Super Market Sales Analysis

June 20, 2022

1 Import Basic Libraries

[1]:
```
import numpy as np
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
#Load the dataset
df=pd.read_csv("supermarket_sales - Sheet1.csv")
```

```
<IPython.core.display.Javascript object>
```

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

[3]:

	Invoice ID	Branch	City	Customer type	Gender	\
0	750-67-8428	A	Yangon	Member	Female	
1	226-31-3081	C	Naypyitaw	Normal	Female	
2	631-41-3108	A	Yangon	Normal	Male	
3	123-19-1176	A	Yangon	Member	Male	
4	373-73-7910	A	Yangon	Normal	Male	

	Product line	Unit price	Quantity	Tax 5%	Total	Date	\
0	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	
1	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	
2	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	
3	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	
4	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	

	Time	Payment	cogs	gross margin percentage	gross income	Rating
0	13:08	Ewallet	522.83	4.761905	26.1415	9.1
1	10:29	Cash	76.40	4.761905	3.8200	9.6
2	13:23	Credit card	324.31	4.761905	16.2155	7.4
3	20:33	Ewallet	465.76	4.761905	23.2880	8.4
4	10:37	Ewallet	604.17	4.761905	30.2085	5.3

2 Exploratory Data Analysis

[4]:
```
df.shape
```

[4]:
```
(1000, 17)
```

[5]:
```
df.columns
```

[5]:
```
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')
```

cogs stands for cost of good sale

[6]:
```
#check for null values
df.isnull().sum()
```

[6]:
```
Invoice ID                 0
Branch                     0
City                       0
Customer type              0
Gender                     0
Product line               0
Unit price                 0
Quantity                   0
Tax 5%                     0
Total                      0
Date                       0
Time                       0
Payment                    0
cogs                       0
gross margin percentage    0
gross income               0
Rating                     0
dtype: int64
```

[7]:
```
df.describe().T
```

[7]:

	count	mean	std	min	\
Unit price	1000.0	55.672130	2.649463e+01	10.080000	
Quantity	1000.0	5.510000	2.923431e+00	1.000000	
Tax 5%	1000.0	15.379369	1.170883e+01	0.508500	
Total	1000.0	322.966749	2.458853e+02	10.678500	
cogs	1000.0	307.587380	2.341765e+02	10.170000	
gross margin percentage	1000.0	4.761905	6.220360e-14	4.761905	
gross income	1000.0	15.379369	1.170883e+01	0.508500	

```
Rating                    1000.0     6.972700  1.718580e+00    4.000000

                                25%         50%         75%          max
Unit price                32.875000   55.230000   77.935000    99.960000
Quantity                   3.000000    5.000000    8.000000    10.000000
Tax 5%                     5.924875   12.088000   22.445250    49.650000
Total                    124.422375  253.848000  471.350250  1042.650000
cogs                     118.497500  241.760000  448.905000   993.000000
gross margin percentage    4.761905    4.761905    4.761905     4.761905
gross income               5.924875   12.088000   22.445250    49.650000
Rating                     5.500000    7.000000    8.500000    10.000000
```

[8]:
```
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
```

[9]:
```
#convert 'Date' column datatype into int
df['Date']
```

[9]:
```
0       1/5/2019
1       3/8/2019
2       3/3/2019
3      1/27/2019
4       2/8/2019
         ...
```

```
995    1/29/2019
996     3/2/2019
997     2/9/2019
998    2/22/2019
999    2/18/2019
Name: Date, Length: 1000, dtype: object
```

[10]:
```
df['Day']=pd.to_datetime(df['Date']).dt.day
df['Month']=pd.to_datetime(df['Date']).dt.month
df['Year']=pd.to_datetime(df['Date']).dt.year
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>
```

[11]:
```
#drop the "Date" column
df.drop(columns='Date',inplace=True)
```

[12]:
```
df.head(2)
```

[12]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Time	Payment	cogs	gross margin percentage	gross income	Rating	Day	Month	Year
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	13:08	Ewallet	522.83	4.761905	26.1415	9.1	5	1	2019
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	10:29	Cash	76.40	4.761905	3.8200	9.6	8	3	2019

[13]:
```
#Convert The 'Time column datatype to int'
df["Time"].unique()
```

[13]:
```
array(['13:08', '10:29', '13:23', '20:33', '10:37', '18:30', '14:36',
       '11:38', '17:15', '13:27', '18:07', '17:03', '10:25', '16:48',
       '19:21', '16:19', '11:03', '10:39', '18:00', '15:30', '11:24',
       '10:40', '12:20', '11:15', '17:36', '19:20', '15:31', '12:17',
       '19:48', '15:36', '19:39', '12:43', '14:49', '10:12', '10:42',
```

```
'12:28', '19:15', '17:17', '13:24', '13:01', '18:45', '10:11',
'13:03', '20:39', '19:47', '17:24', '15:47', '12:45', '17:08',
'10:19', '15:10', '14:42', '15:46', '11:49', '19:01', '11:26',
'11:28', '15:55', '20:36', '17:47', '10:55', '13:40', '12:27',
'14:35', '16:40', '15:43', '15:01', '10:04', '18:50', '12:46',
'18:17', '18:21', '17:04', '14:20', '15:48', '16:24', '18:56',
'19:56', '18:37', '10:17', '14:31', '10:23', '20:35', '16:57',
'17:55', '19:54', '16:42', '12:09', '20:05', '20:38', '13:11',
'10:16', '18:14', '13:22', '11:27', '16:44', '18:19', '14:50',
'20:54', '20:19', '10:43', '14:30', '11:32', '10:41', '12:44',
'20:07', '20:31', '12:29', '15:26', '20:48', '12:02', '17:26',
'19:52', '14:57', '18:44', '13:26', '16:17', '15:57', '13:18',
'20:34', '18:36', '14:40', '16:43', '20:59', '15:39', '12:21',
'19:25', '13:00', '13:48', '19:57', '10:36', '16:37', '17:11',
'15:07', '16:07', '11:56', '18:23', '13:05', '19:40', '13:58',
'14:43', '19:18', '16:21', '19:44', '19:42', '15:24', '14:12',
'13:32', '16:20', '16:31', '11:36', '19:17', '17:34', '12:04',
'17:01', '10:50', '19:16', '16:47', '10:00', '11:51', '15:00',
'11:19', '19:46', '19:00', '10:53', '12:50', '20:50', '13:41',
'19:08', '20:23', '11:30', '19:30', '18:03', '10:13', '19:58',
'10:01', '11:57', '10:02', '14:51', '12:42', '17:38', '20:24',
'18:08', '15:53', '15:05', '18:27', '16:55', '12:58', '18:59',
'13:44', '13:46', '18:06', '12:38', '15:56', '14:29', '19:14',
'10:52', '12:55', '19:28', '13:52', '10:54', '18:31', '18:24',
'18:09', '15:16', '17:07', '19:26', '11:20', '16:49', '12:01',
'11:25', '18:42', '14:47', '19:43', '14:04', '16:11', '19:06',
'15:34', '11:22', '11:23', '10:46', '13:25', '14:53', '19:22',
'11:00', '19:24', '17:22', '20:55', '16:05', '13:34', '18:13',
'11:44', '15:51', '16:52', '20:52', '16:28', '13:29', '11:09',
'15:02', '14:21', '18:01', '13:30', '14:38', '17:37', '17:20',
'20:29', '11:46', '13:42', '14:44', '14:16', '15:54', '10:21',
'16:46', '20:14', '17:09', '17:43', '19:05', '10:08', '13:12',
'20:51', '17:29', '11:34', '18:58', '20:26', '15:08', '13:21',
'12:48', '19:53', '19:09', '16:30', '13:07', '18:48', '17:27',
'15:59', '11:21', '15:49', '13:02', '20:21', '15:04', '16:10',
'12:14', '11:06', '18:22', '19:02', '15:44', '20:01', '13:45',
'15:40', '16:58', '11:12', '15:12', '20:37', '17:44', '16:23',
'12:12', '19:33', '14:28', '17:54', '12:25', '12:52', '19:50',
'15:32', '13:19', '13:37', '14:55', '12:31', '10:26', '20:18',
'20:04', '13:38', '17:30', '15:28', '19:07', '18:55', '19:36',
'10:57', '17:13', '13:57', '13:53', '16:53', '16:51', '15:37',
'20:15', '19:35', '15:42', '14:11', '17:58', '11:02', '15:09',
'13:47', '16:59', '14:15', '15:19', '18:33', '12:10', '11:40',
'16:54', '15:25', '20:47', '18:20', '11:48', '14:14', '11:17',
'12:40', '17:53', '16:36', '10:48', '18:05', '12:07', '19:49',
'15:52', '20:46', '10:34', '13:55', '11:43', '16:03', '20:03',
'19:41', '18:04', '10:31', '13:28', '17:16', '18:43', '10:30',
```

```
       '20:40', '12:08', '17:45', '10:28', '10:49', '12:34', '18:51',
       '19:38', '12:32', '10:33', '19:55', '14:33', '13:54', '12:15',
       '12:37', '15:06', '15:58', '14:03', '16:38', '11:07', '12:23',
       '14:13', '19:11', '18:53', '14:22', '10:06', '20:08', '12:56',
       '10:18', '11:45', '16:08', '12:24', '19:51', '18:10', '15:27',
       '16:04', '14:41', '14:19', '14:08', '11:29', '12:16', '20:00',
       '15:29', '14:58', '11:52', '17:46', '14:45', '11:39', '13:06',
       '20:43', '16:34', '13:10', '17:10', '10:22', '19:29', '14:27',
       '12:22', '11:59', '17:59', '12:51', '13:56', '19:45', '16:18',
       '18:57', '11:18', '14:06', '20:13', '15:14', '16:06', '12:47',
       '20:42', '20:10', '14:24', '11:42', '17:49', '15:33', '10:38',
       '12:39', '14:26', '12:41', '15:20', '16:33', '20:44', '11:16',
       '12:30', '17:48', '20:30', '13:59', '11:58', '16:50', '18:02',
       '17:52', '20:32', '16:09', '11:33', '15:15', '20:06', '16:26',
       '18:38', '16:45', '18:41', '17:12', '14:00', '16:32', '10:10',
       '10:05', '18:15', '11:01', '15:21', '16:16', '11:05', '19:31',
       '18:35', '13:51', '12:35', '11:55', '15:11', '14:48', '12:36',
       '13:35', '15:45', '14:25', '15:18', '10:03', '13:14', '16:35',
       '20:57', '13:50', '17:35', '17:56', '10:44', '10:09', '10:58',
       '13:49', '11:10', '13:33', '14:05', '16:27', '15:23', '18:18',
       '15:17', '19:12'], dtype=object)
```

[14]:
```
df['Hour']=pd.to_datetime(df['Time']).dt.hour
df['Minutes']=pd.to_datetime(df['Time']).dt.minute
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>
```

[15]:
```
#drop the Time column
df.drop(columns='Time',inplace=True)
```

[16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 20 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  Payment                  1000 non-null   object
 11  cogs                     1000 non-null   float64
 12  gross margin percentage  1000 non-null   float64
 13  gross income             1000 non-null   float64
 14  Rating                   1000 non-null   float64
 15  Day                      1000 non-null   int64
 16  Month                    1000 non-null   int64
 17  Year                     1000 non-null   int64
 18  Hour                     1000 non-null   int64
 19  Minutes                  1000 non-null   int64
dtypes: float64(7), int64(6), object(7)
memory usage: 156.4+ KB
```

[17]:
```
df.head(2)
```

[17]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Payment	cogs	gross margin percentage	gross income	Rating	Day	Month	Year	Hour	Minutes
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	Ewallet	522.83	4.761905	26.1415	9.1	5	1	2019	13	8
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	Cash	76.40	4.761905	3.8200	9.6	8	3	2019	10	29

3 In which month the 'gross income' was maximum

[18]:
```
df.groupby(['Month'])["gross income"].max().reset_index()
```

[18]:

	Month	gross income
0	1	49.26
1	2	49.65
2	3	48.69

In January and Feburary the gross income was high

4 Which Products were sold in Maximum Quantity with respect to Month

[19]:
```
df.groupby(["Product line","Month"])['Quantity'].size().reset_index()
```

[19]:

	Product line	Month	Quantity
0	Electronic accessories	1	54
1	Electronic accessories	2	54
2	Electronic accessories	3	62
3	Fashion accessories	1	64
4	Fashion accessories	2	60
5	Fashion accessories	3	54
6	Food and beverages	1	56
7	Food and beverages	2	62
8	Food and beverages	3	56
9	Health and beauty	1	49
10	Health and beauty	2	46
11	Health and beauty	3	57
12	Home and lifestyle	1	59
13	Home and lifestyle	2	38
14	Home and lifestyle	3	63
15	Sports and travel	1	70
16	Sports and travel	2	43
17	Sports and travel	3	53

Electronic accessories : Maximum Quantity sold "62" in Feburary

Fashion accessories : Maximum Quantity sold "64" in January

Food and beverages : Maximum Quantity sold "62" in Feburary

Health and beauty : Maximum Quantity sold "57" in March

Home and lifestyle : Maximum Quantity sold "63" in March

Sports and travel : Maximum Quantity sold "70" in January

5 On Which features "Tax5%" Depends

[20]:
```
#Finding the correlation among integral variables
df.corr()
```

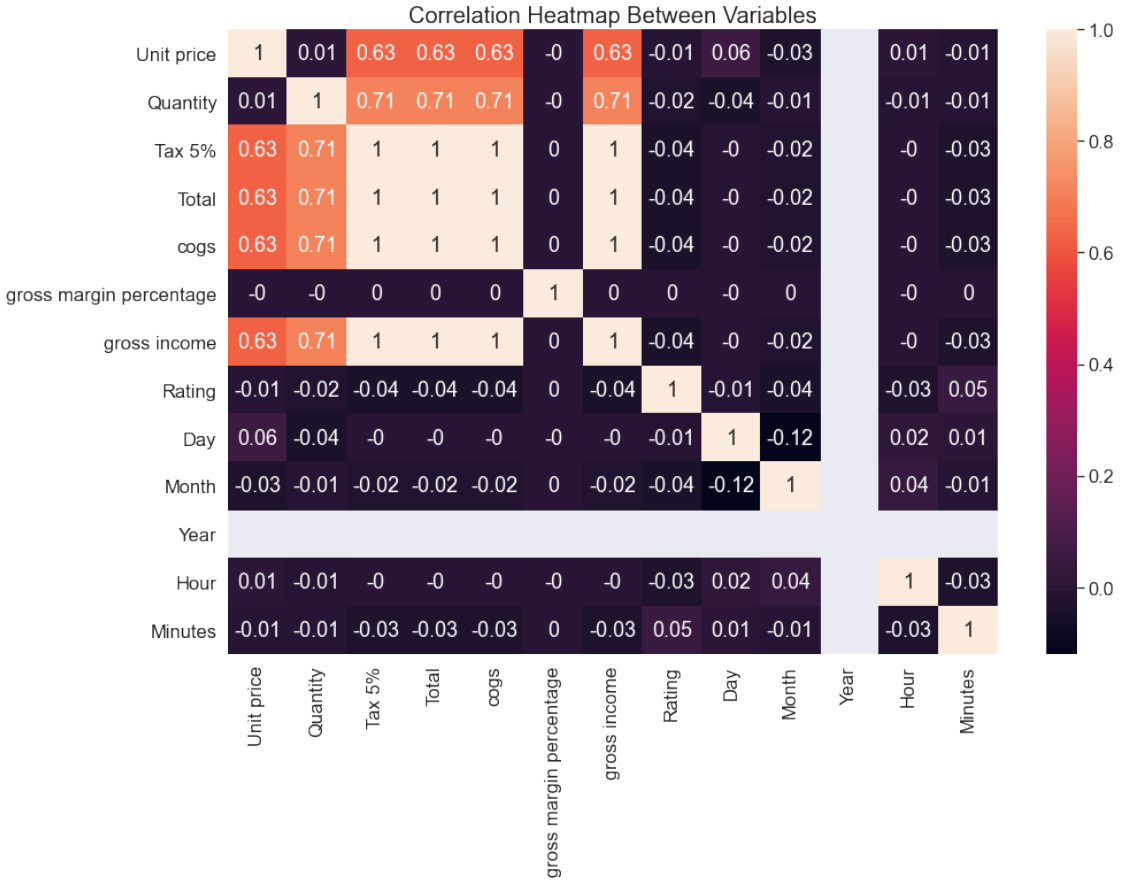
[20]:

	Unit price	Quantity	Tax 5%	\
Unit price	1.000000e+00	1.077756e-02	6.339621e-01	
Quantity	1.077756e-02	1.000000e+00	7.055102e-01	
Tax 5%	6.339621e-01	7.055102e-01	1.000000e+00	
Total	6.339621e-01	7.055102e-01	1.000000e+00	
cogs	6.339621e-01	7.055102e-01	1.000000e+00	
gross margin percentage	-6.998957e-16	-3.849075e-16	2.461896e-16	

```
gross income             6.339621e-01  7.055102e-01  1.000000e+00
Rating                  -8.777507e-03 -1.581490e-02 -3.644170e-02
Day                      5.702090e-02 -4.334686e-02 -2.514770e-03
Month                   -2.738719e-02 -1.452428e-02 -2.230134e-02
Year                              NaN           NaN           NaN
Hour                     8.242210e-03 -7.316886e-03 -2.770440e-03
Minutes                 -6.868818e-03 -1.492856e-02 -2.747990e-02

                                Total          cogs  gross margin percentage  \
Unit price               6.339621e-01  6.339621e-01             -6.998957e-16
Quantity                 7.055102e-01  7.055102e-01             -3.849075e-16
Tax 5%                   1.000000e+00  1.000000e+00              2.461896e-16
Total                    1.000000e+00  1.000000e+00              2.408632e-16
cogs                     1.000000e+00  1.000000e+00              1.439279e-15
gross margin percentage  2.408632e-16  1.439279e-15              1.000000e+00
gross income             1.000000e+00  1.000000e+00              2.461896e-16
Rating                  -3.644170e-02 -3.644170e-02              2.042714e-15
Day                     -2.514770e-03 -2.514770e-03             -4.999832e-16
Month                   -2.230134e-02 -2.230134e-02              8.935965e-17
Year                              NaN           NaN                       NaN
Hour                    -2.770440e-03 -2.770440e-03             -2.114722e-16
Minutes                 -2.747990e-02 -2.747990e-02              1.288421e-16

                         gross income        Rating           Day  \
Unit price               6.339621e-01 -8.777507e-03  5.702090e-02
Quantity                 7.055102e-01 -1.581490e-02 -4.334686e-02
Tax 5%                   1.000000e+00 -3.644170e-02 -2.514770e-03
Total                    1.000000e+00 -3.644170e-02 -2.514770e-03
cogs                     1.000000e+00 -3.644170e-02 -2.514770e-03
gross margin percentage  2.461896e-16  2.042714e-15 -4.999832e-16
gross income             1.000000e+00 -3.644170e-02 -2.514770e-03
Rating                  -3.644170e-02  1.000000e+00 -7.075821e-03
Day                     -2.514770e-03 -7.075821e-03  1.000000e+00
Month                   -2.230134e-02 -4.288037e-02 -1.189964e-01
Year                              NaN           NaN           NaN
Hour                    -2.770440e-03 -3.058764e-02  2.066810e-02
Minutes                 -2.747990e-02  5.055848e-02  1.264550e-02

                                Month  Year          Hour       Minutes
Unit price              -2.738719e-02   NaN  8.242210e-03 -6.868818e-03
Quantity                -1.452428e-02   NaN -7.316886e-03 -1.492856e-02
Tax 5%                  -2.230134e-02   NaN -2.770440e-03 -2.747990e-02
Total                   -2.230134e-02   NaN -2.770440e-03 -2.747990e-02
cogs                    -2.230134e-02   NaN -2.770440e-03 -2.747990e-02
gross margin percentage  8.935965e-17   NaN -2.114722e-16  1.288421e-16
gross income            -2.230134e-02   NaN -2.770440e-03 -2.747990e-02
Rating                  -4.288037e-02   NaN -3.058764e-02  5.055848e-02
```

Day	-1.189964e-01	NaN	2.066810e-02	1.264550e-02
Month	1.000000e+00	NaN	4.376174e-02	-6.553809e-03
Year	NaN	NaN	NaN	NaN
Hour	4.376174e-02	NaN	1.000000e+00	-2.538363e-02
Minutes	-6.553809e-03	NaN	-2.538363e-02	1.000000e+00

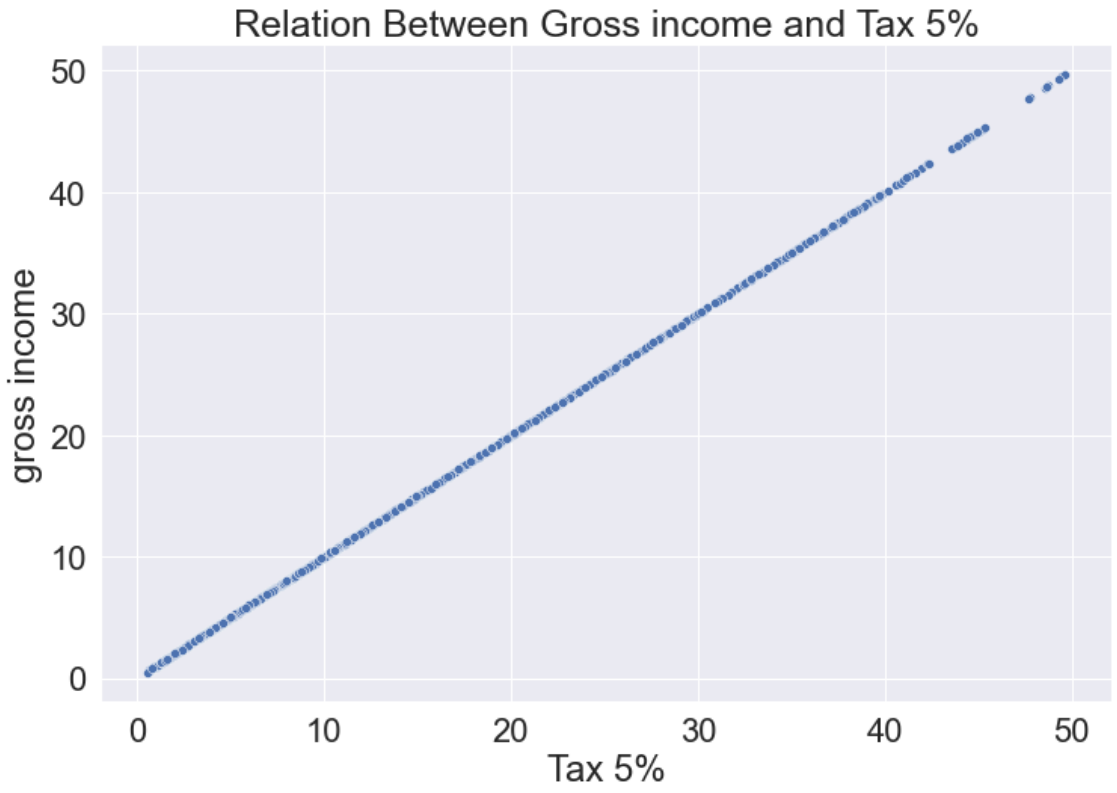
[21]:
```
#Heatmap of variables
plt.figure(figsize=(15,10))
sns.set(font_scale=1.5)
heatmap=sns.heatmap(np.round(df.corr(),2),annot=True,linecolor='white')
heatmap.set_title('Correlation Heatmap Between Variables',fontsize=20);
```



'Tax5%' depends on 'gross income' and "Quantity"

6 What is the Relation Between 'Tax5%' and 'gross income'

[22]:
```
plt.figure(figsize=(12,8))
sns.set(font_scale=2)
sns.scatterplot(x='Tax 5%',y='gross income',data=df)
plt.title('Relation Between Gross income and Tax 5%',fontsize=25);
```



'gross income' and 'Tax5% highly Positive correlated

7 What kind of relation between "Quantity" and "Tax5%"

[23]:
```
plt.figure(figsize=(12,8))
sns.regplot(x='Quantity',y='Tax 5%',data=df,color='blue')
plt.title('Relation Between Quantity and Tax 5%',fontsize=25)
```

[23]:
```
Text(0.5, 1.0, 'Relation Between Quantity and Tax 5%')
```

"Quantity" and 'Tax5% highly Positive correlated

[24]:
```
#which city has more gross income
df['City'].unique()
```

[24]:
```
array(['Yangon', 'Naypyitaw', 'Mandalay'], dtype=object)
```

[25]:
```
df.groupby(['City'])['gross income'].median()
```

[25]:
```
City
Mandalay     12.04200
Naypyitaw    12.92475
Yangon       11.46800
Name: gross income, dtype: float64
```

Naypyitaw has more gross income than other

8 In which year gross income was maximum

[26]:
```
#df.groupby(['Year'])['gross income'].max()
```

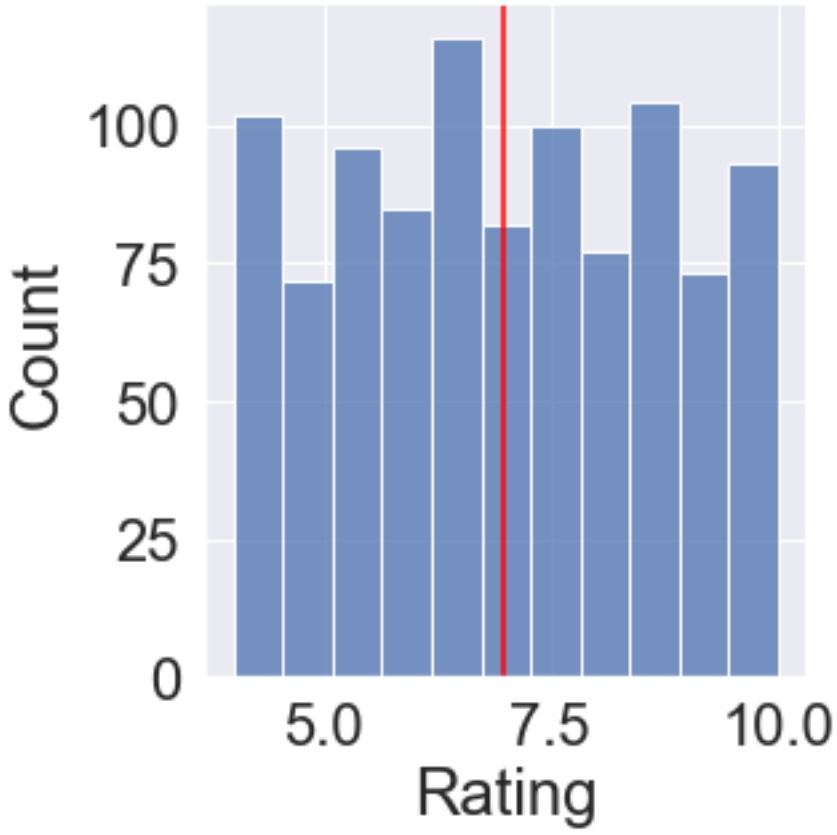
9 What is the mean of rating

[27]:
```
df['Rating'].mean()
```

[27]:
```
6.972700000000003
```

[28]:
```
#find mean ratiing and visualize it
plt.figure(figsize=(8,8))
sns.set(font_scale=2)
sns.displot(df['Rating'])
plt.axvline(x=np.mean(df['Rating']),c='red',label="Avg Rating")
plt.show()
```

```
<Figure size 576x576 with 0 Axes>
```



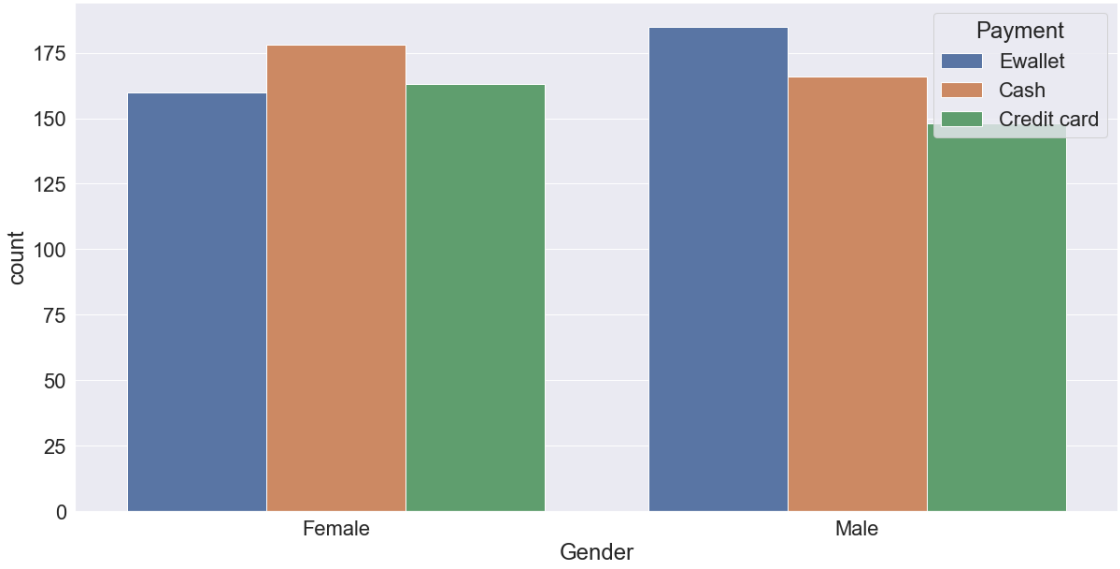
10 Which kind of Payment method is mostly used by Male and Females

[29]:
```
df['Gender'].value_counts()
```

[29]:
```
Female    501
Male      499
Name: Gender, dtype: int64
```

[30]:
```
plt.figure(figsize=(20,10))
sns.set(font_scale=2)
sns.countplot('Gender',data=df,hue='Payment')
plt.show()
```

```
C:\Users\hp\AppData\Roaming\Python\Python38\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable
as a keyword arg: x. From version 0.12, the only valid positional argument will
be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```



Females are mostly paying through Cash

Males are mostly paying through Ewallet

[31]:
```
df.columns
```

[31]:
```
Index(['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender',
       'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Payment',
```

```
       'cogs', 'gross margin percentage', 'gross income', 'Rating', 'Day',
       'Month', 'Year', 'Hour', 'Minutes'],
      dtype='object')
```

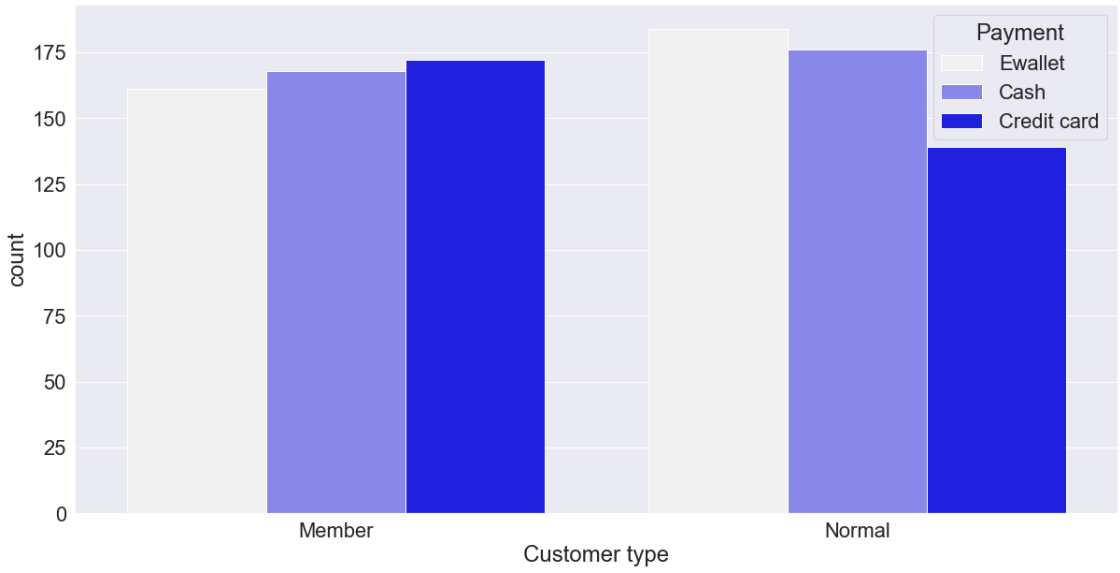
11 Which kind of Payment method is mostly used by Member And Normal people

[32]:
```
df['Customer type'].unique()
```

[32]:
```
array(['Member', 'Normal'], dtype=object)
```

[33]:
```
plt.figure(figsize=(20,10))
sns.set(font_scale=2)
sns.countplot('Customer type',data=df,hue='Payment',color='blue')
plt.show()
```

```
C:\Users\hp\AppData\Roaming\Python\Python38\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable
as a keyword arg: x. From version 0.12, the only valid positional argument will
be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```



Members are mostly paying through Credit Card

Normal people are mostly paying through Ewallet

[34]:
```
df['Product line'].unique()
```

[34]:
```
array(['Health and beauty', 'Electronic accessories',
       'Home and lifestyle', 'Sports and travel', 'Food and beverages',
       'Fashion accessories'], dtype=object)
```

[35]:
```
df['gross income']=df['gross income'].astype(int)
```

[36]:
```
df.head(2)
```

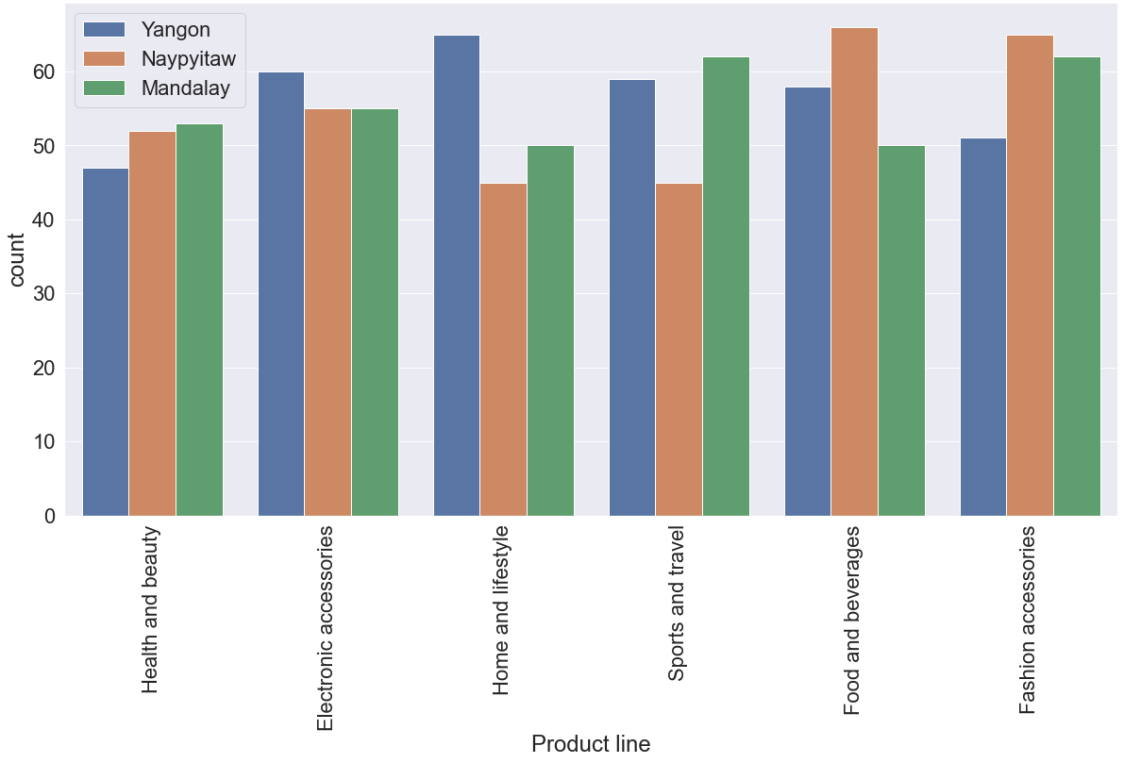
[36]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Payment	cogs	gross margin percentage	gross income	Rating	Day	Month	Year	Hour	Minutes
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	Ewallet	522.83	4.761905	26	9.1	5	1	2019	13	8
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	Cash	76.40	4.761905	3	9.6	8	3	2019	10	29

12 Which type of products are mostly sold in each city

[37]:
```
plt.figure(figsize=(20,10))
sns.set(font_scale=2)
sns.countplot('Product line',data=df,hue='City')
plt.xticks(rotation=90)
plt.legend(loc='upper left');
```

```
C:\Users\hp\AppData\Roaming\Python\Python38\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable
as a keyword arg: x. From version 0.12, the only valid positional argument will
be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```



13 What is relation between gross income and unit price

[38]:
```
plt.figure(figsize=(7,7))
sns.regplot(x='gross income',y='Unit price',data=df,color='g')
plt.title('Relation Between Unit price and gross income',fontsize=20);
```

Unit price and gross income are positively correlated

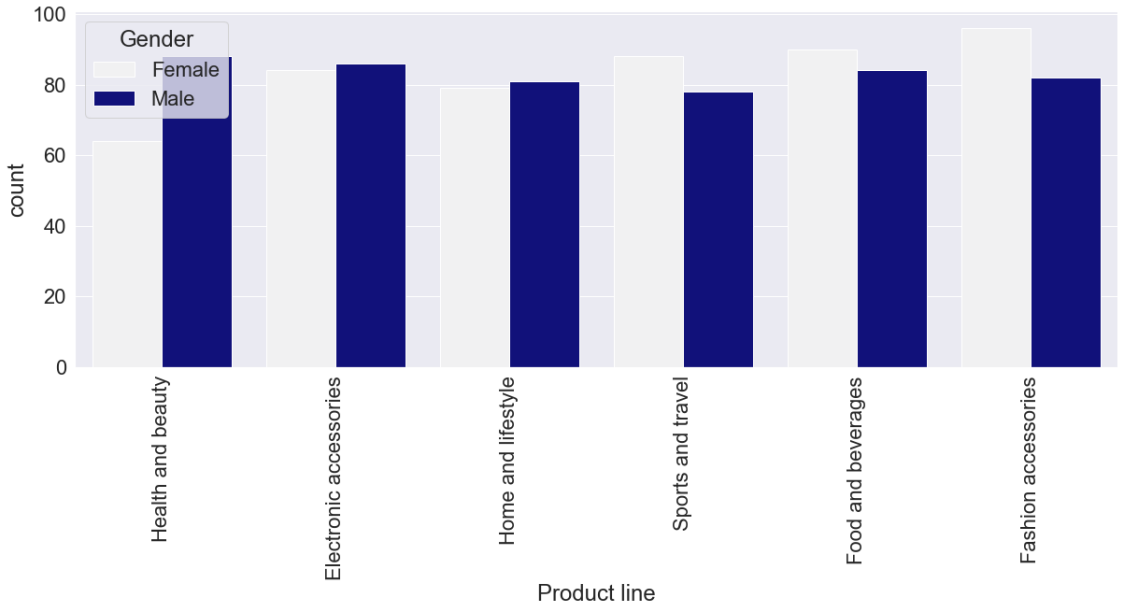
gross income increases with unit price

14 Which products are mostly purchased by Male And female

[39]:
```
plt.figure(figsize=(20,7))
sns.countplot('Product line',data=df,hue='Gender',color='darkblue')
plt.xticks(rotation=90)
plt.show()
```

```
C:\Users\hp\AppData\Roaming\Python\Python38\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable
as a keyword arg: x. From version 0.12, the only valid positional argument will
```

```
be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```



Mostly males purchased health and beauty products

Mostly females purchased Fashion accessories

15 How much members are in each city

[40]:
```
df.groupby(['City',"Customer type"]).size().reset_index()
```

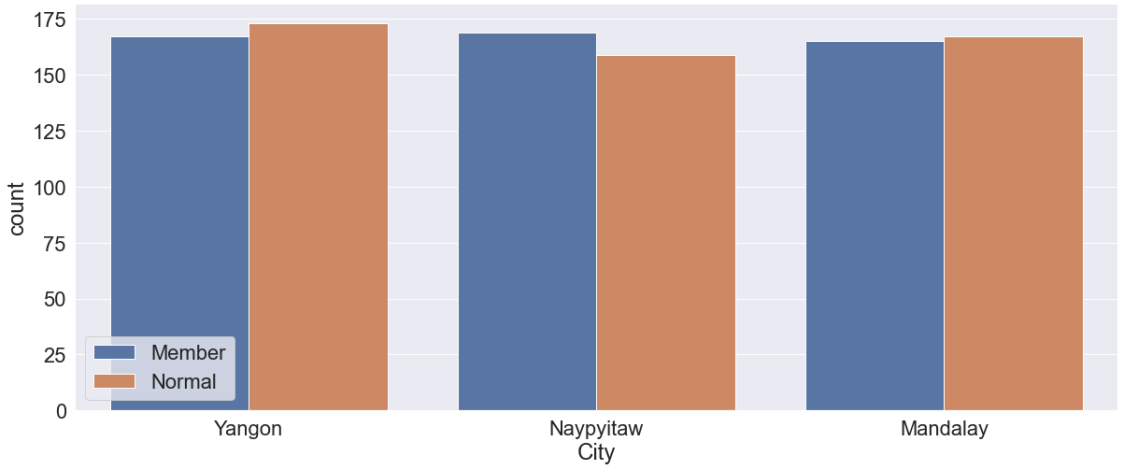
[40]:

	City	Customer type	0
0	Mandalay	Member	165
1	Mandalay	Normal	167
2	Naypyitaw	Member	169
3	Naypyitaw	Normal	159
4	Yangon	Member	167
5	Yangon	Normal	173

[41]:
```
#Count city based on custumer type
plt.figure(figsize=(20,8))
sns.countplot('City',data=df,hue='Customer type')
plt.legend(loc="lower left")
plt.show()
```

```
C:\Users\hp\AppData\Roaming\Python\Python38\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable
```

```
as a keyword arg: x. From version 0.12, the only valid positional argument will
be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```

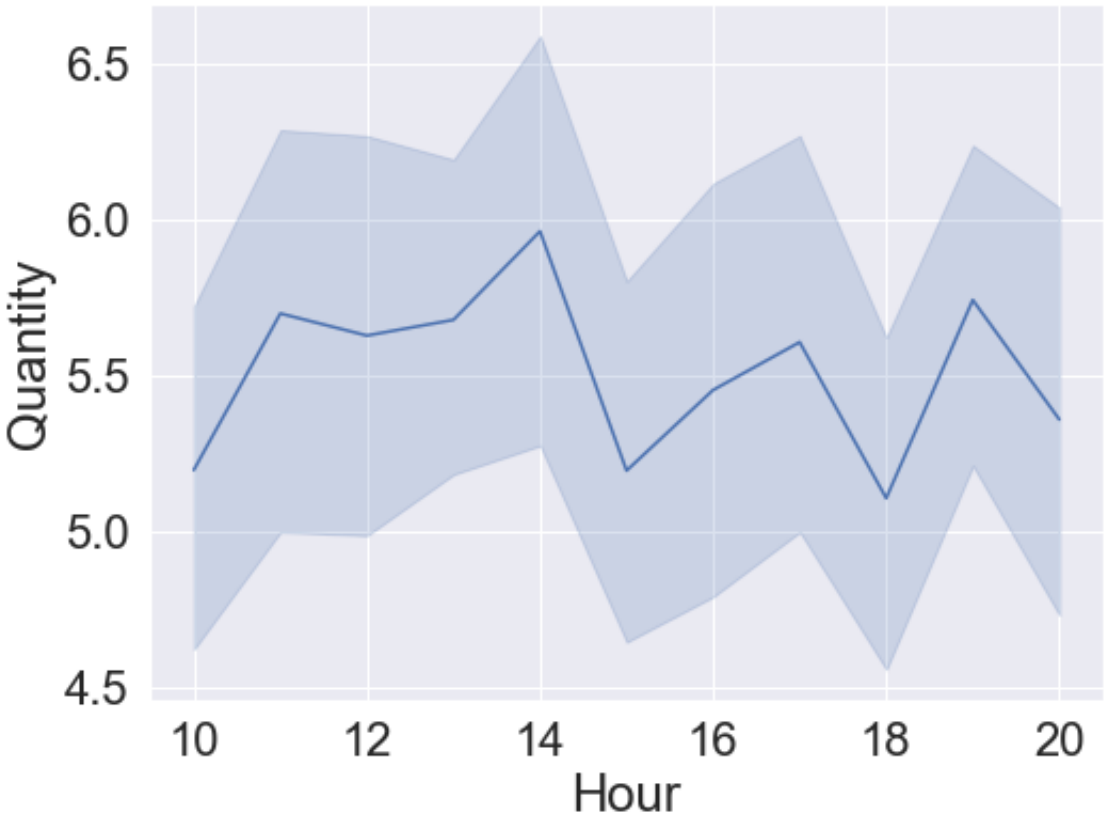


Maximum members are from Naypyitaw city

16 On which hour maximum quantities were sold

[42]:
```
#trend of selling products by time
plt.figure(figsize=(8,6))
sns.lineplot('Hour','Quantity',data=df);
```

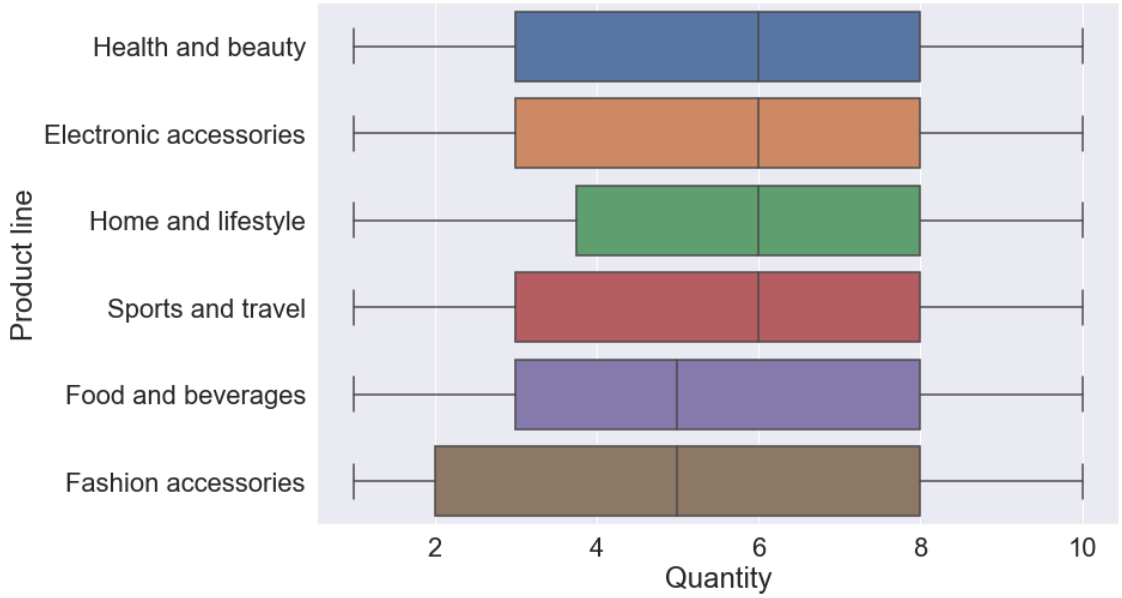
```
C:\Users\hp\AppData\Roaming\Python\Python38\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```



At 2pm products were sold in maximum quantity

[43]:
```
plt.figure(figsize=(12,8))
sns.boxplot('Quantity','Product line',data=df)
plt.show()
```

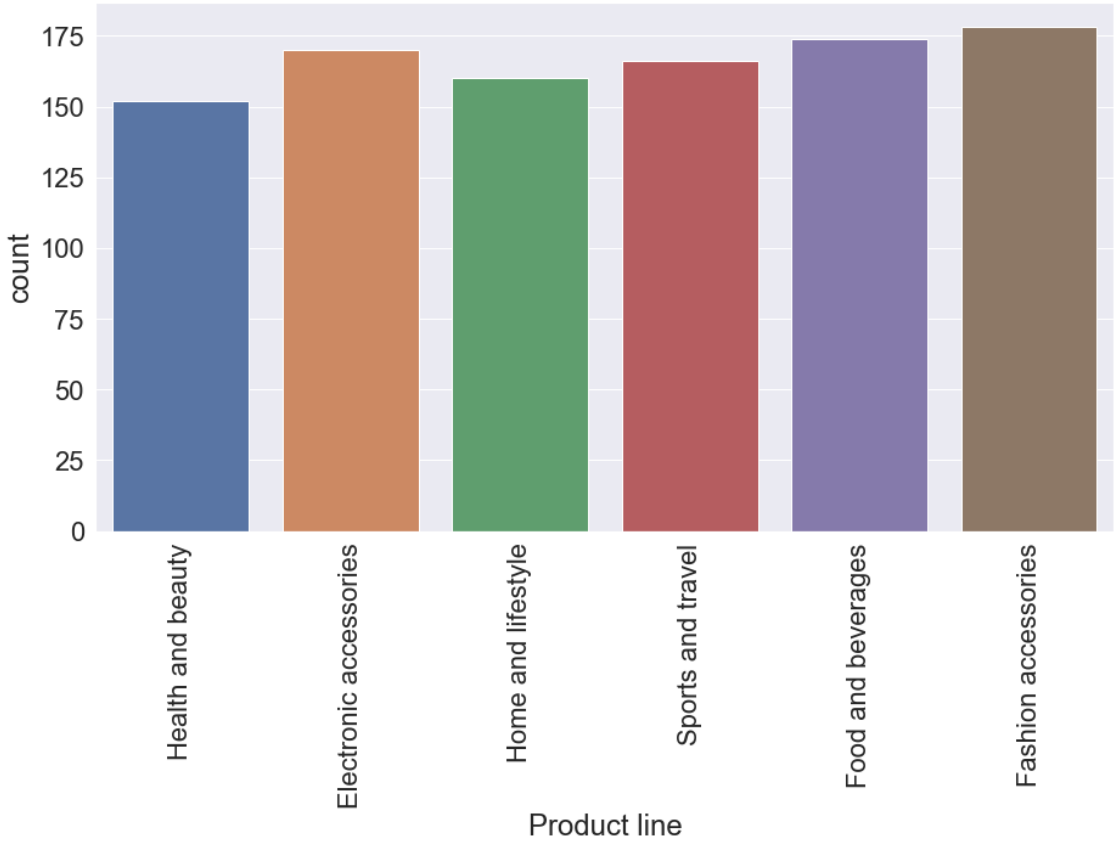
```
C:\Users\hp\AppData\Roaming\Python\Python38\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```



17 Which product is sold more

[44]:
```
plt.figure(figsize=(15,8))
sns.countplot('Product line',data=df)
plt.xticks(rotation=90)
plt.show()
```

```
C:\Users\hp\AppData\Roaming\Python\Python38\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable
as a keyword arg: x. From version 0.12, the only valid positional argument will
be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```

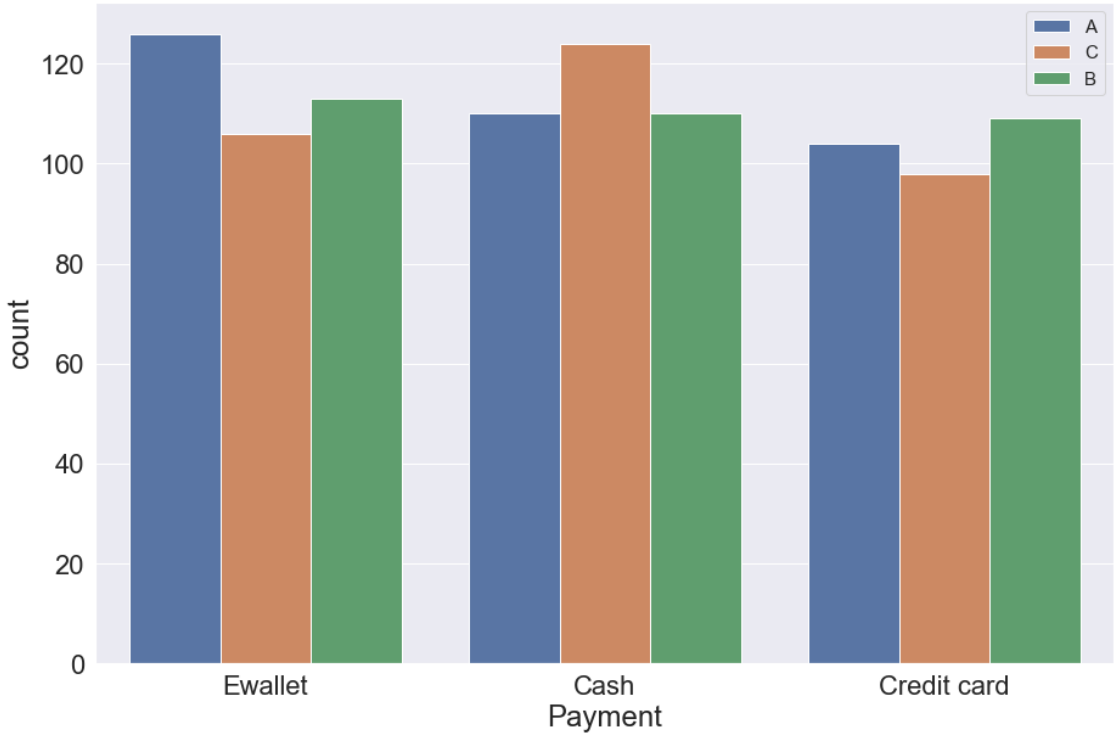


Fashion accessories are sold more than other products

18 On Each branch which payment method is mostly prefferd by Customers

[45]:
```
#Customers payment in this bussiness
plt.figure(figsize=(15,10))
sns.countplot('Payment',data=df,hue='Branch')
plt.legend(loc='upper right',fontsize=15)
plt.show()
```

```
C:\Users\hp\AppData\Roaming\Python\Python38\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable
as a keyword arg: x. From version 0.12, the only valid positional argument will
be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```



On branch 'A' mostly prefferd payment method:Ewallet

On branch 'B' mostly prefferd payment method:Ewallet

On branch 'C' mostly prefferd payment method:Cash

19 How much maximum rating has each product

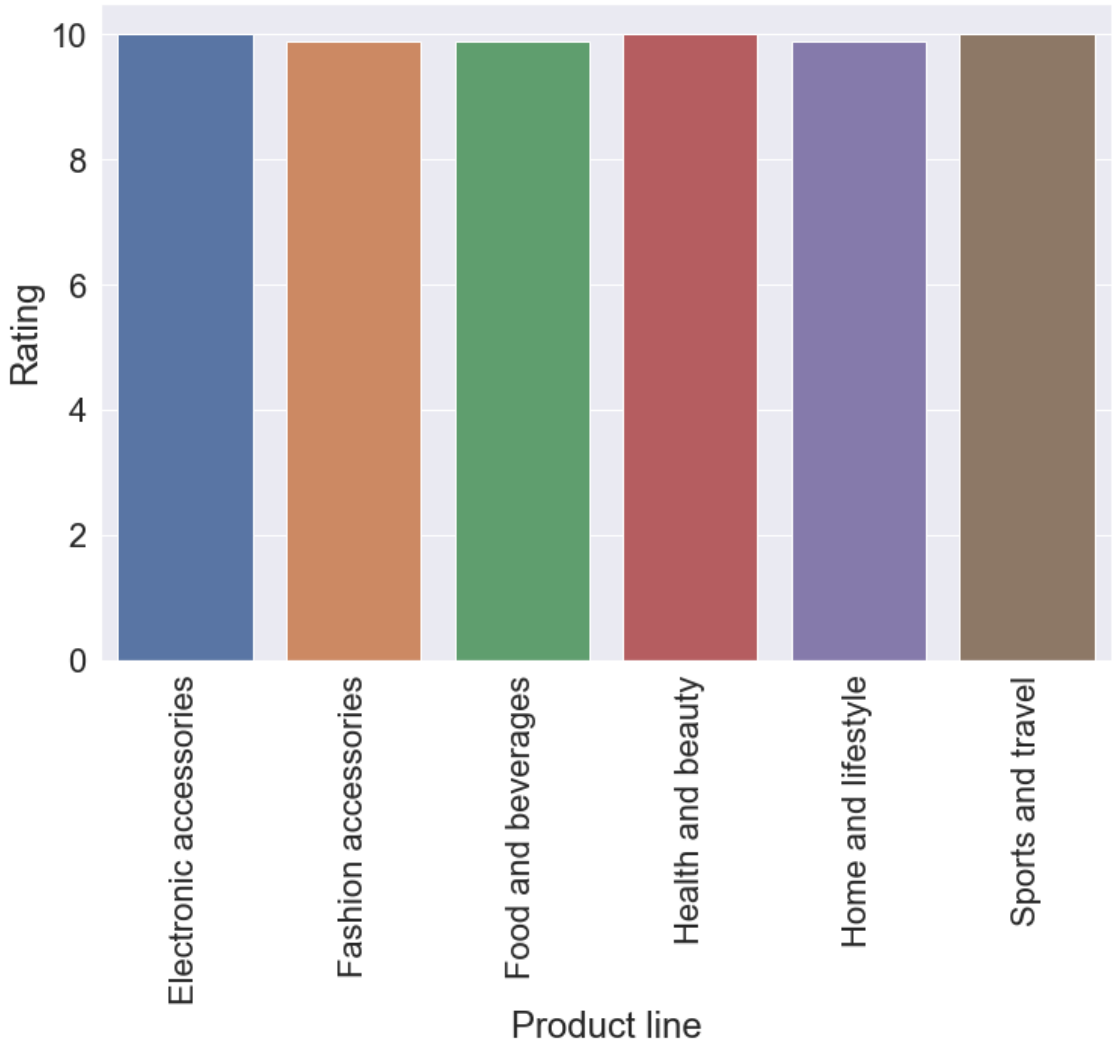
[46]:
```
max_rat=df.groupby(['Product line'])['Rating'].max().reset_index()
```

[47]:
```
max_rat
```

[47]:

	Product line	Rating
0	Electronic accessories	10.0
1	Fashion accessories	9.9
2	Food and beverages	9.9
3	Health and beauty	10.0
4	Home and lifestyle	9.9
5	Sports and travel	10.0

[56]:
```
plt.figure(figsize=(12,8))
sns.barplot(x='Product line',y='Rating',data=max_rat)
plt.xticks(rotation=90);
```


10
8
6
4
2
0
Rating
Electronic accessories
Fashion accessories
Food and beverages
Health and beauty
Home and lifestyle
Sports and travel
Product line

20 THANK YOU