importing the libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Read the Data

In [2]:
```
df=pd.read_csv(r"C:/Users/DD/Desktop/Mobile Prices.csv")
```

In [3]:
```
df
```

Out[3]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	r
0	842	0	2.2	0	1	0	7	0.6	188	
1	1021	1	0.5	1	0	1	53	0.7	136	
2	563	1	0.5	1	2	1	41	0.9	145	
3	615	1	2.5	0	0	0	10	0.8	131	
4	1821	1	1.2	0	13	1	44	0.6	141	
...	...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	1	0	1	2	0.8	106	
1996	1965	1	2.6	1	0	0	39	0.2	187	
1997	1911	0	0.9	1	1	1	36	0.7	108	
1998	1512	0	0.9	0	4	1	46	0.1	145	
1999	510	1	2.0	1	5	1	45	0.9	168	

2000 rows × 21 columns

Data Preprocessing

In [4]:
```
df.head()
```

Out[4]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cc
0	842	0	2.2	0	1	0	7	0.6	188	
1	1021	1	0.5	1	0	1	53	0.7	136	
2	563	1	0.5	1	2	1	41	0.9	145	
3	615	1	2.5	0	0	0	10	0.8	131	
4	1821	1	1.2	0	13	1	44	0.6	141	

5 rows × 21 columns

In [5]:
```
df.shape
```

Out[5]:
```
(2000, 21)
```

In [6]:
```
df.describe()
```

Out[6]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memor
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.0000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.0465
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.1457
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.0000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.0000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.0000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.0000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.0000

8 rows × 21 columns

In [7]:
```
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: 328.3 KB
```

In [37]:
```
df.tail(3)
```

Out[37]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n
1997	1911	0	0.9	1	1	1	36	0.7	108	
1998	1512	0	0.9	0	4	1	46	0.1	145	
1999	510	1	2.0	1	5	1	45	0.9	168	

3 rows × 21 columns

In [38]:
```
df.nunique()#no. of unique values
```

Out[38]:
```
battery_power    1094
blue                2
clock_speed        26
dual_sim            2
fc                 20
four_g              2
int_memory         63
m_dep              10
mobile_wt         121
n_cores             8
pc                 21
px_height        1137
px_width         1109
ram              1562
sc_h               15
sc_w               19
talk_time          19
three_g             2
touch_screen        2
wifi                2
price_range         4
dtype: int64
```

In [39]:
```
df.isnull().sum()    #missing value show
```

Out[39]:
```
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
```

In [40]:
```
(df.isnull().sum()/(len(df)))*100 #precentage missing value show
```

Out[40]:
```
battery_power    0.0
blue             0.0
clock_speed      0.0
dual_sim         0.0
fc               0.0
four_g           0.0
int_memory       0.0
m_dep            0.0
mobile_wt        0.0
n_cores          0.0
pc               0.0
px_height        0.0
px_width         0.0
ram              0.0
sc_h             0.0
sc_w             0.0
talk_time        0.0
three_g          0.0
touch_screen     0.0
wifi             0.0
price_range      0.0
dtype: float64
```

In [45]:
```
df.nunique().count()
```

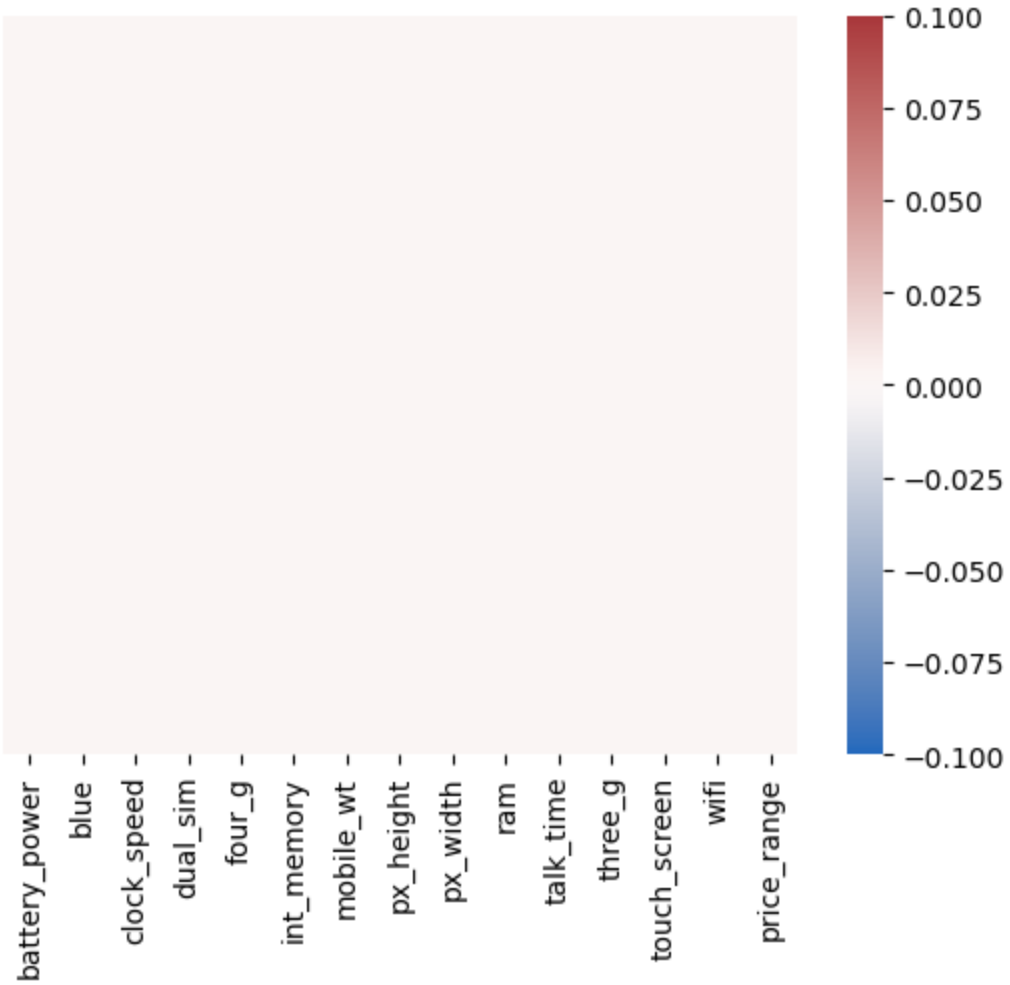
Out[45]:
```
15
```

Data Visulization

Heat map

In [46]:
```
sns.heatmap(df.isnull(), yticklabels=False, cmap="vlag") #Visualizing␣ null va
```

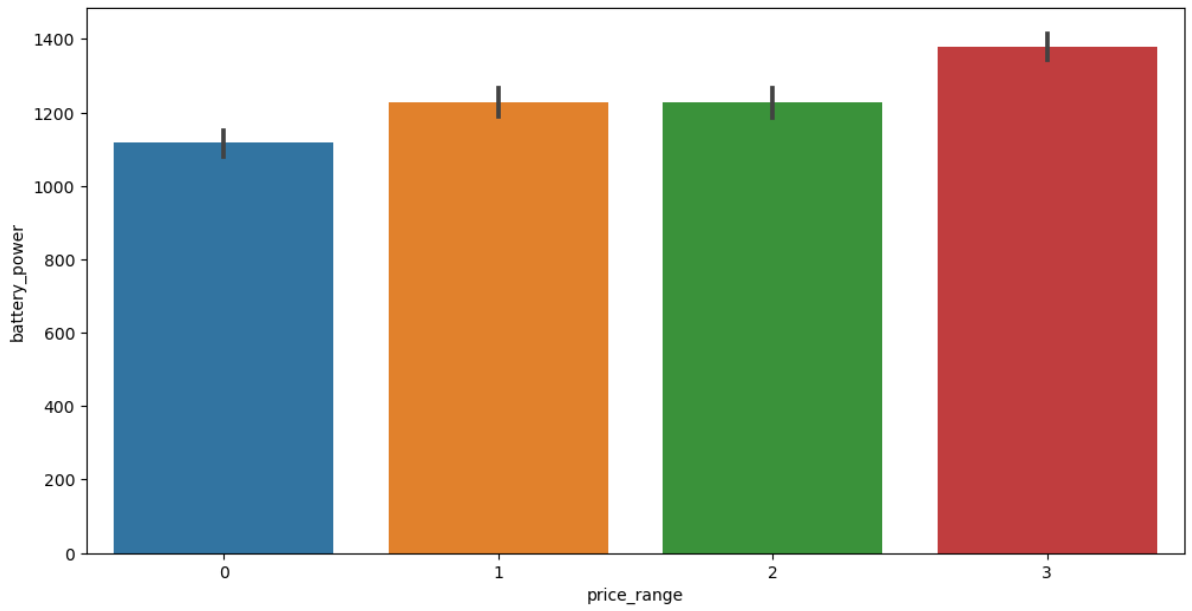
Out[46]: <Axes: >



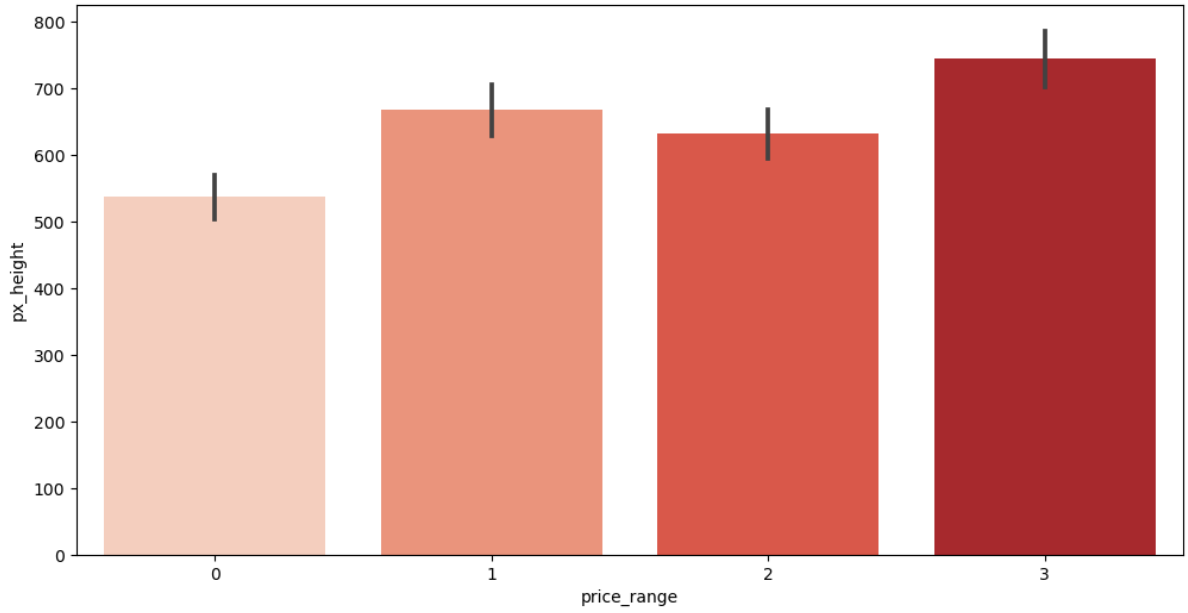
In [47]:
```
plt.figure(figsize=(12,6))
sns.heatmap(df.corr(),annot=True)
plt.show()
```

plotting relationbetween price range & battery power

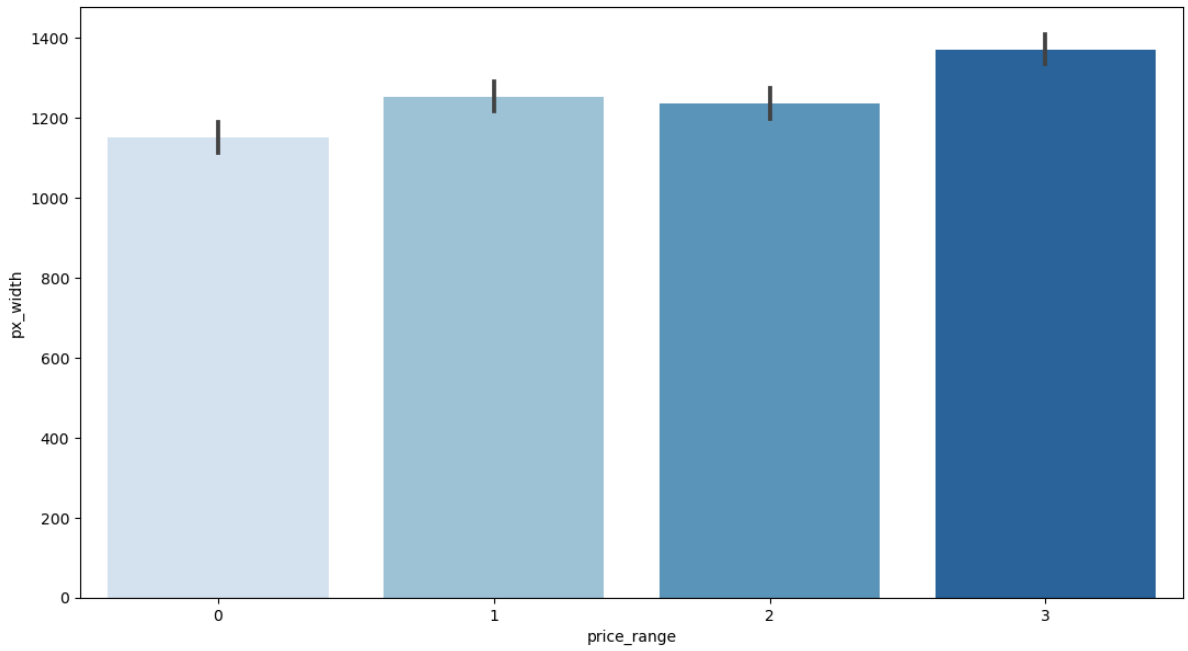
In [9]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='price_range',y='battery_power',data=df)
plt.show()
```



In [10]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='price_range',y='px_height',data=df,palette='Reds')
plt.show()
```

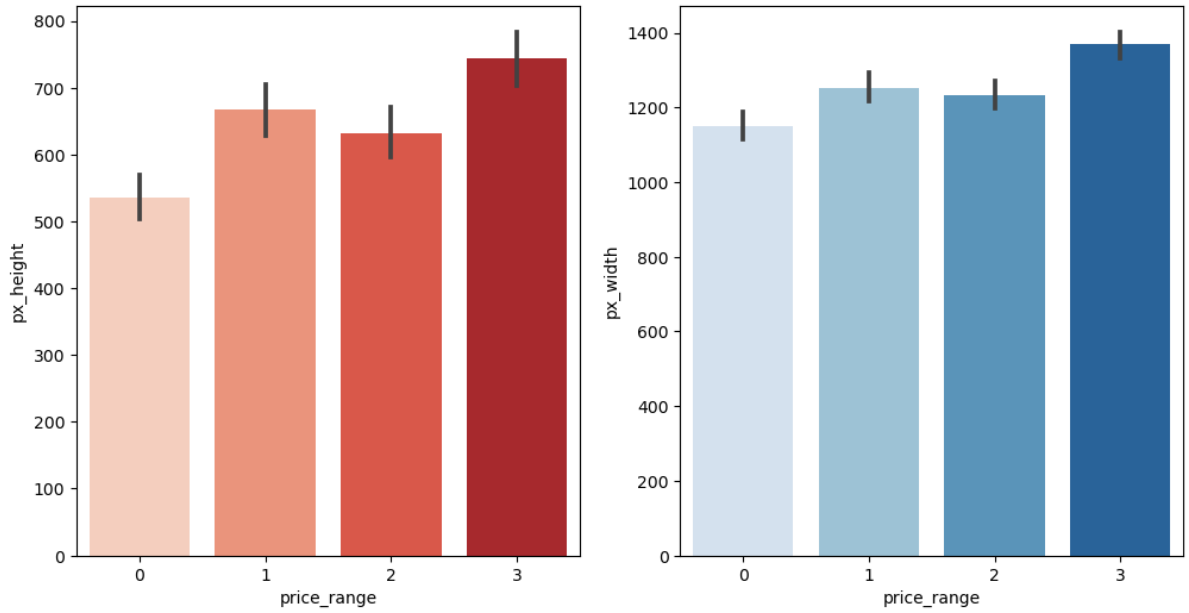


In [11]:
```
plt.figure(figsize=(13,7))
sns.barplot(x='price_range',y='px_width',data=df,palette='Blues')
plt.show()
```



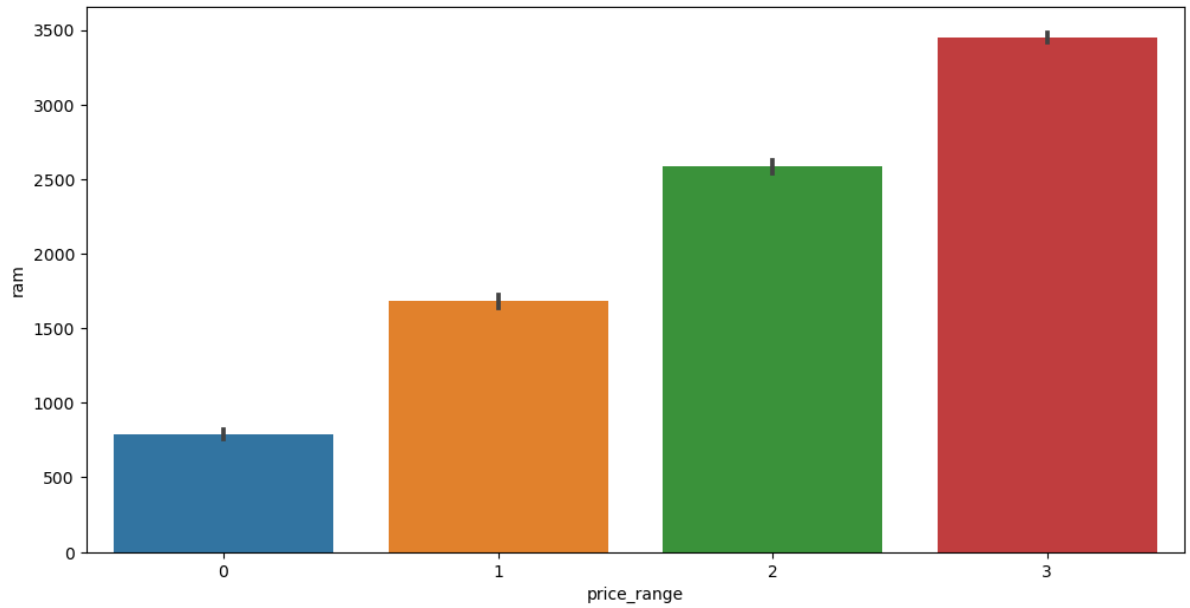
plotting relation between price range & px height,width

In [12]:
```
plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
sns.barplot(x='price_range',y='px_height',data=df,palette='Reds')
plt.subplot(1,2,2)
sns.barplot(x='price_range',y='px_width',data=df,palette='Blues')
plt.show()
```



plotting relation between price range & ram

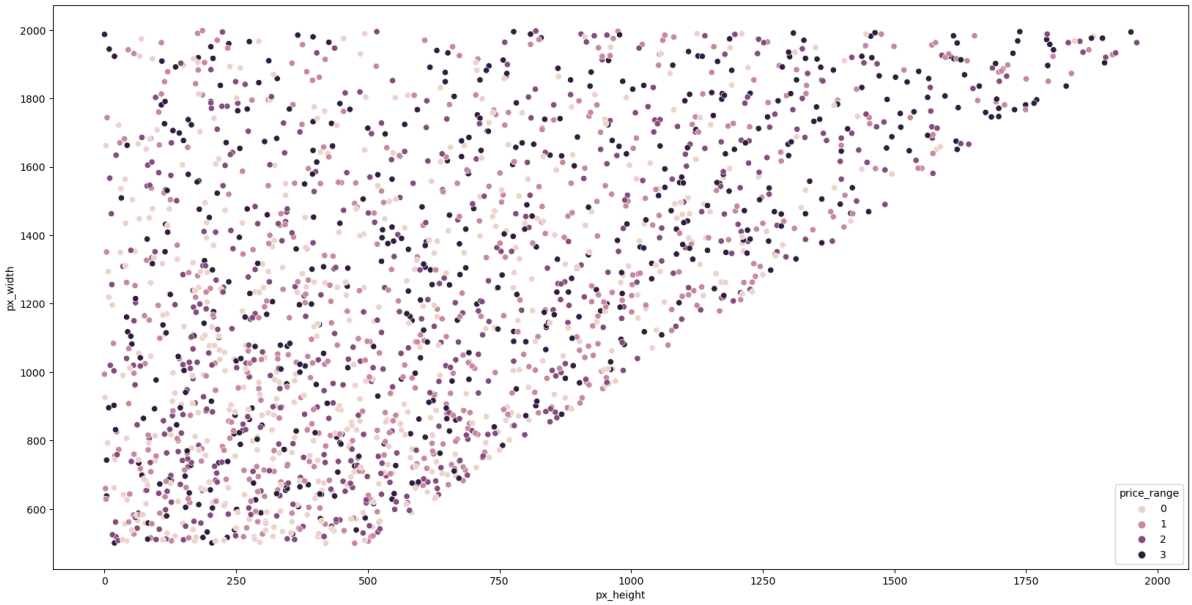
In [13]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='price_range',y='ram',data=df)
plt.show()
```



relation between price range &3g/4g

In [16]:
```
plt.figure(figsize=(20,10))
sns.scatterplot(x = "px_height", y = "px_width", data=df, hue = "price_range")
```

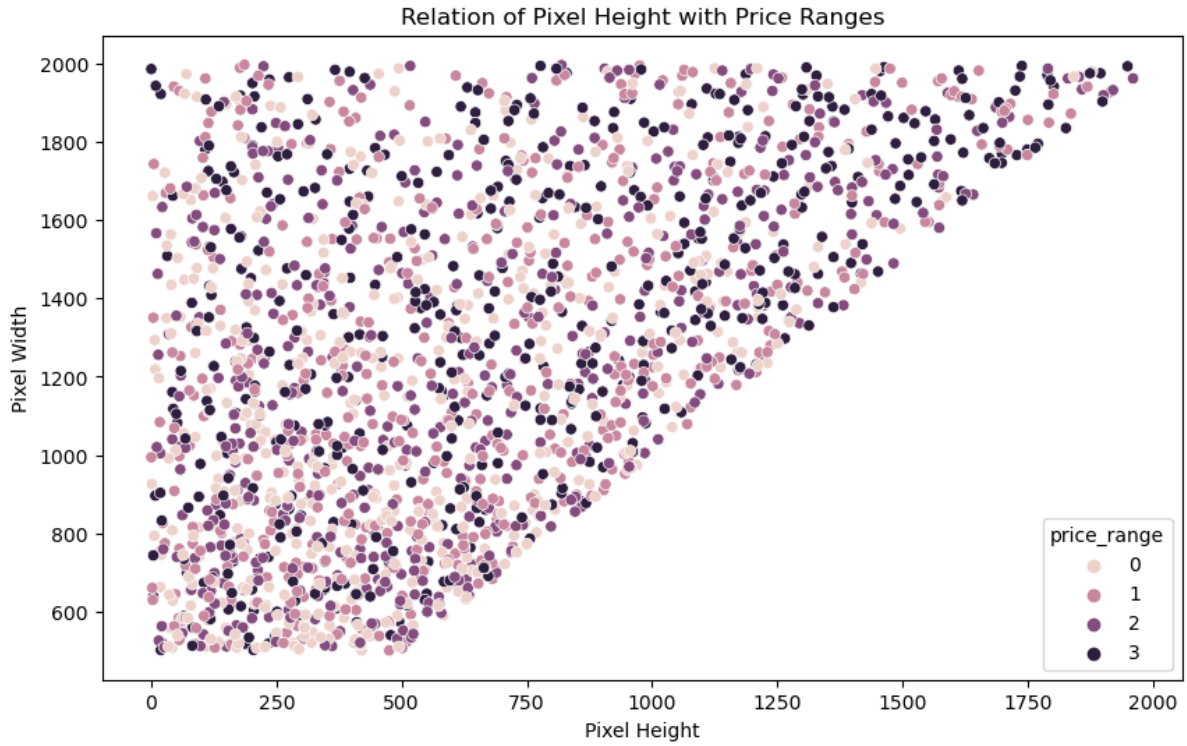
Out[16]:
```
<Axes: xlabel='px_height', ylabel='px_width'>
```



Relation of Pixel Height with Price Ranges

In [17]:
```
ure(figsize=(10,6))
tterplot(x = "px_height", y = "px_width", data=df, hue = "price_range").set(tit
                                                                    xlabel = "Pixel He
                                                                    ,ylabel = "Pixel Wi
```

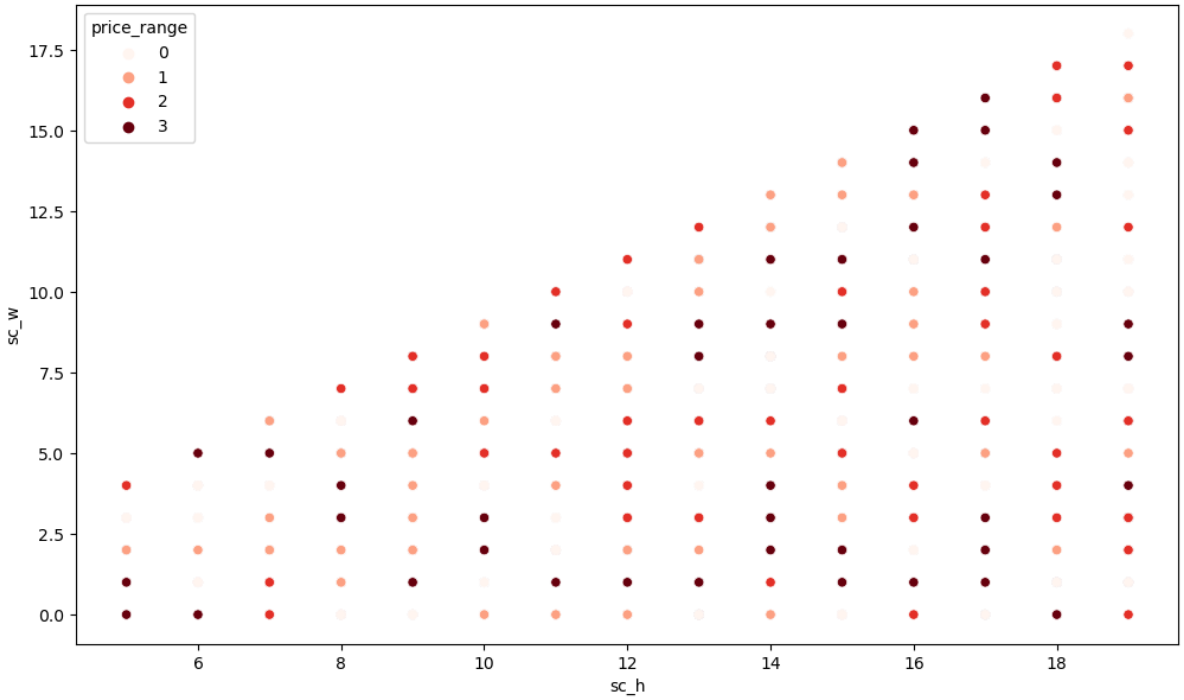
Out[17]:
```
[Text(0.5, 1.0, 'Relation of Pixel Height with Price Ranges'),
 Text(0.5, 0, 'Pixel Height'),
 Text(0, 0.5, 'Pixel Width')]
```



Relation of Screen Height and Screen Width with Price Ranges

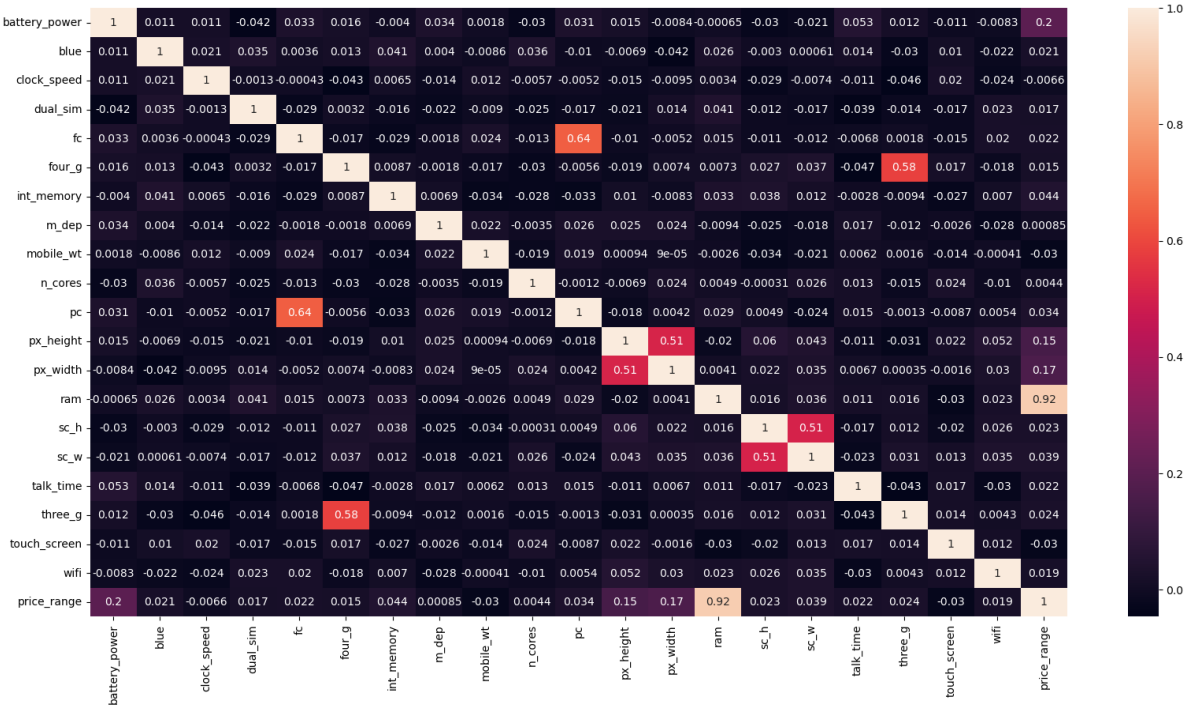
In [19]:
```
plt.figure(figsize=(12,7))
sns.scatterplot(x="sc_h",y="sc_w",data=df, hue = "price_range",palette='Reds')
```

Out[19]: <Axes: xlabel='sc_h', ylabel='sc_w'>



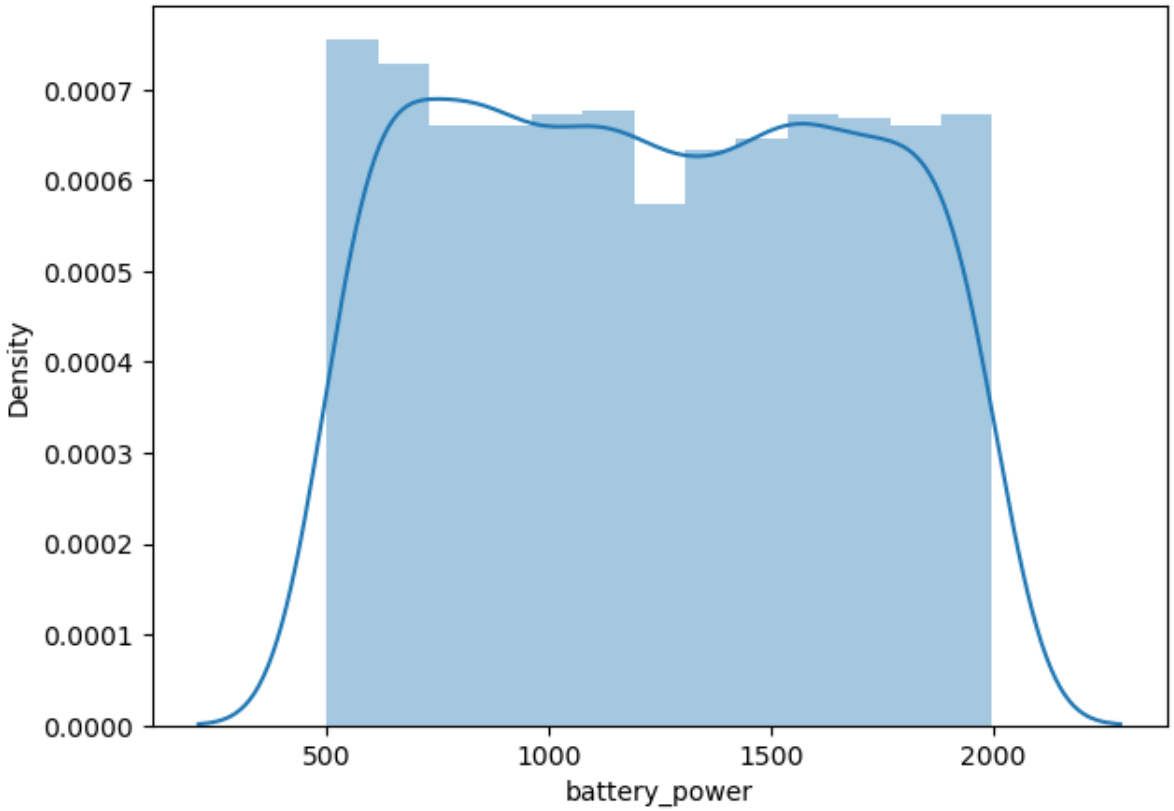
In [20]:
```
plt.figure(figsize = (20, 10))
sns.heatmap(df.corr(), annot = True)
```

Out[20]: <Axes: >



In [21]:
```
plt.figure(figsize = (7, 5))
sns.distplot(df['battery_power'])
plt.show()
```

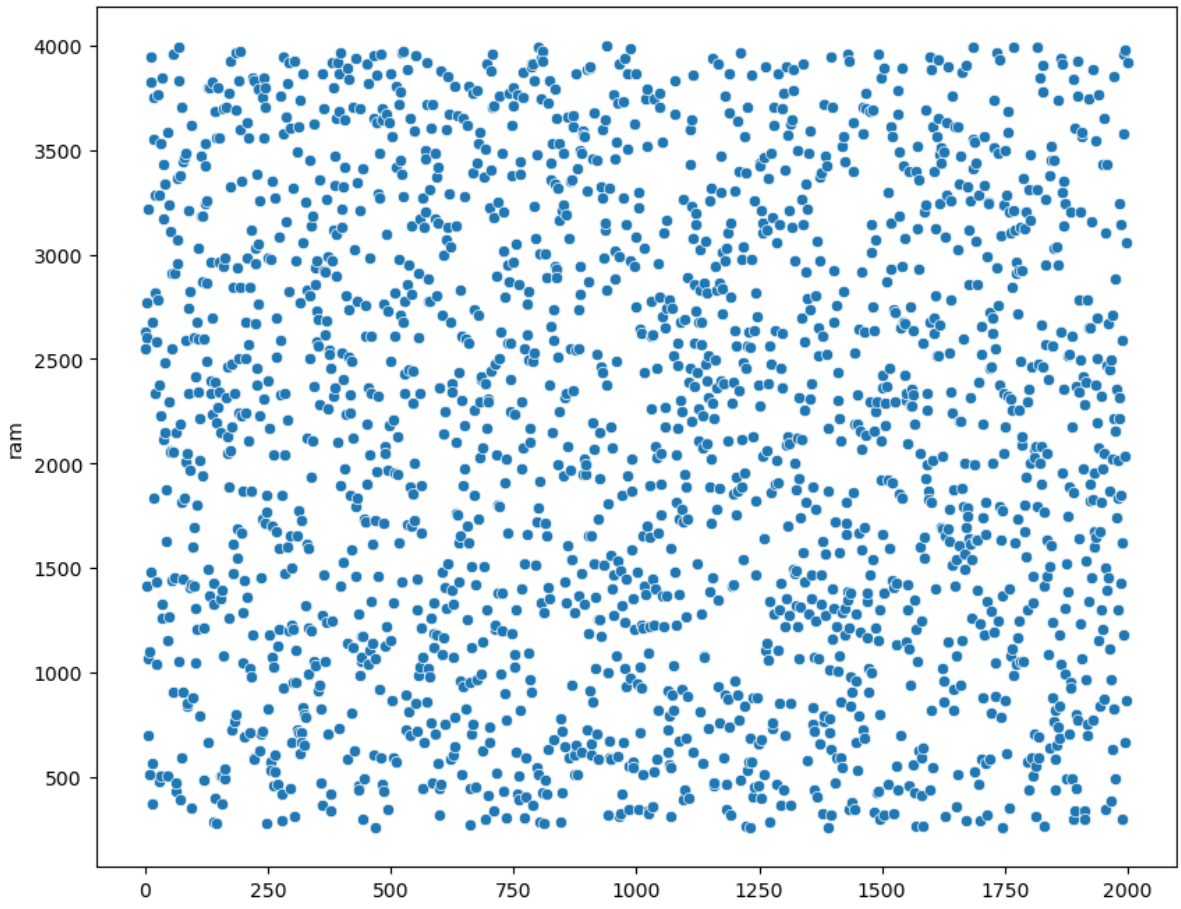
```
C:\Users\DD\AppData\Local\Temp\ipykernel_10772\3355353234.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df['battery_power'])
```



In [22]:
```
plt.figure(figsize = (10, 8))
sns.scatterplot(df['ram'], palette = 'icefire')
```

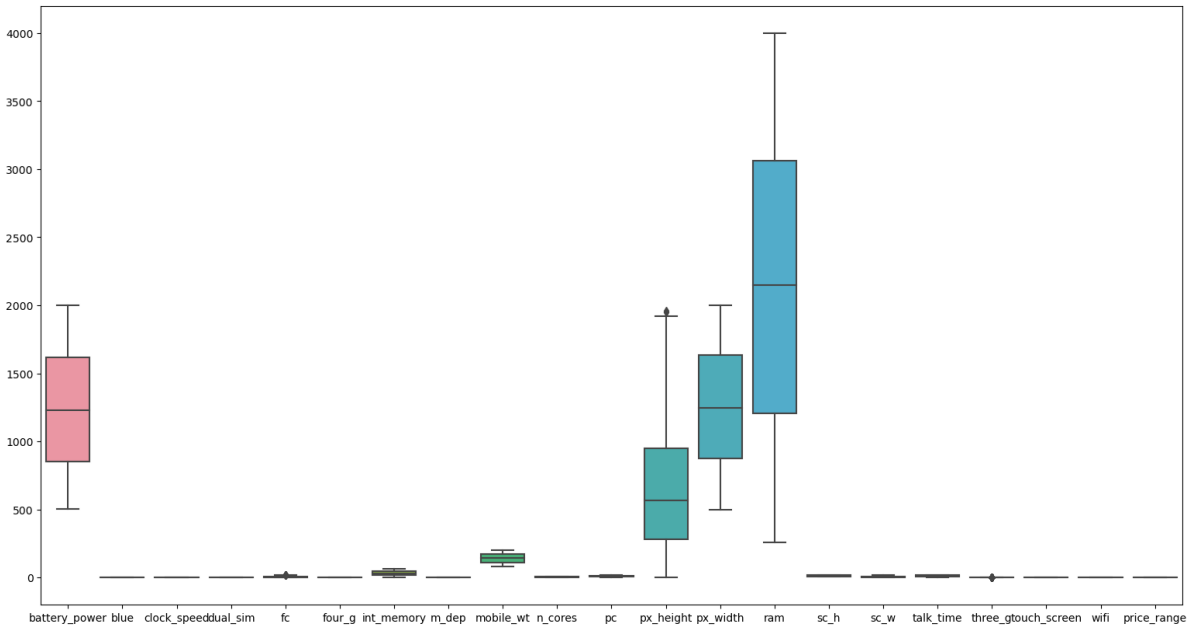
```
C:\Users\DD\AppData\Local\Temp\ipykernel_10772\1212288893.py:2: UserWarning:
Ignoring `palette` because no `hue` variable has been assigned.
  sns.scatterplot(df['ram'], palette = 'icefire')
```

Out[22]: <Axes: ylabel='ram'>



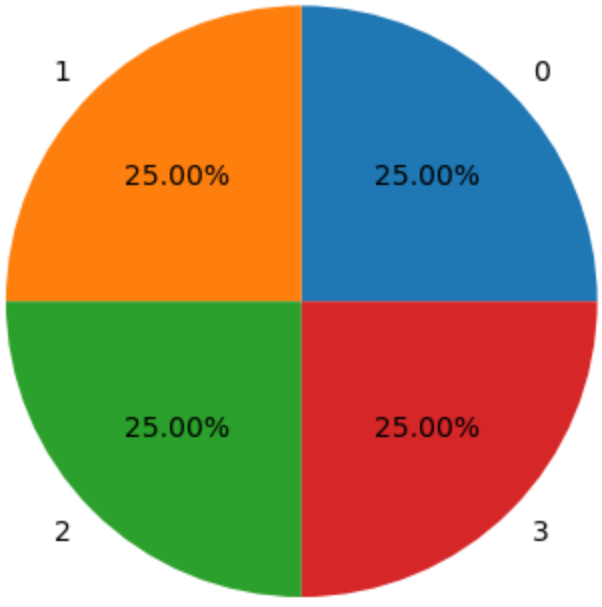
In [27]:
```
plt.figure(figsize = (19,10))
sns.boxplot(df)
```

Out[27]: <Axes: >



In [34]:
```
c=df.groupby("price_range")["price_range"].count()
```

In [36]:
```
plt.pie(c,labels=c.index,autopct="%.2f%%")
plt.show()
```



In [ ]: