Initial Data Exploration

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pandas_profiling import ProfileReport
```

In [ ]:
```
supermarket_sales= pd.read_csv('supermarket_sales.csv')
```

In [ ]:
```
supermarket_sales.head()
```

Out[ ]:

	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

In [ ]:
```
supermarket_sales.tail()
```

Out[ ]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	T
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	42.3
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.5920	33.4
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.2910	69.1
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190	649.2

In [ ]:
```
supermarket_sales.columns
```

Out[ ]:
```
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')
```

In [ ]:
```
supermarket_sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Invoice ID              1000 non-null   object 
 1   Branch                  1000 non-null   object 
 2   City                    1000 non-null   object 
 3   Customer type           1000 non-null   object 
 4   Gender                  1000 non-null   object 
 5   Product line            1000 non-null   object 
 6   Unit price              1000 non-null   float64
 7   Quantity                1000 non-null   int64  
 8   Tax 5%                  1000 non-null   float64
 9   Total                   1000 non-null   float64
 10  Date                    1000 non-null   object 
 11  Time                    1000 non-null   object 
 12  Payment                 1000 non-null   object 
 13  cogs                    1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income            1000 non-null   float64
 16  Rating                  1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
```

In [ ]:
```
supermarket_sales['Date']=pd.to_datetime(supermarket_sales['Date'])
```

In [ ]:
```
supermarket_sales.dtypes
```

Out[ ]:
```
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
Date                    datetime64[ns]
Time                            object
Payment                         object
cogs                           float64
gross margin percentage        float64
gross income                   float64
Rating                         float64
dtype: object
```

In [ ]:
```
supermarket_sales.set_index('Date', inplace=True)
```

In [ ]:
```
supermarket_sales.head()
```

Out[ ]:

Date	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	
2019-01-05	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.
2019-03-08	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.
2019-03-03	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.
2019-01-27	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.
2019-02-08	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.

In [ ]:
```
supermarket_sales.describe()
```

Out[ ]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gros incom
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1.000000e+03	1000.00000
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905e+00	15.37936
std	26.494628	2.923431	11.708825	245.885335	234.17651	6.131498e-14	11.70882
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905e+00	0.50850
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905e+00	5.92487
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905e+00	12.08800
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905e+00	22.44525
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905e+00	49.65000

UNIVARIATE ANALYSIS

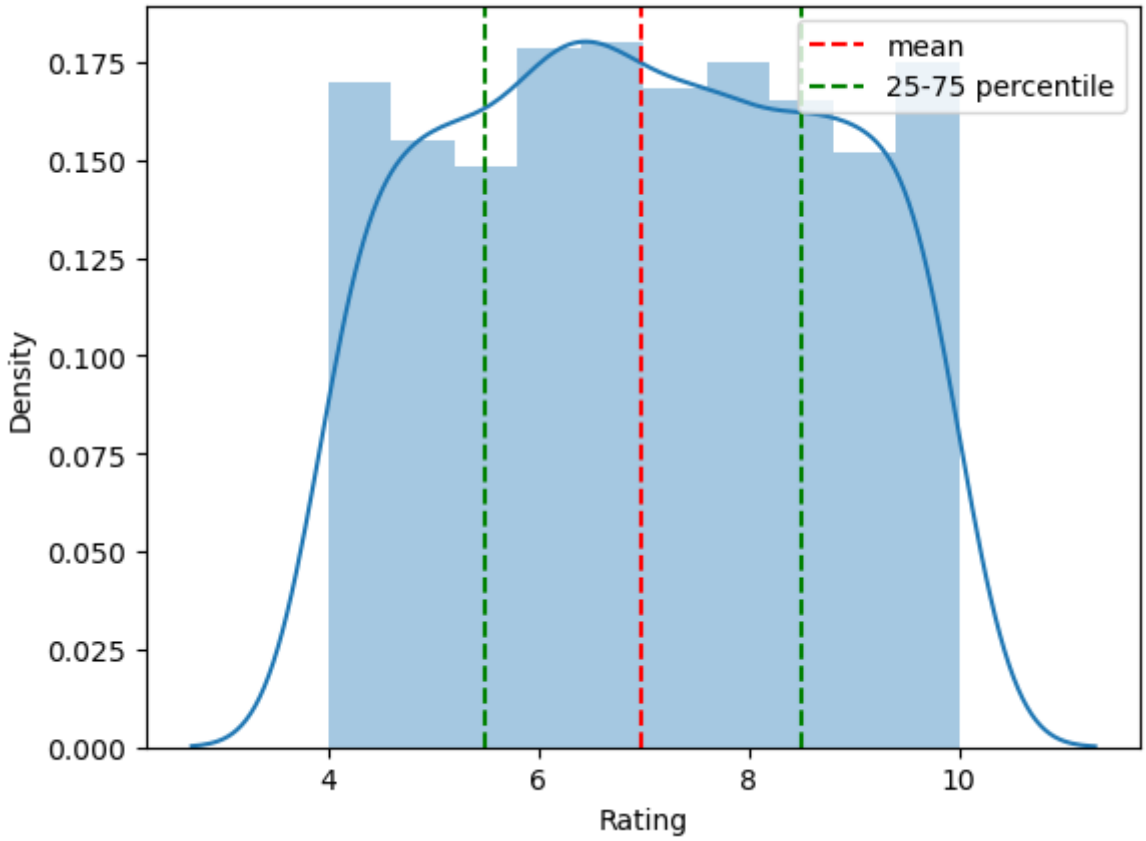
- CHeck the distribution of Rating , if it is skewed or not

In [ ]:
```
from cProfile import label


sns.distplot(supermarket_sales['Rating'])
plt.axvline(x=np.mean(supermarket_sales['Rating']),c='red',ls='--',label='mean')
plt.axvline(x=np.percentile(supermarket_sales['Rating'],25),c='green',ls='--',label
plt.axvline(x=np.percentile(supermarket_sales['Rating'],75),c='green',ls='--')
plt.legend()
```

```
c:\Users\HP\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. P
lease adapt your code to use either `displot` (a figure-level function with simila
r flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

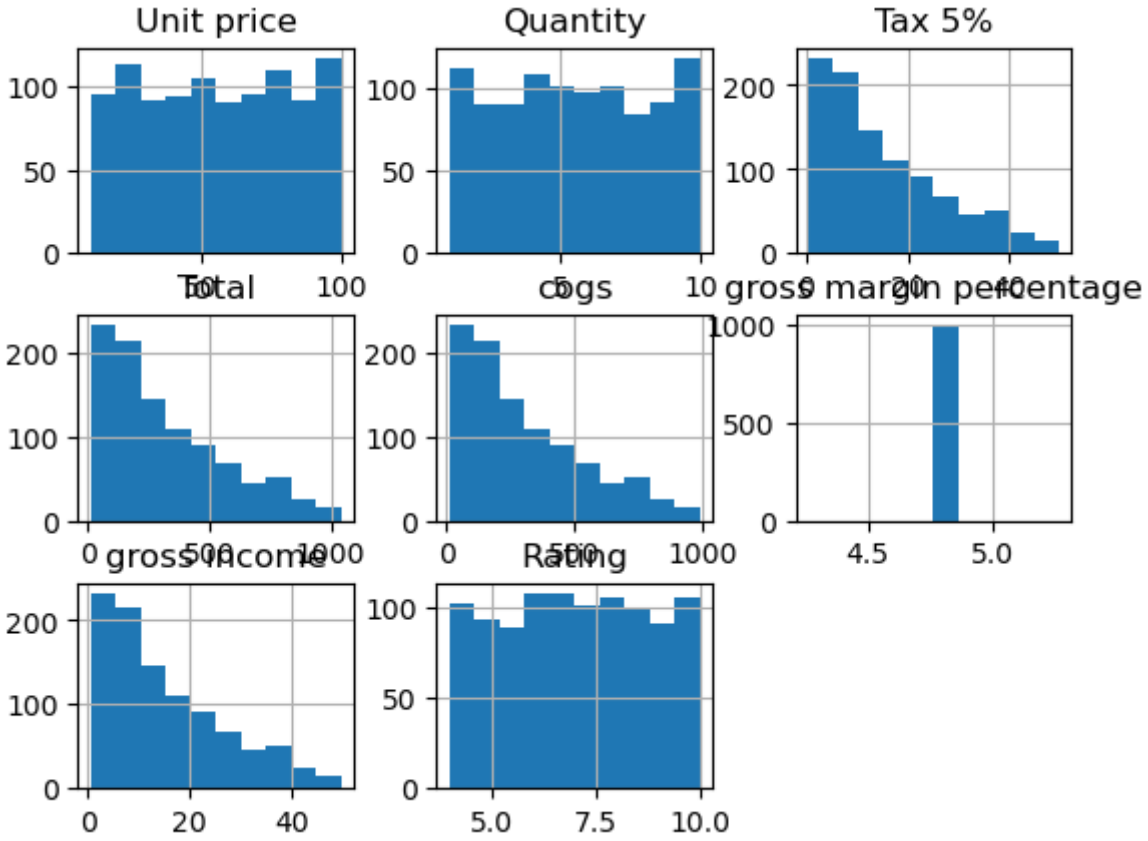
Out[ ]:
```
<matplotlib.legend.Legend at 0x23fb0695190>
```



Observations: The distribution of Rating looks like uniform and there is no skewness either on left or right side.

In [ ]:
```
supermarket_sales.hist()
```

Out[ ]:
```
array([[<AxesSubplot:title={'center':'Unit price'}>,
        <AxesSubplot:title={'center':'Quantity'}>,
        <AxesSubplot:title={'center':'Tax 5%'}>],
       [<AxesSubplot:title={'center':'Total'}>,
        <AxesSubplot:title={'center':'cogs'}>,
        <AxesSubplot:title={'center':'gross margin percentage'}>],
       [<AxesSubplot:title={'center':'gross income'}>,
        <AxesSubplot:title={'center':'Rating'}>, <AxesSubplot:>]],
      dtype=object)
```



Observations:

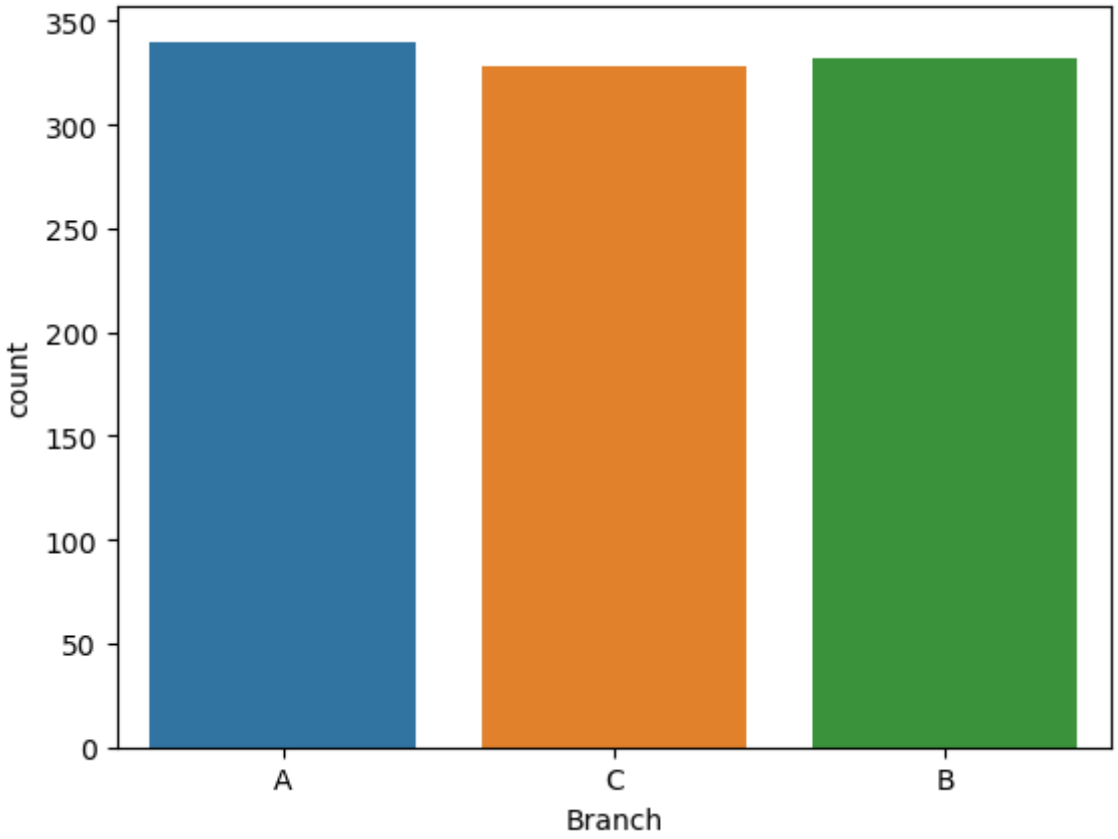
- Unit Price, Quantity & Rating have uniform distribution.
- Gross Margin Percentage has fixed percentage.
- Tax 5%, Cog, Total and Gross income is right skewed.

Do Aggregate Sales distribution differ bewtween branches

In [ ]:
```
sns.countplot(supermarket_sales['Branch'])
```

```
c:\Users\HP\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 explici
t keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[ ]:
```
<AxesSubplot:xlabel='Branch', ylabel='count'>
```



Observations: There is no such big difference between aggregate sales of different branch category. But Branch A is at the top with slight difference.

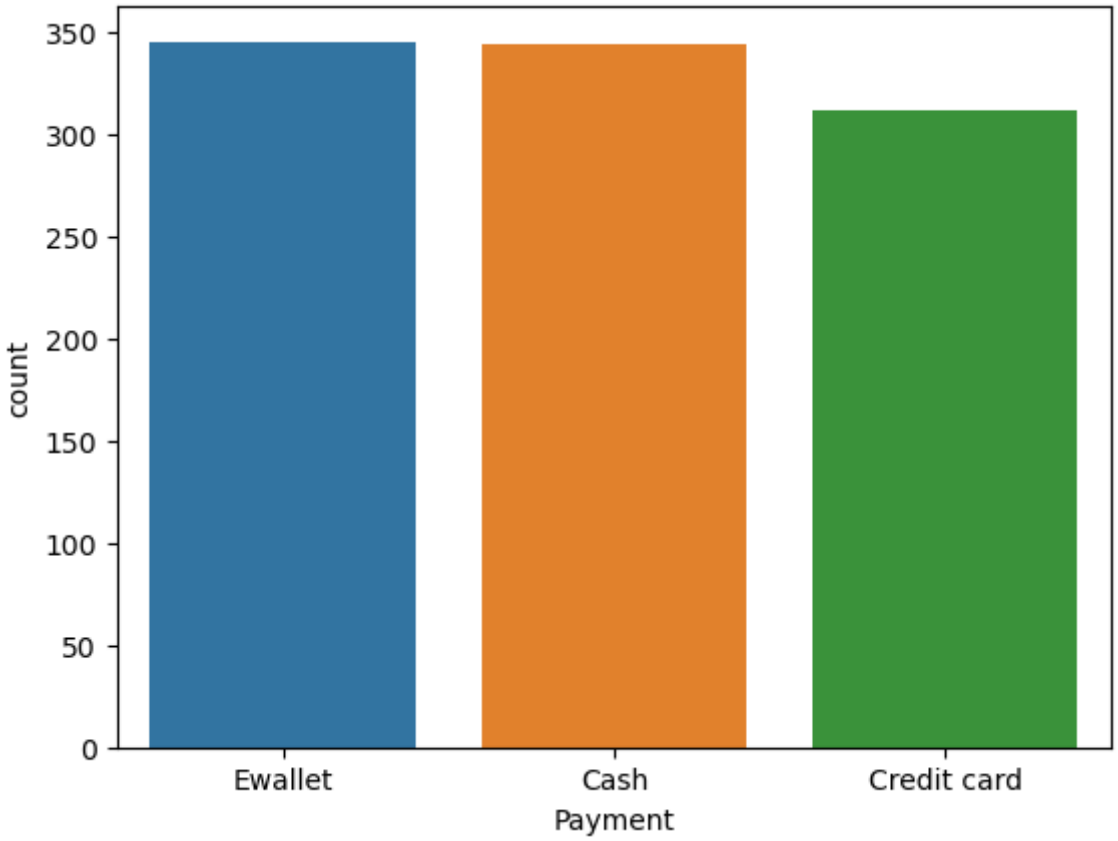
In [ ]:
```
supermarket_sales['Branch'].value_counts()
```

Out[ ]:
```
A    340
B    332
C    328
Name: Branch, dtype: int64
```

In [ ]:
```
sns.countplot(supermarket_sales['Payment'])
```

```
c:\Users\HP\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 explici
t keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[ ]:
```
<AxesSubplot:xlabel='Payment', ylabel='count'>
```



Observations:

- Mostly customers make payment through Ewallet and Cash Payment. Credit Card payment is slightly less usable payment method as compare to Ewallet and Cash Payment.

In [ ]:
```
supermarket_sales.columns
```

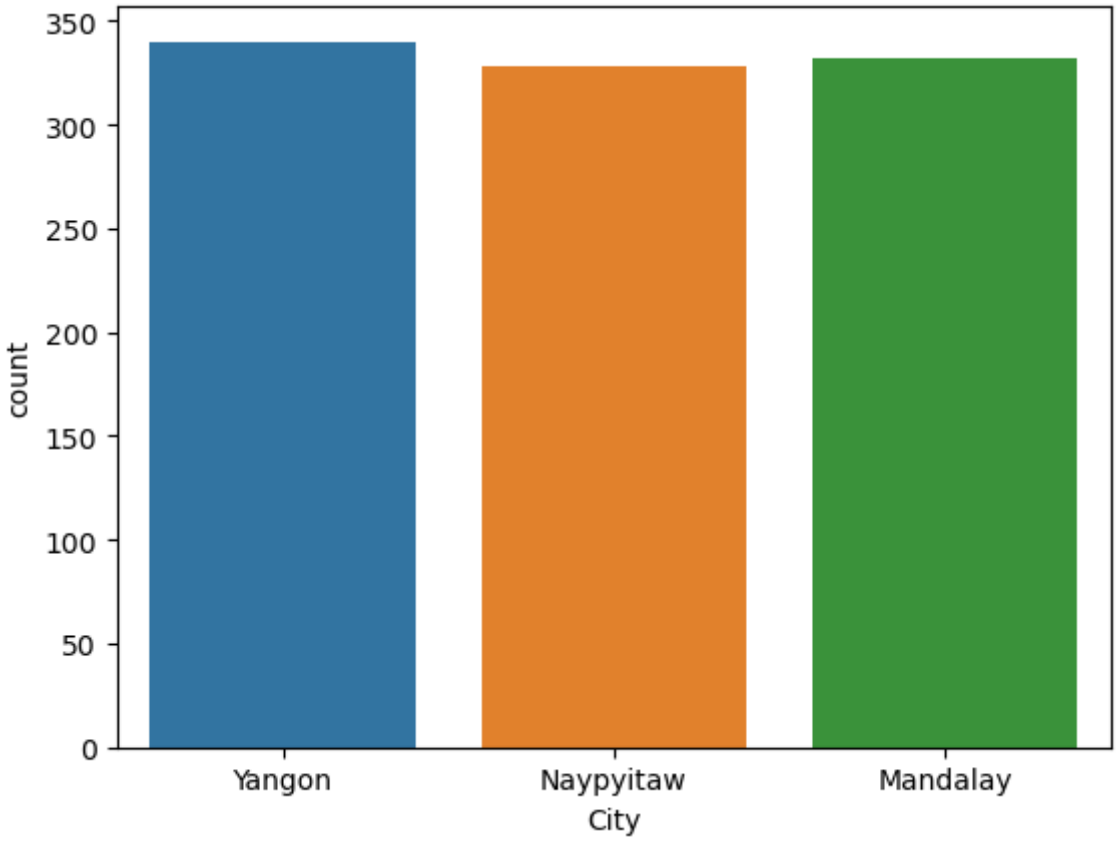
Out[ ]:
```
Index(['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender',
       'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Time',
       'Payment', 'cogs', 'gross margin percentage', 'gross income', 'Rating'],
      dtype='object')
```

Which city has more number of aggregate Sale

In [ ]:
```
sns.countplot(supermarket_sales['City'])
```

```
c:\Users\HP\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 explici
t keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[ ]:
```
<AxesSubplot:xlabel='City', ylabel='count'>
```



Observations:

- Although there is slight difference between sales of each city but still Yangon is at the top with slight difference from Mandalay (2nd) and Naypyitaw(3rd).

In [ ]:
```
supermarket_sales['City'].value_counts()
```

Out[ ]:
```
Yangon       340
Mandalay     332
Naypyitaw    328
Name: City, dtype: int64
```

BIVARIATE ANALYSIS

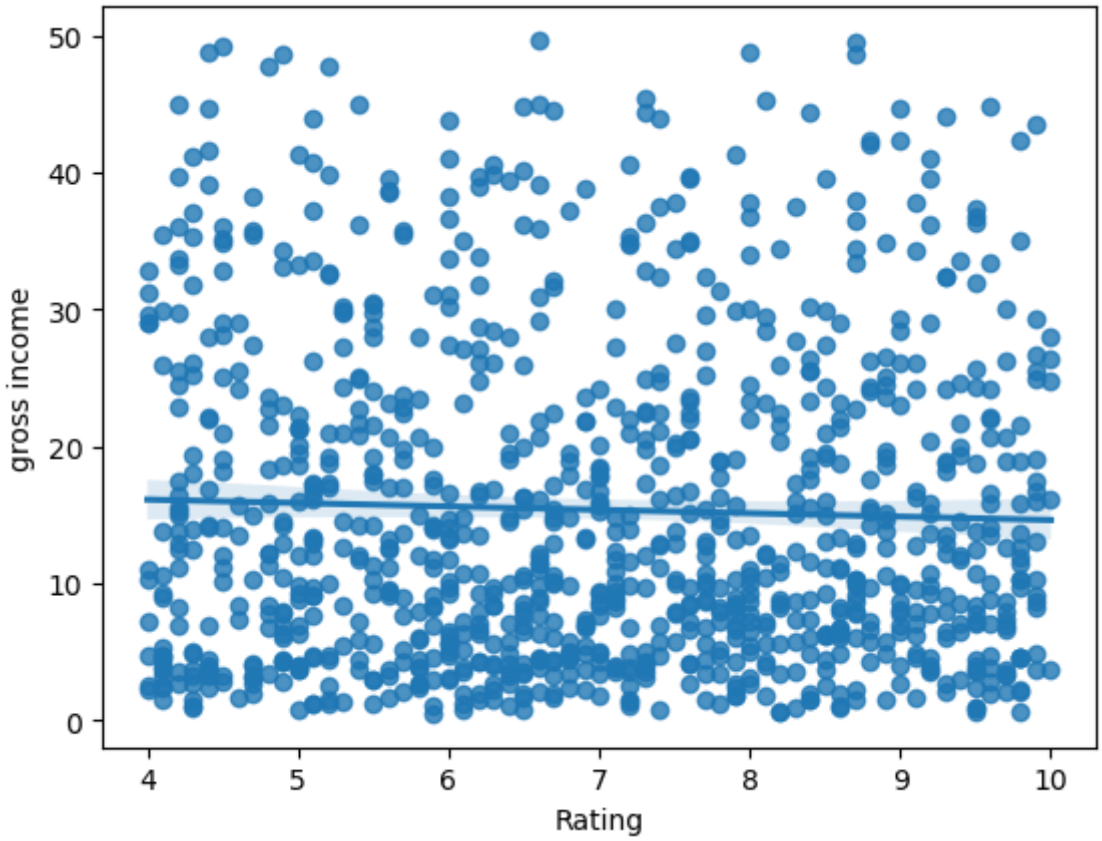
In [ ]:
```
supermarket_sales.columns
```

Out[ ]:
```
Index(['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender',
       'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Time',
       'Payment', 'cogs', 'gross margin percentage', 'gross income', 'Rating'],
      dtype='object')
```

- Check relationship between Customer Rating and Gross Income

In [ ]:
```
sns.regplot(x=supermarket_sales['Rating'], y=supermarket_sales['gross income'])
```

Out[ ]:
```
<AxesSubplot:xlabel='Rating', ylabel='gross income'>
```

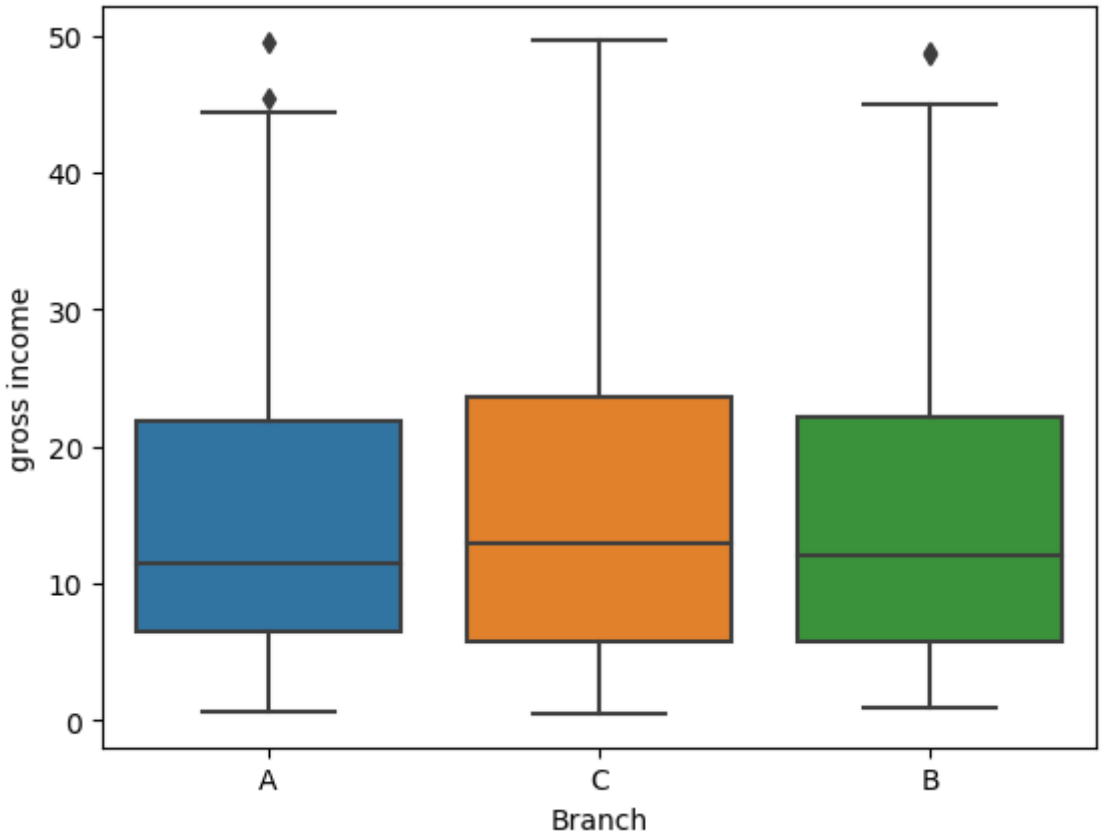


Observations:

- No considerable relationship between Customer Rating and Gross Income as plotted regression line show no such movement between them.

In [ ]:
```
sns.boxplot(x=supermarket_sales['Branch'], y=supermarket_sales['gross income'])
```

Out[ ]:
```
<AxesSubplot:xlabel='Branch', ylabel='gross income'>
```



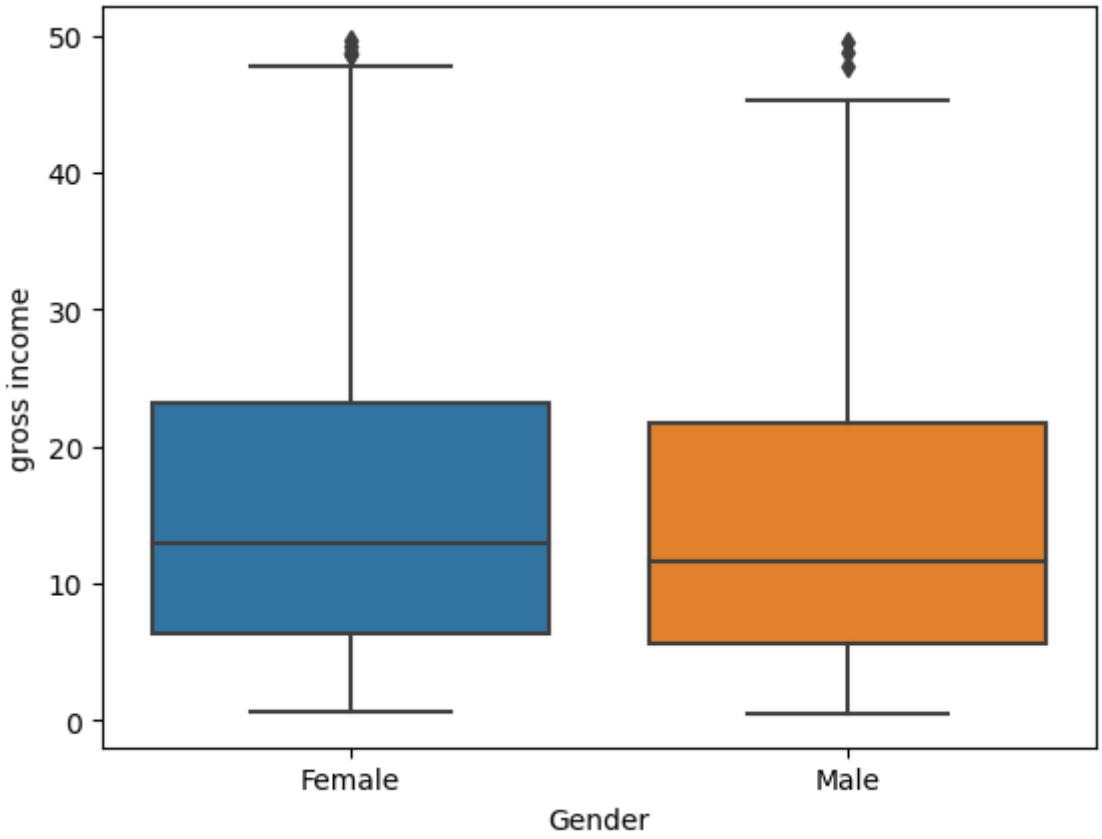
Observations:

- There is not much variation between the Gross Income of different branches.
- Branch C Gross Income is slightly higher than Branch A & B as by comparing the median values in boxplot.

- Comparison between Gross Income Genderwise.

In [ ]:
```
sns.boxplot(x=supermarket_sales['Gender'], y=supermarket_sales['gross income'])
```

Out[ ]:
```
<AxesSubplot:xlabel='Gender', ylabel='gross income'>
```



Observations:

- In this dataset, its look alike that overall women and men spends near about same with slight difference.
- At 75% Quarantile Females spend slightly more than men.

- Check Gross income variation in time frame

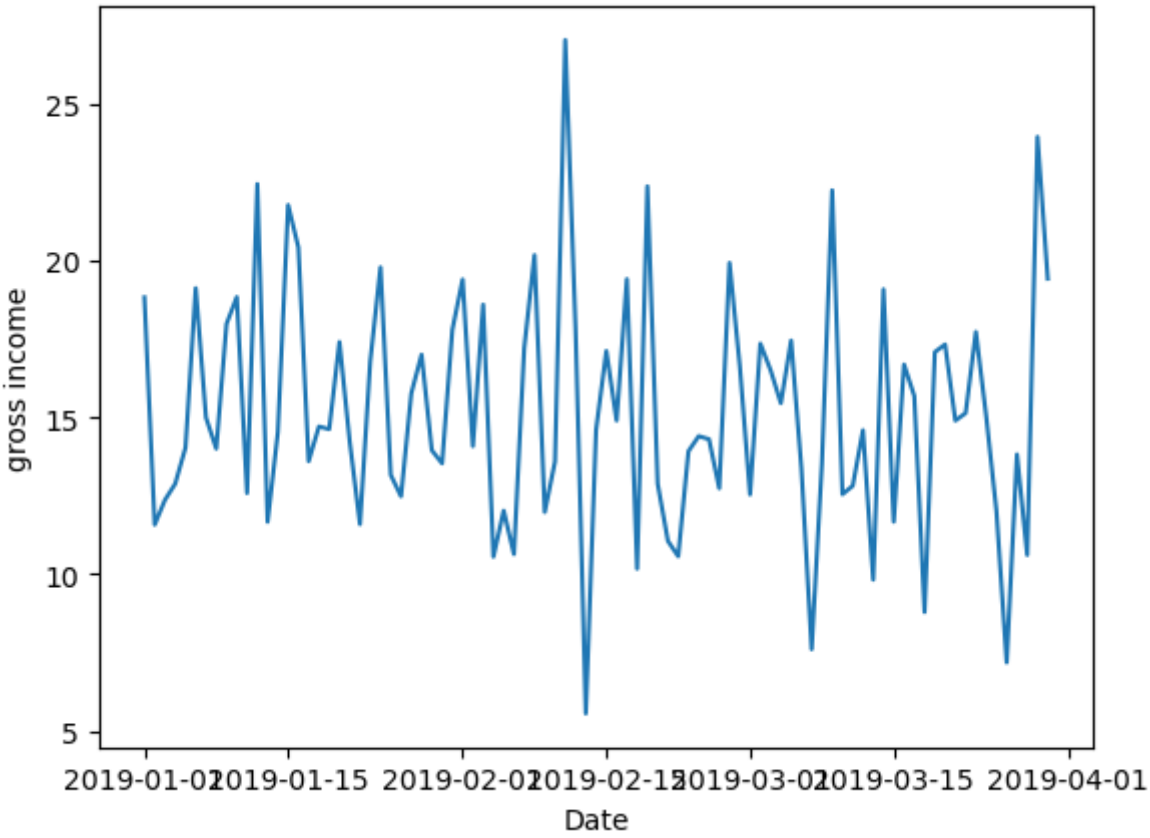
In [ ]:
```
supermarket_sales.groupby(supermarket_sales.index).mean()
```

Out[ ]:
```
DatetimeIndex(['2019-01-01', '2019-01-02', '2019-01-03', '2019-01-04',
               '2019-01-05', '2019-01-06', '2019-01-07', '2019-01-08',
               '2019-01-09', '2019-01-10', '2019-01-11', '2019-01-12',
               '2019-01-13', '2019-01-14', '2019-01-15', '2019-01-16',
               '2019-01-17', '2019-01-18', '2019-01-19', '2019-01-20',
               '2019-01-21', '2019-01-22', '2019-01-23', '2019-01-24',
               '2019-01-25', '2019-01-26', '2019-01-27', '2019-01-28',
               '2019-01-29', '2019-01-30', '2019-01-31', '2019-02-01',
               '2019-02-02', '2019-02-03', '2019-02-04', '2019-02-05',
               '2019-02-06', '2019-02-07', '2019-02-08', '2019-02-09',
               '2019-02-10', '2019-02-11', '2019-02-12', '2019-02-13',
               '2019-02-14', '2019-02-15', '2019-02-16', '2019-02-17',
               '2019-02-18', '2019-02-19', '2019-02-20', '2019-02-21',
               '2019-02-22', '2019-02-23', '2019-02-24', '2019-02-25',
               '2019-02-26', '2019-02-27', '2019-02-28', '2019-03-01',
               '2019-03-02', '2019-03-03', '2019-03-04', '2019-03-05',
               '2019-03-06', '2019-03-07', '2019-03-08', '2019-03-09',
               '2019-03-10', '2019-03-11', '2019-03-12', '2019-03-13',
               '2019-03-14', '2019-03-15', '2019-03-16', '2019-03-17',
               '2019-03-18', '2019-03-19', '2019-03-20', '2019-03-21',
               '2019-03-22', '2019-03-23', '2019-03-24', '2019-03-25',
               '2019-03-26', '2019-03-27', '2019-03-28', '2019-03-29',
               '2019-03-30'],
              dtype='datetime64[ns]', name='Date', freq=None)
```

In [ ]:
```
sns.lineplot(supermarket_sales.groupby(supermarket_sales.index).mean().index, y=su
```

```
c:\Users\HP\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 explici
t keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[ ]:
```
<AxesSubplot:xlabel='Date', ylabel='gross income'>
```



Observations:

- We cant see any time series trend in Gross Income.
- In few dates Gross income touches the high level which in other few dates Gross income is very low.
- So we can see any considerable trend in income during the time frame.

Check Duplicate and NUll values

In [ ]:
```
supermarket_sales.isnull().sum()
```

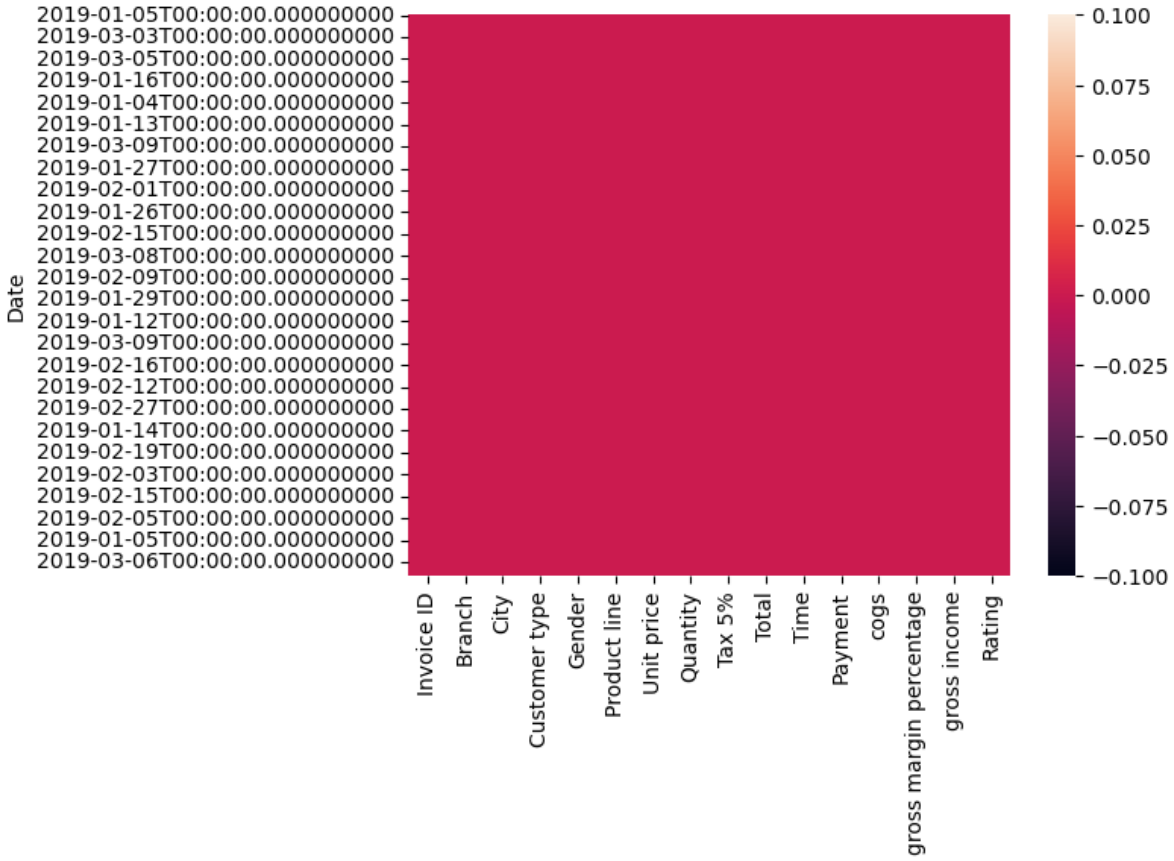
Out[ ]:
```
Invoice ID                 0
Branch                     0
City                       0
Customer type              0
Gender                     0
Product line               0
Unit price                 0
Quantity                   0
Tax 5%                     0
Total                      0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

In [ ]:
```
supermarket_sales.duplicated().sum()
```

Out[ ]:
```
0
```

In [ ]:
```
sns.heatmap(supermarket_sales.isnull())
```

Out[ ]:
```
<AxesSubplot:ylabel='Date'>
```



In [ ]:
```
ProfileReport(supermarket_sales)
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

Overview

Dataset statistics

Number of variables	17
Number of observations	1000
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	132.9 KiB
Average record size in memory	136.1 B

Variable types

DateTime	1
Categorical	9
Numeric	7

Alerts

`gross margin percentage` has constant value "4.761904762"	**Constant**
`Invoice ID` has a high cardinality: 1000 distinct values	**High cardinality**

Out[ ]:

Corelation Analysis

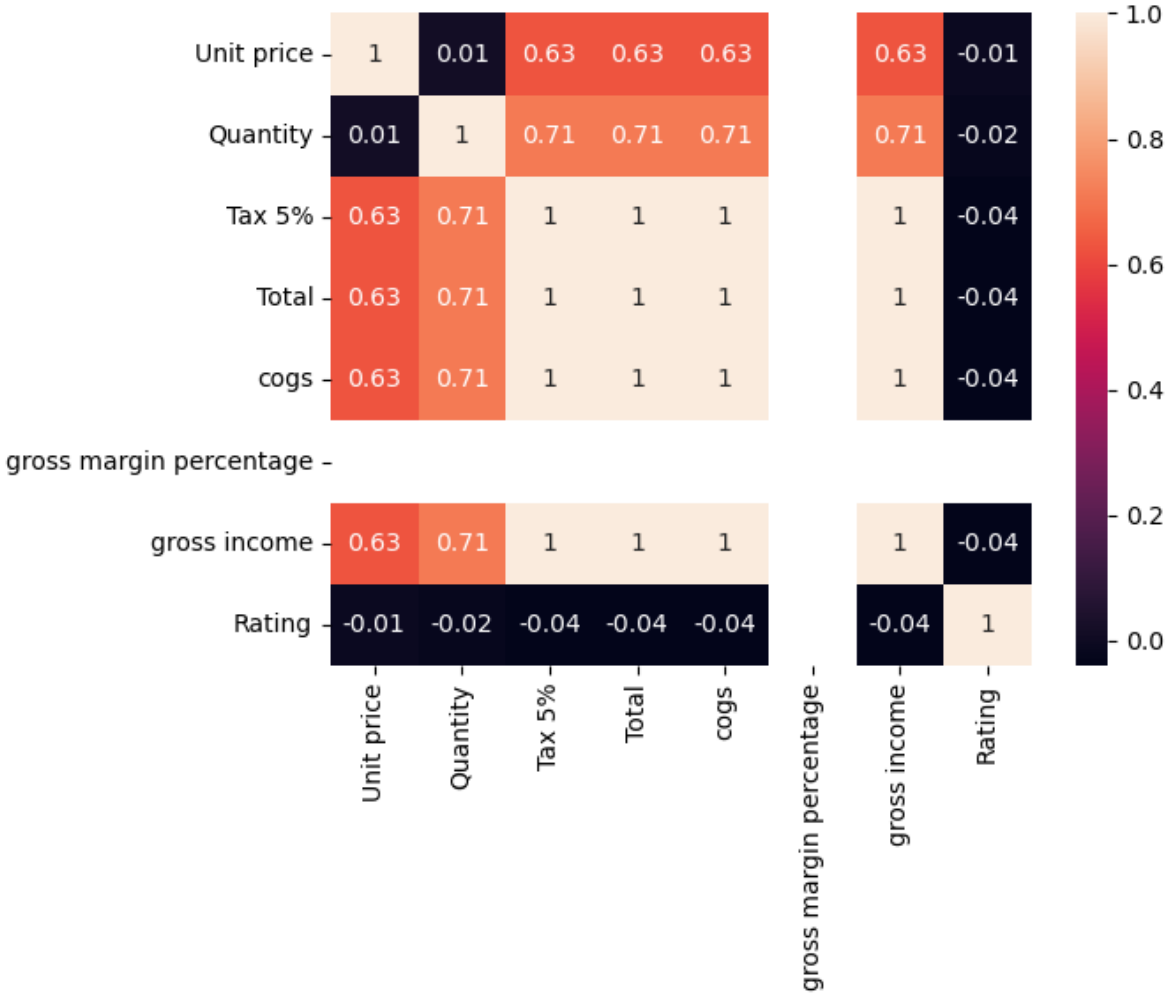
In [ ]:
```
round(supermarket_sales.corr(),2)
```

Out[ ]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
Unit price	1.00	0.01	0.63	0.63	0.63	NaN	0.63	-0.01
Quantity	0.01	1.00	0.71	0.71	0.71	NaN	0.71	-0.02
Tax 5%	0.63	0.71	1.00	1.00	1.00	NaN	1.00	-0.04
Total	0.63	0.71	1.00	1.00	1.00	NaN	1.00	-0.04
cogs	0.63	0.71	1.00	1.00	1.00	NaN	1.00	-0.04
gross margin percentage	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
gross income	0.63	0.71	1.00	1.00	1.00	NaN	1.00	-0.04
Rating	-0.01	-0.02	-0.04	-0.04	-0.04	NaN	-0.04	1.00

In [ ]:
```
sns.heatmap(round(supermarket_sales.corr(),2), annot=True)
```

Out[ ]:
```
<AxesSubplot:>
```



In [ ]: