Supermart Grocery Sales- Retail Analytics Dataset

April 19, 2025

1 Lakshman Chaudhary

[1]:
```python
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```python
sales = pd.read_csv('Supermart Grocery Sales - Retail Analytics Dataset.csv')
```

[3]:
```python
sales.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
```

[4]:
```python
sales.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
```

By inspection, the 'Date' datatype is an object, we need to change it to datetime

[5]:
```
sales['date'] = pd.to_datetime(sales['Date'])
```

[6]:
```
sales['date'].dtype
```

[6]:
```
dtype('<M8[ns]')
```

[7]:
```
type(sales['date'])
```

[7]:
```
pandas.core.series.Series
```

[8]:
```
sales['date'] = pd.to_datetime(sales['date'])
```

[9]:
```
sales['day'] = (sales['date']).dt.day
sales['month'] = (sales['date']).dt.month
sales['year'] = (sales['date']).dt.year
```

[10]:
```
sales['Time'] = pd.to_datetime(sales['Time'])
```

```
C:\Users\laksh\AppData\Local\Temp\ipykernel_31852\721023929.py:1: UserWarning:
Could not infer format, so each element will be parsed individually, falling
back to `dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  sales['Time'] = pd.to_datetime(sales['Time'])
```

[11]:
```
 sales['Hour'] = (sales['Time']).dt.hour    #type(sales['Time'])
```

Let's see the unique hours of sales in this dataset

[12]:
```
sales['Hour'].nunique()  #gives us the number of unique hours
```

[12]:
```
11
```

[13]:
```
  sales['Hour'].unique()
```

[13]:
```
array([13, 10, 20, 18, 14, 11, 17, 16, 19, 15, 12])
```

[14]:
```
sales.describe()
```

[14]:
```
        Unit price     Quantity       Tax 5%        Total  \
count  1000.000000  1000.000000  1000.000000  1000.000000
mean     55.672130     5.510000    15.379369   322.966749
min      10.080000     1.000000     0.508500    10.678500
25%      32.875000     3.000000     5.924875   124.422375
50%      55.230000     5.000000    12.088000   253.848000
75%      77.935000     8.000000    22.445250   471.350250
max      99.960000    10.000000    49.650000  1042.650000
std      26.494628     2.923431    11.708825   245.885335

                             Time        cogs  gross margin percentage  \
count                        1000  1000.00000              1000.000000
mean   2025-04-19 15:24:41.880000   307.58738                 4.761905
min           2025-04-19 10:00:00    10.17000                 4.761905
25%           2025-04-19 12:43:00   118.49750                 4.761905
50%           2025-04-19 15:19:00   241.76000                 4.761905
75%           2025-04-19 18:15:00   448.90500                 4.761905
max           2025-04-19 20:59:00   993.00000                 4.761905
std                           NaN   234.17651                 0.000000

       gross income      Rating                        date          day  \
count   1000.000000  1000.00000                        1000  1000.000000
mean      15.379369     6.97270  2019-02-14 00:05:45.600000    15.256000
min        0.508500     4.00000         2019-01-01 00:00:00     1.000000
25%        5.924875     5.50000         2019-01-24 00:00:00     8.000000
50%       12.088000     7.00000         2019-02-13 00:00:00    15.000000
75%       22.445250     8.50000         2019-03-08 00:00:00    23.000000
max       49.650000    10.00000         2019-03-30 00:00:00    31.000000
std       11.708825     1.71858                         NaN     8.693563

             month    year         Hour
count  1000.000000  1000.0  1000.000000
mean      1.993000  2019.0    14.910000
min       1.000000  2019.0    10.000000
25%       1.000000  2019.0    12.000000
50%       2.000000  2019.0    15.000000
75%       3.000000  2019.0    18.000000
```

```
max        3.000000   2019.0     20.000000
std        0.835254      0.0      3.186857
```

Let's find the number of unique values in columns with object datatype

[15]:
```python
categorical_columns = [cname for cname in sales.columns if sales[cname].dtype == "object"]
```

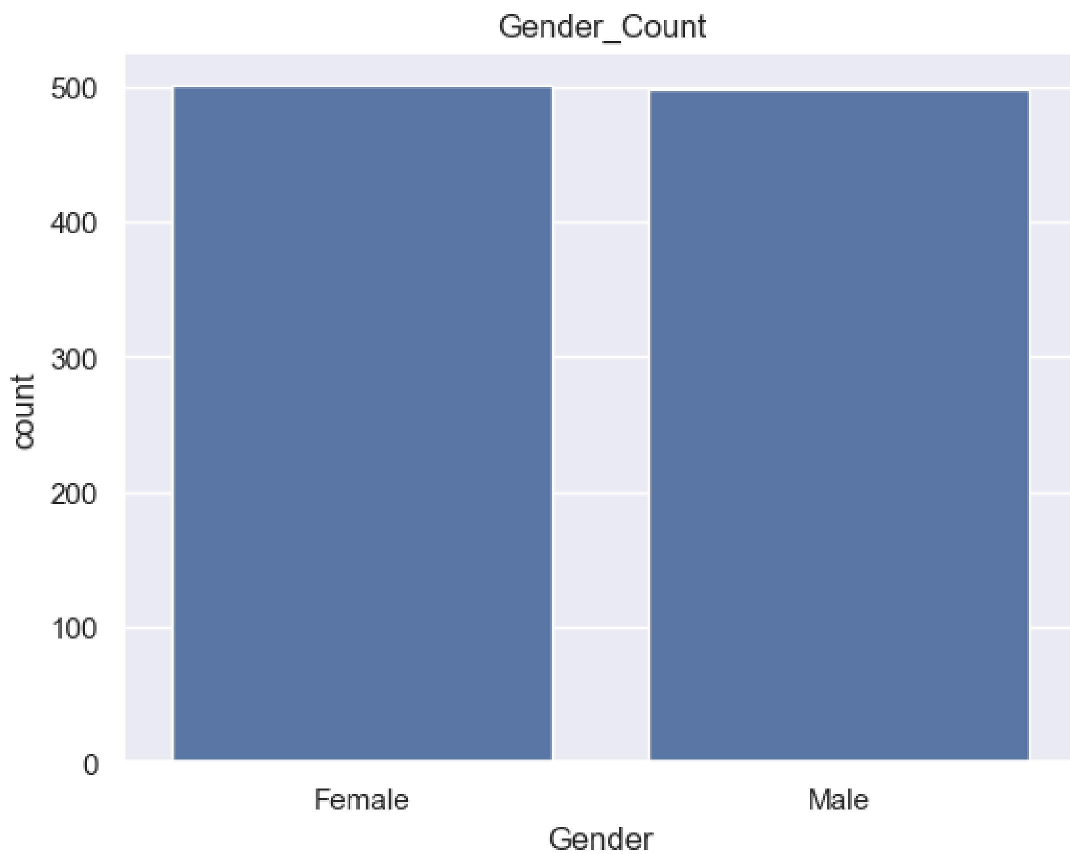
[16]:
```python
categorical_columns
```

[16]:
```
['Invoice ID',
 'Branch',
 'City',
 'Customer type',
 'Gender',
 'Product line',
 'Date',
 'Payment']
```

[17]:
```python
print("# unique values in Branch: {0}".format(len(sales['Branch'].unique().tolist())))
print("# unique values in City: {0}".format(len(sales['City'].unique().tolist())))
print("# unique values in Customer Type: {0}".format(len(sales['Customer type'].unique().tolist())))
print("# unique values in Gender: {0}".format(len(sales['Gender'].unique().tolist())))
print("# unique values in Product Line: {0}".format(len(sales['Product line'].unique().tolist())))
print("# unique values in Payment: {0}".format(len(sales['Payment'].unique().tolist())))
```

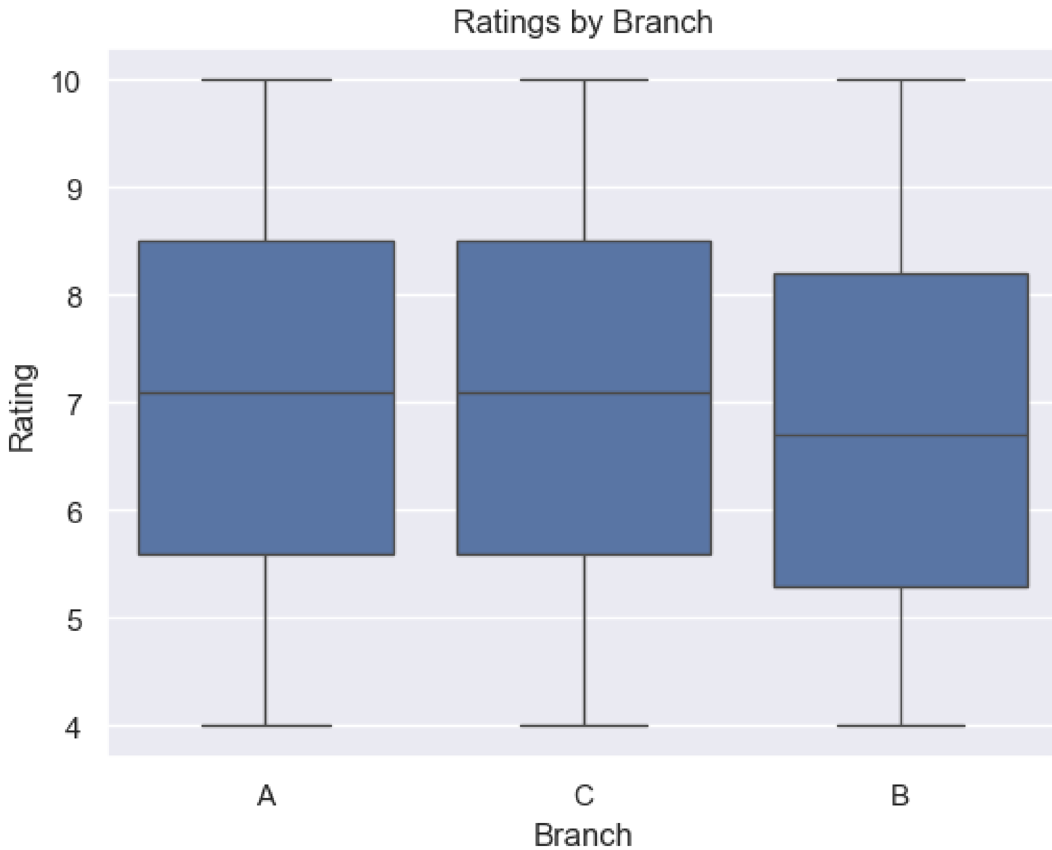
```
# unique values in Branch: 3
# unique values in City: 3
# unique values in Customer Type: 2
# unique values in Gender: 2
# unique values in Product Line: 6
# unique values in Payment: 3
```

[18]:
```python
sns.set(style="darkgrid")       #style the plot background to become a grid
genderCount  = sns.countplot(x="Gender", data =sales).set_title("Gender_Count")
```



[19]:
```
sns.boxplot(x="Branch", y = "Rating" ,data =sales).set_title("Ratings by␣
↪Branch")
```

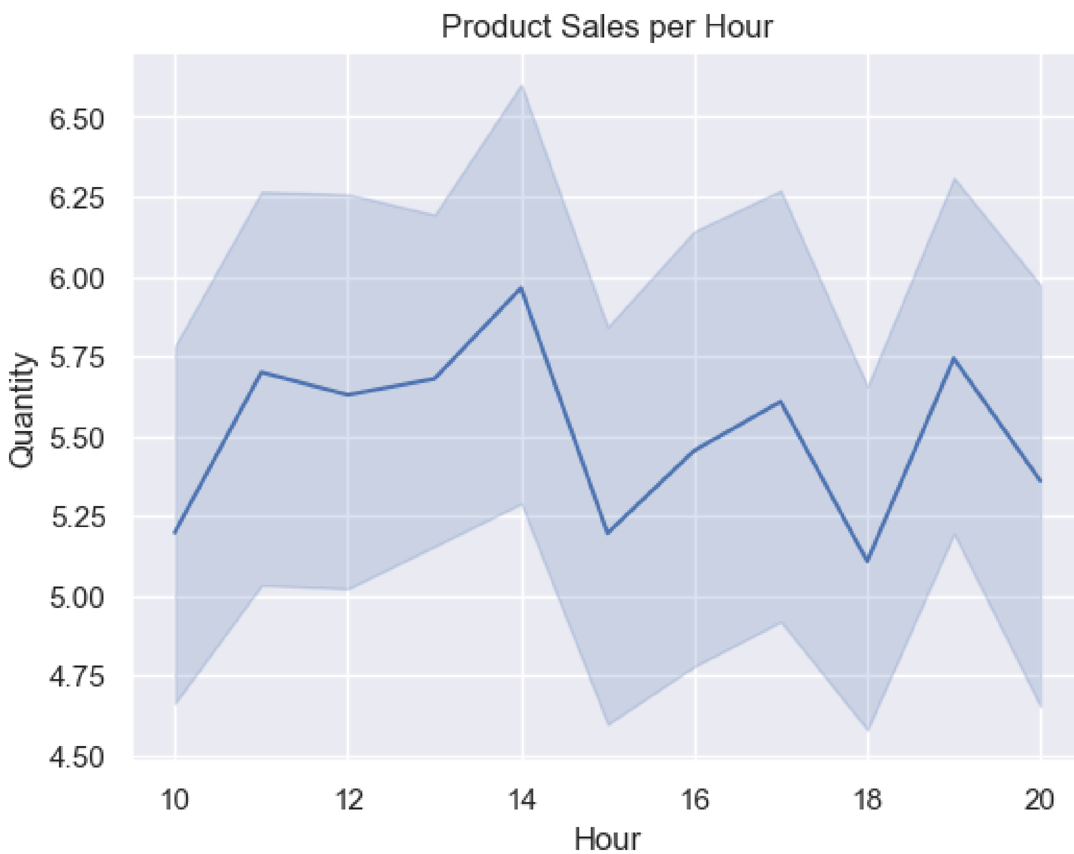
[19]: Text(0.5, 1.0, 'Ratings by Branch')



Branch B has the lowest rating among all the branches

Sales by the hour in the comapny Most of the item were sold around 14:00 hrs local time

[20]:
```
genderCount  = sns.lineplot(x="Hour",  y = 'Quantity',data =sales).
 ↪set_title("Product Sales per Hour")
```

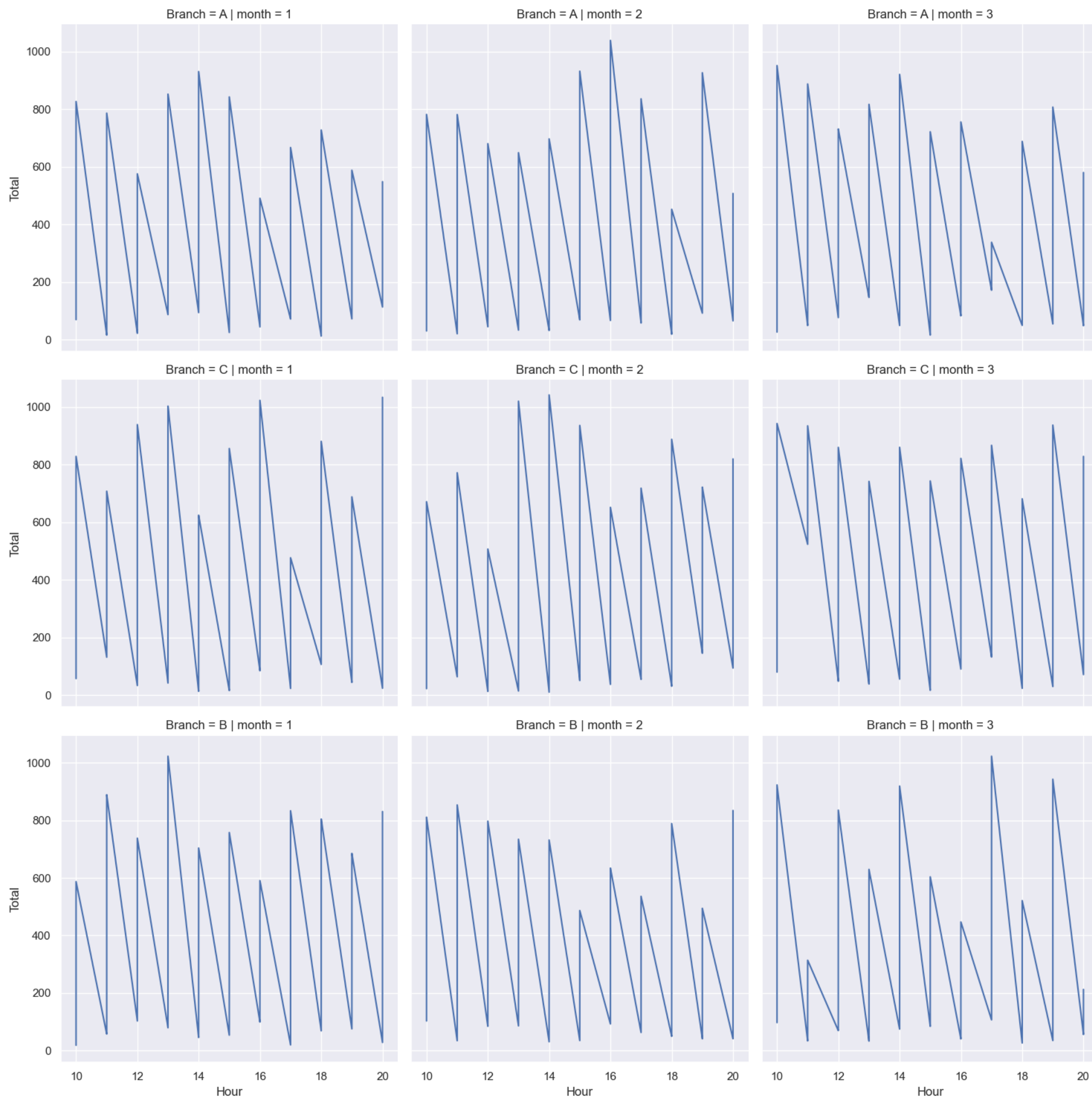


Below we can see how each branch's sales quantity looks like by the hour in a monthly fashion

[21]:
```
genderCount  = sns.relplot(x="Hour",  y = 'Quantity', col= 'month' , row=␣
↪'Branch', kind="line", hue="Gender", style="Gender", data =sales)
```

Below we can see each branch's sales by the hour in a monthly fashion

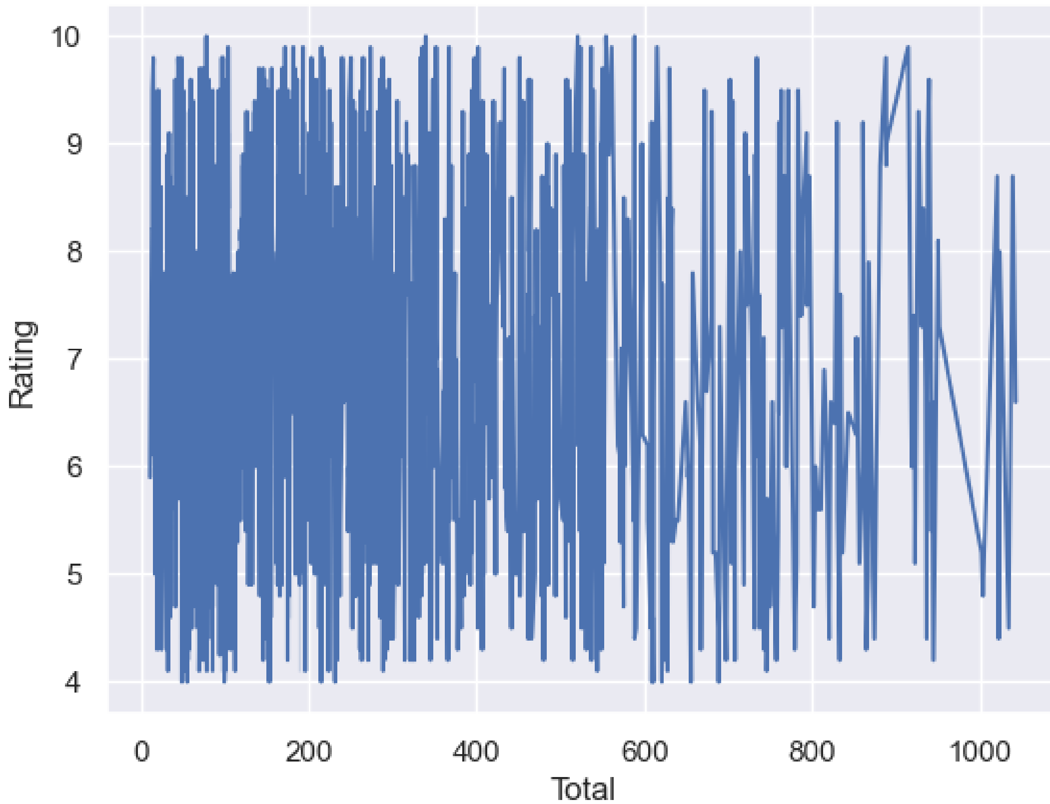
[22]:
```
genderCount  = sns.relplot(x="Hour",  y = 'Total', col= 'month' , row=␣
↪'Branch', estimator = None, kind="line", data =sales)
```



[23]:
```
sales['Rating'].unique()
```

[23]:
```
array([ 9.1,  9.6,  7.4,  8.4,  5.3,  4.1,  5.8,  8. ,  7.2,  5.9,  4.5,
        6.8,  7.1,  8.2,  5.7,  4.6,  6.9,  8.6,  4.4,  4.8,  5.1,  9.9,
        6. ,  8.5,  6.7,  7.7,  7.5,  7. ,  4.7,  7.6,  7.9,  6.3,  5.6,
        9.5,  8.1,  6.5,  6.1,  6.6,  5.4,  9.3, 10. ,  6.4,  4.3,  4. ,
        8.7,  9.4,  5.5,  8.3,  7.3,  4.9,  4.2,  9.2,  7.8,  5.2,  9. ,
        8.8,  6.2,  9.8,  9.7,  5. ,  8.9])
```

[24]:
```
ageDisSpend = sns.lineplot(x="Total", y = "Rating", data =sales)
```

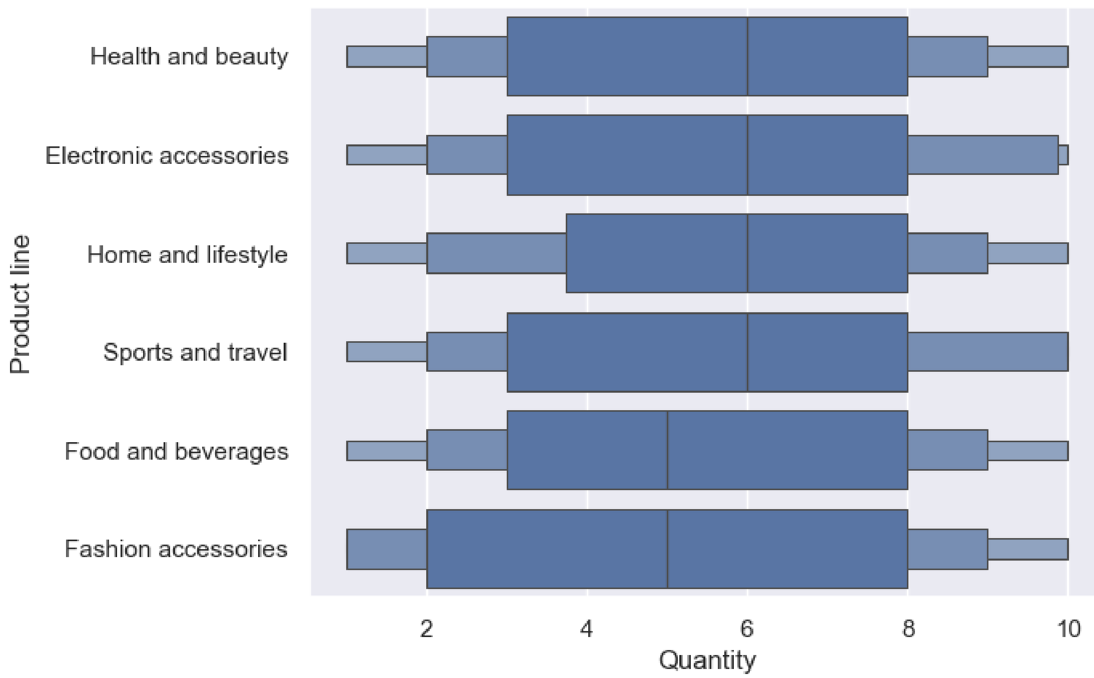


1.1 Product Analysis

Let's look at the various products' performance.

[25]:
```
sns.boxenplot(y = 'Product line', x = 'Quantity', data=sales )
```

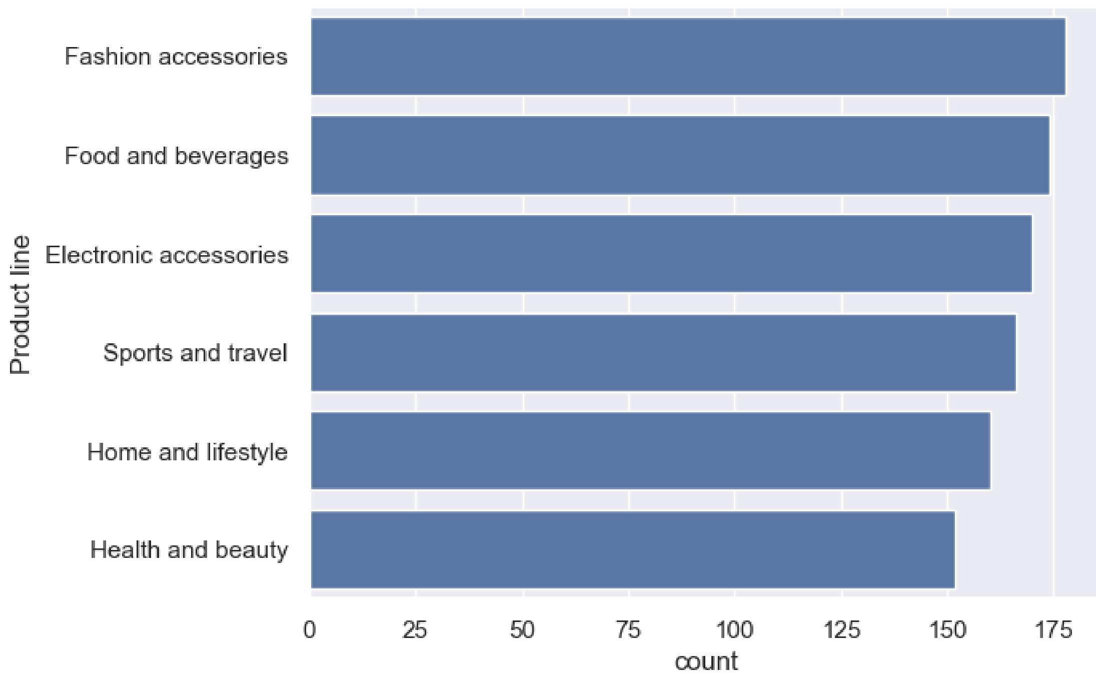
[25]:
```
<Axes: xlabel='Quantity', ylabel='Product line'>
```



From the above visual, Health and Beauty,Electronic accessories, Homem and lifestyle, Sports and travel have a better average quantity sales that food and beverages as well as Fashion accessories.

[26]:
```
sns.countplot(y = 'Product line', data=sales, order = sales['Product line'].
  ↪value_counts().index )
```

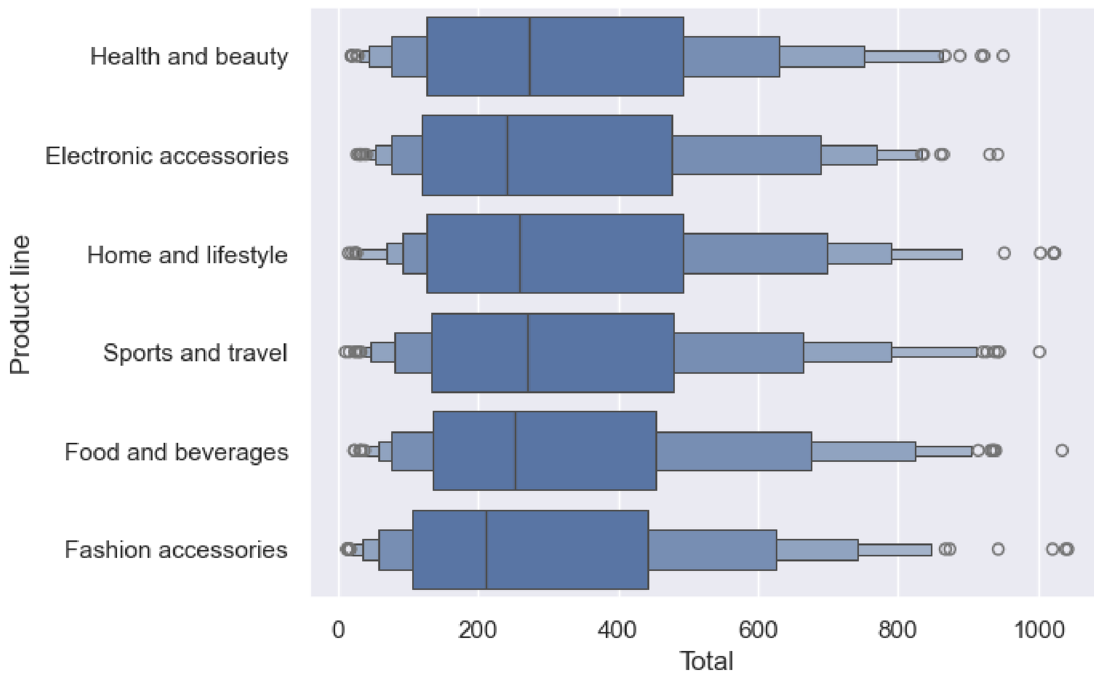
[26]: <Axes: xlabel='count', ylabel='Product line'>



From the above image shows the top product line item type sold in the given dataset. Fashion Accessories is the highest while Health and beauty is the lowest

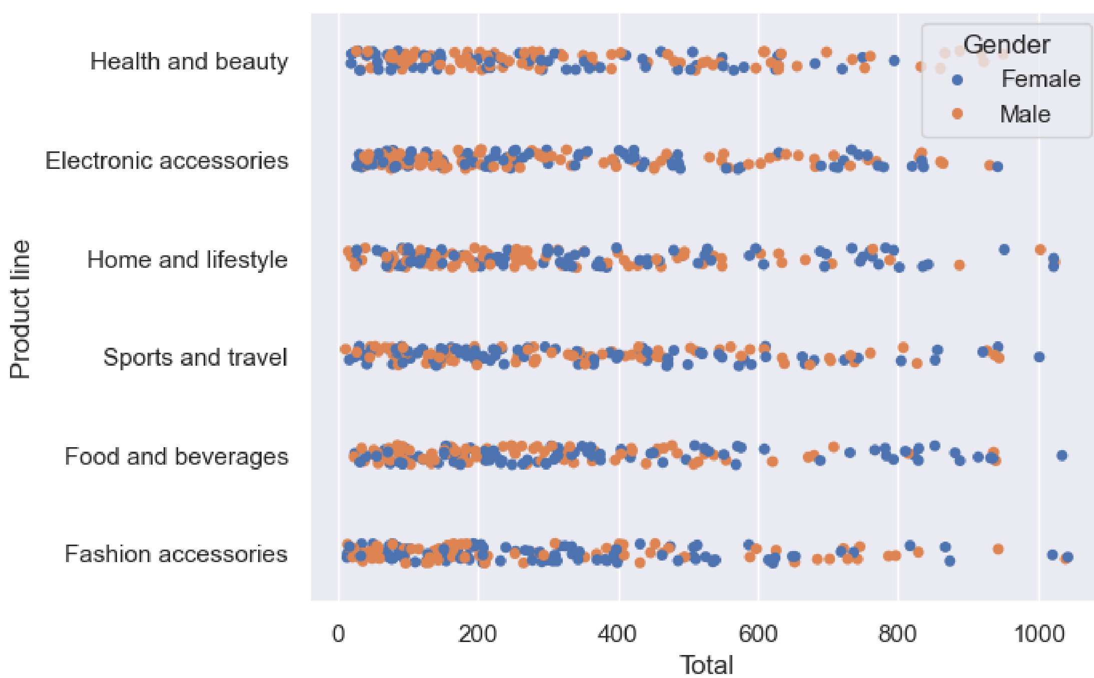
[27]:
```
sns.boxenplot(y = 'Product line', x = 'Total', data=sales )
```

[27]: <Axes: xlabel='Total', ylabel='Product line'>



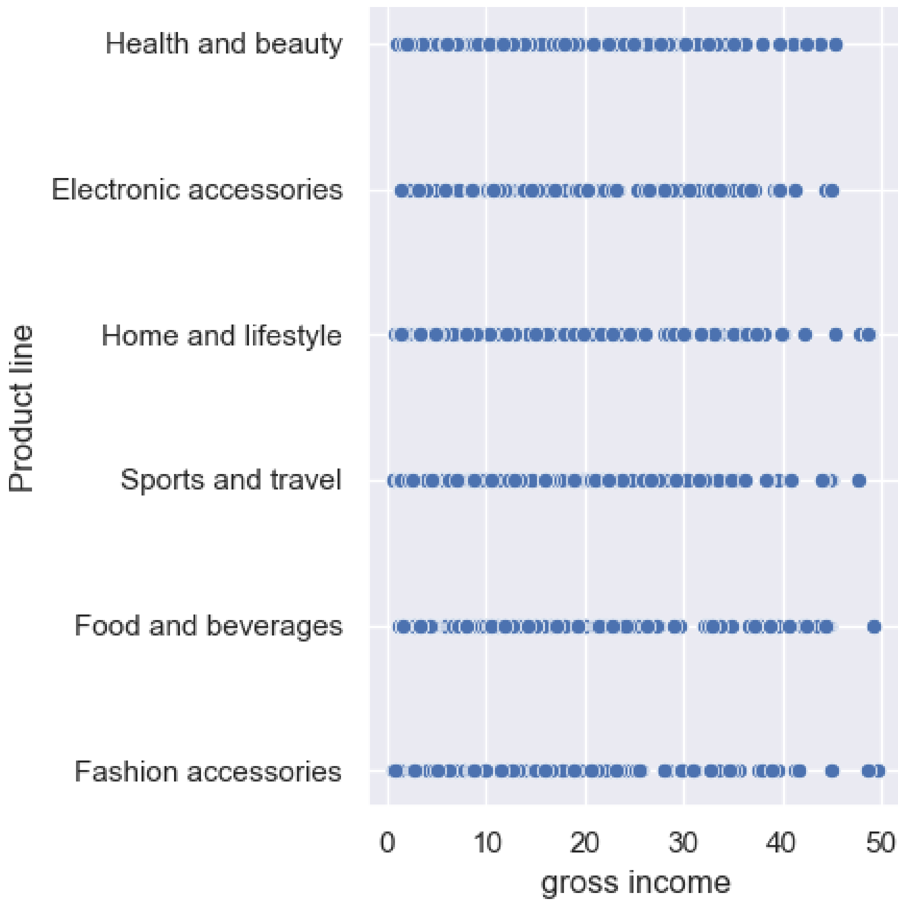
[28]:
```
sns.stripplot(y = 'Product line', x = 'Total', hue = 'Gender', data=sales )
```

[28]: <Axes: xlabel='Total', ylabel='Product line'>



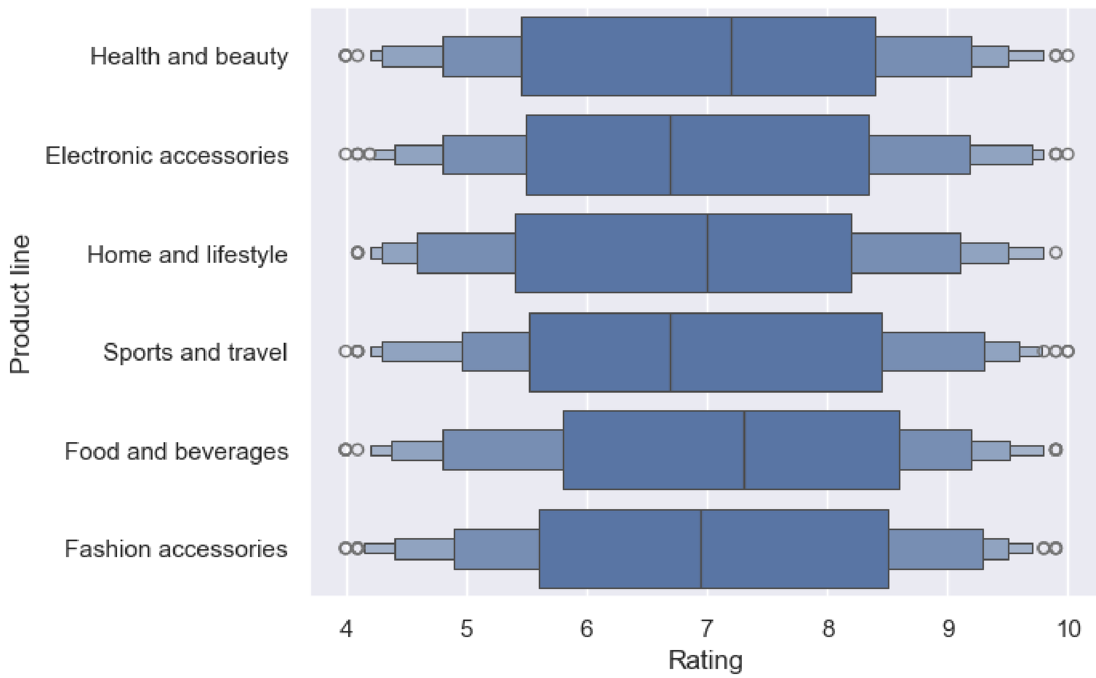
[29]:
```
sns.relplot(y = 'Product line', x = 'gross income', data=sales )
```

[29]: <seaborn.axisgrid.FacetGrid at 0x19a0fa7df70>



[30]:
```
sns.boxenplot(y = 'Product line', x = 'Rating', data=sales )
```

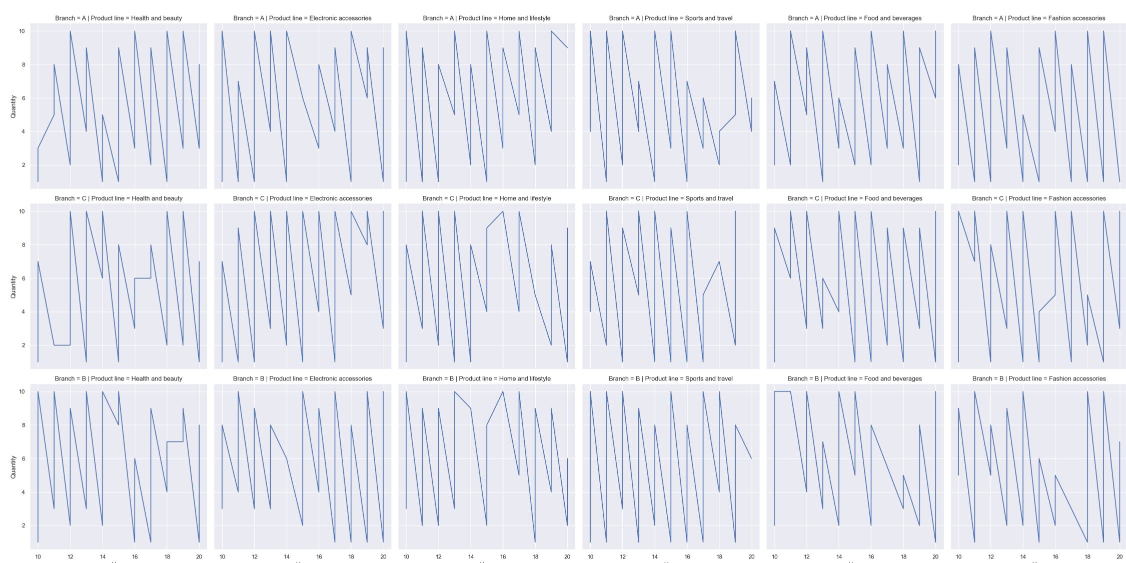
[30]: <Axes: xlabel='Rating', ylabel='Product line'>



Food and Beverages have the highest average rating while sports and travel the lowest

Let's see when customers buy certain products in the various branches.

[31]:
```
productCount  = sns.relplot(x="Hour",  y = 'Quantity', col= 'Product line' ,␣
 ↪row= 'Branch', estimator = None, kind="line", data =sales)
```



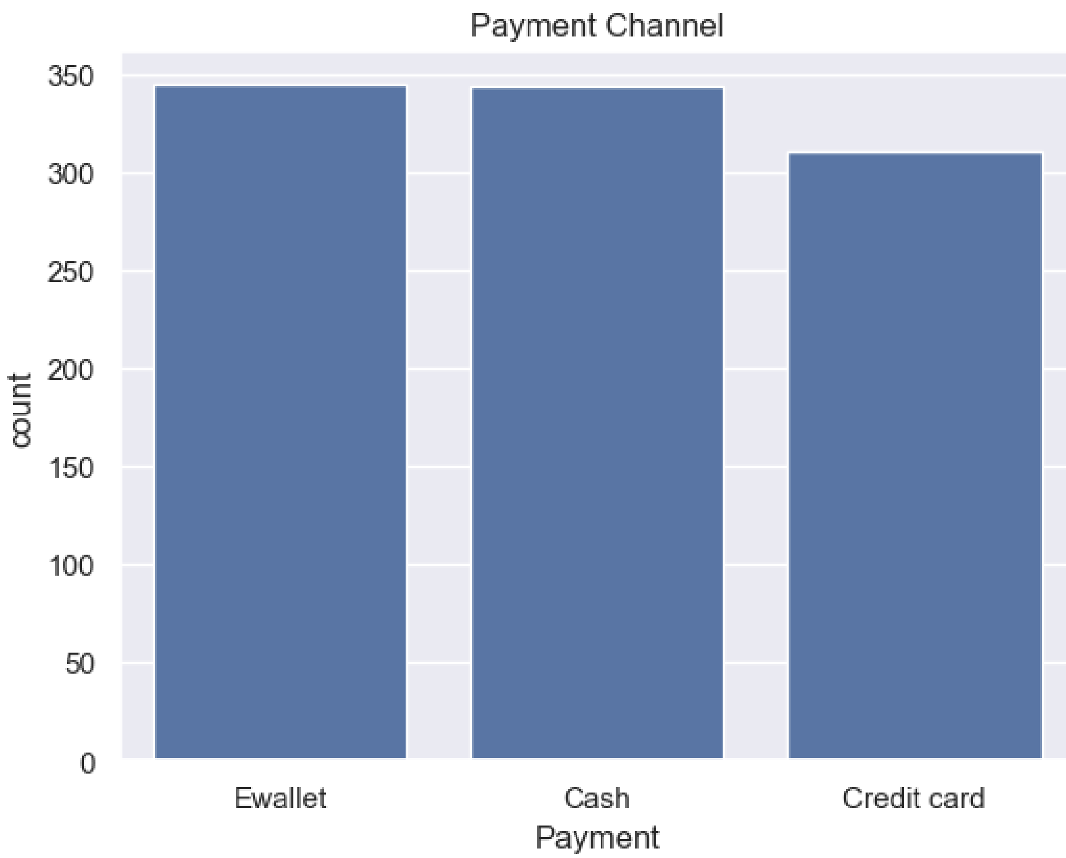
From the above plots, we can see that food and beverages sales usually high in all three branches at evening especially around 19:00

2 Payment Channel

Let see how customers make payment in this business

[32]:
```
sns.countplot(x="Payment", data =sales).set_title("Payment Channel")
```

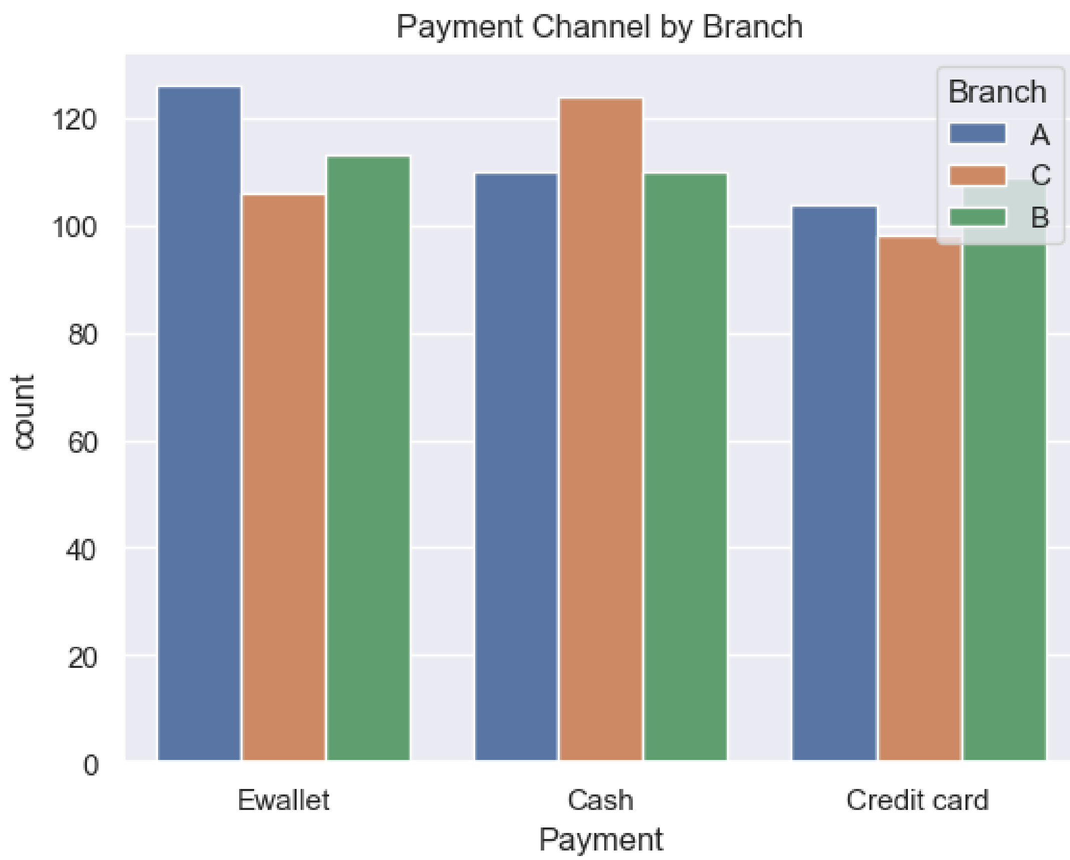
[32]: Text(0.5, 1.0, 'Payment Channel')



Most of the customers pay through the Ewallet and Cash Payment while under 40 percent of them pay with their credit card. We would also like to see this payment type distribution across all the branches

[33]:
```
sns.countplot(x="Payment", hue = "Branch", data =sales).set_title("Payment␣
↪Channel by Branch")
```

[33]: Text(0.5, 1.0, 'Payment Channel by Branch')



3 Customer Analysis

From inspection, there are two types of customers. Members and Normal. Let's see how many they are and where they are

[34]:
```
sales['Customer type'].nunique()
```

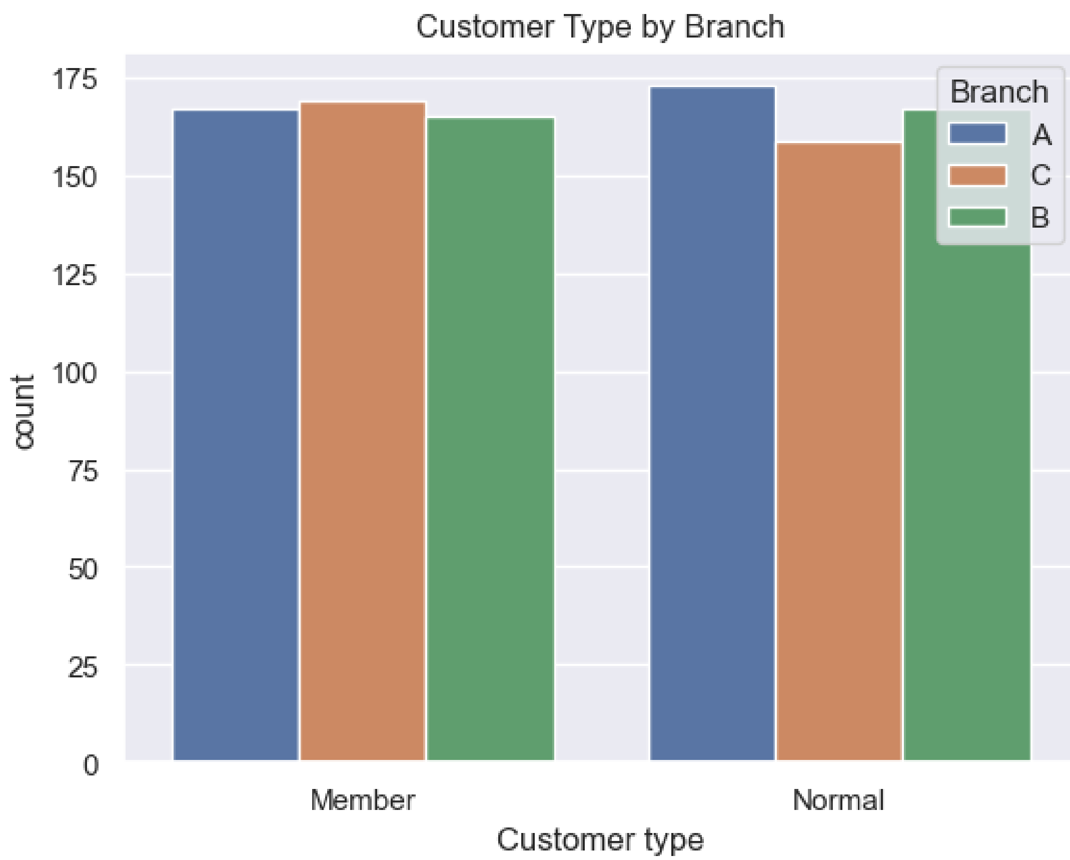
[34]:
```
2
```

[35]:
```
sns.countplot(x="Customer type", data =sales).set_title("Customer Type")
```

[35]:
```
Text(0.5, 1.0, 'Customer Type')
```

[36]:
```
sns.countplot(x="Customer type", hue = "Branch", data =sales).
↪set_title("Customer Type by Branch")
```

[36]: Text(0.5, 1.0, 'Customer Type by Branch')



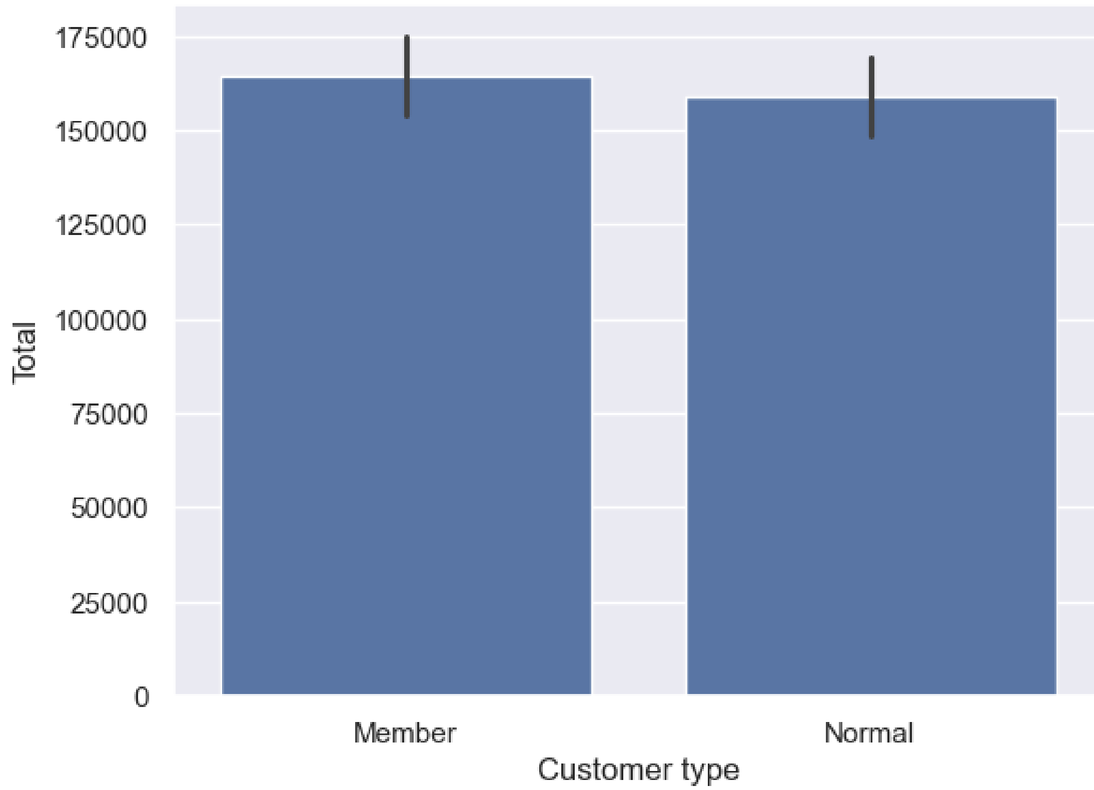
3.1 Does customer type influences the sales

[37]:
```
sales.groupby(['Customer type']).agg({'Total': 'sum'})
```

[37]:
```
                    Total
Customer type
Member         164223.444
Normal         158743.305
```

[38]:
```
sns.barplot(x="Customer type", y="Total", estimator = sum, data=sales)
```

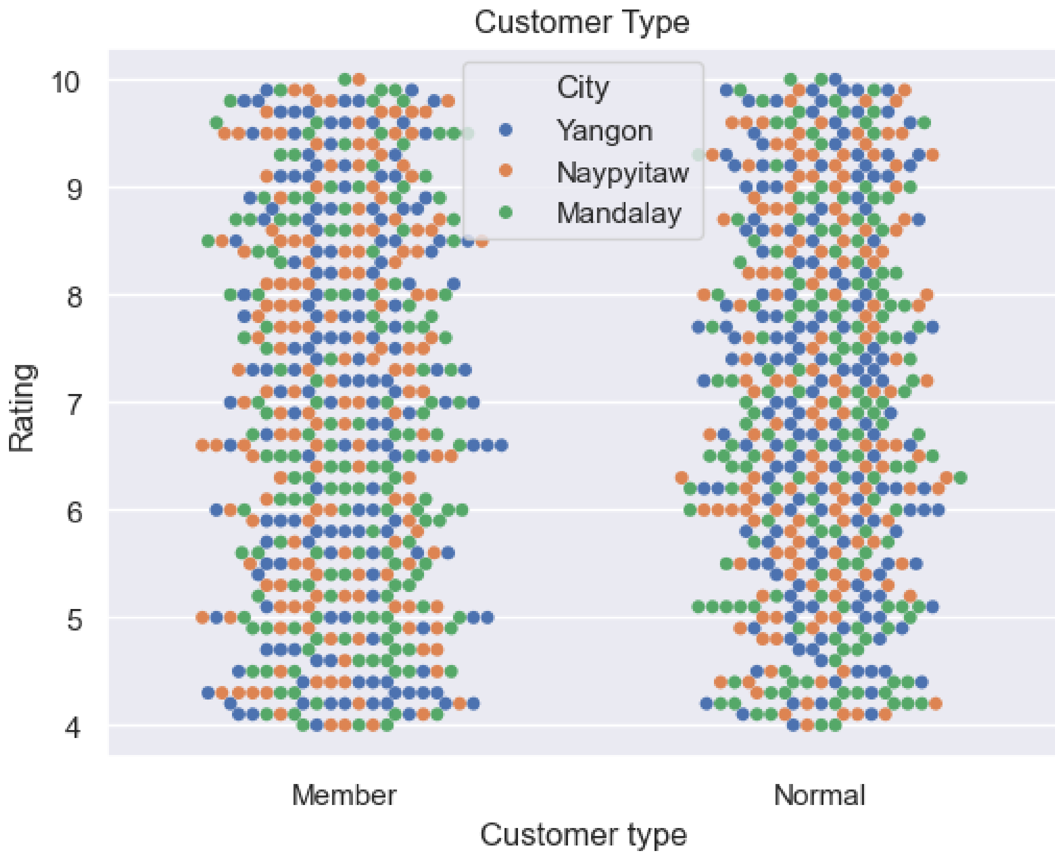
[38]: <Axes: xlabel='Customer type', ylabel='Total'>



Do the customer type influence customer rating? Let's find out

[39]:
```
sns.swarmplot(x="Customer type",  y = "Rating",  hue = "City", data =sales).
↪set_title("Customer Type")
```

[39]: Text(0.5, 1.0, 'Customer Type')

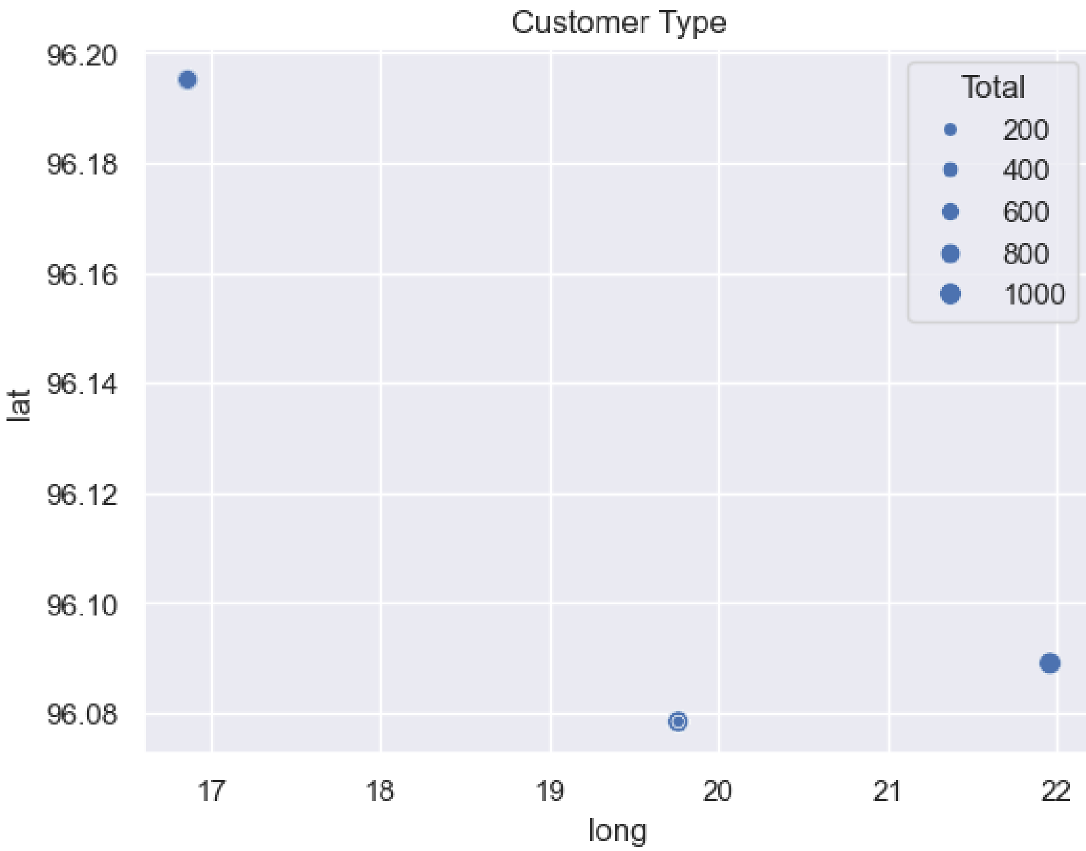


With the use of google search, I was able to get the longitude and latitude of each cities. We can

[40]:
```
long = {"Yangon": 16.8661, "Naypyitaw": 19.7633, "Mandalay": 21.9588 }
lat = {"Yangon": 96.1951, "Naypyitaw": 96.0785, "Mandalay": 96.0891 }
for set in sales:
    sales['long'] = sales['City'].map(long)
    sales['lat'] = sales['City'].map(lat)
```

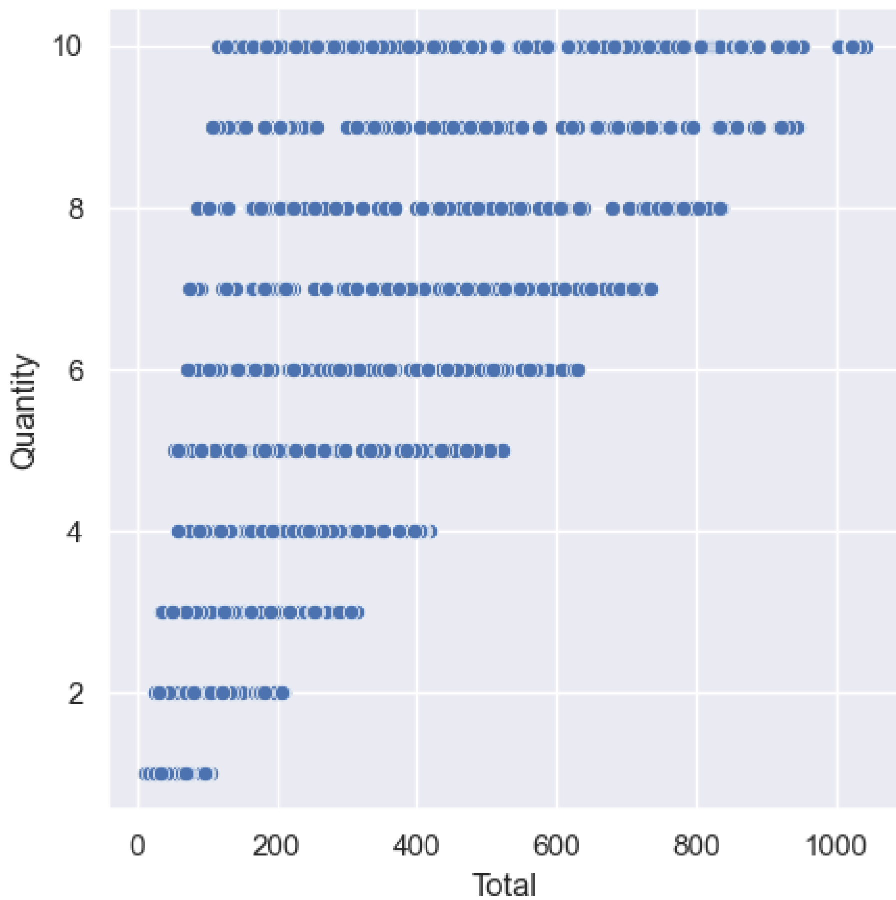
[41]:
```
sns.scatterplot(x="long",  y = "lat",size = "Total", data =sales, legend =␣
↪"brief").set_title("Customer Type")
```

[41]:
```
Text(0.5, 1.0, 'Customer Type')
```



[42]:
```
sns.relplot(x="Total",  y = "Quantity", data =sales)
```

[42]: <seaborn.axisgrid.FacetGrid at 0x19a0eaef9e0>


Quantity
Total
0
200
400
600
800
1000
2
4
6
8
10

[ ]:

[ ]: