Python初級數據分析員證書

（六）數據分析 及可視化專案

13. 數據分析專案

Demo 9 – Supermarket

Review

- Statistics
- Hypothesis testing
- Algebra
- Linear regression
- Propositional logic
- Python
- R
- SQL

- Pandas, NumPy, SciPy
- Data Visualization, Matplotlib, Seaborn, Plotly
- Dashboard Visualization, Business Intelligence
- Storytelling

Chapter Summary

- Scenario
- Data Import
- Handling date and time
- Branch sales
- Customer behaviour
- Product line
- Product rating
- Recommendation

Scenario

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.

You are a data analyst of a supermarket and are required to suggest at least 5 solutions to increase sales in coming 3rd quarter.

Data Import

```
1 import numpy as np
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
```

```
1 sales = pd.read_csv('supermarket_sales.csv')
2 sales.sample(3)
```

	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
75	502-05-1910	A	Yangon	Normal	Male	Health and beauty	65.18	3	9.777	205.317	2/25/2019	20:35	Credit card	195.54	4.761905	9.777	6.3
42	860-73-6466	A	Yangon	Member	Female	Sports and travel	39.47	2	3.947	82.887	3/2/2019	16:16	Credit card	78.94	4.761905	3.947	5.0
37	137-63-5492	C	Naypyitaw	Normal	Male	Electronic accessories	58.76	10	29.380	616.980	1/29/2019	14:26	Ewallet	587.60	4.761905	29.380	9.0

Change the object Date and Time to datetime

```
1 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
```

Since we need to investigate the shopping behaviour, we need to change the dtype that pandas understand as time.

Change to obj [‘Date’] to datetime [‘date’]

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

```
1 sales['date']
```

```
0      2019-01-05
1      2019-03-08
2      2019-03-03
3      2019-01-27
4      2019-02-08
          ...
995    2019-01-29
996    2019-03-02
997    2019-02-09
998    2019-02-22
999    2019-02-18
Name: date, Length: 1000, dtype: datetime64[ns]
```

```
1 sales['date'].dtype
```

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

Change the obj [‘Date’] to datetime [‘date’]

```
1 sales['day'] = (sales['date']).dt.day
2 sales['month'] = (sales['date']).dt.month
3 sales['year'] = (sales['date']).dt.year
```

```
1 sales[['day','month','year']]
```

	day	month	year
0	5	1	2019
1	8	3	2019
2	3	3	2019

Also, we add day, month, year columns for further specific analysis.

Change the obj to datetime and classify as Hour

```
1 sales['Time'] = pd.to_datetime(sales['Time'])
2 sales['Hour'] = (sales['Time']).dt.hour
```

```
1 # the opening hour
2 sorted(sales['Hour'].unique())
```

```
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

Investigate the categorical columns

```
1  # the object columns
2  [col for col in sales.columns if sales[col].dtype == "object"]
```

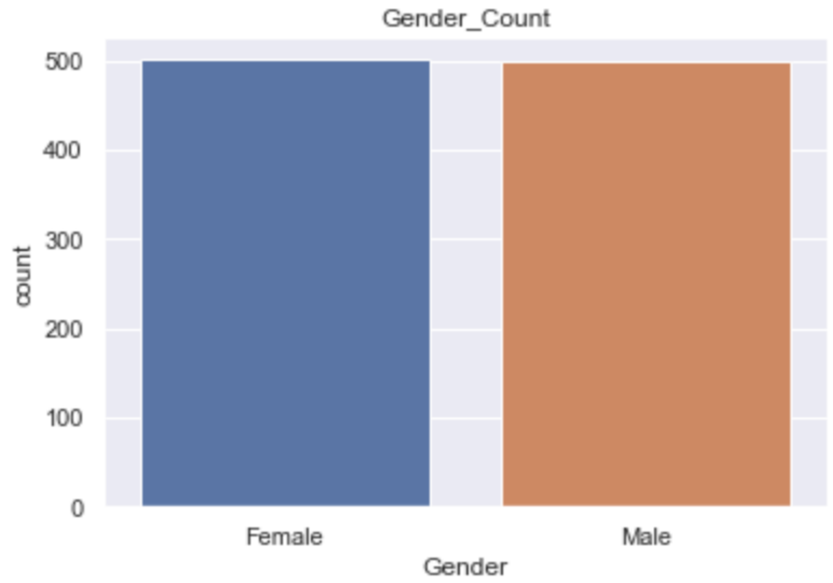
```
['Invoice ID',
 'Branch',
 'City',
 'Customer type',
 'Gender',
 'Product line',
 'Date',
 'Payment']
```

```
1  print("# of unique values in Branch: {0}".format(len(sales['Branch'].unique().tolist())))
2  print("# of unique values in City: {0}".format(len(sales['City'].unique().tolist())))
3  print("# of unique values in Customer Type: {0}".format(len(sales['Customer type'].unique().tolist())))
4  print("# of unique values in Gender: {0}".format(len(sales['Gender'].unique().tolist())))
5  print("# of unique values in Product Line: {0}".format(len(sales['Product line'].unique().tolist())))
6  print("# of unique values in Payment: {0}".format(len(sales['Payment'].unique().tolist())))
```

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

Client gender

```
1 sns.set(style="darkgrid")
2 genderCount  = sns.countplot(x="Gender", data =sales).set_title("Gender_Count")
```




```
1 # Female client
2 sales['Gender'].loc[sales['Gender']=='Female'].count()
```

```
501
```

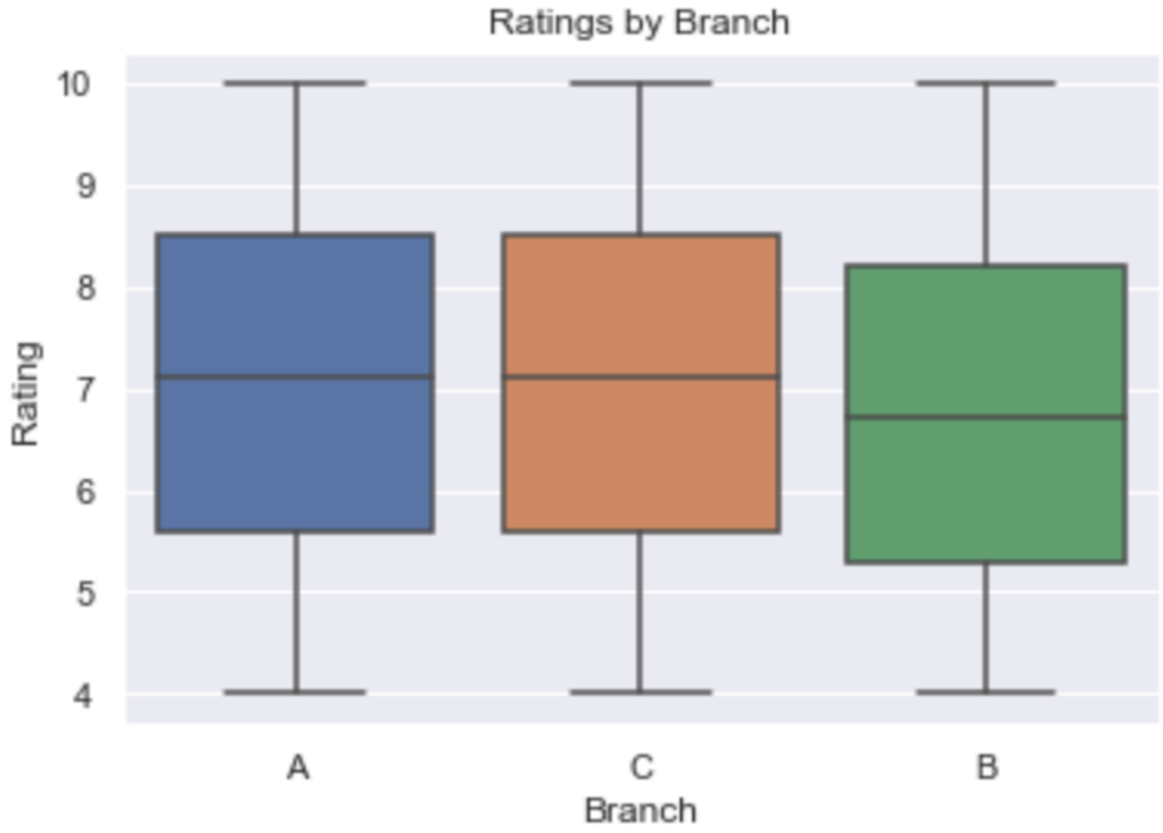
```
1 # Male client
2 sales['Gender'].loc[sales['Gender']=='Male'].count()
```

```
499
```

Branch ratings

```
1 sns.boxplot(x="Branch", y = "Rating" ,data =sales).set_title("Ratings by Branch")
```

```
Text(0.5, 1.0, 'Ratings by Branch')
```



Something bad in branch B ?

Product sales per hour

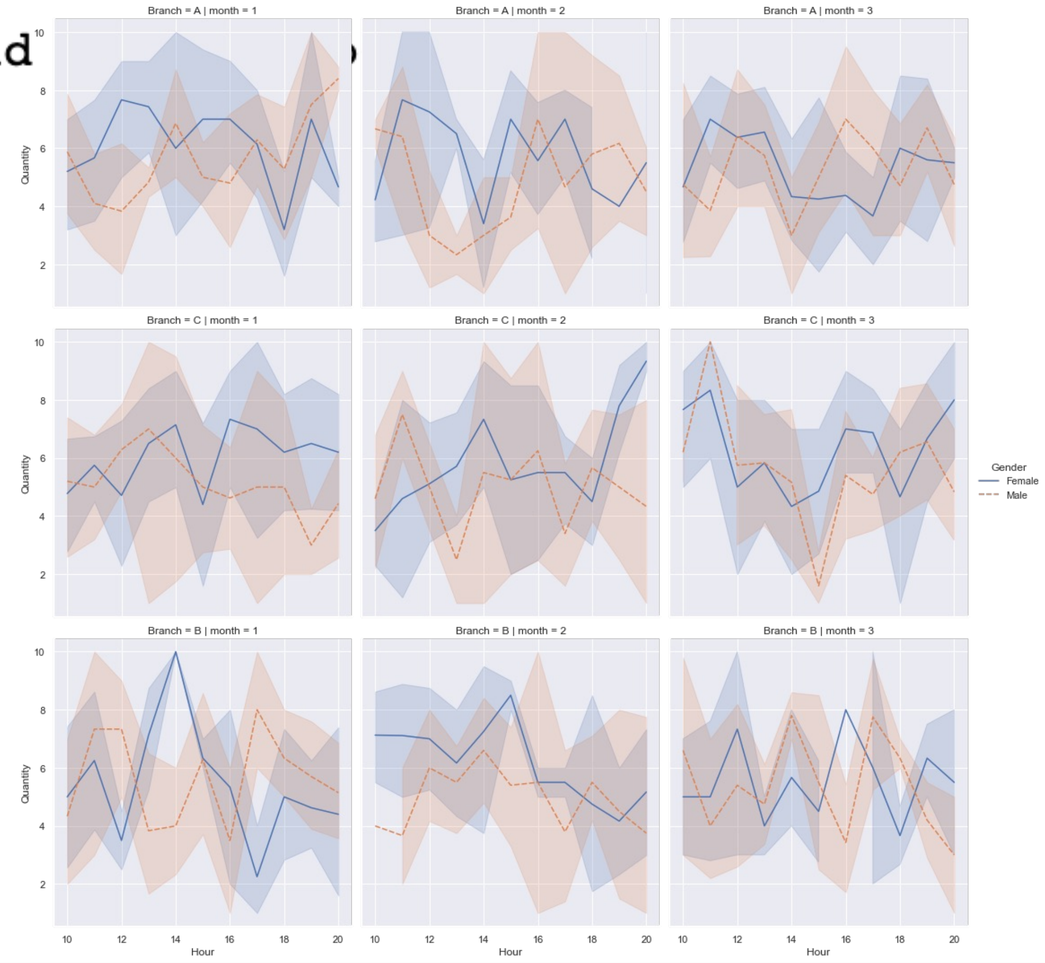
```
1  sns.lineplot(x="Hour",  y = 'Quantity',data =sales)\
2  .set_title("Product Sales per Hour")
```

The peak hour is around 14:00, what do you think the clients' probably are?

Compare each branch sales quantity in 3 months

```
1  sns.relplot(x="Hour",  y = 'Quantity', col= 'month' , row= 'Branch',
2              kind="line", hue="Gender", style="Gender", data =sales)
```

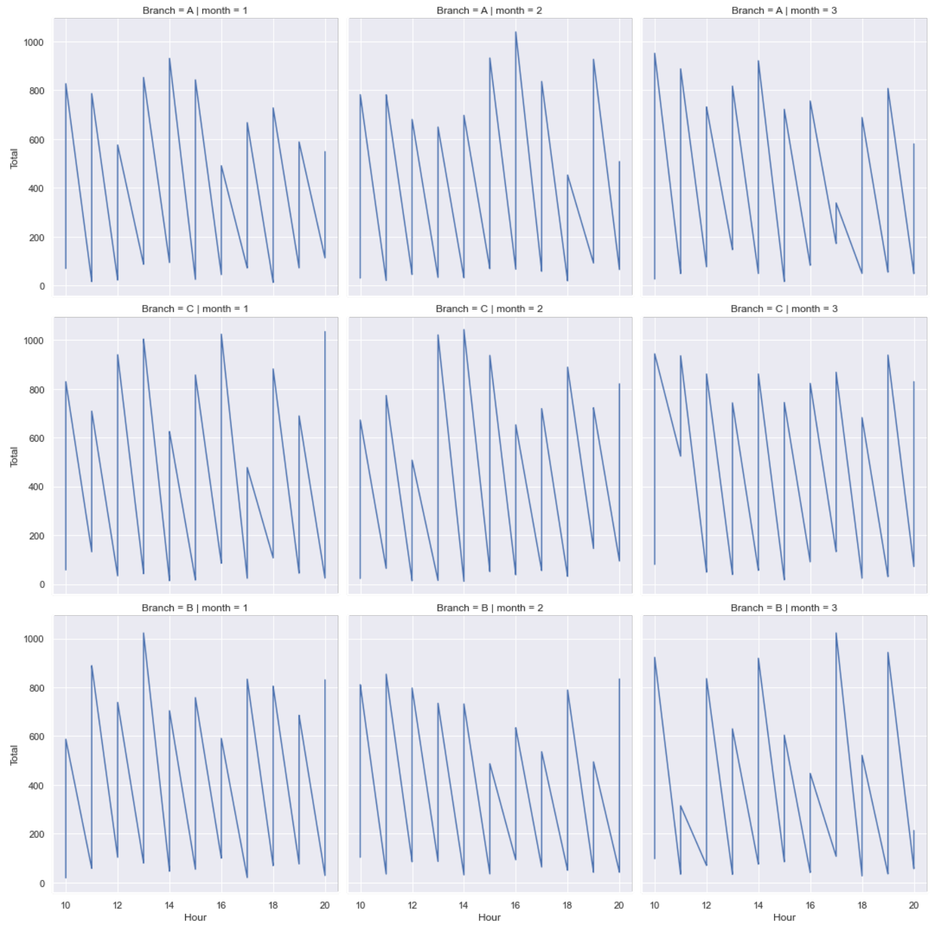
```
<seaborn.axisgrid.FacetGrid
```



Looks like each branch has their own peak hour and typical gender client.

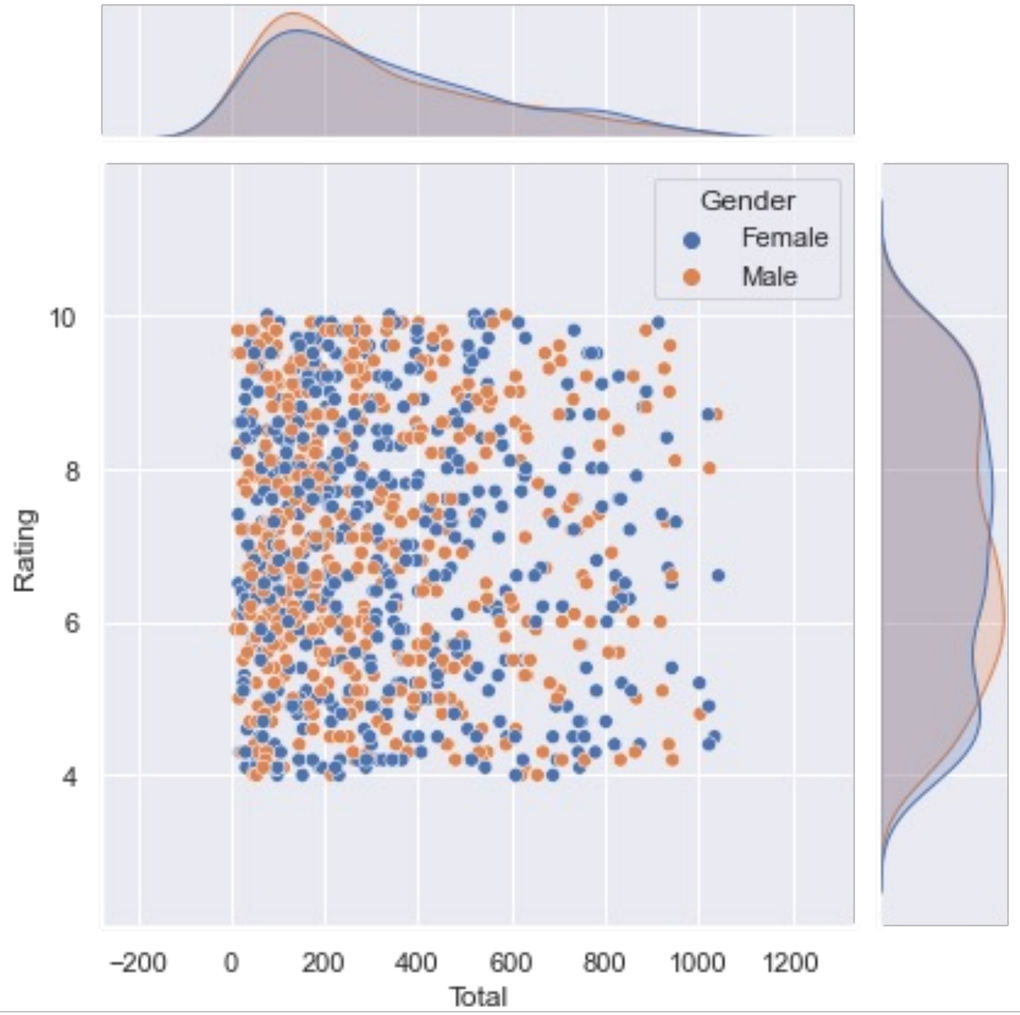
Compare each branch sales amount in 3 months

```
1  sns.relplot(x="Hour",  y = 'Total', col= 'month' , row= 'Branch',
2              estimator = None, kind="line", data =sales)
```



Ratings vs Amount by gender

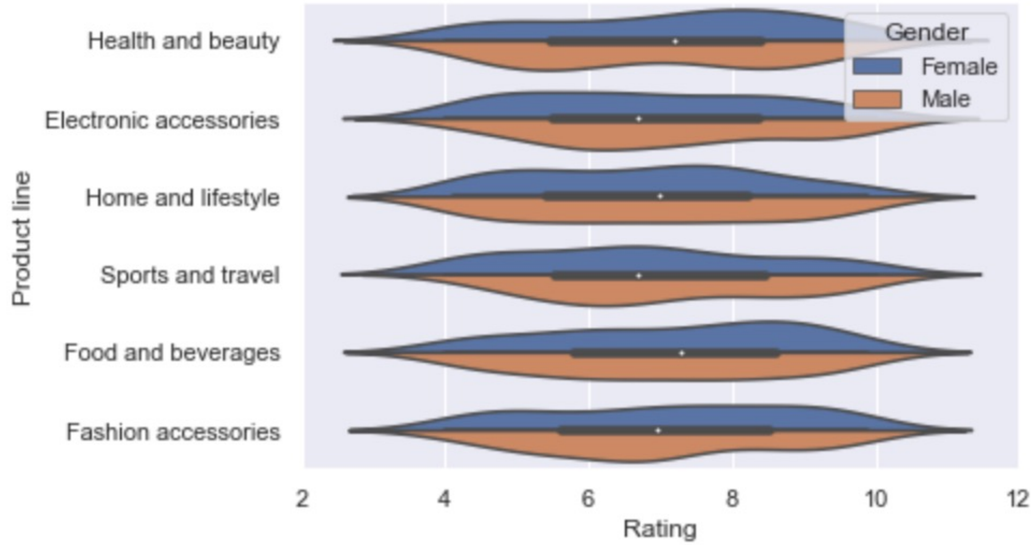
```
1 sns.jointplot(data =sales, x="Total", y = "Rating", hue="Gender",height=8 )
```



Product ratings by gender

```
1 sns.violinplot( data=sales,y = 'Product line', x = 'Rating',
2                 hue = 'Gender',split=True)
```

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




```
1 sales['Rating'].loc[sales['Gender']=='Female'].mean()
```

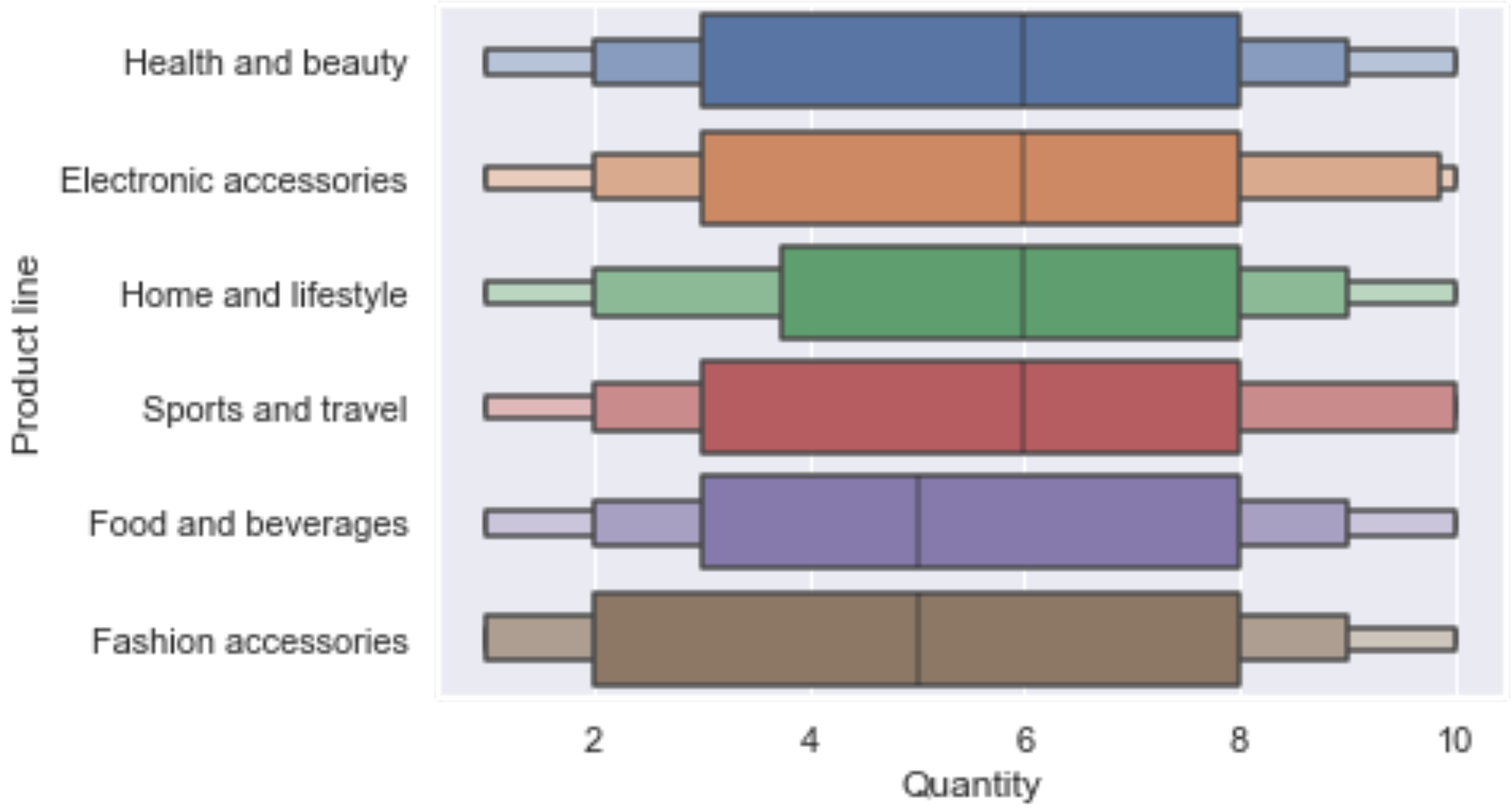
```
6.964471057884231
```

```
1 sales['Rating'].loc[sales['Gender']=='Male'].mean()
```

```
6.980961923847695
```

Product type vs Shopping quantity

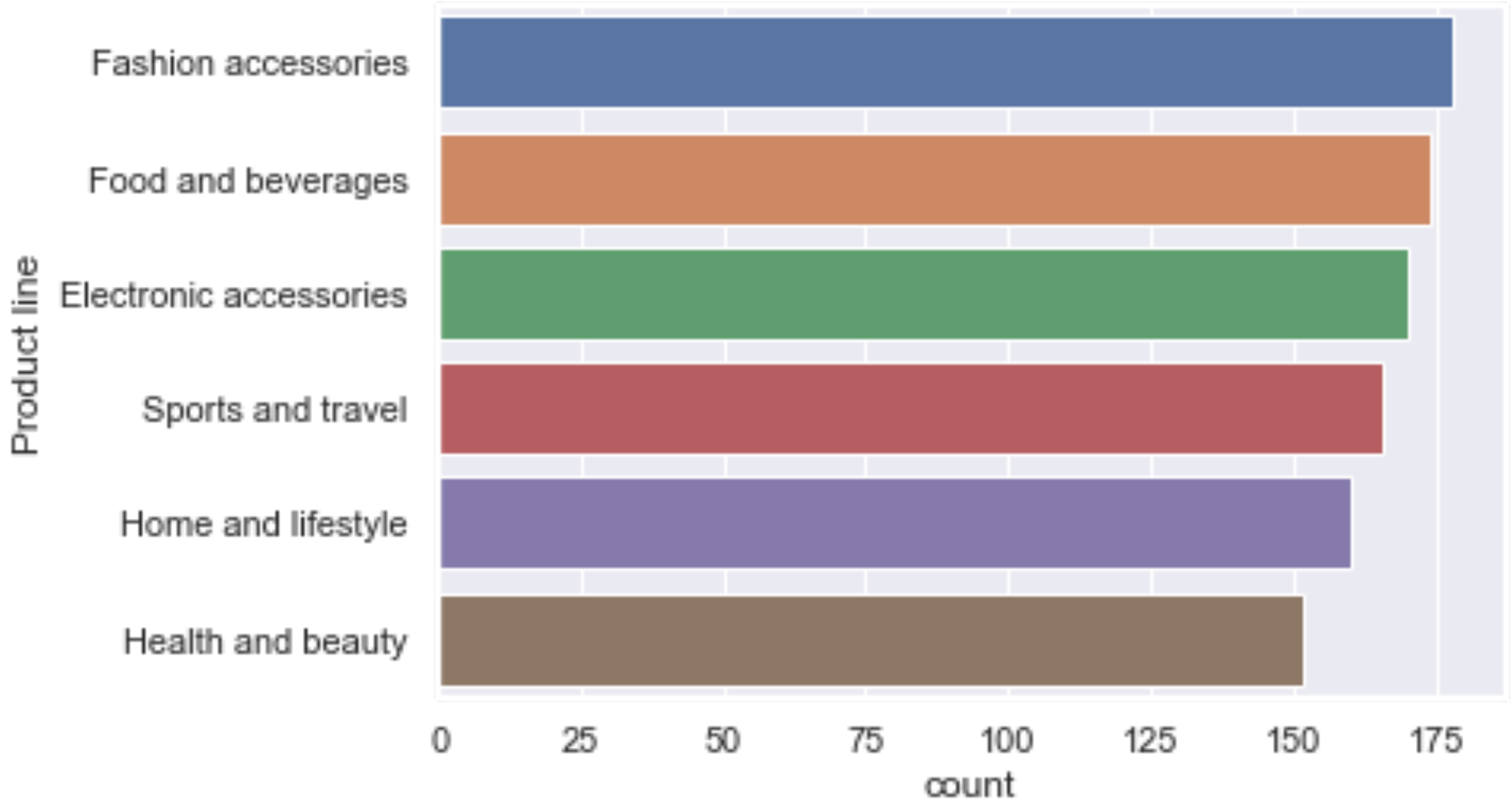
```
1 sns.boxenplot(y = 'Product line', x = 'Quantity', data=sales )
```



F&B is not the largest?!

Product line

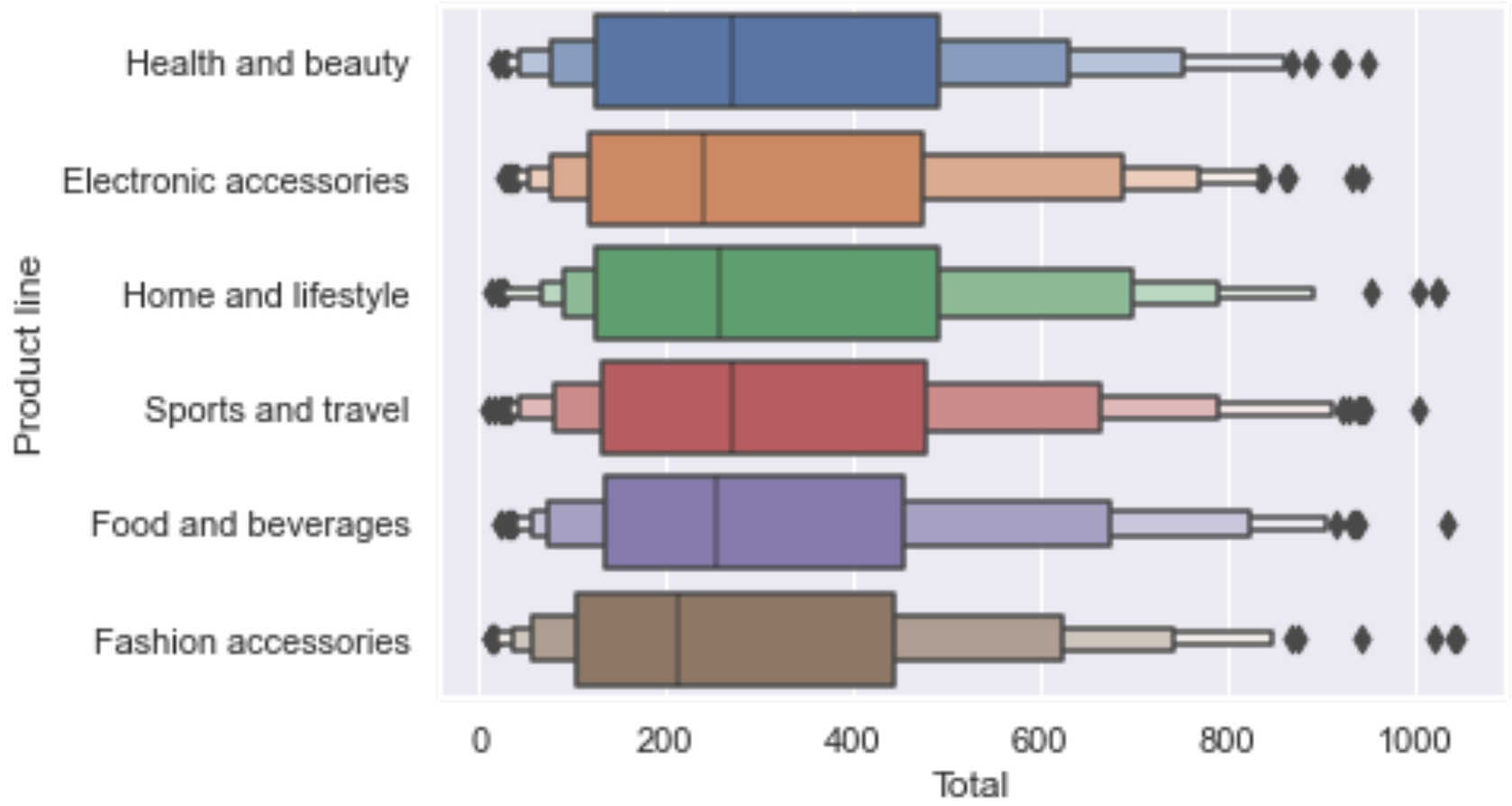
```
sns.countplot(y = 'Product line', data=sales,
              order = sales['Product line'].value_counts().index )
```



The top 2 supply is the bottom in demand?!

Product type vs Shopping amount

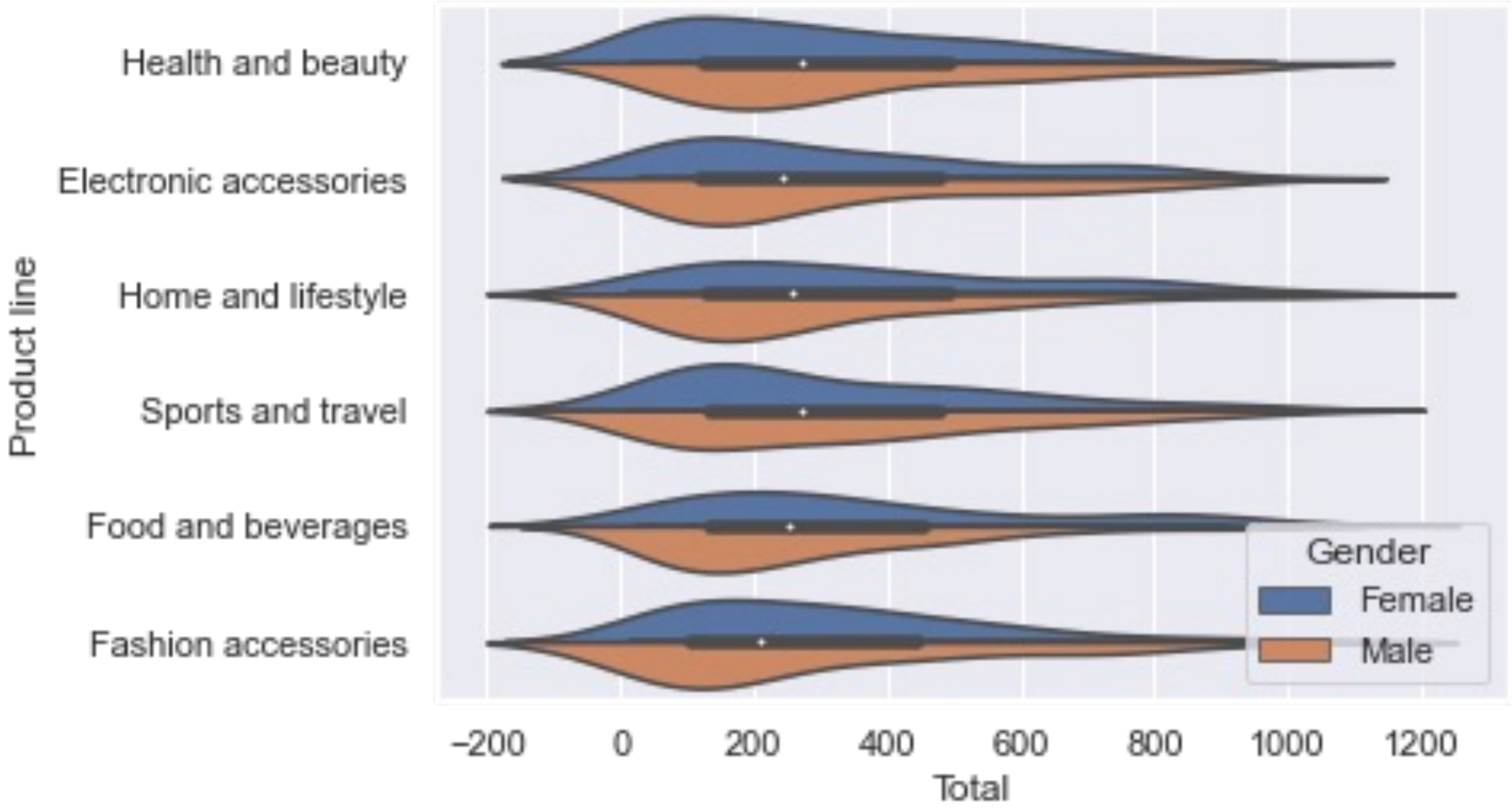
```
1  sns.boxenplot(y = 'Product line', x = 'Total', data=sales )
```



The sales total of F&B and Fashion accessories are the lower in amount!!

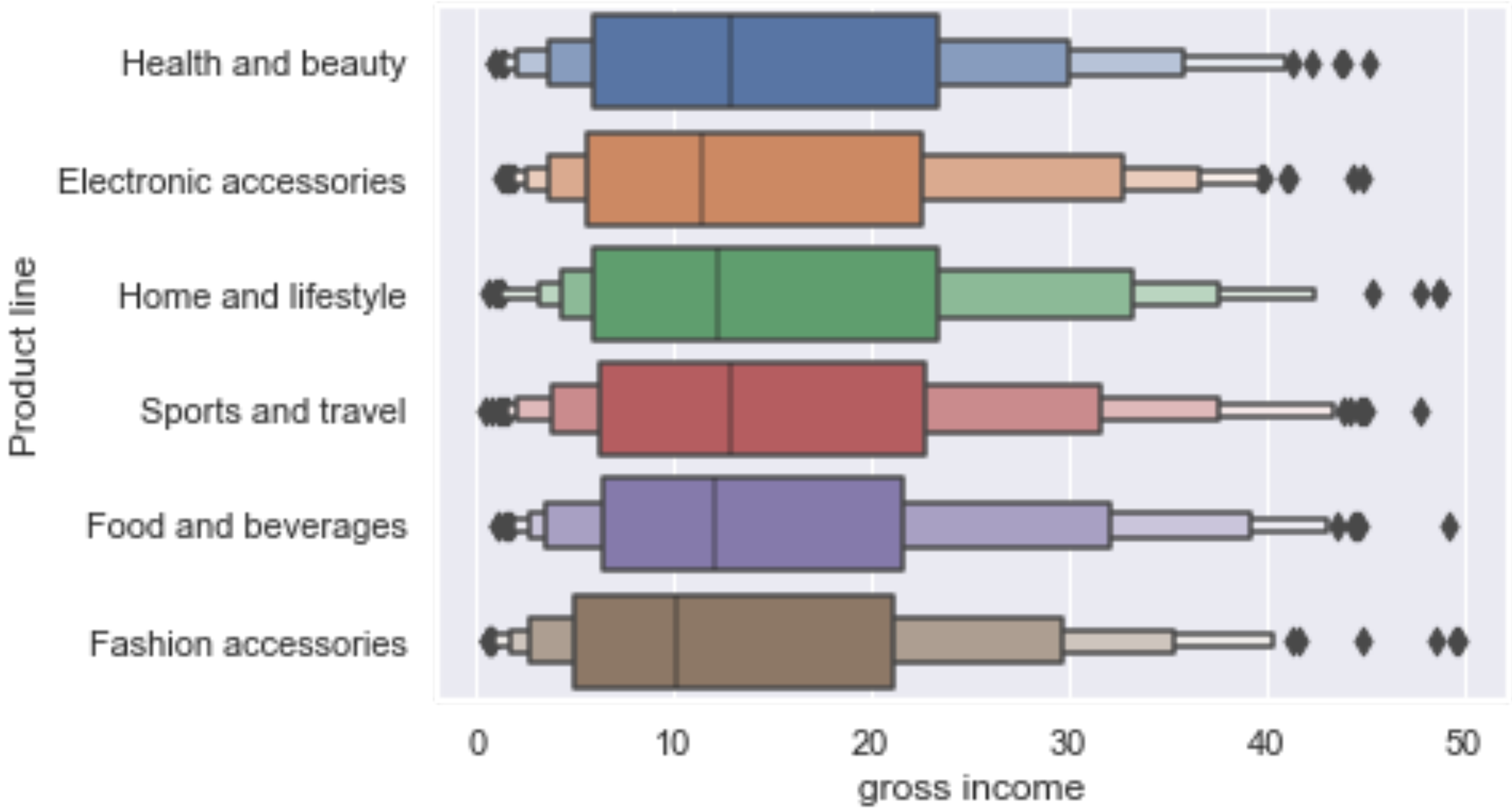
Shopping amount vs Products on Gender

```
1  sns.violinplot( data=sales,y = 'Product line', x = 'Total',
2                  hue = 'Gender',split=True)
```



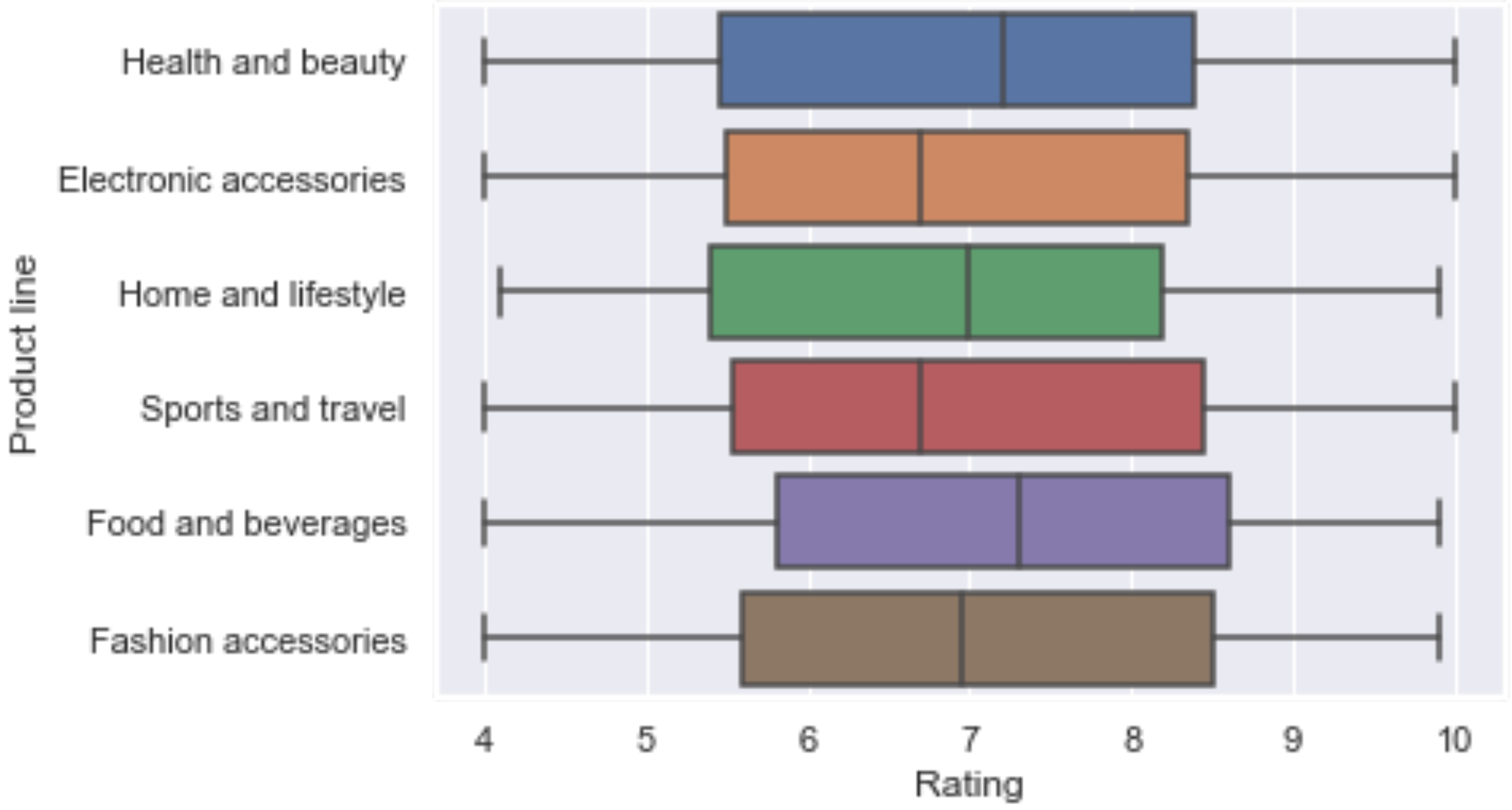
Gross income on Product

```
1 sns.boxenplot(y = 'Product line', x = 'gross income', data=sales )
```



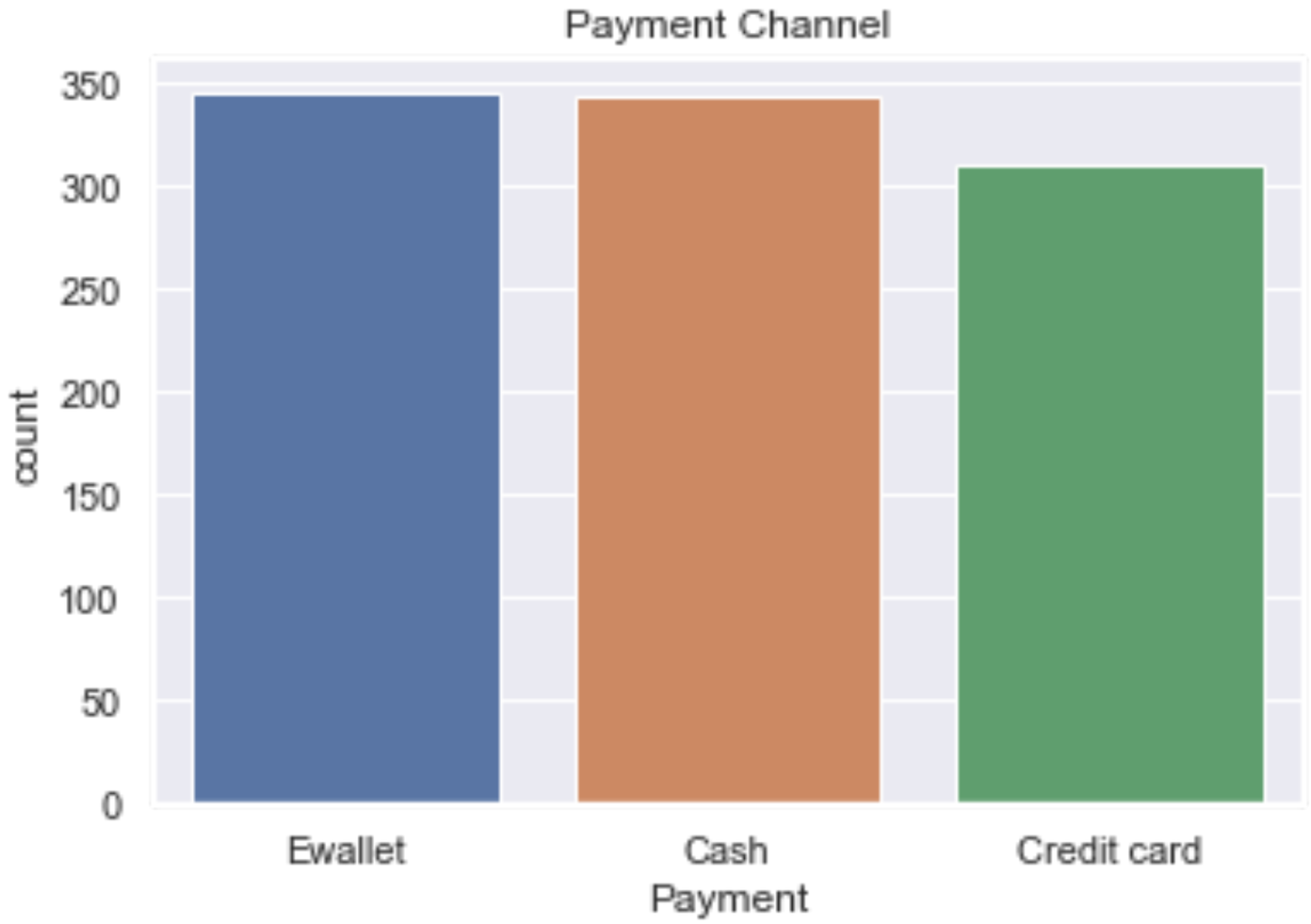
Product rating

```
1  sns.boxplot(y = 'Product line', x = 'Rating', data=sales )
```



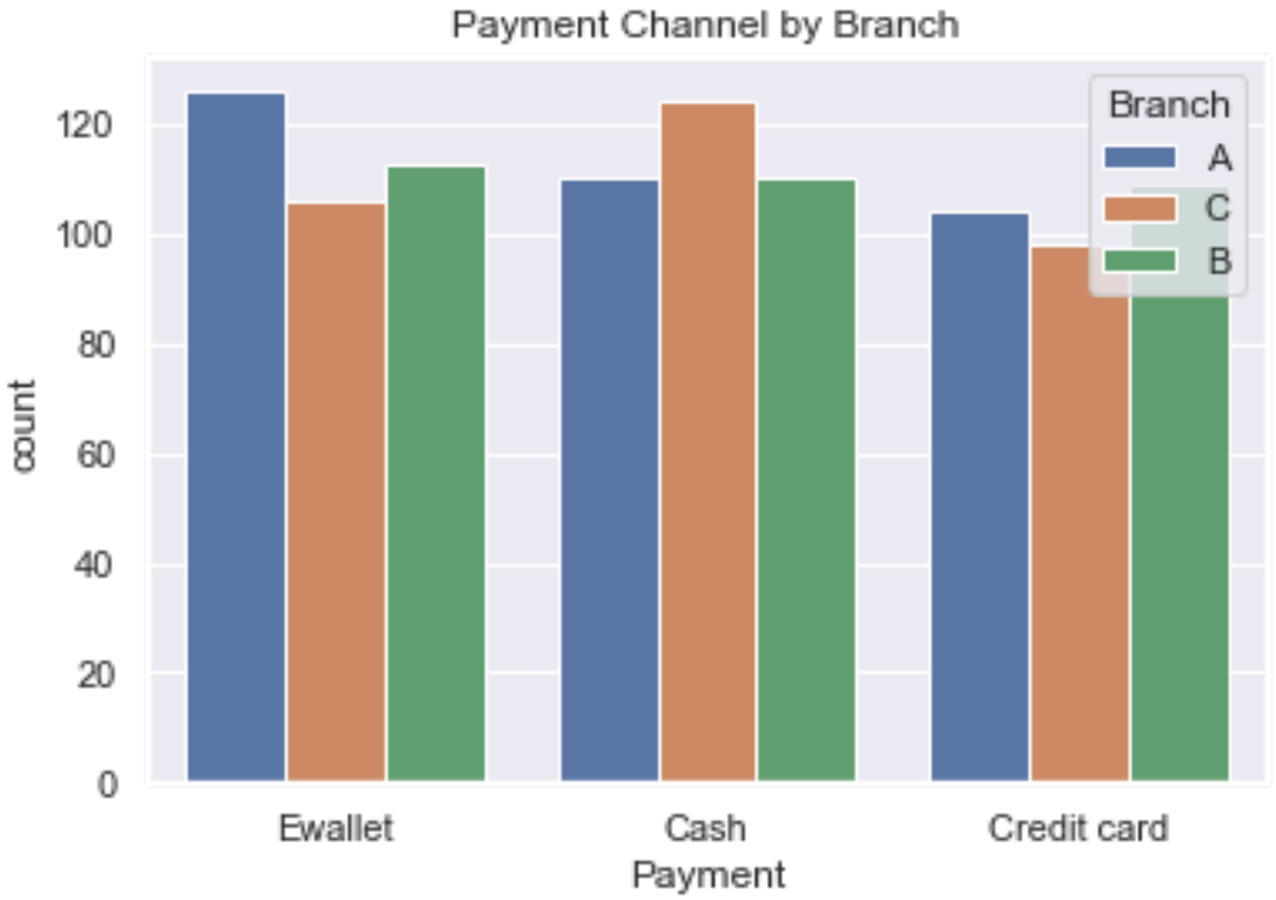
Payment channel

```
1 sns.countplot(x="Payment", data =sales).set_title("Payment Channel")
```



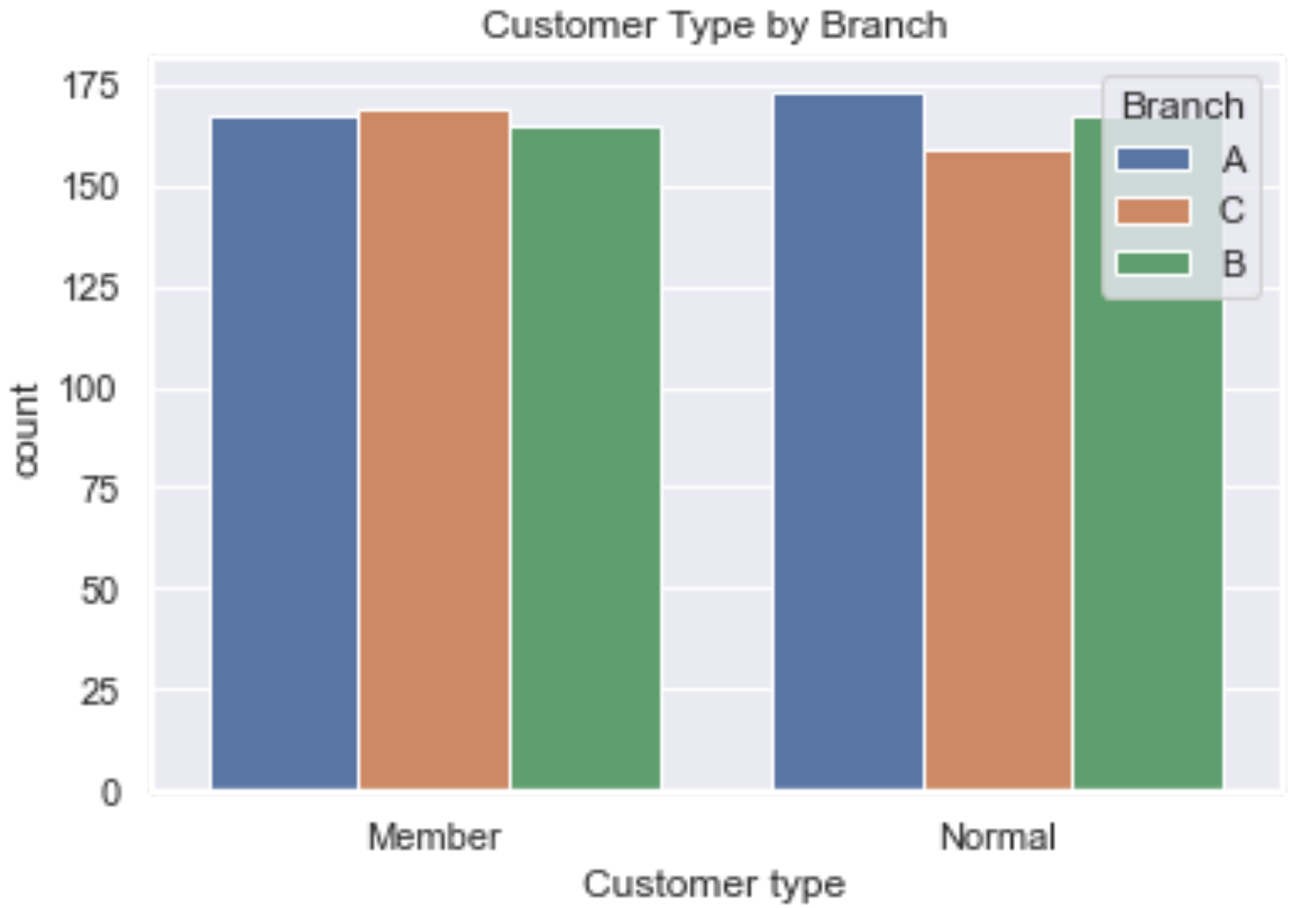
Payment channel on each branch

```
1  sns.countplot(x="Payment", hue = "Branch",
2                data =sales).set_title("Payment Channel by Branch")
```



Customer membership

```
1  sns.countplot(x="Customer type", hue = "Branch",
2                data =sales).set_title("Customer Type by Branch")
```

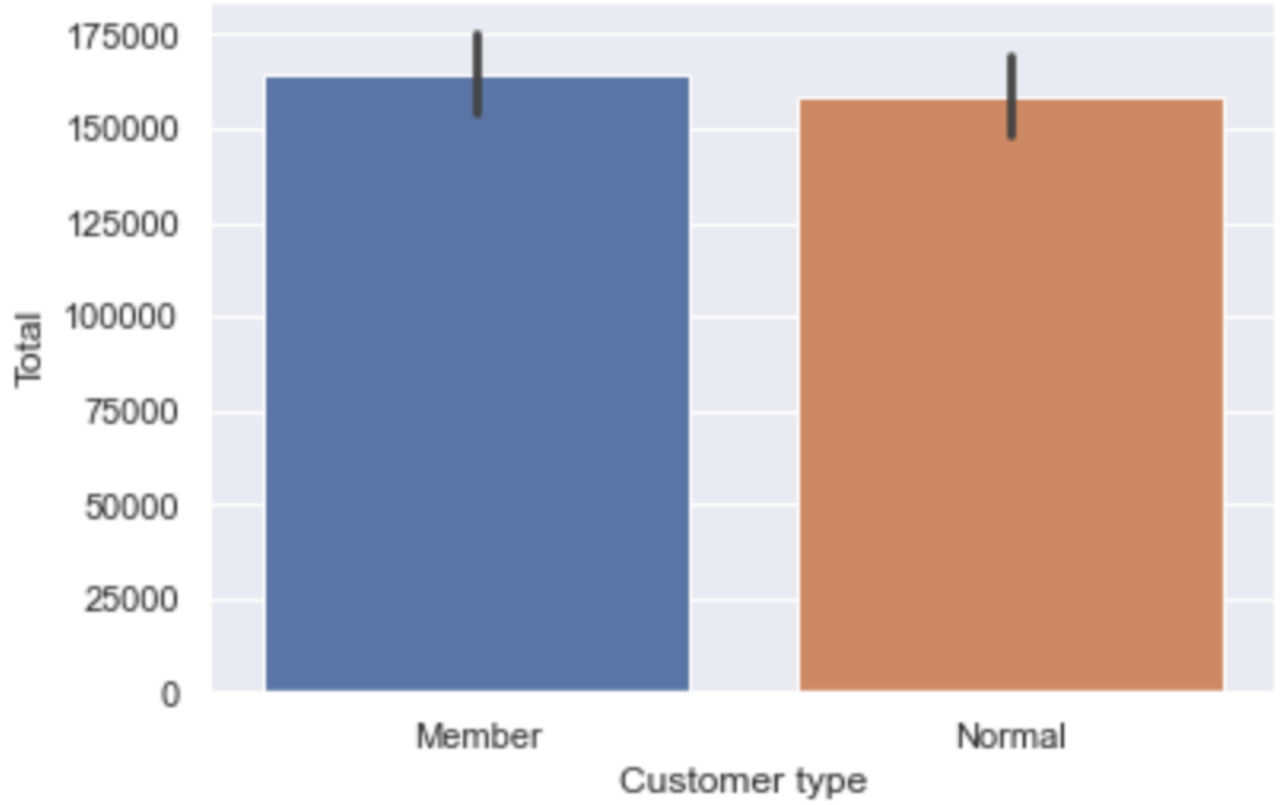


Customer in branch C is not too loyal to the supermarket?

Customer type

```
1 sns.barplot(x="Customer type", y="Total", estimator = sum, data=sales)
```

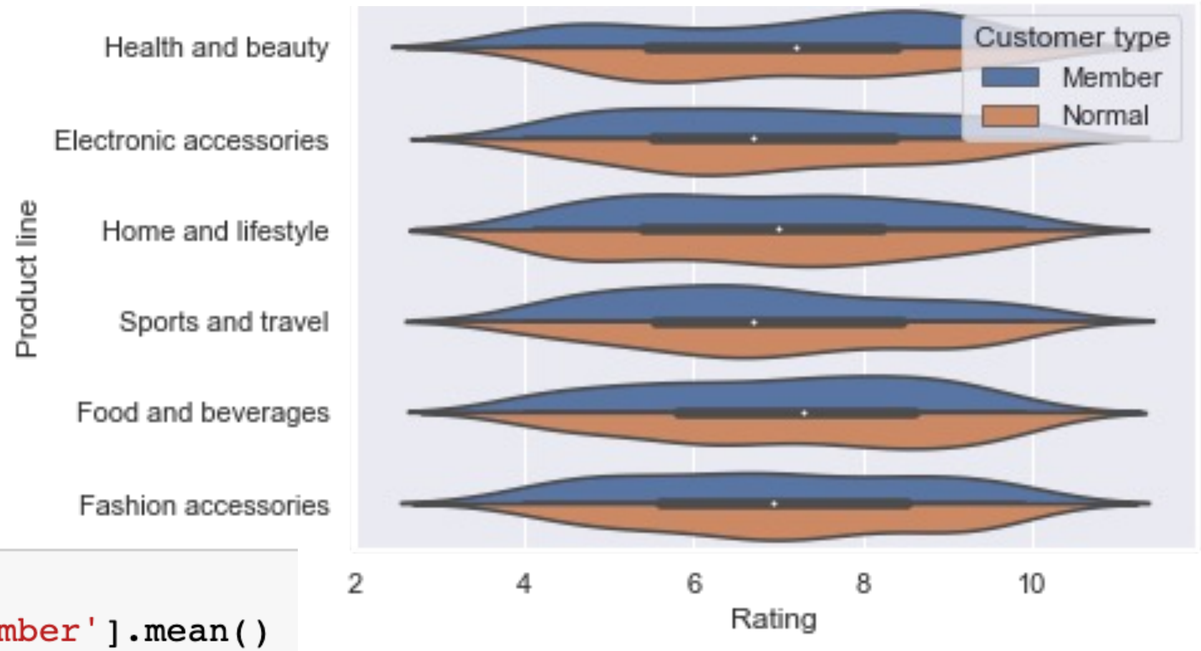
```
<AxesSubplot:xlabel='Customer type', ylabel='Total'>
```



There is plenty of room to convert Normal to Member.

Customer type influence customer rating?

```
sns.violinplot(data=sales, y = 'Product line', x = 'Rating',
               hue = 'Customer type',split=True)
```



No significant difference.

```
# Member's average rating
sales['Rating'].loc[sales['Customer type']=='Member'].mean()
```

```
6.940319361277445
```

```
# Non-Member's average rating
sales['Rating'].loc[sales['Customer type']=='Normal'].mean()
```

```
7.005210420841683
```

Customer loyalty vs Total sales

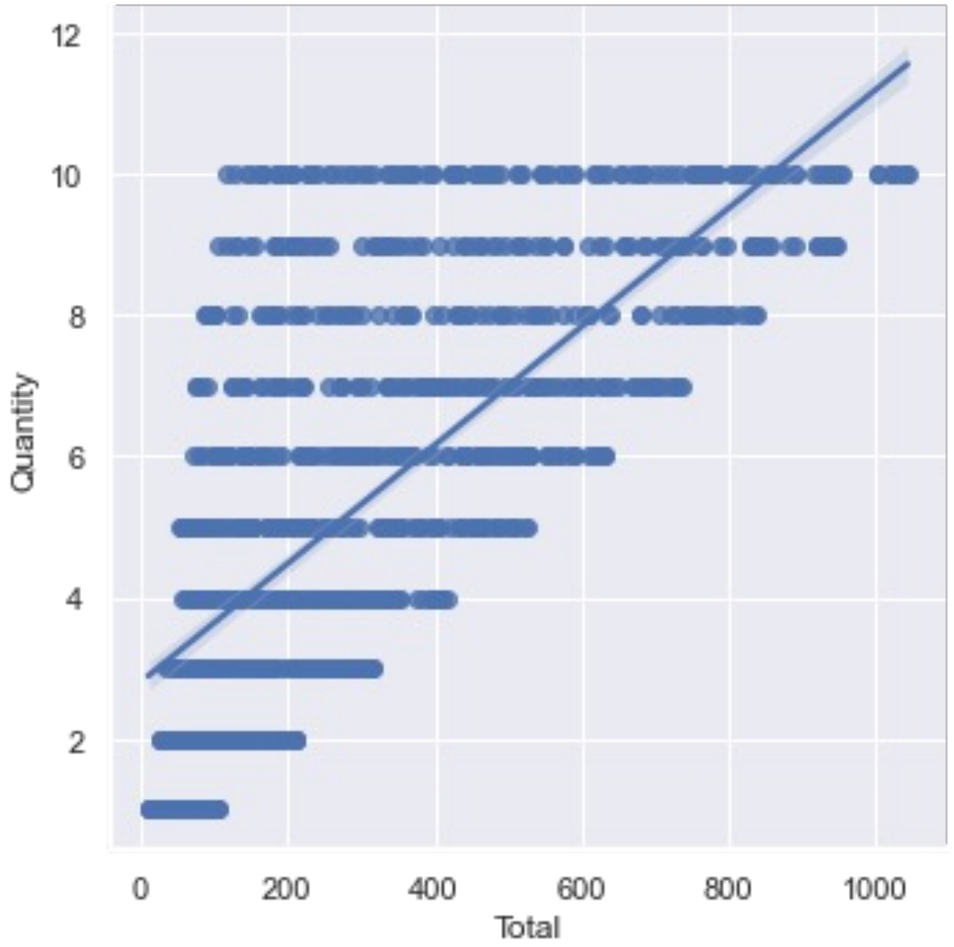
```
1 sales.groupby(['Customer type']).agg({'Total': 'sum'})
```

Customer type	Total
Member	164223.444
Normal	158743.305

What should you suggest to marketing team?

Quantity vs Total sales

```
1 sns.lmplot(x="Total",  y = "Quantity", data =sales)
```



Motivate customer to shop more in quantity, the whole sales amount will be increased as well. Bundle sales or bulk buy could be introduced?

Suggestion to management

Open discussion:

Based on the above visualization and statistics, can you suggest some recommendation to increase the branch sales?

What else information could we gather?

Suggestion to management

- Increase some of the product line?
- Customer loyalty campaign?
- Branch management?
- Timely promotion?
- VIP discount?
- VIP credit card?
- Feedback award?
- Bulk buy delivery?
- On-line shopping?
- Free gift / Sample gift?

Chapter Wrap Up

Every business is unique and you may probably keep it that way.

The key is discover the consumer preference and satisfy their needs.

As data analysts, it is important to think critically and sceptically.

Proofing the ideas with solid evidence and visualization.

Reference & Resources

Official Website:

https://plotly.com/python/

Plotly Graph Objects:

https://plotly.com/python/graph-objects/

Kaggle dataset:

https://www.kaggle.com/datasets

WorldBank API:

- https://blogs.worldbank.org/opendata/introducing-wbgapi-new-python-package-accessing-world-bank-data
- https://nbviewer.org/github/tgherzog/wbgapi/blob/master/examples/wbgapi-cookbook.ipynb
- https://pypi.org/project/wbgapi/

GitHub Open Source Code:

https://github.com/plotly/plotly.py