Supermarket Analysis

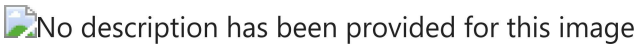


Table of Contents

1. Problem Statement

1.1. Introduction

What is Supermarket Sales?

The growth of supermarkets in most populated cities are increasing and market competitions are also high. The dataset is one of the historical sales of supermarket

company which has recorded in 3 different branches for 3 months data. Predictive data analytics methods are easy to apply with this dataset.

1.2. Obejctives

In this projects below questions will be answered:

- Customer Analysis
- Product Line Analysis
- Sales and Revenue Analysis
- Branch and City Analysis
- Time Series Analysis
- Predictive Modeling and gross income?ty?redicting churn?

Customer Analysis:-

1. What is the distribution of customer types (e.g. regular, loyalty, etc.) and how do they impact sales?
2. What is the gender-wise distribution of customers and how do they differ in their purchasing behavior?
3. What is the average rating given by customers and how does it vary by branch and city?

Product Line Analysis:-

1. Which product lines are the most profitable and which ones are the least profitable?
2. What is the average unit price and quantity sold for each product line?
3. How do the product lines vary in terms of COGS, gross margin percentage, and gross income?

Sales and Revenue Analysis:-

1. What is the total sales revenue and gross income by branch and city?
2. What is the average transaction value and how does it vary by time of day and day of the week?
3. What is the impact of payment method on sales revenue and gross income?

Branch and City Analysis:-

1. Which branches and cities have the highest sales revenue and gross income?
2.

How do the branches and cities vary in terms of customer demographics and purchasing behavior 3. What is the impact of branch and city on product line sales and profitability?

Time Series Analysis:-

1. What are the seasonal and trend patterns in sales revenue and gross income?
2. How do the sales revenue and gross income vary by day of the week and time of day?
3.

What is the impact of holidays and special events on sales revenue and gross income?ome?

Predictive Modeling:-

1. Can we predict the likelihood of a customer making a purchase based on their demographics and purchase history?
2.

Can we predict the sales revenue and gross income for a given branch and city based on historical data 3. Can we identify the most important factors that influence sales revenue and gross income?

1.3. Dataset Features

- `Invoice ID`: A unique ID that is computer-generated sales slip invoice identification number
- `Branch:` Branch of supercenter (3 branches are available identified by A, B, and C).
- `City:` Location of supercenters.
- `Customer Type:` Type of customers, recorded by: Members and Normal
- `Gender:` Type of custom
- `Invoice ID:` A unique ID that is computer-generated sales slip invoice identification number.
- `Branch:` Branch of supercenter (3 branches are available identified by A, B, and C).
- `City:` Location of supercenters.
- `Customer Type:` Type of customers, recorded
- `Gender:` Gender type of customer.
- `Product Line:` General item categorization groups
- `Unit Price:` Price of each product in $.
- `Quantity:` Number of products purchased by customer.
- `Tax:` 5% tax fee for customer buying.
- `Total:` Total price including tax.
- `Date:` Date of purchase (Record available from January 2019 to March 2019).
- `Time:` Purchase time (10am to 9pm).
- `Payment:` Payment used by customer for purchase
- `COGS:` Cost of goods sold.
- `Gross Margin Percentage:` Gross margin percentage.

- `Gross Income:` Gross income.
- `Rating:` Customer stratification rating on their overall shopping experience (On a scale of 1 to 10).

2. Import Libraries and Data

In [18]:
```
!pip install mlens
```

```
Requirement already satisfied: mlens in c:\users\anmol singh rajput\anaconda3\lib\si
te-packages (0.2.3)
Requirement already satisfied: numpy>=1.11 in c:\users\anmol singh rajput\anaconda3
\lib\site-packages (from mlens) (1.26.4)
Requirement already satisfied: scipy>=0.17 in c:\users\anmol singh rajput\anaconda3
\lib\site-packages (from mlens) (1.13.1)
```

In [19]:
```
# handle table-like data and matrices
import pandas as pd
import numpy as np

# visualisation
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.figure_factory as ff
from plotly.offline import download_plotlyjs, init_notebook_mode, iplot
init_notebook_mode(connected=True)

# preprocessing
# balance data
# models
# evaluations

# ignore warnings
import warnings
warnings.filterwarnings('ignore')

# to display the total number columns present in the dataset
pd.set_option('display.max_columns', None)
```

In [25]:
```
data = pd.read_csv('DataSet/SuperMerket_Sales.csv')
```

3. Handling Missing Values

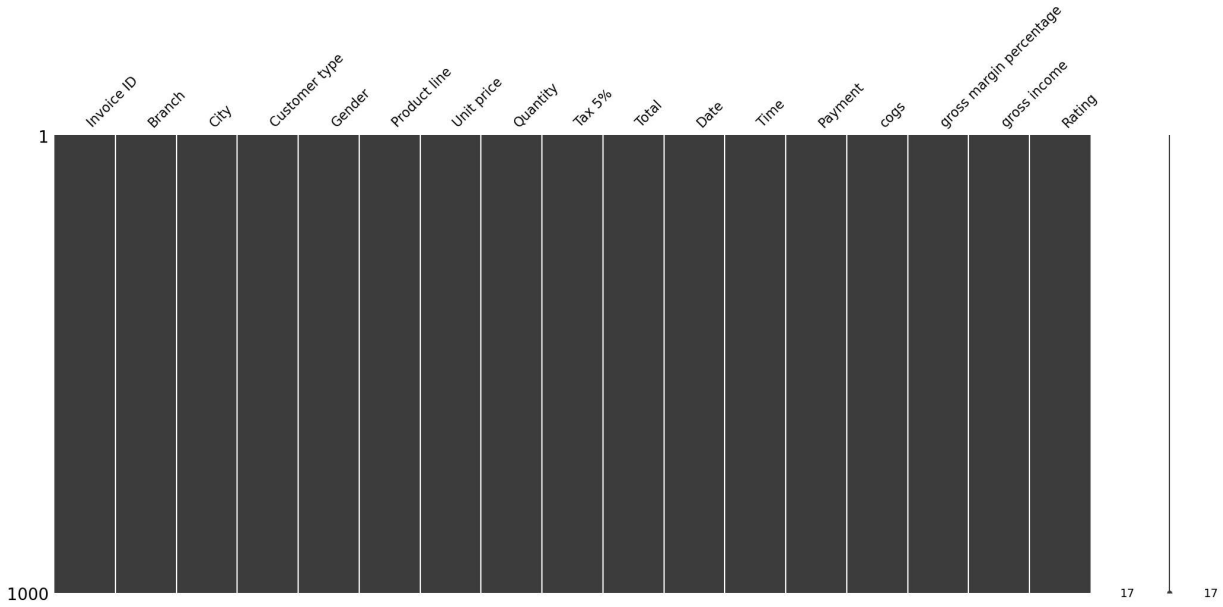
let's find if we have missing values in the dataset.

In [29]:
```
data = data.replace(r'^\s*$', np.nan, regex=True)
```

In [31]:
```
data.isnull().sum()
```

Out[31]:
```
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
```

In [33]:
```
msno.matrix(data);
```



If we examine the data carefully, we can actually estimate that their is no missing data.

let's find if we have duplicate rows.

In [37]:
```
data.duplicated().sum()
```

Out[37]:
```
0
```

4. Data Analysis and Visualization

In [40]:
```
data.head(3)
```

Out[40]:

	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	54
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	8
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	34

In [42]:
```
data.shape
```

Out[42]:
```
(1000, 17)
```

There are 1000 cutomers and 17 features in the dataset.

In [45]:
```
data.columns
```

Out[45]:
```
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')
```

In [47]:
```
data.rename(columns={"Customer type": "Customer_type", "Product line": "Product_lin
                     "Unit price": "Unit_price", "Tax 5%": "Tax_5", "gross margin per
                     "gross income": "gross_income"}, inplace="True")
```

In [49]:
```
data.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

In [52]:
```
data['date'] = pd.to_datetime(data['Date'])
```

In [54]:
```
data['date'].dtype
```

Out[54]:
```
dtype('<M8[ns]')
```

In [56]:
```
type(data['date'])
```

Out[56]:
```
pandas.core.series.Series
```

In [58]:
```
data['date'] = pd.to_datetime(data['date'])
```

In [60]:
```
data['day'] = (data['date']).dt.day
data['month'] = (data['date']).dt.month
data['year'] = (data['date']).dt.year
```

In [62]:
```
data['Time'] = pd.to_datetime(data['Time'])
```

In [64]:
```
data['Hour'] = (data['Time']).dt.hour    #type(sales['Time'])
```

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

In [67]:
```
data['Hour'].nunique()  #gives us the number of unique hours
```

Out[67]:
```
11
```

In [69]:
```
data['Hour'].unique()
```

Out[69]:
```
array([13, 10, 20, 18, 14, 11, 17, 16, 19, 15, 12])
```

In [71]:
```
data.describe()
```

Out[71]:

	Unit_price	Quantity	Tax_5	Total	Time	cogs	gr
count	1000.000000	1000.000000	1000.000000	1000.000000	1000	1000.00000	
mean	55.672130	5.510000	15.379369	322.966749	2024-08-25 15:24:41.880000	307.58738	
min	10.080000	1.000000	0.508500	10.678500	2024-08-25 10:00:00	10.17000	
25%	32.875000	3.000000	5.924875	124.422375	2024-08-25 12:43:00	118.49750	
50%	55.230000	5.000000	12.088000	253.848000	2024-08-25 15:19:00	241.76000	
75%	77.935000	8.000000	22.445250	471.350250	2024-08-25 18:15:00	448.90500	
max	99.960000	10.000000	49.650000	1042.650000	2024-08-25 20:59:00	993.00000	
std	26.494628	2.923431	11.708825	245.885335	NaN	234.17651	

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

In [74]:
```
categorical_columns = [cname for cname in data.columns if data[cname].dtype == "obj
```

In [76]:
```
categorical_columns
```

Out[76]:
```
['Invoice ID',
 'Branch',
 'City',
 'Customer_type',
 'Gender',
 'Product_line',
 'Date',
 'Payment']
```

In [78]:
```
print("# unique values in Branch: {0}".format(len(data['Branch'].unique().tolist())
print("# unique values in City: {0}".format(len(data['City'].unique().tolist())))
print("# unique values in Customer Type: {0}".format(len(data['Customer_type'].uniq
print("# unique values in Gender: {0}".format(len(data['Gender'].unique().tolist())
print("# unique values in Product Line: {0}".format(len(data['Product_line'].unique
print("# unique values in Payment: {0}".format(len(data['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
```

In [80]:
```
data.describe(include=object).T
```

Out[80]:

	count	unique	top	freq
Invoice ID	1000	1000	750-67-8428	1
Branch	1000	3	A	340
City	1000	3	Yangon	340
Customer_type	1000	2	Member	501
Gender	1000	2	Female	501
Product_line	1000	6	Fashion accessories	178
Date	1000	89	2/7/2019	20
Payment	1000	3	Ewallet	345

In [82]:
```
import plotly.express as px

fig = px.pie(data, names="Gender", color_discrete_sequence=px.colors.qualitative.Se
fig.update_layout(legend=dict(title="Gender"), title_text="Gender Distribution", ti

fig.show()
```

EDA on Branch:-

In [84]:
```
# EDA on Branch column
branch_ratings = data.groupby('Branch')['Rating'].describe()

print("Branch Ratings Summary:")
print(branch_ratings)
```

```
Branch Ratings Summary:
        count      mean       std  min  25%  50%  75%   max
Branch
A       340.0  7.027059  1.731345  4.0  5.6  7.1  8.5  10.0
B       332.0  6.818072  1.713719  4.0  5.3  6.7  8.2  10.0
C       328.0  7.072866  1.704526  4.0  5.6  7.1  8.5  10.0
```

In [90]:
```
# Histograms for each branch
fig = px.histogram(data, x="Rating", color="Branch", barmode="group")
fig.update_layout(title={"text": "Rating Distribution by Branch",
                         "x": 0.5,
                         "xanchor": "center"},
                  xaxis_title="Rating",
                  yaxis_title="Frequency")
fig.show()
```

In [113…

```
fig = px.box(data, x="Branch", y="Rating")
fig.update_layout(
    title="Ratings by Branch",
    title_x=0.5,  # center the title
)
fig.show()
```

Branch B has the lowest rating among all the branches

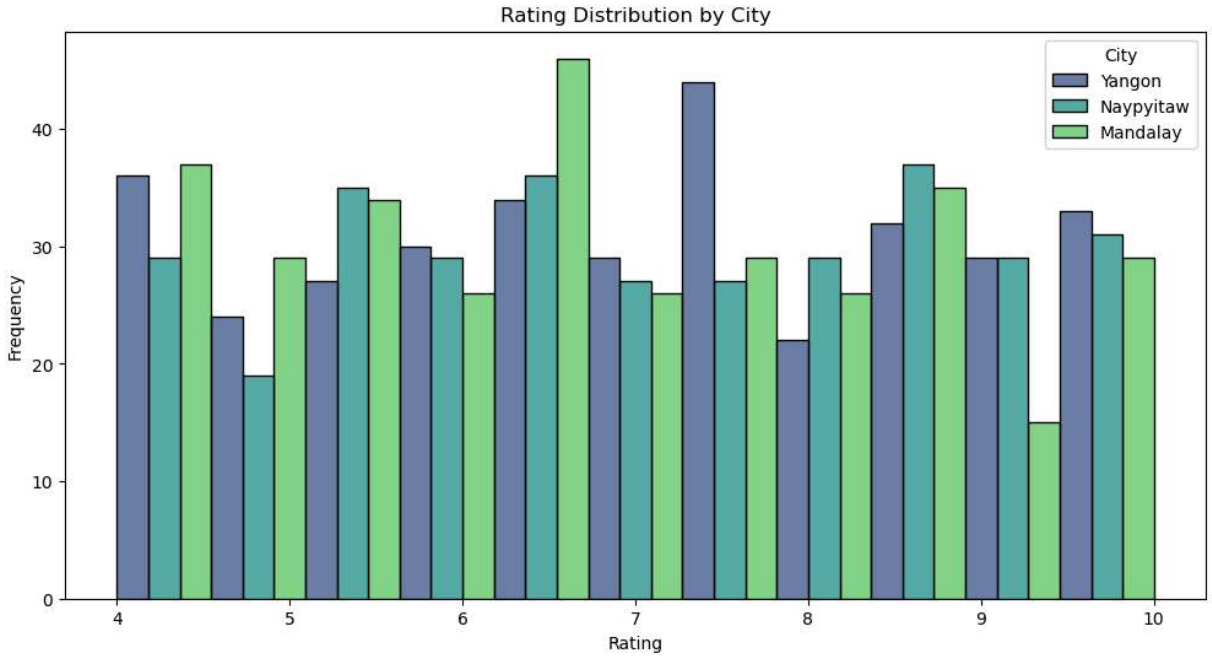
EDA on City:-

In [96]:
```
# EDA on City column
city_ratings = data.groupby('City')['Rating'].describe()

print("City Ratings Summary:")
print(city_ratings)
```

```
City Ratings Summary:
           count      mean       std  min  25%  50%  75%   max
City
Mandalay   332.0  6.818072  1.713719  4.0  5.3  6.7  8.2  10.0
Naypyitaw  328.0  7.072866  1.704526  4.0  5.6  7.1  8.5  10.0
Yangon     340.0  7.027059  1.731345  4.0  5.6  7.1  8.5  10.0
```

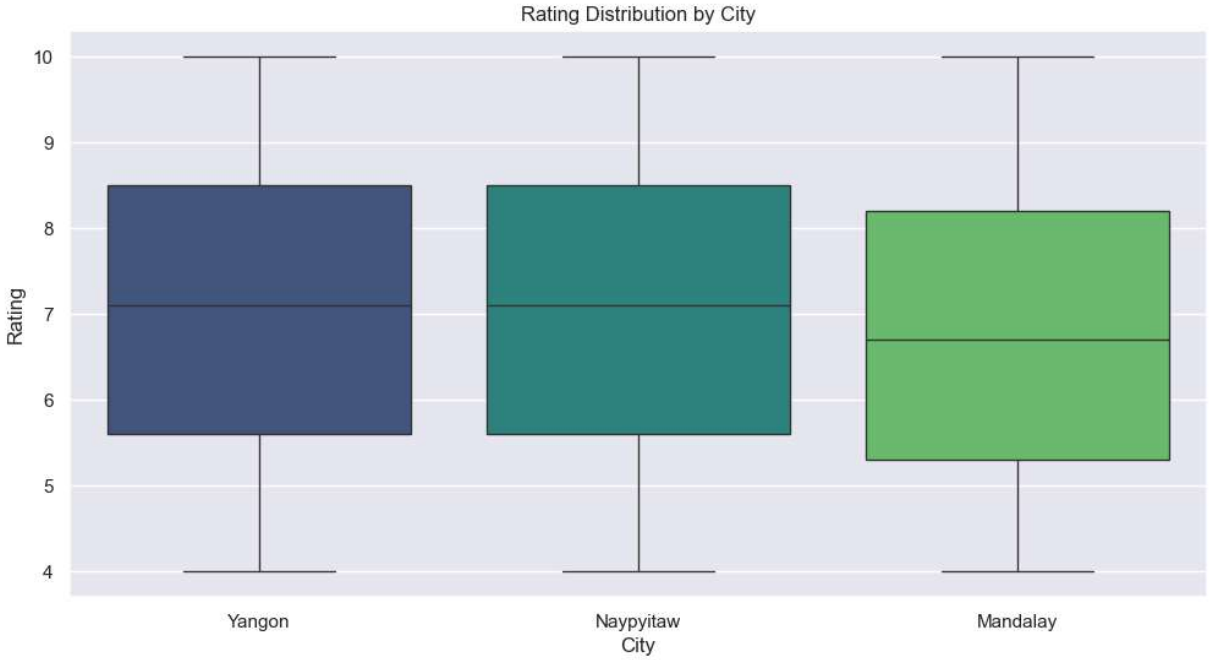
In [100…
```
# Histograms for each city
plt.figure(figsize=(12, 6))

sns.histplot(data=data, x='Rating', hue='City', multiple='dodge', palette='viridis'
plt.title('Rating Distribution by City')
plt.xlabel('Rating')
```

```
plt.ylabel('Frequency')
plt.show()
```



In [137…

```
# Box plots for each city
plt.figure(figsize=(12, 6))

sns.boxplot(data=data, x='City', y='Rating', palette='viridis')
plt.title('Rating Distribution by City')
plt.xlabel('City')
plt.ylabel('Rating')
plt.show()
```

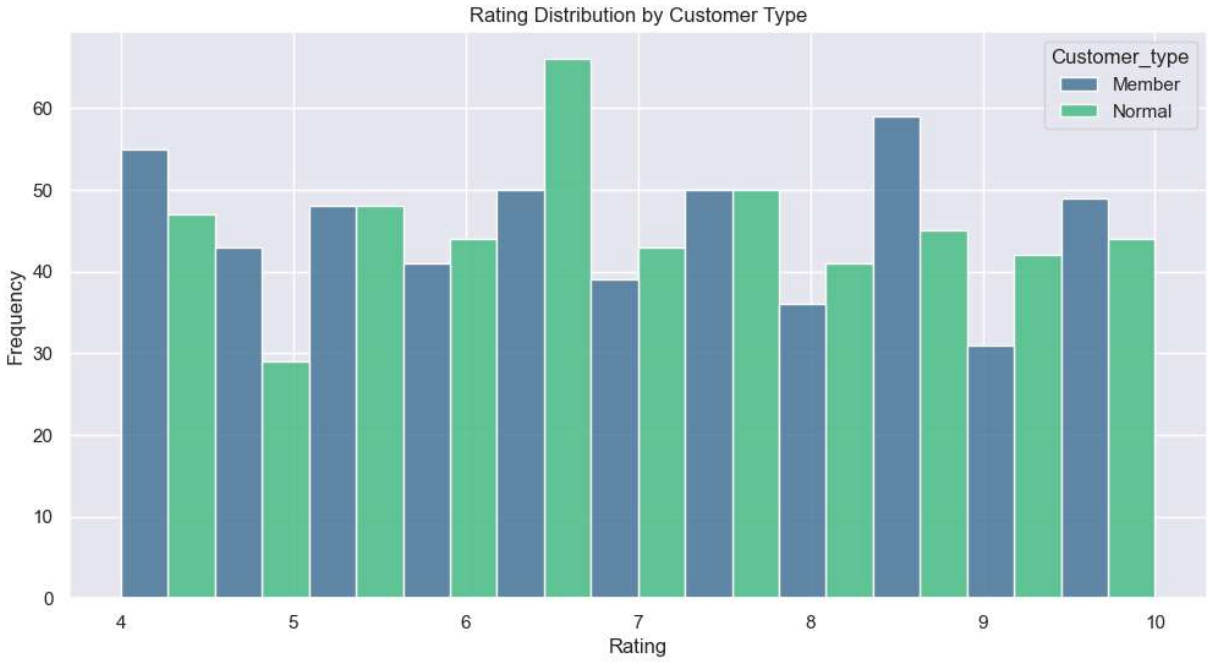


EDA on Customer:-

In [149…
```
# EDA on Customer Type column
customer_type_ratings = data.groupby('Customer_type')['Rating'].describe()

print("Customer Type Ratings Summary:")
print(customer_type_ratings)
```

```
Customer Type Ratings Summary:
               count      mean       std  min  25%  50%  75%   max
Customer_type
Member         501.0  6.940319  1.749380  4.0  5.4  7.0  8.5  10.0
Normal         499.0  7.005210  1.688222  4.0  5.7  7.0  8.4  10.0
```

In [151…
```
# Histograms for each customer type
plt.figure(figsize=(12, 6))

sns.histplot(data=data, x='Rating', hue='Customer_type', multiple='dodge', palette=
plt.title('Rating Distribution by Customer Type')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()
```



In [153…
```
# Box plots for each customer type
plt.figure(figsize=(12, 6))

sns.boxplot(data=data, x='Customer_type', y='Rating', palette='viridis')
plt.title('Rating Distribution by Customer Type')
plt.xlabel('Customer Type')
plt.ylabel('Rating')
plt.show()
```

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

In [104…

```
fig = px.line(data, x="Hour", y="Quantity")
fig.update_layout(title={"text": "Product Sales per Hour",
                         "x": 0.5,
                         "xanchor": "center"},
                  xaxis_title="Hour",
                  yaxis_title="Quantity")
fig.show()
```

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

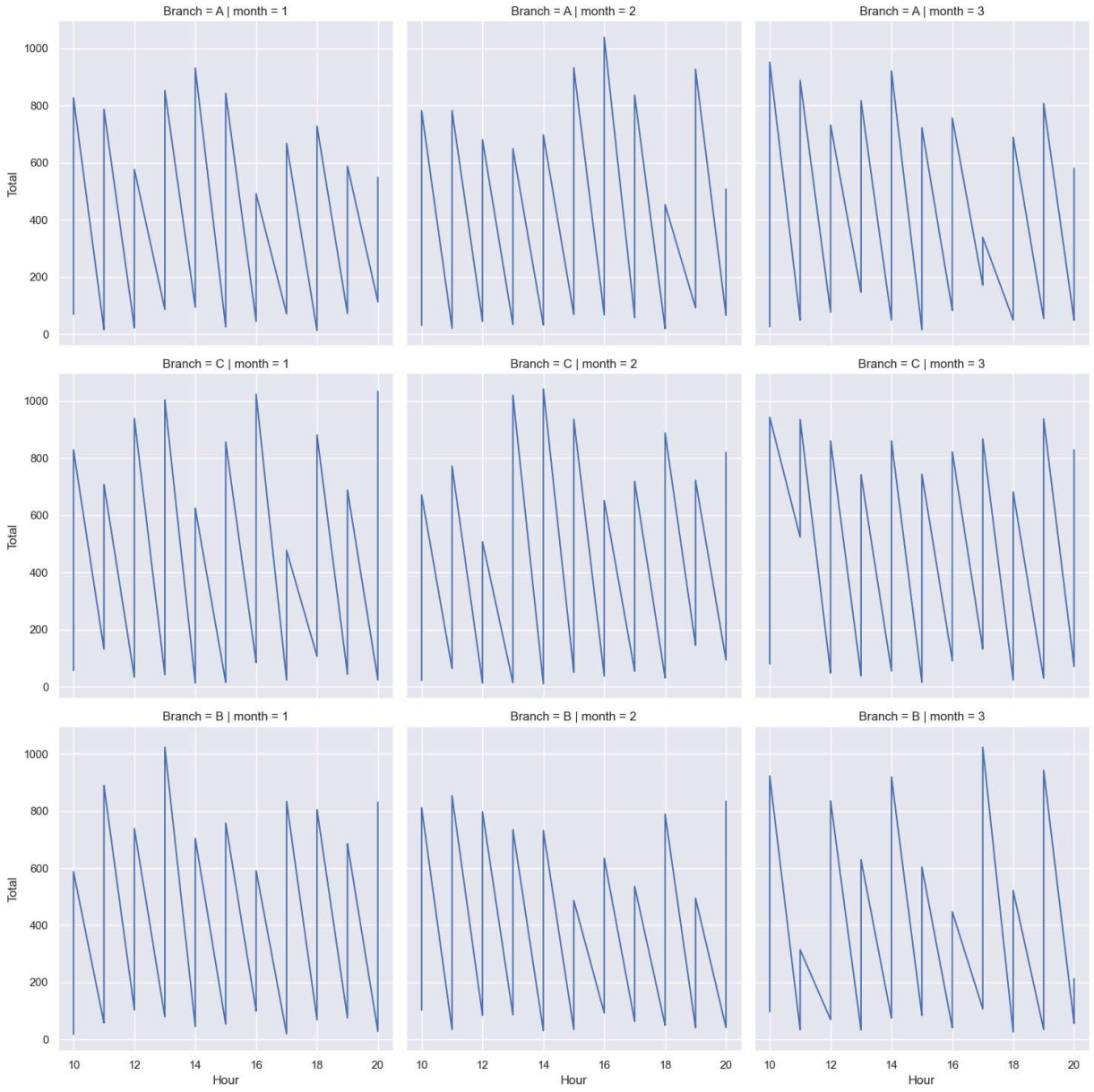
In [106…

```
fig = px.line(data, x="Hour", y="Quantity", color="Gender", facet_col="month", face
fig.update_layout(title={"text": "Product Sales per Hour by Month and Branch",
                         "x": 0.5,
                         "xanchor": "center"},
                  xaxis_title="Hour",
                  yaxis_title="Quantity")
fig.show()
```

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

In [168…

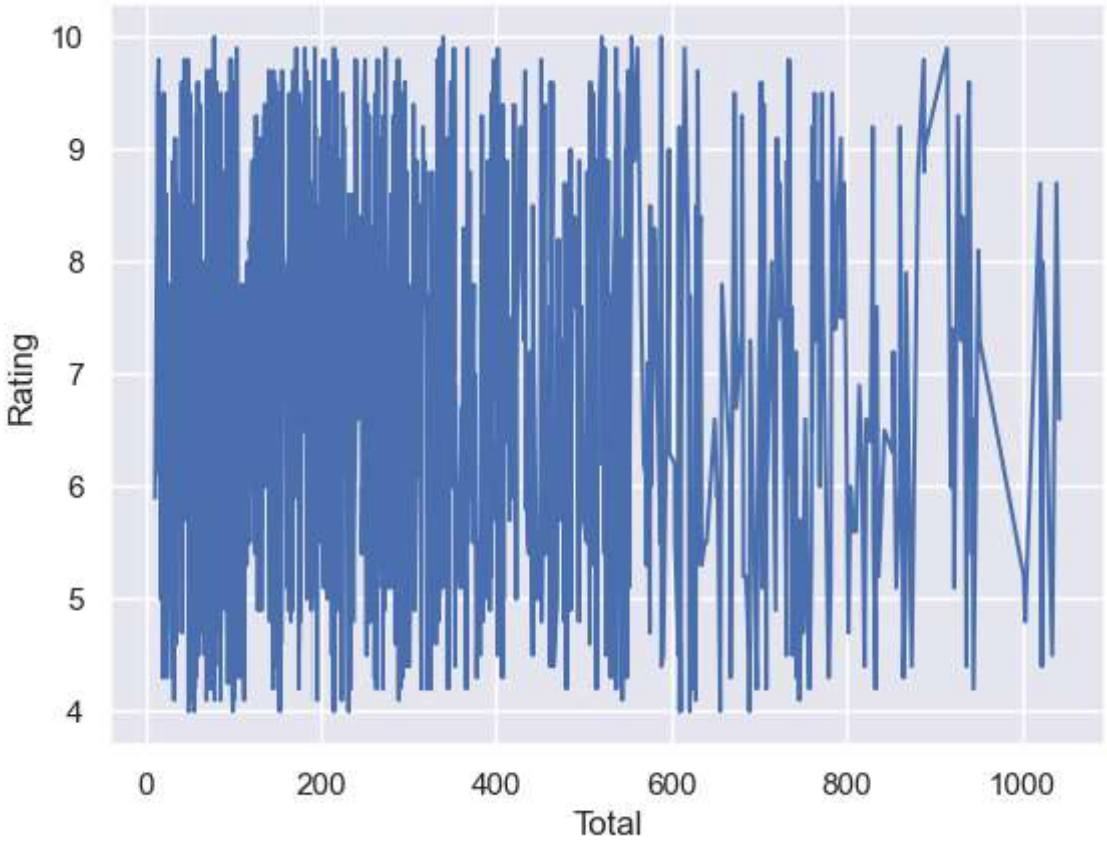
```
genderCount  = sns.relplot(x="Hour",  y = 'Total', col= 'month' , row= 'Branch', es
```



In [170…
```
data['Rating'].unique()
```

Out[170…
```
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])
```

In [172…
```
ageDisSpend = sns.lineplot(x="Total", y = "Rating", data =data)
```

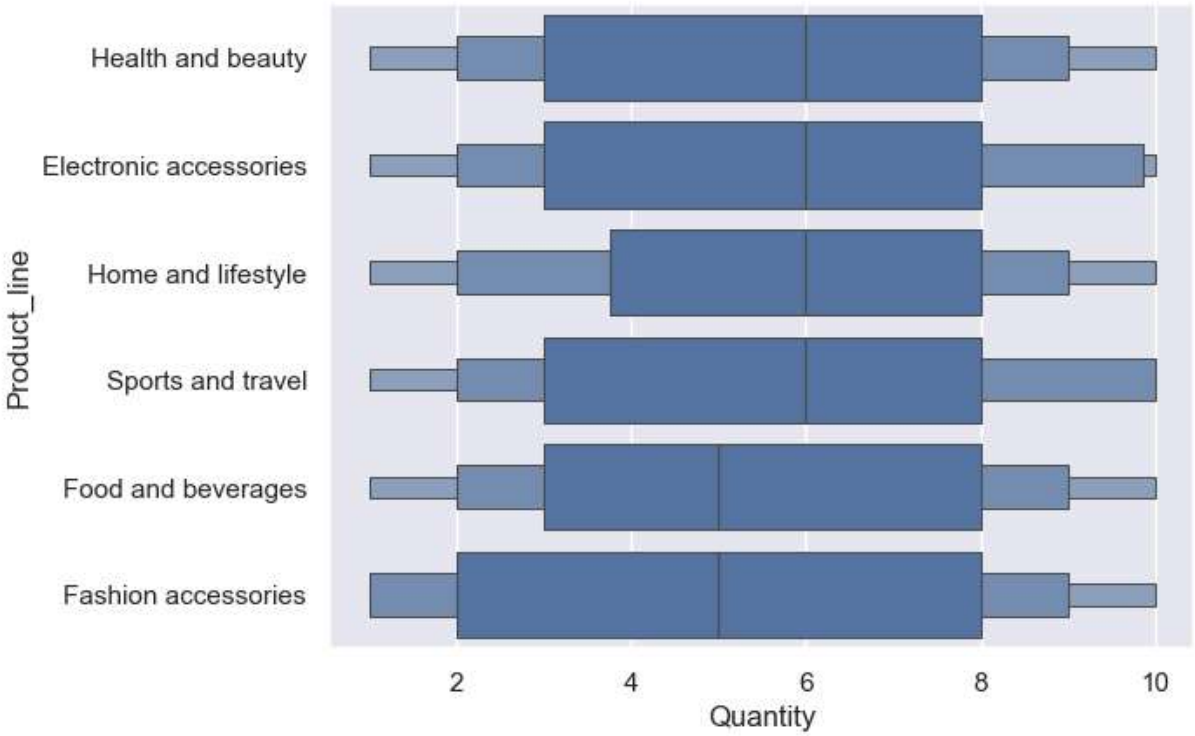


Product Analysis

Let's look at the various products' performance.

In [178…
```
sns.boxenplot(y = 'Product_line', x = 'Quantity', data=data )
```

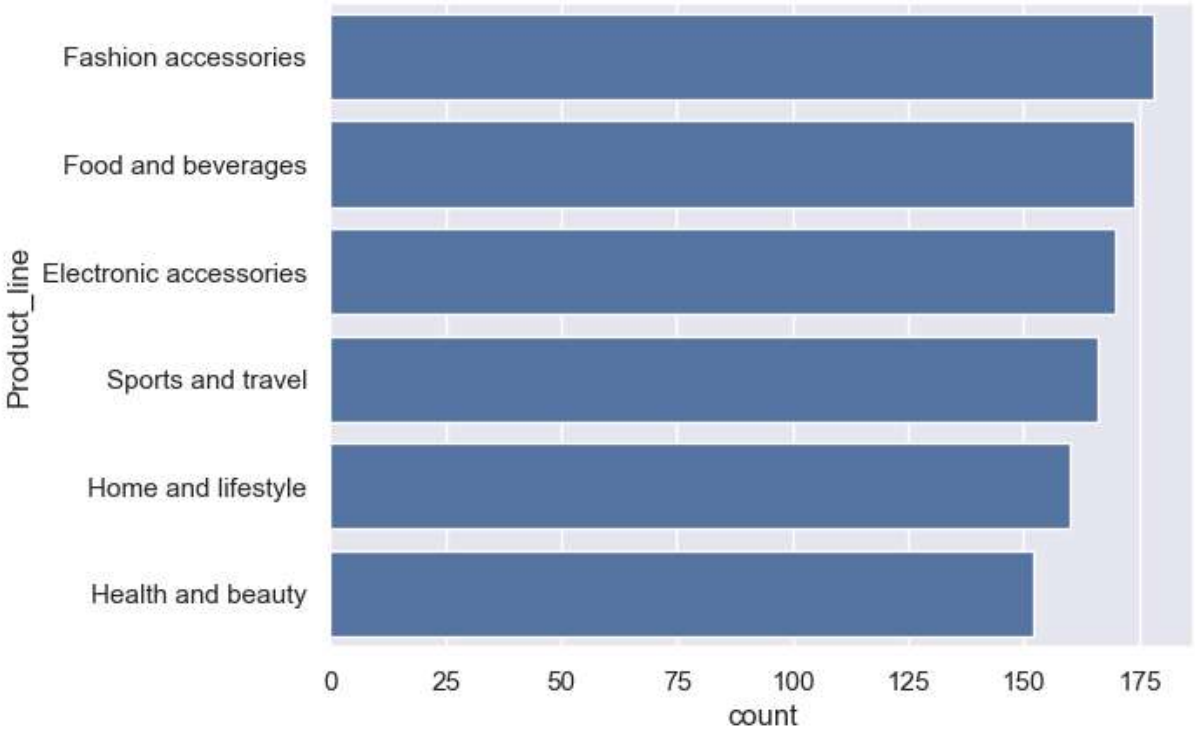
Out[178…
```
<Axes: xlabel='Quantity', ylabel='Product_line'>
```



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.

In [185…
```
sns.countplot(y = 'Product_line', data=data, order = data['Product_line'].value_cou
```

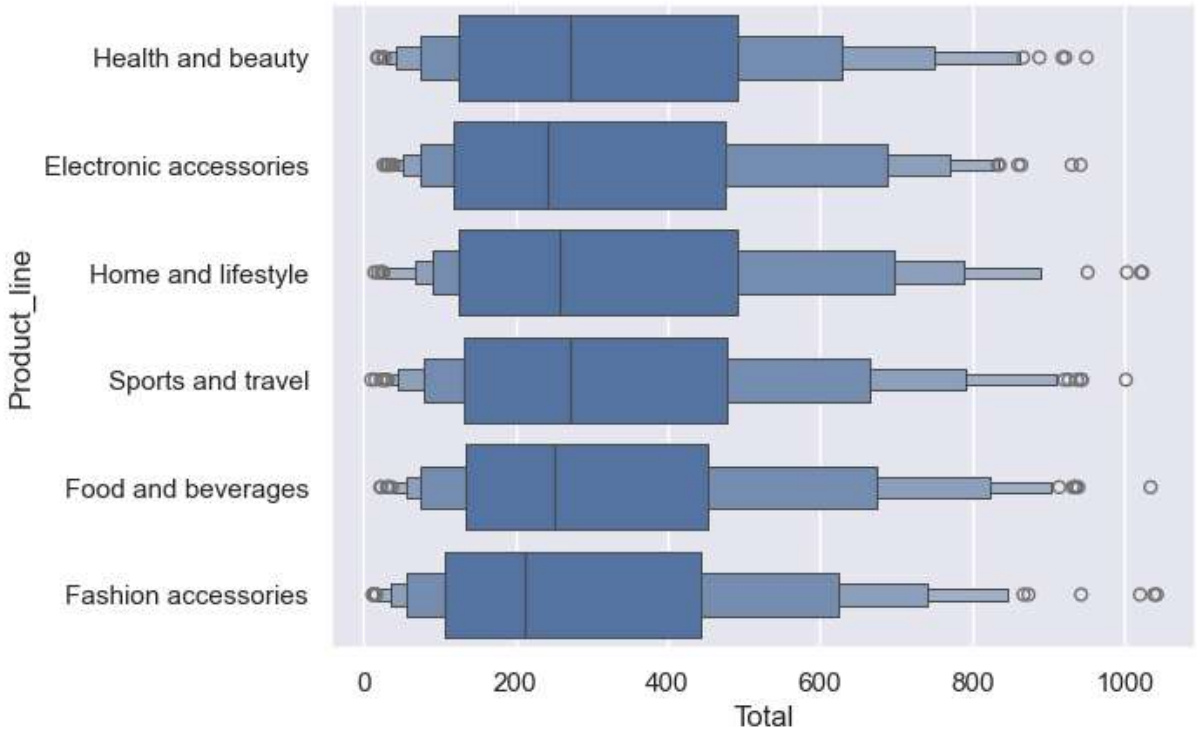
Out[185…
```
<Axes: xlabel='count', ylabel='Product_line'>
```



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

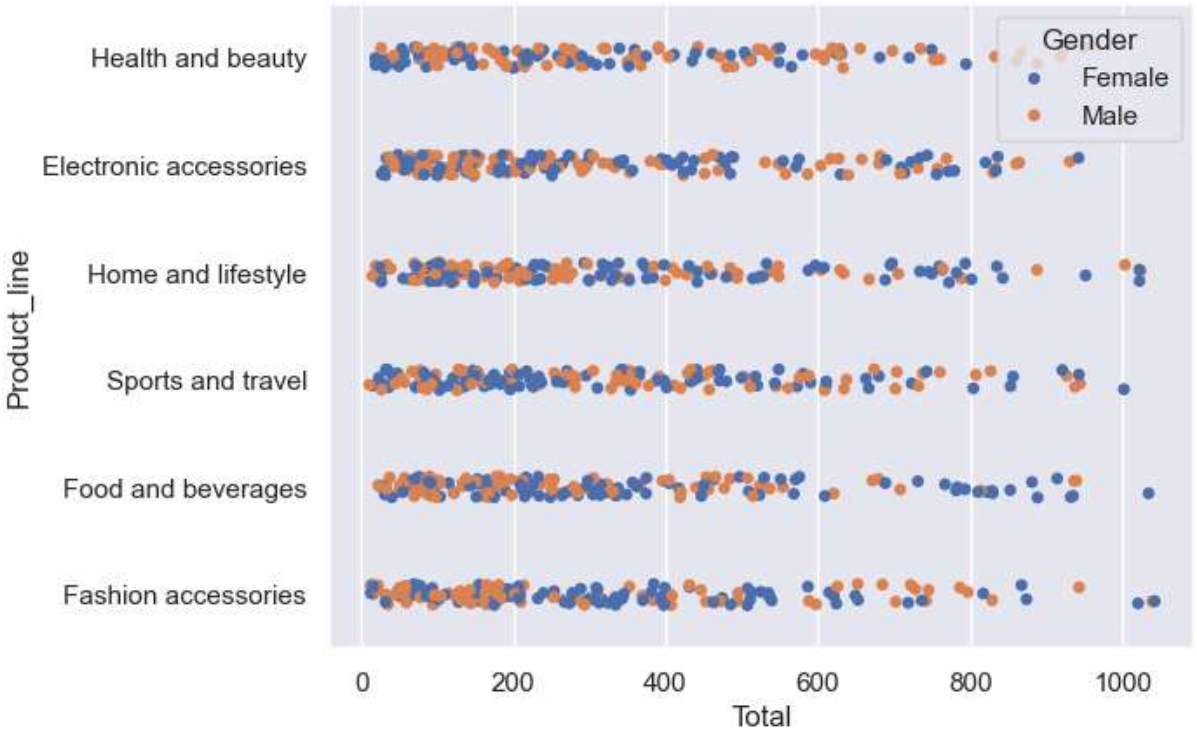
In [188…
```
sns.boxenplot(y = 'Product_line', x = 'Total', data=data )
```

Out[188…
```
<Axes: xlabel='Total', ylabel='Product_line'>
```



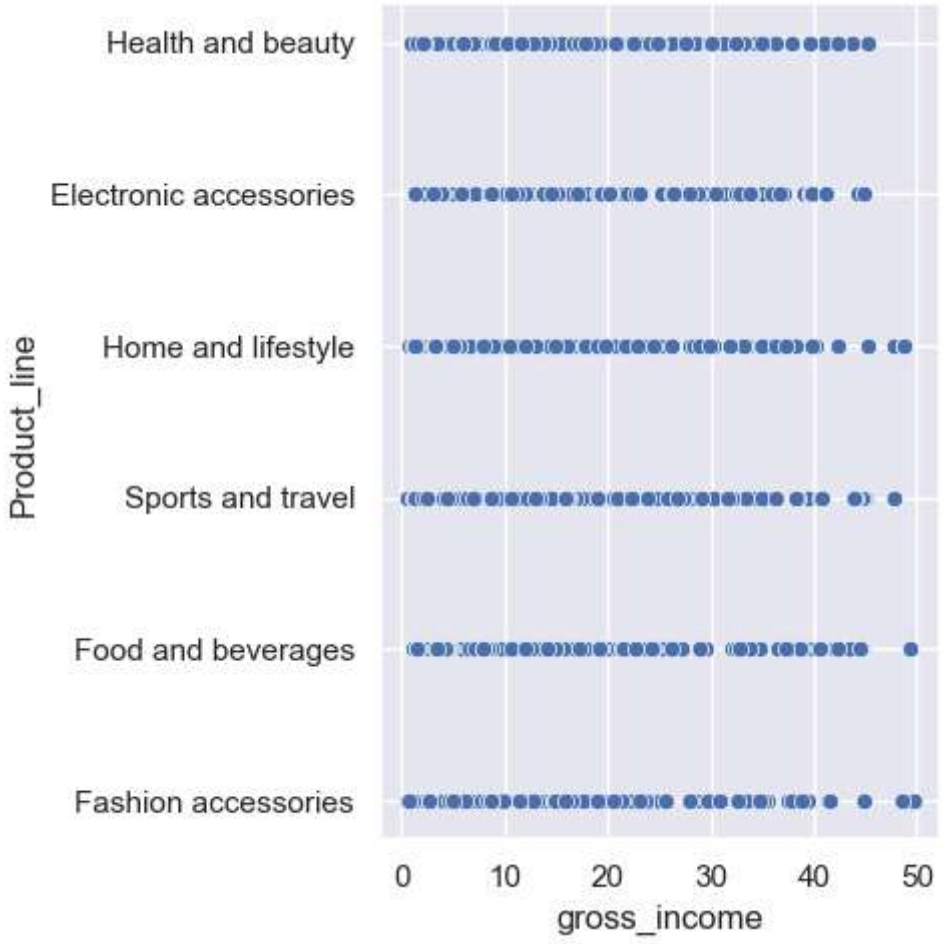
In [190…
```
sns.stripplot(y = 'Product_line', x = 'Total', hue = 'Gender', data=data )
```

Out[190…
```
<Axes: xlabel='Total', ylabel='Product_line'>
```



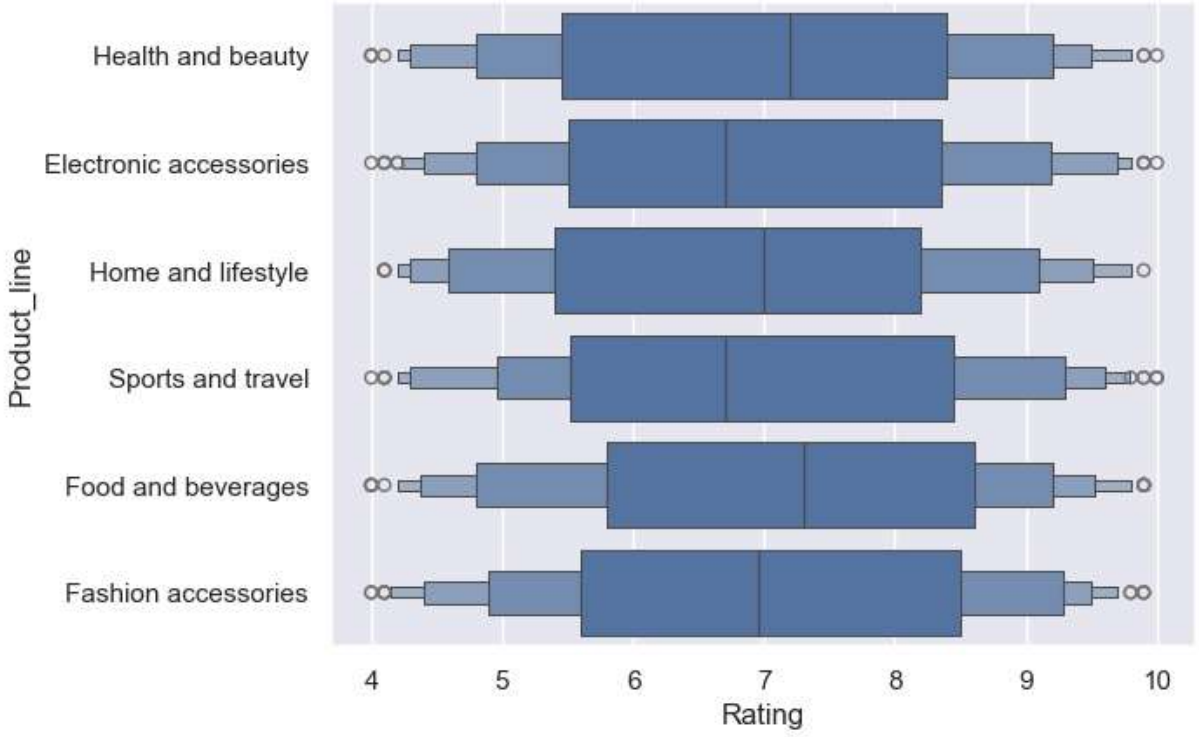
In [194…
```
sns.relplot(y = 'Product_line', x = 'gross_income', data=data )
```

Out[194…
```
<seaborn.axisgrid.FacetGrid at 0x213f005d510>
```



In [196…
```
sns.boxenplot(y = 'Product_line', x = 'Rating', data=data )
```

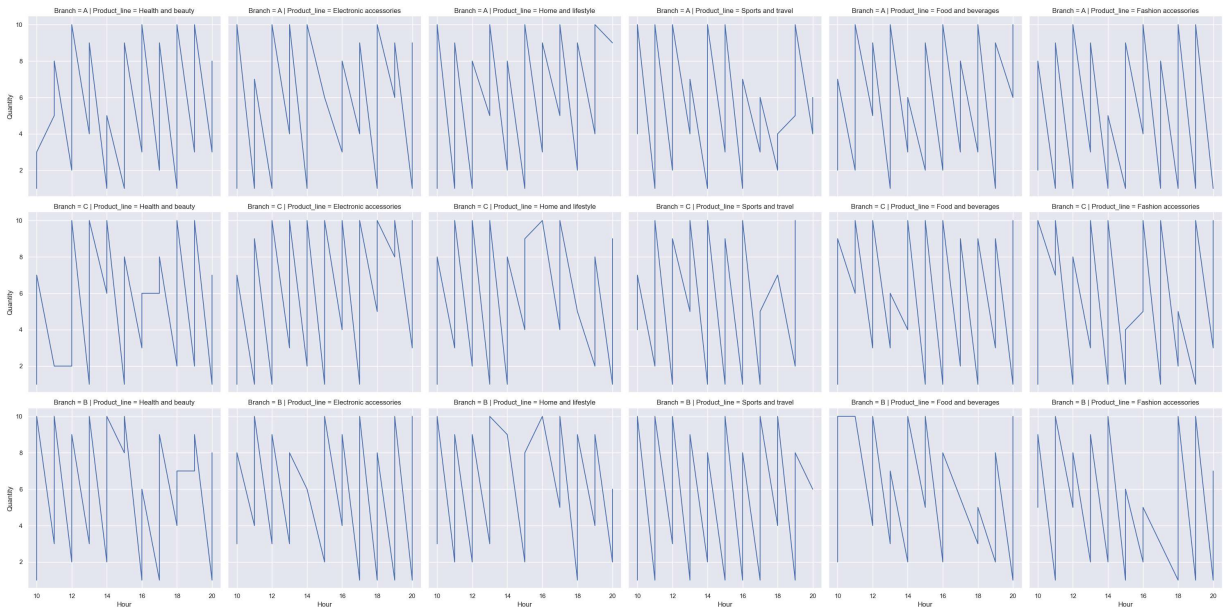
Out[196…
```
<Axes: xlabel='Rating', ylabel='Product_line'>
```



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.

In [200…
```
productCount  = sns.relplot(x="Hour",  y = 'Quantity', col= 'Product_line' , row= '
```



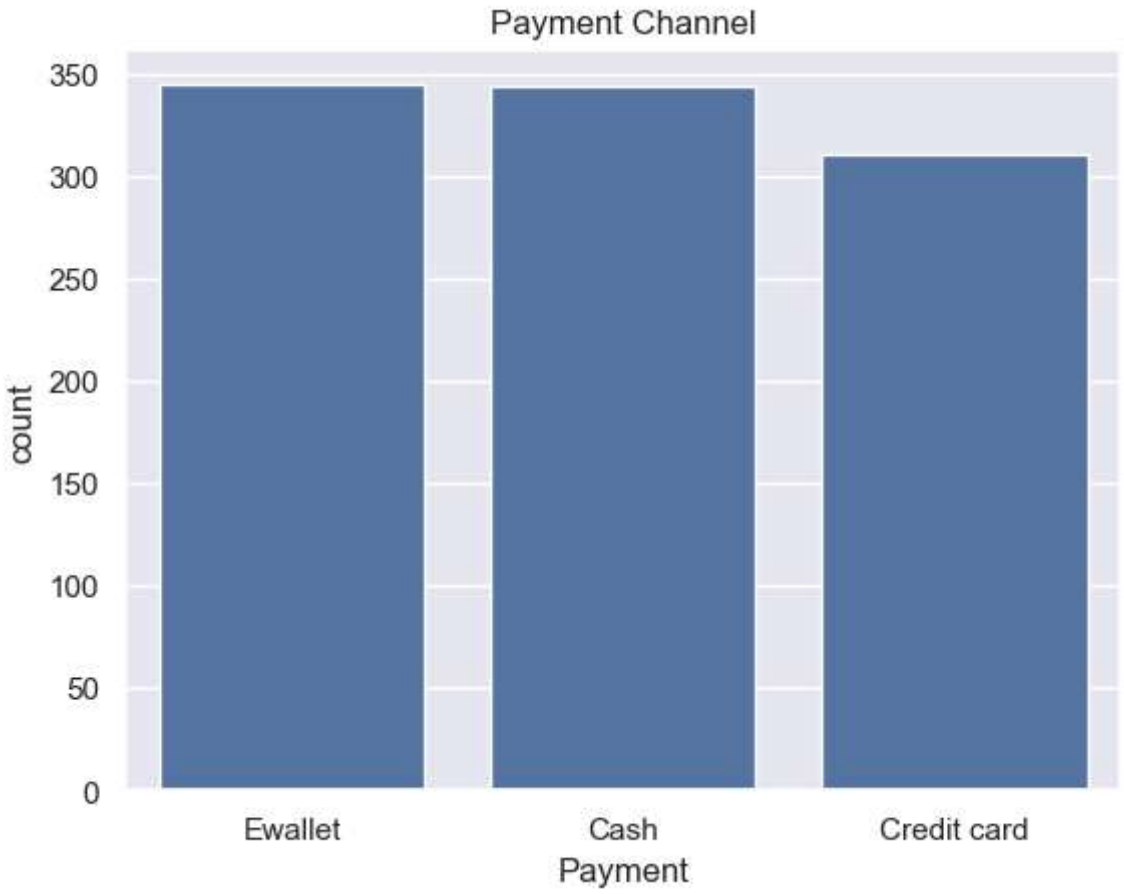
From the above plots, we can see that food and beverages sales usually high in all three branches at evening especially around 19:00

Payment Channel

Let see how customers make payment in this business

In [205…
```
sns.countplot(x="Payment", data =data).set_title("Payment Channel")
```

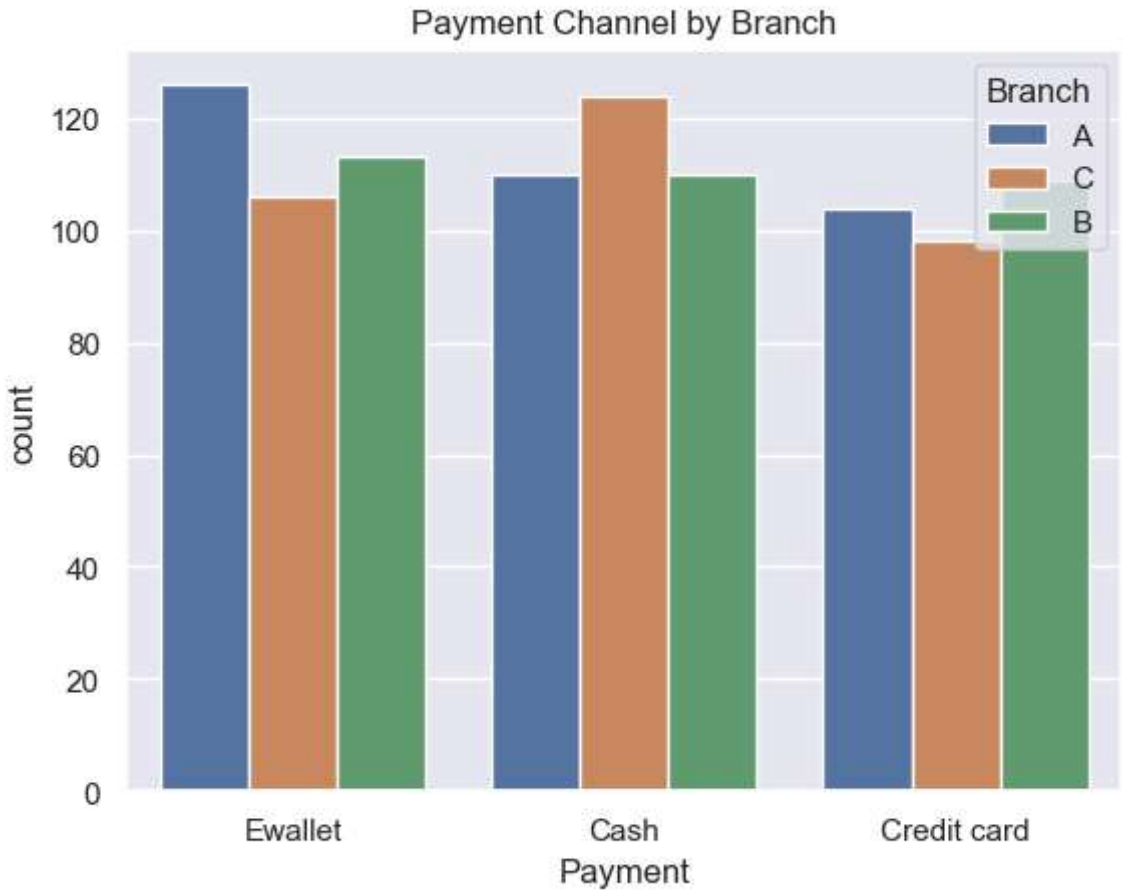
Out[205…
```
Text(0.5, 1.0, 'Payment Channel')
```



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

In [208…
```
sns.countplot(x="Payment", hue = "Branch", data =data).set_title("Payment Channel b
```

Out[208…
```
Text(0.5, 1.0, 'Payment Channel by Branch')
```



Customer Analysis

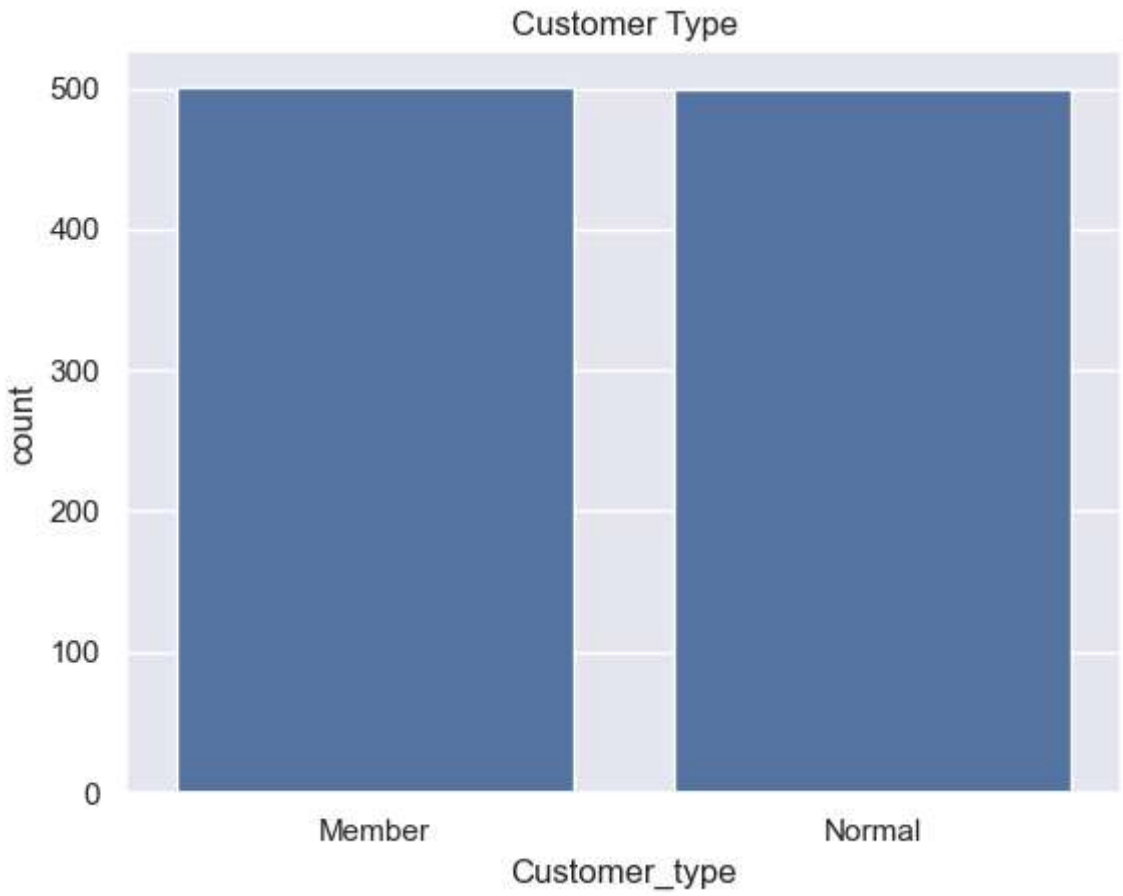
From inspection, there are two types of customers. Members and Normal. Let's see how many they are and where they are

In [212…
```
data['Customer_type'].nunique()
```

Out[212…
```
2
```

In [214…
```
sns.countplot(x="Customer_type", data =data).set_title("Customer Type")
```

Out[214…
```
Text(0.5, 1.0, 'Customer Type')
```



In [216…
```
sns.countplot(x="Customer_type", hue = "Branch", data =data).set_title("Customer Ty
```

Out[216…
```
Text(0.5, 1.0, 'Customer Type by Branch')
```

Does customer type influences the sales

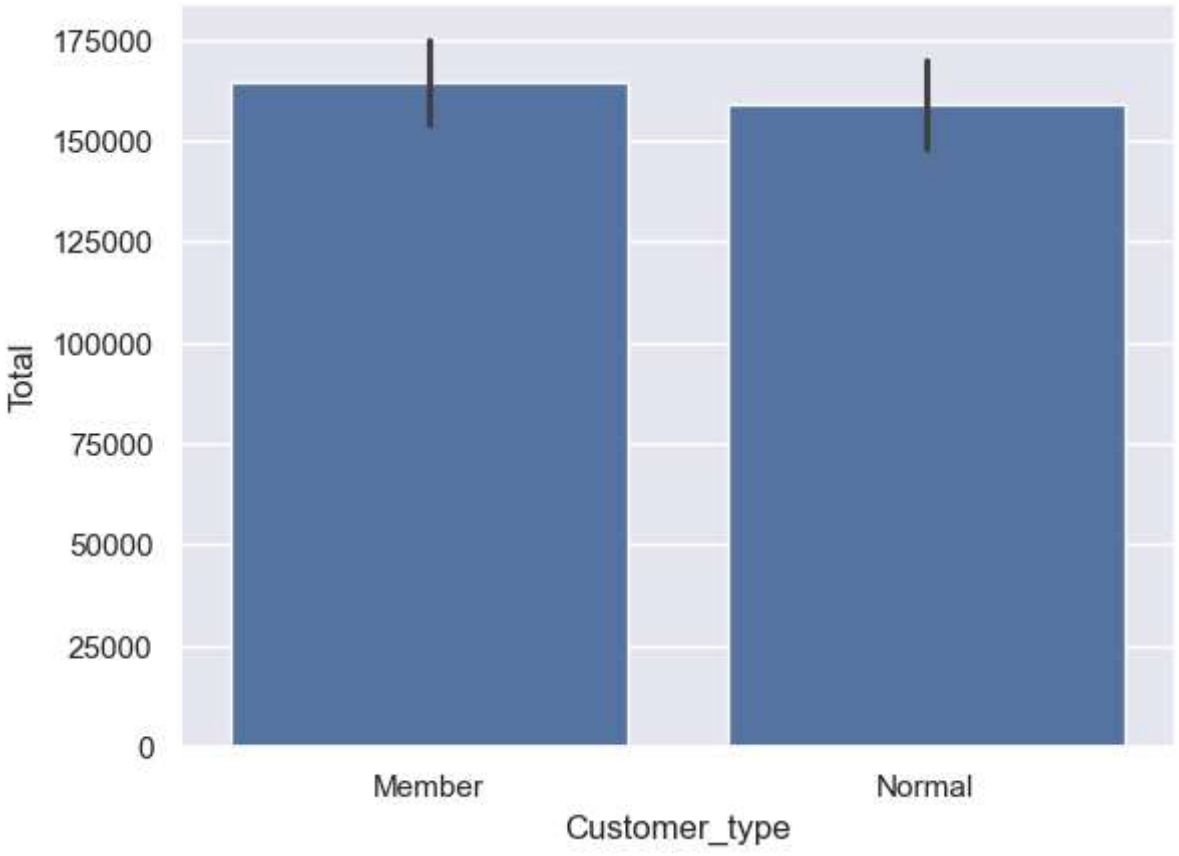
In [219…
```
data.groupby(['Customer_type']).agg({'Total': 'sum'})
```

Out[219…

Customer_type	Total
Member	164223.444
Normal	158743.305

In [221…
```
sns.barplot(x="Customer_type", y="Total", estimator = sum, data=data)
```

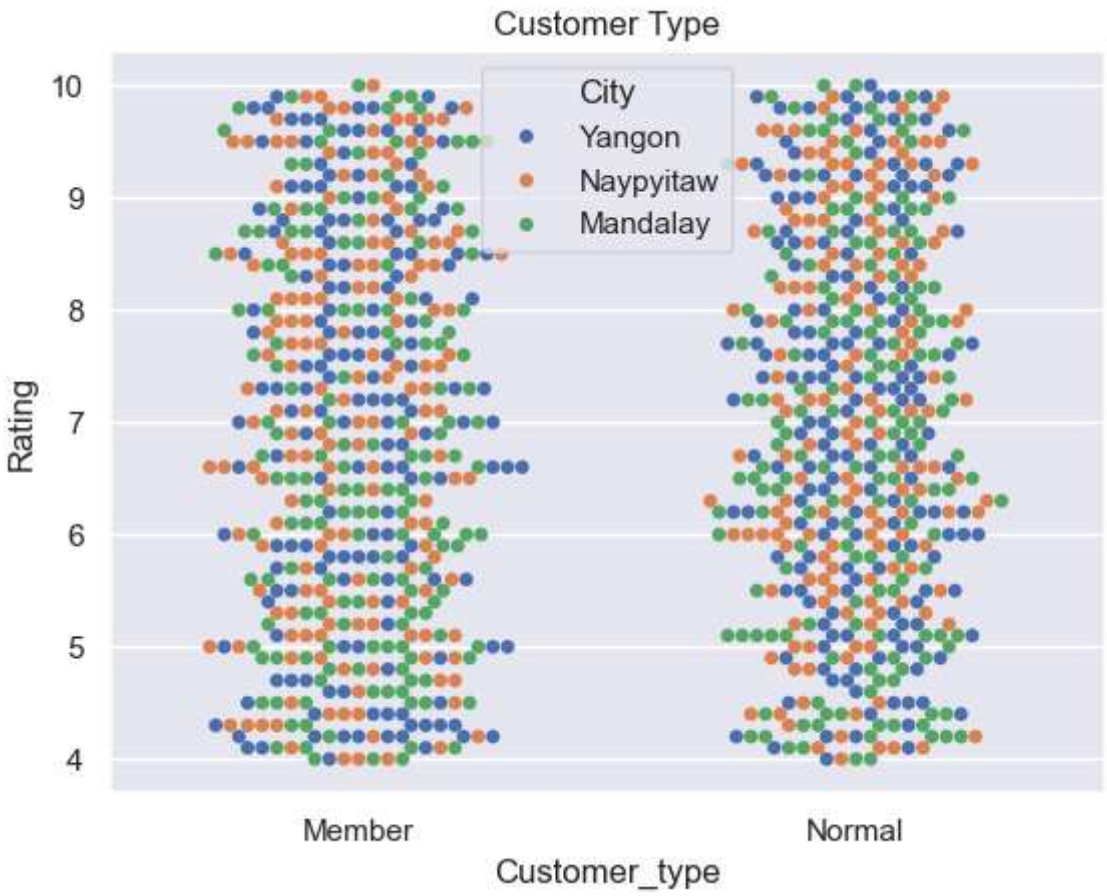
Out[221… `<Axes: xlabel='Customer_type', ylabel='Total'>`



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

In [224…
```
sns.swarmplot(x="Customer_type",  y = "Rating",  hue = "City", data =data).set_titl
```

Out[224…
```
Text(0.5, 1.0, 'Customer Type')
```

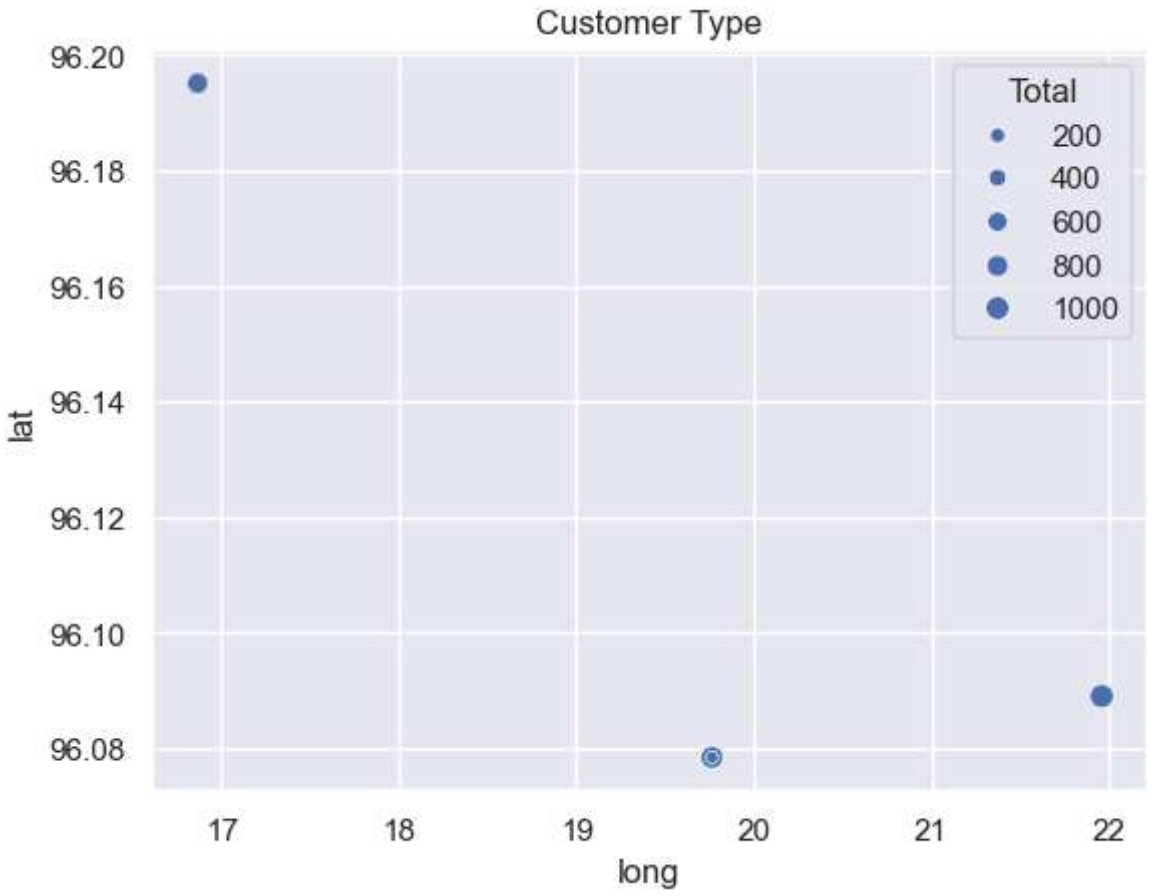


With the use of google search, I was able to get the longitude and latitude of each cities. We can

In [227…
```
long = {"Yangon": 16.8661, "Naypyitaw": 19.7633, "Mandalay": 21.9588 }
lat = {"Yangon": 96.1951, "Naypyitaw": 96.0785, "Mandalay": 96.0891 }
for set in data:
    data['long'] = data['City'].map(long)
    data['lat'] = data['City'].map(lat)
```

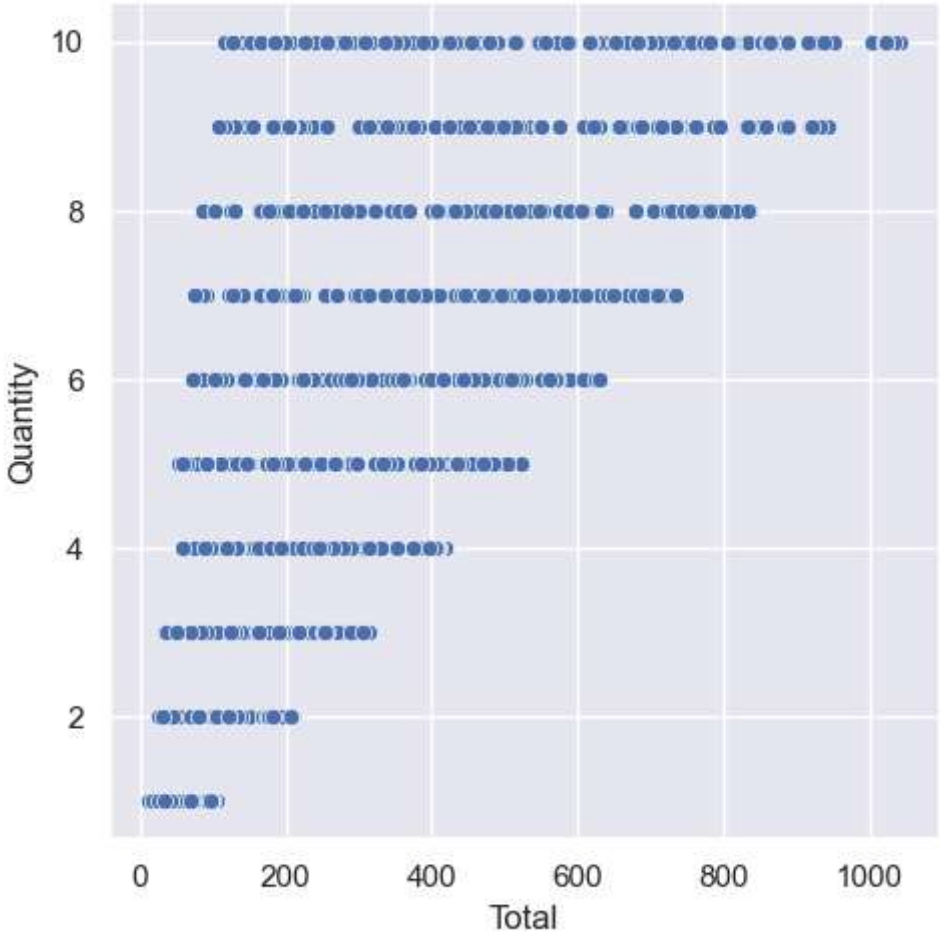
In [229…
```
sns.scatterplot(x="long",  y = "lat",size = "Total", data =data, legend = "brief").
```

Out[229…
```
Text(0.5, 1.0, 'Customer Type')
```



In [231…
```
sns.relplot(x="Total",  y = "Quantity", data =data)
```

Out[231…
```
<seaborn.axisgrid.FacetGrid at 0x213ea298410>
```



5. Outlier Detection

The presence of outliers in a classification or regression dataset can result in a poor fit and lower predictive modeling performance, therefore we should see there are ouliers in the data.

In [246…
```
data=data.drop(labels=['Invoice ID'],axis=1)
```

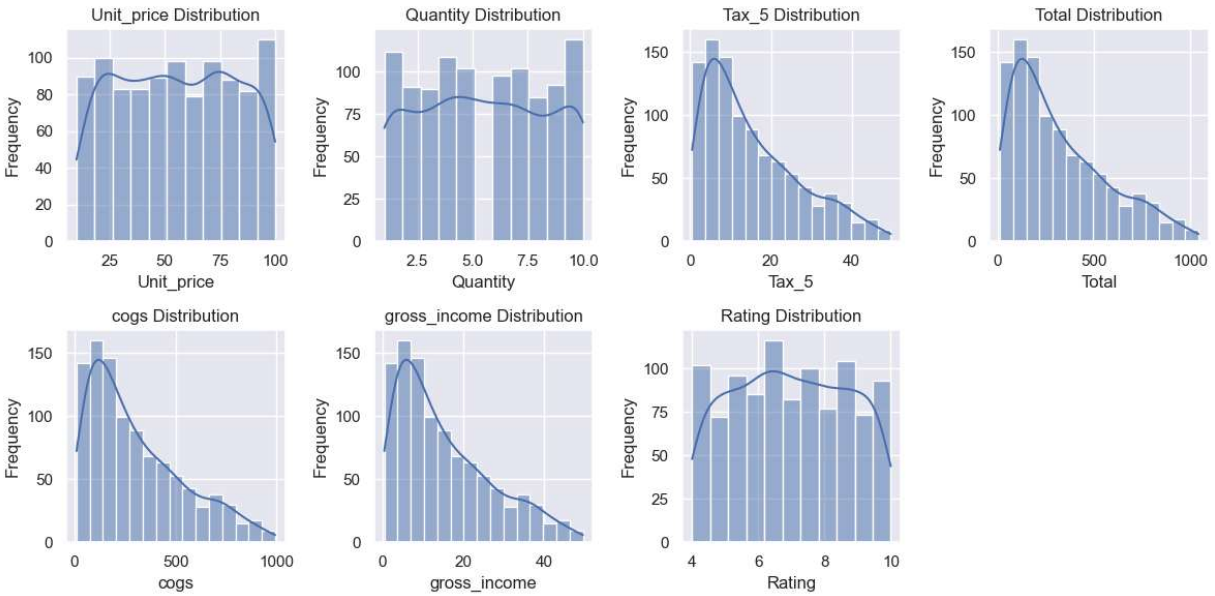
In [260…
```
# Select numerical columns
numerical_cols = ['Unit_price', 'Quantity', 'Tax_5', 'Total', 'cogs', 'gross_income
```

In [264…
```
# Calculate summary statistics
summary_stats = data[numerical_cols].describe()

print("Summary Statistics:")
print(summary_stats)
```

```
Summary Statistics:
       Unit_price     Quantity        Tax_5        Total        cogs  \
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.00000
mean     55.672130     5.510000    15.379369   322.966749   307.58738
std      26.494628     2.923431    11.708825   245.885335   234.17651
min      10.080000     1.000000     0.508500    10.678500    10.17000
25%      32.875000     3.000000     5.924875   124.422375   118.49750
50%      55.230000     5.000000    12.088000   253.848000   241.76000
75%      77.935000     8.000000    22.445250   471.350250   448.90500
max      99.960000    10.000000    49.650000  1042.650000   993.00000

       gross_income      Rating
count   1000.000000  1000.00000
mean      15.379369     6.97270
std       11.708825     1.71858
min        0.508500     4.00000
25%        5.924875     5.50000
50%       12.088000     7.00000
75%       22.445250     8.50000
max       49.650000    10.00000
```

In [268…

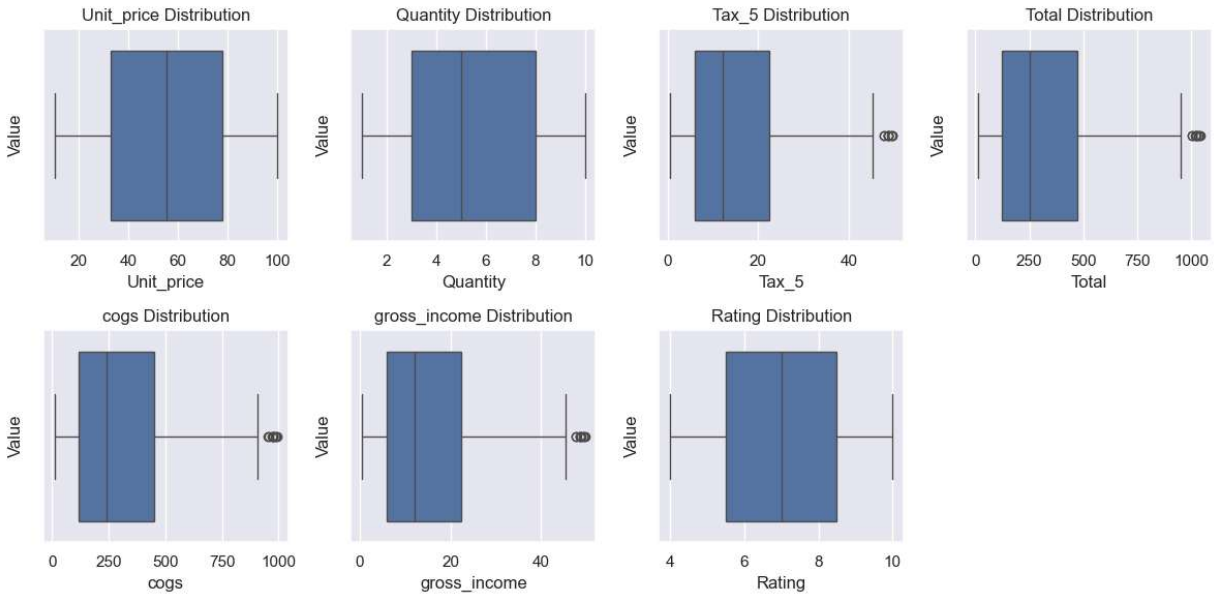
```
# Histograms for each numerical column
plt.figure(figsize=(12, 6))

for i, col in enumerate(numerical_cols):
    plt.subplot(2, 4, i+1)
    sns.histplot(data=data, x=col, kde=True)
    plt.title(f"{col} Distribution")
    plt.xlabel(col)
    plt.ylabel("Frequency")

plt.tight_layout()
plt.show()
```



In [270…

```
# Box plots for each numerical column
plt.figure(figsize=(12, 6))
```

```
for i, col in enumerate(numerical_cols):
    plt.subplot(2, 4, i+1)
    sns.boxplot(data=data, x=col)
    plt.title(f"{col} Distribution")
    plt.xlabel(col)
    plt.ylabel("Value")

plt.tight_layout()
plt.show()
```



In [276…

```
# Outlier detection using IQR method
outliers = {}
for col in numerical_cols:
    q1 = data[col].quantile(0.25)
    q3 = data[col].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    outliers[col] = data[(data[col] < lower_bound) | (data[col] > upper_bound)]

print("Outliers by Column:")
for col, outlier_values in outliers.items():
    print(f"{col}: {outlier_values.shape[0]} outliers")
```

```
Outliers by Column:
Unit_price: 0 outliers
Quantity: 0 outliers
Tax_5: 9 outliers
Total: 9 outliers
cogs: 9 outliers
gross_income: 9 outliers
Rating: 0 outliers
```

6. Check for Rare Categories

Some categories may appear a lot in the dataset, whereas some other categories appear only in a few number of observations.

Rare values in categorical variables tend to cause over-fitting, particularly in tree based methods. Rare labels may be present in training set, but not in test set, therefore causing over-fitting to the train set. Rare labels may appear in the test set, and not in the train set. Thus, the machine learning model will not know how to evaluate it.

In [283…
```
categorical = [var for var in data.columns if data[var].dtype=='O']
```

In [288…
```
for var in categorical:
    print(data[var].value_counts() / np.float64(len(data)))
    print()
    print()
```

```
Branch
A    0.340
B    0.332
C    0.328
Name: count, dtype: float64


City
Yangon       0.340
Mandalay     0.332
Naypyitaw    0.328
Name: count, dtype: float64


Customer_type
Member    0.501
Normal    0.499
Name: count, dtype: float64


Gender
Female    0.501
Male      0.499
Name: count, dtype: float64


Product_line
Fashion accessories       0.178
Food and beverages        0.174
Electronic accessories    0.170
Sports and travel         0.166
Home and lifestyle        0.160
Health and beauty         0.152
Name: count, dtype: float64


Date
2/7/2019     0.020
2/15/2019    0.019
3/14/2019    0.018
3/2/2019     0.018
1/8/2019     0.018
             ...
1/4/2019     0.006
2/28/2019    0.006
2/21/2019    0.006
3/17/2019    0.006
2/1/2019     0.006
Name: count, Length: 89, dtype: float64


Payment
Ewallet        0.345
Cash           0.344
Credit card    0.311
Name: count, dtype: float64
```

As shown above, there is no rare category in the categorical variables.