importing libraries

In [78]:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler,MinMaxScaler

#Data Visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Reading CSV file

In [79]:

```
path ="D:\\Coursera IBM course\\final-project\\supermarket_sales.xls"
data =pd.read_csv(path)
```

In [80]:

```
data.head(5)
```

Out[80]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gros margi percentag
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7.0	26.1415	548.9715	1/5/19	13:08	Ewallet	522.83	4.76190
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5.0	3.8200	80.2200	3/8/19	10:29	Cash	76.40	4.76190
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7.0	16.2155	340.5255	3/3/19	13:23	Credit card	324.31	4.76190
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8.0	23.2880	489.0480	1/27/19	20:33	Ewallet	465.76	4.76190
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7.0	30.2085	634.3785	2/8/19	10:37	Ewallet	604.17	4.76190

Exploratory Data Analysis

In [81]:

```
data.shape
```

Out[81]:

```
(1003, 17)
```

In [82]:

```
#creating a list of columns
data.columns.tolist()
```

Out[82]:

```
['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']
```

In [83]:

```
data.describe().round()
```

Out[83]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
count	996.0	983.0	1003.0	1003.0	1003.0	1003.0	1003.0	1003.0
mean	56.0	6.0	15.0	323.0	308.0	5.0	15.0	7.0
std	27.0	3.0	12.0	246.0	234.0	0.0	12.0	2.0
min	10.0	1.0	1.0	11.0	10.0	5.0	1.0	4.0
25%	33.0	3.0	6.0	124.0	118.0	5.0	6.0	6.0
50%	55.0	5.0	12.0	254.0	242.0	5.0	12.0	7.0
75%	78.0	8.0	23.0	473.0	451.0	5.0	23.0	8.0
max	100.0	10.0	50.0	1043.0	993.0	5.0	50.0	10.0

In [84]:

```
#some more informations about dataset
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1003 entries, 0 to 1002
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Invoice ID               1003 non-null   object 
 1   Branch                   1003 non-null   object 
 2   City                     1003 non-null   object 
 3   Customer type            924 non-null    object 
 4   Gender                   1003 non-null   object 
 5   Product line             960 non-null    object 
 6   Unit price               996 non-null    float64
 7   Quantity                 983 non-null    float64
 8   Tax 5%                   1003 non-null   float64
 9   Total                    1003 non-null   float64
 10  Date                     1003 non-null   object 
 11  Time                     1003 non-null   object 
 12  Payment                  1003 non-null   object 
 13  cogs                     1003 non-null   float64
 14  gross margin percentage  1003 non-null   float64
 15  gross income             1003 non-null   float64
 16  Rating                   1003 non-null   float64
dtypes: float64(8), object(9)
memory usage: 133.3+ KB
```

In [85]:

```
#convert 'Date' and 'Time' from 'object' Type into 'datetime' type
data['Date']=pd.to_datetime(data['Date'])
data['Date'].dtype
```

Out[85]:

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

In [86]:

```
data['Time']=pd.to_datetime(data['Time'])
data['Time'].dtype
```

Out[86]:

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

checking missing values

In [87]:

```
null_val=data.isna().sum().sort_values(ascending=False)
```

In [88]:

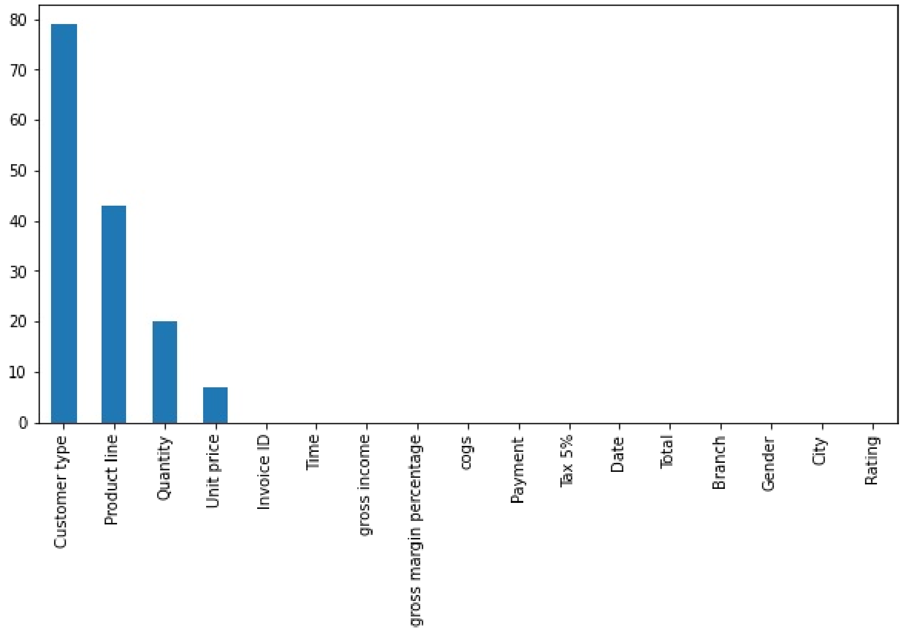
```
null_val
```

Out[88]:

```
Customer type              79
Product line               43
Quantity                   20
Unit price                  7
Invoice ID                  0
Time                        0
gross income                0
gross margin percentage     0
cogs                        0
Payment                     0
Tax 5%                      0
Date                        0
Total                       0
Branch                      0
Gender                      0
City                        0
Rating                      0
dtype: int64
```

In [89]:

```
plt.figure(figsize=(10,5))
null_val.plot(kind='bar')
plt.show()
```



In [90]:

```
# filling the missing values for the 'Object'type data set by the most freqenct (mode)
data['Customer type']=data['Customer type'].fillna(data['Customer type'].mode()[0])
data['Product line']=data['Product line'].fillna(data['Product line'].mode()[0])
```

In [91]:

```
# Drop NAN values
data.dropna(subset=['Unit price'],axis=0,inplace=True)
data.dropna(subset=['Quantity'],axis=0,inplace=True)
```

In [92]:

```
data.isna().sum()
```

Out[92]:

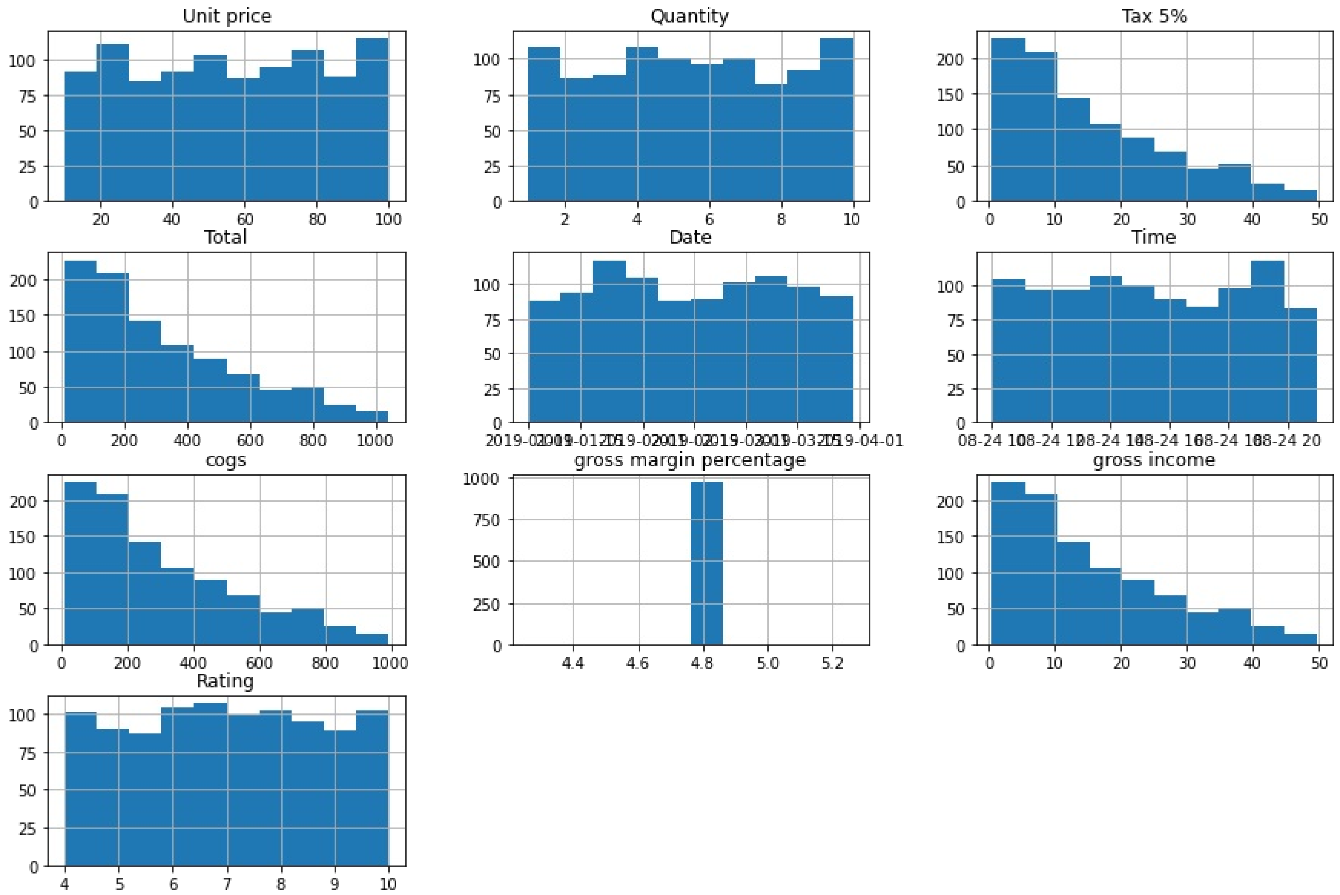
```
Invoice ID                 0
Branch                     0
City                       0
Customer type              0
Gender                     0
Product line               0
Unit price                 0
Quantity                   0
Tax 5%                     0
Total                      0
Date                       0
Time                       0
Payment                    0
cogs                       0
gross margin percentage    0
gross income               0
Rating                     0
dtype: int64
```

Data visualization

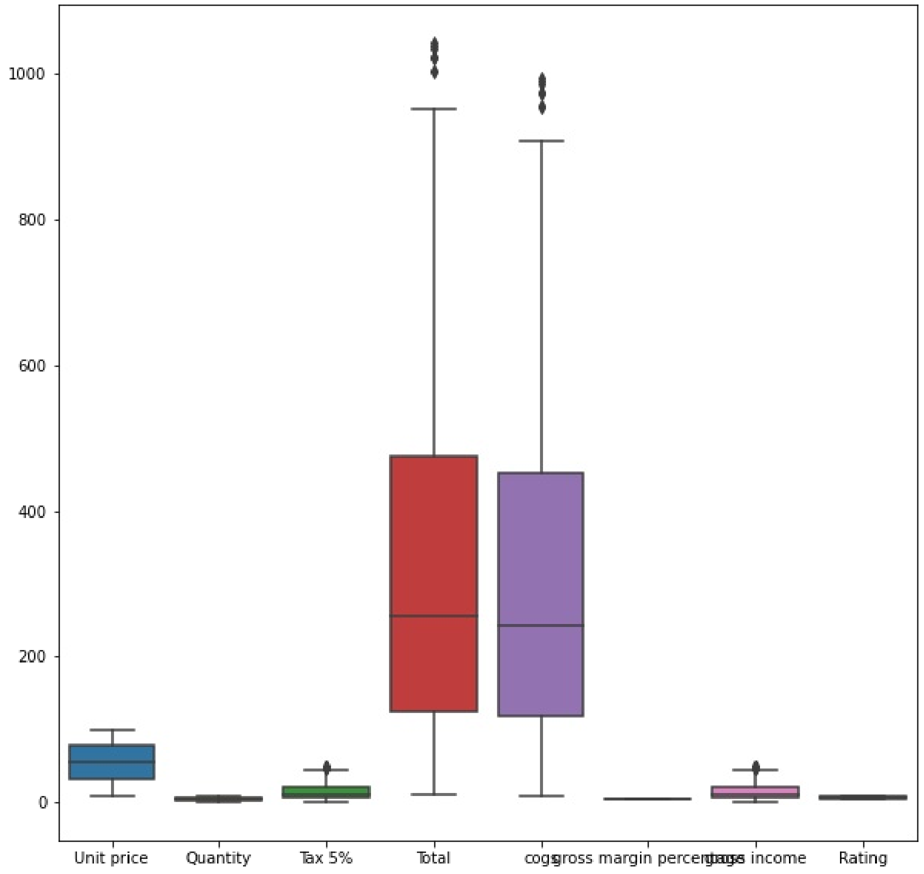
Outliers

In [93]:

```
data.hist(figsize=(15,10))
plt.show()
```



```
In [94]: plt.figure(figsize=(10,10))
         sns.boxplot(data=data)
         plt.show()
```



sales trend

```
In [95]: data['year_month']=data['Date'].apply(lambda x:x.strftime('%Y-%m'))
```

```
In [96]:  data['year_month'].head()
```

Out[96]:

```
0    2019-01
1    2019-03
2    2019-03
3    2019-01
4    2019-02
Name: year_month, dtype: object
```

In [97]:

```
data_temp=data.groupby('year_month').sum()['gross income'].reset_index()
```

In [98]:

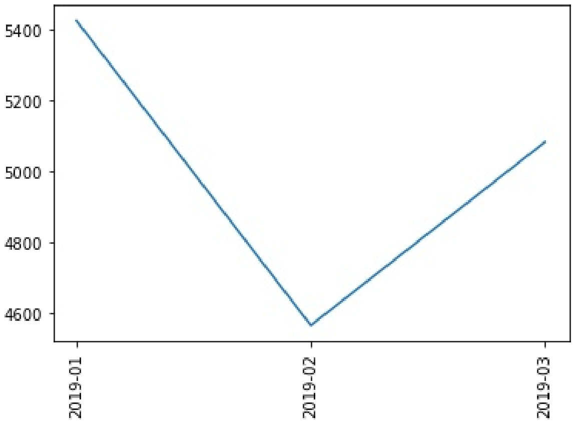
```
data_temp
```

Out[98]:

	year_month	gross income
0	2019-01	5425.4995
1	2019-02	4566.3460
2	2019-03	5083.5280

In [99]:

```
plt.plot(data_temp['year_month'],data_temp['gross income'])
plt.xticks(rotation='vertical')
plt.show()
```



In [100…

```
#the gross income for each branch
branch_count=data.groupby('Branch').sum()['gross income'].reset_index()
branch_count
```

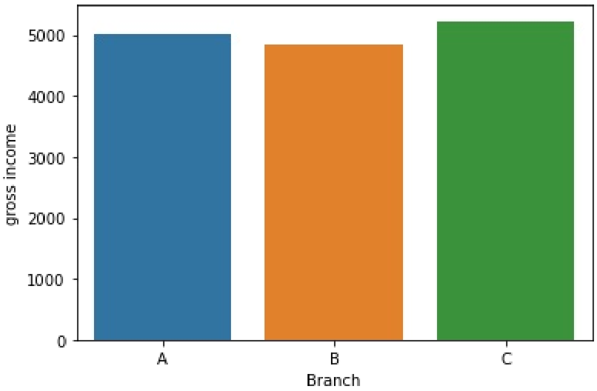
Out[100]:

	Branch	gross income
0	A	5019.1275
1	B	4831.9420
2	C	5224.3040

In [101…

```
sns.barplot(branch_count['Branch'],branch_count['gross income'])
plt.show()
```

```
C:\Users\Public\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variab
les as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [102…

```
#Branch Count
branch_count=data.groupby('Branch').count()['City']
branch_count
```

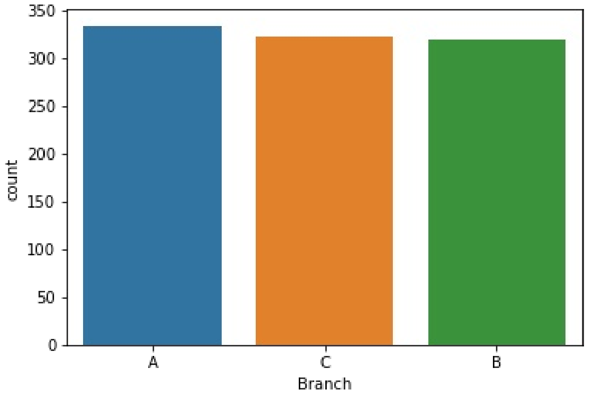
Out[102]:

```
Branch
A    334
B    319
C    323
Name: City, dtype: int64
```

In [103…

```
sns.countplot(data['Branch'])
sns.countplot(data['Branch'])
plt.show()
```

```
C:\Users\Public\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variab
le 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 [104…

```
#another example for count number of branches
```

In [105…

```
data.Branch.unique().tolist()
```

Out[105]:

```
['A', 'C', 'B']
```

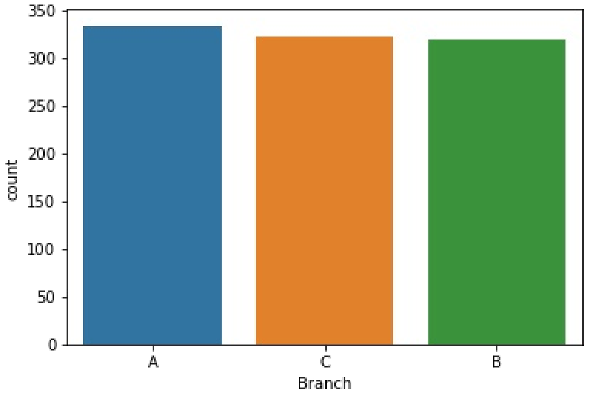
In [106…

```
A,C,B=data.Branch.value_counts()
print(f'A={A}')
print(f'C={C}')
print(f'B={B}')

sns.countplot(data['Branch'])
plt.show()
```

```
A=334
C=323
B=319
```

```
C:\Users\Public\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variab
le 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 [107…

```
most_paymethods=data.groupby('Payment').count()
```

In [108…

```
most_paymethods
```

Out[108]:

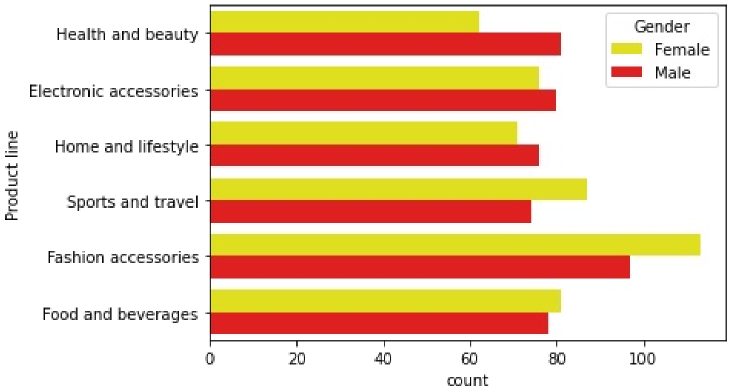
	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	cogs	gross margin percentage	gross income	Rating	y
Payment																	
Cash	336	336	336	336	336	336	336	336	336	336	336	336	336	336	336	336	
Credit card	305	305	305	305	305	305	305	305	305	305	305	305	305	305	305	305	
Ewallet	335	335	335	335	335	335	335	335	335	335	335	335	335	335	335	335	

In [109…

```
data.columns
```

Out[109]:
```
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', 'year_month'],
      dtype='object')
```

In [110…
```
sns.countplot(data=data,y='Product line',hue='Gender',palette=sns.color_palette(['yellow','red']))
plt.show()
```



Feature engineering

In [111…
```
data.head(5)
```

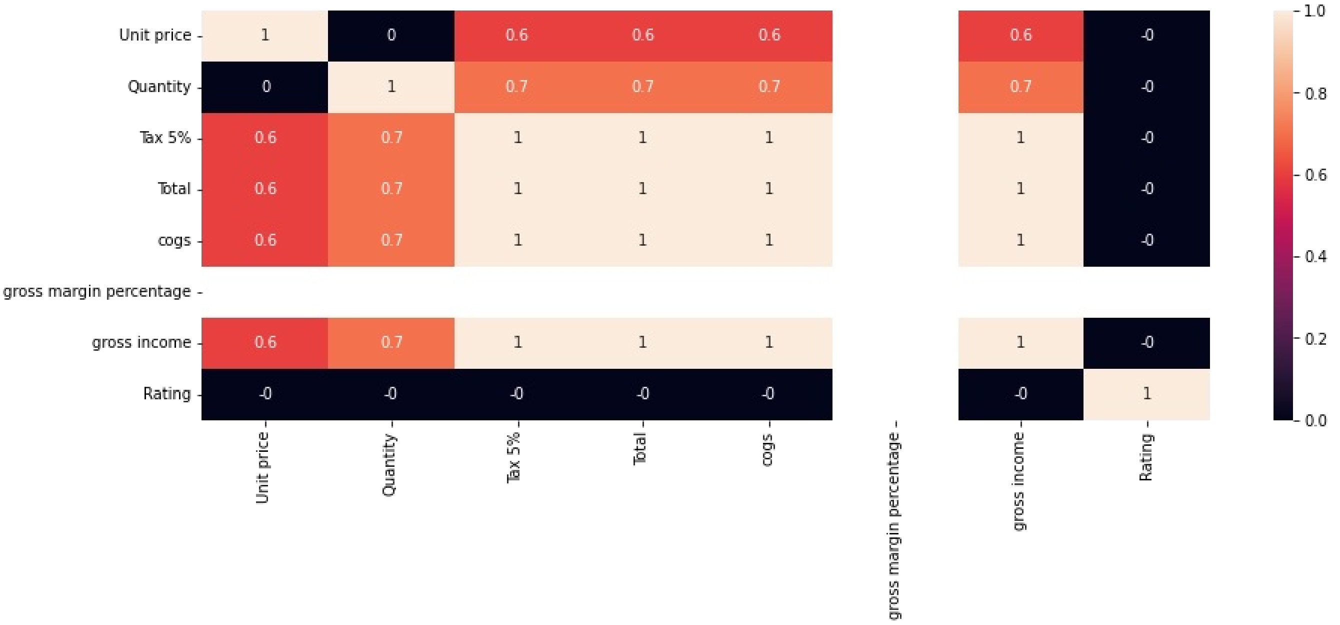
Out[111]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gı ma percent
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7.0	26.1415	548.9715	2019-01-05	2022-08-24 13:08:00	Ewallet	522.83	4.761
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5.0	3.8200	80.2200	2019-03-08	2022-08-24 10:29:00	Cash	76.40	4.761
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7.0	16.2155	340.5255	2019-03-03	2022-08-24 13:23:00	Credit card	324.31	4.761
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8.0	23.2880	489.0480	2019-01-27	2022-08-24 20:33:00	Ewallet	465.76	4.761
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7.0	30.2085	634.3785	2019-02-08	2022-08-24 10:37:00	Ewallet	604.17	4.761

In [112…
```
data_num=data.select_dtypes(['float64','int64'])
data_num_corr=data_num.corr()['gross income']
data_num_corr
```

Out[112]:
```
Unit price                 0.634655
Quantity                   0.708505
Tax 5%                     1.000000
Total                      1.000000
cogs                       1.000000
gross margin percentage         NaN
gross income               1.000000
Rating                    -0.034109
Name: gross income, dtype: float64
```

In [113…
```
plt.figure(figsize=(15,5))
sns.heatmap(np.round(data_num.corr(),1),annot=True)
plt.show()
```



Ploynomial Feature

In [114…
```
from sklearn.preprocessing import PolynomialFeatures
```

In [115…
```
pf=PolynomialFeatures(degree=2)
```

In [116…
```
features=['Unit price','Tax 5%','gross income']
pf.fit(data[features])
```

Out[116]:
```
PolynomialFeatures()
```

In [117…
```
pf.get_feature_names_out()
```

Out[117]:
```
array(['1', 'Unit price', 'Tax 5%', 'gross income', 'Unit price^2',
       'Unit price Tax 5%', 'Unit price gross income', 'Tax 5%^2',
       'Tax 5% gross income', 'gross income^2'], dtype=object)
```

In [118…
```
feat_array=pf.transform(data[features])
df=pd.DataFrame(feat_array,columns=pf.get_feature_names_out(input_features=features))
```

In [119…
```
df.head()
```

Out[119]:

	1	Unit price	Tax 5%	gross income	Unit price^2	Unit price Tax 5%	Unit price gross income	Tax 5%^2	Tax 5% gross income	gross income^2
0	1.0	74.69	26.1415	26.1415	5578.5961	1952.508635	1952.508635	683.378022	683.378022	683.378022
1	1.0	15.28	3.8200	3.8200	233.4784	58.369600	58.369600	14.592400	14.592400	14.592400
2	1.0	46.33	16.2155	16.2155	2146.4689	751.264115	751.264115	262.942440	262.942440	262.942440
3	1.0	58.22	23.2880	23.2880	3389.5684	1355.827360	1355.827360	542.330944	542.330944	542.330944
4	1.0	86.31	30.2085	30.2085	7449.4161	2607.295635	2607.295635	912.553472	912.553472	912.553472

One Hot Encoding (OHE)

In [120…
```
data.select_dtypes('object')
```

Out[120]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Payment	year_month
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	Ewallet	2019-01
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	Cash	2019-03
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	Credit card	2019-03
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	Ewallet	2019-01
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	Ewallet	2019-02
...	...	...	...	...	...	...	...	...
990	886-18-2897	A	Yangon	Normal	Female	Food and beverages	Credit card	2019-03
991	602-16-6955	B	Mandalay	Normal	Female	Sports and travel	Ewallet	2019-01
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	Cash	2019-02
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	Cash	2019-02
1000	849-09-3807	A	Yangon	Member	Female	Fashion accessories	Cash	2019-02

976 rows × 8 columns

In [121…

```
data.drop(['Invoice ID'],axis=1,inplace=True)
```

In [122…

```
columns=data.select_dtypes('object').columns.to_list()
columns
```

Out[122]:

```
['Branch',
 'City',
 'Customer type',
 'Gender',
 'Product line',
 'Payment',
 'year_month']
```

In [123…

```
data[columns].head().T
```

Out[123]:

	0	1	2	3	4
Branch	A	C	A	A	A
City	Yangon	Naypyitaw	Yangon	Yangon	Yangon
Customer type	Member	Normal	Normal	Member	Normal
Gender	Female	Female	Male	Male	Male
Product line	Health and beauty	Electronic accessories	Home and lifestyle	Health and beauty	Sports and travel
Payment	Ewallet	Cash	Credit card	Ewallet	Ewallet
year_month	2019-01	2019-03	2019-03	2019-01	2019-02

In [124…

```
#dummy variables
dv=pd.get_dummies(data,columns=columns,drop_first=True)
dv.head()
```

Out[124]:

	Unit price	Quantity	Tax 5%	Total	Date	Time	cogs	gross margin percentage	gross income	Rating	...	Gender_Male	Product line_Fashion accessories	Product line_Food and beverages	P line_ and l
0	74.69	7.0	26.1415	548.9715	2019-01-05	2022-08-24 13:08:00	522.83	4.761905	26.1415	9.1	...	0	0	0	
1	15.28	5.0	3.8200	80.2200	2019-03-08	2022-08-24 10:29:00	76.40	4.761905	3.8200	9.6	...	0	0	0	
2	46.33	7.0	16.2155	340.5255	2019-03-03	2022-08-24 13:23:00	324.31	4.761905	16.2155	7.4	...	1	0	0	
3	58.22	8.0	23.2880	489.0480	2019-01-27	2022-08-24 20:33:00	465.76	4.761905	23.2880	8.4	...	1	0	0	
4	86.31	7.0	30.2085	634.3785	2019-02-08	2022-08-24 10:37:00	604.17	4.761905	30.2085	5.3	...	1	0	0	

5 rows × 25 columns

Hypothesis Testing

In [125…

```
from scipy.stats import binom
```

In [126…

```
#the probabilty of getting 180 from 350
```

```
In [126…  #the probabilty of getting 180 from 350
          probabilty=1-binom.cdf(180,350,0.5)
          print(f'{str(round(probabilty*100 + 1 , 1))} %')
```

```
28.8 %
```

```
In [127…  #the probabilty of getting 90 from 150
          probabilty=1-binom.cdf(90,150,0.5)
          print(f'{str(round(probabilty*100 + 1 , 1))} %')
```

```
1.6 %
```

```
In [129…  #the probabilty of getting 80 from 450
          probabilty=1-binom.cdf(80,150,0.5)
          print(f'{str(round(probabilty*100 + 1 , 1))} %')
```

```
19.5 %
```

Dataset Summary

at first this data was covered most majority of what I have learned in this course but the quality of the dataset isn't so good we have need more information about customer ages , works and locations etc..

```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js