Telecommunication Industry Project

Introduction

Welcome to this Jupyter notebook, a crucial part of your applied statistics learning experience. In this project, you'll be working with a dataset containing mobile phone prices and specifications.

Dataset Columns Information

Column	Description
PID	Unique identifier for the phone model
Blue	Bluetooth support (Yes/No)
Wi_Fi	WiFi support (Yes/No)
Tch_Scr	Touch screen support (Yes/No)
Ext_Mem	External Memory support (Yes/No)
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 phone screen (in cm)
Scr_w	Width of the phone screen (in cm)
Int_Mem	Internal memory of the phone (in MB)
Bty_Pwr	Maximum battery energy (in mAh)
PC	Primary camera resolution (in MP)
FC	Front camera resolution (in MP)
RAM	Random access memory (in GB)
Depth	Depth of the mobile phone (in cm)
Weight	Weight of the mobile phone (in 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
# Importing the basic libraries
import numpy as np
import pandas as pd
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
df = pd.read_csv("Telecom.csv")
```

In [3]:
```
# Take a brief look at the data
df
```

Out[3]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_M
0	AAB346A	yes	yes	no	no	780	460	3	1	2	2	
1	AAC347I	yes	yes	no	no	780	560	2	1	4	2	
2	BAB657J	no	yes	no	no	840	720	2	1	4	2	
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	
10	EOP657N	yes	yes	yes	yes	2580	1920	6	3	32	16	
11	ELS333L	yes	yes	yes	yes	2580	1920	8	6	64	32	
12	ETT987D	no	yes	yes	yes	2580	1980	6	5	64	32	
13	NAJ56GL	no	yes	yes	no	2880	2120	6	5	32	16	
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	

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_M
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	
35	ULI999T	no	yes	yes	yes	2440	2120	5	3	32	16	
36	UST000T	yes	yes	yes	yes	2580	1980	10	8	64	32	
37	USZ111S	yes	yes	yes	yes	2440	1980	5	3	48	32	
38	VWV532Y	yes	yes	yes	yes	2580	1920	5	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	
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

In [4]:
```
# Get the dimensions of the dataframe
df.shape
```

Out[4]:
```
(50, 17)
```

In [5]:
```
# Get the row names of the dataframe

df.head()
```

Out[5]:

	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 [6]:
```
# Get the column names of the dataframe
df.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
df.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 [8]:
```
df.isnull().sum()
```

Out[8]:
```
PID         0
Blue        0
Wi_Fi       0
Tch_Scr     0
Ext_Mem     0
Px_h        0
Px_w        0
Scr_h       0
Scr_w       0
PC          0
FC          0
Int_Mem     0
Bty_Pwr     0
RAM         0
Depth       0
Weight      0
Price       0
dtype: int64
```

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

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

In [9]:
```
# Get the feature names of the dataframe
df.columns
```

Out[9]:
```
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 [10]:
```
# Creating a function to count the values of a variable
def count_values(dataframe, column_list):
    for column in column_list:
        print(f"Value counts for {column}:")
```

```
        print(dataframe[column].value_counts())
        print("-" * 20)
```

In [11]:
```
l = ["Blue", "Wi_Fi", "Tch_Scr", "Ext_Mem"]
```

In [12]:
```
count_values(df,l)
```

```
Value counts for Blue:
Blue
yes    27
no     23
Name: count, dtype: int64
--------------------
Value counts for Wi_Fi:
Wi_Fi
yes    42
no      8
Name: count, dtype: int64
--------------------
Value counts for Tch_Scr:
Tch_Scr
yes    45
no      5
Name: count, dtype: int64
--------------------
Value counts for Ext_Mem:
Ext_Mem
yes    38
no     12
Name: count, dtype: int64
--------------------
```

The children want phones that have the following: Bluetooth, WiFi, touch screen and external memory support

Create a logical condition for this situation and store the logical values as "con1"

In [13]:
```
con1 = (df['Blue'] == 'yes') & (df['Wi_Fi'] == 'yes') & (df['Tch_Scr'] == 'yes') &
```

In [14]:
```
con1.head()
```

Out[14]:
```
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 [15]:
```
# Get the feature names of the dataframe
l2= ['Px_h','Px_w']
```

In [16]:
```
# Let's tackle these features: "Px_h", "Px_w"
count_values(df,l2)
```

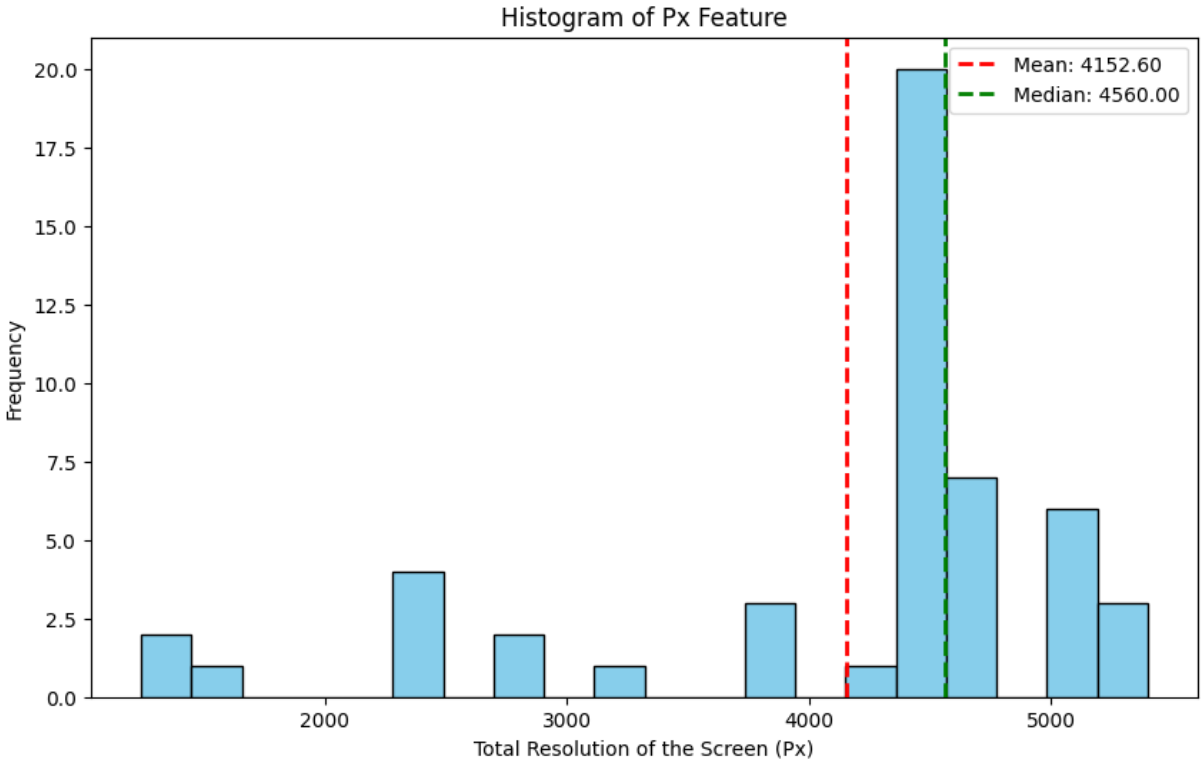
```
Value counts for Px_h:
Px_h
2580    24
2880     9
1980     4
1280     3
2440     3
780      2
1440     2
840      1
1390     1
2380     1
Name: count, dtype: int64
--------------------
Value counts for Px_w:
Px_w
1980    15
2120    14
1920     4
1080     3
1760     3
1280     3
2520     3
460      1
560      1
720      1
1120     1
1820     1
Name: count, dtype: int64
--------------------
```

In [17]:
```
# Create a new feature called "Px" which stores the total resolution of the screen
df['Px'] = df['Px_h'] + df['Px_w']
```

In [18]:
```
df['Px'].value_counts()
```

Out[18]:
```
Px
4560    14
4700     7
5000     6
4500     4
3740     3
5400     3
2360     2
2720     2
4420     2
1240     1
1340     1
1560     1
2400     1
2470     1
4200     1
3260     1
Name: count, dtype: int64
```

In [19]:
```
# Create a histogram of the "Px" feature and also show the mean and the median
# Plotting the histogram
plt.figure(figsize=(10, 6))
plt.hist(df['Px'], bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram of Px Feature')
plt.xlabel('Total Resolution of the Screen (Px)')
plt.ylabel('Frequency')

# Adding mean and median lines
mean_value = df['Px'].mean()
median_value = df['Px'].median()
plt.axvline(mean_value, color='red', linestyle='dashed', linewidth=2, label=f'Mean:
plt.axvline(median_value, color='green', linestyle='dashed', linewidth=2, label=f'M
plt.legend()
plt.show()
```



The children want phones that have good screen resolutions

Consider the phones that have screen resolutions greater than or equal to the median value in the data set

Create a logical condition for this situation and store the logical values as "con2"

In [20]:
```
# Calculate the median screen resolution
median_resolution = df['Px'].median()

# Create the logical condition con2
con2 = df['Px'] >= median_resolution
```

In [21]:
```
con2.head()
```

Out[21]:
```
0    False
1    False
2    False
3    False
4    False
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 [22]:
```
# Let's tackle these features: "Scr_h", "Scr_w"
l3 = ['Scr_h','Scr_w']
```

In [23]:
```
count_values(df,l3)
```

```
Value counts for Scr_h:
Scr_h
8     20
6      9
5      8
4      6
10     4
2      2
3      1
Name: count, dtype: int64
--------------------
Value counts for Scr_w:
Scr_w
6    20
3    19
8     4
1     3
5     3
4     1
Name: count, dtype: int64
--------------------
```

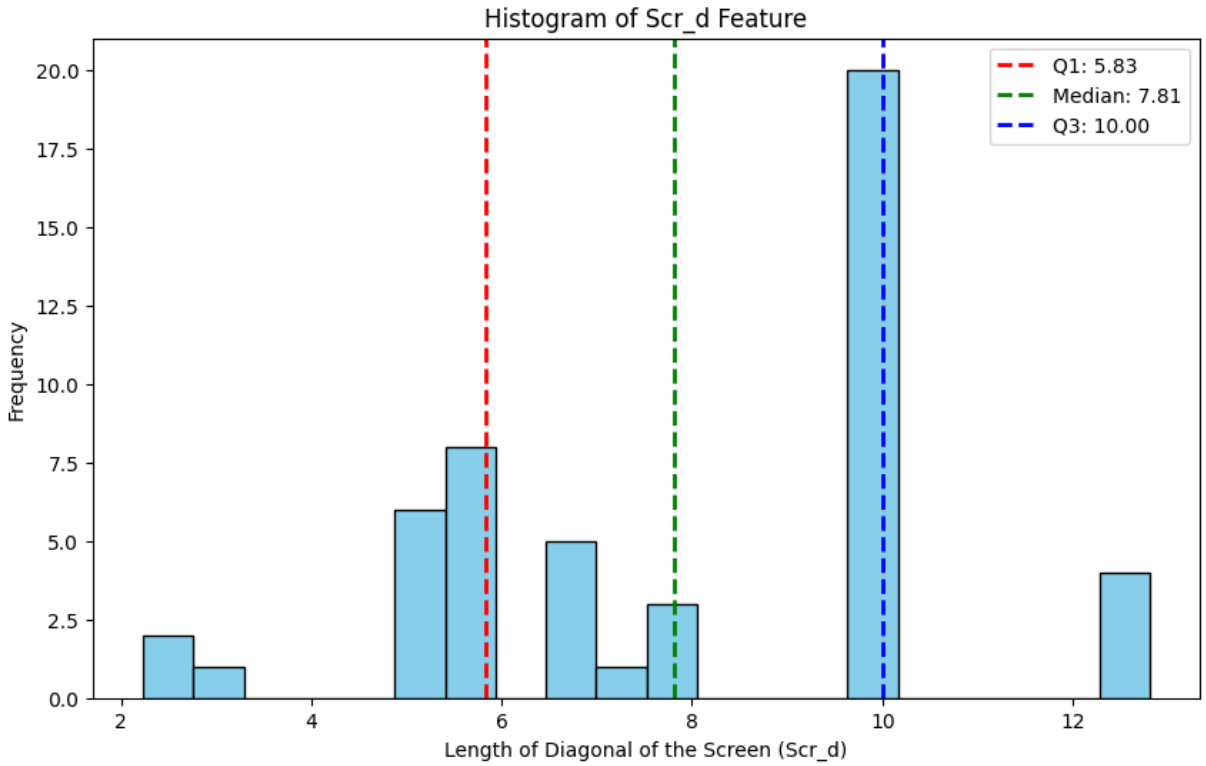
In [24]:
```
# Create a new feature called "Scr_d" which stores the length of the diagonal of th
df['Scr_d'] = np.sqrt(df['Scr_h']**2 + df['Scr_w']**2)
```

In [25]:
```
df['Scr_d'].head()
```

Out[25]:
```
0    3.162278
1    2.236068
2    2.236068
3    5.830952
4    5.000000
Name: Scr_d, dtype: float64
```

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

In [26]:
```
# Plotting the histogram
plt.figure(figsize=(10, 6))
plt.hist(df['Scr_d'], bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram of Scr_d Feature')
plt.xlabel('Length of Diagonal of the Screen (Scr_d)')
plt.ylabel('Frequency')

# Adding quartile lines
q25 = df['Scr_d'].quantile(0.25)
q50 = df['Scr_d'].quantile(0.50)
q75 = df['Scr_d'].quantile(0.75)
plt.axvline(q25, color='red', linestyle='dashed', linewidth=2, label=f'Q1: {q25:.2f
plt.axvline(q50, color='green', linestyle='dashed', linewidth=2, label=f'Median: {q
plt.axvline(q75, color='blue', linestyle='dashed', linewidth=2, label=f'Q3: {q75:.2
plt.legend()
plt.show()
```



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"

In [27]:
```
# Calculate the upper quartile value for the Scr_d feature
upper_quartile = df['Scr_d'].quantile(0.75)
```

```
# Create the logical condition con3
con3 = df['Scr_d'] >= upper_quartile
```

In [28]:
```
con3.head()
```

Out[28]:
```
0    False
1    False
2    False
3    False
4    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"

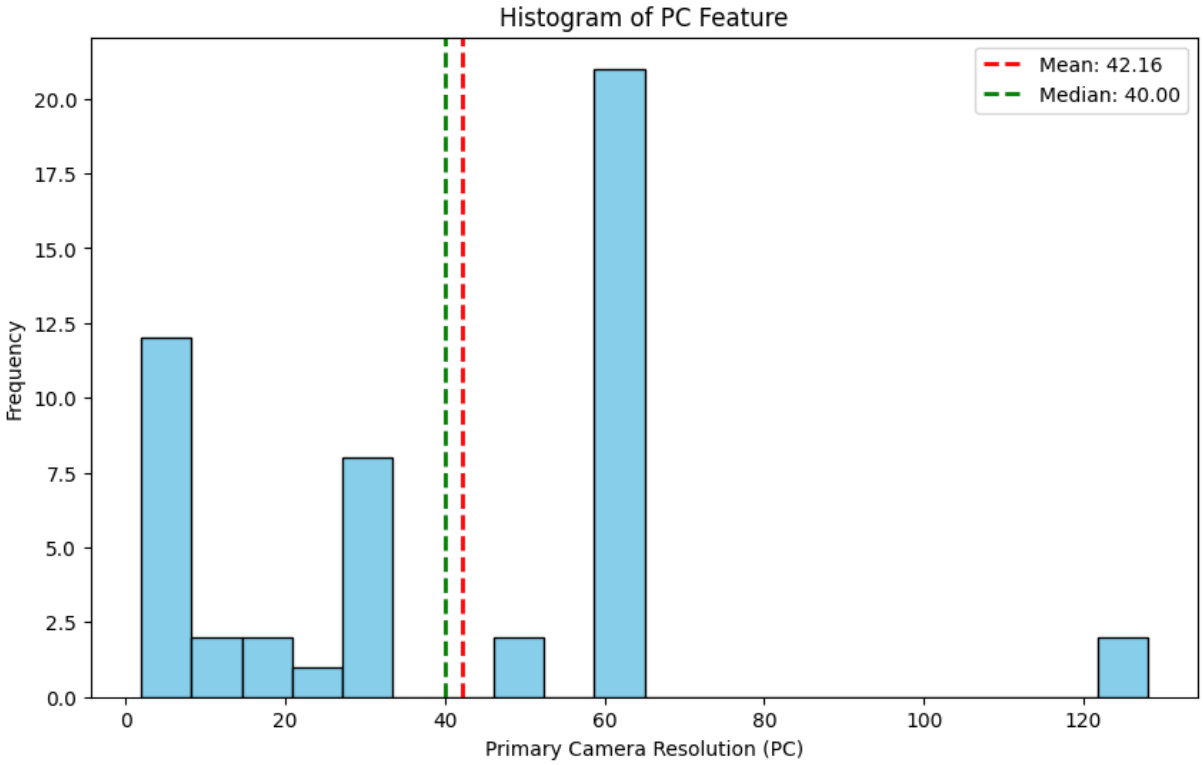
In [29]:
```
# Let's tackle these features: "PC", "FC"
l4 = ['PC','FC']
```

In [30]:
```
count_values(df,l4)
```

```
Value counts for PC:
PC
64     21
32      8
8       6
6       3
4       2
12      2
16      2
48      2
128     2
2       1
24      1
Name: count, dtype: int64
--------------------
Value counts for FC:
FC
32    20
16    12
4      6
2      5
8      4
64     2
12     1
Name: count, dtype: int64
--------------------
```

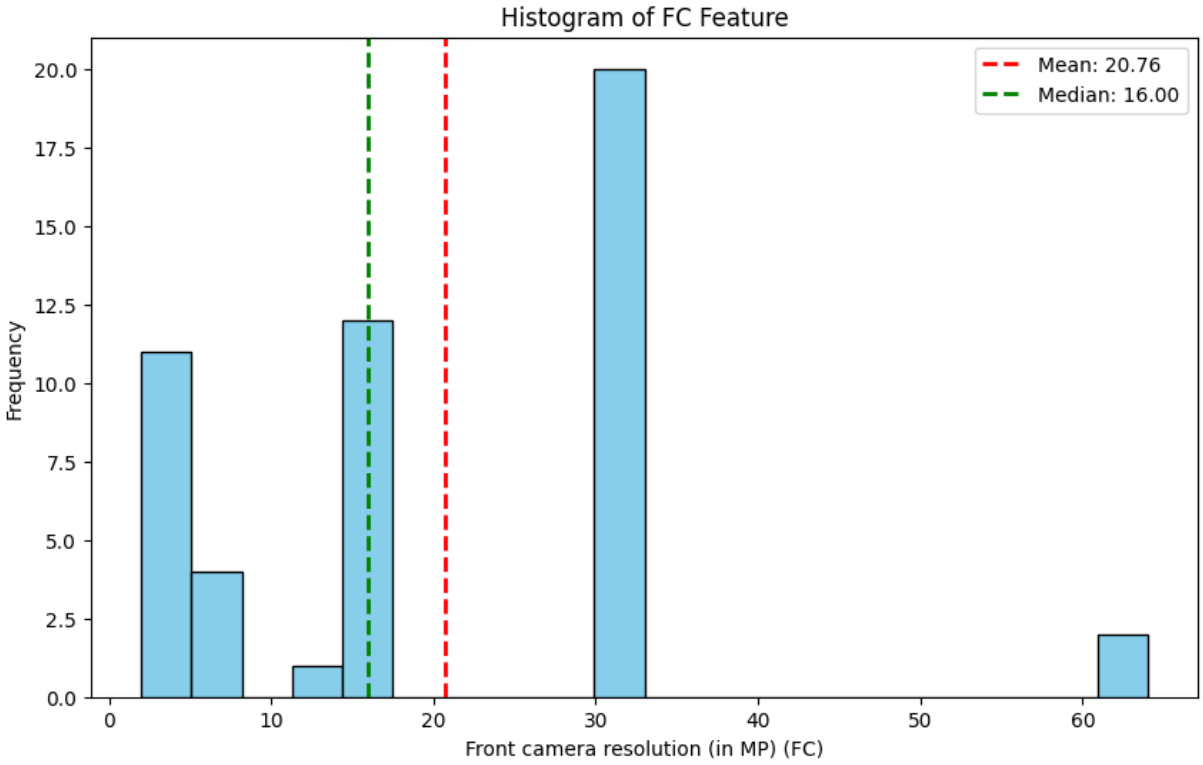
Create a histogram of the "PC" feature and also show the mean and the median

In [31]:
```python
# Plotting the histogram
plt.figure(figsize=(10, 6))
plt.hist(df['PC'], bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram of PC Feature')
plt.xlabel('Primary Camera Resolution (PC)')
plt.ylabel('Frequency')

# Adding mean and median lines
mean_value = df['PC'].mean()
median_value = df['PC'].median()
plt.axvline(mean_value, color='red', linestyle='dashed', linewidth=2, label=f'Mean:
plt.axvline(median_value, color='green', linestyle='dashed', linewidth=2, label=f'M
plt.legend()
plt.show()
```



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

In [32]:
```
# Plotting the histogram
plt.figure(figsize=(10, 6))
plt.hist(df['FC'], bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram of FC Feature')
plt.xlabel('Front camera resolution (in MP) (FC)')
plt.ylabel('Frequency')

# Adding mean and median lines
mean_value = df['FC'].mean()
median_value = df['FC'].median()
plt.axvline(mean_value, color='red', linestyle='dashed', linewidth=2, label=f'Mean:
plt.axvline(median_value, color='green', linestyle='dashed', linewidth=2, label=f'M
plt.legend()
plt.show()
```



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 equal to their respective mean values

Create a logical condition for this situation and store the logical values as "con4"

In [33]:
```
# Calculate the mean values for PC and FC features
mean_pc = df['PC'].mean()
mean_fc = df['FC'].mean()

# Create the logical condition con4
con4 = (df['PC'] >= mean_pc) & (df['FC'] >= mean_fc)
```

In [34]:
```
con4.head()
```

Out[34]:
```
0    False
1    False
2    False
3    False
4    False
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"

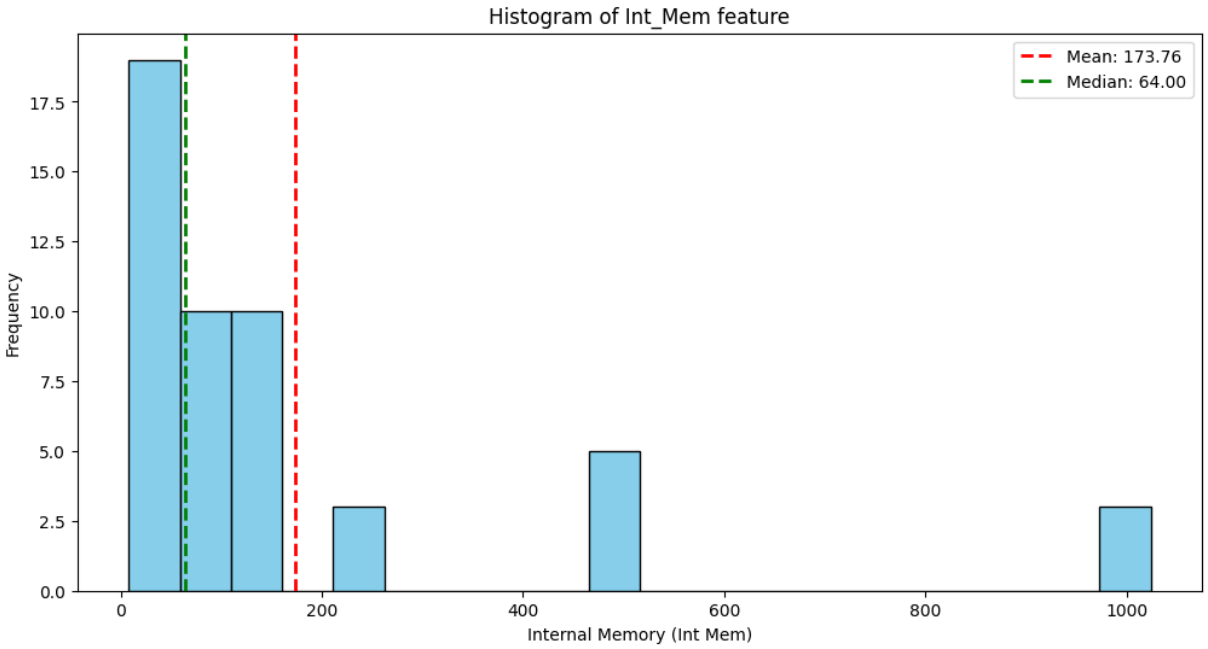
In [35]:
```
# Let's tackle these features: "Int_Mem", "Bty_Pwr", "RAM"
l5 = ["Int_Mem", "Bty_Pwr", "RAM"]
count_values(df,l5)
```

```
Value counts for Int_Mem:
Int_Mem
64      10
128     10
8        8
32       8
512      5
16       3
256      3
1024     3
Name: count, dtype: int64
--------------------
Value counts for Bty_Pwr:
Bty_Pwr
2800    13
3000     6
5600     4
4500     4
3300     3
4860     3
4800     2
4400     2
3900     2
3200     2
4000     1
4380     1
4300     1
4200     1
5000     1
3400     1
2980     1
4560     1
2300     1
Name: count, dtype: int64
--------------------
Value counts for RAM:
RAM
8     18
4     17
2      8
6      4
12     3
Name: count, dtype: int64
--------------------
```

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

In [36]:
```
plt.figure(figsize=[12,6])
plt.hist(df['Int_Mem'], bins=20,color='skyblue',edgecolor='black')
plt.title('Histogram of Int_Mem feature')
plt.xlabel('Internal Memory (Int Mem)')
plt.ylabel('Frequency')

mean_value = df['Int_Mem'].mean()
```

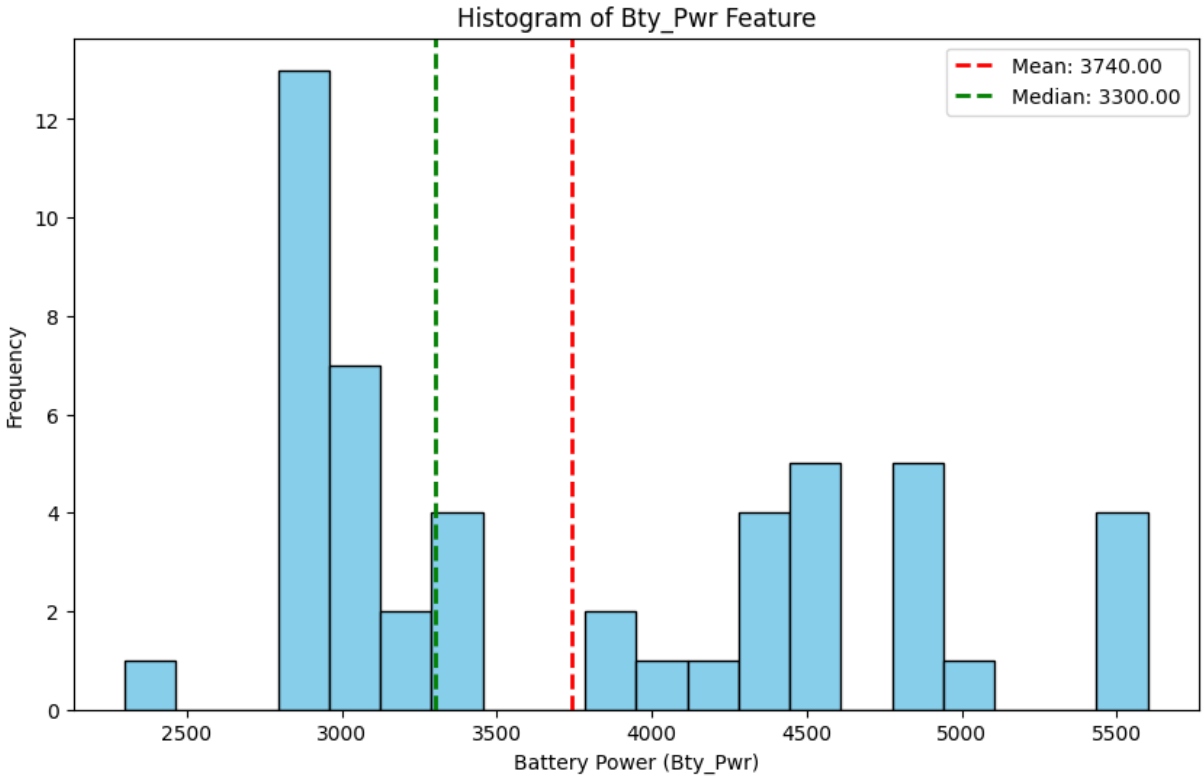
```
median_value = df['Int_Mem'].median()
plt.axvline(mean_value, color='red', linestyle='dashed', linewidth=2, label=f'Mean:
plt.axvline(median_value, color='green', linestyle='dashed', linewidth=2, label=f'M
plt.legend()
plt.show()
```



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

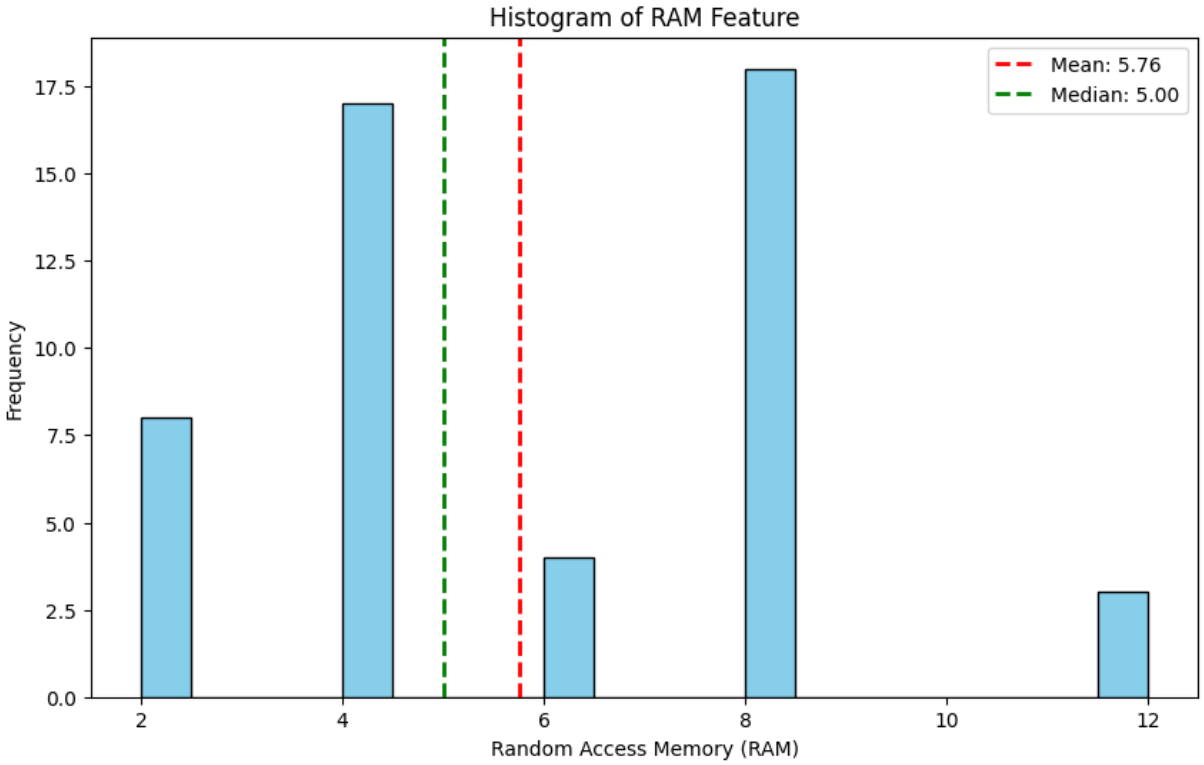
In [37]:

```
# Plotting the histogram
plt.figure(figsize=(10, 6))
plt.hist(df['Bty_Pwr'], bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram of Bty_Pwr Feature')
plt.xlabel('Battery Power (Bty_Pwr)')
plt.ylabel('Frequency')

# Adding mean and median lines
mean_value = df['Bty_Pwr'].mean()
median_value = df['Bty_Pwr'].median()
plt.axvline(mean_value, color='red', linestyle='dashed', linewidth=2, label=f'Mean:
plt.axvline(median_value, color='green', linestyle='dashed', linewidth=2, label=f'M
plt.legend()
plt.show()
```



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

In [38]:
```
# Plotting the histogram
plt.figure(figsize=(10, 6))
plt.hist(df['RAM'], bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram of RAM Feature')
plt.xlabel('Random Access Memory (RAM)')
plt.ylabel('Frequency')

# Adding mean and median lines
mean_value = df['RAM'].mean()
median_value = df['RAM'].median()
plt.axvline(mean_value, color='red', linestyle='dashed', linewidth=2, label=f'Mean:
plt.axvline(median_value, color='green', linestyle='dashed', linewidth=2, label=f'M
plt.legend()
plt.show()
```



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"

In [39]:
```
# Calculate the mean values for Int_Mem, Bty_Pwr, and RAM features
mean_int_mem = df['Int_Mem'].mean()
mean_bty_pwr = df['Bty_Pwr'].mean()
mean_ram = df['RAM'].mean()

# Create the logical condition con5
con5 = (df['Int_Mem'] >= mean_int_mem) & (df['Bty_Pwr'] >= mean_bty_pwr) & (df['RAM
```

In [40]:
```
con5.head()
```

Out[40]:
```
0    False
1    False
2    False
3    False
4    False
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

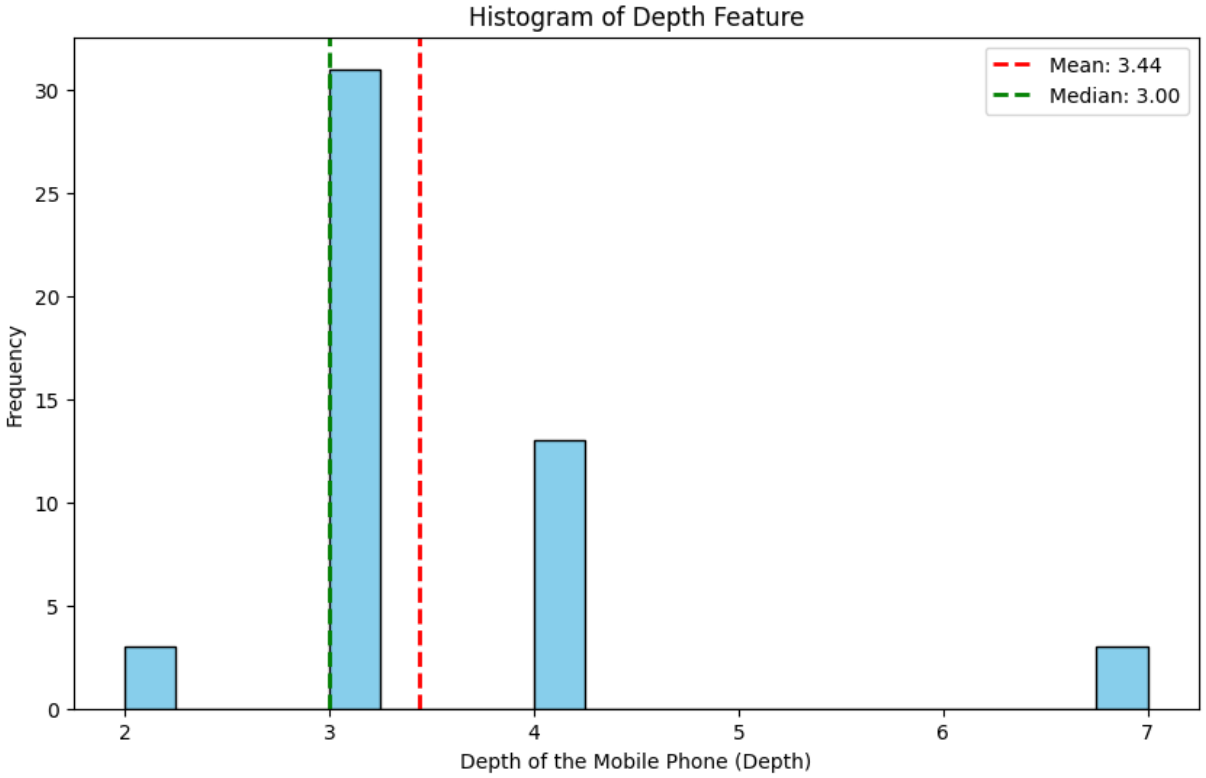
Task 7 - Obtain the logical conditions for the features "Depth" and "Weight"

In [41]:
```
# Let's tackle these features: "Depth", "Weight"
l6 = ["Depth", "Weight"]
count_values(df,l6)
```

```
Value counts for Depth:
Depth
3    31
4    13
7     3
2     3
Name: count, dtype: int64
--------------------
Value counts for Weight:
Weight
210    5
150    5
120    4
160    3
200    3
80     3
180    2
110    2
170    2
250    2
320    2
270    2
230    2
300    2
280    1
100    1
330    1
220    1
90     1
400    1
240    1
310    1
140    1
260    1
130    1
Name: count, dtype: int64
--------------------
```

Create a histogram of the "Depth" feature and also show the mean and the median

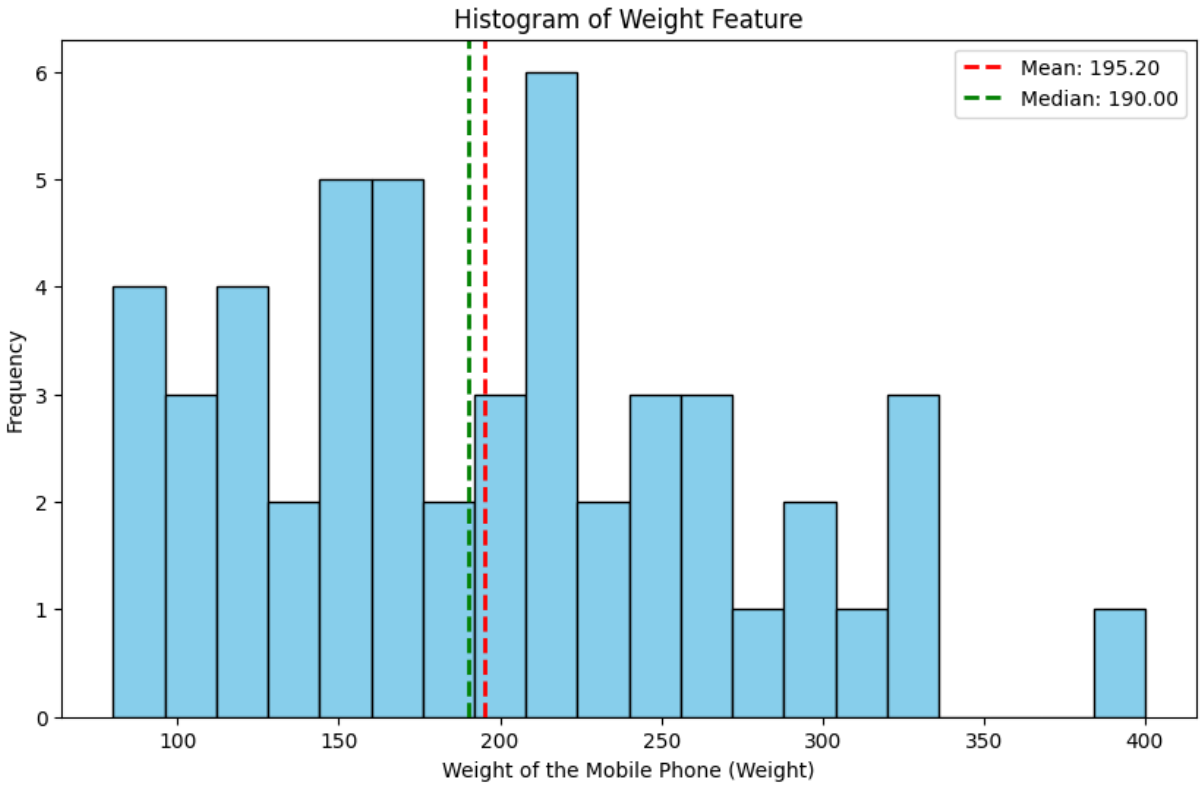
In [42]:
```
# Plotting the histogram
plt.figure(figsize=(10, 6))
plt.hist(df['Depth'], bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram of Depth Feature')
plt.xlabel('Depth of the Mobile Phone (Depth)')
plt.ylabel('Frequency')

# Adding mean and median lines
mean_value = df['Depth'].mean()
median_value = df['Depth'].median()
plt.axvline(mean_value, color='red', linestyle='dashed', linewidth=2, label=f'Mean:
plt.axvline(median_value, color='green', linestyle='dashed', linewidth=2, label=f'M
plt.legend()
plt.show()
```



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

In [43]:
```
# Plotting the histogram
plt.figure(figsize=(10, 6))
plt.hist(df['Weight'], bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram of Weight Feature')
plt.xlabel('Weight of the Mobile Phone (Weight)')
plt.ylabel('Frequency')

# Adding mean and median lines
```

```
mean_value = df['Weight'].mean()
median_value = df['Weight'].median()
plt.axvline(mean_value, color='red', linestyle='dashed', linewidth=2, label=f'Mean:
plt.axvline(median_value, color='green', linestyle='dashed', linewidth=2, label=f'M
plt.legend()
plt.show()
```



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 median values in the data set

Create a logical condition for this situation and store the logical values as "con6"

In [44]:
```
# Calculate the median values for Depth and Weight features
median_depth = df['Depth'].median()
median_weight = df['Weight'].median()

# Create the logical condition con6
con6 = (df['Depth'] <= median_depth) & (df['Weight'] <= median_weight)
```

In [45]:
```
con6.head()
```

Out[45]:
```
0    False
1    False
2    False
3    False
4    False
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 [46]:
```
# Subset the DataFrame using logical conditions
df1 = df[con1 & con2 & con3 & con4 & con5 & con6]
```

In [47]:
```
df1.head()
```

Out[47]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Men
30	TVF078Y	yes	yes	yes	yes	2580	2120	8	6	64	32	51
32	TYS938L	yes	yes	yes	yes	2580	2120	8	6	64	32	102
42	WZB298K	yes	yes	yes	yes	2580	1980	8	6	64	32	102

In [48]:
```
# Get the dimensions of the dataframe
df1.shape
```

Out[48]:
```
(3, 19)
```

In [49]:
```
# Sort the dataframe according to the "Price" feature in ascending order and displa
df1_sorted = df1.sort_values(by='Price',ascending=True)
```

In [50]:
```
df1_sorted.head()
```

Out[50]:

	PID	Blue	Wi_Fi	Tch_Scr	Ext_Mem	Px_h	Px_w	Scr_h	Scr_w	PC	FC	Int_Men
30	TVF078Y	yes	yes	yes	yes	2580	2120	8	6	64	32	51
42	WZB298K	yes	yes	yes	yes	2580	1980	8	6	64	32	102
32	TYS938L	yes	yes	yes	yes	2580	2120	8	6	64	32	102

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 set

Calculate the ratio of the standard deviation to the mean for all the numerical features in the dataframe

Store these values in a new series wherein the rows are the features and the only column is the calculated ratio

In [51]:
```
# Calculate the ratio of standard deviation to mean for all numerical features
deviations = (df1_sorted.select_dtypes(include='number').std() / df1_sorted.select_
```

In [52]:
```
deviations.head()
```

Out[52]:
```
Px_h     0.000000
Px_w     0.038985
Scr_h    0.000000
Scr_w    0.000000
PC       0.000000
Name: Ratio, dtype: float64
```

In [53]:
```
# Sort the "deviations" Series in descending order
deviations_sorted = deviations.sort_values(ascending=False)
```

In [54]:
```
deviations_sorted.head()
```

Out[54]:
```
Int_Mem    0.346410
Weight     0.284747
Price      0.217569
Bty_Pwr    0.083663
Px_w       0.038985
Name: Ratio, dtype: float64
```

Observations

- The ratio of the standard deviation to the mean of a feature normalizes it, allowing 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 highly 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 applied descriptive statistics concepts to analyze and work with a dataset containing mobile phone specifications.

2. Based on the analysis, we recommend three phone models to the client, which she can propose to her children.

In [ ]: