INT - 353 EDA PROJECT

Name : Shaik Julfeen Ahmadh

Registration Number : 12110554

Section : K21UG

Roll No : RK21UGB53

Mobile Trends Analysis

Introduction

The advent of technology has significantly transformed the landscape of mobile devices, offering an array of choices to consumers with diverse preferences and requirements. This report delves into the dynamic world of mobile phones, aiming to analyse and comprehend the trends prevalent in the market. The dataset, sourced from 'mobile_trends.csv', encapsulates a comprehensive array of information pertaining to various mobile phone models, providing insights into crucial aspects such as brand, model name, price, rating, SIM types, and several other key features.

As we navigate through the dataset, we will uncover patterns, correlations, and noteworthy trends that define the contemporary mobile market. From the operating systems that power these devices to the intricacies of their processors, the dataset offers a holistic view of the technological landscape. Furthermore, factors like battery capacity, display size, and camera specifications contribute to the nuanced decision-making process of consumers. Our analysis will not only shed light on the individual attributes of mobile phones but also explore potential interdependencies among these features. Whether it's the impact of the operating system on user satisfaction or the correlation between price and device specifications, our goal is to extract meaningful insights that can inform both consumers and industry stakeholders.

DOMAIN

Domain: Mobile Technology Market Analysis

This dataset resides within the domain of the Mobile Technology Market, offering a detailed exploration of the contemporary landscape of mobile phones. Capturing essential attributes such as brand, model name, pricing, user ratings, SIM types, and diverse technical specifications, the dataset serves as a valuable resource for understanding the intricacies of the mobile technology sector.

Within this domain, we aim to uncover patterns, trends, and relationships that define consumer preferences, technological advancements, and competitive dynamics. The dataset provides a

comprehensive overview of mobile phone features, enabling a thorough analysis of factors influencing consumer choices and market trends. The insights derived from this exploration will contribute to a deeper understanding of the evolving dynamics within the Mobile Technology Market.

Why this dataset?

Potential Reasons for Choosing This Dataset for an EDA (Exploratory Data Analysis) project are…

Personal Interest: • The topic of mobile technology aligns with a personal interest in staying updated on the latest advancements and trends in the tech world. Exploring mobile phones, their features, and market dynamics is inherently engaging and satisfying.

Relevance: • The dataset is highly relevant to the current technological landscape. Mobile phones are integral to daily life, and understanding the nuances of different models provides insights into consumer behaviour and industry competitiveness. Market Insights: • The dataset promises to deliver valuable market insights, allowing for a deeper understanding of consumer preferences, brand dynamics, and emerging trends within the Mobile Technology Market. This information is essential for anyone seeking to stay informed about market shifts.

Practical Application: • The practical application of insights derived from this dataset extends to real-world scenarios. Whether making informed purchasing decisions as a consumer or strategizing within the mobile technology industry, the practical implications make this dataset particularly valuable.

Data Availability: • The dataset offers a rich array of information, providing a comprehensive snapshot of the mobile phone market. The availability of diverse variables ensures that there is ample data to explore, making it suitable for in-depth analysis.

Educational Value: • This project is a part of coursework, and the dataset's relevance to the educational objectives makes it an ideal choice. The exploration of the Mobile Technology Market aligns with the learning goals of the course, allowing for practical application of concepts and methodologies.

Availability of Variety: • The inclusion of various brands, models, and technical specifications ensures a diverse dataset. This variety not only makes the analysis interesting but also provides a holistic view of the mobile technology landscape, enhancing the educational and analytical value of the project.

Dataset Overview:

The dataset encapsulates information from the Mobile Technology Market, spanning various brands and models. It comprises 26 columns, each representing a distinct attribute, offering a holistic view of mobile phone specifications and features.

Libraries used and Approaches:

The following libraries were used for the project and their brief description as follows: NumPy: NumPy, short for "Numerical Python," is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a variety of high-level mathematical functions to operate on these arrays. NumPy is often a foundational library for data manipulation and analysis in Python.

Pandas: Pandas is a powerful data manipulation and analysis library for Python. It provides data structures like DataFrames and Series, which make it easy to work with structured data, such as CSV files, Excel spreadsheets, or SQL databases. Pandas is widely used for data cleaning, transformation, aggregation, and exploration

Matplotlib: Matplotlib is a popular Python library for creating static, animated, and interactive visualizations in Python. It offers a wide range of plotting options, allowing you to create various types of charts and graphs, from simple line plots to complex heat maps and 3D plots. Matplotlib is highly customizable and can be used alongside other libraries like NumPy and Pandas for data visualisation.

Seaborn: Seaborn is a data visualisation library built on top of Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics. Seaborn simplifies the process of creating complex visualizations, such as scatter plots, bar plots, and heatmaps, and it also offers built-in support for working with Pandas DataFrames.

warnings Module:Purpose: The warnings module is part of the Python Standard Library and provides a way to handle warning messages in a program. Warnings are messages that indicate potential issues or usage of deprecated features but don't necessarily halt the execution of the program.

scipy.stats Module: The scipy.stats module is part of the SciPy library, which is built on top of NumPy and provides additional functionality for scientific computing. The stats module, in particular, focuses on statistical functions and tests, offering a wide range of tools for analyzing and manipulating statistical data.

Importing the libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from scipy import stats
```

Data Description

The dataset is a comprehensive collection of information about mobile phones, encompassing crucial attributes such as brand, model name, price, rating, SIM type, and a variety of hardware features. Each mobile phone entry provides details on display size, operating system, processor core, internal storage, camera specifications, network type, and connectivity features like

Bluetooth and Wi-Fi. This well-structured dataset proves valuable for a thorough analysis of the diverse characteristics exhibited by different mobile phone models.

Notably, the dataset stands out for its cleanliness, as there are no missing values in any of its columns. This absence of missing data enhances the reliability of the dataset, ensuring that each entry is complete and accurate. With a wealth of detailed information and a lack of missing values, this dataset becomes a robust resource for gaining insights into the features and specifications of various mobile phones, facilitating informed decision-making and analysis in the realm of mobile technology.

Information On Variables

```
Brand (Type: Object): The brand of the mobile phone.
Model Name (Type: Object): The specific model name of the mobile phone.
Price (Type: Int64): The price of the mobile phone.
Rating (Type: Float64): The rating assigned to the mobile phone.
SIM Type (Type: Object): The type of SIM card supported.
Hybrid Sim Slot (Type: Object): Indicates if the mobile phone has a hybrid SIM slot.
Touchscreen (Type: Object): Indicates if the mobile phone has a touchscreen.
Display_size (Type: Float64): Size of the mobile phone display.
Operating System (Type: Object): The operating system installed on the mobile phone.
Processor Core (Type: Object): The number of processor cores.
Internal Storage (Type: Float64): Amount of internal storage available.
Primary Camera (Type: Object): Specifications of the primary (main) camera.
Secondary Camera (Type: Object): Specifications of the secondary (front) camera.
Network Type (Type: Object): Type of network supported.
Bluetooth Version (Type: Object): The version of Bluetooth supported.
Wi-Fi (Type: Object): Indicates if Wi-Fi is supported.
GPS Support (Type: Object): Indicates if the mobile phone has GPS support.
SIM Size (Type: Object): Size of the SIM card.
Battery Capacity (Type: Object): The capacity of the mobile phone's battery.
Width (Type: Float64): Width of the mobile phone.
```

Steps of EDA

Data Exploration

Loading the dataset

```
mobile = pd.read_csv('mobile_trends.csv')
```

Investigating the structure

```
mobile.head()
```

```
     Brand          Model Name  Price  Rating  SIM Type Hybrid Sim Slot
\
0    APPLE           iPhone 13  52499     4.7  Dual Sim               No

1     POCO                 C51   6499     4.1  Dual Sim               No

2  OnePlus  Nord CE 2 Lite 5G  17196     4.4  Dual Sim               No

3   realme              11x 5G  15999     4.3  Dual Sim               No

4   realme              11x 5G  14999     4.4  Dual Sim               No


  Touchscreen  Display_size Operating System Processor Core  ... GPS
Support  \
0         Yes          6.10           iOS 15      Hexa Core  ...
Yes
1         Yes          6.52       Android 13      Octa Core  ...
Yes
2         Yes          6.59       Android 13      Octa Core  ...
NaN
3         Yes          6.72       Android 13      Octa Core  ...
Yes
4         Yes          6.72       Android 13      Octa Core  ...
Yes

      SIM Size Battery Capacity     Width    Height Weight SIM Type.1
\
0  Nano + eSIM         3240 mAh   71.5 mm  146.7 mm  173 g   Dual Sim

1     Nano Sim         5000 mAh  76.75 mm  164.9 mm  192 g   Dual Sim

2     Nano Sim         5000 mAh  76.75 mm  164.9 mm  192 g   Dual Sim

3     Nano Sim         5000 mAh     76 mm  165.7 mm  190 g   Dual Sim

4     Nano Sim         5000 mAh     76 mm  165.7 mm  190 g   Dual Sim
```

```
  Hybrid Sim Slot.1 Dual Camera Lens           Color
0                No   Primary Camera           Pink
1                No   Primary Camera     Power Black
2                No              NaN      Black Dusk
3                No   Primary Camera     Purple Dawn
4                No   Primary Camera  Midnight Black

[5 rows x 26 columns]
```

mobile.tail()

```
     Brand                                        Model Name  Price
Rating  \
979  APPLE                                        iPhone 12  51999
4.6
980  Tecno                                           Pova 3  13999
4.1
981  REDMI                                        Note 11 SE  12890
4.3
982   vivo                                              Y16  10699
4.2
983  Tecno  Camon 19 Pro Multi-Colour Changing Back-Panel  14940
4.1

     SIM Type Hybrid Sim Slot Touchscreen  Display_size Operating
System  \
979  Dual Sim              No         Yes          6.10           iOS
14
980  Dual Sim              No         Yes          6.90       Android
12
981  Dual Sim              No         Yes          6.43       Android
11
982  Dual Sim              No         Yes          6.51       Android
12
983  Dual Sim              No         Yes          6.80       Android
12

    Processor Core  ... GPS Support     SIM Size Battery Capacity
Width  \
979            NaN  ...         Yes  Nano + eSIM         4500 mAh
71.5 mm
980      Octa Core  ...         Yes     Nano-SIM         7000 mAh
78.46 mm
981      Octa Core  ...         Yes     Nano Sim         5000 mAh
74.5 mm
982      Octa Core  ...         Yes     Nano Sim         5000 mAh
75.55 mm
983      Octa Core  ...         Yes         Nano         5000 mAh
```

```
74.55 mm

        Height   Weight SIM Type.1 Hybrid Sim Slot.1 Dual Camera Lens
\
979   146.7 mm    162 g   Dual Sim                No   Primary Camera

980   173.1 mm      NaN   Dual Sim                No   Primary Camera

981  160.46 mm  178.8 g   Dual Sim                No   Primary Camera

982  163.95 mm    183 g   Dual Sim                No   Primary Camera

983  166.79 mm      NaN   Dual Sim                No   Primary Camera


               Color
979            White
980     Tech Silver
981  Thunder Purple
982   Steller Black
983         Mondrian

[5 rows x 26 columns]

mobile.shape

(984, 26)
```

What is the target variable?

The target variable is `Price` because it is necessary for us to analyse any mobile and is a must to evaluate the features for any mobile.

2.Data Cleaning

Investigating the Quality

```
# Obtain basic information about columns
mobile.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 984 entries, 0 to 983
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Brand              984 non-null    object
 1   Model Name         984 non-null    object
 2   Price              984 non-null    int64
 3   Rating             984 non-null    float64
```

```
 4   SIM Type            984 non-null    object
 5   Hybrid Sim Slot     979 non-null    object
 6   Touchscreen         981 non-null    object
 7   Display_size        984 non-null    float64
 8   Operating System    818 non-null    object
 9   Processor Core      722 non-null    object
 10  Internal Storage    973 non-null    object
 11  Primary Camera      955 non-null    object
 12  Secondary Camera    617 non-null    object
 13  Network Type        979 non-null    object
 14  Bluetooth Version   569 non-null    object
 15  Wi-Fi               600 non-null    object
 16  GPS Support         604 non-null    object
 17  SIM Size            834 non-null    object
 18  Battery Capacity    984 non-null    object
 19  Width               880 non-null    object
 20  Height              875 non-null    object
 21  Weight              940 non-null    object
 22  SIM Type.1          984 non-null    object
 23  Hybrid Sim Slot.1   979 non-null    object
 24  Dual Camera Lens    663 non-null    object
 25  Color               984 non-null    object
dtypes: float64(2), int64(1), object(23)
memory usage: 200.0+ KB
```

mobile.dtypes

```
Brand                 object
Model Name            object
Price                  int64
Rating               float64
SIM Type              object
Hybrid Sim Slot       object
Touchscreen           object
Display_size         float64
Operating System      object
Processor Core        object
Internal Storage      object
Primary Camera        object
Secondary Camera      object
Network Type          object
Bluetooth Version     object
Wi-Fi                 object
GPS Support           object
SIM Size              object
Battery Capacity      object
Width                 object
Height                object
Weight                object
SIM Type.1            object
```

```
Hybrid Sim Slot.1     object
Dual Camera Lens      object
Color                 object
dtype: object
```

```
#Statistical Summary of numerical columns
mobile.describe()
```

```
                Price      Rating  Display_size
count      984.000000  984.000000    984.000000
mean     19280.004065    4.165041      5.315948
std      26030.388513    0.281070      2.145569
min        597.000000    2.900000      0.660000
25%       2098.000000    4.000000      2.550000
50%      11634.500000    4.200000      6.550000
75%      21999.000000    4.300000      6.700000
max     199900.000000    5.000000     16.510000
```

```
# Removing units from few categorical columns to convert them to
numerical
mobile['Internal Storage'] = mobile['Internal
Storage'].str.strip('GB')
mobile['Internal Storage'] = mobile['Internal
Storage'].str.strip('MB')
#mobile['Battery Capacity'] = mobile['Battery
Capacity'].str.strip('mAh')
mobile['Width'] = mobile['Width'].str.strip('mm')
mobile['Height'] = mobile['Height'].str.strip('mm')
mobile['Weight'] = mobile['Weight'].str.strip('g')

# Converting few categorical columns into nmerical for future use
mobile['Internal Storage']=mobile['Internal Storage'].astype(float)
#mobile['Battery Capacity']=mobile['Battery Capacity'].astype(float)
mobile['Width']=mobile['Width'].astype(float)
mobile['Height']=mobile['Height'].astype(float)
mobile['Weight']=mobile['Weight'].astype(float)

mapping_dict = {'V5.1': 'v5.1', '5.2': 'v5.2', 'V5.0': 'v5.0', '4.2':
'v4.2', '5': 'v5.3'}

mobile['Bluetooth Version'] = mobile['Bluetooth
Version'].replace(mapping_dict)

mobile.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 984 entries, 0 to 983
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Brand              984 non-null    object
```

```
 1   Model Name          984 non-null    object
 2   Price               984 non-null    int64
 3   Rating              984 non-null    float64
 4   SIM Type            984 non-null    object
 5   Hybrid Sim Slot     979 non-null    object
 6   Touchscreen         981 non-null    object
 7   Display_size        984 non-null    float64
 8   Operating System    818 non-null    object
 9   Processor Core      722 non-null    object
 10  Internal Storage    973 non-null    float64
 11  Primary Camera      955 non-null    object
 12  Secondary Camera    617 non-null    object
 13  Network Type        979 non-null    object
 14  Bluetooth Version   569 non-null    object
 15  Wi-Fi               600 non-null    object
 16  GPS Support         604 non-null    object
 17  SIM Size            834 non-null    object
 18  Battery Capacity    984 non-null    object
 19  Width               880 non-null    float64
 20  Height              875 non-null    float64
 21  Weight              940 non-null    float64
 22  SIM Type.1          984 non-null    object
 23  Hybrid Sim Slot.1   979 non-null    object
 24  Dual Camera Lens    663 non-null    object
 25  Color               984 non-null    object
dtypes: float64(6), int64(1), object(19)
memory usage: 200.0+ KB
```

```
# Checking for Null values
mobile.isnull().sum()
```

```
Brand                   0
Model Name              0
Price                   0
Rating                  0
SIM Type                0
Hybrid Sim Slot         5
Touchscreen             3
Display_size            0
Operating System      166
Processor Core        262
Internal Storage       11
Primary Camera         29
Secondary Camera      367
Network Type            5
Bluetooth Version     415
Wi-Fi                 384
GPS Support           380
SIM Size              150
Battery Capacity        0
```

```
Width                104
Height               109
Weight                44
SIM Type.1             0
Hybrid Sim Slot.1      5
Dual Camera Lens     321
Color                  0
dtype: int64
```

```
# Calculate Null value percentage
null_percentage=round(100*(mobile.isnull().sum()/len(mobile.index)),2)
null_percentage
```

```
Brand                 0.00
Model Name            0.00
Price                 0.00
Rating                0.00
SIM Type              0.00
Hybrid Sim Slot       0.51
Touchscreen           0.30
Display_size          0.00
Operating System     16.87
Processor Core       26.63
Internal Storage      1.12
Primary Camera        2.95
Secondary Camera     37.30
Network Type          0.51
Bluetooth Version    42.17
Wi-Fi                39.02
GPS Support          38.62
SIM Size             15.24
Battery Capacity      0.00
Width                10.57
Height               11.08
Weight                4.47
SIM Type.1            0.00
Hybrid Sim Slot.1     0.51
Dual Camera Lens     32.62
Color                 0.00
dtype: float64
```

Drop duplicate values

```
mobile = mobile.drop_duplicates()
mobile.shape
```

```
(972, 26)
```

- 14 duplicate rows were removed from the dataset

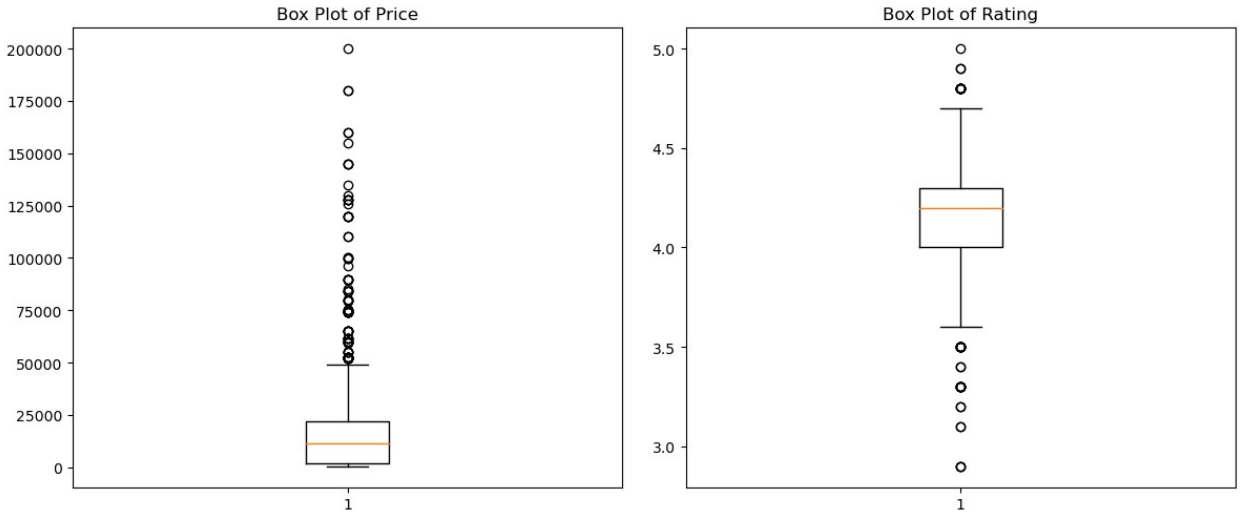
```
# Droping null values in few rows
#mobile.dropna(subset=['Width','Height','Weight'],inplace=True)

#Filling the null values with appropriate median and mode asper dtype
of column
warnings.filterwarnings("ignore")
for col in mobile.columns:
    if col in mobile.select_dtypes(include=[np.number]).columns:
    # Filling missing values in numerical columns with their
respective medians
        mobile[col] = mobile[col].fillna(mobile[col].mean())
    else:
    # Filling missing values in categorical columns with their
respective modes
        mode_val = mobile[col].mode().iloc[0]
        mobile[col] = mobile[col].fillna(mode_val)

#mobile =mobile.dropna()

# Checking for Null values again
mobile.isnull().sum()
```

```
Brand                0
Model Name           0
Price                0
Rating               0
SIM Type             0
Hybrid Sim Slot      0
Touchscreen          0
Display_size         0
Operating System     0
Processor Core       0
Internal Storage     0
Primary Camera       0
Secondary Camera     0
Network Type         0
Bluetooth Version    0
Wi-Fi                0
GPS Support          0
SIM Size             0
Battery Capacity     0
Width                0
Height               0
Weight               0
SIM Type.1           0
Hybrid Sim Slot.1    0
Dual Camera Lens     0
Color                0
dtype: int64
```

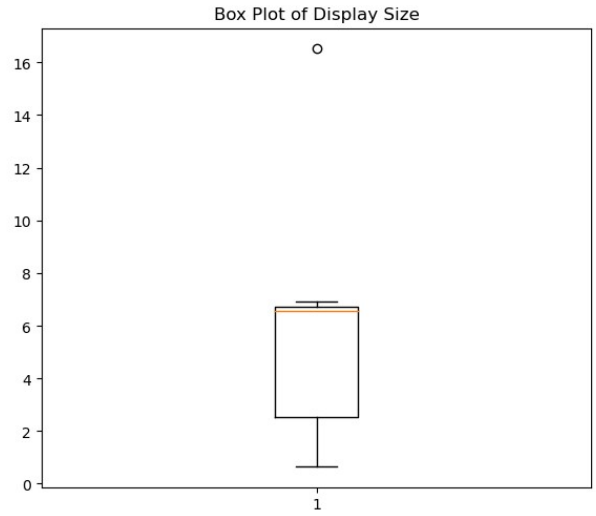
Data Cleaning is successful as finally we getridof all the null values from the data set

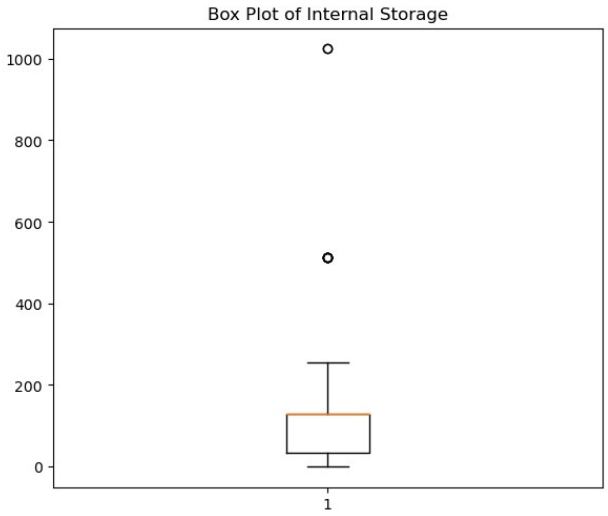
Check for outliers

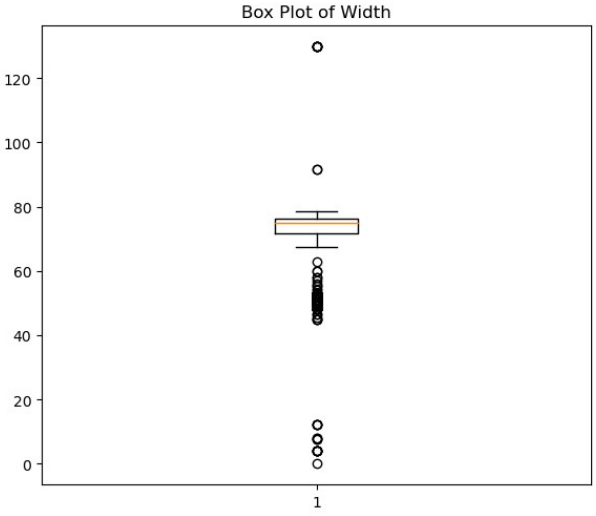
```
# Boxplot is used to detect outliers
fig, axes = plt.subplots(4, 2, figsize=(12, 20))

axes[0, 0].boxplot(mobile['Price'])
axes[0, 0].set_title('Box Plot of Price')

axes[0, 1].boxplot(mobile['Rating'])
axes[0, 1].set_title('Box Plot of Rating')

axes[1, 0].boxplot(mobile['Display_size'])
axes[1, 0].set_title('Box Plot of Display Size')

axes[1, 1].boxplot(mobile['Internal Storage'])
axes[1, 1].set_title('Box Plot of Internal Storage')

axes[2, 0].boxplot(mobile['Width'])
axes[2, 0].set_title('Box Plot of Width')

axes[2, 1].boxplot(mobile['Height'])
axes[2, 1].set_title('Box Plot of Height')

axes[3, 0].boxplot(mobile['Weight'])
axes[3, 0].set_title('Box Plot of Weight')

axes[3, 1].axis('off')


plt.tight_layout()
plt.show()
```

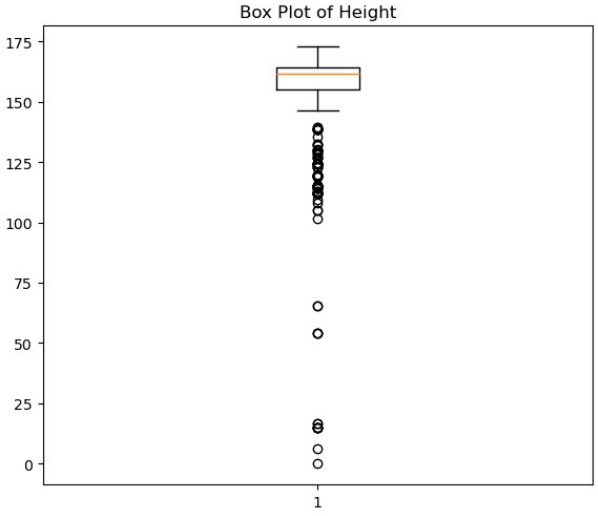

Box Plot of Price
200000
175000
150000
125000
100000
75000
50000
25000
0
1

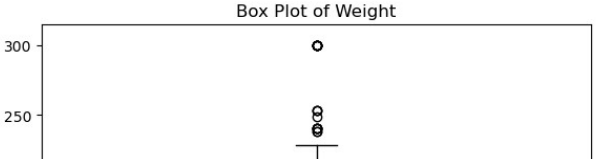
Box Plot of Rating
5.0
4.5
4.0
3.5
3.0
1


Box Plot of Display Size
16
14
12
10
8
6
4
2
0
1


Box Plot of Internal Storage
1000
800
600
400
200
0
1


Box Plot of Width
120
100
80
60
40
20
0
1

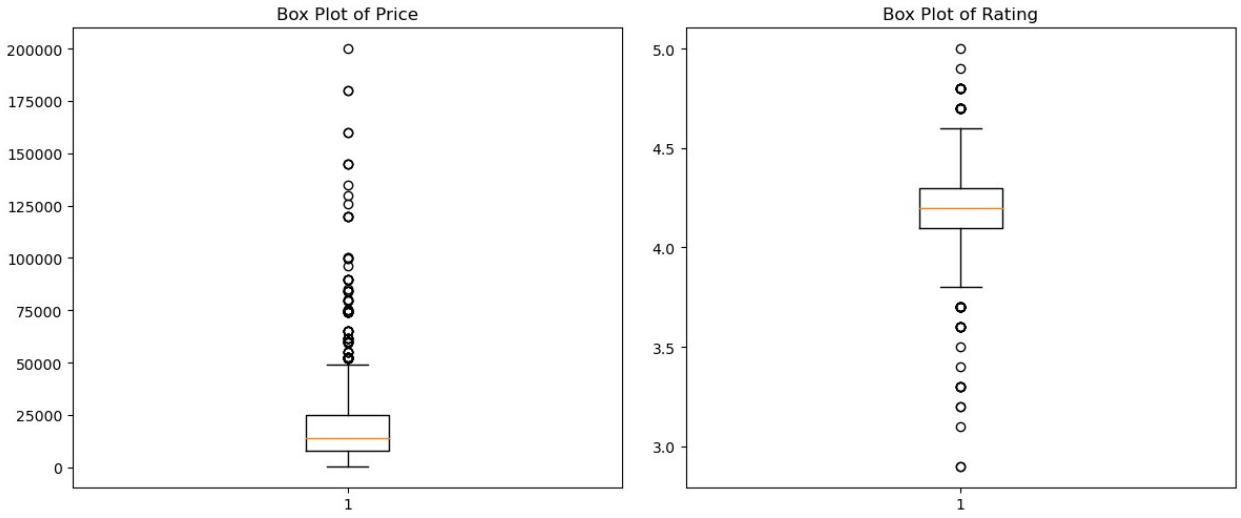

Box Plot of Height
175
150
125
100
75
50
25
0
1


Box Plot of Weight
300
250

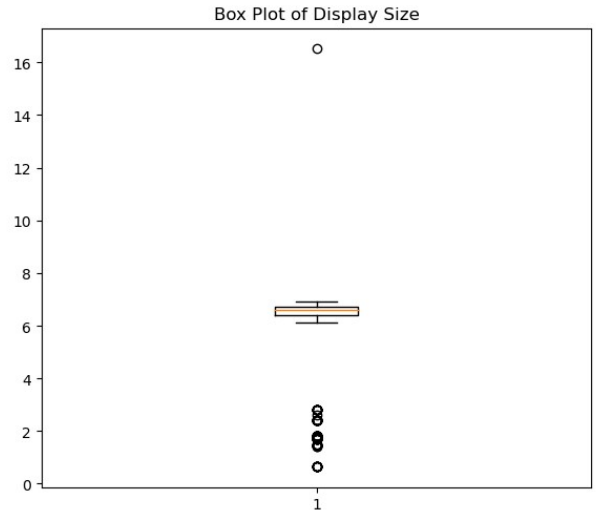
Removing Outliers

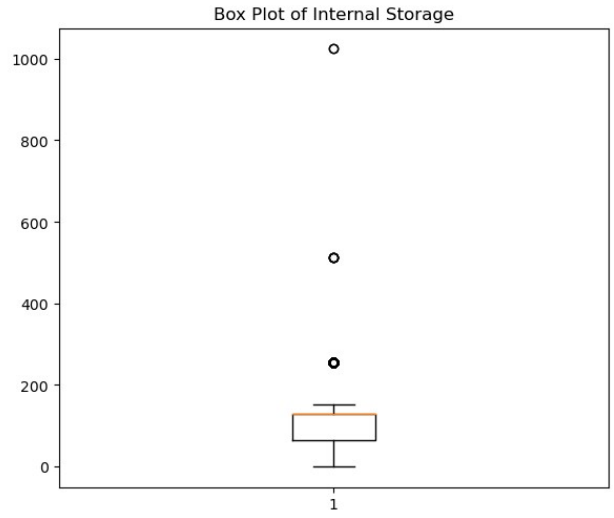
```
# Calculate the IQR for each of the columns
Q1_height = mobile['Height'].quantile(0.25)
Q3_height = mobile['Height'].quantile(0.75)
IQR_height = Q3_height - Q1_height

Q1_width = mobile['Width'].quantile(0.25)
Q3_width = mobile['Width'].quantile(0.75)
IQR_width = Q3_width - Q1_width

Q1_weight = mobile['Weight'].quantile(0.25)
Q3_weight = mobile['Weight'].quantile(0.75)
IQR_weight = Q3_weight - Q1_weight

# Define the lower and upper bounds for each column
lower_bound_height = Q1_height - 1.5 * IQR_height
upper_bound_height = Q3_height + 1.5 * IQR_height

lower_bound_width = Q1_width - 1.5 * IQR_width
upper_bound_width = Q3_width + 1.5 * IQR_width

lower_bound_weight = Q1_weight - 1.5 * IQR_weight
upper_bound_weight = Q3_weight + 1.5 * IQR_weight

# Remove outliers based on the bounds
mobile = mobile[(mobile['Height'] >= lower_bound_height) &
(mobile['Height'] <= upper_bound_height)]
mobile = mobile[(mobile['Width'] >= lower_bound_width) &
(mobile['Width'] <= upper_bound_width)]
mobile = mobile[(mobile['Weight'] >= lower_bound_weight) &
(mobile['Weight'] <= upper_bound_weight)]

# Boxplot is used to detect outliers
fig, axes = plt.subplots(4, 2, figsize=(12, 20))

# Create boxplots and set titles for each subplot
axes[0, 0].boxplot(mobile['Price'])
axes[0, 0].set_title('Box Plot of Price')

axes[0, 1].boxplot(mobile['Rating'])
axes[0, 1].set_title('Box Plot of Rating')

axes[1, 0].boxplot(mobile['Display_size'])
axes[1, 0].set_title('Box Plot of Display Size')

axes[1, 1].boxplot(mobile['Internal Storage'])
axes[1, 1].set_title('Box Plot of Internal Storage')

axes[2, 0].boxplot(mobile['Width'])
axes[2, 0].set_title('Box Plot of Width')
```

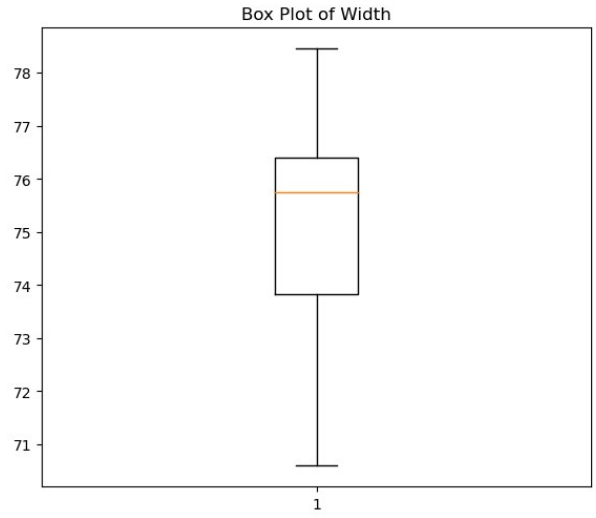
```
axes[2, 1].boxplot(mobile['Height'])
axes[2, 1].set_title('Box Plot of Height')

axes[3, 0].boxplot(mobile['Weight'])
axes[3, 0].set_title('Box Plot of Weight')

axes[3, 1].axis('off')


plt.tight_layout()
plt.show()
```

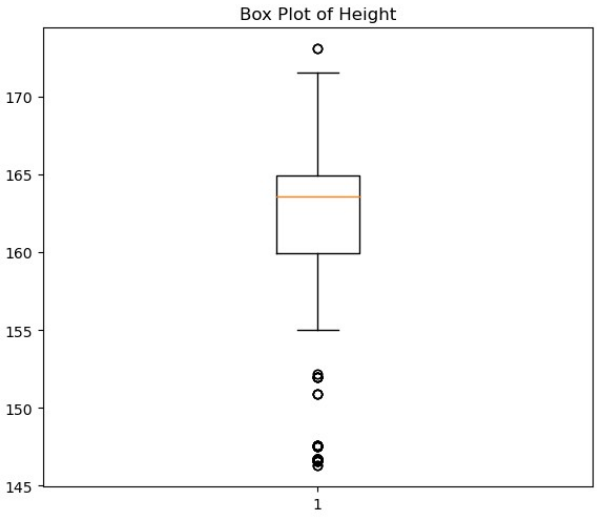

Box Plot of Price
200000
175000
150000
125000
100000
75000
50000
25000
0
1

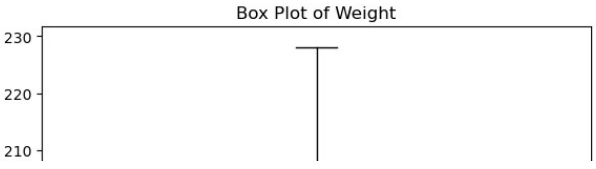
Box Plot of Rating
5.0
4.5
4.0
3.5
3.0
1


Box Plot of Display Size
16
14
12
10
8
6
4
2
0
1


Box Plot of Internal Storage
1000
800
600
400
200
0
1


Box Plot of Width
78
77
76
75
74
73
72
71
1


Box Plot of Height
170
165
160
155
150
145
1


Box Plot of Weight
230
220
210

3.Data Exploration

```
mobile.columns
```

```
Index(['Brand', 'Model Name', 'Price', 'Rating', 'SIM Type', 'Hybrid
Sim Slot',
       'Touchscreen', 'Display_size', 'Operating System', 'Processor
Core',
       'Internal Storage', 'Primary Camera', 'Secondary Camera',
       'Network Type', 'Bluetooth Version', 'Wi-Fi', 'GPS Support',
'SIM Size',
       'Battery Capacity', 'Width', 'Height', 'Weight', 'SIM Type.1',
       'Hybrid Sim Slot.1', 'Dual Camera Lens', 'Color'],
      dtype='object')
```

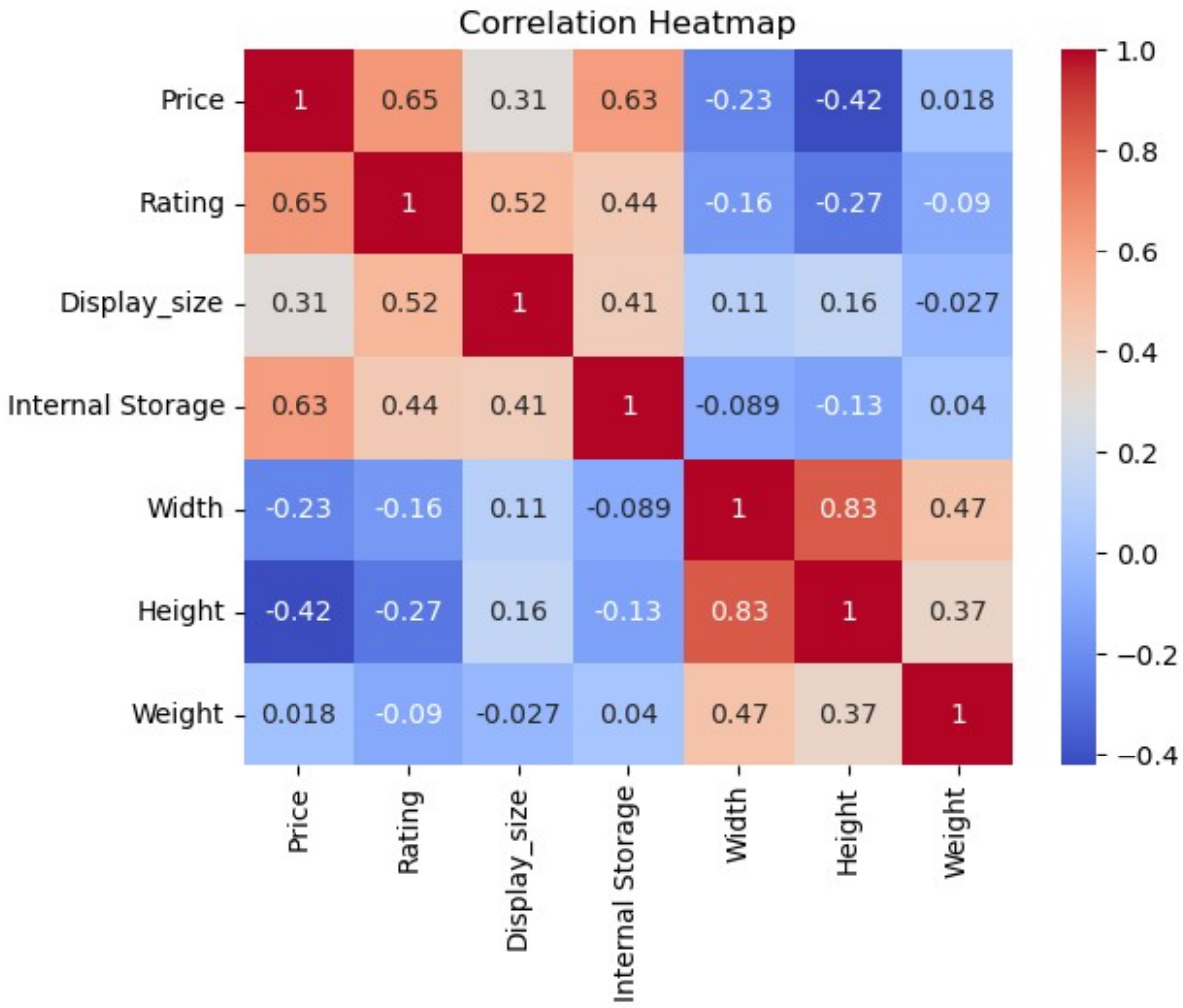
Summary

```
mobile.describe()
```

```
                Price      Rating  Display_size  Internal Storage
Width  \
count     778.000000  778.000000    778.000000        778.000000
778.000000
mean    21796.984576    4.205398      5.803436        126.308916
74.996629
std     26024.828639    0.258292      1.814605         92.579396
1.963075
min       597.000000    2.900000      0.660000          0.000000
70.600000
25%      7991.250000    4.100000      6.380000         64.000000
73.820000
50%     13999.000000    4.200000      6.590000        128.000000
75.750000
75%     24999.000000    4.300000      6.700000        128.000000
76.400000
max    199900.000000    5.000000     16.510000       1024.000000
78.460000

           Height      Weight
count  778.000000  778.000000
mean   161.308723  189.686804
std      6.249525   12.964685
min    146.300000  155.000000
25%    159.900000  181.000000
50%    163.600000  189.500000
75%    164.900000  199.800000
max    173.100000  228.000000
```

Visualization

```
correlation_matrix = mobile.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



4.Univariate Analysis

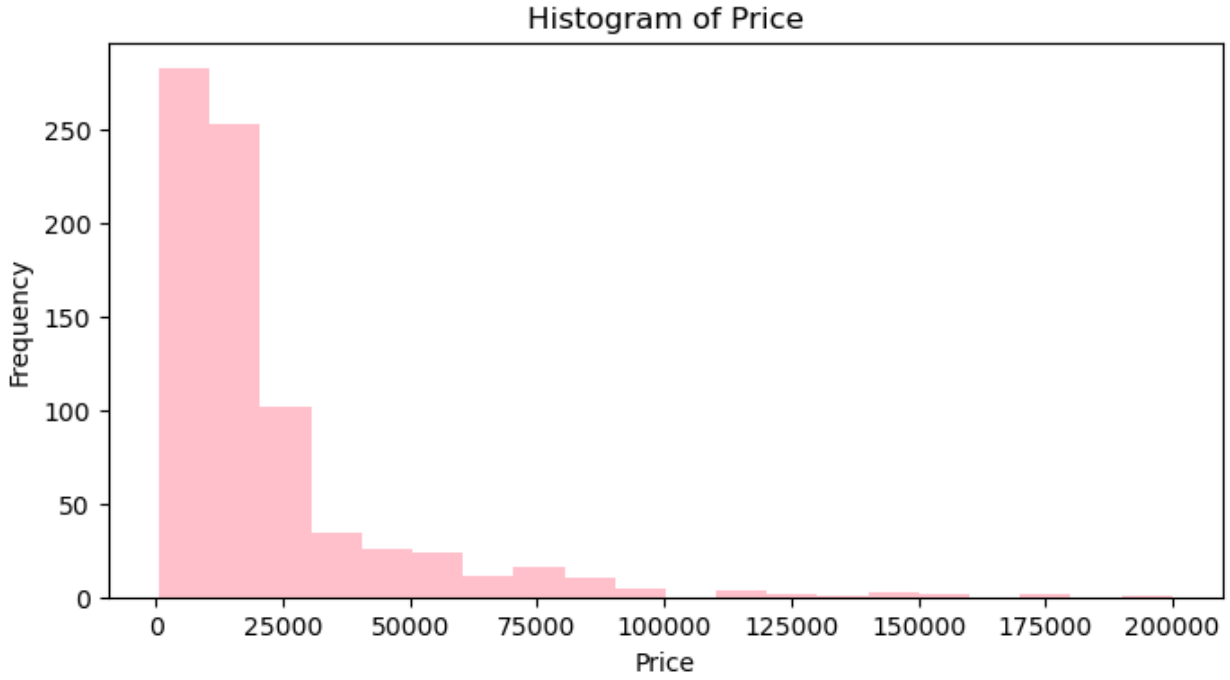
A) Numrerical Variables

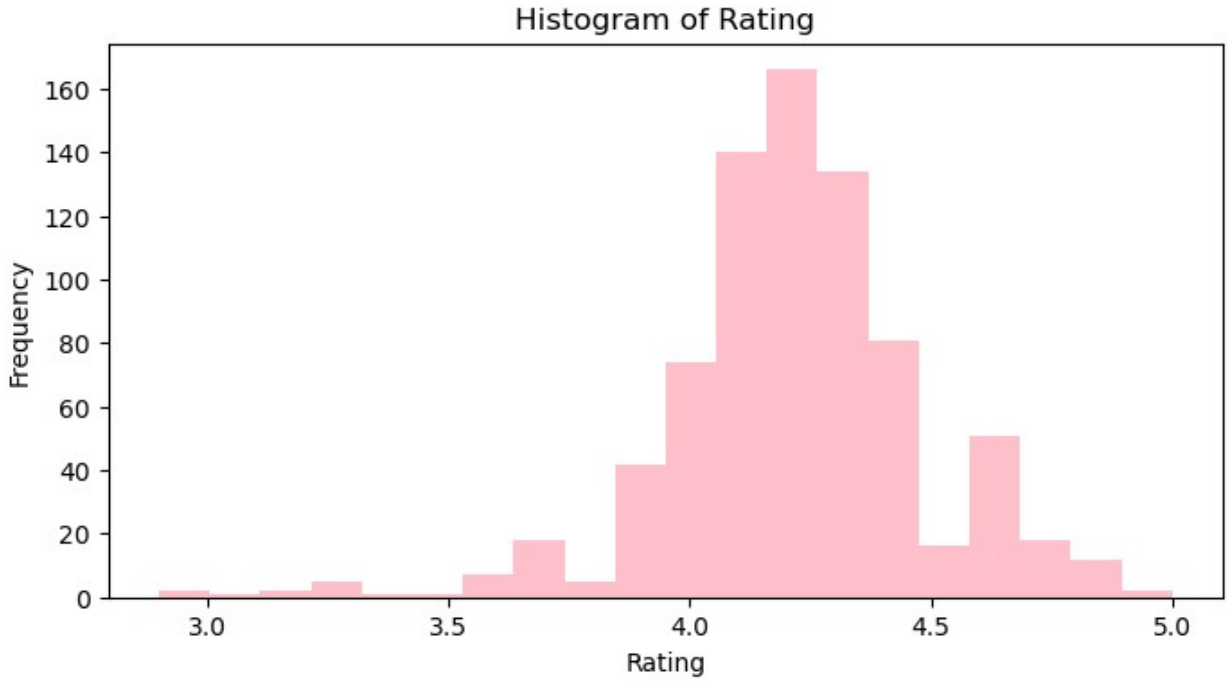
```
numerical_cols = mobile.select_dtypes(include=[np.number]).columns
```

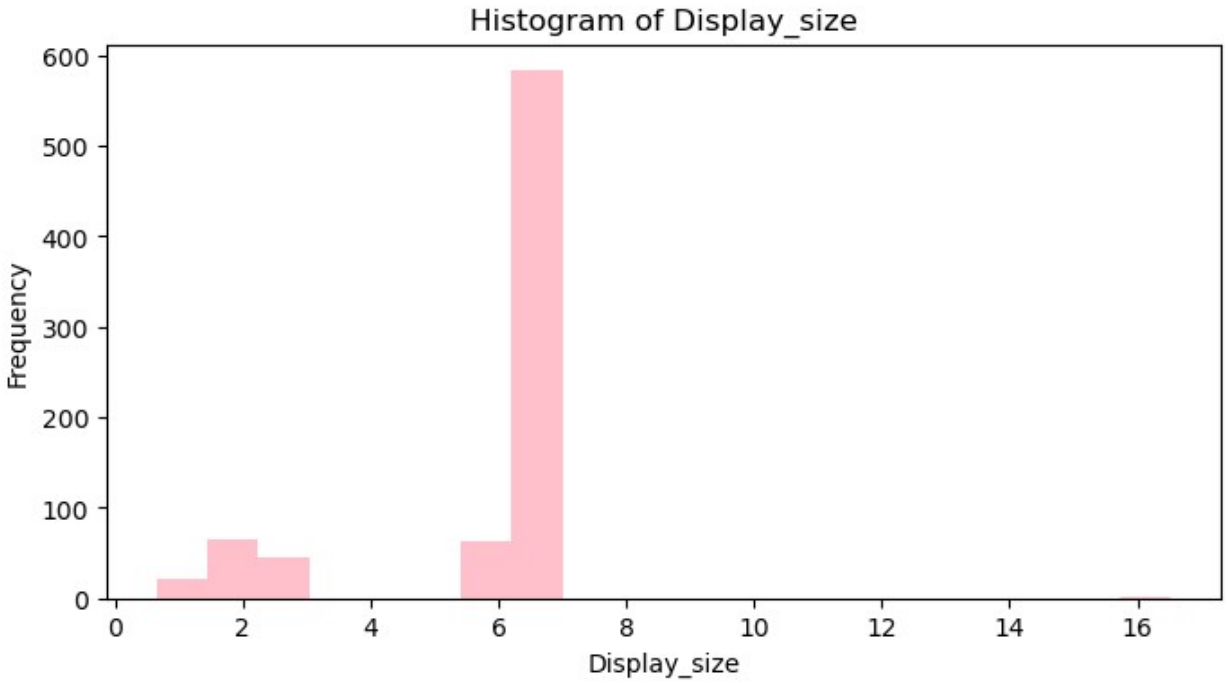
Histograms for numerical variables

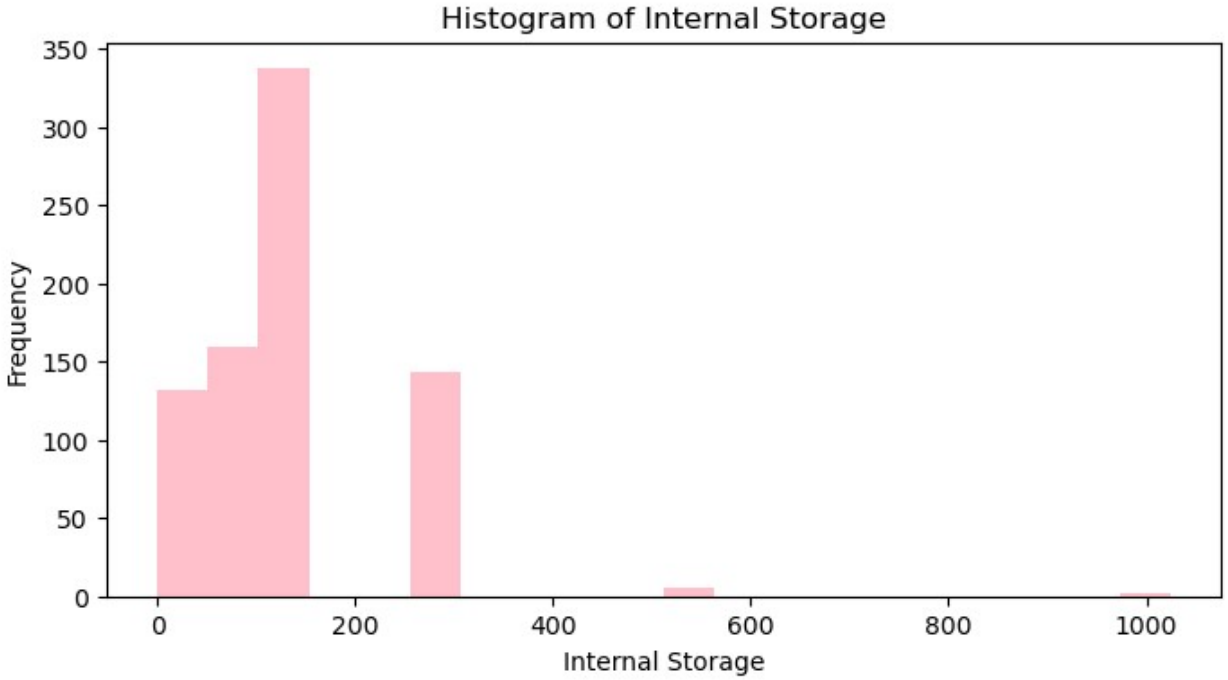
```
for col in numerical_cols:
    plt.figure(figsize=(8, 4))
    plt.hist(mobile[col], bins=20 ,color='pink' )
    plt.title(f'Histogram of {col}')
```

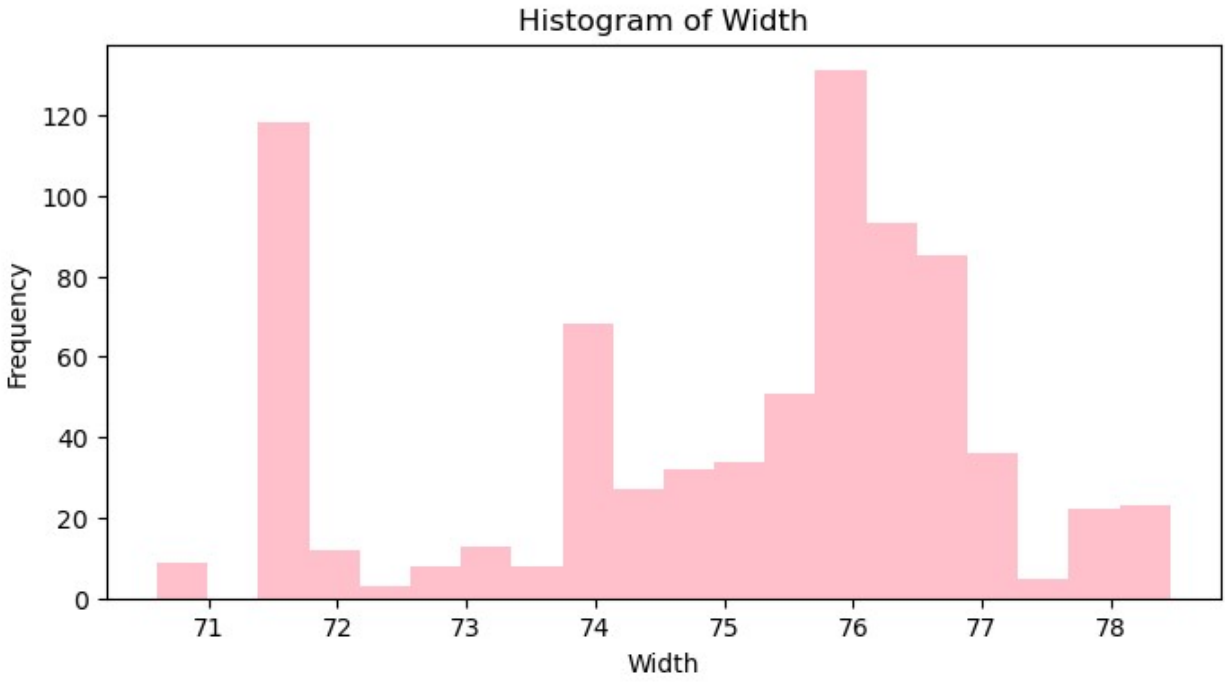
```
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.show()
```

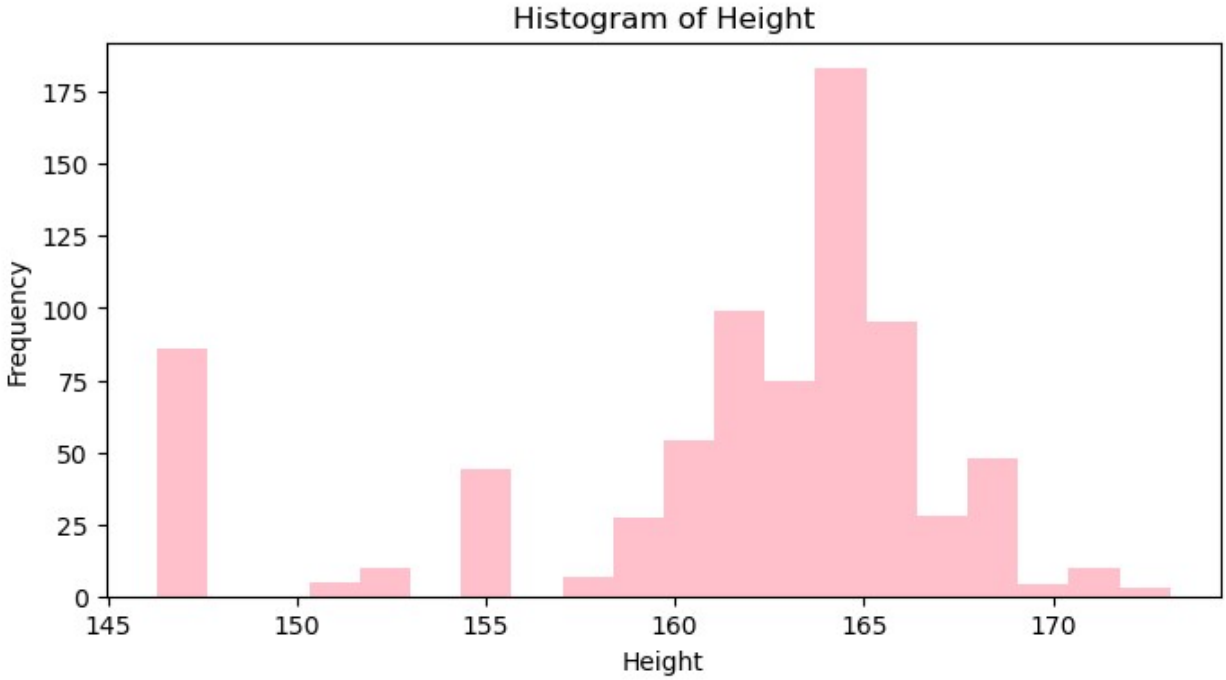


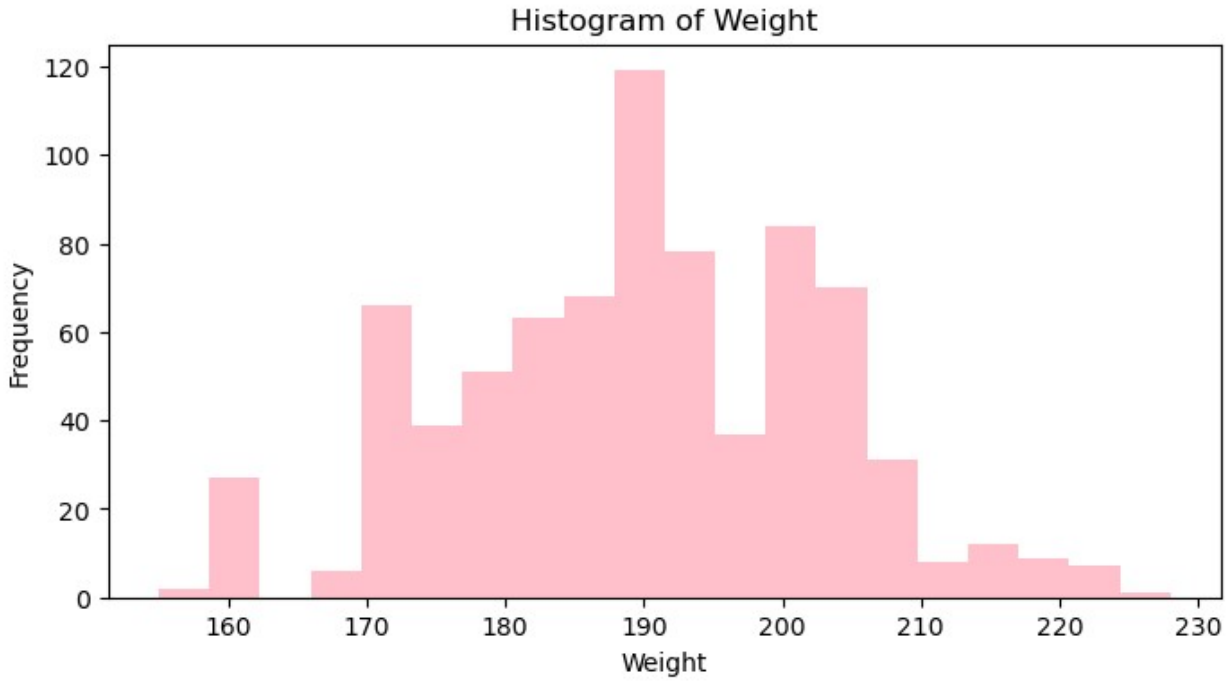



Histogram of Display_size
Frequency
Display_size
0
100
200
300
400
500
600
0
2
4
6
8
10
12
14
16


Histogram of Internal Storage
Frequency
Internal Storage
0
50
100
150
200
250
300
350
0
200
400
600
800
1000

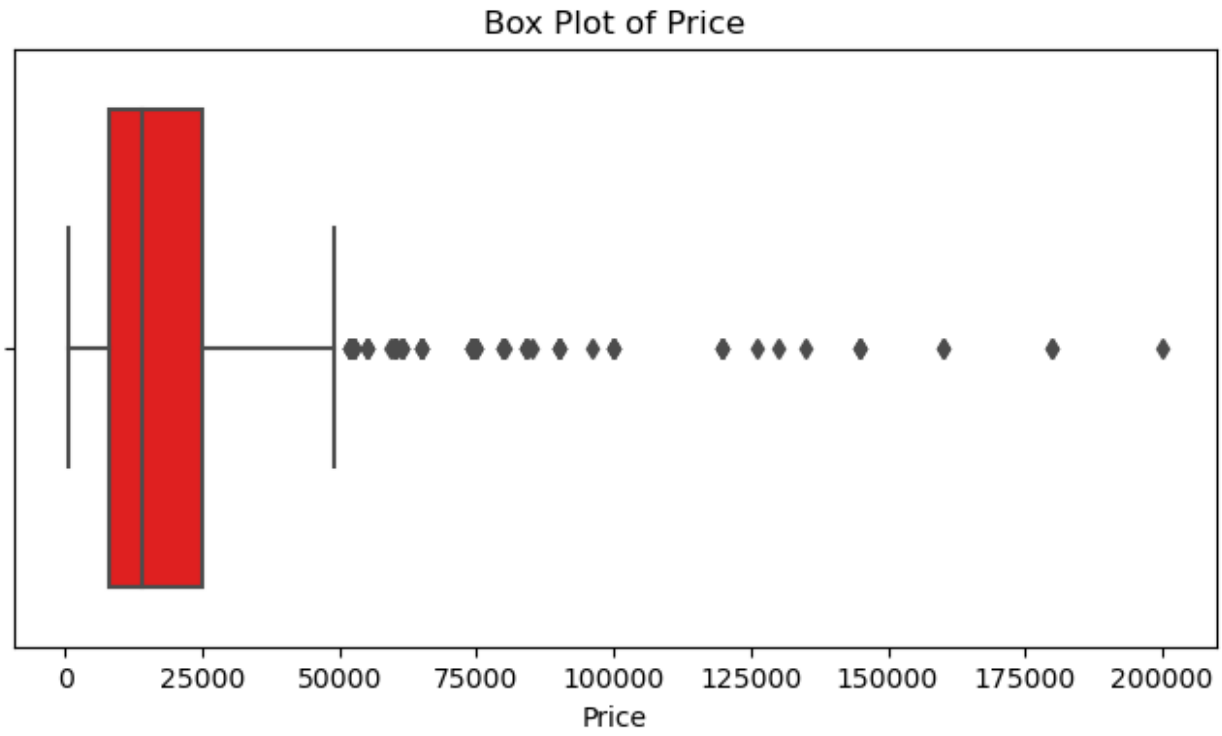

Histogram of Width
Frequency
Width
0
20
40
60
80
100
120
71
72
73
74
75
76
77
78

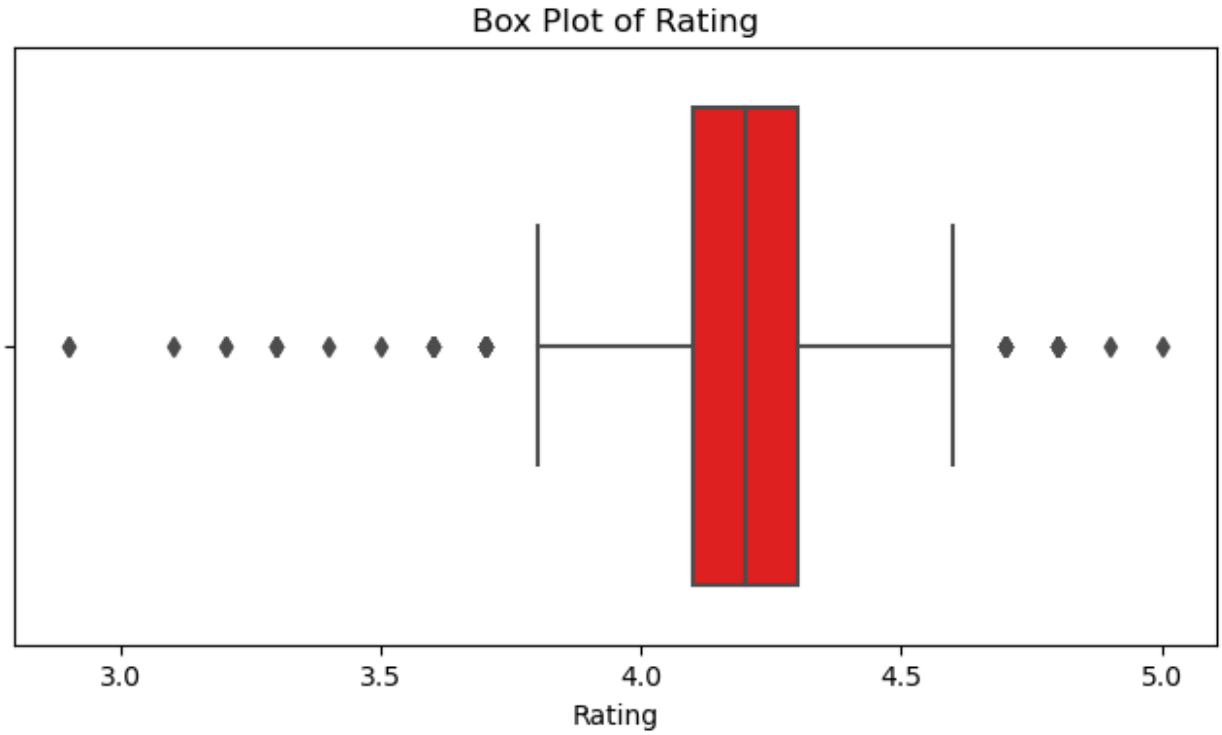

Histogram of Height
Frequency
Height
0
25
50
75
100
125
150
175
145
150
155
160
165
170

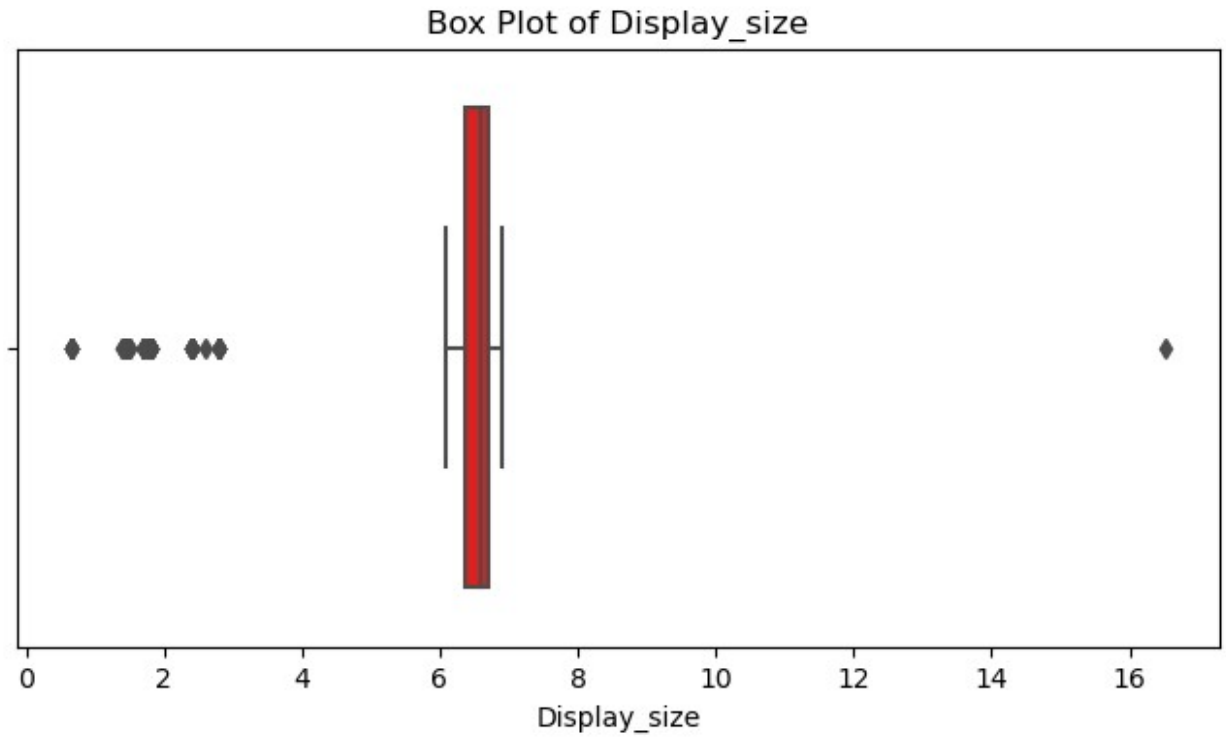


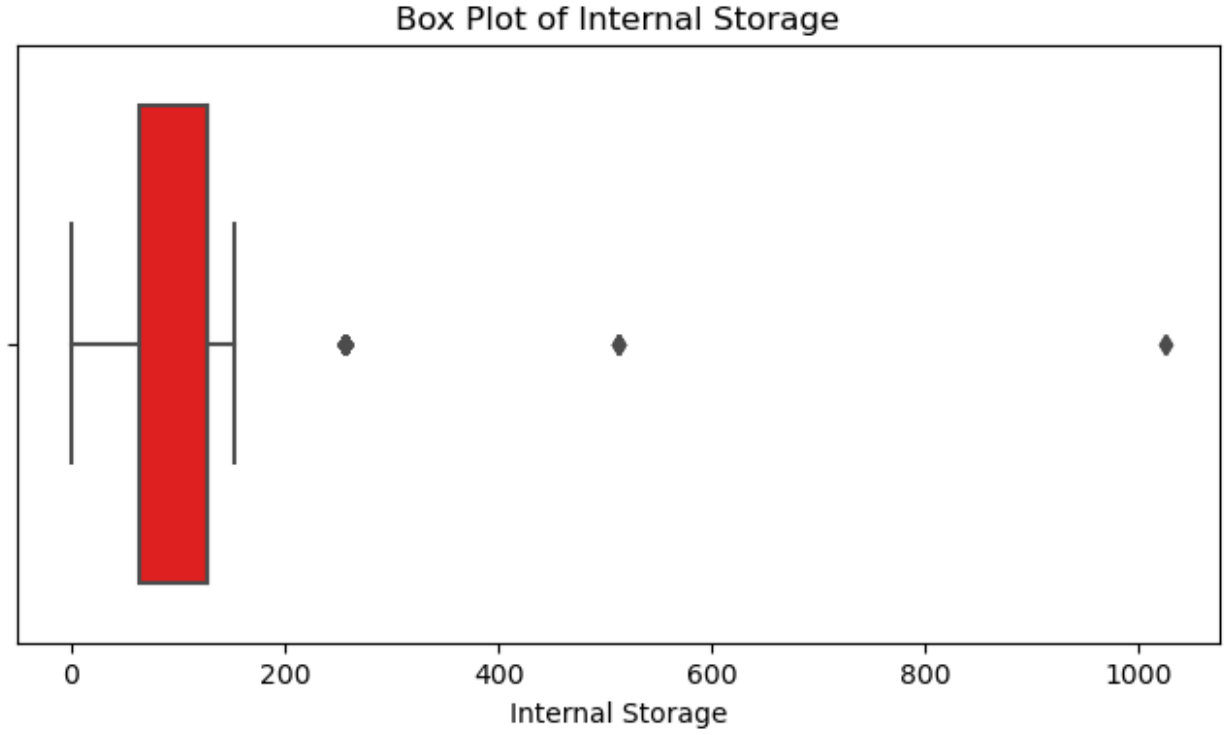
Box plots for numerical columns

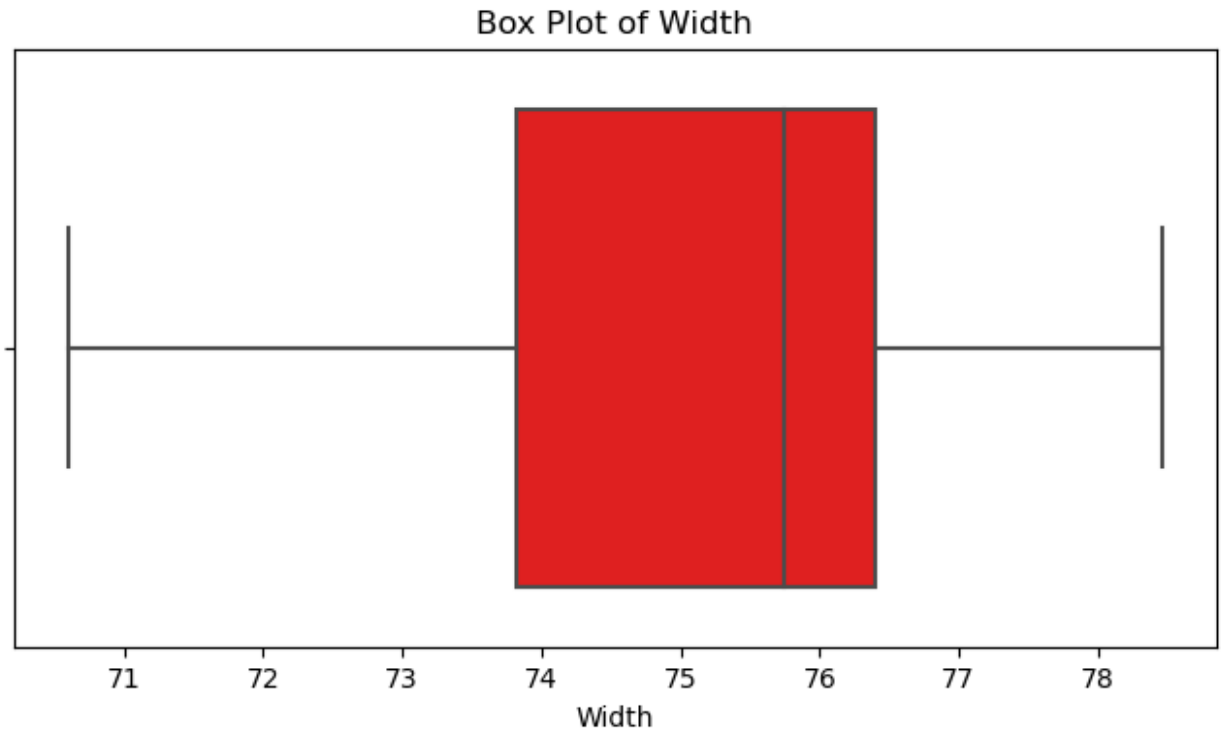
```
for col in numerical_cols:
    plt.figure(figsize=(8, 4))
    sns.boxplot(x=mobile[col],color='red')
    plt.title(f'Box Plot of {col}')
    plt.show()
```

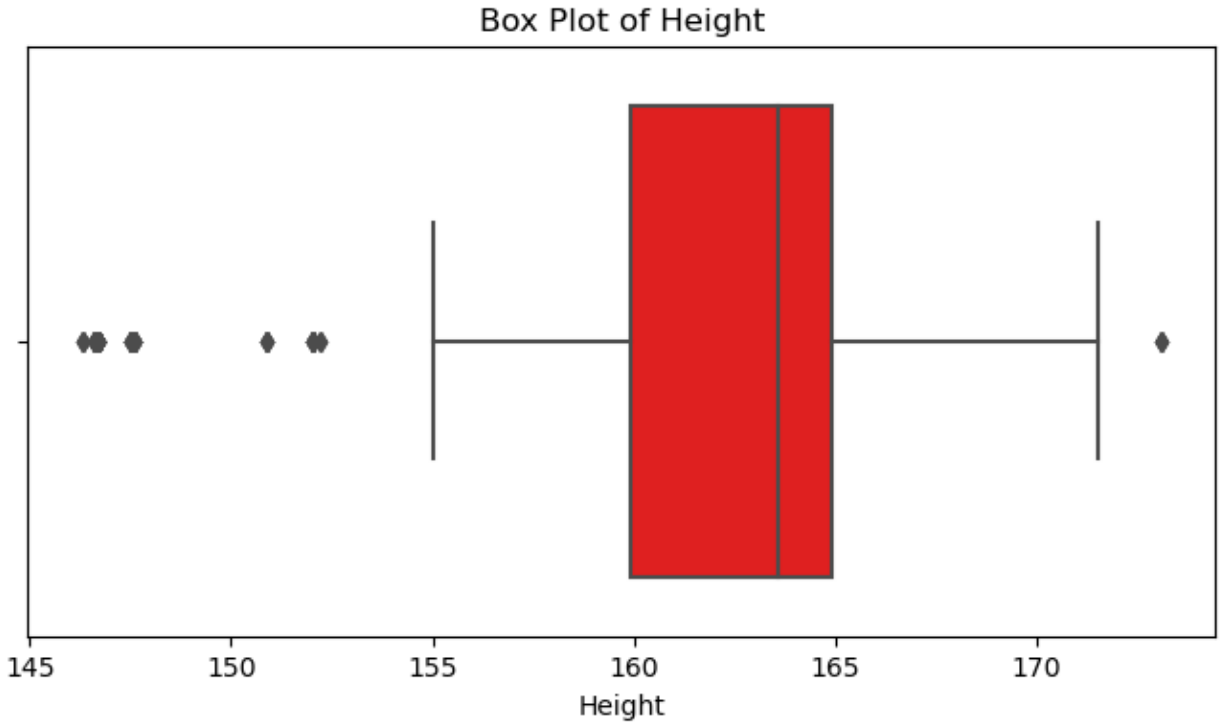

Box Plot of Price
0
25000
50000
75000
100000
125000
150000
175000
200000
Price

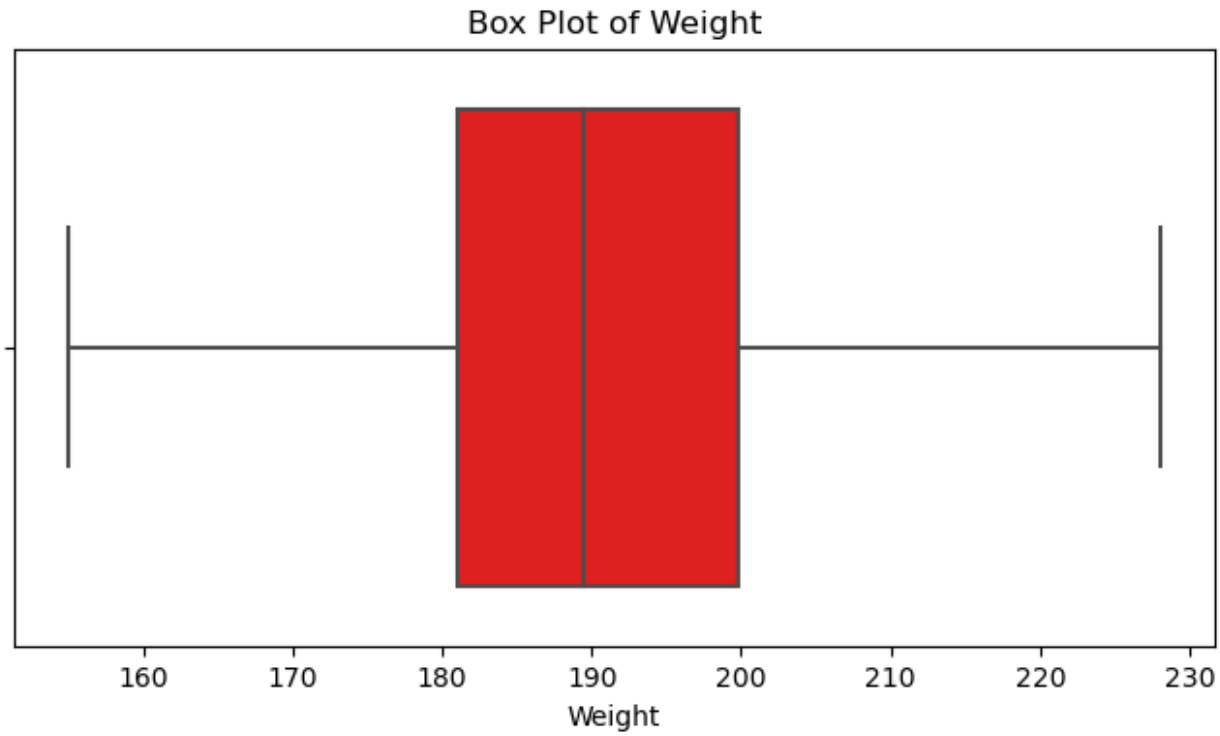

Box Plot of Rating
3.0
3.5
4.0
4.5
5.0
Rating


Box Plot of Display_size
0
2
4
6
8
10
12
14
16
Display_size


Box Plot of Internal Storage
0
200
400
600
800
1000
Internal Storage

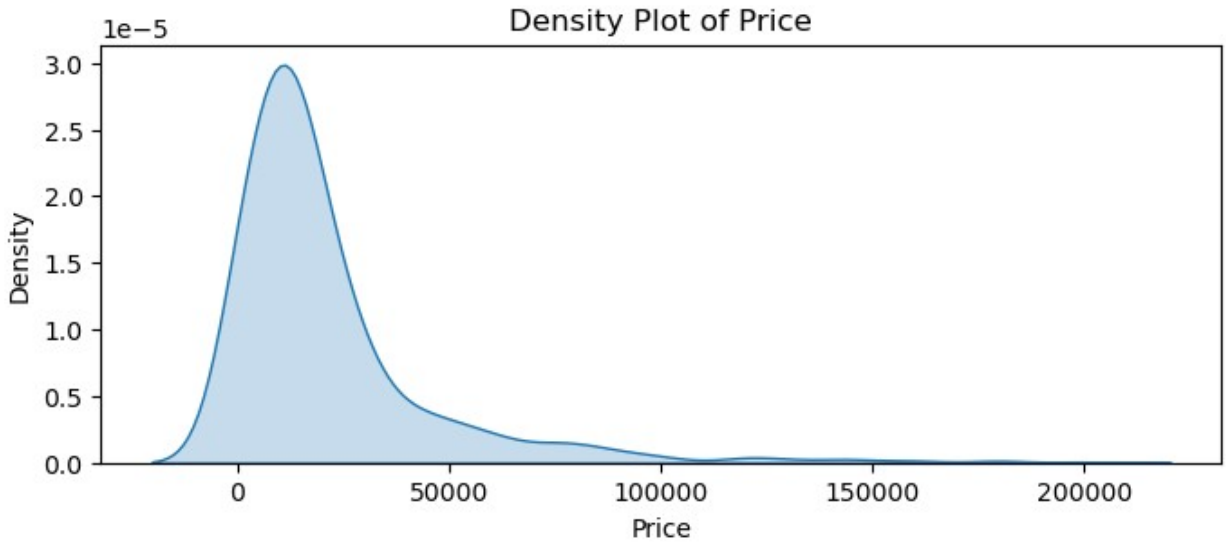

Box Plot of Width
71
72
73
74
75
76
77
78
Width

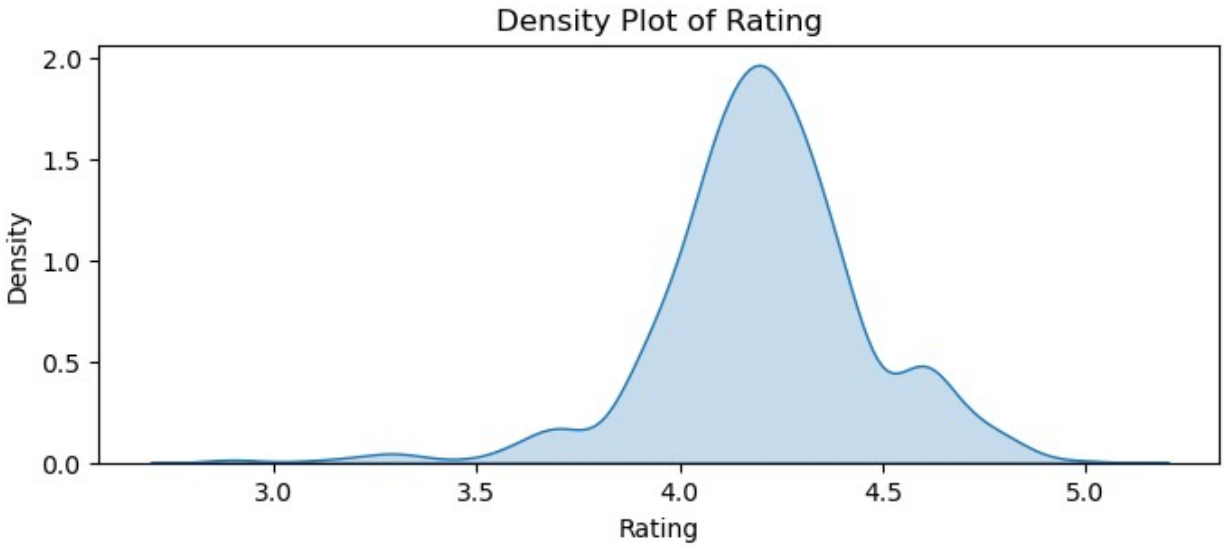

Box Plot of Height
145
150
155
160
165
170
Height

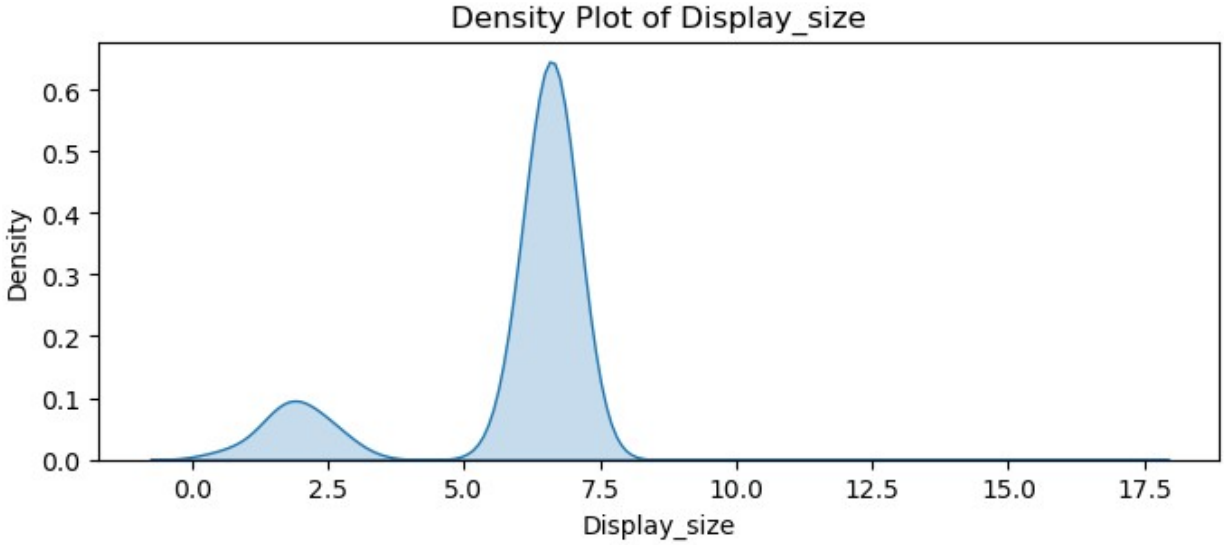


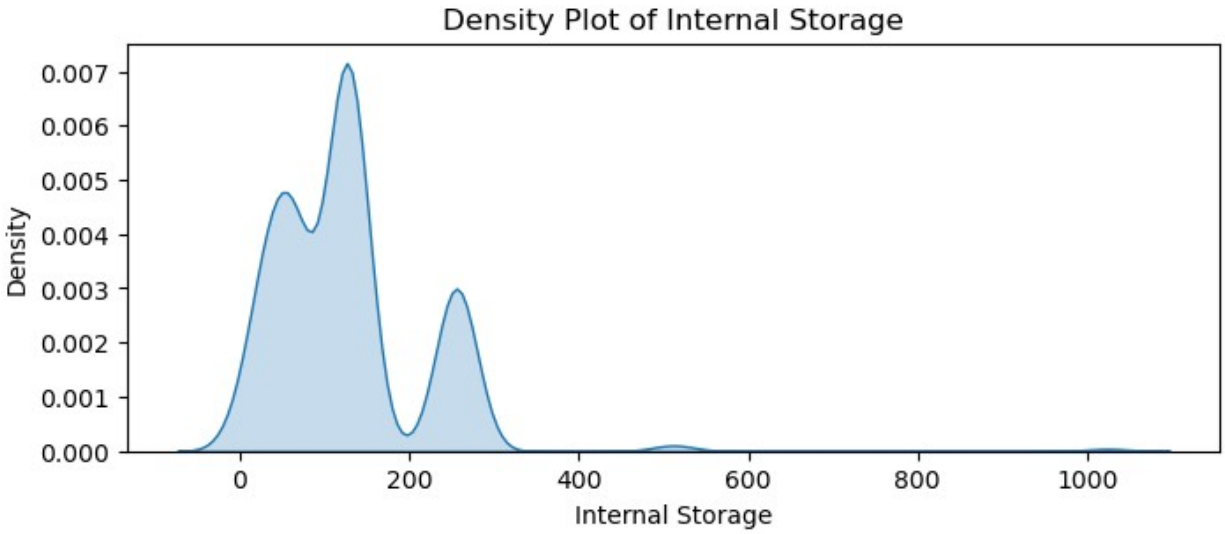
Density plots for numerical columns

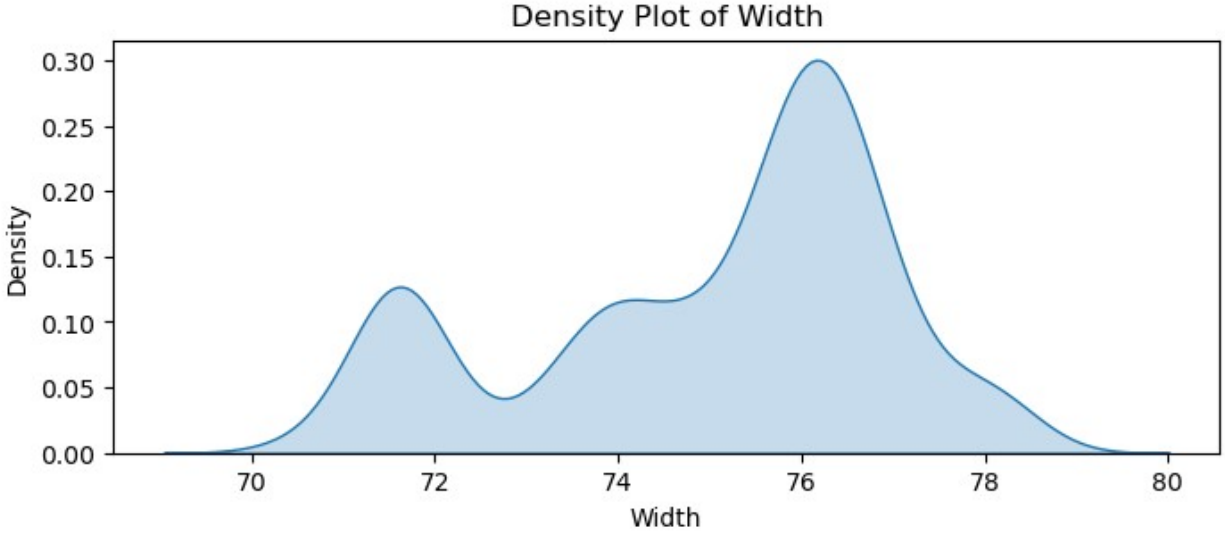
```
for col in numerical_cols:
    plt.figure(figsize=(8, 3))
    sns.kdeplot(mobile[col], shade=True)
    plt.title(f'Density Plot of {col}')
    plt.show()
```

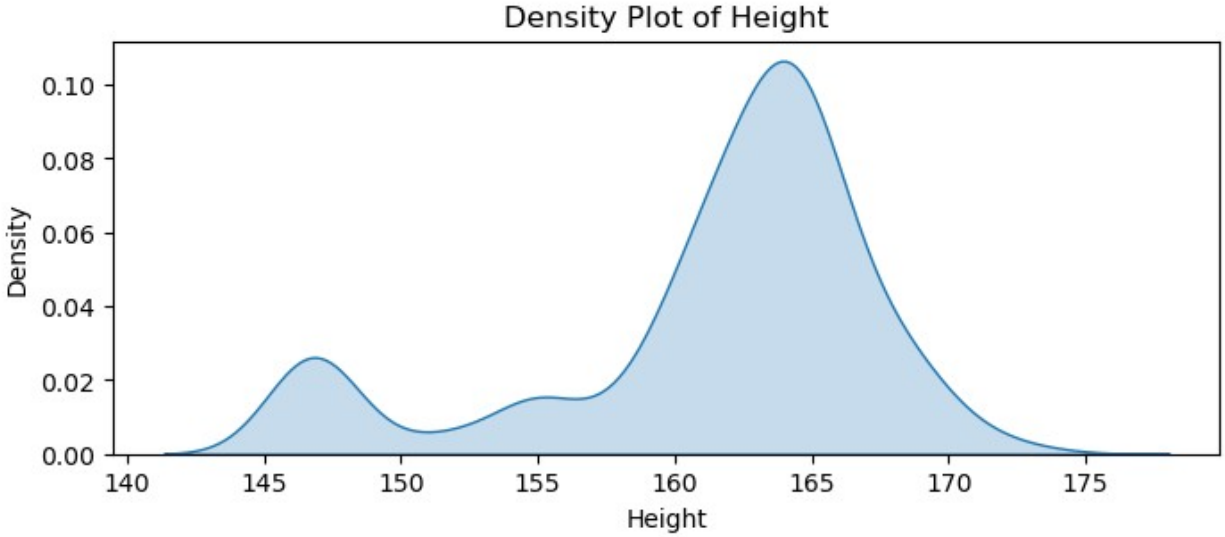


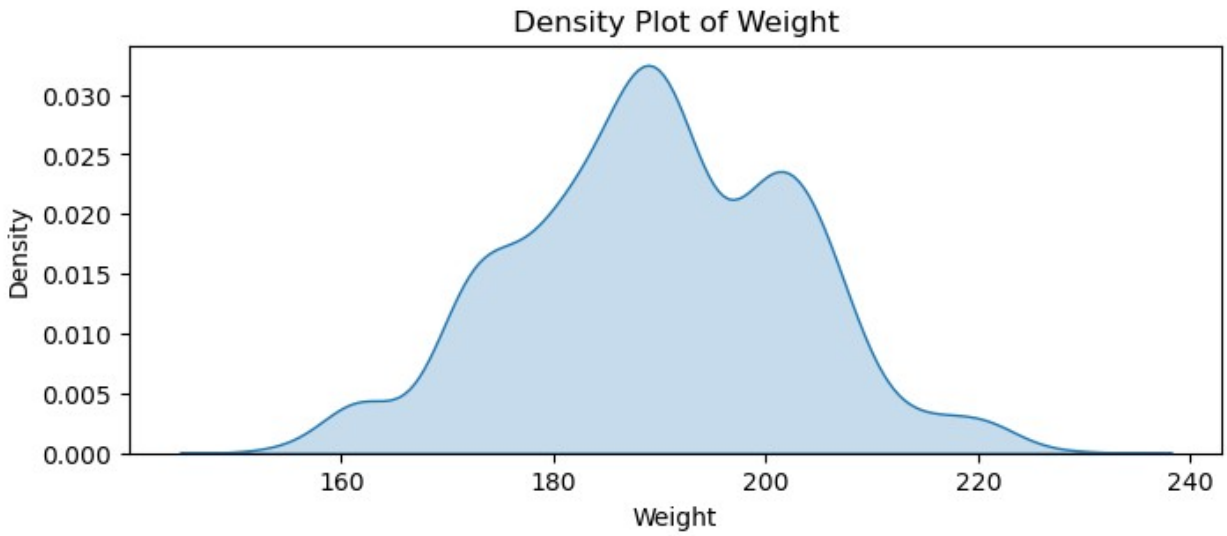

Density Plot of Rating
Density
Rating
2.0
1.5
1.0
0.5
0.0
3.0
3.5
4.0
4.5
5.0


Density Plot of Display_size
Density
Display_size
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Density Plot of Internal Storage
Density
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
0
200
400
600
800
1000
Internal Storage

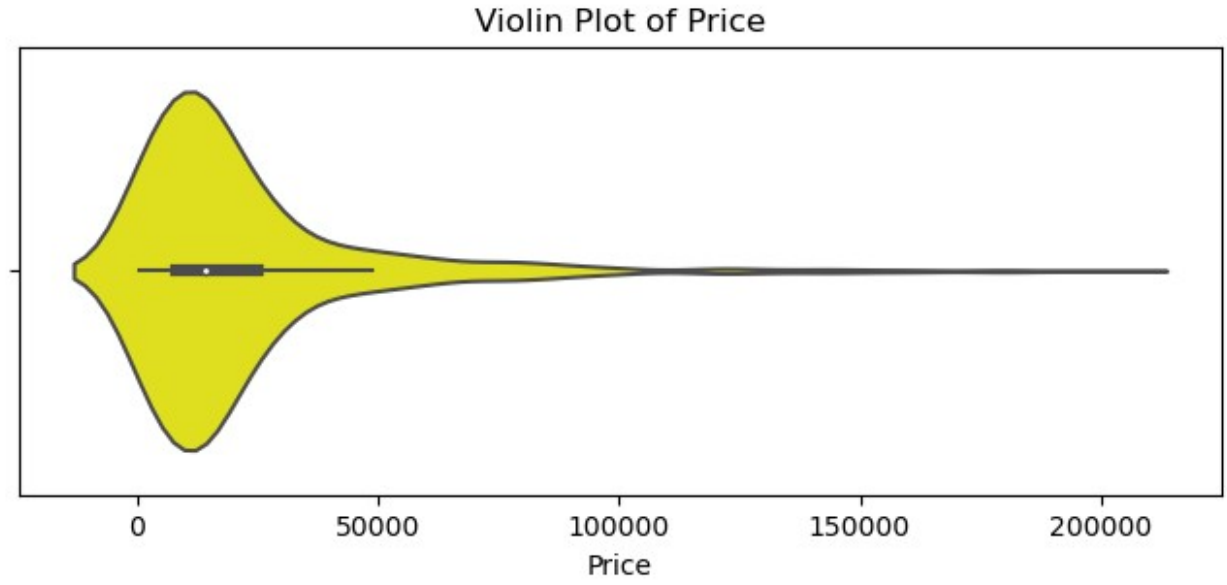

Density Plot of Width
Density
0.30
0.25
0.20
0.15
0.10
0.05
0.00
70
72
74
76
78
80
Width

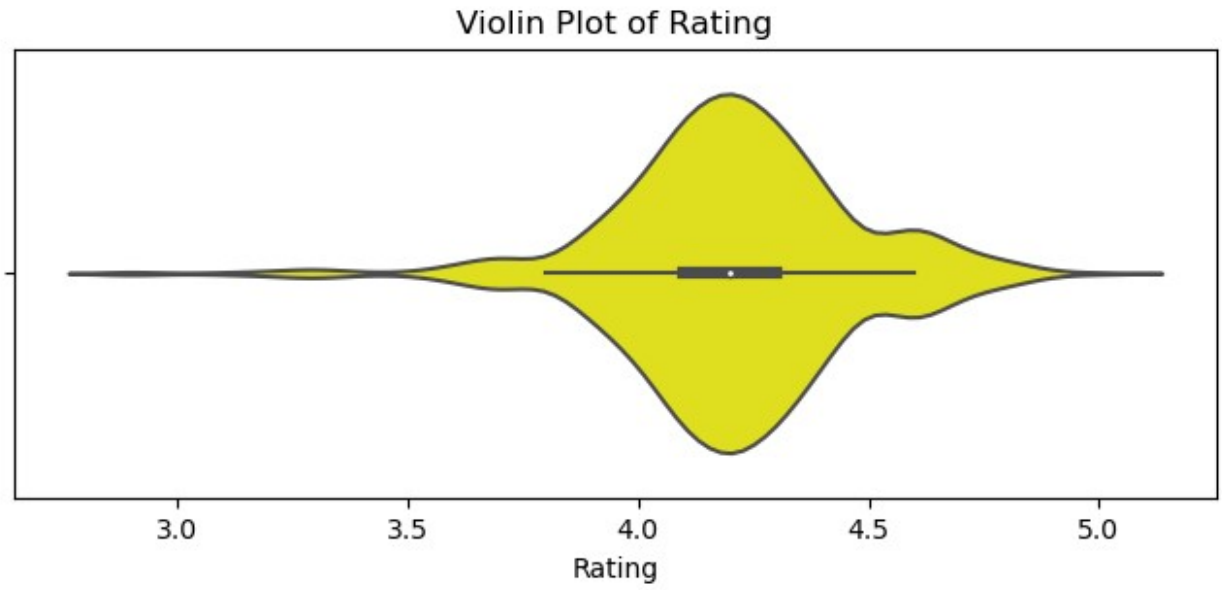

Density Plot of Height
Density
0.10
0.08
0.06
0.04
0.02
0.00
140
145
150
155
160
165
170
175
Height

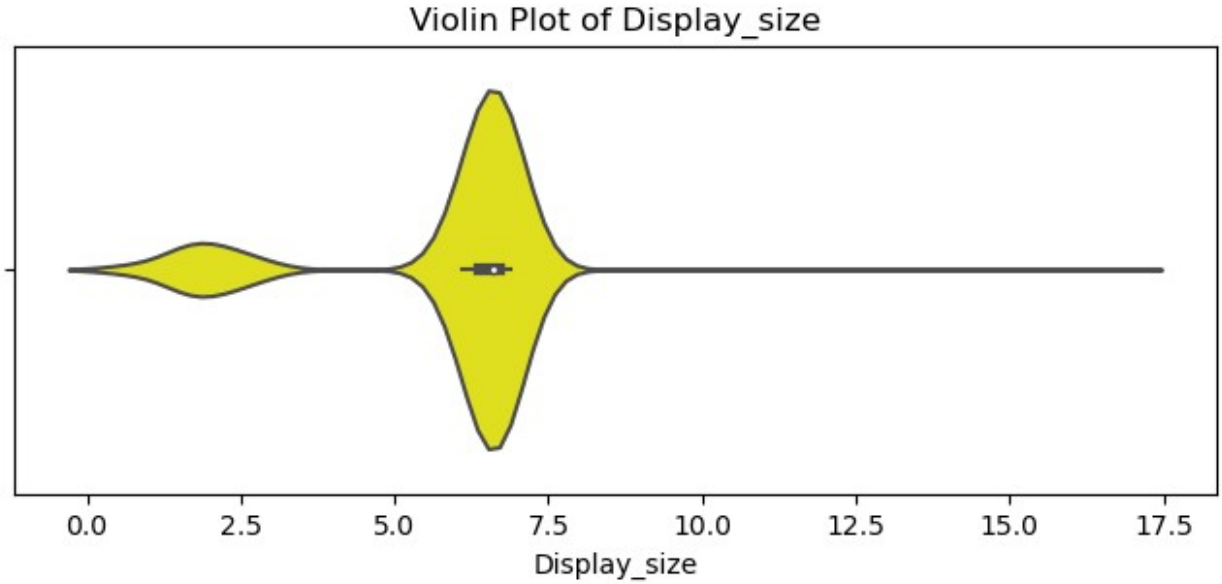


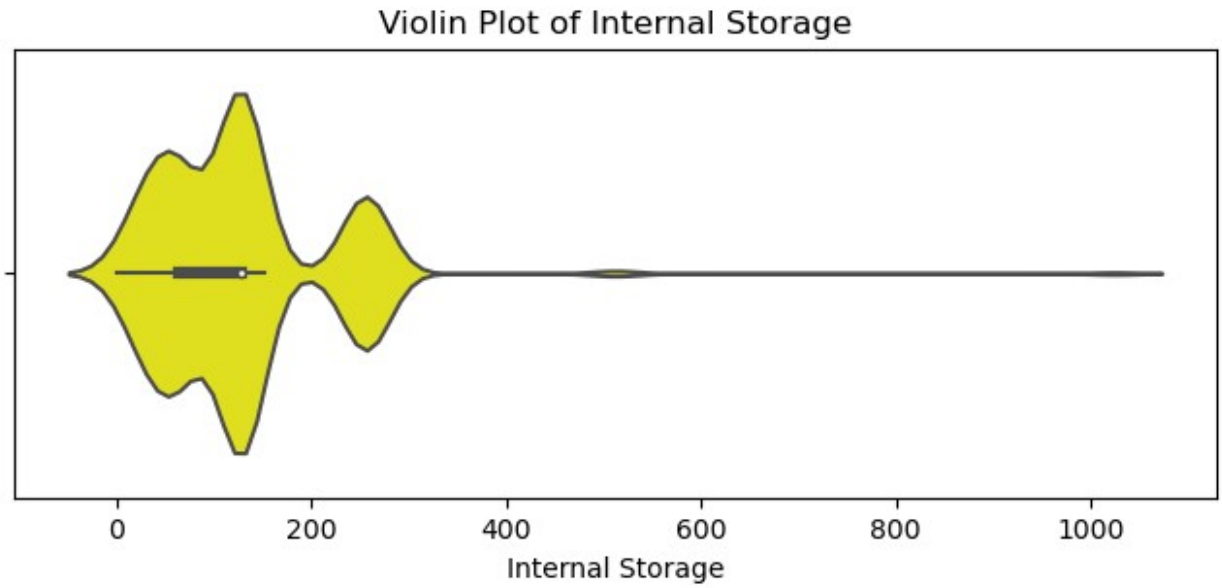
Violin plots for numerical columns

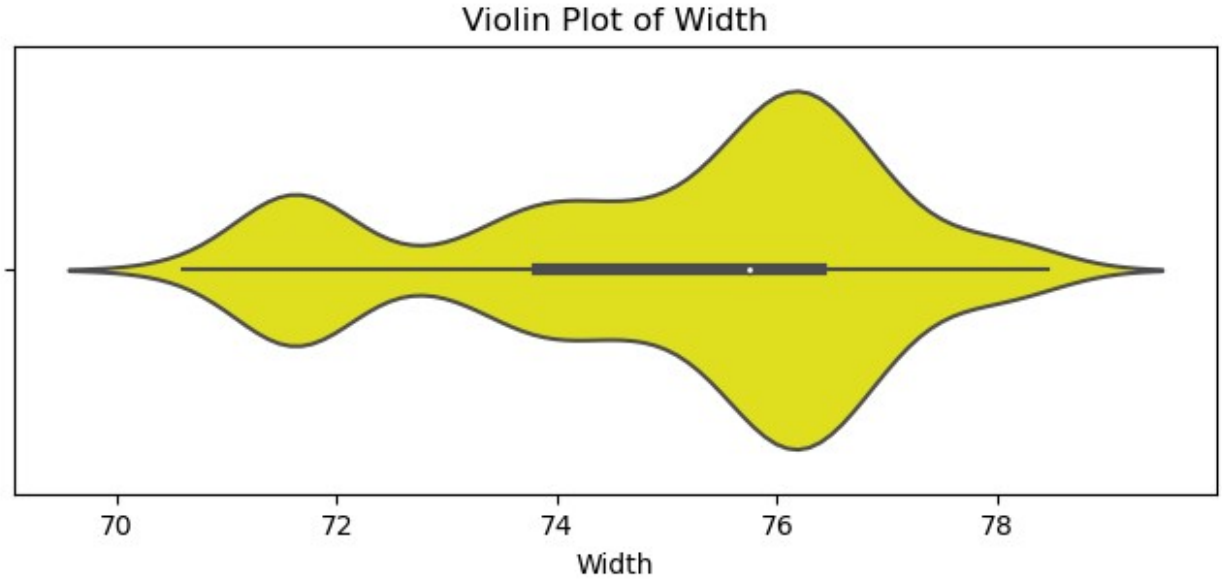
```
for col in numerical_cols:
    plt.figure(figsize=(8, 3))
    sns.violinplot(x=mobile[col],color='yellow')
    plt.title(f'Violin Plot of {col}')
    plt.show()
```

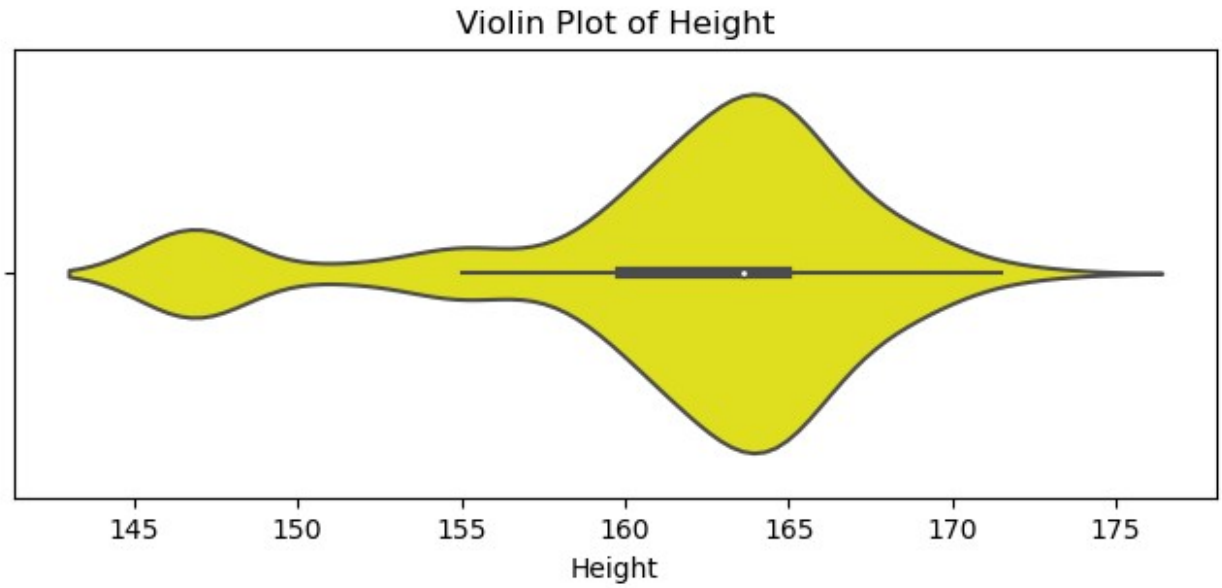


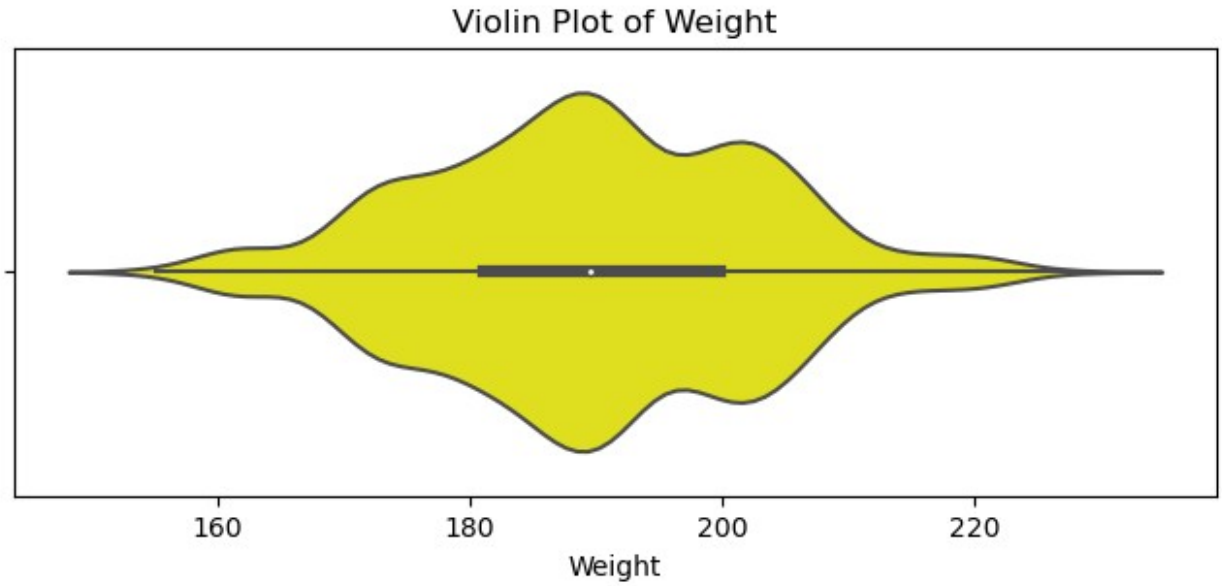

Violin Plot of Rating
3.0
3.5
4.0
4.5
5.0
Rating


Violin Plot of Display_size
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Display_size


Violin Plot of Internal Storage
0
200
400
600
800
1000
Internal Storage

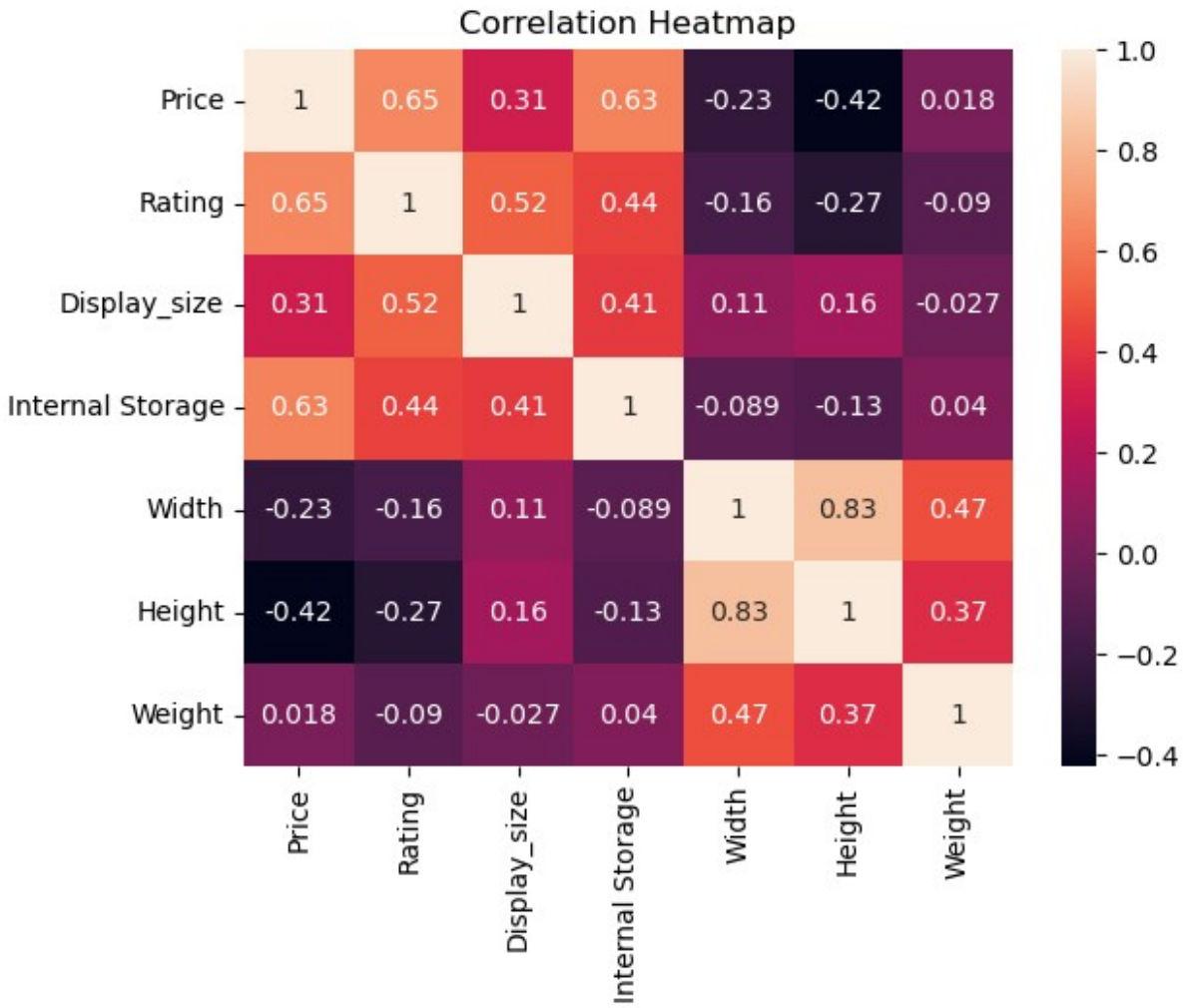

Violin Plot of Width
70
72
74
76
78
Width





Heat Map for Numerical columns

```
correlation_matrix = mobile.corr()
sns.heatmap(correlation_matrix, annot=True)
plt.title('Correlation Heatmap')
plt.show()
```



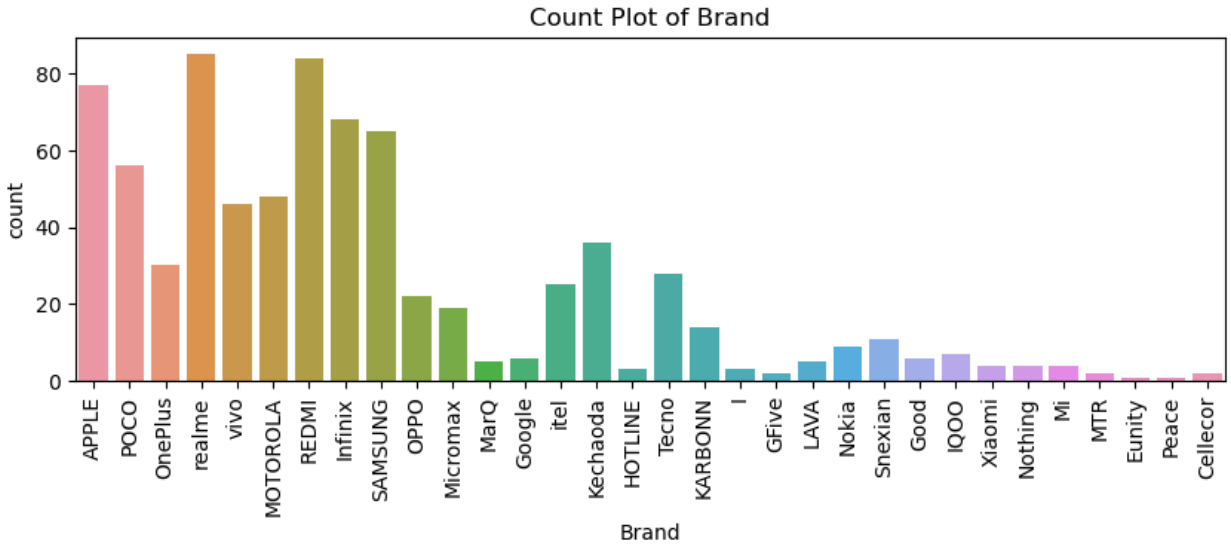
B)Categorical Variables

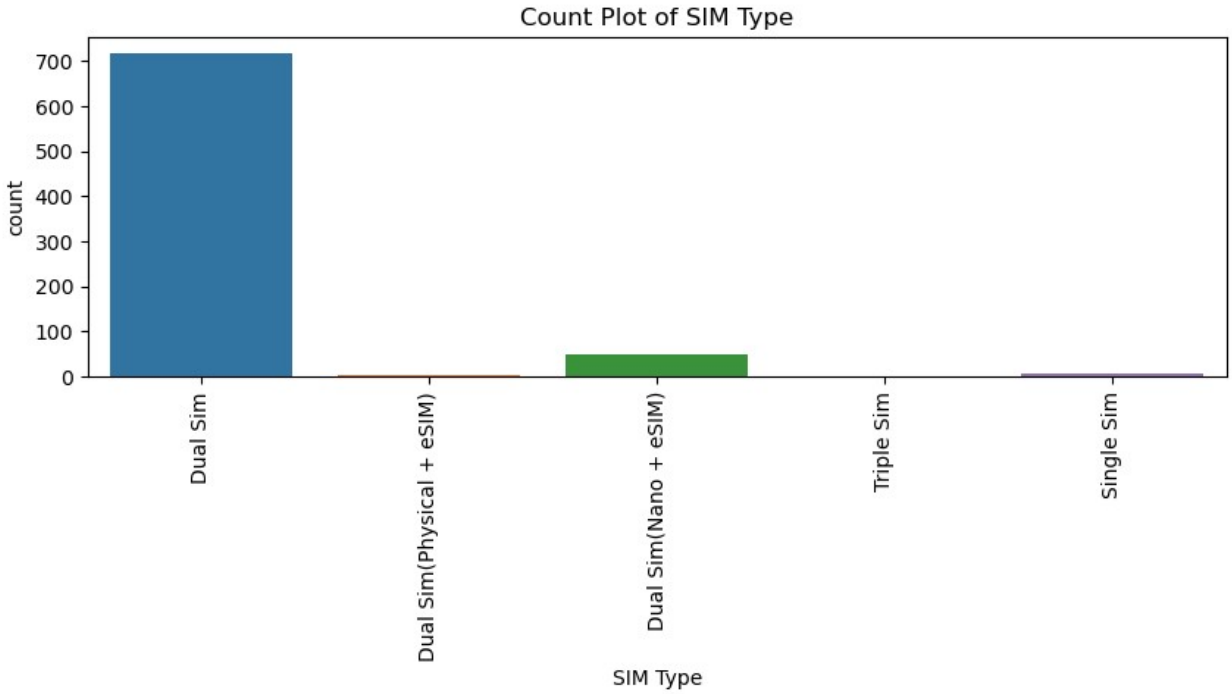
```
categorical_cols = mobile.select_dtypes(exclude=[np.number]).columns
categorical_cols = ['Brand',  'SIM Type', 'Hybrid Sim Slot',
'Touchscreen',
       'Operating System', 'Processor Core', 'Primary Camera',
       'Secondary Camera', 'Network Type', 'Bluetooth Version', 'Wi-
Fi',
       'GPS Support', 'SIM Size', 'Battery Capacity', 'SIM Type.1',
       'Hybrid Sim Slot.1', 'Dual Camera Lens', 'Color']
```

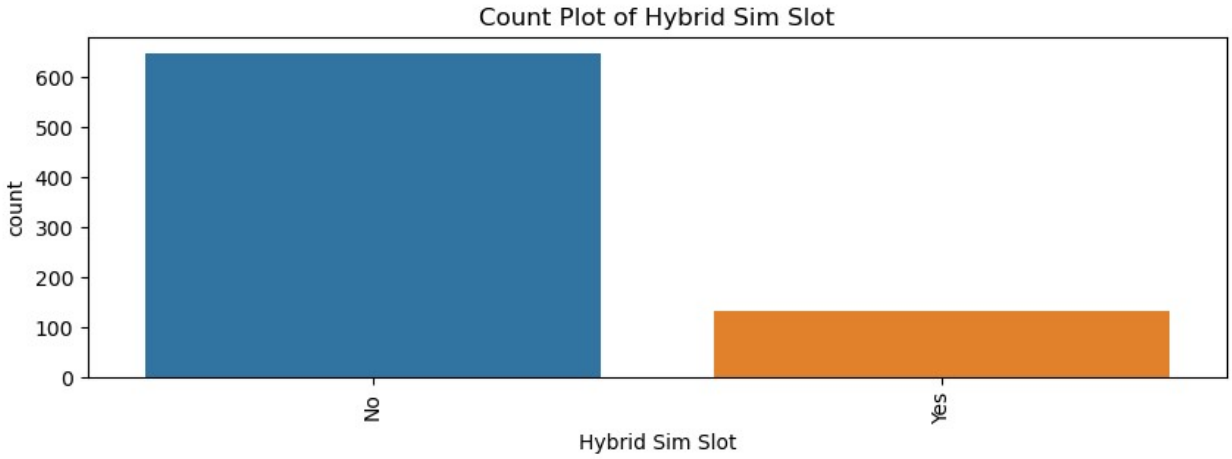
Count plot for categorical columns

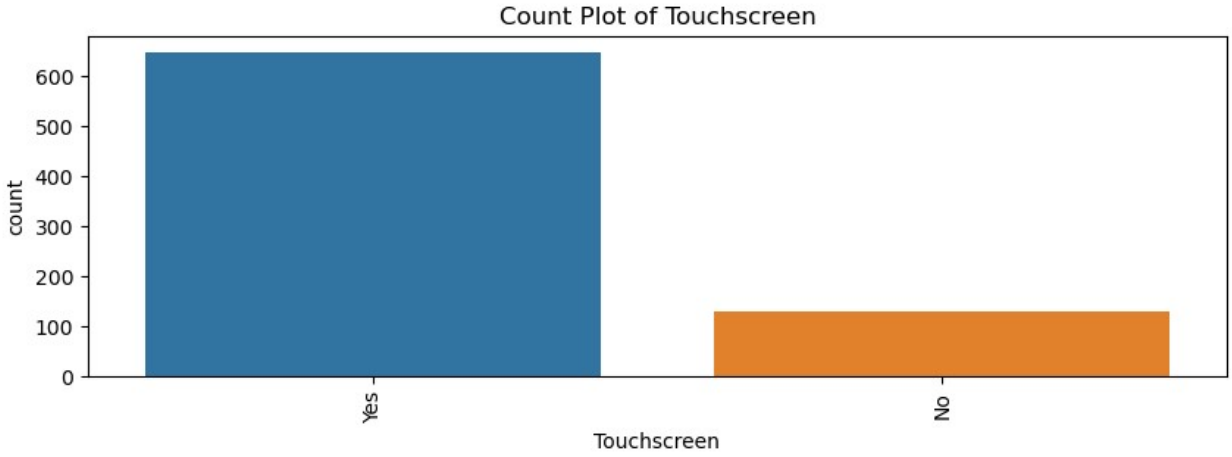
```
categorical_cols = ['Brand',  'SIM Type', 'Hybrid Sim Slot',
'Touchscreen',
       'Operating System', 'Processor Core',
       'Secondary Camera', 'Network Type', 'Bluetooth Version', 'Wi-
Fi',
       'GPS Support', 'SIM Size', 'Battery Capacity', 'SIM Type.1',
```

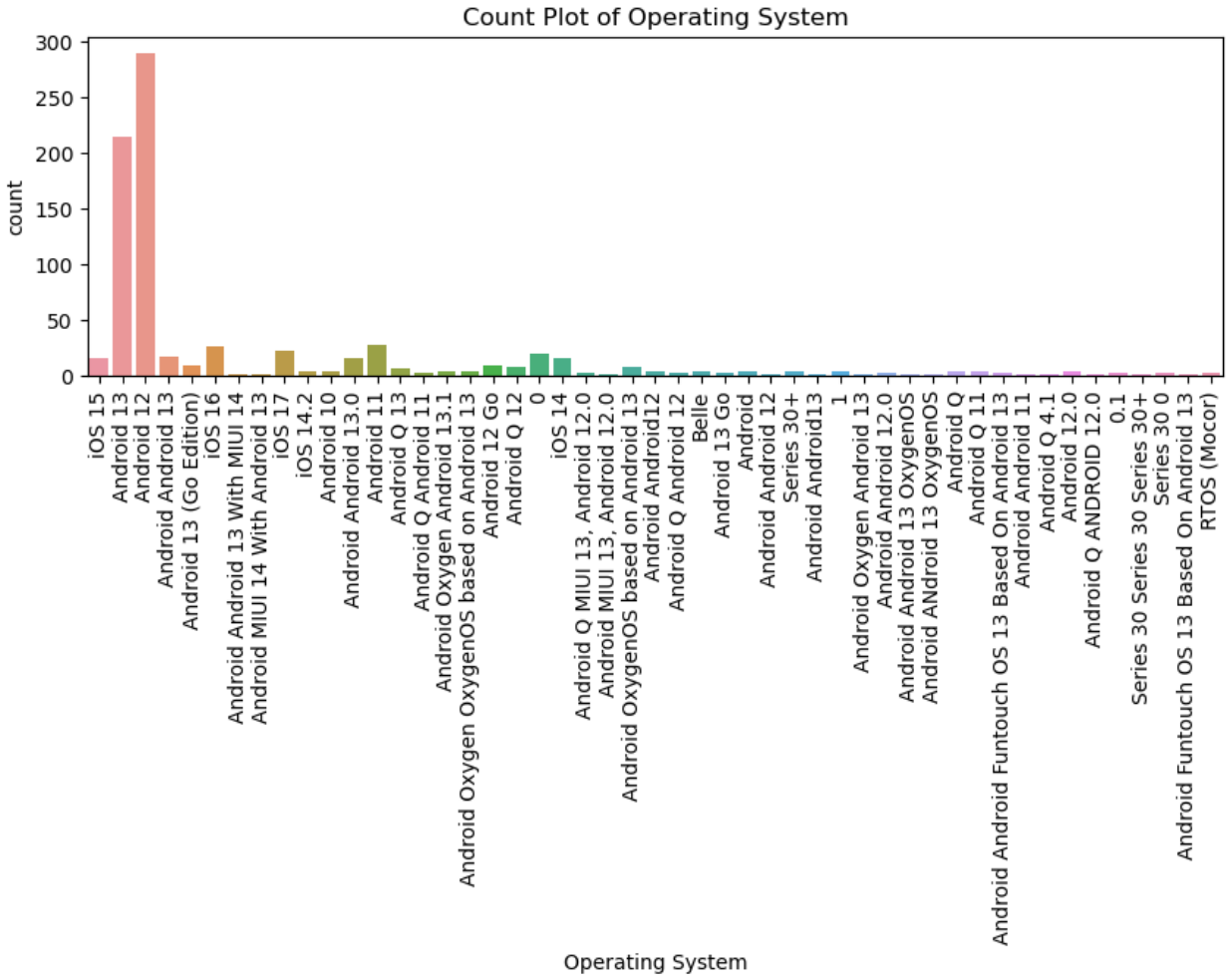
```
       'Hybrid Sim Slot.1', 'Dual Camera Lens']
for col in categorical_cols:
    plt.figure(figsize=(10, 3))
    sns.countplot(data=mobile, x=col)
    plt.xticks(rotation=90)
    plt.title(f'Count Plot of {col}')
    plt.show()
```

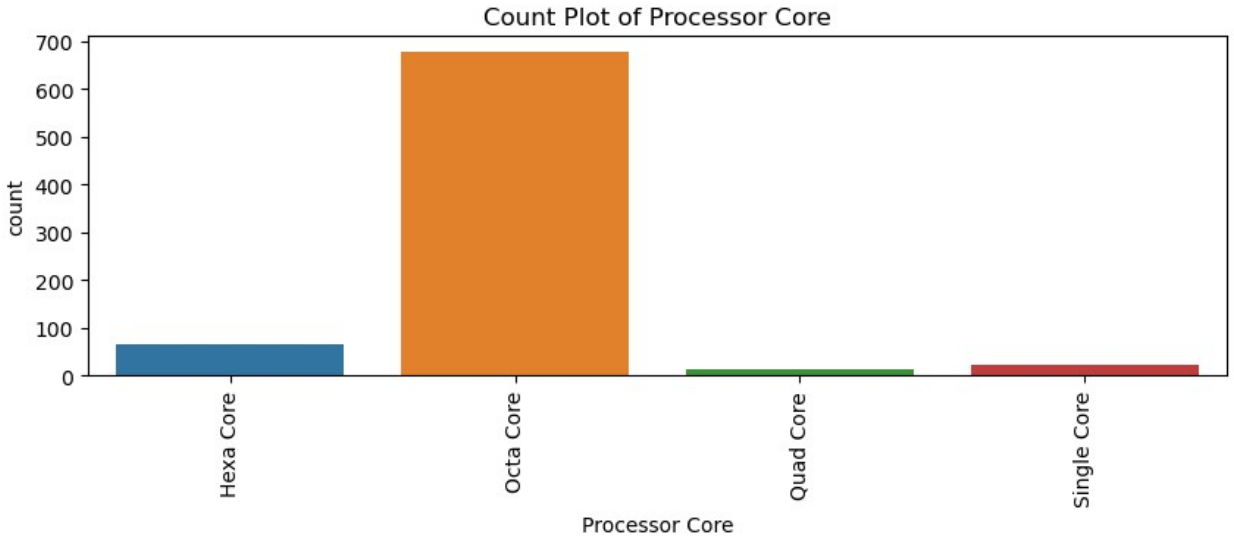


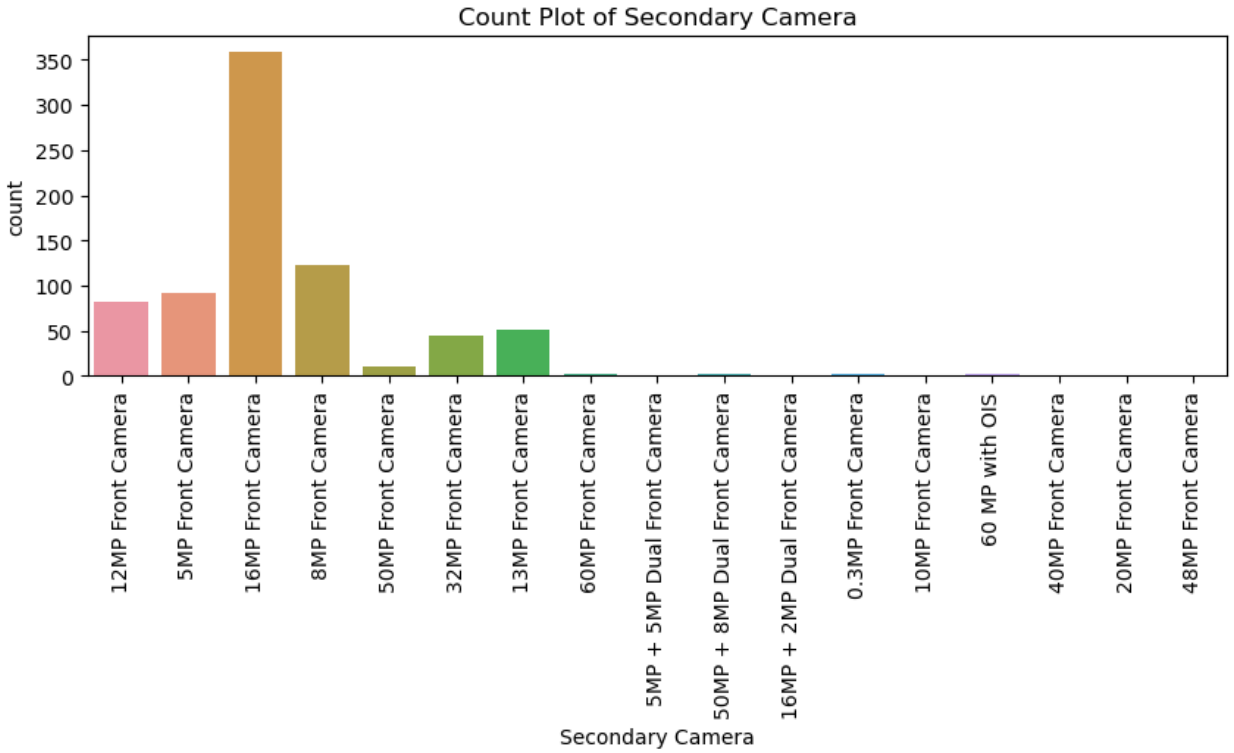


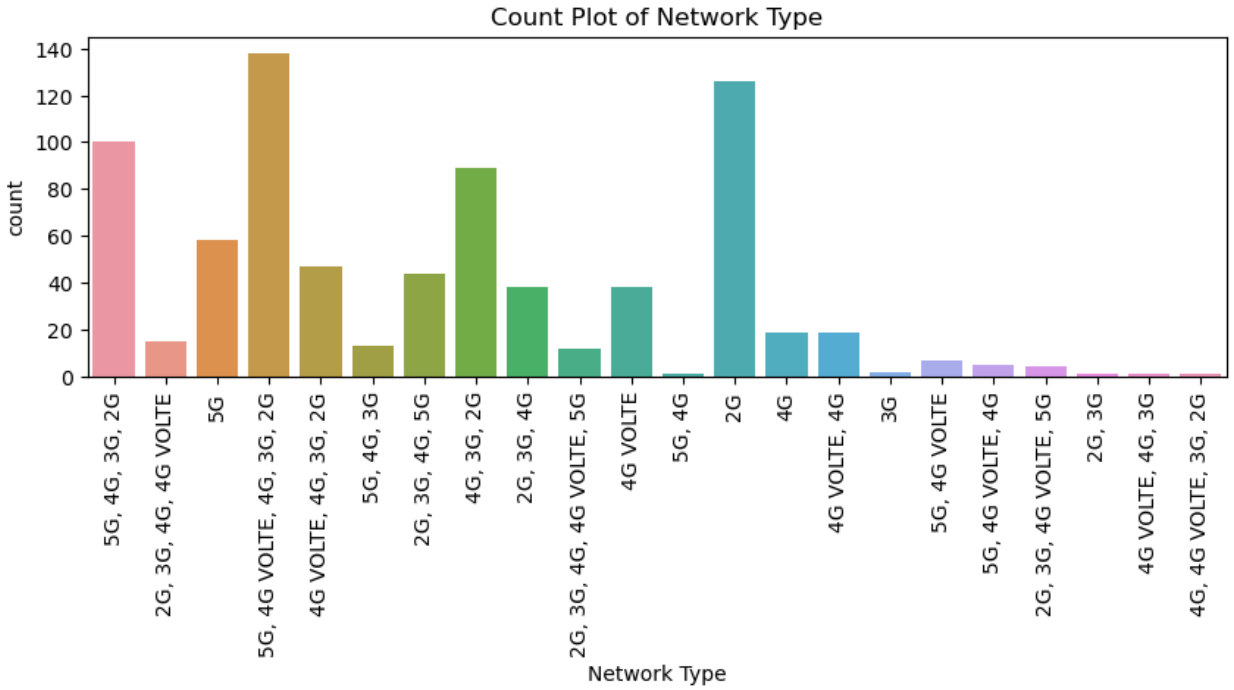

Count Plot of Hybrid Sim Slot
count
600
500
400
300
200
100
0
No
Yes
Hybrid Sim Slot

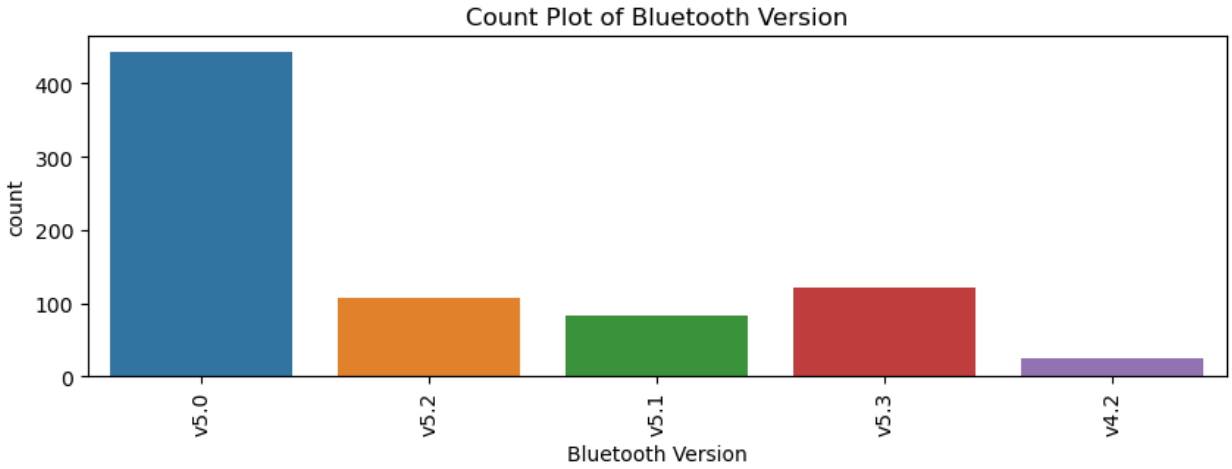

Count Plot of Touchscreen
count
600
500
400
300
200
100
0
Yes
No
Touchscreen

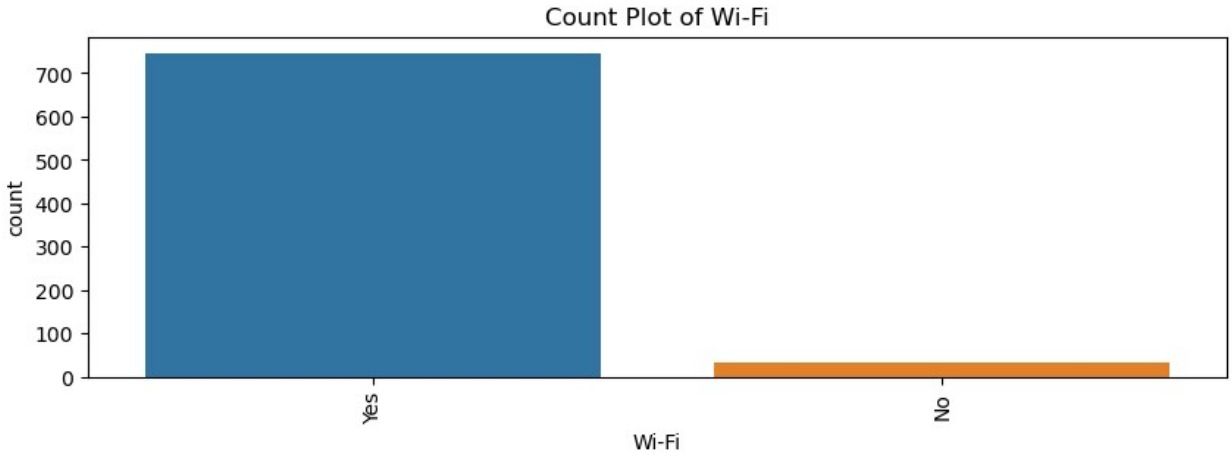

Count Plot of Operating System
count
300
250
200
150
100
50
0
iOS 15
Android 13
Android 12
Android Android 13
Android 13 (Go Edition)
iOS 16
Android Android 13 With MIUI 14
Android MIUI 14 With Android 13
iOS 17
iOS 14.2
Android 10
Android Android 13.0
Android 11
Android Q 13
Android Q Android 11
Android Oxygen OxygenOS based on Android 13.1
Android Oxygen OxygenOS based on Android 13
Android 12 Go
Android Q 12
0
iOS 14
Android Q MIUI 13, Android 12.0
Android MIUI 13, Android 12.0
Android OxygenOS based on Android 13
Android Android12
Android Q Android 12
Belle
Android 13 Go
Android
Android Android 12
Series 30+
Android Android13
1
Android Oxygen Android 13
Android Android 12.0
Android Android 13 OxygenOS
Android ANdroid 13 OxygenOS
Android Q
Android Q 11
Android Android Funtouch OS 13 Based On Android 13
Android Android 11
Android Q 4.1
Android 12.0
Android Q ANDROID 12.0
0.1
Series 30 Series 30+
Series 30 0
Android Funtouch OS 13 Based On Android 13
RTOS (Mocor)
Operating System

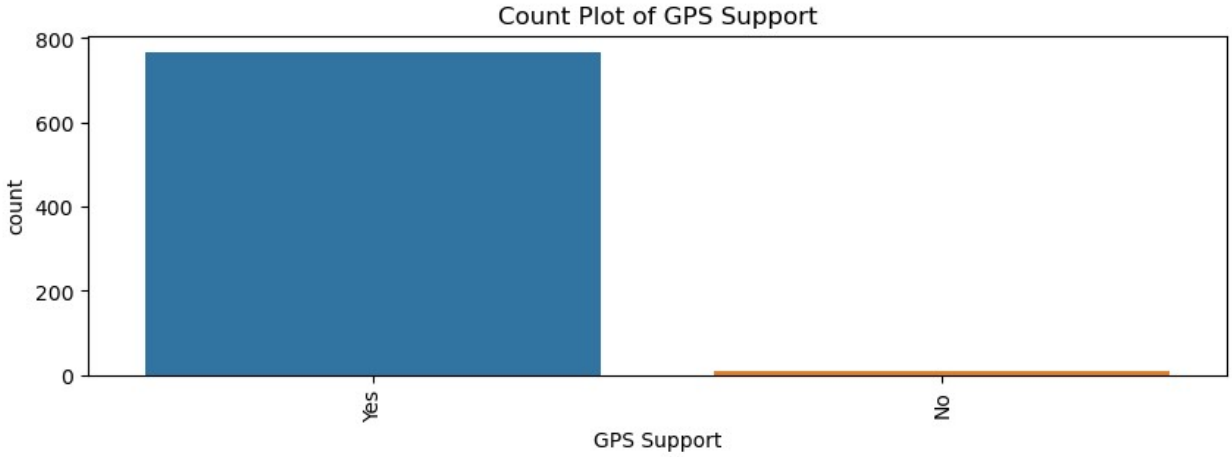

Count Plot of Processor Core
count
700
600
500
400
300
200
100
0
Hexa Core
Octa Core
Quad Core
Single Core
Processor Core

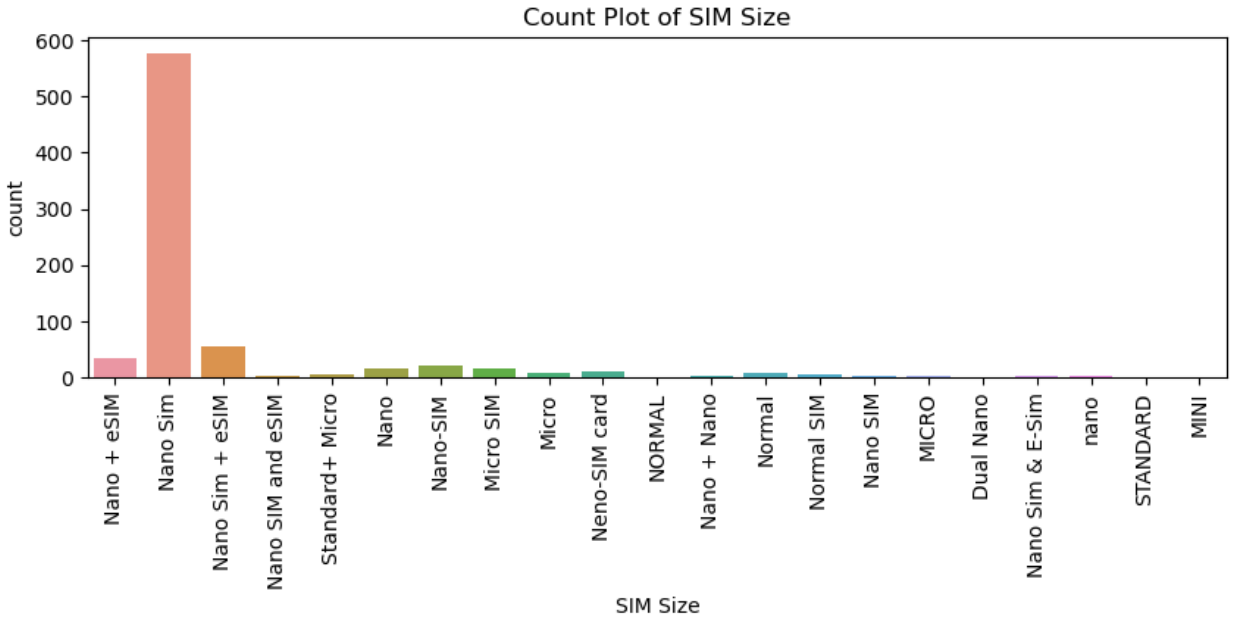

Count Plot of Secondary Camera
count
0
50
100
150
200
250
300
350
12MP Front Camera
5MP Front Camera
16MP Front Camera
8MP Front Camera
50MP Front Camera
32MP Front Camera
13MP Front Camera
60MP Front Camera
5MP + 5MP Dual Front Camera
50MP + 8MP Dual Front Camera
16MP + 2MP Dual Front Camera
0.3MP Front Camera
10MP Front Camera
60 MP with OIS
40MP Front Camera
20MP Front Camera
48MP Front Camera
Secondary Camera

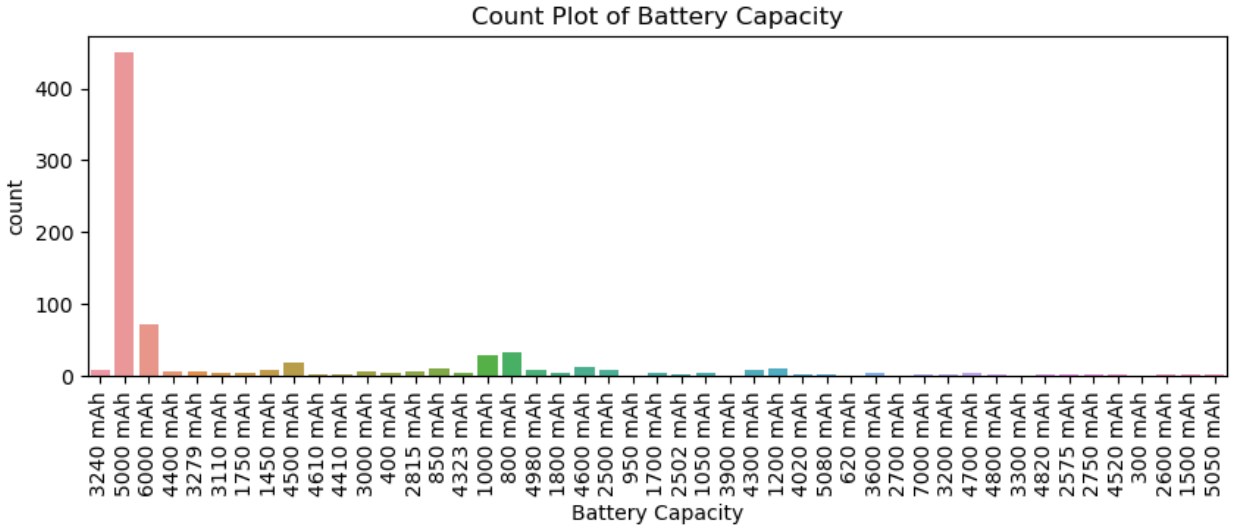

Count Plot of Network Type
count
0
20
40
60
80
100
120
140
5G, 4G, 3G, 2G
2G, 3G, 4G, 4G VOLTE
5G
5G, 4G VOLTE, 4G, 3G, 2G
4G VOLTE, 4G, 3G, 2G
5G, 4G, 3G
2G, 3G, 4G, 5G
4G, 3G, 2G
2G, 3G, 4G
2G, 3G, 4G, 4G VOLTE, 5G
4G VOLTE
5G, 4G
2G
4G
4G VOLTE, 4G
3G
5G, 4G VOLTE
5G, 4G VOLTE, 4G
2G, 3G, 4G VOLTE, 5G
2G, 3G
4G VOLTE, 4G, 3G
4G, 4G VOLTE, 3G, 2G
Network Type

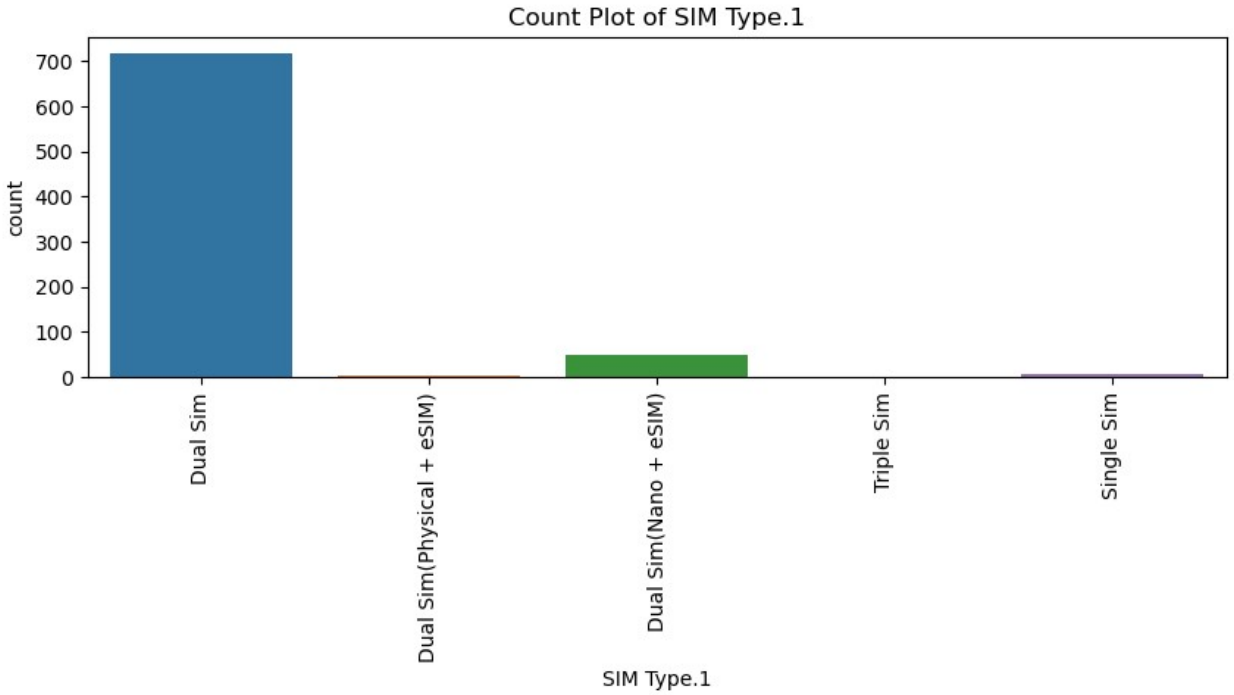

Count Plot of Bluetooth Version
count
0
100
200
300
400
v5.0
v5.2
v5.1
v5.3
v4.2
Bluetooth Version

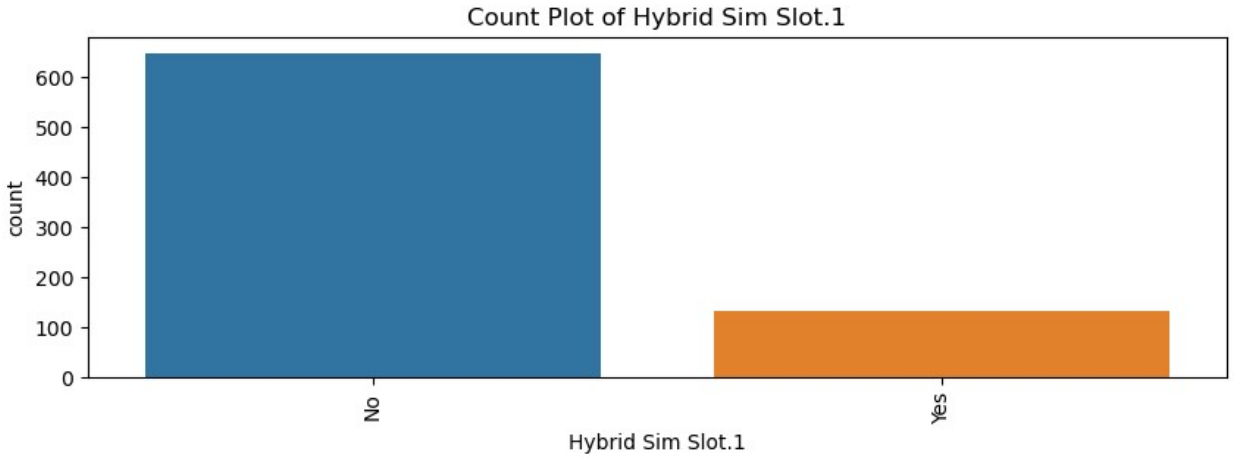

Count Plot of Wi-Fi
count
0
100
200
300
400
500
600
700
Yes
No
Wi-Fi

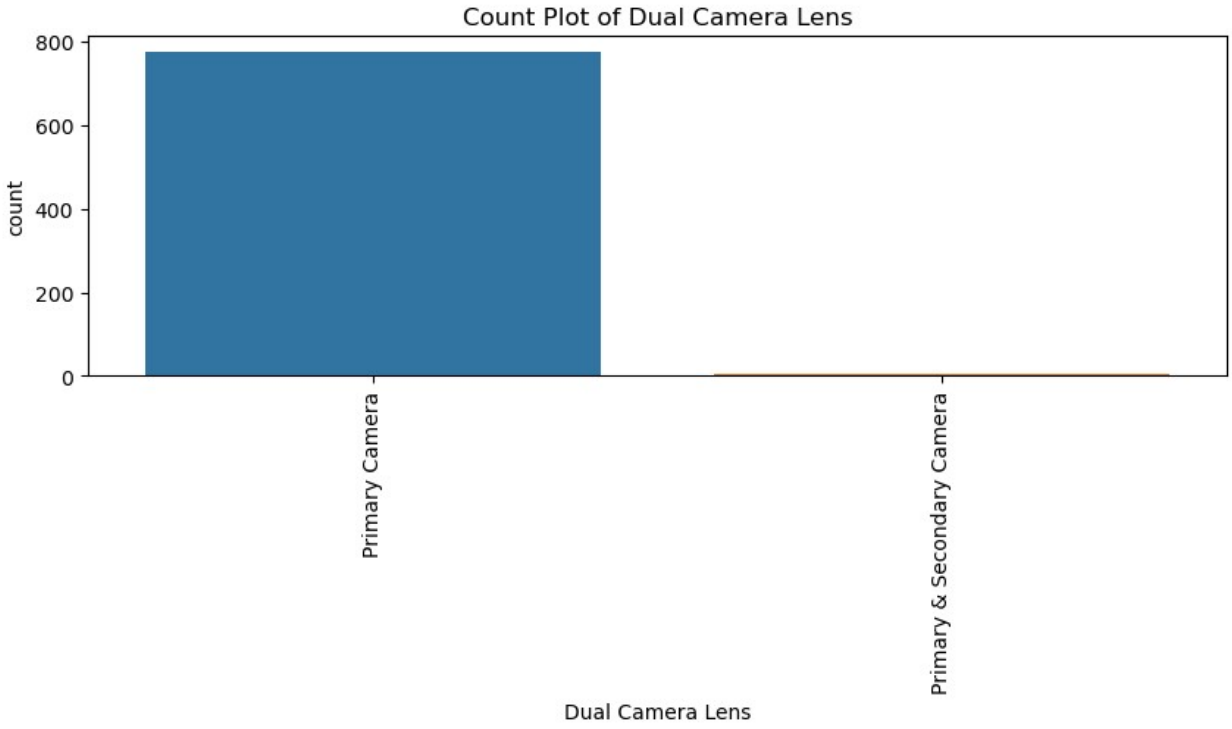

Count Plot of GPS Support
count
0
200
400
600
800
Yes
No
GPS Support


Count Plot of SIM Size
count
SIM Size
Nano + eSIM
Nano Sim
Nano Sim + eSIM
Nano SIM and eSIM
Standard+ Micro
Nano
Nano-SIM
Micro SIM
Micro
Neno-SIM card
NORMAL
Nano + Nano
Normal
Normal SIM
Nano SIM
MICRO
Dual Nano
Nano Sim & E-Sim
nano
STANDARD
MINI


Count Plot of Battery Capacity
count
Battery Capacity
3240 mAh
5000 mAh
6000 mAh
4400 mAh
3279 mAh
3110 mAh
1750 mAh
1450 mAh
4500 mAh
4610 mAh
4410 mAh
3000 mAh
400 mAh
2815 mAh
850 mAh
4323 mAh
1000 mAh
800 mAh
4980 mAh
1800 mAh
4600 mAh
2500 mAh
950 mAh
1700 mAh
2502 mAh
1050 mAh
3900 mAh
4300 mAh
1200 mAh
4020 mAh
5080 mAh
620 mAh
3600 mAh
2700 mAh
7000 mAh
3200 mAh
4700 mAh
4800 mAh
3300 mAh
4820 mAh
2575 mAh
2750 mAh
4520 mAh
300 mAh
2600 mAh
1500 mAh
5050 mAh


Count Plot of SIM Type.1
count
700
600
500
400
300
200
100
0
Dual Sim
Dual Sim(Physical + eSIM)
Dual Sim(Nano + eSIM)
Triple Sim
Single Sim
SIM Type.1


Count Plot of Hybrid Sim Slot.1
count
600
500
400
300
200
100
0
No
Yes
Hybrid Sim Slot.1

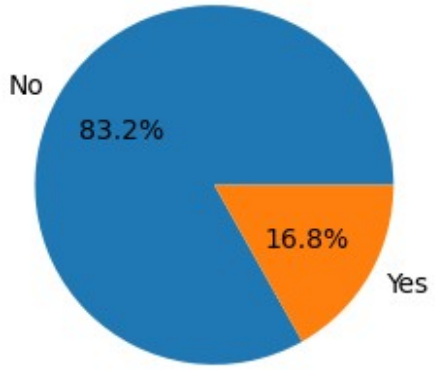


Count plot Clearly shows the Realme , Redme , Apple are the brands which are widely in use

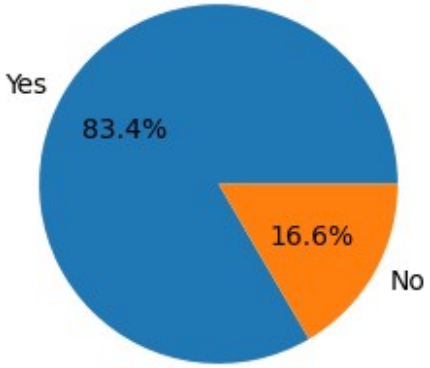
Pie charts for categorical columns

```
categorical_cols = [   'Hybrid Sim Slot', 'Touchscreen',
        'Processor Core',
         'Bluetooth Version', 'Wi-Fi',
       'GPS Support',  'SIM Type.1',
       'Hybrid Sim Slot.1', 'Dual Camera Lens']
for col in categorical_cols:
    plt.figure(figsize=(4, 3))
    data_counts = mobile[col].value_counts()
    plt.pie(data_counts, labels=data_counts.index, autopct='%1.1f%%')
    plt.title(f'Pie Chart of {col}')
    plt.show()
```

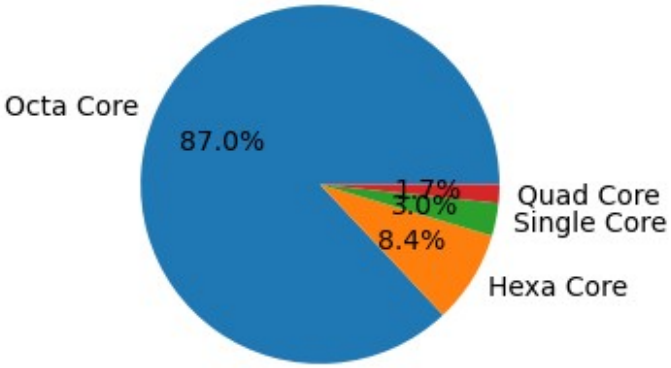
Pie Chart of Hybrid Sim Slot



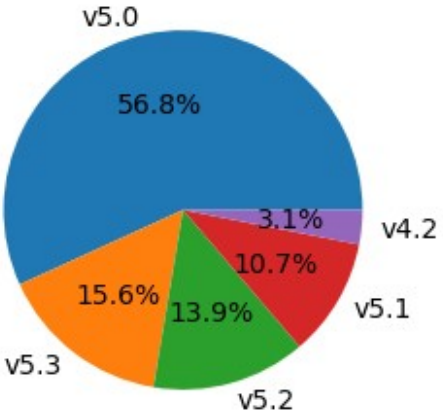
Pie Chart of Touchscreen



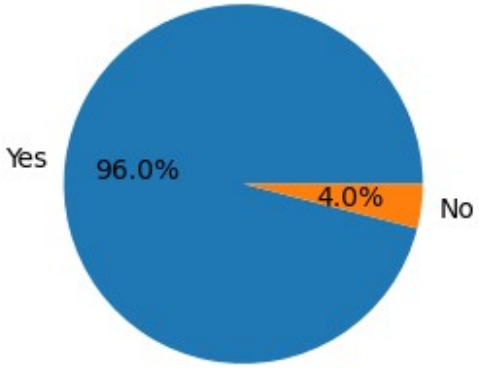
Pie Chart of Processor Core



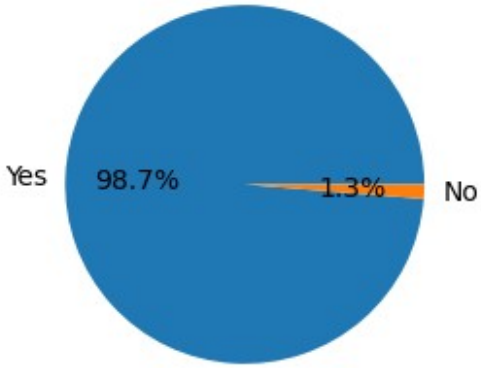
Pie Chart of Bluetooth Version



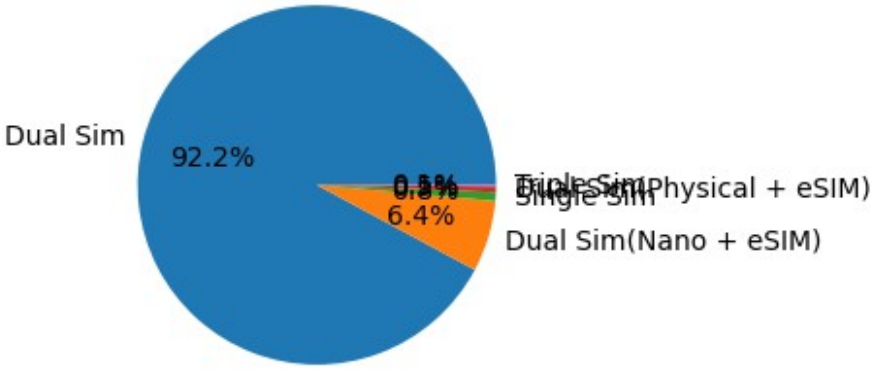
Pie Chart of Wi-Fi



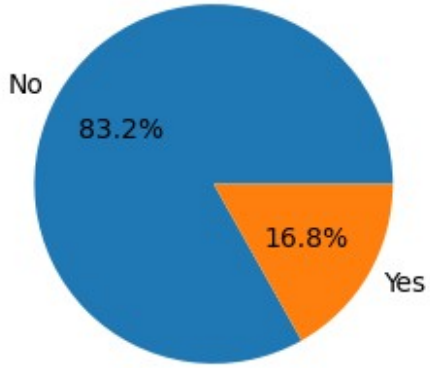
Pie Chart of GPS Support



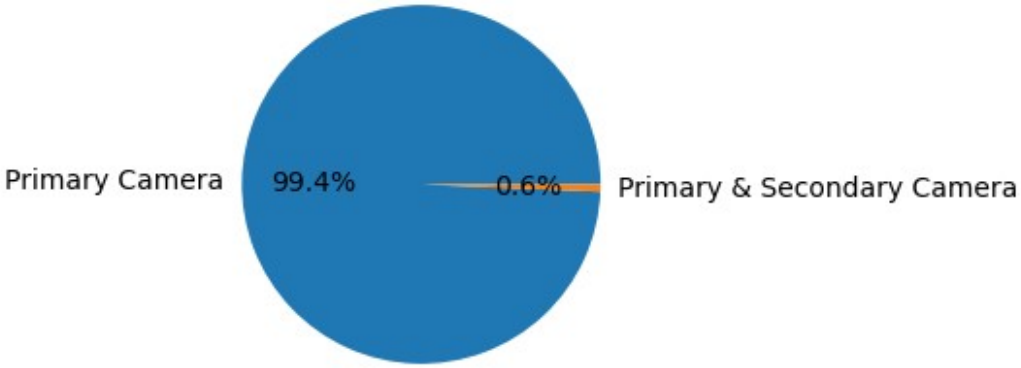
Pie Chart of SIM Type.1



Pie Chart of Hybrid Sim Slot.1



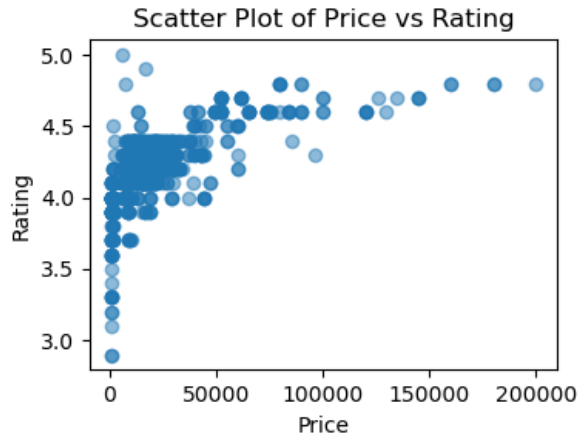
Pie Chart of Dual Camera Lens

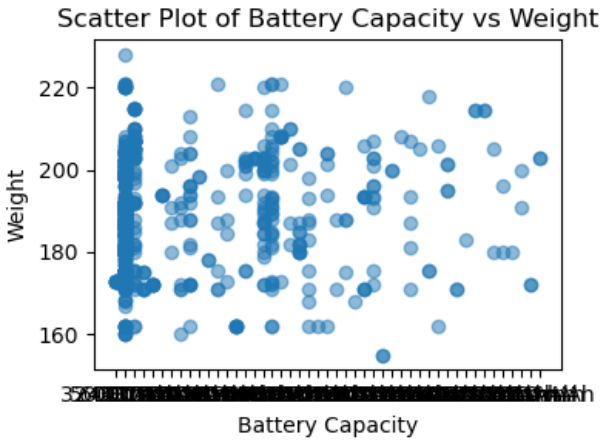


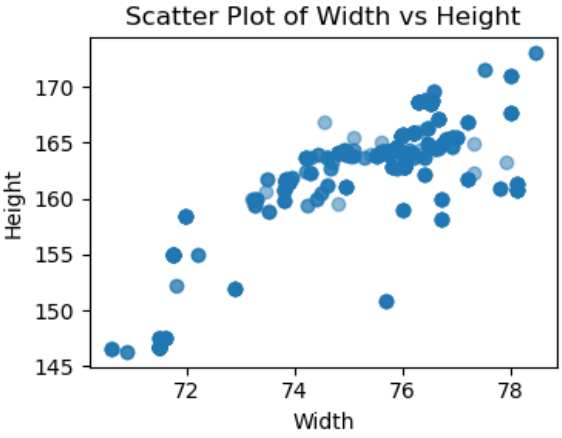
5.Bivariate Analysis

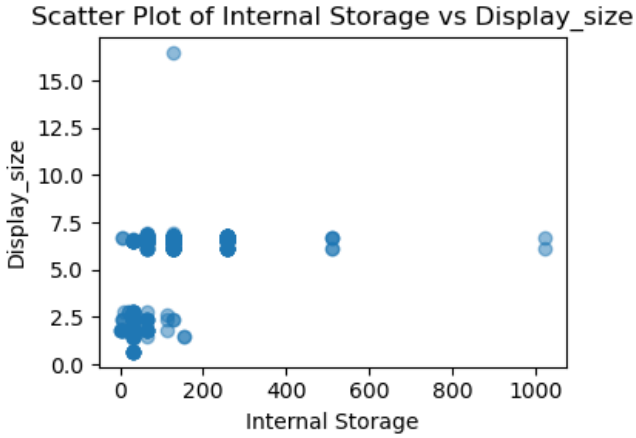
Scatter plots

```
fig, axes = plt.subplots(2, 2, figsize=(8, 6))

axes[0, 0].scatter(mobile['Price'], mobile['Rating'], alpha=0.5)
axes[0, 0].set_xlabel('Price')
axes[0, 0].set_ylabel('Rating')
axes[0, 0].set_title('Scatter Plot of Price vs Rating')

axes[0, 1].scatter(mobile['Battery Capacity'], mobile['Weight'],
alpha=0.5)
axes[0, 1].set_xlabel('Battery Capacity')
axes[0, 1].set_ylabel('Weight')
axes[0, 1].set_title('Scatter Plot of Battery Capacity vs Weight')

axes[1, 0].scatter(mobile['Width'], mobile['Height'], alpha=0.5)
axes[1, 0].set_xlabel('Width')
axes[1, 0].set_ylabel('Height')
axes[1, 0].set_title('Scatter Plot of Width vs Height')

axes[1, 1].scatter(mobile['Internal Storage'], mobile['Display_size'],
alpha=0.5)
axes[1, 1].set_xlabel('Internal Storage')
axes[1, 1].set_ylabel('Display_size')
axes[1, 1].set_title('Scatter Plot of Internal Storage vs
Display_size')


plt.tight_layout()
plt.show()
```









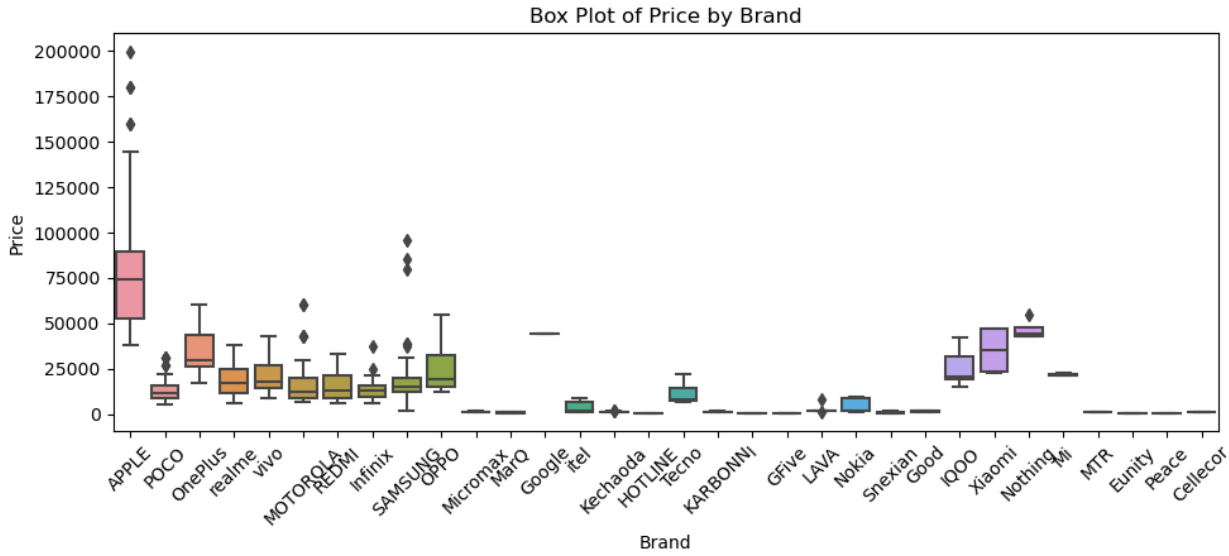
Corelatioon Matrix

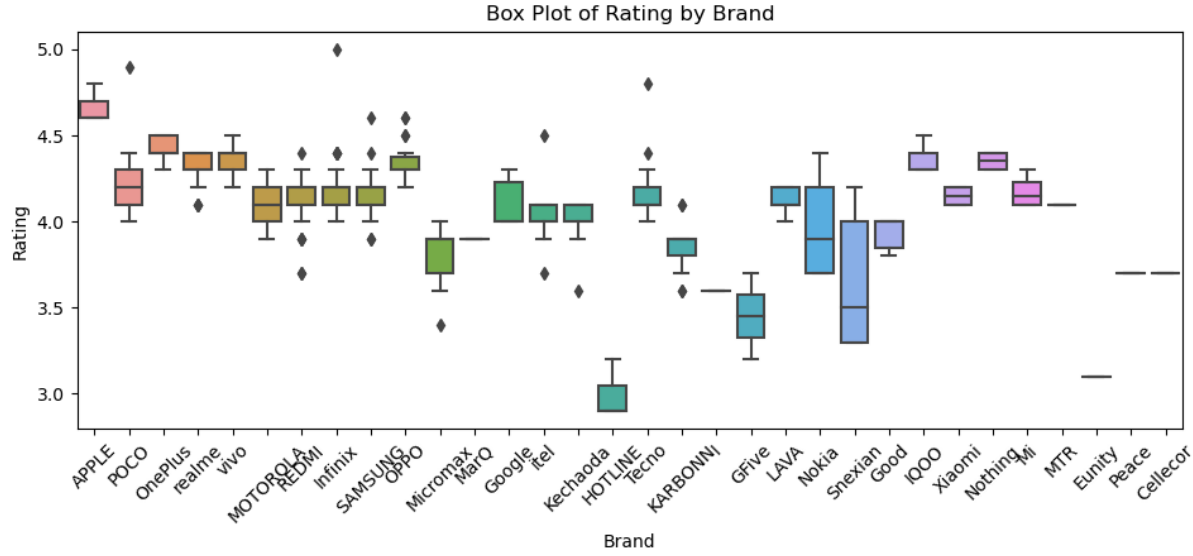
```
correlation_matrix = mobile.select_dtypes(include=[np.number]).corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='viridis')
plt.title('Correlation Matrix')
plt.show()
```

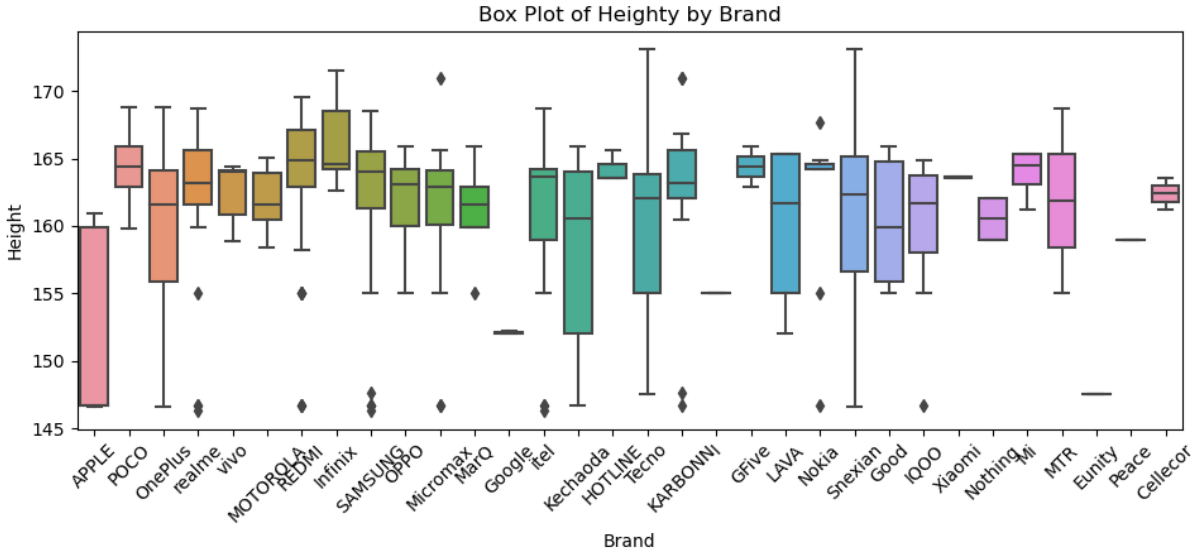
Box plot by Category

```
fig, axes = plt.subplots(4, 1, figsize=(10, 18))

sns.boxplot(data=mobile, x='Brand', y='Price', ax=axes[0])
axes[0].set_title('Box Plot of Price by Brand')
axes[0].tick_params(axis='x', rotation=45)

sns.boxplot(data=mobile, x='Brand', y='Rating', ax=axes[1])
axes[1].set_title('Box Plot of Rating by Brand')
axes[1].tick_params(axis='x', rotation=45)

sns.boxplot(data=mobile, x='Brand', y='Height', ax=axes[2])
axes[2].set_title('Box Plot of Heighty by Brand')
axes[2].tick_params(axis='x', rotation=45)

sns.boxplot(data=mobile, x='Brand', y='Weight', ax=axes[3])
axes[3].set_title('Box Plot of Weight by Brand')
axes[3].tick_params(axis='x', rotation=45)
```

```
plt.tight_layout()
plt.show()
```



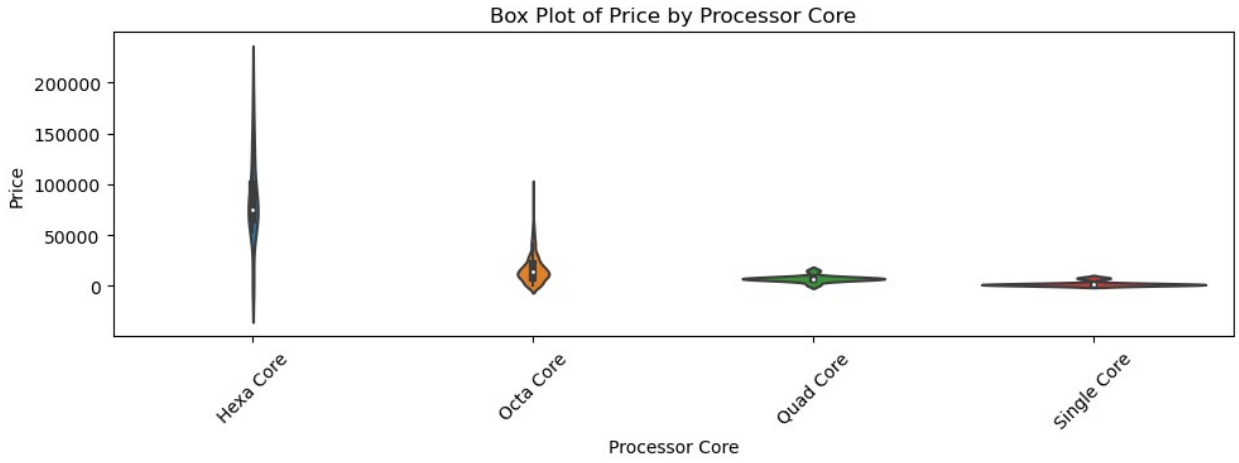


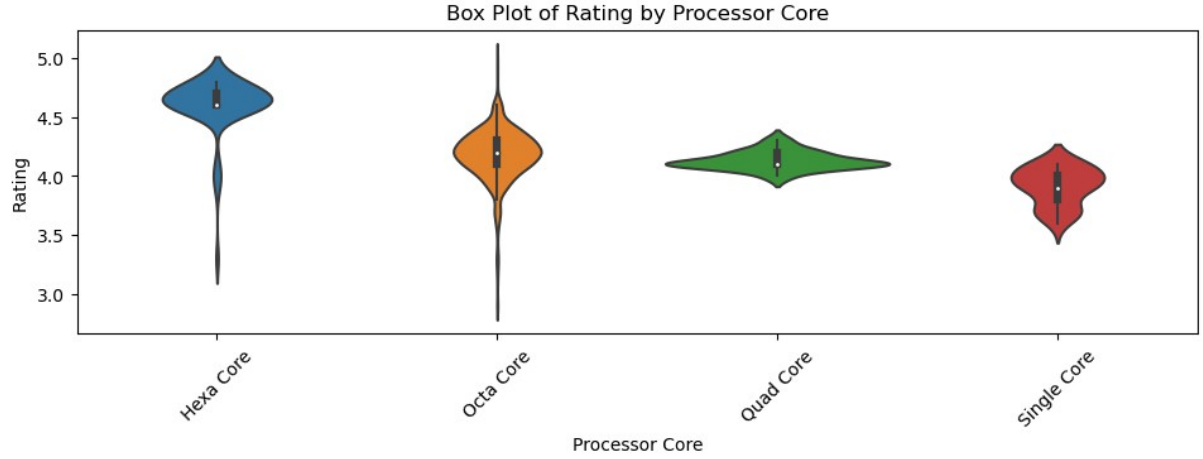


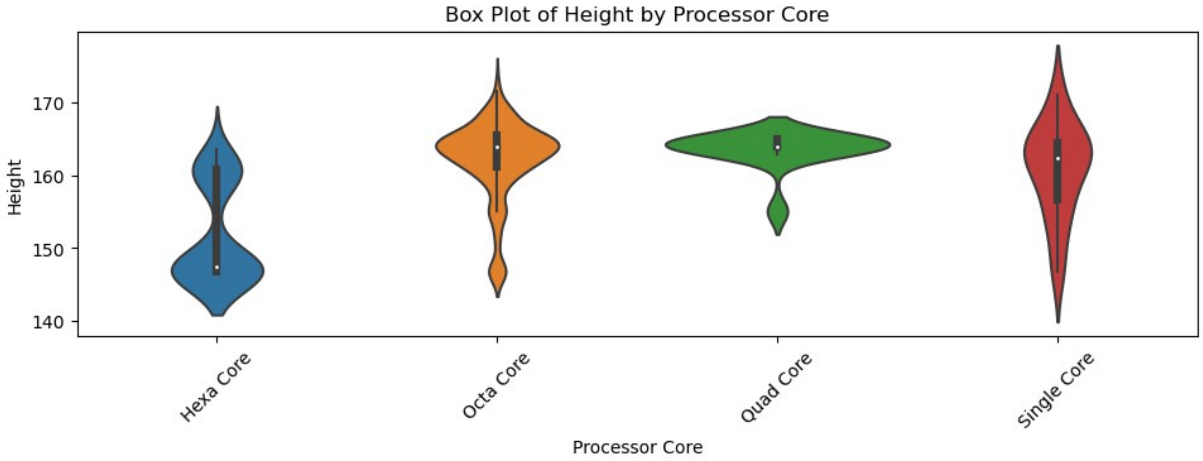
Box Plot of Weight by Brand

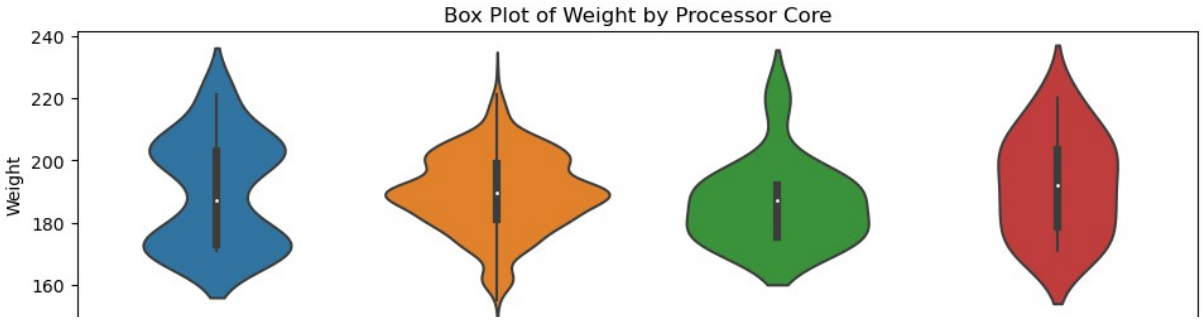
Violin plot by Category

```
fig, axes = plt.subplots(4, 1, figsize=(10, 15))

sns.violinplot(data=mobile, x='Processor Core', y='Price', ax=axes[0])
axes[0].set_title('Box Plot of Price by Processor Core')
axes[0].tick_params(axis='x', rotation=45)

sns.violinplot(data=mobile, x='Processor Core', y='Rating',
ax=axes[1])
axes[1].set_title('Box Plot of Rating by Processor Core')
axes[1].tick_params(axis='x', rotation=45)

sns.violinplot(data=mobile, x='Processor Core', y='Height',
ax=axes[2])
axes[2].set_title('Box Plot of Height by Processor Core')
axes[2].tick_params(axis='x', rotation=45)

sns.violinplot(data=mobile, x='Processor Core', y='Weight',
ax=axes[3])
axes[3].set_title('Box Plot of Weight by Processor Core')
axes[3].tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```


Box Plot of Price by Processor Core
Price
200000
150000
100000
50000
0
Hexa Core
Octa Core
Quad Core
Single Core
Processor Core

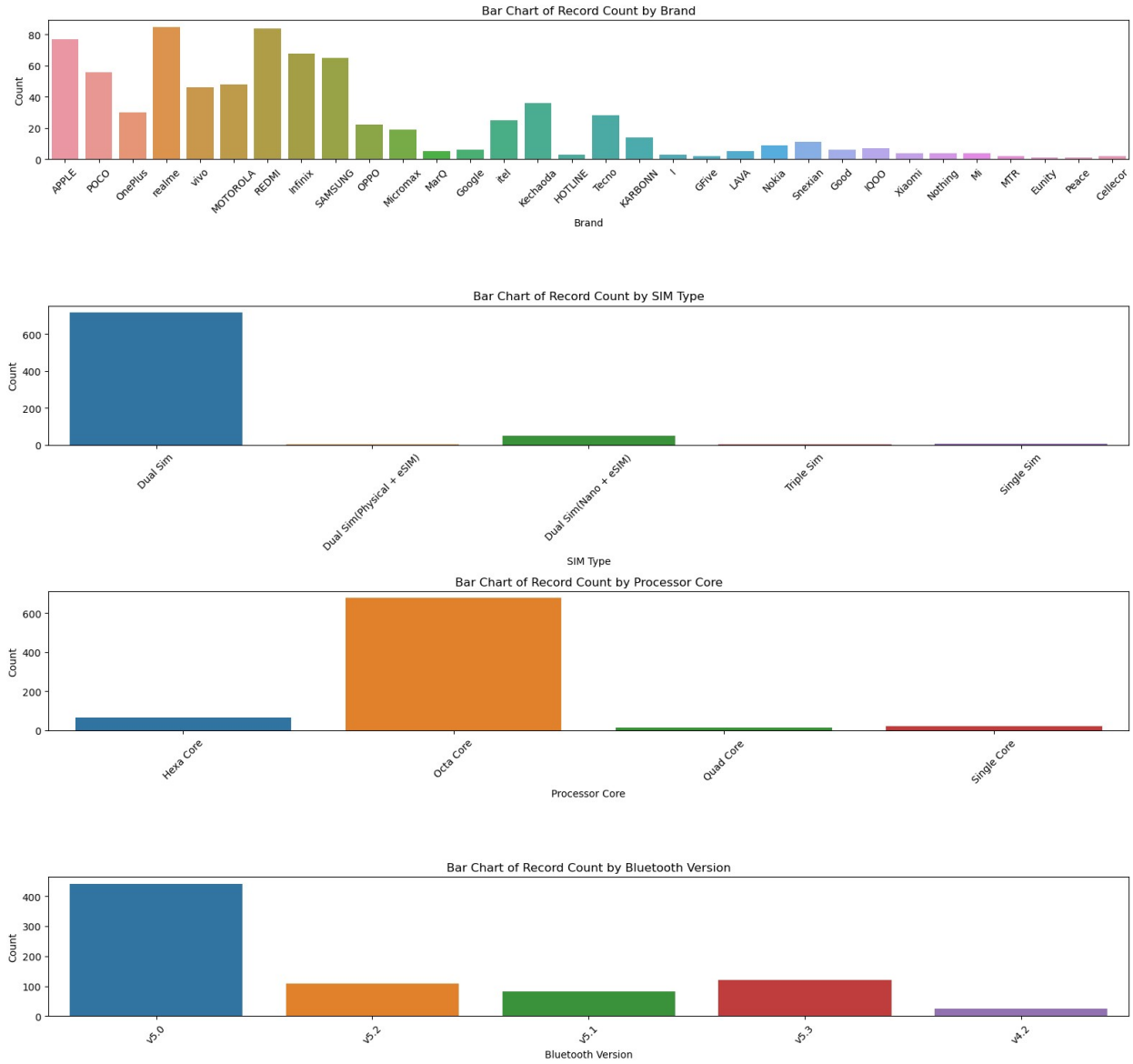

Box Plot of Rating by Processor Core
Rating
5.0
4.5
4.0
3.5
3.0
Hexa Core
Octa Core
Quad Core
Single Core
Processor Core


Box Plot of Height by Processor Core
Height
170
160
150
140
Hexa Core
Octa Core
Quad Core
Single Core
Processor Core


Box Plot of Weight by Processor Core
Weight
240
220
200
180
160

Bar Chart by Category

```
fig, axes = plt.subplots(4, 1, figsize=(16, 15))


sns.countplot(data=mobile, x='Brand', ax=axes[0])
axes[0].set_title('Bar Chart of Record Count by Brand')
axes[0].set_xlabel('Brand')
axes[0].set_ylabel('Count')
axes[0].tick_params(axis='x', rotation=45)

sns.countplot(data=mobile, x='SIM Type', ax=axes[1])
axes[1].set_title('Bar Chart of Record Count by SIM Type')
axes[1].set_xlabel('SIM Type')
axes[1].set_ylabel('Count')
axes[1].tick_params(axis='x', rotation=45)

sns.countplot(data=mobile, x='Processor Core', ax=axes[2])
axes[2].set_title('Bar Chart of Record Count by Processor Core')
axes[2].set_xlabel('Processor Core')
axes[2].set_ylabel('Count')
axes[2].tick_params(axis='x', rotation=45)

sns.countplot(data=mobile, x='Bluetooth Version', ax=axes[3])
axes[3].set_title('Bar Chart of Record Count by Bluetooth Version')
axes[3].set_xlabel('Bluetooth Version')
axes[3].set_ylabel('Count')
axes[3].tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```

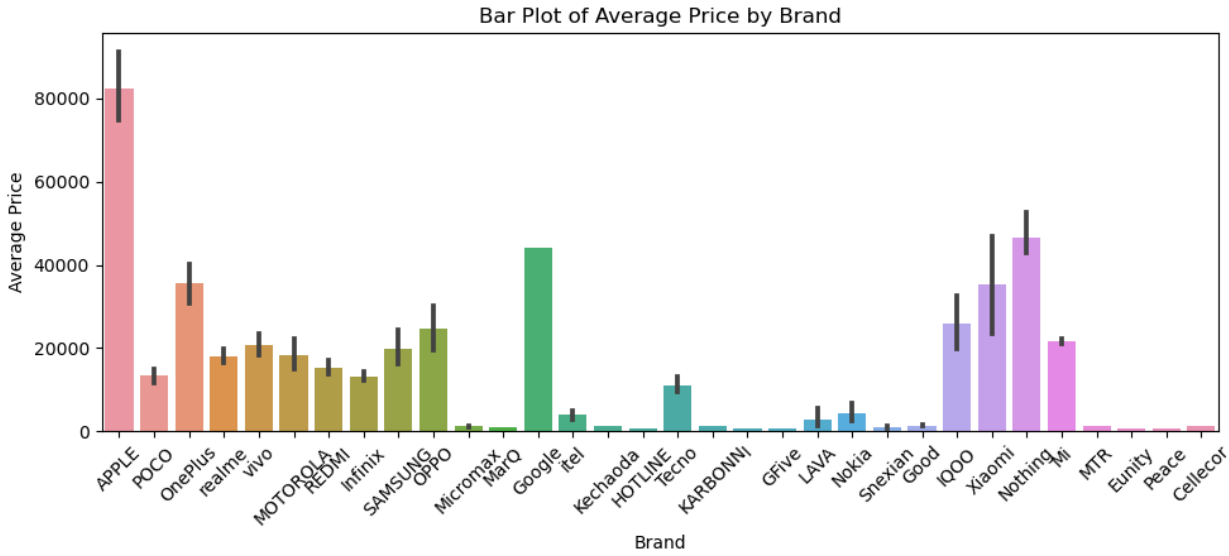


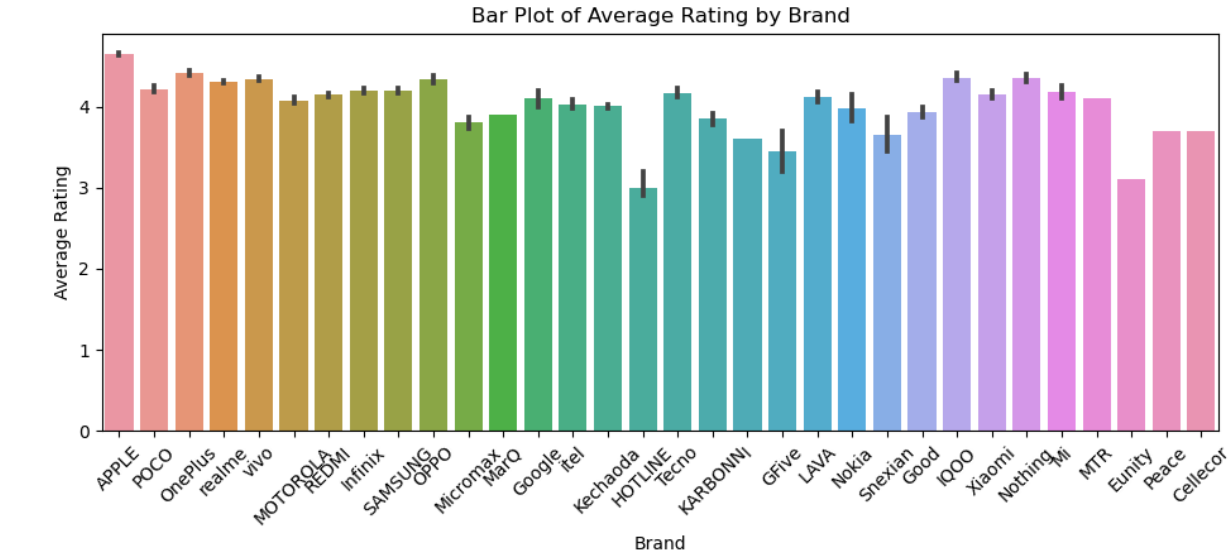
Bar Charts clears that : Dual Sim is the most common configuration among all types Octa Core is the most usage processor in every mobile v5.0 is the most specified Bluetooth Version

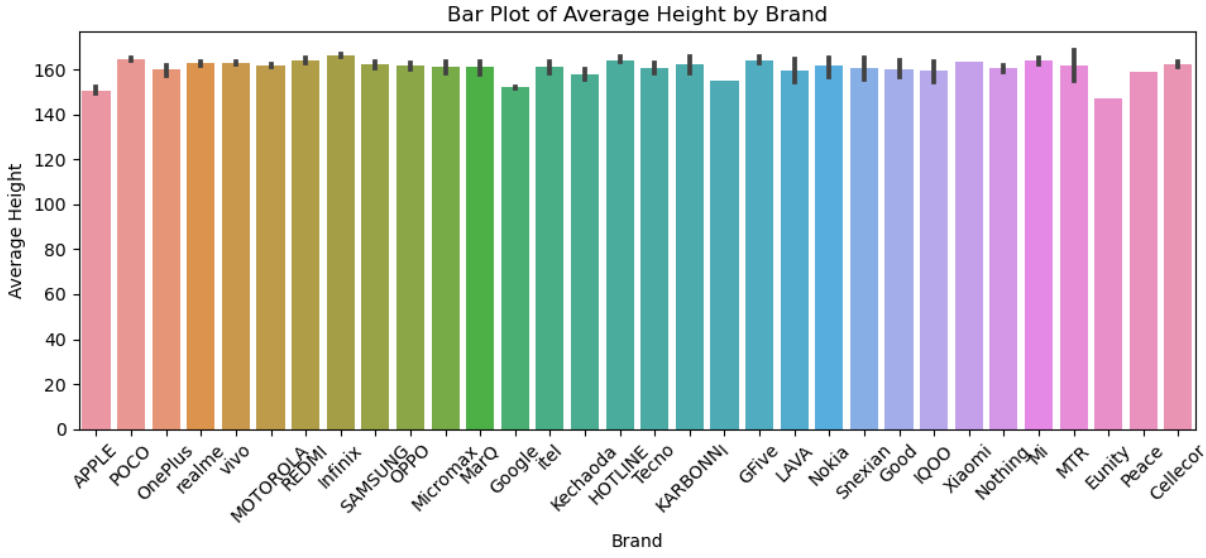
Barplot by Category

```
fig, axes = plt.subplots(4, 1, figsize=(10, 18))

sns.barplot(data=mobile, x='Brand', y='Price', ax=axes[0])
axes[0].set_title('Bar Plot of Average Price by Brand')
axes[0].set_xlabel('Brand')
axes[0].set_ylabel('Average Price')
axes[0].tick_params(axis='x', rotation=45)


sns.barplot(data=mobile, x='Brand', y='Rating', ax=axes[1])
axes[1].set_title('Bar Plot of Average Rating by Brand')
```

```
axes[1].set_xlabel('Brand')
axes[1].set_ylabel('Average Rating')
axes[1].tick_params(axis='x', rotation=45)

sns.barplot(data=mobile, x='Brand', y='Height', ax=axes[2])
axes[2].set_title('Bar Plot of Average Height by Brand')
axes[2].set_xlabel('Brand')
axes[2].set_ylabel('Average Height')
axes[2].tick_params(axis='x', rotation=45)

sns.barplot(data=mobile, x='Brand', y='Weight', ax=axes[3])
axes[3].set_title('Bar Plot of Average Weight by Brand')
axes[3].set_xlabel('Brand')
axes[3].set_ylabel('Average Weight')
axes[3].tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```





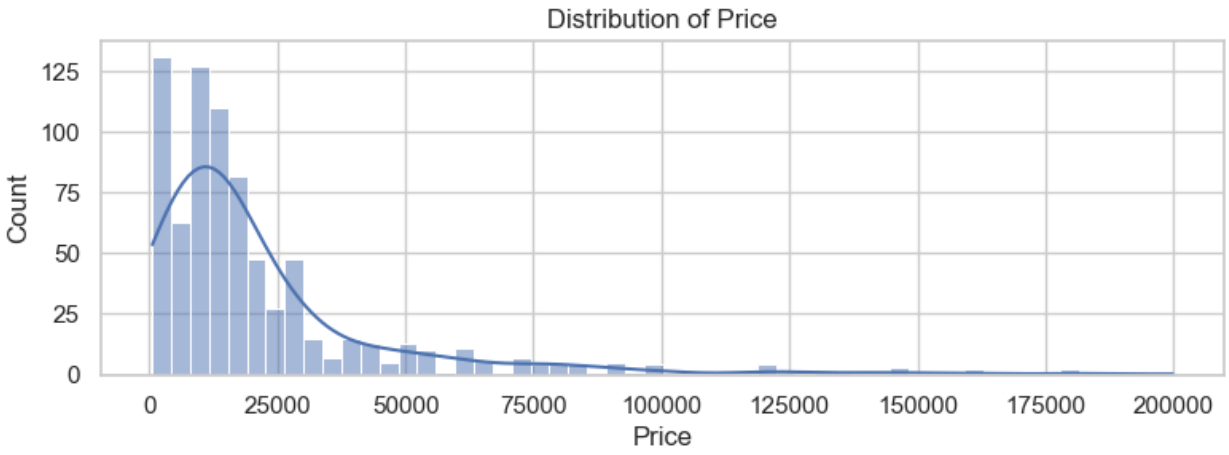


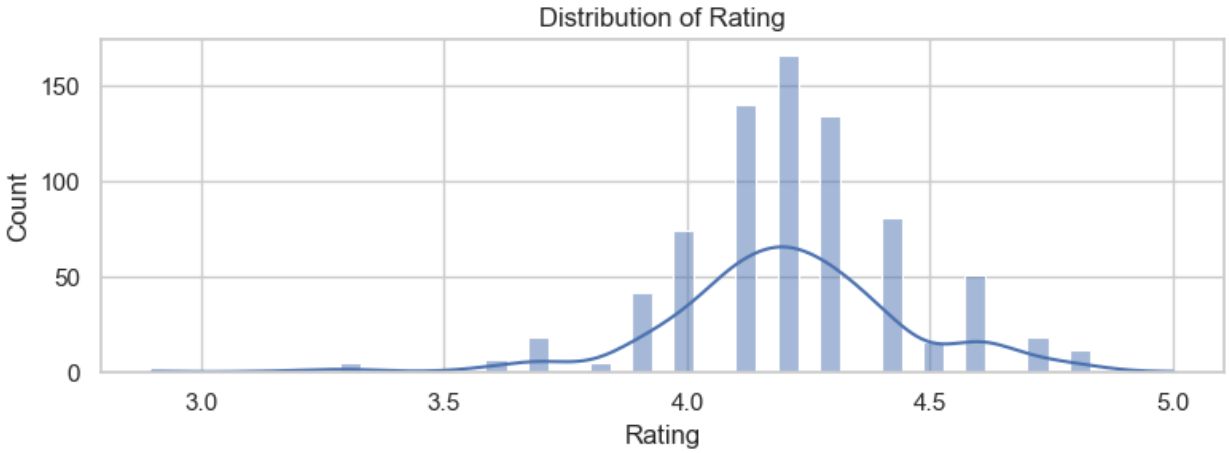
Bar Plot of Average Weight by Brand

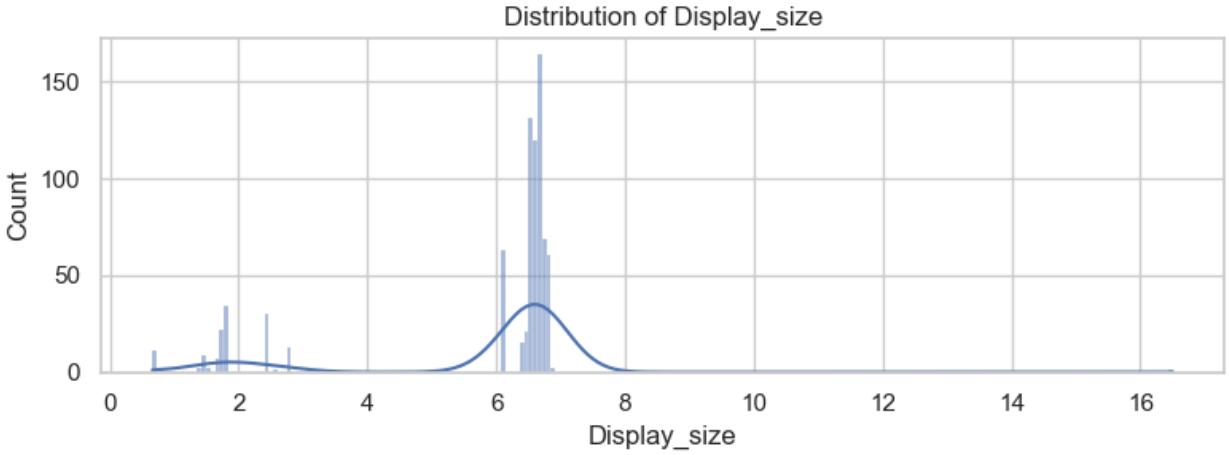
Undoubtly Apple Is the most costliest mobile brand which is highly priced mobile

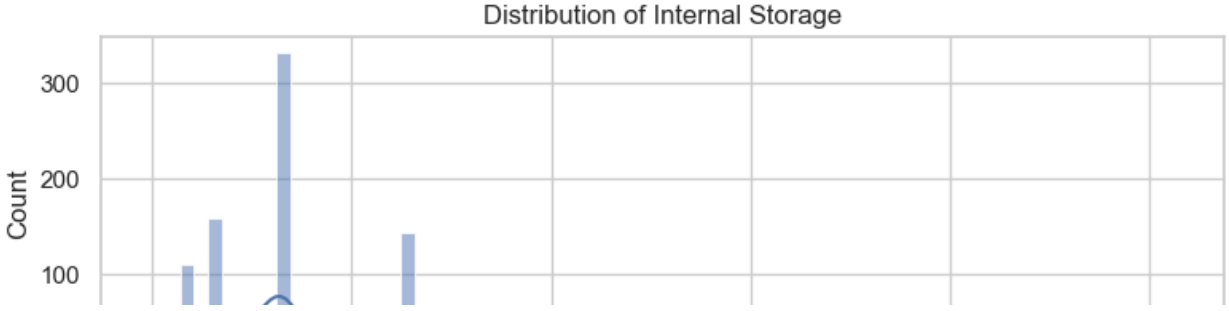
Displot

```
sns.set(style="whitegrid")

# Create a 4x1 subplot grid
fig, axes = plt.subplots(nrows=4, ncols=1, figsize=(8, 12))

# Distribution plot for 'Price'
sns.histplot(mobile['Price'], kde=True, ax=axes[0])
axes[0].set_title('Distribution of Price')

# Distribution plot for 'Rating'
sns.histplot(mobile['Rating'], kde=True, ax=axes[1])
axes[1].set_title('Distribution of Rating')

# Distribution plot for 'Display_size'
sns.histplot(mobile['Display_size'], kde=True, ax=axes[2])
axes[2].set_title('Distribution of Display_size')

# Distribution plot for 'Internal Storage'
sns.histplot(mobile['Internal Storage'], kde=True, ax=axes[3])
axes[3].set_title('Distribution of Internal Storage')

# Adjust spacing
plt.tight_layout()
plt.show()
```


Distribution of Price
Count
Price
0
25
50
75
100
125
0
25000
50000
75000
100000
125000
150000
175000
200000

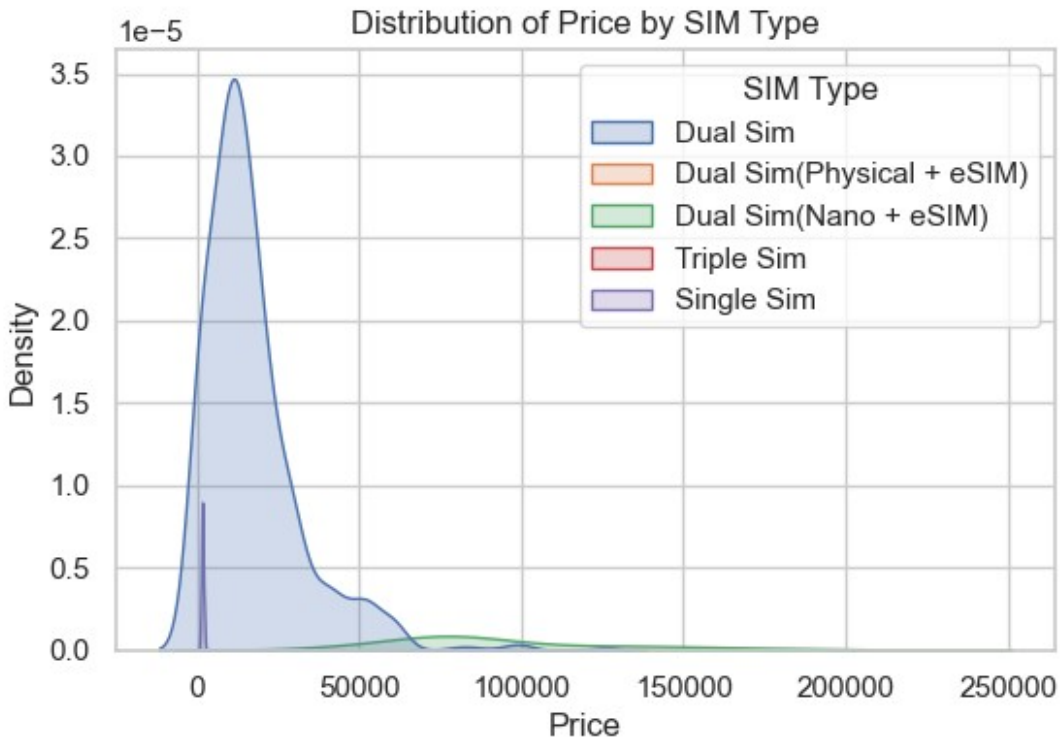

Distribution of Rating
Count
Rating
0
50
100
150
3.0
3.5
4.0
4.5
5.0

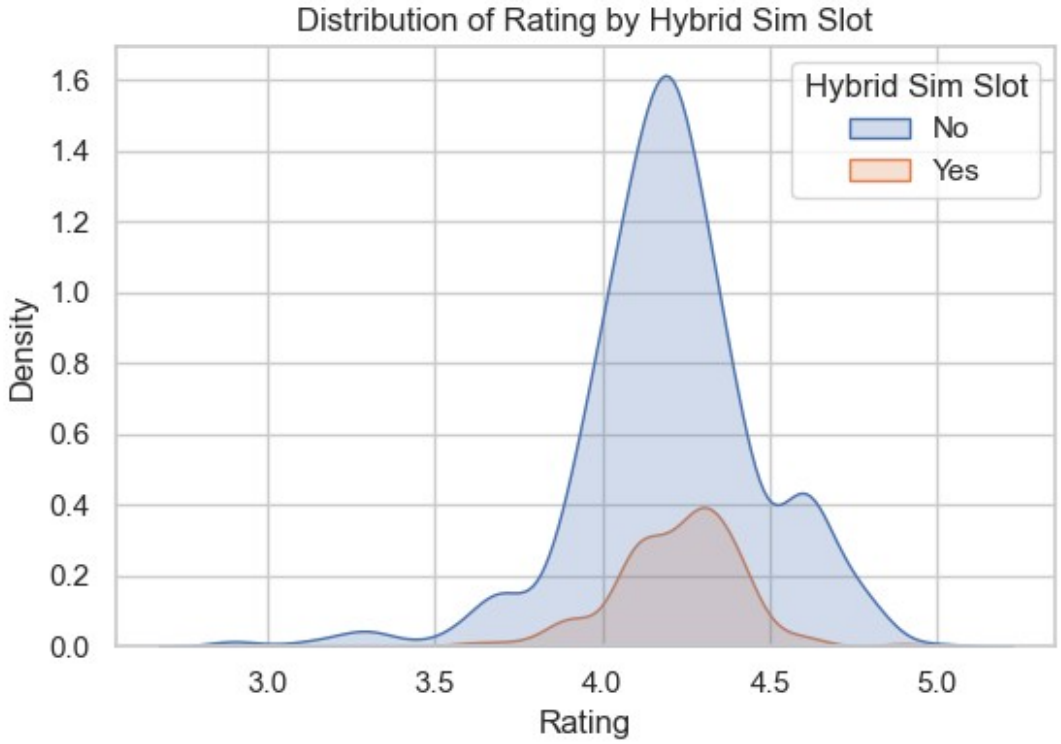

Distribution of Display_size
Count
Display_size
0
50
100
150
0
2
4
6
8
10
12
14
16

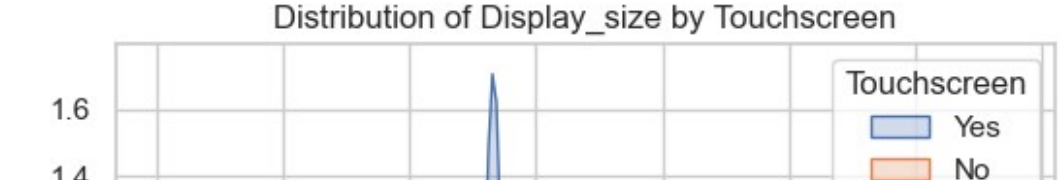

Distribution of Internal Storage
Count
100
200
300

KDE plot

```
sns.set(style="whitegrid")

# Create a 4x1 subplot grid
fig, axes = plt.subplots(nrows=4, ncols=1, figsize=(6, 16))

# Distribution plot 1: 'Price' with 'SIM Type'
sns.kdeplot(data=mobile, x='Price', hue='SIM Type', ax=axes[0],
fill=True)
axes[0].set_title('Distribution of Price by SIM Type')

# Distribution plot 2: 'Rating' with 'Hybrid Sim Slot'
sns.kdeplot(data=mobile, x='Rating', hue='Hybrid Sim Slot',
ax=axes[1], fill=True)
axes[1].set_title('Distribution of Rating by Hybrid Sim Slot')

# Distribution plot 3: 'Display_size' with 'Color'
sns.kdeplot(data=mobile, x='Display_size', hue='Touchscreen',
ax=axes[2], fill=True)
axes[2].set_title('Distribution of Display_size by Touchscreen')

# Distribution plot 4: 'Internal Storage' with 'Processor Core'
sns.kdeplot(data=mobile, x='Internal Storage', hue='Processor Core',
ax=axes[3], fill=True)
axes[3].set_title('Distribution of Internal Storage by Processor
Core')

# Adjust spacing
plt.tight_layout()
plt.show()
```


Distribution of Price by SIM Type
1e−5
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Density
SIM Type
Dual Sim
Dual Sim(Physical + eSIM)
Dual Sim(Nano + eSIM)
Triple Sim
Single Sim
0
50000
100000
150000
200000
250000
Price

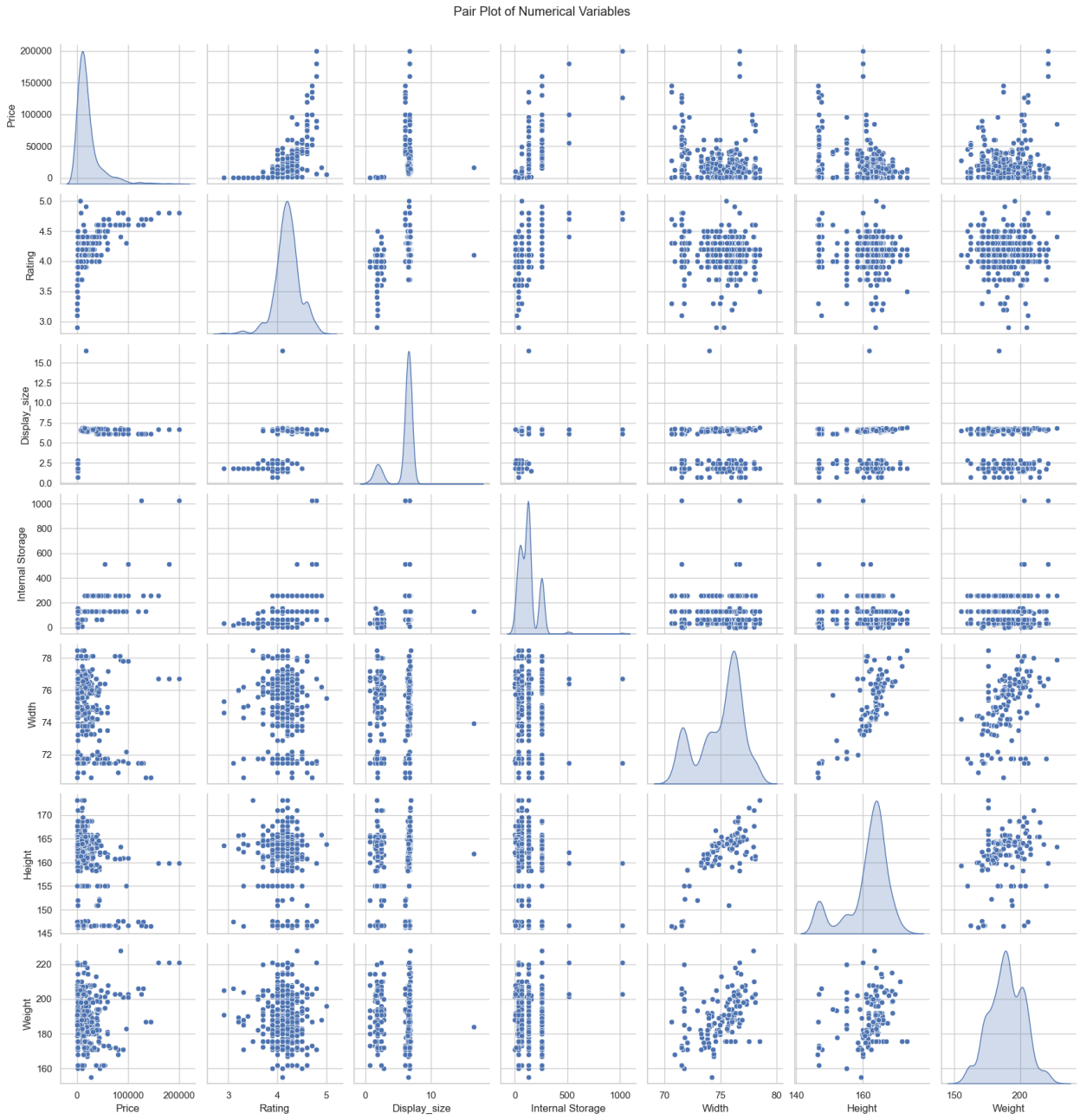

Distribution of Rating by Hybrid Sim Slot
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Density
Hybrid Sim Slot
No
Yes
3.0
3.5
4.0
4.5
5.0
Rating


Distribution of Display_size by Touchscreen
1.6
Touchscreen
Yes
No

6.Multivariate Analysis

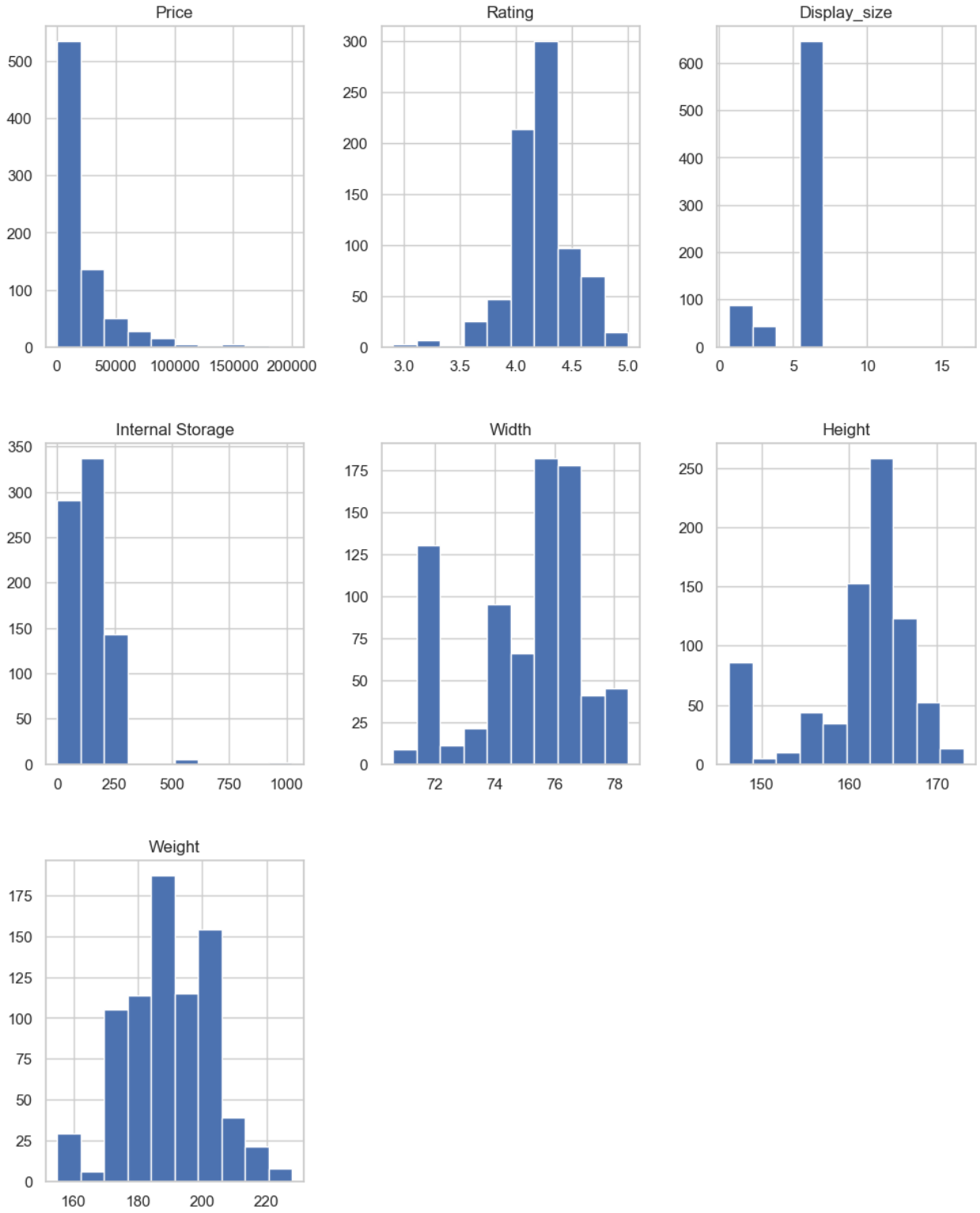
Pair plots

```
sns.pairplot(mobile[numerical_cols], diag_kind='kde', markers='o')
plt.suptitle("Pair Plot of Numerical Variables", y=1.02)
plt.show()
```



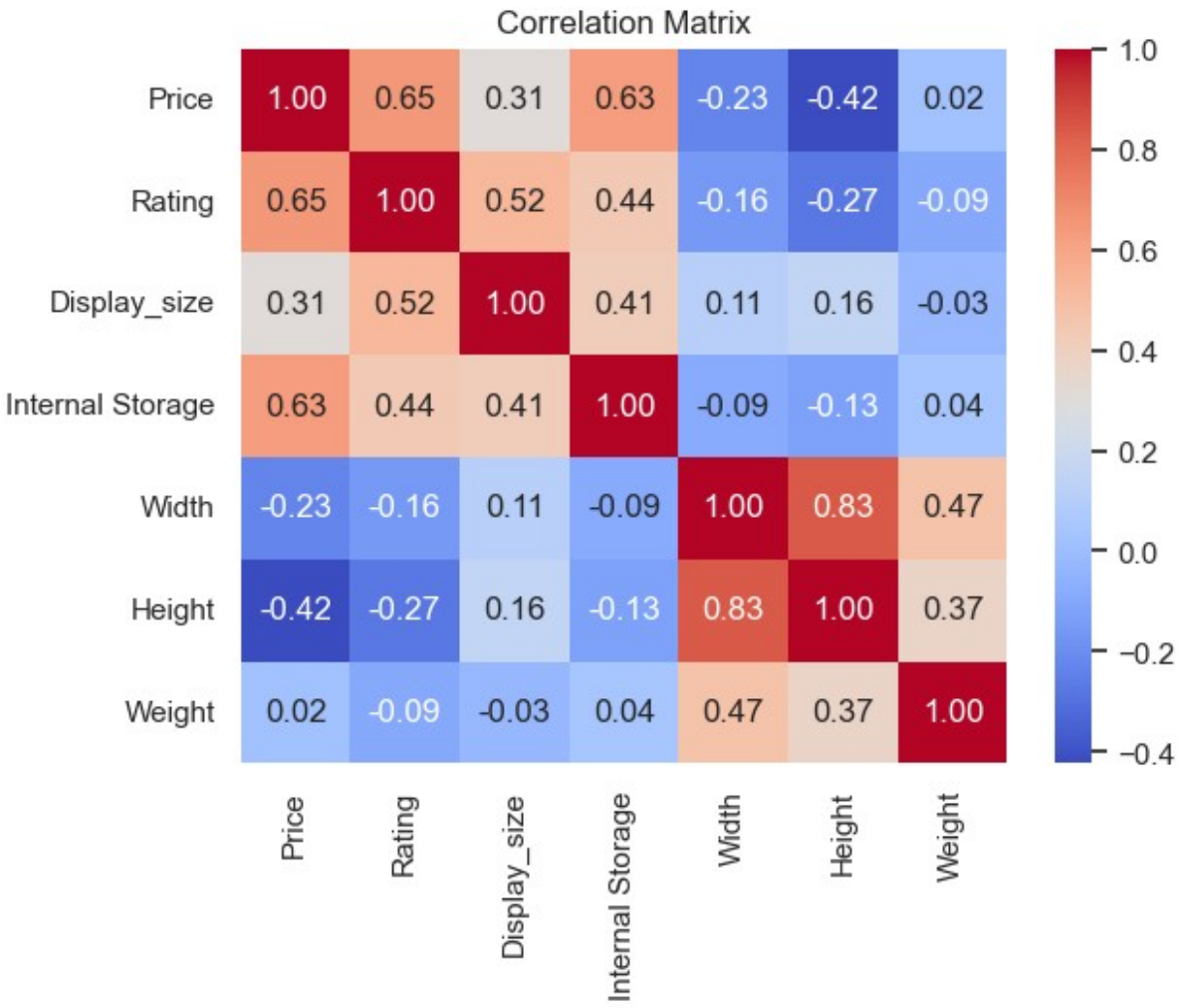
Histogram

```
mobile.hist(bins=10,figsize=[12,15])
plt.show()
```



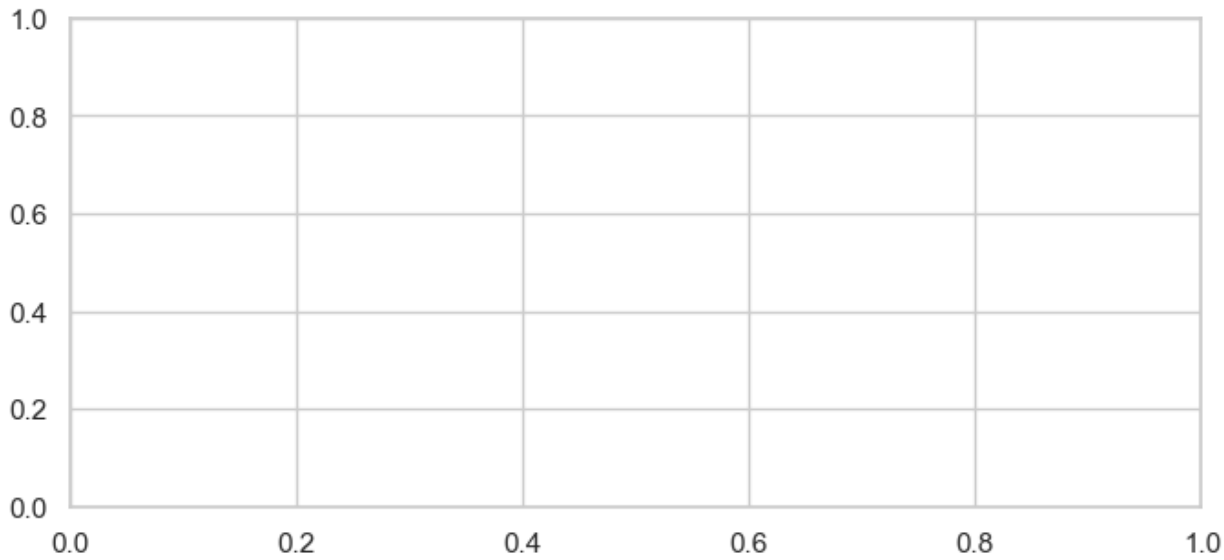
Heatmap

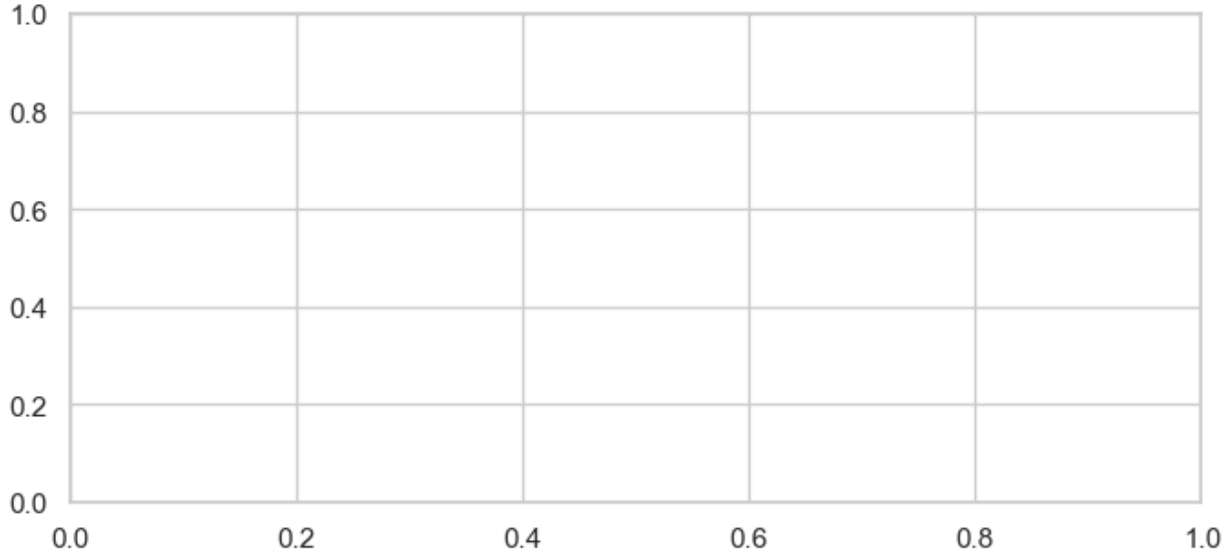
```
correlation_matrix = mobile.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```

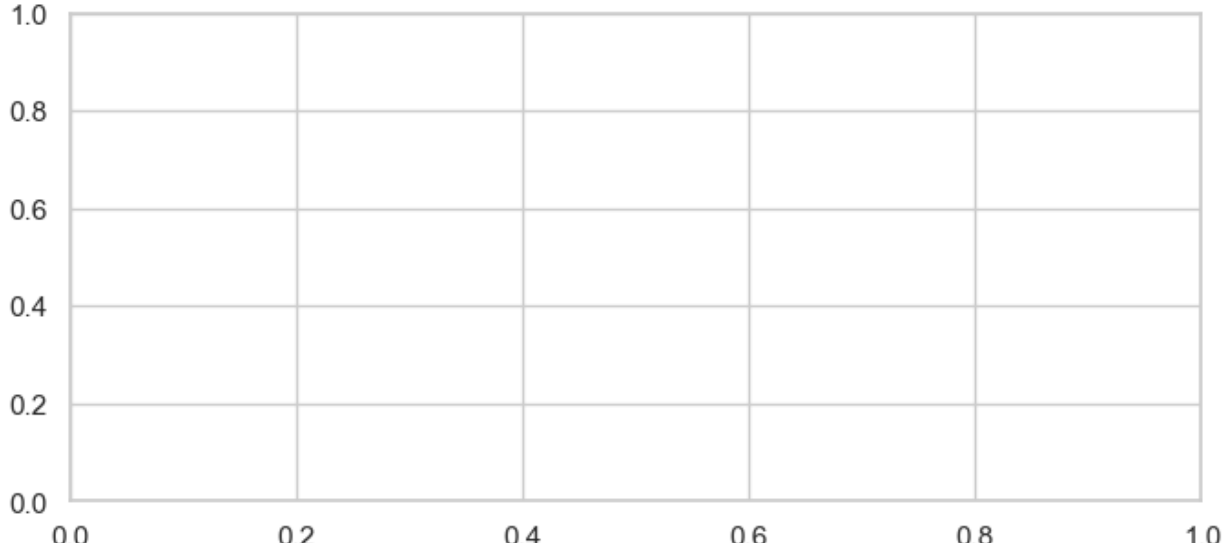


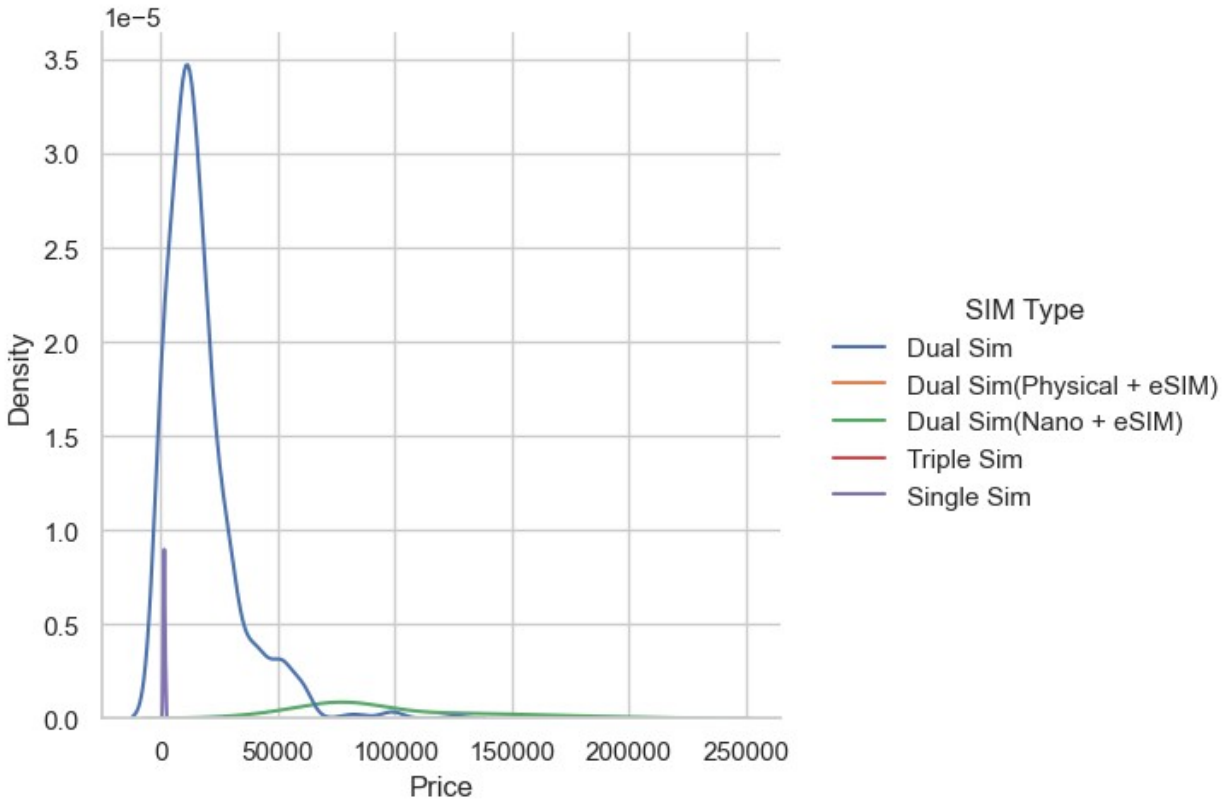

```
sns.set(style="whitegrid")

fig, axes = plt.subplots(4, 1, figsize=(8, 16))

sns.displot(data=mobile, x='Price', hue='SIM Type', ax=axes[0],
kind='kde')

sns.displot(data=mobile, x='Rating', hue='SIM Type', ax=axes[1],
kind='kde')

sns.displot(data=mobile, x='Display_size', hue='SIM Type', ax=axes[2],
```

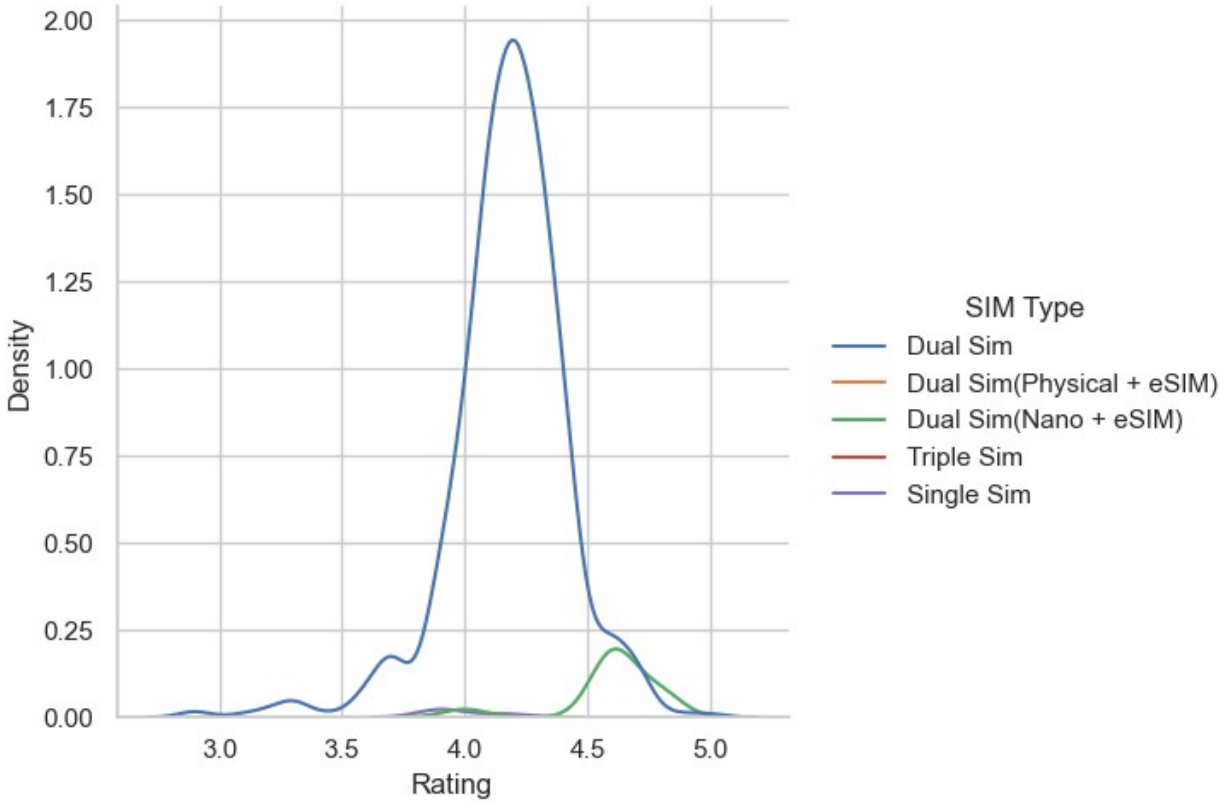
```
kind='kde')

sns.displot(data=mobile, x='Weight', hue='SIM Type', ax=axes[3],
kind='kde')

plt.tight_layout()
plt.show()
```

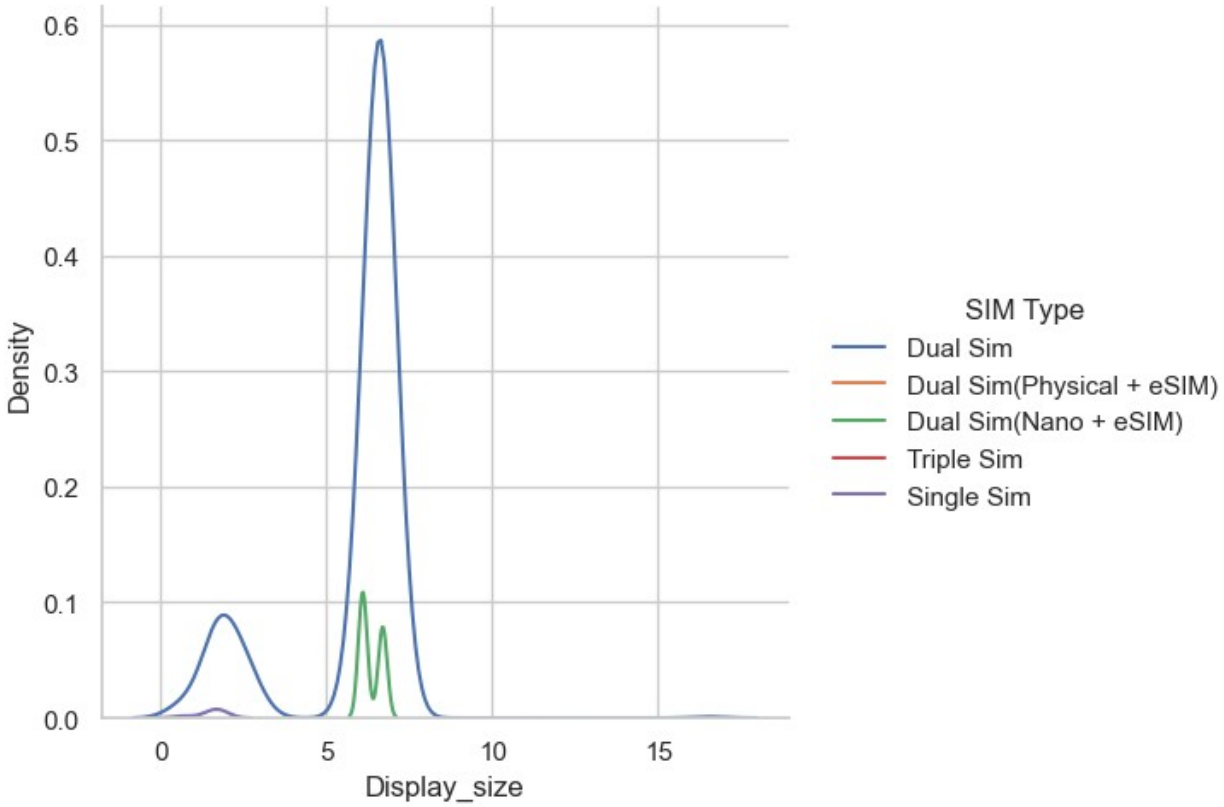

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

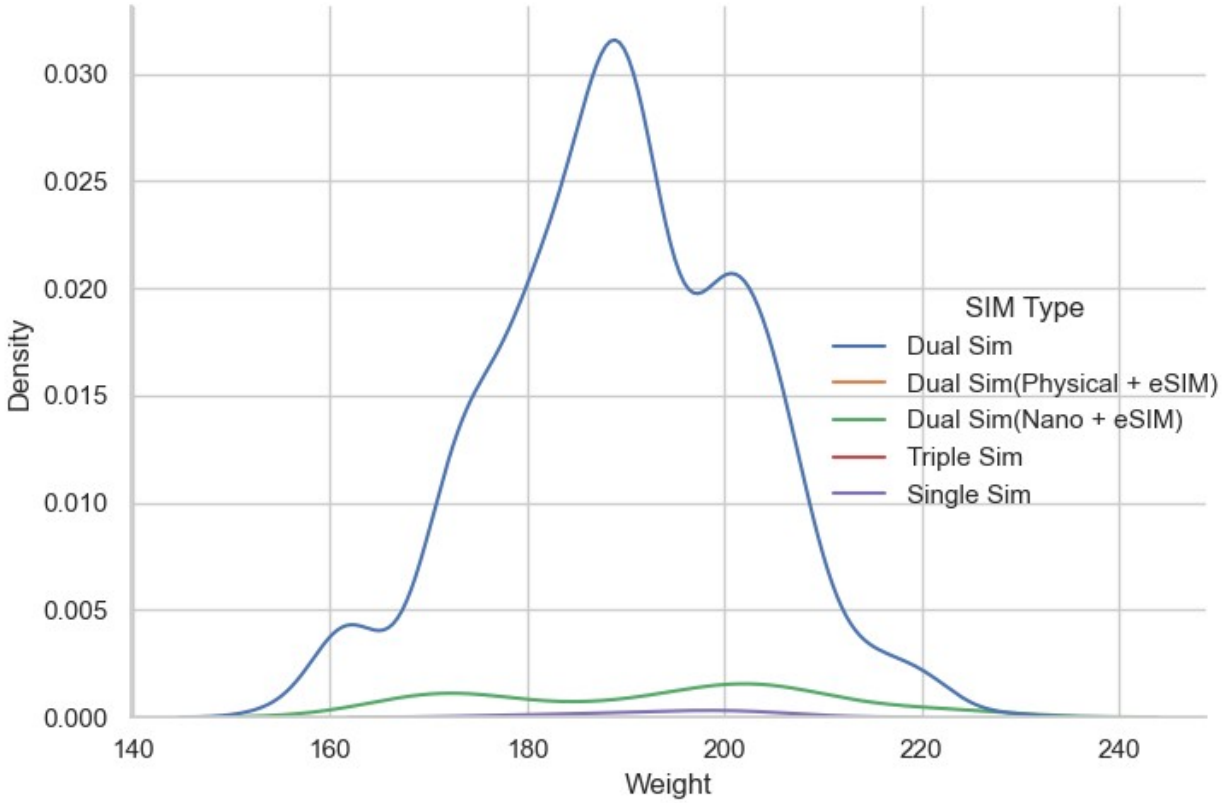

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


1e−5
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Density
0
50000
100000
150000
200000
250000
Price
SIM Type
Dual Sim
Dual Sim(Physical + eSIM)
Dual Sim(Nano + eSIM)
Triple Sim
Single Sim


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Density
3.0
3.5
4.0
4.5
5.0
Rating
SIM Type
Dual Sim
Dual Sim(Physical + eSIM)
Dual Sim(Nano + eSIM)
Triple Sim
Single Sim


Density
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
5
10
15
Display_size
SIM Type
Dual Sim
Dual Sim(Physical + eSIM)
Dual Sim(Nano + eSIM)
Triple Sim
Single Sim


Density
0.030
0.025
0.020
0.015
0.010
0.005
0.000
140
160
180
200
220
240
Weight
SIM Type
Dual Sim
Dual Sim(Physical + eSIM)
Dual Sim(Nano + eSIM)
Triple Sim
Single Sim

7.Hypothesis Testing

1. Comparison of Means: Independent Two-Sample t-test

Null Hypothesis (H0): The mean prices of smartphones from brand 'A' and brand 'B' are equal.
Alternative Hypothesis (H1): The mean prices of smartphones from brand 'A' and brand 'B' are not equal. Steps:

```
from scipy.stats import ttest_ind

Apple_prices = mobile[mobile['Brand'] == 'APPLE']['Price']
Samsung_prices = mobile[mobile['Brand'] == 'SAMSUNG']['Price']

t_stat, p_value = ttest_ind(Apple_prices, Samsung_prices,
equal_var=False)

# Display results
print("T-statistic:", t_stat)
print("P-value:", p_value)
print("\n")

# Assumed significance level 95%
alpha = 0.05

if p_value < alpha:
    print("Reject the null hypothesis: There is evidence that the
Apple_mean is not equal to Samsung_mean.")
else:
    print("Fail to reject the null hypothesis: There is no evidence to
suggest that the Apple_mean is not equal to Samsung_mean.")
```

```
T-statistic: 13.406278892944565
P-value: 5.981714444739929e-25


Reject the null hypothesis: There is evidence that the Apple_mean is
not equal to Samsung_mean.
```

2.Correlation Test: Pearson Correlation

Null Hypothesis (H0): There is no correlation between battery capacity and phone weight.
Alternative Hypothesis (H1): There is a significant correlation between battery capacity and phone weight.

```
from scipy.stats import pearsonr

correlation, p_value = pearsonr(mobile['Display_size'],
mobile['Weight'])
```

```
# Display results
print('Correlation:', correlation)
print('p-value:' ,p_value)
print("\n")

# Assumed significance level 95%
alpha = 0.05

# Assumed significance level 95%
alpha = 0.05

if p_value < alpha:
    print("Reject the null hypothesis: There is evidence that there is
no correlation between display size and phone weight.")
else:
    print("Fail to reject the null hypothesis: There is no evidence to
suggest that there is no correlation between display size and phone
weight.")
```

```
Correlation: -0.02741574134167696
p-value: 0.4450986675322601


Fail to reject the null hypothesis: There is no evidence to suggest
that there is no correlation between display size and phone weight.
```

8.Questions for Analysis

1. What are the names and data types of the columns?

```
# Gives an overview of columns in teh dataset
mobile.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 778 entries, 0 to 983
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Brand              778 non-null    object
 1   Model Name         778 non-null    object
 2   Price              778 non-null    int64
 3   Rating             778 non-null    float64
 4   SIM Type           778 non-null    object
 5   Hybrid Sim Slot    778 non-null    object
 6   Touchscreen        778 non-null    object
 7   Display_size       778 non-null    float64
 8   Operating System   778 non-null    object
 9   Processor Core     778 non-null    object
 10  Internal Storage   778 non-null    float64
 11  Primary Camera     778 non-null    object
```

```
 12  Secondary Camera   778 non-null    object
 13  Network Type       778 non-null    object
 14  Bluetooth Version  778 non-null    object
 15  Wi-Fi              778 non-null    object
 16  GPS Support        778 non-null    object
 17  SIM Size           778 non-null    object
 18  Battery Capacity   778 non-null    object
 19  Width              778 non-null    float64
 20  Height             778 non-null    float64
 21  Weight             778 non-null    float64
 22  SIM Type.1         778 non-null    object
 23  Hybrid Sim Slot.1  778 non-null    object
 24  Dual Camera Lens   778 non-null    object
 25  Color              778 non-null    object
dtypes: float64(6), int64(1), object(19)
memory usage: 180.3+ KB
```

All ready mentioned in above Section

2. What are the basic summary statistics?

```
# Basic description about columns in the dataset
mobile.describe()
```

```
                Price      Rating  Display_size  Internal Storage
Width  \
count     778.000000  778.000000    778.000000        778.000000
778.000000
mean    21796.984576    4.205398      5.803436        126.308916
74.996629
std     26024.828639    0.258292      1.814605         92.579396
1.963075
min       597.000000    2.900000      0.660000          0.000000
70.600000
25%      7991.250000    4.100000      6.380000         64.000000
73.820000
50%     13999.000000    4.200000      6.590000        128.000000
75.750000
75%     24999.000000    4.300000      6.700000        128.000000
76.400000
max    199900.000000    5.000000     16.510000       1024.000000
78.460000

           Height      Weight
count  778.000000  778.000000
mean   161.308723  189.686804
std      6.249525   12.964685
min    146.300000  155.000000
25%    159.900000  181.000000
50%    163.600000  189.500000
```

```
75%    164.900000  199.800000
max    173.100000  228.000000

Provides a statistical summary of a mobile phone dataset.
Includes central tendency, dispersion, and shape measures.
Useful for identifying outliers, trends, and relationships between
metrics.
```

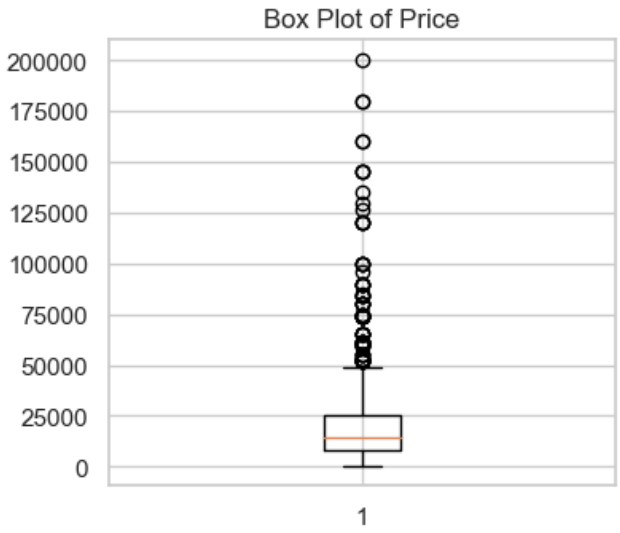
3.Are there any categorical variables and missing values? If so, print it

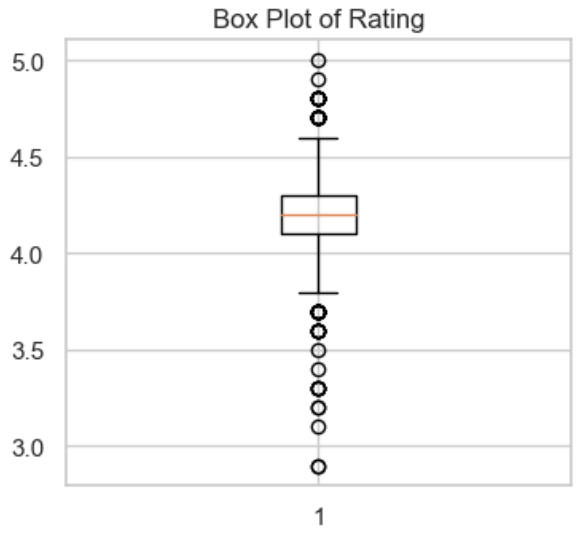
```
categorical_cols = mobile.select_dtypes(exclude=[np.number]).columns
print(categorical_cols)

#Null values
mobile.isnull().sum()

Index(['Brand', 'Model Name', 'SIM Type', 'Hybrid Sim Slot',
'Touchscreen',
       'Operating System', 'Processor Core', 'Primary Camera',
       'Secondary Camera', 'Network Type', 'Bluetooth Version', 'Wi-
Fi',
       'GPS Support', 'SIM Size', 'Battery Capacity', 'SIM Type.1',
       'Hybrid Sim Slot.1', 'Dual Camera Lens', 'Color'],
      dtype='object')

Brand                0
Model Name           0
Price                0
Rating               0
SIM Type             0
Hybrid Sim Slot      0
Touchscreen          0
Display_size         0
Operating System     0
Processor Core       0
Internal Storage     0
Primary Camera       0
Secondary Camera     0
Network Type         0
Bluetooth Version    0
Wi-Fi                0
GPS Support          0
SIM Size             0
Battery Capacity     0
Width                0
Height               0
Weight               0
SIM Type.1           0
Hybrid Sim Slot.1    0
Dual Camera Lens     0
```

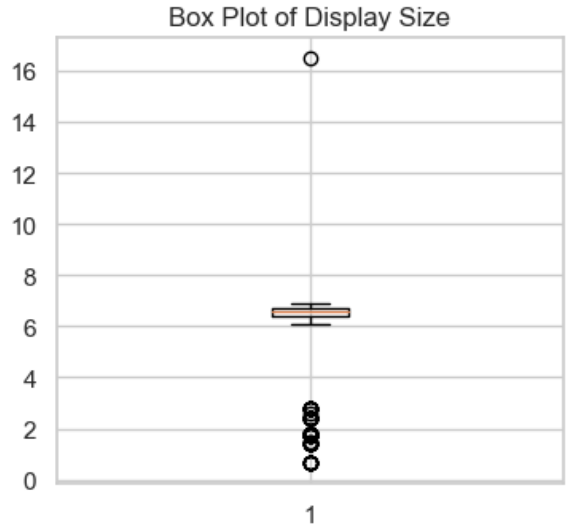
```
Color                 0
dtype: int64

Mobile data set contains a categorical data.
As a significant amount column are belong to categorical.
```

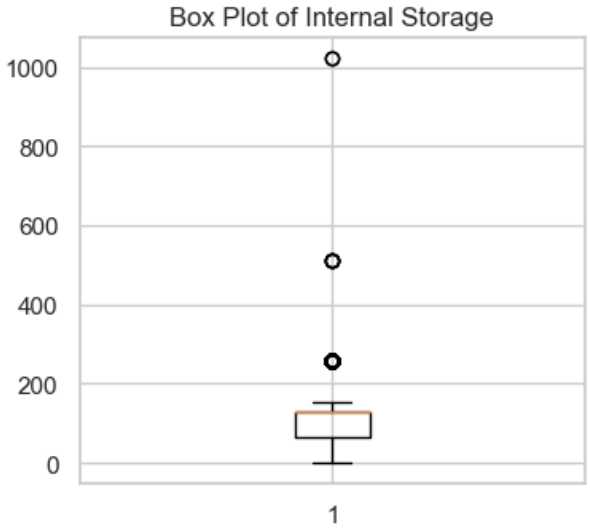
4. Are there any outliers in the data? If so, use box plots, histograms and visualize.

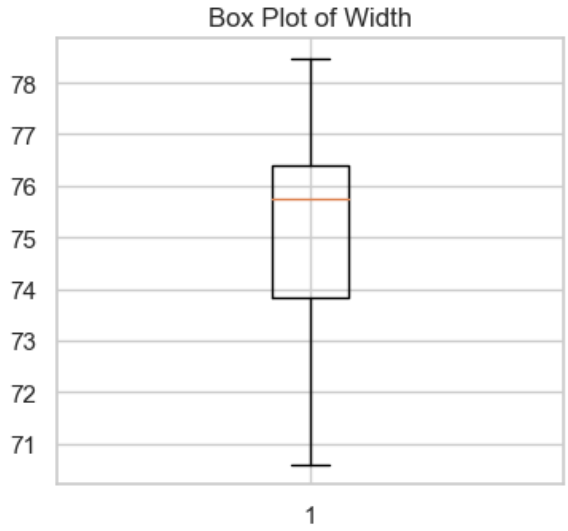
```
# Outlier detection using boxplot
fig, axes = plt.subplots(4, 2, figsize=(8, 14))

axes[0, 0].boxplot(mobile['Price'])
axes[0, 0].set_title('Box Plot of Price')

axes[0, 1].boxplot(mobile['Rating'])
axes[0, 1].set_title('Box Plot of Rating')

axes[1, 0].boxplot(mobile['Display_size'])
axes[1, 0].set_title('Box Plot of Display Size')

axes[1, 1].boxplot(mobile['Internal Storage'])
axes[1, 1].set_title('Box Plot of Internal Storage')

axes[2, 0].boxplot(mobile['Width'])
axes[2, 0].set_title('Box Plot of Width')

axes[2, 1].boxplot(mobile['Height'])
axes[2, 1].set_title('Box Plot of Height')

axes[3, 0].boxplot(mobile['Weight'])
axes[3, 0].set_title('Box Plot of Weight')

axes[3, 1].axis('off')


plt.tight_layout()
plt.show()
```

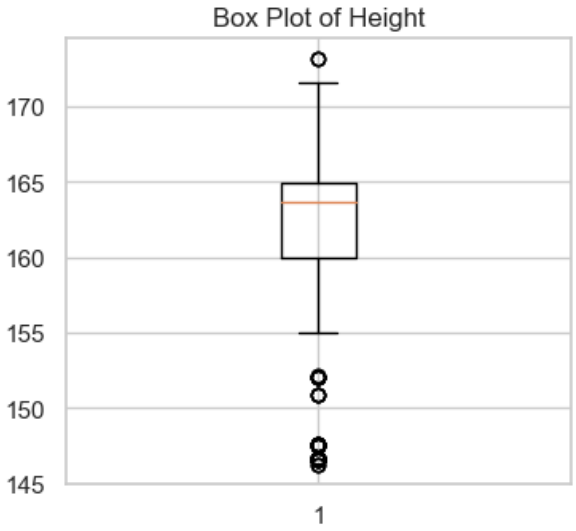

Box Plot of Price
200000
175000
150000
125000
100000
75000
50000
25000
0
1

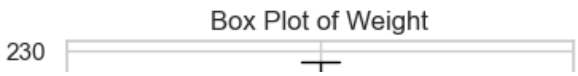

Box Plot of Rating
5.0
4.5
4.0
3.5
3.0
1


Box Plot of Display Size
16
14
12
10
8
6
4
2
0
1


Box Plot of Internal Storage
1000
800
600
400
200
0
1


Box Plot of Width
78
77
76
75
74
73
72
71
1


Box Plot of Height
170
165
160
155
150
145
1


Box Plot of Weight
230

The code generates a set of box plots to visually identify outliers in key numerical columns of the mobile dataset.

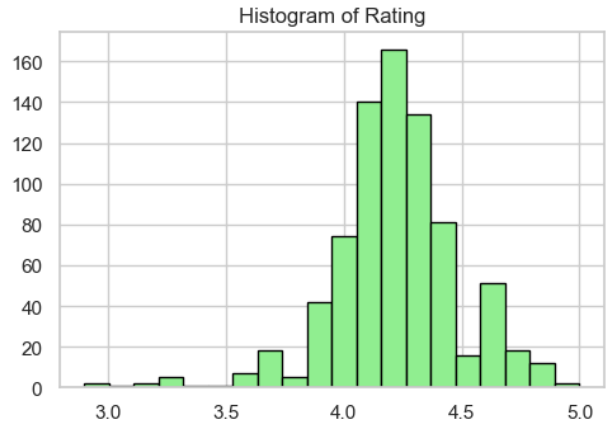
Each box plot represents the distribution of a specific variable, such as 'Price,' 'Rating,' 'Display Size,' 'Internal Storage,' 'Width,' 'Height,' and 'Weight.'

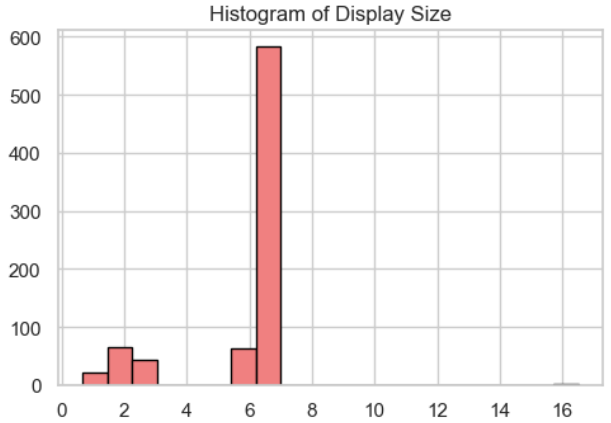
Outliers, or extreme values, can be detected by observing data points that fall outside the whiskers of the box plots.

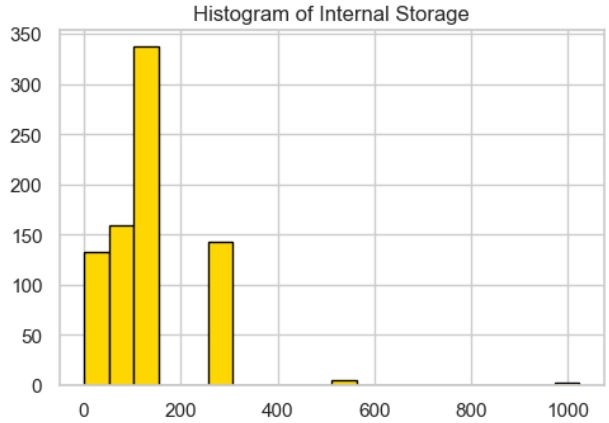
The absence of a box plot in the last subplot indicates that the 'Weight' column is not plotted for outlier detection.

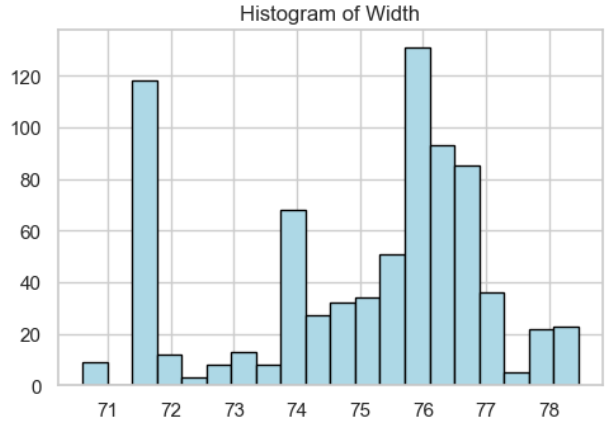
```
# Outlier detection using histograms
# Histograms for columns
fig, axes = plt.subplots(4, 2, figsize=(10, 14))
6
# Create histograms and set titles for each subplot
axes[0, 0].hist(mobile['Price'], bins=20, color='skyblue',
edgecolor='black')
axes[0, 0].set_title('Histogram of Price')

axes[0, 1].hist(mobile['Rating'], bins=20, color='lightgreen',
edgecolor='black')
axes[0, 1].set_title('Histogram of Rating')

axes[1, 0].hist(mobile['Display_size'], bins=20, color='lightcoral',
edgecolor='black')
axes[1, 0].set_title('Histogram of Display Size')

axes[1, 1].hist(mobile['Internal Storage'], bins=20, color='gold',
edgecolor='black')
axes[1, 1].set_title('Histogram of Internal Storage')

axes[2, 0].hist(mobile['Width'], bins=20, color='lightblue',
edgecolor='black')
axes[2, 0].set_title('Histogram of Width')

axes[2, 1].hist(mobile['Height'], bins=20, color='pink',
edgecolor='black')
axes[2, 1].set_title('Histogram of Height')

axes[3, 0].hist(mobile['Weight'], bins=20, color='lightgrey',
edgecolor='black')
axes[3, 0].set_title('Histogram of Weight')

axes[3, 1].axis('off')

plt.tight_layout()
plt.show()
```

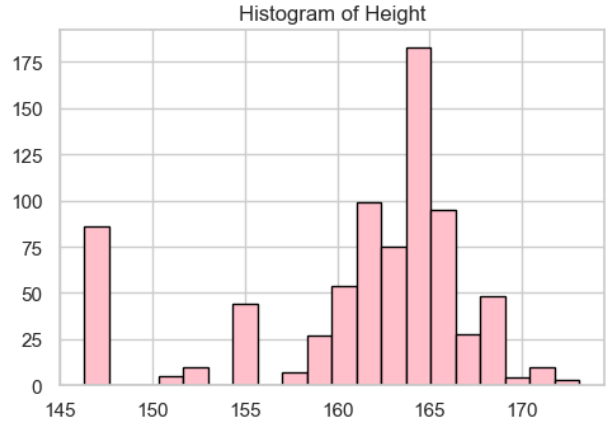

Histogram of Price
0
50
100
150
200
250
0
25000
50000
75000
100000
125000
150000
175000
200000

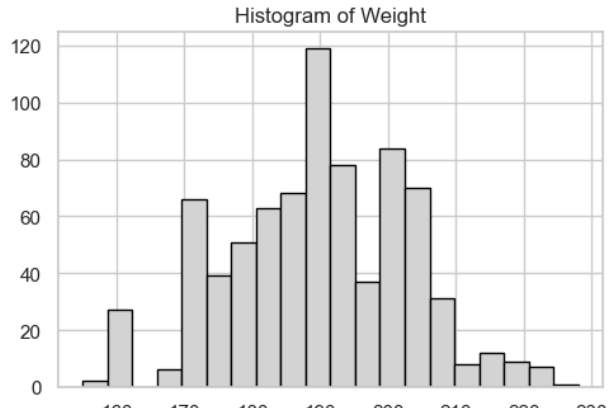

Histogram of Rating
0
20
40
60
80
100
120
140
160
3.0
3.5
4.0
4.5
5.0


Histogram of Display Size
0
100
200
300
400
500
600
0
2
4
6
8
10
12
14
16


Histogram of Internal Storage
0
50
100
150
200
250
300
350
0
200
400
600
800
1000

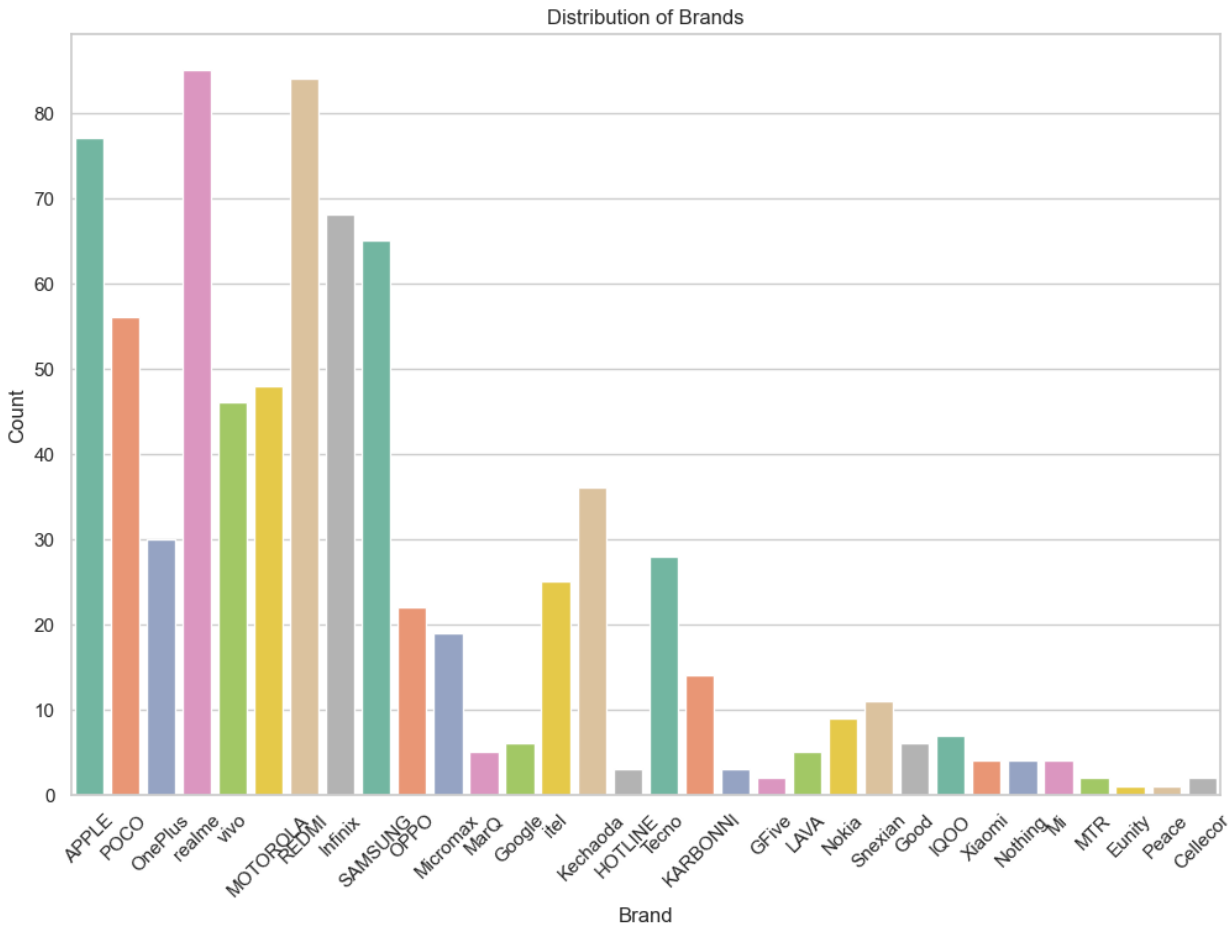

Histogram of Width
0
20
40
60
80
100
120
71
72
73
74
75
76
77
78


Histogram of Height
0
25
50
75
100
125
150
175
145
150
155
160
165
170


Histogram of Weight
0
20
40
60
80
100
120

5. Is the data balanced or imbalanced? Visualize.

```
# Asuuming target variable as Brand

plt.figure(figsize=(12, 8))
sns.countplot(data=mobile, x='Brand', palette='Set2')
plt.title('Distribution of Brands')
plt.xlabel('Brand')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



From the above graphical representation we ca reach unambiguous conclusion that the data is not balanced

6. What is the target variable (if any).

The target variable is Price because it is necessary for us to analyse any mobile and is a must to evaluate the features for any mobile.

7. What are the units of measurement for numerical columns?

-- The units of numerical columns are as follows :

Price : Rupees

Rating : No units

Display_size Inches

Internal Storage : GB

Width : Millimeter(mm)

Height : Millimeter(mm)

Weight : Grams(g)

8. Do you have domain clarification? Brief it.

Domain: Mobile Technology Market Analysis

This dataset resides within the domain of the Mobile Technology Market, offering a detailed exploration of the contemporary landscape of mobile phones. Capturing essential attributes such as brand, model name, pricing, user ratings, SIM types, and diverse technical specifications, the dataset serves as a valuable resource for understanding the intricacies of the mobile technology sector.

Within this domain, we aim to uncover patterns, trends, and relationships that define consumer preferences, technological advancements, and competitive dynamics. The dataset provides a comprehensive overview of mobile phone features, enabling a thorough analysis of factors influencing consumer choices and market trends. The insights derived from this exploration will contribute to a deeper understanding of the evolving dynamics within the Mobile Technology Market.

9. Are there any time-based trends or patterns?

Upon thorough analysis, I am unable to discern discernible trends and patterns within the provided data, particularly in relation to time patterns.

10. Are there any correlations between variables? Calculate correlations.

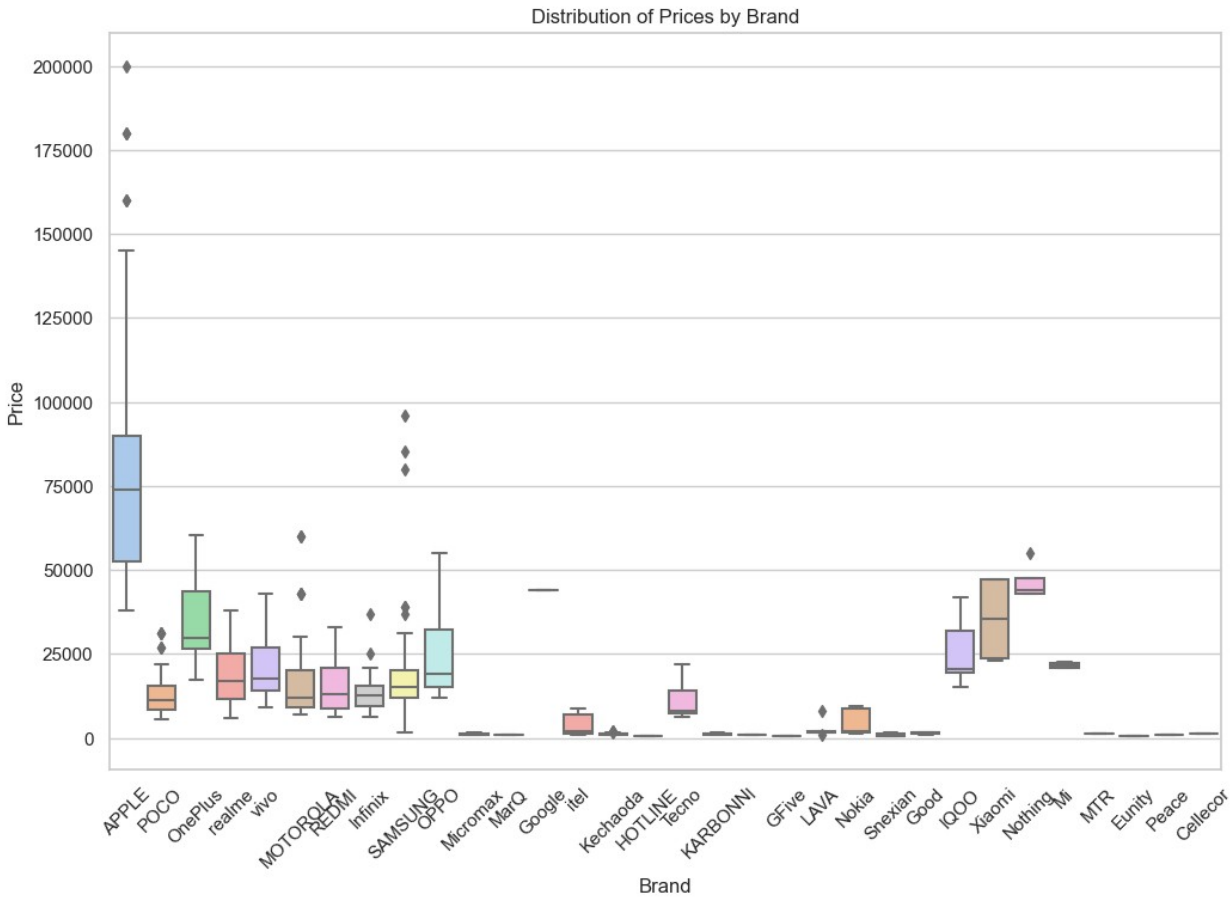
-- The corelation between variables is represented usinga heatmap.

```
correlation_matrix = mobile[numerical_cols].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='Pastel1')
plt.title('Correlation Matrix')
plt.show()
```

This concise visualization offers insights into the linear relationships between numerical features, with a title indicating its purpose as a 'Correlation Matrix.'

11. Do different 'Brands' have significantly different 'Prices'?

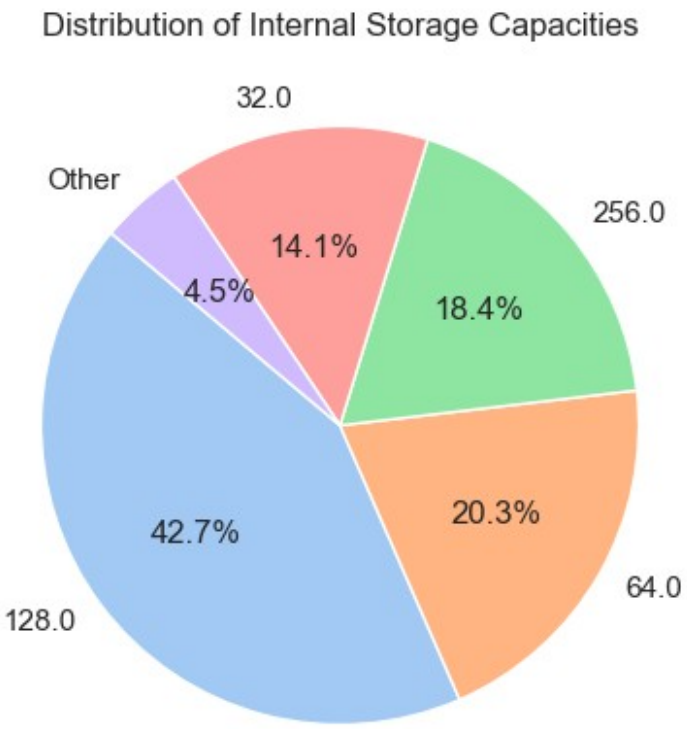
```
sns.set(style="whitegrid")

# Create a box plot for 'Prices' by 'Brand'
plt.figure(figsize=(12, 8))
sns.boxplot(data=mobile, x='Brand', y='Price', palette='pastel')
plt.title('Distribution of Prices by Brand')
plt.xlabel('Brand')
plt.ylabel('Price')
plt.xticks(rotation=45)
plt.show()
```



Without any second doubt it is clear that there is significant difference in the prices. Mostly Apple has a lot of variation from every brand

12.What is the distribution of internal storage capacities among mobile devices?

```
internal_storage_counts = mobile['Internal Storage'].value_counts()

threshold = 0.1
small_categories = internal_storage_counts[internal_storage_counts /
internal_storage_counts.sum() < threshold].index

mobile['Internal Storage'] = mobile['Internal
Storage'].replace(small_categories, 'Other')

internal_storage_counts = mobile['Internal Storage'].value_counts()

plt.figure(figsize=(5, 5))
plt.pie(internal_storage_counts, labels=internal_storage_counts.index,
autopct='%1.1f%%', startangle=140, colors=sns.color_palette('pastel'))
plt.title('Distribution of Internal Storage Capacities')
plt.show()
```



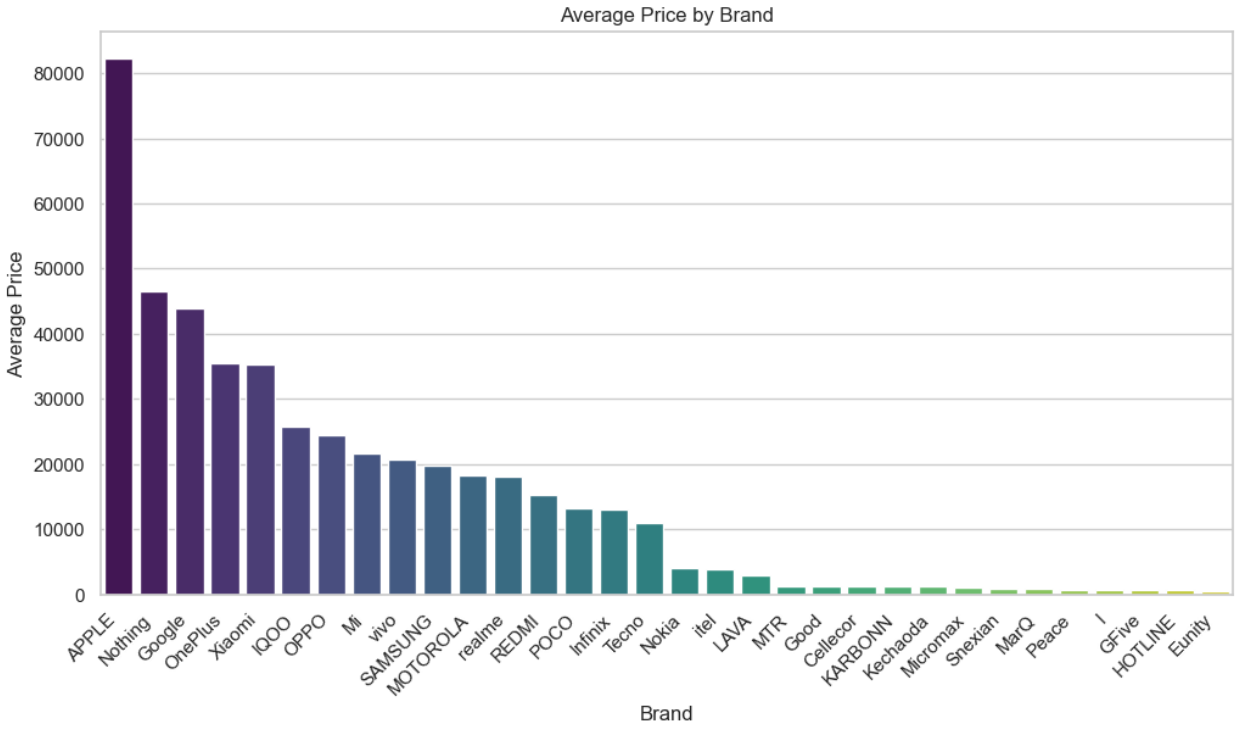
A large chunk of Brands in Present time prefer 128 Gb Storage Capacity followed by 64 gb and 256 gb. But from patterns in data set clearly shows that usage of 256 gb capacity mobiles increasing with a rapid speed

13.Which brand has the highest average price?

```
avg_brand_price = mobile.groupby('Brand')
['Price'].mean().sort_values(ascending=False)
print(avg_brand_price)

plt.figure(figsize=(12, 6))
sns.barplot(x=avg_brand_price.index, y=avg_brand_price.values,
palette='viridis')
plt.title('Average Price by Brand')
plt.xlabel('Brand')
plt.ylabel('Average Price')
plt.xticks(rotation=45, ha='right')
plt.show()
```

```
Brand
APPLE       82285.038961
Nothing     46499.000000
Google      43999.000000
OnePlus     35459.266667
Xiaomi      35246.750000
```

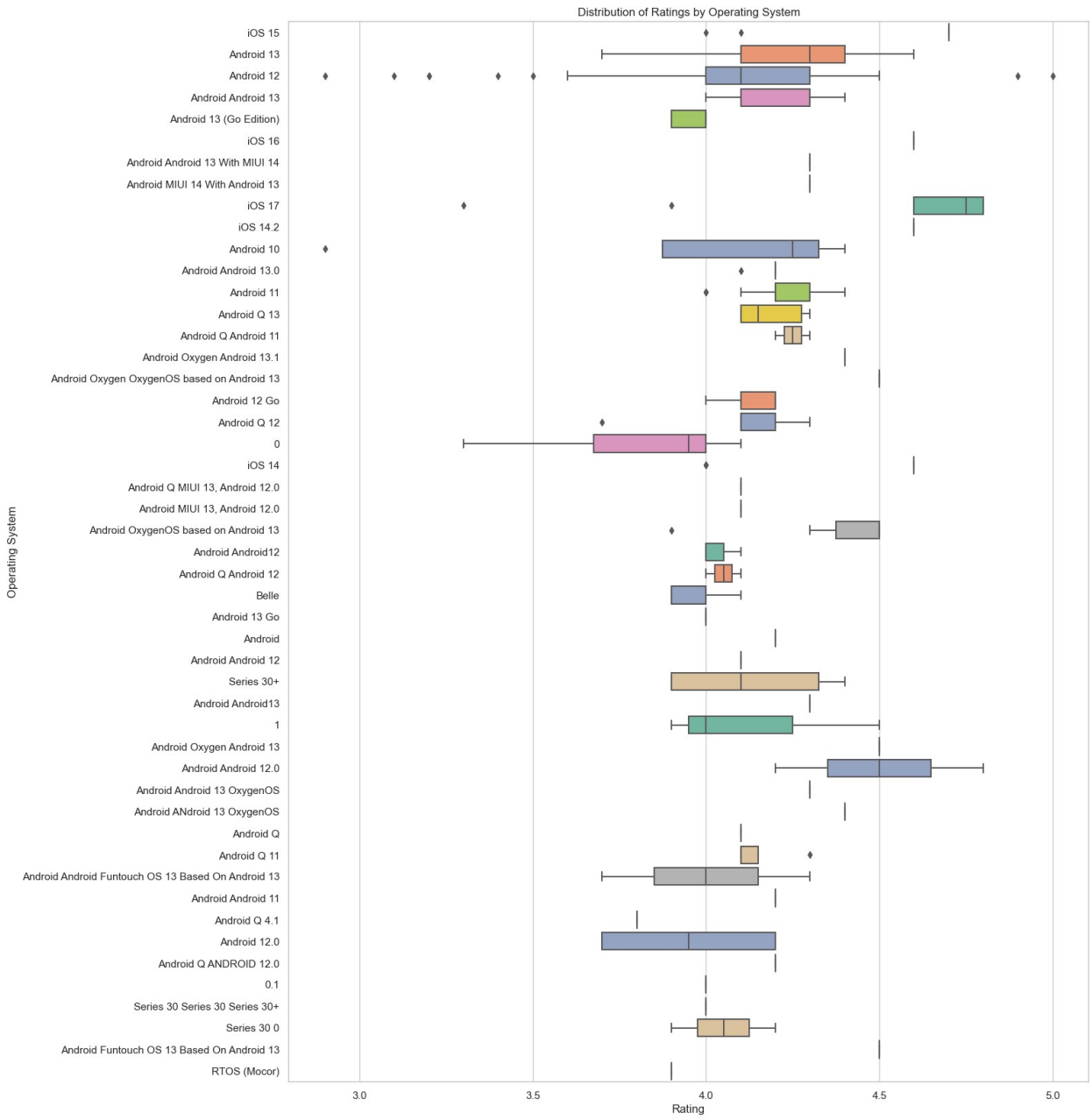
```
IQOO         25757.571429
OPPO         24512.090909
Mi           21619.500000
vivo         20695.326087
SAMSUNG      19854.200000
MOTOROLA     18339.645833
realme       18023.000000
REDMI        15372.440476
POCO         13339.910714
Infinix      13144.485294
Tecno        11056.714286
Nokia         4142.333333
itel          3883.640000
LAVA          2883.000000
MTR           1350.000000
Good          1324.166667
Cellecor      1313.000000
KARBONN       1227.714286
Kechaoda      1191.638889
Micromax      1098.842105
Snexian        942.181818
MarQ           899.000000
Peace          749.000000
I              744.000000
GFive          699.000000
HOTLINE        649.000000
Eunity         597.000000
Name: Price, dtype: float64
```



As earlier questions without any hesitation answer will be Apple if it's about price and premium quality as it offers. But interestingly new player Nothing and rising giant google are trying to compete with apple

14.What is the distribution of ratings across different operating systems?

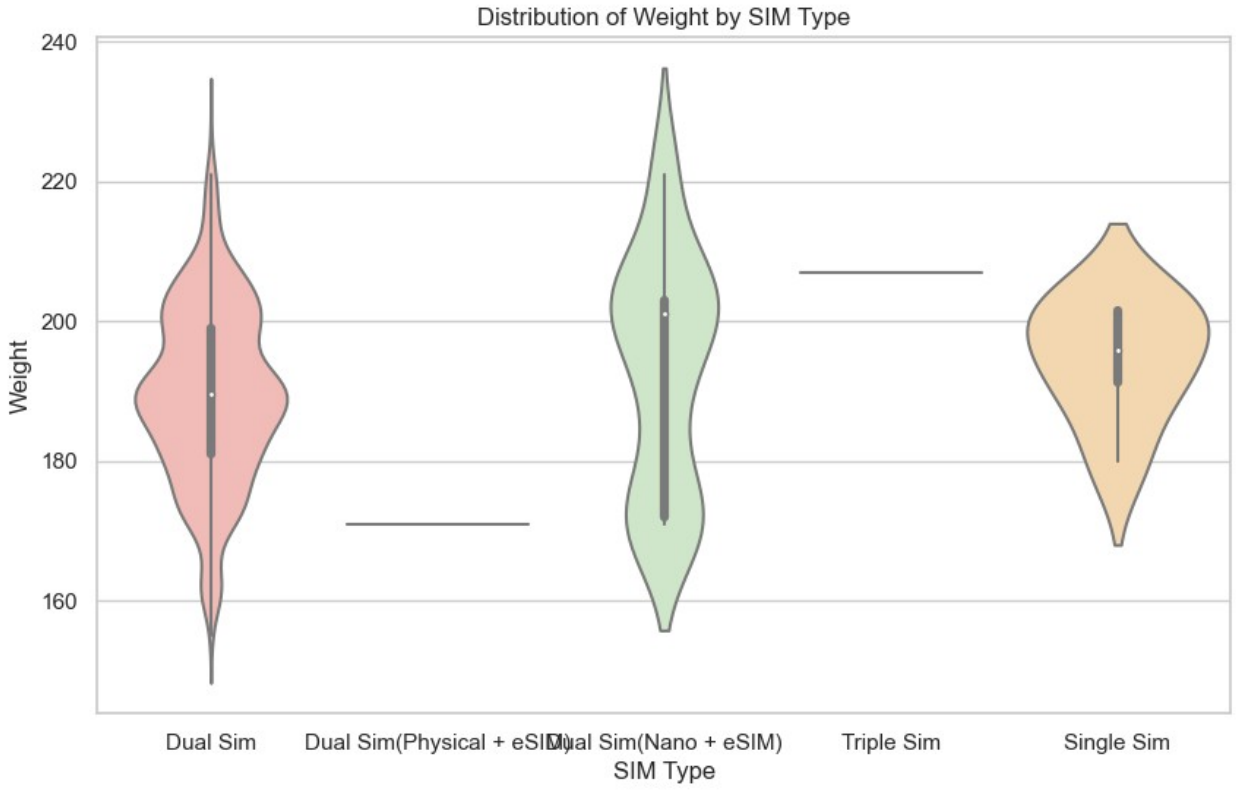
```
plt.figure(figsize=(15, 20))
sns.boxplot(data=mobile, x='Rating', y='Operating System',
palette='Set2')
plt.title('Distribution of Ratings by Operating System')
plt.xlabel('Rating')
plt.ylabel('Operating System')
plt.show()
```



Latest Operating Systems occupies TOP ratings and mostly IOS and new Android Versions are performing very Well

15.What is the distribution of 'Weight' for different SIM types?

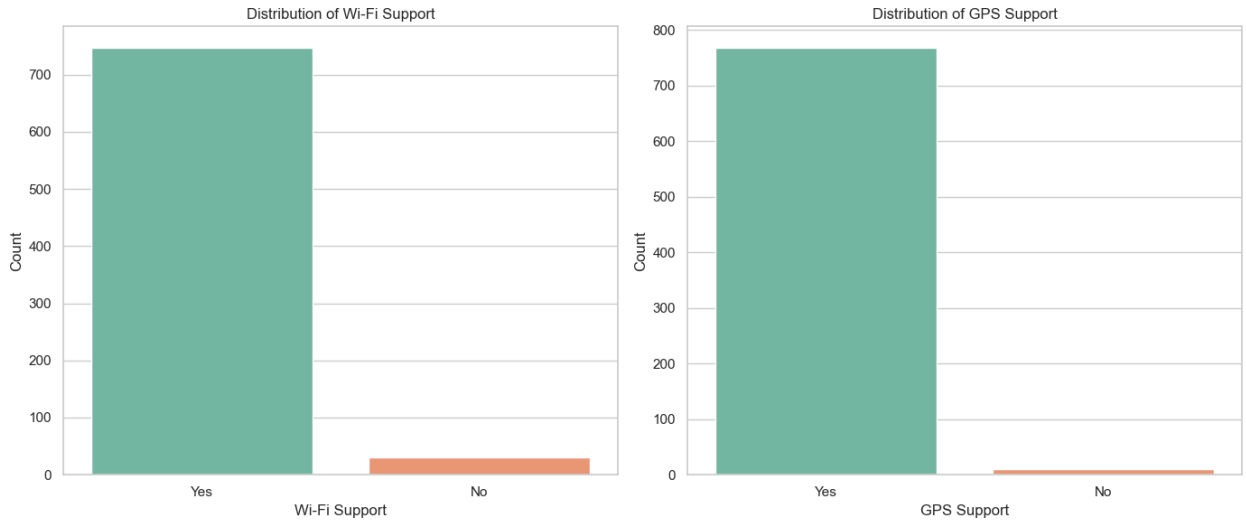
```
plt.figure(figsize=(10, 6))
sns.violinplot(data=mobile, x='SIM Type', y='Weight',
palette='Pastel1')
plt.title('Distribution of Weight by SIM Type')
plt.xlabel('SIM Type')
plt.ylabel('Weight')
plt.show()
```



Although Single accounts only sim but as it uses old technology which weight significantly some how high when it is compared yo its counterparts

16.How is the distribution of Wi-Fi and GPS support in the dataset?

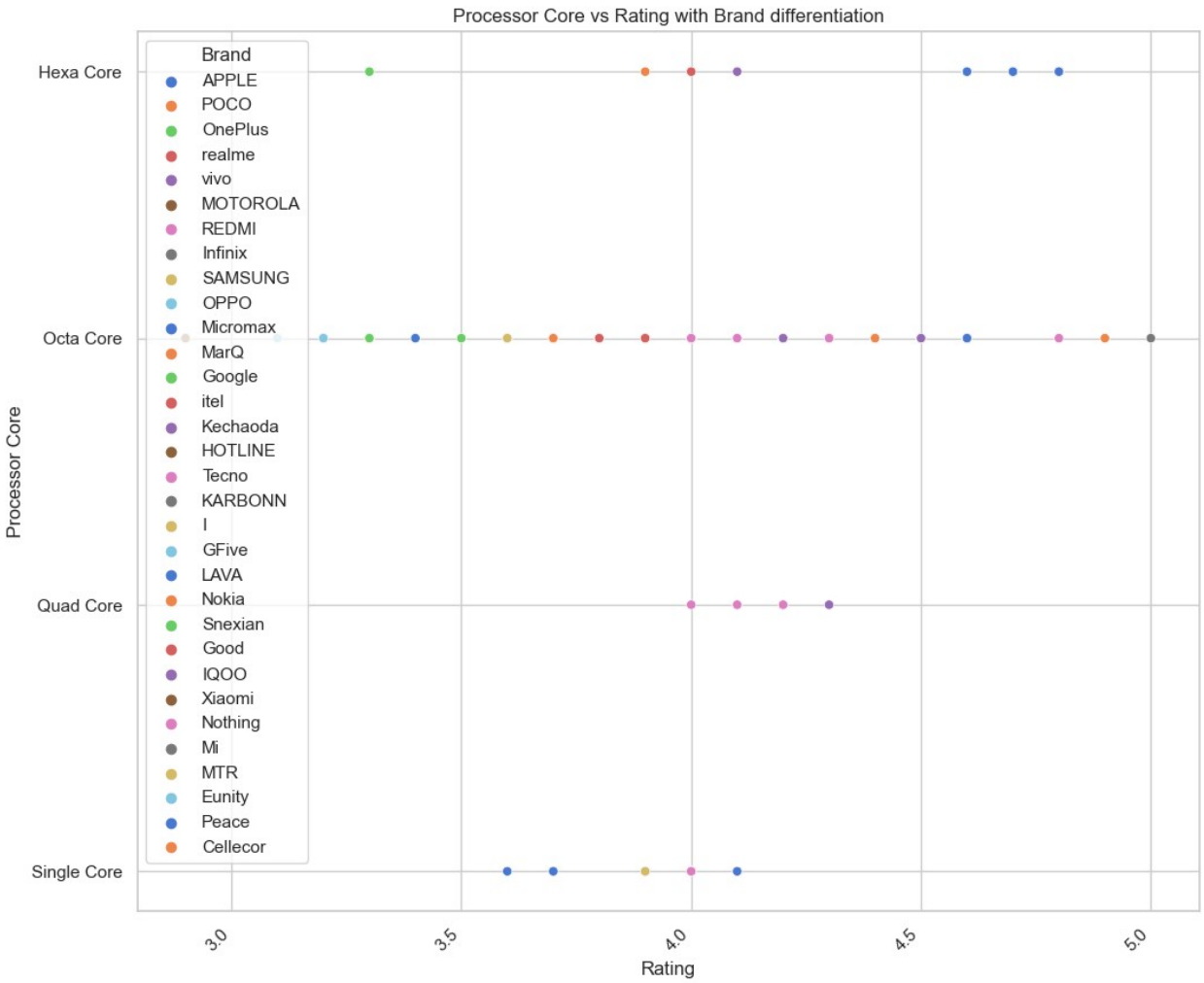
```
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# Count plot for Wi-Fi support
sns.countplot(data=mobile, x='Wi-Fi', palette='Set2', ax=axes[0])
axes[0].set_title('Distribution of Wi-Fi Support')
axes[0].set_xlabel('Wi-Fi Support')
axes[0].set_ylabel('Count')

# Count plot for GPS support
sns.countplot(data=mobile, x='GPS Support', palette='Set2',
ax=axes[1])
axes[1].set_title('Distribution of GPS Support')
axes[1].set_xlabel('GPS Support')
axes[1].set_ylabel('Count')

plt.tight_layout()
plt.show()
```



In recent days every mobile supports wi-fi and GPS as increasing purchase of smartPhones

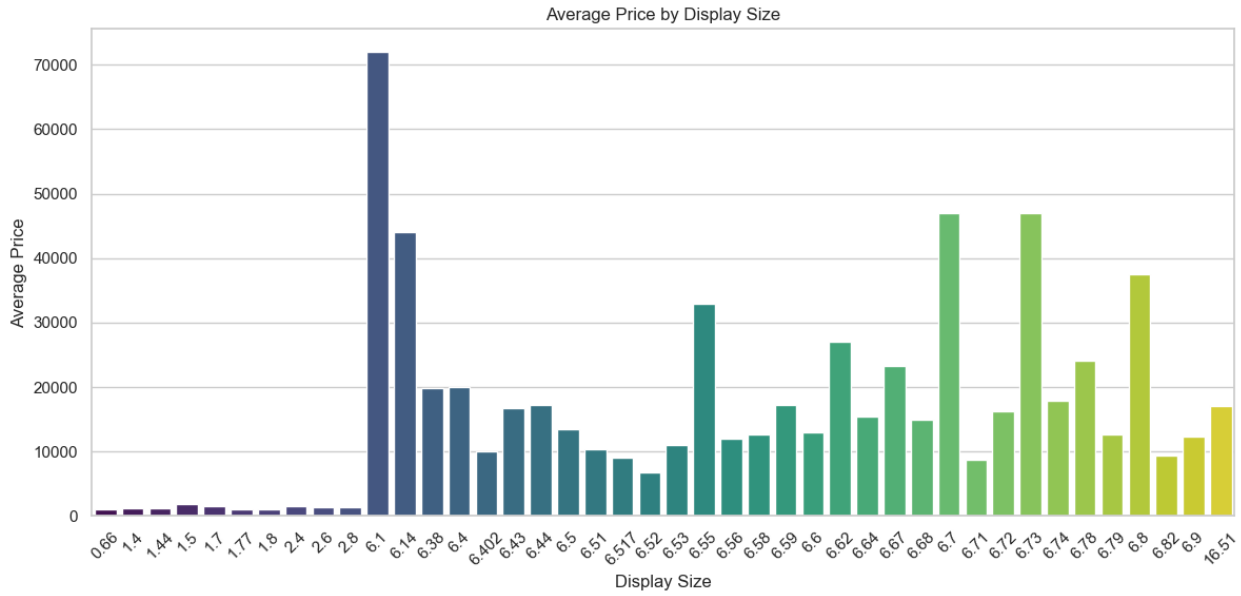
17.Is there a noticeable trend in ratings based on the processor core of device?

```
plt.figure(figsize=(12, 10))
sns.scatterplot(data=mobile, x='Rating', y='Processor Core',
hue='Brand', palette='muted')
plt.title('Processor Core vs Rating with Brand differentiation')
plt.xlabel('Rating')
plt.ylabel('Processor Core')
plt.xticks(rotation=45, ha='right')
plt.show()
```



18.Compare the average price of phones with different screen sizes

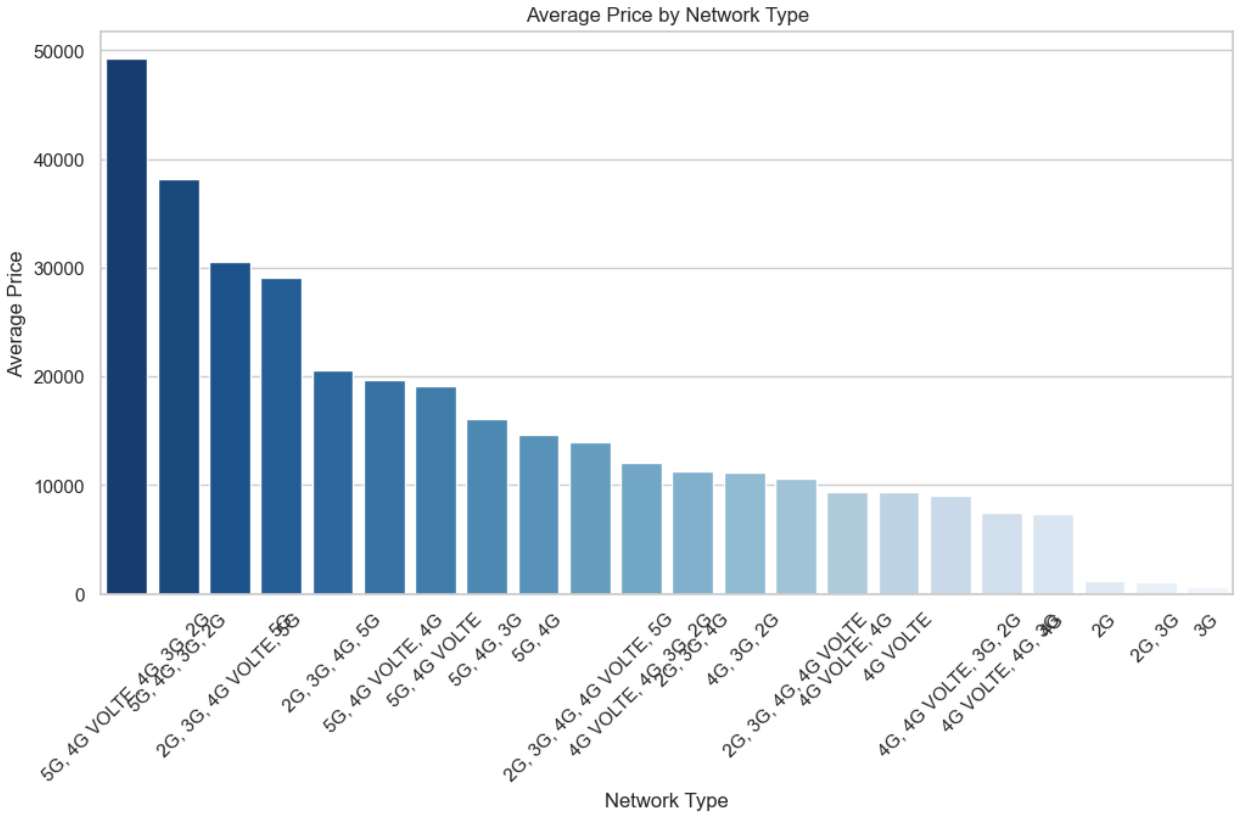
```
avg_price_by_display_size = mobile.groupby('Display_size')
['Price'].mean().sort_values(ascending=False)
plt.figure(figsize=(14, 6))
sns.barplot(x=avg_price_by_display_size.index,
y=avg_price_by_display_size.values, palette='viridis')
plt.title('Average Price by Display Size')
plt.xlabel('Display Size')
plt.ylabel('Average Price')
plt.xticks(rotation=45)
plt.show()
```



6.1 represents the most specialized category, containing the maximum number of mobile brands but price is very high .

19.How is the distribution of Network Types across different brands, and does the Network Type correlate with the average Price of the mobile devices?

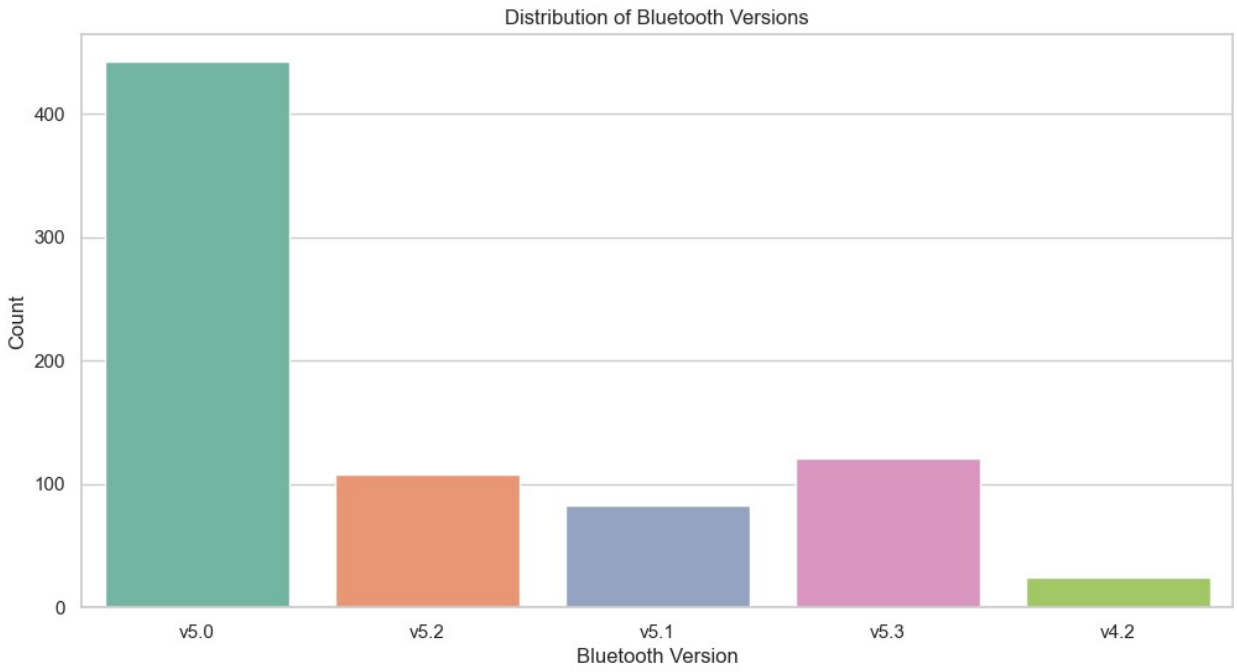
```
avg_price_by_network = mobile.groupby('Network Type')
['Price'].mean().sort_values(ascending=False)
plt.figure(figsize=(12, 6))
sns.barplot(x=avg_price_by_network.index,
y=avg_price_by_network.values, palette='Blues_r')  # Use the Blues_r
palette
plt.title('Average Price by Network Type')
plt.xlabel('Network Type')
plt.ylabel('Average Price')
plt.xticks(rotation=45)
plt.show()
```



While the new generation network type, 5G, boasts the most supported features, it also comes with the highest price

20.What is the distribution of Bluetooth versions among the mobile devices in the dataset, and how does it vary across different brands?

```
plt.figure(figsize=(12, 6))
sns.countplot(data=mobile, x='Bluetooth Version', palette='Set2')  # Use a Set2 palette
plt.title('Distribution of Bluetooth Versions')
plt.xlabel('Bluetooth Version')
plt.ylabel('Count')
plt.show()
```



Version 5.0 boasts a substantial enhancement in the number of features within the Bluetooth functionality.

Insights

Price Distribution by Brand: There is significant variation in smartphone prices among different brands. Brands like Apple tend to have higher average prices compared to others.

Relationship Between Price and Rating: There is no clear linear relationship between the price and the rating of smartphones. Price alone does not determine the rating of a smartphone.

Distribution of Ratings: The majority of smartphones have ratings between 4.0 and 4.5, indicating a generally positive reception.

Operating System Impact on Ratings: There might be variations in ratings based on the operating system, as shown in the distribution by operating system.

Internal Storage and Battery Capacity: Scatter plots indicate that there is no strong linear correlation between internal storage and battery capacity.

Brand Analysis: The dataset contains smartphones from various brands, and each brand has its own pricing strategy. Some brands dominate the market, while others may cater to specific niches.

Network Type and Price: The average price varies depending on the network type. SIM Type and Hybrid SIM Slot Impact: The type of SIM card supported (Single or Dual) and the presence of a Hybrid SIM slot may influence the price.

Limitations:

Assumption of Normality: Some statistical tests assume normality in the distribution of data. If the data does not meet this assumption, the results of certain analyses may be less reliable.

Outliers Handling: Outliers were visualized using box plots but not systematically addressed. Extreme values can skew statistical measures and impact the generalizability of findings.

Limited Statistical Tests: The analysis has used basic statistical tests, and more sophisticated methods might be needed for a deeper understanding of relationships.

Unexplored External Factors: External factors such as market trends, economic conditions, or technological advancements that might influence smartphone attributes are not considered.

Recomendations:

Outlier Management: Further investigate and manage outliers in the dataset. Outliers may represent unique cases or errors in data entry. Understanding their nature is crucial for accurate analysis.

Time-Based Analysis: If available, consider obtaining a time-series dataset to analyze trends and changes over time in the smartphone market. This can provide insights into evolving customer preferences and technological advancements.

Market Positioning: Identify the brands with the highest average prices and ratings. Consider the market positioning of these brands and assess whether there are specific features or strategies contributing to their success.

Deeper Statistical Tests: If required, perform more advanced statistical tests or hypothesis testing to explore relationships between variables in greater detail. This can provide a more nuanced understanding of the dataset.

Monitoring Industry Trends: Stay informed about broader industry trends, technological advancements, and changing consumer behaviours. This external context is essential for understanding the implications of your findings in a dynamic market.

Conclusion:

Unravelling the Tapestry of Mobile Technology Trends

In the dynamic landscape of the Mobile Technology Market.

our exploration into the dataset has provided valuable insights into the diverse facets of mobile phones.

As we conclude our analysis, several key findings and considerations emerge, shaping a comprehensive understanding of this evolving industry.

References

Data Sources: Kaggle. Mobile Trends Dataset.

Acknowledgement

This comprehensive analysis would not have been possible without the collaboration and contribution of various entities and resources. We express our gratitude to: Kaggle Community: The dataset used in this analysis was obtained from Kaggle, and we extend our appreciation to the contributors who curated and shared this valuable dataset with the community.

Open-Source Libraries: The Python programming language and open-source libraries such as NumPy, Pandas, Matplotlib, and Seaborn provided the foundational tools for data manipulation, analysis, and visualization. We thank the dedicated developers and contributors behind these libraries for their continuous efforts in making robust tools accessible to the community.

Educational Institution: This project is a part of our coursework, and we acknowledge our educational institution for providing the framework and resources for learning and applying data analysis techniques. The knowledge gained through this project contributes to our academic and professional growth.

Mentors and Instructors: Special thanks to our mentors and instructors who provided guidance, feedback, and support throughout the project. Their expertise and insights have been instrumental in navigating challenges and refining the analysis.

Upgrad: The broader data science and technology community has been a valuable source of information and assistance. We are grateful for the shared knowledge that has contributed to the success of this project.

*** THE END ***