```
import pandas as pd
df=pd.read_csv('/content/supermarket.csv')
df
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	p
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	42.3675	1/29/2019	13:46	Ewallet	40.35	
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4900	3/2/2019	17:16	Ewallet	973.80	
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.5920	33.4320	2/9/2019	13:22	Cash	31.84	
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.2910	69.1110	2/22/2019	15:33	Cash	65.82	
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190	649.2990	2/18/2019	13:28	Cash	618.38	

```
df.head()
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085

```
df.tail()
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.017
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.690
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.592
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.291
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.919

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Invoice ID               1000 non-null   object
 1   Branch                   1000 non-null   object
 2   City                     1000 non-null   object
 3   Customer type            1000 non-null   object
 4   Gender                   1000 non-null   object
 5   Product line             1000 non-null   object
 6   Unit price               1000 non-null   float64
 7   Quantity                 1000 non-null   int64
 8   Tax 5%                   1000 non-null   float64
 9   Total                    1000 non-null   float64
 10  Date                     1000 non-null   object
 11  Time                     1000 non-null   object
 12  Payment                  1000 non-null   object
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  Rating                   1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
```

```
df.size
```

```
17000
```

```
df.describe()
```

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000	100
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905	1
std	26.494628	2.923431	11.708825	245.885335	234.17651	0.000000	1
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905	
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905	
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905	1
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905	2

```
df.shape
```

```
(1000, 17)
```

```
df.columns
```

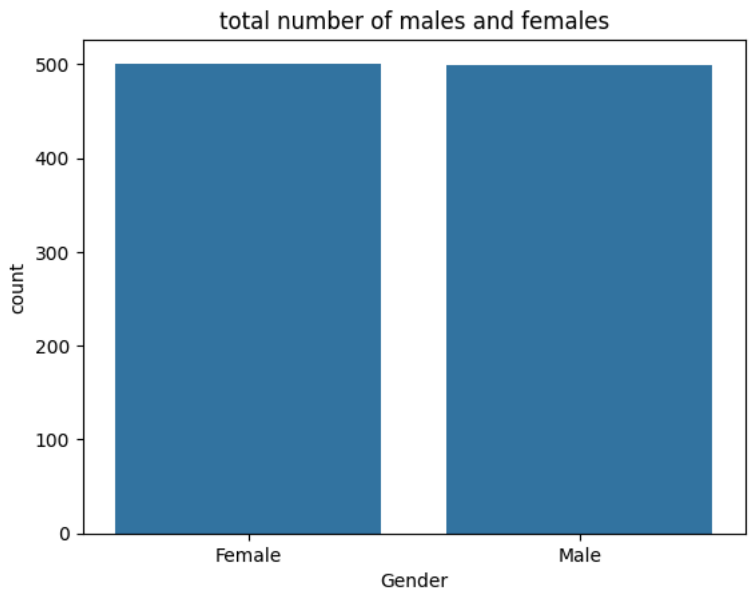
```
Index(['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender',
       'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Date',
       'Time', 'Payment', 'cogs', 'gross margin percentage', 'gross income',
       'Rating'],
      dtype='object')
```

```
g=df['Gender'].value_counts()#counting number of males and females
g
```

```
Gender
Female    501
Male      499
Name: count, dtype: int64
```

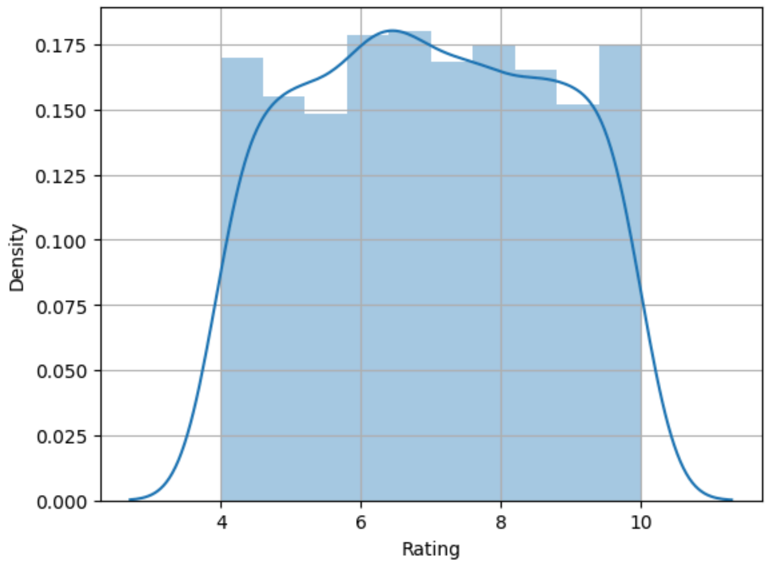
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x=df['Gender'])
plt.title("total number of males and females")
```

```
Text(0.5, 1.0, 'total number of males and females')
```



```
import numpy as np
sns.distplot(df['Rating'])
plt.grid()
```

```
<ipython-input-11-aed383c7a89a>: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

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

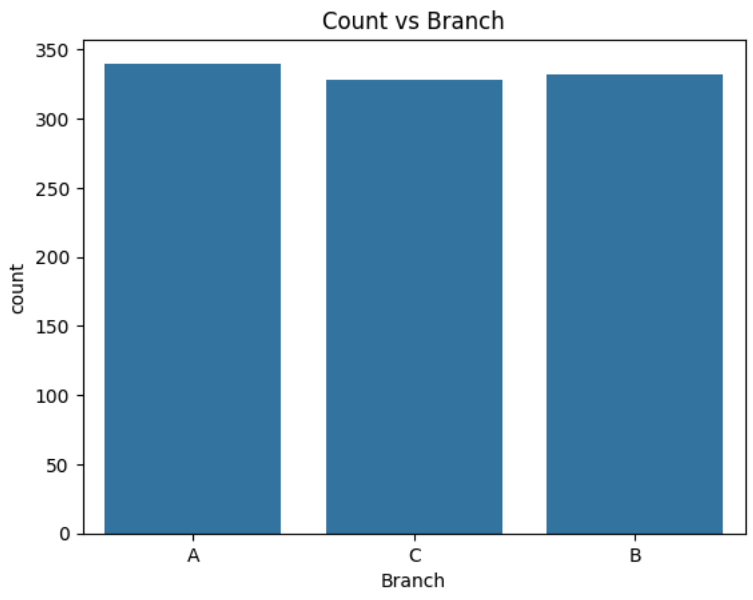


```
df["Branch"].value_counts()
```

```
Branch
A    340
B    332
C    328
Name: count, dtype: int64
```

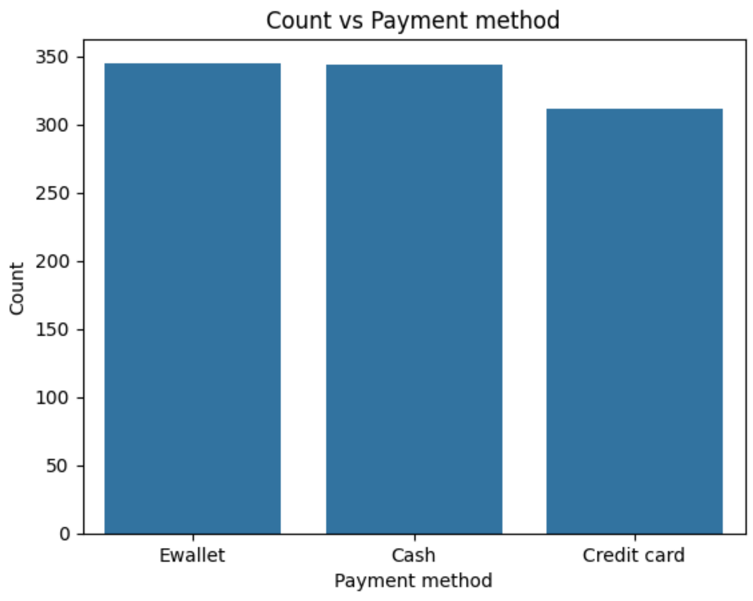
```
sns.countplot(x=df['Branch'])
plt.title("Count vs Branch")
```

```
Text(0.5, 1.0, 'Count vs Branch')
```



```
sns.countplot(x=df['Payment'])
plt.title("Count vs Payment method")
plt.xlabel("Payment method")
plt.ylabel("Count")
```

```
Text(0, 0.5, 'Count')
```

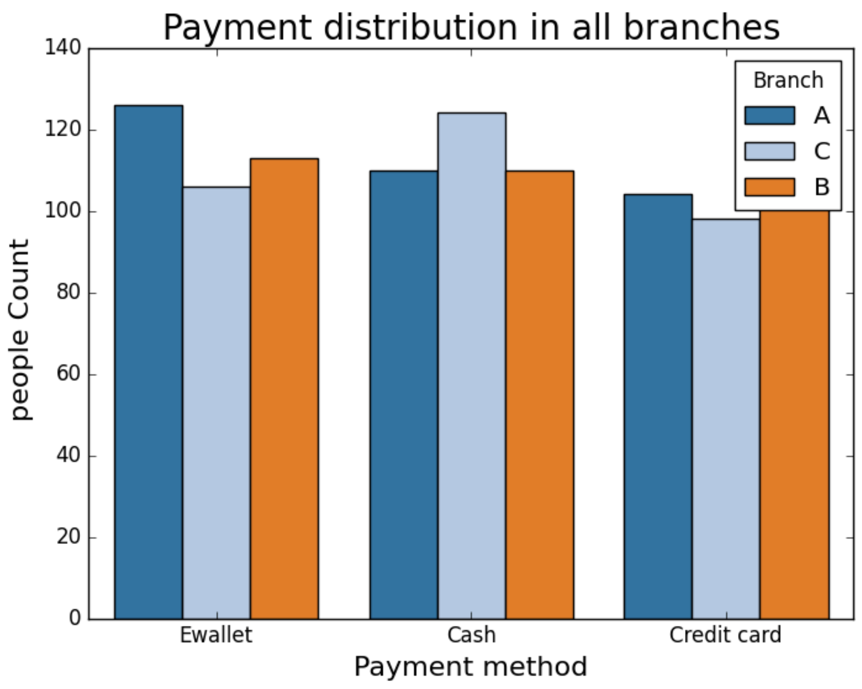


```
a=df["Payment"].value_counts()
a
```

```
Payment
Ewallet        345
Cash           344
Credit card    311
Name: count, dtype: int64
```

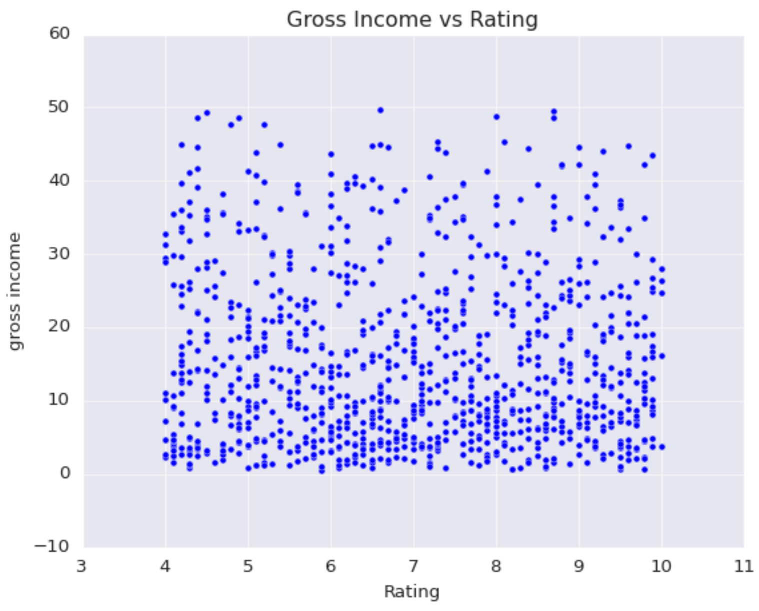
```
plt.figure(figsize=(8,6))
plt.style.use('classic')
ax=sns.countplot(x="Payment",hue="Branch",data=df,palette="tab20")
ax.set_title(label="Payment distribution in all branches",fontsize=20)
ax.set_xlabel(xlabel="Payment method",fontsize=16)
ax.set_ylabel(ylabel=" people Count",fontsize=16)
```

```
Text(0, 0.5, ' people Count')
```



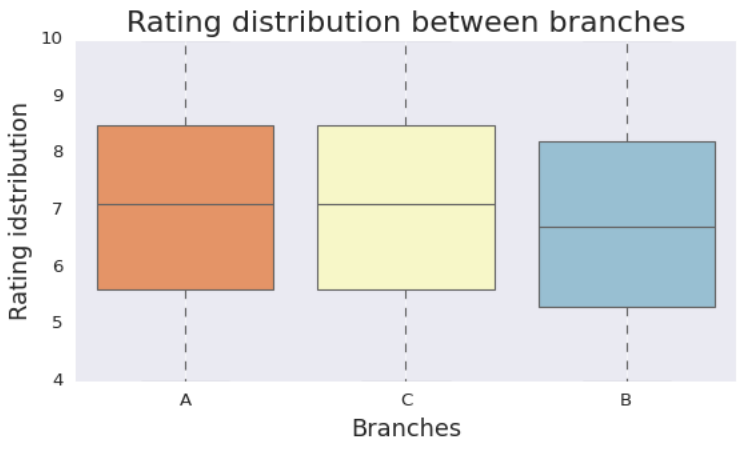
```
sns.set_style('darkgrid')
sns.scatterplot(x=df['Rating'],y=df['gross income'])
plt.title("Gross Income vs Rating")
```

```
Text(0.5, 1.0, 'Gross Income vs Rating')
```



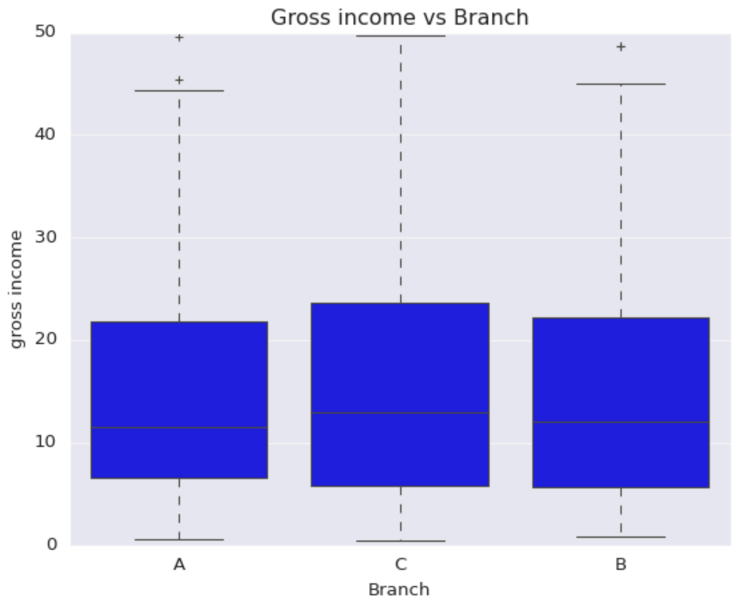
```
plt.figure(figsize=(8,4))
ax=sns.boxplot(x="Branch",y="Rating",data=df,palette="RdYlBu")
ax.set_title(label="Rating distribution between branches",fontsize=20)
ax.set_xlabel(xlabel="Branches",fontsize=16)
ax.set_ylabel(ylabel="Rating idstribution",fontsize=16)
plt.grid()
```

```
<ipython-input-18-a7ee5a544007>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  ax=sns.boxplot(x="Branch",y="Rating",data=df,palette="RdYlBu")
```



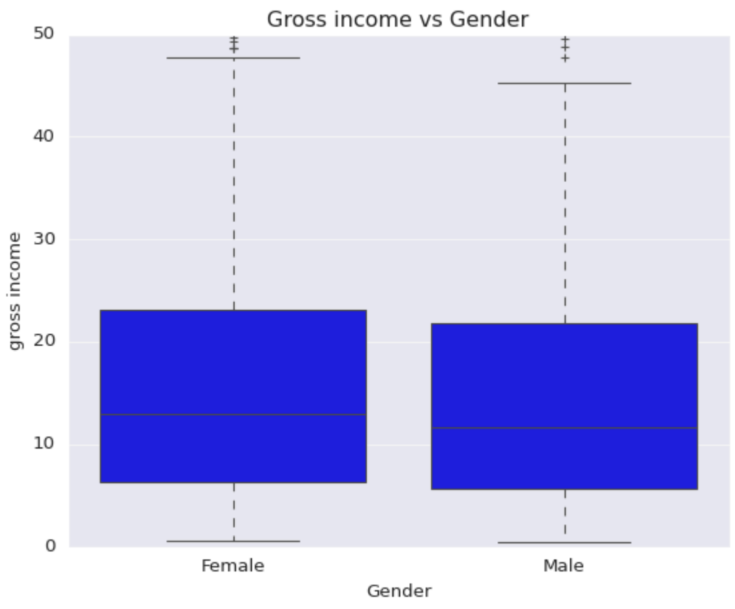
```
sns.boxplot(x=df['Branch'],y=df['gross income'])
plt.title("Gross income vs Branch")
```

```
Text(0.5, 1.0, 'Gross income vs Branch')
```



```
sns.boxplot(x=df['Gender'],y=df['gross income'])
plt.title("Gross income vs Gender")
```

```
Text(0.5, 1.0, 'Gross income vs Gender')
```



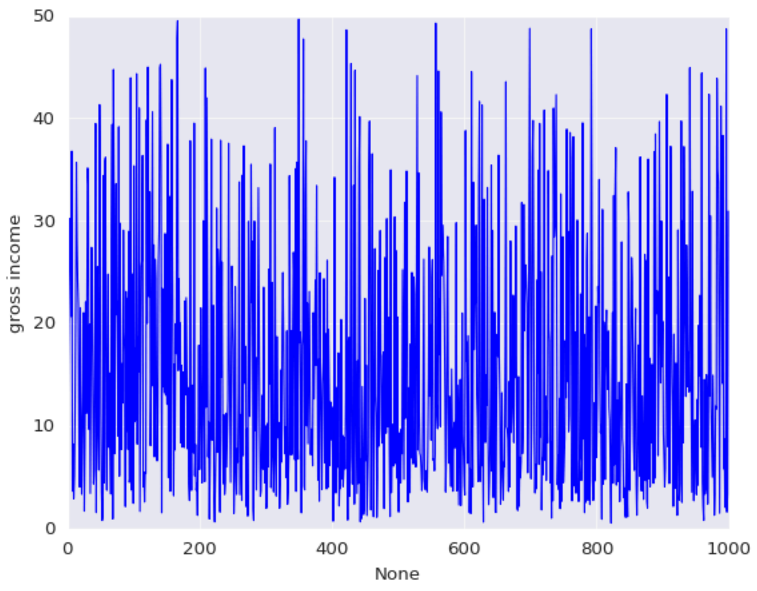
```
df1=df.select_dtypes(include='number')
df1.groupby(df.index).mean()
```

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
0	74.69	7.0	26.1415	548.9715	522.83	4.761905	26.1415	9.1
1	15.28	5.0	3.8200	80.2200	76.40	4.761905	3.8200	9.6
2	46.33	7.0	16.2155	340.5255	324.31	4.761905	16.2155	7.4
3	58.22	8.0	23.2880	489.0480	465.76	4.761905	23.2880	8.4
4	86.31	7.0	30.2085	634.3785	604.17	4.761905	30.2085	5.3
...	...	...	...	...	...	...	...	...
995	40.35	1.0	2.0175	42.3675	40.35	4.761905	2.0175	6.2
996	97.38	10.0	48.6900	1022.4900	973.80	4.761905	48.6900	4.4
997	31.84	1.0	1.5920	33.4320	31.84	4.761905	1.5920	7.7
998	65.82	1.0	3.2910	69.1110	65.82	4.761905	3.2910	4.1
999	88.34	7.0	30.9190	649.2990	618.38	4.761905	30.9190	6.6

1000 rows × 8 columns

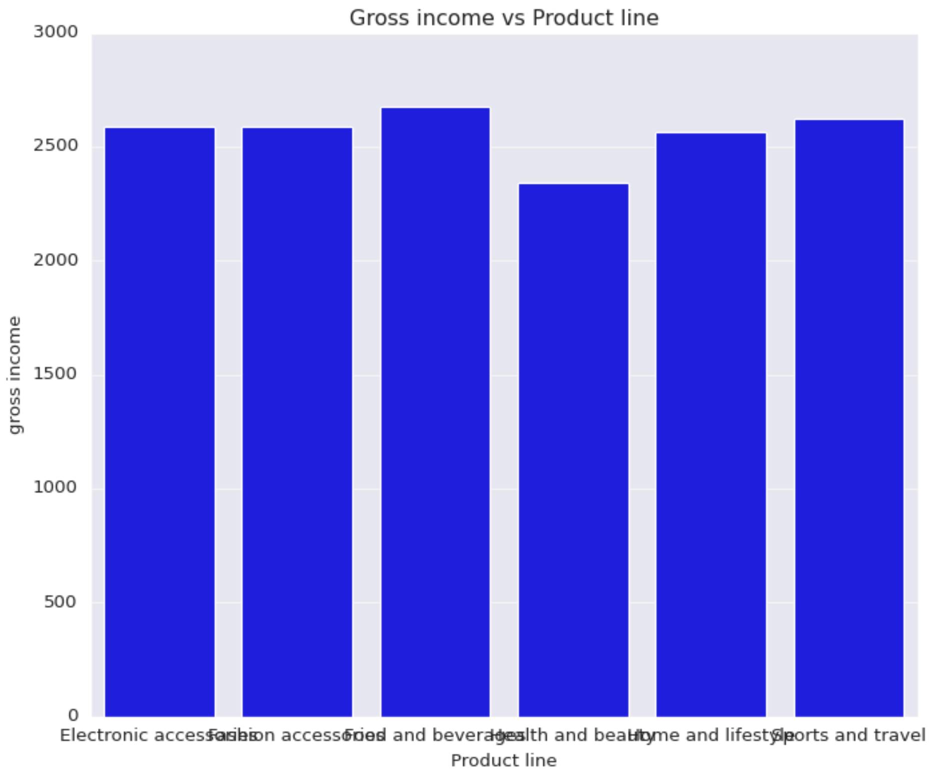
```
sns.lineplot(x=df1.groupby(df.index).mean().index,#to seperate identical data into groups to allow for further aggregation and analysis
             y=df1.groupby(df.index).mean()['gross income'])
```

```
<Axes: xlabel='None', ylabel='gross income'>
```



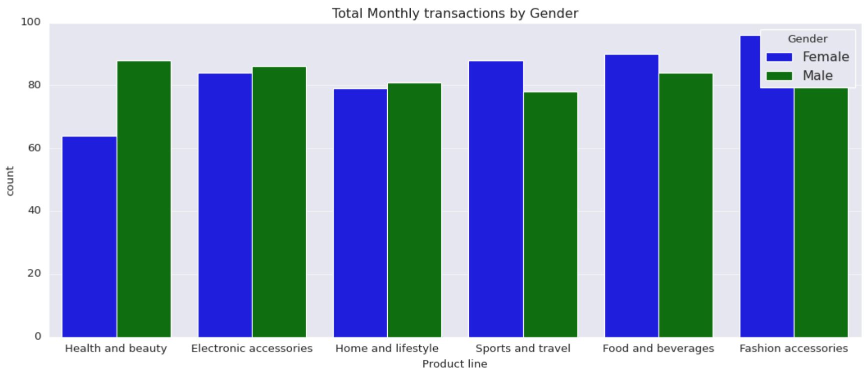
```
cat=df[["Product line","gross income"]].groupby(['Product line']).sum().reset_index()
plt.figure(figsize=(10,8))
sns.barplot(x='Product line',y='gross income',data=cat)
plt.title("Gross income vs Product line")
```

```
Text(0.5, 1.0, 'Gross income vs Product line')
```



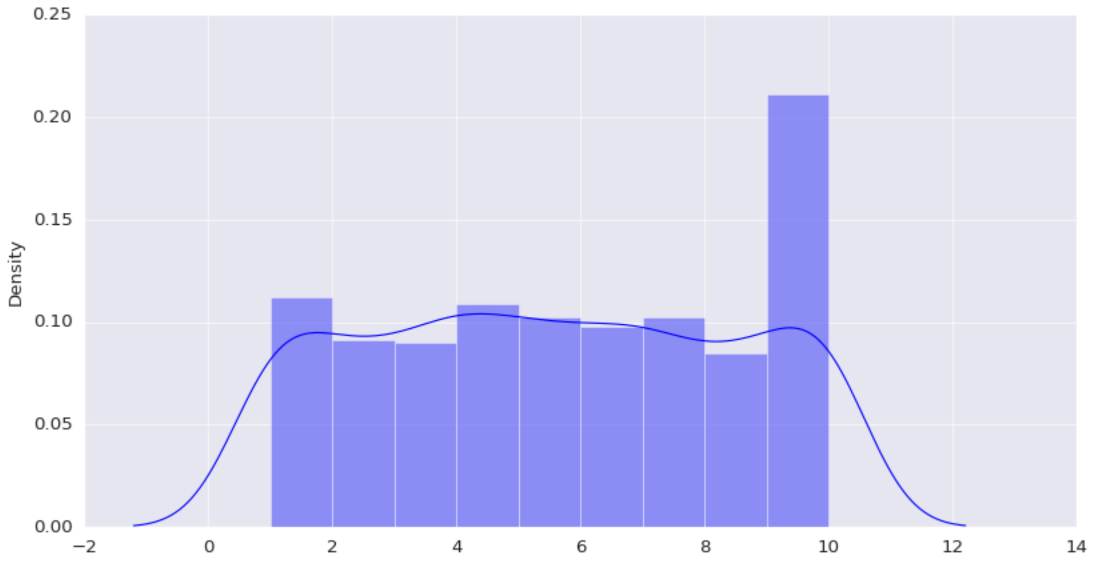
```
plt.figure(figsize=(16,6))
plt.title('Total Monthly transactions by Gender')
sns.countplot(x=df['Product line'],hue=df.Gender)#to color plot based on the values of a specific variables we used hue
```

```
<Axes: title={'center': 'Total Monthly transactions by Gender'}, xlabel='Product
line', ylabel='count'>
```



```
plt.figure(figsize=(12,6))
sns.distplot(x=df['Quantity'])
```

```
<ipython-input-25-559964f3843b>: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

  sns.distplot(x=df['Quantity'])
<Axes: ylabel='Density'>
```

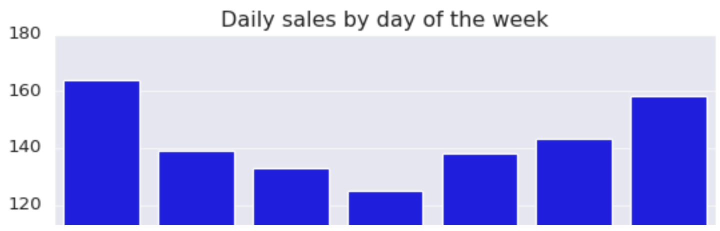


```
df['Date']=pd.to_datetime(df['Date'])
df['weekday']=df['Date'].dt.day_name()
df.set_index('Date',inplace=True)
df.head()
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax
Date									
2019-01-05	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.14
2019-03-08	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.82
2019-03-03	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.21
2019-01-27	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.28
2019-02-08	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.20

```
plt.figure(figsize=(8,6))
plt.title('Daily sales by day of the week')
sns.countplot(x=df['weekday'])
```

```
<Axes: title={'center': 'Daily sales by day of the week'}, xlabel='weekday', ylabel='count'>
```

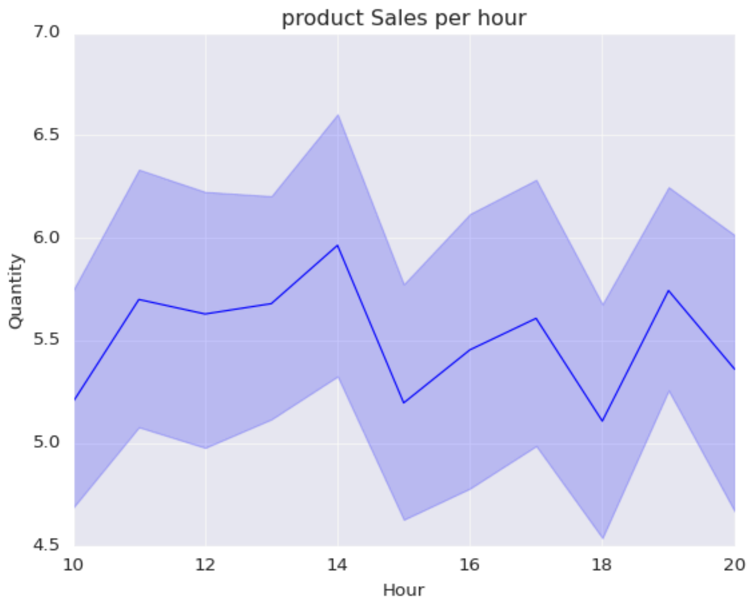


```
df['Time']=pd.to_datetime(df['Time'])
df['Hour']=df['Time'].dt.hour
df['Hour'].unique()
```

```
<ipython-input-28-41a1fe9ea6d1>:1: UserWarning: Could not infer format, so each element will be parsed individually, falling back to
  df['Time']=pd.to_datetime(df['Time'])
array([13, 10, 20, 18, 14, 11, 17, 16, 19, 15, 12], dtype=int32)
```

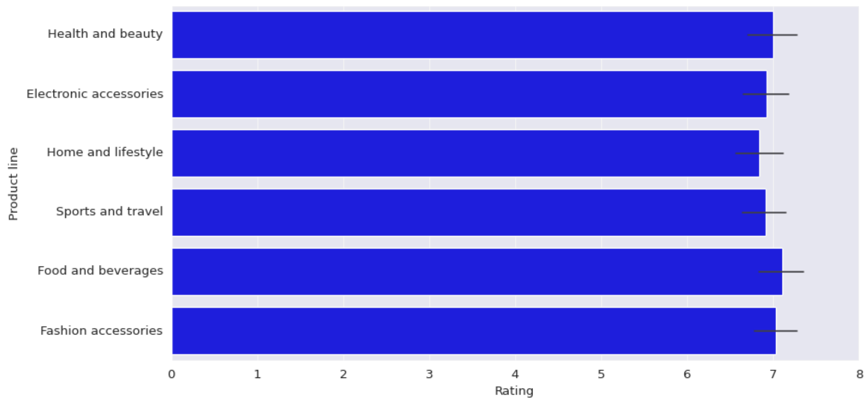
```
sns.lineplot(x="Hour",y='Quantity',data=df).set_title("product Sales per hour")
```

```
Text(0.5, 1.0, 'product Sales per hour')
```



```
plt.figure(figsize=(12,6))
sns.barplot(y=df['Product line'],x=df['Rating'])
```

```
<Axes: xlabel='Rating', ylabel='Product line'>
```


Health and beauty
Electronic accessories
Home and lifestyle
Sports and travel
Food and beverages
Fashion accessories
Product line
Rating