Mobile Phone in India Predection

In [206…

```
Image(filename="India Bazar Phone.png", width=1150)
```

Out[206…

In [152…
```
# Project By: Uvesh Ahmad
# DataSet Link: https://github.com/Uvesh-Ahmad/Mobile-Phone-Price-in-india
# Portfolio: https://uvesh-ahmad.github.io/uvesh.ah/
# Linkedin : https://www.linkedin.com/in/uvesh-ahmad-a2aa6816a/
```

In [153…
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

In [154…
```
sns.set_theme(color_codes=True)
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", None)
```

In [155…
```
df = pd.read_csv("mobiles.csv")
df.head()
```

Out[155...

	Unnamed: 0	Brand	Title	Model Name	Model Number	Price	Rating	No_of_Ratings	No_of_Reviews	In The Box	Colo
0	0	APPLE	APPLE iPhone 13 (Pink, 128 GB)	iPhone 13	MLPH3HN/A	52499	4.7	259109.0	12745.0	iPhone, USB-C to Lightning Cable, Documentation	Pink
1	1	POCO	POCO C51 (Power Black, 64 GB)	C51	MZB0E6DIN	6499	4.1	78642.0	4449.0	Handset, 10W Adapter, USB Cable, Sim Eject Too...	Powe Black
2	2	OnePlus	OnePlus Nord CE 2 Lite 5G (Black Dusk, 128 GB)	Nord CE 2 Lite 5G	CPH2381/CVD	17196	4.4	110949.0	7728.0	Phone,SIM Tray Ejector, Adapter, Phone Case, U...	Black Dusk
3	3	realme	realme 11x 5G (Purple Dawn, 128 GB)	11x 5G	RMX3785	15999	4.3	2633.0	220.0	Handset, Adapter, USB Cable, Important Info Bo...	Purple Dawn

	Unnamed: 0	Brand	Title	Model Name	Model Number	Price	Rating	No_of_Ratings	No_of_Reviews	In The Box	Colo
4	4	realme	realme 11x 5G (Midnight Black, 128 GB)	11x 5G	RMX3785	14999	4.4	13807.0	988.0	Handset, Adapter, USB Cable, Important Info Bo...	Midnight Black

Data Preprocessing Part - 1

In [156…

```
# Drop identifier column
df.drop(columns=["Unnamed: 0", "Title", "Model Number"], inplace=True)
df.head()
```

Out[156...

	Brand	Model Name	Price	Rating	No_of_Ratings	No_of_Reviews	In The Box	Color	Browse Type	SIM Type	Hybrid Sim Slot	Touchs
0	APPLE	iPhone 13	52499	4.7	259109.0	12745.0	iPhone, USB-C to Lightning Cable, Documentation	Pink	Smartphones	Dual Sim	No	
1	POCO	C51	6499	4.1	78642.0	4449.0	Handset, 10W Adapter, USB Cable, Sim Eject Too...	Power Black	Smartphones	Dual Sim	No	
2	OnePlus	Nord CE 2 Lite 5G	17196	4.4	110949.0	7728.0	Phone,SIM Tray Ejector, Adapter, Phone Case, U...	Black Dusk	Smartphones	Dual Sim	No	
3	realme	11x 5G	15999	4.3	2633.0	220.0	Handset, Adapter, USB Cable, Important Info Bo...	Purple Dawn	Smartphones	Dual Sim	No	

	Brand	Model Name	Price	Rating	No_of_Ratings	No_of_Reviews	In The Box	Color	Browse Type	SIM Type	Hybrid Sim Slot	Touchs
4	realme	11x 5G	14999	4.4	13807.0	988.0	Handset, Adapter, USB Cable, Important Info Bo...	Midnight Black	Smartphones	Dual Sim	No	

In [157…
```
df.shape
```

Out[157…
```
(984, 71)
```

In [158…
```
# Check the number of unique value from all of the object datatype
df.select_dtypes(include="object").nunique()
```

Out[158…

Brand	38
Model Name	327
In The Box	240
Color	376
Browse Type	2
SIM Type	5
Hybrid Sim Slot	2
Touchscreen	2
OTG Compatible	4
Quick Charging	2
Sound Enhancements	41
Resolution	174
Resolution Type	16
Display Type	76
Other Display Features	133
Operating System	59
Processor Type	125
Processor Core	5
Operating Frequency	132
Internal Storage	29
Primary Camera Available	2
Primary Camera	91
Primary Camera Features	162
Secondary Camera Available	2
Secondary Camera	20
Secondary Camera Features	139
Flash	44
HD Recording	2
Full HD Recording	2
Video Recording	2
Video Recording Resolution	83
Digital Zoom	26
Frame Rate	28
Dual Camera Lens	3
Call Wait/Hold	1
Network Type	24
Supported Networks	32
Internet Connectivity	33
3G	2
Pre-installed Browser	14
Bluetooth Support	2
Bluetooth Version	20

```
Wi-Fi                          2
Wi-Fi Version                 71
Wi-Fi Hotspot                  2
NFC                            2
EDGE                           2
Map Support                   11
GPS Support                    2
Smartphone                     2
SIM Size                      39
Mobile Tracker                 2
Removable Battery              2
SMS                            1
Graphics PPI                  33
Sensors                      105
Browser                       20
Other Features               158
GPS Type                      52
Video Formats                 47
Battery Capacity              55
Width                        115
Height                       130
Depth                         98
Weight                       104
dtype: int64
```

In [159…

```
# Change Numerical column datatype from object to int and remove string data

df["Battery Capacity"] = df["Battery Capacity"].str.replace("[^0-9.]", "", regex=True)
df["Battery Capacity"] = pd.to_numeric(df["Battery Capacity"], errors="coerce")

df["Width"] = df["Width"].str.replace("[^0-9.]", " ", regex=True)
df["Width"] = pd.to_numeric(df["Width"], errors="coerce")

df["Height"] = df["Height"].str.replace("[^0-9.]", "", regex=True)
df["Height"] = pd.to_numeric(df["Height"], errors="coerce")

df["Depth"] = df["Depth"].str.replace("[^0-9.]", "", regex=True)
df["Depth"] = pd.to_numeric(df["Depth"], errors="coerce")

df["Weight"] = df["Weight"].str.replace("[^0-9.]", "", regex=True)
df["Weight"] = pd.to_numeric(df["Weight"], errors="coerce")
```

```
df["Graphics PPI"] = df["Graphics PPI"].str.replace("[^0-9.]", " ", regex=True)
df["Graphics PPI"] = pd.to_numeric(df["Graphics PPI"], errors="coerce")
```

In [160…

```
df.head()
```

Out[160...

	Brand	Model Name	Price	Rating	No_of_Ratings	No_of_Reviews	In The Box	Color	Browse Type	SIM Type	Hybrid Sim Slot	Touchs
0	APPLE	iPhone 13	52499	4.7	259109.0	12745.0	iPhone, USB-C to Lightning Cable, Documentation	Pink	Smartphones	Dual Sim	No	
1	POCO	C51	6499	4.1	78642.0	4449.0	Handset, 10W Adapter, USB Cable, Sim Eject Too...	Power Black	Smartphones	Dual Sim	No	
2	OnePlus	Nord CE 2 Lite 5G	17196	4.4	110949.0	7728.0	Phone,SIM Tray Ejector, Adapter, Phone Case, U...	Black Dusk	Smartphones	Dual Sim	No	
3	realme	11x 5G	15999	4.3	2633.0	220.0	Handset, Adapter, USB Cable, Important Info Bo...	Purple Dawn	Smartphones	Dual Sim	No	

	Brand	Model Name	Price	Rating	No_of_Ratings	No_of_Reviews	In The Box	Color	Browse Type	SIM Type	Hybrid Sim Slot	Touchs
4	realme	11x 5G	14999	4.4	13807.0	988.0	Handset, Adapter, USB Cable, Important Info Bo...	Midnight Black	Smartphones	Dual Sim	No	

In [161…

```
# Drop columns with unique values and columns with only 1 unique value
columns_to_drop = [
    "Model Name",
    "In The Box",
    "Color",
    "Sound Enhancements",
    "Resolution",
    "Display Type",
    "Other Display Features",
    "Processor Type",
    "Operating Frequency",
    "Primary Camera",
    "Primary Camera Features",
    "Secondary Camera Features",
    "Flash",
    "Video Recording Resolution",
    "Digital Zoom",
    "Call Wait/Hold",
    "Wi-Fi Version",
    "Sensors",
    "Other Features",
    "GPS Type",
    "Video Formats",
    "SMS",
]
```

```
df.drop(columns=columns_to_drop, inplace=True)
df.head()
```

Out[161…

	Brand	Price	Rating	No_of_Ratings	No_of_Reviews	Browse Type	SIM Type	Hybrid Sim Slot	Touchscreen	OTG Compatible	Quick Charging	Dis
0	APPLE	52499	4.7	259109.0	12745.0	Smartphones	Dual Sim	No	Yes	No	Yes	
1	POCO	6499	4.1	78642.0	4449.0	Smartphones	Dual Sim	No	Yes	Yes	Yes	
2	OnePlus	17196	4.4	110949.0	7728.0	Smartphones	Dual Sim	No	Yes	No	NaN	
3	realme	15999	4.3	2633.0	220.0	Smartphones	Dual Sim	No	Yes	Yes	NaN	
4	realme	14999	4.4	13807.0	988.0	Smartphones	Dual Sim	No	Yes	Yes	NaN	

In [162…

```
# Check the number of unique value from all of the object datatype
df.select_dtypes(include="object").nunique()
```

Out[162…

```
Brand                        38
Browse Type                   2
SIM Type                      5
Hybrid Sim Slot               2
Touchscreen                   2
OTG Compatible                4
Quick Charging                2
Resolution Type              16
Operating System             59
Processor Core                5
Internal Storage             29
Primary Camera Available      2
Secondary Camera Available    2
Secondary Camera             20
HD Recording                  2
Full HD Recording             2
Video Recording               2
Frame Rate                   28
Dual Camera Lens              3
Network Type                 24
Supported Networks           32
Internet Connectivity        33
3G                            2
Pre-installed Browser        14
Bluetooth Support             2
Bluetooth Version            20
Wi-Fi                         2
Wi-Fi Hotspot                 2
NFC                           2
EDGE                          2
Map Support                  11
GPS Support                   2
Smartphone                    2
SIM Size                     39
Mobile Tracker                2
Removable Battery             2
Browser                      20
dtype: int64
```

In [163…

```
df.shape
```

Out[163… (984, 49)

Segement Brand into Smaller Unique Value

In [164…
```
df.Brand.unique()
```

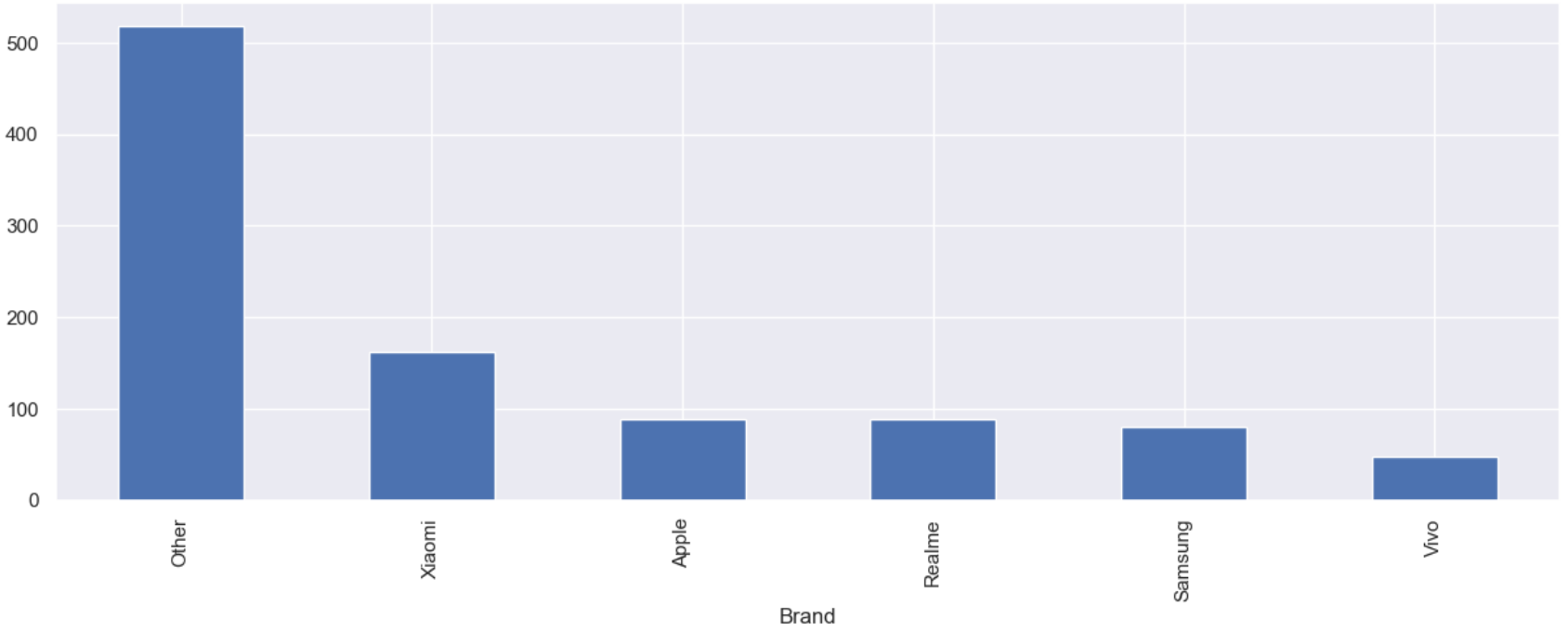
Out[164…
```
array(['APPLE', 'POCO', 'OnePlus', 'realme', 'vivo', 'MOTOROLA', 'REDMI',
       'Infinix', 'Nokia', 'SAMSUNG', 'OPPO', 'Micromax', 'MarQ', 'LAVA',
       'Google', 'itel', 'Kechaoda', 'HOTLINE', 'Tecno', 'KARBONN', 'I',
       'GFive', 'DIZO', 'Snexian', 'Good', 'Eunity', 'Energizer', 'IAIR',
       'Cellecor', 'IQOO', 'Xiaomi', 'MTR', 'Nothing', 'Mi', 'SAREGAMA',
       'Peace', 'UiSmart', 'Itel'], dtype=object)
```

In [165…
```
def segment_brand(brand):
    # List of segment
    apple_brands = ["APPLE", "iPhone"]
    samsung_brands = ["SAMSUNG", "Galaxy"]
    xiaomi_brand = ["Xiaomi", "Mi", "REDMI", "POCO"]
    oneplus_brands = ["OnlePlus"]
    realme_brands = ["realme"]
    vivo_brands = ["vivo"]
    other_brands = [
        "APPLE",
        "POCO",
        "OnePlus",
        "realme",
        "vivo",
        "MOTOROLA",
        "REDMI",
        "Infinix",
        "Nokia",
        "SAMSUNG",
        "OPPO",
        "Micromax",
        "MarQ",
        "LAVA",
        "Google",
        "itel",
        "Kechaoda",
        "HOTLINE",
```

```
        "Tecno",
        "KARBONN",
        "I",
        "GFive",
        "DIZO",
        "Snexian",
        "Good",
        "Eunity",
        "Energizer",
        "IAIR",
        "Cellecor",
        "IQOO",
        "Xiaomi",
        "MTR",
        "Nothing",
        "Mi",
        "SAREGAMA",
        "Peace",
        "UiSmart",
        "Itel",
    ]

    if brand in apple_brands:
        return "Apple"
    elif brand in samsung_brands:
        return "Samsung"
    elif brand in xiaomi_brand:
        return "Xiaomi"
    elif brand in oneplus_brands:
        return "OnePlus"
    elif brand in realme_brands:
        return "Realme"
    elif brand in vivo_brands:
        return "Vivo"
    else:
        return "Other"


# Apple the segment_brand function to the 'Brand' column
df["Brand"] = df["Brand"].apply(segment_brand)
```

In [166…
```
plt.figure(figsize=(15, 5))
df["Brand"].value_counts().plot(kind="bar")
```

Out[166…
```
<Axes: xlabel='Brand'>
```



Segment Resolution Type into Smaller Unique Value

In [167…
```
df["Resolution Type"].unique()
```

Out[167…
```
array(['Super Retina XDR Display', 'HD+', nan, 'Full HD+', 'HD',
       'Full HD', 'Liquid Retina HD Display', 'QVGA',
       'Full HD+ AMOLED Display', 'Quarter QVGA',
       'Full HD+ Super AMOLED Display', 'FWVGA', 'HQVGA',
       'Retina HD Display', 'Full HD+ E3 Super AMOLED Display', 'Quad HD',
       'Quad HD+'], dtype=object)
```

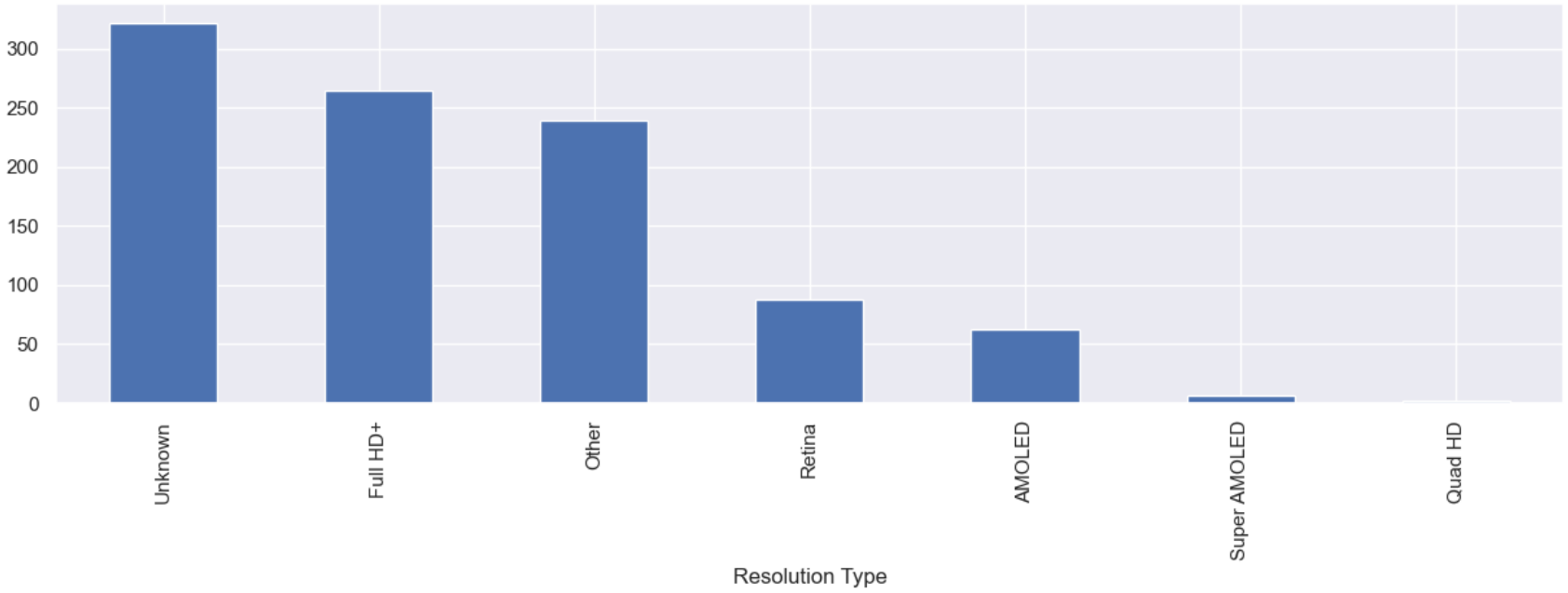
In [168…
```
def segment_resoluation_type(resolution):
    if pd.isna(resolution):
        return "Unknown"
    elif "Super AMOLED" in resolution:
```

```
        return "Super AMOLED"
    elif "AMOLED" in resolution:
        return "AMOLED"
    elif "Retina" in resolution:
        return "Retina"
    elif "Full HD+" in resolution:
        return "Full HD+"
    elif "Quad HD" in resolution:
        return "Quad HD"
    else:
        return "Other"


# Apply the segment_resolution_type function to the 'Resolution Type' column
df["Resolution Type"] = df["Resolution Type"].apply(segment_resoluation_type)
```

In [169…

```
plt.figure(figsize=(15, 4))
df["Resolution Type"].value_counts().plot(kind="bar")
```

Out[169… <Axes: xlabel='Resolution Type'>



Segment Operation System into Smaller Unique Value

In [170…
```
df["Operating System"].unique()
```

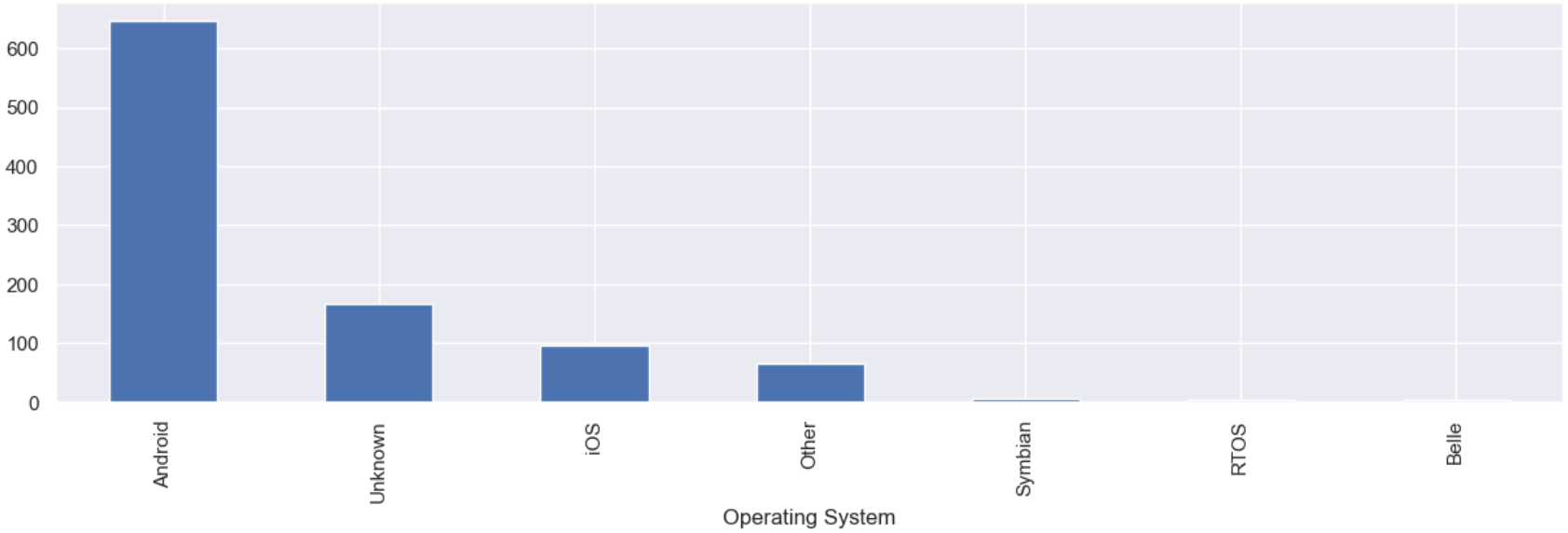
Out[170…
```
array(['iOS 15', 'Android 13', 'Android 12', 'Android Android 13',
       'Android 13 (Go Edition)', nan, 'iOS 16',
       'Android Android 13 With MIUI 14',
       'Android MIUI 14 With Android 13', 'iOS 17', 'iOS 14.2', '1',
       'Android 10', 'Android Android 13.0', 'Android 11', 'Android Q 13',
       'Android Q Android 11', 'Android Oxygen Android 13.1',
       'Android Oxygen OxygenOS based on Android 13', 'Android 12 Go',
       'Symbian', '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 Android 13 OxygenOS', 'Android Q Android 12', 'Belle',
       'Android 13 Go', 'RTOS', 'Android', 'Android Android 12',
       'Series 30+', 'Android Android13', 'Android ANdroid 13 OxygenOS',
       'Android ?MIUI 13, Android 12.0', 'Android Oxygen Android 13',
       'Android Android 12.0', '4.1', 'Android Q', 'Nucleus',
       'Android Q 13.0', 'Android Q 11', 'Android Q 11.0', 'Android Q 10',
       'Android 12.0',
       'Android Android Funtouch OS 13 Based On Android 13',
       'Android Android 11', 'Android Q 4.1', '0.2', '1.41',
       'Android Q ANDROID 12.0', '0.1', 'Series 30 Series 30 Series 30+',
       'Series 30 0', 'Android Funtouch OS 13 Based On Android 13',
       'RTOS (Mocor)'], dtype=object)
```

In [171…
```
def segment_os_type(os):
    if pd.isna(os):
        return "Unknown"
    elif "iOS" in os:
        return "iOS"
    elif "Android" in os:
        return "Android"
    elif "Symbian" in os:
        return "Symbian"
    elif "Belle" in os:
        return "Belle"
    elif "RTOS" in os:
        return "RTOS"
    else:
        return "Other"
```

```
# Apply the Segment_os_type function to the 'Operation System' column
df["Operating System"] = df["Operating System"].apply(segment_os_type)
```

In [172…
```
plt.figure(figsize=(15, 4))
df["Operating System"].value_counts().plot(kind="bar")
```

Out[172… <Axes: xlabel='Operating System'>



Segment Internal Storage into smaller Unique Value

In [173…
```
df["Internal Storage"].unique()
```

Out[173…
```
array(['128 GB', '64 GB', '256 GB', '32 MB', '32 GB', nan, '24 MB',
       '4 MB', '153 MB', '0 GB', '64 MB', '0.125 GB', '128 MB', '16 MB',
       '32 KB', '32+3 GB', '1 TB', '3 MB', '512 GB', '0 MB', '8 MB',
       '10 MB', '20 MB', '2 GB', '56 MB', '256 MB', '6 GB', 'NA KB',
       '8 GB', '31 MB'], dtype=object)
```

In [174…
```
def convert_and_segment_storage(storage):
    if pd.isna(storage):
        return "Unknown"
    elif storage == "0.125 GB":
```

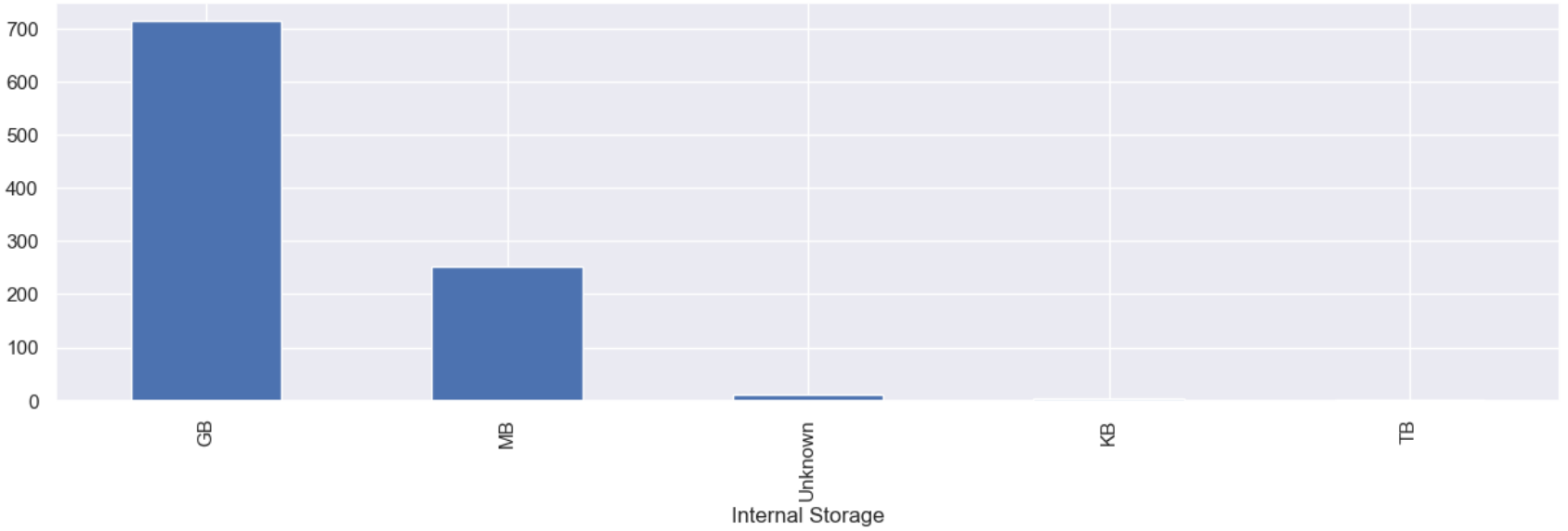
```
        return "MB"
    elif "GB" in storage:
        return "GB"
    elif "MB" in storage:
        return "MB"
    elif "KB" in storage:
        return "KB"
    elif "TB" in storage:
        return "TB"

    else:
        return "Other"


# Apply the convert_and_segment_storage function to the 'Storage' column
df["Internal Storage"] = df["Internal Storage"].apply(convert_and_segment_storage)
```

In [175…

```
plt.figure(figsize=(15, 4))
df["Internal Storage"].value_counts().plot(kind="bar")
```

Out[175…

```
<Axes: xlabel='Internal Storage'>
```



In [176…

```
# Drop Secondary Camera Column
```

```
df.drop(columns="Secondary Camera", inplace=True)
```

Segment Frame Rate into smaller Unique Value

In [177…
```
df["Frame Rate"].unique()
```

Out[177…
```
array(['24 fps, 25 fps, 30 fps, 60 fps', '60 Hz', nan, '120 fps, 30 fps',
       '30 fps', '960 fps, 240 fps, 120 fps, 60 fps, 30 fps', '90Hz fps',
       '120 fps, 60 fps, 30 fps', '60 fps, 30 fps',
       '960 fps, 480 fps, 120 fps, 60 fps, 30 fps',
       '240 fps, 120 fps, 60 fps, 30 fps', 'NA fps',
       '240 fps, 120 fps, 60 fps, 30 fps, 25 fps, 24 fps',
       'Rear Camera - 4K Video Recording at 24 fps, 30 fps or 60 fps, 1080p HD Video Recording at 30 fps or 60 fps,
720p HD Video Recording at 30 fps, Extended Dynamic Range for Video upto 60 fps, Slow-motion Video Support for 1080p
at 120 fps or 240 fps | True Depth Camera - 4K Video Recording at 24 fps, 30 fps or 60 fps, 1080p HD Video Recording
at 30 fps or 60 fps, Slow-motion Video Support for 1080p at 120 fps, Extended Dynamic Range for Video at 30 fps fp
s',
       '0 fps', '10 fps', '30 fps, 60 fps',
       '960 fps, 320 fps, 240 fps, 120 fps, 60 fps, 30 fps',
       '240 fps, 60 fps, 30 fps', '30 fps, 60 Hz', '120 Hz, 30 fps',
       '24 fps, 30 fps, 60 fps, 120 fps, 240 fps', '960 fps, 30 fps',
       '120 Hz', '60 fps', 'Yes, 1080p Video Recording fps', '144 Hz',
       '480 fps, 60 fps, 30 fps',
       '240 fps, 120 fps, 60 fps, 30 fps, 24 fps'], dtype=object)
```

In [178…
```
import re


def hz_to_fps(hz):
    return hz * 2


def segment_frame_rate(frame_rate):
    if pd.isna(frame_rate):
        return "Unknown"
    elif "fps" in frame_rate:
        # Extract numerical frame rates and convert to integer

        fps_values = [int(value) for value in re.findall(r"\d+", frame_rate)]
        if fps_values:
            max_fps = max(fps_values)
```

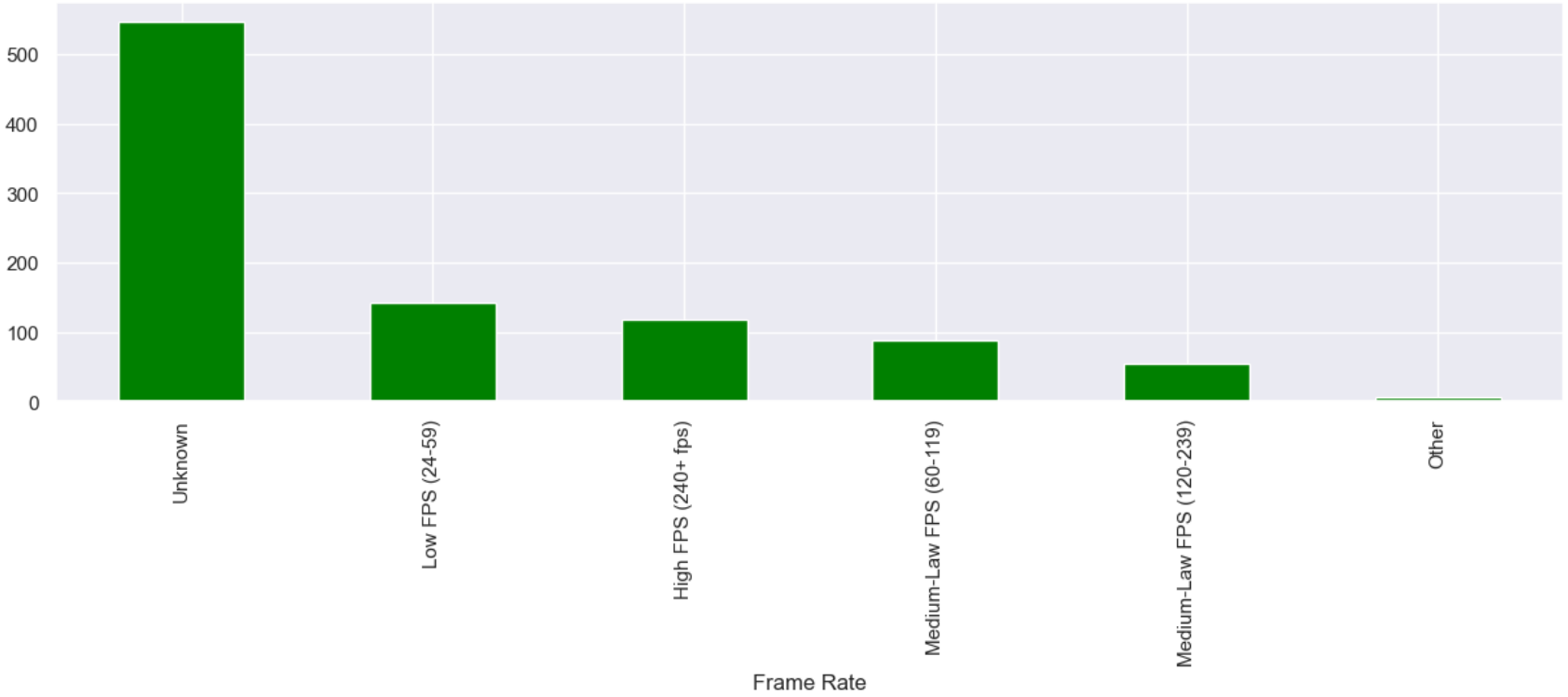
```
            if max_fps >= 240:
                return "High FPS (240+ fps)"
            elif max_fps >= 120:
                return "Medium-Law FPS (120-239)"
            elif max_fps >= 60:
                return "Medium-Law FPS (60-119)"
            else:
                return "Low FPS (24-59)"
        elif "Hz" in frame_rate:
            hz_value = float(frame_rate.replace(" Hz", ""))
            fps_values = hz_to_fps(hz_value)
            if fps_values >= 240:
                return "High FPS (240 fps)"
            elif fps_values >= 120:
                return "Medium FPS (120-239)"
            elif fps_values >= 60:
                return "Low FPS (60-119)"
            else:
                return "Low FPS (24-59)"
        else:
            return "Other"


# Apply the segment_fram_rate function to the 'Frame Rate' column

df["Frame Rate"] = df["Frame Rate"].apply(segment_frame_rate)
```

In [179…

```
plt.figure(figsize=(15, 4))  # Adjust the figure size
# Count the values and create a bar plot
df["Frame Rate"].value_counts().plot(kind="bar", color="green")
```

Out[179…

```
<Axes: xlabel='Frame Rate'>
```



Segment Network Type into Smaller Unique Value

In [180…
```
df["Network Type"].unique()
```

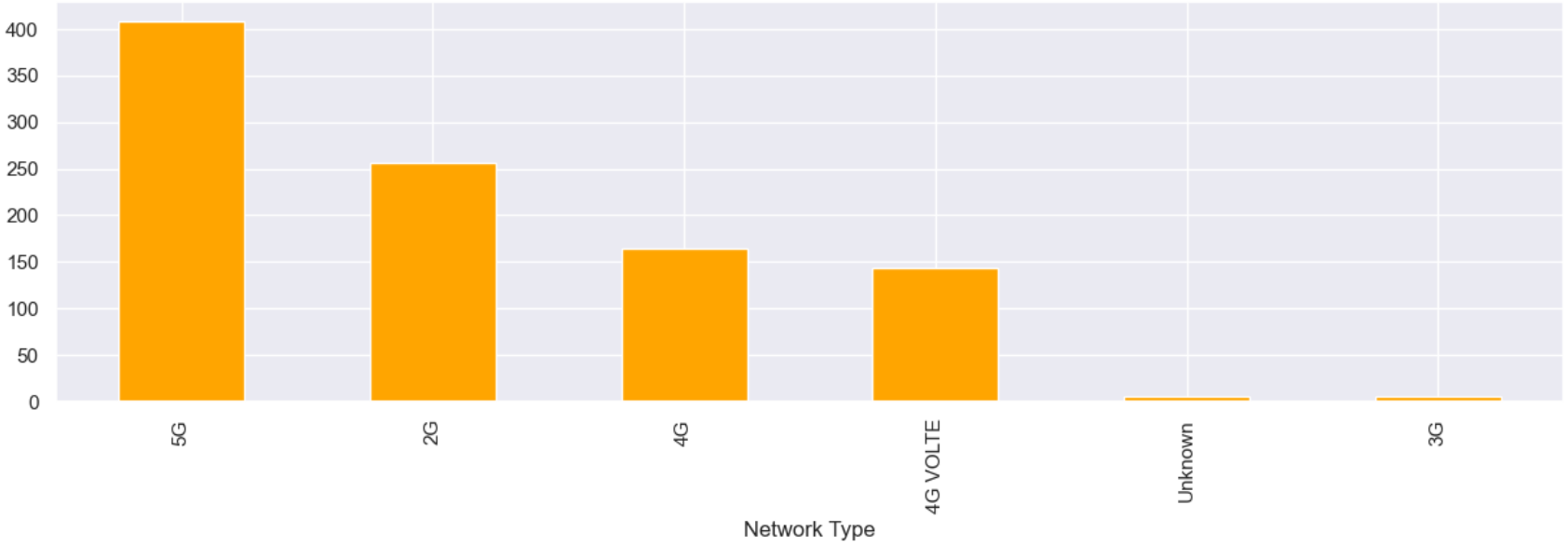
Out[180…
```
array(['5G, 4G, 3G, 2G', '2G, 3G, 4G, 4G VOLTE', '5G',
       '5G, 4G VOLTE, 4G, 3G, 2G', '4G VOLTE, 4G, 3G, 2G', '5G, 4G, 3G',
       '2G, 3G, 4G, 5G', '4G, 3G, 2G', '2G, 3G, 4G',
       '2G, 3G, 4G, 4G VOLTE, 5G', '4G VOLTE', '2G', '5G, 4G', '4G', nan,
       '4G VOLTE, 4G', '5G, 4G VOLTE', '3G', '5G, 4G VOLTE, 4G', '4G, 3G',
       '2G, 3G, 4G VOLTE, 5G', '2G, 3G', '4G VOLTE, 4G, 3G',
       '4G VOLTE, 3G, 2G', '4G, 4G VOLTE, 3G, 2G'], dtype=object)
```

In [181…
```
def segment_network_type(network_type):
    if pd.isna(network_type):
        return "Unknown"
    elif "5G" in network_type:
        return "5G"
    elif "4G VOLTE" in network_type:
```

```
        return "4G VOLTE"
    elif "4G" in network_type:
        return "4G"
    elif "3G" in network_type:
        return "3G"
    elif "2G" in network_type:
        return "2G"
    else:
        return "Mixed"


# Apply the segment_netwroek_type function to the 'Network Type' colum
df["Network Type"] = df["Network Type"].apply(segment_network_type)
```

In [182…
```
plt.figure(figsize=(15, 4))
df["Network Type"].value_counts().plot(kind="bar", color="orange")
```

Out[182… <Axes: xlabel='Network Type'>



Segment Supported Networks into Smaller Unique Value

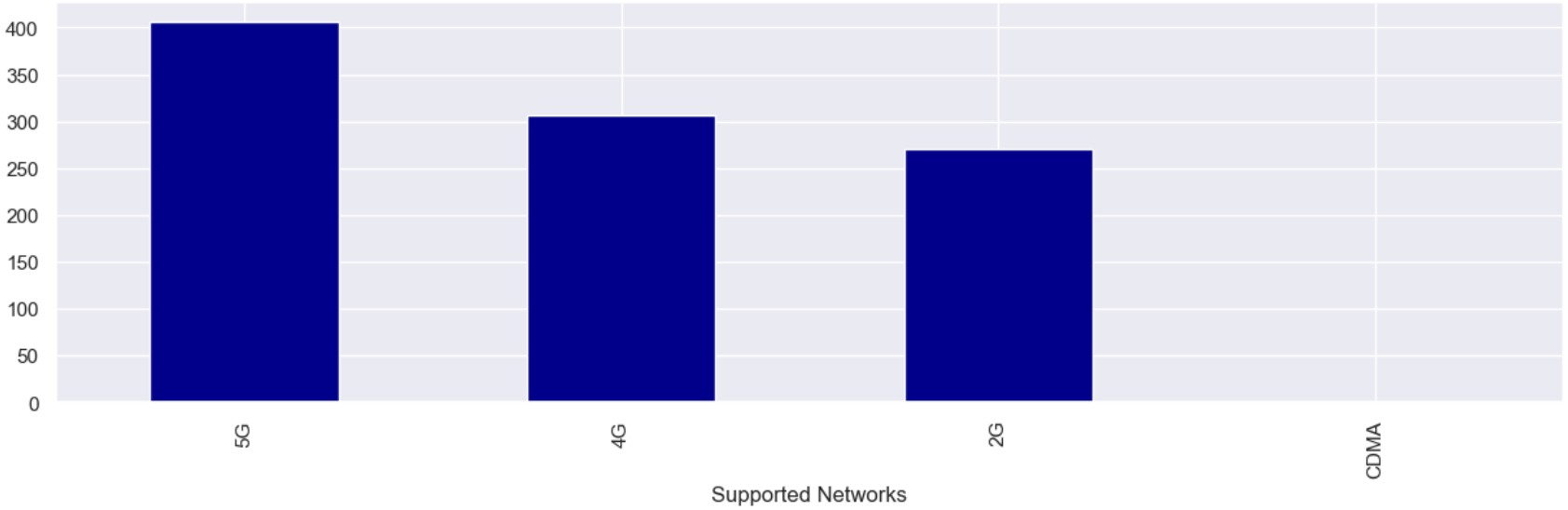
In [183…
```
df["Supported Networks"].unique()
```

Out[183…
```
array(['5G, 4G VoLTE, 4G LTE, UMTS, GSM', '4G LTE, 4G VoLTE, GSM, WCDMA',
       '5G', '5G, 4G VoLTE, 4G LTE, WCDMA, GSM',
       '4G VoLTE, 4G LTE, WCDMA, GSM', '5G, 4G LTE, WCDMA',
       '4G LTE, 5G, GSM, UMTS', '4G LTE, WCDMA, GSM',
       '4G LTE, GSM, WCDMA', '4G LTE, 5G, GSM, WCDMA',
       '4G LTE, 4G VoLTE, 5G, GSM, WCDMA', '4G VoLTE',
       '4G LTE, UMTS, GSM', '5G, 4G LTE, WCDMA, GSM', 'GSM',
       '4G VoLTE, 4G LTE, UMTS, GSM', 'GSM, GSM', '5G, 4G LTE, UMTS, GSM',
       '4G LTE', '4G VoLTE, 4G LTE', '5G, 4G VoLTE',
       '4G VoLTE, 4G LTE, GSM', 'GSM, WCDMA, 4G LTE', '4G LTE, 4G VoLTE',
       'GSM, CDMA, 4G LTE', '4G LTE, 4G VoLTE, 5G',
       '5G, 4G VoLTE, 4G LTE', 'CDMA', '4G LTE, CDMA, GSM', '5G, 4G LTE',
       '4G LTE, GSM, 4G VoLTE, WCDMA', '4G VoLTE, WCDMA, GSM'],
      dtype=object)
```

In [184…
```python
def segment_support_networks(networks2):
    if pd.isna(networks2):
        return "Unknown"
    elif "5G" in networks2:
        return "5G"
    elif "4G VoLTE" in networks2 or "4G LTE" in networks2:
        return "4G"
    elif "UMTS" in networks2 or "WCDMA" in networks2:
        return "3G"
    elif "GSM" in networks2:
        return "2G"
    elif "CDMA" in networks2:
        return "CDMA"
    else:
        return "Mixed"


# Apply the segment_support_networks functions to the 'Supported Networks' column
df["Supported Networks"] = df["Supported Networks"].apply(segment_support_networks)
```

In [185…
```python
plt.figure(figsize=(15, 4))
df["Supported Networks"].value_counts().plot(kind="bar", color="darkblue")
```

Out[185…
```
<Axes: xlabel='Supported Networks'>
```



Segment Internet Connectivity into Smaller Unique Value

In [186…
```
df["Internet Connectivity"].unique()
```

Out[186…
```
array(['5G, 4G, 3G, Wi-Fi, EDGE', '4G, 3G, Wi-Fi, EDGE, GPRS', nan,
       '5G, 4G, 3G, EDGE, GPRS, Wi-Fi', '5G, 4G, 3G, Wi-Fi',
       '4G, 3G, EDGE, GPRS, Wi-Fi', '4G, 3G, Wi-Fi',
       '5G, 4G, 3G, Wi-Fi, EDGE, GPRS', '2G', '4G, 3G, Wi-Fi, GPRS', '0',
       '4G. 3G, Wi-Fi', '4G, 3G, EDGE, Wi-Fi',
       '5G, 4G, 3G, GPRS, EDGE, Wi-Fi', '5G, 4G, 3G, EDGE, Wi-Fi',
       '5G, 4G, 3G, 2G', 'GPRS, WAP', '4G, 3G, GPRS, EDGE, Wi-Fi', 'GSM',
       '5G, 3G, 4G, EDGE, GPRS, Wi-Fi',
       'Google Play Store, Gmail, Youtube, Google, Google Assistant, Maps, Files, Facebook',
       '4G, 3G, LTE', '4G, 3G, WiFi', 'Wap', 'Yes', '4G', 'GPRS',
       '5G, 4G, 3G, EDGE, Wi-Fi, GPRS', '5G,4GLTE,3G,2G',
       '5G, 4G, 3G, Wi-Fi, GPRS, EDGE', '4G VOLTE + WIFI', '5G,4G,3G,2G',
       '5G, 4G, 3G, Wi-Fi, GPRS', '4G, 3G, Wi-Fi, EDGE'], dtype=object)
```

In [187…
```
def segment_internet_connectivity(connectivity):
    if pd.isna(connectivity):
        return "Unknown"
    elif "5G" in connectivity:
        return "5G"
    elif "4G" in connectivity:
```

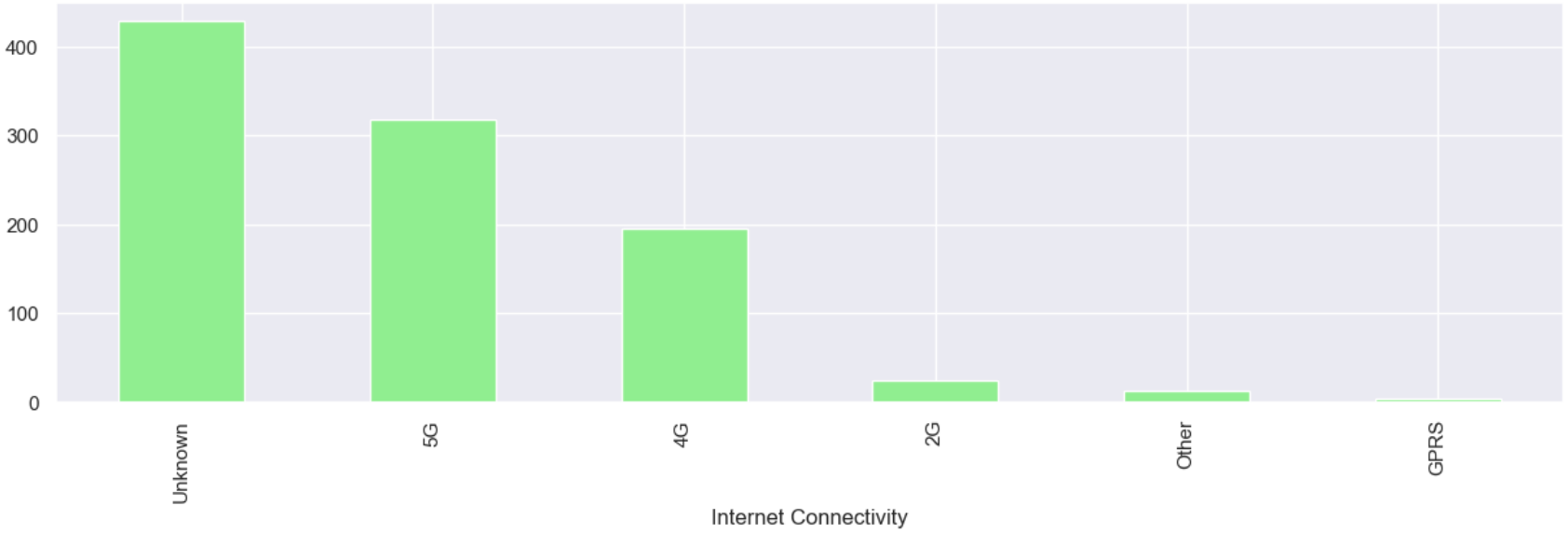
```
        return "4G"
    elif "3G" in connectivity:
        return "3G"
    elif "2G" in connectivity:
        return "2G"
    elif "Wi-Fi" in connectivity:
        return "Wi-Fi"
    elif "EDGE" in connectivity:
        return "EDGE"
    elif "GPRS" in connectivity:
        return "GPRS"
    else:
        return "Other"


# Apply the segment_internet_connectivity function to the 'Internet Connectivity' column
df["Internet Connectivity"] = df["Internet Connectivity"].apply(
    segment_internet_connectivity
)
```

In [188…

```
plt.figure(figsize=(15, 4))
df["Internet Connectivity"].value_counts().plot(kind="bar", color="LightGreen")
```

Out[188…

```
<Axes: xlabel='Internet Connectivity'>
```



Segment Pre-Installed Browser into smaller unique value

In [189…
```
df["Pre-installed Browser"].unique()
```

Out[189…
```
array(['Safari', nan, 'Yes', 'Google Chrome, Samsung S Browser 14.0',
       'Google Chrome', 'Google Chrome, Samsung S-Browser 13.2', 'WAP',
       'Google Chrome | Samsung Internet', 'CHROME', 'No',
       'Google Chrome, Samsung S-Browser 11.2',
       'Google Chrome | Samsung S-Browser 14.0',
       'Google Chrome, Samsung Browser 15.0', 'Google Chrome, Internet',
       'Google Chrome, Samsung S-Browser 9.0'], dtype=object)
```

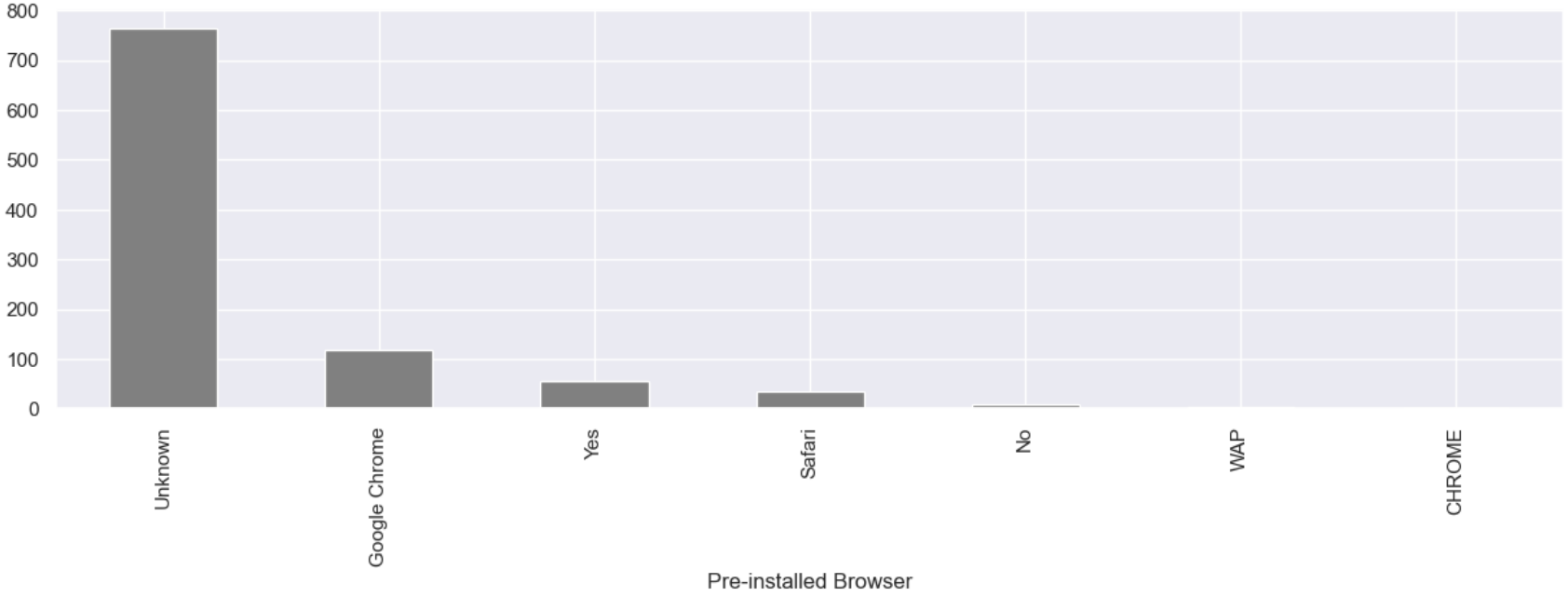
In [190…
```
def segment_preinstalled_browser(browser):
    if pd.isna(browser):
        return "Unknown"
    elif "Safari" in browser:
        return "Safari"
    elif "Google Chrome" in browser:
        return "Google Chrome"
    elif "Samsung S-Browser" in browser:
        return "Samsung S-Browser"
    elif "Internet" in browser:
        return "Internet"
```

```
    elif "WAP" in browser:
        return "WAP"
    elif "CHROME" in browser:
        return "CHROME"
    elif "Yes" in browser:
        return "Yes"
    elif "No" in browser:
        return "No"
    else:
        return "Other"


# Apply the segment_preinstalled_browser function to the 'Pre-Installed Browser' column
df["Pre-installed Browser"] = df["Pre-installed Browser"].apply(
    segment_preinstalled_browser
)
```

In [191…

```
plt.figure(figsize=(15, 4))
df["Pre-installed Browser"].value_counts().plot(kind="bar", color="Gray")
```

Out[191… <Axes: xlabel='Pre-installed Browser'>



Segment Bluetooth Version into Smaller Unique Value

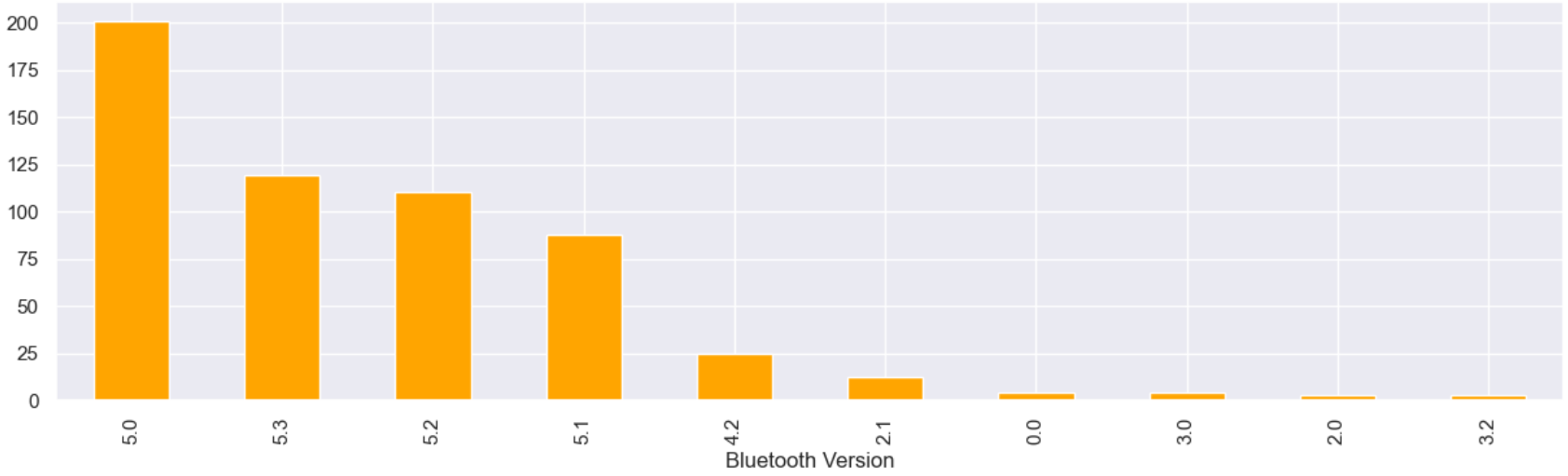
In [192…
```
df["Bluetooth Version"].unique()
```

Out[192…
```
array(['v5.0', nan, 'v5.2', 'v5.1', 'v5.3', 'v4.2', '0', '5.0', '2',
       '4.2', 'v3.0', 'V5.1', 'v2.1', '3.2', 'v3.2', '2.1', '5.1', '5.2',
       '5.3', 'V5.0', '3'], dtype=object)
```

In [193…
```
df["Bluetooth Version"] = df["Bluetooth Version"].str.replace("[^0-9.]", "", regex=True)
df["Bluetooth Version"] = pd.to_numeric(df["Bluetooth Version"], errors="coerce")
```

In [194…
```
plt.figure(figsize=(15, 4))
df["Bluetooth Version"].value_counts().plot(kind="bar", color="Orange")
```

Out[194…
```
<Axes: xlabel='Bluetooth Version'>
```



Segment SIM Size into Smaller Unique Value

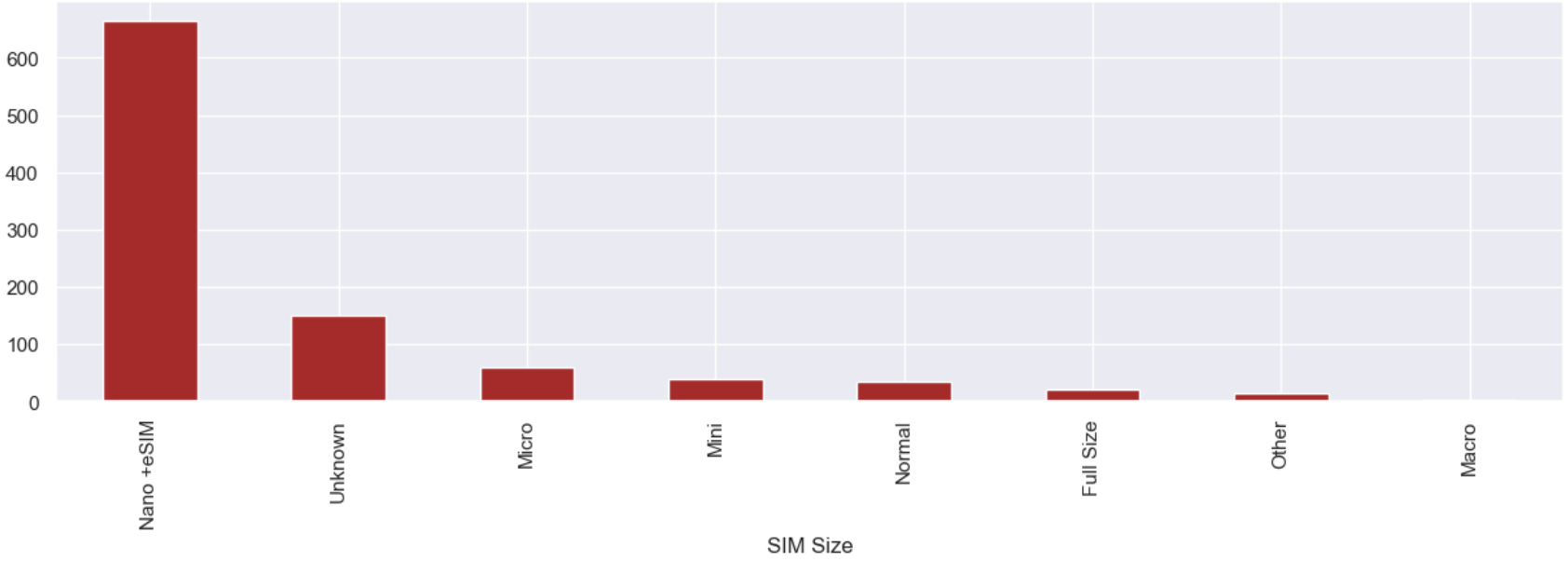
In [195…
```
df["SIM Size"].unique()
```

Out[195…
```
array(['Nano + eSIM', 'Nano Sim', nan, 'Mini Sim', 'Nano Sim + eSIM',
       'Nano SIM and eSIM', 'Full Size SIM', 'Standard+ Micro', 'Micro',
       'Nano', 'Mini SIM', 'Nano-SIM', 'Normal', 'Full Size Sim',
       'Micro SIM', 'Neno-SIM card', 'NORMAL', 'Micro SIM + Micro SIM',
       'Normal SIM', 'Macro Sim', 'Nano + Nano', 'Standard', 'MICRO',
       'Nano SIM', 'Regular Sim', 'Mini SIM + Mini SIM', 'Full Size',
       'Micro Sim', 'MINI', 'Dual Nano', 'Nano Sim & E-Sim', 'nano', 'F',
       'Micro+Micro', '2 x Mini SIM + 1 x Micro SIM', 'STANDARD',
       'mini SIM + mini SIM', 'Full Sim Size', 'Nano SIM,Standard SIM',
       'Standard SIM,Nano SIM'], dtype=object)
```

In [196…
```
def segment_sim_size(size):
    if pd.isna(size):
        return "Unknown"
    size = size.lower()
    if "nano" in size or "e-sim" in size:
        return "Nano +eSIM"
    elif "micro" in size:
        return "Micro"
    elif "mini" in size or "mini sim" in size:
        return "Mini"
    elif "full" in size or "standard" in size:
        return "Full Size"
    elif "normal" in size or "regular" in size:
        return "Normal"
    elif "macro" in size:
        return "Macro"
    else:
        return "Other"


# Apply the sagment_sim functions to the 'SIM_SIZE' Column
df["SIM Size"] = df["SIM Size"].apply(segment_sim_size)
```

In [197…
```
plt.figure(figsize=(15, 4))
df["SIM Size"].value_counts().plot(kind="bar", color="Brown")
```

Out[197…
```
<Axes: xlabel='SIM Size'>
```



Segment Browser into Smaller Unique Value

In [198…
```
df["Browser"].unique()
```

Out[198…
```
array(['Safari', 'Google Chrome', nan,
       'Google Chrome, Samsung S Browser 14.0',
       'Google Chrome | Samsung S-Browser 18.0', 'NO',
       'Google Chrome, Samsung S-Browser 13.2', 'Supports HTML5', 'WAP',
       'Google Chrome, Samsung Internet',
       'Google Chrome, Samsung S-Browser 19.0', 'Vivo Browser', 'Google',
       'Google Chrome, Samsung S-Browser 11.2',
       'Google Chrome, Samsung S-Browser 16.0',
       'Vivo Browser, Google Chrome',
       'Google Chrome | Samsung S-Browser 14.0', 'Feature Phone',
       'Google Chrome | Samsung S-Browser 16.0', 'Internet',
       'Google Chrome, Samsung S-Browser 9.0'], dtype=object)
```

In [199…
```
def segment_browser(browser):
    if pd.isna(browser):
        return "Unknown"
    browser = browser.lower()
```

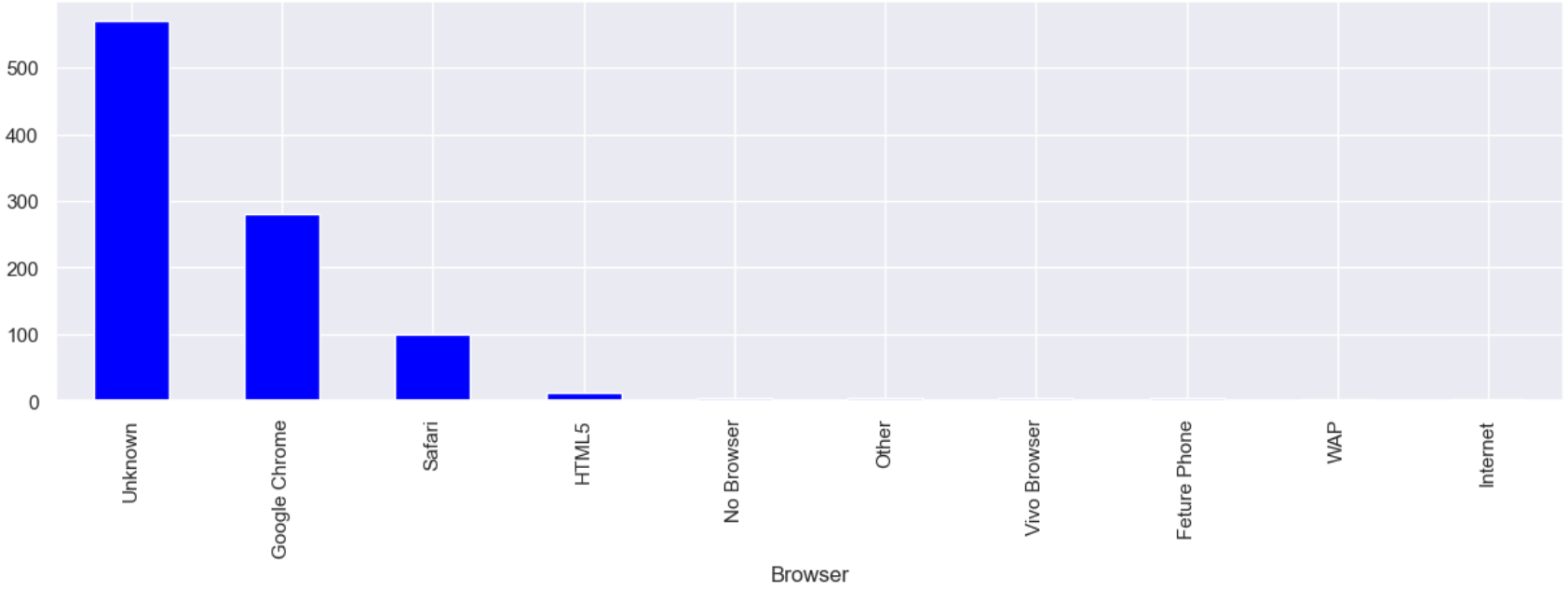
```
    if "chrome" in browser:
        return "Google Chrome"
    elif "safari" in browser:
        return "Safari"
    elif "samsung" in browser:
        return "Samsung Browser"
    elif "vivo" in browser:
        return "Vivo Browser"
    elif "html5" in browser:
        return "HTML5"
    elif "wap" in browser:
        return "WAP"
    elif "feature phone" in browser:
        return "Feture Phone"
    elif "internet" in browser:
        return "Internet"
    elif "no" in browser:
        return "No Browser"
    else:
        return "Other"


# Applying the segment_browser function to the "Browser" column
df["Browser"] = df["Browser"].apply(segment_browser)
```

In [200…

```
plt.figure(figsize=(15, 4))
df["Browser"].value_counts().plot(kind="bar", color="Blue")
```

Out[200…

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



In [201…

```
# Check the number of unique value all of the object datatype
df.select_dtypes(include="object").nunique()
```

Out[201…

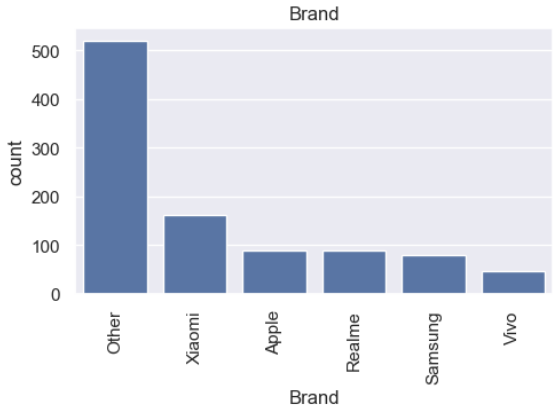
```
Brand                          6
Browse Type                    2
SIM Type                       5
Hybrid Sim Slot                2
Touchscreen                    2
OTG Compatible                 4
Quick Charging                 2
Resolution Type                7
Operating System               7
Processor Core                 5
Internal Storage               5
Primary Camera Available       2
Secondary Camera Available     2
HD Recording                   2
Full HD Recording              2
Video Recording                2
Frame Rate                     6
Dual Camera Lens               3
Network Type                   6
Supported Networks             4
Internet Connectivity          6
3G                             2
Pre-installed Browser          7
Bluetooth Support              2
Wi-Fi                          2
Wi-Fi Hotspot                  2
NFC                            2
EDGE                           2
Map Support                   11
GPS Support                    2
Smartphone                     2
SIM Size                       8
Mobile Tracker                 2
Removable Battery              2
Browser                       10
dtype: int64
```

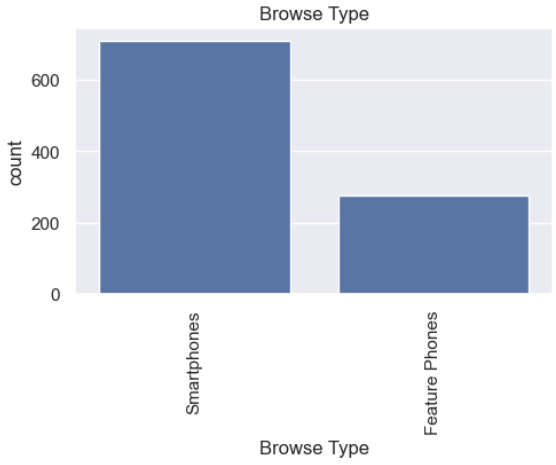
Exploratory Data Analysis

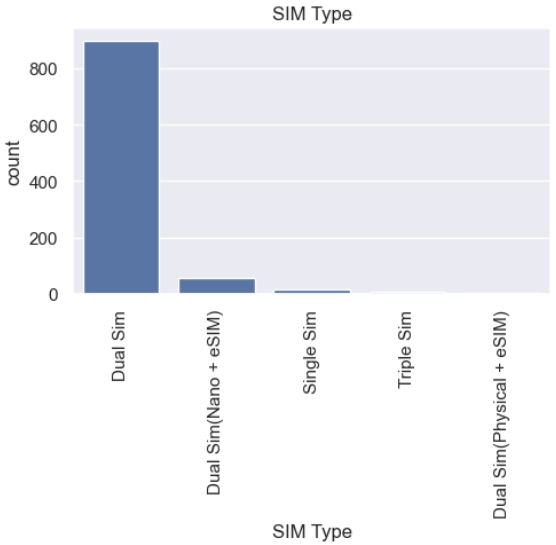
In [202…

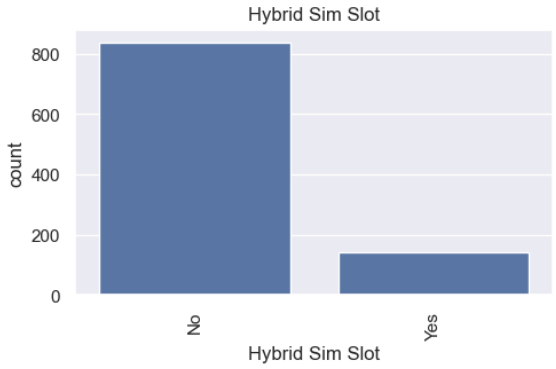
```
import matplotlib.pyplot as plt
import seaborn as sns
```

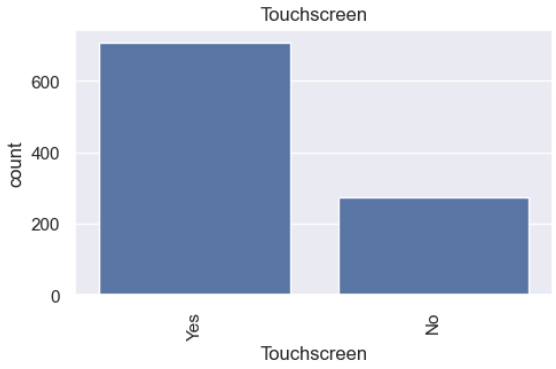
```
# Get the names of all columns with data 'object' (categorical columns) excluding 'Map Support'
cat_vars = df.select_dtypes(include="object").columns.tolist()
if "Map Support" in cat_vars:
    cat_vars.remove(
        "Map Support"
    )  # Remove 'Map Support' from the list of categorical columns

# Create a figure with subplots
num_cols = len(cat_vars)
num_rows = (num_cols + 2) // 3
fig, axs = plt.subplots(nrows=num_rows, ncols=3, figsize=(15, 5 * num_rows))
axs = axs.flatten()

# Create a countplot for the top 6 values of each categorical variable using Seaborn
for i, var in enumerate(cat_vars):
    top_values = df[var].value_counts().head(6).index  # Get top 6 values
    filtered_df = df[df[var].isin(top_values)]
    sns.countplot(x=var, data=filtered_df, ax=axs[i], order=top_values)
    axs[i].set_title(var)
    axs[i].tick_params(
        axis="x", rotation=90
    )  # Corrected parameter name from 'xaxis' to 'axis'

# Remove any extra empty subplots if needed
if num_cols < len(axs):
    for i in range(num_cols, len(axs)):
        fig.delaxes(axs[i])

# Adjust spacing between subplots
fig.tight_layout()

# Show plot
plt.show()
```

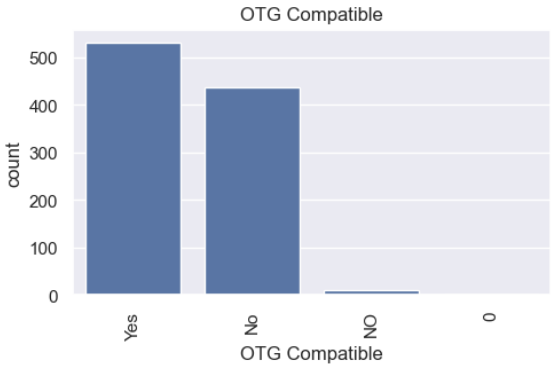

Brand
count
0
100
200
300
400
500
Other
Xiaomi
Apple
Realme
Samsung
Vivo
Brand

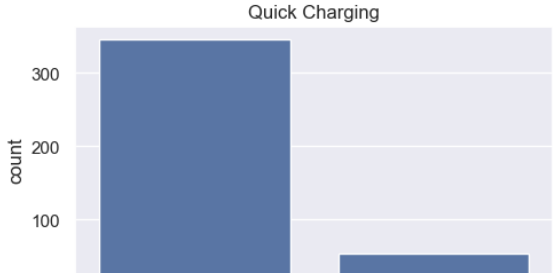

Browse Type
count
0
200
400
600
Smartphones
Feature Phones
Browse Type

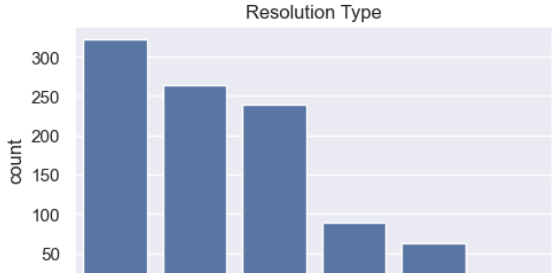

SIM Type
count
0
200
400
600
800
Dual Sim
Dual Sim(Nano + eSIM)
Single Sim
Triple Sim
Dual Sim(Physical + eSIM)
SIM Type

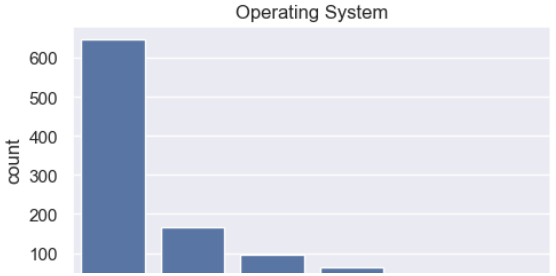

Hybrid Sim Slot
count
0
200
400
600
800
No
Yes
Hybrid Sim Slot


Touchscreen
count
0
200
400
600
Yes
No
Touchscreen

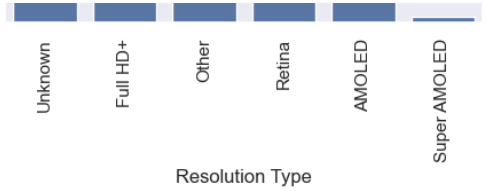

OTG Compatible
count
0
100
200
300
400
500
Yes
No
NO
0
OTG Compatible

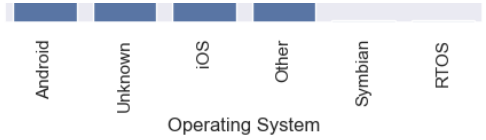

Quick Charging
count
100
200
300

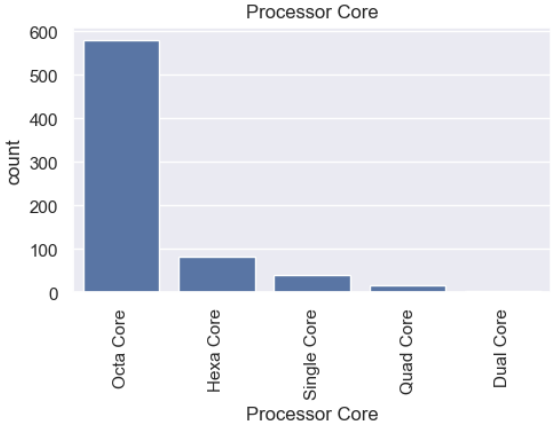

Resolution Type
count
50
100
150
200
250
300

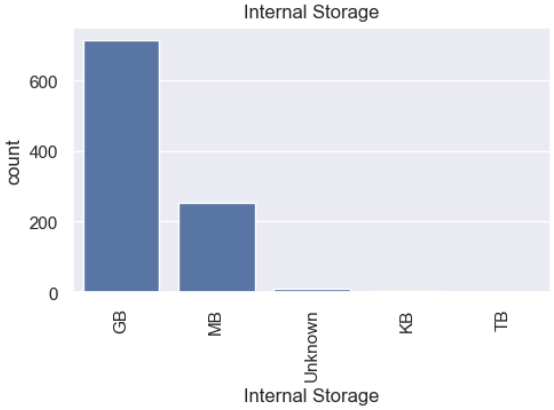

Operating System
count
100
200
300
400
500
600

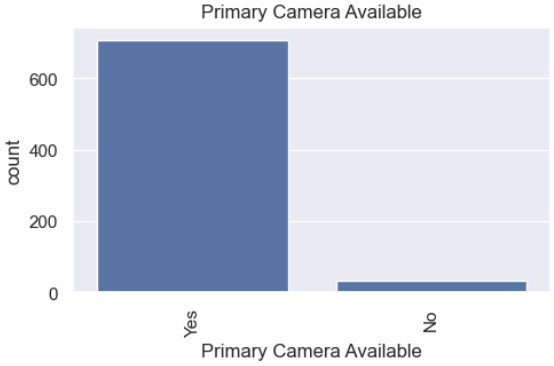

0
Yes
No
Quick Charging


0
Unknown
Full HD+
Other
Retina
AMOLED
Super AMOLED
Resolution Type

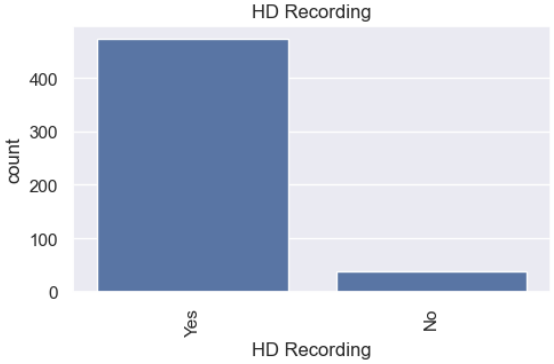

0
Android
Unknown
iOS
Other
Symbian
RTOS
Operating System

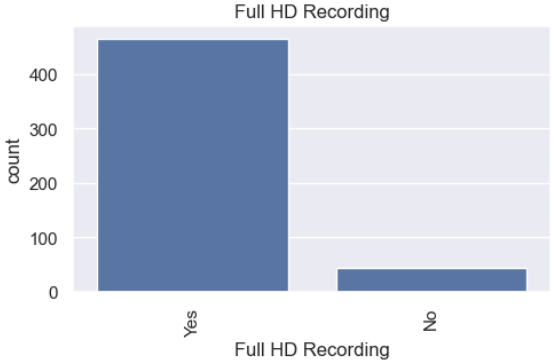

Processor Core
600
500
400
300
200
100
0
count
Octa Core
Hexa Core
Single Core
Quad Core
Dual Core
Processor Core

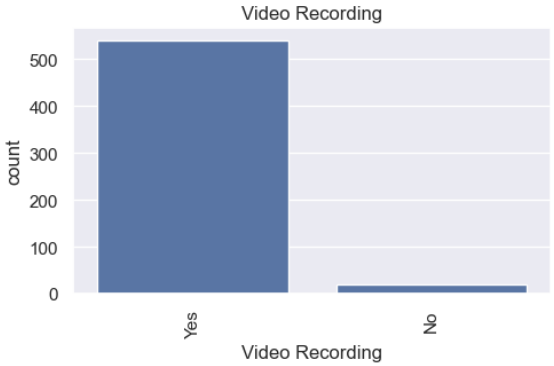

Internal Storage
600
400
200
0
count
GB
MB
Unknown
KB
TB
Internal Storage

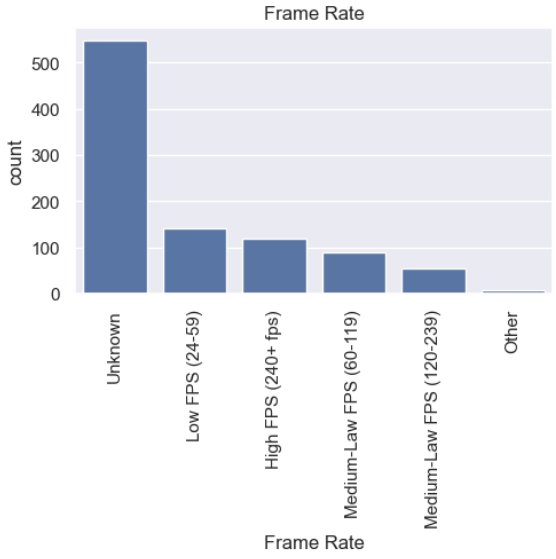

Primary Camera Available
600
400
200
0
count
Yes
No
Primary Camera Available

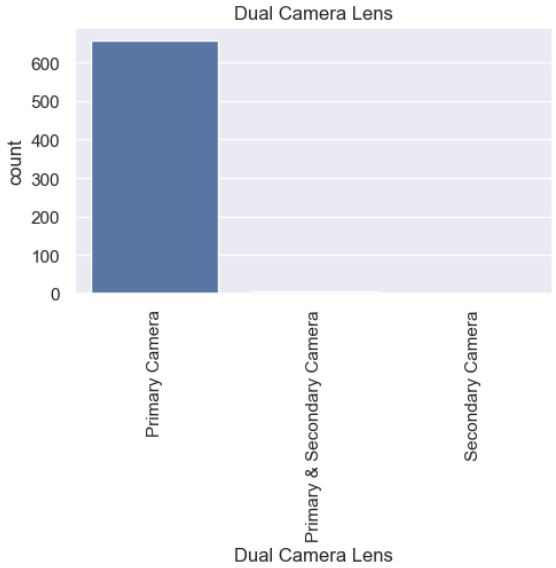

Secondary Camera Available
600
500
400
300
200
100
0
count
Yes
No
Secondary Camera Available

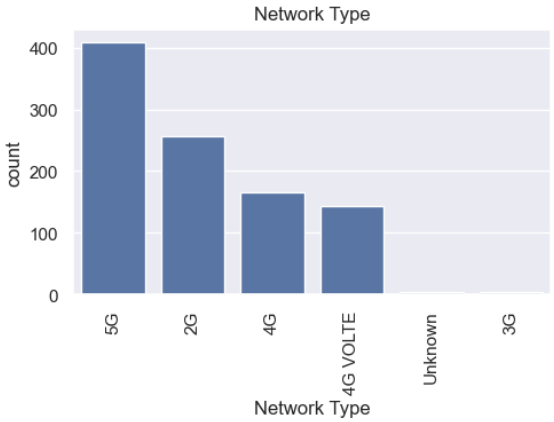

HD Recording
400
300
200
100
0
count
Yes
No
HD Recording

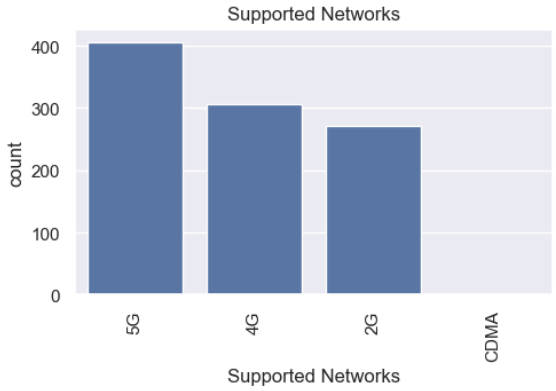

Full HD Recording
400
300
200
100
0
count
Yes
No
Full HD Recording

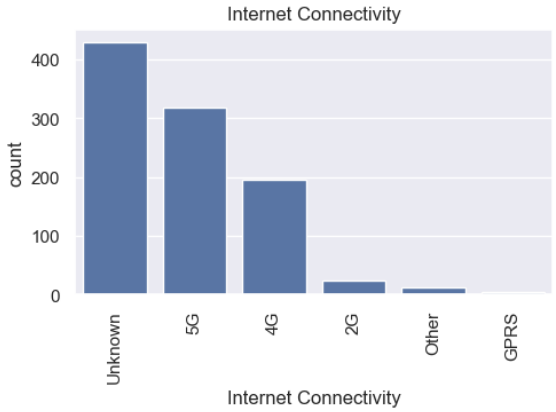

Video Recording
count
0
100
200
300
400
500
Yes
No
Video Recording

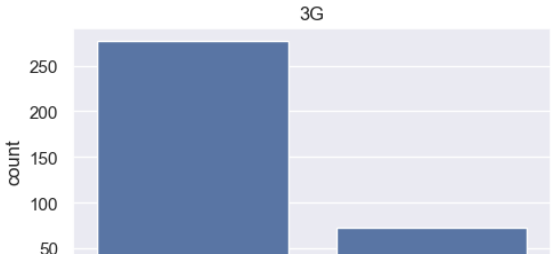

Frame Rate
count
0
100
200
300
400
500
Unknown
Low FPS (24-59)
High FPS (240+ fps)
Medium-Law FPS (60-119)
Medium-Law FPS (120-239)
Other
Frame Rate


Dual Camera Lens
count
0
100
200
300
400
500
600
Primary Camera
Primary & Secondary Camera
Secondary Camera
Dual Camera Lens

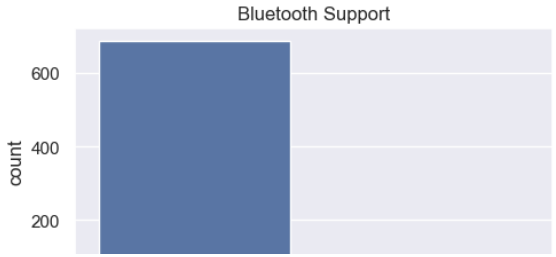

Network Type
count
0
100
200
300
400
5G
2G
4G
4G VOLTE
Unknown
3G
Network Type

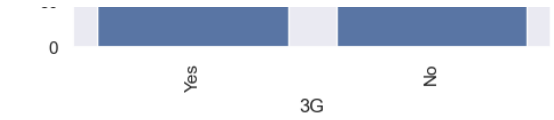

Supported Networks
count
0
100
200
300
400
5G
4G
2G
CDMA
Supported Networks

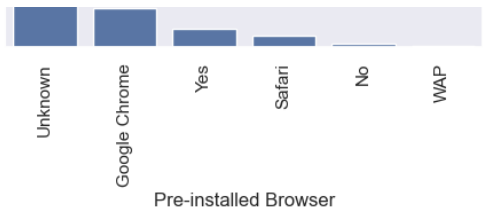

Internet Connectivity
count
0
100
200
300
400
Unknown
5G
4G
2G
Other
GPRS
Internet Connectivity

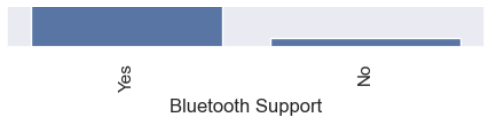

3G
count
50
100
150
200
250

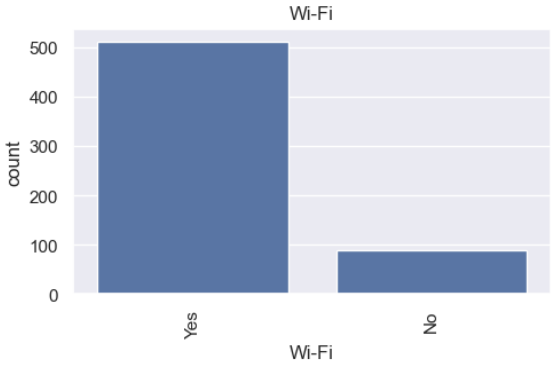

Pre-installed Browser
count
200
400
600
800

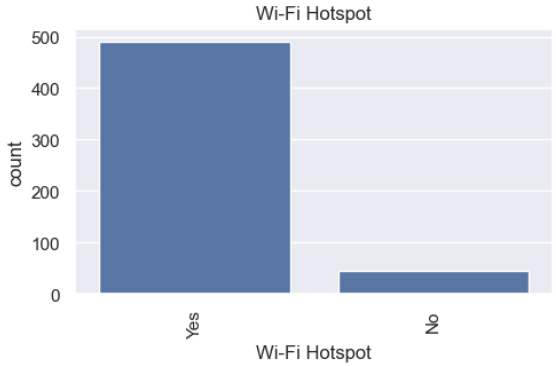

Bluetooth Support
count
200
400
600

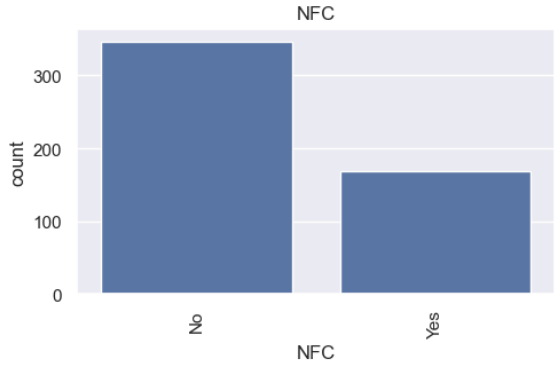

0
Yes
No
3G

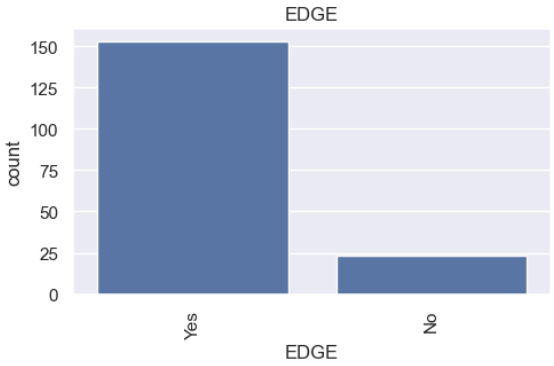

0
Unknown
Google Chrome
Yes
Safari
No
WAP
Pre-installed Browser

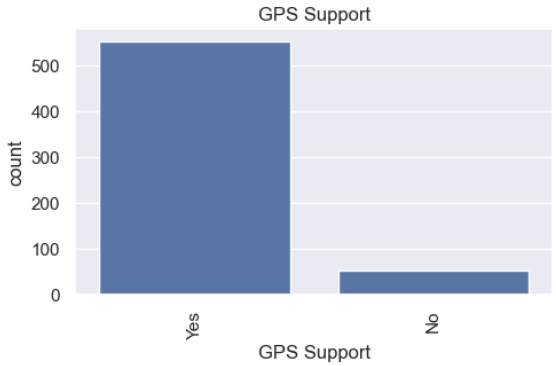

0
Yes
No
Bluetooth Support

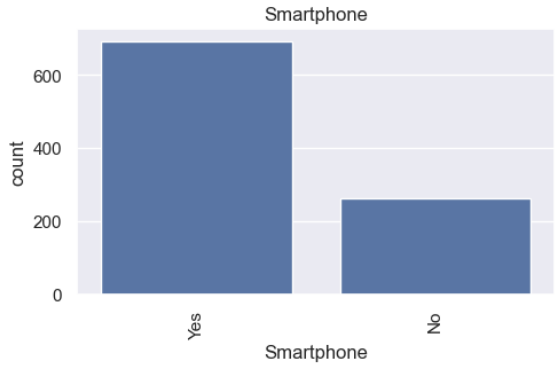

Wi-Fi
count
500
400
300
200
100
0
Yes
No
Wi-Fi

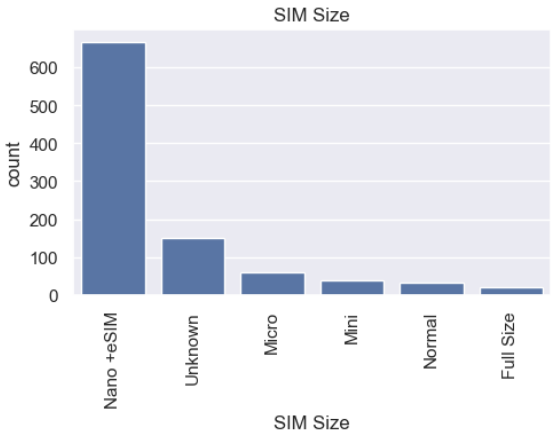

Wi-Fi Hotspot
count
500
400
300
200
100
0
Yes
No
Wi-Fi Hotspot

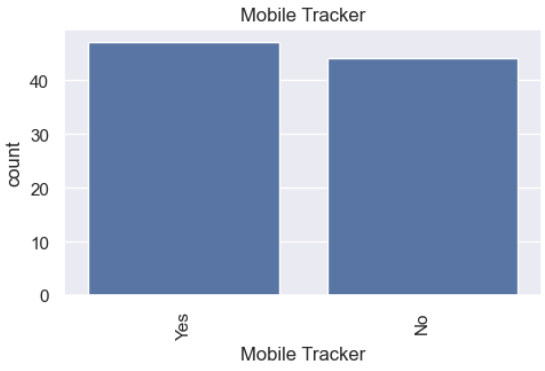

NFC
count
300
200
100
0
No
Yes
NFC

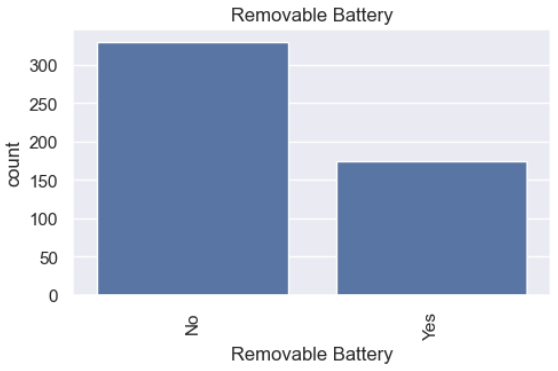

EDGE
count
150
125
100
75
50
25
0
Yes
No
EDGE

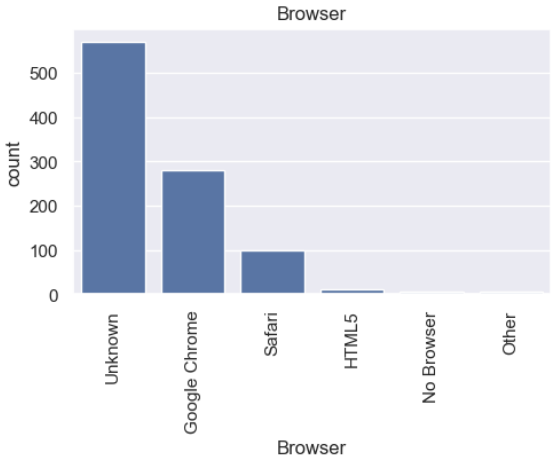

GPS Support
count
500
400
300
200
100
0
Yes
No
GPS Support


Smartphone
count
600
400
200
0
Yes
No
Smartphone





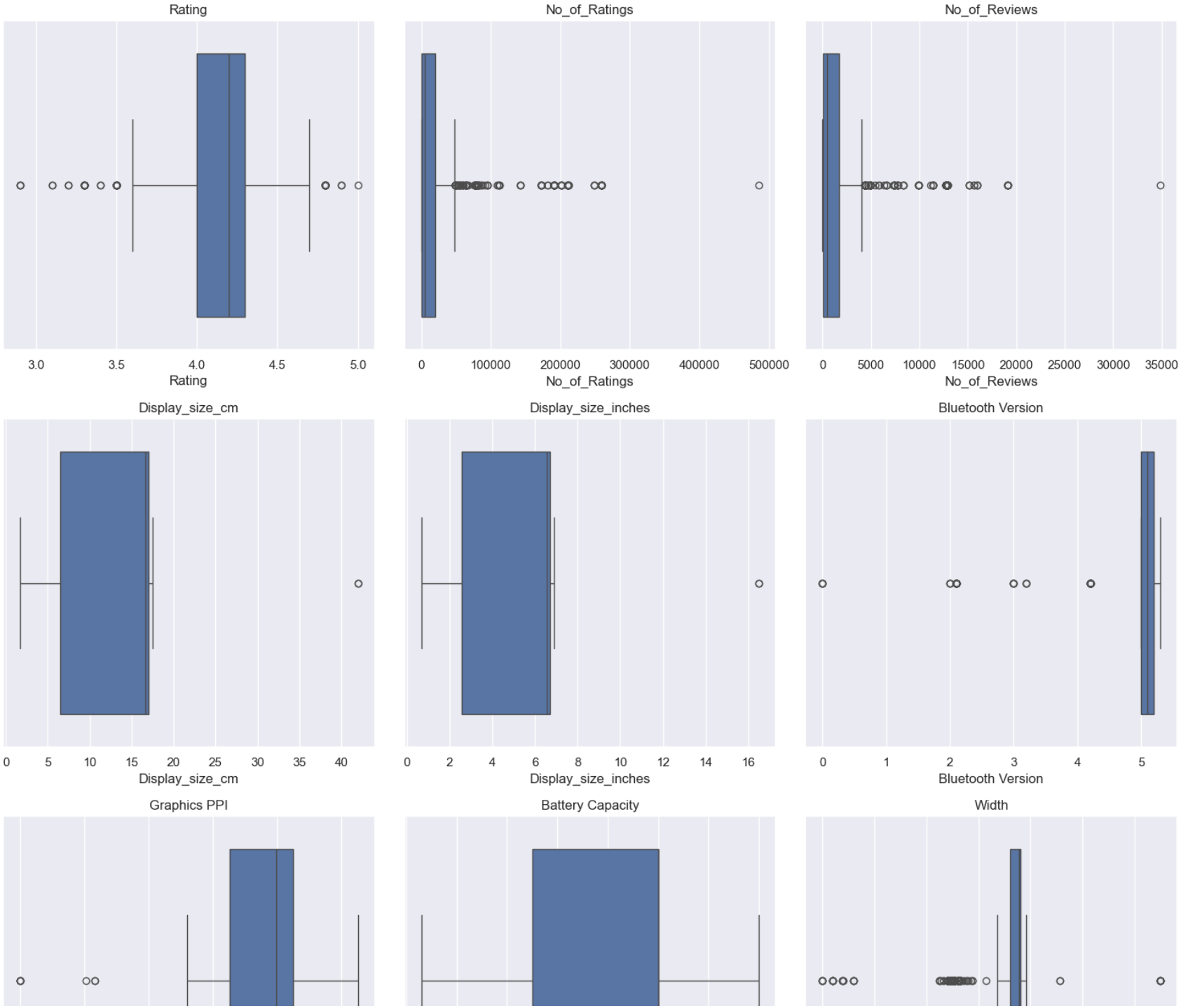


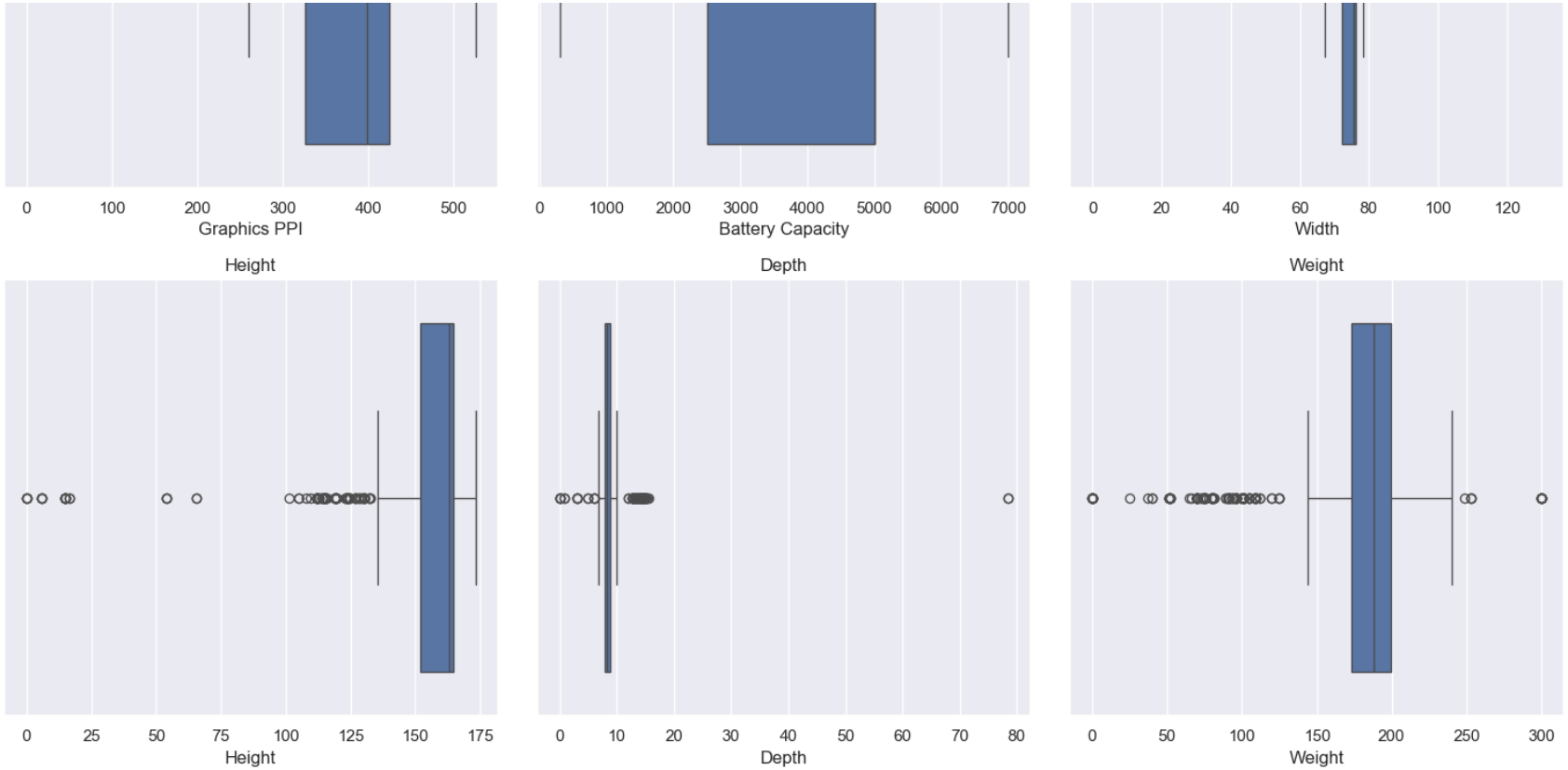


In [203…

```
# Get the name of all colums with data type 'int' or 'float', excluding 'Price'
num_vars = df.select_dtypes(include=["int", "float"]).columns.tolist()
num_vars.remove("Price")  # Remove 'Price'  from the lsit of numerical columns

# Create a figure with subplots
num_cols = len(num_vars)
num_rows = (num_cols + 2) // 3
fig, axs = plt.subplots(nrows=num_rows, ncols=3, figsize=(15, 5 * num_rows))
axs = axs.flatten()
```

```
# Create a box plot for each numerical variable using Seaborn
for i, var in enumerate(num_vars):
    sns.boxplot(x=df[var], ax=axs[i])
    axs[i].set_title(var)

# Remove any extra empty sublpot if needed
if num_cols < len(axs):
    for i in range(num_cols, len(axs)):
        fig.delaxes(axs[i])

# Adjust specing between subplots
fig.tight_layout()

# show plot
plt.show()
```

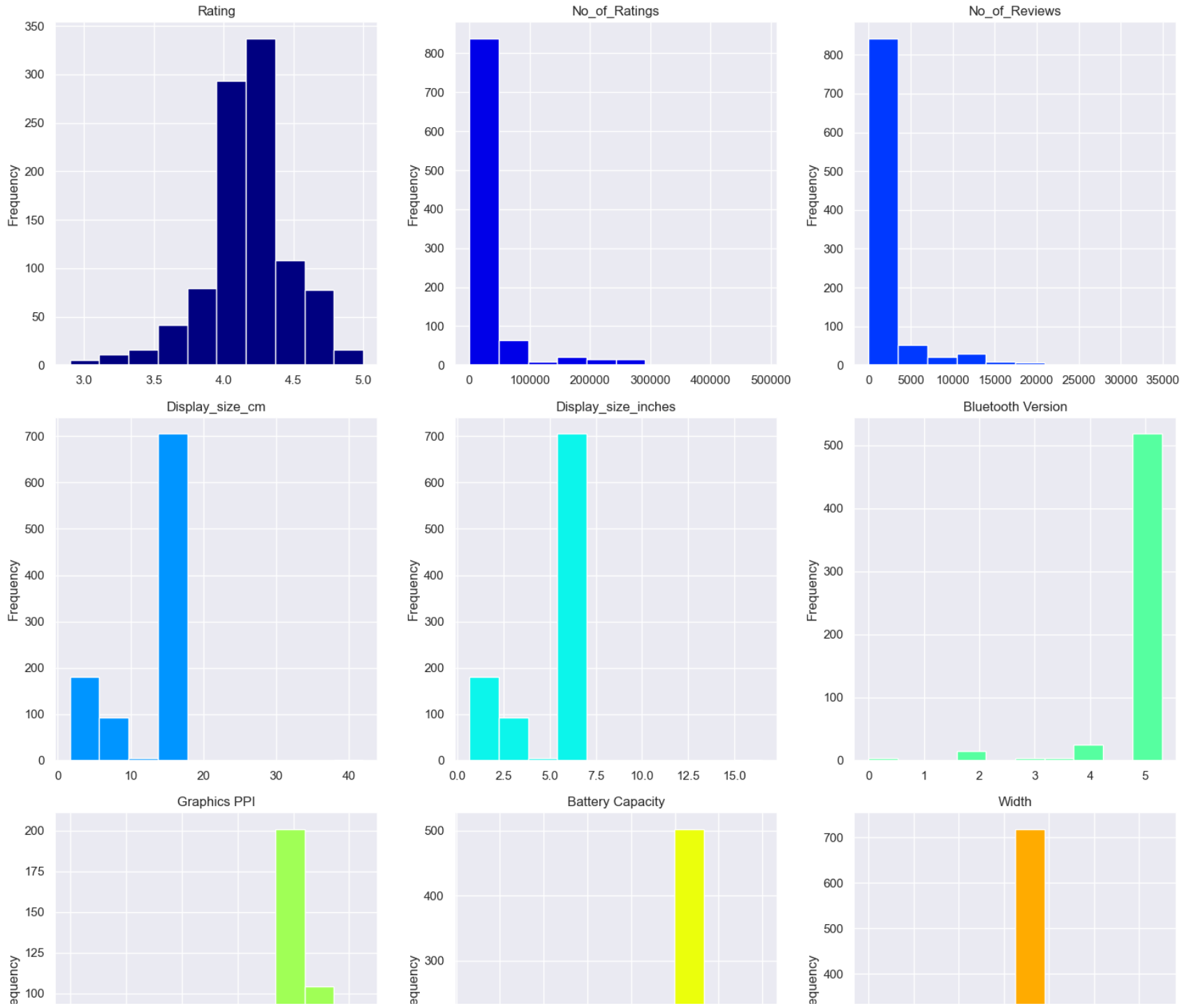

Rating
3.0
3.5
4.0
4.5
5.0
Rating
No_of_Ratings
0
100000
200000
300000
400000
500000
No_of_Ratings
No_of_Reviews
0
5000
10000
15000
20000
25000
30000
35000
No_of_Reviews
Display_size_cm
0
5
10
15
20
25
30
35
40
Display_size_cm
Display_size_inches
0
2
4
6
8
10
12
14
16
Display_size_inches
Bluetooth Version
0
1
2
3
4
5
Bluetooth Version
Graphics PPI
Battery Capacity
Width

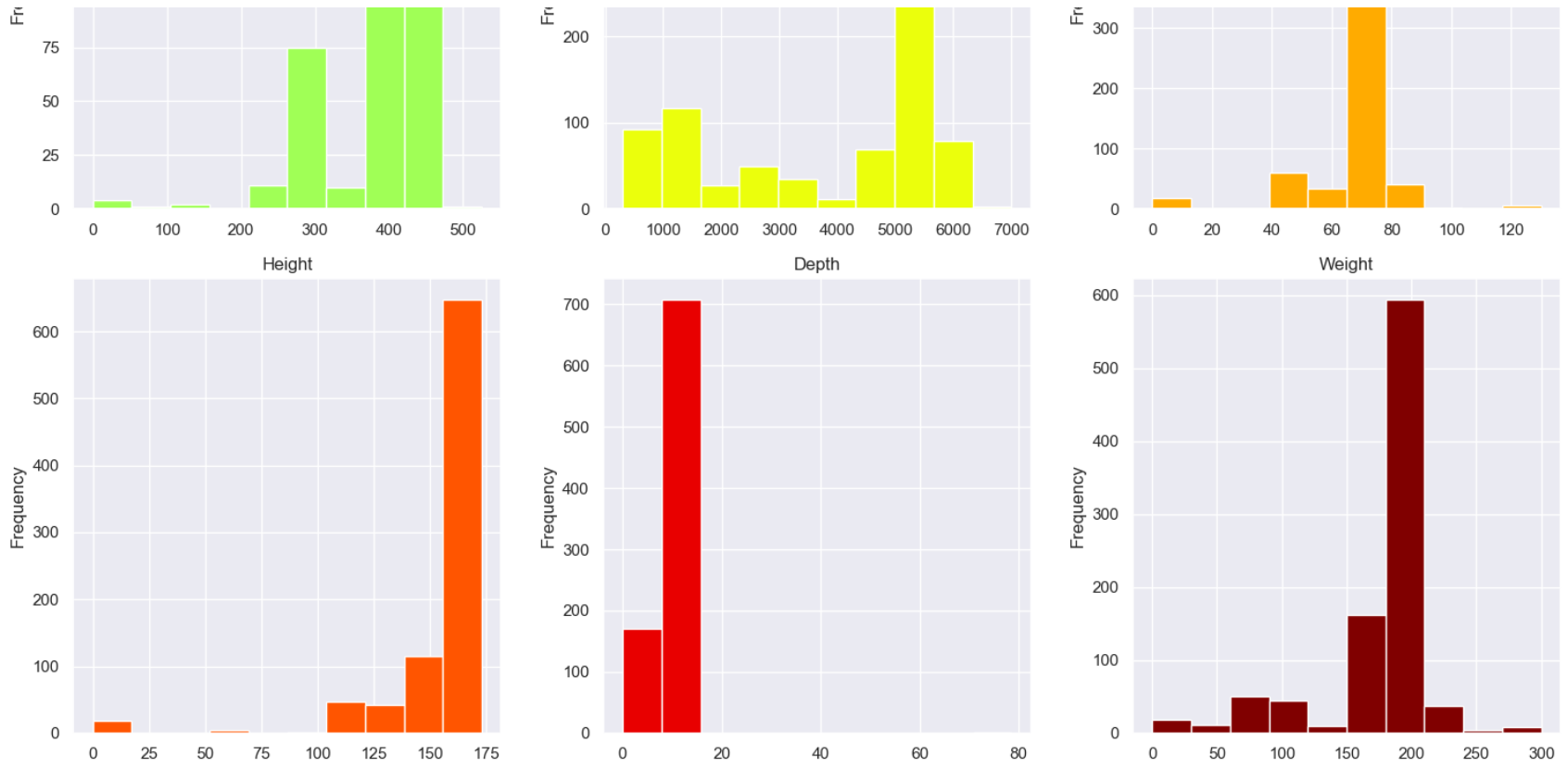


In [204…

```
# Get the name of all columns with data type 'int'
int_vars = df.select_dtypes(include=["int", "float"]).columns.tolist()
int_vars.remove("Price")

# Create a figure with subplots
num_cols = len(int_vars)
num_rows = (num_cols + 2) // 3  # To make sure there are enugh rows fro the subplots
fig, axs = plt.subplots(nrows=num_rows, ncols=3, figsize=(15, 5 * num_rows))
axs = axs.flatten()


# Create colorful histograms for each integer variable
colors = plt.cm.jet(np.linspace(0, 1, num_cols))  # Generate a range of colors
for i, var in enumerate(int_vars):
    df[var].plot.hist(ax=axs[i], color=colors[i])
    axs[i].set_title(var)
```

```
# Remove any extra empty subplots if needed
if num_cols < len(axs):
    for i in range(num_cols, len(axs)):
        fig.delaxes(axs[i])

# Adjust specing between subplots
fig.tight_layout()

# Show plot
plt.show()
```


Rating
Frequency
No_of_Ratings
Frequency
No_of_Reviews
Frequency
Display_size_cm
Frequency
Display_size_inches
Frequency
Bluetooth Version
Frequency
Graphics PPI
Battery Capacity
Width



In [ ]:

Data Preprocessing Part - 2

In [ ]:
```
# Remove column that alot of 'Unknown' Value
df.drop(
    columns=["Browser", "Pre-install Browser", "Frame Rate", "Recursion Type"],
    inplace=True,
)
df.head()
```

In [ ]:
```
# Remove column that have alot of 'Unknown' Value
# Print the column names to check for discrepancies
print(df.columns)
```

```
# Attempt to drop the columns
columns_to_drop = ["Browser", "Pre-installed Browser"]
if all(col in df.columns for col in columns_to_drop):
    df.drop(columns=columns_to_drop, inplace=True)
else:
    print("One or more columns do not exist in the DataFrame.")
```

```
Index(['Brand', 'Price', 'Rating', 'No_of_Ratings', 'No_of_Reviews',
       'Browse Type', 'SIM Type', 'Hybrid Sim Slot', 'Touchscreen',
       'OTG Compatible', 'Quick Charging', 'Display_size_cm',
       'Display_size_inches', 'Resolution Type', 'Operating System',
       'Processor Core', 'Internal Storage', 'Primary Camera Available',
       'Secondary Camera Available', 'HD Recording', 'Full HD Recording',
       'Video Recording', 'Frame Rate', 'Dual Camera Lens', 'Network Type',
       'Supported Networks', 'Internet Connectivity', '3G',
       'Pre-installed Browser', 'Bluetooth Support', 'Bluetooth Version',
       'Wi-Fi', 'Wi-Fi Hotspot', 'NFC', 'EDGE', 'Map Support', 'GPS Support',
       'Smartphone', 'SIM Size', 'Mobile Tracker', 'Removable Battery',
       'Graphics PPI', 'Browser', 'Battery Capacity', 'Width', 'Height',
       'Depth', 'Weight'],
      dtype='object')
```

In [ ]:
```
# Check the amount of missing value
check_missing = df.isnull().sum() * 100 / df.shape[0]
check_missing[check_missing > 0].sort_values(ascending=False)
```

Out[ ]:
```
Mobile Tracker               90.752033
EDGE                         82.113821
3G                           64.532520
Quick Charging               59.451220
Graphics PPI                 58.434959
Map Support                  55.081301
Removable Battery            48.882114
Full HD Recording            48.373984
HD Recording                 48.069106
NFC                          47.560976
Wi-Fi Hotspot                45.528455
Video Recording              43.292683
Bluetooth Version            42.174797
Wi-Fi                        39.024390
GPS Support                  38.617886
Dual Camera Lens             32.621951
Secondary Camera Available   29.674797
Bluetooth Support            27.947154
Processor Core               26.626016
Primary Camera Available     24.695122
Height                       11.077236
Depth                        10.670732
Width                        10.569106
Weight                        4.471545
Smartphone                    3.150407
Frame Rate                    2.743902
No_of_Reviews                 2.439024
No_of_Ratings                 2.439024
OTG Compatible                0.508130
Hybrid Sim Slot               0.508130
Touchscreen                   0.304878
dtype: float64
```

In [ ]:
```
# Remove all of the colum where the null value > 20%
# Get the column with more the 20% missing values
column_to_remove = check_missing[check_missing > 20].index
# Drop the columns with more the 20% missing values
df = df.drop(columns=column_to_remove)
df.head()
```

Out[ ]:

	Brand	Price	Rating	No_of_Ratings	No_of_Reviews	Browse Type	SIM Type	Hybrid Sim Slot	Touchscreen	OTG Compatible	Display_size_cm
0	Apple	52499	4.7	259109.0	12745.0	Smartphones	Dual Sim	No	Yes	No	15.49
1	Xiaomi	6499	4.1	78642.0	4449.0	Smartphones	Dual Sim	No	Yes	Yes	16.56
2	Other	17196	4.4	110949.0	7728.0	Smartphones	Dual Sim	No	Yes	No	16.74
3	Realme	15999	4.3	2633.0	220.0	Smartphones	Dual Sim	No	Yes	Yes	17.07
4	Realme	14999	4.4	13807.0	988.0	Smartphones	Dual Sim	No	Yes	Yes	17.07

In [ ]:
```
df.shape
```

Out[ ]: (984, 26)

In [ ]:
```
# Fill the null value median in Height, Depth, Width, Weight column
df["Height"].fillna(df["Height"].median(), inplace=True)
df["Depth"].fillna(df["Depth"].median(), inplace=True)
df["Width"].fillna(df["Width"].median(), inplace=True)
df["Weight"].fillna(df["Weight"].median(), inplace=True)
```

In [ ]:
```
# Drop all the null value for the rest of column
df.dropna(inplace=True)
df.shape
```

Out[ ]: (899, 26)

Label Encoding for Object Datatypes

In [ ]:
```
# Loop over each column in the DataFrame where is 'object'
for col in df.select_dtypes(include=["object"]).columns:
    # Print the column name and the unique values
    print(f"{col}: {df[col].unique()}")
```

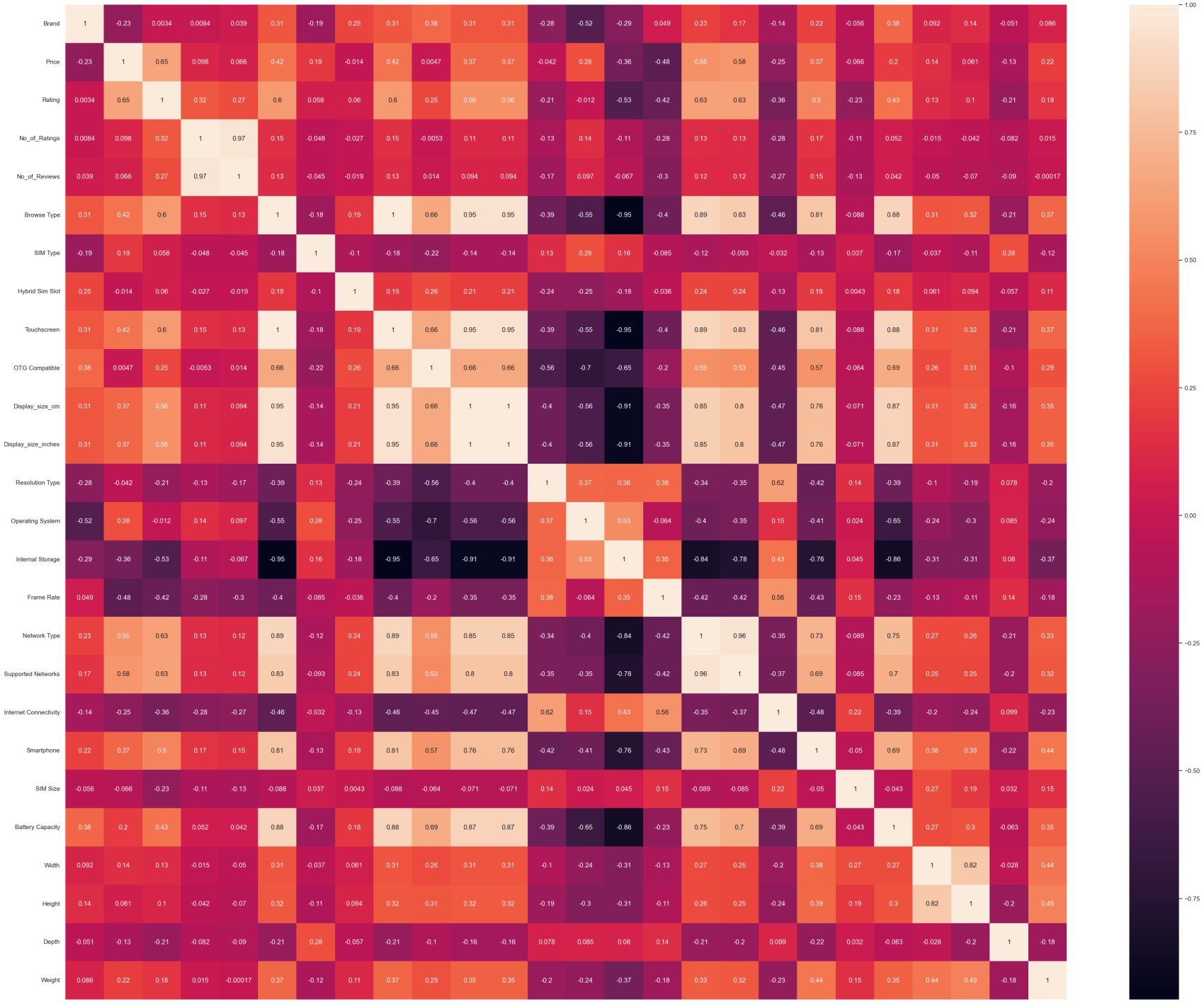
```
Brand: ['Apple' 'Other' 'Realme' 'Xiaomi' 'Vivo' 'Samsung']
Browse Type: ['Smartphones' 'Feature Phones']
SIM Type: ['Dual Sim' 'Dual Sim(Physical + eSIM)' 'Single Sim'
 'Dual Sim(Nano + eSIM)' 'Triple Sim']
Hybrid Sim Slot: ['No' 'Yes']
Touchscreen: ['Yes' 'No']
OTG Compatible: ['No' 'Yes' 'NO' '0']
Resolution Type: ['Retina' 'Unknown' 'Full HD+' 'Other' 'AMOLED' 'Super AMOLED' 'Quad HD']
Operating System: ['iOS' 'Android' 'Unknown' 'Other' 'Symbian' 'RTOS' 'Belle']
Internal Storage: ['GB' 'MB' 'Unknown' 'KB' 'TB']
Frame Rate: ['Medium-Law FPS (60-119)' 'Unknown' 'Medium-Law FPS (120-239)'
 'Low FPS (24-59)' 'High FPS (240+ fps)' 'Other']
Network Type: ['5G' '4G VOLTE' '4G' '2G' '3G']
Supported Networks: ['5G' '4G' '2G' 'CDMA']
Internet Connectivity: ['5G' 'Unknown' '4G' '2G' 'Other' 'GPRS']
Smartphone: ['Yes' 'No']
SIM Size: ['Nano +eSIM' 'Unknown' 'Mini' 'Full Size' 'Micro' 'Normal' 'Other'
 'Macro']
```

In [ ]:
```
from sklearn import preprocessing

# Loop over each column in the DataFrame where dtype is 'object'
for col in df.select_dtypes(include=["object"]).columns:
    # Initialize a LabelEncoder object
    label_encoder = preprocessing.LabelEncoder()

    # Fit the encoder to the unique value in the column
    label_encoder.fit(df[col].unique())

    # Transform the column using the encoder
    df[col] = label_encoder.transform(df[col])

    # Print the column name and the unique encoded values
    print(f"{col}: {df[col].unique()}")
```

```
Brand: [0 1 2 5 4 3]
Browse Type: [1 0]
SIM Type: [0 2 3 1 4]
Hybrid Sim Slot: [0 1]
Touchscreen: [1 0]
OTG Compatible: [2 3 1 0]
Resolution Type: [4 6 1 2 0 5 3]
Operating System: [6 0 5 2 4 3 1]
Internal Storage: [0 2 4 1 3]
Frame Rate: [3 5 2 1 0 4]
Network Type: [4 3 2 0 1]
Supported Networks: [2 1 0 3]
Internet Connectivity: [2 5 1 0 4 3]
Smartphone: [1 0]
SIM Size: [4 7 3 0 2 5 6 1]
```

In [ ]:
```
# Correlation Heatmap
plt.figure(figsize=(40, 32))
sns.heatmap(df.corr(), fmt=".2g", annot=True)
```

Out[ ]:
```
<Axes: >
```


Brand
Price
Rating
No_of_Ratings
No_of_Reviews
Browse Type
SIM Type
Hybrid Sim Slot
Touchscreen
OTG Compatible
Display_size_cm
Display_size_inches
Resolution Type
Operating System
Internal Storage
Frame Rate
Network Type
Supported Networks
Internet Connectivity
Smartphone
SIM Size
Battery Capacity
Width
Height
Depth
Weight
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

In [ ]:
```
# Project By: Uvesh Ahmad
# DataSet Link: https://github.com/Uvesh-Ahmad/Mobile-Phone-Price-in-india
# Portfolio: https://uvesh-ahmad.github.io/uvesh.ah/
# Linkedin : https://www.linkedin.com/in/uvesh-ahmad-a2aa6816a/
```

Thank You !! 😊