In [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

import os
print(os.listdir("../input"))

# Any results you write to the current directory are saved as output.
```

```
['supermarket_sales - Sheet1.csv']
```

In [2]:
```python
sales = pd.read_csv('../input/supermarket_sales - Sheet1.csv')
```

In [3]:
```python
sales.head()
```

Out[3]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax !
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.14
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.82
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.21
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.28
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.20

In [4]:
```python
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
Invoice ID                 1000 non-null object
Branch                     1000 non-null object
City                       1000 non-null object
Customer type              1000 non-null object
Gender                     1000 non-null object
Product line               1000 non-null object
Unit price                 1000 non-null float64
Quantity                   1000 non-null int64
Tax 5%                     1000 non-null float64
Total                      1000 non-null float64
Date                       1000 non-null object
Time                       1000 non-null object
Payment                    1000 non-null object
cogs                       1000 non-null float64
gross margin percentage    1000 non-null float64
gross income               1000 non-null float64
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 [5]: sales['date'] = pd.to_datetime(sales['Date'])
```

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

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

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

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

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

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

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

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

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

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

```
Out[12]: 11
```

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

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

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

Out[14]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1.000000e+03	1C
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905e+00	
std	26.494628	2.923431	11.708825	245.885335	234.17651	6.220360e-14	
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905e+00	
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905e+00	
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905e+00	
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905e+00	
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905e+00	

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

```
In [15]: categorical_columns = [cname for cname in sales.columns if sales[cname].d
```

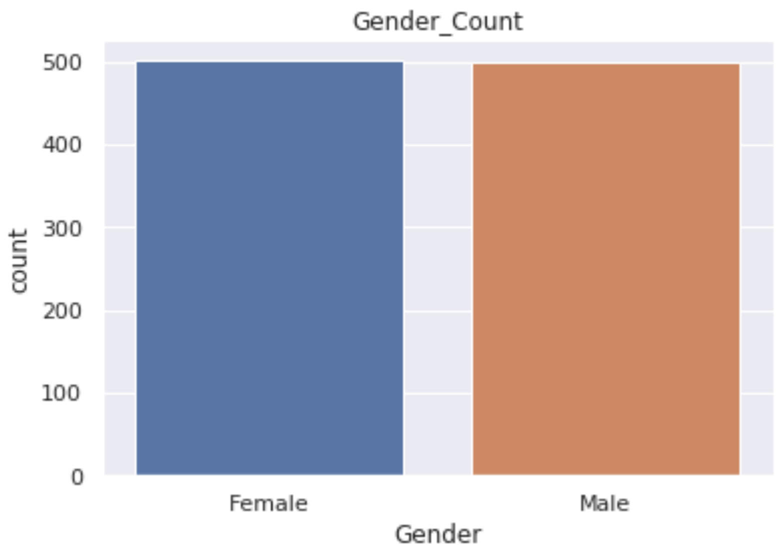
```
In [16]: categorical_columns
```

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

In [17]:
```
print("# unique values in Branch: {0}".format(len(sales['Branch'].unique(
print("# unique values in City: {0}".format(len(sales['City'].unique().to
print("# unique values in Customer Type: {0}".format(len(sales['Customer 
print("# unique values in Gender: {0}".format(len(sales['Gender'].unique(
print("# unique values in Product Line: {0}".format(len(sales['Product li
print("# unique values in Payment: {0}".format(len(sales['Payment'].uniqu
```

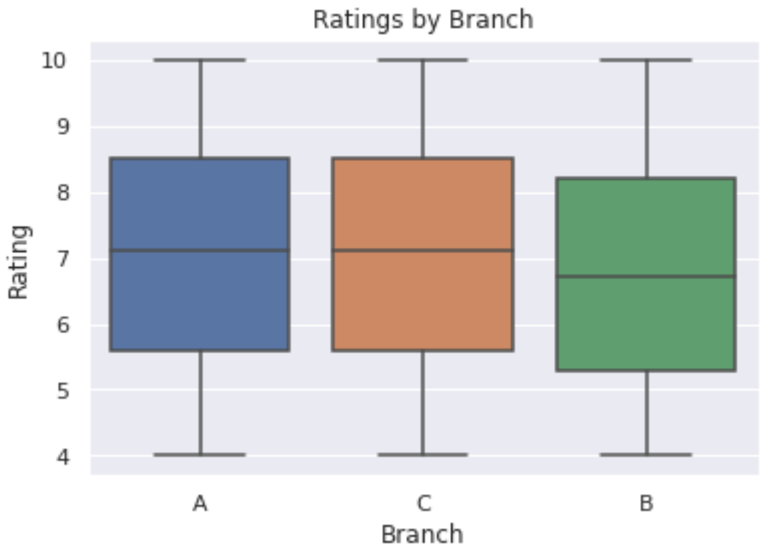
```
# unique values in Branch: 3
# unique values in City: 3
# unique values in Customer Type: 2
# unique values in Gender: 2
# unique values in Product Line: 6
# unique values in Payment: 3
```

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



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

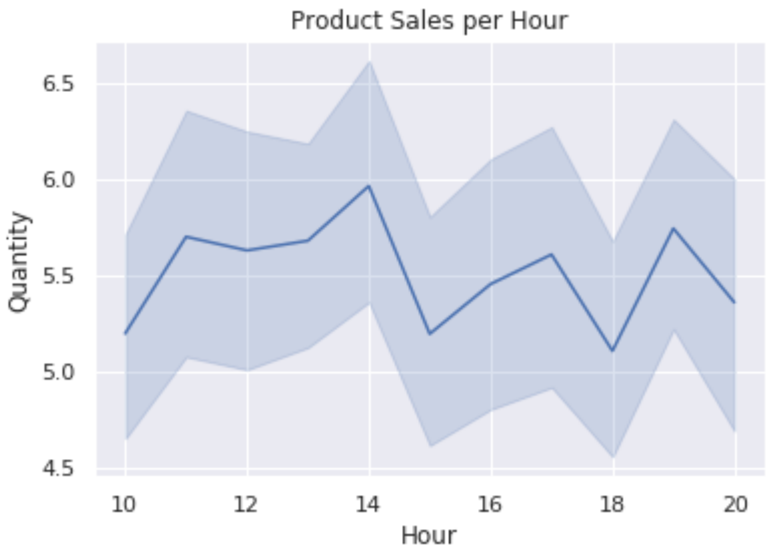
Out[19]:
```
Text(0.5, 1.0, 'Ratings by Branch')
```



Branch B has the lowest rating among all the branches

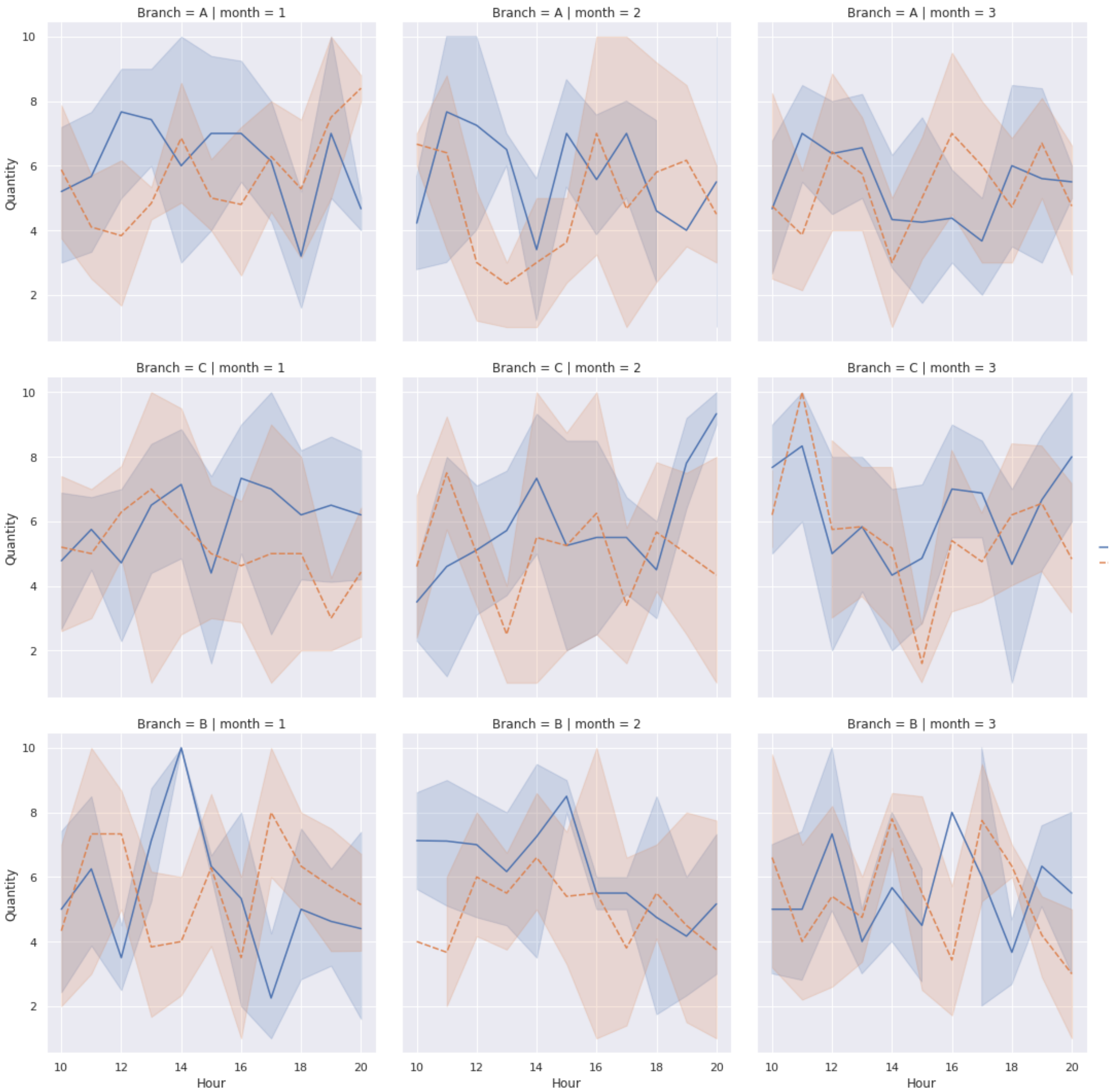
Sales by the hour in the comapny Most of the item were sold around 14:00 hrs local time

In [20]:
```
genderCount  = sns.lineplot(x="Hour",  y = 'Quantity',data =sales).set_ti
```



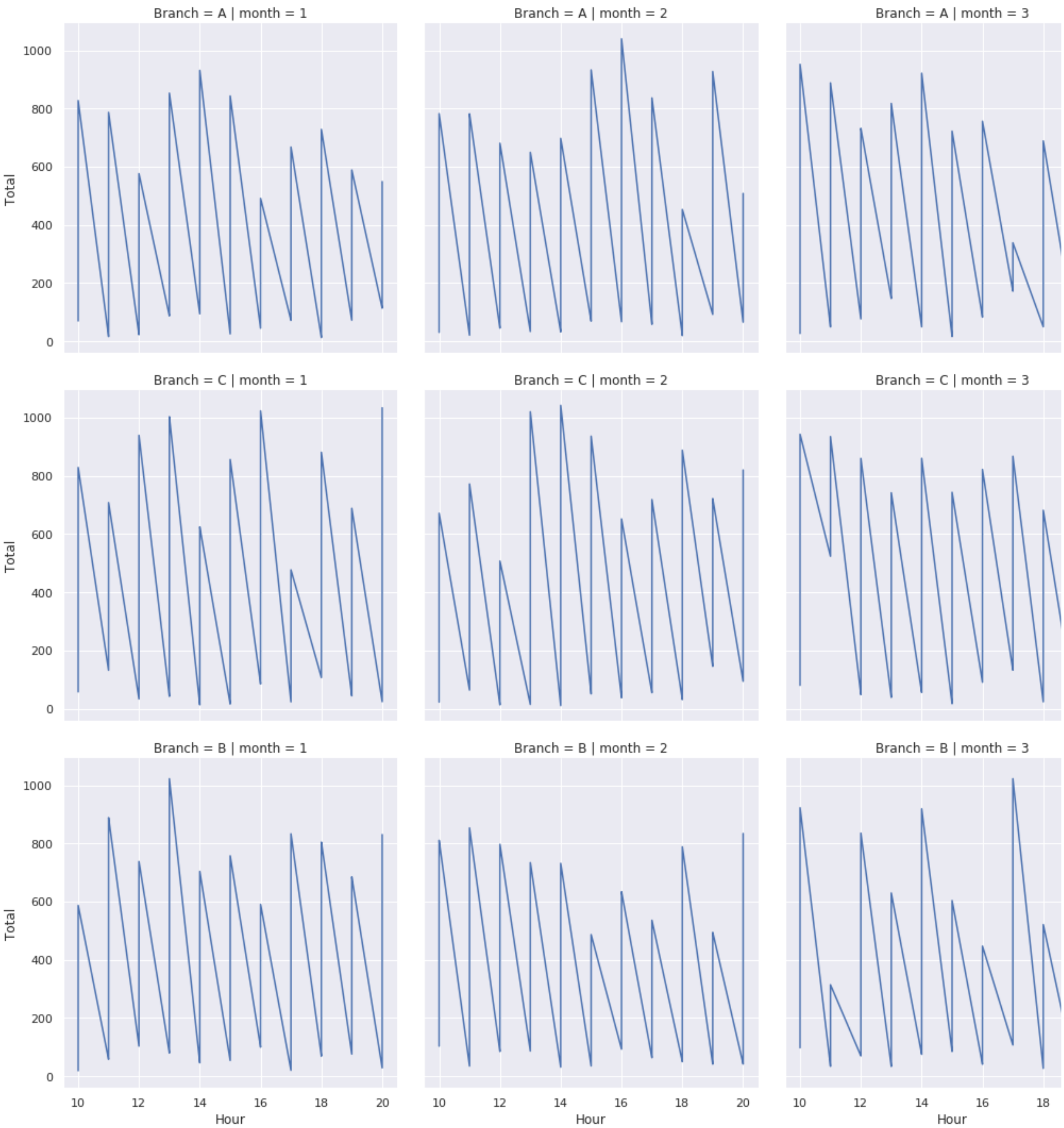
Below we can see how each branch's sales quantity looks like by the hour in a monthly fashion

In [21]:
```
genderCount  = sns.relplot(x="Hour",  y = 'Quantity', col= 'month' , row=
```



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

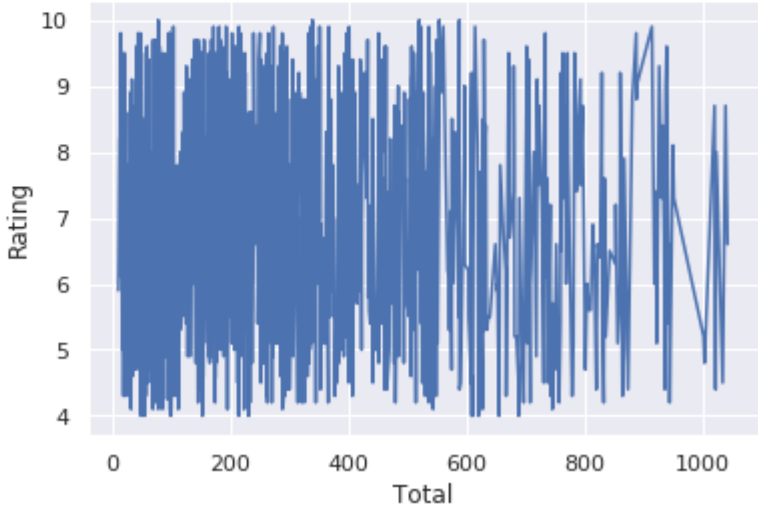
```
In [22]: genderCount  = sns.relplot(x="Hour",  y = 'Total', col= 'month' , row= 'B
```




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

```
Out[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])
```

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

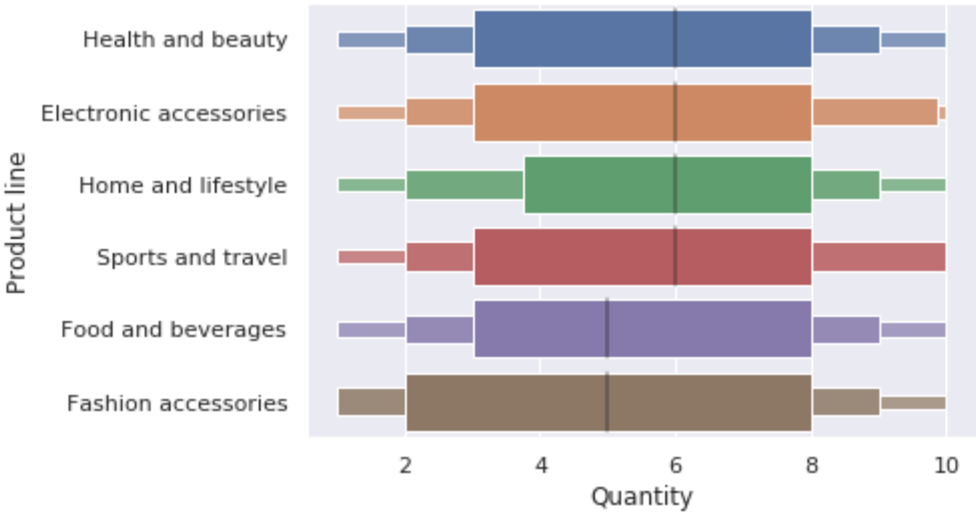


Product Analysis

Let's look at the various products' performance.

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

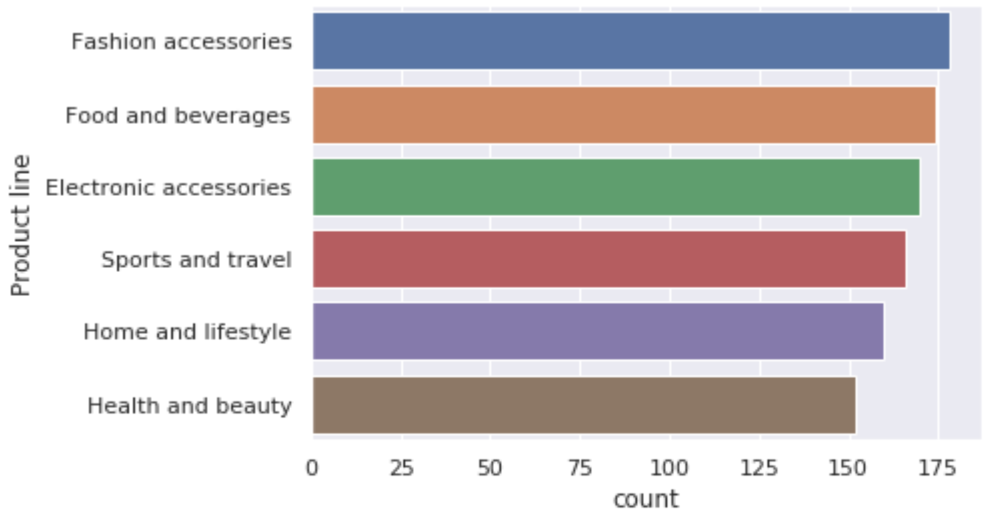
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1409d744e0>



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

In [26]:
```
sns.countplot(y = 'Product line', data=sales, order = sales['Product line
```

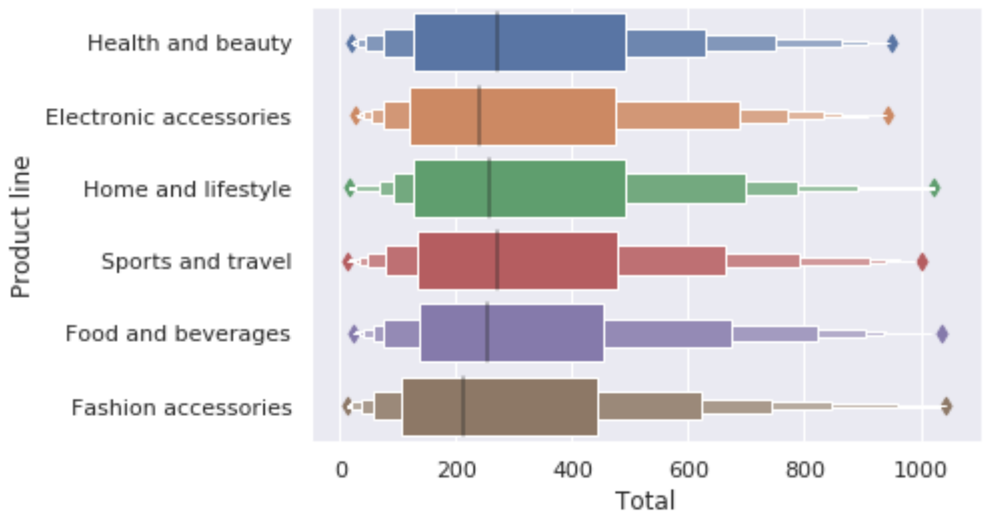
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x7f140e0beb00>



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

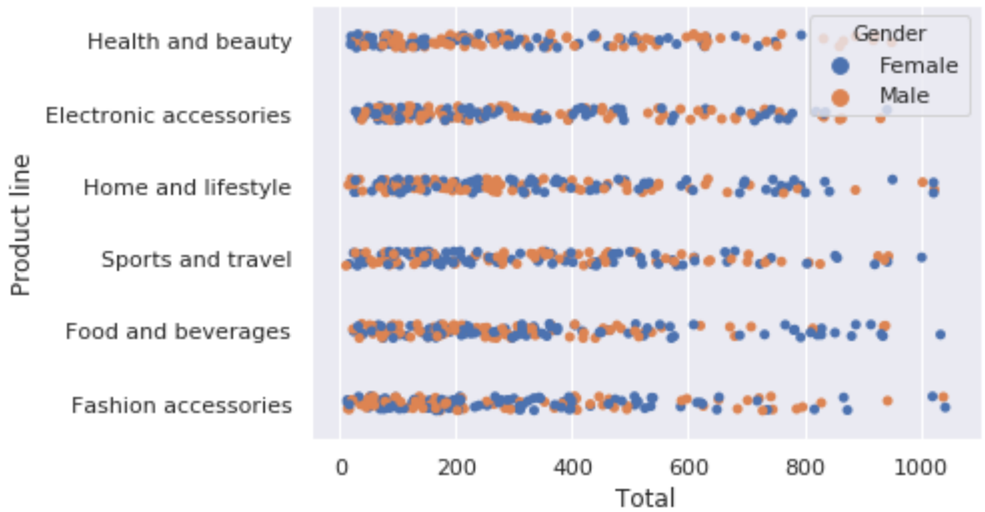
```
In [27]: sns.boxenplot(y = 'Product line', x = 'Total', data=sales )
```

```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1409c73048>
```



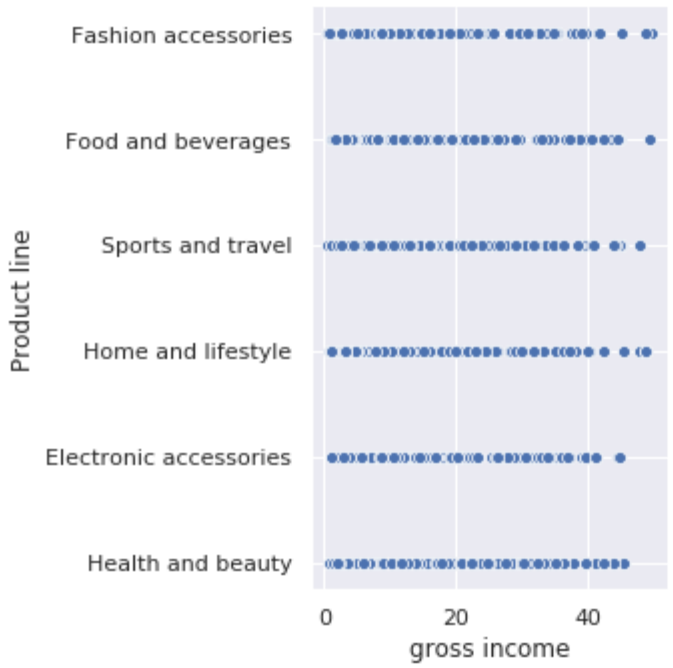

```
In [28]: sns.stripplot(y = 'Product line', x = 'Total', hue = 'Gender', data=sales
```

```
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1409bfc7f0>
```



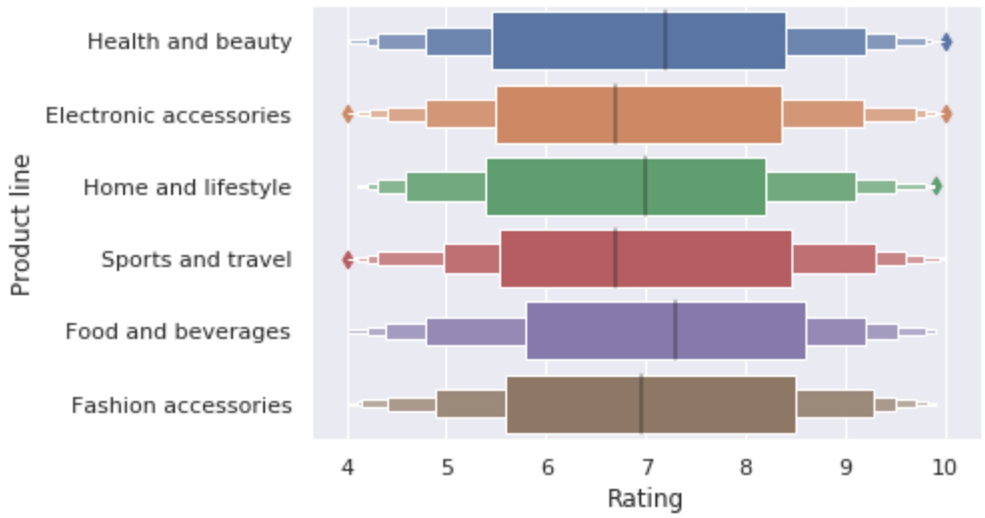

```
In [29]: sns.relplot(y = 'Product line', x = 'gross income', data=sales )
```

```
Out[29]: <seaborn.axisgrid.FacetGrid at 0x7f1409c73b00>
```




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

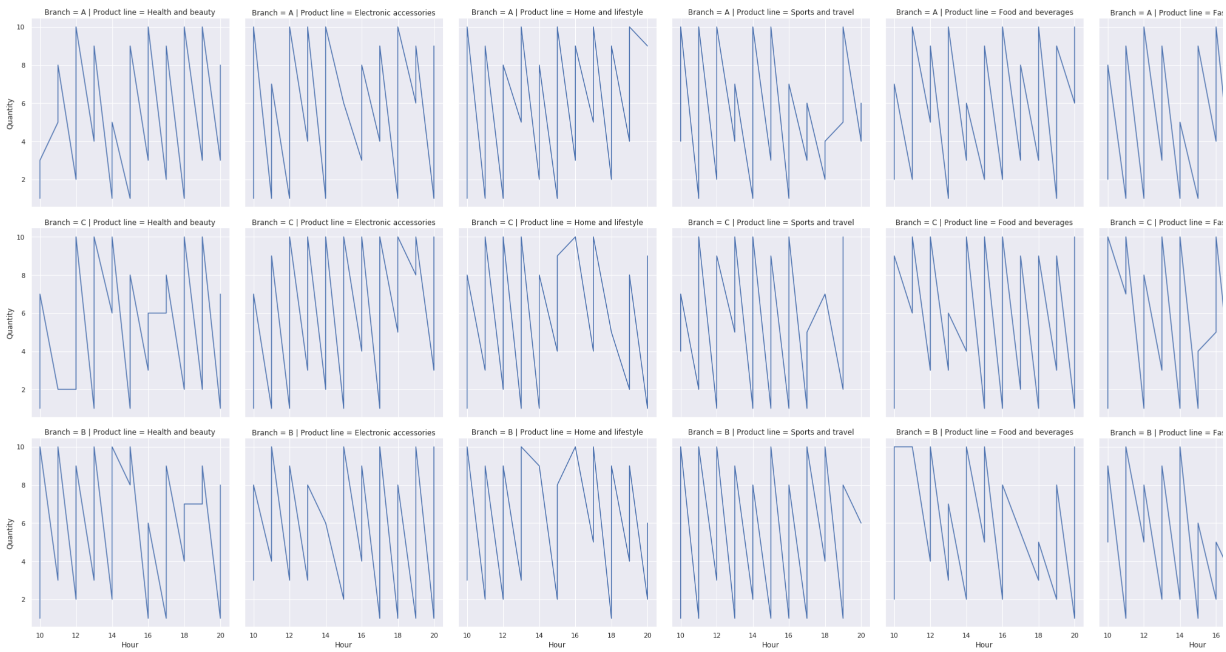
```
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1409c2d550>
```



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 [31]:
```
productCount  = sns.relplot(x="Hour",  y = 'Quantity', col= 'Product line
```



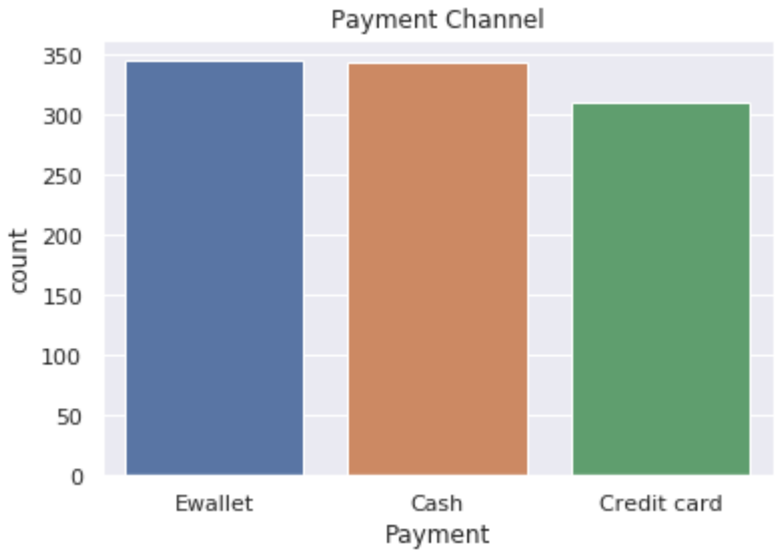
From the above plots, we can see that food and beverages sales usually high in all three branch
evening especially around 19:00

Payment Channel

Let see how customers make payment in this business

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

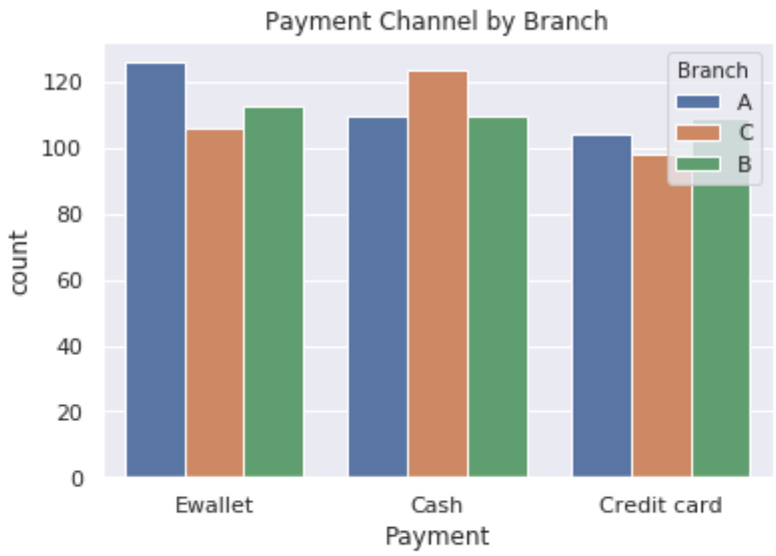
Out[32]:
```
Text(0.5, 1.0, 'Payment Channel')
```



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

In [33]:
```
sns.countplot(x="Payment", hue = "Branch", data =sales).set_title("Payme
```

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



Customer Analysis

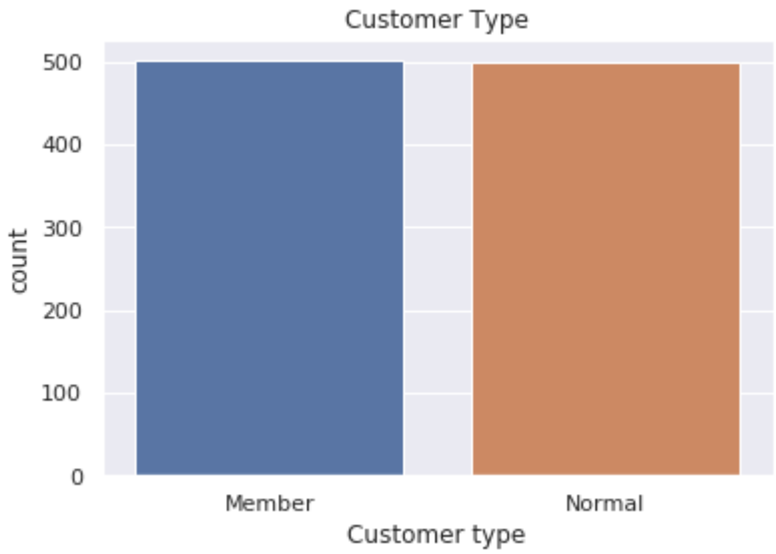
From inspection, there are two types of customers. Members and Normal. Let's see how many th
and where they are

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

Out[34]:
```
2
```

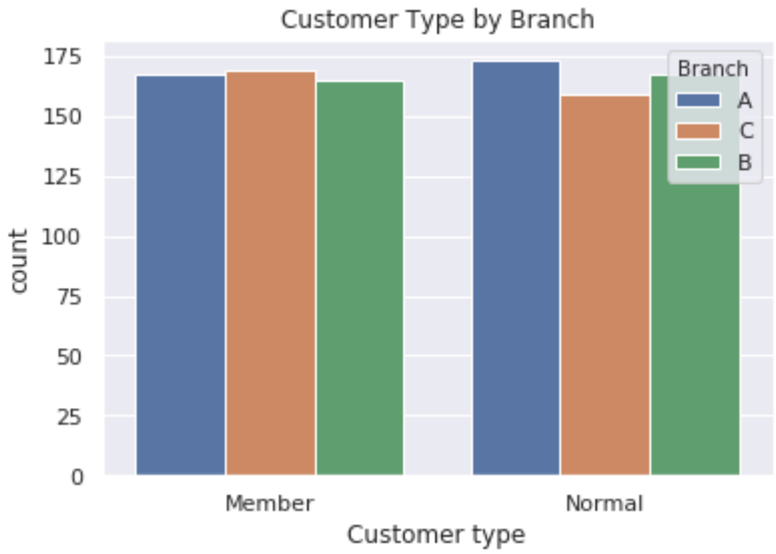
In [35]:
```
sns.countplot(x="Customer type", data =sales).set_title("Customer Type")
```

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




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

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



Does customer type influences the sales

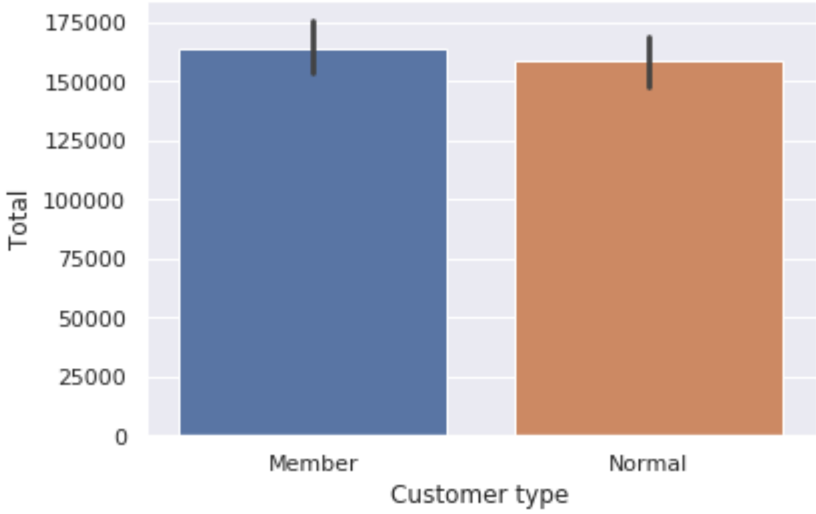
```
In [37]: sales.groupby(['Customer type']).agg({'Total': 'sum'})
```

Out[37]:

	Total
Customer type	
Member	164223.444
Normal	158743.305

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

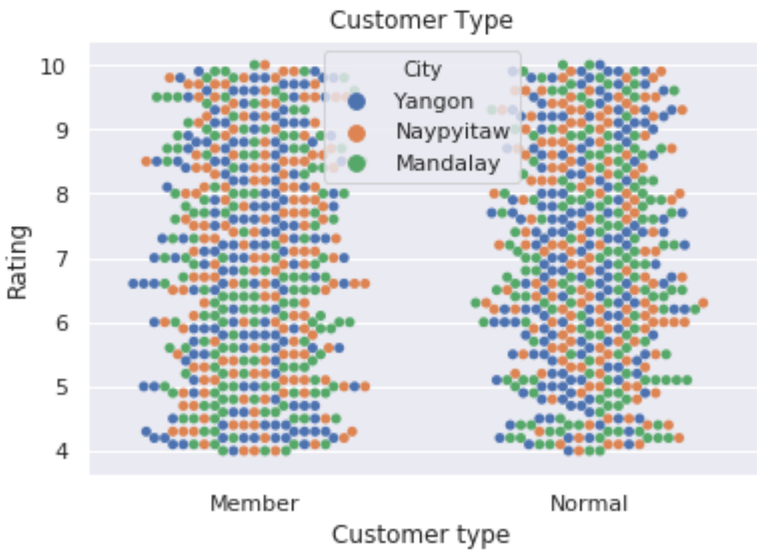
```
Out[38]:  <matplotlib.axes._subplots.AxesSubplot at 0x7f140a557550>
```



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

```
In [39]:  sns.swarmplot(x="Customer type",  y = "Rating",  hue = "City", data =sal
```

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

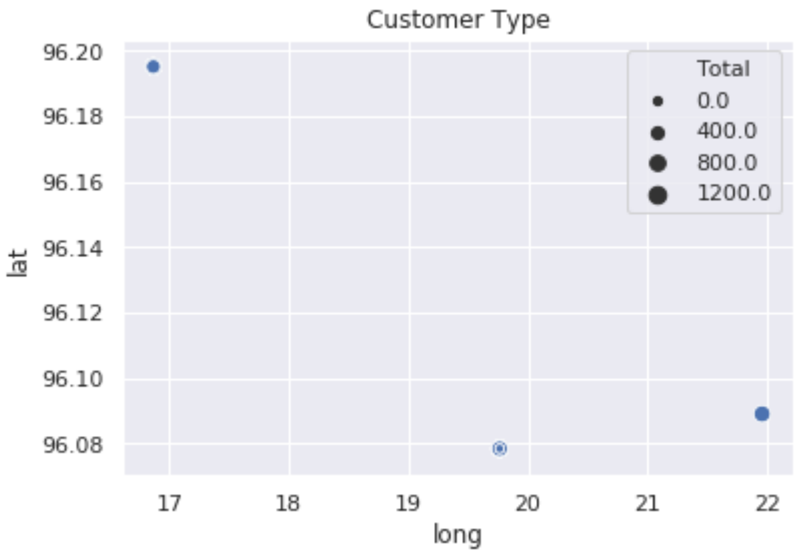


With the use of google search, I was able to get the longitude and latitude of each cities. We can

```
In [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)
```

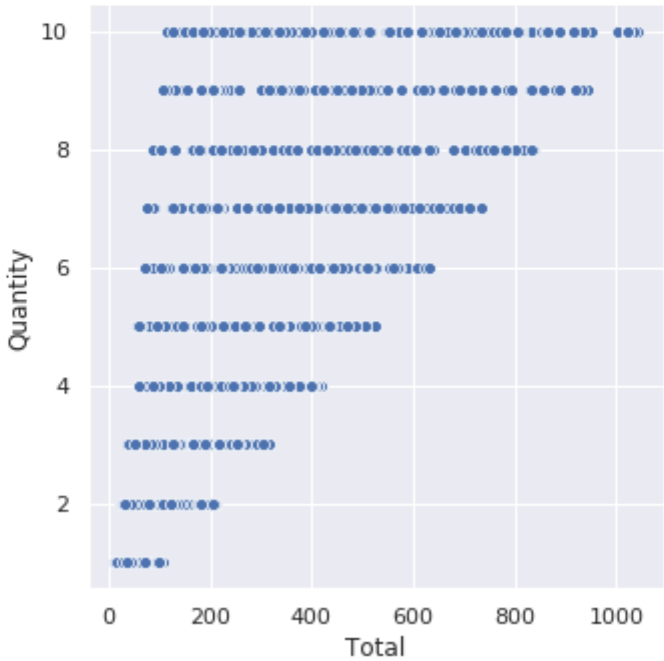
```
In [41]:  sns.scatterplot(x="long",  y = "lat",size = "Total", data =sales, legend
```

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




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

```
Out[42]: <seaborn.axisgrid.FacetGrid at 0x7f140a479160>
```




```
In [ ]:
```