Capstone Project:3 - Exploratory Data Analysis Projects(EDA)

Project 4 -Mobile Price Analysis

Modules needed: Numpy,pandas,matplotlib,seaborn.

```
#import the needed libraries:
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
#Loading the mobile dataset file:
df= pd.read_csv("/content/train.csv")
print(df)
```

```
      battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
0               842     0          2.2         0   1       0           7   
1              1021     1          0.5         1   0       1          53   
2               563     1          0.5         1   2       1          41   
3               615     1          2.5         0   0       0          10   
4              1821     1          1.2         0  13       1          44   
...             ...   ...          ...       ...  ..     ...         ...   
1995            794     1          0.5         1   0       1           2   
1996           1965     1          2.6         1   0       0          39   
1997           1911     0          0.9         1   1       1          36   
1998           1512     0          0.9         0   4       1          46   
1999            510     1          2.0         1   5       1          45   

      m_dep  mobile_wt  n_cores  ...  px_height  px_width   ram  sc_h  sc_w  \
0       0.6        188        2  ...         20       756  2549     9     7   
1       0.7        136        3  ...        905      1988  2631    17     3   
2       0.9        145        5  ...       1263      1716  2603    11     2   
3       0.8        131        6  ...       1216      1786  2769    16     8   
4       0.6        141        2  ...       1208      1212  1411     8     2   
...     ...        ...      ...  ...        ...       ...   ...   ...   ...   
1995    0.8        106        6  ...       1222      1890   668    13     4   
1996    0.2        187        4  ...        915      1965  2032    11    10   
1997    0.7        108        8  ...        868      1632  3057     9     1   
1998    0.1        145        5  ...        336       670   869    18    10   
1999    0.9        168        6  ...        483       754  3919    19     4   

      talk_time  three_g  touch_screen  wifi  price_range  
0            19        0             0     1            1  
1             7        1             1     0            2  
2             9        1             1     0            2  
3            11        1             0     0            2  
4            15        1             1     0            1  
...         ...      ...           ...   ...          ...  
1995         19        1             1     0            0  
1996         16        1             1     1            2  
1997          5        1             1     0            3  
1998         19        1             1     1            0  
1999          2        1             1     1            3  

[2000 rows x 21 columns]
```

```
#First 5 rows of the data:
df.head()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0	2
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0	2
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1	0	0	2
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1	1	0	1

5 rows × 21 columns

```
#Last 5 rows of the data:
df.tail()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
1995	794	1	0.5	1	0	1	2	0.8	106	6	...	1222	1890	668	13	4	19	1	1	0	0
1996	1965	1	2.6	1	0	0	39	0.2	187	4	...	915	1965	2032	11	10	16	1	1	1	2
1997	1911	0	0.9	1	1	1	36	0.7	108	8	...	868	1632	3057	9	1	5	1	1	0	3
1998	1512	0	0.9	0	4	1	46	0.1	145	5	...	336	670	869	18	10	19	1	1	1	0
1999	510	1	2.0	1	5	1	45	0.9	168	6	...	483	754	3919	19	4	2	1	1	1	3

5 rows × 21 columns

```
#Shape of the dataset:
df.shape
```

```
(2000, 21)
```

```
#Description of the dataset:
df.describe()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	ta
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	...	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	200
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	0.501750	140.249000	4.520500	...	645.108000	1251.515500	2124.213000	12.306500	5.767000	1
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.399655	2.287837	...	443.780811	432.199447	1084.732044	4.213245	4.356398	
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000	1.000000	...	0.000000	500.000000	256.000000	5.000000	0.000000	
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.000000	3.000000	...	282.750000	874.750000	1207.500000	9.000000	2.000000	
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.000000	4.000000	...	564.000000	1247.000000	2146.500000	12.000000	5.000000	1
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.000000	7.000000	...	947.250000	1633.000000	3064.500000	16.000000	9.000000	1
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000	8.000000	...	1960.000000	1998.000000	3998.000000	19.000000	18.000000	2

8 rows × 21 columns

```
#Cleaning the data for missing values and null values:
#Checking for null values:
df.isnull().sum()
```

	0
battery_power	0
blue	0
clock_speed	0
dual_sim	0
fc	0
four_g	0
int_memory	0
m_dep	0
mobile_wt	0
n_cores	0
pc	0
px_height	0
px_width	0
ram	0
sc_h	0
sc_w	0
talk_time	0
three_g	0
touch_screen	0
wifi	0
price_range	0

dtype: int64

```
#Checking for missing and null values:
null_df = df[df.isna().any(axis=1)]
null_df
```

battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range

0 rows × 21 columns

```
#Checking for missing and null values:
df.isnull().any().any()
```

```
False
```

```
#Info of the dataset:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2000 non-null   int64  
 1   blue           2000 non-null   int64  
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64  
 4   fc             2000 non-null   int64  
 5   four_g         2000 non-null   int64  
 6   int_memory     2000 non-null   int64  
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64  
 9   n_cores        2000 non-null   int64  
 10  pc             2000 non-null   int64  
 11  px_height      2000 non-null   int64  
 12  px_width       2000 non-null   int64  
 13  ram            2000 non-null   int64  
 14  sc_h           2000 non-null   int64  
 15  sc_w           2000 non-null   int64  
 16  talk_time      2000 non-null   int64  
 17  three_g        2000 non-null   int64  
 18  touch_screen   2000 non-null   int64  
 19  wifi           2000 non-null   int64  
 20  price_range    2000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

```
#Printing the column names:
df.columns
```

```
Index(['battery_power', 'blue', 'clock_speed', 'dual_sim', 'fc', 'four_g',
       'int_memory', 'm_dep', 'mobile_wt', 'n_cores', 'pc', 'px_height',
       'px_width', 'ram', 'sc_h', 'sc_w', 'talk_time', 'three_g',
       'touch_screen', 'wifi', 'price_range'],
      dtype='object')
```

```
#Changing the column names:
df=df.rename(columns={"blue":"bluetooth","fc":"front_camera","pc": "Primary Camera mega pixels"})
print("Column Names has Updated Successfully")
```

```
Column Names has Updated Successfully
```

```
#First 3 rows of the data:
df.head(3)
```

	battery_power	bluetooth	clock_speed	dual_sim	front_camera	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0

3 rows × 21 columns

```
#Printing the column names:
df.columns
```

```
Index(['battery_power', 'bluetooth', 'clock_speed', 'dual_sim', 'front_camera',
       'four_g', 'int_memory', 'm_dep', 'mobile_wt', 'n_cores',
       'Primary Camera mega pixels', 'px_height', 'px_width', 'ram', 'sc_h',
       'sc_w', 'talk_time', 'three_g', 'touch_screen', 'wifi', 'price_range'],
      dtype='object')
```

```
#Changing the data:
mobile_data = pd.DataFrame()
mobile_data["bluetooth"] = np.where(df["bluetooth"]<1,"No","yes")
mobile_data["dual_sim"] = np.where(df["dual_sim"]<1,"No","yes")
```

```
mobile_data["four_g"] = np.where(df["four_g"]<1,"No","yes")
mobile_data["three_g"] = np.where(df["three_g"]<1,"No","yes")
mobile_data["touch_screen"] = np.where(df["touch_screen"]<1,"No","yes")
mobile_data["wifi"] = np.where(df["wifi"]<1,"No","yes")
print("Column Data has updated Successfully")
```

```
Column Data has updated Successfully
```

```
#First 3 rows of the data:
mobile_data.head(3)
```

	bluetooth	dual_sim	four_g	three_g	touch_screen	wifi
0	No	No	No	No	No	yes
1	yes	yes	yes	yes	yes	No
2	yes	yes	yes	yes	yes	No

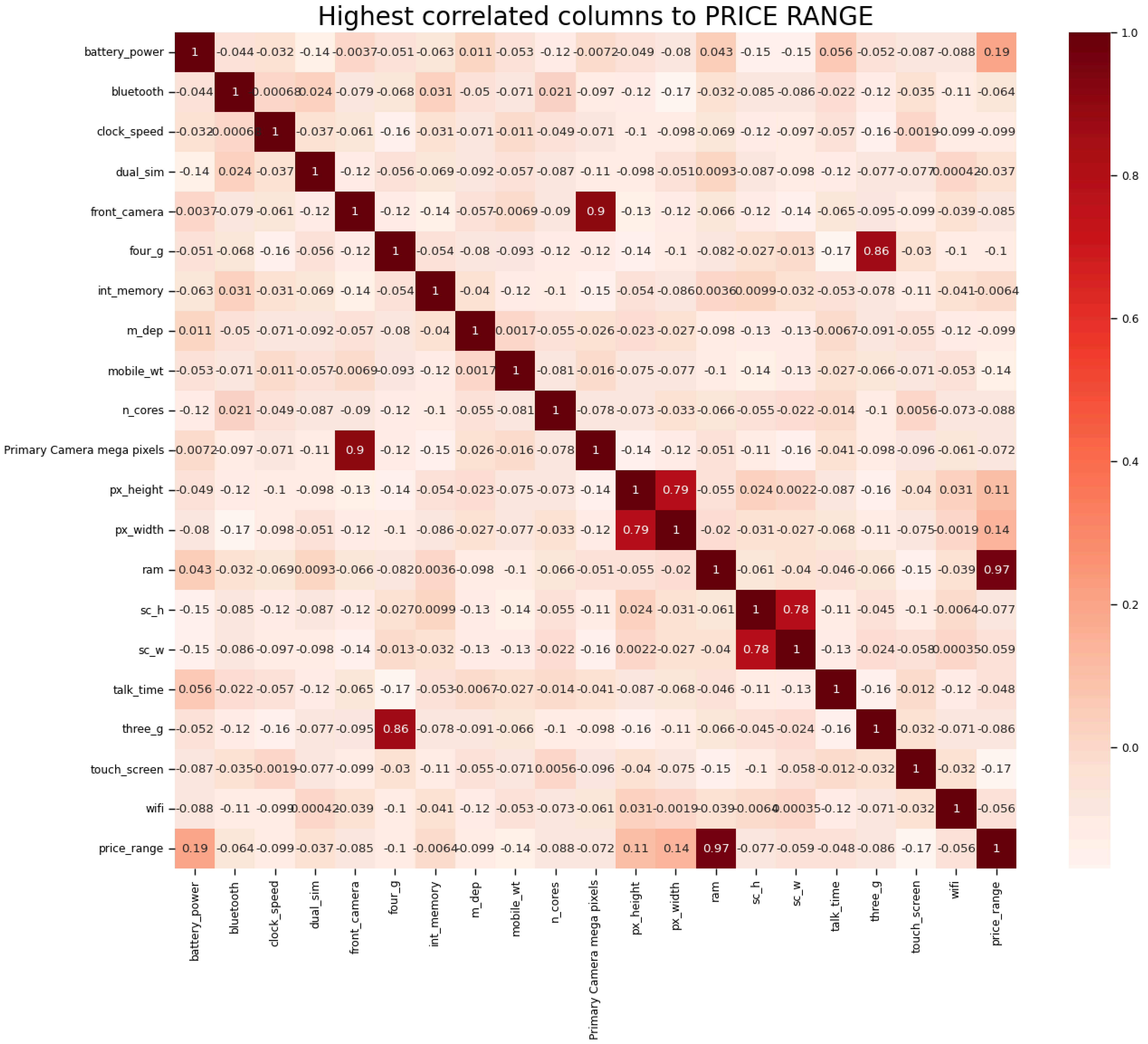
```
#First 3 rows of the data:
df.head(3)
```

	battery_power	bluetooth	clock_speed	dual_sim	front_camera	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0

3 rows × 21 columns

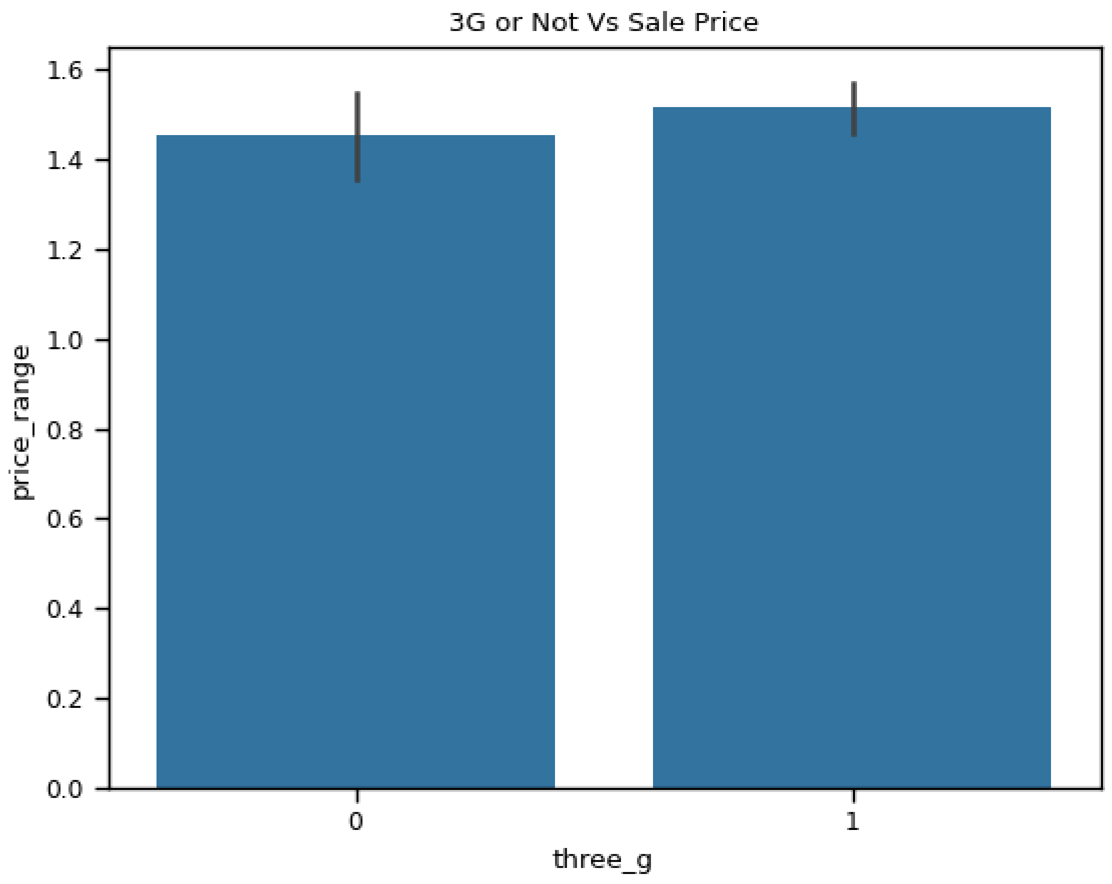
VISUALIZATION:

```
#Highest Correlated columns to PRICE RANGE:
plt.figure(figsize=(15, 12))
sns.set_context('paper')
mobile_corr=df.corr(numeric_only=True)
sns.heatmap(mobile_corr.corr(),cmap='Reds',annot=True)
plt.title(" Highest correlated columns to PRICE RANGE ",fontsize=20)
plt.show()
```

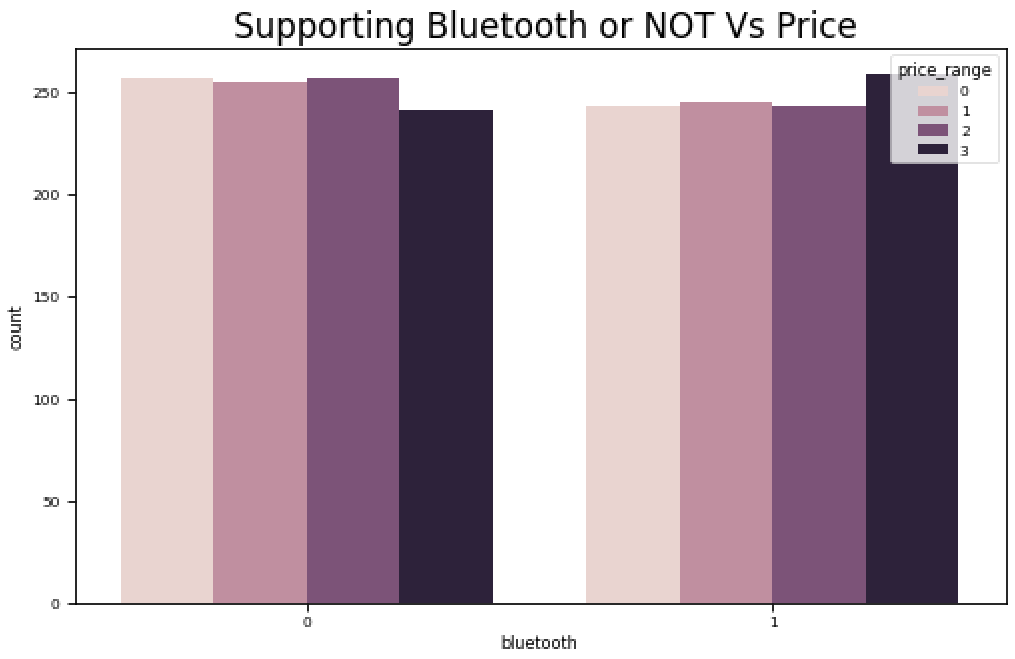


Highest correlated columns to PRICE RANGE

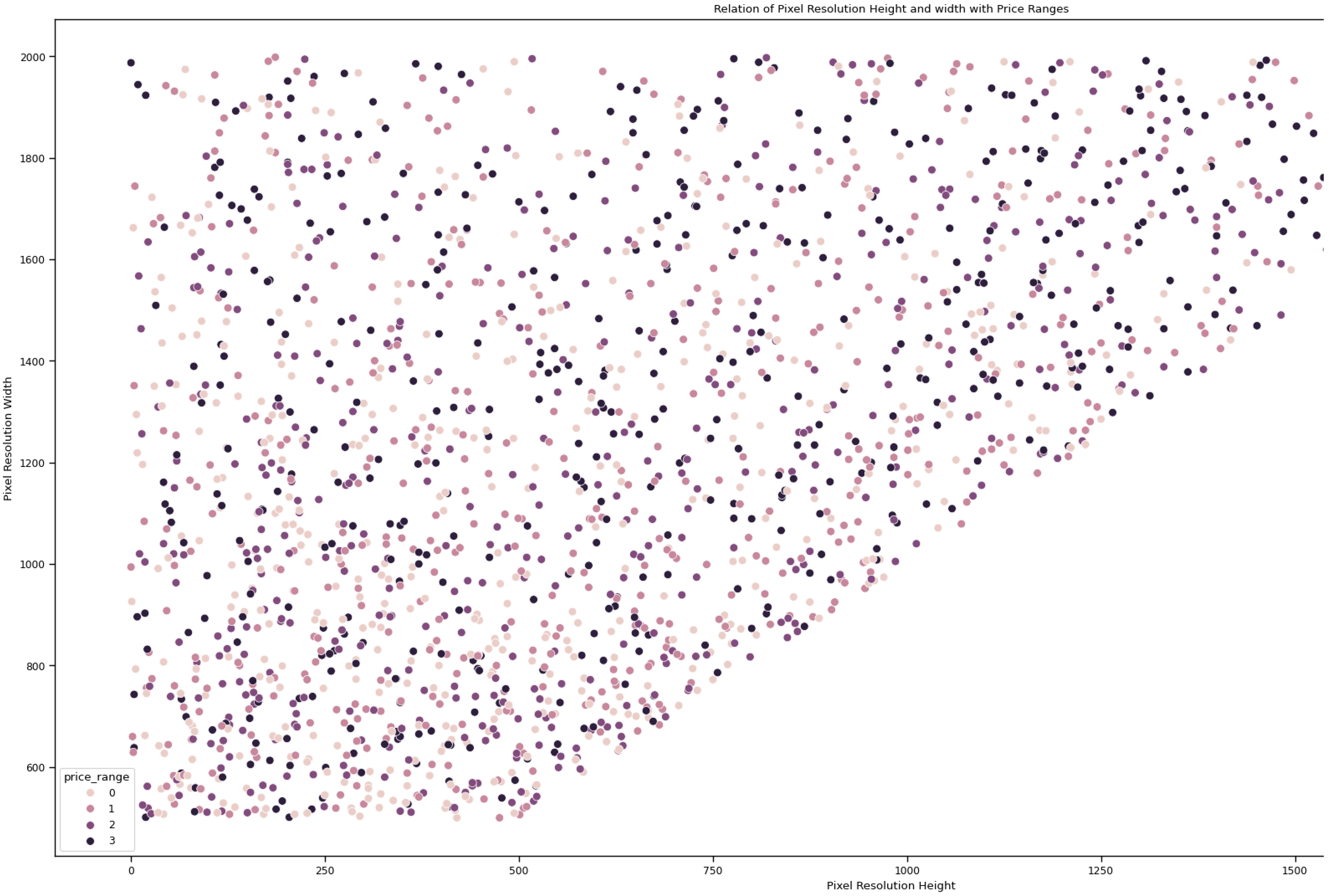
```
#3G or Not 3G Mobile VS Sale Price:
sns.barplot(x='three_g', y='price_range', data=df)
plt.title('3G or Not Vs Sale Price')
plt.show()
```



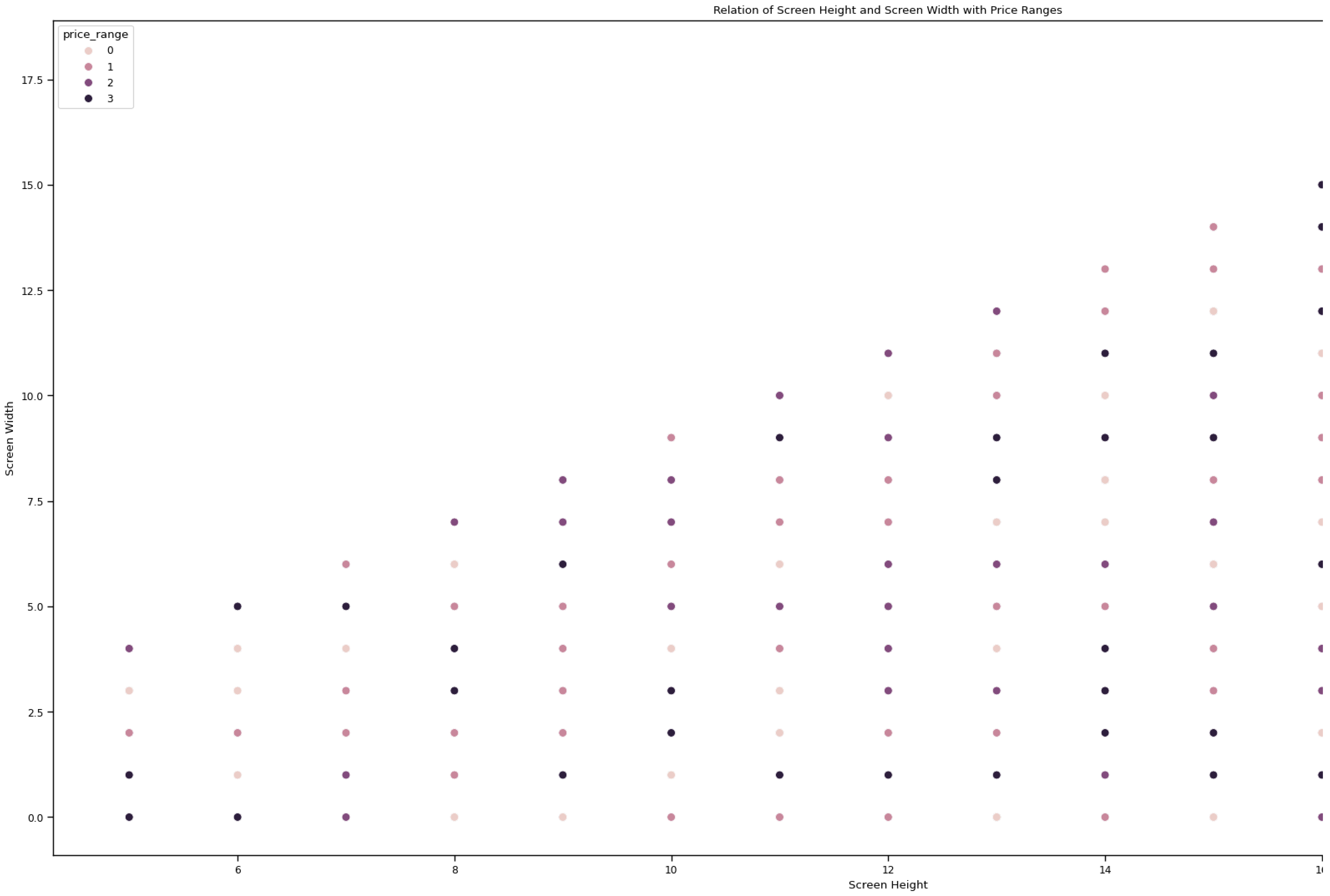
```
#Count Plot For Supporting Bluetooth or NOT Vs Price:
plt.figure(figsize = (10,6),dpi=60)
sns.countplot(data=df, x="bluetooth", hue="price_range")
plt.title(" Supporting Bluetooth or NOT Vs Price",fontsize=20)
plt.show()
```



```
# Relation of Pixel Resolution Height and Pixel Resolution Width with Price Ranges
fig=plt.figure(figsize=(20,10))
ax=fig.add_axes([0,0,1,1])
sns.scatterplot(x = "px_height", y = "px_width", data = df, hue = "price_range",s=50)
ax.set(xlabel = "Pixel Resolution Height" , ylabel = "Pixel Resolution Width")
ax.set(title = "Relation of Pixel Resolution Height and width with Price Ranges")
plt.show()
```



```
# Relation of Screen Height and Screen Width with Price Ranges
fig=plt.figure(figsize=(20,10))
ax=fig.add_axes([0,0,1,1])
sns.scatterplot(x = "sc_h", y = "sc_w", data = df, hue = "price_range",s=50)
ax.set(xlabel = "Screen Height",ylabel = "Screen Width")
ax.set(title = "Relation of Screen Height and Screen Width with Price Ranges")
plt.show()
```



```
#Changing of data:
df["price_range"].replace({1:"Low Cost", 2:"Medium Cost", 3:"High Cost", 4:"Very High Cost"}, inplace=True)
print("Column Data has updated Successfully")
```

```
Column Data has updated Successfully
```

```
#First five rows of the data:
df.head()
```

	battery_power	bluetooth	clock_speed	dual_sim	front_camera	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1	0	0
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1	1	0

5 rows × 21 columns