mobile-india

October 2, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
df = pd.read_csv('C:/Users/Admin/Downloads/mobiles.csv/mobiles.csv')
```

[3]:
```
df.head()
```

[3]:
```
   Unnamed: 0   Brand                                        Title  \
0           0   APPLE                    APPLE iPhone 13 (Pink, 128 GB)
1           1    POCO                     POCO C51 (Power Black, 64 GB)
2           2  OnePlus  OnePlus Nord CE 2 Lite 5G (Black Dusk, 128 GB)
3           3   realme              realme 11x 5G (Purple Dawn, 128 GB)
4           4   realme           realme 11x 5G (Midnight Black, 128 GB)

          Model Name Model Number  Price  Rating  No_of_Ratings  \
0          iPhone 13    MLPH3HN/A  52499     4.7       259109.0
1                C51    MZB0E6DIN   6499     4.1        78642.0
2  Nord CE 2 Lite 5G  CPH2381/CVD  17196     4.4       110949.0
3             11x 5G      RMX3785  15999     4.3         2633.0
4             11x 5G      RMX3785  14999     4.4        13807.0

   No_of_Reviews                                      In The Box  …  \
0        12745.0    iPhone, USB-C to Lightning Cable, Documentation  …
1         4449.0  Handset, 10W Adapter, USB Cable, Sim Eject Too…  …
2         7728.0  Phone,SIM Tray Ejector, Adapter, Phone Case, U…  …
3          220.0  Handset, Adapter, USB Cable, Important Info Bo…  …
4          988.0  Handset, Adapter, USB Cable, Important Info Bo…  …

                                          Sensors        Browser  \
0  Face ID, Barometer, Three-axis Gyro, Accelerom…         Safari
1                 Accelerometer, Fingerprint Sensor  Google Chrome
2                 Accelerometer, Fingerprint Sensor  Google Chrome
3  Magnetic Induction Sensor, Light Sensor, Proxi…            NaN
4  Magnetic Induction Sensor, Light Sensor, Proxi…            NaN
```

```
                                      Other Features  \
0  Splash, Water and Dust Resistant IP68 Rated (M…
1  Splash Resistant Coating, MIUI Dialer, Upto 3 …
2  Splash Resistant Coating, MIUI Dialer, Upto 3 …
3  COG Sealing Process, In-Cell Touch Panel Techn…
4  COG Sealing Process, In-Cell Touch Panel Techn…

                                          GPS Type  \
0  Built-in GPS, GLONASS, Galileo, QZSS and BeiDou
1                      GPS, AGPS, GLONASS, BEIDOU
2                      GPS, AGPS, GLONASS, BEIDOU
3             GPS, AGPS, BEIDOU, GALILEO, GLONASS
4             GPS, AGPS, BEIDOU, GALILEO, GLONASS

                                       Video Formats Battery Capacity  \
0  HEVC, H.264, MPEG-4 Part 2 and Motion JPEG, HD…         3240 mAh
1                                                NaN          5000 mAh
2                                                NaN          5000 mAh
3  MP4, 3GP, ASF, AVI, FLV, M2TS, MKV, MPG, TS, W…         5000 mAh
4  MP4, 3GP, ASF, AVI, FLV, M2TS, MKV, MPG, TS, W…         5000 mAh

      Width    Height    Depth  Weight
0   71.5 mm  146.7 mm  7.65 mm   173 g
1  76.75 mm  164.9 mm  9.09 mm   192 g
2  76.75 mm  164.9 mm  9.09 mm   192 g
3     76 mm  165.7 mm  7.89 mm   190 g
4     76 mm  165.7 mm  7.89 mm   190 g

[5 rows x 74 columns]
```

[4]:
```
df.columns
```

[4]:
```
Index(['Unnamed: 0', 'Brand', 'Title', 'Model Name', 'Model Number', 'Price',
       'Rating', 'No_of_Ratings', 'No_of_Reviews', 'In The Box', 'Color',
       'Browse Type', 'SIM Type', 'Hybrid Sim Slot', 'Touchscreen',
       'OTG Compatible', 'Quick Charging', 'Sound Enhancements',
       'Display_size_cm', 'Display_size_inches', 'Resolution',
       'Resolution Type', 'Display Type', 'Other Display Features',
       'Operating System', 'Processor Type', 'Processor Core',
       'Operating Frequency', 'Internal Storage', 'Primary Camera Available',
       'Primary Camera', 'Primary Camera Features',
       'Secondary Camera Available', 'Secondary Camera',
       'Secondary Camera Features', 'Flash', 'HD Recording',
       'Full HD Recording', 'Video Recording', 'Video Recording Resolution',
       'Digital Zoom', 'Frame Rate', 'Dual Camera Lens', 'Call Wait/Hold',
       'Network Type', 'Supported Networks', 'Internet Connectivity', '3G',
       'Pre-installed Browser', 'Bluetooth Support', 'Bluetooth Version',
```

```
       'Wi-Fi', 'Wi-Fi Version', 'Wi-Fi Hotspot', 'NFC', 'EDGE', 'Map Support',
       'GPS Support', 'Smartphone', 'SIM Size', 'Mobile Tracker',
       'Removable Battery', 'SMS', 'Graphics PPI', 'Sensors', 'Browser',
       'Other Features', 'GPS Type', 'Video Formats', 'Battery Capacity',
       'Width', 'Height', 'Depth', 'Weight'],
      dtype='object')
```

1 Data Preprocessing

[5]:
```
df.drop(columns=['Unnamed: 0','Title','Model Name'],inplace=True)
df.head()
```

[5]:

	Brand	Model Number	Price	Rating	No_of_Ratings	No_of_Reviews	In The Box	Color	Browse Type	SIM Type	…	Sensors	Browser	Other Features
0	APPLE	MLPH3HN/A	52499	4.7	259109.0	12745.0	iPhone, USB-C to Lightning Cable, Documentation	Pink	Smartphones	Dual Sim	…	Face ID, Barometer, Three-axis Gyro, Accelerom…	Safari	Splash, Water and Dust Resistant IP68 Rated (M…
1	POCO	MZB0E6DIN	6499	4.1	78642.0	4449.0	Handset, 10W Adapter, USB Cable, Sim Eject Too…	Power Black	Smartphones	Dual Sim	…	Accelerometer, Fingerprint Sensor	Google Chrome	Splash Resistant Coating, MIUI Dialer, Upto 3 …
2	OnePlus	CPH2381/CVD	17196	4.4	110949.0	7728.0	Phone,SIM Tray Ejector, Adapter, Phone Case, U…	Black Dusk	Smartphones	Dual Sim	…	Accelerometer, Fingerprint Sensor	Google Chrome	Splash Resistant Coating, MIUI Dialer, Upto 3 …
3	realme	RMX3785	15999	4.3	2633.0	220.0	Handset, Adapter, USB Cable, Important Info Bo…	Purple Dawn	Smartphones	Dual Sim	…	Magnetic Induction Sensor, Light Sensor, Proxi…	NaN	COG Sealing Process, In-Cell Touch Panel Techn…
4	realme	RMX3785	14999	4.4	13807.0	988.0	Handset, Adapter, USB Cable, Important Info Bo…	Midnight Black	Smartphones	Dual Sim	…	Magnetic Induction Sensor, Light Sensor, Proxi…	NaN	COG Sealing Process, In-Cell Touch Panel Techn…

```
                                          GPS Type  \
0  Built-in GPS, GLONASS, Galileo, QZSS and BeiDou
1                      GPS, AGPS, GLONASS, BEIDOU
2                      GPS, AGPS, GLONASS, BEIDOU
3             GPS, AGPS, BEIDOU, GALILEO, GLONASS
4             GPS, AGPS, BEIDOU, GALILEO, GLONASS

                                      Video Formats  Battery Capacity  \
0  HEVC, H.264, MPEG-4 Part 2 and Motion JPEG, HD…          3240 mAh
1                                              NaN            5000 mAh
2                                              NaN            5000 mAh
3  MP4, 3GP, ASF, AVI, FLV, M2TS, MKV, MPG, TS, W…          5000 mAh
4  MP4, 3GP, ASF, AVI, FLV, M2TS, MKV, MPG, TS, W…          5000 mAh

      Width    Height    Depth Weight
0   71.5 mm  146.7 mm  7.65 mm  173 g
1  76.75 mm  164.9 mm  9.09 mm  192 g
2  76.75 mm  164.9 mm  9.09 mm  192 g
3     76 mm  165.7 mm  7.89 mm  190 g
4     76 mm  165.7 mm  7.89 mm  190 g

[5 rows x 71 columns]
```

[6]:
```
df.shape
```

[6]:
```
(984, 71)
```

[7]:
```
df.select_dtypes(include='object').nunique()
```

[7]:
```
Brand                38
Model Number        563
In The Box          240
Color               376
Browse Type           2
                    …
Battery Capacity     55
Width               115
Height              130
Depth                98
Weight              104
Length: 65, dtype: int64
```

[8]:
```
df['Battery Capacity'] = df['Battery Capacity'].str.replace('[^0-9.
  ↪]','',regex=True)
```

[9]:
```
df['Battery Capacity'] = pd.to_numeric(df['Battery Capacity'])
```

[10]:
```
df['Width'][0]
```

[10]:
```
'71.5 mm'
```

[11]:
```
df['Width'] = df['Width'].str.replace('[^0-9.]','',regex = True)
df['Width'] = pd.to_numeric(df['Width'])
```

[12]:
```
df['Height'][0]
```

[12]:
```
'146.7 mm'
```

[13]:
```
df['Height'] = df['Height'].str.replace('[^0-9.]','',regex=True)
df['Height'] = pd.to_numeric(df['Height'])
```

[14]:
```
df['Depth'][0]
```

[14]:
```
'7.65 mm'
```

[15]:
```
df['Depth'] = df['Depth'].str.replace('[^0-9.]','',regex=True)
df['Depth'] = pd.to_numeric(df['Depth'])
```

[16]:
```
df['Weight'][0]
```

[16]:
```
'173 g'
```

[17]:
```
df['Weight'] = df['Weight'].str.replace('[^0-9.]','',regex=True)
df['Weight'] = pd.to_numeric(df['Weight'])
```

[18]:
```
df['Graphics PPI'][0]
```

[18]:
```
'460 PPI'
```

[19]:
```
df['Graphics PPI'] = df['Graphics PPI'].str.replace('[^0-9.]','',regex=True)
df['Graphics PPI'] = pd.to_numeric(df['Graphics PPI'])
```

[20]:
```
df.head()
```

[20]:

	Brand	Model Number	Price	Rating	No_of_Ratings	No_of_Reviews	\
0	APPLE	MLPH3HN/A	52499	4.7	259109.0	12745.0	
1	POCO	MZB0E6DIN	6499	4.1	78642.0	4449.0	
2	OnePlus	CPH2381/CVD	17196	4.4	110949.0	7728.0	
3	realme	RMX3785	15999	4.3	2633.0	220.0	
4	realme	RMX3785	14999	4.4	13807.0	988.0	

	In The Box	Color	\
0	iPhone, USB-C to Lightning Cable, Documentation	Pink	
1	Handset, 10W Adapter, USB Cable, Sim Eject Too…	Power Black	

```
2  Phone,SIM Tray Ejector, Adapter, Phone Case, U…      Black Dusk
3  Handset, Adapter, USB Cable, Important Info Bo…     Purple Dawn
4  Handset, Adapter, USB Cable, Important Info Bo…  Midnight Black

   Browse Type  SIM Type  …  \
0  Smartphones  Dual Sim  …
1  Smartphones  Dual Sim  …
2  Smartphones  Dual Sim  …
3  Smartphones  Dual Sim  …
4  Smartphones  Dual Sim  …

                                              Sensors        Browser  \
0  Face ID, Barometer, Three-axis Gyro, Accelerom…         Safari
1                  Accelerometer, Fingerprint Sensor  Google Chrome
2                  Accelerometer, Fingerprint Sensor  Google Chrome
3  Magnetic Induction Sensor, Light Sensor, Proxi…            NaN
4  Magnetic Induction Sensor, Light Sensor, Proxi…            NaN

                                       Other Features  \
0  Splash, Water and Dust Resistant IP68 Rated (M…
1  Splash Resistant Coating, MIUI Dialer, Upto 3 …
2  Splash Resistant Coating, MIUI Dialer, Upto 3 …
3  COG Sealing Process, In-Cell Touch Panel Techn…
4  COG Sealing Process, In-Cell Touch Panel Techn…

                                         GPS Type  \
0  Built-in GPS, GLONASS, Galileo, QZSS and BeiDou
1                      GPS, AGPS, GLONASS, BEIDOU
2                      GPS, AGPS, GLONASS, BEIDOU
3             GPS, AGPS, BEIDOU, GALILEO, GLONASS
4             GPS, AGPS, BEIDOU, GALILEO, GLONASS

                                       Video Formats  Battery Capacity  Width  \
0  HEVC, H.264, MPEG-4 Part 2 and Motion JPEG, HD…              3240  71.50
1                                                NaN              5000  76.75
2                                                NaN              5000  76.75
3  MP4, 3GP, ASF, AVI, FLV, M2TS, MKV, MPG, TS, W…              5000  76.00
4  MP4, 3GP, ASF, AVI, FLV, M2TS, MKV, MPG, TS, W…              5000  76.00

  Height Depth Weight
0  146.7  7.65  173.0
1  164.9  9.09  192.0
2  164.9  9.09  192.0
3  165.7  7.89  190.0
4  165.7  7.89  190.0

[5 rows x 71 columns]
```

[21]:
```
df.columns
```

[21]:
```
Index(['Brand', 'Model Number', 'Price', 'Rating', 'No_of_Ratings',
       'No_of_Reviews', 'In The Box', 'Color', 'Browse Type', 'SIM Type',
       'Hybrid Sim Slot', 'Touchscreen', 'OTG Compatible', 'Quick Charging',
       'Sound Enhancements', 'Display_size_cm', 'Display_size_inches',
       'Resolution', 'Resolution Type', 'Display Type',
       'Other Display Features', 'Operating System', 'Processor Type',
       'Processor Core', 'Operating Frequency', 'Internal Storage',
       'Primary Camera Available', 'Primary Camera', 'Primary Camera Features',
       'Secondary Camera Available', 'Secondary Camera',
       'Secondary Camera Features', 'Flash', 'HD Recording',
       'Full HD Recording', 'Video Recording', 'Video Recording Resolution',
       'Digital Zoom', 'Frame Rate', 'Dual Camera Lens', 'Call Wait/Hold',
       'Network Type', 'Supported Networks', 'Internet Connectivity', '3G',
       'Pre-installed Browser', 'Bluetooth Support', 'Bluetooth Version',
       'Wi-Fi', 'Wi-Fi Version', 'Wi-Fi Hotspot', 'NFC', 'EDGE', 'Map Support',
       'GPS Support', 'Smartphone', 'SIM Size', 'Mobile Tracker',
       'Removable Battery', 'SMS', 'Graphics PPI', 'Sensors', 'Browser',
       'Other Features', 'GPS Type', 'Video Formats', 'Battery Capacity',
       'Width', 'Height', 'Depth', 'Weight'],
      dtype='object')
```

[22]:
```
df.drop(columns =['Model Number','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'],inplace = True)
```

[23]:
```
df.drop(columns=['SMS'],inplace=True)
```

[24]:
```
df.head()
```

[24]:

	Brand	Price	Rating	No_of_Ratings	No_of_Reviews	Browse Type	\
0	APPLE	52499	4.7	259109.0	12745.0	Smartphones	
1	POCO	6499	4.1	78642.0	4449.0	Smartphones	
2	OnePlus	17196	4.4	110949.0	7728.0	Smartphones	
3	realme	15999	4.3	2633.0	220.0	Smartphones	
4	realme	14999	4.4	13807.0	988.0	Smartphones	

	SIM Type	Hybrid Sim Slot	Touchscreen	OTG Compatible	…	SIM Size	\
0	Dual Sim	No	Yes	No	…	Nano + eSIM	
1	Dual Sim	No	Yes	Yes	…	Nano Sim	
2	Dual Sim	No	Yes	No	…	Nano Sim	

```
3  Dual Sim              No         Yes            Yes  …     Nano Sim
4  Dual Sim              No         Yes            Yes  …     Nano Sim

   Mobile Tracker  Removable Battery Graphics PPI        Browser  \
0             Yes                 No        460.0         Safari
1             NaN                 No          NaN  Google Chrome
2             NaN                 No          NaN  Google Chrome
3             NaN                NaN        391.0            NaN
4             NaN                NaN        391.0            NaN

  Battery Capacity  Width Height Depth Weight
0             3240  71.50  146.7  7.65  173.0
1             5000  76.75  164.9  9.09  192.0
2             5000  76.75  164.9  9.09  192.0
3             5000  76.00  165.7  7.89  190.0
4             5000  76.00  165.7  7.89  190.0

[5 rows x 49 columns]
```

[25]:
```
df.select_dtypes(include='object').nunique()
```

[25]:
```
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
```

[26]:
```
df.shape
```

[26]:
```
(984, 49)
```

[27]:
```
df['Brand'].unique()
```

[27]:
```
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)
```

[28]:
```
def segment_brand(brand):
    apple_brand = ['APPLE']
    samsung_brand = ['SAMSUNG']
    xiaomi_brand = ['Mi','Xiaomi','REDMI','POCO']
    oneplus_brand = ['OnePlus']
    realme_brand = ['realme']
    vivo_brand = ['vivo']
    other_brand = ['MOTOROLA','Infinix', 'Nokia','OPPO', 'Micromax', 'MarQ',␣
 ↪'LAVA',
        'Google', 'itel', 'Kechaoda', 'HOTLINE', 'Tecno', 'KARBONN', 'I',
        'GFive', 'DIZO', 'Snexian', 'Good', 'Eunity', 'Energizer', 'IAIR',
        'Cellecor', 'IQOO','MTR', 'Nothing','SAREGAMA',
        'Peace', 'UiSmart', 'Itel']

    if brand in apple_brand:
        return 'Apple'
    if brand in samsung_brand:
        return 'SAMSUNG'
    if brand in xiaomi_brand:
        return 'Xiaomi'
    if brand in oneplus_brand:
```

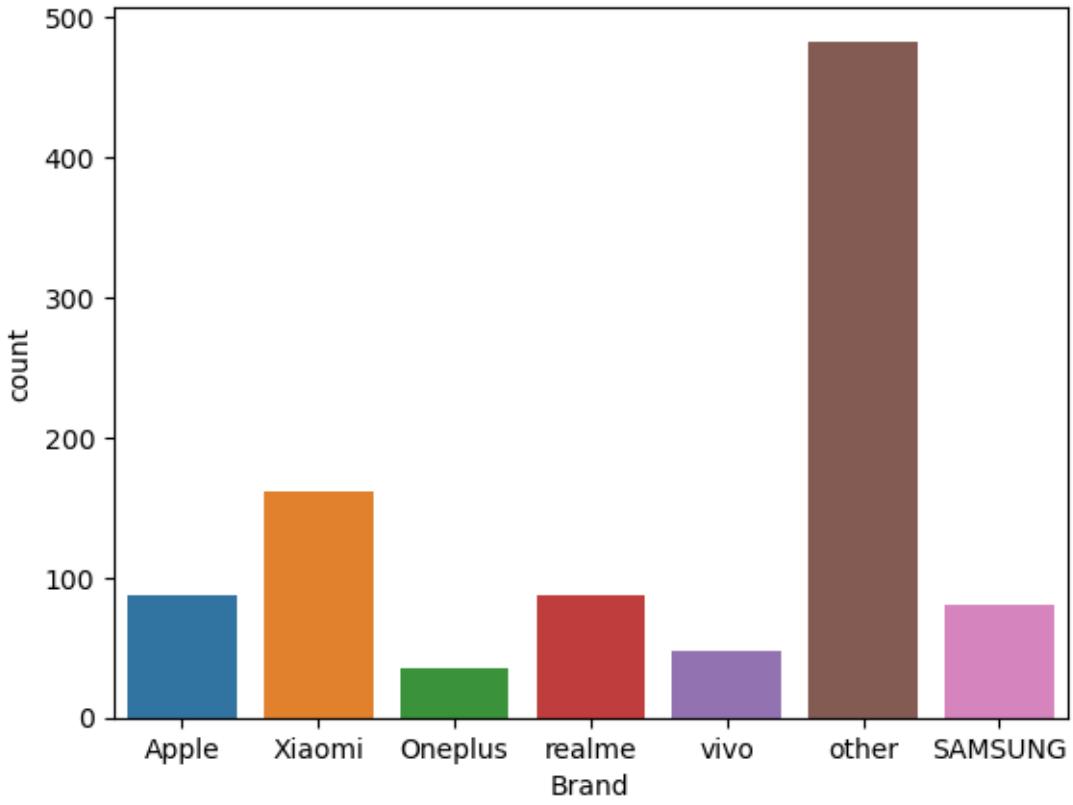
```
        return 'Oneplus'
    if brand in realme_brand:
        return 'realme'
    if brand in vivo_brand:
        return 'vivo'
    else:
        return 'other'
```

[29]:
```
df['Brand'] = df['Brand'].apply(segment_brand)
```

[30]:
```
def return_countplot(column):
    return sns.countplot(x = column,data=df)
```

[31]:
```
return_countplot('Brand')
```

[31]: <AxesSubplot:xlabel='Brand', ylabel='count'>



[32]:
```
df['Resolution Type'].unique()
```

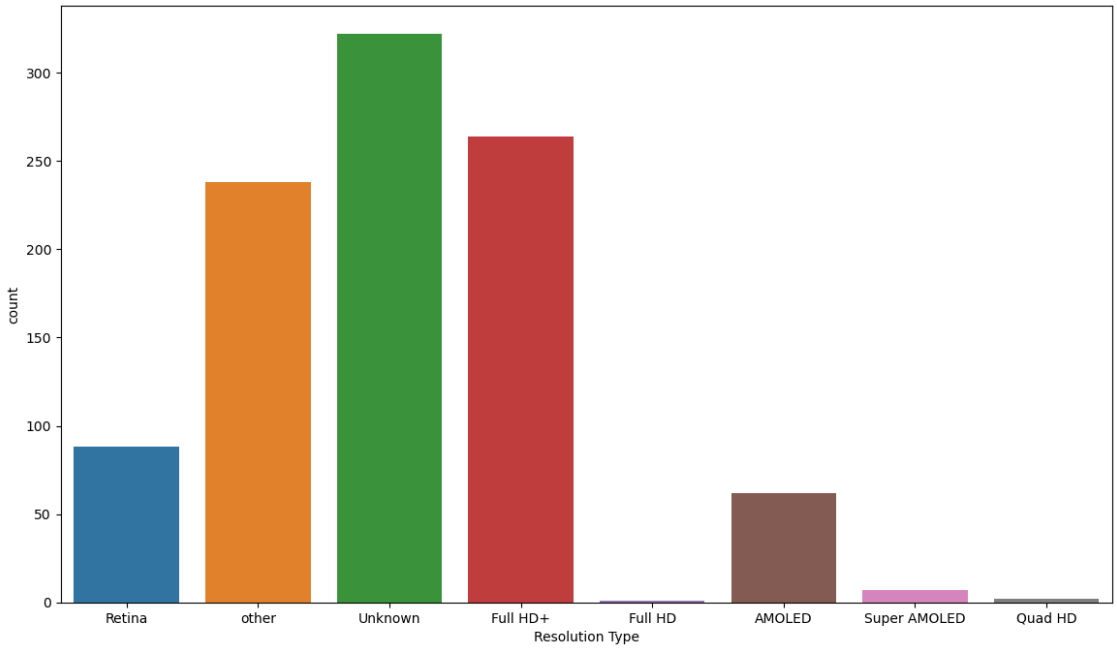
[32]:
```
array(['Super Retina XDR Display', 'HD+', nan, 'Full HD+', 'HD',
       'Full HD', 'Liquid Retina HD Display', 'QVGA',
```

```
       '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)
```

[33]:
```
def segment_resolution_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 'Full HD' in resolution:
        return 'Full HD'
    elif 'Quad HD' in resolution:
        return 'Quad HD'
    else:
        return 'other'

df['Resolution Type'] = df['Resolution Type'].apply(segment_resolution_type)
```

[34]:
```
plt.figure(figsize=(14,8))
return_countplot('Resolution Type')
```

[34]:
```
<AxesSubplot:xlabel='Resolution Type', ylabel='count'>
```



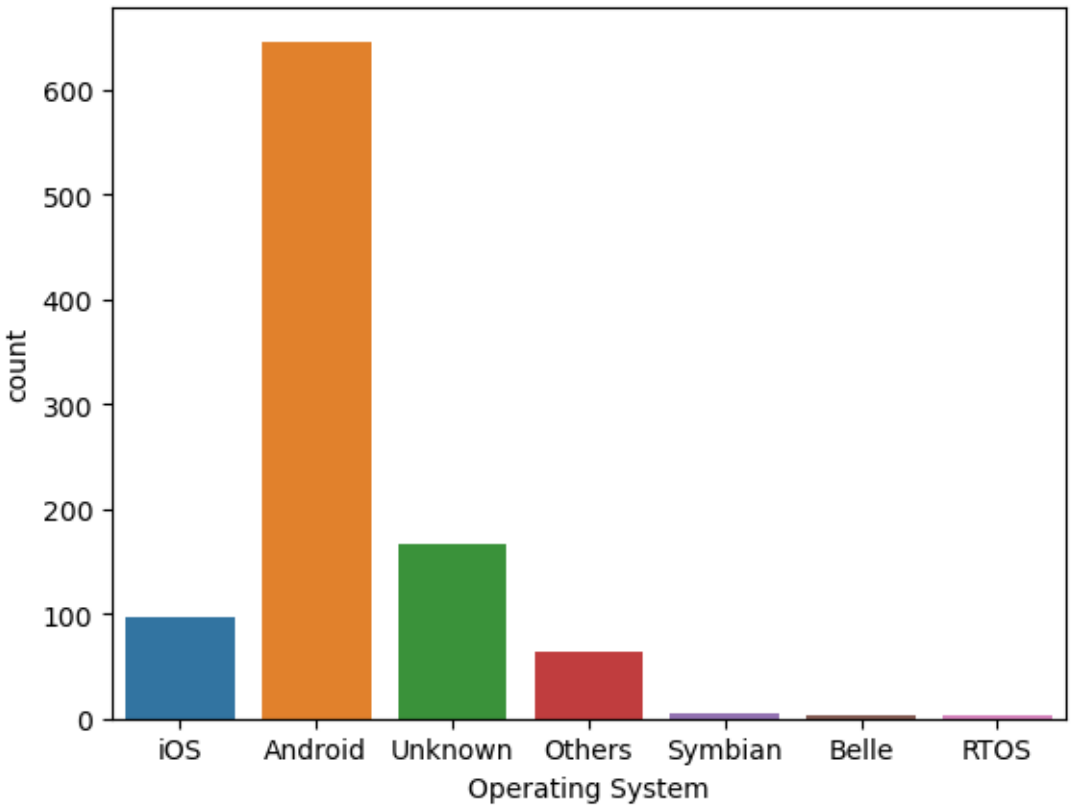
[35]:
```
df['Operating System'].unique()
```

[35]:
```
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)
```

[36]:
```
def segment_operating_system(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 'Others'


df['Operating System'] = df['Operating System'].apply(segment_operating_system)
```

[37]:
```
return_countplot('Operating System')
```

[37]: <AxesSubplot:xlabel='Operating System', ylabel='count'>



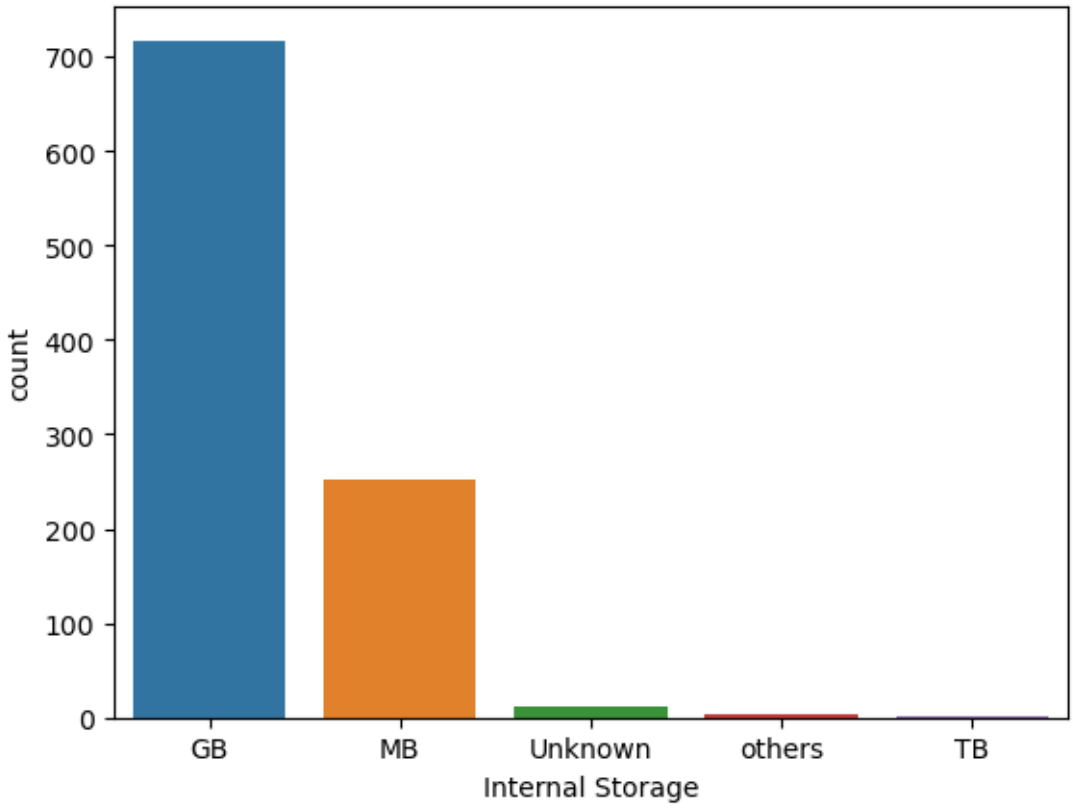
[38]:
```
df['Internal Storage'].unique()
```

[38]:
```
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)
```

[39]:
```
def segment_internal_storage(storage):
    if pd.isna(storage):
        return 'Unknown'
    elif 'TB' in storage:
        return 'TB'
    elif 'GB' in storage:
        return 'GB'
    elif 'MB' in storage:
        return 'MB'
    else:
        return 'others'

df['Internal Storage'] = df['Internal Storage'].apply(segment_internal_storage)
```

[40]:
```
return_countplot('Internal Storage')
```

[40]:
```
<AxesSubplot:xlabel='Internal Storage', ylabel='count'>
```



[41]:
```
df['Frame Rate'][0:10]
```

[41]:
```
0               24 fps, 25 fps, 30 fps, 60 fps
1                                       60 Hz
2                                         NaN
3                              120 fps, 30 fps
4                              120 fps, 30 fps
5                                       30 fps
6               24 fps, 25 fps, 30 fps, 60 fps
7                                       60 Hz
8                                         NaN
9    960 fps, 240 fps, 120 fps, 60 fps, 30 fps
Name: Frame Rate, dtype: object
```

[42]:
```
df['Frame Rate'].dtypes
```

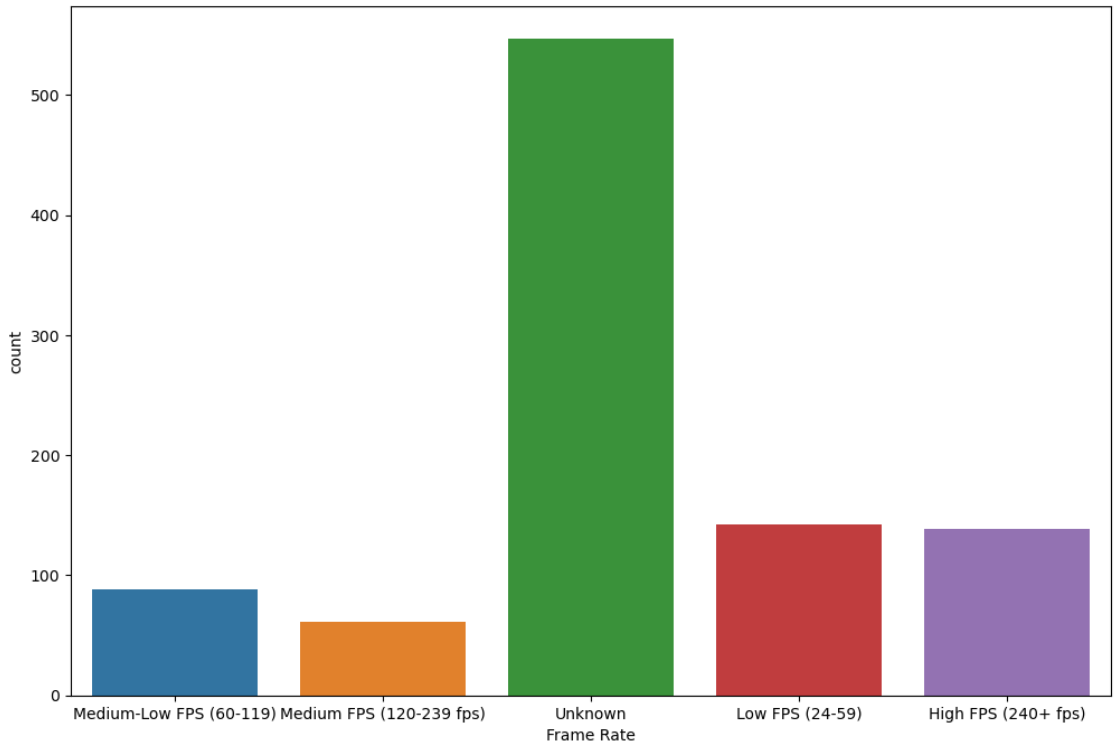
[42]:
```
dtype('O')
```

[43]:
```
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:
        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 FPS (120-239 fps)'
            elif max_fps >=60:
                return 'Medium-Low FPS (60-119)'
            else:
                return 'Low FPS (24-59)'
    elif 'Hz' in frame_rate:
        hz_value = float(frame_rate.replace(' Hz',''))
        fps_value = hz_to_fps(hz_value)
        if fps_value >=240:
            return 'High FPS (240+ fps)'
        elif fps_value >=120:
            return 'Medium FPS (120-239 fps)'
        elif fps_value >=60:
            return 'Medium-Low FPS (60-119 fps)'
        else:
            return 'Low FPS(24-59 fps)'
    else:
        return 'Other'

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

[44]:
```
plt.figure(figsize=(12,8))
return_countplot('Frame Rate')
```

[44]:
```
<AxesSubplot:xlabel='Frame Rate', ylabel='count'>
```



[45]:
```
df['Network Type'].unique()
```

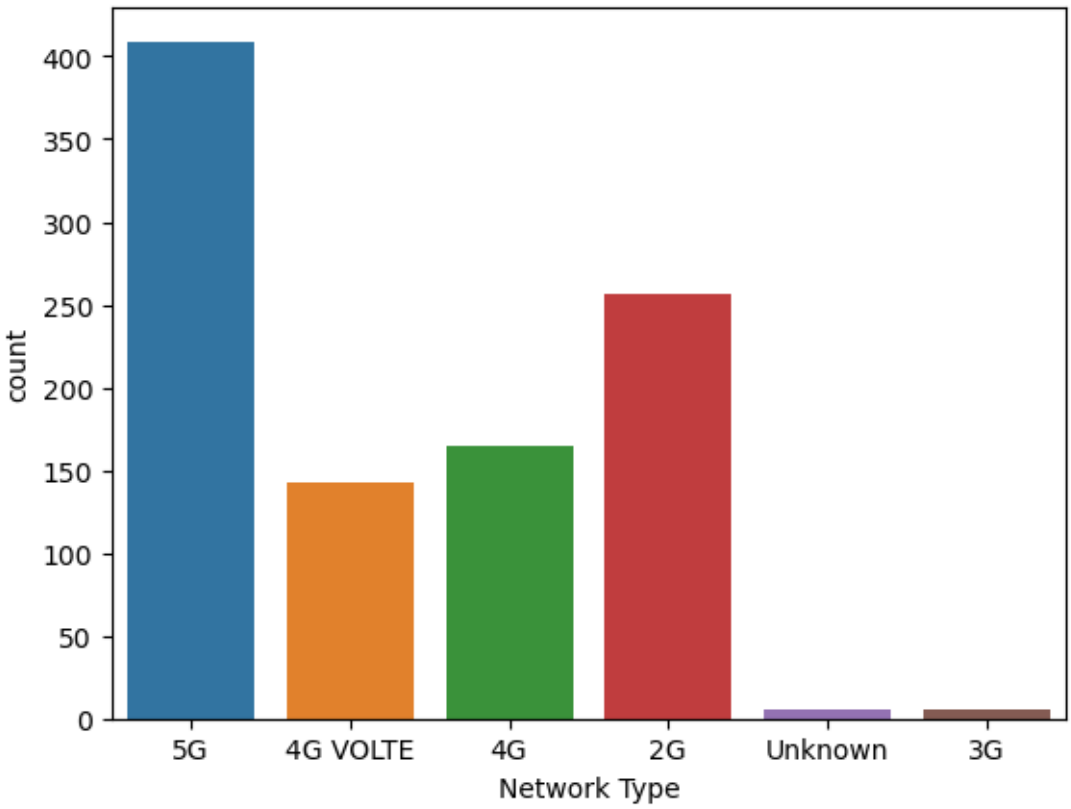
[45]:
```
array(['5G, 4G, 3G, 2G', '2G, 3G, 4G, 4G VOLTE', '5G',
       '5G, 4G VOLTE, 4G, 3G, 2G', '4G VOLTE, 4G, 3G, 2G', '5G, 4G, 3G',
       '2G, 3G, 4G, 5G', '4G, 3G, 2G', '2G, 3G, 4G',
       '2G, 3G, 4G, 4G VOLTE, 5G', '4G VOLTE', '2G', '5G, 4G', '4G', nan,
       '4G VOLTE, 4G', '5G, 4G VOLTE', '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)
```

[46]:
```
def segment_network(network):
    if pd.isna(network):
        return 'Unknown'
    elif '5G' in network:
        return '5G'
    elif '4G VOLTE' in network:
        return '4G VOLTE'
    elif '4G' in network:
        return '4G'
    elif '3G' in network:
        return '3G'
    elif '2G' in network:
        return '2G'
```

```
    else:
        return 'Other'
df['Network Type'] = df['Network Type'].apply(segment_network)
```

[47]:
```
return_countplot('Network Type')
```

[47]:
```
<AxesSubplot:xlabel='Network Type', ylabel='count'>
```



[48]:
```
df['Supported Networks'].unique()
```

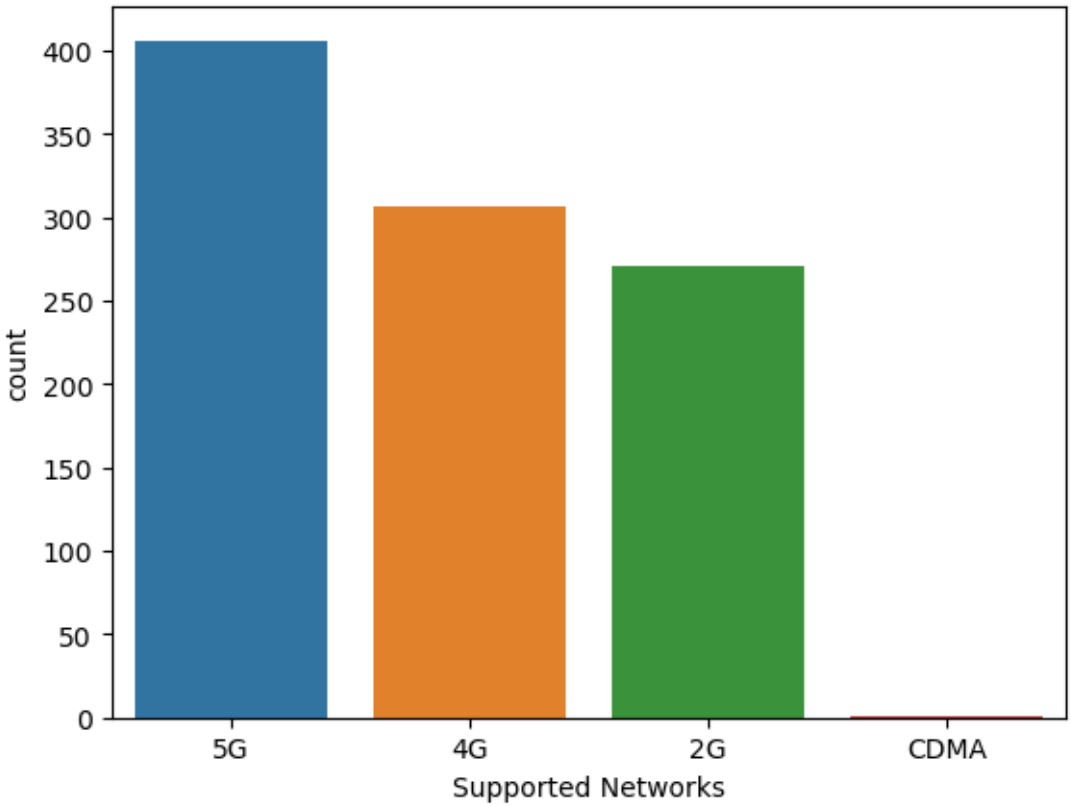
[48]:
```
array(['5G, 4G VoLTE, 4G LTE, UMTS, GSM', '4G LTE, 4G VoLTE, GSM, WCDMA',
       '5G', '5G, 4G VoLTE, 4G LTE, WCDMA, GSM',
       '4G VoLTE, 4G LTE, WCDMA, GSM', '5G, 4G LTE, WCDMA',
       '4G LTE, 5G, GSM, UMTS', '4G LTE, WCDMA, GSM',
       '4G LTE, GSM, WCDMA', '4G LTE, 5G, GSM, WCDMA',
       '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)
```

[49]:
```
def segment_network(network):
    if pd.isna(network):
        return 'Unknown'
    elif '5G' in network:
        return '5G'
    elif '4G VoLTE' in network or '4G LTE' in network:
        return '4G'
    elif 'UMTS' in network or 'WCDMA' in network:
        return '3G'
    elif 'GSM' in network:
        return '2G'
    elif 'CDMA' in network:
        return 'CDMA'
    else:
        return 'Mixed'
df['Supported Networks'] = df['Supported Networks'].apply(segment_network)
```

[50]:
```
return_countplot('Supported Networks')
```

[50]:
```
<AxesSubplot:xlabel='Supported Networks', ylabel='count'>
```



[51]:
```
df['Internet Connectivity'].unique()
```

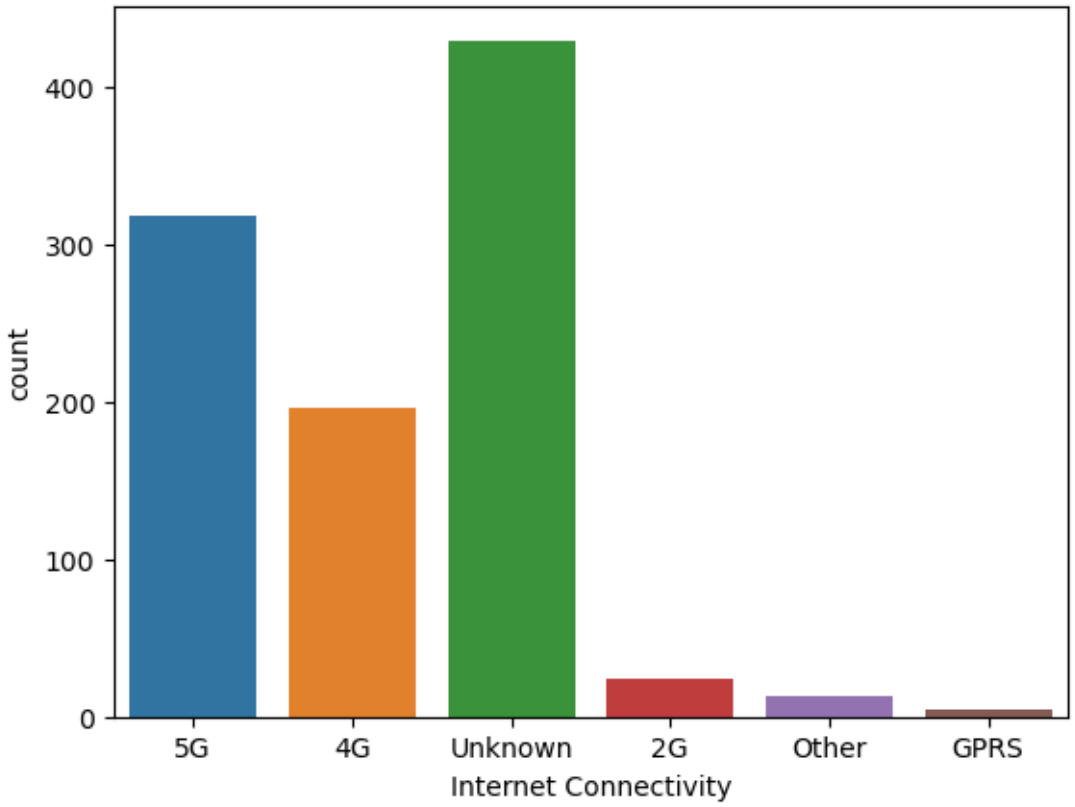
[51]:
```
array(['5G, 4G, 3G, Wi-Fi, EDGE', '4G, 3G, Wi-Fi, EDGE, GPRS', nan,
       '5G, 4G, 3G, EDGE, GPRS, Wi-Fi', '5G, 4G, 3G, Wi-Fi',
       '4G, 3G, EDGE, GPRS, Wi-Fi', '4G, 3G, Wi-Fi',
       '5G, 4G, 3G, Wi-Fi, EDGE, GPRS', '2G', '4G, 3G, Wi-Fi, GPRS', '0',
       '4G. 3G, Wi-Fi', '4G, 3G, EDGE, 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)
```

[52]:
```
def segment_connect(connect):
    if pd.isna(connect):
        return 'Unknown'
    elif '5G' in connect:
```

```
        return '5G'
    elif '4G' in connect:
        return '4G'
    elif '3G' in connect:
        return '3G'
    elif '2G' in connect:
        return '2G'
    elif 'Wi-Fi' in connect:
        return 'Wi-Fi'
    elif 'GPRS' in connect:
        return 'GPRS'
    else:
        return 'Other'

df['Internet Connectivity'] = df['Internet Connectivity'].apply(segment_connect)
```

[53]:
```
return_countplot('Internet Connectivity')
```

[53]:
```
<AxesSubplot:xlabel='Internet Connectivity', ylabel='count'>
```



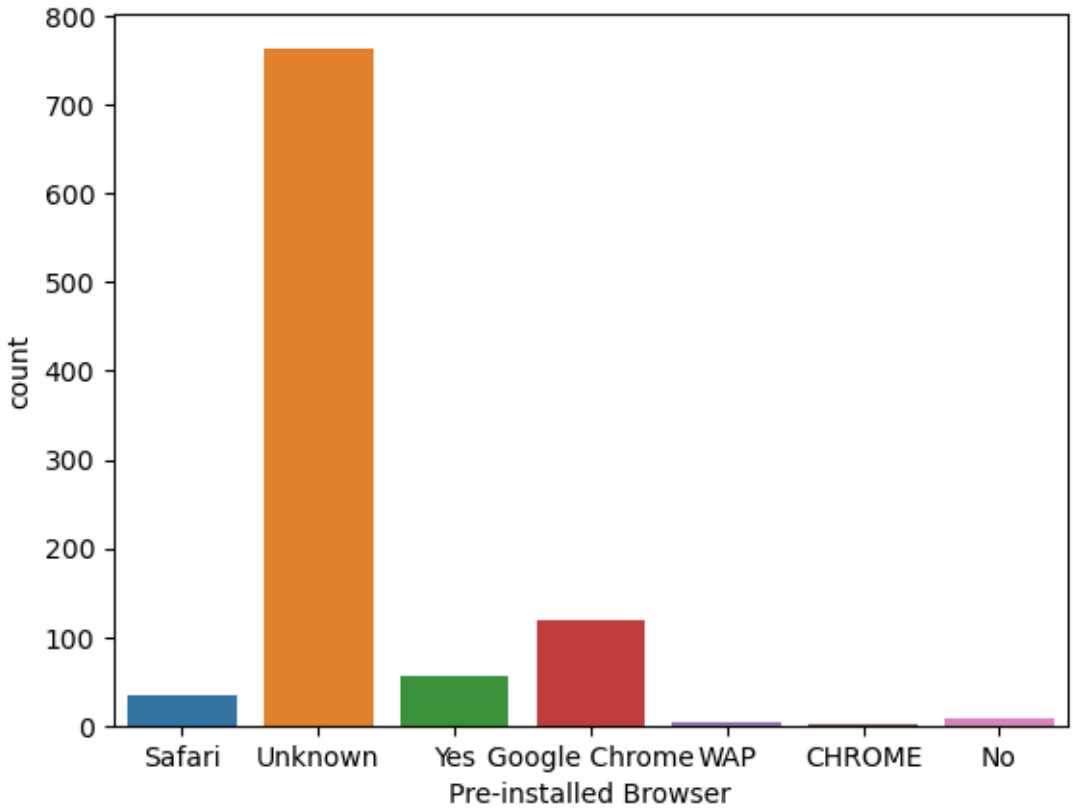
[54]:
```
df['Pre-installed Browser'].unique()
```

[54]:
```
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)
```

[55]:
```
def segment_preinstall(browser):
    if pd.isna(browser):
        return 'Unknown'
    elif 'Safari' in browser:
        return 'Safari'
    elif 'Google Chrome' in browser:
        return 'Google Chrome'
    elif 'Samsung Internet' in browser:
        return 'Samsung Internet'
    elif 'CHROME' in browser:
        return 'CHROME'
    elif 'Internet' in browser:
        return 'Internet'
    elif 'Samsung S-Browser' in browser:
        return 'Samsung S-Browser'
    elif 'WAP' in browser:
        return 'WAP'
    elif 'No' in browser:
        return 'No'
    elif 'Yes' in browser:
        return 'Yes'
    else:
        return 'Other'
df['Pre-installed Browser']=df['Pre-installed Browser'].
 ↪apply(segment_preinstall)
```

[56]:
```
return_countplot('Pre-installed Browser')
```

[56]:
```
<AxesSubplot:xlabel='Pre-installed Browser', ylabel='count'>
```



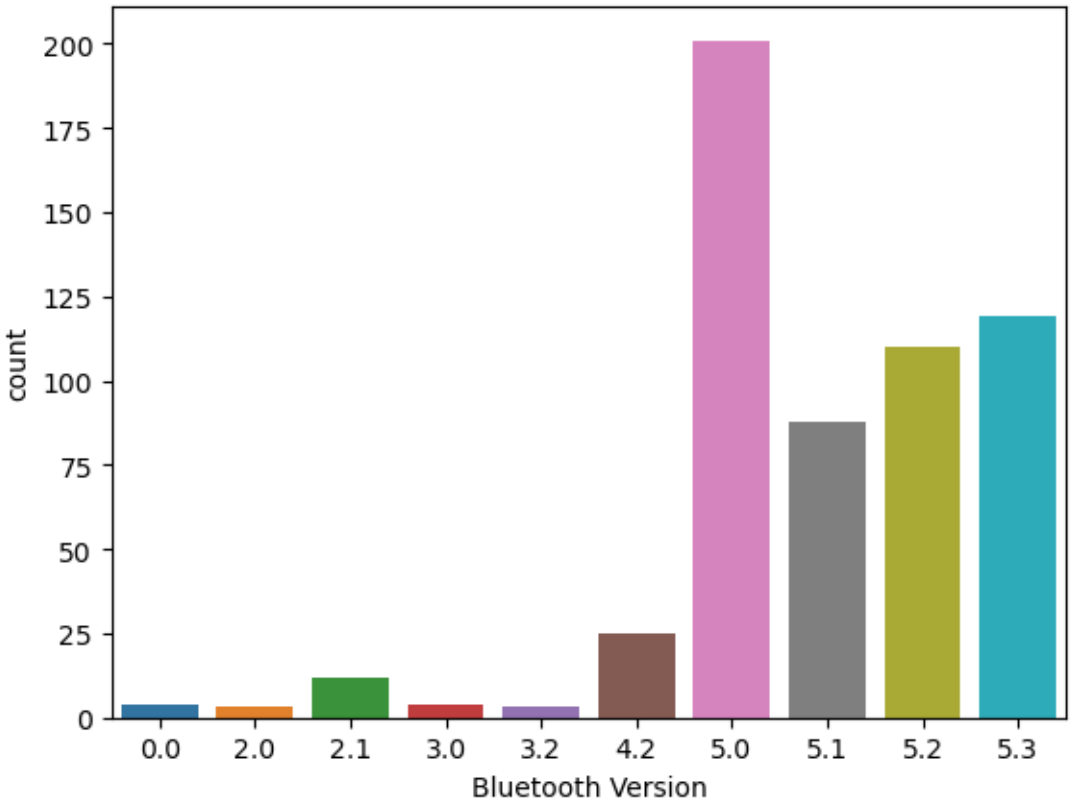
[57]:
```
df['Bluetooth Version'].unique()
```

[57]:
```
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)
```

[58]:
```
df['Bluetooth Version'] = df['Bluetooth Version'].str.replace('[^0-9.
  ↪]','',regex=True)
df['Bluetooth Version'] =pd.to_numeric(df['Bluetooth Version'])
```

[59]:
```
return_countplot('Bluetooth Version')
```

[59]:
```
<AxesSubplot:xlabel='Bluetooth Version', ylabel='count'>
```



[60]:
```
df['SIM Size'].unique()
```

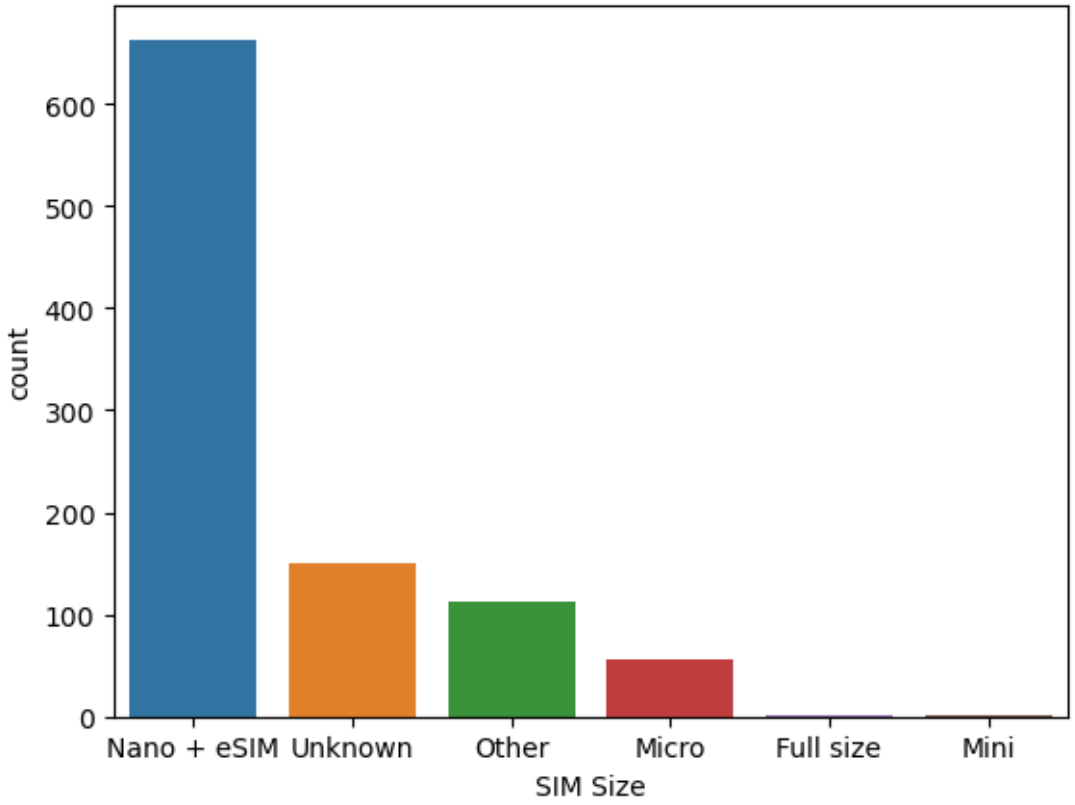
[60]:
```
array(['Nano + eSIM', 'Nano Sim', nan, 'Mini Sim', 'Nano Sim + eSIM',
       'Nano SIM and eSIM', 'Full Size SIM', 'Standard+ Micro', 'Micro',
       'Nano', 'Mini SIM', 'Nano-SIM', 'Normal', 'Full Size Sim',
       'Micro SIM', 'Neno-SIM card', 'NORMAL', 'Micro SIM + Micro SIM',
       'Normal 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)
```

[61]:
```
def segment_sim_size(size):
    if pd.isna(size):
        return 'Unknown'
    elif 'Nano' in size or 'eSIM' in size:
        return 'Nano + eSIM'
    elif 'Micro' in size or 'Micro SIM' 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'
df['SIM Size'] = df['SIM Size'].apply(segment_sim_size)
```

[62]:
```
return_countplot('SIM Size')
```

[62]:
```
<AxesSubplot:xlabel='SIM Size', ylabel='count'>
```



[63]:
```
df['Browser'].unique()
```

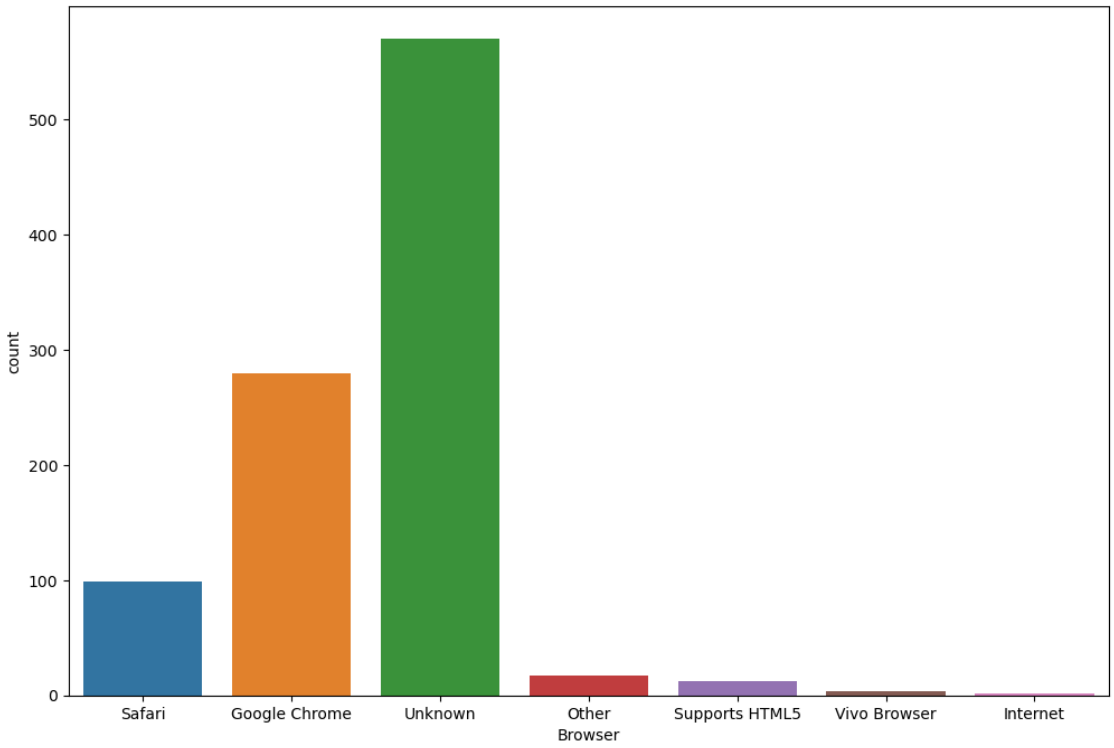
[63]:
```
array(['Safari', 'Google Chrome', nan,
       'Google Chrome, Samsung S Browser 14.0',
       'Google Chrome | Samsung S-Browser 18.0', 'NO',
       'Google Chrome, Samsung S-Browser 13.2', 'Supports HTML5', 'WAP',
```

```
        '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)
```

[64]:
```
def segment_browser(browser):
    if pd.isna(browser):
        return 'Unknown'
    elif 'Safari' in browser:
        return 'Safari'
    elif 'Google Chrome' in browser:
        return 'Google Chrome'
    elif 'Samsung Internet' in browser:
        return 'Samsung Internet'
    elif 'Samsung S-Browser' in browser:
        return 'Samsung S-Browser'
    elif 'Vivo Browser' in browser:
        return 'Vivo Browser'
    elif 'Supports HTML5' in browser:
        return 'Supports HTML5'
    elif 'Internet' in browser:
        return 'Internet'
    else:
        return 'Other'

df['Browser'] = df['Browser'].apply(segment_browser)
```

[65]:
```
plt.figure(figsize=(12,8))
return_countplot('Browser')
```

[65]:
```
<AxesSubplot:xlabel='Browser', ylabel='count'>
```



[66]:
```
df['Map Support'].unique()
```

[66]:
```
array(['Maps', 'Google Maps', nan, 'Yes',
       'Google Maps and Other Third Party Map Apps Supported',
       'Google maps', 'NO', 'YES', 'no', 'Google Maps (Go Edition)',
       'Google Map', 'No'], dtype=object)
```

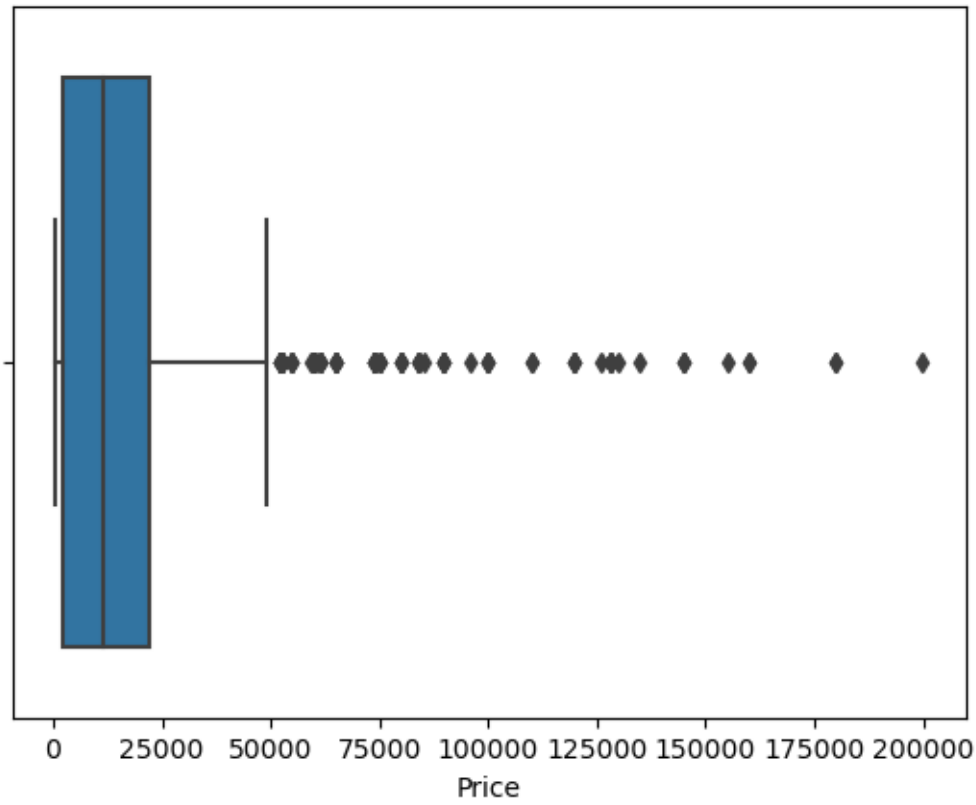
[67]:
```
cat_vars = df.select_dtypes(include='object').columns.tolist()
```

[68]:
```
def boxplot(column):
    sns.boxplot(x=column,data=df)
```

[69]:
```
df.select_dtypes(include = ['int','float']).columns
```

[69]:
```
Index(['Price', 'Rating', 'No_of_Ratings', 'No_of_Reviews', 'Display_size_cm',
       'Display_size_inches', 'Bluetooth Version', 'Graphics PPI',
       'Battery Capacity', 'Width', 'Height', 'Depth', 'Weight'],
      dtype='object')
```

[70]:
```
boxplot('Price')
```



[71]:
```
# Data Preprocessing
```

[72]:
```
df.drop(columns = ['Browser','Pre-installed Browser','Frame Rate','Resolution␣
↪Type'],inplace=True)
df.head()
```

[72]:

	Brand	Price	Rating	No_of_Ratings	No_of_Reviews	Browse Type	\
0	Apple	52499	4.7	259109.0	12745.0	Smartphones	
1	Xiaomi	6499	4.1	78642.0	4449.0	Smartphones	
2	Oneplus	17196	4.4	110949.0	7728.0	Smartphones	
3	realme	15999	4.3	2633.0	220.0	Smartphones	
4	realme	14999	4.4	13807.0	988.0	Smartphones	

	SIM Type	Hybrid Sim Slot	Touchscreen	OTG Compatible	…	Smartphone	\
0	Dual Sim	No	Yes	No	…	Yes	
1	Dual Sim	No	Yes	Yes	…	Yes	
2	Dual Sim	No	Yes	No	…	Yes	
3	Dual Sim	No	Yes	Yes	…	Yes	
4	Dual Sim	No	Yes	Yes	…	Yes	

	SIM Size	Mobile Tracker	Removable Battery	Graphics PPI	\

```
0  Nano + eSIM            Yes                No        460.0
1  Nano + eSIM            NaN                No          NaN
2  Nano + eSIM            NaN                No          NaN
3  Nano + eSIM            NaN               NaN        391.0
4  Nano + eSIM            NaN               NaN        391.0

  Battery Capacity  Width Height Depth Weight
0             3240  71.50  146.7  7.65  173.0
1             5000  76.75  164.9  9.09  192.0
2             5000  76.75  164.9  9.09  192.0
3             5000  76.00  165.7  7.89  190.0
4             5000  76.00  165.7  7.89  190.0

[5 rows x 45 columns]
```

[73]:
```
df.shape
```

[73]:
```
(984, 45)
```

[74]:
```
check_missing = df.isnull().sum() *100 /df.shape[0]
check_missing[check_missing>0].sort_values(ascending = False)
```

[74]:
```
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
Secondary Camera             37.296748
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
No_of_Reviews                 2.439024
No_of_Ratings                 2.439024
OTG Compatible                0.508130
Hybrid Sim Slot               0.508130
Touchscreen                   0.304878
dtype: float64
```

[75]:
```
# remove all of the column where the null value > 20%
columns_to_remove = check_missing[check_missing > 20].index
columns_to_remove
```

[75]:
```
Index(['Quick Charging', 'Processor Core', 'Primary Camera Available',
       'Secondary Camera Available', 'Secondary Camera', 'HD Recording',
       'Full HD Recording', 'Video Recording', 'Dual Camera Lens', '3G',
       'Bluetooth Support', 'Bluetooth Version', 'Wi-Fi', 'Wi-Fi Hotspot',
       'NFC', 'EDGE', 'Map Support', 'GPS Support', 'Mobile Tracker',
       'Removable Battery', 'Graphics PPI'],
      dtype='object')
```

[76]:
```
df = df.drop(columns =columns_to_remove )
```

[77]:
```
df.head()
```

[77]:

	Brand	Price	Rating	No_of_Ratings	No_of_Reviews	Browse Type	\
0	Apple	52499	4.7	259109.0	12745.0	Smartphones	
1	Xiaomi	6499	4.1	78642.0	4449.0	Smartphones	
2	Oneplus	17196	4.4	110949.0	7728.0	Smartphones	
3	realme	15999	4.3	2633.0	220.0	Smartphones	
4	realme	14999	4.4	13807.0	988.0	Smartphones	

	SIM Type	Hybrid Sim Slot	Touchscreen	OTG Compatible	…	Network Type	\
0	Dual Sim	No	Yes	No	…	5G	
1	Dual Sim	No	Yes	Yes	…	4G VOLTE	
2	Dual Sim	No	Yes	No	…	5G	
3	Dual Sim	No	Yes	Yes	…	5G	
4	Dual Sim	No	Yes	Yes	…	5G	

	Supported Networks	Internet Connectivity	Smartphone	SIM Size	\
0	5G	5G	Yes	Nano + eSIM	
1	4G	4G	Yes	Nano + eSIM	
2	5G	Unknown	Yes	Nano + eSIM	
3	5G	5G	Yes	Nano + eSIM	
4	5G	5G	Yes	Nano + eSIM	

	Battery Capacity	Width	Height	Depth	Weight
0	3240	71.50	146.7	7.65	173.0

```
1             5000  76.75  164.9  9.09   192.0
2             5000  76.75  164.9  9.09   192.0
3             5000  76.00  165.7  7.89   190.0
4             5000  76.00  165.7  7.89   190.0

[5 rows x 24 columns]
```

[78]:
```
df.shape
```

[78]:
```
(984, 24)
```

[79]:
```
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)
```

[80]:
```
df.dropna(inplace=True)
df.shape
```

[80]:
```
(926, 24)
```

2 Label Encoding for Object Datatype

[81]:
```
# loop over each column in dataframe where dtype is object
for col in df.select_dtypes(include = 'object').columns:

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

```
Brand:['Apple' 'Xiaomi' 'Oneplus' 'realme' 'vivo' 'other' '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']
Operating System:['iOS' 'Android' 'Unknown' 'Others' 'Symbian' 'RTOS' 'Belle']
Internal Storage:['GB' 'MB' 'Unknown' 'others' 'TB']
Network Type:['5G' '4G VOLTE' '4G' '2G' '3G']
Supported Networks:['5G' '4G' '2G' 'CDMA']
Internet Connectivity:['5G' '4G' 'Unknown' '2G' 'Other' 'GPRS']
Smartphone:['Yes' 'No']
SIM Size:['Nano + eSIM' 'Unknown' 'Other' 'Micro' 'Full size' 'Mini']
```

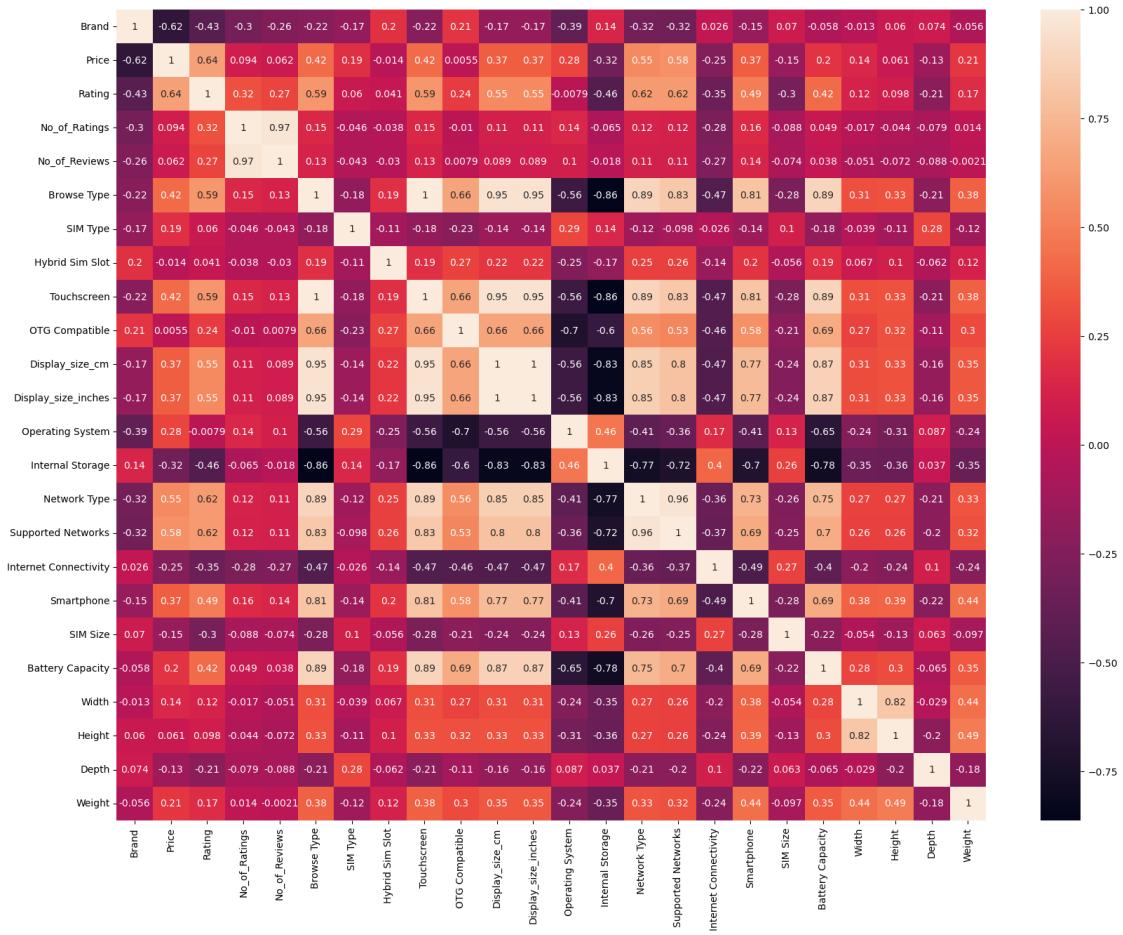
[82]:
```
from sklearn import preprocessing
```

```
for col in df.select_dtypes(include='object').columns:

    label_encoder = preprocessing.LabelEncoder()

    label_encoder.fit(df[col].unique())

    df[col] = label_encoder.transform(df[col])


    print(f"{col}:{df[col].unique()}")
```

```
Brand:[0 3 1 5 6 4 2]
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]
Operating System:[6 0 5 2 4 3 1]
Internal Storage:[0 1 3 4 2]
Network Type:[4 3 2 0 1]
Supported Networks:[2 1 0 3]
Internet Connectivity:[2 1 5 0 4 3]
Smartphone:[1 0]
SIM Size:[3 5 4 1 0 2]
```

[83]:
```
# correlation heatmap
plt.figure(figsize=(20,15))
sns.heatmap(df.corr(),annot=True)
```

[83]: <AxesSubplot:>



[84]:
```
df.drop(columns = ['No_of_Ratings','Browse Type','Display_size_inches'],inplace
↪= True)

df.head()
```

[84]:

	Brand	Price	Rating	No_of_Reviews	SIM Type	Hybrid Sim Slot	\
0	0	52499	4.7	12745.0	0	0	
1	3	6499	4.1	4449.0	0	0	
2	1	17196	4.4	7728.0	0	0	
3	5	15999	4.3	220.0	0	0	
4	5	14999	4.4	988.0	0	0	

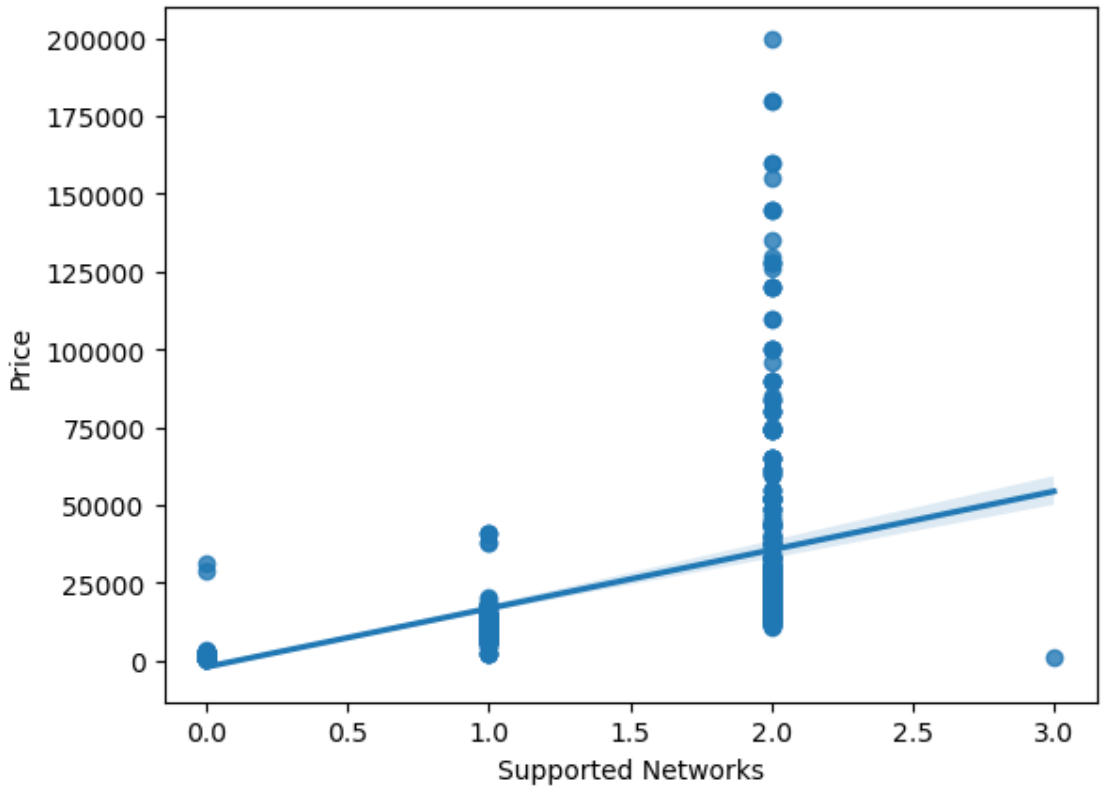
	Touchscreen	OTG Compatible	Display_size_cm	Operating System	…	\
0	1	2	15.49	6	…	
1	1	3	16.56	0	…	
2	1	2	16.74	0	…	
3	1	3	17.07	0	…	
4	1	3	17.07	0	…	

```
   Network Type  Supported Networks  Internet Connectivity  Smartphone  \
0             4                   2                      2           1
1             3                   1                      1           1
2             4                   2                      5           1
3             4                   2                      2           1
4             4                   2                      2           1

   SIM Size  Battery Capacity  Width  Height  Depth  Weight
0         3              3240  71.50   146.7   7.65   173.0
1         3              5000  76.75   164.9   9.09   192.0
2         3              5000  76.75   164.9   9.09   192.0
3         3              5000  76.00   165.7   7.89   190.0
4         3              5000  76.00   165.7   7.89   190.0

[5 rows x 21 columns]
```

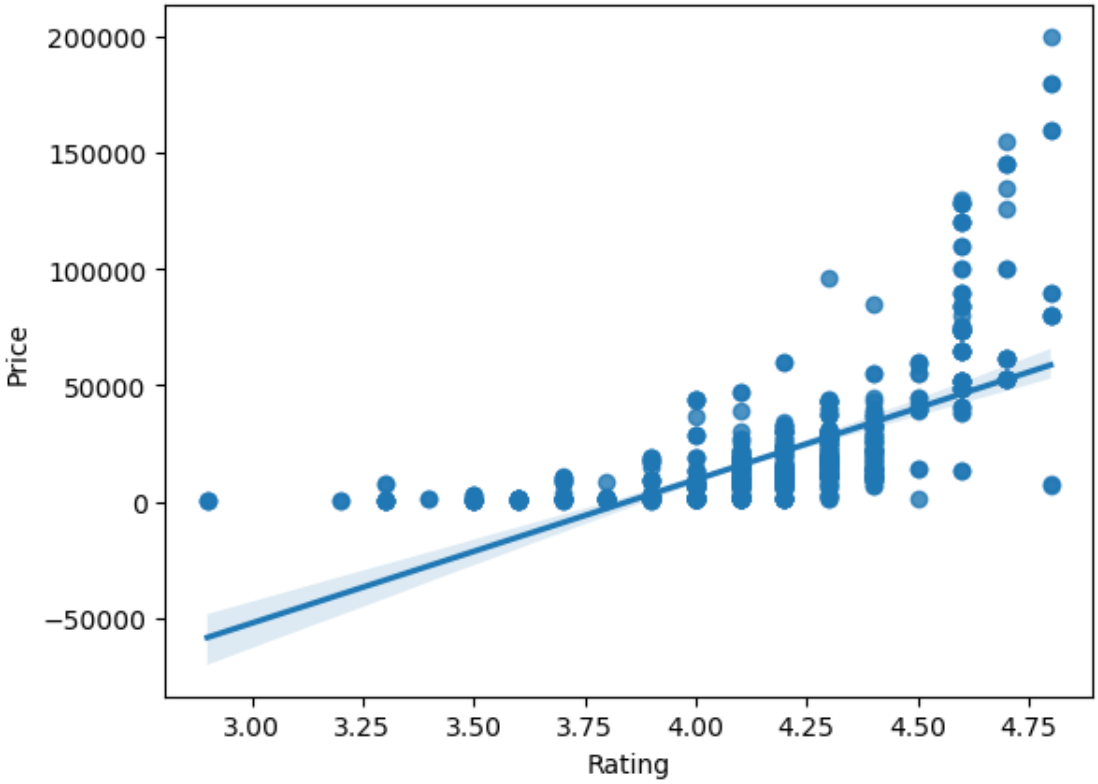
[91]:
```
def rel_plot(col_name,row_name):
    sns.regplot(x=col_name,y=row_name,data=df)
```

[90]:
```
sns.regplot(x='Supported Networks',y='Price',data=df)
```

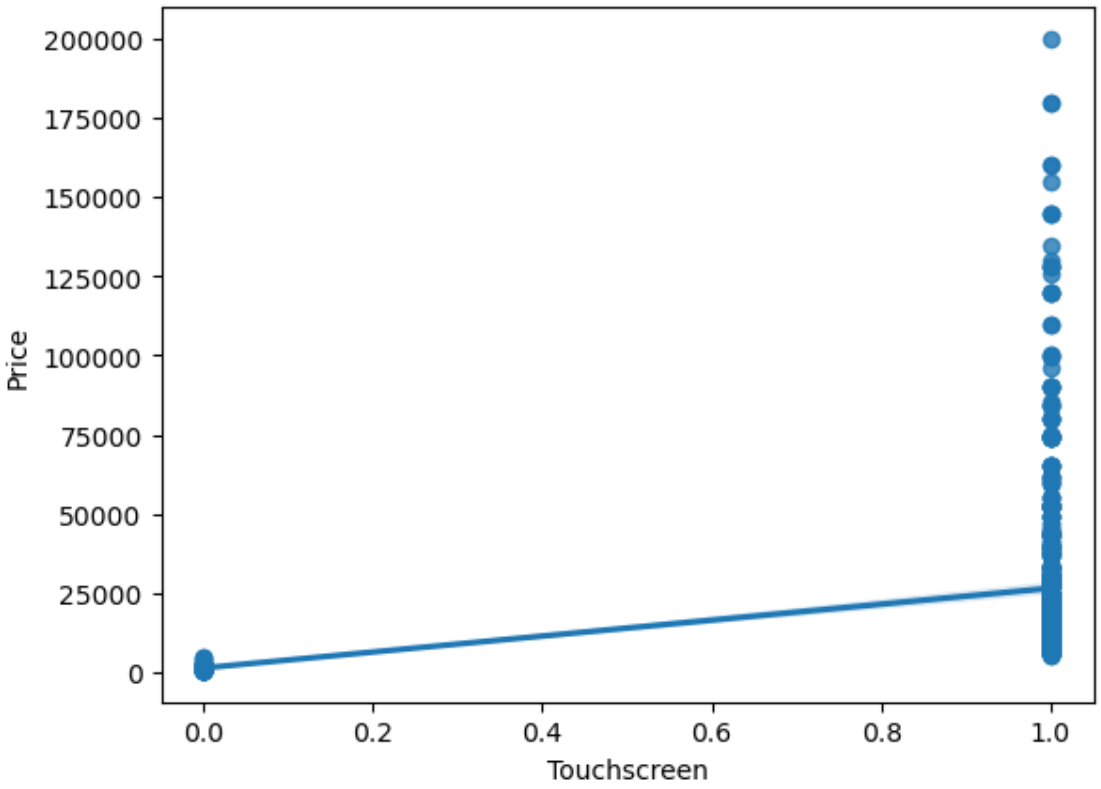
[90]:
```
<AxesSubplot:xlabel='Supported Networks', ylabel='Price'>
```



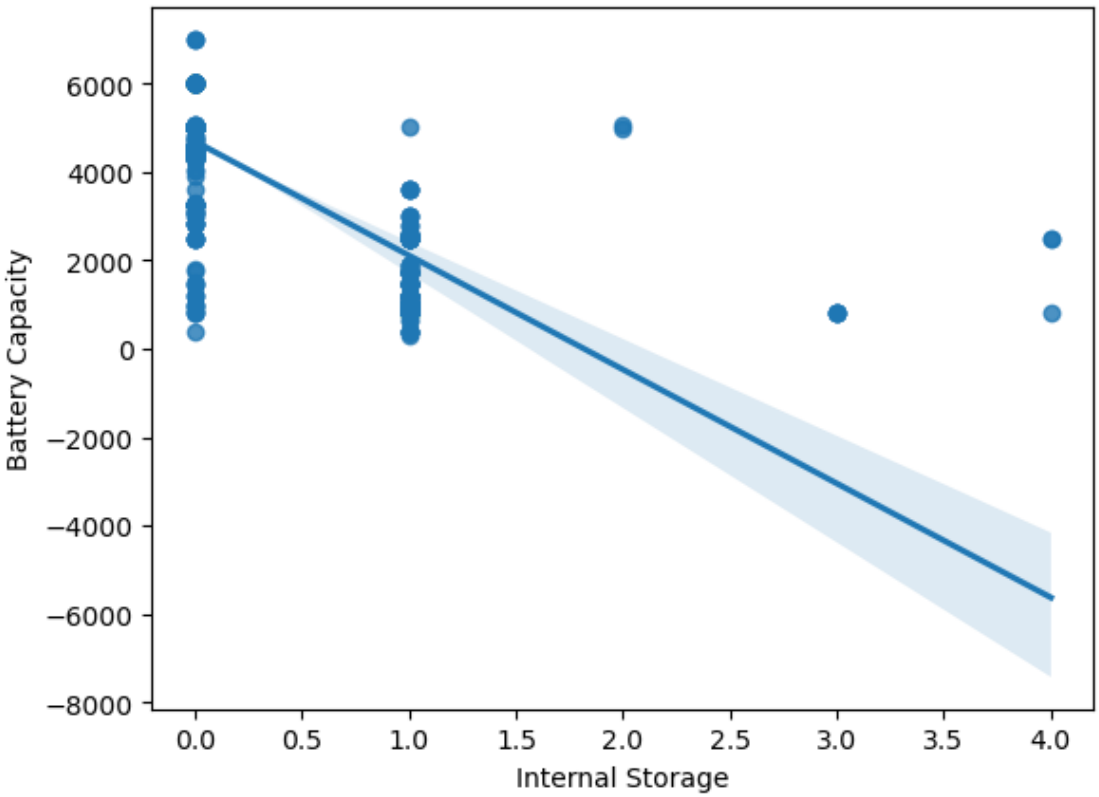
[92]:
```
rel_plot('Rating','Price')
```



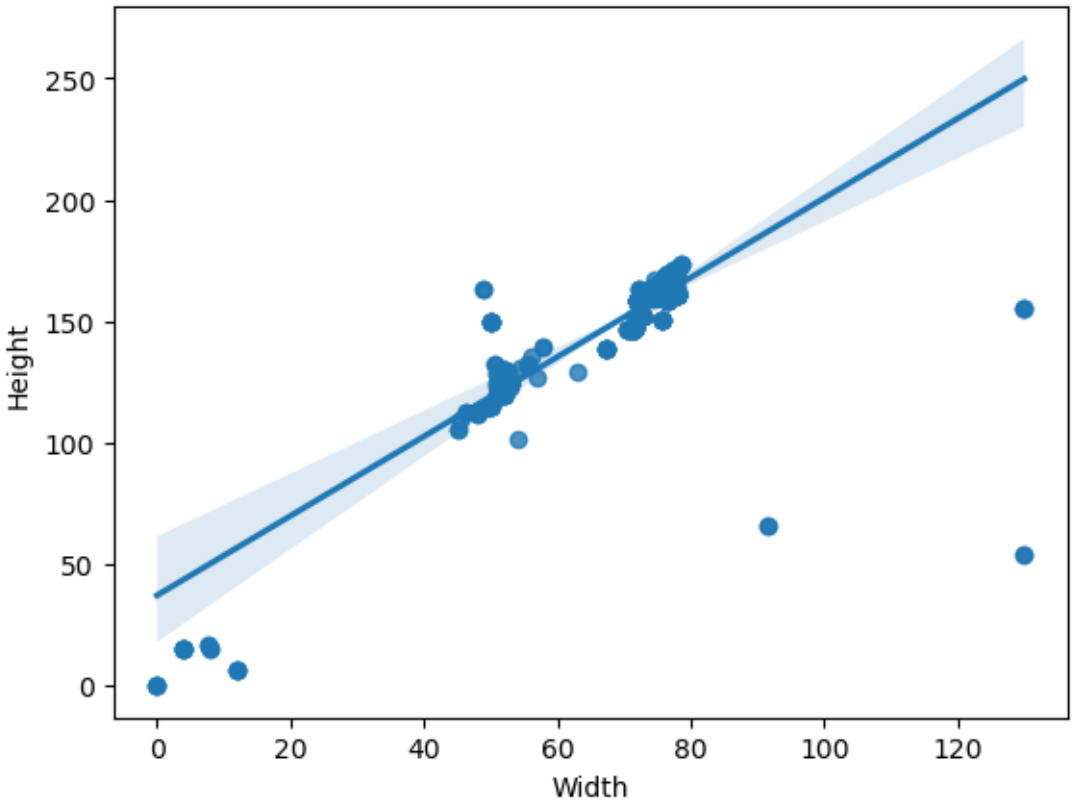
[93]:
```
rel_plot('Touchscreen','Price')
```



[95]:
```
rel_plot('Internal Storage','Battery Capacity')
```



[96]:
```
rel_plot('Width','Height')
```



[97]:
```
x = df.drop('Price',axis=1)
y = df['Price']

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train,X_test,Y_train,Y_test = train_test_split(x,y,test_size=0.
 ↪2,random_state=0)
```

3 Remove Outlier from Train Data using Z-score

[100]:
```
from scipy import stats

columns_to_keep_outliers = ['Rating','Bluetooth Version']

threshold = 3

numerical_columns = X_train.select_dtypes(include=['int','float']).columns

for col in numerical_columns:
    if col not in columns_to_keep_outliers:
```

```
        z_scores = np.abs(stats.zscore(X_train[col]))
        outlier_indices = np.where(z_scores > threshold)[0]
        X_train = X_train.drop(X_train.index[outlier_indices])
        Y_train = Y_train.drop(Y_train.index[outlier_indices])
```

4 Decision Tree Regressor

[101]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV


dtree = DecisionTreeRegressor()


param_grid = {
    'max_depth':[2,4,6,8],
    'min_samples_split':[2,4,6,8],
    'min_samples_leaf' :[1,2,3,4],
    'max_features':['auto','sqrt','log2'],
    'random_state':[0,42]
}


grid_search  =GridSearchCV(dtree,param_grid,cv=5, scoring =␣
 ↪'neg_mean_squared_error')

grid_search.fit(X_train,Y_train)


print(grid_search.best_params_)
```

```
{'max_depth': 8, 'max_features': 'auto', 'min_samples_leaf': 1,
'min_samples_split': 6, 'random_state': 0}
```

[102]:
```
dtree =␣
 ↪DecisionTreeRegressor(random_state=0,max_depth=8,max_features='auto',min_samples_split=6,mi

dtree.fit(X_train,Y_train)
```

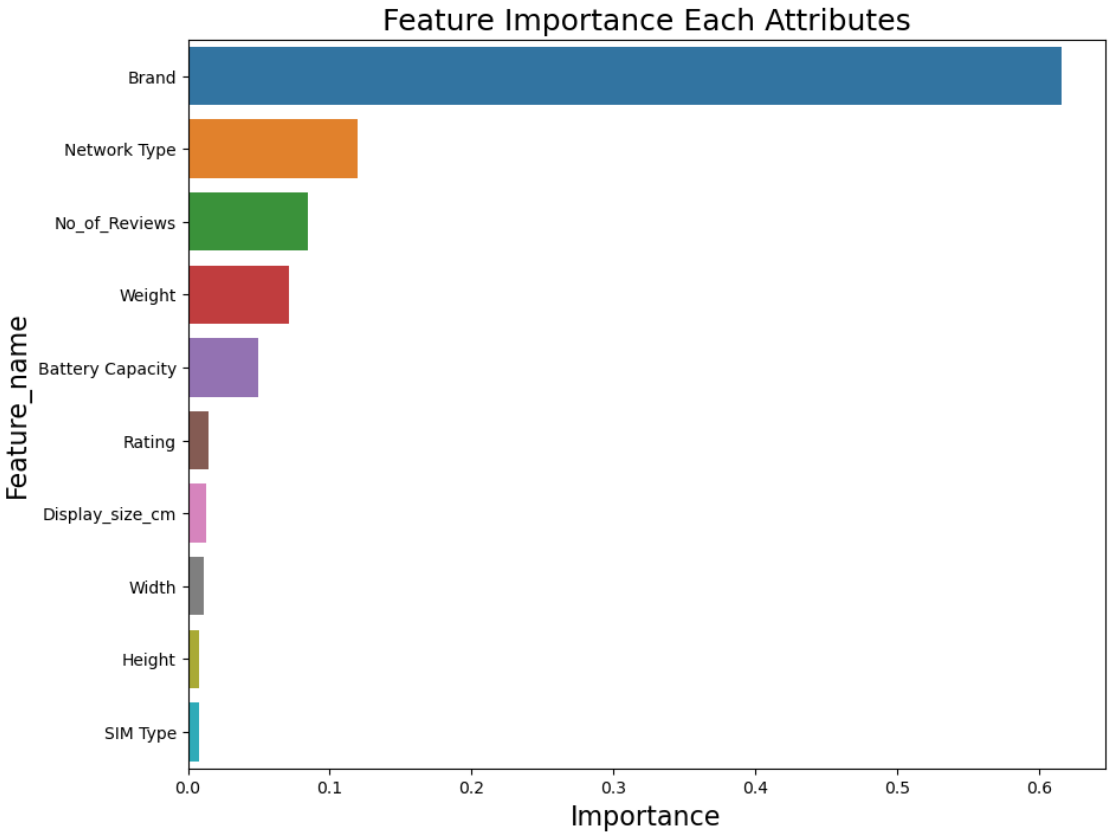
[102]:
```
DecisionTreeRegressor(max_depth=8, max_features='auto', min_samples_split=6,
                      random_state=0)
```

[103]:
```
from sklearn import metrics
from sklearn.metrics import mean_absolute_percentage_error
import math
```

```
y_pred = dtree.predict(X_test)

mae  = metrics.mean_absolute_error(Y_test,y_pred)

mape = metrics.mean_absolute_percentage_error(Y_test,y_pred)

mse = metrics.mean_squared_error(Y_test,y_pred)

r2 = metrics.r2_score(Y_test,y_pred)

rmse = math.sqrt(mse)


print('MAE is {}'.format(mae))
print('MAPE is {}'.format(mape))
print('MSE is {}'.format(mse))
print('R2 is {}'.format(r2))
print('RMSE is {}'.format(rmse))
```

```
MAE is 3117.0552534446565
MAPE is 0.1902928007881226
MSE is 36138492.87802894
R2 is 0.959855465852626
RMSE is 6011.529994770794
```

[104]:
```
imp_df = pd.DataFrame({
    'Feature_name':X_train.columns,
    'Importance':dtree.feature_importances_

})

fi = imp_df.sort_values(by='Importance',ascending = False)

fi2 = fi.head(10)
plt.figure(figsize=(10,8))
sns.barplot(data=fi2,x='Importance',y='Feature_name')
plt.title('Feature Importance Each Attributes',fontsize=18)
plt.xlabel('Importance',fontsize=16)
plt.ylabel('Feature_name',fontsize=16)
plt.show()
```


Feature Importance Each Attributes
Brand
Network Type
No_of_Reviews
Weight
Battery Capacity
Rating
Display_size_cm
Width
Height
SIM Type
Feature_name
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Importance

[ ]: