Libraries

In [ ]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
import calmap
from ydata_profiling import ProfileReport
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please update jup
yter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_
install.html
  from .autonotebook import tqdm as notebook_tqdm
```

Task:1 Initial Data exploration

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

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

Out[ ]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Ta
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30

In [ ]:
```
#last 10 rows
df.tail(10)
```

Out[ ]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity
990	886-18-2897	A	Yangon	Normal	Female	Food and beverages	56.56	5
991	602-16-6955	B	Mandalay	Normal	Female	Sports and travel	76.60	10
992	745-74-0715	A	Yangon	Normal	Male	Electronic accessories	58.03	2
993	690-01-6631	B	Mandalay	Normal	Male	Fashion accessories	17.49	10
994	652-49-6720	C	Naypyitaw	Member	Female	Electronic accessories	60.95	1
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7

In [ ]:
```
#shape of the data
df.shape
```

Out[ ]:
```
(1000, 17)
```

In [ ]:
```
#data types
df.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                       object
Time                       object
Payment                    object
cogs                      float64
gross margin percentage   float64
gross income              float64
Rating                    float64
dtype: object
```

In [ ]:
```
#data info
df.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 [ ]:
```
#data description
df.describe()
```

Out[ ]:

	Unit price	Quantity	Tax 5%	Total	cogs	g ma percen
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1000.000
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.76
std	26.494628	2.923431	11.708825	245.885335	234.17651	0.000
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.76
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.76
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.76
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.76
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.76

In [ ]:
```
#data columns
df.columns
```

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

In [ ]:
```
#date object is object type
df['Date'].dtype
```

Out[ ]:
```
dtype('O')
```

In [ ]:
```
#convert data object to datetime
df['Date'] = pd.to_datetime(df['Date'])
```

In [ ]:
```
#set the date to permanent change
df.set_index('Date', inplace=True)
```

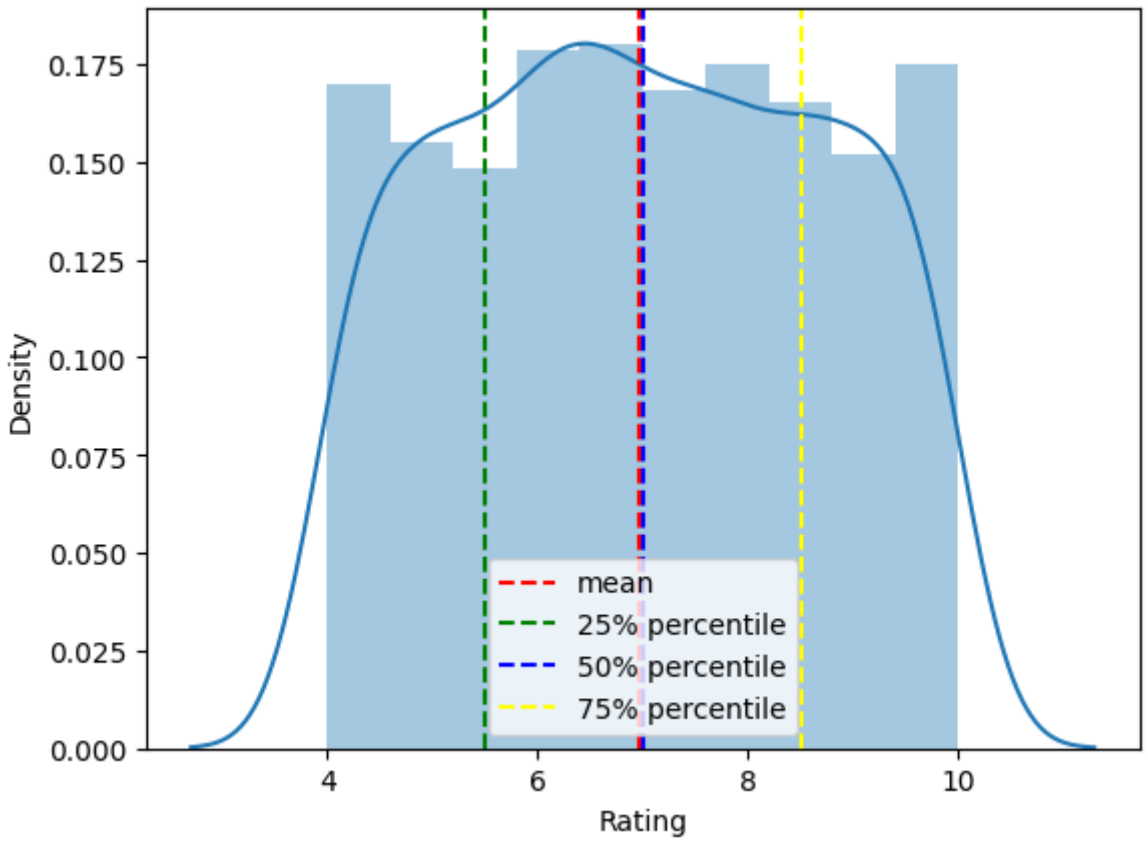
Task 2 Univariate Analysis

** Question 1: What does the distribution of customer ratings looks like? Is it skewed?
**

In [ ]:
```
sns.distplot(df['Rating'])
#mean of rating in plt
plt.axvline(df['Rating'].mean(), color='red', linestyle='--', label='mean
#show 25% percentile in plt
plt.axvline(df['Rating'].quantile(0.25), color='green', linestyle='--', l
#show 50% percentile in plt
plt.axvline(df['Rating'].quantile(0.50), color='blue', linestyle='--', la
#show 75% percentile in plt
plt.axvline(df['Rating'].quantile(0.75), color='yellow', linestyle='--',
plt.legend()
```

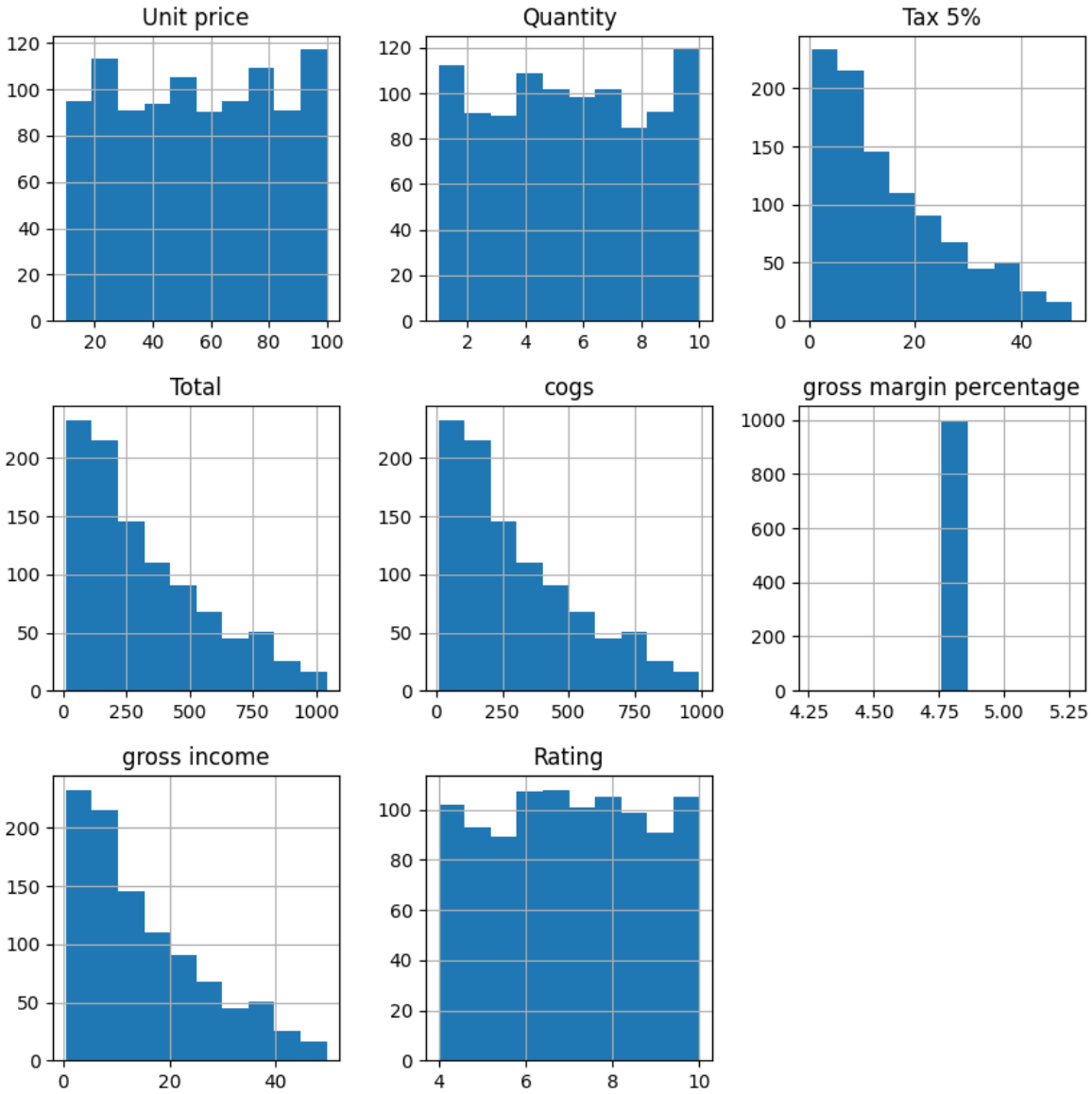
```
/var/folders/mr/xqkrmqvj7ljfrz59pxqfk6rc0000gn/T/ipykernel_20549/326280470
6.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Rating'])
```

Out[ ]: <matplotlib.legend.Legend at 0x1213db100>



In [ ]:
```
#histogram of each column
df.hist(figsize=(10,10))
```

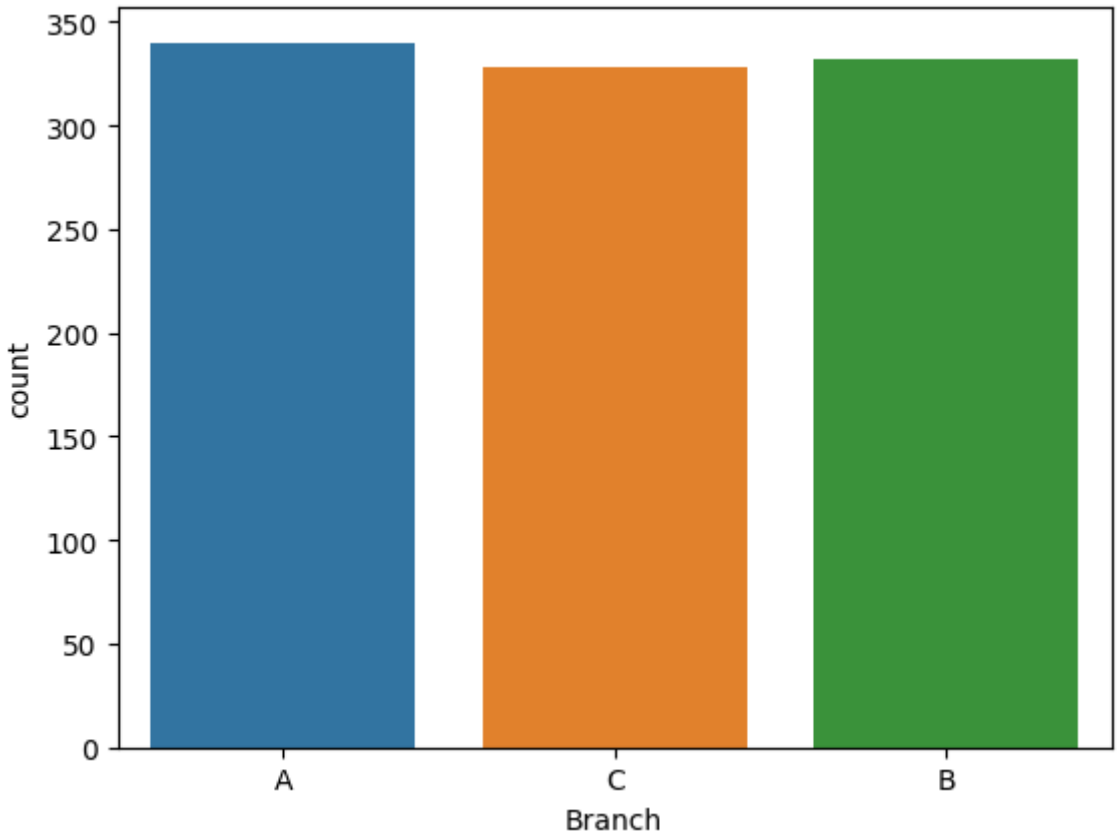
Out[ ]:
```
array([[<Axes: title={'center': 'Unit price'}>,
        <Axes: title={'center': 'Quantity'}>,
        <Axes: title={'center': 'Tax 5%'}>],
       [<Axes: title={'center': 'Total'}>,
        <Axes: title={'center': 'cogs'}>,
        <Axes: title={'center': 'gross margin percentage'}>],
  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



Question 2: Do aggregate sales numbers differ by much between branches?

In [ ]:
```
#sns.countplot(df["Branch"])
#got this errot running above line code could not convert string to float
#so i convert the column to string
#df["Branch"] = df["Branch"].astype(str)
#still getting the same error
#so i convert the column to float
#df["Branch"] = df["Branch"].astype(float)
#could not convert string to float: 'A'
#so i convert the column to int
#put xlable in the branch column and ylable in the count column
sns.countplot(x="Branch", data=df)
```

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

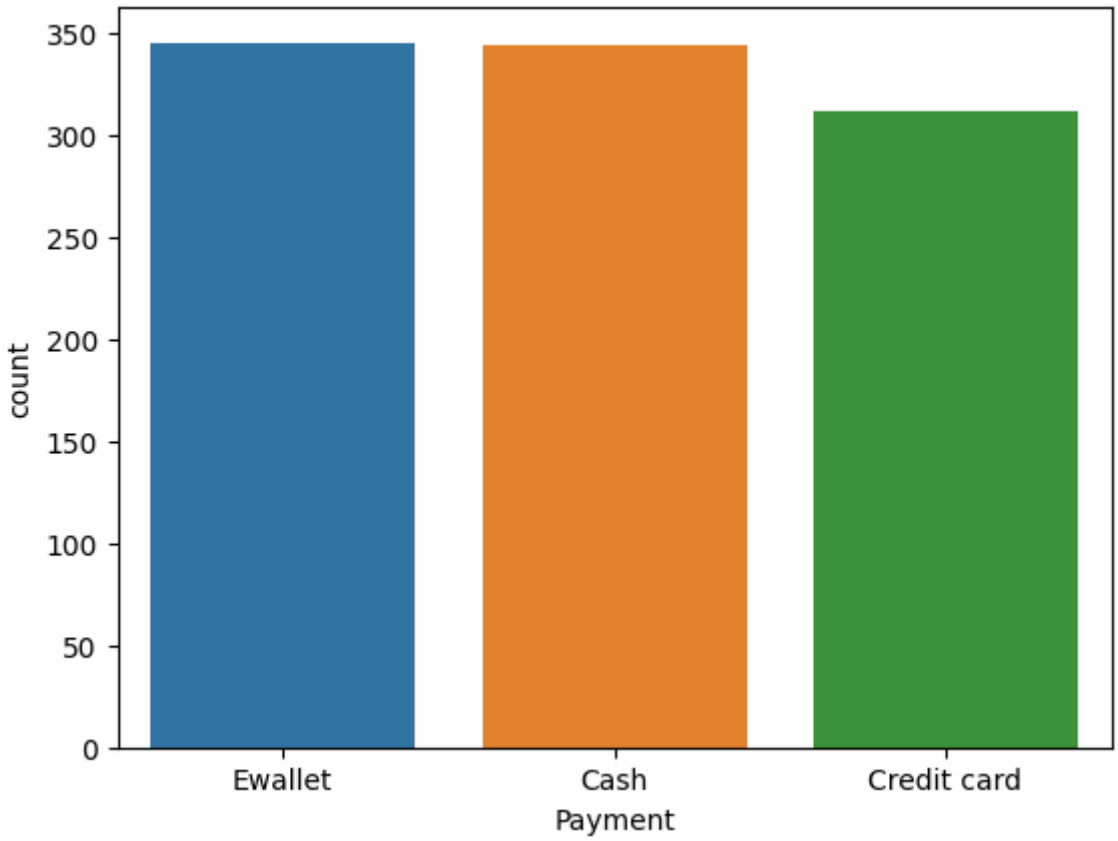


In [ ]:
```
#calculate the total of each branch in df
df["Branch"].value_counts()
```

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

In [ ]:
```
#put xlable payment method and ylable in the count column
sns.countplot(x="Payment", data=df)
```

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



In [ ]:
```
#calculate the total of each payment method in df
df["Payment"].value_counts()
```

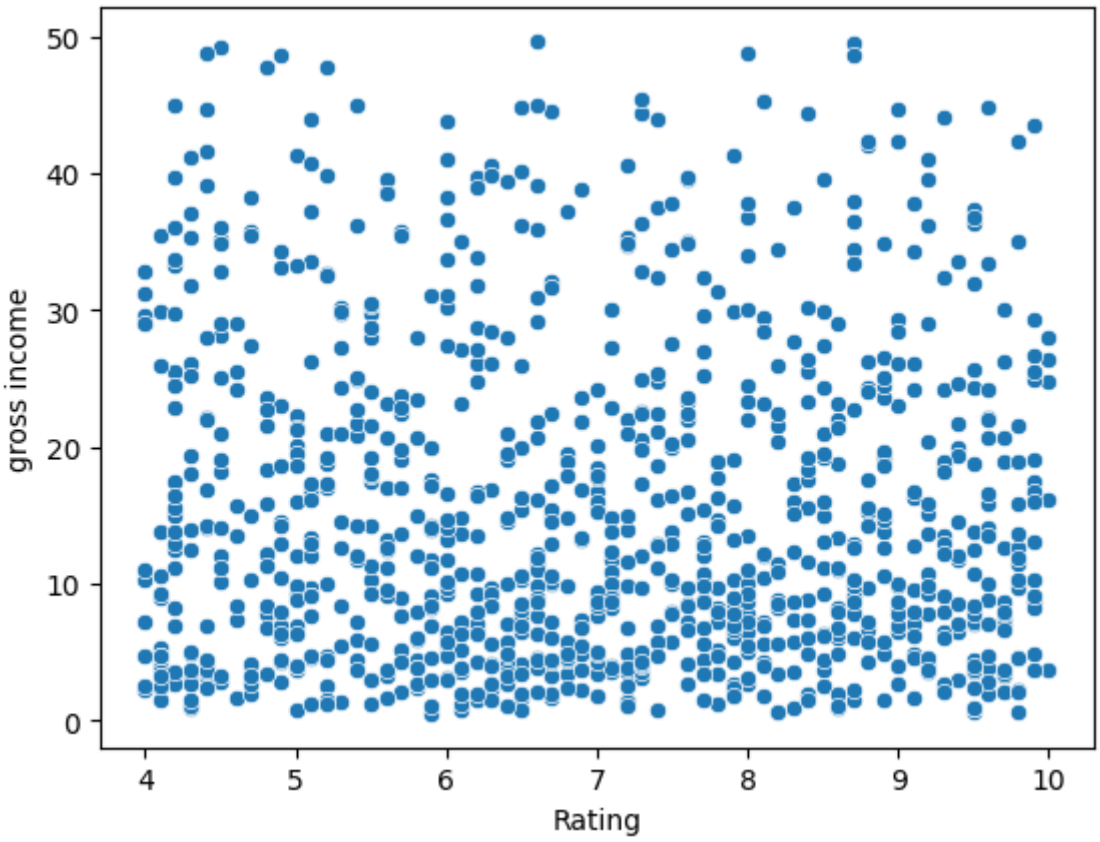
Out[ ]:
```
Ewallet        345
Cash           344
Credit card    311
Name: Payment, dtype: int64
```

Task 3 Bivariate Analysis

Question 3: Is there a relationship between gross income and customer ratings?

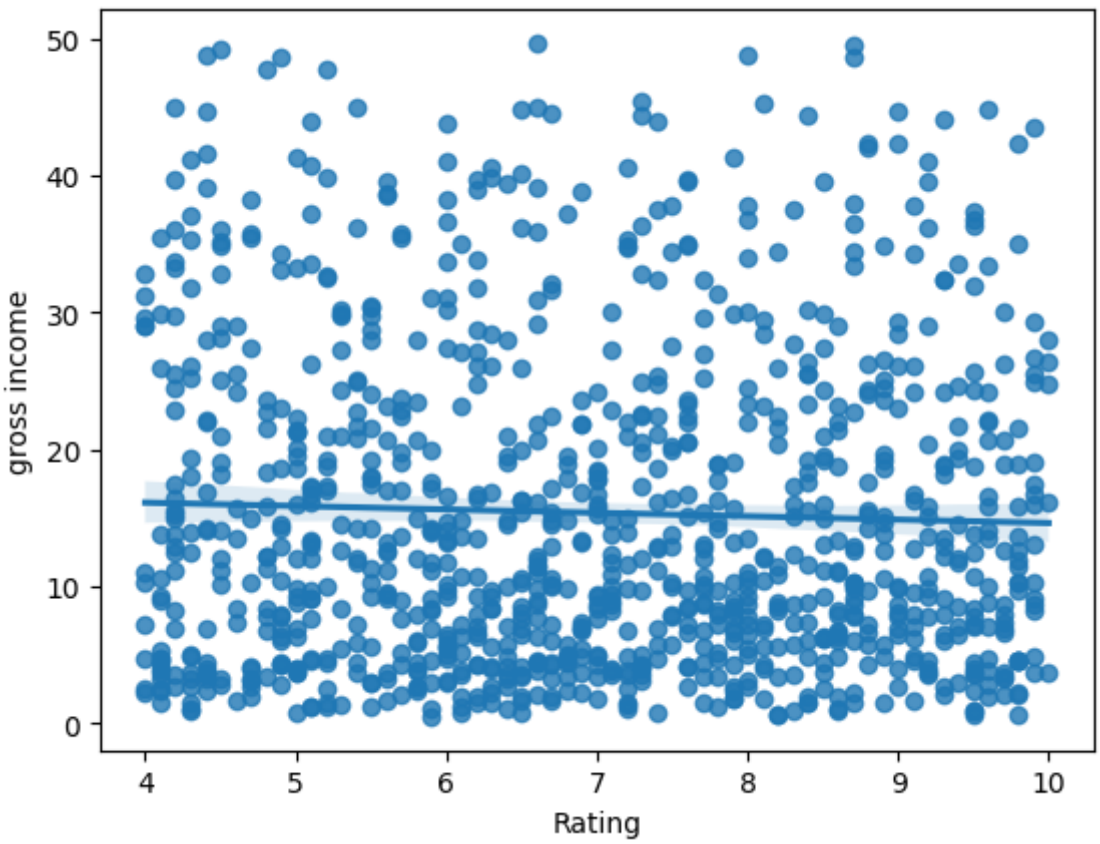
In [ ]:
```
sns.scatterplot(x="Rating", y="gross income", data=df)
```

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



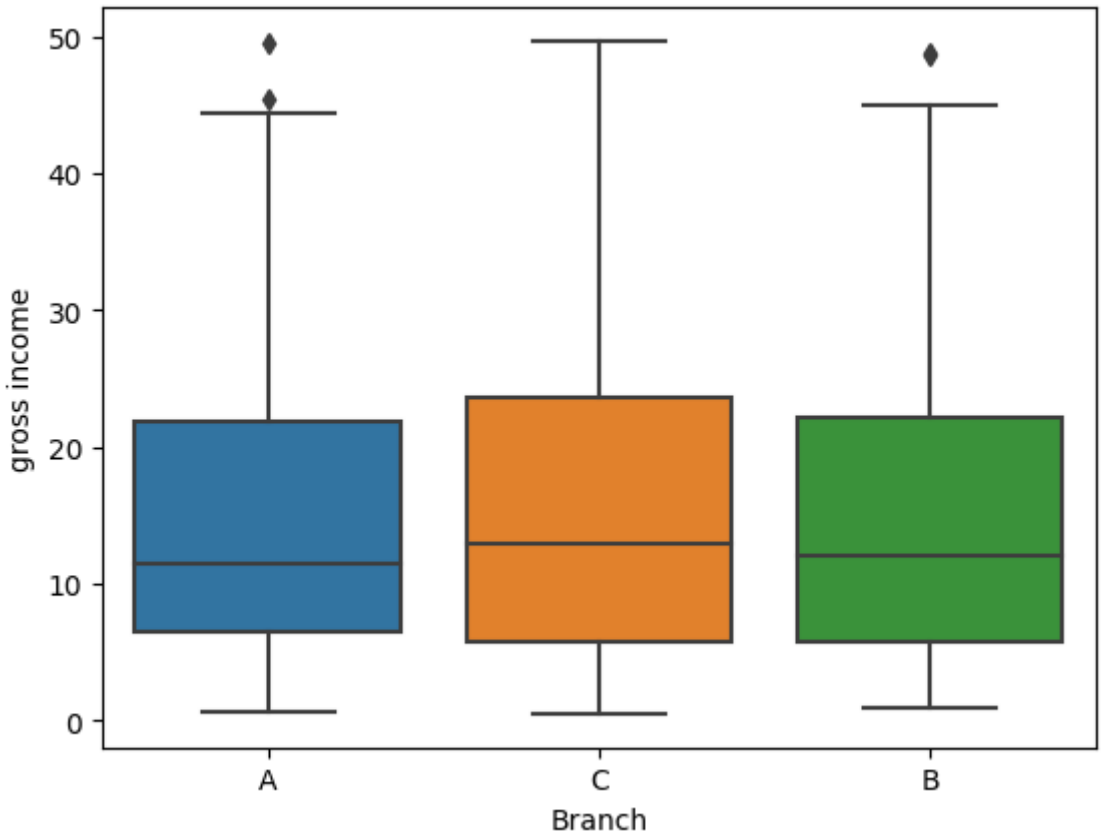
In [ ]:
```
# regression plot using seaborn library
sns.regplot(x="Rating", y="gross income", data=df)
```

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



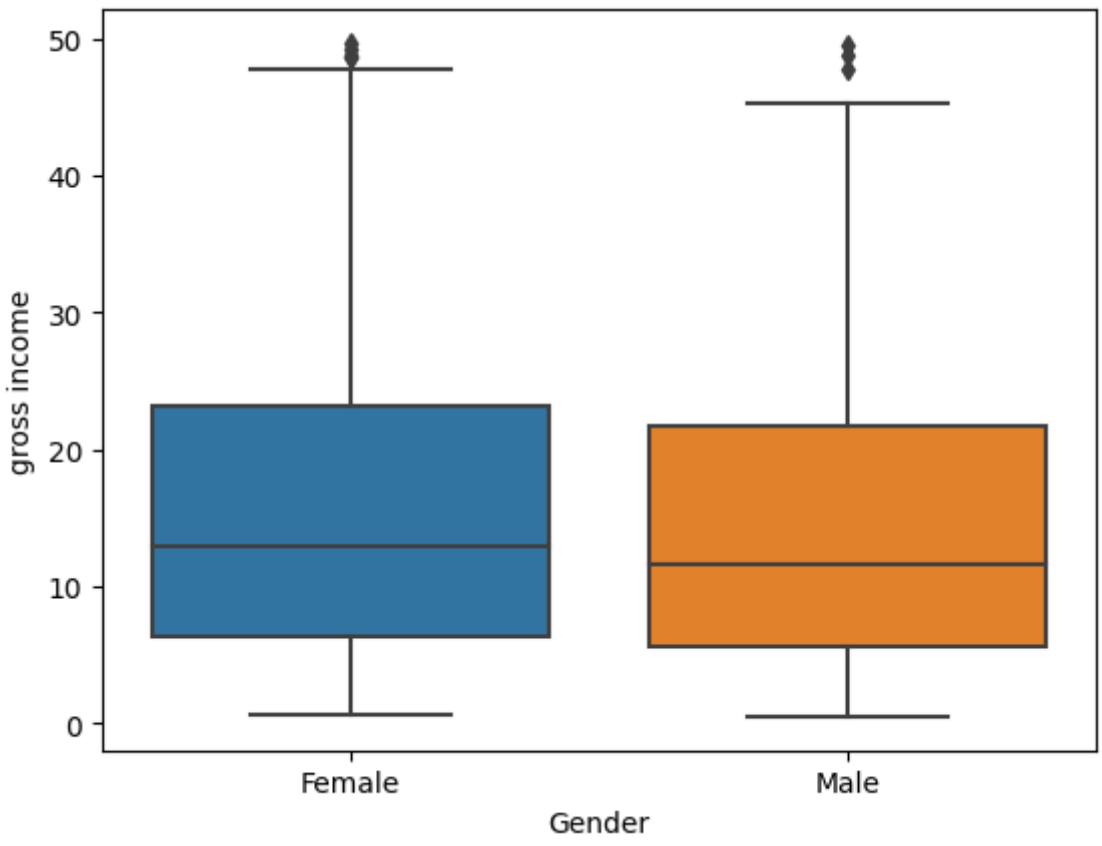
In [ ]:
```
# boxplot using seaborn library between branch and gross income
sns.boxplot(x="Branch", y="gross income", data=df)
```

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



In [ ]:
```
#boxplot uing seaborn between gender and gross income
sns.boxplot(x="Gender", y="gross income", data=df)
```

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



Question 4: Is there a noticeable time trend in gross income?

In [ ]:
```
#df groupby index and mean
df.groupby(df.index).mean()
```

```
/var/folders/mr/xqkrmqvj7ljfrz59pxqfk6rc0000gn/T/ipykernel_20549/48615760
1.py:2: FutureWarning: The default value of numeric_only in DataFrameGroup
By.mean is deprecated. In a future version, numeric_only will default to F
alse. Either specify numeric_only or select only columns which should be v
alid for the function.
  df.groupby(df.index).mean()
```

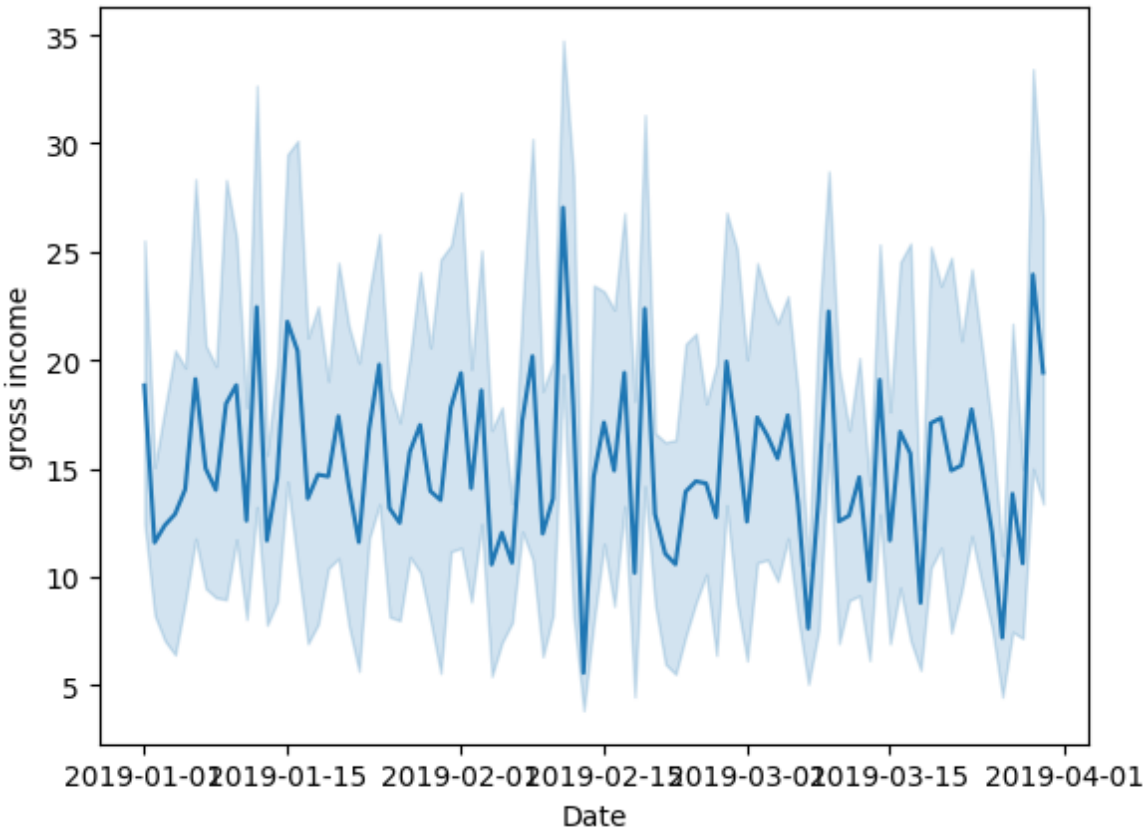
Out[ ]:

Date	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	
2019-01-01	54.995833	6.750000	18.830083	395.431750	376.601667	4.761905	18.8
2019-01-02	44.635000	6.000000	11.580375	243.187875	231.607500	4.761905	11.
2019-01-03	59.457500	4.625000	12.369813	259.766062	247.396250	4.761905	12.
2019-01-04	51.743333	5.333333	12.886417	270.614750	257.728333	4.761905	12.
2019-01-05	61.636667	4.583333	14.034458	294.723625	280.689167	4.761905	14.(
...	...	...	...	...	...	...	
2019-03-26	42.972308	4.000000	7.188692	150.962538	143.773846	4.761905	7.
2019-03-27	56.841000	4.500000	13.822950	290.281950	276.459000	4.761905	13.8
2019-03-28	45.525000	4.800000	10.616200	222.940200	212.324000	4.761905	10.
2019-03-29	66.346250	6.750000	23.947875	502.905375	478.957500	4.761905	23.
2019-03-30	67.408182	6.090909	19.424500	407.914500	388.490000	4.761905	19.4

89 rows × 8 columns

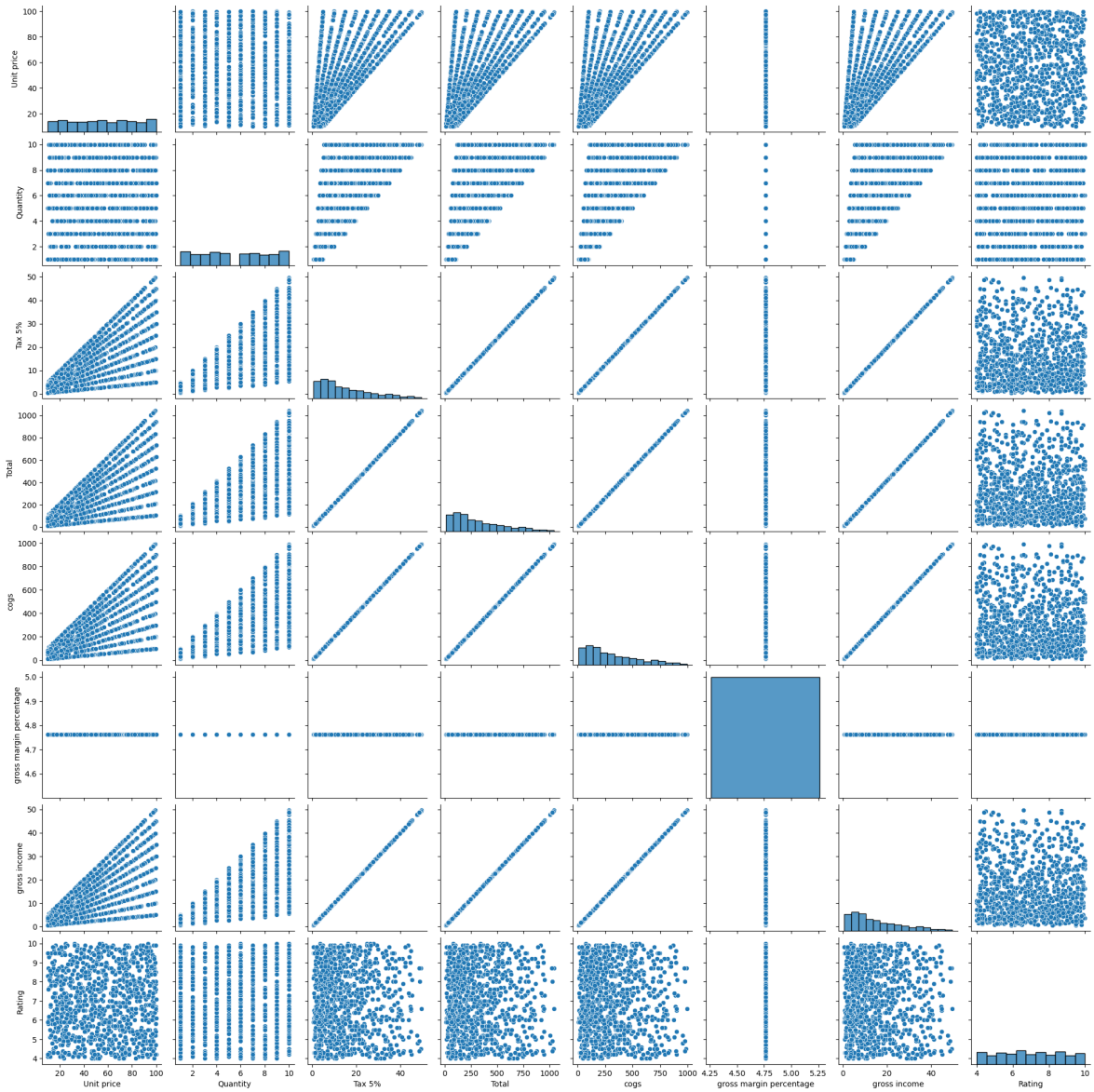
In [ ]:
```
sns.lineplot(x=df.index, y="gross income", data=df)
```

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



In [ ]:
```
#pairplot using seaborn library
sns.pairplot(df)
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x123413ca0>
```



Task 4: Dealing With Duplicate Rows and Missing Values

In [ ]:
```
#duplicate rows
df.duplicated()
```

Out[ ]:
```
Date
2019-01-05    False
2019-03-08    False
2019-03-03    False
2019-01-27    False
2019-02-08    False
              ...
2019-01-29    False
2019-03-02    False
2019-02-09    False
2019-02-22    False
2019-02-18    False
Length: 1000, dtype: bool
```

In [ ]:
```
#count duplicate rows
df.duplicated().sum()
```

Out[ ]: 0

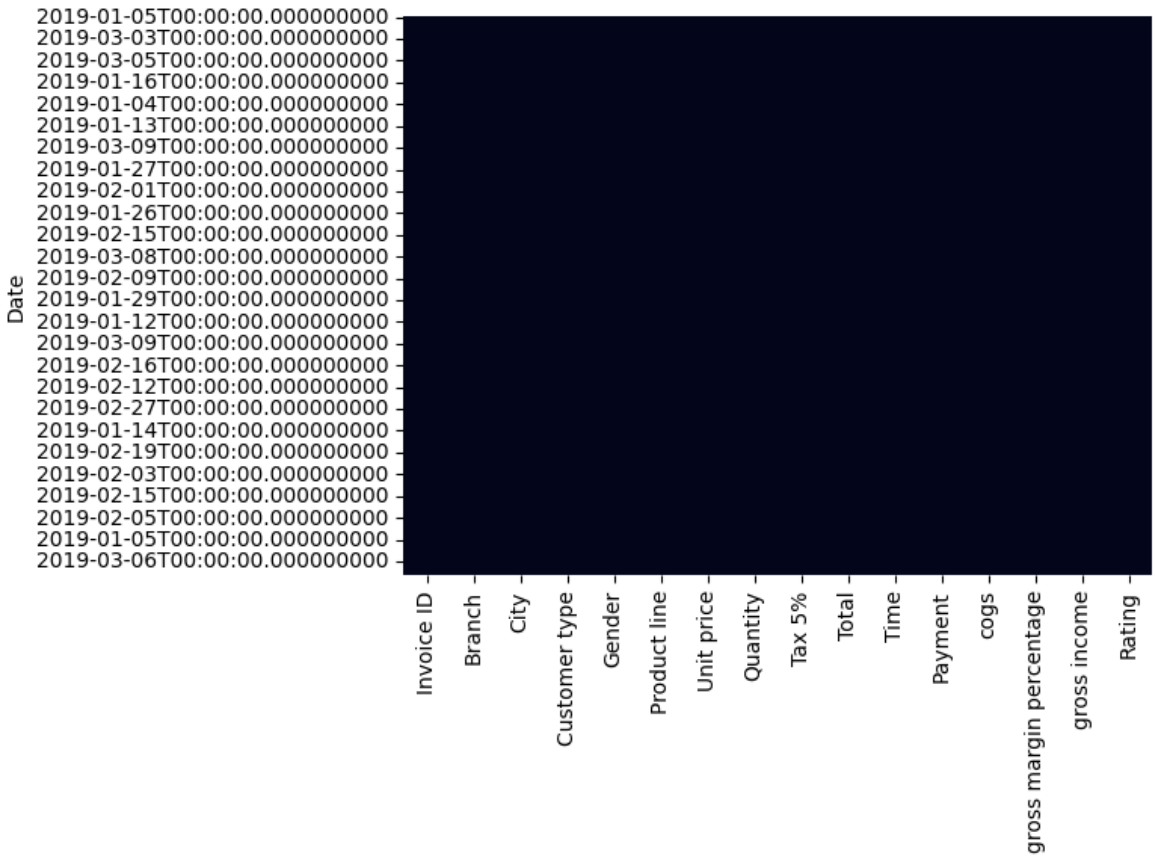
In [ ]:
```
#drop duplicate rows if any
df.drop_duplicates(inplace=True)
```

In [ ]:
```
#check NAN values in df
df.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 [ ]:
```
#heatmap using seaborn library to see the NAN values
sns.heatmap(df.isnull(), cbar=False)
```

Out[ ]: <Axes: ylabel='Date'>



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

```
prof.to_file(output_file='output.html')
```

```
Summarize dataset: 100%|██████████| 75/75 [00:20<00:00,  3.66it/s, Completed]
Generate report structure: 100%|██████████| 1/1 [00:11<00:00, 11.54s/it]
Render HTML: 100%|██████████| 1/1 [00:04<00:00,  4.47s/it]
Export report to file: 100%|██████████| 1/1 [00:00<00:00, 15.33it/s]
```

Task 5: Correlation Analysis

In [ ]:
```
#correlation between gross income and rating
corr = np.corrcoef(df['gross income'], df['Rating'])[1][0]
round_val = round(corr, 2)
print(f"Correlation between gross income and rating is {round_val}")
```

```
Correlation between gross income and rating is -0.04
```

In [ ]:
```
datafram_corr = df.corr()
round_df= datafram_corr.round(2)
round_df
```

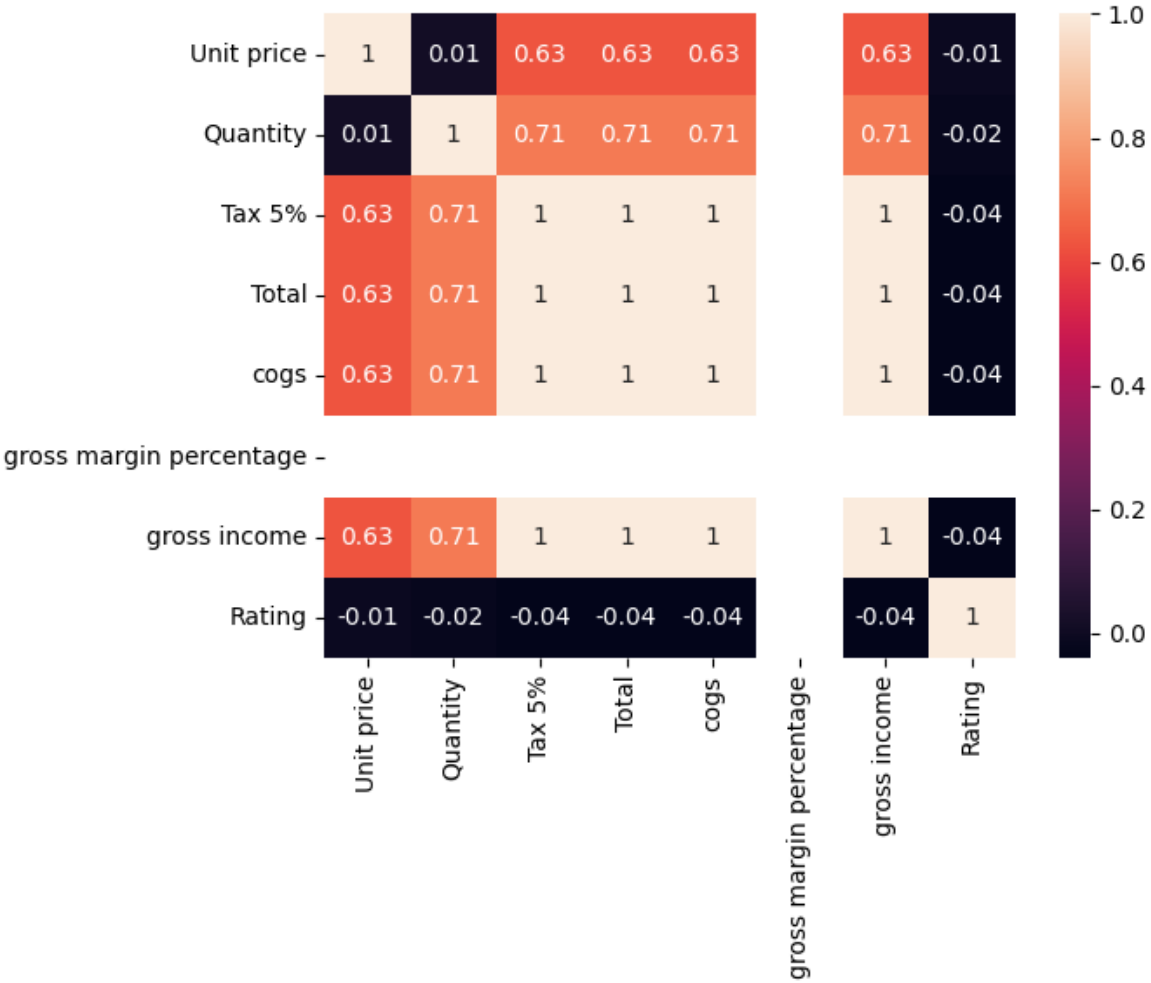
```
/var/folders/mr/xqkrmqvj7ljfrz59pxqfk6rc0000gn/T/ipykernel_20549/637010295.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  datafram_corr = df.corr()
```

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_df, annot=True)
```

Out[ ]:
```
<Axes: >
```


Unit price
Quantity
Tax 5%
Total
cogs
gross margin percentage
gross income
Rating
1 0.01 0.63 0.63 0.63 0.63 -0.01
0.01 1 0.71 0.71 0.71 0.71 -0.02
0.63 0.71 1 1 1 1 -0.04
0.63 0.71 1 1 1 1 -0.04
0.63 0.71 1 1 1 1 -0.04
0.63 0.71 1 1 1 1 -0.04
-0.01 -0.02 -0.04 -0.04 -0.04 -0.04 1
1.0
0.8
0.6
0.4
0.2
0.0