Exploratory Data Analysis on Smartphone Datasets 📱📱

Introduction:

Smartphones have become an integral part of our daily lives, serving a wide range of functions from communication and entertainment to productivity and beyond. In this exploratory data analysis (EDA) I'm using Real World Smartphone's Dataset from kaggle, Here we dive into a dataset that captures a snapshot of the real-world smartphone landscape. Our dataset is a collection of information about various smartphone models, features, and user-related metrics. Through this EDA, we aim to gain a deeper understanding of this dynamic industry and the factors that influence consumer choices.

Data Source:

The dataset used for this analysis was obtained from kaggle.

Data Description:

The dataset includes a diverse set of features and information about smartphone, such as brand, model, screen size, camera specifications, battery life, user ratings, and more. this data also contain some missing value.

Objective:

Our primary objective in this EDA is to explore the dataset to uncover insights and trends related to smartphone attributes and user preferences. We seek to answer questions such as:

What are the most popular smartphone brands and models?
Is there a correlation between smartphone prices and user ratings?
Are there any distinctive features that influence consumer choices?
How do user reviews and ratings vary across different brands and models?
Smartphone Handsets has 5g Compatibility or not
Average price versus Smartphone specifications like Processor, Battery, rear camera.
Brand vs Number of models
Brand vs Avg. Price
Brand vs Avg. Rating
Processor vs Avg. Price
Battery vs Avg. Price
Battery vs Brand
Fast Charging Available
Number of Rear cameras vs Avg. Price
Rear camera Resolution vs Avg. Price
Operating System vs Brand

Methods:

To accomplish our objectives, we will employ various data analysis methods, including data visualization, statistical analysis, and potentially, machine learning techniques. Our aim is to present clear and informative visualizations and summaries that can provide valuable insights.

Audience:

This EDA report is intended for a broad audience, including technology enthusiasts, industry professionals, and data analysts interested in the smartphone market. Our goal is to provide information that is both accessible and insightful.

Expected Outcomes:

Through this EDA, we anticipate uncovering interesting patterns, relationships, and trends within the smartphone dataset. These insights can be valuable for consumers, manufacturers, and the broader technology industry, helping them better understand the evolving landscape of smartphones.

Challenges and Limitations:

It's important to acknowledge potential challenges, such as missing or incomplete data, data biases, or the rapidly changing nature of the smartphone market. These limitations may affect the comprehensiveness of our analysis.

Exploratory Data Analysis on Smartphone Datasets 📱📱

Introduction:

Smartphones have become an integral part of our daily lives, serving a wide range of functions from communication and entertainment to productivity and beyond. In this exploratory data analysis (EDA) I'm using Real World Smartphone's Dataset from kaggle, Here we dive into a dataset that captures a snapshot of the real-world smartphone landscape. Our dataset is a collection of information about various smartphone models, features, and user-related metrics. Through this EDA, we aim to gain a deeper understanding of this dynamic industry and the factors that influence consumer choices.

Data Source:

The dataset used for this analysis was obtained from kaggle.

Data Description:

The dataset includes a diverse set of features and information about smartphone, such as brand, model, screen size, camera specifications, battery life, user ratings, and more. this data also contain some missing value.

Objective:

Our primary objective in this EDA is to explore the dataset to uncover insights and trends related to smartphone attributes and user preferences. We seek to answer questions such as:

What are the most popular smartphone brands and models?
Is there a correlation between smartphone prices and user ratings?
Are there any distinctive features that influence consumer choices?
How do user reviews and ratings vary across different brands and models?
Smartphone Handsets has 5g Compatibility or not
Average price versus Smartphone specifications like Processor, Battery, rear camera.
Brand vs Number of models
Brand vs Avg. Price
Brand vs Avg. Rating

Processor vs Avg. Price
Battery vs Avg. Price
Battery vs Brand
Fast Charging Available
Number of Rear cameras vs Avg. Price
Rear camera Resolution vs Avg. Price
Operating System vs Brand

Methods:

To accomplish our objectives, we will employ various data analysis methods, including data visualization, statistical analysis, and potentially, machine learning techniques. Our aim is to present clear and informative visualizations and summaries that can provide valuable insights.

Audience:

This EDA report is intended for a broad audience, including technology enthusiasts, industry professionals, and data analysts interested in the smartphone market. Our goal is to provide information that is both accessible and insightful.

Expected Outcomes:

Through this EDA, we anticipate uncovering interesting patterns, relationships, and trends within the smartphone dataset. These insights can be valuable for consumers, manufacturers, and the broader technology industry, helping them better understand the evolving landscape of smartphones.

Challenges and Limitations:

It's important to acknowledge potential challenges, such as missing or incomplete data, data biases, or the rapidly changing nature of the smartphone market. These limitations may affect the comprehensiveness of our analysis.

Ethical Considerations:

We are committed to ethical data use and respect for user privacy. Any personal or sensitive data within the dataset will be handled with the utmost care, and results will be presented in an anonymized and aggregated manner to ensure data privacy and security.

Importing library

```
# basic
import numpy as np
import pandas as pd

# Visualization
import seaborn as sns
sns.set_style('darkgrid')
import matplotlib.pyplot as plt

# Warnings
```

```
import warnings
warnings.filterwarnings('ignore')
```

Downloading the Dataset

Here we are using opendatasets module to download datasets directly from your url in your local :-
opendatasets is a Python library for downloading datasets from online sources like Kaggle and Google Drive using a simple Python command within a Jupyter notebook or Python script.

```
!pip install jovian opendatasets --upgrade --quiet
!pip install jovian kaggle --upgrade --quiet
```

```
dataset_url = 'https://www.kaggle.com/datasets/abhijitdahatonde/real-world-smartphones-
```

```
import opendatasets
opendatasets.download(dataset_url)
# Due to some issue with i will skiping this way i will directly load data from locally
```

```
Please provide your Kaggle credentials to download this dataset. Learn more:
http://bit.ly/kaggle-creds
Your Kaggle username: abhishekyv
Your Kaggle Key: ·····
Downloading real-world-smartphones-dataset.zip to ./real-world-smartphones-dataset
100%|████████████████| 16.8k/16.8k [00:00<00:00, 1.55MB/s]
```

```
# import pandas as pd
# df = pd.read_csv(r"C:\Users\admin\Downloads\smartphones.csv")
# Path to the Local Data
data_dir = './real-world-smartphones-dataset'
```

```
import os
os.listdir(data_dir)
```

```
['smartphones.csv']
```

```
!pip install jovian --upgrade -q
```

```
import jovian
```

```
jovian.commit(project='zerotopandas-Real world smartphone data analysis')
```

```
[jovian] Updating notebook "ay29020/zerotopandas-real-world-smartphone-data-analysis"
```

```
on https://jovian.com
[jovian] Committed successfully! https://jovian.com/ay29020/zerotopandas-real-world-smartphone-data-analysis
```

'https://jovian.com/ay29020/zerotopandas-real-world-smartphone-data-analysis'

Data Preparation and Cleaning

We have a `Real World Smartphone's Dataset` that contains information about various smartphone models, features, and user-related metrics. **

> So We are going to do things one by one

> - Load the dataset into a data frame using Pandas
> - Explore the number of rows & columns, ranges of values etc.
> - Handle missing, incorrect and invalid data
> - Perform any additional steps (parsing dates, creating additional columns, merging multiple dataset etc.)

* Load the Dataset

```
import pandas as pd
import numpy as np
df = pd.read_csv('./real-world-smartphones-dataset/smartphones.csv')
```

* Explore the Dataset

```
df.head(5)
```

	brand_name	model	price	avg_rating	5G_or_not	processor_brand	num_cores	processor_speed	battery_capacit
0	apple	Apple iPhone 11	38999	7.3	0	bionic	6.0	2.65	3110.
1	apple	Apple iPhone 11 (128GB)	46999	7.5	0	bionic	6.0	2.65	3110.
2	apple	Apple iPhone 11 Pro Max	109900	7.7	0	bionic	6.0	2.65	3500.
3	apple	Apple iPhone 12	51999	7.4	1	bionic	6.0	3.10	Nal
4	apple	Apple iPhone 12 (128GB)	55999	7.5	1	bionic	6.0	3.10	Nal

5 rows × 22 columns

```
df.tail(5)
```

	brand_name	model	price	avg_rating	5G_or_not	processor_brand	num_cores	processor_speed	battery_capaci
975	xiaomi	Xiaomi Redmi Note 9 Pro	13999	7.5	0	snapdragon	8.0	2.3	5020
976	xiaomi	Xiaomi Redmi Note 9 Pro (4GB RAM + 128GB)	14439	7.7	0	snapdragon	8.0	2.3	5020
977	xiaomi	Xiaomi Redmi Note 9 Pro Max	16490	8.0	0	snapdragon	8.0	2.3	5020
978	zte	ZTE Axon 30S	19999	8.2	1	snapdragon	8.0	3.2	4200
979	zte	ZTE Axon 40 Ultra 5G	61990	8.9	1	snapdragon	8.0	3.0	5000

5 rows × 22 columns

```
df.index
```

```
RangeIndex(start=0, stop=980, step=1)
```

```
# columns
df.columns
```

```
Index(['brand_name', 'model', 'price', 'avg_rating', '5G_or_not',
       'processor_brand', 'num_cores', 'processor_speed', 'battery_capacity',
       'fast_charging_available', 'fast_charging', 'ram_capacity',
       'internal_memory', 'screen_size', 'refresh_rate', 'num_rear_cameras',
       'os', 'primary_camera_rear', 'primary_camera_front',
       'extended_memory_available', 'resolution_height', 'resolution_width'],
      dtype='object')
```

```
df.nunique()
```

```
brand_name                 46
model                     980
price                     379
avg_rating                 30
5G_or_not                   2
processor_brand            13
```

```
num_cores                      3
processor_speed               35
battery_capacity              89
fast_charging_available        2
fast_charging                 32
ram_capacity                   9
internal_memory                8
screen_size                   79
refresh_rate                   6
num_rear_cameras               4
os                             3
primary_camera_rear           18
primary_camera_front          19
extended_memory_available      2
resolution_height             65
resolution_width              40
dtype: int64
```

```
print('\n'.join([colmn for colmn in df.columns]), sep='\n')
```

```
brand_name
model
price
avg_rating
5G_or_not
processor_brand
num_cores
processor_speed
battery_capacity
fast_charging_available
fast_charging
ram_capacity
internal_memory
screen_size
refresh_rate
num_rear_cameras
os
primary_camera_rear
primary_camera_front
extended_memory_available
resolution_height
resolution_width
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 980 entries, 0 to 979
Data columns (total 22 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   brand_name                 980 non-null    object 
 1   model                      980 non-null    object 
 2   price                      980 non-null    int64  
 3   avg_rating                 879 non-null    float64
 4   5G_or_not                  980 non-null    int64  
 5   processor_brand            960 non-null    object 
 6   num_cores                  974 non-null    float64
 7   processor_speed            938 non-null    float64
 8   battery_capacity           969 non-null    float64
 9   fast_charging_available    980 non-null    int64  
 10  fast_charging              769 non-null    float64
 11  ram_capacity               980 non-null    int64  
 12  internal_memory            980 non-null    int64  
 13  screen_size                980 non-null    float64
 14  refresh_rate               980 non-null    int64  
 15  num_rear_cameras           980 non-null    int64  
 16  os                         966 non-null    object 
 17  primary_camera_rear        980 non-null    float64
 18  primary_camera_front       975 non-null    float64
 19  extended_memory_available  980 non-null    int64  
 20  resolution_height          980 non-null    int64  
 21  resolution_width           980 non-null    int64  
dtypes: float64(8), int64(10), object(4)
memory usage: 168.6+ KB
```

```
print("This Dataframe contains {} row and column respectively".format(df.shape))
```

```
This Dataframe contains (980, 22) row and column respectively
```

* Handling missing, incorrect and invalid data

```
df.isna().sum()
```

```
brand_name                   0
model                        0
price                        0
avg_rating                 101
5G_or_not                    0
processor_brand             20
num_cores                    6
processor_speed             42
```

```
battery_capacity              11
fast_charging_available        0
fast_charging                211
ram_capacity                   0
internal_memory                0
screen_size                    0
refresh_rate                   0
num_rear_cameras               0
os                            14
primary_camera_rear            0
primary_camera_front           5
extended_memory_available      0
resolution_height              0
resolution_width               0
dtype: int64
```

```
# Calculate the median of the columns
avg_rating = df['avg_rating'].median()
num_cores = df['num_cores'].median()
battery_cap = df['battery_capacity'].median()
processor_speed = df['processor_speed'].median()


# Fill NaN values in the Missing columns with the median value
df['avg_rating'].fillna(avg_rating,inplace=True)
df['num_cores'].fillna(num_cores,inplace=True)
df['battery_capacity'].fillna(battery_cap,inplace=True)
df['processor_speed'].fillna(processor_speed,inplace=True)
```

```
df.isna().sum()
```

```
brand_name                     0
model                          0
price                          0
avg_rating                     0
5G_or_not                      0
processor_brand               20
num_cores                      0
processor_speed                0
battery_capacity               0
fast_charging_available        0
fast_charging                211
ram_capacity                   0
internal_memory                0
screen_size                    0
refresh_rate                   0
num_rear_cameras               0
os                            14
primary_camera_rear            0
```

```
primary_camera_front           5
extended_memory_available      0
resolution_height              0
resolution_width               0
dtype: int64
```

```
df
```

	brand_name	model	price	avg_rating	5G_or_not	processor_brand	num_cores	processor_speed	battery_capa
0	apple	Apple iPhone 11	38999	7.3	0	bionic	6.0	2.65	31
1	apple	Apple iPhone 11 (128GB)	46999	7.5	0	bionic	6.0	2.65	31
2	apple	Apple iPhone 11 Pro Max	109900	7.7	0	bionic	6.0	2.65	35(
3	apple	Apple iPhone 12	51999	7.4	1	bionic	6.0	3.10	50(
4	apple	Apple iPhone 12 (128GB)	55999	7.5	1	bionic	6.0	3.10	50(
...	...	...	...	...	...	...	...	...	
975	xiaomi	Xiaomi Redmi Note 9 Pro	13999	7.5	0	snapdragon	8.0	2.30	50
976	xiaomi	Xiaomi Redmi Note 9 Pro (4GB RAM + 128GB)	14439	7.7	0	snapdragon	8.0	2.30	50
977	xiaomi	Xiaomi Redmi Note 9 Pro Max	16490	8.0	0	snapdragon	8.0	2.30	50
978	zte	ZTE Axon 30S	19999	8.2	1	snapdragon	8.0	3.20	42(
979	zte	ZTE Axon 40 Ultra 5G	61990	8.9	1	snapdragon	8.0	3.00	50(

980 rows × 22 columns

```
df= df.dropna()
df.isna().sum()
```

```
brand_name                  0
model                       0
price                       0
avg_rating                  0
5G_or_not                   0
processor_brand             0
num_cores                   0
processor_speed             0
battery_capacity            0
fast_charging_available     0
fast_charging               0
ram_capacity                0
internal_memory             0
screen_size                 0
refresh_rate                0
num_rear_cameras            0
os                          0
primary_camera_rear         0
primary_camera_front        0
extended_memory_available   0
resolution_height           0
resolution_width            0
dtype: int64
```

```
df
```

	brand_name	model	price	avg_rating	5G_or_not	processor_brand	num_cores	processor_speed	battery_capa
2	apple	Apple iPhone 11 Pro Max	109900	7.7	0	bionic	6.0	2.65	350
38	apple	Apple iPhone 15 Pro Max	142990	7.9	1	bionic	8.0	2.30	435
42	apple	Apple iPhone SE 2020	39900	6.3	0	bionic	6.0	2.65	182
46	asus	Asus ROG Phone 5s 5G	39999	8.7	1	snapdragon	8.0	2.90	600
47	asus	Asus ROG Phone 6	71999	8.6	1	snapdragon	8.0	3.20	600
...	...	...	...	...	...	...	...	...	
975	xiaomi	Xiaomi Redmi Note 9 Pro	13999	7.5	0	snapdragon	8.0	2.30	502

	brand_name	model	price	avg_rating	5G_or_not	processor_brand	num_cores	processor_speed	battery_capa
976	xiaomi	Xiaomi Redmi Note 9 Pro (4GB RAM + 128GB)	14439	7.7	0	snapdragon	8.0	2.30	502
977	xiaomi	Xiaomi Redmi Note 9 Pro Max	16490	8.0	0	snapdragon	8.0	2.30	502
978	zte	ZTE Axon 30S	19999	8.2	1	snapdragon	8.0	3.20	420
979	zte	ZTE Axon 40 Ultra 5G	61990	8.9	1	snapdragon	8.0	3.00	500

750 rows × 22 columns

```
len(df)
```

```
750
```

* Some additional steps

```
df['My_index'] = range(1, len(df) + 1)
df.set_index('My_index')
```

My_index	brand_name	model	price	avg_rating	5G_or_not	processor_brand	num_cores	processor_speed	battery_
1	apple	Apple iPhone 11 Pro Max	109900	7.7	0	bionic	6.0	2.65	
2	apple	Apple iPhone 15 Pro Max	142990	7.9	1	bionic	8.0	2.30	
3	apple	Apple iPhone SE 2020	39900	6.3	0	bionic	6.0	2.65	
4	asus	Asus ROG Phone 5s 5G	39999	8.7	1	snapdragon	8.0	2.90	
5	asus	Asus ROG Phone 6	71999	8.6	1	snapdragon	8.0	3.20	
...	...	...	...	...	...	...	...	...	

My_index	brand_name	model	price	avg_rating	5G_or_not	processor_brand	num_cores	processor_speed	battery
746	xiaomi	Xiaomi Redmi Note 9 Pro	13999	7.5	0	snapdragon	8.0	2.30	
747	xiaomi	Xiaomi Redmi Note 9 Pro (4GB RAM + 128GB)	14439	7.7	0	snapdragon	8.0	2.30	
748	xiaomi	Xiaomi Redmi Note 9 Pro Max	16490	8.0	0	snapdragon	8.0	2.30	
749	zte	ZTE Axon 30S	19999	8.2	1	snapdragon	8.0	3.20	
750	zte	ZTE Axon 40 Ultra 5G	61990	8.9	1	snapdragon	8.0	3.00	

750 rows × 22 columns

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "ay29020/zerotopandas-real-world-smartphone-data-analysis"
on https://jovian.com
[jovian] Committed successfully! https://jovian.com/ay29020/zerotopandas-real-world-
smartphone-data-analysis
```

```
'https://jovian.com/ay29020/zerotopandas-real-world-smartphone-data-analysis'
```

Exploratory Analysis and Visualization

Now our dataset is ready for performing Exploratory Analysis

> Here we will :
>
> - Compute the mean, sum, range and other interesting statistics for numeric columns
> - Explore distributions of numeric columns using histograms etc.
> - Explore relationship between columns using scatter plots, bar charts etc.
> - Make a note of interesting insights from the exploratory analysis

Let's begin by importing `matplotlib.pyplot and seaborn .`

```
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (9, 5)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

```
df.dtypes
```

```
brand_name                  object
model                       object
price                        int64
avg_rating                 float64
5G_or_not                    int64
processor_brand             object
num_cores                  float64
processor_speed            float64
battery_capacity           float64
fast_charging_available      int64
fast_charging              float64
ram_capacity                 int64
internal_memory              int64
screen_size                float64
refresh_rate                 int64
num_rear_cameras             int64
os                          object
primary_camera_rear        float64
primary_camera_front       float64
extended_memory_available    int64
resolution_height            int64
resolution_width             int64
My_index                     int64
dtype: object
```

```
df.price.mean()
```

```
30529.662666666667
```

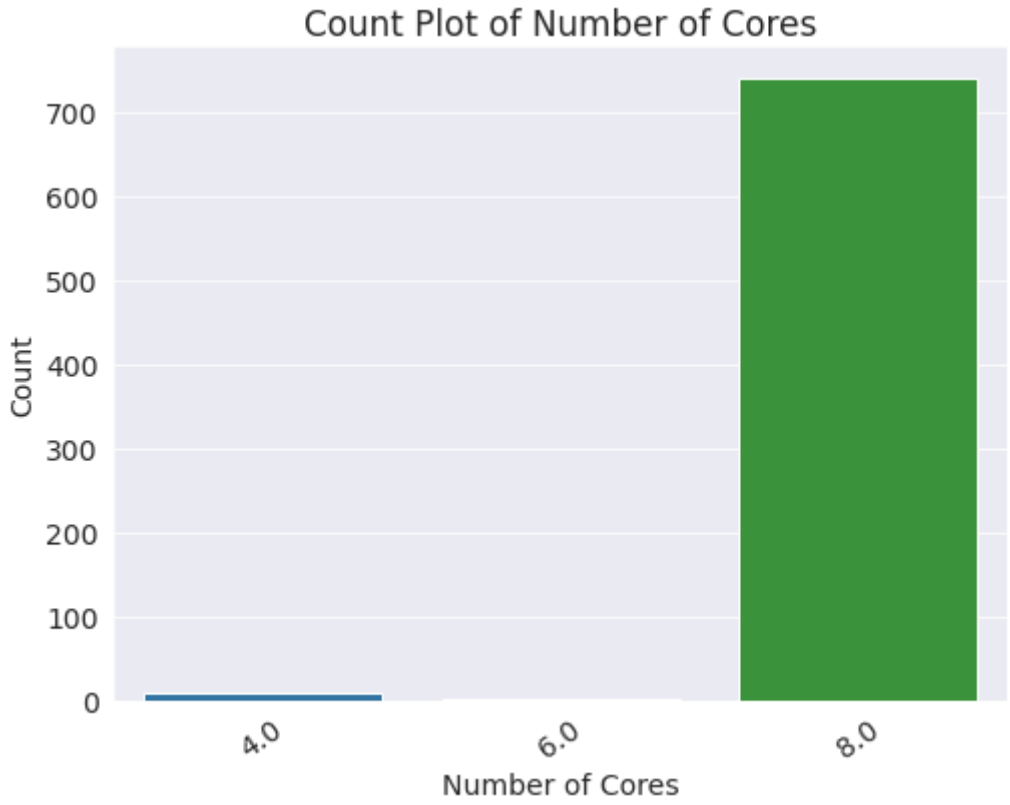
```
df.describe()
```

	price	avg_rating	5G_or_not	num_cores	processor_speed	battery_capacity	fast_charging_availab
count	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.
mean	30529.662667	7.999867	0.633333	7.952000	2.468360	4928.064000	1.

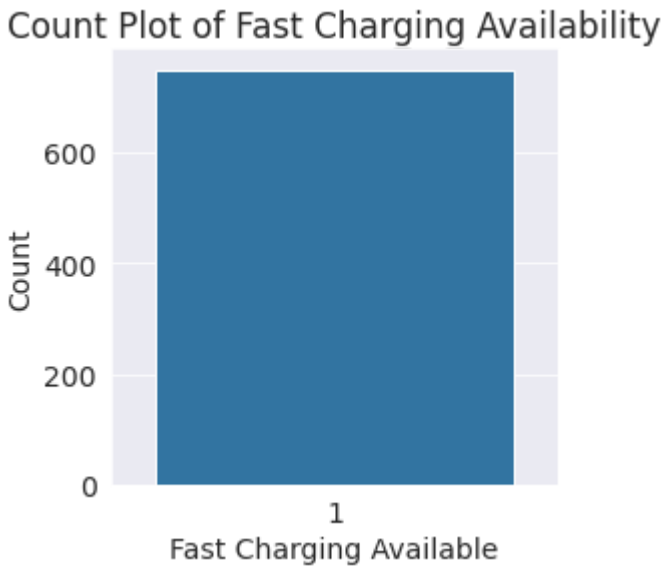
	price	avg_rating	5G_or_not	num_cores	processor_speed	battery_capacity	fast_charging_availab
std	29646.372834	0.589664	0.482216	0.423401	0.420745	987.525223	0.
min	5785.000000	6.000000	0.000000	4.000000	1.600000	1821.000000	1.
25%	14990.000000	7.700000	0.000000	8.000000	2.200000	4500.000000	1.
50%	21839.000000	8.000000	1.000000	8.000000	2.400000	5000.000000	1.
75%	34999.000000	8.400000	1.000000	8.000000	2.850000	5000.000000	1.
max	480000.000000	8.900000	1.000000	8.000000	3.200000	22000.000000	1.

Number Of Cores - Explore one or more columns by plotting a graph below, and add some explanation about it

```
plt.figure(figsize = (8,6))
sns.countplot(data=df, x='num_cores')
plt.title("Count Plot of Number of Cores")
plt.xlabel("Number of Cores")
plt.ylabel("Count")
plt.xticks(rotation=35)
plt.show()
```

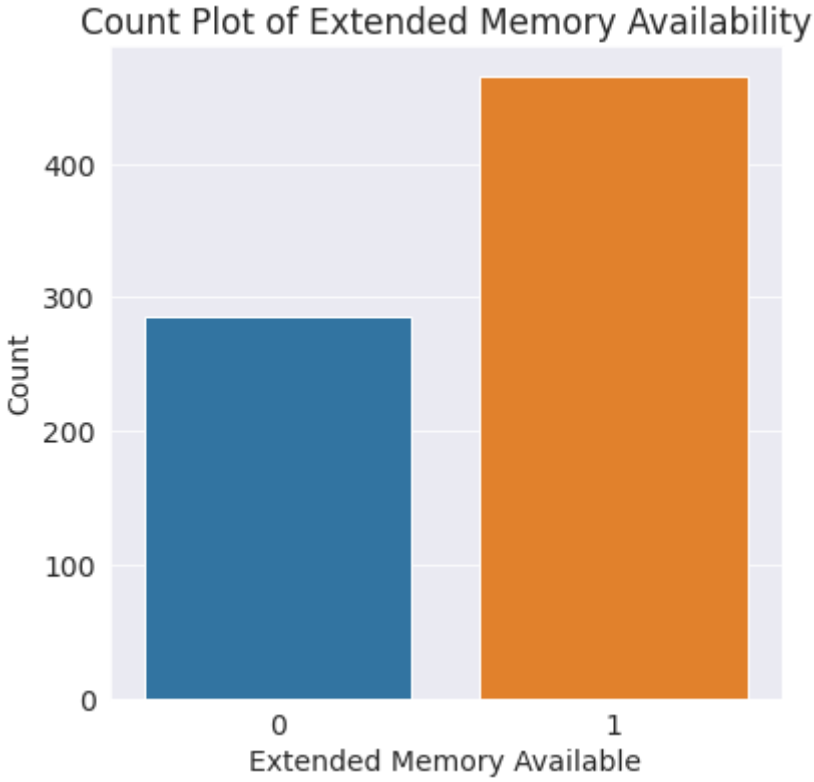



```
plt.figure(figsize=(4, 4))
sns.countplot(data=df, x='fast_charging_available')
plt.title("Count Plot of Fast Charging Availability")
plt.xlabel("Fast Charging Available")
plt.ylabel("Count")
plt.show()
```



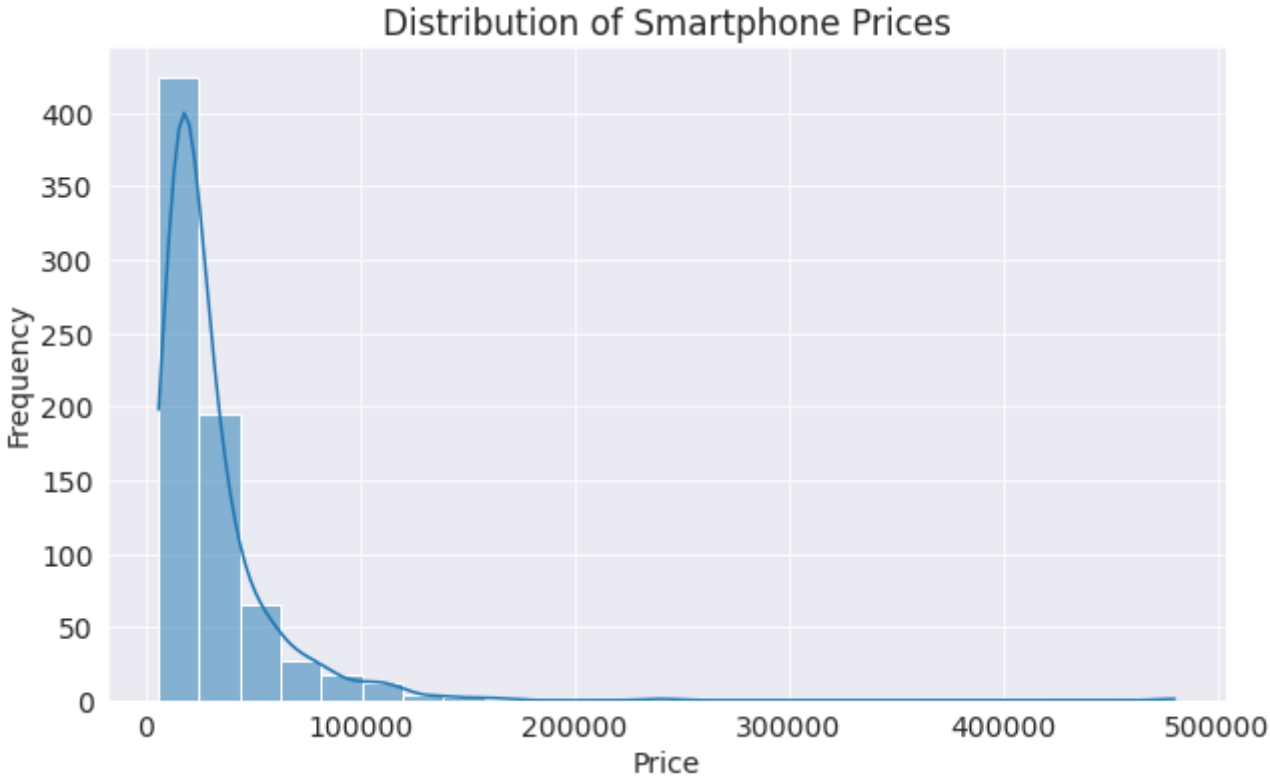
Extended Memory Availability - Explore one or more columns by plotting a graph below, and add some explanation about it

```
plt.figure(figsize = (6,6))
sns.countplot(data=df, x='extended_memory_available')
plt.title("Count Plot of Extended Memory Availability")
plt.xlabel("Extended Memory Available")
plt.ylabel("Count")
plt.show()
```



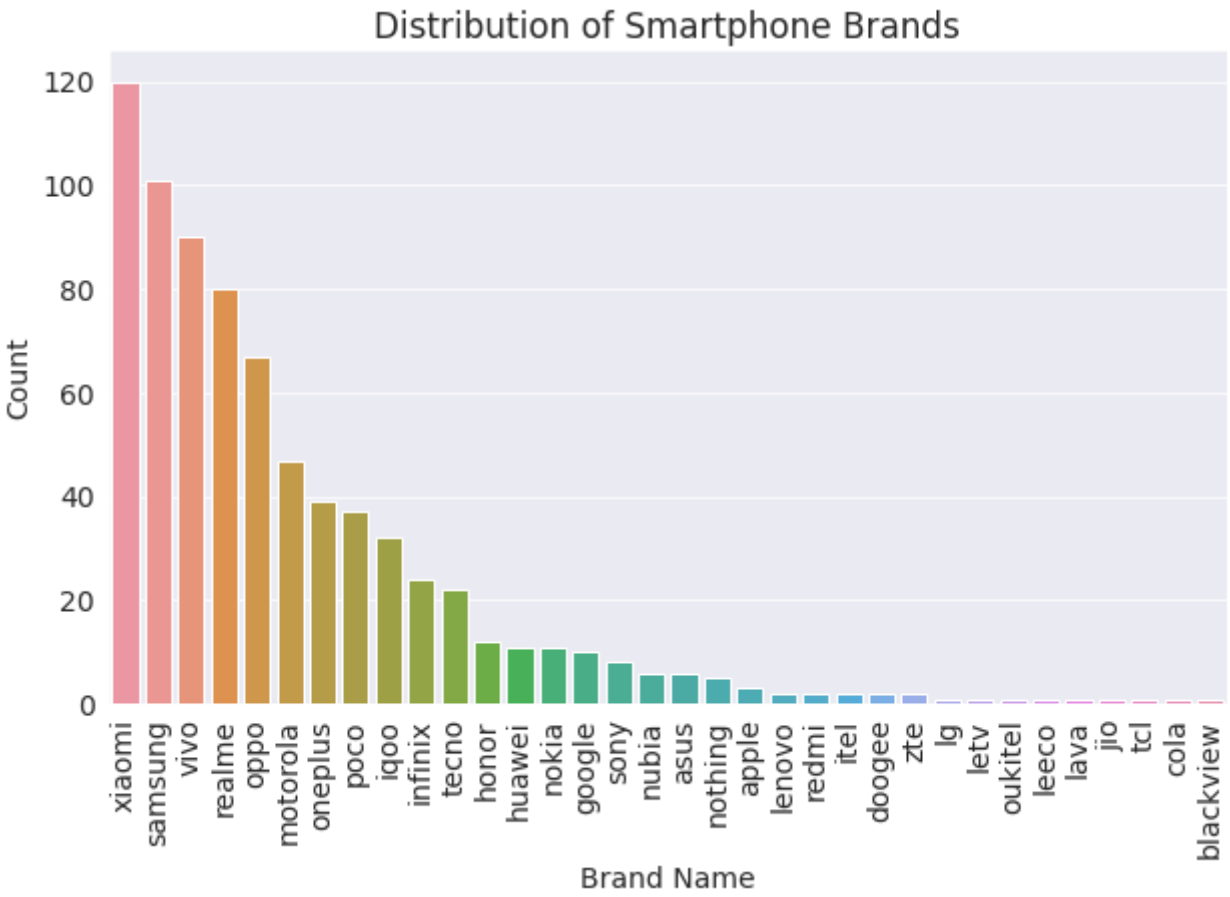
Distribution Of Smartphone Prices - Explore one or more columns by plotting a graph below, and add some explanation about it

```
# Create a histogram of prices
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='price', bins=25, kde=True)
plt.title("Distribution of Smartphone Prices")
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.show()
```



Distribution Of Smartphone Brands¶ - Explore one or more columns by plotting a graph below, and add some explanation about it

```
# Bar plot for distribution of brands
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='brand_name', order=df['brand_name'].value_counts().index)
plt.xticks(rotation=90)
plt.title("Distribution of Smartphone Brands")
plt.xlabel("Brand Name")
plt.ylabel("Count")
plt.show()
```



Distribution Of Processor Brands

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

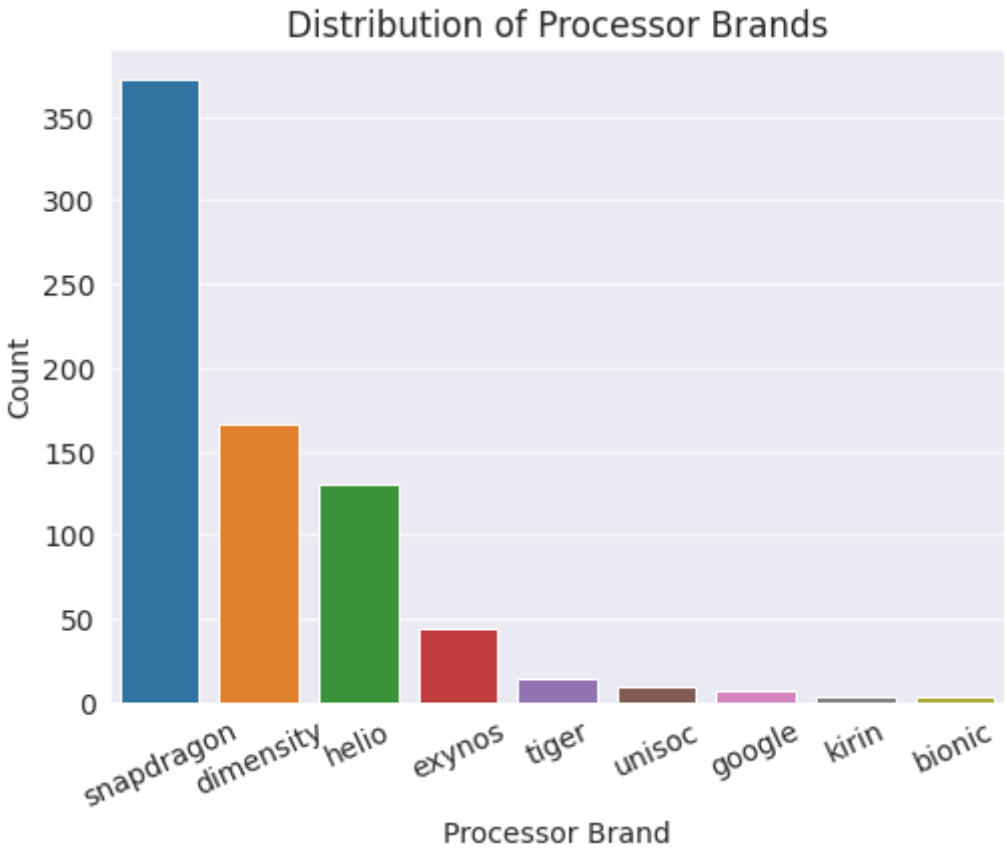
```
snapdragon    372
dimensity     166
helio         130
exynos         44
tiger          14
unisoc         10
google          7
kirin           4
bionic          3
Name: processor_brand, dtype: int64
```

```
df['processor_brand'].value_counts().index
```

```
Index(['snapdragon', 'dimensity', 'helio', 'exynos', 'tiger', 'unisoc',
       'google', 'kirin', 'bionic'],
      dtype='object')
```

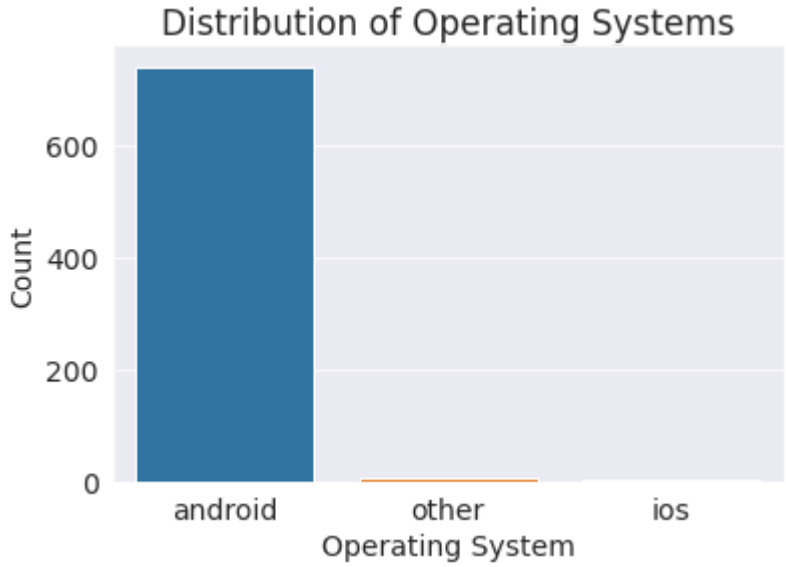
```
# Bar plot for distribution of processor brands
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='processor_brand', order=df['processor_brand'].value_counts().
plt.xticks(rotation=25)
plt.title("Distribution of Processor Brands")
plt.xlabel("Processor Brand")
```

```
plt.ylabel("Count")
plt.show()
```



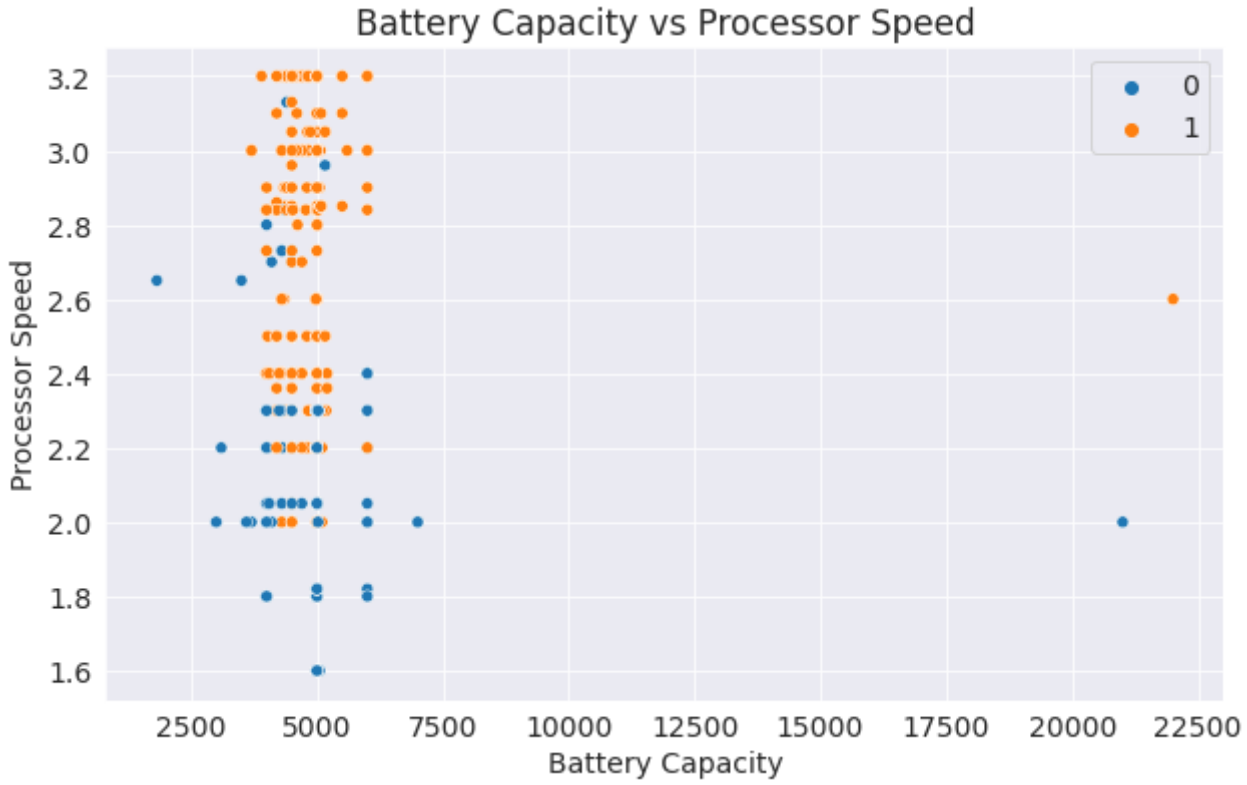
Distribution of Operating Systems¶

```
plt.figure(figsize=(6,4))
sns.countplot(data = df, x = 'os', order = df['os'].value_counts().index)
plt.title("Distribution of Operating Systems")
plt.xlabel("Operating System")
plt.ylabel('Count')
plt.show()
```



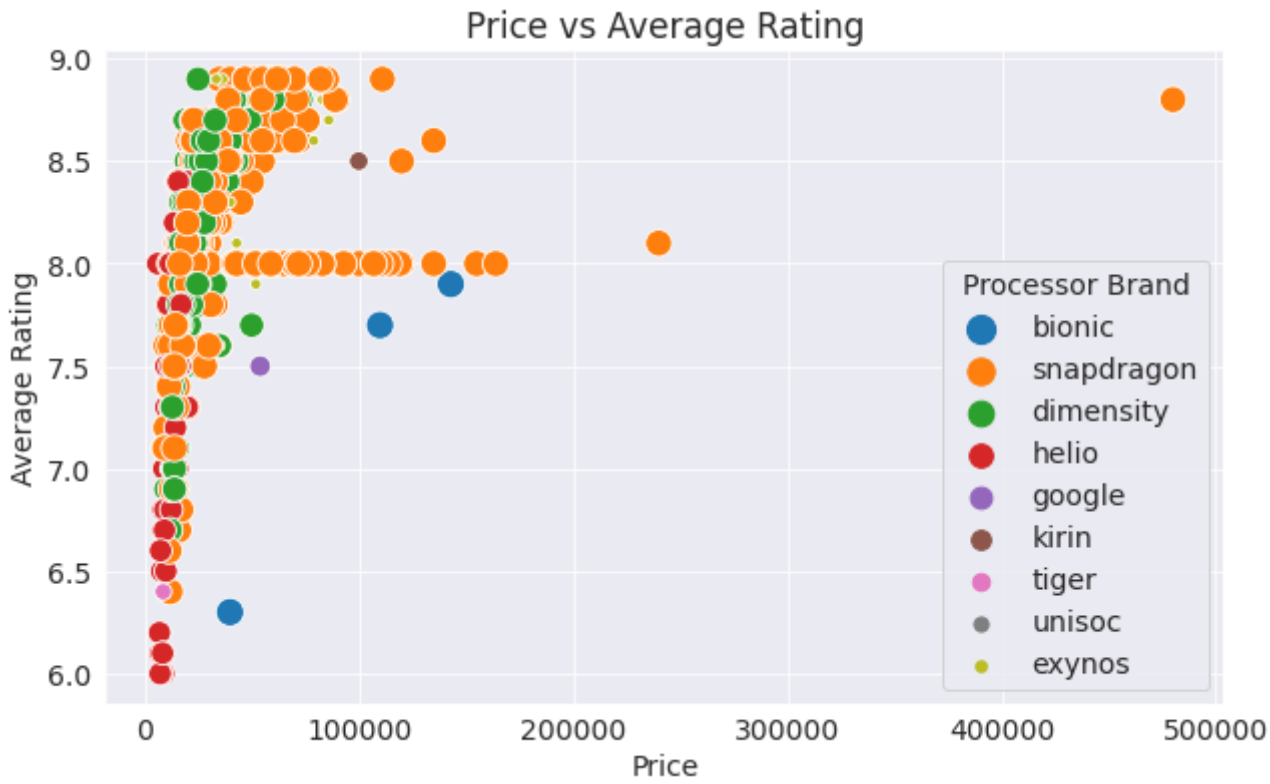
Battery capacity vs. Processor speed, Color-coded by 5G availability

```
# Scatter plot of battery capacity vs. processor speed, color-coded by 5G availability
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='battery_capacity', y='processor_speed', hue='5G_or_not')
plt.title("Battery Capacity vs Processor Speed")
plt.xlabel("Battery Capacity")
plt.ylabel("Processor Speed")
plt.legend()
plt.show()
```



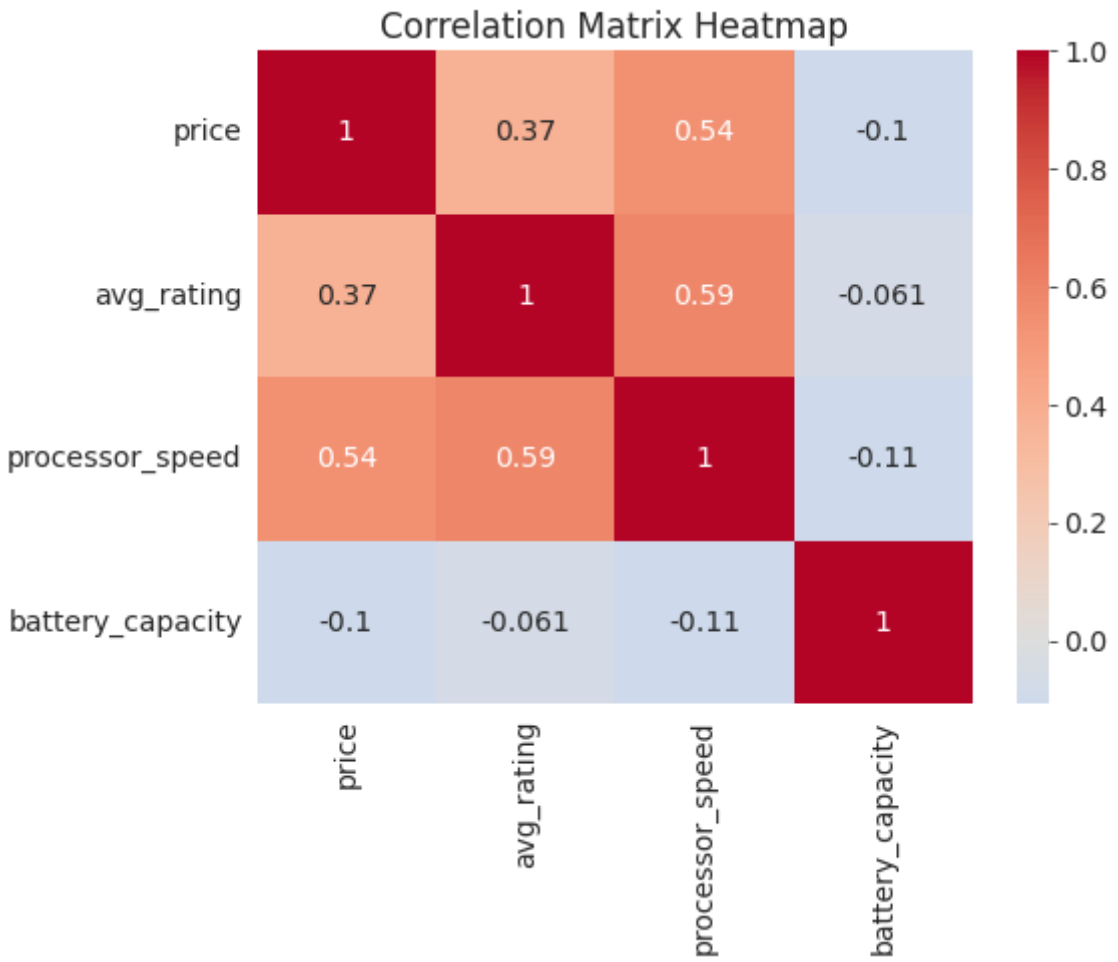
Price vs. Average rating, with Size/Color indicating Processor brand

```
# Scatter plot of price vs. average rating, with size/color indicating processor brand
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='price', y='avg_rating', hue='processor_brand', size='proces
plt.title("Price vs Average Rating")
plt.xlabel("Price")
plt.ylabel("Average Rating")
plt.legend(title="Processor Brand")
plt.show()
```



Correlation Matrix

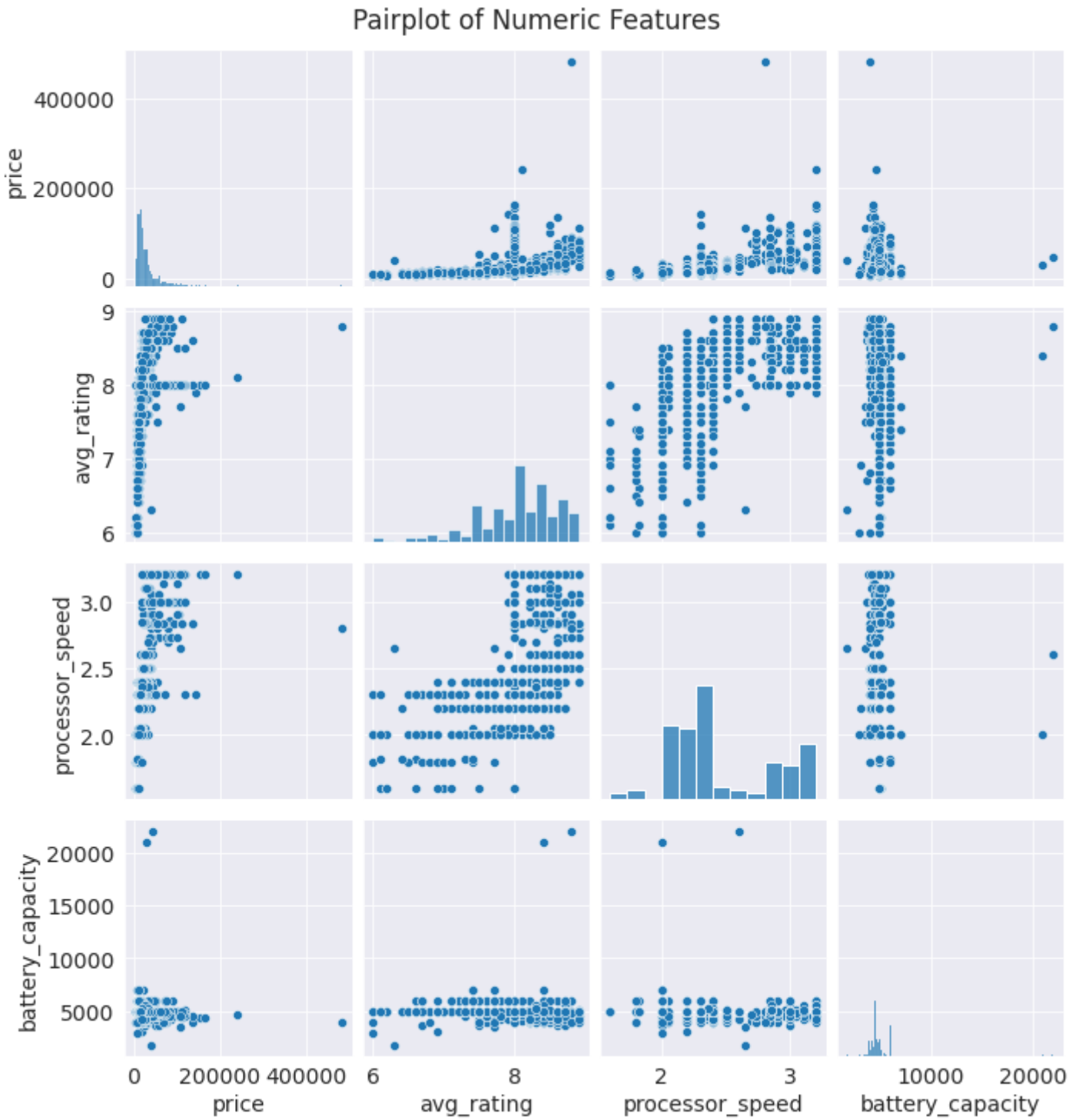
```
# Filter only numeric columns
numeric_columns = ['price', 'avg_rating', 'processor_speed', 'battery_capacity']
numeric_df = df[numeric_columns]

# Calculate the correlation matrix
correlation_matrix = numeric_df.corr()

# Create a heatmap for the correlation matrix
plt.figure(figsize=(8, 6))
sns.heatmap(data=correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title("Correlation Matrix Heatmap")
plt.show()
```



Relationships Of Numeric Features

```
# Create a pairplot for pairwise relationships
plt.figure(figsize= (8,10))
sns.pairplot(data=df[['price', 'avg_rating', 'processor_speed', 'battery_capacity']])
plt.suptitle("Pairplot of Numeric Features", y=1.02)
plt.show()
```

```
<Figure size 576x720 with 0 Axes>
```



Let us save and upload our work to Jovian before continuing

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "ay29020/zerotopandas-real-world-smartphone-data-analysis"
on https://jovian.com
[jovian] Committed successfully! https://jovian.com/ay29020/zerotopandas-real-world-
smartphone-data-analysis
```

```
'https://jovian.com/ay29020/zerotopandas-real-world-smartphone-data-analysis'
```

Asking and Answering Questions

Now we are going to find such 5 intresting answers from our datasets, that people usually have in their mind from smartphone industry.

> Instructions (delete this cell)
>
> - Ask at least 5 interesting questions about your dataset
> - Answer the questions either by computing the results using Numpy/Pandas or by plotting graphs using Matplotlib/Seaborn
> - Create new columns, merge multiple dataset and perform grouping/aggregation wherever necessary
> - Wherever you're using a library function from Pandas/Numpy/Matplotlib etc. explain briefly what it does

Q1: What is top most best expensive brand

```
df.groupby('brand_name').sum().head(5)
```

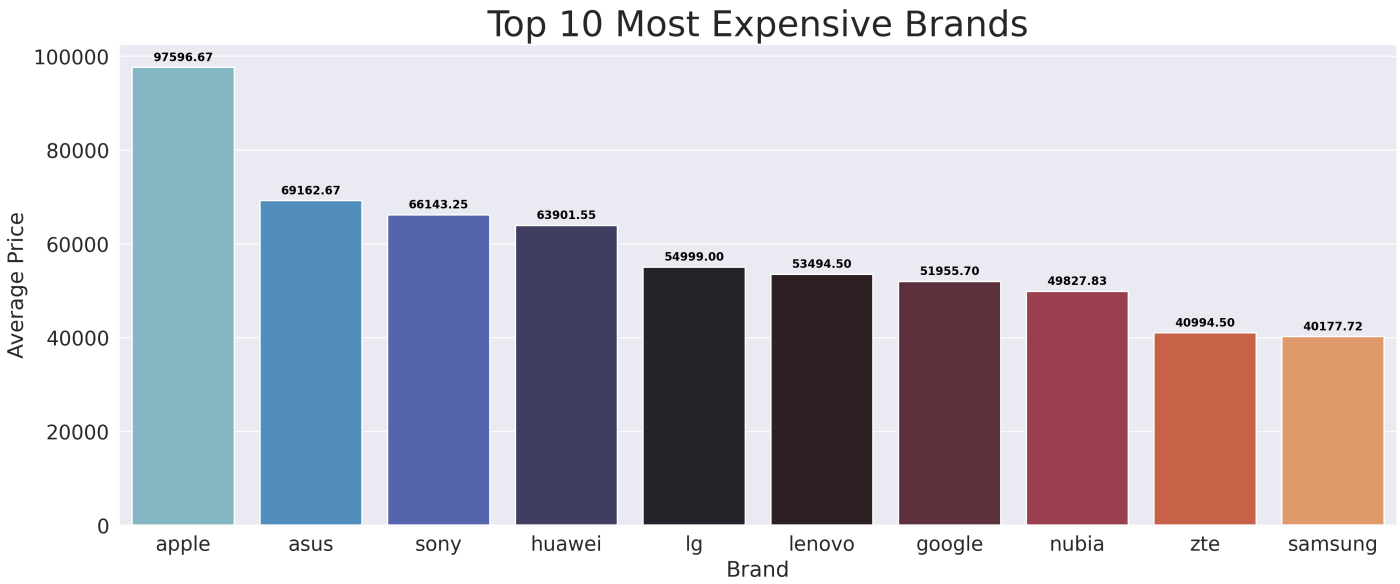
brand_name	price	avg_rating	5G_or_not	num_cores	processor_speed	battery_capacity	fast_charging_available	fast
apple	292790	21.9	1	20.0	7.60	9673.0	3	
asus	414976	52.2	6	48.0	18.90	34300.0	6	
blackview	8990	6.7	1	8.0	2.30	5180.0	1	
cola	14999	7.4	0	8.0	2.20	5000.0	1	
doogee	60998	17.2	1	16.0	4.65	28000.0	2	

```
avg_price = df.groupby('brand_name')['price'].mean().reset_index()
avg_price = avg_price.sort_values(by='price',ascending=False)
avg_price.head()
```

	brand_name	price
0	apple	97596.666667
1	asus	69162.666667
28	sony	66143.250000
7	huawei	63901.545455
16	lg	54999.000000

```
# plot for top 10 most expensive brands
# figsize
plt.figure(figsize=(14, 6), dpi=300)
```

```
# Plot
ax = sns.barplot(data=avg_price.head(10), x='brand_name', y='price', palette='icefire',

# Add labels to the bars
ax.bar_label(ax.containers[0], fmt='%.2f', label_type='edge', fontsize=8, color='black'

# Labels
plt.xlabel('Brand', fontsize=15)
plt.ylabel('Average Price', fontsize=15)
plt.title('Top 10 Most Expensive Brands', fontsize=25)
plt.tight_layout()
plt.show()
```



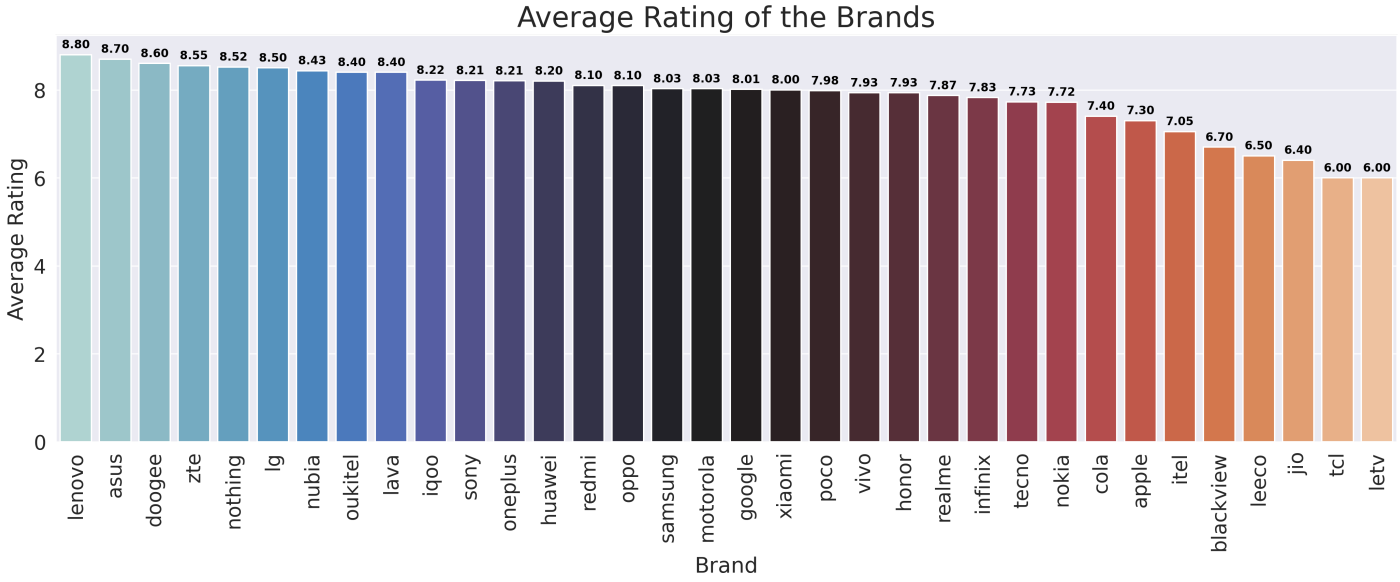
Q2: What is average rating for each brand

```
# average rating for each brand

avg_rating = df.groupby('brand_name')['avg_rating'].mean().reset_index()
avg_rating = avg_rating.sort_values(by='avg_rating',ascending=False)
avg_rating.head()
```

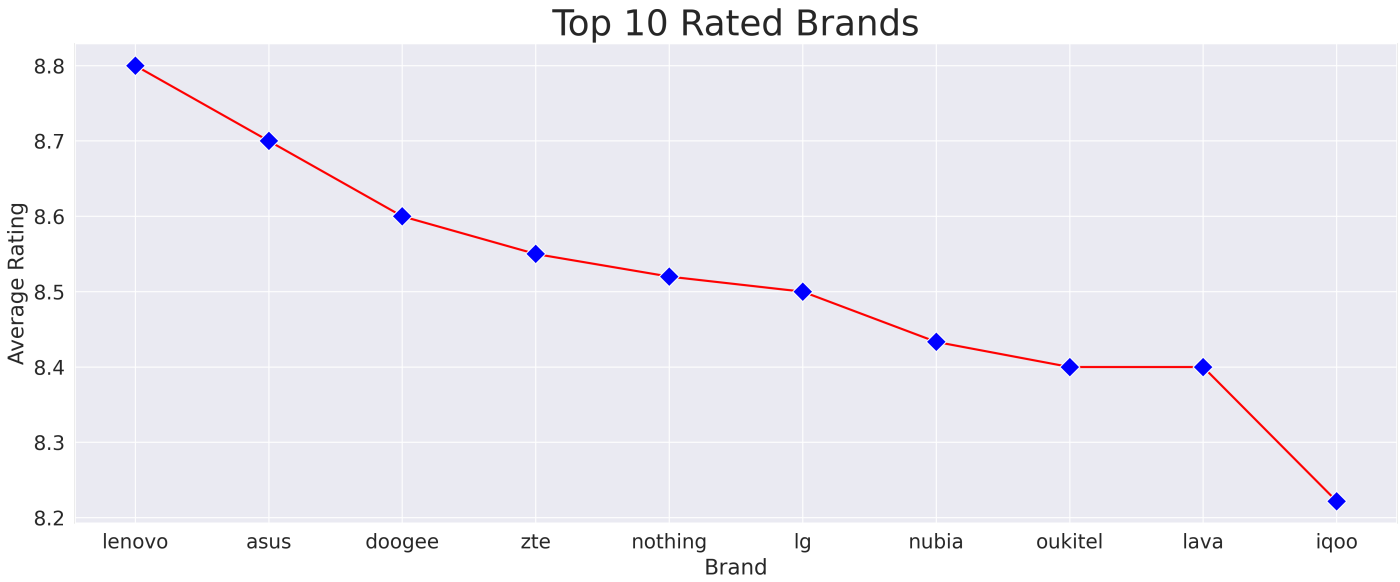
	brand_name	avg_rating
14	lenovo	8.80
1	asus	8.70
4	doogee	8.60
33	zte	8.55
19	nothing	8.52

```
# Barplot for average rating for each brand

plt.figure(figsize=(14, 6), dpi=300)

# Plot
```

```
ax = sns.barplot(data=avg_rating.head(44), x='brand_name',y='avg_rating',palette='icefi

# Add labels to the bars
ax.bar_label(ax.containers[0], fmt='%.2f', label_type='edge', fontsize=8, color='black'

# Labels
plt.xlabel('Brand',fontsize=15)
plt.ylabel('Average Rating',fontsize=15)
plt.title('Average Rating of the Brands',fontsize=20)
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



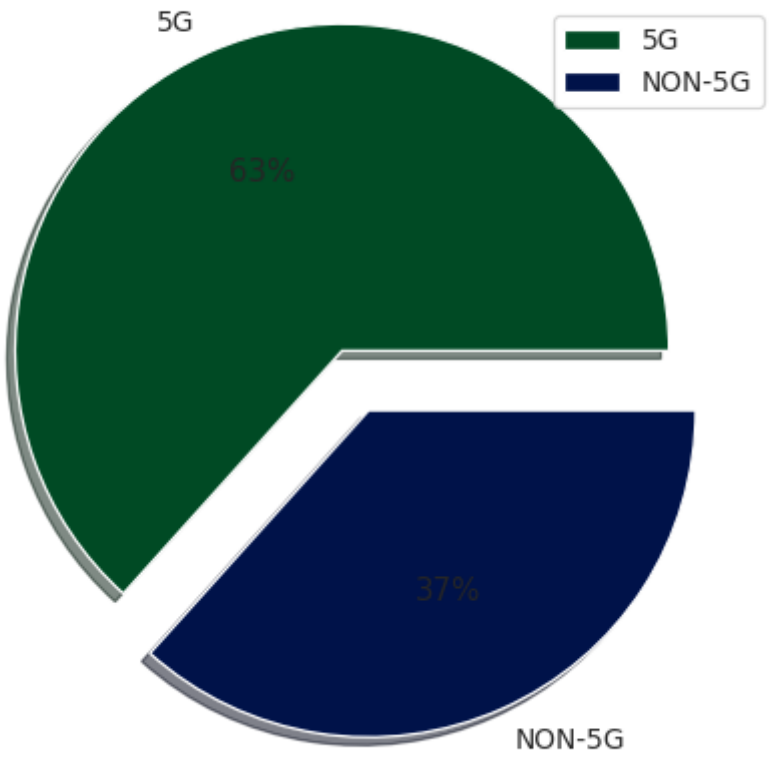

```
# Lineplot for top 10 rated brands
# figsize
plt.figure(figsize=(14,6),dpi=400)

# plot
sns.lineplot(data=avg_rating.head(10),x='brand_name',y='avg_rating',marker='D',color='r

#Labels
plt.xlabel('Brand',fontsize=15)
plt.ylabel('Average Rating',fontsize=15)
plt.title('Top 10 Rated Brands',fontsize=25)
plt.tight_layout()
plt.show()
```



Q3: how maany smartphones have 5G

```
# Pie plot for how many smartphones have 5G
# figsize
plt.figure(figsize=(14, 6), dpi=90)
palette_color = sns.color_palette('ocean')

# Plot
plt.pie(df['5G_or_not'].value_counts(), labels=['5G', 'NON-5G'], colors=palette_color,
        shadow=True, explode=[0.2, 0], autopct='%.0f%%')

# Labels
plt.legend(loc=1)
plt.suptitle('5G vs. Non-5G Smartphones', fontsize=20)
# Error "Figure size 1260x540 with 0 Axes."  often occurs when plt.show() is used more
```

```
Text(0.5, 0.98, '5G vs. Non-5G Smartphones')
```



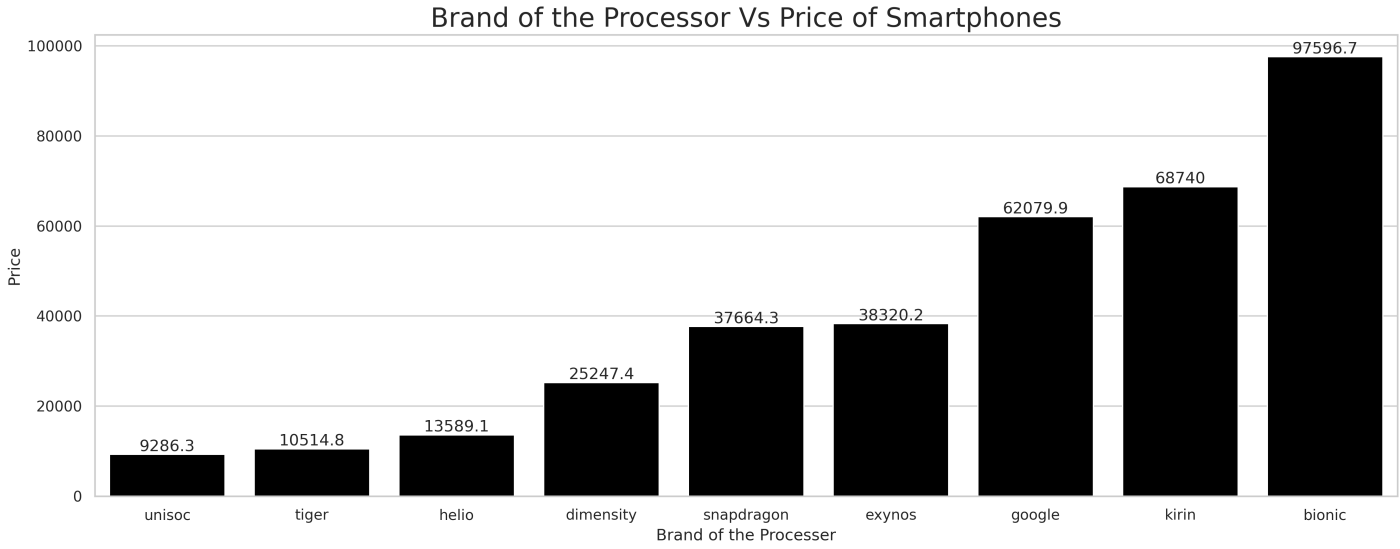
Q4: what is average price for each processor

```
process_brand_price = df.groupby('processor_brand')['price'].mean().reset_index()
process_brand_price = process_brand_price.sort_values(by='price',ascending=True)
process_brand_price.head()
```

	processor_brand	price
8	unisoc	9286.300000
7	tiger	10514.785714
4	helio	13589.115385
1	dimensity	25247.355422
6	snapdragon	37664.330645

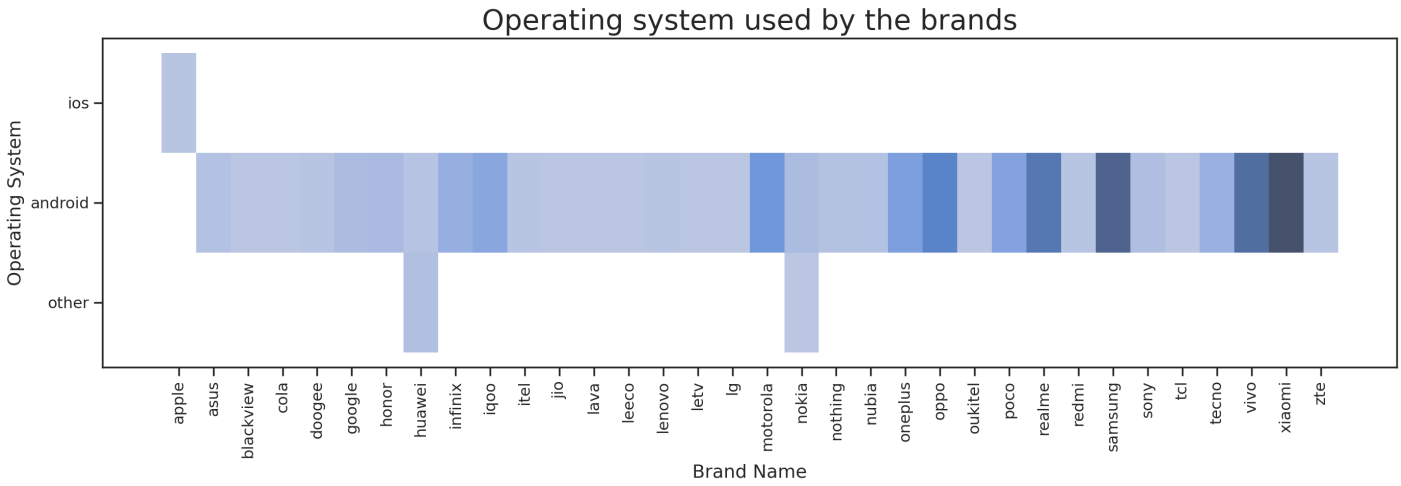
```
# Barplot for brand of the processor and average price of the smartphone
#figsize
plt.figure(figsize=(15,6),dpi=300)

# Plot
ax = sns.barplot(data=process_brand_price,x='processor_brand',y='price',color ='black')
```

```
ax.bar_label(ax.containers[0])

# Labels
plt.xlabel("Brand of the Processer",fontsize= 12)
plt.ylabel('Price',fontsize= 12)
plt.title('Brand of the Processor Vs Price of Smartphones',fontsize=20)
plt.tight_layout()
plt.show()
```



Q5: TODO - What types of operatig system used in the brands.

```
# Histplot to show Operatig system used in the brands.
# figsize
plt.figure(figsize=(14,5),dpi=200)
sns.set_style('ticks')
# plot
sns.histplot(df,x='brand_name',y='os',binwidth=.3)

# labels
plt.xticks(rotation=90)
plt.xlabel('Brand Name',fontsize=13)
plt.ylabel('Operating System',fontsize=13)
plt.title('Operating system used by the brands',fontsize=20)
plt.tight_layout()
```



Let us save and upload our work to Jovian before continuing.

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "ay29020/zerotopandas-real-world-smartphone-data-analysis"
on https://jovian.com
[jovian] Committed successfully! https://jovian.com/ay29020/zerotopandas-real-world-
smartphone-data-analysis
```

```
'https://jovian.com/ay29020/zerotopandas-real-world-smartphone-data-analysis'
```

Inferences and Conclusion

TODO - Write some explanation here: a summary of all the inferences drawn from the analysis, and any conclusions you may have drawn by answering various questions.

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "ay29020/zerotopandas-real-world-smartphone-data-analysis"
on https://jovian.com
[jovian] Committed successfully! https://jovian.com/ay29020/zerotopandas-real-world-
smartphone-data-analysis
```

```
'https://jovian.com/ay29020/zerotopandas-real-world-smartphone-data-analysis'
```

References and Future Work

TODO - Write some explanation here: ideas for future projects using this dataset, and links to resources you found useful.

> Submission Instructions (delete this cell)
>
> - Upload your notebook to your Jovian.ml profile using `jovian.commit`.
> - **Make a submission here**: https://jovian.ml/learn/data-analysis-with-python-zero-to-pandas/assignment/course-project
> - Share your work on the forum: https://jovian.ml/forum/t/course-project-on-exploratory-data-analysis-discuss-and-share-your-work/11684
> - Share your work on social media (Twitter, LinkedIn, Telegram etc.) and tag @JovianML
>
> (Optional) Write a blog post
>
> - A blog post is a great way to present and showcase your work.
> - Sign up on Medium.com to write a blog post for your project.
> - Copy over the explanations from your Jupyter notebook into your blog post, and embed code cells & outputs
> - Check out the Jovian.ml Medium publication for inspiration: https://medium.com/jovianml

```
import jovian
```

```
jovian.commit()
```

```

```