Telecommunication Industry Project

Introduction

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

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

Load the csv file as pandas dataframe.

In [2]:

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

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

In [3]:

```
# Take a brief look at the data
dataset.head()
```

Out[3]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Mem	Bty_Pwr	RAM	Depth	Weight	I
0	AAB346A	yes	yes	no	no	780	460	3	1	2	2	8	2800	2	7	320	
1	AAC347I	yes	yes	no	no	780	560	2	1	4	2	8	3000	2	7	280	
2	BAB657J	no	yes	no	no	840	720	2	1	4	2	8	3300	2	7	400	
3	BBD456K	no	yes	yes	no	1280	1120	5	3	6	2	32	3000	2	3	300	
4	CCP761U	no	yes	yes	no	1280	1080	4	3	6	2	16	3000	2	3	210	

In [4]:

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

Out[4]:

```
(50, 17)
```

In [5]:

```
# Get the row names of the dataframe
dataset.index
```

Out[5]:

```
RangeIndex(start=0, stop=50, step=1)
```

In [6]:

```
# Get the column names of the dataframe

dataset.columns
```

Out[6]:

```
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', 'Weight',
       'Price'],
      dtype='object')
```

In [7]:

```
# Look at basic information about the dataframe

dataset.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.

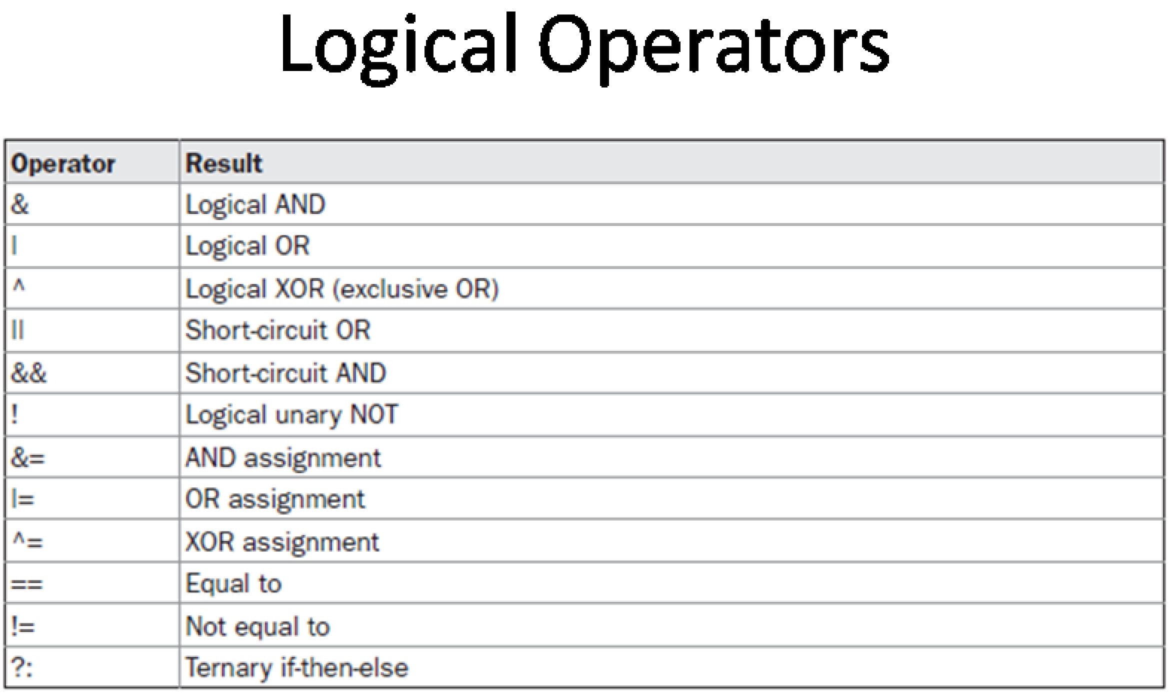
There are no missing values in the data set.

In [7]:

```

```

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



Logical Operators

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

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

In [8]:

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

```
dataset.columns
```

Out[8]:

```
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', 'Weight',
       'Price'],
      dtype='object')
```

In [9]:

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

In [10]:

```
# The children want phones that have the following: Bluetooth, WiFi, touch screen and ext
ernal memory support
# Create a logical condition for this situation and store the logical values as "con1"
#dataset.head()
con1=(dataset["Blue"]=="yes") & (dataset["Wi_Fi"]=="yes") & (dataset["Tch_Scr"]=="yes")
& (dataset["Ext_Mem"]=="yes")
con1.head()
```

Out[10]:

```
0    False
1    False
2    False
3    False
4    False
dtype: bool
```

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 [11]:

```
# Get the feature names of the dataframe

dataset.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', 'Weight',
       'Price'],
      dtype='object')
```

In [12]:

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

In [13]:

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

#dataset.head()
dataset["Px"]=dataset["Px_h"]*dataset["Px_w"]
dataset.head()
```

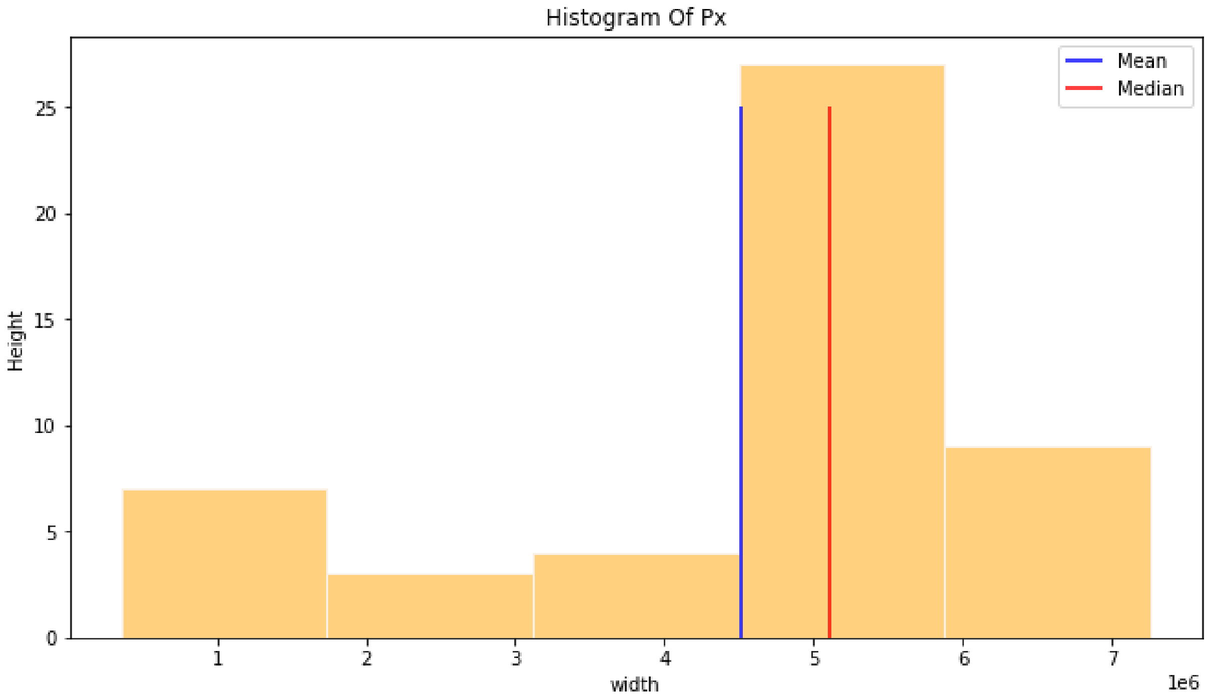
Out[13]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Mem	Bty_Pwr	RAM	Depth	Weight	I

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Mem	Bty_Pwr	RAM	Depth	Weight	I
0	AAB346A	yes	yes	no	no	780	460	3	1	2	2	8	2800	2	7	320	
1	AAC347I	yes	yes	no	no	780	560	2	1	4	2	8	3000	2	7	280	
2	BAB657J	no	yes	no	no	840	720	2	1	4	2	8	3300	2	7	400	
3	BBD456K	no	yes	yes	no	1280	1120	5	3	6	2	32	3000	2	3	300	
4	CCP761U	no	yes	yes	no	1280	1080	4	3	6	2	16	3000	2	3	210	

In [14]:

```
# Create a histogram of the "Px" feature and also show the mean and the median
plt.figure(figsize=(11,6))
sns.histplot(data=dataset,x="Px",color='orange',edgecolor='linen',alpha=0.5,bins=5)
plt.title("Histogram Of Px")
plt.xlabel("width")
plt.ylabel("Height")
plt.vlines(dataset['Px'].mean(), ymin = 0, ymax = 25, colors='blue', label='Mean')
plt.vlines(dataset['Px'].median(), ymin = 0, ymax = 25, colors='red', label='Median')
plt.legend()
plt.show()
```



In [15]:

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

con2=dataset["Px"]>=dataset["Px"].median()
con2
```

Out[15]:

```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7       True
8      False
9       True
10     False
11     False
12      True
13      True
```

```
13      True
14     False
15     False
16      True
17      True
18     False
19      True
20     False
21      True
22     False
23      True
24     False
25      True
26     False
27      True
28      True
29      True
30      True
31      True
32      True
33      True
34      True
35      True
36      True
37     False
38     False
39     False
40      True
41      True
42      True
43      True
44      True
45      True
46      True
47      True
48      True
49      True
Name: Px, dtype: bool
```

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 [16]:

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

In [17]:

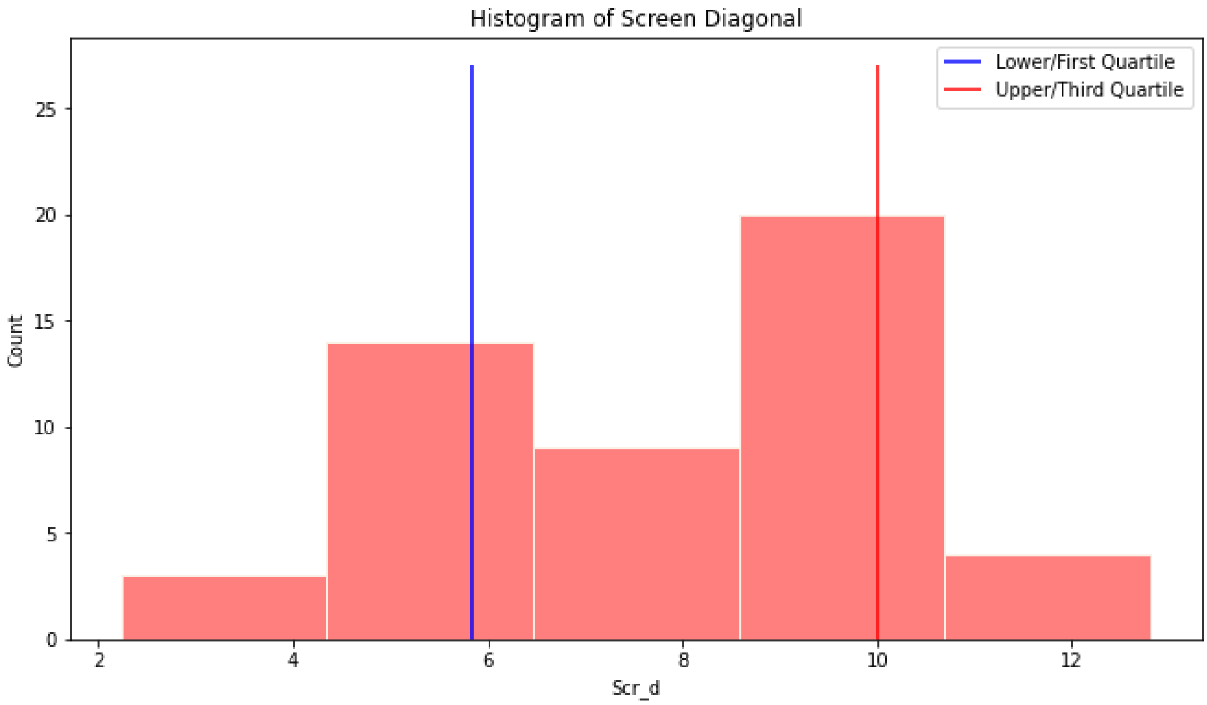
```
# Create a new feature called "Scr_d" which stores the length of the diagonal of the scre
en of the phone
import math
# (dataset.Scr_h).apply(lambda x: float(x))
# (dataset.Scr_w).apply(lambda x: float(x))
Scr_d=np.sqrt((dataset["Scr_h"]**2) + (dataset["Scr_w"]**2))
dataset['Scr_d']=Scr_d
dataset.head()
```

Out[17]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Mem	Bty_Pwr	RAM	Depth	Weight	I
0	AAB346A	yes	yes	no	no	780	460	3	1	2	2	8	2800	2	7	320	
1	AAC347I	yes	yes	no	no	780	560	2	1	4	2	8	3000	2	7	280	
2	BAB657J	no	yes	no	no	840	720	2	1	4	2	8	3300	2	7	400	
3	BBD456K	no	yes	yes	no	1280	1120	5	3	6	2	32	3000	2	3	300	
4	CCP761U	no	yes	yes	no	1280	1080	4	3	6	2	16	3000	2	3	210	

In [18]:

```
# Create a histogram of the "Scr_d" feature and also show the quartiles
Q3,Q1=np.percentile(dataset["Scr_d"],[75,25])
IQR=round(Q3-Q1,2)
plt.figure(figsize=(11,6))
sns.histplot(data=dataset,x="Scr_d",color="red",edgecolor="linen",alpha=0.5,bins=5)
plt.title("Histogram of Screen Diagonal")
plt.xlabel('Scr_d')
plt.ylabel('Count')
plt.vlines(Q1,ymin=0,ymax=27,colors='blue',label='Lower/First Quartile')
plt.vlines(Q3,ymin=0,ymax=27,colors='red',label='Upper/Third Quartile')
#plt.vlines(dataset['Scr_d'].mean(), ymin = 0, ymax = 25, colors='purple', label='Mean')
plt.legend()
plt.show()
```



In [19]:

```
# 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
value in the data set
# Create a logical condition for this situation and store the logical values as "con3"
#if diagonal > Q3
con3=(dataset['Scr_d']>=Q3)
con3
```

Out[19]:

```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      True
```

```
7      True
8     False
9     False
10    False
11     True
12    False
13    False
14    False
15     True
16     True
17     True
18     True
19    False
20    False
21     True
22    False
23     True
24    False
25    False
26    False
27     True
28     True
29     True
30     True
31     True
32     True
33     True
34    False
35    False
36     True
37    False
38    False
39    False
40    False
41     True
42     True
43     True
44     True
45     True
46    False
47     True
48     True
49     True
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 [20]:

```
# Let's tackle these features: "PC", "FC"
dataset.head()
```

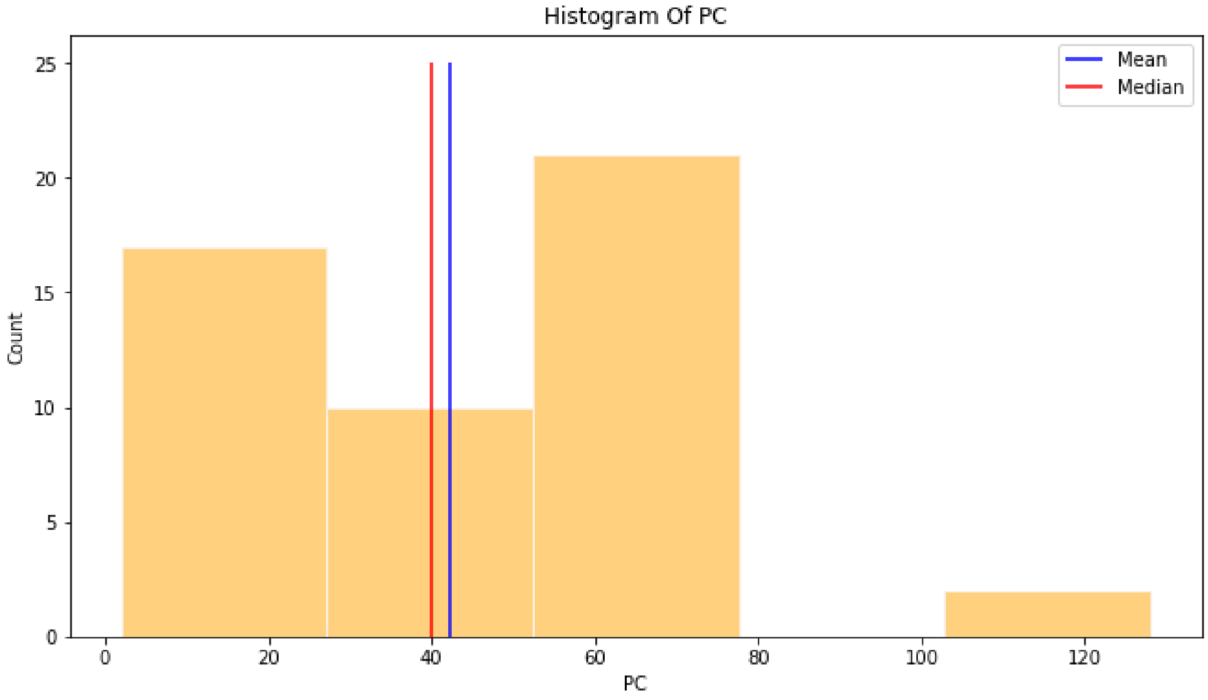
Out[20]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Mem	Bty_Pwr	RAM	Depth	Weight	I
0	AAB346A	yes	yes	no	no	780	460	3	1	2	2	8	2800	2	7	320	

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Mem	Bty_Pwr	RAM	Depth	Weight	I
1	AAC347I	yes	yes	no	no	780	560	2	1	4	2	8	3000	2	7	280	
2	BAB657J	no	yes	no	no	840	720	2	1	4	2	8	3300	2	7	400	
3	BBD456K	no	yes	yes	no	1280	1120	5	3	6	2	32	3000	2	3	300	
4	CCP761U	no	yes	yes	no	1280	1080	4	3	6	2	16	3000	2	3	210	

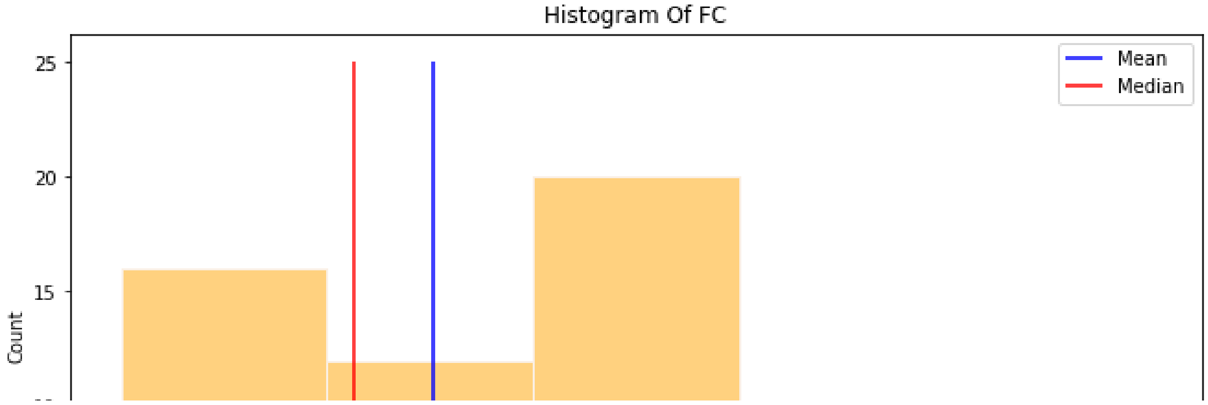
In [21]:

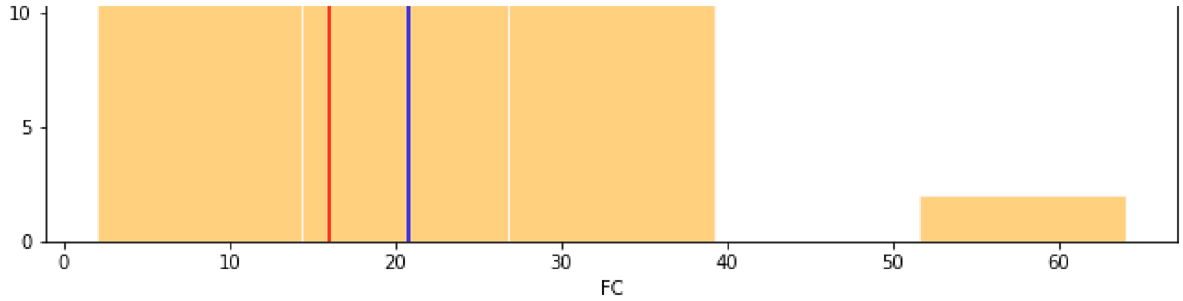
```
# Create a histogram of the "PC" feature and also show the mean and the median
plt.figure(figsize=(11,6))
sns.histplot(data=dataset,x="PC",color='orange',edgecolor='linen',alpha=0.5,bins=5)
plt.title("Histogram Of PC")
plt.xlabel("PC")
plt.ylabel("Count")
plt.vlines(dataset['PC'].mean(), ymin = 0, ymax = 25, colors='blue', label='Mean')
plt.vlines(dataset['PC'].median(), ymin = 0, ymax = 25, colors='red', label='Median')
plt.legend()
plt.show()
```



In [22]:

```
# Create a histogram of the "FC" feature and also show the mean and the median
plt.figure(figsize=(11,6))
sns.histplot(data=dataset,x="FC",color='orange',edgecolor='linen',alpha=0.5,bins=5)
plt.title("Histogram Of FC")
plt.xlabel("FC")
plt.ylabel("Count")
plt.vlines(dataset['FC'].mean(), ymin = 0, ymax = 25, colors='blue', label='Mean')
plt.vlines(dataset['FC'].median(), ymin = 0, ymax = 25, colors='red', label='Median')
plt.legend()
plt.show()
```





In [23]:

```
# 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 equa
l to their respective mean values
# Create a logical condition for this situation and store the logical values as "con4"

con4=(dataset['PC']>=dataset['PC'].mean()) & (dataset['FC']>=dataset['FC'].mean())
con4
```

Out[23]:

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

```
49      True
dtype: bool
```

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 [24]:

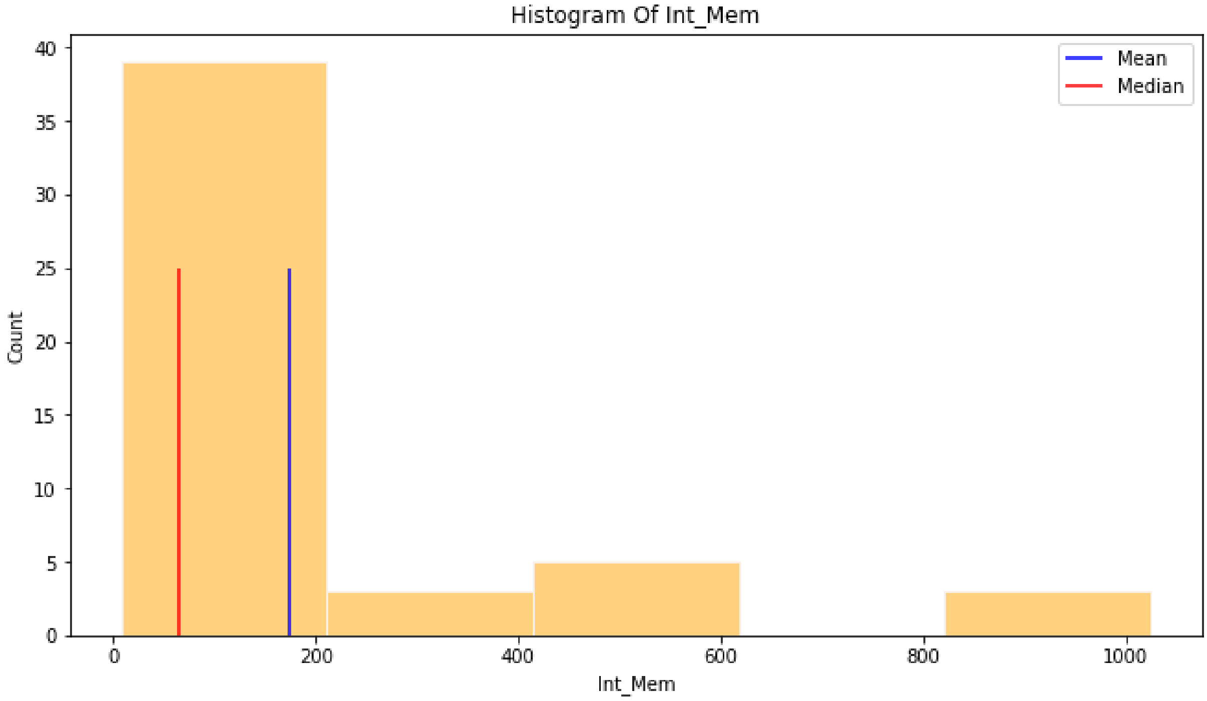
```
# Let's tackle these features: "Int_Mem", "Bty_Pwr", "RAM"
dataset['Int_Mem'].mean()
```

Out[24]:

```
173.76
```

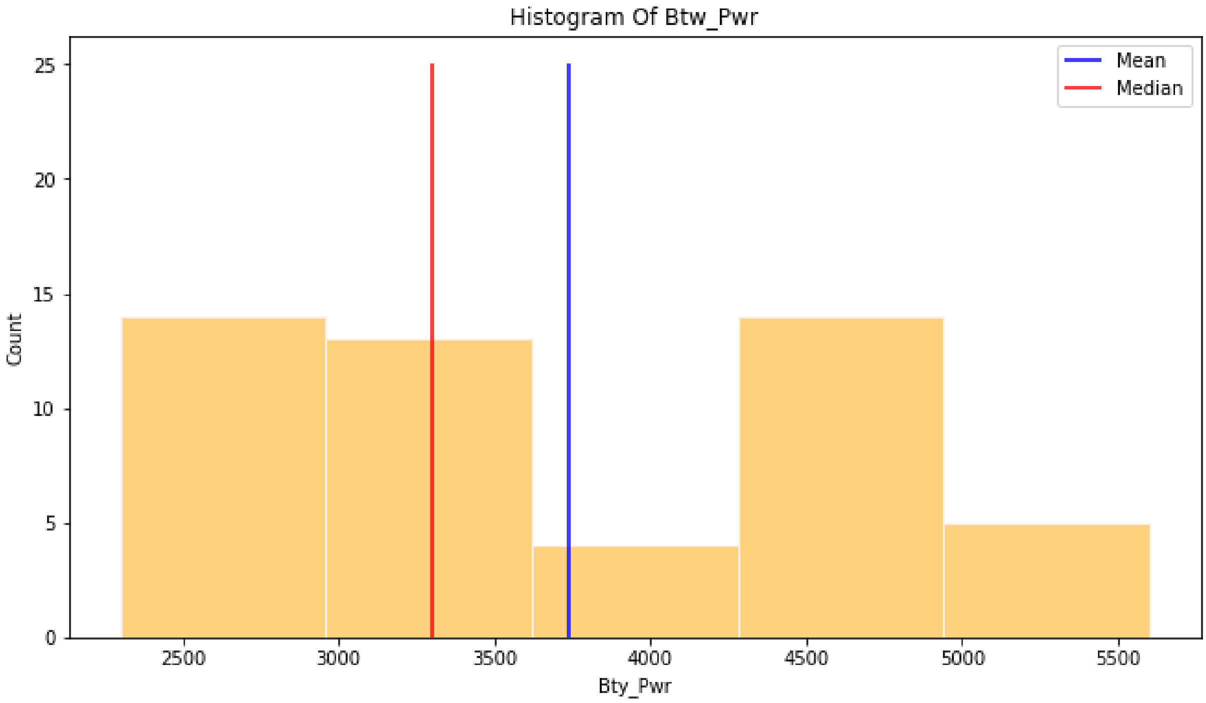
In [25]:

```
# Create a histogram of the "Int_Mem" feature and also show the mean and the median
plt.figure(figsize=(11,6))
sns.histplot(data=dataset,x="Int_Mem",color='orange',edgecolor='linen',alpha=0.5,bins=5)
plt.title("Histogram Of Int_Mem")
plt.xlabel("Int_Mem")
plt.ylabel("Count")
plt.vlines(dataset['Int_Mem'].mean(), ymin = 0, ymax = 25, colors='blue', label='Mean')
plt.vlines(dataset['Int_Mem'].median(), ymin = 0, ymax = 25, colors='red', label='Median
')
plt.legend()
plt.show()
```



In [26]:

```
# Create a histogram of the "Bty_Pwr" feature and also show the mean and the median
plt.figure(figsize=(11,6))
sns.histplot(data=dataset,x="Bty_Pwr",color='orange',edgecolor='linen',alpha=0.5,bins=5)
plt.title("Histogram Of Btw_Pwr")
plt.xlabel("Bty_Pwr")
plt.ylabel("Count")
plt.vlines(dataset['Bty_Pwr'].mean(), ymin = 0, ymax = 25, colors='blue', label='Mean')
plt.vlines(dataset['Bty_Pwr'].median(), ymin = 0, ymax = 25, colors='red', label='Median
')
plt.legend()
plt.show()
```



In [27]:

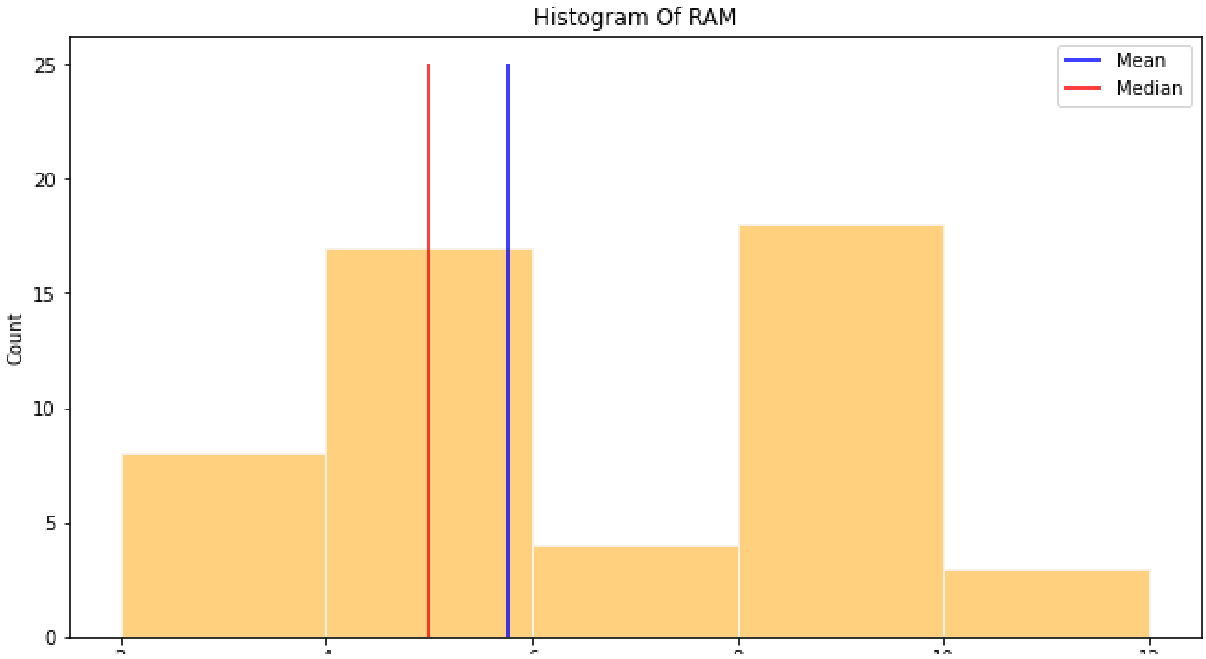
```
dataset['Bty_Pwr'].mean()
```

Out[27]:

```
3740.0
```

In [28]:

```
# Create a histogram of the "RAM" feature and also show the mean and the median
plt.figure(figsize=(11,6))
sns.histplot(data=dataset,x="RAM",color='orange',edgecolor='linen',alpha=0.5,bins=5)
plt.title("Histogram Of RAM")
plt.xlabel("RAM")
plt.ylabel("Count")
plt.vlines(dataset['RAM'].mean(), ymin = 0, ymax = 25, colors='blue', label='Mean')
plt.vlines(dataset['RAM'].median(), ymin = 0, ymax = 25, colors='red', label='Median')
plt.legend()
plt.show()
```



In [29]:

```
dataset['RAM'].mean()
```

Out[29]:

```
5.76
```

In [30]:

```
# 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 equal to their respective mean values
# Create a logical condition for this situation and store the logical values as "con5"
con5=(dataset['Int_Mem']>=dataset['Int_Mem'].mean()) & (dataset['Bty_Pwr']>=dataset['Bty_Pwr'].mean()) & (dataset['RAM']>=dataset['RAM'].mean())
con5
```

Out[30]:

```
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     True
29     True
30     True
31    False
32     True
33    False
34    False
35    False
36    False
37    False
38    False
39    False
40    False
41    False
42     True
43    False
44     True
45    False
46    False
47     True
48     True
49     True
```

```
dtype: bool
```

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

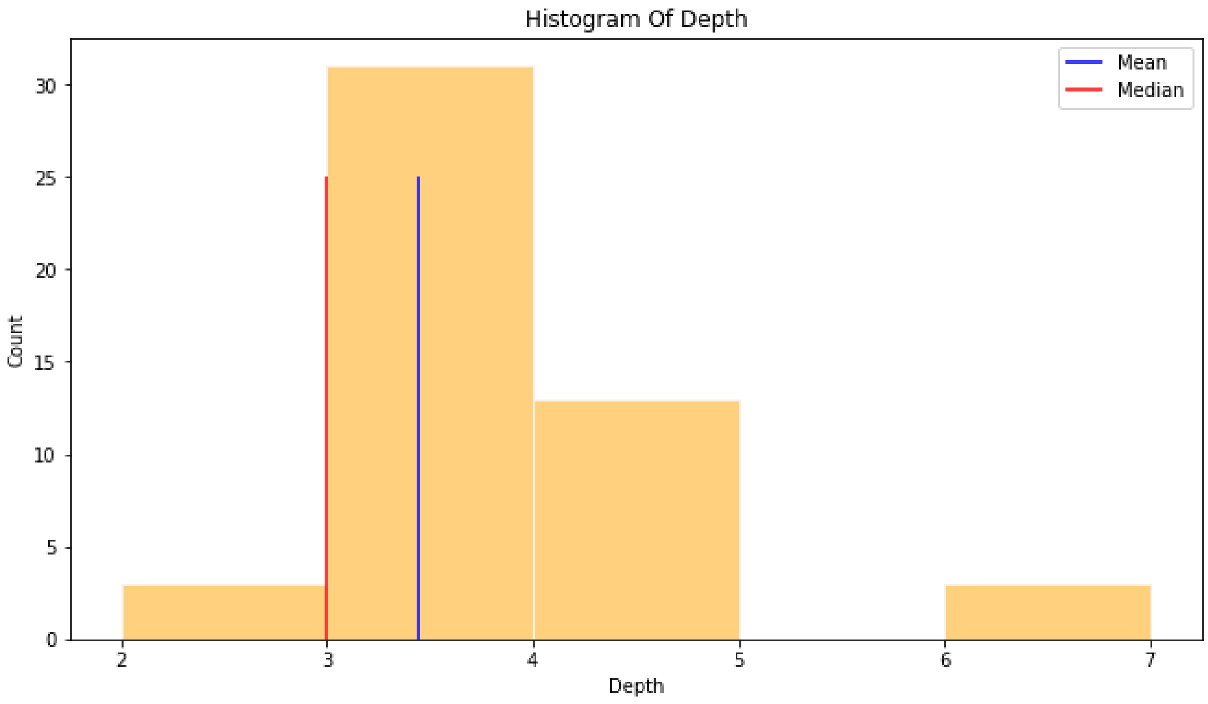
Task 7 - Obtain the logical conditions for the features "Depth" and "Weight"

In [31]:

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

In [32]:

```
# Create a histogram of the "Depth" feature and also show the mean and the median
plt.figure(figsize=(11,6))
sns.histplot(data=dataset,x="Depth",color='orange',edgecolor='linen',alpha=0.5,bins=5)
plt.title("Histogram Of Depth")
plt.xlabel("Depth")
plt.ylabel("Count")
plt.vlines(dataset['Depth'].mean(), ymin = 0, ymax = 25, colors='blue', label='Mean')
plt.vlines(dataset['Depth'].median(), ymin = 0, ymax = 25, colors='red', label='Median')
plt.legend()
plt.show()
```



In [33]:

```
dataset['Depth'].mean()
```

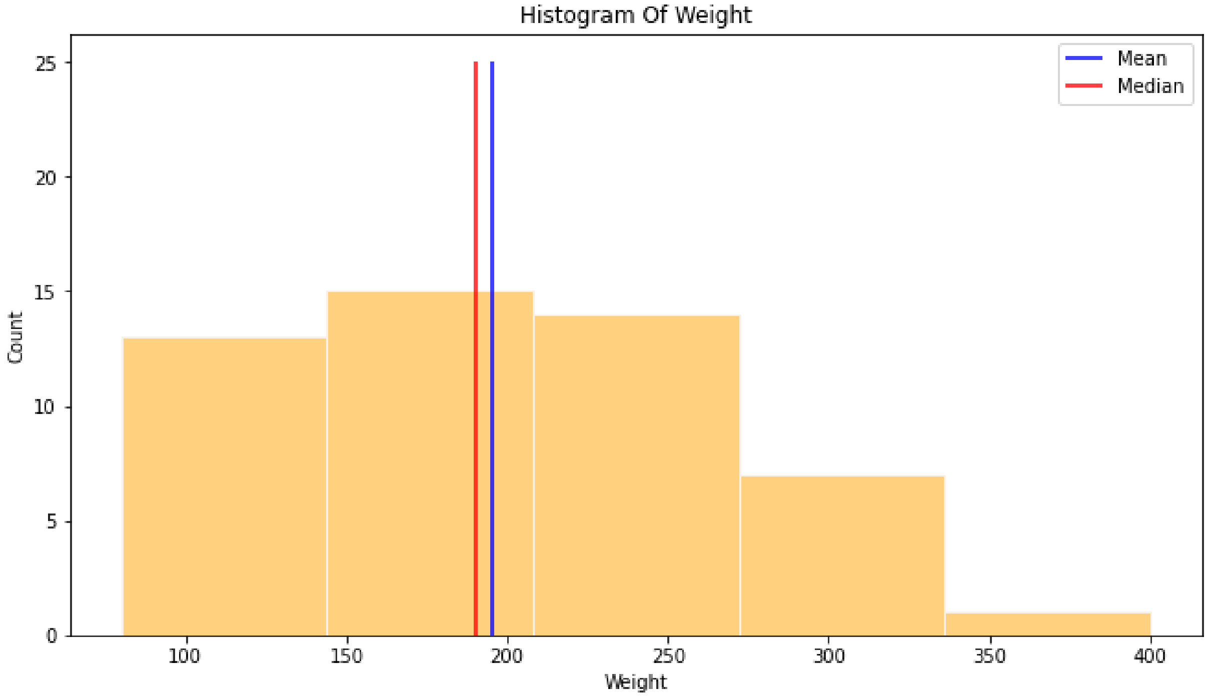
Out[33]:

```
3.44
```

In [34]:

```
# Create a histogram of the "Weight" feature and also show the mean and the median
plt.figure(figsize=(11,6))
sns.histplot(data=dataset,x="Weight",color='orange',edgecolor='linen',alpha=0.5,bins=5)
```

```
plt.title("Histogram Of Weight")
plt.xlabel("Weight")
plt.ylabel("Count")
plt.vlines(dataset['Weight'].mean(), ymin = 0, ymax = 25, colors='blue', label='Mean')
plt.vlines(dataset['Weight'].median(), ymin = 0, ymax = 25, colors='red', label='Median'
)
plt.legend()
plt.show()
```



In [35]:

```
dataset['Weight'].mean()
```

Out[35]:

```
195.2
```

In [36]:

```
# 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 med
ian values in the data set
# Create a logical condition for this situation and store the logical values as "con6"
con6=(dataset['Depth']<=dataset['Depth'].median()) & (dataset['Weight']<=dataset['Weight
'].median())
con6
```

Out[36]:

```
0      False
1      False
2      False
3      False
4      False
5      False
6       True
7       True
8       True
9       True
10     False
11     False
12     False
13     False
14     False
15     False
16      True
17     False
18     False
19     False
```

```
20      True
21      True
22      True
23      True
24      True
25     False
26     False
27     False
28     False
29     False
30      True
31      True
32      True
33      True
34     False
35     False
36     False
37     False
38     False
39     False
40     False
41      True
42      True
43     False
44     False
45     False
46     False
47      True
48      True
49      True
dtype: bool
```

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 [37]:

```
# 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= dataset[(dataset['Depth']<=dataset['Depth'].median()) & (dataset['Weight']<=dataset['Weight'].median()) & (dataset['Int_Mem']>=dataset['Int_Mem'].mean()) & (dataset['Bty_Pwr']>=dataset['Bty_Pwr'].mean()) & (dataset['RAM']>=dataset['RAM'].mean()) & (dataset['PC']>=dataset['PC'].mean()) & (dataset['FC']>=dataset['FC'].mean()) & (dataset['Scr_d']>=Q3) & (dataset["Px"]>=dataset["Px"].median()) & (dataset["Blue"]=="yes") & (dataset["Wi_Fi"]=="yes") & (dataset["Tch_Scr"]=="yes") & (dataset["Ext_Mem"]=="yes")]
df1
```

Out[37]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Mem	Bty_Pwr	RAM	Depth	Weight
30	TVF078Y	yes	yes	yes	yes	2580	2120	8	6	64	32	512	4860	8	3	90
32	TYS938L	yes	yes	yes	yes	2580	2120	8	6	64	32	1024	4860	8	3	120
42	WZB298K	yes	yes	yes	yes	2580	1980	8	6	64	32	1024	5600	8	3	160

In [38]:

```
df1= dataset[con1 & con2 & con3 & con4 & con5 & con6]
```

```
df1
```

Out[38]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Mem	Bty_Pwr	RAM	Depth	Weight
30	TVF078Y	yes	yes	yes	yes	2580	2120	8	6	64	32	512	4860	8	3	90
32	TYS938L	yes	yes	yes	yes	2580	2120	8	6	64	32	1024	4860	8	3	120
42	WZB298K	yes	yes	yes	yes	2580	1980	8	6	64	32	1024	5600	8	3	160

In [39]:

```
# Get the dimensions of the dataframe
df1.shape
```

Out[39]:

```
(3, 19)
```

In [40]:

```
# Sort the dataframe according to the "Price" feature in ascending order and display it
df1.sort_values(by=["Price"],inplace=True)
df1
```

```
/usr/local/lib/python3.8/dist-packages/pandas/util/_decorators.py:311: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  return func(*args, **kwargs)
```

Out[40]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Mem	Bty_Pwr	RAM	Depth	Weight
30	TVF078Y	yes	yes	yes	yes	2580	2120	8	6	64	32	512	4860	8	3	90
42	WZB298K	yes	yes	yes	yes	2580	1980	8	6	64	32	1024	5600	8	3	160
32	TYS938L	yes	yes	yes	yes	2580	2120	8	6	64	32	1024	4860	8	3	120

In [41]:

```
df1=df1.sort_values(by=["Price"],ascending=False)
df1
```

Out[41]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Mem	Bty_Pwr	RAM	Depth	Weight
32	TYS938L	yes	yes	yes	yes	2580	2120	8	6	64	32	1024	4860	8	3	120
42	WZB298K	yes	yes	yes	yes	2580	1980	8	6	64	32	1024	5600	8	3	160
30	TVF078Y	yes	yes	yes	yes	2580	2120	8	6	64	32	512	4860	8	3	90

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.

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

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

In [42]:

```
# Calculate the ratio of the standard deviation to the mean for all the numerical feature
s in the dataframe
# Store these values in a new series wherein the rows are the features and the only colum
n is the calculated ratio
# Name the series as "deviations"

deviations=pd.Series(index=['Px_h','Px_w','Scr_h','Scr_w','PC','FC','Int_Mem','Bty_Pwr',
'RAM','Depth','Weight','Price','Px','Scr_d'],data=[(dataset['Px_h'].std()/dataset['Px_h'
].mean()),(dataset['Px_w'].std()/dataset['Px_w'].mean()),
                                                                (dataset['Scr_h'].std(
)/dataset['Scr_h'].mean()),(dataset['Scr_w'].std()/dataset['Scr_w'].mean()),
                                                                (dataset['PC'].std()/d
ataset['PC'].mean()),(dataset['FC'].std()/dataset['FC'].mean()),
                                                                (dataset['Int_Mem'].st
d()/dataset['Int_Mem'].mean()),(dataset['Bty_Pwr'].std()/dataset['Bty_Pwr'].mean()),
                                                                (dataset['RAM'].std()/
dataset['RAM'].mean()),(dataset['Depth'].std()/dataset['Depth'].mean()),
                                                                (dataset['Weight'].std
()/dataset['Weight'].mean()),(dataset['Price'].std()/dataset['Price'].mean()),
                                                                (dataset['Px'].std()/d
ataset['Px'].mean()),(dataset['Scr_d'].std()/dataset['Scr_d'].mean())])
```

In [43]:

```
deviations
```

Out[43]:

```
Px_h       0.257998
Px_w       0.256226
Scr_h      0.314293
Scr_w      0.407624
PC         0.715716
FC         0.712184
Int_Mem    1.506514
Bty_Pwr    0.256368
RAM        0.479075
Depth      0.306072
Weight     0.388121
Price      0.740868
Px         0.398680
Scr_d      0.340469
dtype: float64
```

In [44]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 19 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 
 13   RAM       50  non-null     int64
 14   Depth     50  non-null     int64
 15   Weight    50  non-null     int64
 16   Price     50  non-null     int64
 17   Px        50  non-null     int64
 18   Scr_d     50  non-null     float64
dtypes: float64(1), int64(13), object(5)
memory usage: 7.5+ KB
```

In [45]:

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

deviations.sort_values()
```

Out[45]:

```
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
Px          0.398680
Scr_w       0.407624
RAM         0.479075
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.

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 [45]:

```

```