Project Title	**Laptop Price Analysis**
language	Machine learning, python, SQL, Excel
Tools	VS code, Jupyter notebook
Domain	Data Analyst
Project Difficulties level	Advance

Dataset : Dataset is available in the given link. You can download it at your convenience.

[Click here to download data set]

About Dataset

The original dataset was pretty compact with a lot of details in each column. The columns mostly consisted of long strings of data, which was pretty human-readable and concise but for Machine Learning algorithms to work more efficiently it's better to separate the different details into their own columns. After doing so, 28 duplicate rows were exposed and removed with this dataset being the final result.

Formatting Issues

The file was saved in standard encoding so there shouldn't be any problems reading it in pandas. Though if it gives you any trouble you could try reading it with the encoding=`'ISO-8859-1'` parameter, as this was the original dataset's formatting.

Columns:

- `Company`: Laptop Manufacturer.
- `Product`: Brand and Model.
- `TypeName`: Laptop Type (Notebook, Ultrabook, Gaming, …etc).
- `Inches`: Screen Size.
- `Ram`: Total amount of RAM in laptop (GBs).
- `OS`: Operating System installed.
- `Weight`: Laptop Weight in kilograms.
- `Price_euros`: Price of Laptop in Euros. (Target)
- `Screen`: screen definition (Standard, Full HD, 4K Ultra HD, Quad HD+).
- `ScreenW`: screen width (pixels).
- `ScreenH`: screen height (pixels).
- `Touchscreen`: whether or not the laptop has a touchscreen.
- `IPSpanel`: whether or not the laptop has an IPSpanel.

- `RetinaDisplay`: whether or not the laptop has retina display.
- `CPU_company`
- `CPU_freq`: frequency of laptop CPU (Hz).
- `CPU_model`
- `PrimaryStorage`: primary storage space (GB).
- `PrimaryStorageType`: primary storage type (HDD, SSD, Flash Storage, Hybrid).
- `SecondaryStorage`: secondary storage space if any (GB).
- `SecondaryStorageType`: secondary storage type (HDD, SSD, Hybrid, None).
- `GPU_company`
- `GPU_model`

Machine Learning Project for Beginners: Laptop Price Analysis

This project will help you understand how to analyze and predict laptop prices using a dataset containing laptop specifications. It is a simple regression task where we predict the price of a laptop based on its features like brand, processor, RAM, storage, etc.

Steps in the Project:

1. **Problem Statement**:
    - The task is to build a machine learning model that can predict the price of laptops based on their features.

2. **Dataset**:
   - You can either scrape data from e-commerce websites or use a public dataset.
   - Here is a sample structure of the dataset:
3.

Brand	Processor	RAM	Storage	Screen Size	GPU	Weight	Price
Dell	i5	8GB	512GB	15.6	None	2.5	600
HP	i7	16GB	1TB	14	Nvidia	2.0	1000
Apple	M1	8GB	256GB	13.3	None	1.4	1200

4. You can find datasets like this on Kaggle or other open sources.

Step-by-Step Project Implementation:

Step 1: Import Libraries

First, you need to import the required libraries for data manipulation and machine learning.

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split
```

```python
from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score
```

Step 2: Load Dataset

Load the dataset using **pandas**.

```python
# Example of loading a dataset

df = pd.read_csv('laptop_price_data.csv')


# Check the first few rows of the dataset

df.head()
```

Step 3: Data Preprocessing

a. Handle Missing Values

You need to handle missing data by either filling it or dropping rows with missing values.

```python
# Check for missing values

df.isnull().sum()


# Fill missing values if any (for simplicity, you can drop missing values)

df = df.dropna()
```

b. Convert Categorical Data to Numerical

Since machine learning models don't work with categorical data directly, you need to convert columns like `Brand`, `Processor`, and `GPU` into numerical format using **Label Encoding** or **One-Hot Encoding**.

```
# Convert categorical columns to numerical using One-Hot Encoding
df = pd.get_dummies(df, columns=['Brand', 'Processor', 'GPU'], drop_first=True)
```

c. Feature Selection

You need to select the features and the target variable.

```
X = df.drop('Price', axis=1)  # Features (independent variables)
y = df['Price']               # Target variable (dependent variable)
```

Step 4: Train-Test Split

Split the data into training and testing sets to evaluate the model's performance.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Step 5: Train the Model

Here, we'll use **Linear Regression** to train the model. Linear regression is simple and great for beginners.

```
# Initialize and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

Step 6: Make Predictions

After training the model, make predictions on the test set.

```
# Predicting the price using the test set
y_pred = model.predict(X_test)
```

Step 7: Evaluate the Model

You can evaluate your model using common regression metrics like **Mean Squared Error (MSE)** and **R-squared (R^2)**.

```
# Calculate Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

```
# Calculate R-squared

r2 = r2_score(y_test, y_pred)

print(f"R-squared: {r2}")
```

Step 8: Visualize Results

Finally, you can visualize the results to compare predicted vs actual values.

```
plt.scatter(y_test, y_pred)

plt.xlabel("Actual Prices")

plt.ylabel("Predicted Prices")

plt.title("Actual vs Predicted Laptop Prices")

plt.show()
```

Project Summary:

- **Problem**: Predict the price of laptops based on their specifications.
- **Steps**:
  - Load and preprocess the dataset.
  - Convert categorical data to numerical format using One-Hot Encoding.
  - Train a linear regression model to predict laptop prices.
  - Evaluate the model using MSE and R-squared.

    - Visualize the actual vs predicted prices.

Key Points:

- **Data Preprocessing** is crucial for converting categorical data to numerical data.
- Use **train-test split** to evaluate the model on unseen data.
- **Linear Regression** is a good starting point, but you can experiment with other models like **Random Forest** or **XGBoost** to improve performance.

This project is great for beginners to learn how to clean data, build models, and make predictions.

[Sample link]

Importing Necessary Libraries¶

In [2]:

```
import seaborn as sns

import matplotlib.pyplot as plt
```

Loading the data

In [3]:

```python
data = pd.read_csv('/kaggle/input/laptop-prices/laptop_prices.csv')

data.head()
```

Out[3]:

	Company	Product	TypeName	Inches	Ram	OS	Weight	Price_euros	Screen	ScreenW	...	RetinaDisplay	CPU_company	CPU_freq	CPU_model	PrimaryStorage	SecondaryStorage	PrimaryStorageType	SecondaryStorageType	GPU_company	GPU_model
0	Apple	MacBook Pro	Ultrabook	13.3	8	macOS	1.37	1339.69	Standard	2560	...	Yes	Intel	2.3	Core i5	128	0	SSD	No	Intel	Iris Plus Graphics 640
1	Ap	Ma	Ultr	13	8	ma	1.	898.	Sta	14	..	No	Inte	1.	Core	128	0	Flash Stora	No	Inte	HD

	ple	cbook Air	abook	.3		cOS	34	94	ndard	40	.		l	8	i5			ge		l	Graphics 6000
2	HP	250 G6	Notebook	15.6	8	No OS	1.86	575.00	Full HD	1920	...	No	Intel	2.5	Core i5 7200U	256	0	SSD	No	Intel	HD Graphics 620
3	Apple	MacBook Pro	Ultrabook	15.4	16	macOS	1.83	2537.45	Standard	2880	...	Yes	Intel	2.7	Core i7	512	0	SSD	No	AMD	Radeon Pro 455

4	Apple	MacBook Pro	Ultrabook	13.3	8	macOS	1.37	1803.60	Standard	2560	...	Yes	Intel	3.1	Core i5	256	0	SSD	No	Intel	Iris Plus Graphics 650

5 rows × 23 columns

Information Related to Data

In [4]:

```
data.shape
```

Out[4]:

```
(1275, 23)
```

In [5]:

```
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 [6]:

```
data.isnull().sum()
```

Out[6]:

```
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 [7]:

```
data.describe()
```

Out[7]:

	Inches	Ram	Weight	Price_eu	Screen	ScreenH	CPU_fre	PrimarySt	SecondarySt

				ros	W		q	orage	orage
count	1275.000000	1275.000000	1275.000000	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	2.302980	444.517647	176.069020
std	1.429470	5.097809	0.669196	700.752504	493.346186	283.883940	0.503846	365.537726	415.960655
min	10.100000	2.000000	0.690000	174.000000	1366.000000	768.000000	0.900000	8.000000	0.000000
25%	14.000000	4.000000	1.500000	609.000000	1920.000000	1080.000000	2.000000	256.000000	0.000000
50%	15.600000	8.000000	2.040000	989.000000	1920.000000	1080.000000	2.500000	256.000000	0.000000

75%	15.600000	8.000000	2.310000	1496.500000	1920.000000	1080.000000	2.700000	512.000000	0.000000
max	18.400000	64.000000	4.700000	6099.000000	3840.000000	2160.000000	3.600000	2048.000000	2048.000000

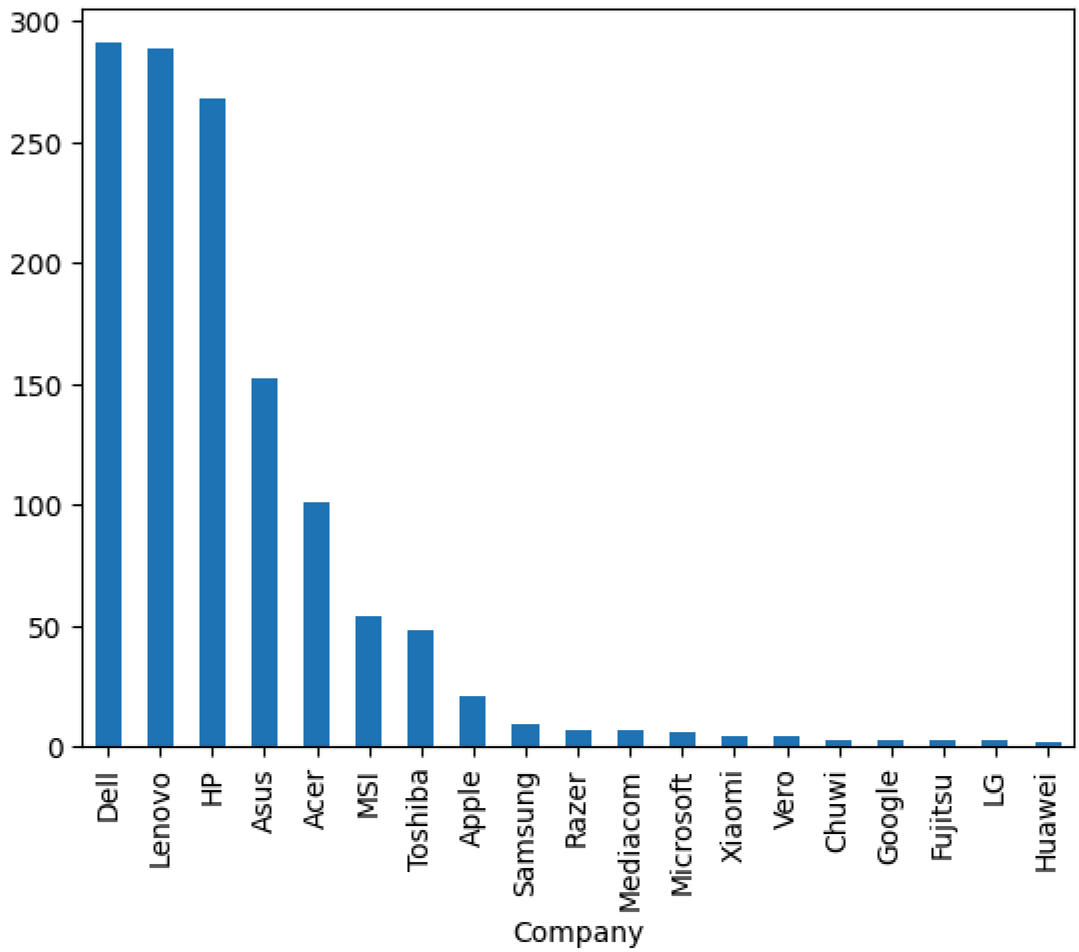
Exploratory Data Analysis : Univeriate Analysis

In [8]:

```
data['Company'].value_counts().plot(kind = 'bar')
```

Out[8]:

```
<Axes: xlabel='Company'>
```

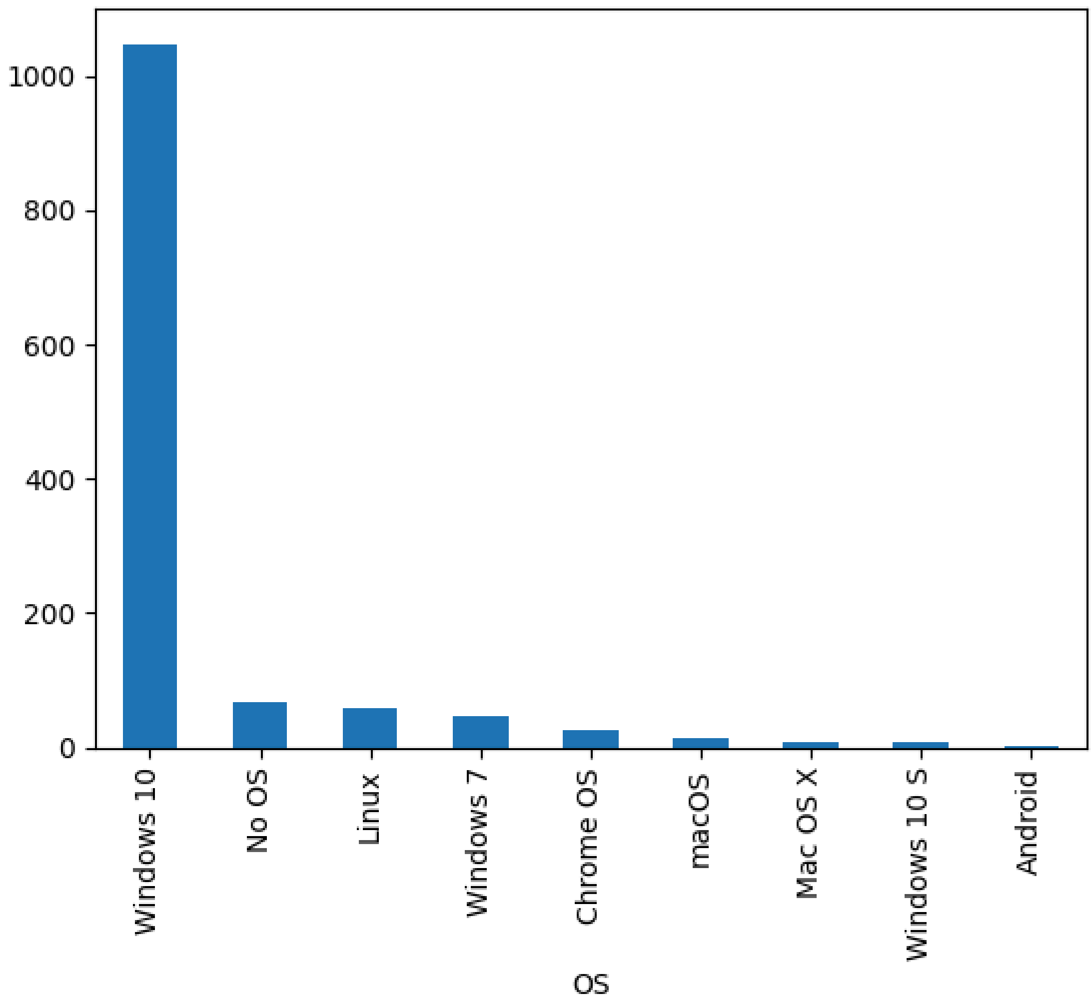


In [9]:

```
data['OS'].value_counts().plot(kind = 'bar' , x = data['OS'])
```

Out[9]:

```
<Axes: xlabel='OS'>
```

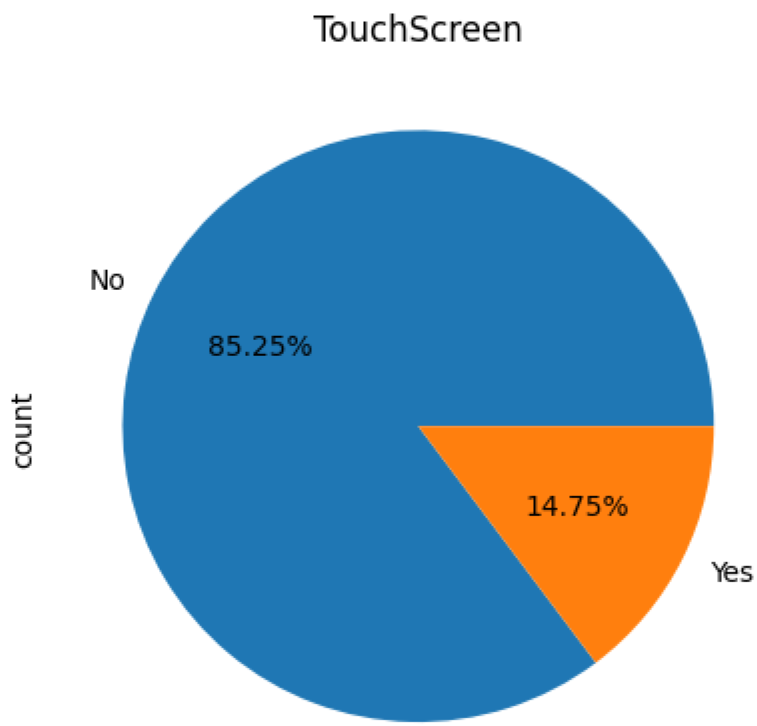


In [10]:

```
data['Touchscreen'].value_counts().plot(kind = 'pie', autopct = '%.2f%%' ,
title = 'TouchScreen')
```

Out[10]:

```
<Axes: title={'center': 'TouchScreen'}, ylabel='count'>
```

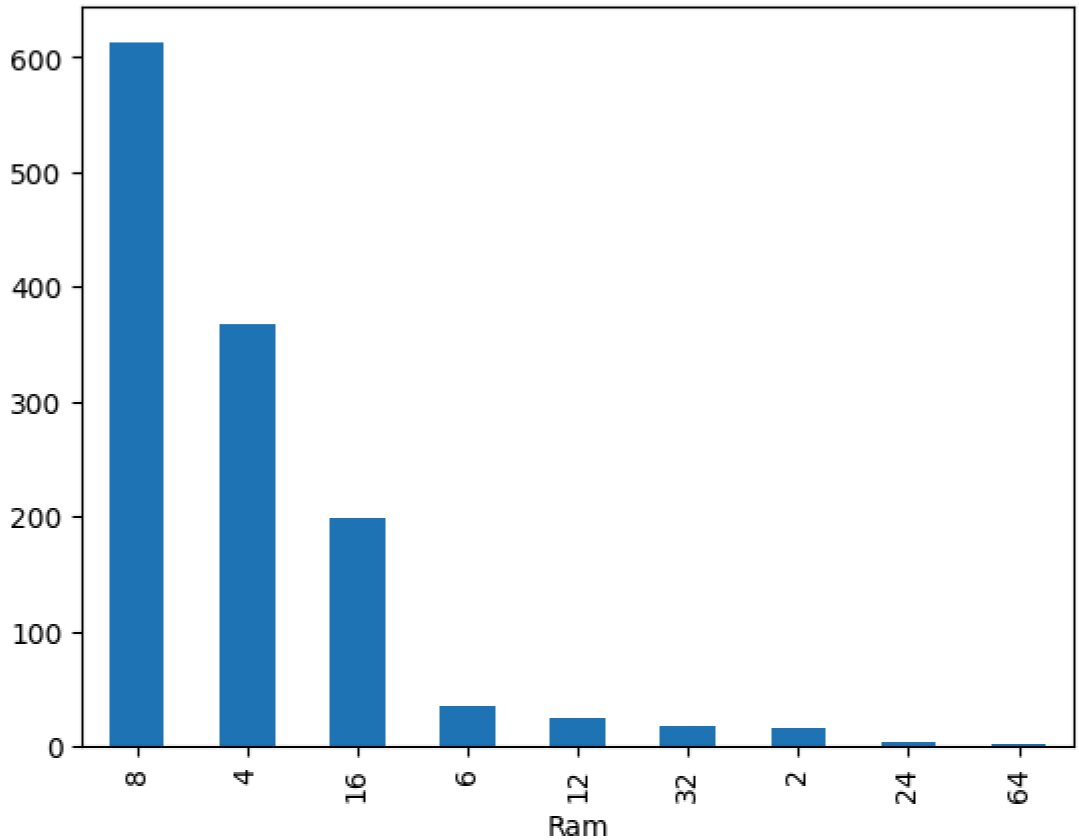


In [11]:

```
data['Ram'].value_counts().plot(kind = 'bar')
```

Out[11]:

```
<Axes: xlabel='Ram'>
```

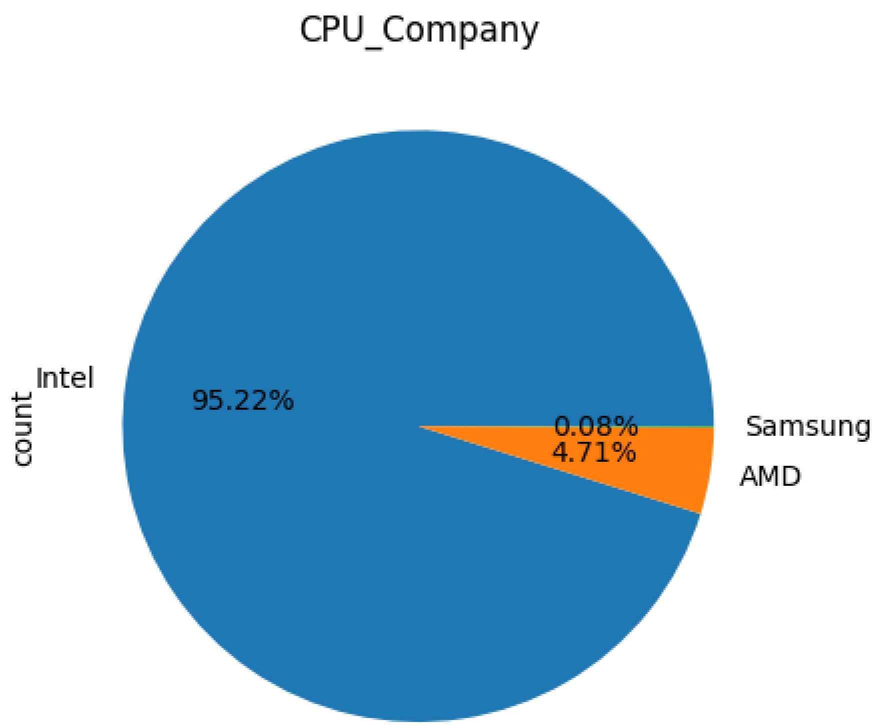


In [12]:

```
data['CPU_company'].value_counts().plot(kind = 'pie' , autopct  = '%.2f%%',
title = 'CPU_Company')
```

Out[12]:

```
<Axes: title={'center': 'CPU_Company'}, ylabel='count'>
```

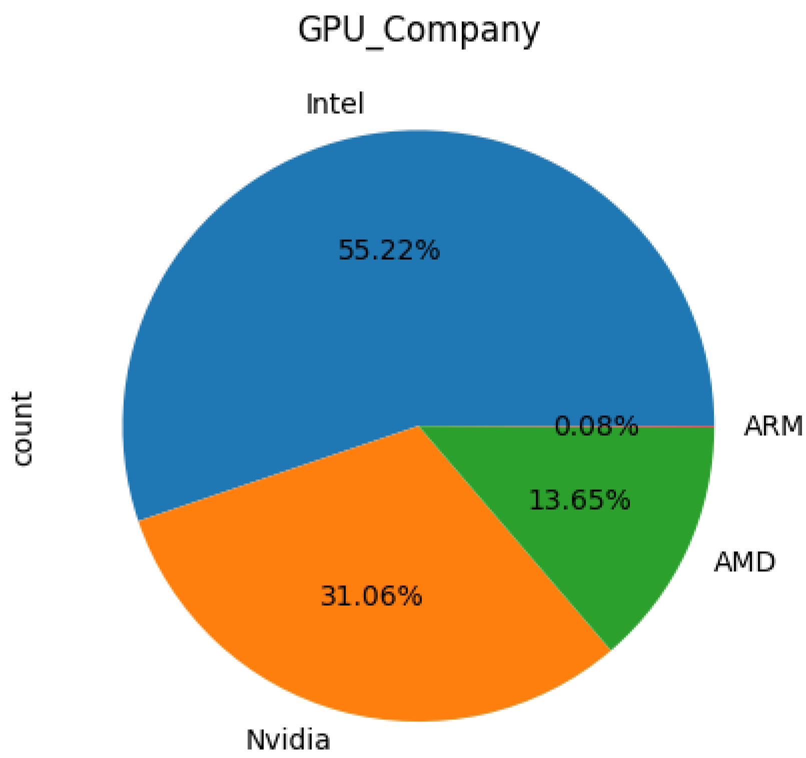


In [13]:

```
data['GPU_company'].value_counts().plot(kind = 'pie' , autopct  = '%.2f%%',
title = 'GPU_Company')
```

Out[13]:

```
<Axes: title={'center': 'GPU_Company'}, ylabel='count'>
```

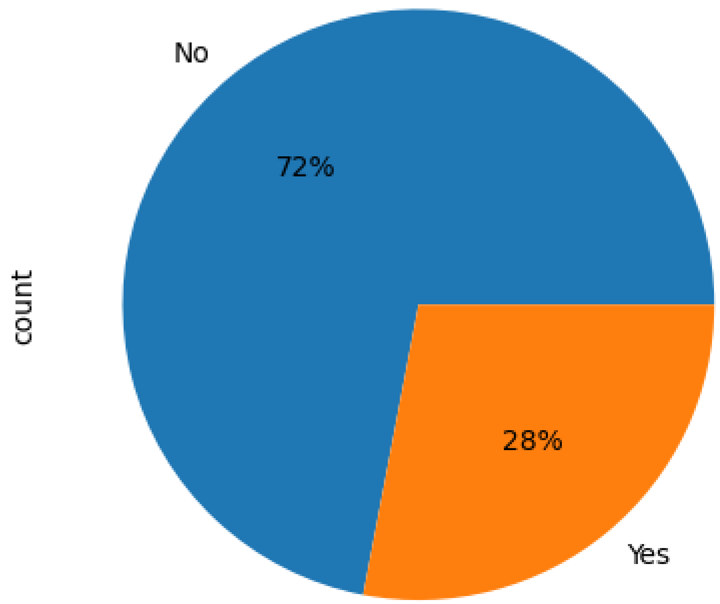


In [14]:

```
data['IPSpanel'].value_counts().plot(kind = 'pie' , autopct = '%.f%%')
```

Out[14]:

```
<Axes: ylabel='count'>
```

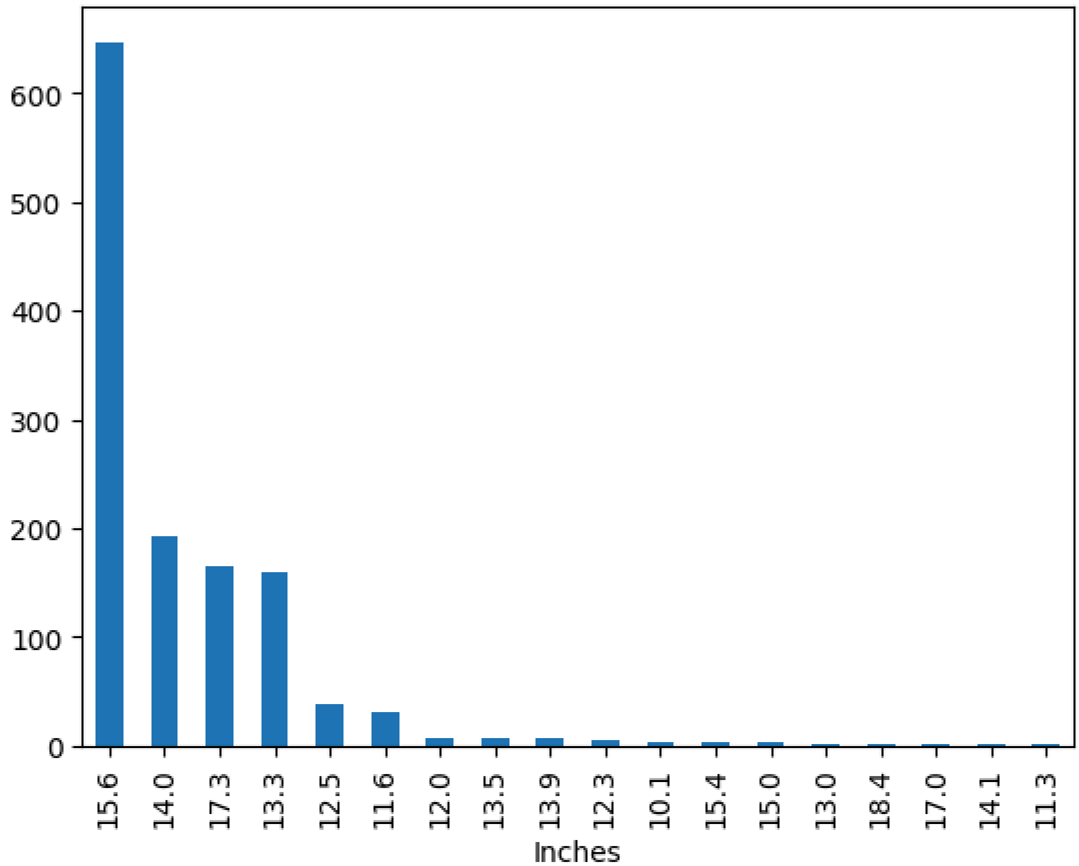


In [15]:

```
data['Inches'].value_counts().plot(kind = 'bar')
```

Out[15]:

```
<Axes: xlabel='Inches'>
```

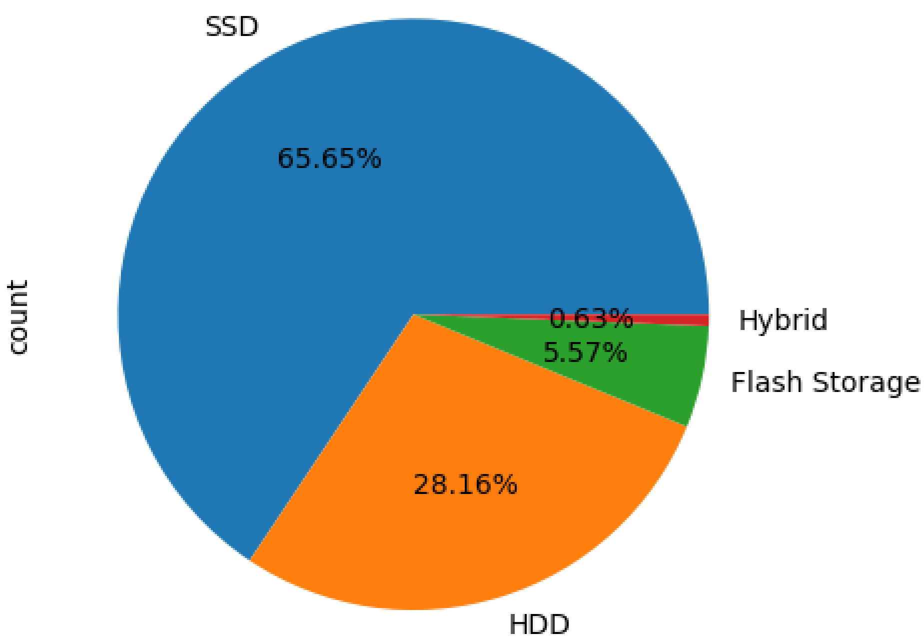


In [16]:

```
data['PrimaryStorageType'].value_counts().plot(kind = 'pie' , autopct = '%.2f%%')
```

Out[16]:

```
<Axes: ylabel='count'>
```

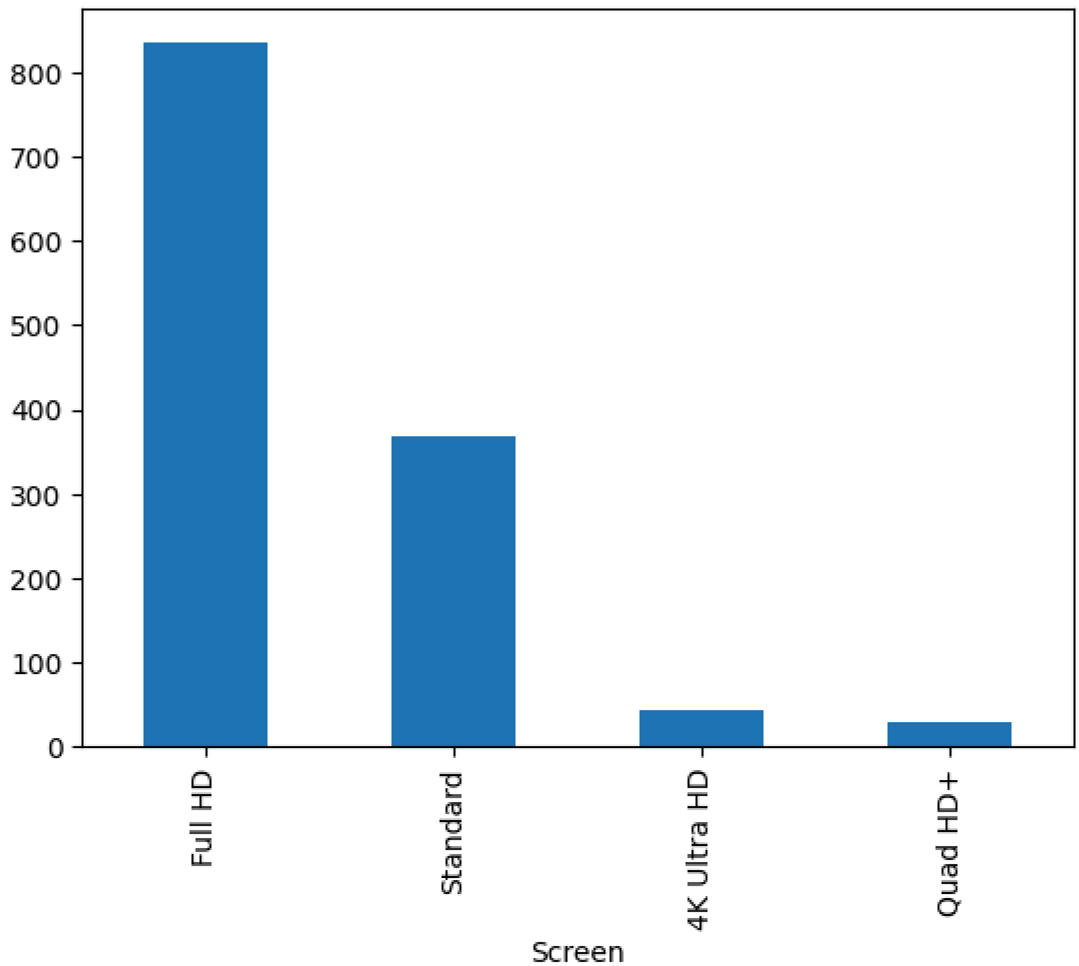


In [17]:

```
data['Screen'].value_counts().plot(kind = 'bar')
```

Out[17]:

```
<Axes: xlabel='Screen'>
```

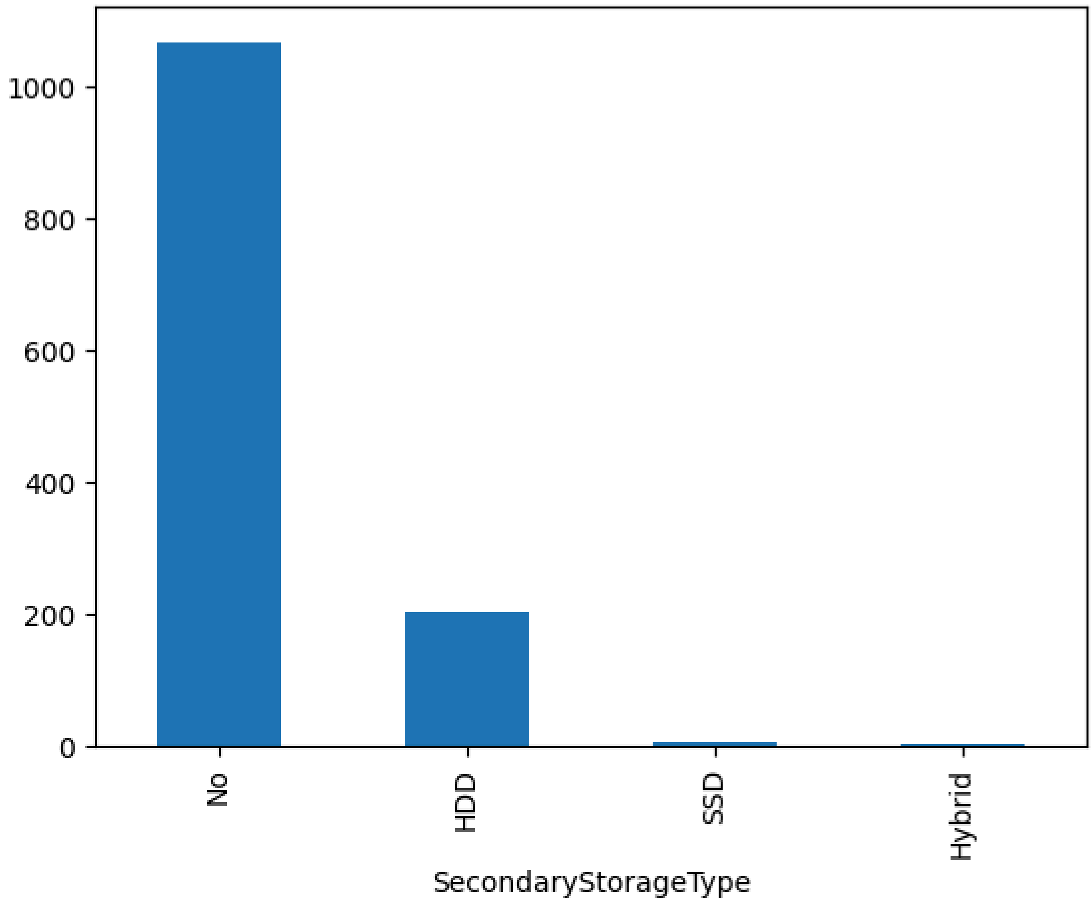


In [18]:

```
data['SecondaryStorageType'].value_counts().plot(kind = 'bar')
```

Out[18]:

```
<Axes: xlabel='SecondaryStorageType'>
```



Bivariate Analysis

In [19]:

```
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
```

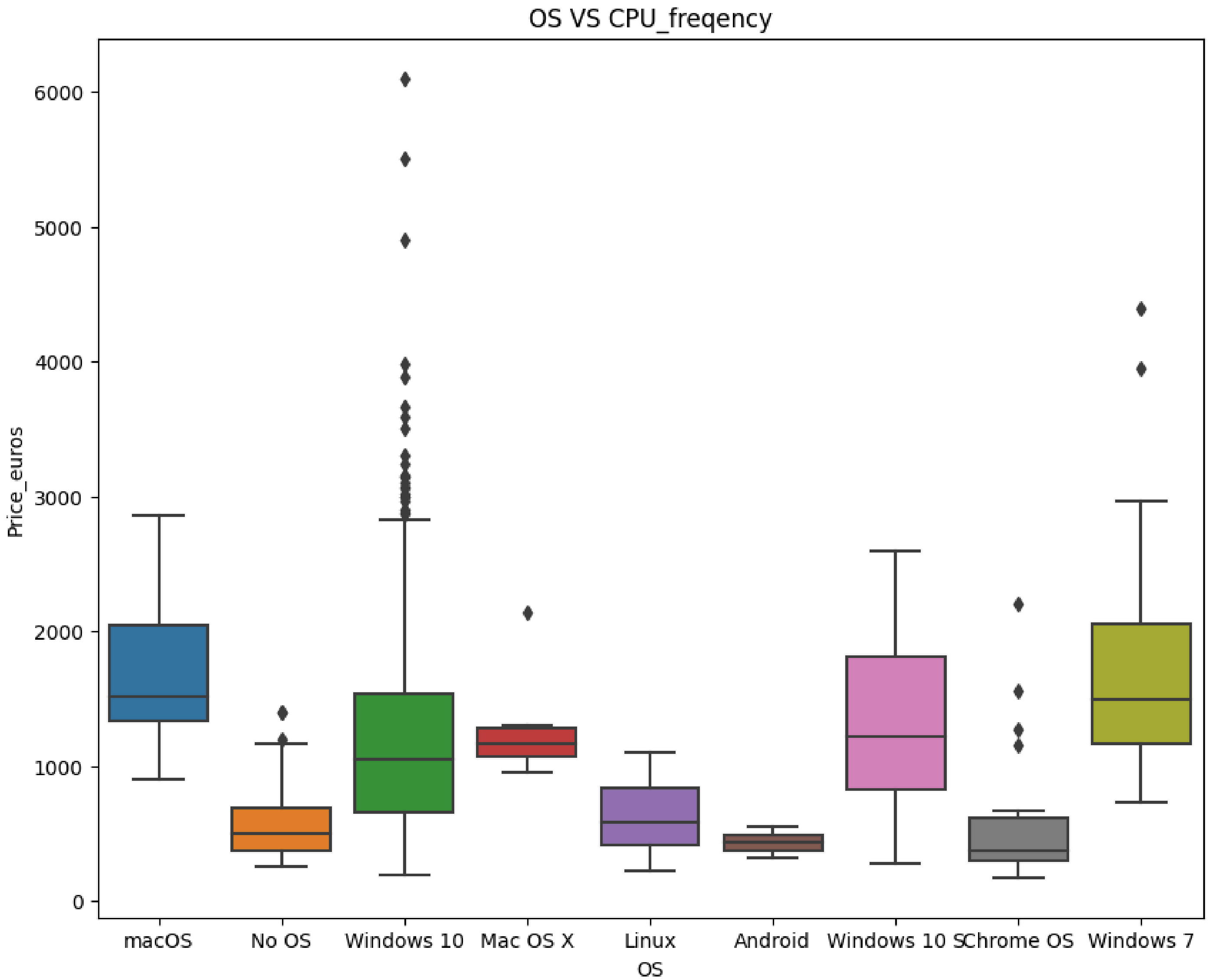
In [20]:

```
plt.figure(figsize = (10,8))

sns.boxplot(x = data['OS'], y= data['Price_euros'])

plt.title('OS VS CPU_freqency')

plt.show()
```

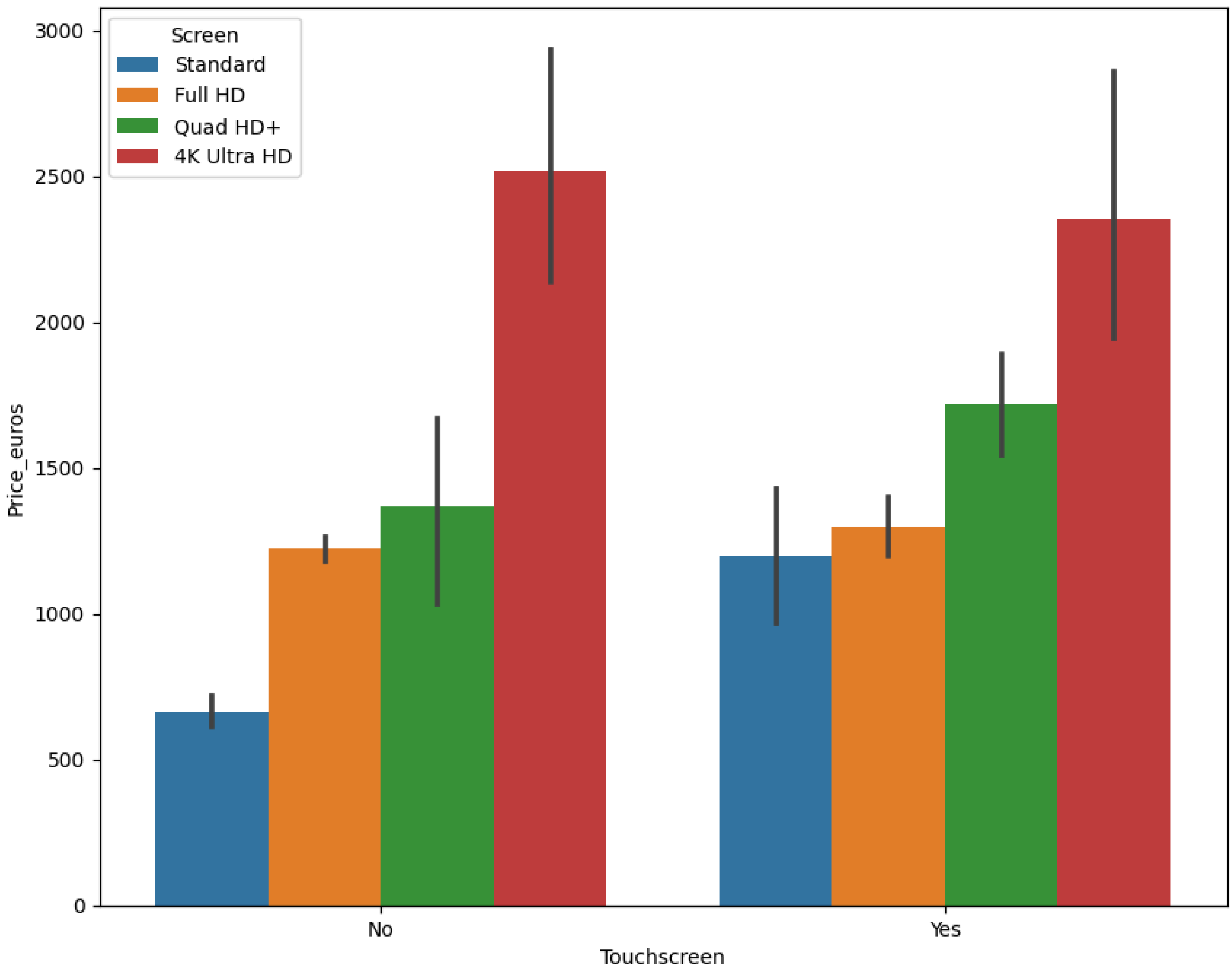


In [21]:

```
plt.figure(figsize = (10,8))

sns.barplot(x  =  data['Touchscreen'],  y=  data['Price_euros']  ,  hue  =
data['Screen'])

plt.show()
```

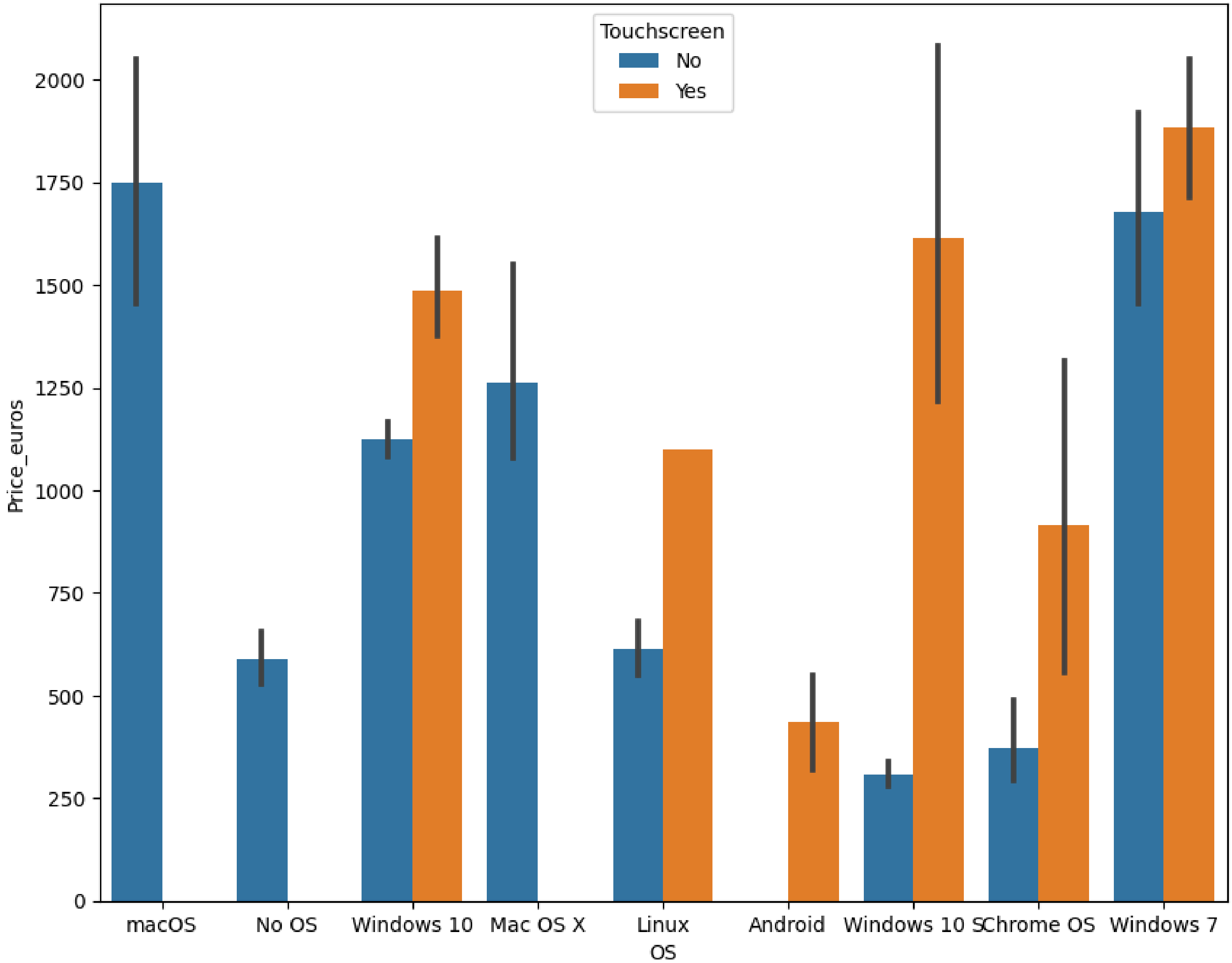


In [22]:

```
plt.figure(figsize = (10,8))

sns.barplot(x = data['OS'], y= data['Price_euros'] , hue = data['Touchscreen'])

plt.show()
```



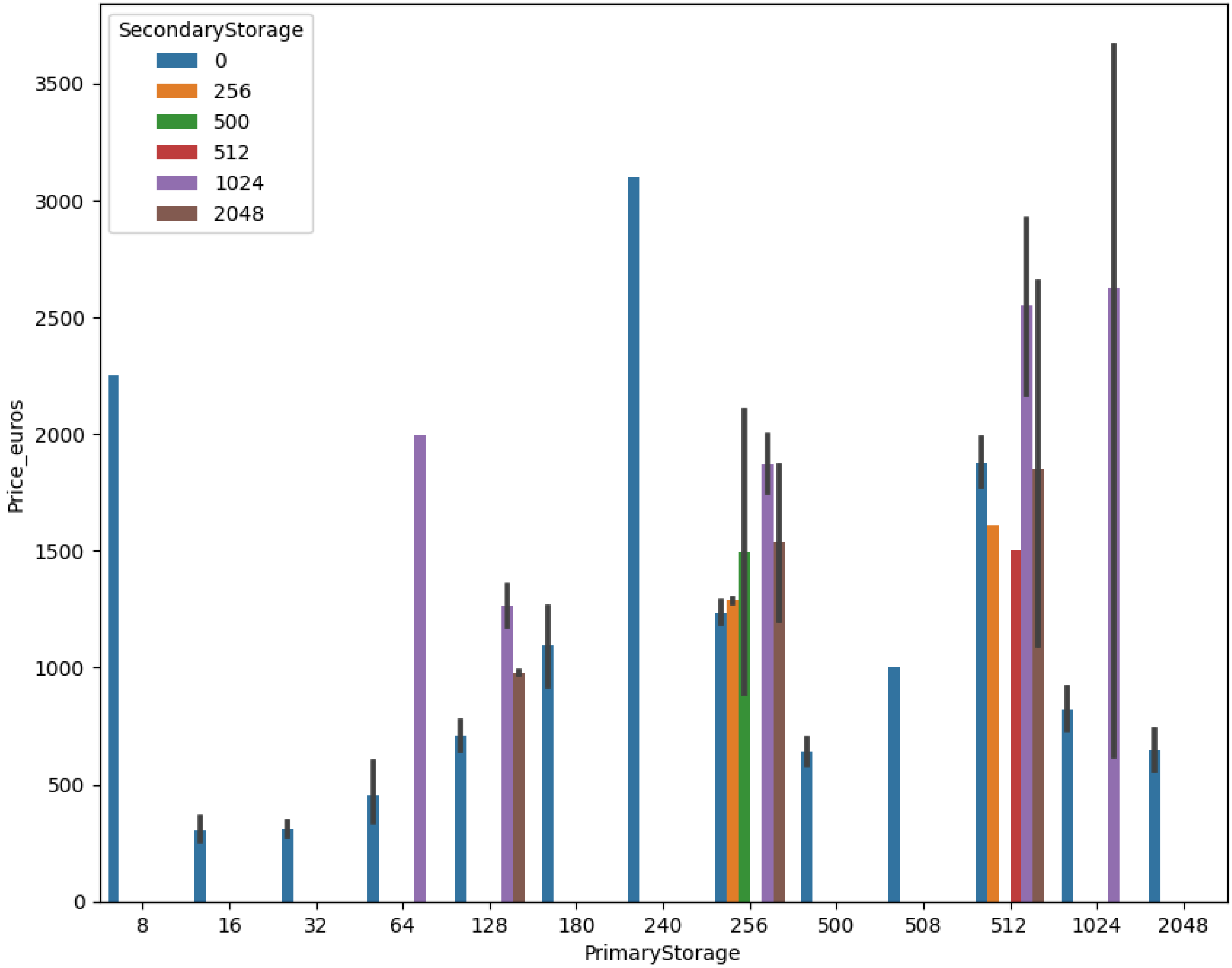
In [23]:

```
plt.figure(figsize = (10,8))

sns.barplot(x  =  data['PrimaryStorage'],  y=  data['Price_euros']  ,  hue  =

data['SecondaryStorage'])

plt.show()
```

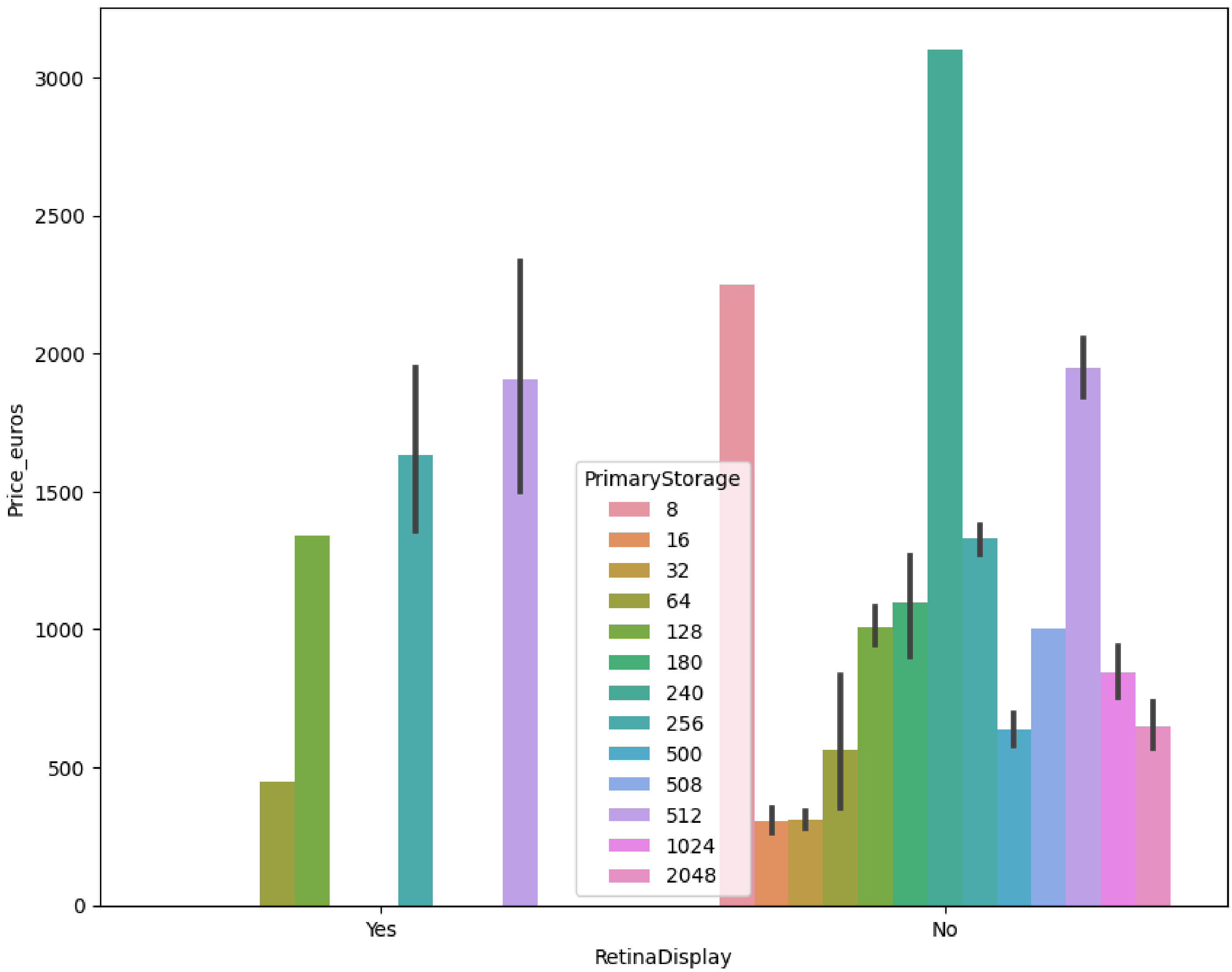


In [24]:

```
plt.figure(figsize = (10,8))

sns.barplot(x  =  data['RetinaDisplay'],  y=  data['Price_euros'],  hue  =
data['PrimaryStorage'])

plt.show()
```

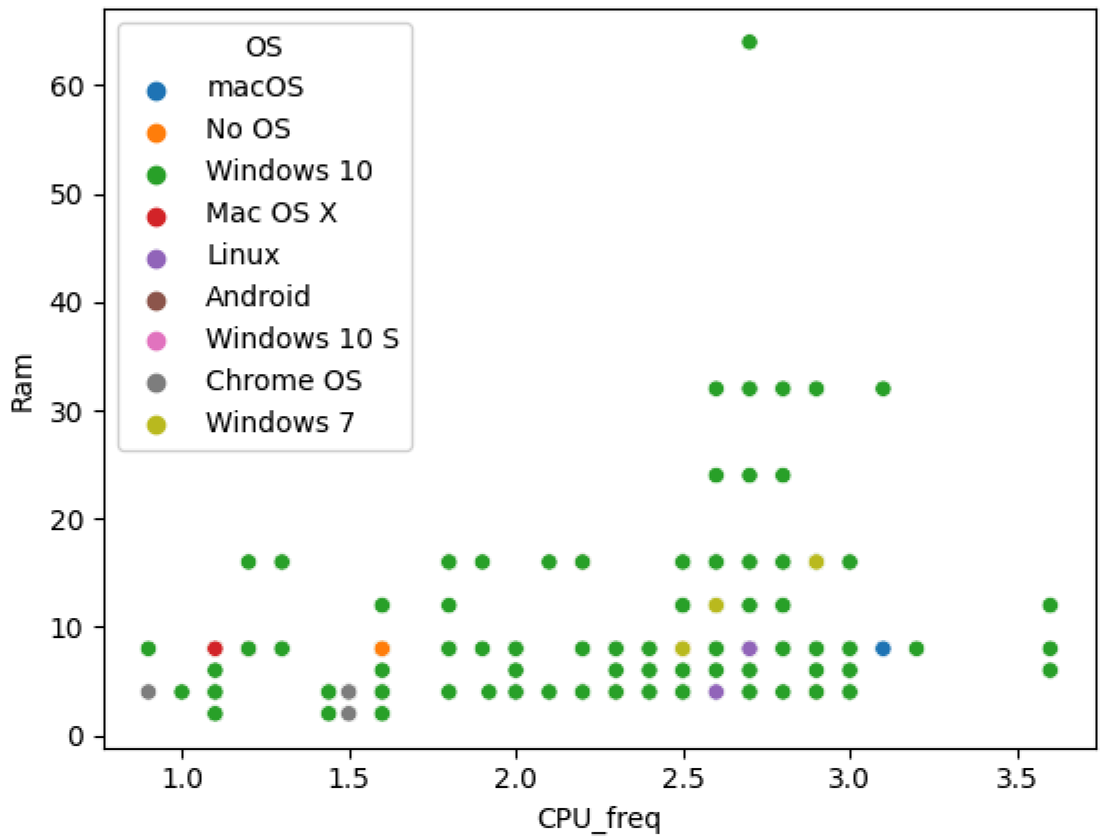


In [25]:

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

Out[25]:

```
<Axes: xlabel='CPU_freq', ylabel='Ram'>
```

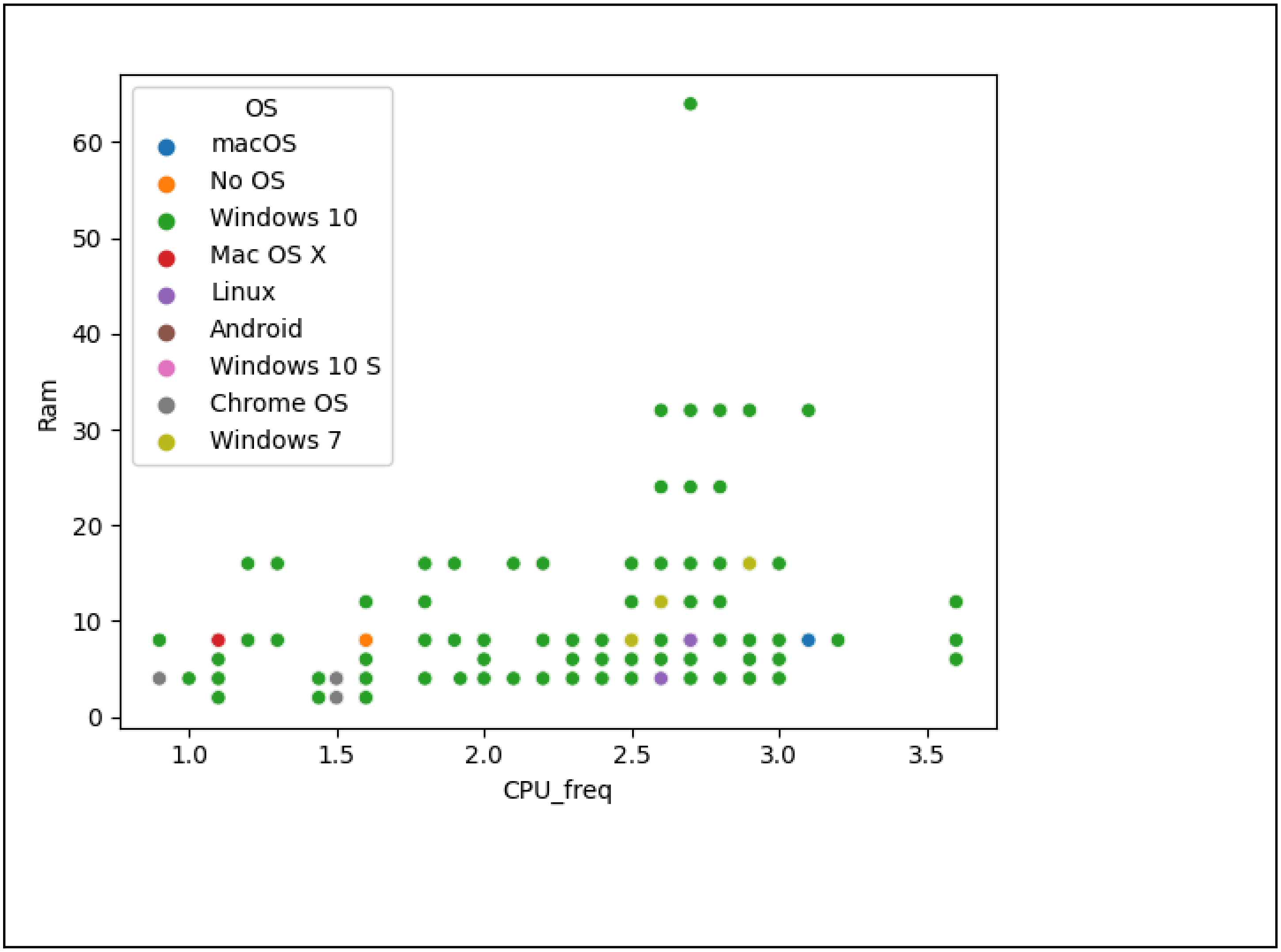


In [26]:

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

Out[26]:

```
<Axes: xlabel='CPU_freq', ylabel='Ram'>
```



Reference link