Supermarket_exp

April 19, 2025

[9]:
```
import pandas as pd
import numpy as np
import matplotlib as plt
import matplotlib.pyplot as plt
import plotly.express as px # for making it interactive
from sklearn.linear_model import LinearRegression # for prediction of the next␣
 ↪month revenue
import seaborn as sns
```

[7]:
```
data0=pd.read_csv('ORIGNAL SUPERMARKET_DATA.csv')
```

[11]:
```
data0.head()
```

[11]:

	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

[12]:
```
data0.columns
```

[12]:
```
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')
```

[ ]:
```
### Added new columns to the dataset.csv to keep the orignal data unchanged
```

[12]:
```
data= pd.read_csv('dataset.csv')

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 22 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Unnamed: 0.2             1000 non-null   int64
 1   Unnamed: 0.1             1000 non-null   int64
 2   Unnamed: 0               1000 non-null   int64
 3   Invoice ID               1000 non-null   object
 4   Branch                   1000 non-null   object
 5   City                     1000 non-null   object
 6   Customer type            1000 non-null   object
 7   Gender                   1000 non-null   object
 8   Product line             1000 non-null   object
 9   Unit price               1000 non-null   float64
 10  Quantity                 1000 non-null   int64
 11  Tax 5%                   1000 non-null   float64
 12  Total                    1000 non-null   float64
 13  Time                     1000 non-null   object
 14  Payment                  1000 non-null   object
 15  cogs                     1000 non-null   float64
 16  gross margin percentage  1000 non-null   float64
 17  gross income             1000 non-null   float64
 18  Rating                   1000 non-null   float64
 19  Day                      1000 non-null   object
 20  Date(dd-mm-yyyy)         1000 non-null   object
 21  Month                    1000 non-null   object
dtypes: float64(7), int64(4), object(11)
memory usage: 172.0+ KB
```

[ ]:
```
data['Date(dd-mm-yyyy)']=pd.to_datetime(data['Date(dd-mm-yyyy)'] ,␣
 ↪format='%d-%m-%Y' )
data['Day']=data['Date(dd-mm-yyyy)'].dt.day_name()
```

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

[14]:
```
       Unnamed: 0.2  Unnamed: 0.1   Unnamed: 0   Unit price     Quantity  \
count   1000.000000   1000.000000  1000.000000  1000.000000  1000.000000
mean     499.500000    499.500000   499.500000    55.672130     5.510000
std      288.819436    288.819436   288.819436    26.494628     2.923431
min        0.000000      0.000000     0.000000    10.080000     1.000000
25%      249.750000    249.750000   249.750000    32.875000     3.000000
50%      499.500000    499.500000   499.500000    55.230000     5.000000
75%      749.250000    749.250000   749.250000    77.935000     8.000000
max      999.000000    999.000000   999.000000    99.960000    10.000000

             Tax 5%        Total         cogs  gross margin percentage  \
count  1000.000000  1000.000000  1000.00000              1000.000000
mean     15.379369   322.966749   307.58738                 4.761905
std      11.708825   245.885335   234.17651                 0.000000
min       0.508500    10.678500    10.17000                 4.761905
25%       5.924875   124.422375   118.49750                 4.761905
50%      12.088000   253.848000   241.76000                 4.761905
75%      22.445250   471.350250   448.90500                 4.761905
max      49.650000  1042.650000   993.00000                 4.761905

       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
```

[15]:
```
data.dtypes
```

[15]:
```
Unnamed: 0.2              int64
Unnamed: 0.1              int64
Unnamed: 0                int64
Invoice ID               object
Branch                   object
City                     object
Customer type            object
Gender                   object
Product line             object
Unit price              float64
Quantity                  int64
Tax 5%                  float64
Total                   float64
Time                     object
Payment                  object
```

```
cogs                        float64
gross margin percentage     float64
gross income                float64
Rating                      float64
Day                          object
Date(dd-mm-yyyy)             object
Month                        object
dtype: object
```

[16]:
```
data['Quantity'].max()
```

[16]:
```
np.int64(10)
```

[17]:
```
data['Quantity'].min()
```

[17]:
```
np.int64(1)
```

[18]:
```
data.groupby('Quantity')['Customer type'].apply(list)
```

[18]:
```
Quantity
1     [Member, Normal, Normal, Normal, Normal, Norma…
2     [Member, Normal, Normal, Member, Normal, Norma…
3     [Normal, Member, Normal, Member, Member, Norma…
4     [Member, Member, Normal, Member, Member, Membe…
5     [Normal, Normal, Member, Member, Normal, Norma…
6     [Member, Normal, Normal, Normal, Member, Membe…
7     [Member, Normal, Normal, Member, Member, Membe…
8     [Member, Member, Normal, Normal, Normal, Norma…
9     [Normal, Normal, Normal, Member, Member, Norma…
10    [Normal, Normal, Member, Member, Member, Norma…
Name: Customer type, dtype: object
```

1 HOW MANY MEMBERS in the dataset purchased the product where the Quantity is 6??

[19]:
```
data.groupby('Quantity')['Customer type'].count()
```

[19]:
```
Quantity
1     112
2      91
3      90
4     109
5     102
6      98
7     102
8      85
```

```
9      92
10    119
Name: Customer type, dtype: int64
```

[20]:
```
data.groupby('Quantity')['Customer type'].size()
```

[20]:
```
Quantity
1     112
2      91
3      90
4     109
5     102
6      98
7     102
8      85
9      92
10    119
Name: Customer type, dtype: int64
```

[21]:
```
# need to declare a avatiable which will store the grouped  values
group= data.groupby(['Quantity' , 'Customer type']).size()

group.loc[ 6 , 'Member']
```

[21]:
```
np.int64(42)
```

[22]:
```
data.groupby(['Quantity' , 'Customer type']).size()
```

[22]:
```
Quantity  Customer type
1         Member           57
          Normal           55
2         Member           43
          Normal           48
3         Member           44
          Normal           46
4         Member           61
          Normal           48
5         Member           48
          Normal           54
6         Member           42
          Normal           56
7         Member           51
          Normal           51
8         Member           41
          Normal           44
9         Member           51
          Normal           41
```

```
10        Member          63
          Normal          56
dtype: int64
```

1.1 now checking types of product and their count

[23]:
```
data.groupby(['Product line','Quantity']).size()
```

[23]:
```
Product line            Quantity
Electronic accessories  1           20
                        2            8
                        3           16
                        4           19
                        5           17
                        6           19
                        7           16
                        8           17
                        9           16
                        10          22
Fashion accessories     1           30
                        2           22
                        3           14
                        4           21
                        5           15
                        6            8
                        7           21
                        8           12
                        9           14
                        10          21
Food and beverages      1           15
                        2           16
                        3           23
                        4           18
                        5           21
                        6           17
                        7           12
                        8           15
                        9           17
                        10          20
Health and beauty       1           15
                        2           13
                        3           13
                        4           15
                        5           18
                        6           14
                        7           19
                        8           15
```

```
                        9           13
                        10          17
Home and lifestyle      1           13
                        2           14
                        3           13
                        4           22
                        5           13
                        6           21
                        7           12
                        8           17
                        9           18
                        10          17
Sports and travel       1           19
                        2           18
                        3           11
                        4           14
                        5           18
                        6           19
                        7           22
                        8            9
                        9           14
                        10          22
dtype: int64
```

1.2 no of people who brought electronic items

[24]:
```
#Storing this data
product=data.groupby(['Product line','Quantity']).size()

product.loc['Electronic accessories'].sum()
```

[24]:
```
np.int64(170)
```

2 SORTING DATA ACCORDING TO THE MONTH

[25]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Unnamed: 0.2            1000 non-null   int64
 1   Unnamed: 0.1            1000 non-null   int64
 2   Unnamed: 0              1000 non-null   int64
 3   Invoice ID              1000 non-null   object
 4   Branch                  1000 non-null   object
```

```
 5   City                      1000 non-null   object
 6   Customer type             1000 non-null   object
 7   Gender                    1000 non-null   object
 8   Product line              1000 non-null   object
 9   Unit price                1000 non-null   float64
 10  Quantity                  1000 non-null   int64
 11  Tax 5%                    1000 non-null   float64
 12  Total                     1000 non-null   float64
 13  Time                      1000 non-null   object
 14  Payment                   1000 non-null   object
 15  cogs                      1000 non-null   float64
 16  gross margin percentage   1000 non-null   float64
 17  gross income              1000 non-null   float64
 18  Rating                    1000 non-null   float64
 19  Day                       1000 non-null   object
 20  Date(dd-mm-yyyy)          1000 non-null   object
 21  Month                     1000 non-null   object
dtypes: float64(7), int64(4), object(11)
memory usage: 172.0+ KB
```

[26]:
```
data['Date(dd-mm-yyyy)'].head()
```

[26]:
```
0      5/1/2019
1      3/3/2019
2    27-01-2019
3      8/2/2019
4    25-02-2019
Name: Date(dd-mm-yyyy), dtype: object
```

[ ]:
```
#making a new column for the date format which stores the correct format
#https://docs.python.org/3/library/datetime.html#strftime-and-strptime-behavior
#for using dt we need to conver t the column into date time

data['Date']=pd.to_datetime(data['Date'])


# created a new column as Date(dd-mm-yyyy)
data['Date(dd-mm-yyyy)']=data['Date'].dt.strftime('%d-%m-%Y')
```

2.1 need to convert this date format which is in YYYY-MM-DD to DD-MM-YYY

[27]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 22 columns):
 #   Column                    Non-Null Count  Dtype
```

```
---  ------                   --------------  -----
 0   Unnamed: 0.2             1000 non-null   int64
 1   Unnamed: 0.1             1000 non-null   int64
 2   Unnamed: 0               1000 non-null   int64
 3   Invoice ID               1000 non-null   object
 4   Branch                   1000 non-null   object
 5   City                     1000 non-null   object
 6   Customer type            1000 non-null   object
 7   Gender                   1000 non-null   object
 8   Product line             1000 non-null   object
 9   Unit price               1000 non-null   float64
 10  Quantity                 1000 non-null   int64
 11  Tax 5%                   1000 non-null   float64
 12  Total                    1000 non-null   float64
 13  Time                     1000 non-null   object
 14  Payment                  1000 non-null   object
 15  cogs                     1000 non-null   float64
 16  gross margin percentage  1000 non-null   float64
 17  gross income             1000 non-null   float64
 18  Rating                   1000 non-null   float64
 19  Day                      1000 non-null   object
 20  Date(dd-mm-yyyy)         1000 non-null   object
 21  Month                    1000 non-null   object
dtypes: float64(7), int64(4), object(11)
memory usage: 172.0+ KB
```

[28]:
```
#ok so new column is added

data.head(20)
```

[28]:

	Unnamed: 0.2	Unnamed: 0.1	Unnamed: 0	Invoice ID	Branch	City
0	0	0	0	750-67-8428	A	Yangon
1	1	1	1	631-41-3108	A	Yangon
2	2	2	2	123-19-1176	A	Yangon
3	3	3	3	373-73-7910	A	Yangon
4	4	4	4	355-53-5943	A	Yangon
5	5	5	5	665-32-9167	A	Yangon
6	6	6	6	365-64-0515	A	Yangon
7	7	7	7	252-56-2699	A	Yangon
8	8	8	8	829-34-3910	A	Yangon
9	9	9	9	656-95-9349	A	Yangon
10	10	10	10	765-26-6951	A	Yangon
11	11	11	11	329-62-1586	A	Yangon
12	12	12	12	636-48-8204	A	Yangon
13	13	13	13	549-59-1358	A	Yangon
14	14	14	14	227-03-5010	A	Yangon
15	15	15	15	189-17-4241	A	Yangon

16	16	16	16	848-62-7243	A	Yangon
17	17	17	17	595-11-5460	A	Yangon
18	18	18	18	129-29-8530	A	Yangon
19	19	19	19	272-65-1806	A	Yangon

	Customer type	Gender	Product line	Unit price	…	Total	\
0	Member	Female	Health and beauty	74.69	…	548.9715	
1	Normal	Male	Home and lifestyle	46.33	…	340.5255	
2	Member	Male	Health and beauty	58.22	…	489.0480	
3	Normal	Male	Sports and travel	86.31	…	634.3785	
4	Member	Female	Electronic accessories	68.84	…	433.6920	
5	Member	Female	Health and beauty	36.26	…	76.1460	
6	Normal	Female	Electronic accessories	46.95	…	246.4875	
7	Normal	Male	Food and beverages	43.19	…	453.4950	
8	Normal	Female	Health and beauty	71.38	…	749.4900	
9	Member	Female	Health and beauty	68.93	…	506.6355	
10	Normal	Male	Sports and travel	72.61	…	457.4430	
11	Normal	Male	Food and beverages	54.67	…	172.2105	
12	Normal	Male	Electronic accessories	34.56	…	181.4400	
13	Member	Male	Sports and travel	88.63	…	279.1845	
14	Member	Female	Home and lifestyle	52.59	…	441.7560	
15	Normal	Female	Fashion accessories	87.67	…	184.1070	
16	Normal	Male	Health and beauty	24.89	…	235.2105	
17	Normal	Male	Health and beauty	96.58	…	202.8180	
18	Member	Male	Sports and travel	62.62	…	328.7550	
19	Normal	Female	Electronic accessories	60.88	…	575.3160	

	Time	Payment	cogs	gross margin percentage	gross income	\
0	13:08:00	Ewallet	522.83	4.761905	26.1415	
1	13:23:00	Credit card	324.31	4.761905	16.2155	
2	20:33:00	Ewallet	465.76	4.761905	23.2880	
3	10:37:00	Ewallet	604.17	4.761905	30.2085	
4	14:36:00	Ewallet	413.04	4.761905	20.6520	
5	17:15:00	Credit card	72.52	4.761905	3.6260	
6	10:25:00	Ewallet	234.75	4.761905	11.7375	
7	16:48:00	Ewallet	431.90	4.761905	21.5950	
8	19:21:00	Cash	713.80	4.761905	35.6900	
9	11:03:00	Credit card	482.51	4.761905	24.1255	
10	10:39:00	Credit card	435.66	4.761905	21.7830	
11	18:00:00	Credit card	164.01	4.761905	8.2005	
12	11:15:00	Ewallet	172.80	4.761905	8.6400	
13	17:36:00	Ewallet	265.89	4.761905	13.2945	
14	19:20:00	Credit card	420.72	4.761905	21.0360	
15	12:17:00	Credit card	175.34	4.761905	8.7670	
16	15:36:00	Cash	224.01	4.761905	11.2005	
17	10:12:00	Credit card	193.16	4.761905	9.6580	
18	19:15:00	Ewallet	313.10	4.761905	15.6550	

```
19  17:17:00      Ewallet  547.92               4.761905       27.3960

    Rating       Day  Date(dd-mm-yyyy)     Month
0      9.1  Saturday          5/1/2019   January
1      7.4    Sunday          3/3/2019     March
2      8.4    Sunday        27-01-2019   January
3      5.3    Friday          8/2/2019  February
4      5.8    Monday        25-02-2019  February
5      7.2  Thursday         10/1/2019   January
6      7.1   Tuesday         12/2/2019  February
7      8.2  Thursday          7/2/2019  February
8      5.7    Friday        29-03-2019     March
9      4.6    Monday         11/3/2019     March
10     6.9   Tuesday          1/1/2019   January
11     8.6    Monday        21-01-2019   January
12     9.9    Sunday        17-02-2019  February
13     6.0  Saturday          2/3/2019     March
14     8.5    Friday        22-03-2019     March
15     7.7    Sunday         10/3/2019     March
16     7.4    Friday        15-03-2019     March
17     5.1    Friday        15-03-2019     March
18     7.0    Sunday         10/3/2019     March
19     4.7   Tuesday        15-01-2019   January

[20 rows x 22 columns]
```

[29]:
```
#ok so now we have thwcorrect formatted data so we don't need the old column so␣
 ↪we are gonna dele that

data.head()
```

[29]:
```
   Unnamed: 0.2  Unnamed: 0.1  Unnamed: 0   Invoice ID Branch    City  \
0             0             0           0  750-67-8428      A  Yangon
1             1             1           1  631-41-3108      A  Yangon
2             2             2           2  123-19-1176      A  Yangon
3             3             3           3  373-73-7910      A  Yangon
4             4             4           4  355-53-5943      A  Yangon

  Customer type  Gender            Product line  Unit price  …     Total  \
0        Member  Female       Health and beauty       74.69  …  548.9715
1        Normal    Male     Home and lifestyle       46.33  …  340.5255
2        Member    Male       Health and beauty       58.22  …  489.0480
3        Normal    Male       Sports and travel       86.31  …  634.3785
4        Member  Female  Electronic accessories       68.84  …  433.6920

       Time      Payment    cogs gross margin percentage  gross income  \
0  13:08:00      Ewallet  522.83               4.761905       26.1415
```

```
1  13:23:00  Credit card  324.31                4.761905       16.2155
2  20:33:00      Ewallet  465.76                4.761905       23.2880
3  10:37:00      Ewallet  604.17                4.761905       30.2085
4  14:36:00      Ewallet  413.04                4.761905       20.6520

   Rating       Day  Date(dd-mm-yyyy)     Month
0     9.1  Saturday          5/1/2019   January
1     7.4    Sunday          3/3/2019     March
2     8.4    Sunday        27-01-2019   January
3     5.3    Friday          8/2/2019  February
4     5.8    Monday        25-02-2019  February

[5 rows x 22 columns]
```

[ ]:
```
#now sorting according to month
#need to create a column called month
data['Date(dd-mm-yyyy)']=pd.to_datetime(data['Date(dd-mm-yyyy)'],␣
 ↪format='%d-%m-%Y')


data['Month']=data['Date(dd-mm-yyyy)'].dt.month_name()
```

[ ]:
```
# Pandas displays datetime objects in the format YYYY-MM-DD by default.

#to to display the date in dd-mm-yyyy format,  just need to  format it like␣
 ↪this:


#  Useing strftime() if you want the format to look nice for display or export.

#  Don't use it before extracting month/day, because then it becomes a string␣
 ↪again

data['Date(dd-mm-yyyy)']=data['Date(dd-mm-yyyy)'].dt.strftime('%d-%m-%Y')
```

[31]:
```
data.head(5)
```

[31]:
```
   Unnamed: 0.2  Unnamed: 0.1  Unnamed: 0   Invoice ID Branch    City  \
0             0             0           0  750-67-8428      A  Yangon
1             1             1           1  631-41-3108      A  Yangon
2             2             2           2  123-19-1176      A  Yangon
3             3             3           3  373-73-7910      A  Yangon
4             4             4           4  355-53-5943      A  Yangon

  Customer type  Gender            Product line  Unit price  …     Total  \
0        Member  Female       Health and beauty       74.69  …  548.9715
1        Normal    Male  Home and lifestyle       46.33  …  340.5255
```

```
2        Member    Male       Health and beauty       58.22  …  489.0480
3        Normal    Male       Sports and travel       86.31  …  634.3785
4        Member  Female  Electronic accessories       68.84  …  433.6920

       Time      Payment    cogs gross margin percentage  gross income  \
0  13:08:00      Ewallet  522.83                4.761905       26.1415
1  13:23:00  Credit card  324.31                4.761905       16.2155
2  20:33:00      Ewallet  465.76                4.761905       23.2880
3  10:37:00      Ewallet  604.17                4.761905       30.2085
4  14:36:00      Ewallet  413.04                4.761905       20.6520

   Rating       Day  Date(dd-mm-yyyy)     Month
0     9.1  Saturday          5/1/2019   January
1     7.4    Sunday          3/3/2019     March
2     8.4    Sunday        27-01-2019   January
3     5.3    Friday          8/2/2019  February
4     5.8    Monday        25-02-2019  February

[5 rows x 22 columns]
```

[32]:
```
#taking the count of sales according to month
#month order is in alphabetical order
data.groupby(['Month' , 'Product line']).size()
```

[32]:
```
Month     Product line
February  Electronic accessories    54
          Fashion accessories       60
          Food and beverages        62
          Health and beauty         46
          Home and lifestyle        38
          Sports and travel         43
January   Electronic accessories    54
          Fashion accessories       64
          Food and beverages        56
          Health and beauty         49
          Home and lifestyle        59
          Sports and travel         70
March     Electronic accessories    62
          Fashion accessories       54
          Food and beverages        56
          Health and beauty         57
          Home and lifestyle        63
          Sports and travel         53
dtype: int64
```

[33]:
```
data['Month'].count
```

[33]:
```
<bound method Series.count of 0       January
1         March
2       January
3      February
4      February
         …
995     January
996     January
997       March
998    February
999     January
Name: Month, Length: 1000, dtype: object>
```

[34]:
```
data['Month'].nunique
```

[34]:
```
<bound method IndexOpsMixin.nunique of 0       January
1         March
2       January
3      February
4      February
         …
995     January
996     January
997       March
998    February
999     January
Name: Month, Length: 1000, dtype: object>
```

[35]:
```
data['Month'].nunique()
```

[35]:
```
3
```

[36]:
```
data.groupby([data['Month']=='January' , data['Product line'] == 'Sports and␣
 ↪travel'])
```

[36]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x00000209B0DDD350>
```

3 saving the file

[38]:
```
data.to_csv('dataset.csv' ,  index='false')
```

[39]:
```
data.head()
```

[39]:
```
   Unnamed: 0.2  Unnamed: 0.1  Unnamed: 0   Invoice ID Branch    City  \
0             0             0           0  750-67-8428      A  Yangon
1             1             1           1  631-41-3108      A  Yangon
2             2             2           2  123-19-1176      A  Yangon
```

```
3             3             3           3  373-73-7910      A  Yangon
4             4             4           4  355-53-5943      A  Yangon

  Customer type  Gender            Product line  Unit price  …     Total  \
0        Member  Female       Health and beauty       74.69  …  548.9715
1        Normal    Male     Home and lifestyle       46.33  …  340.5255
2        Member    Male       Health and beauty       58.22  …  489.0480
3        Normal    Male       Sports and travel       86.31  …  634.3785
4        Member  Female  Electronic accessories       68.84  …  433.6920

       Time      Payment    cogs gross margin percentage  gross income  \
0  13:08:00      Ewallet  522.83                4.761905       26.1415
1  13:23:00  Credit card  324.31                4.761905       16.2155
2  20:33:00      Ewallet  465.76                4.761905       23.2880
3  10:37:00      Ewallet  604.17                4.761905       30.2085
4  14:36:00      Ewallet  413.04                4.761905       20.6520

   Rating       Day  Date(dd-mm-yyyy)     Month
0     9.1  Saturday          5/1/2019   January
1     7.4    Sunday          3/3/2019     March
2     8.4    Sunday        27-01-2019   January
3     5.3    Friday          8/2/2019  February
4     5.8    Monday        25-02-2019  February

[5 rows x 22 columns]
```

[40]:
```
data['Product line'].unique()
```

[40]:
```
array(['Health and beauty', 'Home and lifestyle', 'Sports and travel',
       'Electronic accessories', 'Food and beverages',
       'Fashion accessories'], dtype=object)
```

[41]:
```
data['Product line'].nunique()
```

[41]:
```
6
```

[42]:
```
data.groupby(['Month' , 'Product line']).size()
```

[42]:
```
Month     Product line
February  Electronic accessories    54
          Fashion accessories       60
          Food and beverages        62
          Health and beauty         46
          Home and lifestyle        38
          Sports and travel         43
January   Electronic accessories    54
          Fashion accessories       64
```

```
           Food and beverages        56
           Health and beauty         49
           Home and lifestyle        59
           Sports and travel         70
March      Electronic accessories    62
           Fashion accessories       54
           Food and beverages        56
           Health and beauty         57
           Home and lifestyle        63
           Sports and travel         53
dtype: int64
```

[43]:
```
data.groupby(['Month' , 'Product line' , 'Quantity']).size()
```

[43]:
```
Month     Product line            Quantity
February  Electronic accessories  1           10
                                  2            2
                                  3            3
                                  4            2
                                  5            5
                                               ..
March     Sports and travel       6            6
                                  7            7
                                  8            4
                                  9            5
                                  10           8
Length: 180, dtype: int64
```

[44]:
```
data.head(10)
```

[44]:

	Unnamed: 0.2	Unnamed: 0.1	Unnamed: 0	Invoice ID	Branch	City	\
0	0	0	0	750-67-8428	A	Yangon	
1	1	1	1	631-41-3108	A	Yangon	
2	2	2	2	123-19-1176	A	Yangon	
3	3	3	3	373-73-7910	A	Yangon	
4	4	4	4	355-53-5943	A	Yangon	
5	5	5	5	665-32-9167	A	Yangon	
6	6	6	6	365-64-0515	A	Yangon	
7	7	7	7	252-56-2699	A	Yangon	
8	8	8	8	829-34-3910	A	Yangon	
9	9	9	9	656-95-9349	A	Yangon	

	Customer type	Gender	Product line	Unit price	…	Total	\
0	Member	Female	Health and beauty	74.69	…	548.9715	
1	Normal	Male	Home and lifestyle	46.33	…	340.5255	
2	Member	Male	Health and beauty	58.22	…	489.0480	
3	Normal	Male	Sports and travel	86.31	…	634.3785	

```
4        Member  Female  Electronic accessories       68.84  …  433.6920
5        Member  Female       Health and beauty       36.26  …   76.1460
6        Normal  Female  Electronic accessories       46.95  …  246.4875
7        Normal    Male      Food and beverages       43.19  …  453.4950
8        Normal  Female       Health and beauty       71.38  …  749.4900
9        Member  Female       Health and beauty       68.93  …  506.6355

       Time      Payment    cogs gross margin percentage  gross income  \
0  13:08:00      Ewallet  522.83                4.761905       26.1415
1  13:23:00  Credit card  324.31                4.761905       16.2155
2  20:33:00      Ewallet  465.76                4.761905       23.2880
3  10:37:00      Ewallet  604.17                4.761905       30.2085
4  14:36:00      Ewallet  413.04                4.761905       20.6520
5  17:15:00  Credit card   72.52                4.761905        3.6260
6  10:25:00      Ewallet  234.75                4.761905       11.7375
7  16:48:00      Ewallet  431.90                4.761905       21.5950
8  19:21:00         Cash  713.80                4.761905       35.6900
9  11:03:00  Credit card  482.51                4.761905       24.1255

   Rating       Day  Date(dd-mm-yyyy)     Month
0     9.1  Saturday          5/1/2019   January
1     7.4    Sunday          3/3/2019     March
2     8.4    Sunday        27-01-2019   January
3     5.3    Friday          8/2/2019  February
4     5.8    Monday        25-02-2019  February
5     7.2  Thursday         10/1/2019   January
6     7.1   Tuesday         12/2/2019  February
7     8.2  Thursday          7/2/2019  February
8     5.7    Friday        29-03-2019     March
9     4.6    Monday         11/3/2019     March

[10 rows x 22 columns]
```

[45]:
```
x=data.groupby(['Month' , 'Product line']).size()

x
```

[45]:
```
Month     Product line
February  Electronic accessories    54
          Fashion accessories       60
          Food and beverages        62
          Health and beauty         46
          Home and lifestyle        38
          Sports and travel         43
January   Electronic accessories    54
          Fashion accessories       64
          Food and beverages        56
```

```
          Health and beauty         49
          Home and lifestyle        59
          Sports and travel         70
March     Electronic accessories    62
          Fashion accessories       54
          Food and beverages        56
          Health and beauty         57
          Home and lifestyle        63
          Sports and travel         53
dtype: int64
```

[46]:
```
#size() = Compute the number of values in each group

y=data.groupby(['Product line' , 'Quantity']).size()
y
```

[46]:
```
Product line            Quantity
Electronic accessories  1           20
                        2            8
                        3           16
                        4           19
                        5           17
                        6           19
                        7           16
                        8           17
                        9           16
                        10          22
Fashion accessories     1           30
                        2           22
                        3           14
                        4           21
                        5           15
                        6            8
                        7           21
                        8           12
                        9           14
                        10          21
Food and beverages      1           15
                        2           16
                        3           23
                        4           18
                        5           21
                        6           17
                        7           12
                        8           15
                        9           17
                        10          20
```

```
Health and beauty       1           15
                        2           13
                        3           13
                        4           15
                        5           18
                        6           14
                        7           19
                        8           15
                        9           13
                        10          17
Home and lifestyle      1           13
                        2           14
                        3           13
                        4           22
                        5           13
                        6           21
                        7           12
                        8           17
                        9           18
                        10          17
Sports and travel       1           19
                        2           18
                        3           11
                        4           14
                        5           18
                        6           19
                        7           22
                        8            9
                        9           14
                        10          22
dtype: int64
```

[47]:
```
data['Quantity'].sum()
```

[47]:
```
np.int64(5510)
```

[48]:
```
pd.concat([x ,y],axis=0).head(10)
```

[48]:
```
February  Electronic accessories    54
          Fashion accessories       60
          Food and beverages        62
          Health and beauty         46
          Home and lifestyle        38
          Sports and travel         43
January   Electronic accessories    54
          Fashion accessories       64
          Food and beverages        56
```

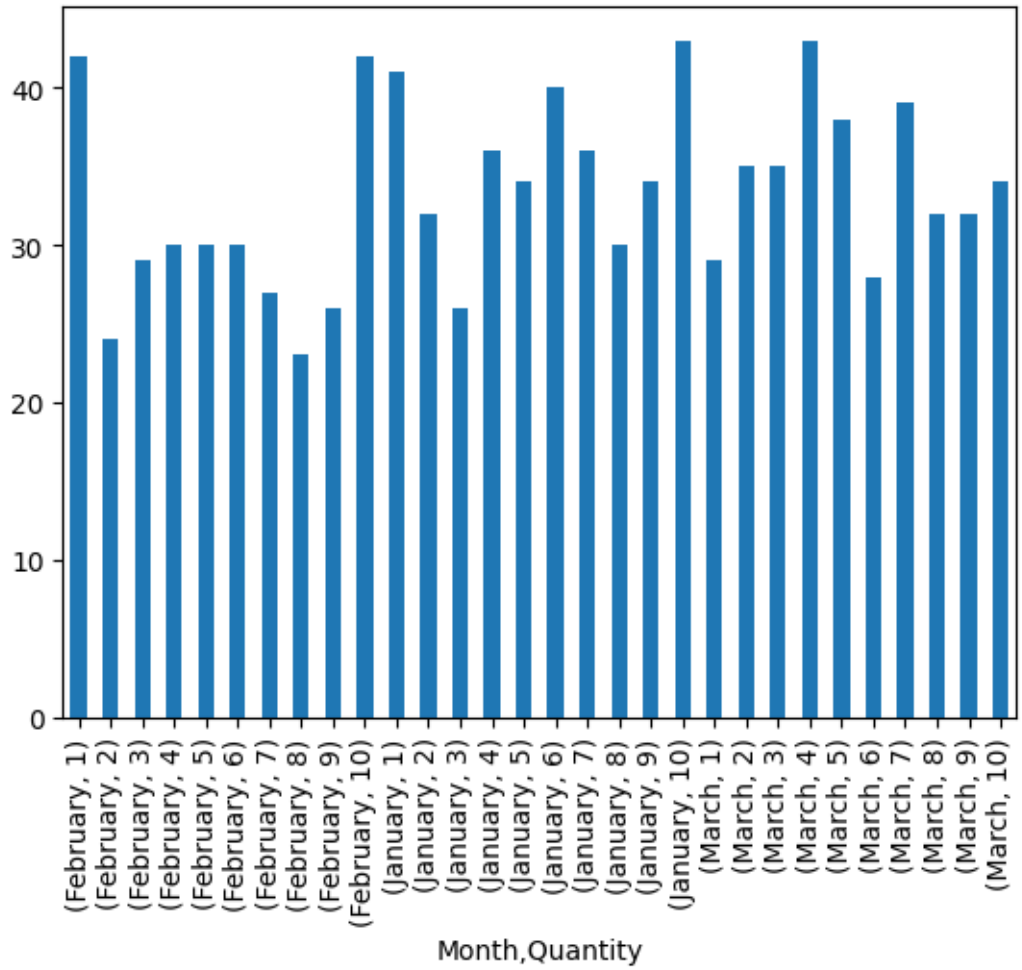
```
          Health and beauty         49
dtype: int64
```

[49]:
```
z=data.groupby(['Month' , 'Quantity']).size()
z
```

[49]:
```
Month     Quantity
February  1           42
          2           24
          3           29
          4           30
          5           30
          6           30
          7           27
          8           23
          9           26
          10          42
January   1           41
          2           32
          3           26
          4           36
          5           34
          6           40
          7           36
          8           30
          9           34
          10          43
March     1           29
          2           35
          3           35
          4           43
          5           38
          6           28
          7           39
          8           32
          9           32
          10          34
dtype: int64
```

[50]:
```
 z.plot.bar()
```

[50]:
```
<Axes: xlabel='Month,Quantity'>
```



[ ]:
```
### Need to make this graph more visually appealing
```

[51]:
```
data.groupby(['Month' ,'Date(dd-mm-yyyy)', 'Product line' , 'Quantity' ]).
↪size().reset_index(name='Count').head(50)
```

[51]:

	Month	Date(dd-mm-yyyy)	Product line	Quantity	Count
0	February	1/2/2019	Electronic accessories	9	1
1	February	1/2/2019	Food and beverages	5	1
2	February	1/2/2019	Food and beverages	6	1
3	February	1/2/2019	Health and beauty	4	1
4	February	1/2/2019	Home and lifestyle	9	1
5	February	1/2/2019	Sports and travel	7	1
6	February	10/2/2019	Electronic accessories	3	1
7	February	10/2/2019	Electronic accessories	6	1
8	February	10/2/2019	Electronic accessories	10	1
9	February	10/2/2019	Fashion accessories	7	2

10	February	10/2/2019	Fashion accessories	10	1
11	February	10/2/2019	Food and beverages	5	1
12	February	10/2/2019	Health and beauty	2	1
13	February	10/2/2019	Home and lifestyle	1	1
14	February	10/2/2019	Home and lifestyle	7	1
15	February	10/2/2019	Sports and travel	4	1
16	February	11/2/2019	Fashion accessories	10	1
17	February	11/2/2019	Food and beverages	10	2
18	February	11/2/2019	Health and beauty	9	1
19	February	11/2/2019	Home and lifestyle	3	2
20	February	11/2/2019	Home and lifestyle	5	1
21	February	11/2/2019	Sports and travel	4	1
22	February	12/2/2019	Electronic accessories	1	1
23	February	12/2/2019	Electronic accessories	5	1
24	February	12/2/2019	Fashion accessories	1	1
25	February	12/2/2019	Fashion accessories	8	1
26	February	12/2/2019	Food and beverages	10	1
27	February	12/2/2019	Health and beauty	6	1
28	February	12/2/2019	Home and lifestyle	10	1
29	February	12/2/2019	Sports and travel	2	1
30	February	13-02-2019	Electronic accessories	1	1
31	February	13-02-2019	Electronic accessories	4	1
32	February	13-02-2019	Fashion accessories	2	1
33	February	13-02-2019	Fashion accessories	5	1
34	February	13-02-2019	Fashion accessories	7	1
35	February	13-02-2019	Food and beverages	2	1
36	February	13-02-2019	Sports and travel	4	1
37	February	13-02-2019	Sports and travel	6	1
38	February	14-02-2019	Electronic accessories	8	1
39	February	14-02-2019	Food and beverages	2	1
40	February	14-02-2019	Food and beverages	10	1
41	February	14-02-2019	Health and beauty	1	1
42	February	14-02-2019	Health and beauty	2	1
43	February	14-02-2019	Health and beauty	4	1
44	February	14-02-2019	Sports and travel	3	1
45	February	14-02-2019	Sports and travel	7	1
46	February	15-02-2019	Electronic accessories	1	1
47	February	15-02-2019	Electronic accessories	9	1
48	February	15-02-2019	Electronic accessories	10	1
49	February	15-02-2019	Fashion accessories	3	2

[53]:
```
GROUP_1_DATA=data.groupby(['Month' ,'Date(dd-mm-yyyy)','Day', 'Product line' ,␣
↪'Quantity' ]).size().reset_index(name='Count')

GROUP_1_DATA.to_csv('GROUP_1_DATA .csv' , index= False )
```

[54]:
```
data2=pd.read_csv('GROUP_1_DATA .csv')

data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 903 entries, 0 to 902
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Month             903 non-null    object
 1   Date(dd-mm-yyyy)  903 non-null    object
 2   Day               903 non-null    object
 3   Product line      903 non-null    object
 4   Quantity          903 non-null    int64
 5   Count             903 non-null    int64
dtypes: int64(2), object(4)
memory usage: 42.5+ KB
```

[55]:
```
data2['Month'].unique()
```

[55]:
```
array(['February', 'January', 'March'], dtype=object)
```

[56]:
```
data2['Product line'].unique()
```

[56]:
```
array(['Electronic accessories', 'Food and beverages',
       'Health and beauty', 'Home and lifestyle', 'Sports and travel',
       'Fashion accessories'], dtype=object)
```

[57]:
```
data2.groupby(['Month', 'Day' , 'Product line' ])['Quantity'].sum().
  ↪reset_index(name='Count').head(20)
```

[57]:

	Month	Day	Product line	Count
0	February	Friday	Electronic accessories	39
1	February	Friday	Fashion accessories	56
2	February	Friday	Food and beverages	24
3	February	Friday	Health and beauty	36
4	February	Friday	Home and lifestyle	38
5	February	Friday	Sports and travel	55
6	February	Monday	Electronic accessories	31
7	February	Monday	Fashion accessories	35
8	February	Monday	Food and beverages	23
9	February	Monday	Health and beauty	39
10	February	Monday	Home and lifestyle	29
11	February	Monday	Sports and travel	15
12	February	Saturday	Electronic accessories	44
13	February	Saturday	Fashion accessories	54
14	February	Saturday	Food and beverages	56
15	February	Saturday	Health and beauty	21

```
16  February  Saturday      Home and lifestyle     20
17  February  Saturday       Sports and travel     35
18  February    Sunday  Electronic accessories     48
19  February    Sunday     Fashion accessories     28
```

[58]:
```
#storing the grouped dataframe in the variable called graph

graph=data2.groupby(['Month' ,'Day','Product line'])['Quantity'].sum().
 ↪reset_index(name='Count')

graph
```

[58]:
```
        Month        Day            Product line  Count
0    February     Friday  Electronic accessories     39
1    February     Friday     Fashion accessories     56
2    February     Friday      Food and beverages     24
3    February     Friday       Health and beauty     36
4    February     Friday      Home and lifestyle     38
..        …        …                       …    …
121     March  Wednesday     Fashion accessories     39
122     March  Wednesday      Food and beverages     37
123     March  Wednesday       Health and beauty     19
124     March  Wednesday      Home and lifestyle     39
125     March  Wednesday       Sports and travel     70

[126 rows x 4 columns]
```

[59]:
```
graph.columns
```

[59]:
```
Index(['Month', 'Day', 'Product line', 'Count'], dtype='object')
```

[60]:
```
graph['Month'].unique()
```

[60]:
```
array(['February', 'January', 'March'], dtype=object)
```

3.1 PLOTTING THE GRAPH

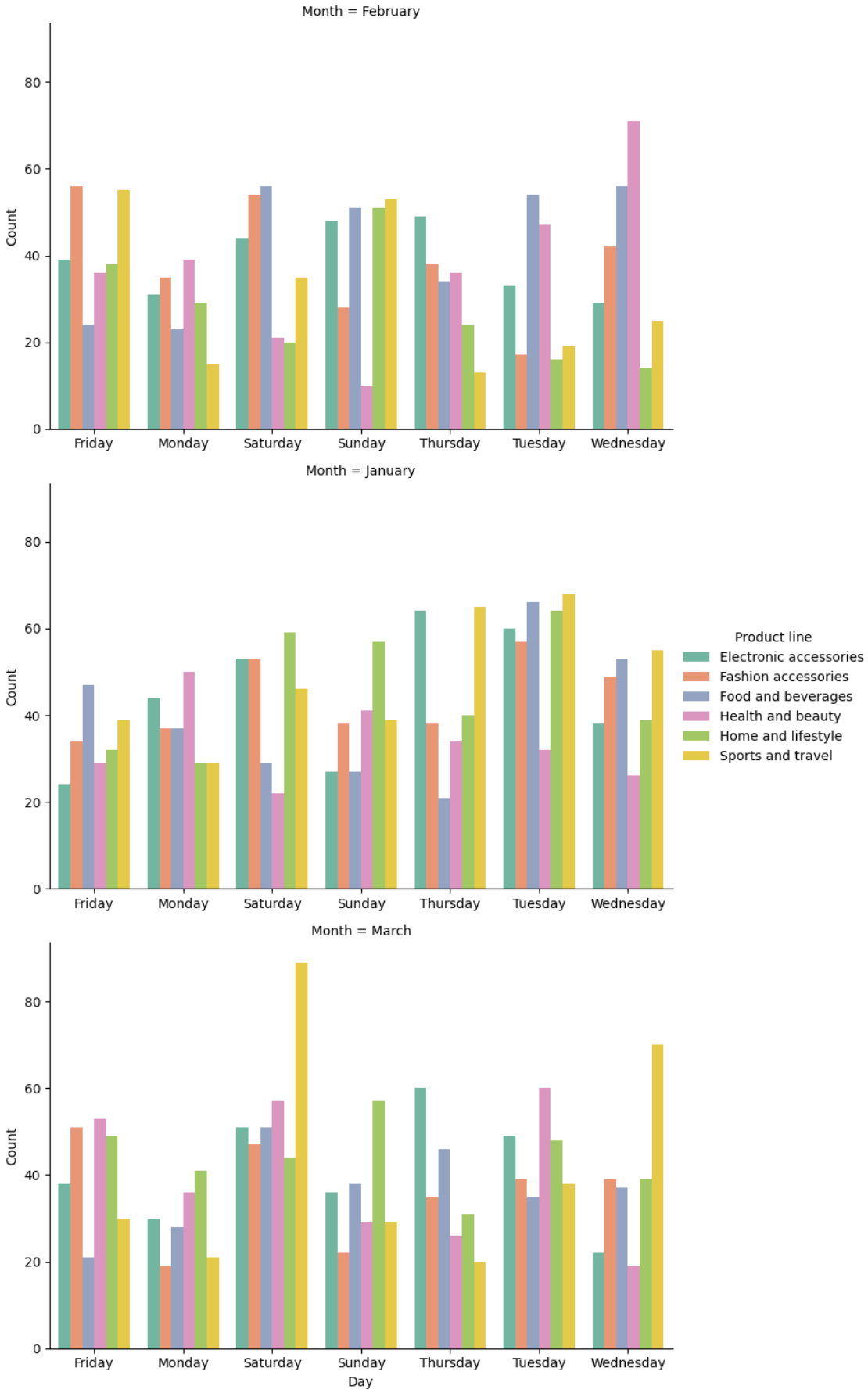
[61]:
```
plt.figure(figsize=(20, 10)) #size of figure 20 bredth 10 height
```

[61]:
```
<Figure size 2000x1000 with 0 Axes>
```

```
<Figure size 2000x1000 with 0 Axes>
```

4 Quantity of product sold on week days of thr month

[63]:
```
g= sns.catplot(data=graph
               ,x='Day' ,
               y='Count' ,
               hue='Product line' ,
               row='Month' ,
               kind='bar'  ,
               height=5,
               aspect=1.5,
               palette='Set2',
               sharex=False)
#sharex=false for dispplaying the days in all the graphs
# change col to row to make it vertical
plt.show()
```


Month = February
Month = January
Month = March
Count
Day
Friday
Monday
Saturday
Sunday
Thursday
Tuesday
Wednesday
0
20
40
60
80
Product line
Electronic accessories
Fashion accessories
Food and beverages
Health and beauty
Home and lifestyle
Sports and travel

4.0.1 MAKING IT INTERACTIVE USING PLOTLY EXPRESS

[64]:
```
g= px.bar(
    graph,
    x="Day",
    y="Count",
    color="Product line",
    facet_row="Month",
    barmode="group",
    height=1500,
    color_discrete_sequence=px.colors.qualitative.Set2,
    title="Sales by Product Line and Day (Interactive)"

)
g.update_xaxes(matches='x')
g.update_yaxes(matches='y')
g.show()
```

[65]:
```
g.for_each_xaxis(lambda axis :
                    axis.update(title='Day'))

g.update_xaxes(showticklabels=True)
```

[66]:
```
data2.columns
```

[66]:
```
Index(['Month', 'Date(dd-mm-yyyy)', 'Day', 'Product line', 'Quantity',
       'Count'],
      dtype='object')
```

[67]:
```
data2['Product line'].unique()
```

[67]:
```
array(['Electronic accessories', 'Food and beverages',
       'Health and beauty', 'Home and lifestyle', 'Sports and travel',
       'Fashion accessories'], dtype=object)
```

[68]:
```
data2['Product line'].nunique()
```

[68]:
```
6
```

[69]:
```
data2['Product line'].count()
```

[69]:
```
np.int64(903)
```

[70]:
```
data2.describe()
```

[70]:
```
          Quantity       Count
count   903.000000  903.000000
mean      5.513843    1.107420
std       2.930378    0.333928
min       1.000000    1.000000
25%       3.000000    1.000000
50%       5.000000    1.000000
75%       8.000000    1.000000
max      10.000000    3.000000
```

[71]:
```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 903 entries, 0 to 902
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Month             903 non-null    object
 1   Date(dd-mm-yyyy)  903 non-null    object
 2   Day               903 non-null    object
 3   Product line      903 non-null    object
 4   Quantity          903 non-null    int64 
 5   Count             903 non-null    int64 
dtypes: int64(2), object(4)
memory usage: 42.5+ KB
```

[ ]:
```
#data is clean and no missing columns are there in it 903== 903`
```

5 adding tax column to data 2

[72]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Unnamed: 0.2            1000 non-null   int64  
 1   Unnamed: 0.1            1000 non-null   int64  
 2   Unnamed: 0              1000 non-null   int64  
 3   Invoice ID              1000 non-null   object 
 4   Branch                  1000 non-null   object 
 5   City                    1000 non-null   object 
 6   Customer type           1000 non-null   object 
 7   Gender                  1000 non-null   object 
 8   Product line            1000 non-null   object 
 9   Unit price              1000 non-null   float64
 10  Quantity                1000 non-null   int64  
```

```
 11  Tax 5%                   1000 non-null   float64
 12  Total                    1000 non-null   float64
 13  Time                     1000 non-null   object
 14  Payment                  1000 non-null   object
 15  cogs                     1000 non-null   float64
 16  gross margin percentage  1000 non-null   float64
 17  gross income             1000 non-null   float64
 18  Rating                   1000 non-null   float64
 19  Day                      1000 non-null   object
 20  Date(dd-mm-yyyy)         1000 non-null   object
 21  Month                    1000 non-null   object
dtypes: float64(7), int64(4), object(11)
memory usage: 172.0+ KB
```

[73]:
```
data.columns
```

[73]:
```
Index(['Unnamed: 0.2', 'Unnamed: 0.1', 'Unnamed: 0', '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',
       'Date(dd-mm-yyyy)', 'Month'],
      dtype='object')
```

[75]:
```
data2['Tax']=data['Tax 5%'].reindex(data2.index) # in data 2 tax  column wil␣
 ↪contain the values from the column tax 5% iin data
```

[76]:
```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 903 entries, 0 to 902
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Month             903 non-null    object
 1   Date(dd-mm-yyyy)  903 non-null    object
 2   Day               903 non-null    object
 3   Product line      903 non-null    object
 4   Quantity          903 non-null    int64
 5   Count             903 non-null    int64
 6   Tax               903 non-null    float64
dtypes: float64(1), int64(2), object(4)
memory usage: 49.5+ KB
```

[77]:
```
data2.describe()
```

[77]:

	Quantity	Count	Tax
count	903.000000	903.000000	903.000000
mean	5.513843	1.107420	15.303181

```
std      2.930378    0.333928   11.657250
min      1.000000    1.000000    0.604500
25%      3.000000    1.000000    5.966000
50%      5.000000    1.000000   12.048000
75%      8.000000    1.000000   22.258500
max     10.000000    3.000000   49.650000
```

[78]:
```
data2.head(10)
```

[78]:
```
      Month Date(dd-mm-yyyy)     Day            Product line  Quantity  Count  \
0  February         1/2/2019  Friday  Electronic accessories         9      1
1  February         1/2/2019  Friday      Food and beverages         5      1
2  February         1/2/2019  Friday      Food and beverages         6      1
3  February         1/2/2019  Friday       Health and beauty         4      1
4  February         1/2/2019  Friday      Home and lifestyle         9      1
5  February         1/2/2019  Friday       Sports and travel         7      1
6  February        10/2/2019  Sunday  Electronic accessories         3      1
7  February        10/2/2019  Sunday  Electronic accessories         6      1
8  February        10/2/2019  Sunday  Electronic accessories        10      1
9  February        10/2/2019  Sunday     Fashion accessories         7      2

       Tax
0  26.1415
1  16.2155
2  23.2880
3  30.2085
4  20.6520
5   3.6260
6  11.7375
7  21.5950
8  35.6900
9  24.1255
```

[79]:
```
data2.columns
```

[79]:
```
Index(['Month', 'Date(dd-mm-yyyy)', 'Day', 'Product line', 'Quantity', 'Count',
       'Tax'],
      dtype='object')
```

[80]:
```
data2.columns.tolist()
```

[80]:
```
['Month',
 'Date(dd-mm-yyyy)',
 'Day',
 'Product line',
 'Quantity',
 'Count',
```

```
       'Tax']
```

[81]:
```
data2.columns
```

[81]:
```
Index(['Month', 'Date(dd-mm-yyyy)', 'Day', 'Product line', 'Quantity', 'Count',
       'Tax'],
      dtype='object')
```

[82]:
```
data2.drop('Tax' , axis=1, ) # axis 1 cuz its column and if it would have been␣
 ↪rows the axis =0
```

[82]:
```
        Month Date(dd-mm-yyyy)       Day            Product line  Quantity  \
0    February         1/2/2019    Friday  Electronic accessories         9
1    February         1/2/2019    Friday      Food and beverages         5
2    February         1/2/2019    Friday      Food and beverages         6
3    February         1/2/2019    Friday       Health and beauty         4
4    February         1/2/2019    Friday      Home and lifestyle         9
..        …              …       …                   …       …
898     March         9/3/2019  Saturday      Home and lifestyle         7
899     March         9/3/2019  Saturday       Sports and travel         2
900     March         9/3/2019  Saturday       Sports and travel         5
901     March         9/3/2019  Saturday       Sports and travel         9
902     March         9/3/2019  Saturday       Sports and travel        10

     Count
0        1
1        1
2        1
3        1
4        1
..     …
898      1
899      1
900      1
901      1
902      1

[903 rows x 6 columns]
```

[85]:
```
data2.to_csv('Tax_dataset.csv' ,index=False)
```

[ ]:
```
# need to remove the tax column from this data set cuz the values are not␣
 ↪correct
```

5.0.1 Need to group data so the values matches correctly

[86]:
```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 903 entries, 0 to 902
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Month             903 non-null    object
 1   Date(dd-mm-yyyy)  903 non-null    object
 2   Day               903 non-null    object
 3   Product line      903 non-null    object
 4   Quantity          903 non-null    int64
 5   Count             903 non-null    int64
 6   Tax               903 non-null    float64
dtypes: float64(1), int64(2), object(4)
memory usage: 49.5+ KB
```

[87]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 22 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Unnamed: 0.2             1000 non-null   int64
 1   Unnamed: 0.1             1000 non-null   int64
 2   Unnamed: 0               1000 non-null   int64
 3   Invoice ID               1000 non-null   object
 4   Branch                   1000 non-null   object
 5   City                     1000 non-null   object
 6   Customer type            1000 non-null   object
 7   Gender                   1000 non-null   object
 8   Product line             1000 non-null   object
 9   Unit price               1000 non-null   float64
 10  Quantity                 1000 non-null   int64
 11  Tax 5%                   1000 non-null   float64
 12  Total                    1000 non-null   float64
 13  Time                     1000 non-null   object
 14  Payment                  1000 non-null   object
 15  cogs                     1000 non-null   float64
 16  gross margin percentage  1000 non-null   float64
 17  gross income             1000 non-null   float64
 18  Rating                   1000 non-null   float64
 19  Day                      1000 non-null   object
 20  Date(dd-mm-yyyy)         1000 non-null   object
 21  Month                    1000 non-null   object
dtypes: float64(7), int64(4), object(11)
```

```
memory usage: 172.0+ KB
```

5.0.2 Grouping the coulumns city , Productline , unit price , Quantity , tax 5% on total , Total ,Month ,

[94]:
```
TAX_GROUP=data.groupby(['Month','Product line', 'Quantity' ,'Unit price','Tax␣
↪5%','Total']).size().reset_index(name='Count')
```

[95]:
```
TAX_GROUP
```

[95]:
```
        Month            Product line  Quantity  Unit price   Tax 5%  \
0    February  Electronic accessories         1       28.96   1.4480
1    February  Electronic accessories         1       39.48   1.9740
2    February  Electronic accessories         1       39.75   1.9875
3    February  Electronic accessories         1       60.30   3.0150
4    February  Electronic accessories         1       60.95   3.0475
..        …                      …       …         …      …
994     March       Sports and travel        10       44.02  22.0100
995     March       Sports and travel        10       52.26  26.1300
996     March       Sports and travel        10       54.55  27.2750
997     March       Sports and travel        10       69.74  34.8700
998     March       Sports and travel        10       76.92  38.4600

        Total  Count
0     30.4080      1
1     41.4540      1
2     41.7375      1
3     63.3150      1
4     63.9975      1
..        …    …
994  462.2100      1
995  548.7300      1
996  572.7750      1
997  732.2700      1
998  807.6600      1

[999 rows x 7 columns]
```

[90]:
```
TAX_GROUP.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Month         999 non-null    object
 1   Product line  999 non-null    object
 2   Quantity      999 non-null    int64
```

```
 3   Unit price    999 non-null    float64
 4   Tax 5%        999 non-null    float64
 5   Total         999 non-null    float64
 6   Count         999 non-null    int64
dtypes: float64(3), int64(2), object(2)
memory usage: 54.8+ KB
```

[96]:
```
TAX_GROUP.to_csv('Tax_dataset.csv', index=False)
```

[97]:
```
data3=pd.read_csv('Tax_dataset.csv')
```

[93]:
```
data3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Month         999 non-null    object
 1   Product line  999 non-null    object
 2   Quantity      999 non-null    int64
 3   Unit price    999 non-null    float64
 4   Tax 5%        999 non-null    float64
 5   Total         999 non-null    float64
 6   Count         999 non-null    int64
dtypes: float64(3), int64(2), object(2)
memory usage: 54.8+ KB
```

[98]:
```
data3.head(50)
```

[98]:

	Month	Product line	Quantity	Unit price	Tax 5%	Total	\
0	February	Electronic accessories	1	28.96	1.4480	30.4080	
1	February	Electronic accessories	1	39.48	1.9740	41.4540	
2	February	Electronic accessories	1	39.75	1.9875	41.7375	
3	February	Electronic accessories	1	60.30	3.0150	63.3150	
4	February	Electronic accessories	1	60.95	3.0475	63.9975	
5	February	Electronic accessories	1	62.48	3.1240	65.6040	
6	February	Electronic accessories	1	71.95	3.5975	75.5475	
7	February	Electronic accessories	1	76.82	3.8410	80.6610	
8	February	Electronic accessories	1	98.84	4.9420	103.7820	
9	February	Electronic accessories	1	99.69	4.9845	104.6745	
10	February	Electronic accessories	2	20.89	2.0890	43.8690	
11	February	Electronic accessories	2	46.61	4.6610	97.8810	
12	February	Electronic accessories	3	48.09	7.2135	151.4835	
13	February	Electronic accessories	3	81.40	12.2100	256.4100	
14	February	Electronic accessories	3	94.64	14.1960	298.1160	
15	February	Electronic accessories	4	32.25	6.4500	135.4500	
16	February	Electronic accessories	4	65.94	13.1880	276.9480	

17	February	Electronic accessories	5	11.81	2.9525	62.0025
18	February	Electronic accessories	5	12.05	3.0125	63.2625
19	February	Electronic accessories	5	34.56	8.6400	181.4400
20	February	Electronic accessories	5	46.95	11.7375	246.4875
21	February	Electronic accessories	5	86.04	21.5100	451.7100
22	February	Electronic accessories	6	12.45	3.7350	78.4350
23	February	Electronic accessories	6	18.93	5.6790	119.2590
24	February	Electronic accessories	6	35.49	10.6470	223.5870
25	February	Electronic accessories	6	46.02	13.8060	289.9260
26	February	Electronic accessories	6	50.45	15.1350	317.8350
27	February	Electronic accessories	6	68.84	20.6520	433.6920
28	February	Electronic accessories	6	87.45	26.2350	550.9350
29	February	Electronic accessories	6	90.70	27.2100	571.4100
30	February	Electronic accessories	7	25.22	8.8270	185.3670
31	February	Electronic accessories	7	26.26	9.1910	193.0110
32	February	Electronic accessories	7	74.58	26.1030	548.1630
33	February	Electronic accessories	7	92.60	32.4100	680.6100
34	February	Electronic accessories	8	14.96	5.9840	125.6640
35	February	Electronic accessories	8	35.74	14.2960	300.2160
36	February	Electronic accessories	8	40.86	16.3440	343.2240
37	February	Electronic accessories	8	57.91	23.1640	486.4440
38	February	Electronic accessories	8	71.89	28.7560	603.8760
39	February	Electronic accessories	8	85.98	34.3920	722.2320
40	February	Electronic accessories	8	99.56	39.8240	836.3040
41	February	Electronic accessories	9	23.07	10.3815	218.0115
42	February	Electronic accessories	9	69.58	31.3110	657.5310
43	February	Electronic accessories	9	75.59	34.0155	714.3255
44	February	Electronic accessories	9	77.63	34.9335	733.6035
45	February	Electronic accessories	9	88.25	39.7125	833.9625
46	February	Electronic accessories	10	17.42	8.7100	182.9100
47	February	Electronic accessories	10	22.95	11.4750	240.9750
48	February	Electronic accessories	10	24.74	12.3700	259.7700
49	February	Electronic accessories	10	32.80	16.4000	344.4000

	Count
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1

```
12      1
13      1
14      1
15      1
16      1
17      1
18      1
19      1
20      1
21      1
22      1
23      1
24      1
25      1
26      1
27      1
28      1
29      1
30      1
31      1
32      1
33      1
34      1
35      1
36      1
37      1
38      1
39      1
40      1
41      1
42      1
43      1
44      1
45      1
46      1
47      1
48      1
49      1
```

[99]:
```
T1=data3.groupby(['Month' , 'Product line'])['Tax 5%'].sum().reset_index()
tax_graph= px.bar(T1 ,y='Tax 5%' ,  x='Product line' , color='Product line'␣
 ↪,facet_row='Month' , title='Tax graph' ,height=1000)

tax_graph.update_xaxes(showticklabels=True)
tax_graph.show()
```

[100]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 22 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Unnamed: 0.2             1000 non-null   int64
 1   Unnamed: 0.1             1000 non-null   int64
 2   Unnamed: 0               1000 non-null   int64
 3   Invoice ID               1000 non-null   object
 4   Branch                   1000 non-null   object
 5   City                     1000 non-null   object
 6   Customer type            1000 non-null   object
 7   Gender                   1000 non-null   object
 8   Product line             1000 non-null   object
 9   Unit price               1000 non-null   float64
 10  Quantity                 1000 non-null   int64
 11  Tax 5%                   1000 non-null   float64
 12  Total                    1000 non-null   float64
 13  Time                     1000 non-null   object
 14  Payment                  1000 non-null   object
 15  cogs                     1000 non-null   float64
 16  gross margin percentage  1000 non-null   float64
 17  gross income             1000 non-null   float64
 18  Rating                   1000 non-null   float64
 19  Day                      1000 non-null   object
 20  Date(dd-mm-yyyy)         1000 non-null   object
 21  Month                    1000 non-null   object
dtypes: float64(7), int64(4), object(11)
memory usage: 172.0+ KB
```

5.0.3 rating vs product line which product line has the maximum rating

[101]:
```
rating_data=data.groupby(['City','Product line' ,'Rating']).size().reset_index()
rating_data
```

[101]:

	City	Product line	Rating	0
0	Mandalay	Electronic accessories	4.0	1
1	Mandalay	Electronic accessories	4.2	1
2	Mandalay	Electronic accessories	4.3	1
3	Mandalay	Electronic accessories	4.8	1
4	Mandalay	Electronic accessories	4.9	1
..	…	…	… ..	
669	Yangon	Sports and travel	9.3	2
670	Yangon	Sports and travel	9.6	3
671	Yangon	Sports and travel	9.7	3
672	Yangon	Sports and travel	9.8	1
673	Yangon	Sports and travel	9.9	1

```
[674 rows x 4 columns]
```

[102]:
```
rating_graph = rating_data.pivot_table( columns='Product line' , index='City',
↪values='Rating' , aggfunc='mean' ).reset_index()
```

[103]:
```
rating_graph.columns.name=None
```

[104]:
```
rating_graph
```

[104]:

	City	Electronic accessories	Fashion accessories	Food and beverages	Health and beauty	Home and lifestyle	Sports and travel
0	Mandalay	7.069231	6.629268	6.897222	7.035294	6.802857	6.682051
1	Naypyitaw	6.887500	7.217073	7.140909	7.102778	7.077419	7.038235
2	Yangon	6.967647	6.902941	7.195238	6.931429	7.157143	7.218919

[186]:
```
#Giving errors need to melt this data so it is correct form to plot
```

5.1 DATA MELTING need to covert this table to correct format for plotting the graph

[106]:
```
melt = pd.melt(rating_graph , id_vars='City' , var_name='Product line' ,
↪value_name='Average rating')

melt
```

[106]:

	City	Product line	Average rating
0	Mandalay	Electronic accessories	7.069231
1	Naypyitaw	Electronic accessories	6.887500
2	Yangon	Electronic accessories	6.967647
3	Mandalay	Fashion accessories	6.629268
4	Naypyitaw	Fashion accessories	7.217073
5	Yangon	Fashion accessories	6.902941
6	Mandalay	Food and beverages	6.897222
7	Naypyitaw	Food and beverages	7.140909
8	Yangon	Food and beverages	7.195238
9	Mandalay	Health and beauty	7.035294
10	Naypyitaw	Health and beauty	7.102778
11	Yangon	Health and beauty	6.931429
12	Mandalay	Home and lifestyle	6.802857
13	Naypyitaw	Home and lifestyle	7.077419
14	Yangon	Home and lifestyle	7.157143
15	Mandalay	Sports and travel	6.682051

```
 16  Naypyitaw       Sports and travel        7.038235
 17     Yangon       Sports and travel        7.218919
```

[107]:
```
RG=px.bar(melt,x='City' , y='Average rating' , color='Product line' ␣
 ↪,barmode='group', title='Average ratings of Product line' )
RG.show()
```

[108]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 22 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Unnamed: 0.2             1000 non-null   int64
 1   Unnamed: 0.1             1000 non-null   int64
 2   Unnamed: 0               1000 non-null   int64
 3   Invoice ID               1000 non-null   object
 4   Branch                   1000 non-null   object
 5   City                     1000 non-null   object
 6   Customer type            1000 non-null   object
 7   Gender                   1000 non-null   object
 8   Product line             1000 non-null   object
 9   Unit price               1000 non-null   float64
 10  Quantity                 1000 non-null   int64
 11  Tax 5%                   1000 non-null   float64
 12  Total                    1000 non-null   float64
 13  Time                     1000 non-null   object
 14  Payment                  1000 non-null   object
 15  cogs                     1000 non-null   float64
 16  gross margin percentage  1000 non-null   float64
 17  gross income             1000 non-null   float64
 18  Rating                   1000 non-null   float64
 19  Day                      1000 non-null   object
 20  Date(dd-mm-yyyy)         1000 non-null   object
 21  Month                    1000 non-null   object
dtypes: float64(7), int64(4), object(11)
memory usage: 172.0+ KB
```

6 SALES BY CITY

[109]:
```
data.groupby(['City' , 'Total']).size().reset_index()
```

[109]:
```
          City      Total  0
0    Mandalay    18.6375  1
1    Mandalay    20.1075  1
2    Mandalay    26.7225  1
```

```
3     Mandalay    28.4235  1
4     Mandalay    30.9960  1
..         …         … ..
992     Yangon   926.9505  1
993     Yangon   931.0350  1
994     Yangon   932.3370  1
995     Yangon   951.8250  1
996     Yangon  1039.2900  1

[997 rows x 3 columns]
```

[110]:
```
sales=data.groupby('City' )['Total'].sum().reset_index()
sales
```

[110]:
```
         City        Total
0   Mandalay  106197.6720
1  Naypyitaw  110568.7065
2     Yangon  106200.3705
```

[111]:
```
sales_graph= px.pie(sales ,  names='City' , values='Total' , title='TOTAL Sales␣
 ↪by city' )
sales_graph.show()
```

6.1 CALCULATING CUSTOMER SATISFACTION INDEX

[112]:
```
rating=data.groupby(['City' , 'Product line' ,'Rating']).size().
 ↪reset_index(name='Count of ratings occured')
rating.head(50)
```

[112]:

	City	Product line	Rating	Count of ratings occured
0	Mandalay	Electronic accessories	4.0	1
1	Mandalay	Electronic accessories	4.2	1
2	Mandalay	Electronic accessories	4.3	1
3	Mandalay	Electronic accessories	4.8	1
4	Mandalay	Electronic accessories	4.9	1
5	Mandalay	Electronic accessories	5.1	1
6	Mandalay	Electronic accessories	5.3	1
7	Mandalay	Electronic accessories	5.5	1
8	Mandalay	Electronic accessories	5.6	1
9	Mandalay	Electronic accessories	5.7	1
10	Mandalay	Electronic accessories	5.8	1
11	Mandalay	Electronic accessories	6.0	3
12	Mandalay	Electronic accessories	6.1	3
13	Mandalay	Electronic accessories	6.2	2
14	Mandalay	Electronic accessories	6.3	1
15	Mandalay	Electronic accessories	6.5	3
16	Mandalay	Electronic accessories	6.6	1

17	Mandalay	Electronic accessories	6.7	2
18	Mandalay	Electronic accessories	6.8	1
19	Mandalay	Electronic accessories	7.0	1
20	Mandalay	Electronic accessories	7.1	1
21	Mandalay	Electronic accessories	7.3	1
22	Mandalay	Electronic accessories	7.6	3
23	Mandalay	Electronic accessories	7.7	1
24	Mandalay	Electronic accessories	7.8	2
25	Mandalay	Electronic accessories	7.9	1
26	Mandalay	Electronic accessories	8.0	2
27	Mandalay	Electronic accessories	8.1	2
28	Mandalay	Electronic accessories	8.2	1
29	Mandalay	Electronic accessories	8.6	2
30	Mandalay	Electronic accessories	8.7	1
31	Mandalay	Electronic accessories	8.8	1
32	Mandalay	Electronic accessories	8.9	2
33	Mandalay	Electronic accessories	9.0	2
34	Mandalay	Electronic accessories	9.4	1
35	Mandalay	Electronic accessories	9.5	1
36	Mandalay	Electronic accessories	9.8	1
37	Mandalay	Electronic accessories	9.9	1
38	Mandalay	Electronic accessories	10.0	1
39	Mandalay	Fashion accessories	4.1	2
40	Mandalay	Fashion accessories	4.2	1
41	Mandalay	Fashion accessories	4.3	1
42	Mandalay	Fashion accessories	4.4	1
43	Mandalay	Fashion accessories	4.5	1
44	Mandalay	Fashion accessories	4.6	2
45	Mandalay	Fashion accessories	4.7	1
46	Mandalay	Fashion accessories	4.8	2
47	Mandalay	Fashion accessories	4.9	1
48	Mandalay	Fashion accessories	5.0	2
49	Mandalay	Fashion accessories	5.1	1

[113]:
```
max_rating=rating['Rating'].max() # maximum rating is 10
max_rating
```

[113]: np.float64(10.0)

[114]:
```
rating['Count of ratings occured'].max()
```

[114]: np.int64(5)

6.1.1 calculation of csi for each product line in the city

[115]:
```
Q=rating.query("`Product line` == 'Electronic accessories' and City ==␣
 ↪'Mandalay'")
Q
```

[115]:

	City	Product line	Rating	Count of ratings occured
0	Mandalay	Electronic accessories	4.0	1
1	Mandalay	Electronic accessories	4.2	1
2	Mandalay	Electronic accessories	4.3	1
3	Mandalay	Electronic accessories	4.8	1
4	Mandalay	Electronic accessories	4.9	1
5	Mandalay	Electronic accessories	5.1	1
6	Mandalay	Electronic accessories	5.3	1
7	Mandalay	Electronic accessories	5.5	1
8	Mandalay	Electronic accessories	5.6	1
9	Mandalay	Electronic accessories	5.7	1
10	Mandalay	Electronic accessories	5.8	1
11	Mandalay	Electronic accessories	6.0	3
12	Mandalay	Electronic accessories	6.1	3
13	Mandalay	Electronic accessories	6.2	2
14	Mandalay	Electronic accessories	6.3	1
15	Mandalay	Electronic accessories	6.5	3
16	Mandalay	Electronic accessories	6.6	1
17	Mandalay	Electronic accessories	6.7	2
18	Mandalay	Electronic accessories	6.8	1
19	Mandalay	Electronic accessories	7.0	1
20	Mandalay	Electronic accessories	7.1	1
21	Mandalay	Electronic accessories	7.3	1
22	Mandalay	Electronic accessories	7.6	3
23	Mandalay	Electronic accessories	7.7	1
24	Mandalay	Electronic accessories	7.8	2
25	Mandalay	Electronic accessories	7.9	1
26	Mandalay	Electronic accessories	8.0	2
27	Mandalay	Electronic accessories	8.1	2
28	Mandalay	Electronic accessories	8.2	1
29	Mandalay	Electronic accessories	8.6	2
30	Mandalay	Electronic accessories	8.7	1
31	Mandalay	Electronic accessories	8.8	1
32	Mandalay	Electronic accessories	8.9	2
33	Mandalay	Electronic accessories	9.0	2
34	Mandalay	Electronic accessories	9.4	1
35	Mandalay	Electronic accessories	9.5	1
36	Mandalay	Electronic accessories	9.8	1
37	Mandalay	Electronic accessories	9.9	1
38	Mandalay	Electronic accessories	10.0	1

[116]:
```
Q.groupby('City')['Rating'].mean().reset_index()
```

[116]:

	City	Rating
0	Mandalay	7.069231

6.1.2 formual of CSI sum of ratings/total number of ratings *100/Max rating

[117]:
```
total_rating=Q['Count of ratings occured'].sum()
total_rating
```

[117]:
```
np.int64(55)
```

[118]:
```
CSI_electronics=(Q['Rating']*Q['Count of ratings occured']).sum()/total_rating/
 ↪max_rating*100
CSI_electronics
```

[118]:
```
np.float64(71.16363636363636)
```

[119]:
```
cities=rating['City'].unique()
cities
```

[119]:
```
array(['Mandalay', 'Naypyitaw', 'Yangon'], dtype=object)
```

[120]:
```
product_lines = rating["Product line"].unique()
cities=rating['City'].unique()

CSI_DATA= []

for city in cities:
    for line in product_lines:
        Q = rating.query("`Product line` == @line and `City` == @city")
        total_rating = Q["Count of ratings occured"].sum()
        weighted_sum = (Q["Rating"] * Q["Count of ratings occured"]).sum()
        max_rating = 10
        csi = (weighted_sum / total_rating) / max_rating * 100
        print(f"{city}: {line}: {csi:.2f}%")

        CSI_DATA.append({
            "City" : city,
            "Product line": line,
                      "CSI(%)":round(csi,2)})

CSI_DF = pd.DataFrame(CSI_DATA)
```

```
Mandalay: Electronic accessories: 71.16%
Mandalay: Fashion accessories: 67.23%
Mandalay: Food and beverages: 69.94%
Mandalay: Health and beauty: 71.00%
```

```
Mandalay: Home and lifestyle: 65.16%
Mandalay: Sports and travel: 65.10%
Naypyitaw: Electronic accessories: 67.47%
Naypyitaw: Fashion accessories: 74.40%
Naypyitaw: Food and beverages: 70.80%
Naypyitaw: Health and beauty: 69.98%
Naypyitaw: Home and lifestyle: 70.60%
Naypyitaw: Sports and travel: 70.29%
Yangon: Electronic accessories: 69.12%
Yangon: Fashion accessories: 68.78%
Yangon: Food and beverages: 72.53%
Yangon: Health and beauty: 69.00%
Yangon: Home and lifestyle: 69.31%
Yangon: Sports and travel: 72.58%
```

[121]:
```
CSI_DF
```

[121]:

	City	Product line	CSI(%)
0	Mandalay	Electronic accessories	71.16
1	Mandalay	Fashion accessories	67.23
2	Mandalay	Food and beverages	69.94
3	Mandalay	Health and beauty	71.00
4	Mandalay	Home and lifestyle	65.16
5	Mandalay	Sports and travel	65.10
6	Naypyitaw	Electronic accessories	67.47
7	Naypyitaw	Fashion accessories	74.40
8	Naypyitaw	Food and beverages	70.80
9	Naypyitaw	Health and beauty	69.98
10	Naypyitaw	Home and lifestyle	70.60
11	Naypyitaw	Sports and travel	70.29
12	Yangon	Electronic accessories	69.12
13	Yangon	Fashion accessories	68.78
14	Yangon	Food and beverages	72.53
15	Yangon	Health and beauty	69.00
16	Yangon	Home and lifestyle	69.31
17	Yangon	Sports and travel	72.58

[122]:
```
m_graph=px.bar(CSI_DF, y='CSI(%)' , x='City' , title='CSI OF CITY'␣
 ↪,barmode='group', color='Product line')

m_graph.show()
```

7 SALES PREDICTION MODEL FOR EACH CITY ACCORDING TO PRODUCT LINE

[123]:
```
sg_data=data.groupby(['City' ,'Product line' ])['Total'].sum().
 ↪reset_index(name='Total Revenue')
sg_data
```

[123]:

	City	Product line	Total Revenue
0	Mandalay	Electronic accessories	17051.4435
1	Mandalay	Fashion accessories	16413.3165
2	Mandalay	Food and beverages	15214.8885
3	Mandalay	Health and beauty	19980.6600
4	Mandalay	Home and lifestyle	17549.1645
5	Mandalay	Sports and travel	19988.1990
6	Naypyitaw	Electronic accessories	18968.9745
7	Naypyitaw	Fashion accessories	21560.0700
8	Naypyitaw	Food and beverages	23766.8550
9	Naypyitaw	Health and beauty	16615.3260
10	Naypyitaw	Home and lifestyle	13895.5530
11	Naypyitaw	Sports and travel	15761.9280
12	Yangon	Electronic accessories	18317.1135
13	Yangon	Fashion accessories	16332.5085
14	Yangon	Food and beverages	17163.1005
15	Yangon	Health and beauty	12597.7530
16	Yangon	Home and lifestyle	22417.1955
17	Yangon	Sports and travel	19372.6995

[124]:
```
sg=px.bar(sg_data , x='Product line' ,y='Total Revenue' , facet_row='City',␣
 ↪color='Product line', height=1200,
          title='TOTAL REVENUE OF EACH PRODUCT LINE IN DIFFERENT CITIES' )

sg.update_xaxes(showticklabels=True)
sg.show()
```

[136]:
```
Month_sales=data.groupby(['City','Month' ,'Product line' ])['Total'].sum().
 ↪reset_index()
Month_sales
```

[136]:

	City	Month	Product line	Total
0	Mandalay	February	Electronic accessories	6686.2530
1	Mandalay	February	Fashion accessories	6137.1135
2	Mandalay	February	Food and beverages	5554.8150
3	Mandalay	February	Health and beauty	5856.4275
4	Mandalay	February	Home and lifestyle	4659.8475
5	Mandalay	February	Sports and travel	5529.8145
6	Mandalay	January	Electronic accessories	6699.7770
7	Mandalay	January	Fashion accessories	6112.5960

8	Mandalay	January	Food and beverages	6609.2775
9	Mandalay	January	Health and beauty	6399.8865
10	Mandalay	January	Home and lifestyle	4586.4420
11	Mandalay	January	Sports and travel	6768.0795
12	Mandalay	March	Electronic accessories	3665.4135
13	Mandalay	March	Fashion accessories	4163.6070
14	Mandalay	March	Food and beverages	3050.7960
15	Mandalay	March	Health and beauty	7724.3460
16	Mandalay	March	Home and lifestyle	8302.8750
17	Mandalay	March	Sports and travel	7690.3050
18	Naypyitaw	February	Electronic accessories	5473.8810
19	Naypyitaw	February	Fashion accessories	7699.1145
20	Naypyitaw	February	Food and beverages	7391.3175
21	Naypyitaw	February	Health and beauty	5830.3455
22	Naypyitaw	February	Home and lifestyle	3002.9055
23	Naypyitaw	February	Sports and travel	3537.4185
24	Naypyitaw	January	Electronic accessories	5730.2385
25	Naypyitaw	January	Fashion accessories	6385.0290
26	Naypyitaw	January	Food and beverages	8315.0235
27	Naypyitaw	January	Health and beauty	6020.6895
28	Naypyitaw	January	Home and lifestyle	5594.7045
29	Naypyitaw	January	Sports and travel	8388.9960
30	Naypyitaw	March	Electronic accessories	7764.8550
31	Naypyitaw	March	Fashion accessories	7475.9265
32	Naypyitaw	March	Food and beverages	8060.5140
33	Naypyitaw	March	Health and beauty	4764.2910
34	Naypyitaw	March	Home and lifestyle	5297.9430
35	Naypyitaw	March	Sports and travel	3835.5135
36	Yangon	February	Electronic accessories	5202.7710
37	Yangon	February	Fashion accessories	5173.6335
38	Yangon	February	Food and beverages	7054.2255
39	Yangon	February	Health and beauty	2915.4825
40	Yangon	February	Home and lifestyle	4771.6305
41	Yangon	February	Sports and travel	4742.3775
42	Yangon	January	Electronic accessories	6401.2725
43	Yangon	January	Fashion accessories	6847.4910
44	Yangon	January	Food and beverages	4646.2290
45	Yangon	January	Health and beauty	3962.5950
46	Yangon	January	Home and lifestyle	10313.5935
47	Yangon	January	Sports and travel	6509.9475
48	Yangon	March	Electronic accessories	6713.0700
49	Yangon	March	Fashion accessories	4311.3840
50	Yangon	March	Food and beverages	5462.6460
51	Yangon	March	Health and beauty	5719.6755
52	Yangon	March	Home and lifestyle	7331.9715
53	Yangon	March	Sports and travel	8120.3745

[126]:
```
Mg=px.bar(Month_sales, x="Month", y="Total", color="Product line"␣
↪,facet_row="City", barmode='group', category_orders={"Month":['January'␣
↪,'February' , 'March']},
          title="MONTHLY SALES OF PRODUCT LINE" ,height=1200)

Mg.update_xaxes(showticklabels=True)
Mg.show()
```

[135]:
```
Month_sales=data.groupby(['City','Month' ,'Product line' ])['Total'].sum()
Month_sales
```

[135]:
```
City       Month     Product line
Mandalay   February  Electronic accessories     6686.2530
                     Fashion accessories        6137.1135
                     Food and beverages         5554.8150
                     Health and beauty          5856.4275
                     Home and lifestyle         4659.8475
                     Sports and travel          5529.8145
           January   Electronic accessories     6699.7770
                     Fashion accessories        6112.5960
                     Food and beverages         6609.2775
                     Health and beauty          6399.8865
                     Home and lifestyle         4586.4420
                     Sports and travel          6768.0795
           March     Electronic accessories     3665.4135
                     Fashion accessories        4163.6070
                     Food and beverages         3050.7960
                     Health and beauty          7724.3460
                     Home and lifestyle         8302.8750
                     Sports and travel          7690.3050
Naypyitaw  February  Electronic accessories     5473.8810
                     Fashion accessories        7699.1145
                     Food and beverages         7391.3175
                     Health and beauty          5830.3455
                     Home and lifestyle         3002.9055
                     Sports and travel          3537.4185
           January   Electronic accessories     5730.2385
                     Fashion accessories        6385.0290
                     Food and beverages         8315.0235
                     Health and beauty          6020.6895
                     Home and lifestyle         5594.7045
                     Sports and travel          8388.9960
           March     Electronic accessories     7764.8550
                     Fashion accessories        7475.9265
                     Food and beverages         8060.5140
                     Health and beauty          4764.2910
                     Home and lifestyle         5297.9430
```

```
                      Sports and travel          3835.5135
Yangon     February   Electronic accessories     5202.7710
                      Fashion accessories        5173.6335
                      Food and beverages         7054.2255
                      Health and beauty          2915.4825
                      Home and lifestyle         4771.6305
                      Sports and travel          4742.3775
           January    Electronic accessories     6401.2725
                      Fashion accessories        6847.4910
                      Food and beverages         4646.2290
                      Health and beauty          3962.5950
                      Home and lifestyle        10313.5935
                      Sports and travel          6509.9475
           March      Electronic accessories     6713.0700
                      Fashion accessories        4311.3840
                      Food and beverages         5462.6460
                      Health and beauty          5719.6755
                      Home and lifestyle         7331.9715
                      Sports and travel          8120.3745
Name: Total, dtype: float64
```

[137]:
```
Month_sales.groupby(['City','Month'])['Total'].sum()
```

[137]:
```
City       Month
Mandalay   February    34424.2710
           January     37176.0585
           March       34597.3425
Naypyitaw  February    32934.9825
           January     40434.6810
           March       37199.0430
Yangon     February    29860.1205
           January     38681.1285
           March       37659.1215
Name: Total, dtype: float64
```

[138]:
```
sales_df=Month_sales.groupby(['City','Month'])['Total'].sum().
  ↪reset_index(name="Revenue")
sales_df
```

[138]:

	City	Month	Revenue
0	Mandalay	February	34424.2710
1	Mandalay	January	37176.0585
2	Mandalay	March	34597.3425
3	Naypyitaw	February	32934.9825
4	Naypyitaw	January	40434.6810
5	Naypyitaw	March	37199.0430
6	Yangon	February	29860.1205

```
7     Yangon   January  38681.1285
8     Yangon     March  37659.1215
```

[139]:
```
sales_df['Month']=pd.Categorical(sales_df['Month'] ,categories=['January' ,'February' ,'March'] , ordered =True)
```

[140]:
```
final_sales=sales_df.groupby(['City','Month','Revenue'] ,observed=True).size().reset_index()
final_sales
```

[140]:
```
        City     Month     Revenue  0
0   Mandalay   January  37176.0585  1
1   Mandalay  February  34424.2710  1
2   Mandalay     March  34597.3425  1
3  Naypyitaw   January  40434.6810  1
4  Naypyitaw  February  32934.9825  1
5  Naypyitaw     March  37199.0430  1
6     Yangon   January  38681.1285  1
7     Yangon  February  29860.1205  1
8     Yangon     March  37659.1215  1
```

[141]:
```
final_sales=sales_df.groupby(['City','Month','Revenue'] ,observed=True).size().reset_index().drop(columns=0)
final_sales
```

[141]:
```
        City     Month     Revenue
0   Mandalay   January  37176.0585
1   Mandalay  February  34424.2710
2   Mandalay     March  34597.3425
3  Naypyitaw   January  40434.6810
4  Naypyitaw  February  32934.9825
5  Naypyitaw     March  37199.0430
6     Yangon   January  38681.1285
7     Yangon  February  29860.1205
8     Yangon     March  37659.1215
```

[142]:
```
sg2=px.line(final_sales,y='Revenue' , x='Month' ,title='MONTHLY REVENUE OF CITIES' ,
       markers=True, height=600, color='City')
sg2.update_xaxes(showticklabels=True)
sg2.show()
```

8 REVENUE PREDICTION FOR APRIL MONTH

[143]:
```
final_sales
```

[143]:

	City	Month	Revenue
0	Mandalay	January	37176.0585
1	Mandalay	February	34424.2710
2	Mandalay	March	34597.3425
3	Naypyitaw	January	40434.6810
4	Naypyitaw	February	32934.9825
5	Naypyitaw	March	37199.0430
6	Yangon	January	38681.1285
7	Yangon	February	29860.1205
8	Yangon	March	37659.1215

[174]:
```
pvt_table=final_sales.pivot_table(index='City' , columns='Month' ,␣
 ↪values='Revenue' , observed=True)

pvt_table
```

[174]:

Month	January	February	March
City			
Mandalay	37176.0585	34424.2710	34597.3425
Naypyitaw	40434.6810	32934.9825	37199.0430
Yangon	38681.1285	29860.1205	37659.1215

[175]:
```
X=pvt_table[['January' ,'February' ,'March']] # X=feature
y=pvt_table['March'] # y == target

model= LinearRegression()
model.fit(X,y)

pvt_table['April_Prediction']=model.predict(X)

sg_predict=pvt_table[['January' ,'February' ,'March','April_Prediction']]
sg_predict
```

[175]:

Month	January	February	March	April_Prediction
City				
Mandalay	37176.0585	34424.2710	34597.3425	34597.3425
Naypyitaw	40434.6810	32934.9825	37199.0430	37199.0430
Yangon	38681.1285	29860.1205	37659.1215	37659.1215

[176]:
```
pvt_table = pvt_table.reset_index()

pvt_table
```

[176]:

Month	City	January	February	March	April_Prediction
0	Mandalay	37176.0585	34424.2710	34597.3425	34597.3425
1	Naypyitaw	40434.6810	32934.9825	37199.0430	37199.0430
2	Yangon	38681.1285	29860.1205	37659.1215	37659.1215

[177]:
```
pvt_table[['City', 'January', 'February', 'March']]
```

[177]:

Month	City	January	February	March
0	Mandalay	37176.0585	34424.2710	34597.3425
1	Naypyitaw	40434.6810	32934.9825	37199.0430
2	Yangon	38681.1285	29860.1205	37659.1215

[178]:
```
sg_long = pd.melt(
    pvt_table,
    id_vars=['City'],
    value_vars=['January', 'February', 'March', 'April_Prediction'],
    var_name='Month',
    value_name='Revenue'
)

sg_long
```

[178]:

	City	Month	Revenue
0	Mandalay	January	37176.0585
1	Naypyitaw	January	40434.6810
2	Yangon	January	38681.1285
3	Mandalay	February	34424.2710
4	Naypyitaw	February	32934.9825
5	Yangon	February	29860.1205
6	Mandalay	March	34597.3425
7	Naypyitaw	March	37199.0430
8	Yangon	March	37659.1215
9	Mandalay	April_Prediction	34597.3425
10	Naypyitaw	April_Prediction	37199.0430
11	Yangon	April_Prediction	37659.1215

[185]:
```
sg_predict_graph=px.line(sg_long , x='Month' ,y='Revenue' ,color='City' ,␣
 ↪markers=True)
sg_predict_graph.show()
```

9 NOW I'LL BE PUTTING ALL OF HE GRAPHS IN THE FORM OF DASBOARD

9.0.1 installing pandoc to export it as PDF

[ ]:
```

```