

Import required libraries

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
```

Read the csv file

```
In [2]: #filepath="D:/project/supermarket_sales - Sheet1.csv"
data=pd.read_csv('D:/project/supermarket_sales - Sheet1.csv')
```

Top 10 records

```
In [40]: data.head(10)
```

```
Out[40]:
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785
5	699-14-3026	C	Naypyitaw	Normal	Male	Electronic accessories	85.39	7	29.8865	627.6165
6	355-53-5943	A	Yangon	Member	Female	Electronic accessories	68.84	6	20.6520	433.6920
7	315-22-5665	C	Naypyitaw	Normal	Female	Home and lifestyle	73.56	10	36.7800	772.3800
8	665-32-9167	A	Yangon	Member	Female	Health and beauty	36.26	2	3.6260	76.1460
9	692-92-5582	B	Mandalay	Member	Female	Food and beverages	54.84	3	8.2260	172.7460

Last 10 records

In [182...]: `data.tail(10)`

Out[182...]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total
990	886-18-2897	A	Yangon	Normal	Female	Food and beverages	56.56	5	14.1400	296.9400
991	602-16-6955	B	Mandalay	Normal	Female	Sports and travel	76.60	10	38.3000	804.3000
992	745-74-0715	A	Yangon	Normal	Male	Electronic accessories	58.03	2	5.8030	121.8630
993	690-01-6631	B	Mandalay	Normal	Male	Fashion accessories	17.49	10	8.7450	183.6450
994	652-49-6720	C	Naypyitaw	Member	Female	Electronic accessories	60.95	1	3.0475	63.9975
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	42.3675
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4900
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.5920	33.4320
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.2910	69.1110
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190	649.2990

Columns present in the data

In [19]: `data.columns`

Out[19]: 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')

Basic information about the dataset

In [20]: `data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
#   Column                                Non-Null Count  Dtype
#   ...
#   Rating                                1000 non-null  object
```

```

-----
0  Invoice ID          1000 non-null object
1  Branch             1000 non-null object
2  City               1000 non-null object
3  Customer type      1000 non-null object
4  Gender             1000 non-null object
5  Product line       1000 non-null object
6  Unit price         1000 non-null float64
7  Quantity           1000 non-null int64
8  Tax 5%             1000 non-null float64
9  Total              1000 non-null float64
10 Date              1000 non-null object
11 Time              1000 non-null object
12 Payment           1000 non-null object
13 cogs              1000 non-null float64
14 gross margin percentage 1000 non-null float64
15 gross income       1000 non-null float64
16 Rating            1000 non-null float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB

```

Statistical information about the dataset

In [21]: `data.describe()`

Out[21]:

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

Check the number of Rows and Columns in the DataSet

In [22]: `data.shape`

Out[22]: (1000, 17)

Check for the Null Value count

In [23]: `data.isna().sum()`

Out[23]:

Invoice ID	0
Branch	0
City	0
Customer type	0
Gender	0
Product line	0
Unit price	0
Quantity	0
Tax 5%	0
Total	0

```
Date          0
Time          0
Payment       0
cogs         0
gross margin percentage  0
gross income  0
Rating        0
dtype: int64
```

-----Qualitative Data Analysis-----

Query - Date given in Dataset

```
In [24]: data['Date'].min()
```

```
Out[24]: '1/1/2019'
```

```
In [26]: data['Date'].max()
```

```
Out[26]: '3/9/2019'
```

Query - Number of Branches

```
In [27]: data['Branch'].unique()
```

```
Out[27]: array(['A', 'C', 'B'], dtype=object)
```

Query - Customer Type

```
In [28]: data['Customer type'].unique()
```

```
Out[28]: array(['Member', 'Normal'], dtype=object)
```

Query - Product categories

```
In [29]: data['Product line'].unique()
```

```
Out[29]: array(['Health and beauty', 'Electronic accessories',
                'Home and lifestyle', 'Sports and travel', 'Food and beverages',
                'Fashion accessories'], dtype=object)
```

Query - What are the Payment methods

```
In [30]: data['Payment'].unique()
```

```
Out[30]: array(['Ewallet', 'Cash', 'Credit card'], dtype=object)
```

Query - Cities mentioned

```
In [31]: data['City'].unique()
```

```
Out[31]: array(['Yangon', 'Naypyitaw', 'Mandalay'], dtype=object)
```

-----Quantitative Analysis-----

Query - Citywise sales

```
In [32]: data.groupby('City')['Total'].sum()
```

```
Out[32]: City
Mandalay      106197.6720
Naypyitaw     110568.7065
Yangon        106200.3705
Name: Total, dtype: float64
```

Query - Product wise sales

```
In [33]: data.groupby('Product line')['Total'].sum()
```

```
Out[33]: Product line
Electronic accessories      54337.5315
Fashion accessories         54305.8950
Food and beverages          56144.8440
Health and beauty           49193.7390
Home and lifestyle           53861.9130
Sports and travel           55122.8265
Name: Total, dtype: float64
```

Query - Number of products purchased by Branch

```
In [36]: print(data.groupby('Branch')['Quantity'].sum())
```

```
Branch
A      1859
B      1820
C      1831
Name: Quantity, dtype: int64
```

Query - Average rating for Branches

```
In [37]: data.groupby('Branch')['Rating'].mean()
```

```
Out[37]: Branch
A      7.027059
B      6.818072
C      7.072866
Name: Rating, dtype: float64
```

Query - Purchases by different payment methods

```
In [39]: data.groupby('Payment')['Quantity'].sum()
```

```
Out[39]: Payment
Cash      1896
Credit card  1722
Ewallet    1892
Name: Quantity, dtype: int64
```

Statistical Analysis

Query - Average Rating Given by Customers

```
In [65]: rating=data['Rating'].mean()
print("Average Rating given is {} out of 10".format(rating))
```

Average Rating given is 6.972700000000003 out of 10

Query - What is the total Cost of Goods Sold?

```
In [61]: print('Total COGS is $ {}'.format(data['cogs'].sum()))
```

Total COGS is \$ 307587.38

Query - Total Amount Collected as Tax

```
In [66]: print("Total TAX AMOUNT Collected $ {}".format(data['Tax 5%'].sum()))
```

Total TAX AMOUNT Collected \$ 15379.368999999999

Query - Average amount customers spend

```
In [70]: print("Average Amount Customers Spend $ {}".format(data['Total'].mean()))
```

Average Amount Customers Spend \$ 322.96674900000005

Query - Standard deviation on the Gross Margin Percentage

```
In [73]: data['gross margin percentage'].std()
```

Out[73]: 6.22035989578277e-14

Query - Variance deviation on the Gross Margin Percentage

```
In [74]: data['gross margin percentage'].var()
```

Out[74]: 3.869287723306264e-27

Query - Standard deviation on the Unit Price

```
In [78]: data['Unit price'].std()
```

Out[78]: 26.494628347919768

Query - Standard Deviation on Rating

```
In [82]: data['Rating'].std()
```

Out[82]: 1.718580294379123

Query - Average Gross Income by Branches

```
In [84]: data.groupby('Branch')['gross income'].mean()
```

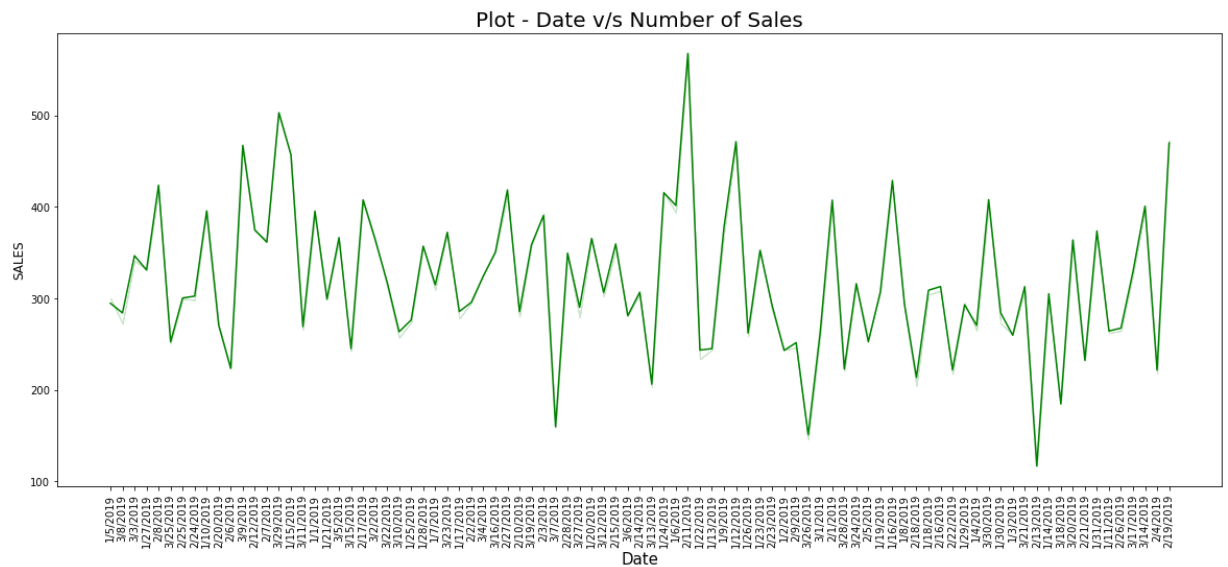
Out[84]: Branch
A 14.874001
B 15.232024
C 16.052367
Name: gross income, dtype: float64

```
In [71]: data['Quantity'].mode()
```

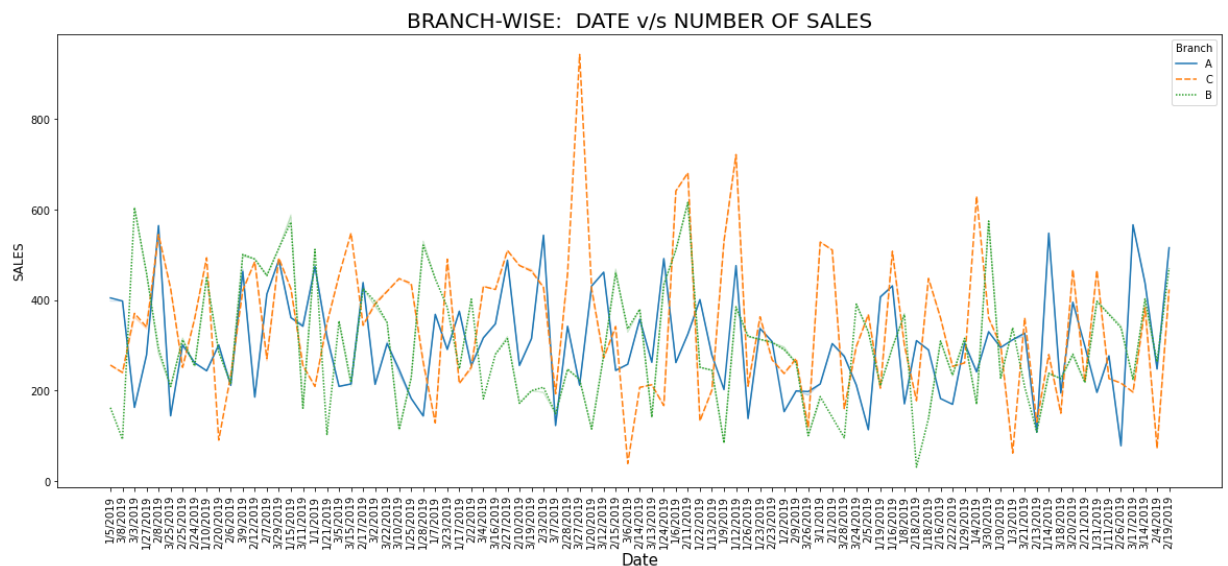
Out[71]: 0 10
dtype: int64

Data Visualisation

```
In [3]: plt.figure(figsize=(20,8))  
sb.lineplot(x='Date',y='Total',data=data,color='g',ci=False)  
plt.xticks(rotation='vertical')  
plt.title('Plot - Date v/s Number of Sales',size=20)  
plt.xlabel('Date',size=15)  
plt.ylabel('SALES',size=12)  
plt.show()
```



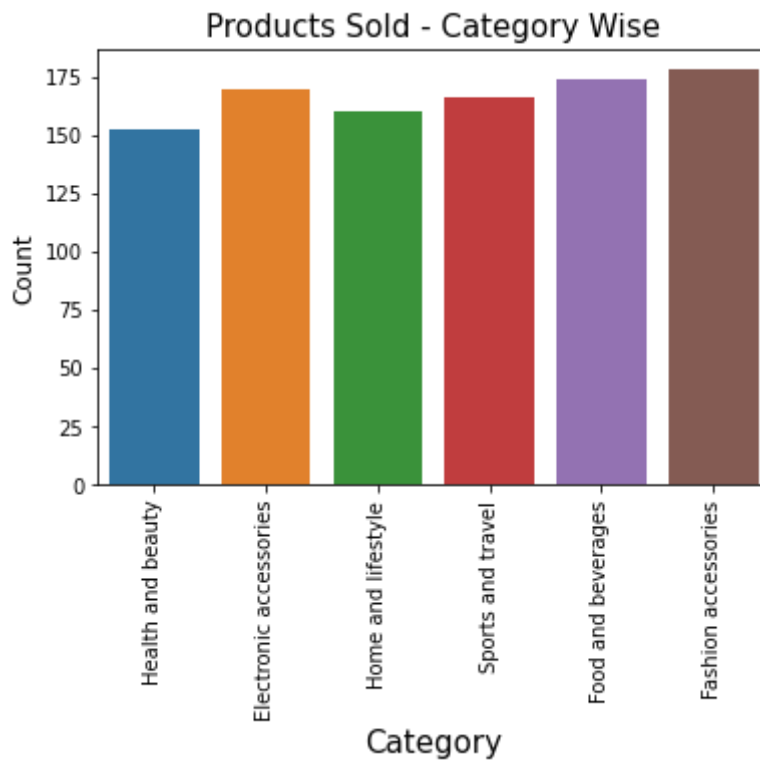
```
In [4]: plt.figure(figsize=(20,8))
sb.lineplot(x='Date',y='Total',data=data,color='g',ci=True,hue='Branch',style='Branch')
plt.xticks(rotation='vertical')
plt.title('BRANCH-WISE: DATE v/s NUMBER OF SALES',size=20)
plt.xlabel('Date',size=15)
plt.ylabel('SALES',size=12)
plt.show()
```



```
In [115]: sb.countplot('Product line',data=data)
plt.title('Products Sold - Category Wise',size=15)
plt.xlabel('Category',size=15)
plt.ylabel('Count',size=12)
plt.xticks(rotation='vertical')
plt.show()
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

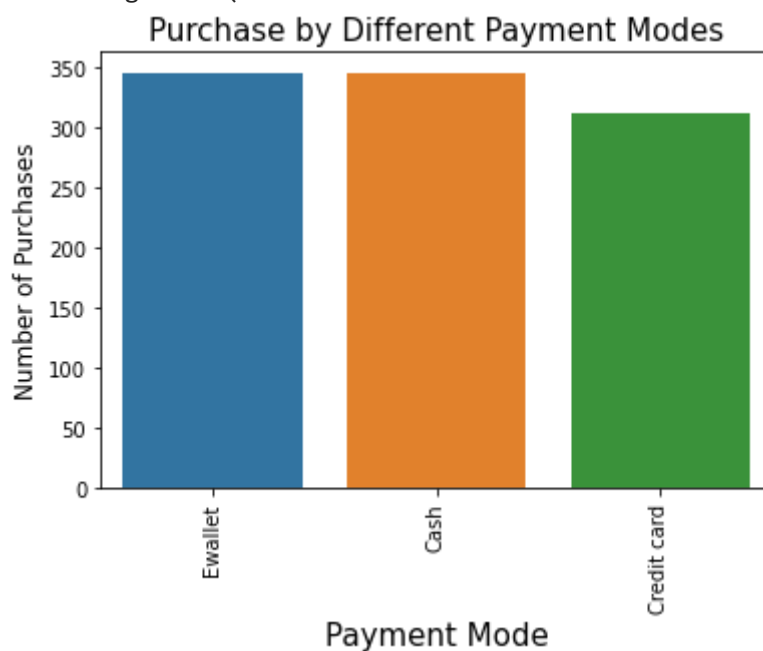
```
warnings.warn(
```



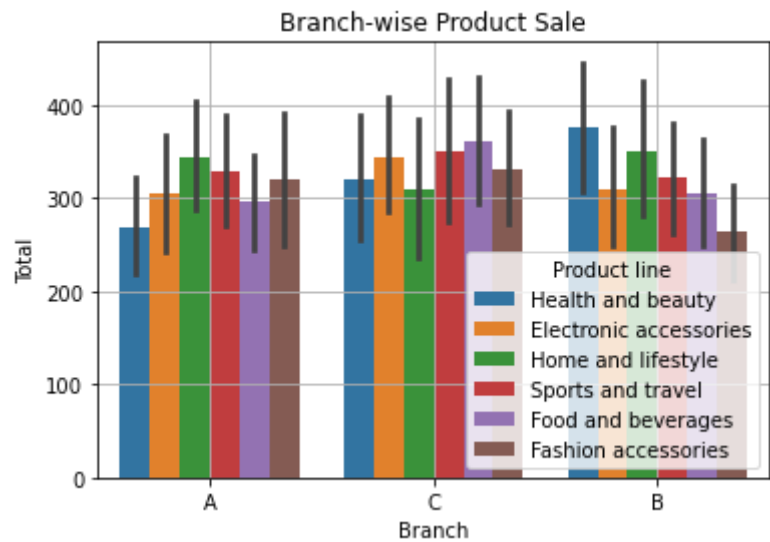
```
In [5]: sb.countplot('Payment',data=data)
plt.title('Purchase by Different Payment Modes',size=15)
plt.xlabel('Payment Mode',size=15)
plt.ylabel('Number of Purchases',size=12)
plt.xticks(rotation='vertical')
plt.show()
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(



```
In [133... sb.barplot(x='Branch',y="Total",data=data,hue='Product line')
plt.title('Branch-wise Product Sale')
plt.grid()
plt.show()
```

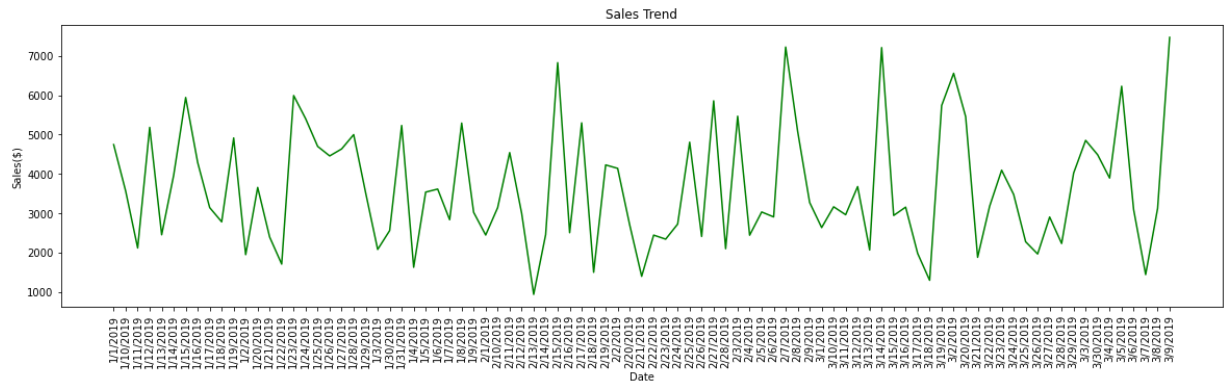
```
In [69]: sales=data.groupby('Date')['Total'].sum().reset_index()
sales
```

Out[69]:

	Date	Total
0	1/1/2019	4745.1810
1	1/10/2019	3560.9490
2	1/11/2019	2114.9625
3	1/12/2019	5184.7635
4	1/13/2019	2451.2040
...
84	3/5/2019	6230.8785
85	3/6/2019	3092.5965
86	3/7/2019	1438.2585
87	3/8/2019	3125.3880
88	3/9/2019	7474.0470

89 rows × 2 columns

```
In [70]: plt.figure(figsize=(20,5))
sb.lineplot(x='Date',y='Total',data=sales,color='g')
plt.title('Sales Trend')
plt.ylabel('Sales($)')
plt.xticks(rotation='vertical')
plt.show()
```

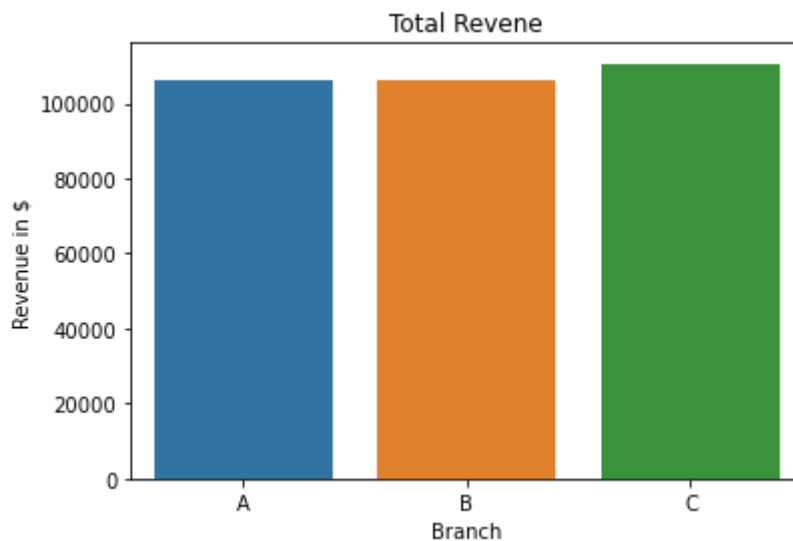


```
In [71]: rev=data.groupby('Branch').sum().reset_index()
rev
```

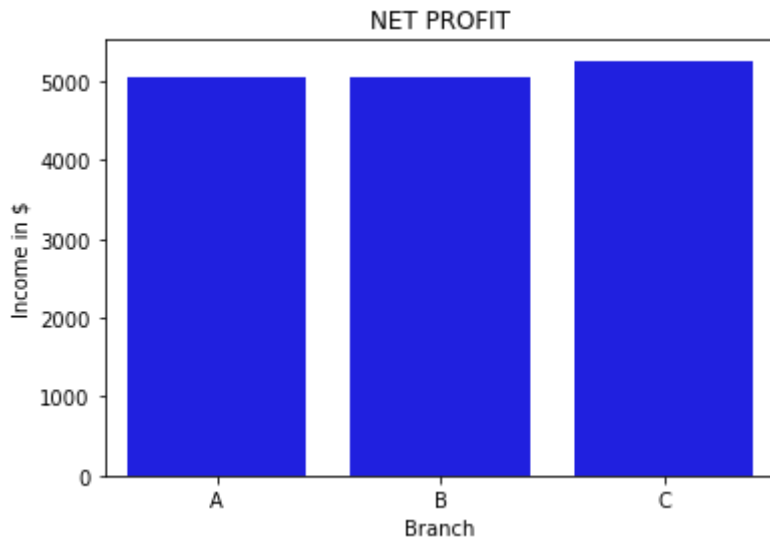
Out[71]:

	Branch	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
0	A	18625.49	1859	5057.1605	106200.3705	101143.21	1619.047619	5057.1605	2389.2
1	B	18478.88	1820	5057.0320	106197.6720	101140.64	1580.952381	5057.0320	2263.6
2	C	18567.76	1831	5265.1765	110568.7065	105303.53	1561.904762	5265.1765	2319.9

```
In [72]: sb.barplot(x='Branch',y='Total',data=rev)
plt.ylabel("Revenue in $")
plt.title('Total Revenue')
plt.show()
```

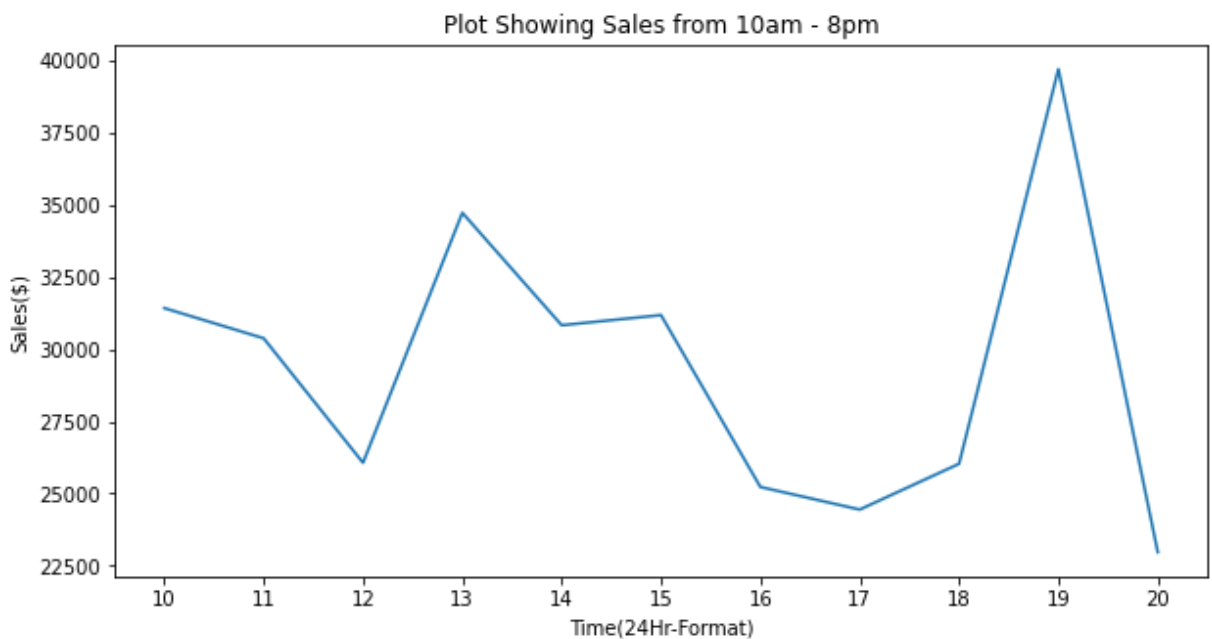


```
In [73]: sb.barplot(x='Branch',y='gross income',data=rev,color='b')
plt.ylabel("Income in $")
plt.title('NET PROFIT')
plt.show()
```



```
In [74]: data['hour']=data['Time'].apply(lambda x: x.split(':')[0])
hour=data.groupby('hour')['Total'].sum().reset_index().sort_values(by='hour')
```

```
In [75]: plt.figure(figsize=(10,5))
sb.lineplot(x=hour['hour'],y=hour['Total'])
plt.title('Plot Showing Sales from 10am - 8pm')
plt.xlabel('Time(24Hr-Format)')
plt.ylabel('Sales($)')
plt.show()
```

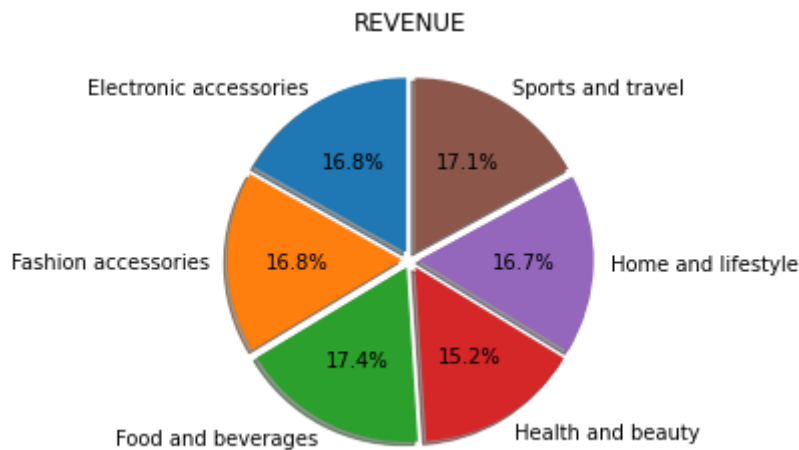


```
In [79]: pie=data.groupby('Product line')['Total'].sum().reset_index()
```

```
In [82]: slices=list(pie['Total'])
name=list(pie['Product line'])
print(slices, name)
```

```
[54337.531500000005, 54305.895, 56144.844000000005, 49193.739000000016, 53861.913000
00001, 55122.826499999996] ['Electronic accessories', 'Fashion accessories', 'Food a
nd beverages', 'Health and beauty', 'Home and lifestyle', 'Sports and travel']
```

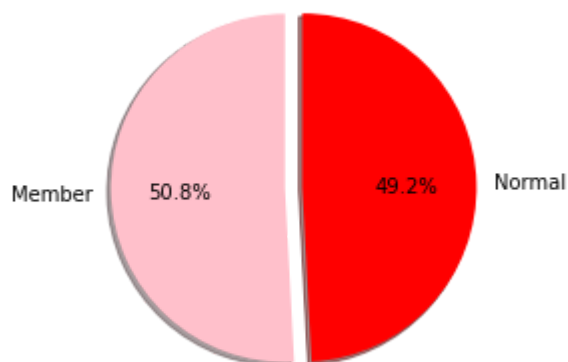
```
In [103... plt.pie(slices,labels=name,shadow=True,startangle=90,explode=[0.05,0.05,0.05,0.05,0.
plt.title('REVENUE')
#plt.axis('auto')
plt.show()
```



```
In [93]: c_type=data.groupby('Customer type')['Total'].sum().reset_index()
        slices1=list(c_type['Total'])
        name1=list(c_type['Customer type'])
```

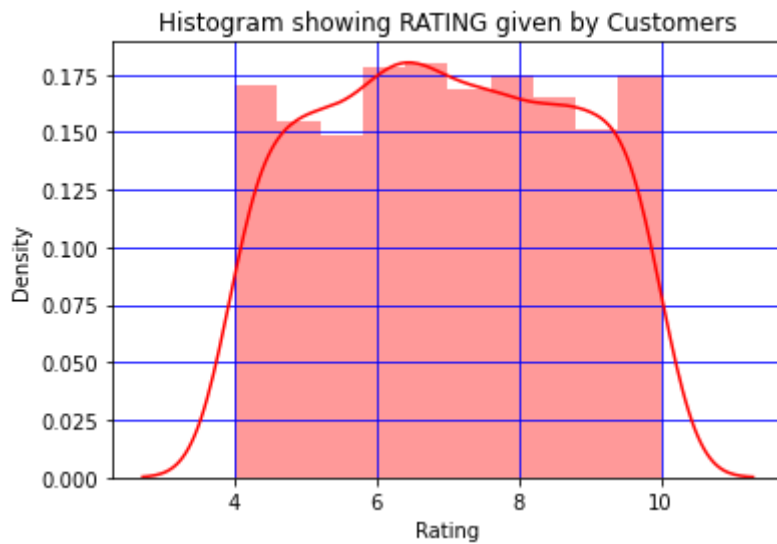
```
In [104...]: plt.pie(slices1,labels=name1,shadow=True,colors=['pink','red'],explode=[0.1,0],start
        plt.title('Revenue shares from different customers')
        plt.show()
```

Revenue shares from different customers

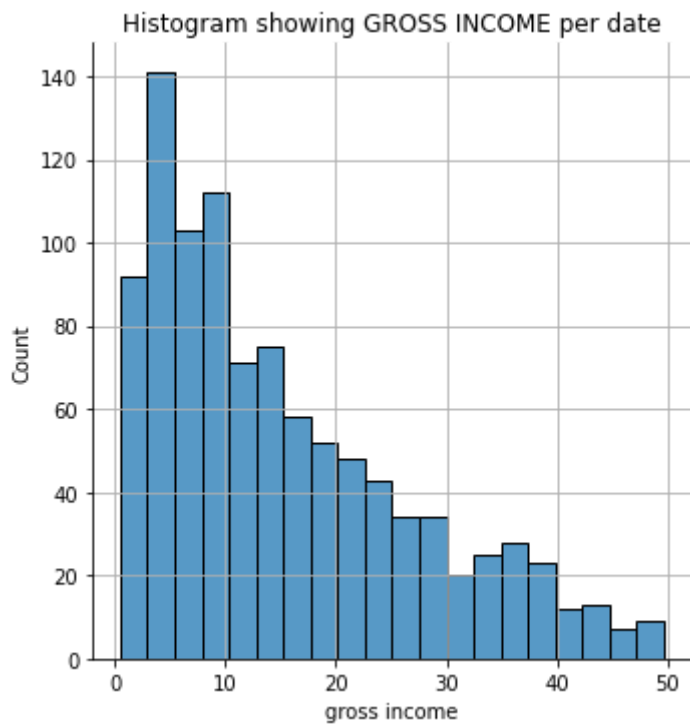


```
In [23]: sb.distplot(data['Rating'],bins=10,color='r')
        plt.title('Histogram showing RATING given by Customers')
        plt.grid(color='b')
        plt.show()
```

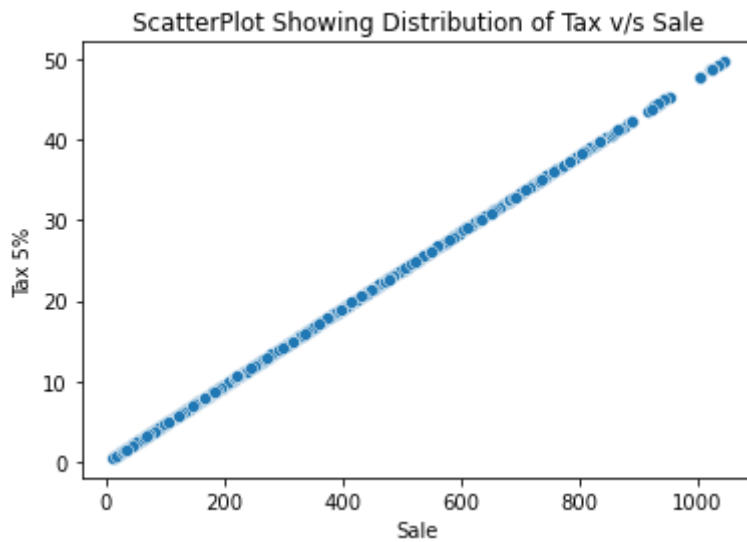
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
 warnings.warn(msg, FutureWarning)



```
In [181... sb.displot(data['gross income'],bins=20)
plt.title('Histogram showing GROSS INCOME per date')
plt.grid()
plt.show()
```

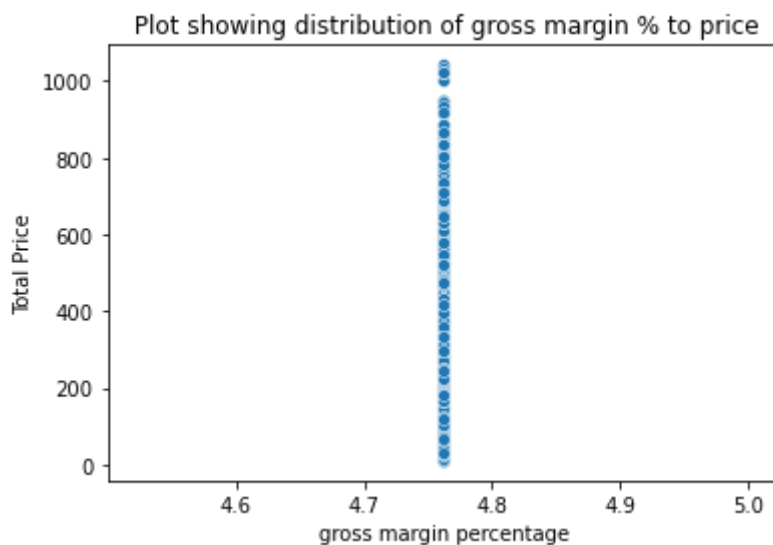


```
In [152... sb.scatterplot(x='Total',y="Tax 5%",data=data)
plt.title('ScatterPlot Showing Distribution of Tax v/s Sale')
plt.xlabel('Sale')
plt.show()
```

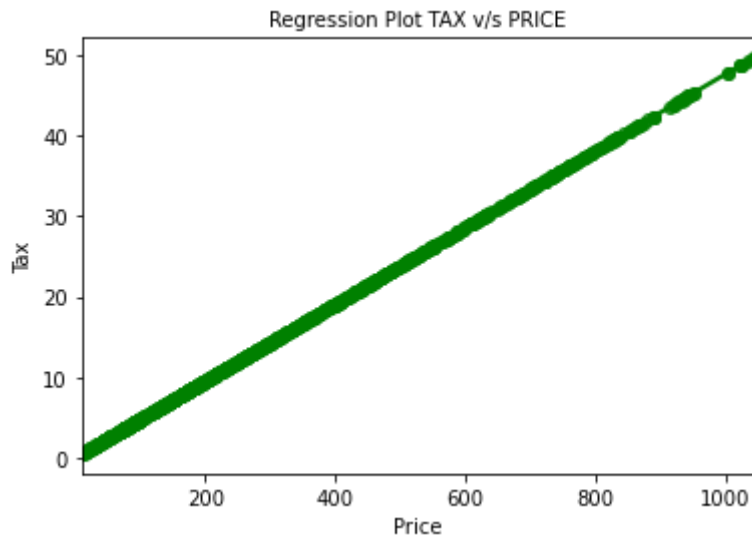


```
In [158... sb.scatterplot(x='gross margin percentage', y='Total', data=data)
plt.title(' Plot showing distribution of gross margin % to price')
plt.ylabel('Total Price')
```

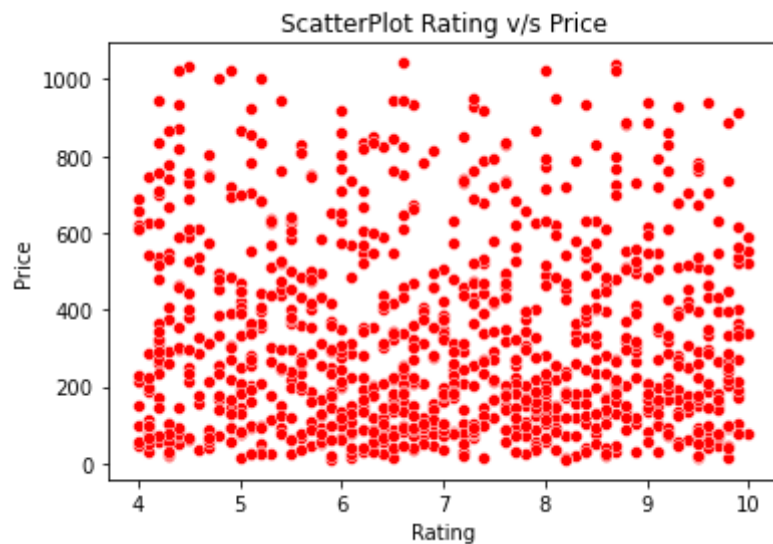
```
Out[158... Text(0, 0.5, 'Total Price')
```



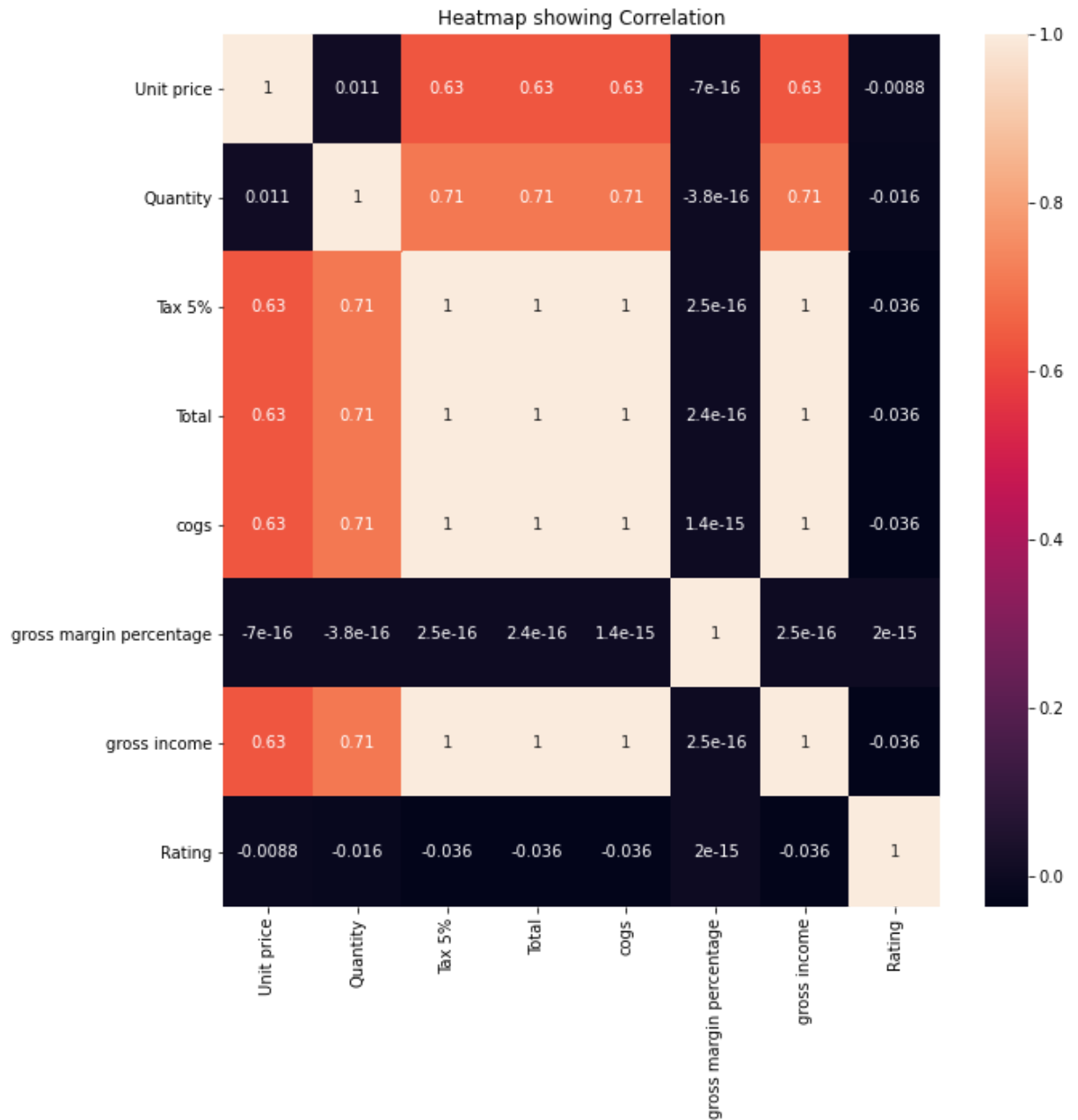
```
In [178... sb.regplot(y=data['Tax 5%'],x=data['Total'],color='g')
plt.title('Regression Plot TAX v/s PRICE',size=10)
plt.xlabel('Price')
plt.ylabel('Tax')
plt.show()
```



```
In [177... sb.scatterplot(x='Rating',y='Total',data=data,color='r')
plt.title('ScatterPlot Rating v/s Price')
plt.ylabel('Price')
plt.show()
```

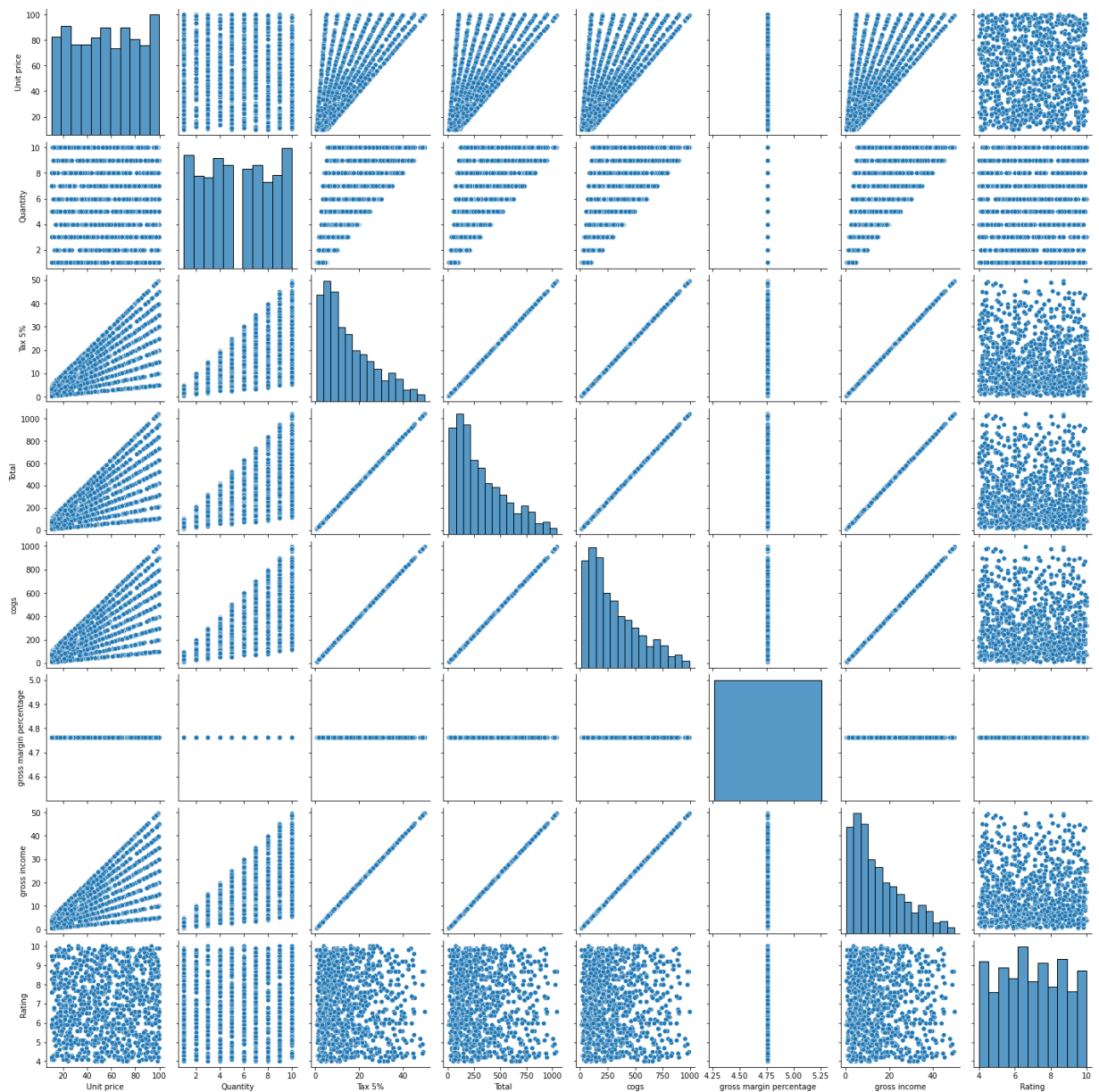


```
In [169... plt.figure(figsize=(10,10))
sb.heatmap(data.corr(),annot=True)
plt.title('Heatmap showing Correlation')
plt.show()
```



In [172... `sb.pairplot(data)`

Out[172... `<seaborn.axisgrid.PairGrid at 0x2cb3df95370>`



In [76]: `data.columns`

Out[76]: 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', 'hour'],
dtype='object')

In []: