Name: Soham Belurgikar

Roll No.: 2019130006

Course: DA (Data Analytics)

Assignment No.: 1

Part: 1

Name of the Assignment: Exploratory Data Analysis

Problem Statement:

The smartphone market in 2022 is filled with variety of phones catering to every person's needs. You can buy phones from brands like Samsung, Apple, Xiaomi, buy a phone which costs as low as Rs. 1000 or as high as Rs. 179900, buy phones with colours like Black, Blue, Rose Gold etc.

But which brand has sold the most phones? Which brand offers phones in all price ranges? Does the overall rating of a phone increase with its price? Which colour is the most in-demand?

This EDA aims to answer these questions with the help of statistics as well as plots.

Implementation:

Dataset link

Colab link

The dataset:

The chosen dataset consists of 2647 samples with 8 attributes, namely:

- Brand - Name of the Mobile Manufacturer
- Model - Model name / number of the Mobile Phone
- Colour - Colour of the model. Missing or Null values indicate no specified colour of the model offered on the ecommerce website.
- Memory - RAM of the model (4GB, 6GB, 8GB, etc.)
- Storage - ROM of the model (32GB, 64GB, 128GB, 256GB, etc.)
- Rating - Rating of the model based on reviews (out of 5). Missing or Null values indicate there are no ratings present for the model.
- Selling Price- Selling Price/Discounted Price of the model in INR when this data was scraped. Ideally price indicates the discounted price of the model

- Original Price- Actual price of the model in INR. Missing values or null values would indicate that the product is being sold at the actual price available in the 'Price' column.

Importing the required libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the data into the dataframe:

```
df_phones = pd.read_csv("/content/drive/MyDrive/Flipkart_mobile_brands_scraped_data.csv")
df_phones
```

	Brand	Model	Color	Memory	Storage	Rating	Selling Price	Original Price
0	OPPO	A53	Moonlight Black	4 GB	64 GB	4.5	11990.0	15990.0
1	OPPO	A53	Mint Cream	4 GB	64 GB	4.5	11990.0	15990.0
2	OPPO	A53	Moonlight Black	6 GB	128 GB	4.3	13990.0	17990.0
3	OPPO	A53	Mint Cream	6 GB	128 GB	4.3	13990.0	17990.0
4	OPPO	A53	Electric Black	4 GB	64 GB	4.5	11990.0	15990.0
...	...	...	...	...	...	...	...	...
2642	Xiaomi	Redmi Y3	Bold Red	4 GB	64 GB	4.3	12999.0	13999.0
2643	Xiaomi	Redmi Y3	Elegant Blue	3 GB	32 GB	4.3	9450.0	NaN
2644	Xiaomi	Redmi Y3	Elegant Blue	4 GB	64 GB	4.2	12999.0	NaN
2645	Xiaomi	Redmi Y3	Prime Black	3 GB	32 GB	4.2	9950.0	NaN
2646	Xiaomi	Redmi Y3	Prime Black	4 GB	64 GB	4.3	12499.0	13999.0

2647 rows × 8 columns

Adding the Name column:

Name of the phone = Name of Brand + Name of Model

```
df_phones["Name"] = df_phones["Brand"].astype(str) +" "+df_phones["Model"].astype(str)
df_phones
```

	Brand	Model	Color	Memory	Storage	Rating	Selling Price	Original Price	Name
0	OPPO	A53	Moonlight Black	4 GB	64 GB	4.5	11990.0	15990.0	OPPO A53
1	OPPO	A53	Mint Cream	4 GB	64 GB	4.5	11990.0	15990.0	OPPO A53
2	OPPO	A53	Moonlight Black	6 GB	128 GB	4.3	13990.0	17990.0	OPPO A53
3	OPPO	A53	Mint Cream	6 GB	128 GB	4.3	13990.0	17990.0	OPPO A53
4	OPPO	A53	Electric Black	4 GB	64 GB	4.5	11990.0	15990.0	OPPO A53
...	...	...	...	...	...	...	...	...	...
2642	Xiaomi	Redmi Y3	Bold Red	4 GB	64 GB	4.3	12999.0	13999.0	Xiaomi Redmi Y3
2643	Xiaomi	Redmi Y3	Elegant Blue	3 GB	32 GB	4.3	9450.0	NaN	Xiaomi Redmi Y3
2644	Xiaomi	Redmi Y3	Elegant Blue	4 GB	64 GB	4.2	12999.0	NaN	Xiaomi Redmi Y3
2645	Xiaomi	Redmi Y3	Prime Black	3 GB	32 GB	4.2	9950.0	NaN	Xiaomi Redmi Y3
2646	Xiaomi	Redmi Y3	Prime Black	4 GB	64 GB	4.3	12499.0	13999.0	Xiaomi Redmi Y3

2647 rows × 9 columns

```
df_phones.shape
```

Using .shape() we can get information about the number of rows and columns of the dataset:

```
(2647, 9)
```

So, the dataset contains 2647 rows (samples) and 9 columns (features).

Removing duplicate rows:

```
duplicate_rows_df = df_phones[df_phones.duplicated()]
print("number of duplicate rows: ", duplicate_rows_df.shape)
```

This gives us the number of rows which have the same values for every column:

```
number of duplicate rows:  (107, 9)
```

So, the dataset contained 107 rows which were duplicates.

```
df_phones.count()
```

You can also check the number of rows that each column contains using the .count() method:

```
Brand              2647
Model              2645
Color              2505
Memory             2605
Storage            2568
Rating             2647
Selling Price      2644
Original Price      969
Name               2647
dtype: int64
```

You can delete the duplicate rows using just a simple method, i.e., .drop_duplicates():

```
[123] df_phones = df_phones.drop_duplicates()
      df_phones
```

	Brand	Model	Color	Memory	Storage	Rating	Selling Price	Original Price	Name
0	OPPO	A53	Moonlight Black	4 GB	64 GB	4.5	11990.0	15990.0	OPPO A53
1	OPPO	A53	Mint Cream	4 GB	64 GB	4.5	11990.0	15990.0	OPPO A53
2	OPPO	A53	Moonlight Black	6 GB	128 GB	4.3	13990.0	17990.0	OPPO A53
3	OPPO	A53	Mint Cream	6 GB	128 GB	4.3	13990.0	17990.0	OPPO A53
4	OPPO	A53	Electric Black	4 GB	64 GB	4.5	11990.0	15990.0	OPPO A53
...	...	...	...	...	...	...	...	...	...
2642	Xiaomi	Redmi Y3	Bold Red	4 GB	64 GB	4.3	12999.0	13999.0	Xiaomi Redmi Y3
2643	Xiaomi	Redmi Y3	Elegant Blue	3 GB	32 GB	4.3	9450.0	NaN	Xiaomi Redmi Y3
2644	Xiaomi	Redmi Y3	Elegant Blue	4 GB	64 GB	4.2	12999.0	NaN	Xiaomi Redmi Y3
2645	Xiaomi	Redmi Y3	Prime Black	3 GB	32 GB	4.2	9950.0	NaN	Xiaomi Redmi Y3
2646	Xiaomi	Redmi Y3	Prime Black	4 GB	64 GB	4.3	12499.0	13999.0	Xiaomi Redmi Y3

2540 rows × 9 columns

```
df_phones.count()
```

```
Brand              2540
Model              2538
Color              2407
Memory             2501
Storage            2463
Rating             2540
Selling Price      2537
Original Price      934
Name               2540
dtype: int64
```

Removing null / missing values:

```
print(df_phones.isnull().sum())
```

The .isnull().sum() command will return the number of values which are missing for every column:

```
Brand                 0
Model                 2
Color               133
Memory               39
Storage              77
Rating                0
Selling Price         3
Original Price     1606
Name                  0
dtype: int64
```

We will drop lines with model unknown or missing memory information or missing storage information. Put missing value of colour to "Base". Drop lines with missing both prices else fill one with the other.

```
df_phones = df_phones.dropna(subset=["Model", "Memory","Storage"])
df_phones["Selling Price"] = df_phones["Selling Price"].fillna(df_phone
s["Original Price"])
df_phones["Original Price"] = df_phones["Original Price"].fillna(df_pho
nes["Selling Price"])
df_phones= df_phones.dropna(subset=["Original Price","Selling Price"])
df_phones["Color"] = df_phones["Color"].fillna("Base")
```

```
print(df_phones.isnull().sum())
```

```
Brand             0
Model             0
Color             0
Memory            0
Storage           0
Rating            0
Selling Price     0
Original Price    0
Name              0
dtype: int64
```

Now our dataset is free of null values.

Statistics:

```
[148] df_phones.describe()
```

	Rating	Selling Price	Original Price
count	2436.000000	2436.000000	2436.000000
mean	4.025041	25524.864943	27507.622742
std	0.928679	28356.236966	30166.558414
min	0.000000	1000.000000	1000.000000
25%	4.000000	9499.000000	9999.000000
50%	4.300000	14999.000000	15999.000000
75%	4.400000	27990.000000	30123.750000
max	5.000000	179900.000000	189999.000000

The .describe() method is very useful for calculating mean, standard deviation, range and percentiles of the data.

We can use .mode() for calculating mode of a particular column:

```
df_phones['Selling Price'].mode()
```

```
0    9999.0
dtype: float64
```

For calculating the standard error, we can use .sem():

```
df_phones['Selling Price'].sem()
```

```
574.5263541029967
```

Variance is calculated using .var():

```
df_phones['Selling Price'].var()
```

```
804076174.8774368
```

Lastly, to calculate the coefficient of variation, we can divide the standard deviation by the mean:

```
cv_sell_price = df_phones['Selling Price'].std() / df_phones['Selling P
rice'].mean()
cv_sell_price
```

```
1.110926033494914
```

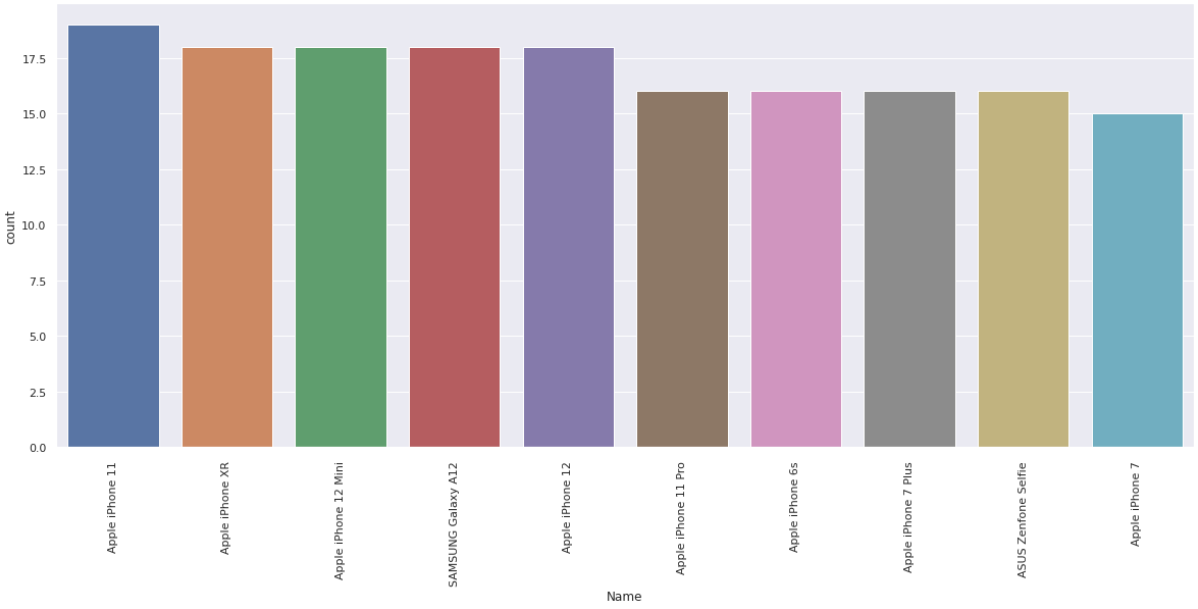
Plots:

The models of a specific smartphone may differ in memory, storage or colour. Such variations can be found using .groupby:

```
group_name = df_phones["Name"].value_counts(ascending=False)[0:10].reset_index()
group_name.columns = ['Name', 'No. of Variants']
group_name
```

	Name	No. of Variants
0	Apple iPhone 11	19
1	Apple iPhone XR	18
2	Apple iPhone 12 Mini	18
3	SAMSUNG Galaxy A12	18
4	Apple iPhone 12	18
5	Apple iPhone 11 Pro	16
6	Apple iPhone 6s	16
7	Apple iPhone 7 Plus	16
8	ASUS Zenfone Selfie	16
9	Apple iPhone 7	15

```
fig, ax = plt.subplots(figsize=(20,8))
ax=sns.countplot(x="Name", data=df_phones, order=df_phones['Name'].valu
e_counts(ascending = False)[:10].index)
plt.xticks(rotation = 90)
```

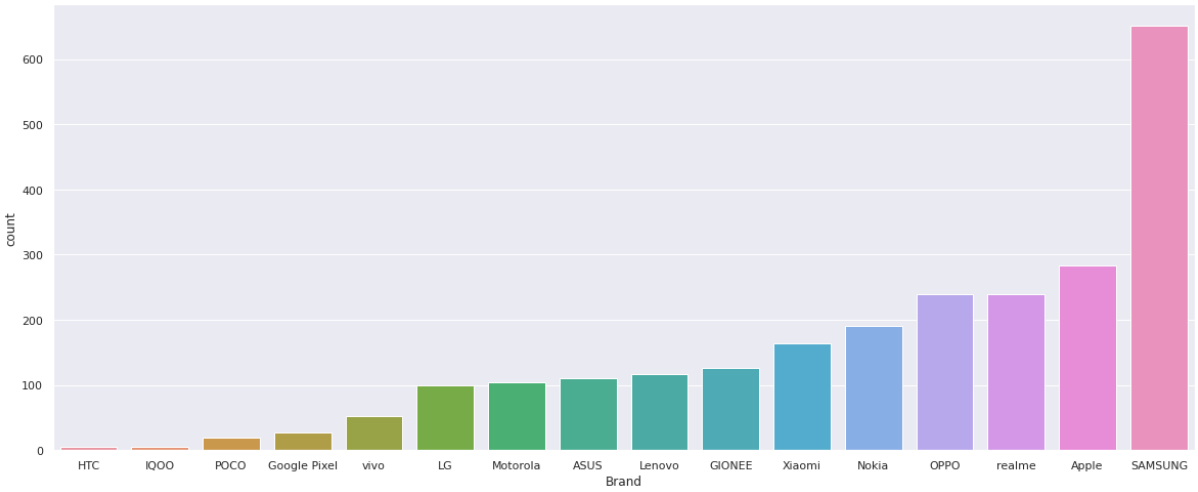


Calculating the no. of smartphones sold by brand:

```
[130] group_brand = df_phones["Brand"].value_counts(ascending=False)[0:10].reset_index()
      group_brand.columns = ['Brand', 'No. of phones']
      group_brand
```

	Brand	No. of phones
0	SAMSUNG	651
1	Apple	283
2	OPPO	240
3	realme	240
4	Nokia	191
5	Xiaomi	164
6	GIONEE	127
7	Lenovo	117
8	ASUS	111
9	Motorola	104

```
fig, ax = plt.subplots(figsize=(20,8))
ax=sns.countplot(x="Brand", data=df_phones, order=df_phones['Brand'].value_counts(ascending = True).index)
```

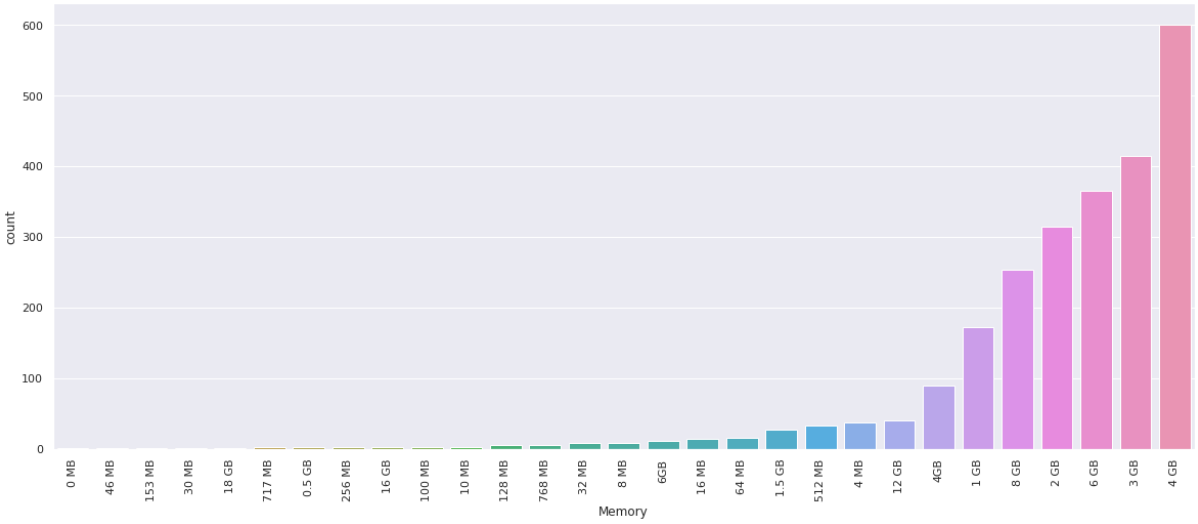


Calculating the no. of smartphones sold by memory:

```
[132] group_memory = df_phones["Memory"].value_counts(ascending=False)[0:10].reset_index()
      group_memory.columns = ['Memory', 'No. of Phones']
      group_memory
```

	Memory	No. of Phones
0	4 GB	601
1	3 GB	414
2	6 GB	365
3	2 GB	314
4	8 GB	254
5	1 GB	173
6	4GB	90
7	12 GB	40
8	4 MB	38
9	512 MB	33

```
fig, ax = plt.subplots(figsize=(20,8))
ax=sns.countplot(x="Memory", data=df_phones, order=df_phones['Memory'].
value_counts(ascending = True).index)
plt.xticks(rotation = 90)
```

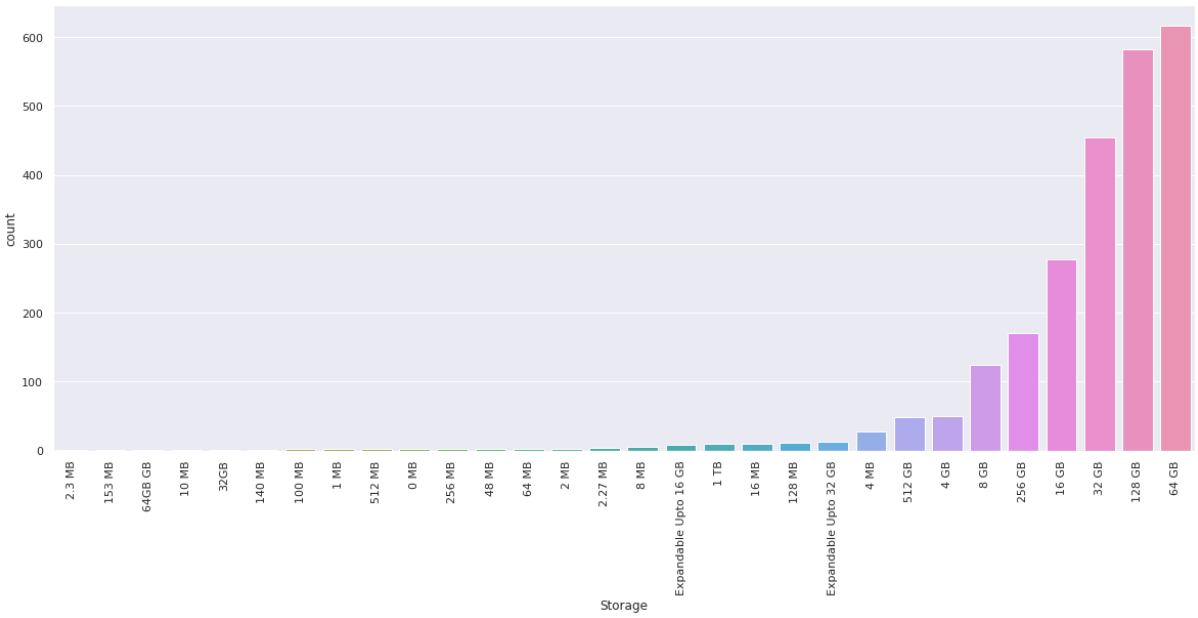


Calculating the no. of smartphones sold by storage:

```
group_storage = df_phones["Storage"].value_counts(ascending=False)[0:10].reset_index()
group_storage.columns = ['Storage', 'No. of Phones']
group_storage
```

	Storage	No. of Phones
0	64 GB	616
1	128 GB	582
2	32 GB	455
3	16 GB	277
4	256 GB	171
5	8 GB	124
6	4 GB	50
7	512 GB	48
8	4 MB	28
9	Expandable Upto 32 GB	12

```
fig, ax = plt.subplots(figsize=(20,8))
ax=sns.countplot(x="Storage", data=df_phones, order=df_phones['Storage'
].value_counts(ascending = True).index)
plt.xticks(rotation = 90)
```

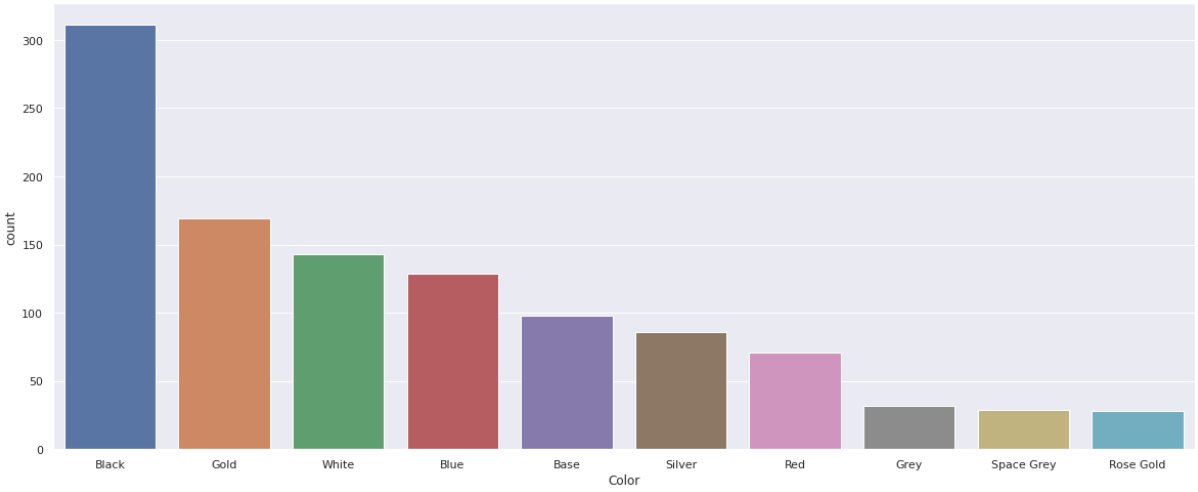


Calculating the no. of smartphones sold by colour:

```
group_color = df_phones["Color"].value_counts(ascending=False)[0:10].reset_index()
group_color.columns = ['Color', 'No. of Phones']
group_color
```

	Color	No. of Phones
0	Black	311
1	Gold	169
2	White	143
3	Blue	129
4	Base	98
5	Silver	86
6	Red	71
7	Grey	32
8	Space Grey	29
9	Rose Gold	28

```
fig, ax = plt.subplots(figsize=(20,8))
ax=sns.countplot(x="Color", data=df_phones, order=df_phones['Color'].value_counts(ascending = False)[:10].index)
```

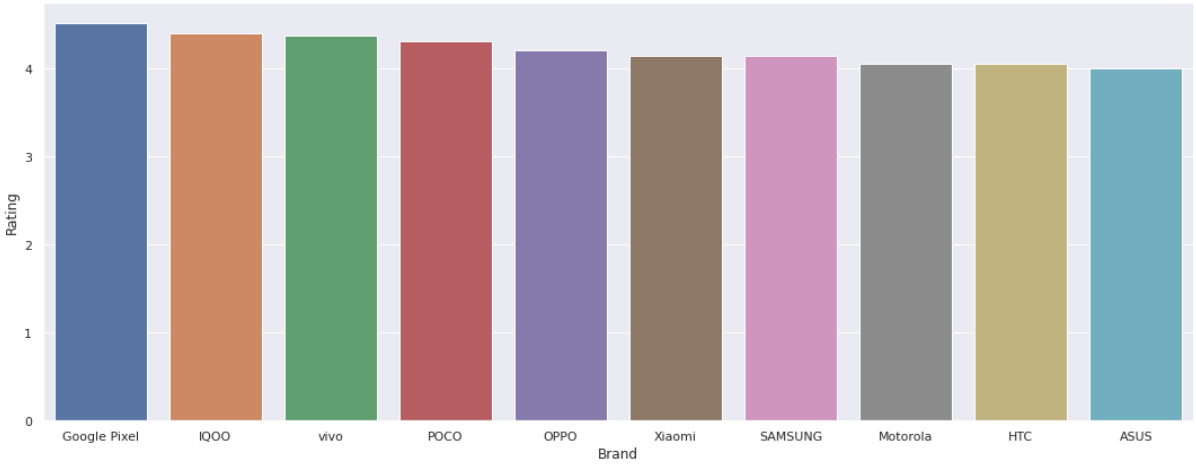


Top 10 brands sorted by average rating:

```
[138] phone_rating = df_phones.groupby('Brand')[['Rating']].mean().sort_values(ascending=False,by='Rating')[:10].reset_index()
      phone_rating
```

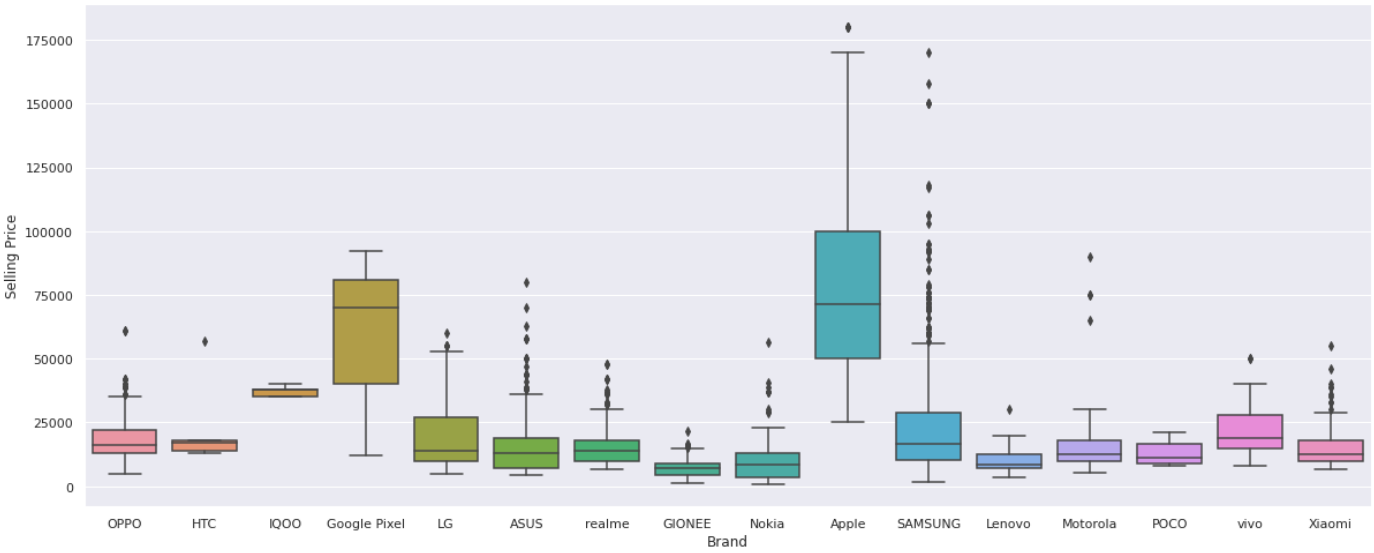
	Brand	Rating
0	Google Pixel	4.514815
1	IQOO	4.400000
2	vivo	4.373077
3	POCO	4.310000
4	OPPO	4.205000
5	Xiaomi	4.150610
6	SAMSUNG	4.145469
7	Motorola	4.062500
8	HTC	4.060000
9	ASUS	4.008108

```
sns.catplot(x="Brand", y="Rating", kind="bar", data=phone_rating, heigh
t=6, aspect=2.5)
```



Finding outliers for selling price of every brand:

```
sns.set(rc={'figure.figsize':(20,8)})
sns.boxplot(x="Brand", y="Selling Price", data=df_phones)
```

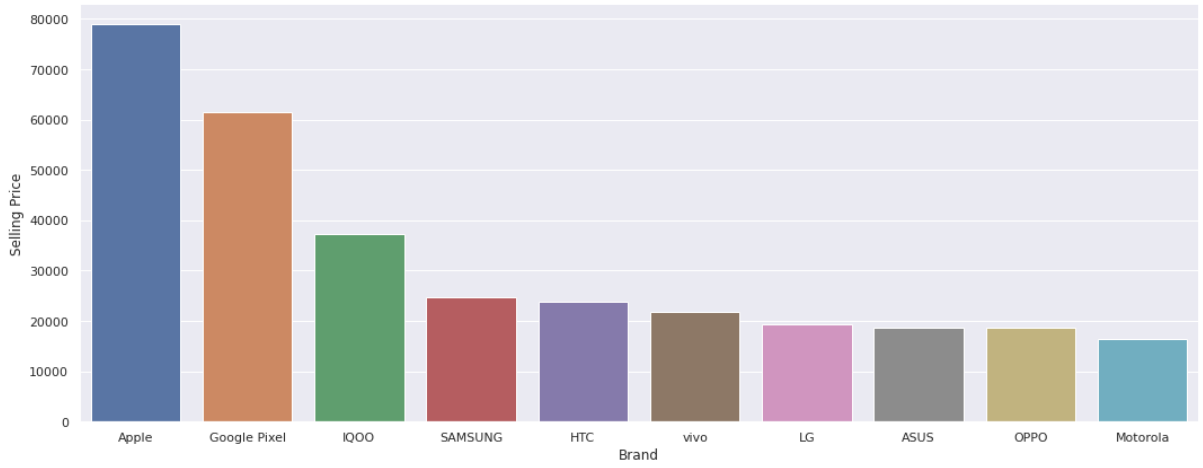


Top 10 brands sorted by average selling price:

```
sell_price = df_phones.groupby('Brand')[['Selling Price']].mean().sort_values(ascending=False,by='Selling Price')[:10].reset_index()
sell_price
```

	Brand	Selling Price
0	Apple	78910.028269
1	Google Pixel	61383.851852
2	IQOO	37190.000000
3	SAMSUNG	24733.596006
4	HTC	23797.200000
5	vivo	21792.884615
6	LG	19323.181818
7	ASUS	18756.684685
8	OPPO	18606.625000
9	Motorola	16370.528846

```
sns.catplot(x="Brand", y="Selling Price", kind="bar", data=sell_price, height=6, aspect=2.5)
```



Heat Map:

We can find the dependent variables using heat map.

```
plt.figure(figsize=(10, 5))
c = df_phones.corr()
sns.heatmap(c,cmap="BrBG",annot=True)
c
```

It first finds the correlation between any two variables and then plots it along with a colour theme using the seaborn library:

	Rating	Selling Price	Original Price
Rating	1.000000	-0.172306	-0.146712
Selling Price	-0.172306	1.000000	0.982184
Original Price	-0.146712	0.982184	1.000000

Relation between Selling Price and Rating:

```
fig, ax = plt.subplots(figsize=(20, 8))
ax.scatter(df_phones['Rating'], df_phones['Selling Price'])
ax.set_xlabel('Rating')
ax.set_ylabel('Selling Price')
plt.show()
```

No. of smartphones by price range:

```
sns.displot(df_phones, x='Selling Price',bins=[5000,10000,15000,20000,2
5000,30000,35000,40000,50000,60000,80000], aspect=2)
plt.xticks(rotation = 90)
```

Conclusion:

The mean rating of all smartphones was 4.02 whereas the mean selling price and original price were Rs. 25524.86 and Rs. 27507.62 respectively.

We can see that no. of variants for Apple iPhones are higher compared to other brands, meaning that Apple offers more variety in terms of colour, memory and storage.

Samsung dominates the smartphone market by selling the most number of smartphones, followed by Apple, OPPO and Realme.

In terms of specifications of mobile phones, 4GB (RAM) and 64 GB (Storage) are the most common types.

Black is the most in-demand colour, followed by Gold and White.

Google Pixel is the highest rated smartphone brand with over 4.5 average rating. Xiaomi and Samsung have the 5th and 6th highest average rating respectively, while Apple doesn't even make it to the top 10.

Selling Price of Samsung contains a relatively higher number of outliers, which can suggest that while Samsung sells majority of its phones in the lower and mid-budget price bracket, it also contains a substantial amount of phones in the high-budget price bracket.

Apple is quite rightly famous for being the most expensive brand, with its average selling price of almost Rs. 79000.

The scatterplot suggests that there might be a positive direct relationship between rating and Selling Price, since Selling Price increases as Rating increases.

The displot allows us to look at each price range and count the no. of phones in it. The low-budget price range is clearly the winner, suggesting that the population prefers a cheaper smartphone even if it sacrifices on some of the specs.

References: Dataset link