Supermarket Sales Basic EDA

In this tutorial, we will examine the Supermarket sales data set using basic tools and statistics with Python.
We will do this with Python Basic EDA and visualize the data with Python such as Matplot and Seaborn libraries.
Content:

1. IMPORT LIBRARIES AND FIRST LOOK AT THE DATA

In [108]:

```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

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      # visualization tool

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/supermarket-sales/supermarket_sales - Sheet1.csv
```

In [109]:

```python
# Read csv document
df = pd.read_csv('/kaggle/input/supermarket-sales/supermarket_sales - Sheet1.csv')
```

In [110]:

```python
# find types
series = df['Gender']          # data['Defense'] = series
print(type(series))
data_frame = df[['Gender']]      # data[['Defense']] = data frame
print(type(data_frame))
```

```
<class 'pandas.core.series.Series'>
<class 'pandas.core.frame.DataFrame'>
```

In [111]:

```
# show 5 values from the beginning - default value is 5 - olso you can use like this example:  df.head(10)  ect.
df.head()
```

Out[111]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.141
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.8200
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.215
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.288
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30.208

In [112]:

```
# show last 5 values - default value is 5 - olso you can use like this example:  df.tail(100)  ect.
df.tail()
```

Out[112]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gro inc
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	42.3675	1/29/2019	13:46	Ewallet	40.35	4.761905	2.0
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4900	3/2/2019	17:16	Ewallet	973.80	4.761905	48
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.5920	33.4320	2/9/2019	13:22	Cash	31.84	4.761905	1.5
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.2910	69.1110	2/22/2019	15:33	Cash	65.82	4.761905	3.2
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190	649.2990	2/18/2019	13:28	Cash	618.38	4.761905	30

In [113]:

```
#how many rows, columns information
df.shape
```

Out[113]:

```
(1000, 17)
```

In [114]:

```
# information about data
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 [115]:

```
# names information of data variables
df.columns
```

Out[115]:

```
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 [116]:

```
# index information
df.index
```

Out[116]:

```
RangeIndex(start=0, stop=1000, step=1)
```

In [117]:

```
# summary statistical information
df.describe()
```

Out[117]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000	1000.000000	1000.00000
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905	15.379369	6.97270
std	26.494628	2.923431	11.708825	245.885335	234.17651	0.000000	11.708825	1.71858
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905	0.508500	4.00000
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905	5.924875	5.50000
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905	12.088000	7.00000
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905	22.445250	8.50000
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905	49.650000	10.00000

In [118]:

```
# transpose it - makes it easier to read
df.describe().T
```

Out[118]:

	count	mean	std	min	25%	50%	75%	max
Unit price	1000.0	55.672130	26.494628	10.080000	32.875000	55.230000	77.935000	99.960000
Quantity	1000.0	5.510000	2.923431	1.000000	3.000000	5.000000	8.000000	10.000000
Tax 5%	1000.0	15.379369	11.708825	0.508500	5.924875	12.088000	22.445250	49.650000
Total	1000.0	322.966749	245.885335	10.678500	124.422375	253.848000	471.350250	1042.650000
cogs	1000.0	307.587380	234.176510	10.170000	118.497500	241.760000	448.905000	993.000000
gross margin percentage	1000.0	4.761905	0.000000	4.761905	4.761905	4.761905	4.761905	4.761905
gross income	1000.0	15.379369	11.708825	0.508500	5.924875	12.088000	22.445250	49.650000
Rating	1000.0	6.972700	1.718580	4.000000	5.500000	7.000000	8.500000	10.000000

In [119]:

```
# check NaN values
df.isnull()
```

Out[119]:

	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
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
996	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
997	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
998	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
999	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

1000 rows × 17 columns

In [120]:

```
# check NaN values and show as a array
df.isnull().values
```

Out[120]:

```
array([[False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       ...,
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False]])
```

In [121]:

```
# find missing data in dataset
df.isnull().values.any()
```

Out[121]:

```
False
```

In [122]:

```
# sum of missing data for each variable
df.isnull().sum()
```

Out[122]:

```
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
```

In [123]:

```
# Is the Gender variable in df
'Gender' in df
```

Out[123]:

```
True
```

In [124]:

```
# show 5 values from the beginning in the Gender variable
df['Gender'].head()
```

Out[124]:

```
0    Female
1    Female
2      Male
3      Male
4      Male
Name: Gender, dtype: object
```

In [125]:

```
# show 5 values from the beginning in the Gender variable - another option
df.Gender.head()
```

Out[125]:

```
0    Female
1    Female
2      Male
3      Male
4      Male
Name: Gender, dtype: object
```

In [126]:

```
# Groups the variables according to their classes and finds how many of each are there?
df['Gender'].value_counts()
```

Out[126]:

```
Gender
Female    501
Male      499
Name: count, dtype: int64
```

In [127]:

```
# slicing from 0 to 13 (13th not included)
df[0:13]
```

Out[127]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross incom
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.14
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.820
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.21
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.28
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30.20
5	699-14-3026	C	Naypyitaw	Normal	Male	Electronic accessories	85.39	7	29.8865	627.6165	3/25/2019	18:30	Ewallet	597.73	4.761905	29.88
6	355-53-5943	A	Yangon	Member	Female	Electronic accessories	68.84	6	20.6520	433.6920	2/25/2019	14:36	Ewallet	413.04	4.761905	20.65
7	315-22-5665	C	Naypyitaw	Normal	Female	Home and lifestyle	73.56	10	36.7800	772.3800	2/24/2019	11:38	Ewallet	735.60	4.761905	36.78
8	665-32-9167	A	Yangon	Member	Female	Health and beauty	36.26	2	3.6260	76.1460	1/10/2019	17:15	Credit card	72.52	4.761905	3.626
9	692-92-5582	B	Mandalay	Member	Female	Food and beverages	54.84	3	8.2260	172.7460	2/20/2019	13:27	Credit card	164.52	4.761905	8.226
10	351-62-0822	B	Mandalay	Member	Female	Fashion accessories	14.48	4	2.8960	60.8160	2/6/2019	18:07	Ewallet	57.92	4.761905	2.896
11	529-56-3974	B	Mandalay	Member	Male	Electronic accessories	25.51	4	5.1020	107.1420	3/9/2019	17:03	Cash	102.04	4.761905	5.102
12	365-64-0515	A	Yangon	Normal	Female	Electronic accessories	46.95	5	11.7375	246.4875	2/12/2019	10:25	Ewallet	234.75	4.761905	11.73

In [128]:

```
# delete in rows (axis = 0), if axis=1 it deletes from columns
df.drop(0, axis=0).head()
```

Out[128]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.8200
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.215
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.288
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30.208
5	699-14-3026	C	Naypyitaw	Normal	Male	Electronic accessories	85.39	7	29.8865	627.6165	3/25/2019	18:30	Ewallet	597.73	4.761905	29.886

In [129]:

```
# type of Gender
type(df['Gender'].head())
```

Out[129]:

```
pandas.core.series.Series
```

In [130]:

```
# selecting multiple variables from dataframe
df[['City', 'Gender']]
```

Out[130]:

	City	Gender
0	Yangon	Female
1	Naypyitaw	Female
2	Yangon	Male
3	Yangon	Male
4	Yangon	Male
...	...	...
995	Naypyitaw	Male
996	Mandalay	Female
997	Yangon	Male
998	Yangon	Male
999	Yangon	Female

1000 rows × 2 columns

In [131]:

```
# add new columns in df
df['Unit price 2'] = df['Unit price'] ** 1.01
df.head()
```

Out[131]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.141
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.8200
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.215
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.288
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30.208

In [132]:

```
# delete Unite price 2
df.drop('Unit price 2', axis = 1, inplace=True)       # inplace= True > makes it permament
```

In [133]:

```
df.head()     # check again to df for deleted value
```

Out[133]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.141
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.8200
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.215
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.288
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30.208

2. LISTS

List in Python is an ordered and mutable data structure. Lists allow you to keep multiple data items (which can also be of different data types) in a single variable.

In [134]:

```
# lists
col_names = ['Invoice ID', 'City', 'Gender', 'Product line']
df[col_names]
```

Out[134]:

	Invoice ID	City	Gender	Product line
0	750-67-8428	Yangon	Female	Health and beauty
1	226-31-3081	Naypyitaw	Female	Electronic accessories
2	631-41-3108	Yangon	Male	Home and lifestyle
3	123-19-1176	Yangon	Male	Health and beauty
4	373-73-7910	Yangon	Male	Sports and travel
...	...	...	...	...
995	233-67-5758	Naypyitaw	Male	Health and beauty
996	303-96-2227	Mandalay	Female	Home and lifestyle
997	727-02-1313	Yangon	Male	Food and beverages
998	347-56-2442	Yangon	Male	Home and lifestyle
999	849-09-3807	Yangon	Female	Fashion accessories

1000 rows × 4 columns

In [135]:

```
# deleting multiple variables

col_names = ['Invoice ID', 'City', 'Gender', 'Product line']
df[col_names]
df.drop(col_names, axis = 1)    # # inplace= True > makes it permament  > df.drop(col_names, axis = 1, inplace=True)
```

Out[135]:

	Branch	Customer type	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating
0	A	Member	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.1415	9.1
1	C	Normal	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.8200	9.6
2	A	Normal	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.2155	7.4
3	A	Member	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.2880	8.4
4	A	Normal	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30.2085	5.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	C	Normal	40.35	1	2.0175	42.3675	1/29/2019	13:46	Ewallet	40.35	4.761905	2.0175	6.2
996	B	Normal	97.38	10	48.6900	1022.4900	3/2/2019	17:16	Ewallet	973.80	4.761905	48.6900	4.4
997	A	Member	31.84	1	1.5920	33.4320	2/9/2019	13:22	Cash	31.84	4.761905	1.5920	7.7
998	A	Normal	65.82	1	3.2910	69.1110	2/22/2019	15:33	Cash	65.82	4.761905	3.2910	4.1
999	A	Member	88.34	7	30.9190	649.2990	2/18/2019	13:28	Cash	618.38	4.761905	30.9190	6.6

1000 rows × 13 columns

3. LOC & ILOC

They both function to select and access data, but they work in different ways

In [136]:

```
df.loc[0:3]        # slicing - 3rd included
```

Out[136]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.141
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.8200
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.215
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.288

In [137]:

```
df.iloc[0:3]           # slicing - 3rd not included
```

Out[137]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.1415
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.8200
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.2155

In [138]:

```
df.iloc[0,3]       # select the 0th row 3rd column value
```

Out[138]:

```
'Member'
```

In [139]:

```
df.iloc[0:4, 4]       # Fetching gender information within a certain range with iloc
```

Out[139]:

```
0    Female
1    Female
2      Male
3      Male
Name: Gender, dtype: object
```

In [140]:

```
df.loc[0 : 3 , 'Gender']       # Fetching gender information within a certain range with loc
```

Out[140]:

```
0    Female
1    Female
2      Male
3      Male
Name: Gender, dtype: object
```

In [141]:

```
# selecting variables containing a certain string expression in the data set with LOC
df.loc[:, df.columns.str.contains("Unit price")].head()
```

Out[141]:

	Unit price
0	74.69
1	15.28
2	46.33
3	58.22
4	86.31

In [142]:

```
# delete variables containing a certain string expression in the data set
df.loc[:, ~ df.columns.str.contains("Unit price")].head()        # with keyboard  ~  = ALT + 0126
```

Out[142]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rati
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.1415	9.1
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.8200	9.6
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.2155	7.4
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.2880	8.4
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30.2085	5.3

4. DICTIONARY

In Python, a dictionary is a data structure that stores key-value pairs. Dictionaries are used to store and access data quickly and efficiently. Each key is linked to a value, and thanks to this association, the data becomes more meaningful.

In [143]:

```
# dictionary     : keys and values
dictionary = {'Customer type': 'Member', 'Payment': 'Ewallet'}
print(dictionary)
print(dictionary.keys())        # dictionary keys
print(dictionary.values())      # dictionary values
```

```
{'Customer type': 'Member', 'Payment': 'Ewallet'}
dict_keys(['Customer type', 'Payment'])
dict_values(['Member', 'Ewallet'])
```

In [144]:

```python
dictionary['Customer type'] = 'Normal'      # change/update key information
print(dictionary)

dictionary['Branch'] = 'A'        # add new value and key to dictionary
print(dictionary)

del dictionary['Branch']          # delete in dictionary
print(dictionary)

print('Payment' in dictionary)     # Is Payment in dictionary?

dictionary.clear()                 # clear dictionary
print(dictionary)
```

```
{'Customer type': 'Normal', 'Payment': 'Ewallet'}
{'Customer type': 'Normal', 'Payment': 'Ewallet', 'Branch': 'A'}
{'Customer type': 'Normal', 'Payment': 'Ewallet'}
True
{}
```

5. WHILE and FOR LOOPS

Loops allow a particular block of code in programming to be executed again based on certain conditions.

In [145]:

```python
# while loop
i = 0
while i != 5:
    print('i is: ', i)
    i += 1
print(i, 'i is equal to 5')
```

```
i is:  0
i is:  1
i is:  2
i is:  3
i is:  4
5 i is equal to 5
```

In [146]:

```
# for loop
liste = [1, 2, 3, 4, 5]     # create a list
for i in liste:
    print('i is: ', i)
print('')


# for loop
# Enumerate index and value of list
# index: value = 0:1, 1:2, 2:3, 3:4, 4:5
for index, value in enumerate(liste):
    print(index, ':', value)
print('')


# for dictionaries
# We can use for loop to achieve key and value of dictionary. We learnt key and value at dictionary part.
dictionary = {'Customer type': 'Member', 'Payment': 'Ewallet'}
for key, value in dictionary.items():
    print(key, ':', value)
print('')

# For pandas we can achieve index and value
for index, value in df[['Gender']][0:1].iterrows():
    print(index, ':', value)
```

```
i is:  1
i is:  2
i is:  3
i is:  4
i is:  5

0 : 1
1 : 2
2 : 3
3 : 4
4 : 5

Customer type : Member
Payment : Ewallet

0 : Gender    Female
Name: 0, dtype: object
```

6. FILTERING

Filtering in Python means selecting data from a data set that meets certain criteria.

In [147]:

```
# Filtering
x = df['Unit price'] > 80        # Unit price greather than 80 values
df[x]
```

Out[147]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	grc inc
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30
5	699-14-3026	C	Naypyitaw	Normal	Male	Electronic accessories	85.39	7	29.8865	627.6165	3/25/2019	18:30	Ewallet	597.73	4.761905	29
15	299-46-1805	B	Mandalay	Member	Female	Sports and travel	93.72	6	28.1160	590.4360	1/15/2019	16:19	Cash	562.32	4.761905	28
20	300-71-4605	C	Naypyitaw	Member	Male	Electronic accessories	86.04	5	21.5100	451.7100	2/25/2019	11:24	Ewallet	430.20	4.761905	21
21	371-85-5789	B	Mandalay	Normal	Male	Health and beauty	87.98	3	13.1970	277.1370	3/5/2019	10:40	Ewallet	263.94	4.761905	13
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
983	148-41-7930	C	Naypyitaw	Normal	Male	Health and beauty	99.96	7	34.9860	734.7060	1/23/2019	10:33	Cash	699.72	4.761905	34
984	189-40-5216	C	Naypyitaw	Normal	Male	Electronic accessories	96.37	7	33.7295	708.3195	1/9/2019	11:40	Cash	674.59	4.761905	33
988	267-62-7380	C	Naypyitaw	Member	Male	Electronic accessories	82.34	10	41.1700	864.5700	3/29/2019	19:12	Ewallet	823.40	4.761905	41
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4900	3/2/2019	17:16	Ewallet	973.80	4.761905	48
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190	649.2990	2/18/2019	13:28	Cash	618.38	4.761905	30

232 rows × 17 columns

In [148]:

```
df[np.logical_and(df['Unit price'] > 80, df['Rating'] > 8)]    # Unit price greater than 80 and Rating greater than 8 values
```

Out[148]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gros inco
28	145-94-9061	B	Mandalay	Normal	Female	Food and beverages	88.36	5	22.0900	463.8900	1/25/2019	19:48	Cash	441.80	4.761905	22.0
45	132-32-9879	B	Mandalay	Member	Female	Electronic accessories	93.96	4	18.7920	394.6320	3/9/2019	18:00	Cash	375.84	4.761905	18.7
50	326-78-5178	C	Naypyitaw	Member	Male	Food and beverages	91.40	7	31.9900	671.7900	2/3/2019	10:19	Cash	639.80	4.761905	31.9
55	399-46-5918	C	Naypyitaw	Normal	Female	Electronic accessories	85.98	8	34.3920	722.2320	2/28/2019	19:01	Cash	687.84	4.761905	34.3
67	109-28-2512	B	Mandalay	Member	Female	Fashion accessories	97.61	6	29.2830	614.9430	1/7/2019	15:01	Ewallet	585.66	4.761905	29.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
959	384-59-6655	A	Yangon	Member	Female	Food and beverages	98.66	9	44.3970	932.3370	2/19/2019	15:07	Cash	887.94	4.761905	44.3
960	256-58-3609	C	Naypyitaw	Member	Male	Fashion accessories	91.98	1	4.5990	96.5790	3/18/2019	15:29	Cash	91.98	4.761905	4.59
967	195-06-0432	A	Yangon	Member	Male	Home and lifestyle	81.01	3	12.1515	255.1815	1/13/2019	12:55	Credit card	243.03	4.761905	12.1
970	746-04-1077	B	Mandalay	Member	Female	Food and beverages	84.63	10	42.3150	888.6150	1/1/2019	11:36	Credit card	846.30	4.761905	42.3
974	744-82-9138	C	Naypyitaw	Normal	Male	Fashion accessories	86.13	2	8.6130	180.8730	2/7/2019	17:59	Cash	172.26	4.761905	8.61

73 rows × 17 columns

In [149]:

```
df[(df['Unit price'] > 80) & (df['Rating'] > 8)]        # Bring Unit price greater than 80 and Rating greater than 8 (another wa
y)
```

Out[149]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gros inco
28	145-94-9061	B	Mandalay	Normal	Female	Food and beverages	88.36	5	22.0900	463.8900	1/25/2019	19:48	Cash	441.80	4.761905	22.0
45	132-32-9879	B	Mandalay	Member	Female	Electronic accessories	93.96	4	18.7920	394.6320	3/9/2019	18:00	Cash	375.84	4.761905	18.7
50	326-78-5178	C	Naypyitaw	Member	Male	Food and beverages	91.40	7	31.9900	671.7900	2/3/2019	10:19	Cash	639.80	4.761905	31.9
55	399-46-5918	C	Naypyitaw	Normal	Female	Electronic accessories	85.98	8	34.3920	722.2320	2/28/2019	19:01	Cash	687.84	4.761905	34.3
67	109-28-2512	B	Mandalay	Member	Female	Fashion accessories	97.61	6	29.2830	614.9430	1/7/2019	15:01	Ewallet	585.66	4.761905	29.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
959	384-59-6655	A	Yangon	Member	Female	Food and beverages	98.66	9	44.3970	932.3370	2/19/2019	15:07	Cash	887.94	4.761905	44.3
960	256-58-3609	C	Naypyitaw	Member	Male	Fashion accessories	91.98	1	4.5990	96.5790	3/18/2019	15:29	Cash	91.98	4.761905	4.59
967	195-06-0432	A	Yangon	Member	Male	Home and lifestyle	81.01	3	12.1515	255.1815	1/13/2019	12:55	Credit card	243.03	4.761905	12.1
970	746-04-1077	B	Mandalay	Member	Female	Food and beverages	84.63	10	42.3150	888.6150	1/1/2019	11:36	Credit card	846.30	4.761905	42.3
974	744-82-9138	C	Naypyitaw	Normal	Male	Fashion accessories	86.13	2	8.6130	180.8730	2/7/2019	17:59	Cash	172.26	4.761905	8.61

73 rows × 17 columns

In [150]:

```
df.loc[(df['City'] == 'Yangon') & (df['Gender'] == 'Female'), ['Product line', 'Payment']]    #Get Product Line and Payment info
rmation for City = Yangon and Gender = Female
```

Out[150]:

	Product line	Payment
0	Health and beauty	Ewallet
6	Electronic accessories	Ewallet
8	Health and beauty	Credit card
12	Electronic accessories	Ewallet
14	Health and beauty	Cash
...	...	...
968	Health and beauty	Cash
976	Food and beverages	Cash
982	Sports and travel	Ewallet
990	Food and beverages	Credit card
999	Fashion accessories	Cash

161 rows × 2 columns

7. CORRELATION MAP

A correlation map is a table that shows correlation relationships between variables in a data set, often visualized as a heatmap.

In [151]:

```
df.drop('gross margin percentage', axis = 1, inplace=True)      # permanently dropped gross margin percentage column
```

In [152]:

```
df.head()        # check dropped values for df
```

Out[152]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross income	Rating
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	26.1415	9.1
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	3.8200	9.6
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	16.2155	7.4
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	23.2880	8.4
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	30.2085	5.3

In [153]:

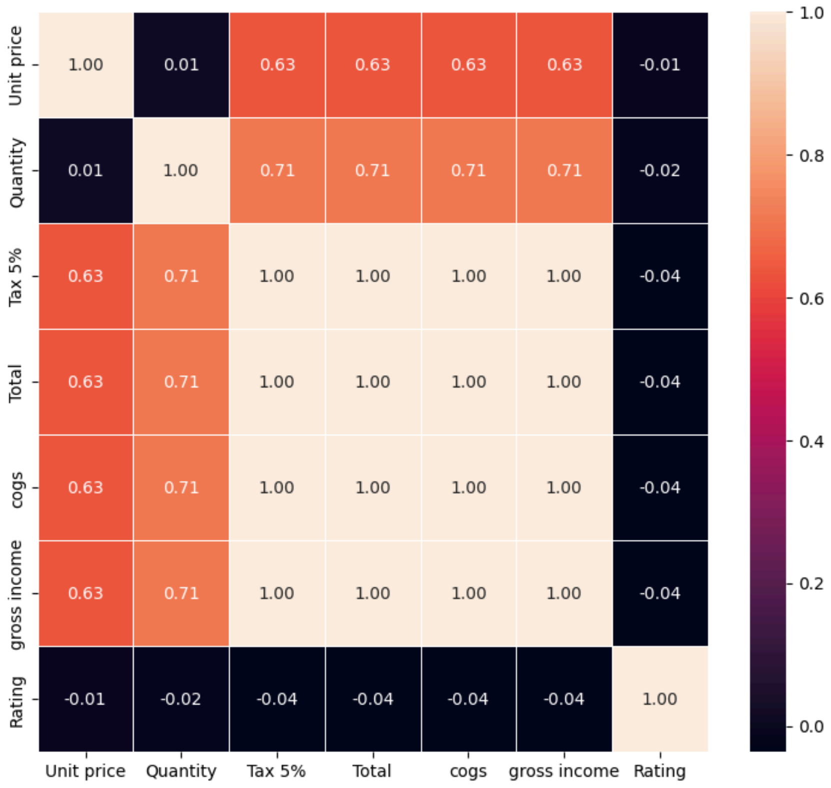
```
df.corr(numeric_only = True)          # calculates correlation coefficients between numeric columns
```

Out[153]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross income	Rating
Unit price	1.000000	0.010778	0.633962	0.633962	0.633962	0.633962	-0.008778
Quantity	0.010778	1.000000	0.705510	0.705510	0.705510	0.705510	-0.015815
Tax 5%	0.633962	0.705510	1.000000	1.000000	1.000000	1.000000	-0.036442
Total	0.633962	0.705510	1.000000	1.000000	1.000000	1.000000	-0.036442
cogs	0.633962	0.705510	1.000000	1.000000	1.000000	1.000000	-0.036442
gross income	0.633962	0.705510	1.000000	1.000000	1.000000	1.000000	-0.036442
Rating	-0.008778	-0.015815	-0.036442	-0.036442	-0.036442	-0.036442	1.000000

In [154]:

```
# visualize correlation map

f, ax = plt.subplots(figsize =(9,8))
sns.heatmap(df.corr(numeric_only=True), annot=True, linewidths=.5, fmt = '0.2f', ax=ax)
plt.show()
```



8. MATPLOTLIB

Matplot is a python library that help us to plot data. The easiest and most basic plots are line, scatter and histogram plots.

In [155]:

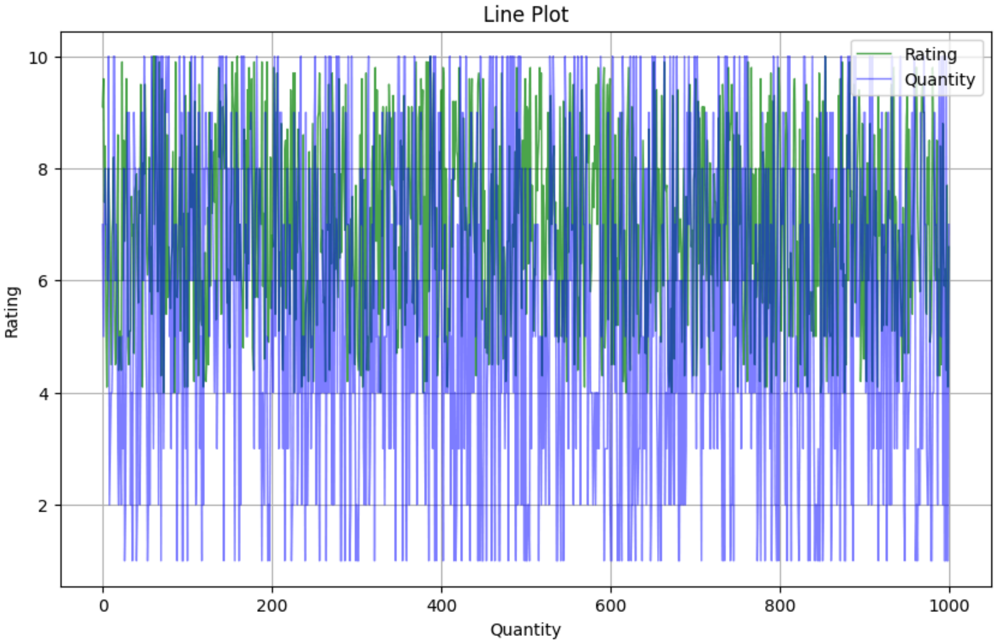
```
df.head()        # quick look at columns for get information in df
```

Out[155]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross income	Rating
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	26.1415	9.1
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	3.8200	9.6
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	16.2155	7.4
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	23.2880	8.4
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	30.2085	5.3

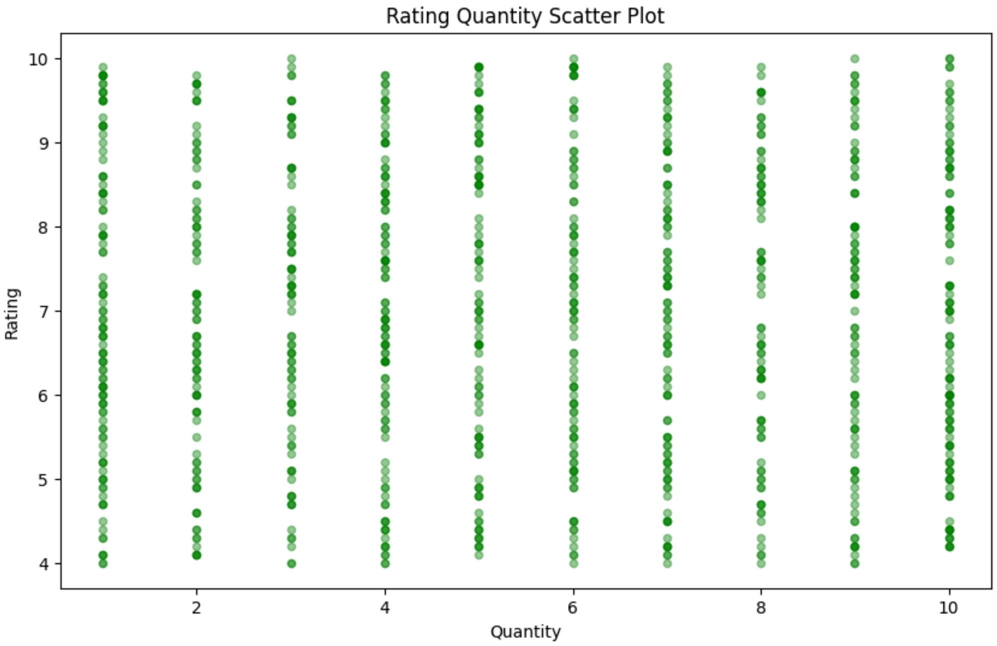
In [156]:

```
# Line Plot    -- Rating to Quality plots

df.Rating.plot(kind='line', color='g', label = 'Rating', linewidth=1, alpha = 0.7, grid = True, figsize= (10, 6))  # line plot for Rating value
df.Quantity.plot(color = 'b', label = 'Quantity', linewidth=1, alpha = 0.5, grid = True )          # line plot for Quantity value
plt.legend(loc = 'upper right')      # location of
plt.xlabel('Quantity')      # label = name of label
plt.ylabel('Rating')        # label = name of label
plt.title('Line Plot')      # title = title of plot
plt.show()
```



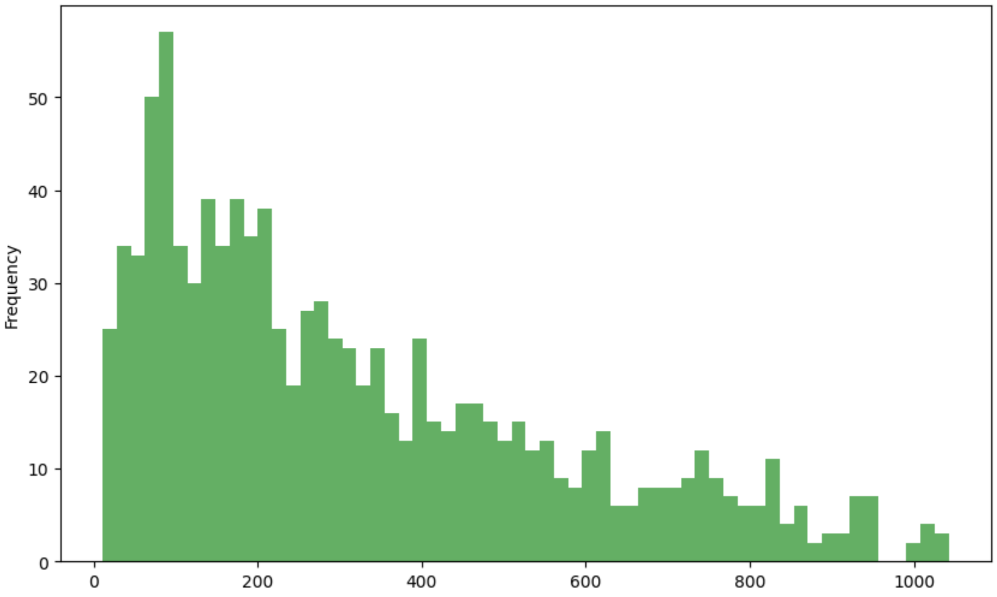
In [157]:

```
# Scatter Plot
# x = attack, y = defense

df.plot(kind='scatter', x='Quantity', y='Rating', alpha = 0.4, color = 'g', figsize = (10,6))
plt.xlabel('Quantity')              # label = name of label
plt.ylabel('Rating')                # label = name of label
plt.title('Rating Quantity Scatter Plot')        # title = title of plot
plt.show()
```



In [158]:

```
# Histogram

df.Total.plot(kind='hist', bins = 60, alpha = 0.6, figsize = (10,6), color = 'g')
plt.show()
```



The end of the notebook. Thank you for your visit.