(X_train, y_train)
```

```
y_pred_xgb = pipe_xgb.predict(X_test)

# Evaluate the test model
mse_test_xgb = mean_squared_error(y_test, y_pred_xgb)
r2_test_xgb = r2_score(y_test, y_pred_xgb)

print(f"Mean Squared Error Test data: {mse_test_xgb}")
print(f"R-squared Test data: {r2_test_xgb}")
```

```
Mean Squared Error Test data: 0.05376753239030201
R-squared Test data: 0.8434611008112847
```

```
D:\Anaconda\Lib\site-packages\sklearn\preprocessing\_encoders.py:242: UserWarnin
g: Found unknown categories in columns [1, 9, 13] during transform. These unknown
categories will be encoded as all zeros
  warnings.warn(
```

Exporting the Model

In [137…

```
import pickle

# pickle.dump(pipe_rid,open('model.pkl','wb'))

# Streamlit-notworking
pickle.dump(data,open('data.pkl','wb'))
pickle.dump(pipe_rid,open('pipe.pkl','wb'))
```