|  |  |  |  |
| --- | --- | --- | --- |
| **import** pandas **as** pd **import** seaborn **as** sns **import** matplotlib.pyplot **as** plt |  |  |  |
|  |  |  |  |
| df**=**pd**.**read\_csv('supermarket\_sales - Sheet1.csv') |  |  |  |
|  |  |  |  |
| df |  |  |  |
| **Invoice Customer**  **Branch City Gender**  **ID type** | **Product line** | **Unit price** | **Quantity Tax 5** |

In [1]:

In [2]:

In [3]:

Out[3]:

750-

Health and

**0** 67- A Yangon Member Female 74.69 7 26.14

beauty 8428

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 226-  313081 | C | Naypyitaw | Normal | Female | Electronic accessories | 15.28 | 5 | 3.820 |
| **2** | 631-  413108 | A | Yangon | Normal | Male | Home and lifestyle | 46.33 | 7 | 16.215 |
| **3** | 123-  191176 | A | Yangon | Member | Male | Health and beauty | 58.22 | 8 | 23.288 |
| **4** | 373-  737910 | A | Yangon | Normal | Male | Sports and travel | 86.31 | 7 | 30.208 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |  |
| **995** | 233-  675758 | C | Naypyitaw | Normal | Male | Health and beauty | 40.35 | 1 | 2.017 |
| **996** | 303-  962227 | B | Mandalay | Normal | Female | Home and lifestyle | 97.38 | 10 | 48.690 |
| **997** | 727-  021313 | A | Yangon | Member | Male | Food and beverages | 31.84 | 1 | 1.592 |
| **998** | 347-  562442 | A | Yangon | Normal | Male | Home and lifestyle | 65.82 | 1 | 3.29 |
| **999** | 849-  093807 | A | Yangon | Member | Female | Fashion accessories | 88.34 | 7 | 30.919 |

1000 rows × 17 columns

|  |
| --- |
| df**.**columns |

In [4]:

Out[4]: 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')

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| df**.**describe() |  |  |  |  |  |
| **Unit price** | **Quantity** | **Tax 5%** | **Total** | **cogs** | **gross margin percentage** |

In [5]:

Out[5]:

**count** 1000.000000 1000.000000 1000.000000 1000.000000 1000.00000 1000.000000 10

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **mean** | 55.672130 | 5.510000 | 15.379369 | 322.966749 | 307.58738 | 4.761905 |
| **std** | 26.494628 | 2.923431 | 11.708825 | 245.885335 | 234.17651 | 0.000000 |
| **min** | 10.080000 | 1.000000 | 0.508500 | 10.678500 | 10.17000 | 4.761905 |
| **25%** | 32.875000 | 3.000000 | 5.924875 | 124.422375 | 118.49750 | 4.761905 |
| **50%** | 55.230000 | 5.000000 | 12.088000 | 253.848000 | 241.76000 | 4.761905 |
| **75%** | 77.935000 | 8.000000 | 22.445250 | 471.350250 | 448.90500 | 4.761905 |
| **max** | 99.960000 | 10.000000 | 49.650000 | 1042.650000 | 993.00000 | 4.761905 |

In [6]: df**.**dtypes

Out[6]: 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 [7]: *# chcvk for nul values* df**.**isnull()**.**sum()

Out[7]: 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

|  |
| --- |
| *#check a duplicate value*  duplicated\_rows**=**df[df**.**duplicated()] print("duplicate rows based on all columns:") print(duplicated\_rows) |

In [8]:

duplicate rows based on all columns:

Empty DataFrame

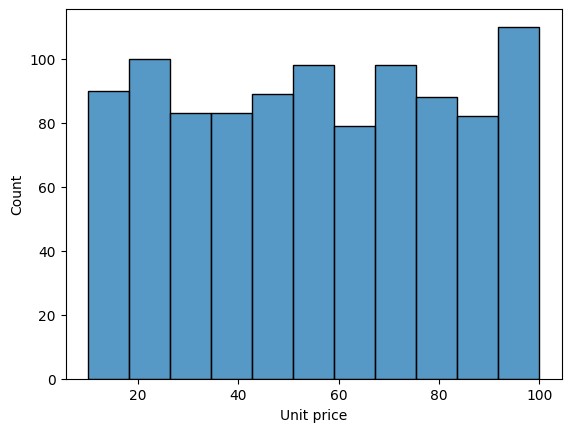
|  |
| --- |
| total\_sum\_duplicated\_valuse**=**df**.**duplicated()**.**sum() print(total\_sum\_duplicated\_valuse) |

Columns: [Invoice ID, Branch, City, Customer type, Gender, Product line, Unit pri ce, Quantity, Tax 5%, Total, Date, Time, Payment, cogs, gross margin percentage, gross income, Rating] Index: [] In [9]:

|  |
| --- |
| *#univariat analyst*  sns**.**histplot(x**=**'Unit price',data**=**df) |

0 In [10]:

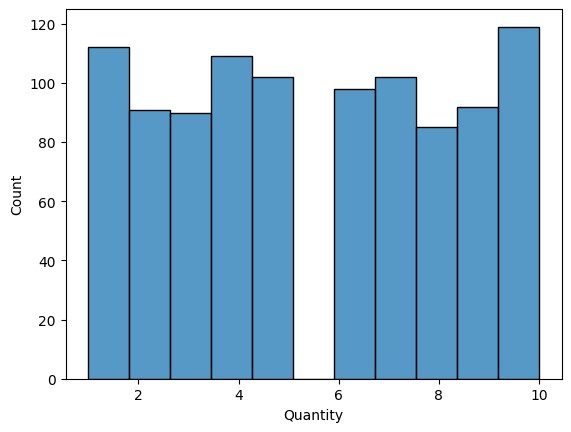
Out[10]: <Axes: xlabel='Unit price', ylabel='Count'>



|  |
| --- |
| *#Quantity*  sns**.**histplot(x**=**'Quantity',data**=**df) |

In [11]:

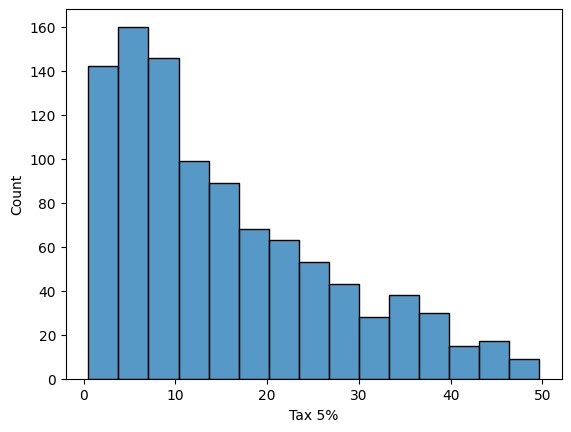
Out[11]: <Axes: xlabel='Quantity', ylabel='Count'>



|  |
| --- |
| sns**.**histplot(x**=**'Tax 5%',data**=**df) |

In [12]:

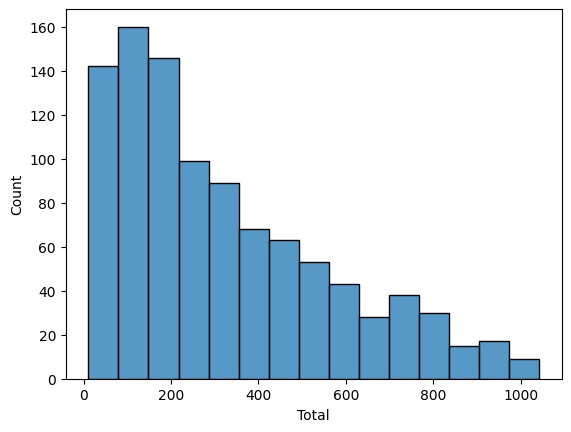
Out[12]: <Axes: xlabel='Tax 5%', ylabel='Count'>



|  |
| --- |
| sns**.**histplot(x**=**'Total',data**=**df) |

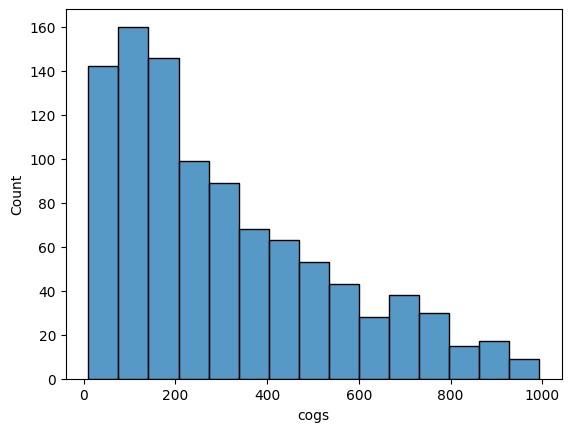
In [13]:

Out[13]: <Axes: xlabel='Total', ylabel='Count'>



In [14]: sns**.**histplot(x**=**'cogs',data**=**df)

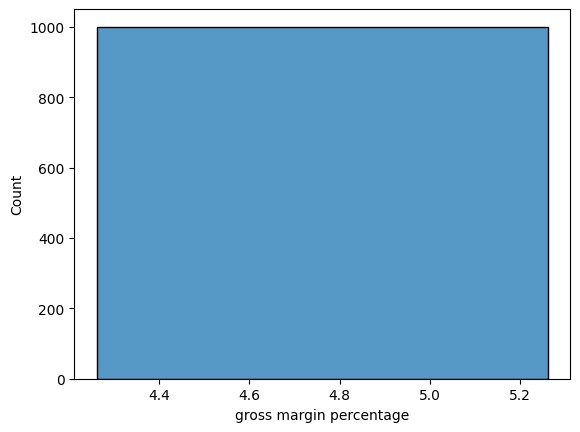
Out[14]: <Axes: xlabel='cogs', ylabel='Count'>



|  |
| --- |
| sns**.**histplot(x**=**'gross margin percentage',data**=**df) |

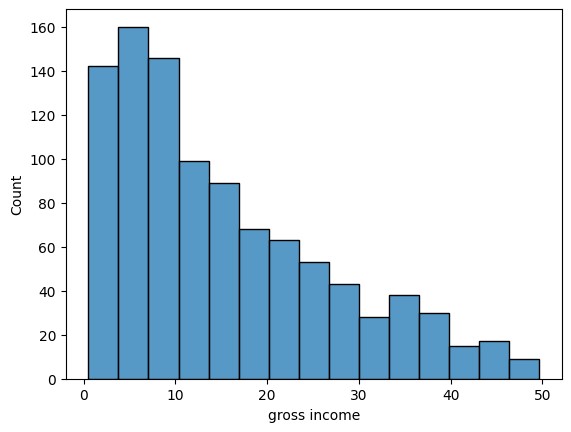
In [15]:

Out[15]: <Axes: xlabel='gross margin percentage', ylabel='Count'>



In [16]: sns**.**histplot(x**=**'gross income',data**=**df)

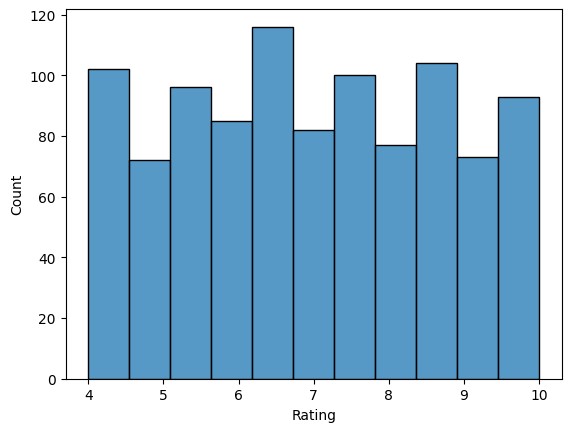
Out[16]: <Axes: xlabel='gross income', ylabel='Count'>



|  |
| --- |
| sns**.**histplot(x**=**'Rating',data**=**df) |

In [17]:

Out[17]: <Axes: xlabel='Rating', ylabel='Count'>



In [18]: df**.**columns

Out[18]: Index(['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender',

'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Date',

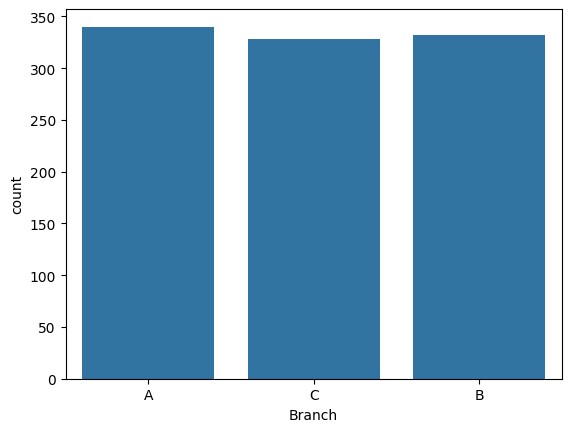
'Time', 'Payment', 'cogs', 'gross margin percentage', 'gross income',

'Rating'], dtype='object')

In [19]: *#count each category #branch, city,*

sns**.**countplot(x**=**'Branch',data**=**df) df['Branch']**.**value\_counts()

|  |  |
| --- | --- |
| Out[19]: | Branch   1. 340 2. 332 3. 328   Name: count, dtype: int64 |



|  |
| --- |
| sns**.**countplot(x**=**'City',data**=**df) df['City']**.**value\_counts() |

In [20]:

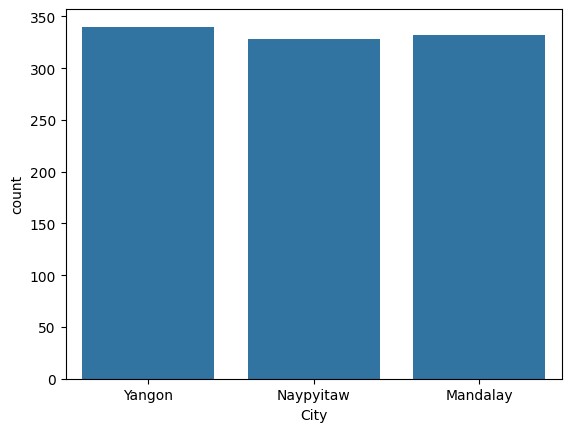
Out[20]: City

Yangon 340

Mandalay 332

Naypyitaw 328

Name: count, dtype: int64



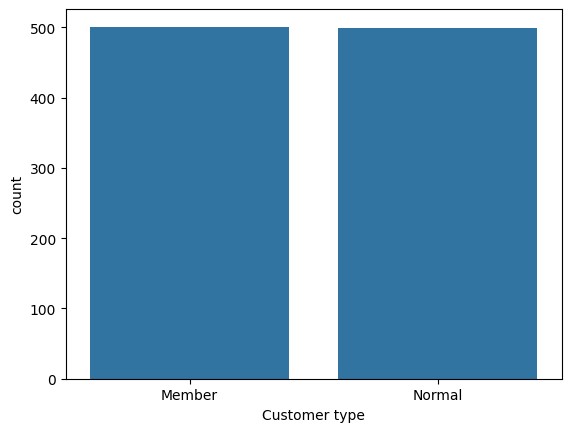
|  |
| --- |
| sns**.**countplot(x**=**'Customer type',data**=**df) df['Customer type']**.**value\_counts() |

In [21]:

Out[21]: Customer type Member 501

Normal 499

Name: count, dtype: int64



|  |
| --- |
| sns**.**countplot(x**=**'Gender',data**=**df) df['Gender']**.**value\_counts() |

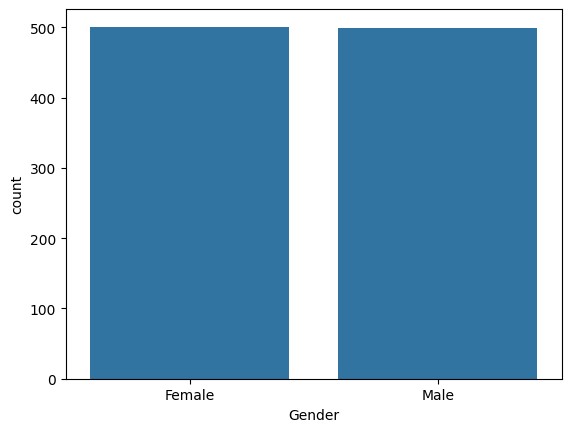
In [22]:

Out[22]: Gender

Female 501

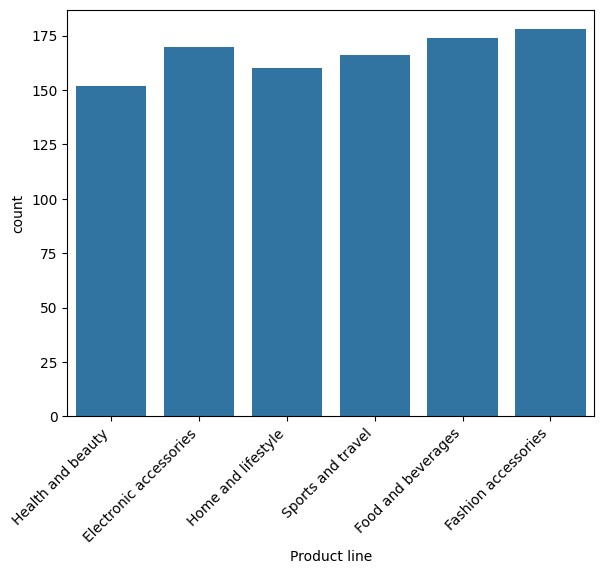
Male 499

Name: count, dtype: int64



|  |
| --- |
| sns**.**countplot(x**=**'Product line',data**=**df) plt**.**tight\_layout() plt**.**xticks(rotation**=**45,ha**=**'right') df['Product line']**.**value\_counts() plt**.**show() |

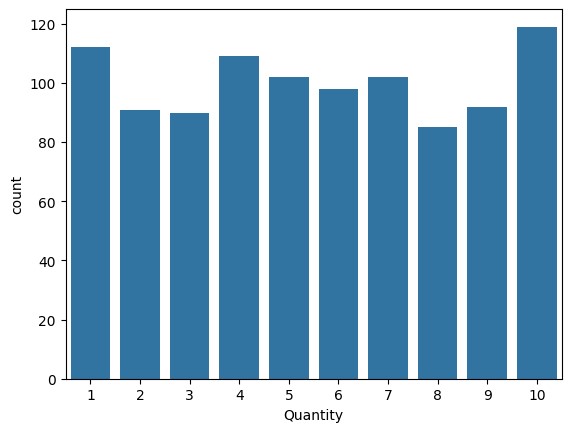
In [23]:



|  |
| --- |
| sns**.**countplot(x**=**'Quantity',data**=**df) |

In [24]:

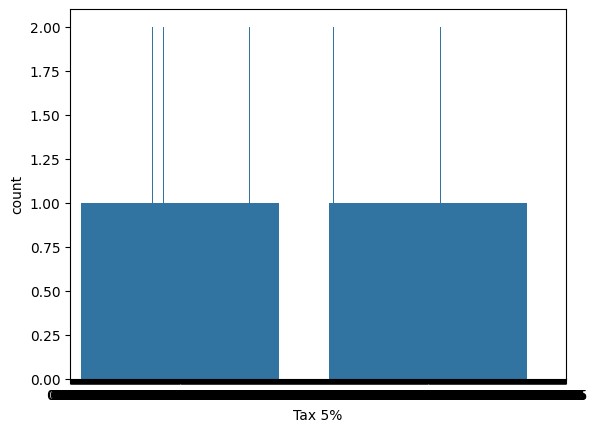
Out[24]: <Axes: xlabel='Quantity', ylabel='count'>



|  |
| --- |
| sns**.**countplot(x**=**'Tax 5%',data**=**df) |

In [25]:

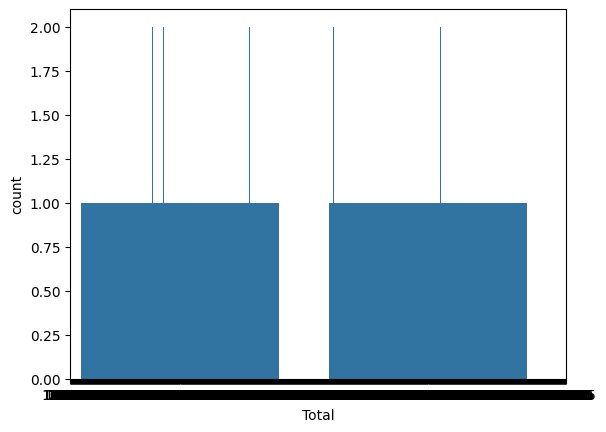
Out[25]: <Axes: xlabel='Tax 5%', ylabel='count'>



|  |
| --- |
| sns**.**countplot(x**=**'Total',data**=**df) |

In [26]:

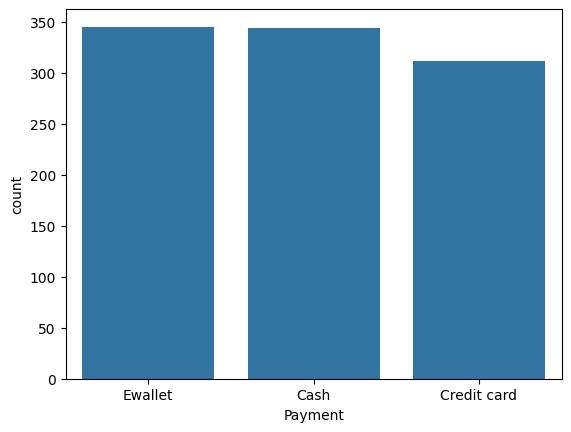
Out[26]: <Axes: xlabel='Total', ylabel='count'>



|  |
| --- |
| sns**.**countplot(x**=**'Payment', data**=**df) |

In [27]:

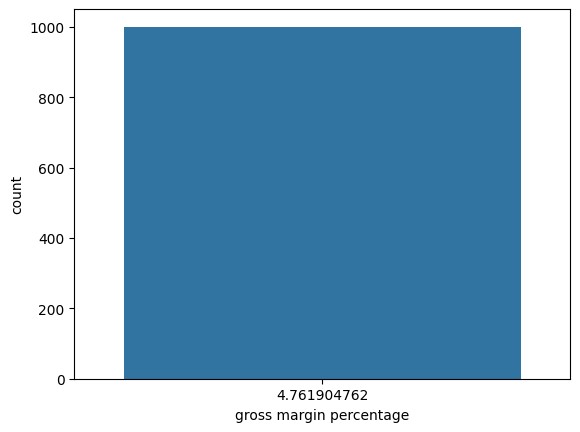
Out[27]: <Axes: xlabel='Payment', ylabel='count'>



|  |
| --- |
| sns**.**countplot(x**=**'gross margin percentage',data**=**df) |

In [28]:

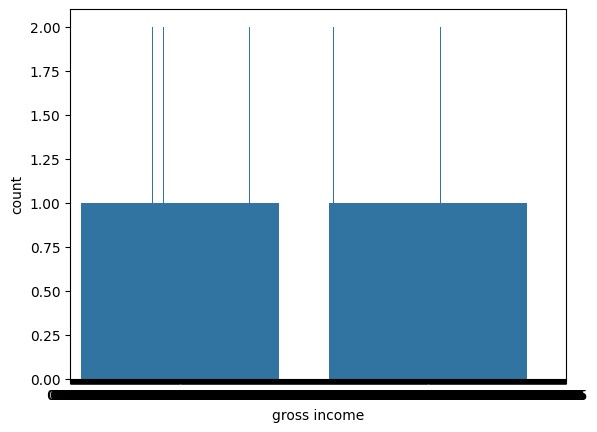
Out[28]: <Axes: xlabel='gross margin percentage', ylabel='count'>



|  |
| --- |
| sns**.**countplot(x**=**'gross income', data**=**df) |

In [29]:

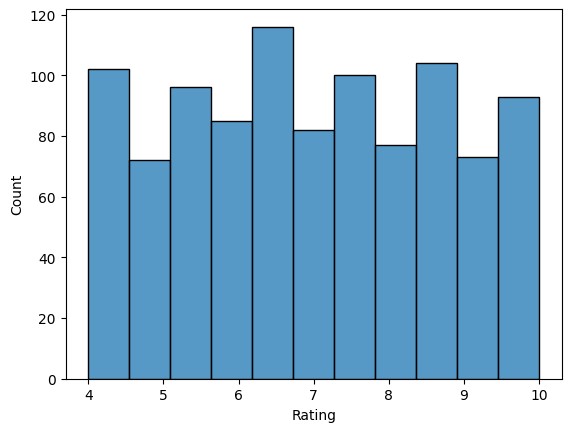
Out[29]: <Axes: xlabel='gross income', ylabel='count'>



|  |
| --- |
| sns**.**histplot(x**=**'Rating', data**=**df) |

In [30]:

Out[30]: <Axes: xlabel='Rating', ylabel='Count'>



|  |
| --- |
| *#counvarting data column into data* df['Date']**=**pd**.**to\_datetime(df['Date']) |

In [31]:

|  |
| --- |
| df['Date']**.**isnull()**.**sum() |

In [32]:

Out[32]: np.int64(0)

|  |
| --- |
| df['Date']**.**fillna(method**=**'ffill',inplace**=True**) |

In [33]:

C:\Users\hp\AppData\Local\Temp\ipykernel\_1208\2146825199.py:1: FutureWarning: A v alue is trying to be set on a copy of a DataFrame or Series through chained assig nment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work becau se the intermediate object on which we are setting values always behaves as a cop y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe rform the operation inplace on the original object.

df['Date'].fillna(method='ffill',inplace=True)

C:\Users\hp\AppData\Local\Temp\ipykernel\_1208\2146825199.py:1: FutureWarning: Ser ies.fillna with 'method' is deprecated and will raise in a future version. Use ob

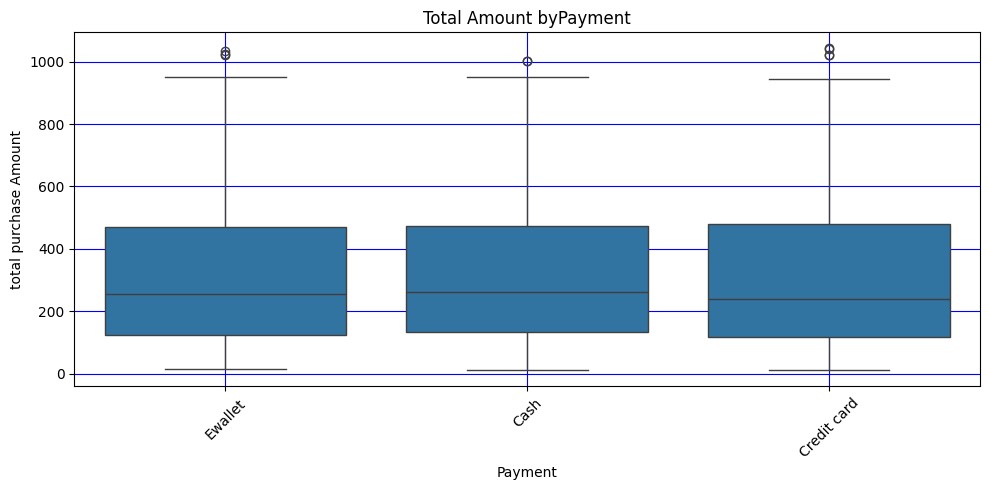
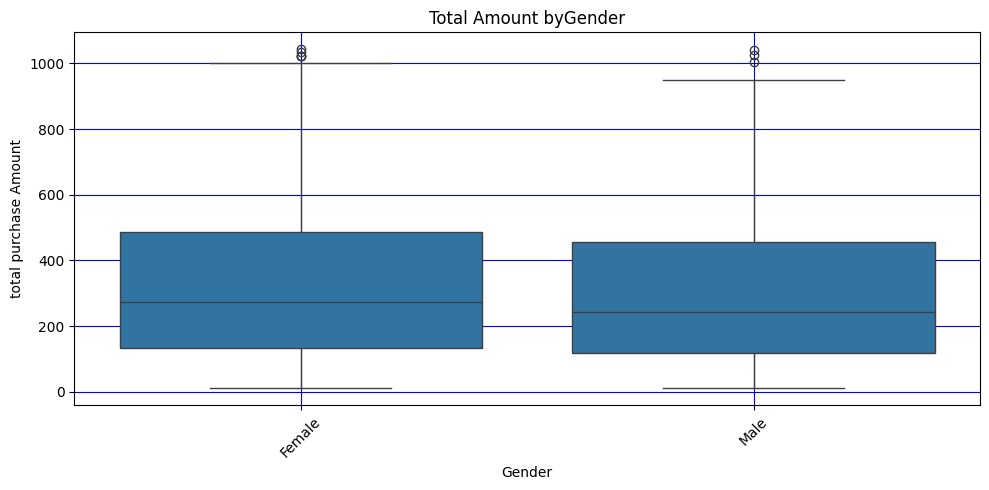
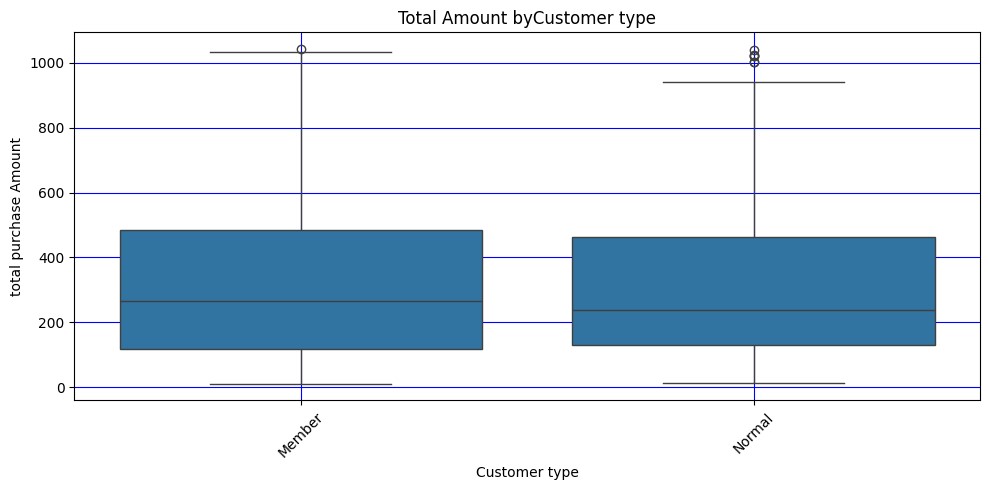
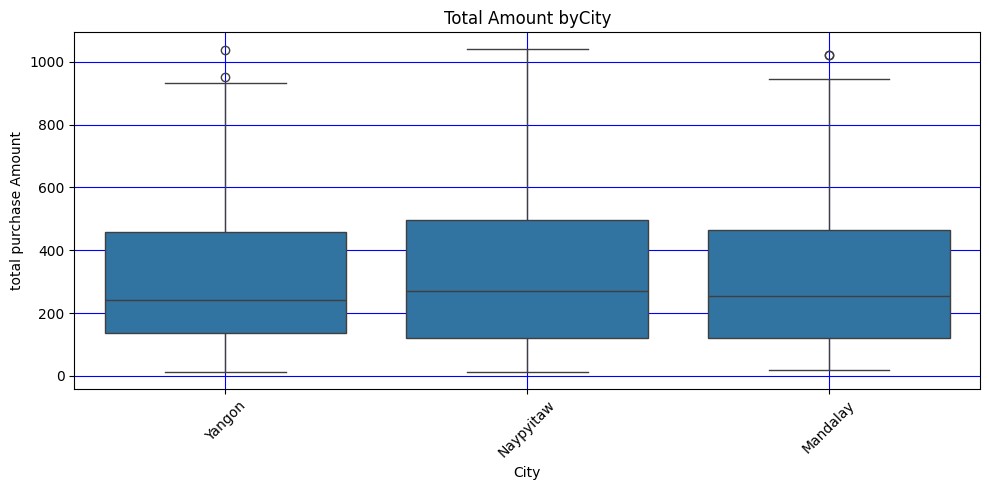
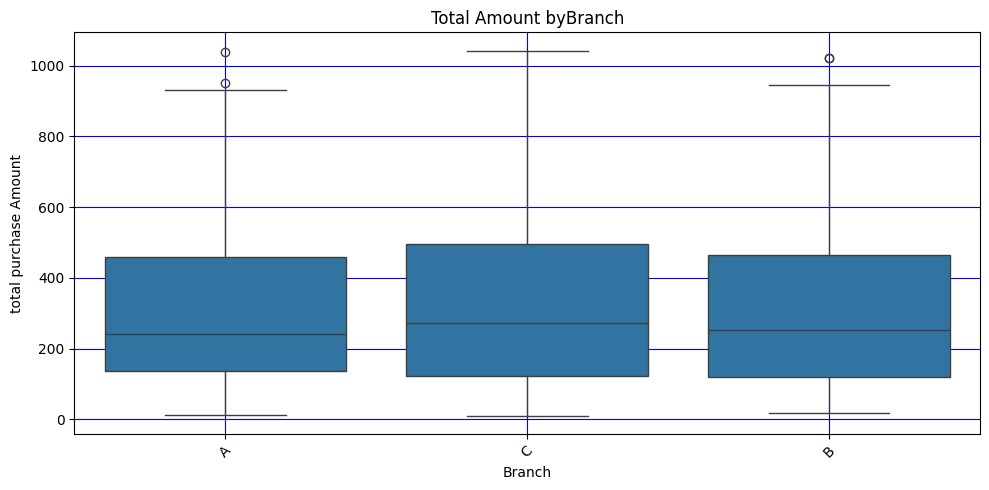
j.ffill() or obj.bfill() instead. df['Date'].fillna(method='ffill',inplace=True)

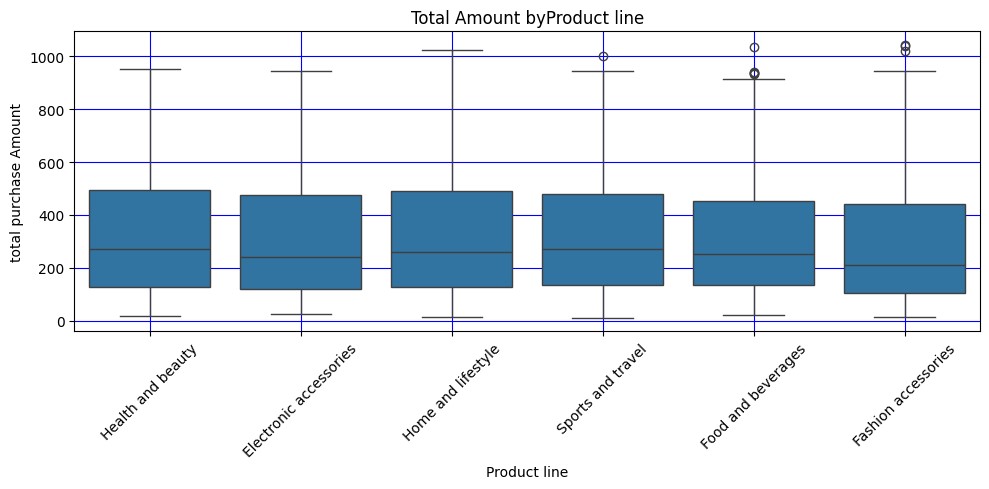
|  |
| --- |
| *#list of categorical columns for bivariate analysis*  categorical\_cols**=**['Branch', 'City', 'Customer type', 'Gender','Payment',  'Product line'] print(categorical\_cols)  *#criating box plot with loop for bivariate analysis*  *#total vs categorical* |

In [34]:

*# plot boxplots for 'total' vs categorical variables* **for** col **in** categorical\_cols: plt**.**figure(figsize**=**(10,5)) sns**.**boxplot(data**=**df,x**=**col,y**=**'Total') plt**.**title(f'Total Amount by{col}') plt**.**xlabel(col) plt**.**ylabel('total purchase Amount') plt**.**xticks(rotation**=**45) plt**.**tight\_layout() plt**.**grid(c**=**'blue') plt**.**show()

['Branch', 'City', 'Customer type', 'Gender', 'Payment', 'Product line']

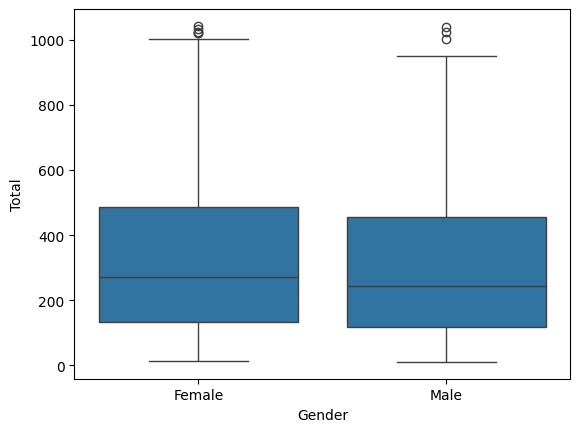




|  |
| --- |
| *#box plot total,pl g,m,p*  sns**.**boxplot(x**=**'Gender',y**=**'Total',data**=**df) |

In [35]:

Out[35]: <Axes: xlabel='Gender', ylabel='Total'>



|  |
| --- |
| *#Identify numarical columns*  df**=**pd**.**read\_csv('supermarket\_sales - Sheet1.csv') df |

In [36]:

Out[36]:

# Invoice Customer Product Unit ID type line price

**Branch City Gender Quantity Tax 5**

750-

Health and

**0** 67- A Yangon Member Female 74.69 7 26.14

beauty 8428

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 226-  313081 | C | Naypyitaw | Normal | Female | Electronic accessories | 15.28 |  | 5 | 3.820 |
| **2** | 631-  413108 | A | Yangon | Normal | Male | Home and lifestyle | 46.33 |  | 7 | 16.215 |
| **3** | 123-  191176 | A | Yangon | Member | Male | Health and beauty | 58.22 |  | 8 | 23.288 |
| **4** | 373-  737910 | A | Yangon | Normal | Male | Sports and travel | 86.31 |  | 7 | 30.208 |
| **...** | ... | ... | ... | ... | ... | ... | ... |  | ... |  |
| **995** | 233-  675758 | C | Naypyitaw | Normal | Male | Health and beauty | 40.35 |  | 1 | 2.017 |
| **996** | 303-  962227 | B | Mandalay | Normal | Female | Home and lifestyle | 97.38 |  | 10 | 48.690 |
| **997** | 727-  021313 | A | Yangon | Member | Male | Food and beverages | 31.84 |  | 1 | 1.592 |
| **998** | 347-  562442 | A | Yangon | Normal | Male | Home and lifestyle | 65.82 |  | 1 | 3.29 |
| **999** | 849-  093807 | A | Yangon | Member | Female | Fashion accessories | 88.34 |  | 7 | 30.919 |

1000 rows × 17 columns

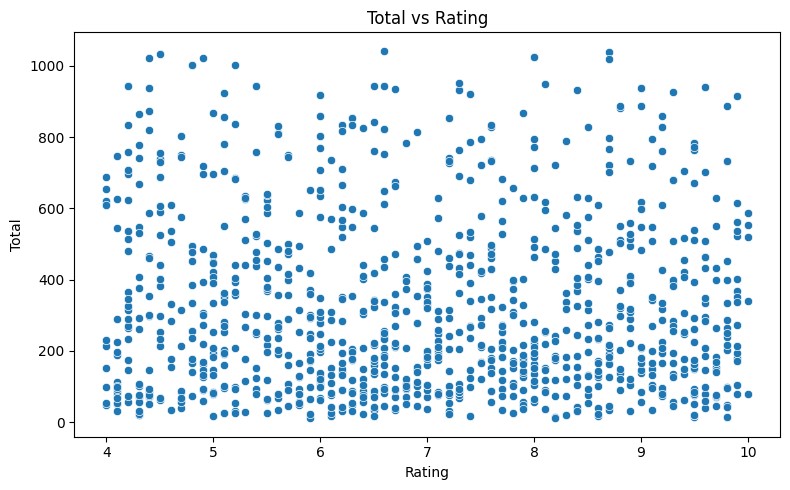
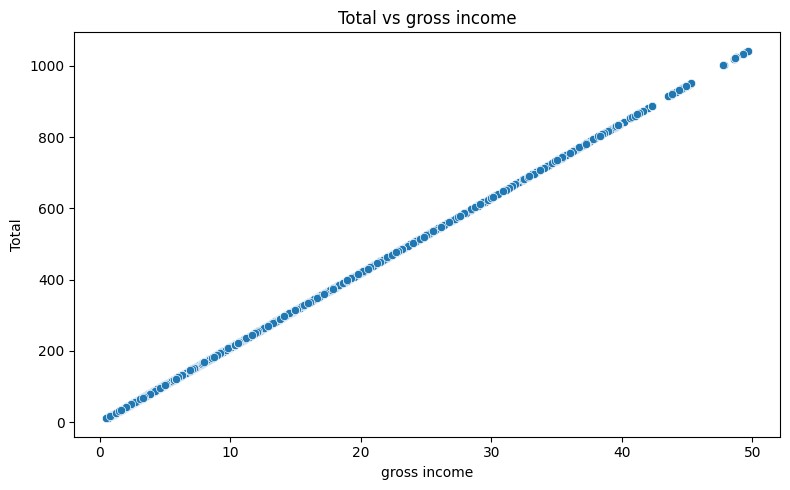
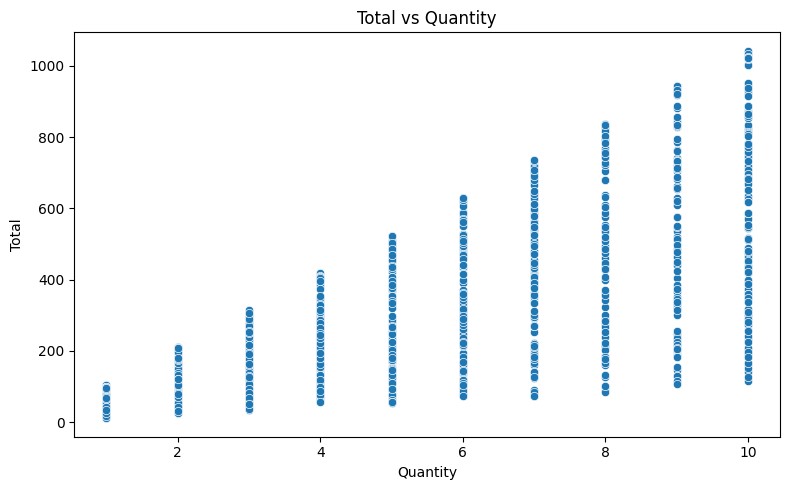
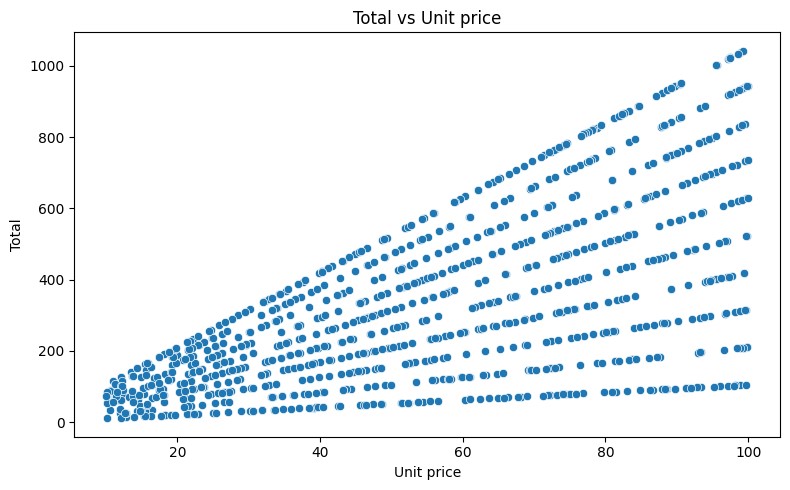
|  |
| --- |
| numerical\_cols**=**df**.**select\_dtypes(include**=**['float64','int64'])**.**columns print(numerical\_cols) scatter\_pairs**=**[  ('Unit price','Total'),  ('Quantity','Total'),  ('gross income','Total'),  ('Rating','Total')  ]  print(scatter\_pairs) **for** x,y **in** scatter\_pairs:  plt**.**figure(figsize**=**(8,5)) sns**.**scatterplot(data**=**df,x**=**x,y**=**y) |

In [41]:

plt**.**title(f'{y} vs {x}') plt**.**tight\_layout()

Index(['Unit price', 'Quantity', 'Tax 5%', 'Total', 'cogs', 'gross margin percentage', 'gross income', 'Rating'], dtype='object')

[('Unit price', 'Total'), ('Quantity', 'Total'), ('gross income', 'Total'), ('Rat ing', 'Total')]



|  |
| --- |
| *#code for time-based Trend Analysis*  *#convert 'Data' and 'Time' To datatime objects* **import** pandas **as** pd df['Date']**=**pd**.**to\_datetime(df['Date'],errors**=**'coerce') df['Time']**=**pd**.**to\_datetime(df['Time'],format**=**'%H:%M')**.**dt**.**time |

|  |
| --- |
| df**.**dtypes |

In [42]: In [43]:

Out[43]: Invoice ID object Branch object

City object

Customer type object

Gender object

Product line object

Unit price float64

Quantity int64

Tax 5% float64

Total float64

Date datetime64[ns]

Time object Payment object cogs float64 gross margin percentage float64 gross income float64 Rating float64 dtype: object

In [44]: *#df['Date'].head(50)*

*#create additional Time-based feature* df['Day']**=**df['Date']**.**dt**.**date df['Month']**=**df['Date']**.**dt**.**to\_period('M') df['Weekday']**=**df['Date']**.**dt**.**day\_name()

In [48]: df

Out[48]:

# Invoice Customer Product Unit ID type line price

**Branch City Gender Quantity Tax 5**

750-

Health and

**0** 67- A Yangon Member Female 74.69 7 26.14

beauty 8428

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 226-  313081 | C | Naypyitaw | Normal | Female | Electronic accessories | 15.28 | 5 | 3.820 |
| **2** | 631-  413108 | A | Yangon | Normal | Male | Home and lifestyle | 46.33 | 7 | 16.215 |
| **3** | 123-  191176 | A | Yangon | Member | Male | Health and beauty | 58.22 | 8 | 23.288 |
| **4** | 373-  737910 | A | Yangon | Normal | Male | Sports and travel | 86.31 | 7 | 30.208 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |  |
| **995** | 233-  675758 | C | Naypyitaw | Normal | Male | Health and beauty | 40.35 | 1 | 2.017 |
| **996** | 303-  962227 | B | Mandalay | Normal | Female | Home and lifestyle | 97.38 | 10 | 48.690 |
| **997** | 727-  021313 | A | Yangon | Member | Male | Food and beverages | 31.84 | 1 | 1.592 |
| **998** | 347-  562442 | A | Yangon | Normal | Male | Home and lifestyle | 65.82 | 1 | 3.29 |
| **999** | 849-  093807 | A | Yangon | Member | Female | Fashion accessories | 88.34 | 7 | 30.919 |

1000 rows × 21 columns

|  |
| --- |
| *#covert 'Time' to hour only*  df['Hour']**=**pd**.**to\_datetime(df['Time'],format**=**'%H,:%M,%S',errors**=**'coerce')**.**apply( |

In [51]: **l**

|  |
| --- |
| df |

In [52]:

Out[52]:

# Invoice Customer Product Unit ID type line price

**Branch City Gender Quantity Tax 5**

750-

Health and

**0** 67- A Yangon Member Female 74.69 7 26.14

beauty 8428

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 226-  313081 | C | Naypyitaw | Normal | Female | Electronic accessories | 15.28 | 5 | 3.820 |
| **2** | 631-  413108 | A | Yangon | Normal | Male | Home and lifestyle | 46.33 | 7 | 16.215 |
| **3** | 123-  191176 | A | Yangon | Member | Male | Health and beauty | 58.22 | 8 | 23.288 |
| **4** | 373-  737910 | A | Yangon | Normal | Male | Sports and travel | 86.31 | 7 | 30.208 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |  |
| **995** | 233-  675758 | C | Naypyitaw | Normal | Male | Health and beauty | 40.35 | 1 | 2.017 |
| **996** | 303-  962227 | B | Mandalay | Normal | Female | Home and lifestyle | 97.38 | 10 | 48.690 |
| **997** | 727-  021313 | A | Yangon | Member | Male | Food and beverages | 31.84 | 1 | 1.592 |
| **998** | 347-  562442 | A | Yangon | Normal | Male | Home and lifestyle | 65.82 | 1 | 3.29 |
| **999** | 849-  093807 | A | Yangon | Member | Female | Fashion accessories | 88.34 | 7 | 30.919 |

1000 rows × 21 columns

|  |
| --- |
| *#date and month wise total sales*  Total\_sum**=**df**.**groupby('Day')['Total']**.**sum() print(Total\_sum)  *#month\_wise*  Total\_monthwise**=**df**.**groupby('Month')['Total']**.**sum() print(Total\_monthwise)  *#day wise Total*  Day\_wise\_salse**=**df**.**groupby('Weekday')['Total']**.**sum() print(Day\_wise\_salse) |

In [53]:

Day

2019-01-01 4745.1810

2019-01-02 1945.5030

2019-01-03 2078.1285

2019-01-04 1623.6885

2019-01-05 3536.6835 ...

2019-03-26 1962.5130

2019-03-27 2902.8195

2019-03-28 2229.4020

2019-03-29 4023.2430

2019-03-30 4487.0595 Name: Total, Length: 89, dtype: float64

Month

2019-01 116291.868

2019-02 97219.374

2019-03 109455.507 Freq: M, Name: Total, dtype: float64

Weekday

Friday 43926.3405

Monday 37899.0780

Saturday 56120.8095

Sunday 44457.8925

Thursday 45349.2480

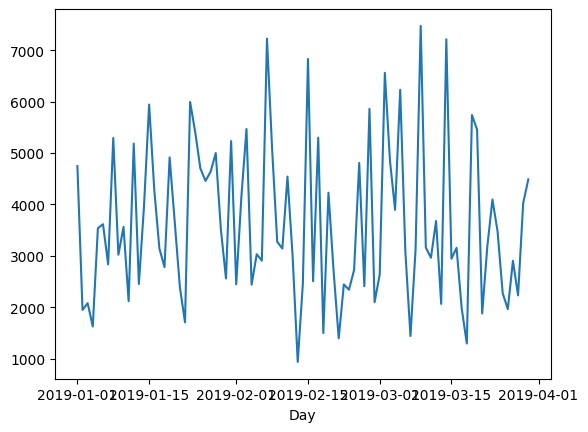
Tuesday 51482.2455

Wednesday 43731.1350

|  |
| --- |
| Total\_sum**.**plot() |

Name: Total, dtype: float64 In [54]:

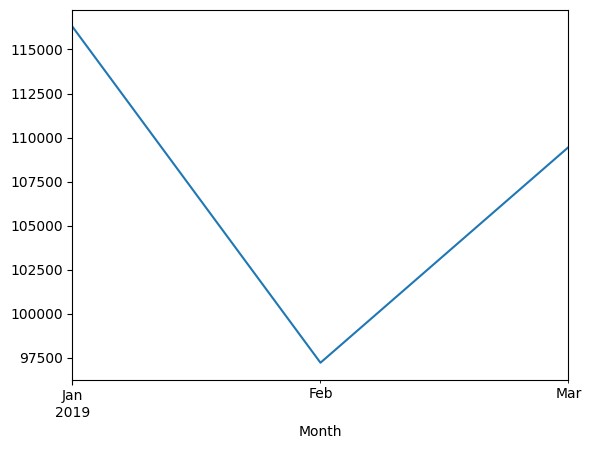
Out[54]: <Axes: xlabel='Day'>



|  |
| --- |
| Total\_monthwise**.**plot() |

In [55]:

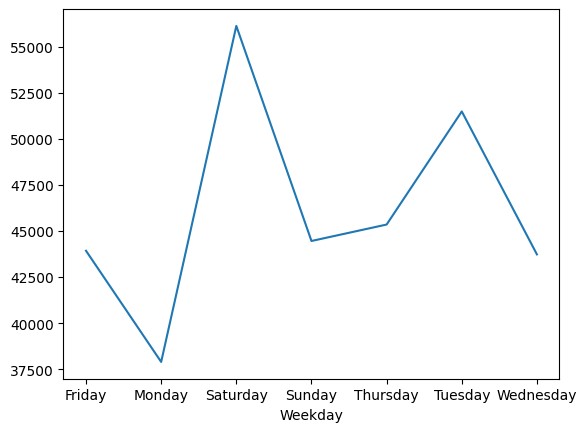
Out[55]: <Axes: xlabel='Month'>



|  |
| --- |
| Day\_wise\_salse**.**plot() |

In [56]:

Out[56]: <Axes: xlabel='Weekday'>

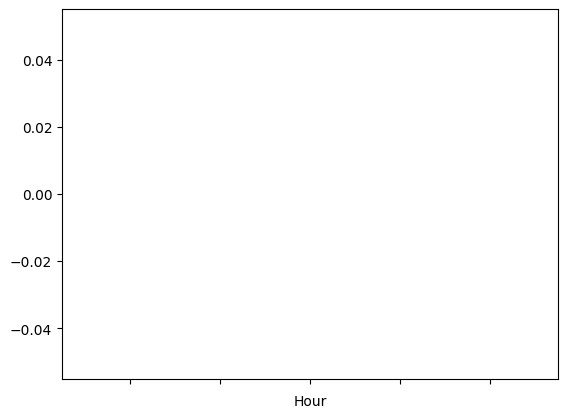


|  |
| --- |
| hourly\_sales**=**df**.**groupby('Hour')['Total']**.**sum() print(hourly\_sales) hourly\_sales**.**plot() |

In [57]:

Series([], Name: Total, dtype: float64)

Out[57]: <Axes: xlabel='Hour'>



|  |
| --- |
|  |

In [ ]: