RV College of Engineering®

Go, change the world

RASHTREEYA SHIKSHAN SAMITHI TRUST

R.V. COLLEGE OF ENGINEERING

(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)

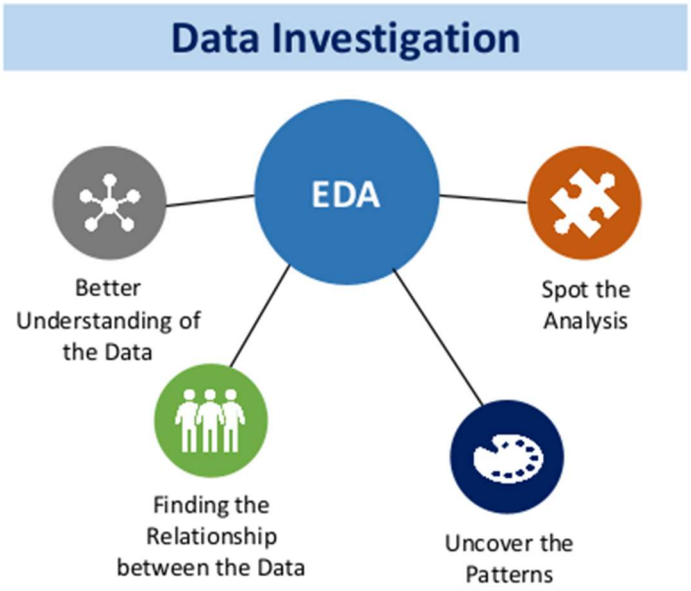
Python EL Report

Submitted by	TALASILA DHEERAJ – RVCE22BCS151 SKANDA P R – RVCE22BCS207 SUHAS RAJ H R – RVCE22BCS219 VAIBHAV U NAVALAGI – RVCE22BCS229
Submitted to	Prof. Rajesh R M Assistant Professor Artificial Intelligence & Cyber Security

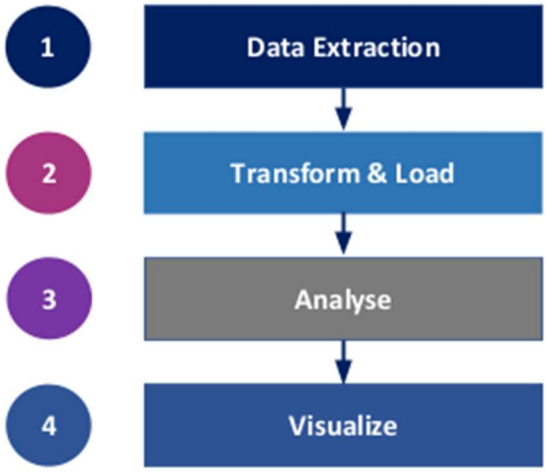
INTRODUCTION TO EDA

Exploratory Data Analysis (EDA) is a process of exploring data in order to gain insights, identify patterns, and uncover relationships. It is a crucial step in the Data Science process and helps to uncover the underlying structure of the data. In this presentation, we will explore the data related to laptops and their features.

We will use a variety of techniques to gain insights into the data such as visualizing the data, computing summary statistics, and performing correlation analysis. Through this process, we will be able to identify trends and relationships between the different features of laptops.



DATA VISUALIZATION



Data visualization is a powerful tool to gain insights into the data. It can help to identify patterns, trends, and relationships between the different features of the data. We can use various plotting techniques such as histograms, bar plots, and scatter plots to visualize the data.

We can also use interactive visualizations such as heatmaps and 3D plots to gain deeper insights into the data. These visualizations can help us identify correlations between different features of the data and help us to understand the underlying structure of the data.

SUMMARY STATISTICS

Summary statistics are numerical metrics that summarize the data. They can help to identify patterns and trends in the data. We can use summary statistics such as mean, median, and mode to understand the central tendency of the data. We can also use measures of variability such as standard deviation and range to understand the spread of the data.

We can also use summary statistics such as correlation and covariance to understand the relationships between the different features of the data. These metrics can help us to identify trends and patterns in the data and gain deeper insights into the data.

Problem statement

To perform EDA on any dataset to understand the domain of any aspect.

Topic chosen

EDA of laptops based on their company, model, price, and other general features of laptop.

Step-1: GETTING THE DATA

First, we import pandas library and open the csv file in read mode and display it.

```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/laptops.csv',encoding = 'latin-1')
df
```

	Unnamed: 0	Company	Product	TypeName	Inches	ScreenResolution
0	1	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600
1	2	Apple	Macbook Air	Ultrabook	13.3	1440x900
2	3	HP	250 G6	Notebook	15.6	Full HD 1920x1080
3	4	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800
4	5	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600
...	...	...	...	...	...	...
1298	1316	Lenovo	Yoga 500-14ISK	2 in 1 Convertible	14.0	IPS Panel Full HD / Touchscreen 1920x1080
1299	1317	Lenovo	Yoga 900-13ISK	2 in 1 Convertible	13.3	IPS Panel Quad HD+ / Touchscreen 3200x1800
1300	1318	Lenovo	IdeaPad 100S-14IBR	Notebook	14.0	1366x768
1301	1319	HP	15-AC110nv (i7-6500U/6GB/1TB/Radeon	Notebook	15.6	1366x768
1302	1320	Asus	X553SA-XX031T (N3050/4GB/500GB/W10)	Notebook	15.6	1366x768

Step-2: INFORMATION OF THE DATASET

df.info()- gives information about the dataset and the datatypes involved in it.

df.shape()-gives no of rows and columns in the dataset.

df.size()- gives total size of the dataset

```
df.info() #gives us the information about our dataframe
```

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

```
df.shape # 1303 rows and 13 cols
```

```
(1303, 13)
```

```
df.size# the total no of elements in df
```

```
16939
```

Step-3: REMOVING NULL VALUES

This is an important method in the process of EDA. This is done in order to clean the data. It removes the unwanted data i.e, the data which is zero/ blank and henceforth making the stats precise.

Df.isnull().sum()- used to check for null values

Df.drop()-used to drop the column/row as specified by data scientist.

```
df.isnull().sum()
```

```
Unnamed: 0          0
Company             0
Product             0
TypeName            0
Inches              0
ScreenResolution    0
Cpu                 0
Ram                 0
Memory              0
Gpu                 0
OpSys               0
Weight              0
Price_euros         0
dtype: int64
```

```
#to remove/drop the Unnamed: 0 column
df = df.drop(columns = 'Unnamed: 0')
```

Step-4: REMOVING OTHER FORMS OF UNNECESSARY DATA

Df.str.replace()- used to remove part of a string and replace it by the need of the user.

Df.astype()-used to convert dataframe of one type to the other as per the requirement of the user.

```
#removing the 'GB' from the Ram column and keeping only integer value
df['Ram'] = df['Ram'].str.replace('GB','')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Company           1303 non-null   object 
 1   Product           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_euros       1303 non-null   float64
dtypes: float64(2), object(10)
memory usage: 122.3+ KB
```

```
df
```

```
#converting the data of the Ram column from string to int
df['Ram'] = df['Ram'].astype('int64')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Company           1303 non-null   object 
 1   Product           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   int64  
 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_euros       1303 non-null   float64
dtypes: float64(2), int64(1), object(9)
memory usage: 122.3+ KB
```

```
#removing the 'kg' from the data of weight column
df['Weight'] = df['Weight'].str.replace('kg','')
df['Weight'] = df['Weight'].astype('float64')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Company           1303 non-null   object 
 1   Product           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   int64
 7   Memory          1303 non-null   object
 8   Gpu             1303 non-null   object
 9   OpSys           1303 non-null   object
 10  Weight          1303 non-null   float64
 11  Price_euros     1303 non-null   float64
dtypes: float64(3), int64(1), object(8)
memory usage: 122.3+ KB
```

Step-5: VISUALIZATION OF THE DATA

Visualization of data is done by plotting graphs of the dataset using matplotlib and seaborn libraries.

sns.displot and **sns.barplot** are two popular functions from the Python data visualization library Seaborn.

sns.displot is used to create a histogram or a kernel density estimate (KDE) plot. It is a versatile function that can handle a wide range of data types and allows for customization of various plot features such as the number of bins, color, and label.

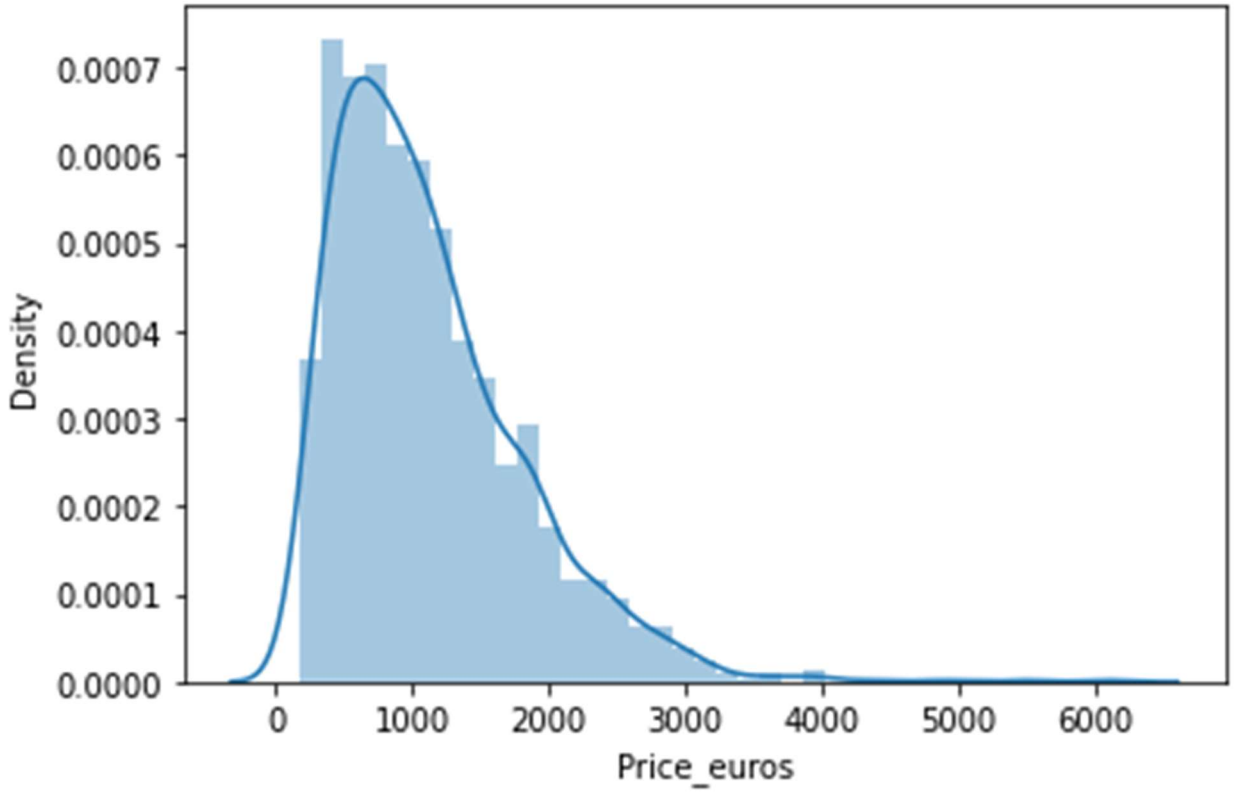
sns.barplot is used to create a bar chart. It is useful for comparing the values of different categories or groups. By default, **sns.barplot**

shows the mean value of the data, but you can specify other summary statistics like median or sum.

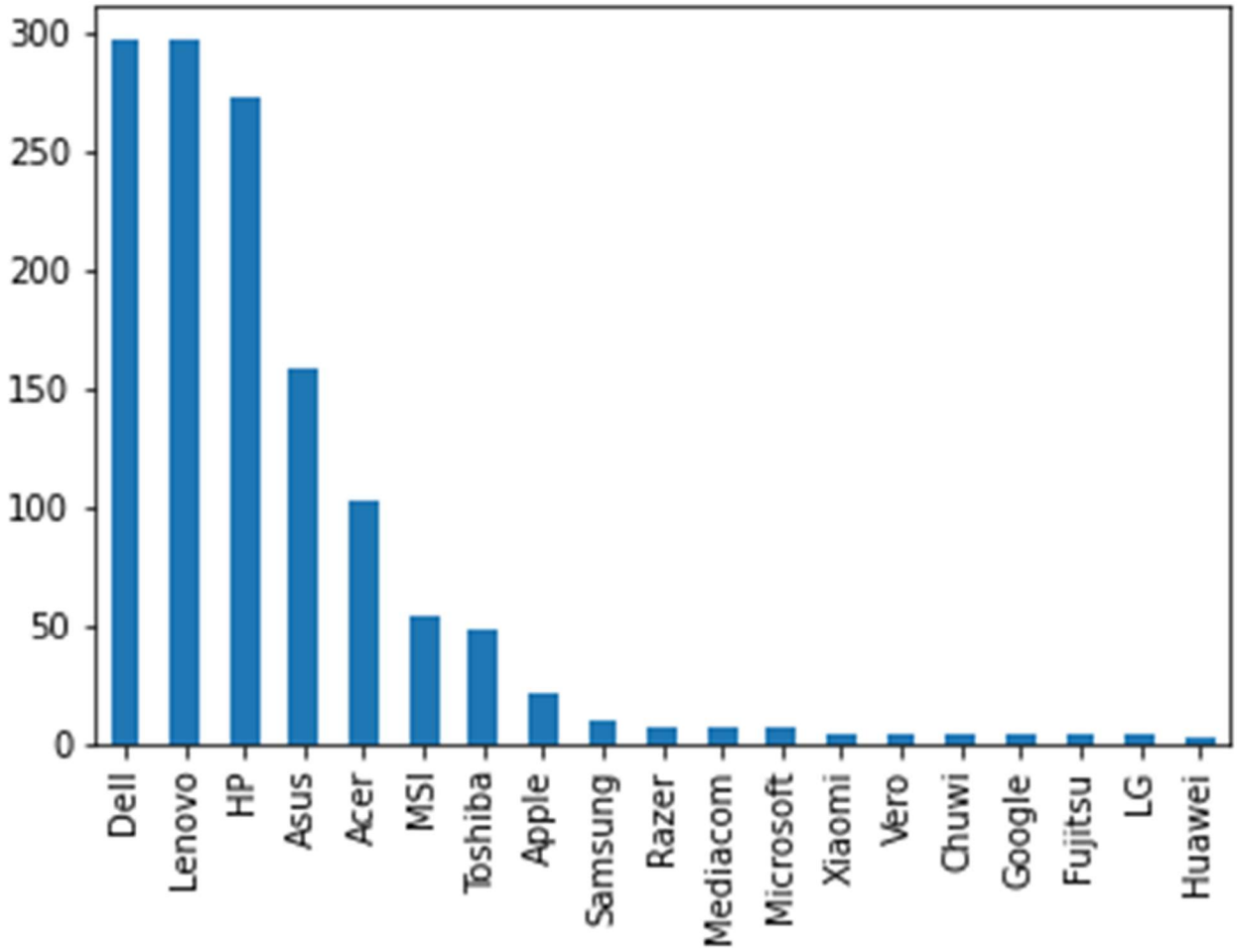
The labels and their angles with respect to the graphs can be modified using xticks(), yticks() functions in matplotlib.

The size of the graph could be controlled by using figsize() function in matplotlib.

Plotting a graph of price of the laptops versus the number of laptops



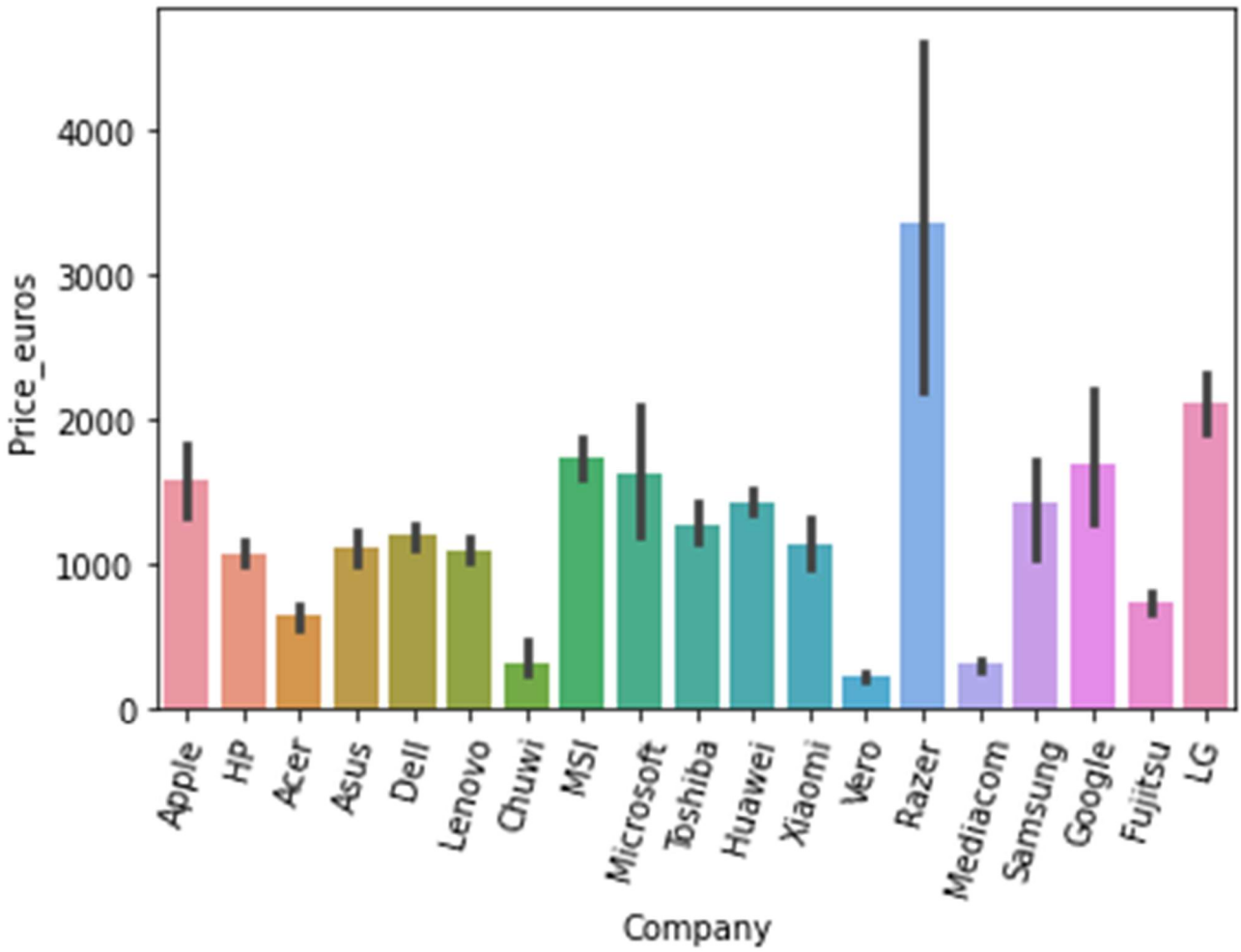
Plotting a graph of company of the laptops versus the number of laptops available.



Plotting a graph of Company of the laptop versus their prices.

```
import matplotlib.pyplot as plt
sns.barplot(x = df['Company'],y = df['Price_euros'])
plt.xticks(rotation = '75')
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18]), <a list of 19 Text major ticklabel objects>)
```



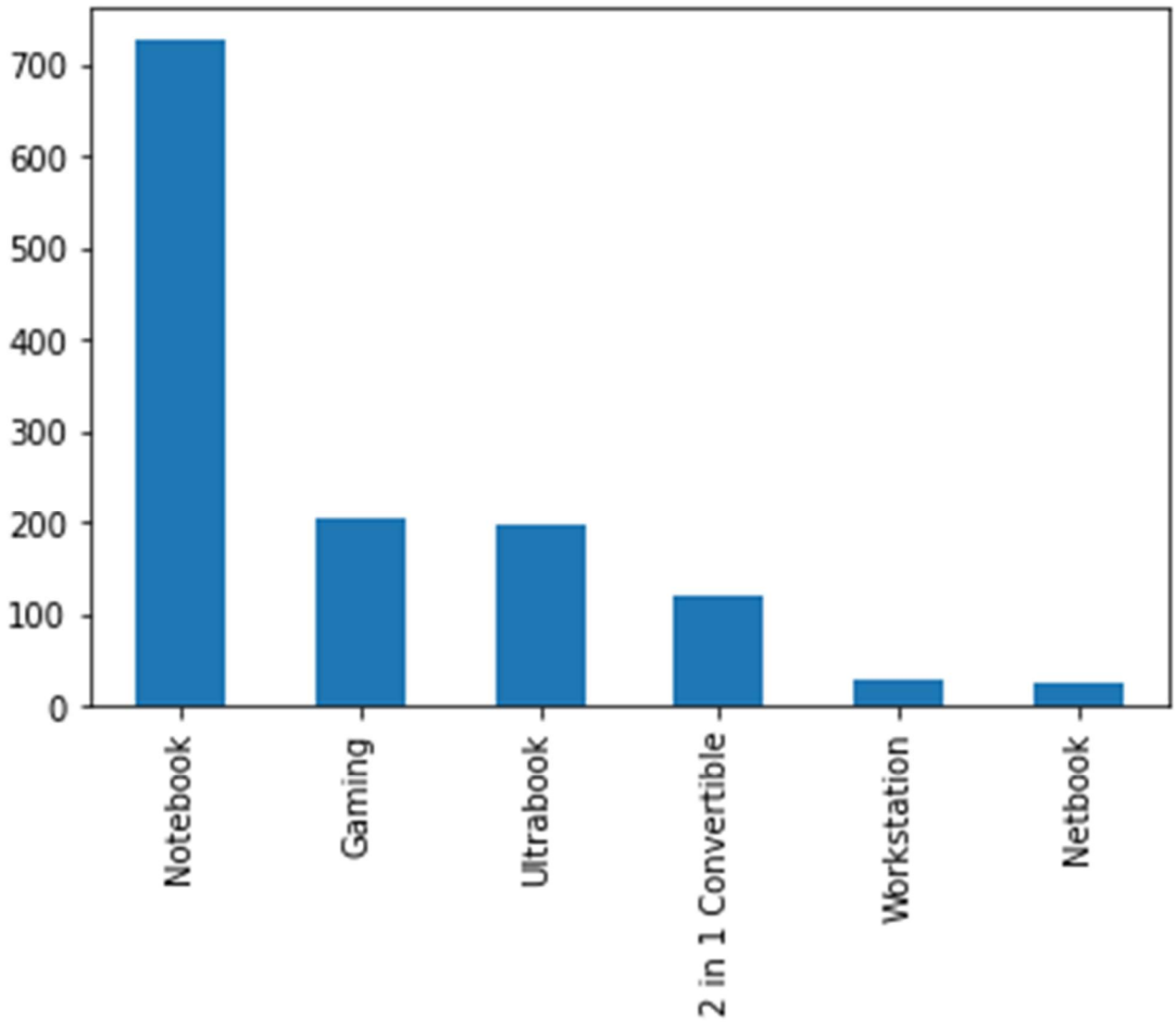
Plotting a graph of Type of laptops and number of laptops available.

```
df['TypeName'].value_counts()
```

```
Notebook              727
Gaming                205
Ultrabook             196
2 in 1 Convertible    121
Workstation            29
Netbook                25
Name: TypeName, dtype: int64
```

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

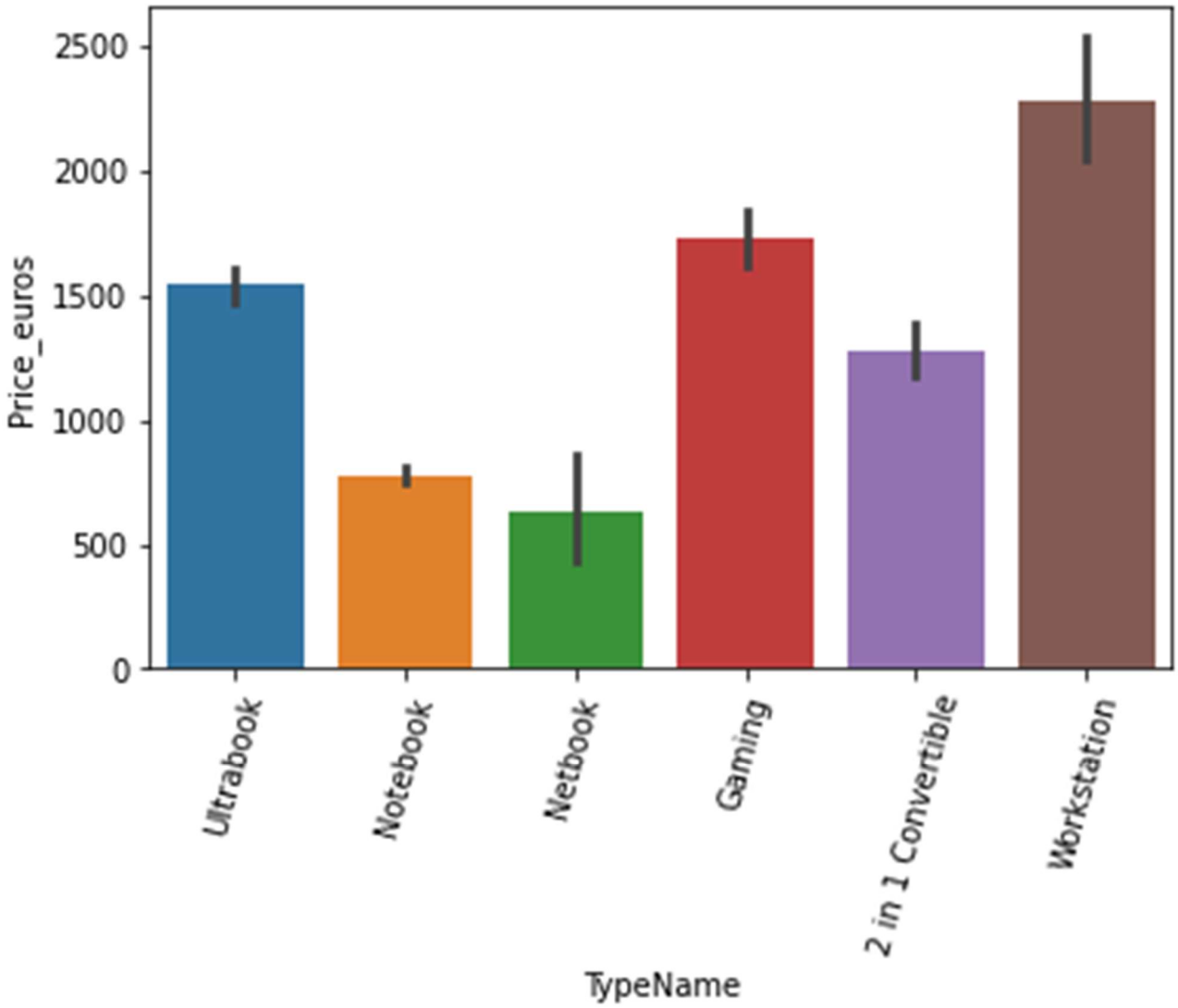
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f47fe728f90>
```



Plotting a graph of Type of laptops and their price.

```
sns.barplot(x = df['TypeName'],y = df['Price_euros'])
plt.xticks(rotation = 75)
```

```
(array([0, 1, 2, 3, 4, 5]), <a list of 6 Text major ticklabel objects>)
```



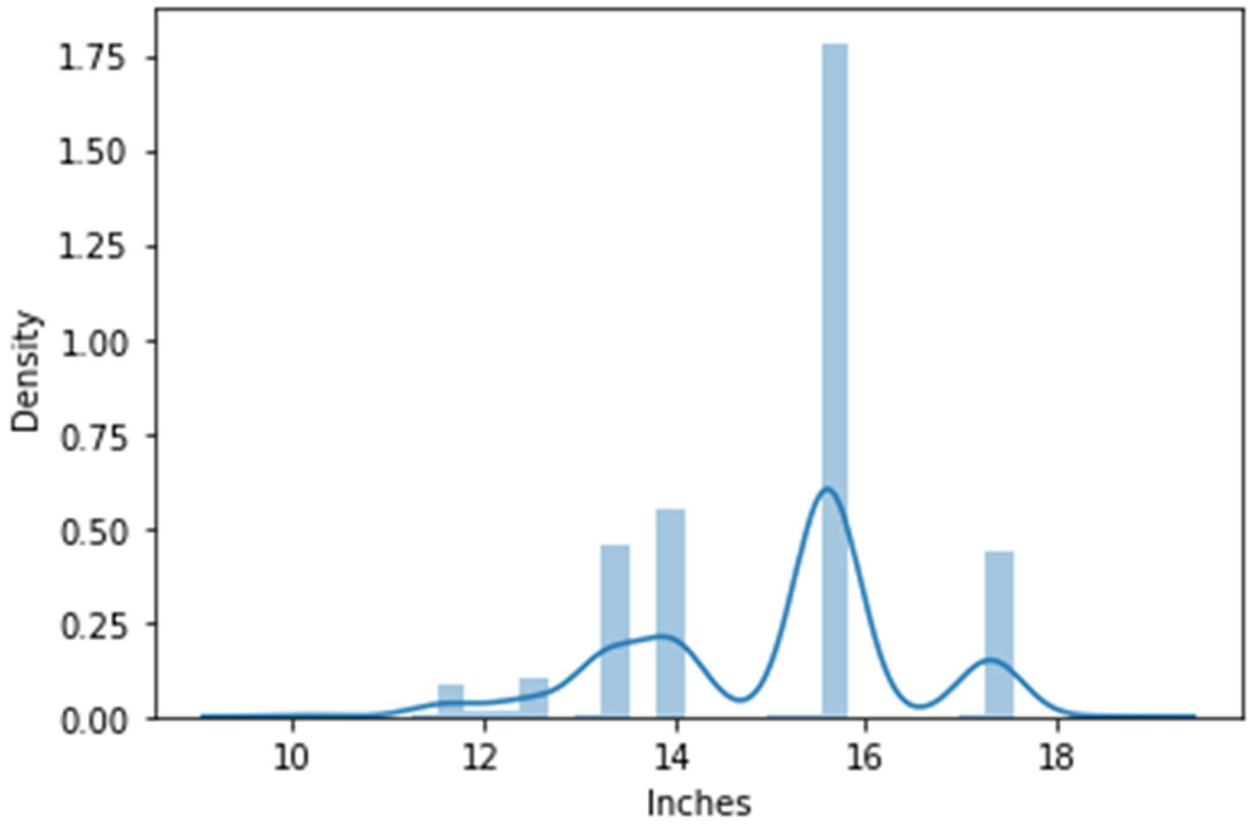
Plotting a graph of Screen size of laptops versus the number of laptops available.

```
df['Inches'].value_counts()
```

```
15.6    665
14.0    197
13.3    164
17.3    164
```

```
12.5     39
11.6     33
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: Inches, dtype: int64
```

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

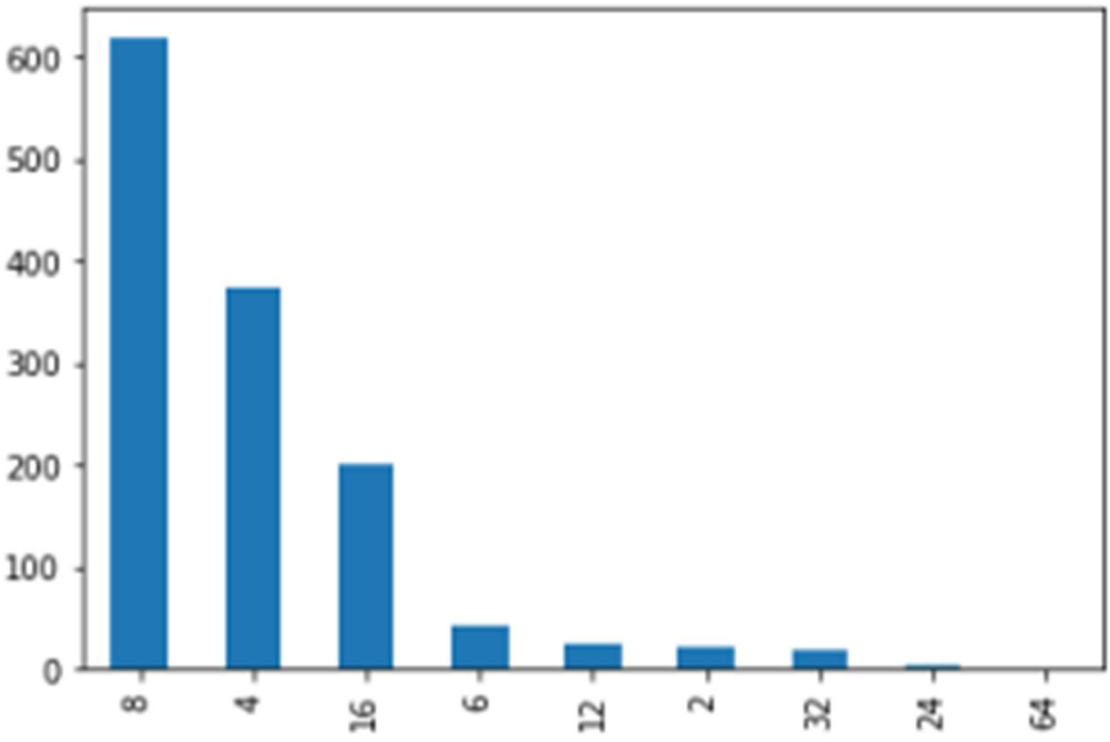


Plotting a graph of configuration of RAM versus the number of laptops available.

```
df['Ram'].value_counts()
```

```
8     619
4     375
16    200
6      41
12     25
2      22
32     17
24      3
64      1
Name: Ram, dtype: int64
```

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

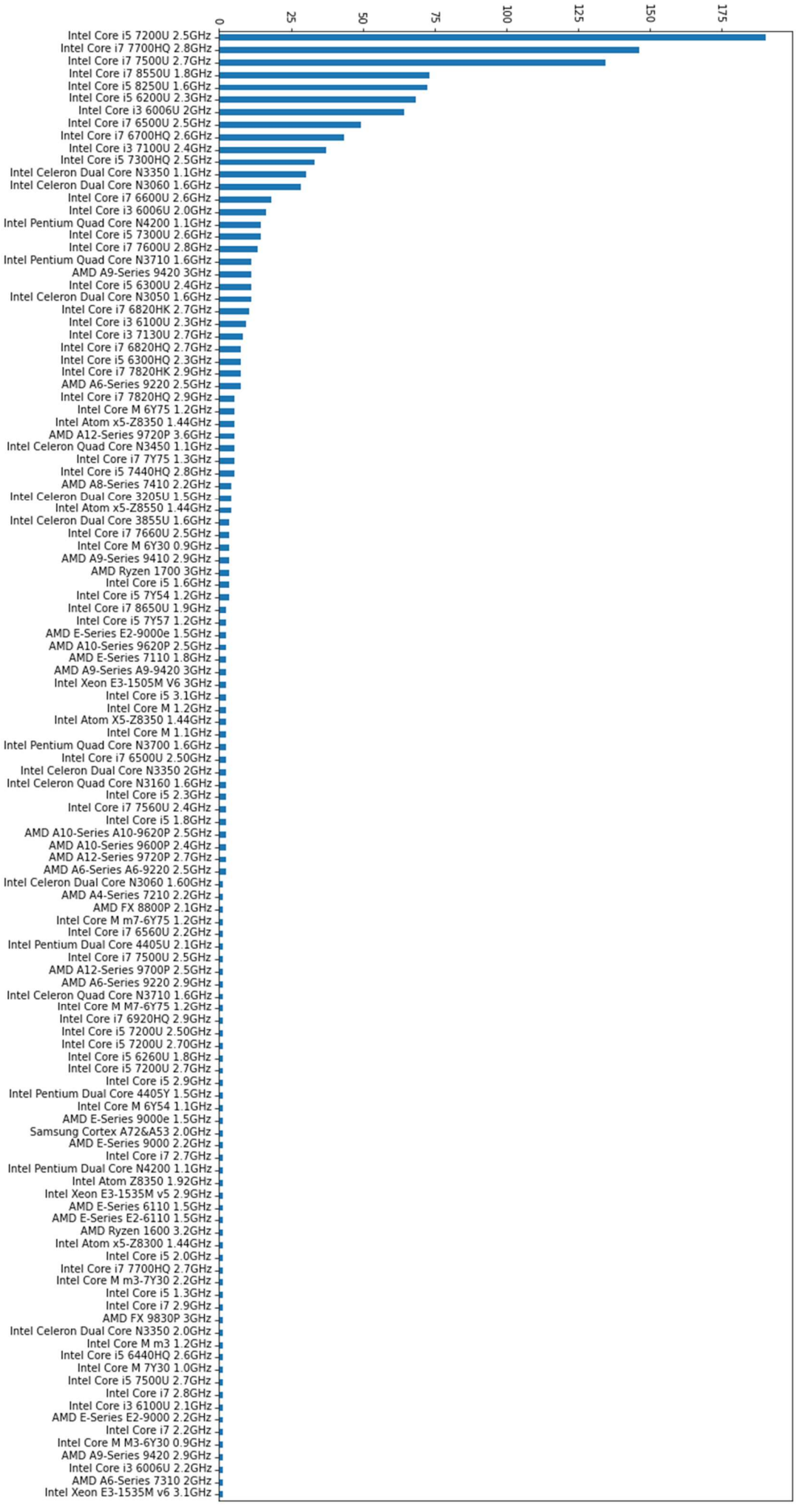


Plotting a graph of type of CPU versus the number of laptops available.

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

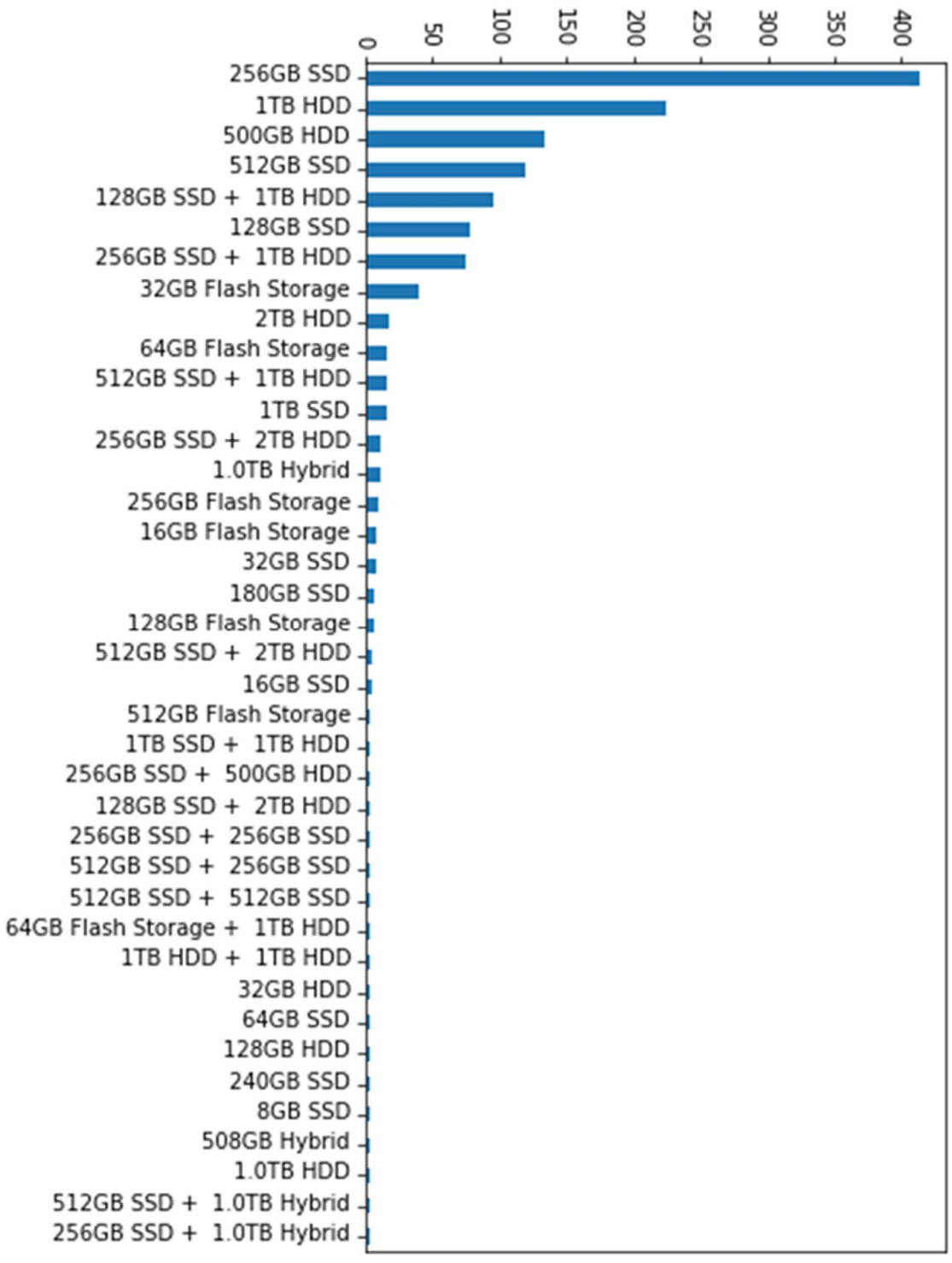
```
Intel Core i5 7200U 2.5GHz       190
Intel Core i7 7700HQ 2.8GHz      146
Intel Core i7 7500U 2.7GHz       134
Intel Core i7 8550U 1.8GHz        73
Intel Core i5 8250U 1.6GHz        72
                                 ...
Intel Core M M3-6Y30 0.9GHz        1
AMD A9-Series 9420 2.9GHz          1
Intel 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
```

```
plt.figure(figsize = (25,10))
df['Cpu'].value_counts().plot(kind = 'bar')
```


0
25
50
75
100
125
150
175
Intel Core i5 7200U 2.5GHz
Intel Core i7 7700HQ 2.8GHz
Intel Core i7 7500U 2.7GHz
Intel Core i7 8550U 1.8GHz
Intel Core i5 8250U 1.6GHz
Intel Core i5 6200U 2.3GHz
Intel Core i3 6006U 2GHz
Intel Core i7 6500U 2.5GHz
Intel Core i7 6700HQ 2.6GHz
Intel Core i3 7100U 2.4GHz
Intel Core i5 7300HQ 2.5GHz
Intel Celeron Dual Core N3350 1.1GHz
Intel Celeron Dual Core N3060 1.6GHz
Intel Core i7 6600U 2.6GHz
Intel Core i3 6006U 2.0GHz
Intel Pentium Quad Core N4200 1.1GHz
Intel Core i5 7300U 2.6GHz
Intel Core i7 7600U 2.8GHz
Intel Pentium Quad Core N3710 1.6GHz
AMD A9-Series 9420 3GHz
Intel Core i5 6300U 2.4GHz
Intel Celeron Dual Core N3050 1.6GHz
Intel Core i7 6820HK 2.7GHz
Intel Core i3 6100U 2.3GHz
Intel Core i3 7130U 2.7GHz
Intel Core i7 6820HQ 2.7GHz
Intel Core i5 6300HQ 2.3GHz
Intel Core i7 7820HK 2.9GHz
AMD A6-Series 9220 2.5GHz
Intel Core i7 7820HQ 2.9GHz
Intel Core M 6Y75 1.2GHz
Intel Atom x5-Z8350 1.44GHz
AMD A12-Series 9720P 3.6GHz
Intel Celeron Quad Core N3450 1.1GHz
Intel Core i7 7Y75 1.3GHz
Intel Core i5 7440HQ 2.8GHz
AMD A8-Series 7410 2.2GHz
Intel Celeron Dual Core 3205U 1.5GHz
Intel Atom x5-Z8550 1.44GHz
Intel Celeron Dual Core 3855U 1.6GHz
Intel Core i7 7660U 2.5GHz
Intel Core M 6Y30 0.9GHz
AMD A9-Series 9410 2.9GHz
AMD Ryzen 1700 3GHz
Intel Core i5 1.6GHz
Intel Core i5 7Y54 1.2GHz
Intel Core i7 8650U 1.9GHz
Intel Core i5 7Y57 1.2GHz
AMD E-Series E2-9000e 1.5GHz
AMD A10-Series 9620P 2.5GHz
AMD E-Series 7110 1.8GHz
AMD A9-Series A9-9420 3GHz
Intel Xeon E3-1505M V6 3GHz
Intel Core i5 3.1GHz
Intel Core M 1.2GHz
Intel Atom X5-Z8350 1.44GHz
Intel Core M 1.1GHz
Intel Pentium Quad Core N3700 1.6GHz
Intel Core i7 6500U 2.50GHz
Intel Celeron Dual Core N3350 2GHz
Intel Celeron Quad Core N3160 1.6GHz
Intel Core i5 2.3GHz
Intel Core i7 7560U 2.4GHz
Intel Core i5 1.8GHz
AMD A10-Series A10-9620P 2.5GHz
AMD A10-Series 9600P 2.4GHz
AMD A12-Series 9720P 2.7GHz
AMD A6-Series A6-9220 2.5GHz
Intel Celeron Dual Core N3060 1.60GHz
AMD A4-Series 7210 2.2GHz
AMD FX 8800P 2.1GHz
Intel Core M m7-6Y75 1.2GHz
Intel Core i7 6560U 2.2GHz
Intel Pentium Dual Core 4405U 2.1GHz
Intel Core i7 7500U 2.5GHz
AMD A12-Series 9700P 2.5GHz
AMD A6-Series 9220 2.9GHz
Intel Celeron Quad Core N3710 1.6GHz
Intel Core M M7-6Y75 1.2GHz
Intel Core i7 6920HQ 2.9GHz
Intel Core i5 7200U 2.50GHz
Intel Core i5 7200U 2.70GHz
Intel Core i5 6260U 1.8GHz
Intel Core i5 7200U 2.7GHz
Intel Core i5 2.9GHz
Intel Pentium Dual Core 4405Y 1.5GHz
Intel Core M 6Y54 1.1GHz
AMD E-Series 9000e 1.5GHz
Samsung Cortex A72&A53 2.0GHz
AMD E-Series 9000 2.2GHz
Intel Core i7 2.7GHz
Intel Pentium Dual Core N4200 1.1GHz
Intel Atom Z8350 1.92GHz
Intel Xeon E3-1535M v5 2.9GHz
AMD E-Series 6110 1.5GHz
AMD E-Series E2-6110 1.5GHz
AMD Ryzen 1600 3.2GHz
Intel Atom x5-Z8300 1.44GHz
Intel Core i5 2.0GHz
Intel Core i7 7700HQ 2.7GHz
Intel Core M m3-7Y30 2.2GHz
Intel Core i5 1.3GHz
Intel Core i7 2.9GHz
AMD FX 9830P 3GHz
Intel Celeron Dual Core N3350 2.0GHz
Intel Core M m3 1.2GHz
Intel Core i5 6440HQ 2.6GHz
Intel Core M 7Y30 1.0GHz
Intel Core i5 7500U 2.7GHz
Intel Core i7 2.8GHz
Intel Core i3 6100U 2.1GHz
AMD E-Series E2-9000 2.2GHz
Intel Core i7 2.2GHz
Intel Core M M3-6Y30 0.9GHz
AMD A9-Series 9420 2.9GHz
Intel Core i3 6006U 2.2GHz
AMD A6-Series 7310 2GHz
Intel Xeon E3-1535M v6 3.1GHz

Plotting a graph of ROM vs laptops available

```
#I want to find out the different storage configuration and their count in the df
plt.figure(figsize = (10,5))
df['Memory'].value_counts().plot(kind = 'bar')
```



STEP-6: GETTING INTO CONCLUSIONS FROM THE DATA

As conclusion of our EDA following data was observed:-

1. Most and least common price of laptop are nearly - 500 and 4000 euros
2. Most and least no of models in the company are - Dell and Huawei
3. Costliest and cheapest laptop belongs to - Razer and Vero
4. Most and least common type of laptop is - Notebook and Netbook
5. Costliest and cheapest type laptop is - Workstation and Netbook
6. Most and least common laptop screen size is - 15.6 and 11.8
7. Most and least common ram - 8GB and 16GB
8. Most and the least common CPU model is- intel core i5 7200U 2.5GHZ and intel xeon E3-1535M V6 3.1GHZ
9. Most and least common rom - 256GB SSD and 256GB SSD + 1.0TB hybrid

LIBRARIES USED

We have used 4 libraries to do this project, they are:

Pandas

NumPy

Seaborn

Matplotlib

PANDAS

- Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively.
- It provides various data structures and operations for manipulating numerical data and time series.
- Pandas is fast and it has high performance & productivity for users.

It has many advantages like:

- Data from different file objects can be loaded.
- Easy handling of missing data.
- Columns can be inserted and deleted from DataFrame and higher dimensional objects
- Data set merging and joining.

- Flexible reshaping and pivoting of data sets.

NUMPY

- NumPy is a general-purpose array-processing package.
- It provides a high-performance multidimensional array object, and tools for working with these arrays.
- It is the fundamental package for scientific computing with Python.
- It is open-source software.

It contains various features like:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions

- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

SEABORN

- Seaborn is a visualization library for statistical graphics plotting in Python.
- It provides beautiful default styles and color palettes to make statistical plots more attractive.
- It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.
- Seaborn aims to make visualization the central part of exploring and understanding data.

- It provides dataset-oriented APIs, so that we can switch between different visual representations for same variables for better understanding of dataset.

MATPLOTLIB

- Matplotlib is a visualization library in Python for 2D plots of arrays.
- Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.
- One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals.
- Matplotlib consists of several plots like line, bar, scatter, histogram etc. Plots helps to understand trends, patterns, and

to make correlations. They’re typically instruments for reasoning about quantitative information.