Mobile Price

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

In [3]:
```
df=pd.read_csv(r"C:/Users/DD/Desktop/Mobile Prices.csv")
```

In [4]:
```
df
```

Out[4]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n
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

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

Out[5]:
```
<bound method DataFrame.info of       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]>
```

In [6]:
```
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 [7]:
```
df.shape
```

Out[7]:
```
(2000, 21)
```

In [8]:
```
df.describe()
```

Out[8]:

	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	18.14571
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 [9]:
```
df.head(5)
```

Out[9]:

	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 [10]:
```
df.tail(3)
```

Out[10]:

	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 [11]:
```
df.isnull().sum()
```

Out[11]:
```
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 [12]:
```
df.nunique()
```

Out[12]:
```
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 [13]:
```
(df.isnull().sum()/(len(df)))*100 #Percentage of Missing Values in each column
```

Out[13]:
```
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 [14]:
```
df.isnull().values.any()
```

Out[14]:
```
False
```

In [15]:
```
df.dropna()
```

Out[15]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n
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

In [16]:
```
df.rename(columns={'blue': 'bluetooth'},inplace=True)# Change Column Names and
```

In [17]:
```
df
```

Out[17]:

	battery_power	bluetooth	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_
0	842	0	2.2	0	1	0	7	0.6	1
1	1021	1	0.5	1	0	1	53	0.7	1
2	563	1	0.5	1	2	1	41	0.9	1
3	615	1	2.5	0	0	0	10	0.8	1
4	1821	1	1.2	0	13	1	44	0.6	1
...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	1	0	1	2	0.8	1
1996	1965	1	2.6	1	0	0	39	0.2	1
1997	1911	0	0.9	1	1	1	36	0.7	1
1998	1512	0	0.9	0	4	1	46	0.1	1
1999	510	1	2.0	1	5	1	45	0.9	1

2000 rows × 21 columns

In [18]:
```
df["dual_sim"].replace({0: "No", 1: "Yes"}, inplace=True)# Dual Sim(change colu

df["four_g"].replace({0: "No", 1: "Yes"}, inplace=True)# 4G(change column name
```

In [19]:
```
df
```

Out[19]:

	battery_power	bluetooth	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_
0	842	0	2.2	No	1	No	7	0.6	1
1	1021	1	0.5	Yes	0	Yes	53	0.7	1
2	563	1	0.5	Yes	2	Yes	41	0.9	1
3	615	1	2.5	No	0	No	10	0.8	1
4	1821	1	1.2	No	13	Yes	44	0.6	1
...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	Yes	0	Yes	2	0.8	1
1996	1965	1	2.6	Yes	0	No	39	0.2	1
1997	1911	0	0.9	Yes	1	Yes	36	0.7	1
1998	1512	0	0.9	No	4	Yes	46	0.1	1
1999	510	1	2.0	Yes	5	Yes	45	0.9	1

2000 rows × 21 columns

In [21]:
```
df["wifi"].replace({0: "No", 1: "Yes"}, inplace=True)# Wifi column name and dat
df["price_range"].replace({0: "Low Cost", 1: "Medium Cost", 2 : "High Cost", 3
# Price Range column name and data)

print("Columns and Data Updated Successfully ")
```

```
Columns and Data Updated Successfully
```

In [22]:
```
df
```

Out[22]:

	battery_power	bluetooth	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_
0	842	0	2.2	No	1	No	7	0.6	1
1	1021	1	0.5	Yes	0	Yes	53	0.7	1
2	563	1	0.5	Yes	2	Yes	41	0.9	1
3	615	1	2.5	No	0	No	10	0.8	1
4	1821	1	1.2	No	13	Yes	44	0.6	1
...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	Yes	0	Yes	2	0.8	1
1996	1965	1	2.6	Yes	0	No	39	0.2	1
1997	1911	0	0.9	Yes	1	Yes	36	0.7	1
1998	1512	0	0.9	No	4	Yes	46	0.1	1
1999	510	1	2.0	Yes	5	Yes	45	0.9	1

2000 rows × 21 columns

In [23]:
```
df.head(3)
```

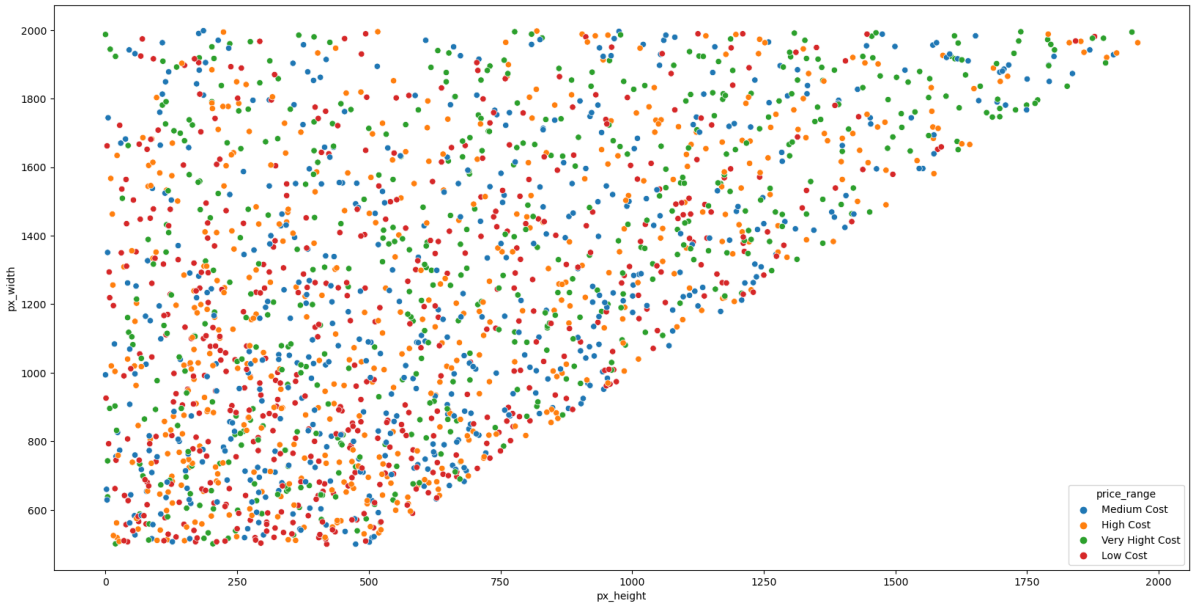
Out[23]:

	battery_power	bluetooth	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt
0	842	0	2.2	No	1	No	7	0.6	188
1	1021	1	0.5	Yes	0	Yes	53	0.7	136
2	563	1	0.5	Yes	2	Yes	41	0.9	145

3 rows × 21 columns

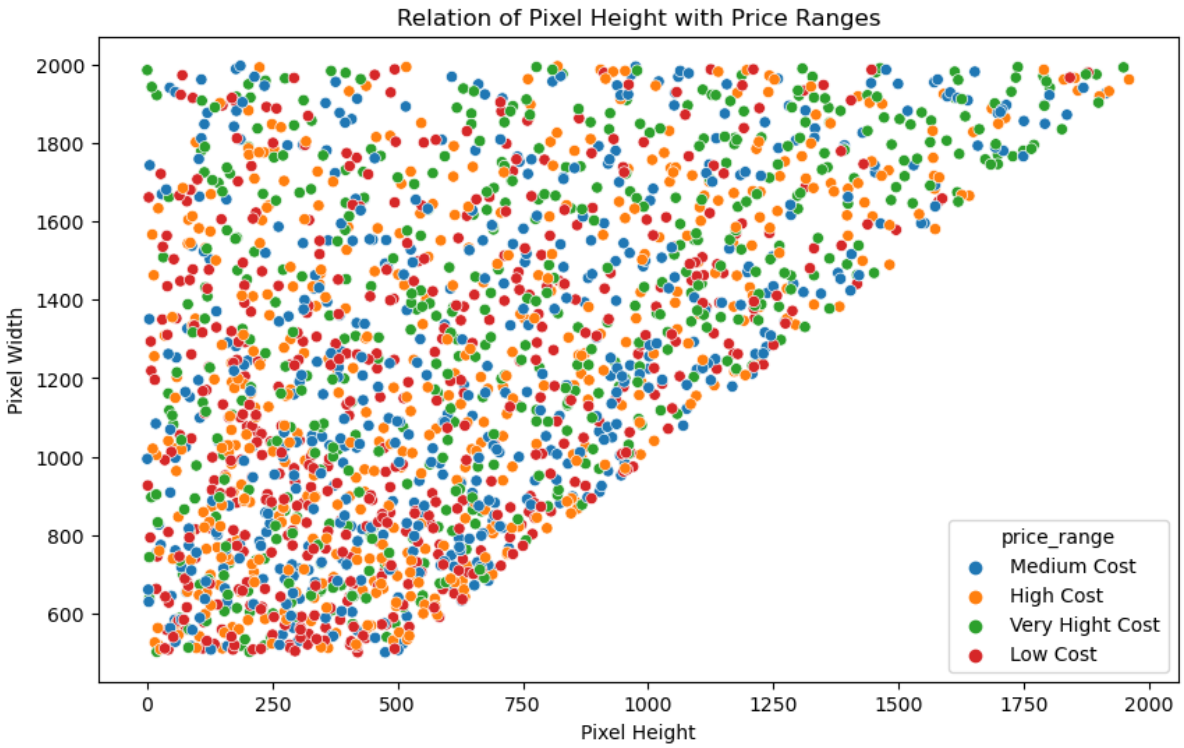
In [24]:
```
plt.figure(figsize=(20,10))
sns.scatterplot(x = "px_height", y = "px_width", data=df, hue = "price_range")
```

Out[24]: <Axes: xlabel='px_height', ylabel='px_width'>



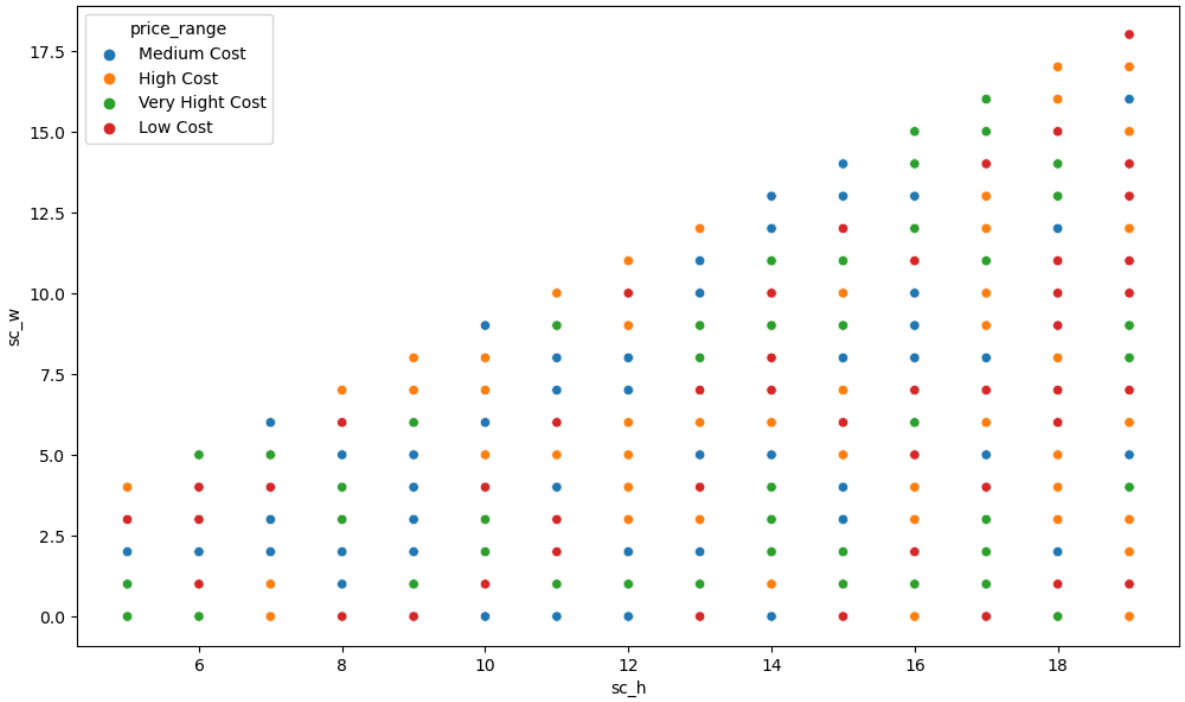
In [31]:
```
plt.figure(figsize=(10,6))
sns.scatterplot(x = "px_height", y = "px_width", data=df, hue = "price_range").
                                                                    xlabel = "P
                                                                   ,ylabel = "P
```

Out[31]:
```
[Text(0.5, 1.0, 'Relation of Pixel Height with Price Ranges'),
 Text(0.5, 0, 'Pixel Height'),
 Text(0, 0.5, 'Pixel Width')]
```



In [32]:
```
plt.figure(figsize=(12,7))
sns.scatterplot(x="sc_h",y="sc_w",data=df, hue = "price_range")
# Relation of Screen Height and Screen Width with Price Ranges
```

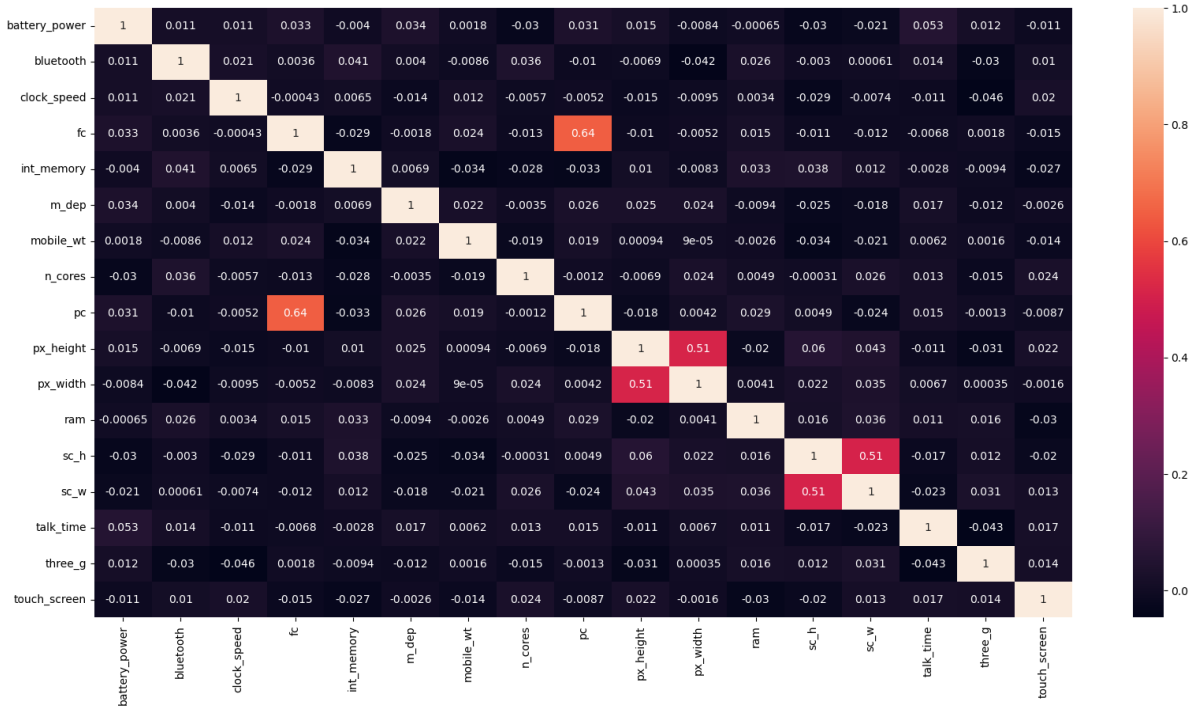
Out[32]: <Axes: xlabel='sc_h', ylabel='sc_w'>



In [33]:
```
plt.figure(figsize = (20, 10))
sns.heatmap(df.corr(), annot = True)
```

```
C:\Users\DD\AppData\Local\Temp\ipykernel_10336\3391398152.py:2: FutureWarnin
g: The default value of numeric_only in DataFrame.corr is deprecated. In a fu
ture version, it will default to False. Select only valid columns or specify
the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(), annot = True)
```

Out[33]: <Axes: >



In [36]:
```
df.groupby('price_range')['px_height', 'px_width'].mean()
```

```
C:\Users\DD\AppData\Local\Temp\ipykernel_10336\3293347090.py:1: FutureWarnin
g: Indexing with multiple keys (implicitly converted to a tuple of keys) will
be deprecated, use a list instead.
  df.groupby('price_range')['px_height', 'px_width'].mean()
```

Out[36]:

price_range	px_height	px_width
High Cost	632.284	1234.046
Low Cost	536.408	1150.270
Medium Cost	666.892	1251.908
Very Hight Cost	744.848	1369.838

In [41]:
```
df.groupby('ram')['int_memory', 'pc'].mean().head(10)
```

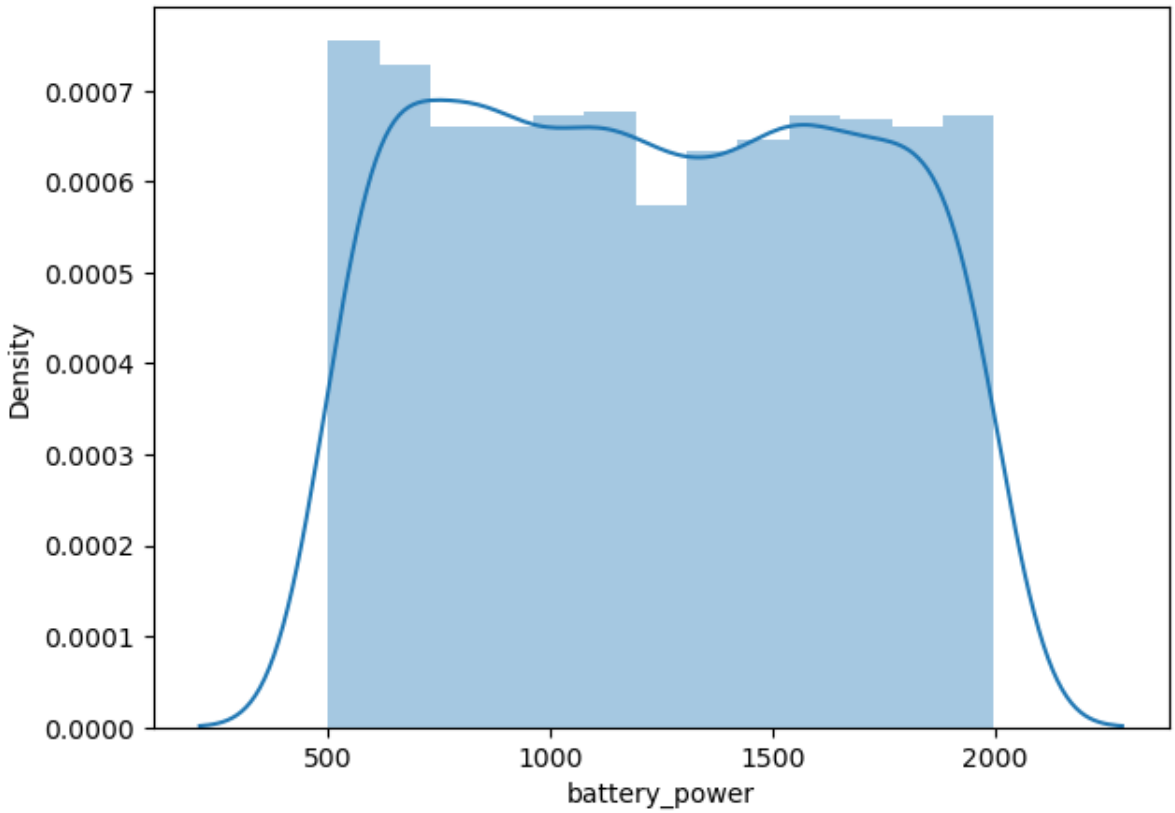
```
C:\Users\DD\AppData\Local\Temp\ipykernel_10336\1221497500.py:1: FutureWarnin
g: Indexing with multiple keys (implicitly converted to a tuple of keys) will
be deprecated, use a list instead.
  df.groupby('ram')['int_memory', 'pc'].mean().head(10)
```

Out[41]:

ram	int_memory	pc
256	59.0	14.0
258	28.0	10.5
259	20.0	6.0
262	56.0	1.0
263	19.0	19.0
265	48.0	8.0
267	27.0	6.0
273	21.0	2.0
277	25.0	10.0
278	43.5	13.0

In [46]:
```
plt.figure(figsize = (7, 5))
sns.distplot(df['battery_power'])
plt.show()
```

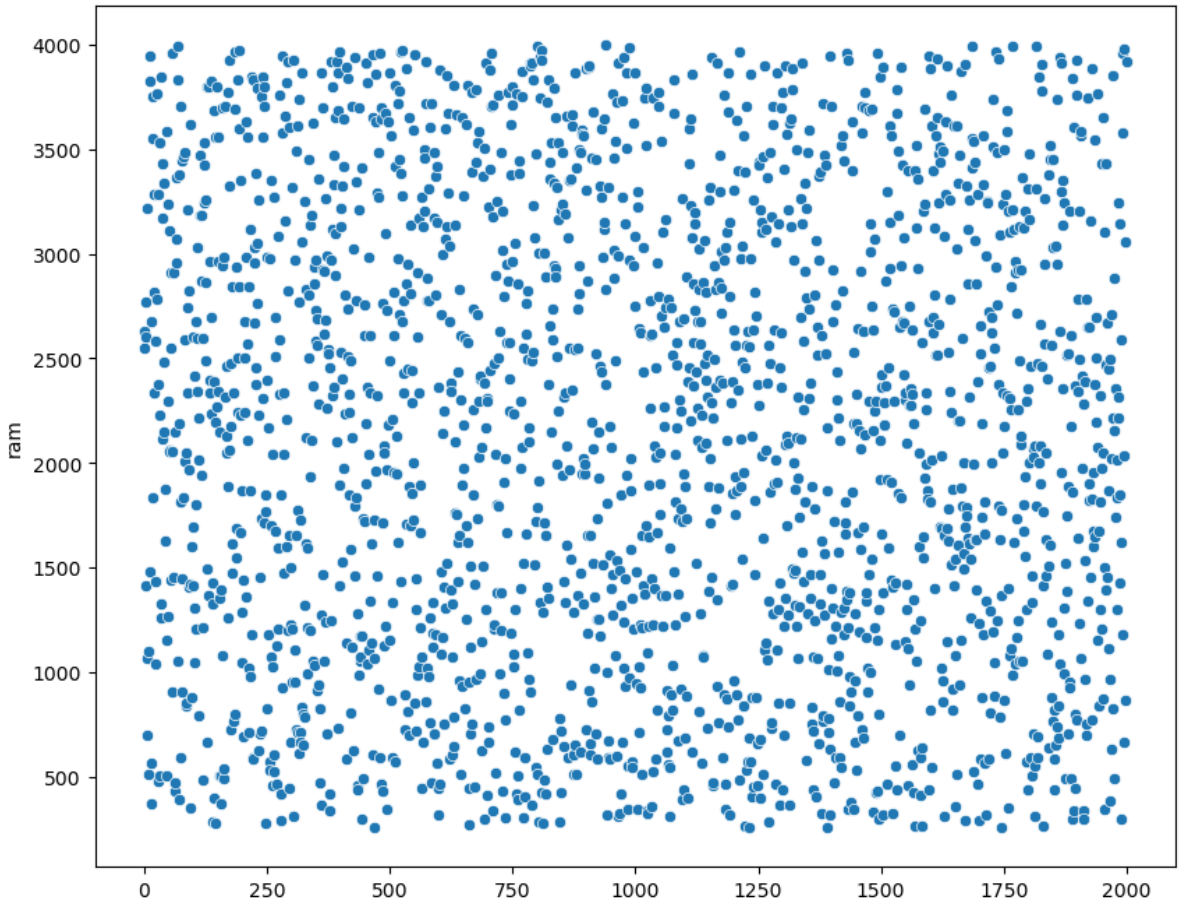
```
C:\Users\DD\AppData\Local\Temp\ipykernel_10336\3355353234.py:2: UserWarning:

`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 [55]:
```
plt.figure(figsize = (10, 8))
sns.scatterplot(df['ram'], palette = 'icefire')
```

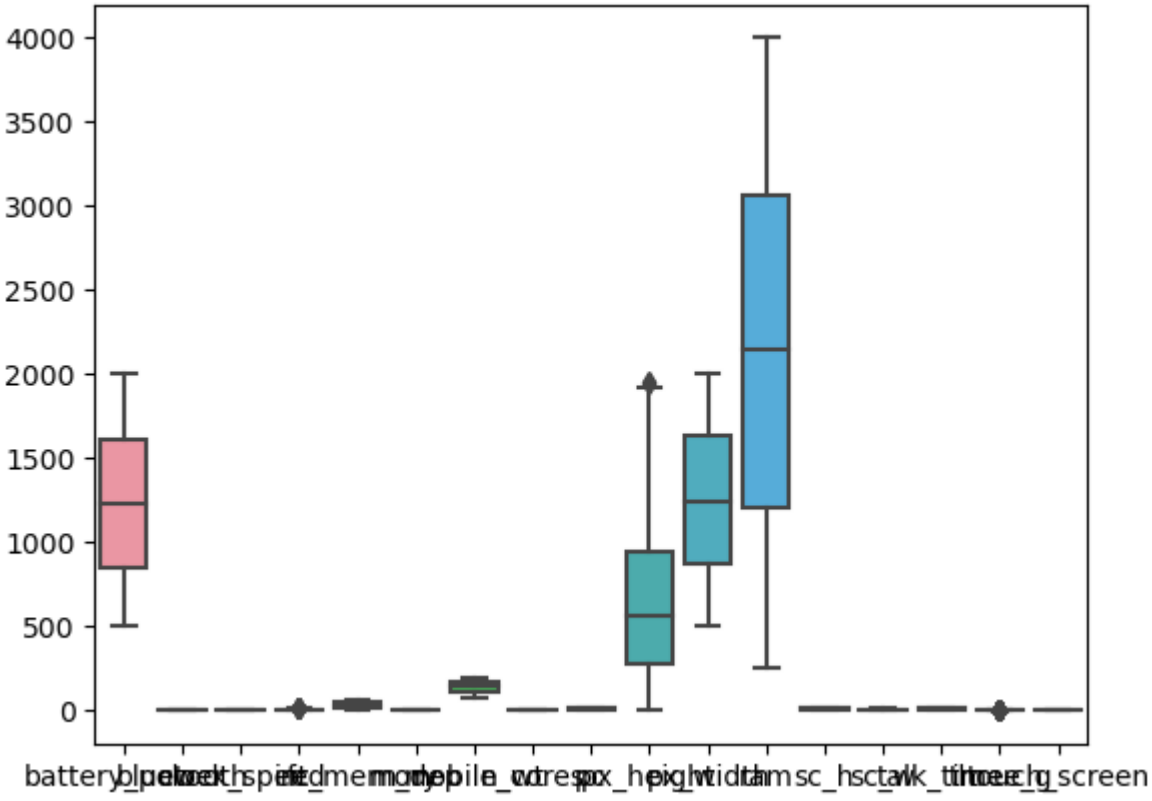
```
C:\Users\DD\AppData\Local\Temp\ipykernel_10336\1212288893.py:2: UserWarning:
Ignoring `palette` because no `hue` variable has been assigned.
  sns.scatterplot(df['ram'], palette = 'icefire')
```

Out[55]: <Axes: ylabel='ram'>



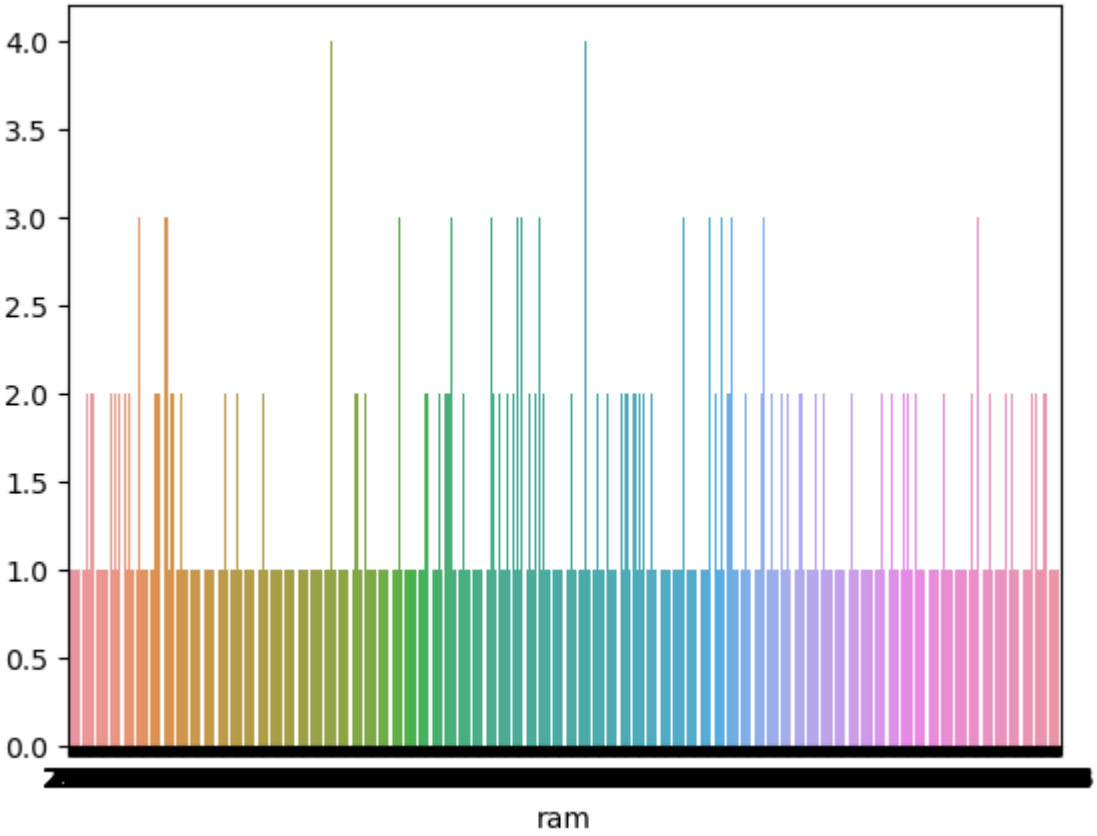
In [60]:
```
sns.boxplot(df)
```

Out[60]: <Axes: >



In [64]:
```
sns.barplot(x=df.index,y=df.values)
```

Out[64]: <Axes: xlabel='ram'>



In [71]:
```
plt.figure(figsize=(12,6))
sns.scatterplot(x="ram",y="battery_power",hue=pricerange,data=df)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[71], line 2
      1 plt.figure(figsize=(12,6))
----> 2 sns.scatterplot(x="ram",y="battery_power",hue=pricerange,data=df)

NameError: name 'pricerange' is not defined


<Figure size 1200x600 with 0 Axes>
```

In [ ]: