Telecommunication Industry Project
¶

Introduction

This Jupyter notebook is part of your learning experience in the study of applied statistics.

You will work with a data set that contains mobile phone prices and their specifications.

Dataset Columns Information

PID = a unique identifier for the phone model

Blue = whether the phone has bluetooth support or not

Wi_Fi = whether the phone has wifi support or not

Tch_Scr = whether the phone has touch screen support or not

Ext_Mem = whether the phone has external memory support or not

Px_h = number of pixels in the vertical axis of the phone

Px_w = number of pixels in the horizontal axis of the phone

Scr_h = height of the screen of the phone in centimetres (cm)

Scr_w = width of the screen of the phone in centimetres (cm)

Int_Mem = internal memory of the phone measured in megabytes (MB)

Bty_Pwr = maximum energy stored by the phone's battery measured in milli-Ampere-hours (mAh)

PC = resolution of the primary camera measued in megapixels (MP)

FC = resolution of the front camera measued in megapixels (MP)

RAM = random access memory available in the phone measured in gigabytes (GB)

Depth = depth of the mobile phone measured in centimetres (cm)

Weight = weight of the mobile phone measured in grams (g)

Price = selling price of the mobile phone in rupees

In [ ]:

Task 1 - Load and study the data

Import the libraries that will be used in this notebook

In [1]:

```
# Load "numpy" and "pandas" for manipulating numbers and data frames
# Load "matplotlib.pyplot" and "seaborn" for data visualisation

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

Load the csv file as pandas dataframe.

In [2]:

```
# Read in the "Dataset" file as a Pandas Data Frame

d= pd.read_csv('D:\Twilearnass1\Mobile_Phones.csv')
```

In [3]:

```
# Take a brief look at the data
top = d.head()
top
```

Out[3]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Mem
0	AAB346A	yes	yes	no	no	780	460	3	1	2	2	8
1	AAC347I	yes	yes	no	no	780	560	2	1	4	2	8
2	BAB657J	no	yes	no	no	840	720	2	1	4	2	8
3	BBD456K	no	yes	yes	no	1280	1120	5	3	6	2	32
4	CCP761U	no	yes	yes	no	1280	1080	4	3	6	2	16

In [15]:

```
# Get the dimensions of the dataframe
d.shape
```

Out[15]:

```
(50, 17)
```

In [18]:

```
# Get the row names of the dataframe# iterate the indices and print each one
for row in d.index:
    print(row, end = " ")
```

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
```

In [19]:

```
# Get the column names of the dataframe

d.columns
```

Out[19]:

```
Index(['PID', 'Blue', 'Wi_Fi', 'Tch_Scr', 'Ext_Mem', 'Px_h', 'Px_w', 'Scr_
h',
       'Scr_w', 'PC', 'FC', 'Int_Mem', 'Bty_Pwr', 'RAM', 'Depth', 'Weigh
t',
       'Price'],
      dtype='object')
```

In [20]:

```
# Look at basic information about the dataframe

d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   PID      50 non-null     object
 1   Blue     50 non-null     object
 2   Wi_Fi    50 non-null     object
 3   Tch_Scr  50 non-null     object
 4   Ext_Mem  50 non-null     object
 5   Px_h     50 non-null     int64
 6   Px_w     50 non-null     int64
 7   Scr_h    50 non-null     int64
 8   Scr_w    50 non-null     int64
 9   PC       50 non-null     int64
 10  FC       50 non-null     int64
 11  Int_Mem  50 non-null     int64
 12  Bty_Pwr  50 non-null     int64
 13  RAM      50 non-null     int64
 14  Depth    50 non-null     int64
 15  Weight   50 non-null     int64
 16  Price    50 non-null     int64
dtypes: int64(12), object(5)
memory usage: 6.8+ KB
```

Observations:

There are 50 phones in the data set.

There are 17 features in the data set including the "PID" feature which is used as the row index labels.

There are no missing values in the data set.

In [ ]:

```
```

Let's try some logical operators to filter the data.

Logical Operators

Operator	Result
&	Logical AND
|	Logical OR
^	Logical XOR (exclusive OR)
||	Short-circuit OR
&&	Short-circuit AND
!	Logical unary NOT
&=	AND assignment
|=	OR assignment
^=	XOR assignment
==	Equal to
!=	Not equal to
?:	Ternary if-then-else

Task 2 - Obtain the logical conditions for the features "Blue", "Wi_Fi", "Tch_Scr" and "Ext_Mem"

In [21]:

```
# Get the feature names of the dataframe

d.columns
```

Out[21]:

```
Index(['PID', 'Blue', 'Wi_Fi', 'Tch_Scr', 'Ext_Mem', 'Px_h', 'Px_w', 'Scr_
h',
       'Scr_w', 'PC', 'FC', 'Int_Mem', 'Bty_Pwr', 'RAM', 'Depth', 'Weigh
t',
       'Price'],
      dtype='object')
```

In [ ]:

```
# Let's tackle these features: "Blue", "Wi_Fi", "Tch_Scr", "Ext_Mem"
```

In [22]:

```
# The children want phones that have the following: Bluetooth, WiFi, touch screen and ex
# Create a logical condition for this situation and store the logical values as "con1"

con1 = d[['Blue','Wi_Fi','Tch_Scr','Ext_Mem']]
con1
```

Out[22]:

	Blue	Wi_Fi	Tch_Scr	Ext_Mem
0	yes	yes	no	no
1	yes	yes	no	no
2	no	yes	no	no
3	no	yes	yes	no
4	no	yes	yes	no
5	yes	no	no	no
6	yes	no	yes	no
7	yes	no	no	no
8	yes	yes	yes	yes
9	yes	yes	yes	yes
10	yes	yes	yes	yes
11	yes	yes	yes	yes
12	no	yes	yes	yes
13	no	yes	yes	no
14	yes	yes	yes	yes
15	no	no	yes	yes
16	no	no	yes	yes
17	no	no	yes	yes
18	no	no	yes	yes
19	no	no	yes	yes
20	no	yes	yes	yes
21	no	yes	yes	yes
22	no	yes	yes	yes
23	no	yes	yes	yes
24	no	yes	yes	yes
25	no	yes	yes	yes
26	no	yes	yes	yes
27	yes	yes	yes	yes
28	yes	yes	yes	yes
29	no	yes	yes	yes
30	yes	yes	yes	yes
31	no	yes	yes	yes
32	yes	yes	yes	yes
33	no	yes	yes	yes
34	yes	yes	yes	yes
35	no	yes	yes	yes
36	yes	yes	yes	yes

	Blue	Wi_Fi	Tch_Scr	Ext_Mem
37	yes	yes	yes	yes
38	yes	yes	yes	yes
39	yes	yes	yes	yes
40	yes	yes	yes	yes
41	yes	yes	yes	yes
42	yes	yes	yes	yes
43	no	yes	yes	yes
44	yes	yes	yes	yes
45	no	yes	yes	yes
46	yes	yes	yes	yes
47	yes	yes	yes	no
48	yes	yes	yes	no
49	yes	yes	yes	no

Observations:

The features "Blue", "Wi_Fi", "Tch_Scr" and "Ext_Mem" are binary in nature.

The children want all these features, so the logical condition "con1" has been obtained accordingly.

Task 3 - Obtain the logical conditions for the features "Px_h" and "Px_w"

In [23]:

```
# Get the feature names of the dataframe
d.columns
```

Out[23]:

```
Index(['PID', 'Blue', 'Wi_Fi', 'Tch_Scr', 'Ext_Mem', 'Px_h', 'Px_w', 'Scr_
h',
       'Scr_w', 'PC', 'FC', 'Int_Mem', 'Bty_Pwr', 'RAM', 'Depth', 'Weigh
t',
       'Price'],
      dtype='object')
```

In [ ]:

```
# Let's tackle these features: "Px_h", "Px_w"
```

In [24]:

```
# Create a new feature called "Px" which stores the total resolution of the screen

Px = (d['Px_h']) & (d['Px_w'])
Pxx = pd.DataFrame(Px)
d['Pxx'] = Pxx
```

In [25]:

```
d['Pxx'].mean()
```

Out[25]:

```
1212.08
```

In [26]:

```
med = d['Pxx'].median()
```

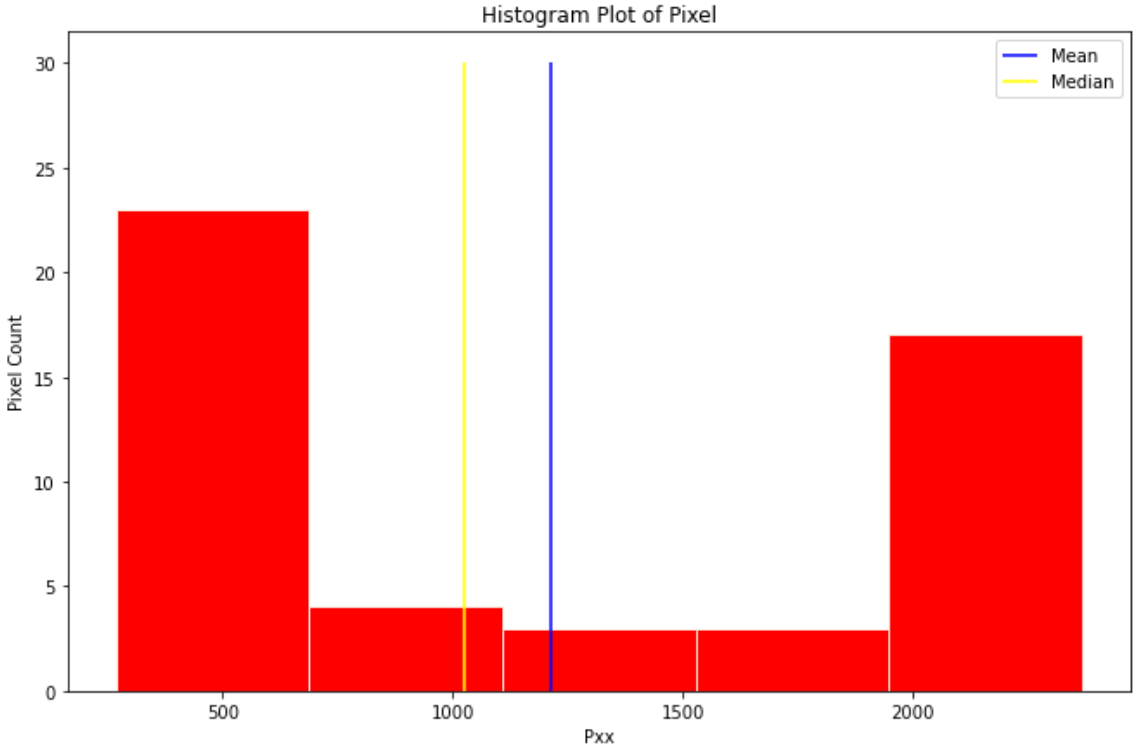
In [29]:

```
# Create a histogram of the "Px" feature and also show the mean and the median

plt.figure(figsize = (11,7))

sns.histplot(data = d , x = 'Pxx', color='red', edgecolor = 'linen', alpha = 1, bins=5)

plt.title("Histogram Plot of Pixel")
plt.xlabel('Pxx')
plt.ylabel('Pixel Count')
plt.vlines(d['Pxx'].mean(),ymin=0,ymax=30,colors='blue',label='Mean')
plt.vlines(d['Pxx'].median(),ymin=0,ymax=30,colors='yellow',label='Median')
plt.legend()
plt.show()
```



In [30]:

```
# The children want phones that have good screen resolutions
# Consider the phones that have screen resolutions greater than or equal to the median v
# Create a logical condition for this situation and store the logical values as "con2"

con2 = d['Pxx'][(d['Pxx']>=med)]
con2
```

Out[30]:

```
3     1024
4     1024
5     1024
6     1064
7     2112
13    2112
15    1696
17    2112
18    1696
19    2112
20    1696
22    1280
23    2112
24    1280
26    1280
27    2048
28    2048
29    2048
30    2048
31    2048
32    2048
33    2048
34    2112
35    2056
47    2368
48    2368
49    2368
Name: Pxx, dtype: int64
```

Observations:

The features "Px_h" and "Px_w" are respectively the number of pixels in the phone screen in the vertical and horizontal axes.

We created a new feature called "Px" which is the product of the features "Px_h" and "Px_w".

The median has been selected as a threshold in this case.

In case it is too strict, we can choose the mean as a threshold.

Task 4 - Obtain the logical conditions for the features "Scr_h" and "Scr_w"

In [ ]:

```
# Let's tackle these features: "Scr_h", "Scr_w"
```

In [33]:

```
# Create a new feature called "Scr_d" which stores the length of the diagonal of the scr

Scr_d = np.sqrt(d['Scr_h']**2 + d['Scr_w']**2)
Scr_d
d['Scr_d'] = Scr_d
```

In [34]:

```
Diagonal = d.Scr_d.values.tolist()
Diagonal
```

Out[34]:

```
[3.1622776601683795,
 2.23606797749979,
 2.23606797749979,
 5.830951894845301,
 5.0,
 5.0,
 6.708203932499369,
 10.0,
 6.708203932499369,
 5.830951894845301,
 6.708203932499369,
 10.0,
 7.810249675906654,
 7.810249675906654,
 5.830951894845301,
 12.806248474865697,
 10.0,
 10.0,
 10.0,
 5.0,
 5.0,
 10.0,
 5.0,
 12.806248474865697,
 7.211102550927978,
 6.708203932499369,
 5.830951894845301,
 10.0,
 10.0,
 10.0,
 10.0,
 10.0,
 10.0,
 10.0,
 6.708203932499369,
 5.830951894845301,
 12.806248474865697,
 5.830951894845301,
 5.830951894845301,
 7.810249675906654,
 5.830951894845301,
 10.0,
 10.0,
 10.0,
 12.806248474865697,
 10.0,
 5.0,
 10.0,
 10.0,
 10.0]
```

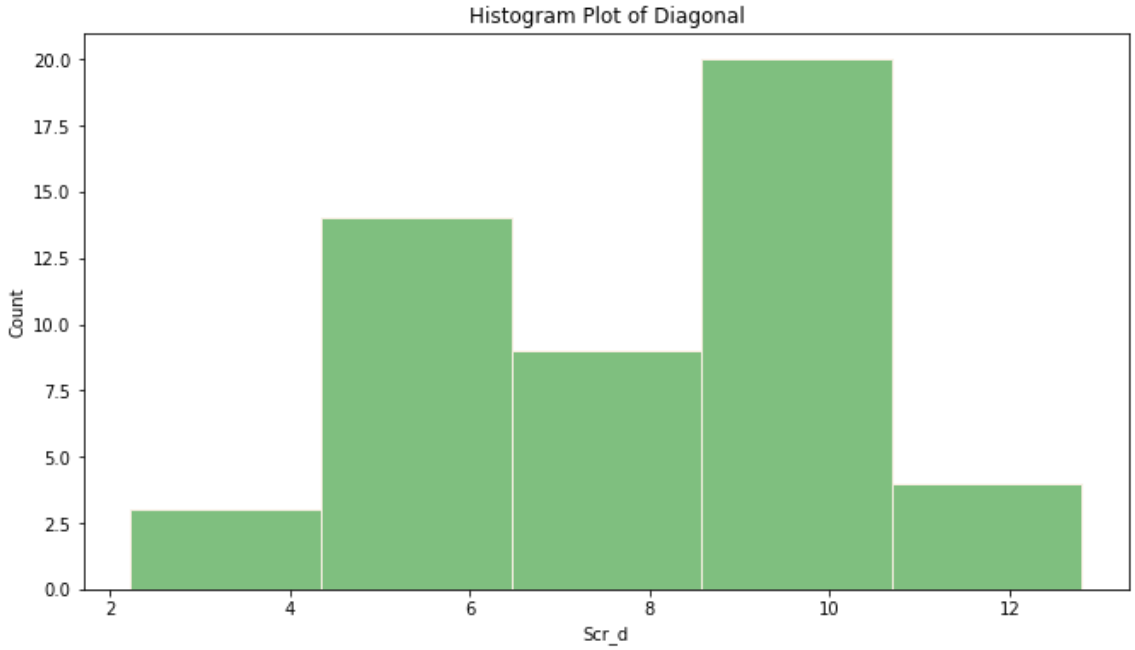
In [35]:

```
# Create a histogram of the "Scr_d" feature and also show the quartiles

plt.figure(figsize = (11,6))

sns.histplot(data = d , x = 'Scr_d', color='green', edgecolor = 'linen', alpha = 0.5, bi

plt.title("Histogram Plot of Diagonal")
plt.xlabel('Scr_d')
plt.ylabel('Count')
Q1 = np.percentile(Diagonal,25)
Q3 = np.percentile(Diagonal,75)
IQR = Q3 - Q1
low = Q1 - 1.5*IQR
upp = Q3 + 1.5*IQR
plt.show()
```



In [36]:

```
# The children want phones that have very good screen sizes
# Consider the phones that have screen sizes greater than or equal to the upper quartile
# Create a logical condition for this situation and store the logical values as "con3"
upp
con3 = [(d['Scr_d']>=upp)]
con3
```

Out[36]:

```
[0     False
 1     False
 2     False
 3     False
 4     False
 5     False
 6     False
 7     False
 8     False
 9     False
 10    False
 11    False
 12    False
 13    False
 14    False
 15    False
 16    False
 17    False
 18    False
 19    False
 20    False
 21    False
 22    False
 23    False
 24    False
 25    False
 26    False
 27    False
 28    False
 29    False
 30    False
 31    False
 32    False
 33    False
 34    False
 35    False
 36    False
 37    False
 38    False
 39    False
 40    False
 41    False
 42    False
 43    False
 44    False
 45    False
 46    False
 47    False
 48    False
 49    False
 Name: Scr_d, dtype: bool]
```

Observations:

The features "Scr_h" and "Scr_w" are respectively the height and the width of the phone screen.

We created a new feature called "Scr_d" which is essentially the length of the screen diagonal.

The upper quartile has been selected as a threshold in this case as the children were very particular on this point.

In case it is too strict, we can choose the mean or the median as a threshold

Task 5 - Obtain the logical conditions for the features "PC" and "FC"

In [ ]:

```
# Let's tackle these features: "PC", "FC"
```

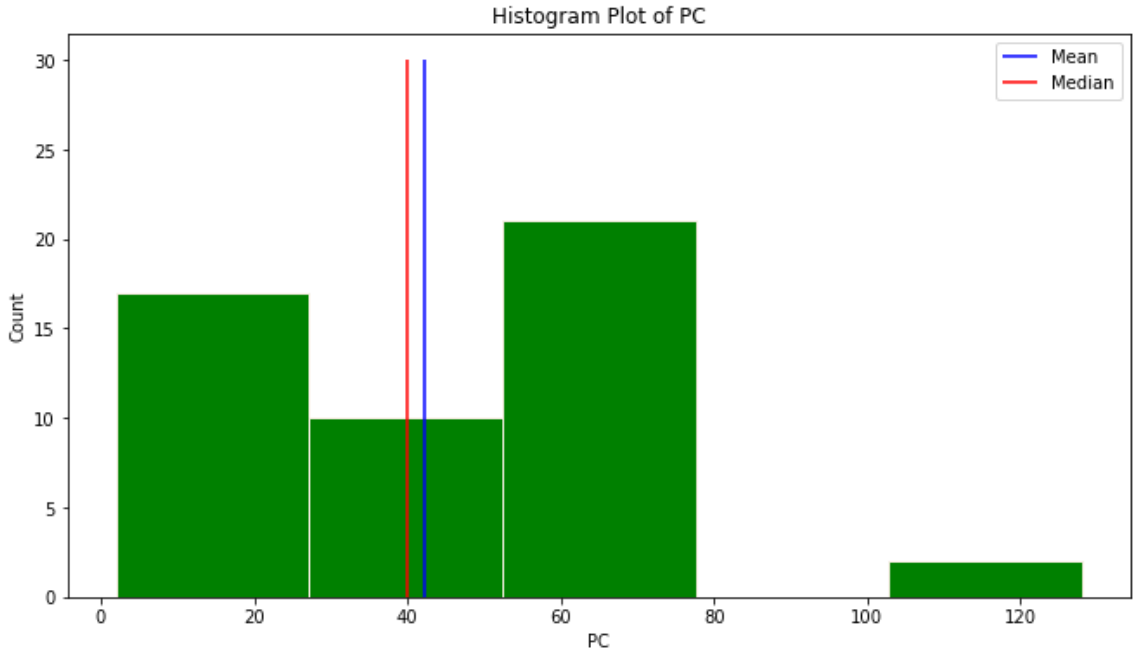
In [37]:

```
# Create a histogram of the "PC" feature and also show the mean and the median

plt.figure(figsize = (11,6))

sns.histplot(data = d , x = 'PC', color='green', edgecolor = 'linen', alpha = 1, bins=5)

plt.title("Histogram Plot of PC")
plt.xlabel('PC')
plt.ylabel('Count')
plt.vlines(d['PC'].mean(),ymin=0,ymax=30,colors='blue',label='Mean')
plt.vlines(d['PC'].median(),ymin=0,ymax=30,colors='red',label='Median')
plt.legend()
plt.show()
```



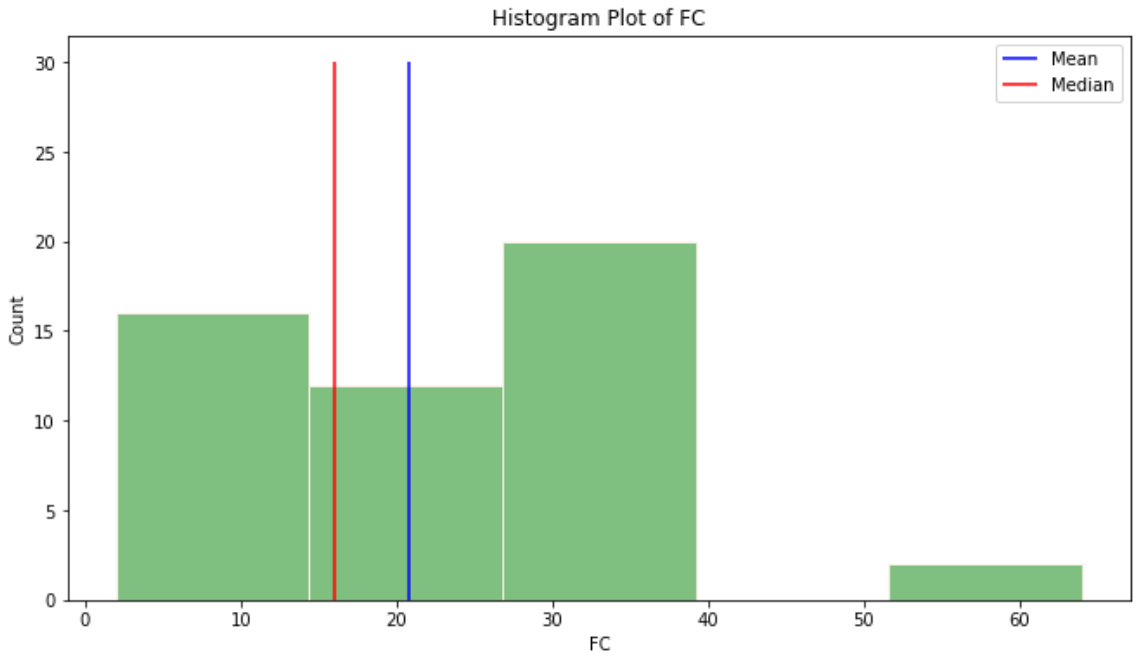
In [38]:

```
# Create a histogram of the "FC" feature and also show the mean and the median


plt.figure(figsize = (11,6))

sns.histplot(data = data , x = 'FC', color='green', edgecolor = 'linen', alpha = 0.5, bi

plt.title("Histogram Plot of FC")
plt.xlabel('FC')
plt.ylabel('Count')
plt.vlines(data['FC'].mean(),ymin=0,ymax=30,colors='blue',label='Mean')
plt.vlines(data['FC'].median(),ymin=0,ymax=30,colors='red',label='Median')
plt.legend()
plt.show()
```



In [39]:

```
pcm = d['PC'].mean()
pcm
```

Out[39]:

42.16

In [40]:

```
fcm = d['FC'].mean()
fcm
```

Out[40]:

20.76

In [41]:

```
# The children want phones that have good primary and front camera resolutions
# Consider the phones that have primary and front camera resolutions greater than or equ
# Create a logical condition for this situation and store the logical values as "con4"

con4 = d[(d['PC']>=pcm) & (d['FC']>=fcm)]
con4
```

Out[41]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Me
9	ENG897N	yes	yes	yes	yes	2580	1980	5	3	64	32	
11	ELS333L	yes	yes	yes	yes	2580	1920	8	6	64	32	
12	ETT987D	no	yes	yes	yes	2580	1980	6	5	64	32	
16	PDF768G	no	no	yes	yes	2580	1980	8	6	64	32	1
21	QWR222Y	no	yes	yes	yes	2580	1980	8	6	64	32	1
25	SDO555G	no	yes	yes	yes	2580	1980	6	3	64	32	
28	SSD000L	yes	yes	yes	yes	2580	2120	8	6	64	32	5
30	TVF078Y	yes	yes	yes	yes	2580	2120	8	6	64	32	5
32	TYS938L	yes	yes	yes	yes	2580	2120	8	6	64	32	10
33	TYU444Q	no	yes	yes	yes	2580	2120	8	6	64	32	1
34	TYY453J	yes	yes	yes	yes	2880	2120	6	3	64	32	1
36	UST000T	yes	yes	yes	yes	2580	1980	10	8	64	32	
37	USZ111S	yes	yes	yes	yes	2440	1980	5	3	48	32	1
38	VWV532Y	yes	yes	yes	yes	2580	1920	5	3	64	32	
40	WER765T	yes	yes	yes	yes	2580	1980	5	3	64	32	1
42	WZB298K	yes	yes	yes	yes	2580	1980	8	6	64	32	10
44	XTL675G	yes	yes	yes	yes	2580	1980	10	8	64	32	5
45	XXV567F	no	yes	yes	yes	2580	1980	8	6	64	32	
46	YTR67TY	yes	yes	yes	yes	2580	1980	4	3	64	32	
47	ZDF789K	yes	yes	yes	no	2880	2520	8	6	64	32	5
48	ZEO567M	yes	yes	yes	no	2880	2520	8	6	128	64	5
49	ZZZ909X	yes	yes	yes	no	2880	2520	8	6	128	64	10

Observations:

The features "PC" and "FC" are respectively the resolutions of the primary camera and the front camera.

The respective means have been selected as thresholds in this case.

In case it is too strict, we can choose the respective medians as thresholds.

Task 6 - Obtain the logical conditions for the features "Int_Mem", "Bty_Pwr" and "RAM"

In [ ]:

```
# Let's tackle these features: "Int_Mem", "Bty_Pwr", "RAM"
```

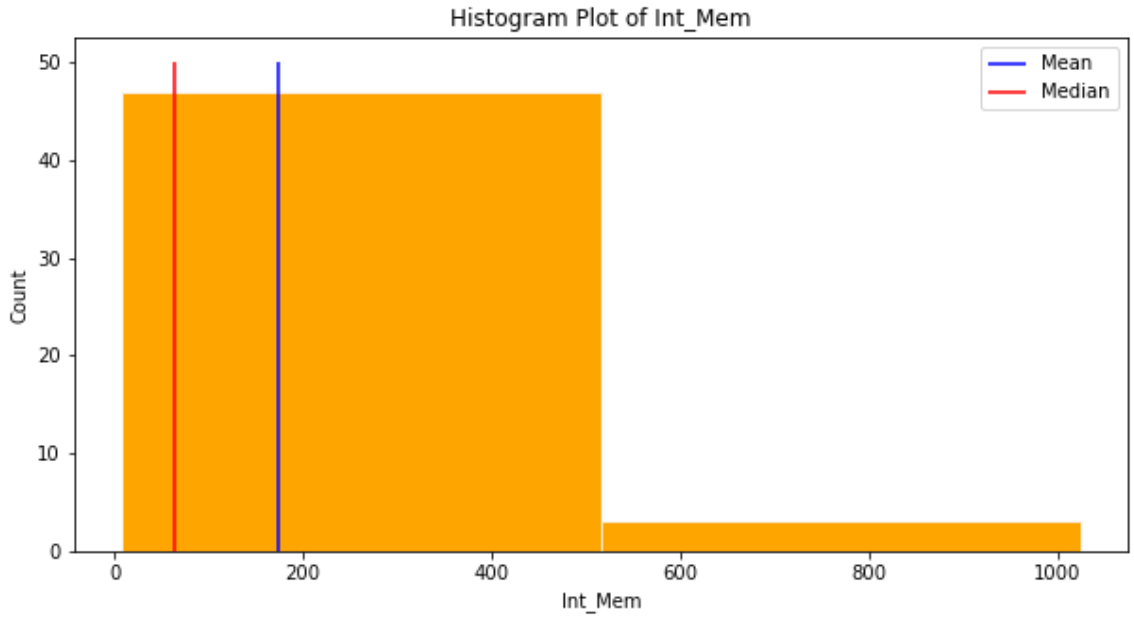
In [42]:

```
# Create a histogram of the "Int_Mem" feature and also show the mean and the median

plt.figure(figsize = (10,5))

sns.histplot(data = d , x = 'Int_Mem', color='orange', edgecolor = 'linen', alpha = 1, b

plt.title("Histogram Plot of Int_Mem")
plt.xlabel('Int_Mem')
plt.ylabel('Count')
plt.vlines(d['Int_Mem'].mean(),ymin=0,ymax=50,colors='blue',label='Mean')
plt.vlines(d['Int_Mem'].median(),ymin=0,ymax=50,colors='red',label='Median')
plt.legend()
plt.show()
```



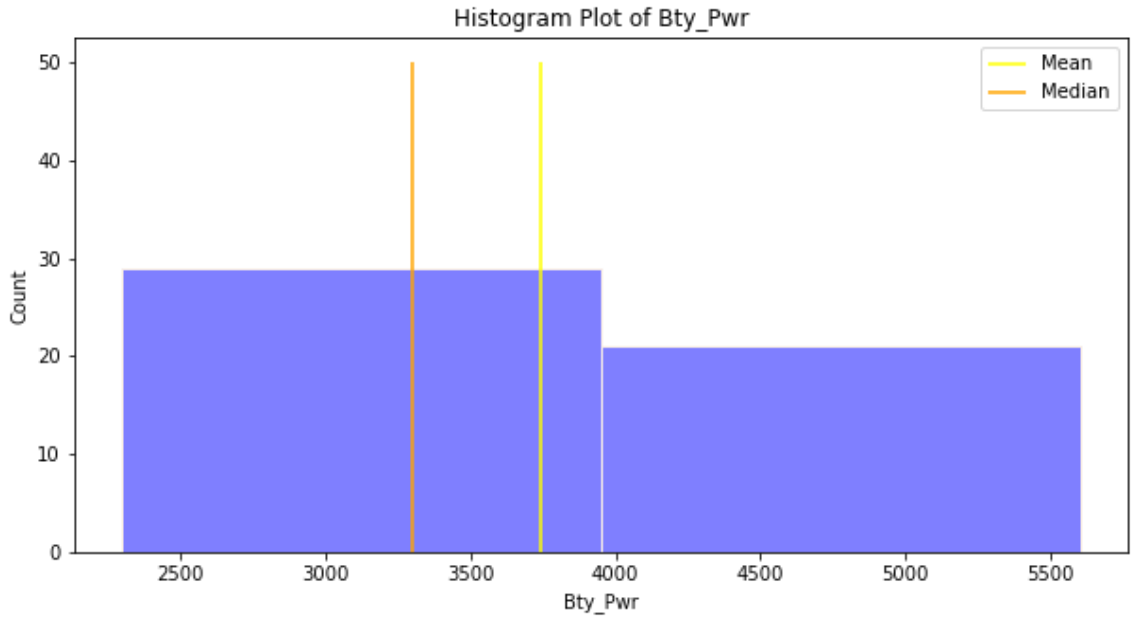
In [47]:

```
# Create a histogram of the "Bty_Pwr" feature and also show the mean and the median


plt.figure(figsize = (10,5))

sns.histplot(data = d , x = 'Bty_Pwr', color='blue', edgecolor = 'linen', alpha = 0.5, b

plt.title("Histogram Plot of Bty_Pwr")
plt.xlabel('Bty_Pwr')
plt.ylabel('Count')
plt.vlines(d['Bty_Pwr'].mean(),ymin=0,ymax=50,colors='yellow',label='Mean')
plt.vlines(d['Bty_Pwr'].median(),ymin=0,ymax=50,colors='orange',label='Median',alpha=1)
plt.legend()
plt.show()
```



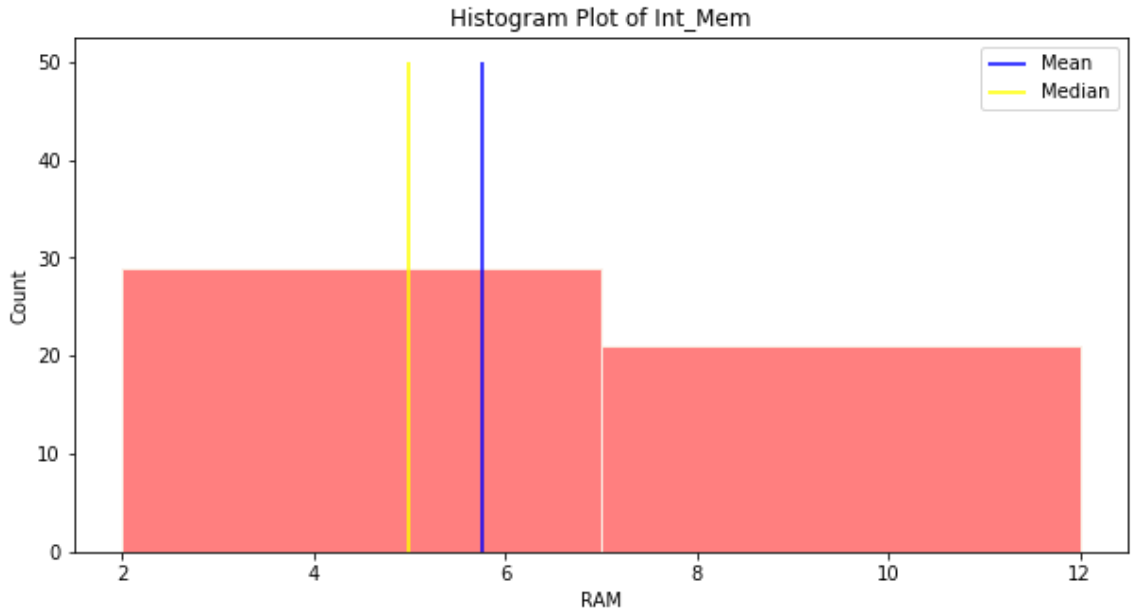
In [48]:

```
# Create a histogram of the "RAM" feature and also show the mean and the median

plt.figure(figsize = (10,5))

sns.histplot(data = d , x = 'RAM', color='red', edgecolor = 'linen', alpha = 0.5, bins=2

plt.title("Histogram Plot of Int_Mem")
plt.xlabel('RAM')
plt.ylabel('Count')
plt.vlines(d['RAM'].mean(),ymin=0,ymax=50,colors='blue',label='Mean')
plt.vlines(d['RAM'].median(),ymin=0,ymax=50,colors='yellow',label='Median')
plt.legend()
plt.show()
```



In [49]:

```
intmean = d['Int_Mem'].mean()
intmean
```

Out[49]:

```
173.76
```

In [50]:

```
Btymean = d['Bty_Pwr'].mean()
Btymean
```

Out[50]:

```
3740.0
```

In [51]:

```
rammean = d['RAM'].mean()
rammean
```

Out[51]:

5.76

In [52]:

```
# The children want phones that have good internal memory, battery power and RAM
# Consider the phones that have internal memory, battery power and RAM greater than or e
# Create a logical condition for this situation and store the logical values as "con5"

con5 = d[(d['Int_Mem']>=intmean) & (d['Bty_Pwr']>=Btymean) & (d['RAM']>=rammean)]
con5
```

Out[52]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Me
28	SSD000L	yes	yes	yes	yes	2580	2120	8	6	64	32	5
29	SYL888P	no	yes	yes	yes	2580	2120	8	6	64	16	2
30	TVF078Y	yes	yes	yes	yes	2580	2120	8	6	64	32	5
32	TYS938L	yes	yes	yes	yes	2580	2120	8	6	64	32	10
42	WZB298K	yes	yes	yes	yes	2580	1980	8	6	64	32	10
44	XTL675G	yes	yes	yes	yes	2580	1980	10	8	64	32	5
47	ZDF789K	yes	yes	yes	no	2880	2520	8	6	64	32	5
48	ZEO567M	yes	yes	yes	no	2880	2520	8	6	128	64	5
49	ZZZ909X	yes	yes	yes	no	2880	2520	8	6	128	64	10

Observations

The features "Int_Mem", "Bty_Pwr" and "RAM" are respectively the internal memory, battery power and RAM of the phones.

The respective means have been selected as thresholds in this case.

.In case it is too strict, we can choose the respective medians as thresholds

Task 7 - Obtain the logical conditions for the features "Depth" and "Weight"

In [ ]:

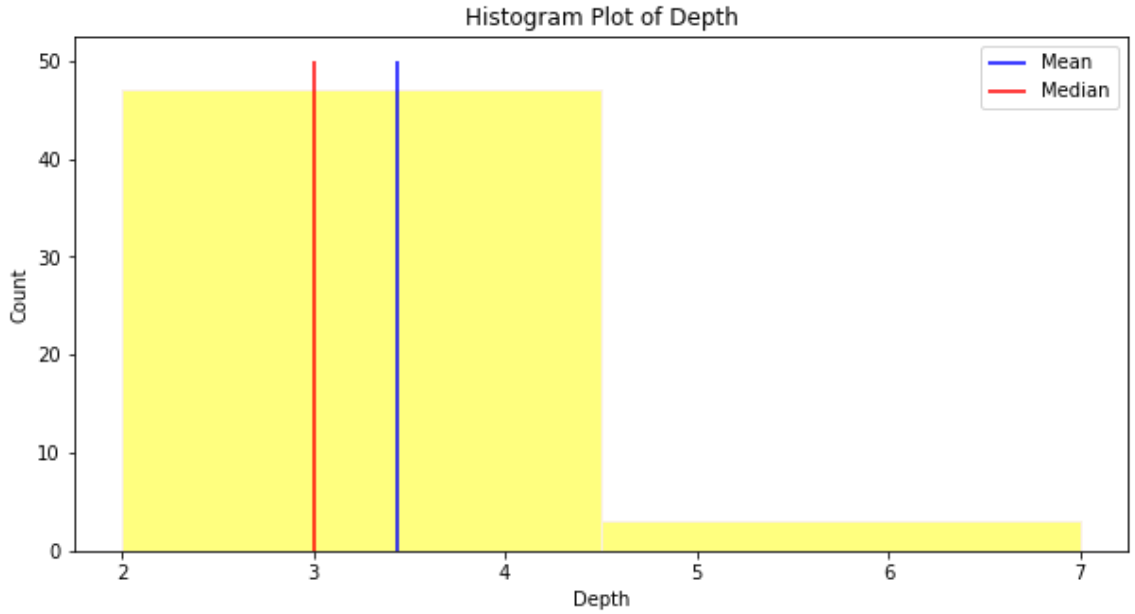
```
# Let's tackle these features: "Depth", "Weight"
```

In [53]:

```
# Create a histogram of the "Depth" feature and also show the mean and the median
plt.figure(figsize = (10,5))

sns.histplot(data = d , x = 'Depth', color='yellow', edgecolor = 'linen', alpha = 0.5, b

plt.title("Histogram Plot of Depth")
plt.xlabel('Depth')
plt.ylabel('Count')
plt.vlines(d['Depth'].mean(),ymin=0,ymax=50,colors='blue',label='Mean')
plt.vlines(d['Depth'].median(),ymin=0,ymax=50,colors='red',label='Median')
plt.legend()
plt.show()
```



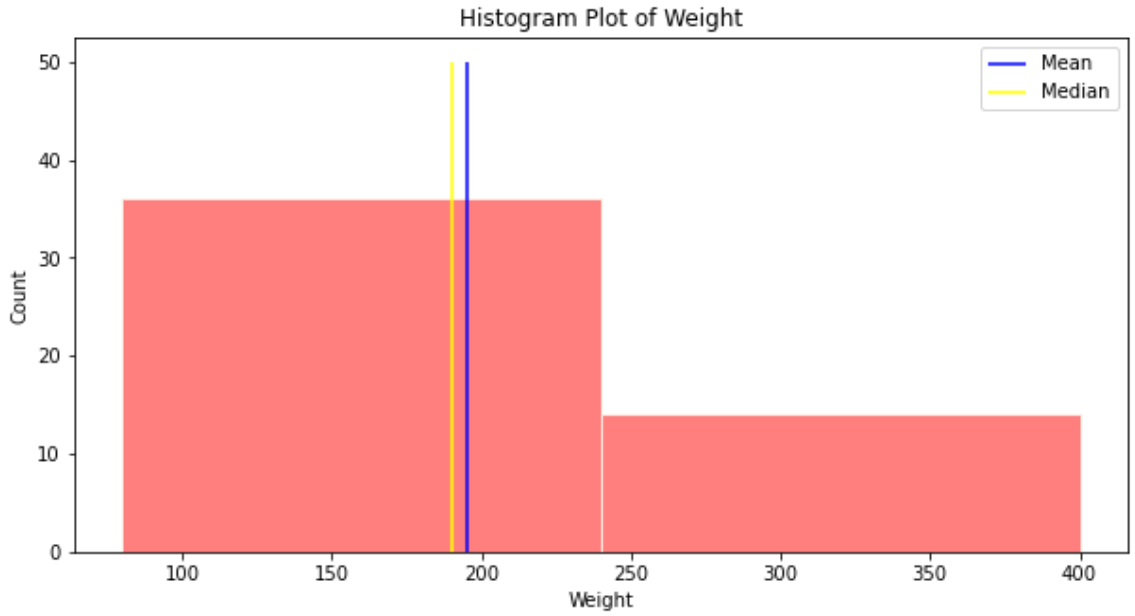
In [54]:

```
# Create a histogram of the "Weight" feature and also show the mean and the median

plt.figure(figsize = (10,5))

sns.histplot(data = d , x = 'Weight', color='red', edgecolor = 'linen', alpha = 0.5, bin

plt.title("Histogram Plot of Weight")
plt.xlabel('Weight')
plt.ylabel('Count')
plt.vlines(d['Weight'].mean(),ymin=0,ymax=50,colors='blue',label='Mean')
plt.vlines(d['Weight'].median(),ymin=0,ymax=50,colors='yellow',label='Median')
plt.legend()
plt.show()
```



In [5]:

```
dmean = d['Depth'].mean()
```

In [6]:

```
wmean = d['Weight'].mean()
```

In [7]:

```
# The children want phones that are light weight and slim
# Consider the phones that have depth and weight less than or equal to the respective me
# Create a logical condition for this situation and store the logical values as "con6"
con6 = d[(d['Depth']<=dmean) | (d['Weight']<=wmean)]
con6
```

Out[7]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_M
3	BBD456K	no	yes	yes	no	1280	1120	5	3	6	2	
4	CCP761U	no	yes	yes	no	1280	1080	4	3	6	2	
5	CCQ674K	yes	no	no	no	1280	1080	4	3	6	4	
6	CTX123L	yes	no	yes	no	1390	1080	6	3	8	4	
7	DFR256N	yes	no	no	no	2880	2120	8	6	12	8	
8	DGS789M	yes	yes	yes	yes	2580	1920	6	3	32	16	
9	ENG897N	yes	yes	yes	yes	2580	1980	5	3	64	32	
11	ELS333L	yes	yes	yes	yes	2580	1920	8	6	64	32	
14	NBN329S	yes	yes	yes	yes	2380	1820	5	3	16	8	
15	NSD450I	no	no	yes	yes	1980	1760	10	8	16	16	
16	PDF768G	no	no	yes	yes	2580	1980	8	6	64	32	
17	PDG234M	no	no	yes	yes	2880	2120	8	6	8	8	
18	PEL111K	no	no	yes	yes	1980	1760	8	6	12	4	
19	PNWD777L	no	no	yes	yes	2880	2120	4	3	24	12	
20	POP857R	no	yes	yes	yes	1980	1760	4	3	32	16	
21	QWR222Y	no	yes	yes	yes	2580	1980	8	6	64	32	
22	QZR577O	no	yes	yes	yes	1440	1280	4	3	8	8	
23	RAY344W	no	yes	yes	yes	2880	2120	10	8	8	4	
24	RBZ451D	no	yes	yes	yes	1440	1280	6	4	8	4	
25	SDO555G	no	yes	yes	yes	2580	1980	6	3	64	32	
26	SET568R	no	yes	yes	yes	1980	1280	5	3	8	4	
27	SFK567Y	yes	yes	yes	yes	2580	2120	8	6	64	16	
28	SSD000L	yes	yes	yes	yes	2580	2120	8	6	64	32	
29	SYL888P	no	yes	yes	yes	2580	2120	8	6	64	16	
30	TVF078Y	yes	yes	yes	yes	2580	2120	8	6	64	32	
31	TYQ109G	no	yes	yes	yes	2580	2120	8	6	32	16	
32	TYS938L	yes	yes	yes	yes	2580	2120	8	6	64	32	1
33	TYU444Q	no	yes	yes	yes	2580	2120	8	6	64	32	
34	TYY453J	yes	yes	yes	yes	2880	2120	6	3	64	32	
39	VYI666I	yes	yes	yes	yes	2440	1980	6	5	32	16	
40	WER765T	yes	yes	yes	yes	2580	1980	5	3	64	32	
41	WUV902Y	yes	yes	yes	yes	2580	1980	8	6	48	16	
42	WZB298K	yes	yes	yes	yes	2580	1980	8	6	64	32	1
43	XKL901R	no	yes	yes	yes	2580	1980	8	6	32	16	
44	XTL675G	yes	yes	yes	yes	2580	1980	10	8	64	32	
45	XXV567F	no	yes	yes	yes	2580	1980	8	6	64	32	
46	YTR67TY	yes	yes	yes	yes	2580	1980	4	3	64	32	

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_M
47	ZDF789K	yes	yes	yes	no	2880	2520	8	6	64	32	
48	ZEO567M	yes	yes	yes	no	2880	2520	8	6	128	64	
49	ZZZ909X	yes	yes	yes	no	2880	2520	8	6	128	64	1

Observations:

The features "Depth" and "Weight" are respectively the depth of the phone and the weight of the phone.

The respective medians have been selected as thresholds in this case.

In case it is too strict, we can choose the respective means as thresholds.

Task 8 - Subset the data based on all the logical conditions

In [60]:

```
# Subset the dataframe using all the logical conditions that have been stored
# Store the subset of the dataframe as a new dataframe called "df1"

df1 = con1
df1
```

Out[60]:

	Blue	Wi_Fi	Tch_Scr	Ext_Mem
0	yes	yes	no	no
1	yes	yes	no	no
2	no	yes	no	no
3	no	yes	yes	no
4	no	yes	yes	no
5	yes	no	no	no
6	yes	no	yes	no
7	yes	no	no	no
8	yes	yes	yes	yes
9	yes	yes	yes	yes
10	yes	yes	yes	yes
11	yes	yes	yes	yes
12	no	yes	yes	yes
13	no	yes	yes	no
14	yes	yes	yes	yes
15	no	no	yes	yes
16	no	no	yes	yes
17	no	no	yes	yes
18	no	no	yes	yes
19	no	no	yes	yes
20	no	yes	yes	yes
21	no	yes	yes	yes
22	no	yes	yes	yes
23	no	yes	yes	yes
24	no	yes	yes	yes
25	no	yes	yes	yes
26	no	yes	yes	yes
27	yes	yes	yes	yes
28	yes	yes	yes	yes
29	no	yes	yes	yes
30	yes	yes	yes	yes
31	no	yes	yes	yes
32	yes	yes	yes	yes
33	no	yes	yes	yes
34	yes	yes	yes	yes
35	no	yes	yes	yes
36	yes	yes	yes	yes

	Blue	Wi_Fi	Tch_Scr	Ext_Mem
37	yes	yes	yes	yes
38	yes	yes	yes	yes
39	yes	yes	yes	yes
40	yes	yes	yes	yes
41	yes	yes	yes	yes
42	yes	yes	yes	yes
43	no	yes	yes	yes
44	yes	yes	yes	yes
45	no	yes	yes	yes
46	yes	yes	yes	yes
47	yes	yes	yes	no
48	yes	yes	yes	no
49	yes	yes	yes	no

In [66]:

```
# Get the dimensions of the dataframe

df1.columns
```

Out[66]:

```
Index(['Blue', 'Wi_Fi', 'Tch_Scr', 'Ext_Mem'], dtype='object')
```

In [67]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   Blue     50 non-null     object
 1   Wi_Fi    50 non-null     object
 2   Tch_Scr  50 non-null     object
 3   Ext_Mem  50 non-null     object
dtypes: object(4)
memory usage: 1.7+ KB
```

In [ ]:

```
# Sort the dataframe according to the "Price" feature in ascending order and display it
df1.sort_values(by = 'Price')
```

Observations:

Based on all the logical conditions obtained through analysis of the features, we are left with three phones.

The most expensive of these phones is the "TYS938L" model and the least expensive is the "TVF078Y" model.

Task 9 - Study the variability of the features in the original data set

In [72]:

```
# Calculate the ratio of the standard deviation to the mean for all the numerical featur
# Store these values in a new series wherein the rows are the features and the only colu
# Name the series as "deviations"

std_dev = d.std()
mean = d.mean()
deviations = std_dev/mean
```

```
C:\Users\kunal\AppData\Local\Temp\ipykernel_19652\1424249030.py:5: FutureW
arning: Dropping of nuisance columns in DataFrame reductions (with 'numeri
c_only=None') is deprecated; in a future version this will raise TypeErro
r.  Select only valid columns before calling the reduction.
  std_dev = d.std()
C:\Users\kunal\AppData\Local\Temp\ipykernel_19652\1424249030.py:6: FutureW
arning: Dropping of nuisance columns in DataFrame reductions (with 'numeri
c_only=None') is deprecated; in a future version this will raise TypeErro
r.  Select only valid columns before calling the reduction.
  mean = d.mean()
```

In [73]:

```
# View the "deviations" series after sorting it in descending order

deviations.sort_values()
```

Out[73]:

```
Px_w       0.256226
Bty_Pwr    0.256368
Px_h       0.257998
Depth      0.306072
Scr_h      0.314293
Scr_d      0.340469
Weight     0.388121
Scr_w      0.407624
RAM        0.479075
Pxx        0.615941
FC         0.712184
PC         0.715716
Price      0.740868
Int_Mem    1.506514
dtype: float64
```

Observations:

The ratio of the standard deviation to the mean of a feature normalises it in a way.

This allows for comparison between multiple features.

The most variable feature in the original data set is the internal memory of the phones.

The least variable feature in the original data set is the number of screen pixels in the horizontal axis.

Although most features don't seem so variable, the prices of the phones are quite variable.

Feel free to investigate what could be the cause of this difference in variability.

Note: We encourage you to extend this analysis further and see what else you can find.

Note: Please refer to the official website of Python and its libraries for various Python documentations.

Conclusion

1. We have used concepts of descriptive statistics to study and work with a data set that contains mobile phone specifications.
2. We were able to recommend three phone models to the client which she can then propose to her children.

In [ ]: