Loading libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

1- Gathering Data

1.1 Loading "Supermarket Sales"

In [2]:
```
df = pd.read_csv(r"Capstone Data - Supermarket Sales.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	Invoice ID	Branch	Yangon	Naypyitaw	Mandalay	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	
0	750-67-8428	A	1	0	0	Normal	Male	Health and beauty	74.69	7	26.1415	
1	226-31-3081	C	0	1	0	Normal	Male	Electronic accessories	15.28	5	3.8200	8
2	631-41-3108	A	1	0	0	Normal	Male	Home and lifestyle	46.33	7	16.2155	34
3	123-19-1176	A	1	0	0	Normal	Male	Health and beauty	58.22	8	NaN	48
4	373-73-7910	A	1	0	0	Normal	Male	Sports and travel	86.31	7	30.2085	63

2. Data Assessing

2.1 Quality issues

In [4]:
```
df.head()
```

Out[4]:

	Invoice ID	Branch	Yangon	Naypyitaw	Mandalay	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	
0	750-67-8428	A	1	0	0	Normal	Male	Health and beauty	74.69	7	26.1415	
1	226-31-3081	C	0	1	0	Normal	Male	Electronic accessories	15.28	5	3.8200	8
2	631-41-3108	A	1	0	0	Normal	Male	Home and lifestyle	46.33	7	16.2155	34
3	123-19-1176	A	1	0	0	Normal	Male	Health and beauty	58.22	8	NaN	48
4	373-73-7910	A	1	0	0	Normal	Male	Sports and travel	86.31	7	30.2085	63

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1006 entries, 0 to 1005
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Invoice ID     1006 non-null   object 
 1   Branch         1006 non-null   object 
 2   Yangon         1006 non-null   int64  
 3   Naypyitaw      1006 non-null   int64  
 4   Mandalay       1006 non-null   int64  
 5   Customer type  1006 non-null   object 
 6   Gender         1006 non-null   object 
 7   Product line   1006 non-null   object 
 8   Unit price     1006 non-null   object 
 9   Quantity       1006 non-null   int64  
 10  Tax 5%         997 non-null    float64
 11  Total          1003 non-null   float64
 12  Date           1006 non-null   object 
 13  Time           1006 non-null   object 
 14  Payment        1006 non-null   object 
 15  Rating         1006 non-null   float64
dtypes: float64(3), int64(4), object(9)
memory usage: 125.9+ KB
```

In [6]:
```
df.shape
```

Out[6]:
```
(1006, 16)
```

In [7]:
```
df.describe().T
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
Yangon	1006.0	0.338966	0.473594	0.0000	0.00000	0.0000	1.0000	1.00
Naypyitaw	1006.0	0.329026	0.470093	0.0000	0.00000	0.0000	1.0000	1.00
Mandalay	1006.0	0.332008	0.471168	0.0000	0.00000	0.0000	1.0000	1.00
Quantity	1006.0	5.469185	3.014153	-8.0000	3.00000	5.0000	8.0000	10.00
Tax 5%	997.0	15.479682	11.728320	0.5085	5.98650	12.2275	22.7205	49.65
Total	1003.0	322.734689	245.865964	10.6785	123.78975	254.0160	471.0090	1042.65
Rating	1006.0	7.056163	3.318751	4.0000	5.50000	7.0000	8.5000	97.00

In [8]:
```
df.isna().sum()
```

Out[8]:
```
Invoice ID       0
Branch           0
Yangon           0
Naypyitaw        0
Mandalay         0
Customer type    0
Gender           0
Product line     0
Unit price       0
Quantity         0
Tax 5%           9
Total            3
Date             0
Time             0
Payment          0
Rating           0
dtype: int64
```

In [9]:
```
df.duplicated().sum()
```

Out[9]:
```
6
```

Missing Values

- Need to handle missing values in "Total"
- Need to handle missing values in "Tax"

Duplicate Records

- 6 columns.

Inaccurate Values

- "97" in Rating.

Validity

- Data Type of Unit price id must be Int.
- Data Type of Date col must be Date.
- Data Type of Time col must be time.
- (-) in Quantity must be deleted.

Inconsistancy

- The time "8-30 PM" is incorrectly formatted and should adhere to a consistent format.
- The column should be modified to better reflect city names than branches.
- In Price column Must remove "USD" from the rows that contain it to standardize the data type in the column as only numerical values .

Spelling Errors

- Spelling error:
- word "memberr" instead of "member"

- "-" instead of "member"
- Using incorrect symbols: like "-" in the "Customer" column instead of the correct text.

2.2 Tidiness issues

The issue is that there are three separate columns for cities: Yangon, Naypyitaw, and Mandalay. According to the tidy data principle, there should be a single City column with the corresponding values, not separate columns for each city.

3. Data Cleaning

3.1 Fixing Tidiness Issues

In [10]:
```
df.head()
```

Out[10]:

	Invoice ID	Branch	Yangon	Naypyitaw	Mandalay	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	
0	750-67-8428	A	1	0	0	Normal	Male	Health and beauty	74.69	7	26.1415	
1	226-31-3081	C	0	1	0	Normal	Male	Electronic accessories	15.28	5	3.8200	8
2	631-41-3108	A	1	0	0	Normal	Male	Home and lifestyle	46.33	7	16.2155	34
3	123-19-1176	A	1	0	0	Normal	Male	Health and beauty	58.22	8	NaN	48
4	373-73-7910	A	1	0	0	Normal	Male	Sports and travel	86.31	7	30.2085	63

A-Define:

- We need to melt the data and make a new colum for city

In [11]:
```
df.iloc[:, 1:5]
```

Out[11]:

	Branch	Yangon	Naypyitaw	Mandalay
0	A	1	0	0
1	C	0	1	0
2	A	1	0	0
3	A	1	0	0
4	A	1	0	0
...	...	...	...	...
1001	C	0	1	0
1002	B	0	0	1
1003	C	0	1	0
1004	B	0	0	1
1005	A	1	0	0

1006 rows × 4 columns

B- Code :

In [12]:
```
df = pd.melt(df, id_vars=['Invoice ID', 'Branch', 'Customer type', 'Gender', 'Product li
                          'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Date',
                          'Time', 'Payment', 'Rating'],
             value_vars=['Yangon', 'Naypyitaw', 'Mandalay'],
             var_name='city', value_name='city_flag')
# Filter out the rows where the flag is 0 (i.e., not that city)
df= df[df['city_flag'] == 1]

# Drop the 'city_flag' column as it's no longer needed
df= df.drop(columns=['city_flag'])
```

C- Test :

In [13]:
```
df.sample(10)
```

Out[13]:

	Invoice ID	Branch	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Pay
1928	838-02-1821	C	Member	Female	Home and lifestyle	12.73	2	1.2730	26.7330	2/22/2019	12:10	
2394	868-52-7573	B	Normal	Female	Food and beverages	99.69	5	24.9225	523.3725	1/14/2019	12:09	
2741	299-29-0180	B	Member	Female	Home and lifestyle	52.18	7	18.2630	383.5230	3/9/2019	10:54	
2198	305-14-0245	B	Member	Female	Home and lifestyle	94.49	8	37.7960	793.7160	3/3/2019	19:00	E
2038	649-29-6775	B	Normal	Male	Fashion accessories	33.52	1	1.6760	35.1960	2/8/2019	15:31	
670	588-47-8641	A	Member	Male	Fashion accessories	56.04	10	28.0200	588.4200	1/14/2019	19:30	E
1011	699-14-3026	C	Normal	Male	Electronic accessories	85.39	7	29.8865	627.6165	3/25/2019	18:30	E
363	462-67-9126	A	Normal	Male	Home and lifestyle	73.22	6	21.9660	461.2860	1/21/2019	17:44	
2197	895-66-0685	B	Member	Male	Food and beverages	18.08	3	2.7120	56.9520	3/5/2019	19:46	E
1379	382-25-8917	C	Normal	Male	Fashion accessories	42.08	6	12.6240	265.1040	1/29/2019	12:25	

In [14]:
```
df.shape
```

Out[14]:
```
(1006, 14)
```

A- Define :

- After using the melt function , the order of index chanded , so we will get back the normal order

In [15]:
```
df.head()
```

Out[15]:

	Invoice ID	Branch	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payme
0	750-67-8428	A	Normal	Male	Health and beauty	74.69	7	26.1415	NaN	1/5/2019	13:08	Ewal
2	631-41-3108	A	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Cre ca
3	123-19-1176	A	Normal	Male	Health and beauty	58.22	8	NaN	489.0480	1/27/2019	8 - 30 PM	Ewal
4	373-73-7910	A	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewal
6	355-53-5943	A	Normal	Male	Electronic accessories	68.84	6	20.6520	433.6920	2/25/2019	14:36	Ewal

B- Code :

In [16]:
```
df.reset_index(drop=True, inplace=True)
df.head()
```

Out[16]:

	Invoice ID	Branch	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payme
0	750-67-8428	A	Normal	Male	Health and beauty	74.69	7	26.1415	NaN	1/5/2019	13:08	Ewal
1	631-41-3108	A	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Cre ca
2	123-19-1176	A	Normal	Male	Health and beauty	58.22	8	NaN	489.0480	1/27/2019	8 - 30 PM	Ewal
3	373-73-7910	A	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewal
4	355-53-5943	A	Normal	Male	Electronic accessories	68.84	6	20.6520	433.6920	2/25/2019	14:36	Ewal

C- Test :

In [17]:
```
df.head()
```

Out[17]:

	Invoice ID	Branch	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payme
0	750-67-8428	A	Normal	Male	Health and beauty	74.69	7	26.1415	NaN	1/5/2019	13:08	Ewal
1	631-41-3108	A	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Cre ca
2	123-19-1176	A	Normal	Male	Health and beauty	58.22	8	NaN	489.0480	1/27/2019	8 - 30 PM	Ewal
3	373-73-7910	A	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewal
4	355-53-5943	A	Normal	Male	Electronic accessories	68.84	6	20.6520	433.6920	2/25/2019	14:36	Ewal

3.2 Fixing Quality Issues

A- Define :

- Remove Negative in Quantity Column

In [18]:
```
df.Quantity.unique()
```

Out[18]:
```
array([ 7,  8,  6,  2,  5, 10,  3,  9,  1,  4, -1, -8, -7], dtype=int64)
```

B- Code :

In [19]:
```
df['Quantity'] = df['Quantity'].abs()
```

C- Test :

In [20]:
```
df['Quantity'].unique()
```

Out[20]:
```
array([ 7,  8,  6,  2,  5, 10,  3,  9,  1,  4], dtype=int64)
```

A- Define :

- Delete "USD" From Pice to standardize the data type in the column as numeric values.

In [21]:
```
df[df["Unit price"].str.contains("usd",case=False)]
```

Out[21]:

	Invoice ID	Branch	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment
213	308-39-1707	A	Normal	Female	Fashion accessories	12.09 USD	1	NaN	12.6945	1/26/2019	18:19	Credit card
279	237-44-6163	A	Normal	Male	Electronic accessories	10.56 USD	8	NaN	88.7040	1/24/2019	17:43	Cash
309	865-41-9075	A	Normal	Male	Food and beverages	11.53 USD	7	NaN	84.7455	1/28/2019	17:35	Cash
375	871-39-9221	C	Normal	Female	Electronic accessories	12.45 USD	6	NaN	78.4350	2/9/2019	13:11	Cash
631	115-38-7388	C	Member	Female	Fashion accessories	10.18 USD	8	NaN	85.5120	3/30/2019	12:51	Credit card

B- Code :

In [22]:
```
df['Unit price'] = df['Unit price'].str.split().str[0]
df['Unit price'] = df['Unit price'].astype(float)
```

C- Test :

In [23]:
```
df['Unit price'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 1006 entries, 0 to 1005
Series name: Unit price
Non-Null Count  Dtype
--------------  -----
1006 non-null   float64
dtypes: float64(1)
memory usage: 8.0 KB
```

A- Define :

- Handle missing values in Tax 5% and Total by Create New Columns based on (Price and Quantity)

In [24]:
```
null_values = df[df['Tax 5%'].isnull()]

print(null_values[['Tax 5%']])
```

```
     Tax 5%
2       NaN
5       NaN
32      NaN
213     NaN
279     NaN
309     NaN
371     NaN
375     NaN
631     NaN
```

B- Code :

In [25]:
```
df['Tax 5%'] = df['Unit price'] * df['Quantity'] * 0.05
df['Total'] = df['Tax 5%'] + (df['Unit price'] * df['Quantity'] )
```

C- Test :

In [26]:
```
df[['Tax 5%','Total']].isna().sum()
```

Out[26]:
```
Tax 5%    0
Total     0
dtype: int64
```

A- Define :

- In Time col, we Need to Repalce '8 - 30 PM' with '20:30'

In [27]:
```
df[df["Time"] == "8 - 30 PM"]
```

Out[27]:

	Invoice ID	Branch	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	R
2	123-19-1176	A	Normal	Male	Health and beauty	58.22	8	23.288	489.048	1/27/2019	8 - 30 PM	Ewallet	

B- Code :

In [28]:
```
df['Time'] = df['Time'].replace("8 - 30 PM", "20:30")
```

C- Test :

In [29]:
```
df[df["Time"] == "8 - 30 PM"]
```

Out[29]:

	Invoice ID	Branch	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	Rating	city

A- Define :

- Convert Data Type of Date From Object To Date

In [32]:
```
df["Date"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 1006 entries, 0 to 1005
Series name: Date
Non-Null Count  Dtype
--------------  -----
1006 non-null   object
dtypes: object(1)
memory usage: 8.0+ KB
```

B- Code :

In [33]:
```
df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y')
```

C- Test :

In [34]:
```
df['Date'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 1006 entries, 0 to 1005
Series name: Date
Non-Null Count  Dtype
--------------  -----
1006 non-null   datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 8.0 KB
```

A-Define:

- In the Customer type col, Replace Spelling error of "Memberr" to "Member" And "-" to "Normal"

In [35]:
```
df[(df["Customer type"]!= "Member") & (df["Customer type"] != "Normal")]
```

Out[35]:

	Invoice ID	Branch	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment
20	162-48-8011	A	-	Female	Food and beverages	44.59	5	11.1475	234.0975	2019-02-10	15:10	Cash
21	106-35-6779	A	-	Male	Home and lifestyle	44.34	2	4.4340	93.1140	2019-03-27	11:26	Cash
22	635-40-6220	A	-	Male	Health and beauty	89.60	8	35.8400	752.6400	2019-02-07	11:28	Ewalle
23	817-48-8732	A	-	Female	Home and lifestyle	72.35	10	36.1750	759.6750	2019-01-20	15:55	Cash
24	199-75-8169	A	-	Male	Sports and travel	15.81	10	7.9050	166.0050	2019-03-06	12:27	Credi card
25	877-22-3308	A	-	Male	Health and beauty	15.87	10	7.9350	166.6350	2019-03-13	16:40	Cash
26	232-11-3025	A	-	Male	Sports and travel	78.77	10	39.3850	827.0850	2019-01-24	10:04	Cash
27	382-03-4532	A	-	Female	Health and beauty	18.33	1	0.9165	19.2465	2019-02-02	18:50	Cash
351	574-22-5561	C	-	Female	Fashion accessories	82.63	10	41.3150	867.6150	2019-03-19	17:08	Ewalle
352	326-78-5178	C	-	Male	Food and beverages	91.40	7	31.9900	671.7900	2019-02-03	10:19	Cash
353	778-71-5554	C	-	Male	Fashion accessories	15.43	1	0.7715	16.2015	2019-01-25	15:46	Credi card
354	399-46-5918	C	-	Female	Electronic accessories	85.98	8	34.3920	722.2320	2019-02-28	19:01	Cash
355	120-06-4233	C	-	Male	Electronic accessories	30.61	6	9.1830	192.8430	2019-03-12	20:36	Cash
356	285-68-5083	C	-	Female	Sports and travel	24.74	3	3.7110	77.9310	2019-02-15	17:47	Credi card
357	803-83-5989	C	-	Male	Home and lifestyle	55.73	6	16.7190	351.0990	2019-02-24	10:55	Ewalle
358	838-78-4295	C	-	Female	Health and beauty	33.47	2	3.3470	70.2870	2019-02-10	15:43	Ewalle
359	393-65-2792	C	-	Male	Food and beverages	89.48	10	44.7400	939.5400	2019-01-06	12:46	Credi card
360	796-12-2025	C	-	Male	Fashion accessories	62.12	10	31.0600	652.2600	2019-02-11	16:19	Cash
688	370-41-7321	B	-	Male	Health and beauty	56.69	9	25.5105	535.7205	2019-02-27	17:24	Credi card
689	727-46-3608	B	-	Male	Food and beverages	20.01	9	9.0045	189.0945	2019-02-06	15:47	Ewalle
690	669-54-1719	B	-	Male	Electronic accessories	18.93	6	5.6790	119.2590	2019-02-10	12:45	Credi card
691	616-24-2851	B	-	Female	Fashion accessories	17.87	4	3.5740	75.0540	2019-03-22	14:42	Ewalle
692	242-55-6721	B	-	Male	Home and lifestyle	16.16	2	1.6160	33.9360	2019-03-07	11:49	Ewalle
693	347-34-2234	B	-	Female	Sports and travel	55.07	9	24.7815	520.4115	2019-02-03	13:40	Ewalle
694	853-23-2453	B	-	Male	Health and beauty	75.74	4	15.1480	318.1080	2019-02-14	14:35	Cash

	Invoice ID	Branch	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Paymen
695	109-28-2512	B	-	Female	Fashion accessories	97.61	6	29.2830	614.9430	2019-01-07	15:01	Ewalle
696	510-95-6347	B	-	Female	Food and beverages	48.52	3	7.2780	152.8380	2019-03-05	18:17	Ewalle
751	346-84-3103	B	Memberr	Female	Electronic accessories	13.22	5	3.3050	69.4050	2019-03-02	19:26	Cash

B- Code :

In [36]:
```
df['Customer type'] = df['Customer type'].replace({'Memberr': 'Member', '-': 'Normal'})
```

C- Test :

In [37]:
```
df['Customer type'].unique()
```

Out[37]:
```
array(['Normal', 'Member'], dtype=object)
```

A-Define:

- Replace "97" in Rating by 9.7

In [38]:
```
df["Rating"].value_counts().sort_index(ascending=False)
```

Out[38]:
```
Rating
97.0     1
10.0     5
9.9     16
9.8     19
9.7     13
        ..
4.4     17
4.3     19
4.2     23
4.1     17
4.0     11
Name: count, Length: 62, dtype: int64
```

B- Code :

In [39]:
```
df['Rating']=df['Rating'].replace(97,9.7)
```

C- Test :

In [40]:
```
df["Rating"].value_counts().sort_index(ascending=False)
```

Out[40]:
```
Rating
10.0     5
9.9     16
9.8     19
9.7     14
9.6     17
        ..
4.4     17
4.3     19
4.2     23
4.1     17
4.0     11
Name: count, Length: 61, dtype: int64
```

A-Define:

- Remove Duplicates

In [41]:
```
df.duplicated().sum()
```

Out[41]:
```
6
```

B- Code :

In [42]:
```
df = df.drop_duplicates()
```

C- Test :

In [43]:
```
df.duplicated().sum()
```

Out[43]:
```
0
```

A-Define:

- Classify Rating

In [44]:
```
df.loc[:, ["Rating"]]
```

Out[44]:

	Rating
0	9.1
1	7.4
2	8.4
3	5.3
4	5.8
...	...
999	6.2
1000	8.4
1001	6.0
1002	6.6
1003	4.4

1000 rows × 1 columns

B- Code :

In [45]:
```
# Define a function to classify ratings
def classify_rating(rating):
    if rating >= 9:
        return 'Excellent'
    elif rating >= 8:
        return 'Very Good'
    elif rating >= 6:
        return 'Good'
    elif rating >= 4:
        return 'Fair'
    else:
        return 'Poor'

# Apply the function to the Rating column
df['Rating Category'] = df['Rating'].apply(classify_rating)
```

C- Test :

In [46]:
```
df['Rating Category'].unique()
```

Out[46]:
```
array(['Excellent', 'Good', 'Very Good', 'Fair'], dtype=object)
```

In [47]:
```
df.loc[:, ["Rating", "Rating Category"]]
```

Out[47]:

	Rating	Rating Category
0	9.1	Excellent
1	7.4	Good
2	8.4	Very Good
3	5.3	Fair
4	5.8	Fair
...	...	...
999	6.2	Good
1000	8.4	Very Good
1001	6.0	Good
1002	6.6	Good
1003	4.4	Fair

1000 rows × 2 columns

Check Final Resul

In [48]:
```
df.head()
```

Out[48]:

	Invoice ID	Branch	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment
0	750-67-8428	A	Normal	Male	Health and beauty	74.69	7	26.1415	548.9715	2019-01-05	13:08	Ewallet
1	631-41-3108	A	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	2019-03-03	13:23	Credit card
2	123-19-1176	A	Normal	Male	Health and beauty	58.22	8	23.2880	489.0480	2019-01-27	20:30	Ewallet
3	373-73-7910	A	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2019-02-08	10:37	Ewallet
4	355-53-5943	A	Normal	Male	Electronic accessories	68.84	6	20.6520	433.6920	2019-02-25	14:36	Ewallet

In [49]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1000 entries, 0 to 1003
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Invoice ID       1000 non-null   object
 1   Branch           1000 non-null   object
 2   Customer type    1000 non-null   object
 3   Gender           1000 non-null   object
 4   Product line     1000 non-null   object
 5   Unit price       1000 non-null   float64
 6   Quantity         1000 non-null   int64
 7   Tax 5%           1000 non-null   float64
 8   Total            1000 non-null   float64
 9   Date             1000 non-null   datetime64[ns]
 10  Time             1000 non-null   object
 11  Payment          1000 non-null   object
 12  Rating           1000 non-null   float64
 13  city             1000 non-null   object
 14  Rating Category  1000 non-null   object
dtypes: datetime64[ns](1), float64(4), int64(1), object(9)
memory usage: 125.0+ KB
```

In [50]:
```
df.isna().sum()
```

Out[50]:
```
Invoice ID         0
Branch             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
Rating             0
city               0
Rating Category    0
dtype: int64
```

In [51]:
```
df.duplicated().sum()
```

Out[51]:
```
0
```

In [52]:
```
df.shape
```

Out[52]:
```
(1000, 15)
```

Save Data to CSV file to visualize it Using Power BI

In [53]:
```
df.to_csv(r'Sales.csv', index=False)
```