Exploratory Data Analysis With Python and Pandas

Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import calmap
from pandas_profiling import ProfileReport
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-1-b0f00164bff8> in <module>
      3 import matplotlib.pyplot as plt
      4 import seaborn as sns
----> 5 import calmap
      6 from pandas_profiling import ProfileReport

ModuleNotFoundError: No module named 'calmap'
```

Link to data source: https://www.kaggle.com/aungpyaeap/supermarket-sales (https://www.kaggle.com/aungpyaeap/supermarket-sales)

Context

The growth of supermarkets in most populated cities are increasing and market competitions are also high. The dataset is one of the historical sales of supermarket company which has recorded in 3 different branches for 3 months data.

Data Dictionary

1. ***Invoice id:*** Computer generated sales slip invoice identification number
2. ***Branch:*** Branch of supercenter (3 branches are available identified by A, B and C).
3. ***City:*** Location of supercenters
4. ***Customer type:*** Type of customers, recorded by Members for customers using member card and Normal for without member card.
5. ***Gender:*** Gender type of customer
6. ***Product line:*** General item categorization groups - Electronic accessories, Fashion accessories, Food and beverages, Health and beauty, Home and lifestyle, Sports and travel
7. ***Unit price:*** Price of each product in USD
8. ***Quantity:*** Number of products purchased by customer
9. ***Tax:*** 5% tax fee for customer buying
10. ***Total:*** Total price including tax
11. ***Date:*** Date of purchase (Record available from January 2019 to March 2019)
12. ***Time:*** Purchase time (10am to 9pm)
13. ***Payment:*** Payment used by customer for purchase (3 methods are available – Cash, Credit card and Ewallet)
14. ***COGS:*** Cost of goods sold
15. ***Gross margin percentage:*** Gross margin percentage
16. ***Gross income:*** Gross income
17. ***Rating:*** Customer stratification rating on their overall shopping experience (On a scale of 1 to 10)

Task 1: Initial Data Exploration

In [3]:

```
df = pd.read_csv('supermarket_sales.csv')
```

In [33]:

```
df['Date'] = pd.to_datetime(df['Date'])
```

In [35]:

```
#it is common convention to set the Date as index
df.set_index('Date', inplace=True)
```

In [40]:

```
df.head()
```

Out[40]:

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	7.0	26.1415	548
2019-03-08	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5.0	3.8200	80
2019-03-03	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7.0	16.2155	340
2019-01-27	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8.0	23.2880	489
2019-02-08	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7.0	30.2085	634

Task 2: Univariate Analysis

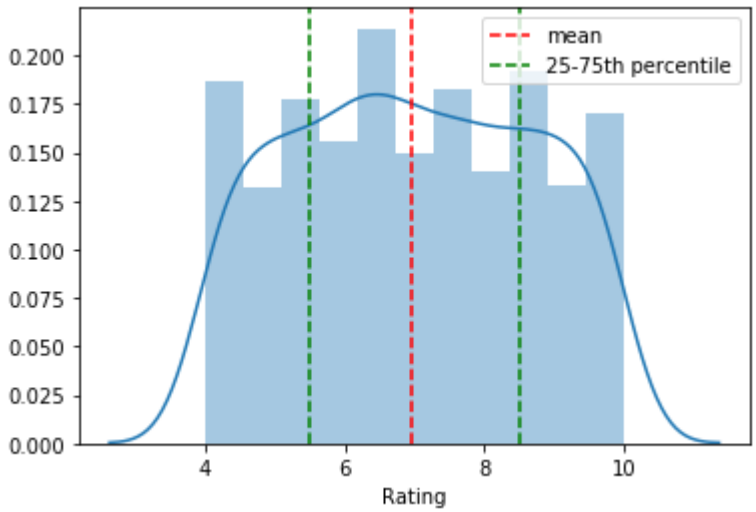
Question 1: What does the distribution of customer ratings looks like? Is it skewed?

In [17]:

```
sns.distplot(df['Rating']) #plot graph
#mean of rating of the dataframe, red line, dotted, labelled
plt.axvline(x=np.mean(df['Rating']), c='red', ls='--', label='mean')
plt.axvline(x=np.percentile(df['Rating'], 25), ls='--', c='green', label='25-75th percenti
le')
plt.axvline(x=np.percentile(df['Rating'], 75), ls='--', c='green')
plt.legend()
#relatively uniform and no skew on left or right
```

Out[17]:

```
<matplotlib.legend.Legend at 0x7f8bda3a9750>
```

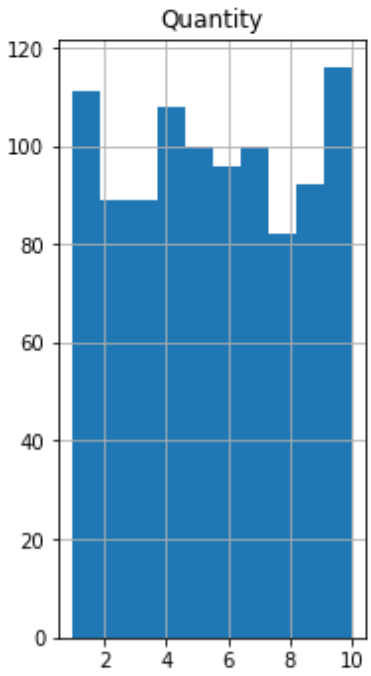


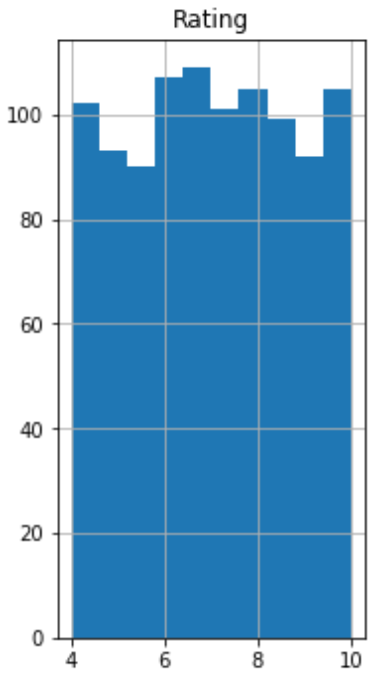
In [22]:

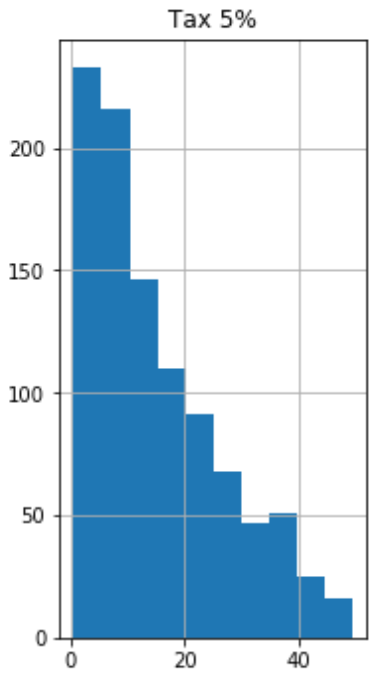
```
#df.histogram, plots all of the graphs
df.hist(figsize=(10,20))
```

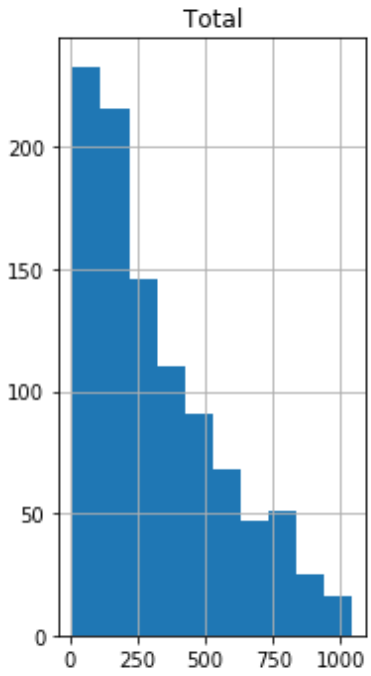
Out[22]:

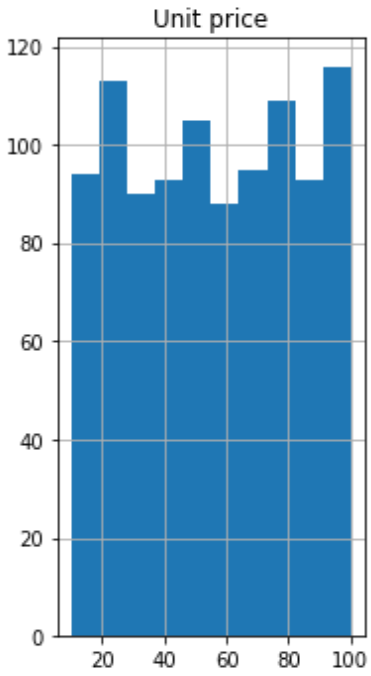
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f8bd8567650>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8bd84c88d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8bd8483f90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f8bd8445850>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8bd83f69d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8bd83b9890>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f8bd83eaa10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8bd832c8d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8bd8336510>]],
      dtype=object)
```

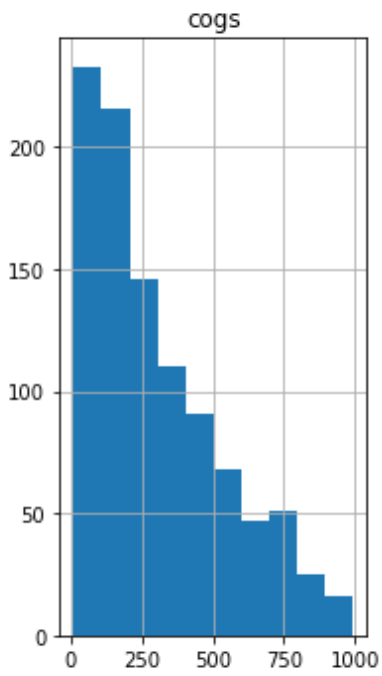

Quantity
120
100
80
60
40
20
0
2
4
6
8
10

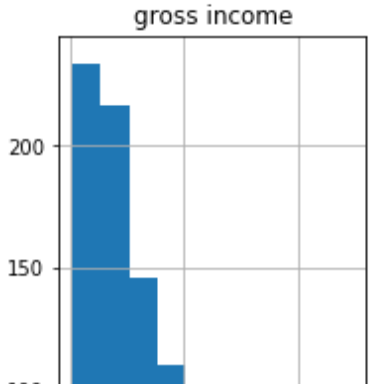

Rating
100
80
60
40
20
0
4
6
8
10

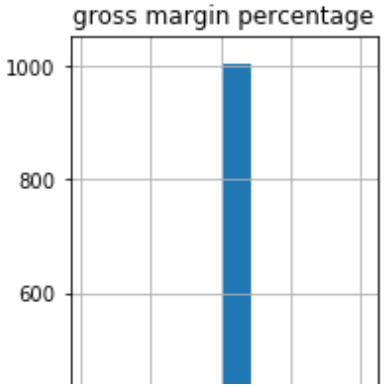

Tax 5%
200
150
100
50
0
0
20
40

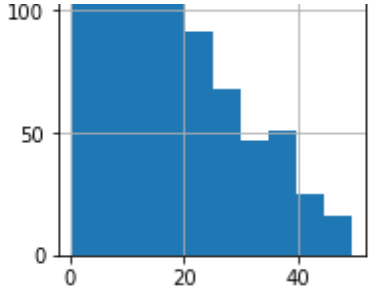

Total
200
150
100
50
0
0
250
500
750
1000

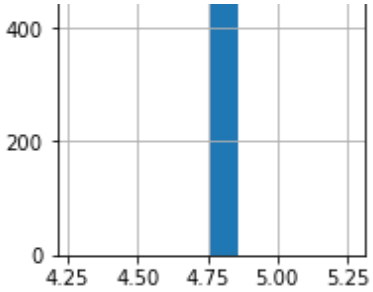

Unit price
120
100
80
60
40
20
0
20
40
60
80
100


cogs
200
150
100
50
0
0
250
500
750
1000


gross income
200
150


gross margin percentage
1000
800
600





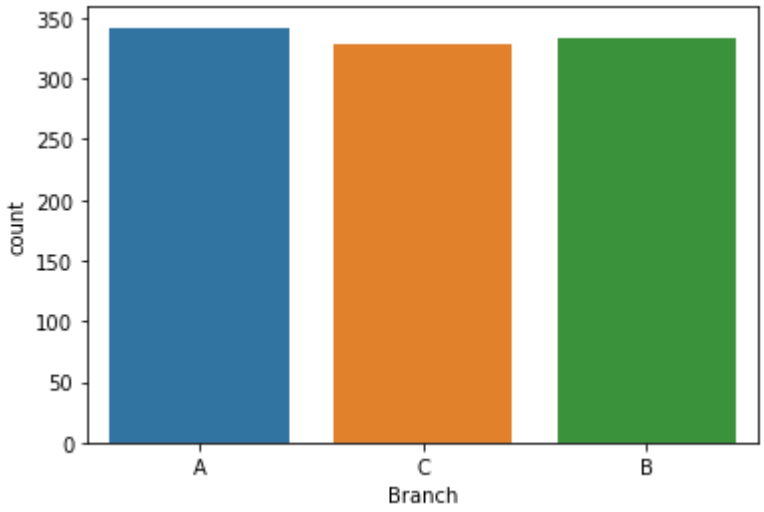
Question 2: Do aggregate sales numbers differ by much between branches?

In [23]:

```
#according to the graph, nope
sns.countplot(df['Branch'])
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8bd813bf50>
```



In [24]:

```
#self explanatory
df['Branch'].value_counts()
```

Out[24]:

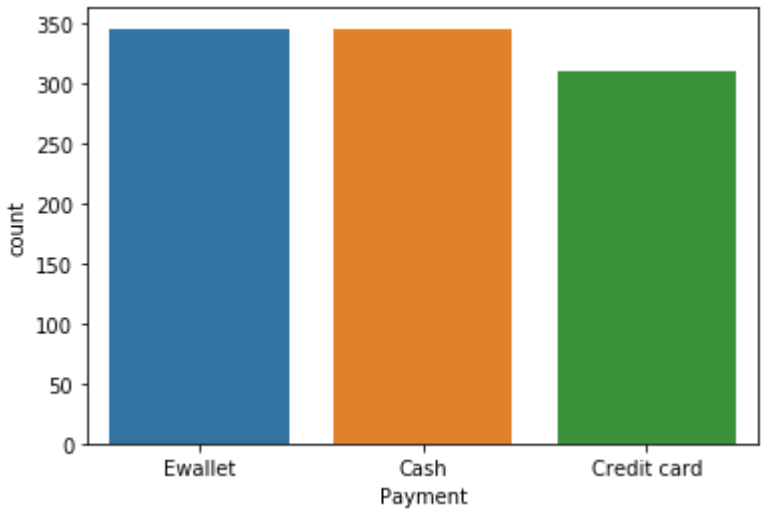
```
A    342
B    333
C    328
Name: Branch, dtype: int64
```

In [25]:

```
sns.countplot(df['Payment'])
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8bd67624d0>
```



Task 3: Bivariate Analysis

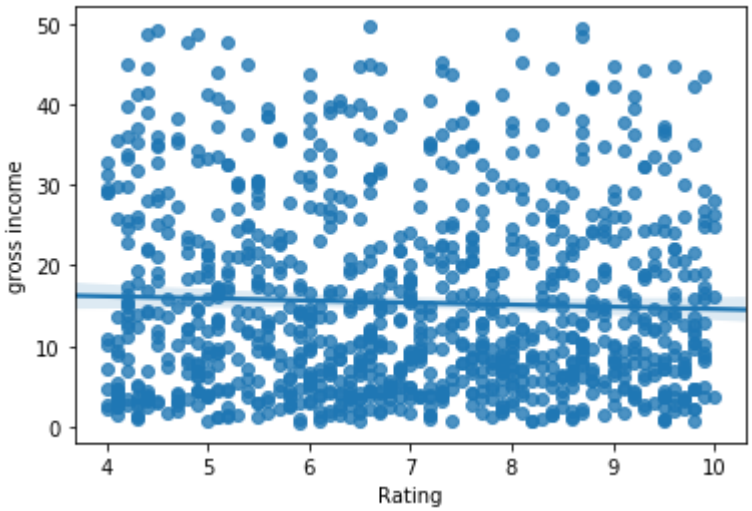
Question 3: Is there a relationship between gross income and customer ratings?

In [29]:

```
sns.regplot(df['Rating'], df['gross income'])
#apparently 0 relationship, since the regression line is literally flat
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8bd6319f50>
```

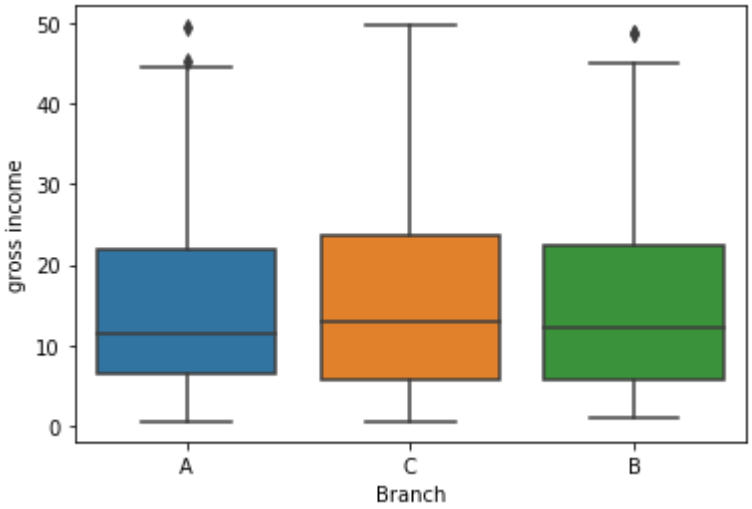


In [30]:

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

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8bd63e04d0>
```

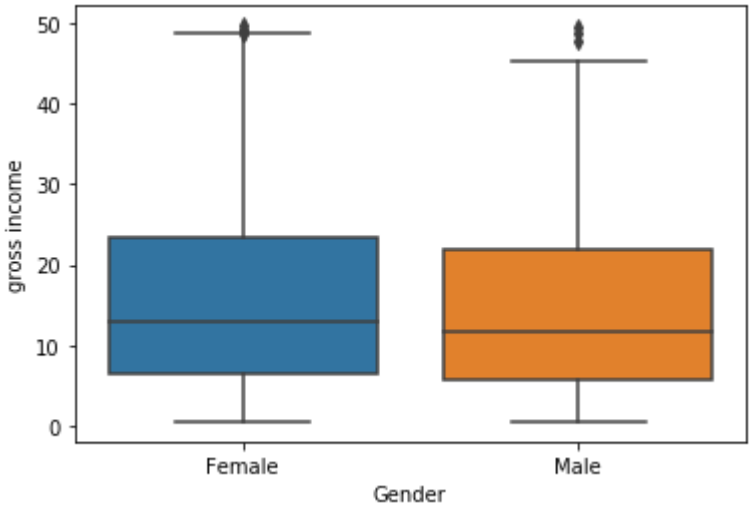


In [31]:

```
#pretty similar, men and women
sns.boxplot(x=df['Gender'], y=df['gross income'])
```

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8bd63e0750>
```



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

In [43]:

```
df.groupby(df.index).mean()
```

Out[43]:

Date	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
2019-01-01	54.995833	6.454545	18.830083	395.431750	376.601667	4.761905	18.830083	6.583333
2019-01-02	44.635000	6.000000	11.580375	243.187875	231.607500	4.761905	11.580375	6.050000
2019-01-03	59.457500	4.625000	12.369813	259.766062	247.396250	4.761905	12.369813	8.112500
2019-01-04	51.743333	5.333333	12.886417	270.614750	257.728333	4.761905	12.886417	6.516667
2019-01-05	61.636667	4.583333	14.034458	294.723625	280.689167	4.761905	14.034458	7.433333
2019-01-06	59.573333	5.777778	19.122778	401.578333	382.455556	4.761905	19.122778	8.177778
2019-01-07	56.898889	5.888889	14.996000	314.916000	299.920000	4.761905	14.996000	7.633333
2019-01-08	48.792778	5.000000	14.004583	294.096250	280.091667	4.761905	14.004583	6.750000
2019-01-09	47.282500	7.250000	17.984188	377.667938	359.683750	4.761905	17.984188	6.137500
2019-01-10	58.051111	6.111111	18.841000	395.661000	376.820000	4.761905	18.841000	6.066667
2019-01-11	59.962500	5.000000	12.589063	264.370313	251.781250	4.761905	12.589063	7.100000
2019-01-12	56.900000	7.272727	22.444864	471.342136	448.897273	4.761905	22.444864	6.845455
2019-01-13	53.150000	4.777778	11.672400	245.120400	233.448000	4.761905	11.672400	6.720000
2019-01-14	56.356923	4.923077	14.529731	305.124346	290.594615	4.761905	14.529731	7.192308
2019-01-15	63.513077	6.769231	21.773846	457.250769	435.476923	4.761905	21.773846	6.561538
2019-01-16	61.212000	6.100000	20.424200	428.908200	408.484000	4.761905	20.424200	6.860000
2019-01-17	60.686364	4.272727	13.605000	285.705000	272.100000	4.761905	13.605000	6.590909
2019-01-18	50.104444	6.000000	14.711500	308.941500	294.230000	4.761905	14.711500	7.622222
2019-01-19	55.493125	5.533333	14.627156	307.170281	292.543125	4.761905	14.627156	7.618750
2019-01-20	62.767000	5.600000	17.406900	365.544900	348.138000	4.761905	17.406900	6.880000
2019-01-21	70.208750	4.000000	14.238687	299.012438	284.773750	4.761905	14.238687	7.375000

Date	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
2019-01-22	47.328571	4.857143	11.597071	243.538500	231.941429	4.761905	11.597071	8.614286
2019-01-23	59.827059	5.588235	16.790441	352.599265	335.808824	4.761905	16.790441	6.970588
2019-01-24	63.929231	6.461538	19.787731	415.542346	395.754615	4.761905	19.787731	6.423077
2019-01-25	56.208824	4.705882	13.166294	276.492176	263.325882	4.761905	13.166294	6.864706
2019-01-26	55.249444	4.375000	13.485583	283.197250	269.711667	4.761905	13.485583	6.733333
2019-01-27	53.985714	6.214286	15.768357	331.135500	315.367143	4.761905	15.768357	7.800000
2019-01-28	49.898571	6.500000	17.005821	357.122250	340.116429	4.761905	17.005821	7.092857
2019-01-29	47.498182	5.583333	13.954625	293.047125	279.092500	4.761905	13.954625	7.450000
2019-01-30	54.185556	4.333333	13.535778	284.251333	270.715556	4.761905	13.535778	6.066667
...	...	...	...	...	...	...	...	...
2019-03-01	50.283000	4.500000	12.544600	263.436600	250.892000	4.761905	12.544600	6.630000
2019-03-02	54.955294	5.294118	17.355306	364.461417	347.106111	4.761905	17.355306	5.938889
2019-03-03	45.857143	6.785714	16.507393	346.655250	330.147857	4.761905	16.507393	6.971429
2019-03-04	50.480000	6.454545	15.454125	324.536625	309.082500	4.761905	15.454125	7.525000
2019-03-05	58.235882	6.058824	17.453441	366.522265	349.068824	4.761905	17.453441	6.723529
2019-03-06	51.197273	6.222222	13.387864	281.145136	267.757273	4.761905	13.387864	7.209091
2019-03-07	47.862222	4.111111	7.609833	159.806500	152.196667	4.761905	7.609833	6.988889
2019-03-08	49.263636	5.454545	13.529818	284.126182	270.596364	4.761905	13.529818	7.618182
2019-03-09	67.346250	6.187500	22.244187	467.127938	444.883750	4.761905	22.244187	7.281250
2019-03-10	49.494545	5.000000	12.033308	252.699462	240.666154	4.761905	12.033308	6.838462
2019-03-11	48.228182	5.454545	12.819273	269.204727	256.385455	4.761905	12.819273	6.554545
2019-03-12	55.510833	5.000000	14.593458	306.462625	291.869167	4.761905	14.593458	7.425000

Date	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
2019-03-13	41.252000	5.700000	9.826700	206.360700	196.534000	4.761905	9.826700	6.430000
2019-03-14	53.251111	6.500000	19.086333	400.813000	381.726667	4.761905	19.086333	6.372222
2019-03-15	52.392500	5.500000	11.676250	245.201250	233.525000	4.761905	11.676250	6.241667
2019-03-16	62.807778	5.222222	16.690333	350.497000	333.806667	4.761905	16.690333	6.955556
2019-03-17	46.331667	6.666667	15.684833	329.381500	313.696667	4.761905	15.684833	6.116667
2019-03-18	57.178571	3.857143	8.794786	184.690500	175.895714	4.761905	8.794786	6.800000
2019-03-19	55.361250	5.666667	17.084500	358.774500	341.690000	4.761905	17.084500	6.881250
2019-03-20	64.494667	5.333333	17.327633	363.880300	346.552667	4.761905	17.327633	6.780000
2019-03-21	47.595000	5.833333	14.900917	312.919250	298.018333	4.761905	14.900917	6.600000
2019-03-22	56.487000	5.200000	15.138800	317.914800	302.776000	4.761905	15.138800	6.780000
2019-03-23	57.154545	6.090909	17.727455	372.276545	354.549091	4.761905	17.727455	7.409091
2019-03-24	60.859091	5.181818	15.053955	316.133045	301.079091	4.761905	15.053955	6.945455
2019-03-25	52.712222	4.666667	12.026278	252.551833	240.525556	4.761905	12.026278	7.600000
2019-03-26	42.972308	4.000000	7.188692	150.962538	143.773846	4.761905	7.188692	6.623077
2019-03-27	56.841000	4.500000	13.822950	290.281950	276.459000	4.761905	13.822950	6.760000
2019-03-28	45.525000	4.800000	10.616200	222.940200	212.324000	4.761905	10.616200	7.050000
2019-03-29	66.346250	6.750000	23.947875	502.905375	478.957500	4.761905	23.947875	6.925000
2019-03-30	67.408182	5.888889	19.424500	407.914500	388.490000	4.761905	19.424500	6.800000

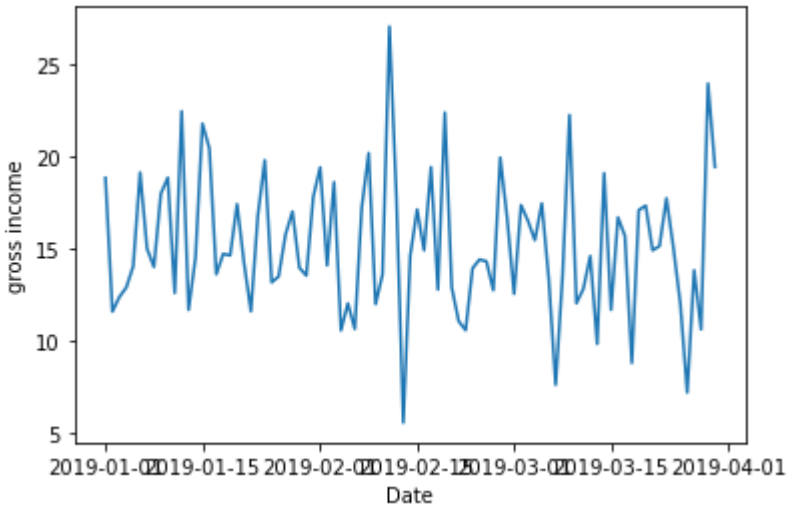
89 rows × 8 columns

In [44]:

```
#a group by is done since there're duplicate dates in our dataset
#basically -> x is mean of our dataframe's index(date), and y, of our gross income
sns.lineplot(x=df.groupby(df.index).mean().index,
             y=df.groupby(df.index).mean()['gross income'])
```

Out[44]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8bd6152d10>
```



Task 4: Dealing With Duplicate Rows and Missing Values

In [46]:

```
#rows duplicated
df.duplicated().sum()
```

Out[46]:

```
3
```

In [47]:

```
#returns those that're duplicated
df[df.duplicated()==True]
```

Out[47]:

Date	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	To
2019-02-18	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7.0	30.919	649.2
2019-03-10	745-74-0715	A	Yangon	Normal	Male	Electronic accessories	NaN	2.0	5.803	121.8
2019-01-26	452-04-8808	B	Mandalay	Normal	Male	Electronic accessories	87.08	NaN	30.478	640.0

In [48]:

```
#self explanatory
df.drop_duplicates(inplace=True)
```

In [50]:

```
df.duplicated().sum()
```

Out[50]:

```
0
```

In [51]:

```
#check if any column has missing values
df.isna().sum()
```

Out[51]:

```
Invoice ID                 0
Branch                     0
City                       0
Customer type             79
Gender                     0
Product line              43
Unit price                 6
Quantity                  19
Tax 5%                     0
Total                      0
Time                       0
Payment                    0
cogs                       0
gross margin percentage    0
gross income               0
Rating                     0
dtype: int64
```

In [52]:

```
#fills the na values with the mean of the column, inplace=True means
#it actually overwrite the dataset, if not specified only change is shown
#but not overwritten
df.fillna(df.mean(), inplace=True)
```

In [53]:

```
#notice here only the quantity has become 0, since customer types and product line
#cuz they obv have no mean
df.isna().sum()
```

Out[53]:

```
Invoice ID                 0
Branch                     0
City                       0
Customer type             79
Gender                     0
Product line              43
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 [57]:

```
#replaces em with the mode(most common one)
df.fillna(df.mode().iloc[0], inplace=True)
```

In [58]:

```
df.isna().sum()
```

Out[58]:

```
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
```

Task 5: Correlation Analysis

In [60]:

```
#coorrelation between gross income and rating
np.corrcoef(df['gross income'], df['Rating'])
```

Out[60]:

```
array([[ 1.       , -0.0364417],
       [-0.0364417,  1.       ]])
```

In [61]:

```
#correlation between everything
df.corr()
```

Out[61]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margi percentag
Unit price	1.000000e+00	1.478639e-02	6.290338e-01	6.290338e-01	6.290338e-01	-4.482629e 1
Quantity	1.478639e-02	1.000000e+00	7.040671e-01	7.040671e-01	7.040671e-01	-8.293717e 1
Tax 5%	6.290338e-01	7.040671e-01	1.000000e+00	1.000000e+00	1.000000e+00	2.461896e-1
Total	6.290338e-01	7.040671e-01	1.000000e+00	1.000000e+00	1.000000e+00	2.408632e-1
cogs	6.290338e-01	7.040671e-01	1.000000e+00	1.000000e+00	1.000000e+00	1.439279e-1
gross margin percentage	-4.482629e-16	-8.293717e-17	2.461896e-16	2.408632e-16	1.439279e-15	1.000000e+0
gross income	6.290338e-01	7.040671e-01	1.000000e+00	1.000000e+00	1.000000e+00	2.461896e-1
Rating	-6.601308e-03	-2.122532e-02	-3.644170e-02	-3.644170e-02	-3.644170e-02	2.042714e-1

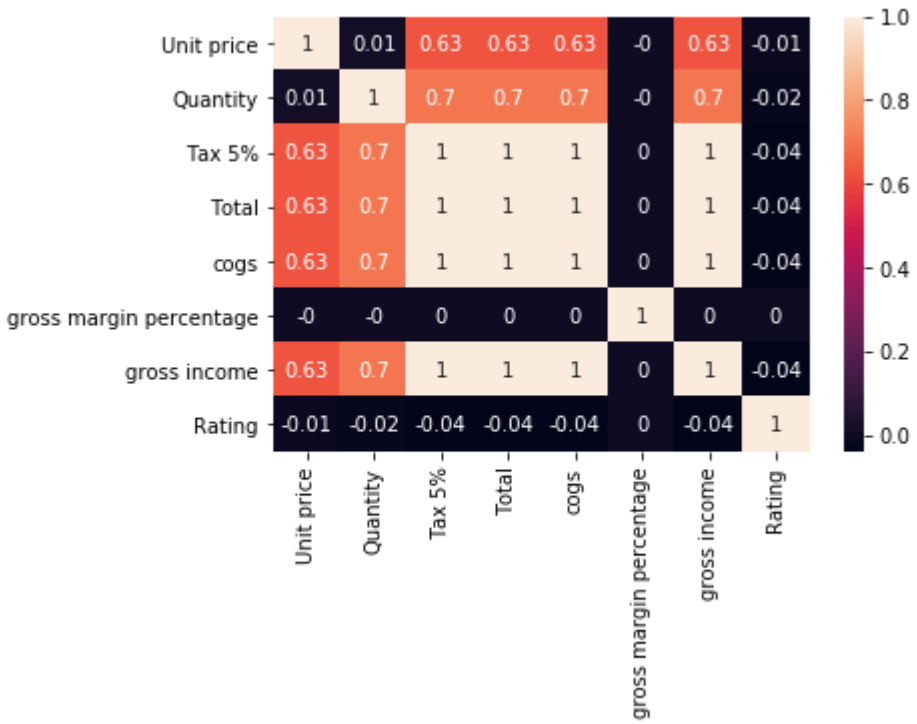
Helpful Links

In [64]:

```
sns.heatmap(np.round(df.corr(), 2), annot=True)
```

Out[64]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8bd607c290>
```



1. More visualizations: https://www.data-to-viz.com/ (https://www.data-to-viz.com/)
2. Seaborn gallery: https://seaborn.pydata.org/examples/index.html (https://seaborn.pydata.org/examples/index.html)
3. Pandas profiling documentation: https://pypi.org/project/pandas-profiling/ (https://pypi.org/project/pandas-profiling/)