<center>**Telecommunication Industry Project**</center>

# 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 [ ]:

```python
# 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 [ ]:

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

data = pd.read_csv('/content/Mobile_Phones.csv')
```

In [ ]:

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

Out[21]:

|   | 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 [ ]:

```python
# Get the dimensions of the dataframe
data.shape
```

Out[5]:

```
(50, 17)
```

In [ ]:

```
# Get the row names of the dataframe
```

In [ ]:

```
# Get the column names of the dataframe

data.columns
```

Out[11]:

```
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 [ ]:

```
# Look at basic information about the dataframe

data.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 |
| I | Logical OR |
| ^ | Logical XOR (exclusive OR) |
| II | Short-circuit OR |
| && | Short-circuit AND |
| ! | Logical unary NOT |
| &= | AND assignment |
| I= | 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 [ ]:

```python
# Get the feature names of the dataframe

data.columns
```

Out[13]:

```
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 [ ]:

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

In [ ]:

```python
# 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 = data[['Blue','Wi_Fi','Tch_Scr','Ext_Mem']]
con1
```

Out[50]:

|     | 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 [ ]:

```python
# Get the feature names of the dataframe
data.columns
```

Out[40]:

```
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 [ ]:

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

In [ ]:

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

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

In [ ]:

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

Out[65]:

1212.08

In [ ]:

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

In [ ]:

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

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

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

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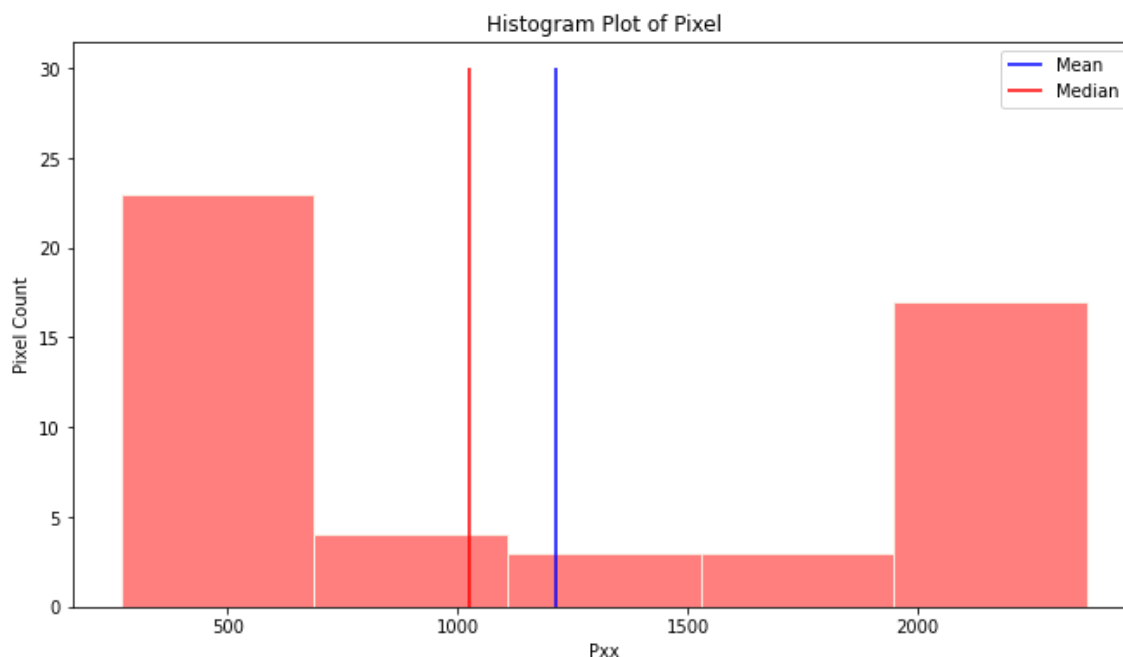
In [ ]:

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

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

Out[74]:

```
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 [ ]:

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

In [ ]:

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

import numpy as np

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

In [ ]:

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

Out[84]:

```
[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 [ ]:

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

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

sns.histplot(data = data , x = 'Scr_d', color='red', edgecolor = 'linen', alpha = 0.5, b

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()
```
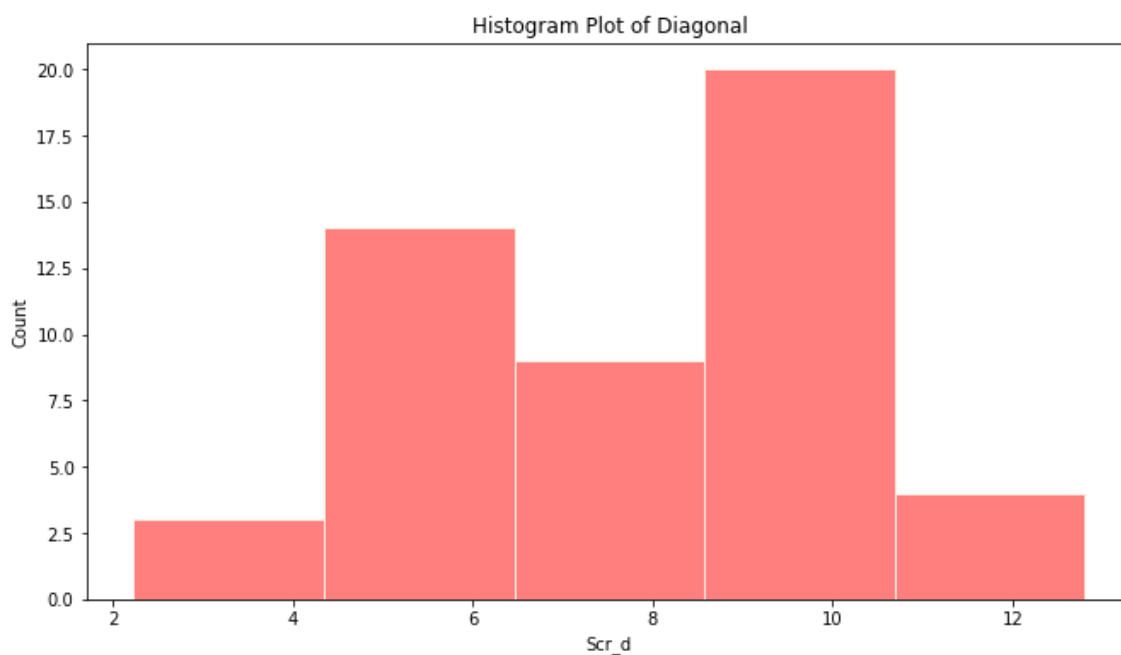


Histogram Plot of Diagonal

In [ ]:

```python
# 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 = [(data['Scr_d']>=upp)]
con3
```

In [ ]:

```python
# 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"
```

Out[99]:

```
[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 [ ]:

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

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

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

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


Histogram Plot of PC

In [ ]:

```python
# 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='yellow', edgecolor = 'linen', alpha = 0.5, b

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 [ ]:
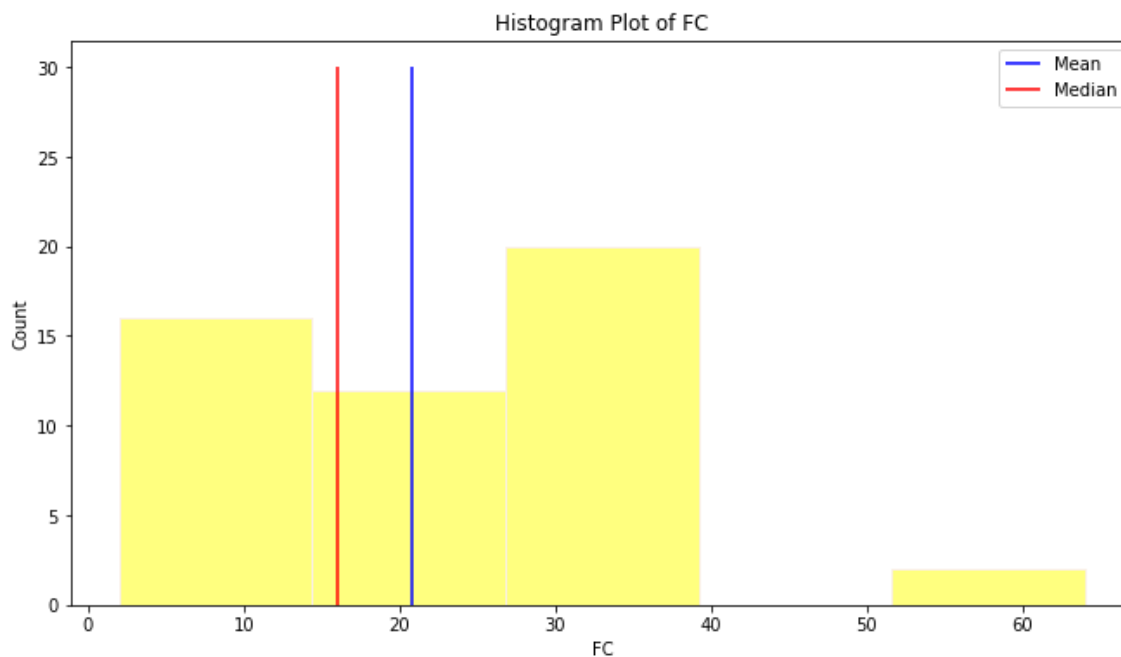
```python
pcm = data['PC'].mean()
pcm
```

Out[106]:

42.16

In [ ]:

```python
fcm = data['FC'].mean()
fcm
```

Out[107]:

20.76

In [ ]:

```python
# 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 = data[(data['PC']>=pcm) & (data['FC']>=fcm)]
con4
```

Out[111]:

| | 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 [ ]:

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

In [ ]:

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

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

sns.histplot(data = data , x = 'Int_Mem', color='red', edgecolor = 'linen', alpha = 0.5,

plt.title("Histogram Plot of Int_Mem")
plt.xlabel('Int_Mem')
plt.ylabel('Count')
plt.vlines(data['Int_Mem'].mean(),ymin=0,ymax=50,colors='blue',label='Mean')
plt.vlines(data['Int_Mem'].median(),ymin=0,ymax=50,colors='red',label='Median')
plt.legend()
plt.show()
```
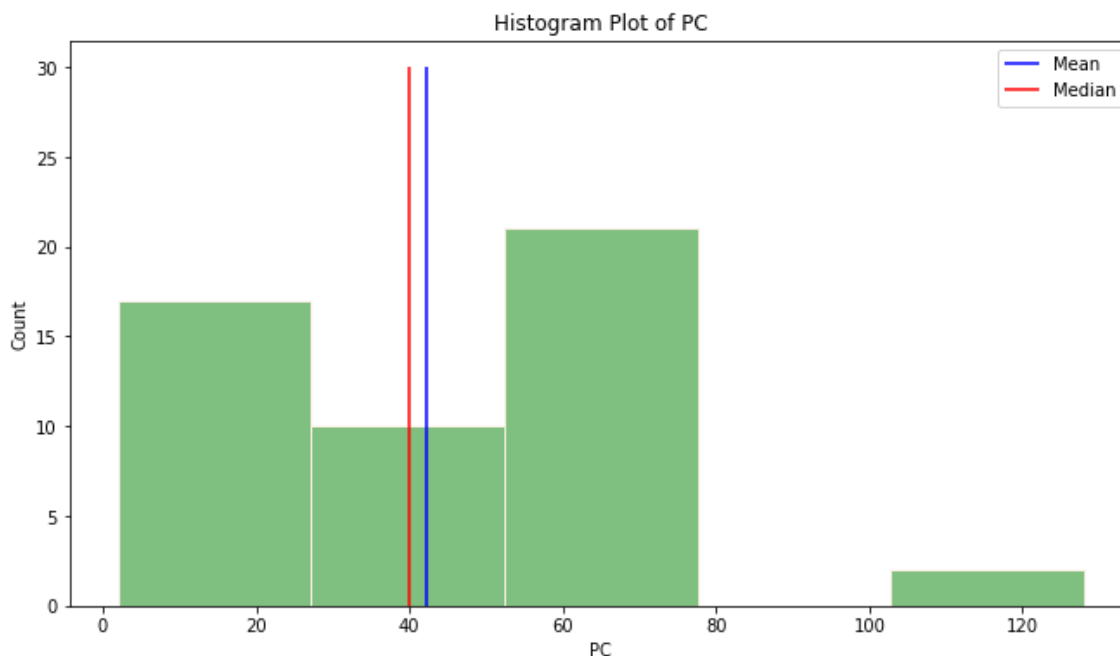
In [ ]:
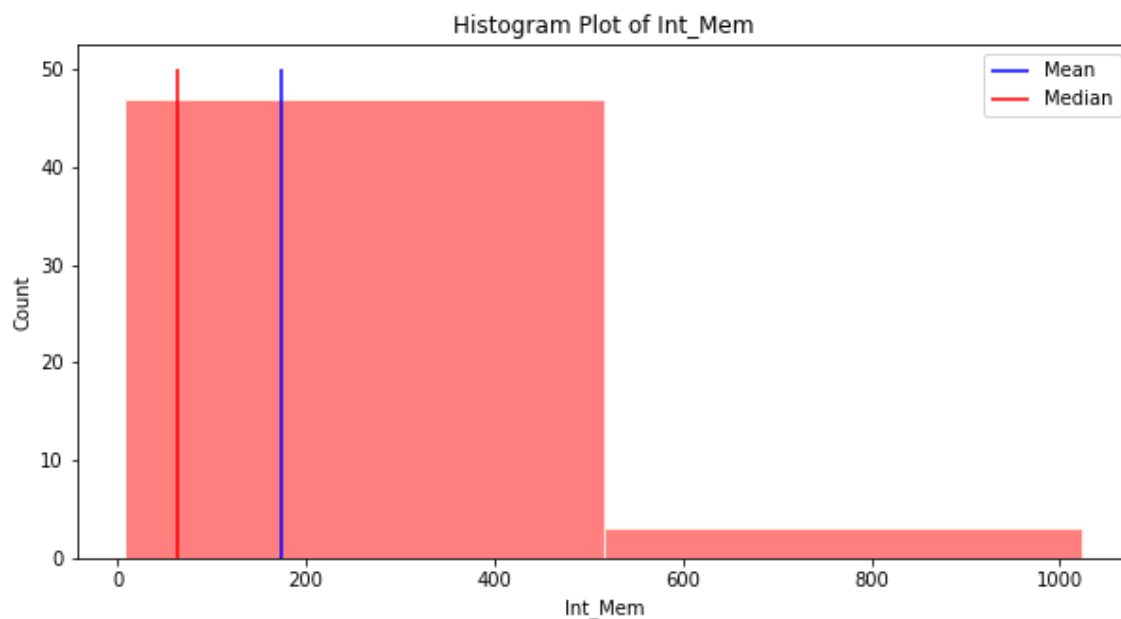
```python
# Create a histogram of the "Bty_Pwr" feature and also show the mean and the median


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

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

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

In [ ]:

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

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

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

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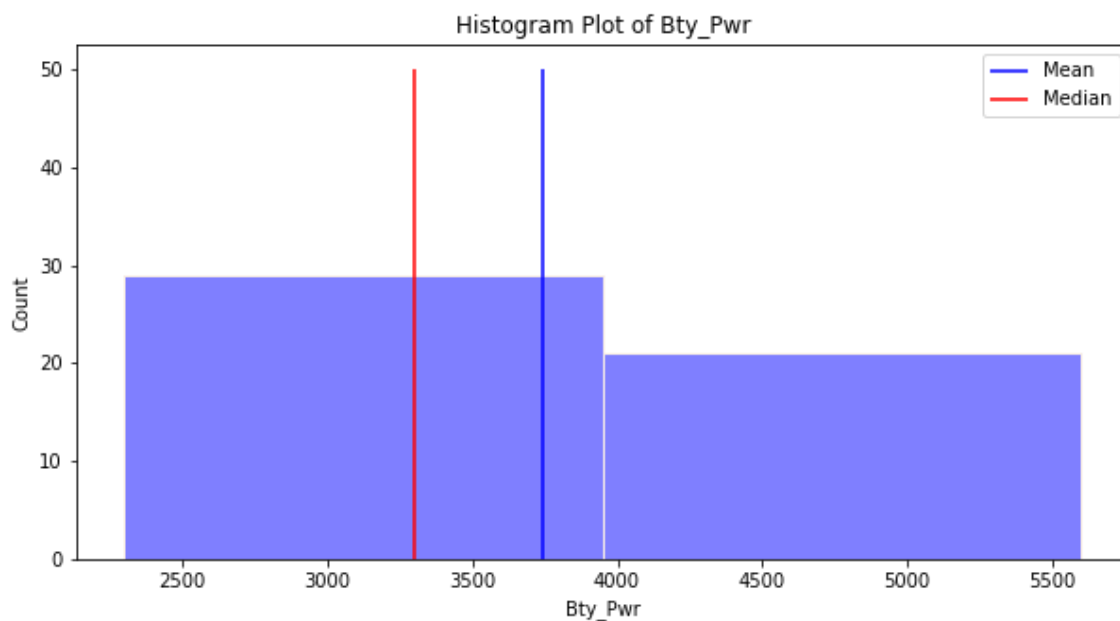


In [ ]:
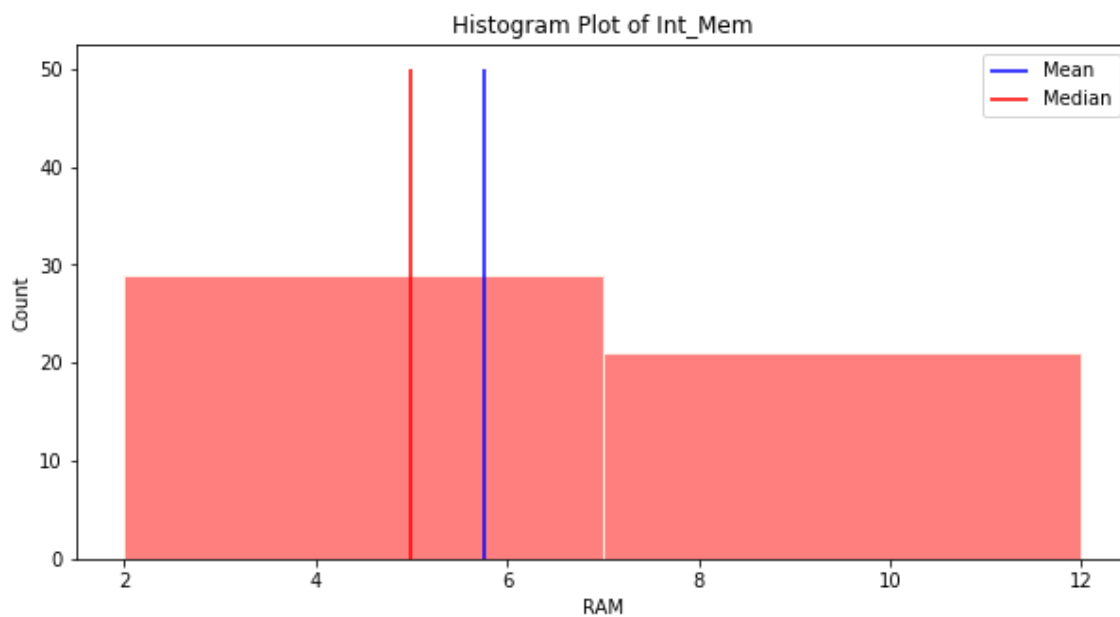
```python
intmean = data['Int_Mem'].mean()
intmean
```

Out[126]:

173.76

In [ ]:

```python
Btymean = data['Bty_Pwr'].mean()
Btymean
```

Out[127]:

3740.0

In [ ]:

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

Out[128]:

5.76

In [ ]:

```
# 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 = data[(data['Int_Mem']>=intmean) & (data['Bty_Pwr']>=Btymean) & (data['RAM']>=ramm
con5
```

Out[125]:

| | 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 [ ]:

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

sns.histplot(data = data , x = 'Depth', color='orange', edgecolor = 'linen', alpha = 0.5

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

In [ ]:
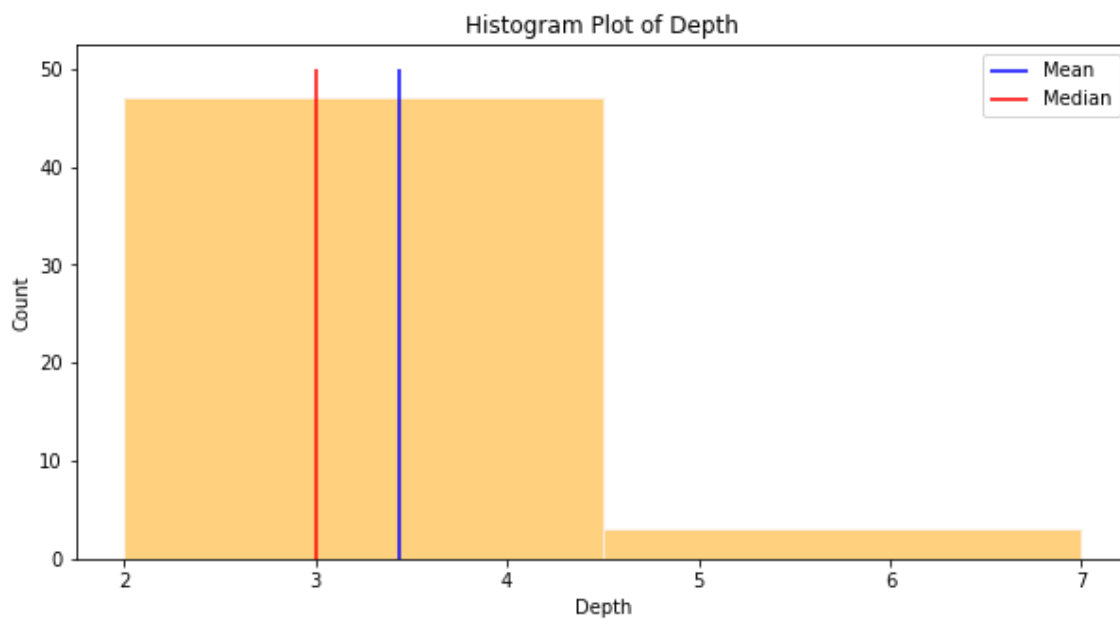
```python
# Create a histogram of the "Weight" feature and also show the mean and the median

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

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

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



In [ ]:
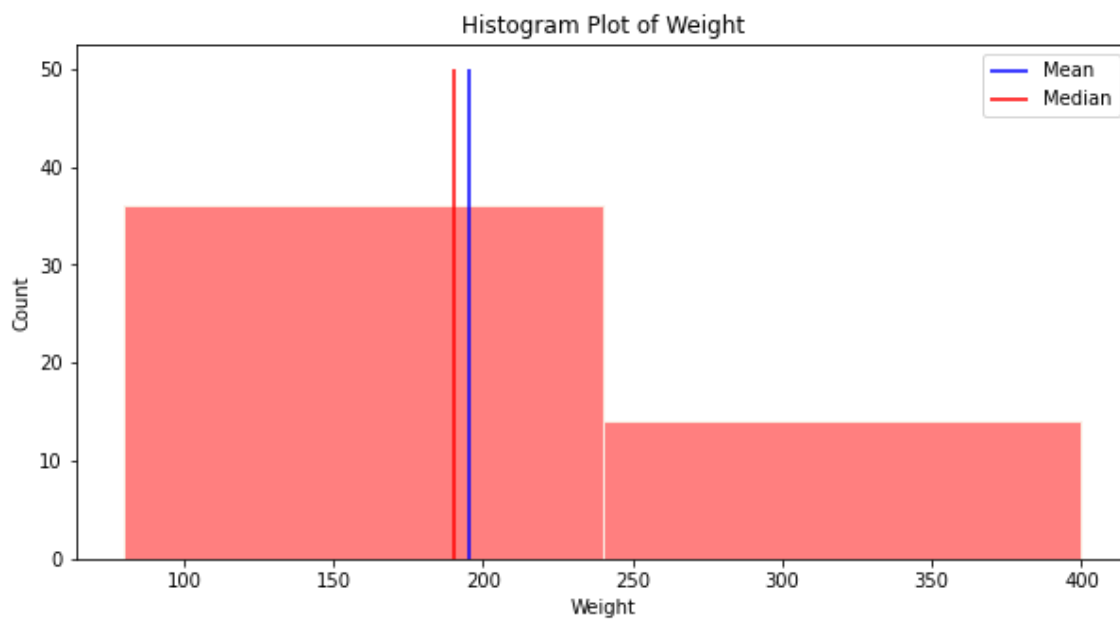
```python
dmean = data['Depth'].mean()
```

In [ ]:

```python
wmean = data['Weight'].mean()
```

In [ ]:

```python
# 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 mea
# Create a logical condition for this situation and store the logical values as "con6"
con6 = data[(data['Depth']<=dmean) & (data['Weight']<=wmean)]
con6
```

Out[136]:

| | PID | Blue | Wi_Fi | Tch_Scr | Ext_Mem | Px_h | Px_w | Scr_h | Scr_w | PC | FC | Int_Me |
|----|---------|------|-------|---------|---------|------|------|-------|-------|-----|-----|--------|
| 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  |        |
| 16 | PDF768G | no   | no    | yes     | yes     | 2580 | 1980 | 8     | 6     | 64  | 32  | 1      |
| 20 | POP857R | no   | yes   | yes     | yes     | 1980 | 1760 | 4     | 3     | 32  | 16  |        |
| 21 | QWR222Y | no   | yes   | yes     | yes     | 2580 | 1980 | 8     | 6     | 64  | 32  | 1      |
| 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   |        |
| 30 | TVF078Y | yes  | yes   | yes     | yes     | 2580 | 2120 | 8     | 6     | 64  | 32  | 5      |
| 31 | TYQ109G | no   | yes   | yes     | yes     | 2580 | 2120 | 8     | 6     | 32  | 16  | 2      |
| 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      |
| 41 | WUV902Y | yes  | yes   | yes     | yes     | 2580 | 1980 | 8     | 6     | 48  | 16  | 1      |
| 42 | WZB298K | yes  | yes   | yes     | yes     | 2580 | 1980 | 8     | 6     | 64  | 32  | 10     |
| 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 "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 [ ]:

```python
# 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[178]:

| | 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 [ ]:

```
# Get the dimensions of the dataframe

df1.columns
```

Out[169]:

```
Index([        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,   'Blue', 'Bty_Pwr',   'Depth', 'Ext_Mem',
            'FC', 'Int_Mem',     'PC',     'PID',   'Price',    'Px_h',
          'Px_w',     'Pxx',    'RAM',   'Scr_d',   'Scr_h',   'Scr_w',
       'Tch_Scr',  'Weight',   'Wi_Fi'],
      dtype='object')
```

In [ ]:

```python
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 51 entries, 0 to Scr_d
Data columns (total 69 columns):
 #    Column    Non-Null Count   Dtype
---   ------    --------------   -----
 0    0         0 non-null       float64
 1    1         0 non-null       float64
 2    2         0 non-null       float64
 3    3         0 non-null       float64
 4    4         0 non-null       float64
 5    5         0 non-null       float64
 6    6         0 non-null       float64
 7    7         0 non-null       float64
 8    8         0 non-null       float64
 9    9         0 non-null       float64
 10   10        0 non-null       float64
 11   11        0 non-null       float64
 12   12        0 non-null       float64
 13   13        0 non-null       float64
 14   14        0 non-null       float64
 15   15        0 non-null       float64
 16   16        0 non-null       float64
 17   17        0 non-null       float64
 18   18        0 non-null       float64
 19   19        0 non-null       float64
 20   20        0 non-null       float64
 21   21        0 non-null       float64
 22   22        0 non-null       float64
 23   23        0 non-null       float64
 24   24        0 non-null       float64
 25   25        0 non-null       float64
 26   26        0 non-null       float64
 27   27        0 non-null       float64
 28   28        0 non-null       float64
 29   29        0 non-null       float64
 30   30        0 non-null       float64
 31   31        0 non-null       float64
 32   32        0 non-null       float64
 33   33        0 non-null       float64
 34   34        0 non-null       float64
 35   35        0 non-null       float64
 36   36        0 non-null       float64
 37   37        0 non-null       float64
 38   38        0 non-null       float64
 39   39        0 non-null       float64
 40   40        0 non-null       float64
 41   41        0 non-null       float64
 42   42        0 non-null       float64
 43   43        0 non-null       float64
 44   44        0 non-null       float64
 45   45        0 non-null       float64
 46   46        0 non-null       float64
 47   47        0 non-null       float64
 48   48        0 non-null       float64
 49   49        0 non-null       float64
 50   Blue      0 non-null       object
 51   Bty_Pwr   0 non-null       float64
 52   Depth     0 non-null       float64
 53   Ext_Mem   0 non-null       object
 54   FC        0 non-null       float64
 55   Int_Mem   0 non-null       float64
```

```
 56  PC        0 non-null      float64
 57  PID       0 non-null      object
 58  Price     0 non-null      float64
 59  Px_h      0 non-null      float64
 60  Px_w      0 non-null      float64
 61  Pxx       0 non-null      float64
 62  RAM       0 non-null      float64
 63  Scr_d     0 non-null      float64
 64  Scr_h     0 non-null      float64
 65  Scr_w     0 non-null      float64
 66  Tch_Scr   0 non-null      object
 67  Weight    0 non-null      float64
 68  Wi_Fi     0 non-null      object
dtypes: float64(64), object(5)
memory usage: 29.9+ KB
```

In [ ]:

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

Out[170]:

|    | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | ... | Px_h | Px_w | Pxx | RAM |
|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|-----|-----|
| 0  | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 1  | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 2  | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 3  | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 4  | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 5  | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 6  | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 7  | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 8  | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 9  | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 10 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 11 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 12 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 13 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 14 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 15 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 16 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 17 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 18 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 19 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 20 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 21 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | Pxx | NaN |
| 22 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 23 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 24 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 25 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 26 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 27 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 28 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 29 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 30 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 31 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 32 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 33 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 34 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 35 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |
| 36 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN  | NaN  | NaN | NaN |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | Px_h | Px_w | Pxx | RAM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 37 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |
| 38 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |
| 39 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |
| 40 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |
| 41 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |
| 42 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |
| 43 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |
| 44 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |
| 45 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |
| 46 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |
| 47 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |
| 48 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |
| 49 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |
| Scr_d | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |

Observations:
51 rows × 69 columns

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.

We could let the children choose from these three phones as per their preferences.

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

In [ ]:

```python
# 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 = data.std()
mean = data.mean()
deviations = std_dev/mean
```

```
<ipython-input-183-27d99c529612>:5: FutureWarning: Dropping of nuisance co
lumns in DataFrame reductions (with 'numeric_only=None') is deprecated; in
a future version this will raise TypeError.  Select only valid columns bef
ore calling the reduction.
  std_dev = data.std()
<ipython-input-183-27d99c529612>:6: FutureWarning: Dropping of nuisance co
lumns in DataFrame reductions (with 'numeric_only=None') is deprecated; in
a future version this will raise TypeError.  Select only valid columns bef
ore calling the reduction.
  mean = data.mean()
```

In [ ]:

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

deviations.sort_values()
```

Out[184]:

```
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 [ ]: