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

#1.Loading the dataset
train_df=pd.read_csv('train.csv')
#2.Print first 5 rows of data
train_df.head(5)
```

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

5 rows × 21 columns

```
#3.Print last 5 rows of data
train_df.tail(5)
```

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

5 rows × 21 columns

```
#4.Shape of the dataset
train_df.shape
```

```
(2000, 21)
```

```
#5.Description of the Dataset
train_df.describe()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	0.501750	140.249000	4.520500
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.399655	2.287837
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000	1.000000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.000000	3.000000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.000000	4.000000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.000000	7.000000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000	8.000000

8 rows × 21 columns

```
#6. Cleaning the data for missing values, null values
train_df.isnull().sum()
train_df=train_df.dropna()
```

```
#7. Info on the dataset
train_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
```

```
#8. Change Column Names and Data
train_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')
```

```
#9.In the following columns : bluetooth, dual_sim, four_g, three_g, touch_screen, wifi
#change (0-No, 1-Yes)
columns_to_change = ['blue', 'dual_sim', 'four_g', 'three_g', 'touch_screen', 'wifi']
for column in columns_to_change:
    train_df[column] = train_df[column].replace({0: 'No', 1: 'Yes'})
```

```
train_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_
0	842	No	2.2	No	1	No	7	0.6	188	2	...	20	756	2549	
1	1021	Yes	0.5	Yes	0	Yes	53	0.7	136	3	...	905	1988	2631	1
2	563	Yes	0.5	Yes	2	Yes	41	0.9	145	5	...	1263	1716	2603	1
3	615	Yes	2.5	No	0	No	10	0.8	131	6	...	1216	1786	2769	1
4	1821	Yes	1.2	No	13	Yes	44	0.6	141	2	...	1208	1212	1411	

5 rows × 21 columns

```
#10.In price_range column change Low Cost-1,Medium Cost-2, High Cost-3 and check
#the data changed or not
train_df['price_range'] = train_df['price_range'].replace({'Low Cost': 1, 'Medium Cost': 2, 'High Cost': 3})
```

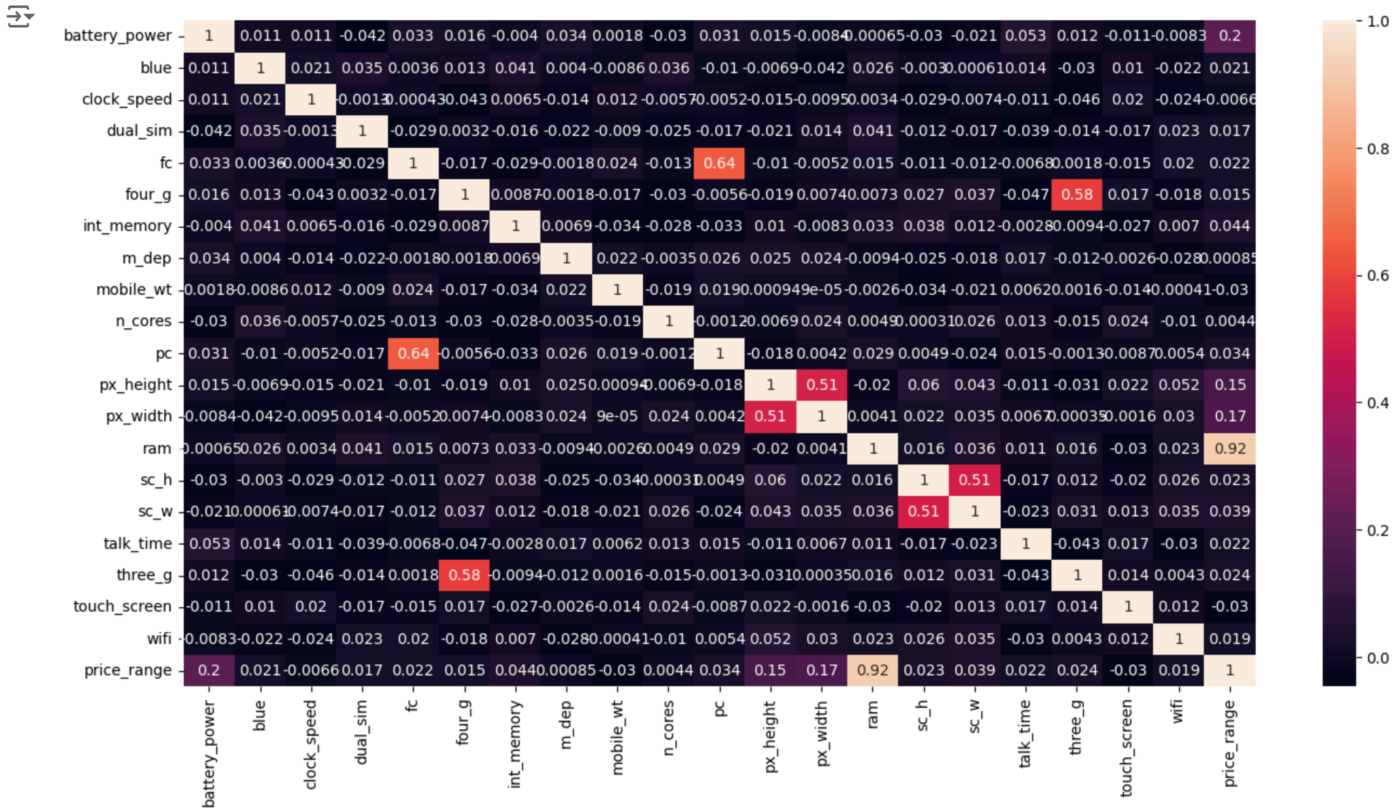
```
train_df['price_range'].unique()
```

```
array([1, 2, 3, 0])
```

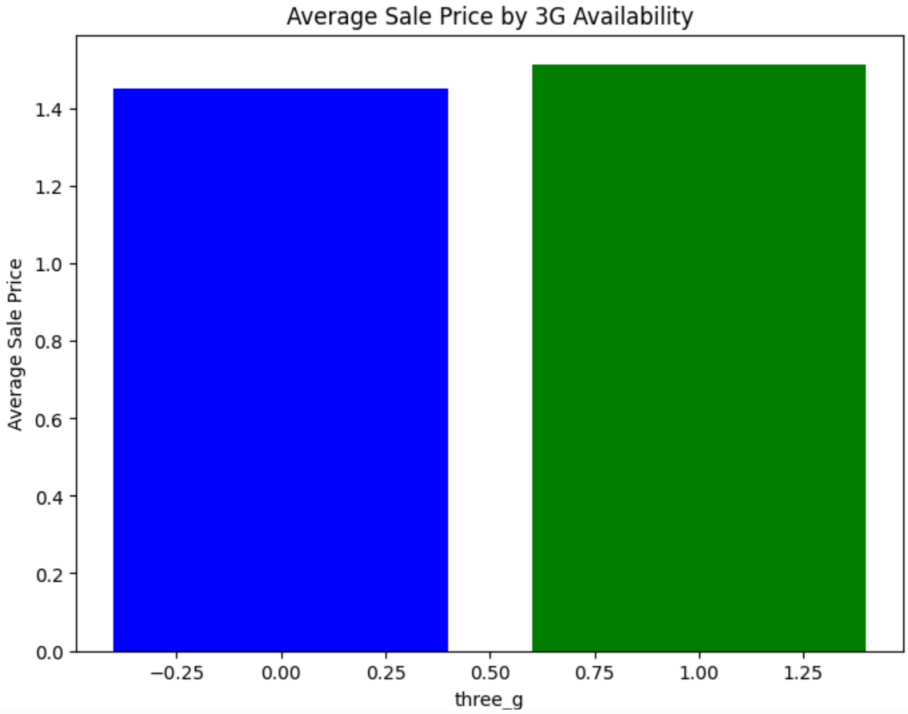
VISUALIZATION

```
#1.Get the highest correlated columns to PRICE RANGE
corr_matrix = train_df.corr()
corr_matrix['price_range'].sort_values(ascending=False)
```

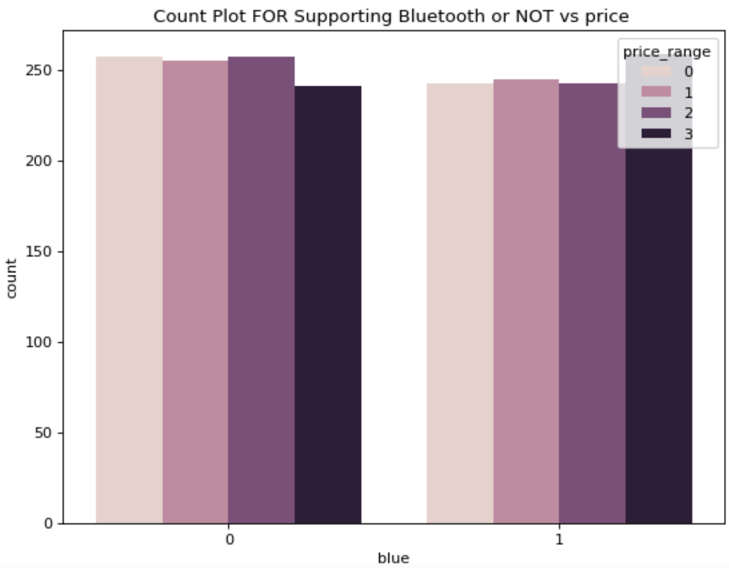
```
plt.figure(figsize=(16,8))
sns.heatmap(corr_matrix,annot=True)
plt.show()
```




```
#2. Show the 3G or Not 3G Mobile VS Sale Price using bar plot
avg_3g = train_df.groupby('three_g')['price_range'].mean().reset_index()
plt.figure(figsize=(8, 6))
plt.bar(avg_3g['three_g'], avg_3g['price_range'], color=['blue', 'green'])
plt.xlabel('three_g')
plt.ylabel('Average Sale Price')
plt.title('Average Sale Price by 3G Availability')
plt.show()
```

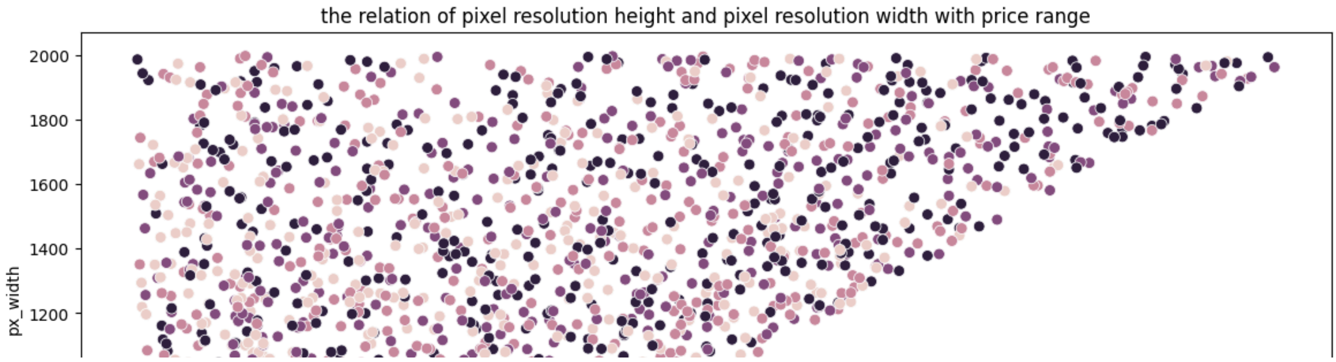



```
#3. Show the Count Plot FOR Supporting Bluetooth or NOT vs price
plt.figure(figsize=(8,6),dpi=80)
sns.countplot(data=train_df,x='blue',hue='price_range').set(title='Count Plot FOR Supporting Bluetooth or NOT vs price
plt.show()
```




```
#Use scatterplot to show the relation of pixel resolution height and pixel resolution
#width with price range.
plt.figure(figsize=(14,6))
plt.title("the relation of pixel resolution height and pixel resolution width with price range")
sns.scatterplot(x=train_df['px_height'],y=train_df['px_width'],hue=train_df['price_range'],s=50)
```

<Axes: title={'center': 'the relation of pixel resolution height and pixel resolution width with price range'}, xlabel='px_height', ylabel='px_width'>




```
#5.Use scatterplot to show the relation of Screen Height and Screen Width with Price
#Ranges.
plt.figure(figsize=(14,6))
plt.title("how the relation of Screen Height and Screen Width with Price Ranges")
sns.scatterplot(x=train_df['sc_h'],y=train_df['sc_w'],hue=train_df['price_range'],s=50)
```

<Axes: title={'center': 'how the relation of Screen Height and Screen Width with Price Ranges'}, xlabel='sc_h', ylabel='sc_w'>