In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
df=pd.read_csv('supermarket_sales - Sheet1.csv')
```

In [3]:
```
df
```

Out[3]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.14
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.82
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.21
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.28
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.20
...	...	...	...	...	...	...	...	...	
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.01
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.69
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.59
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.29
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.91

1000 rows × 17 columns

In [4]:
```
df.columns
```

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')
```

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

Out[5]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	
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
```

In [8]:
```
#check a duplicate value
duplicated_rows=df[df.duplicated()]
print("duplicate rows based on all columns:")
print(duplicated_rows)
```

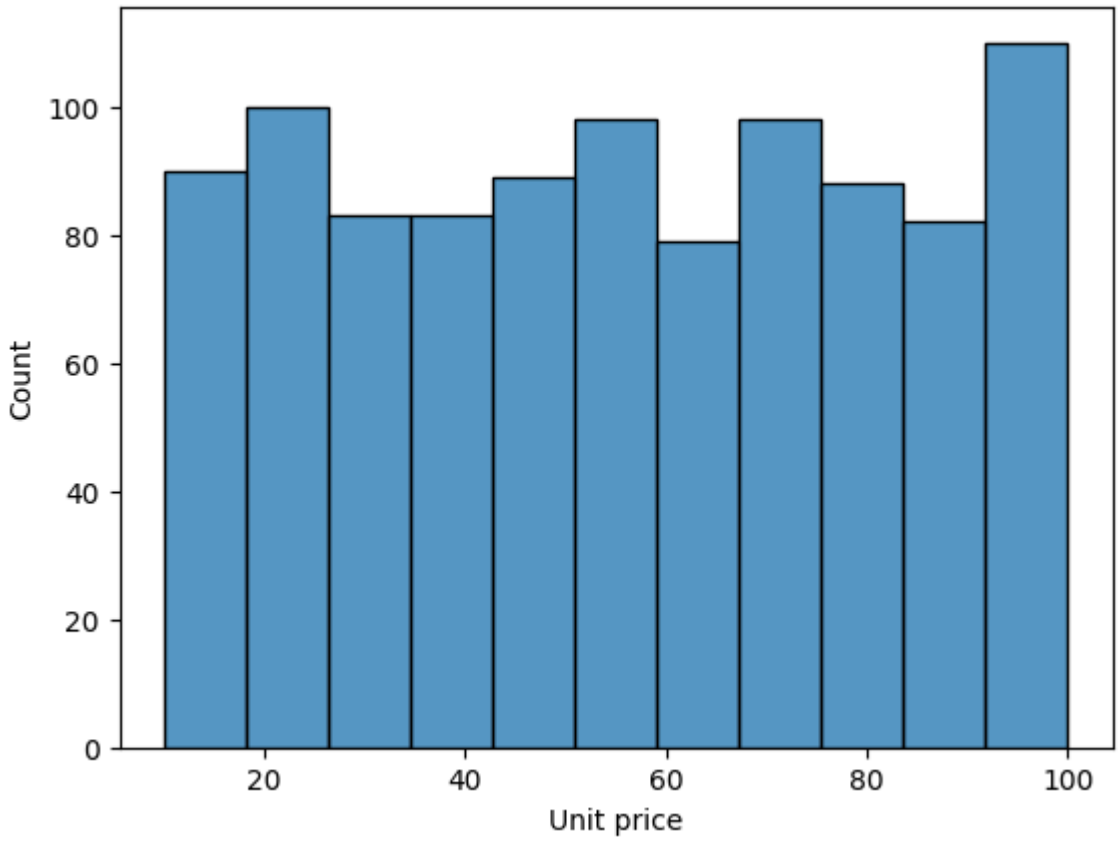
```
duplicate rows based on all columns:
Empty DataFrame
Columns: [Invoice ID, Branch, City, Customer type, Gender, Product line, Unit pri
ce, Quantity, Tax 5%, Total, Date, Time, Payment, cogs, gross margin percentage,
gross income, Rating]
Index: []
```

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

```
0
```

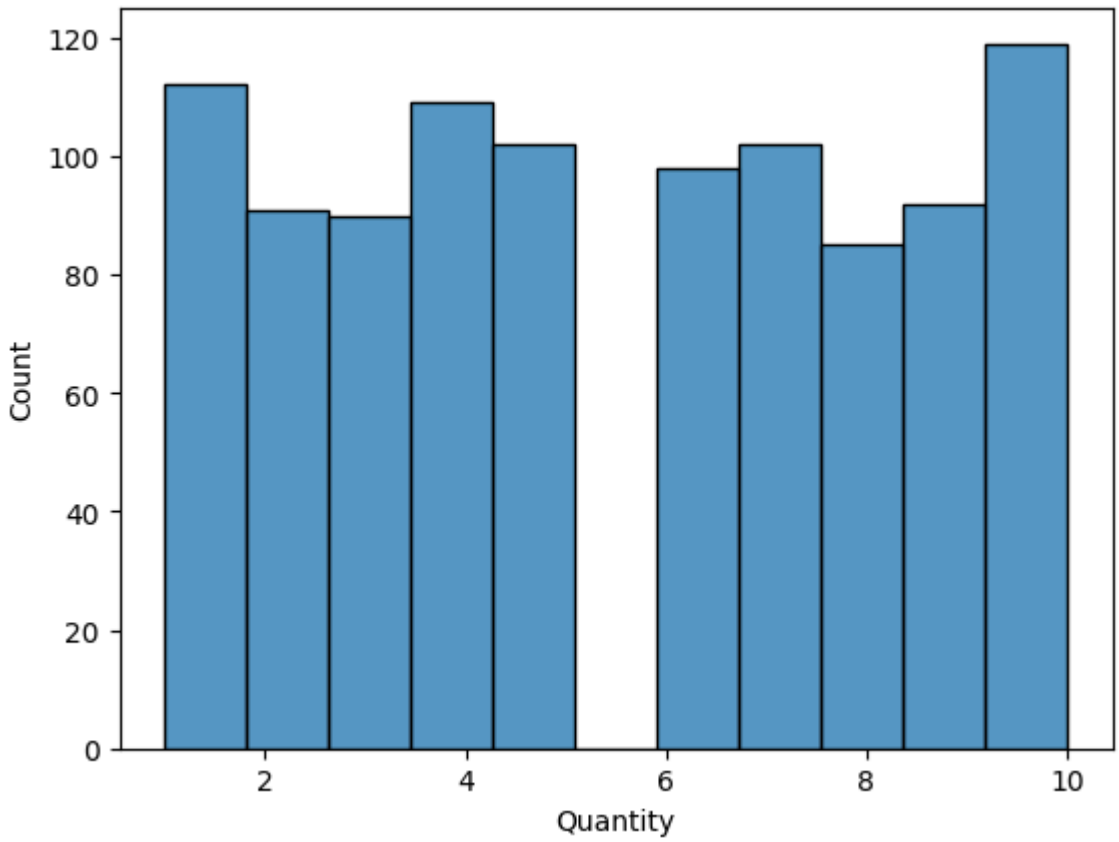
In [10]:
```
#univariat analyst
sns.histplot(x='Unit price',data=df)
```

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



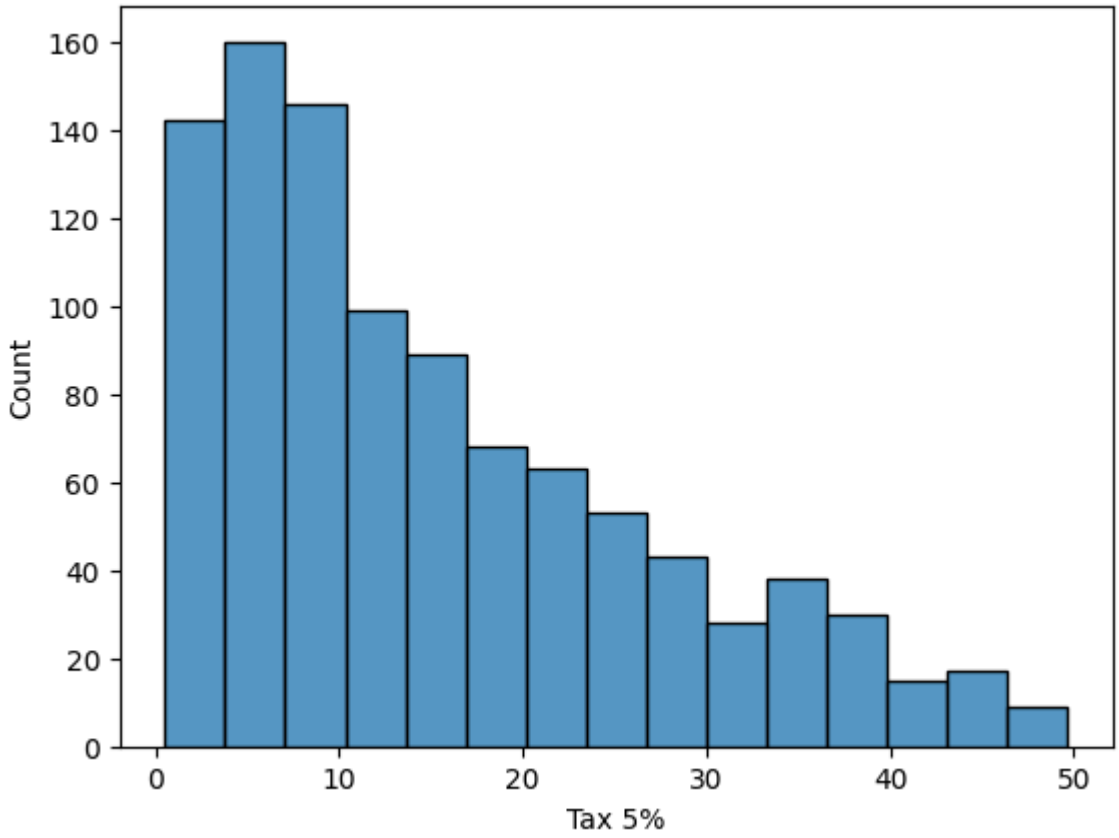
In [11]:
```
#Quantity
sns.histplot(x='Quantity',data=df)
```

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



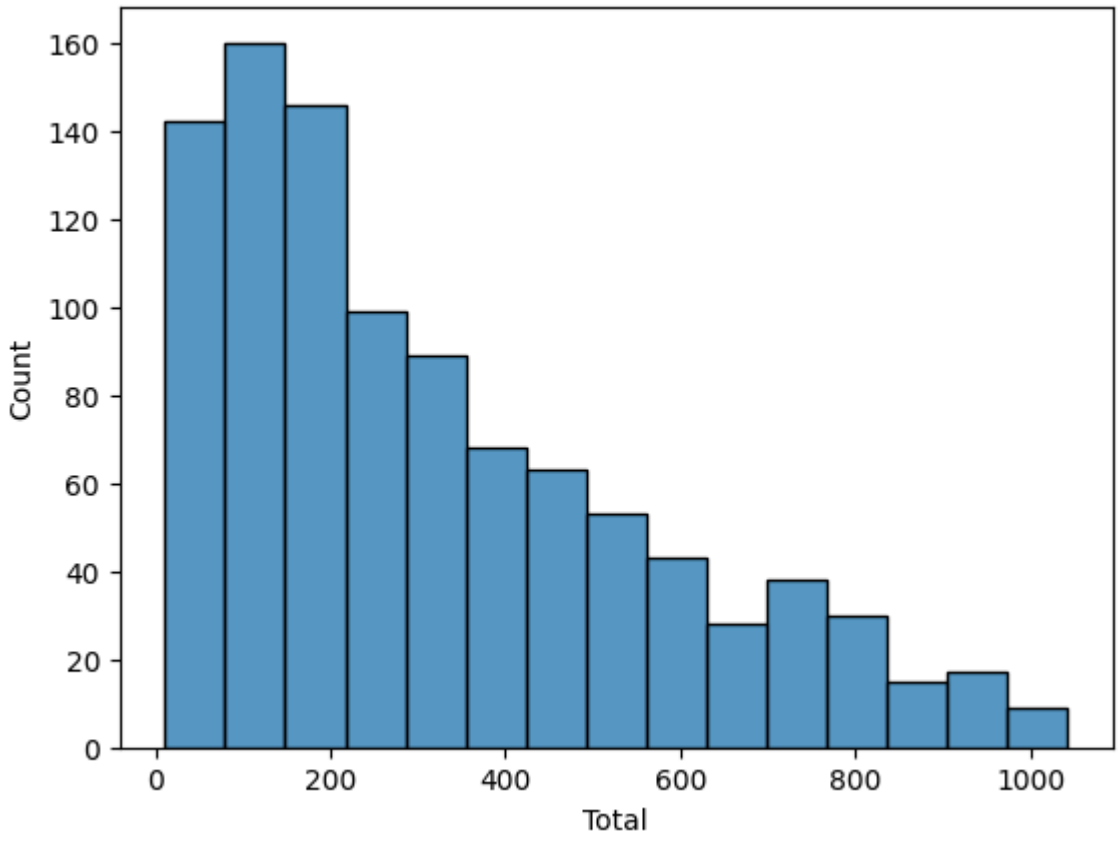
In [12]:
```
sns.histplot(x='Tax 5%',data=df)
```

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



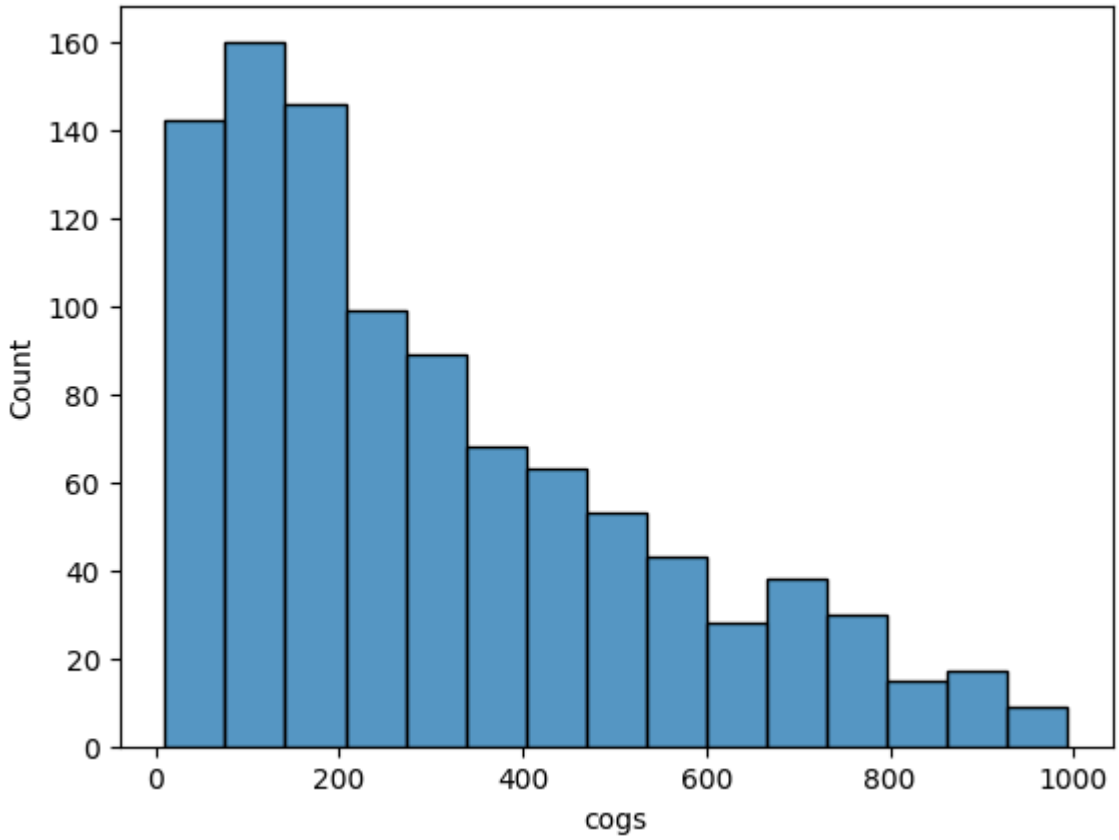
In [13]:
```
sns.histplot(x='Total',data=df)
```

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



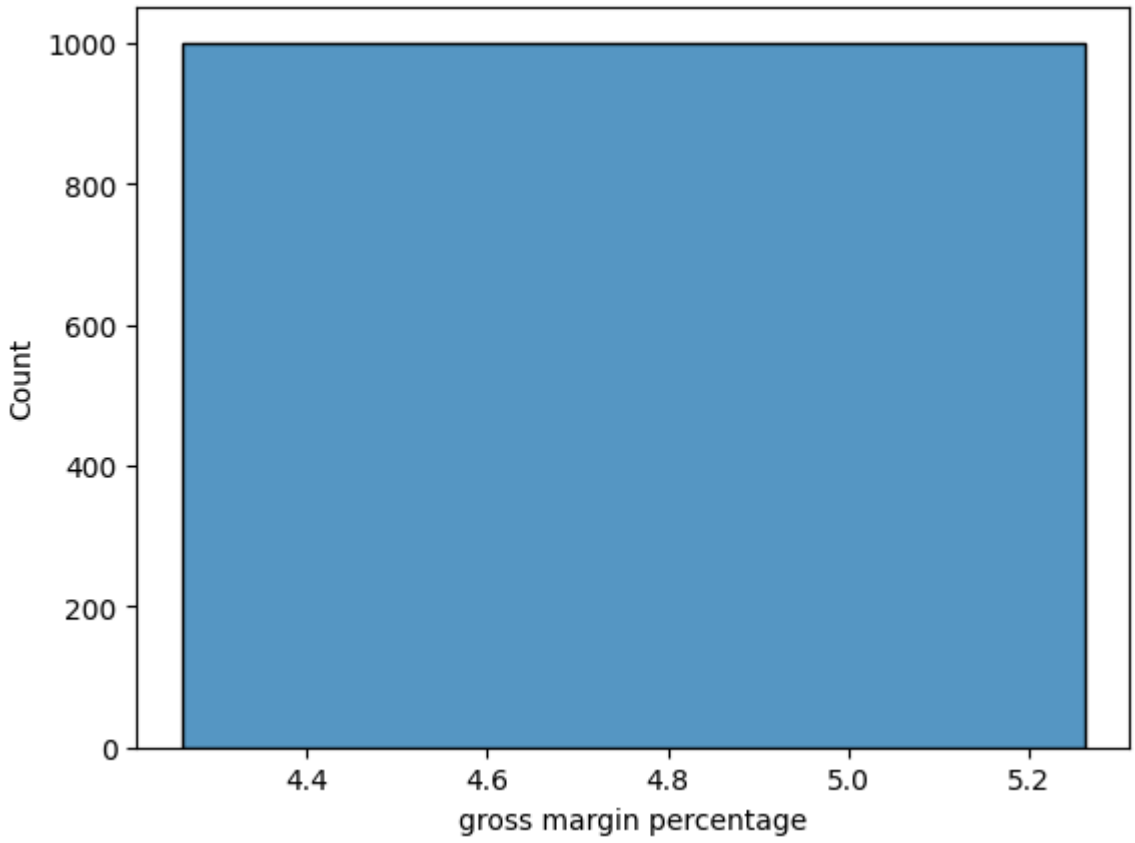
In [14]:
```
sns.histplot(x='cogs',data=df)
```

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



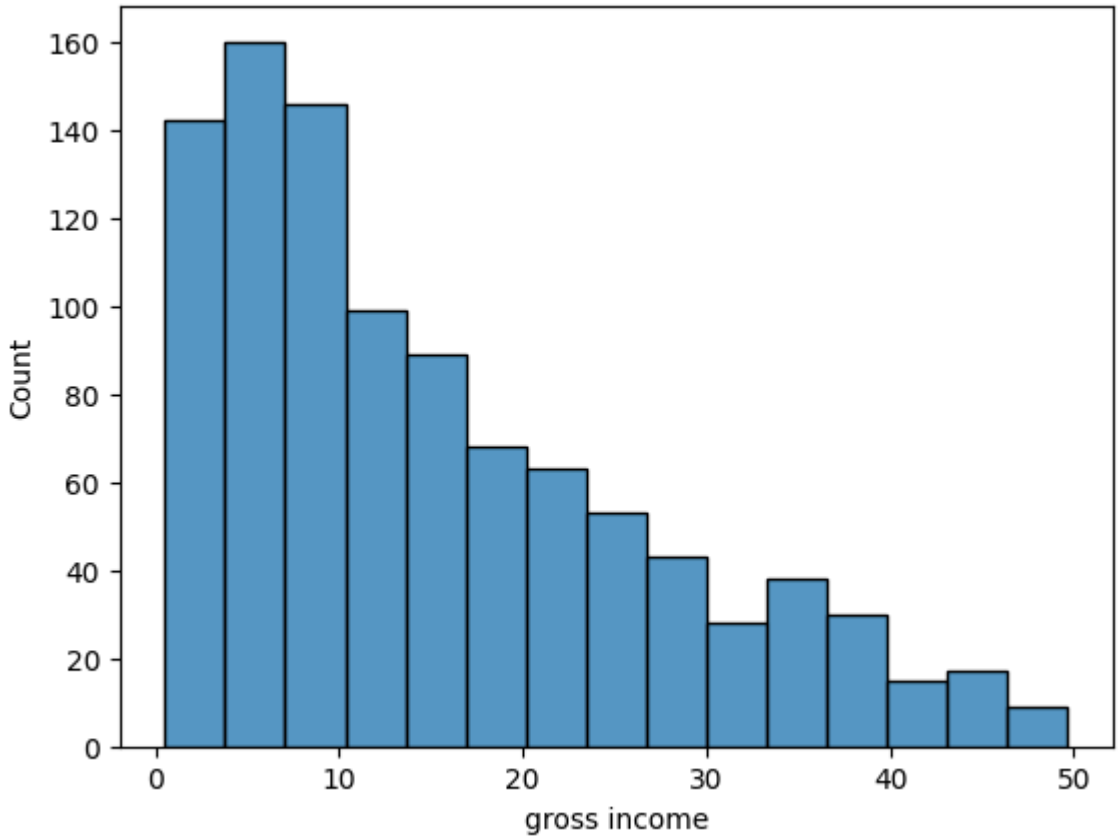
In [15]:
```
sns.histplot(x='gross margin percentage',data=df)
```

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



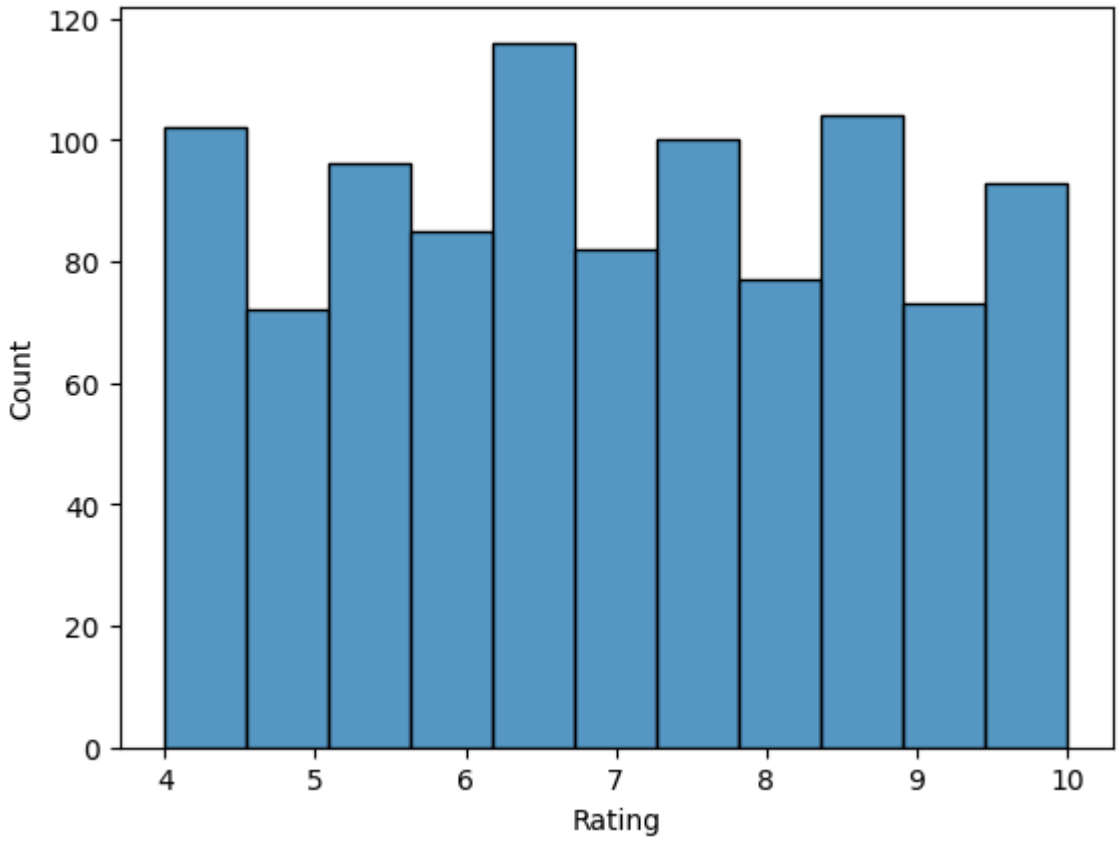
In [16]:
```
sns.histplot(x='gross income',data=df)
```

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



In [17]:
```
sns.histplot(x='Rating',data=df)
```

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

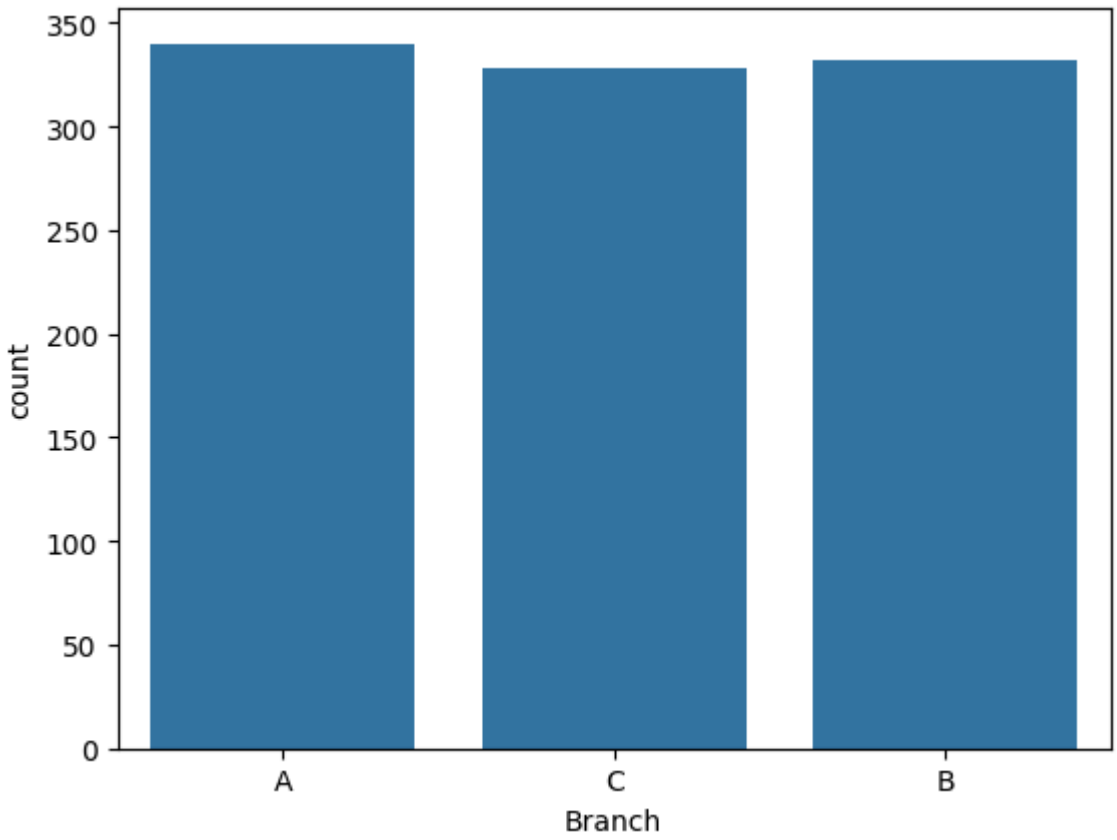


In [18]:
```
df.columns
```

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')
```

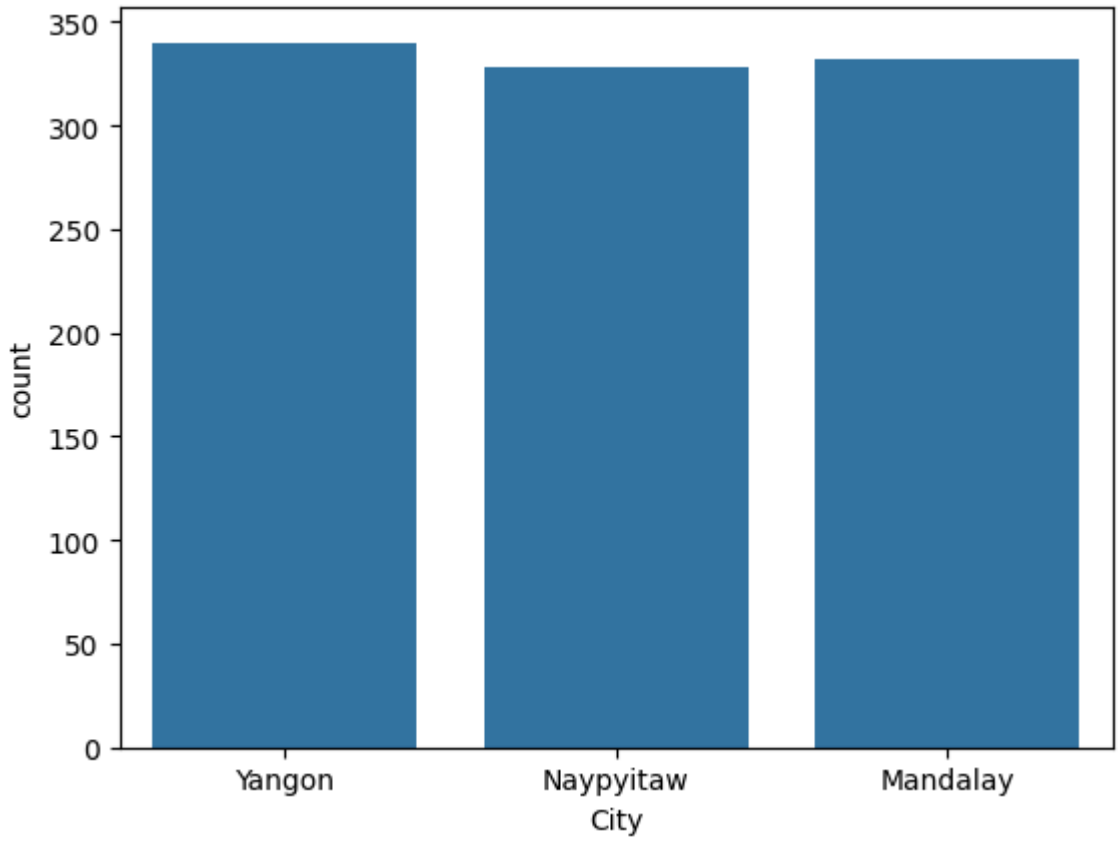
In [19]:
```
#count each category
#branch, city,
sns.countplot(x='Branch',data=df)
df['Branch'].value_counts()
```

Out[19]:
```
Branch
A    340
B    332
C    328
Name: count, dtype: int64
```



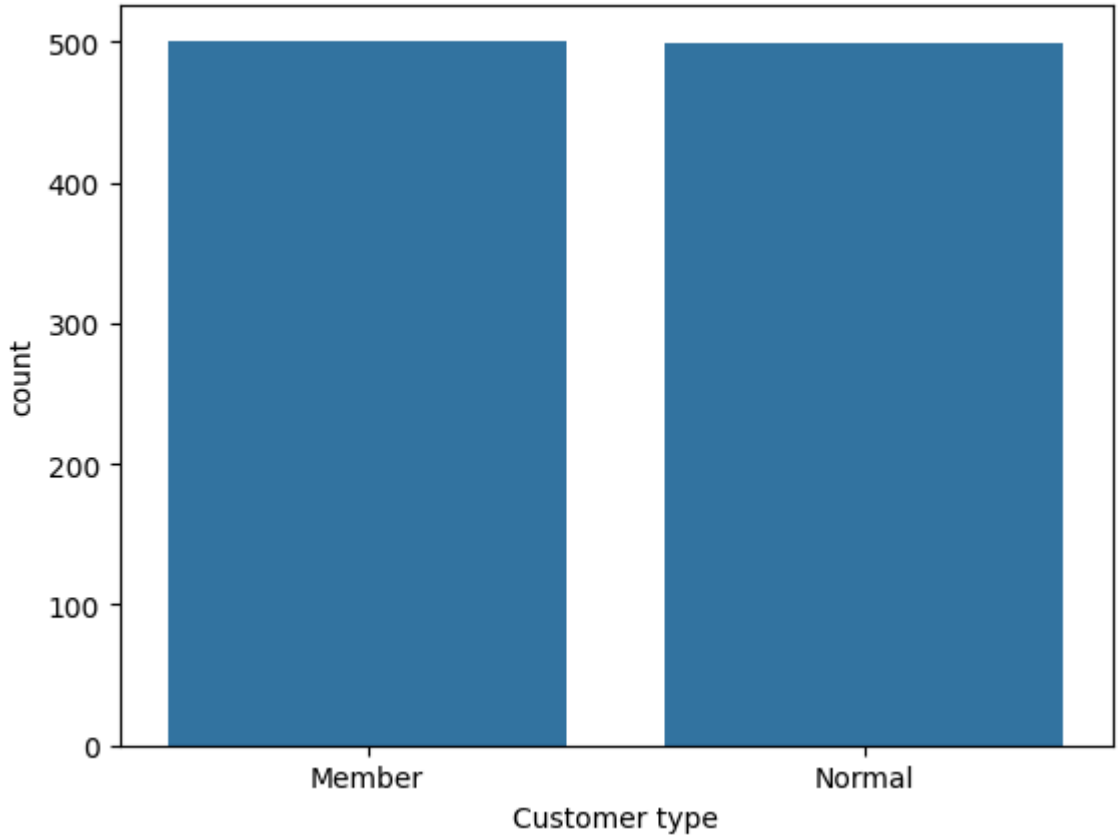
In [20]:
```
sns.countplot(x='City',data=df)
df['City'].value_counts()
```

Out[20]:
```
City
Yangon       340
Mandalay     332
Naypyitaw    328
Name: count, dtype: int64
```



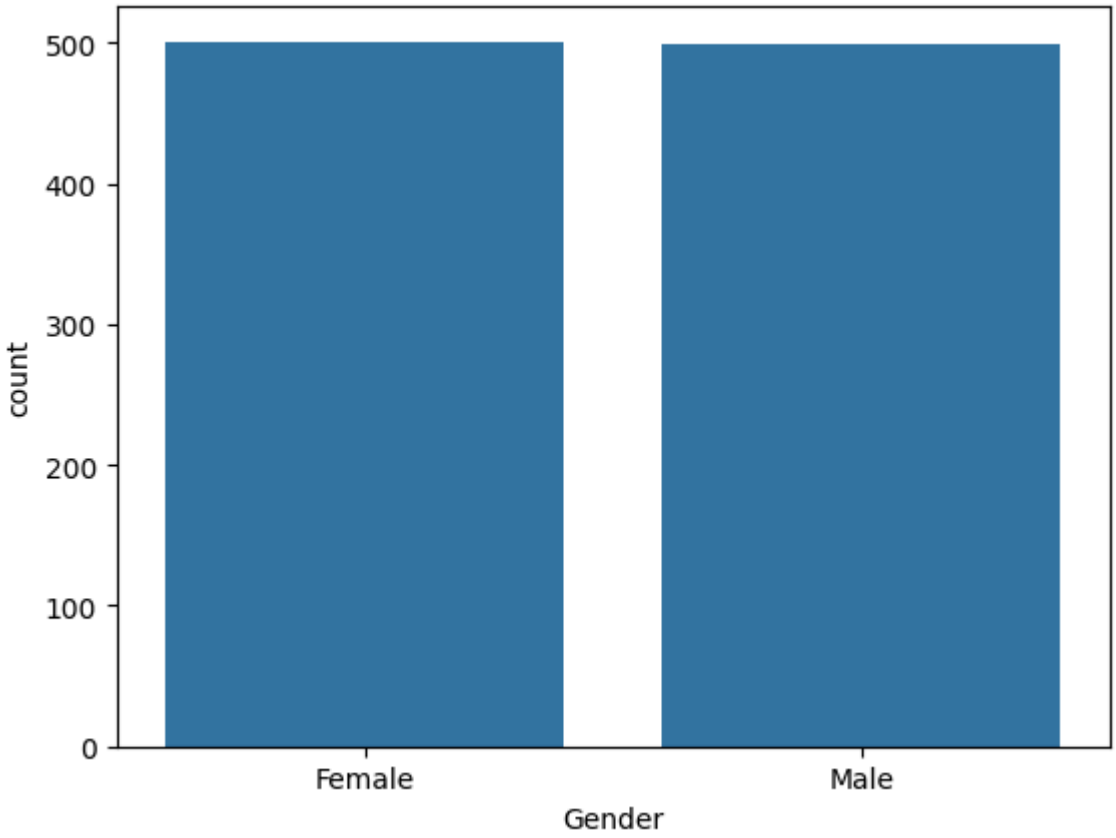
In [21]:
```
sns.countplot(x='Customer type',data=df)
df['Customer type'].value_counts()
```

Out[21]:
```
Customer type
Member    501
Normal    499
Name: count, dtype: int64
```

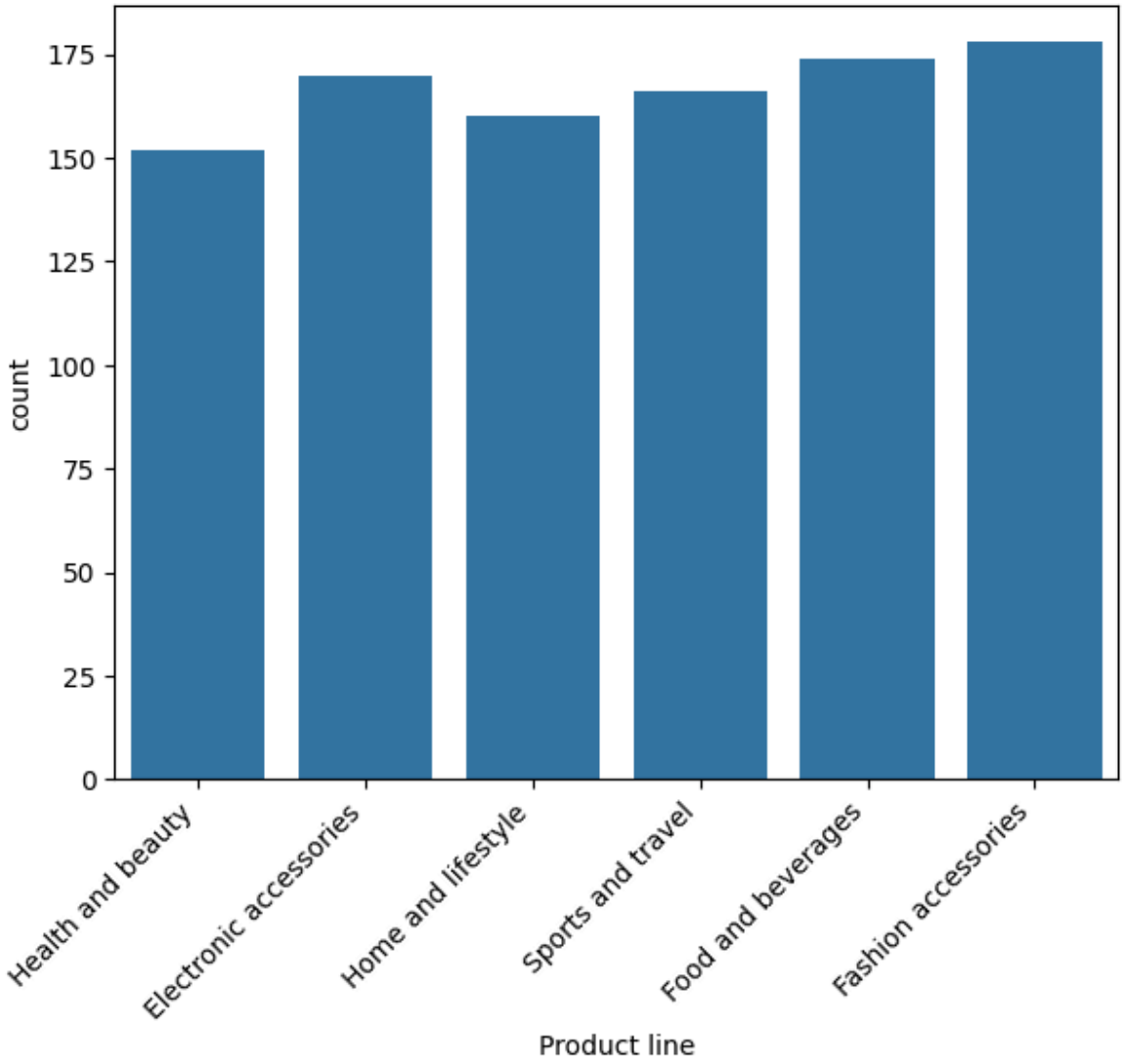


In [22]:
```
sns.countplot(x='Gender',data=df)
df['Gender'].value_counts()
```

Out[22]:
```
Gender
Female    501
Male      499
Name: count, dtype: int64
```

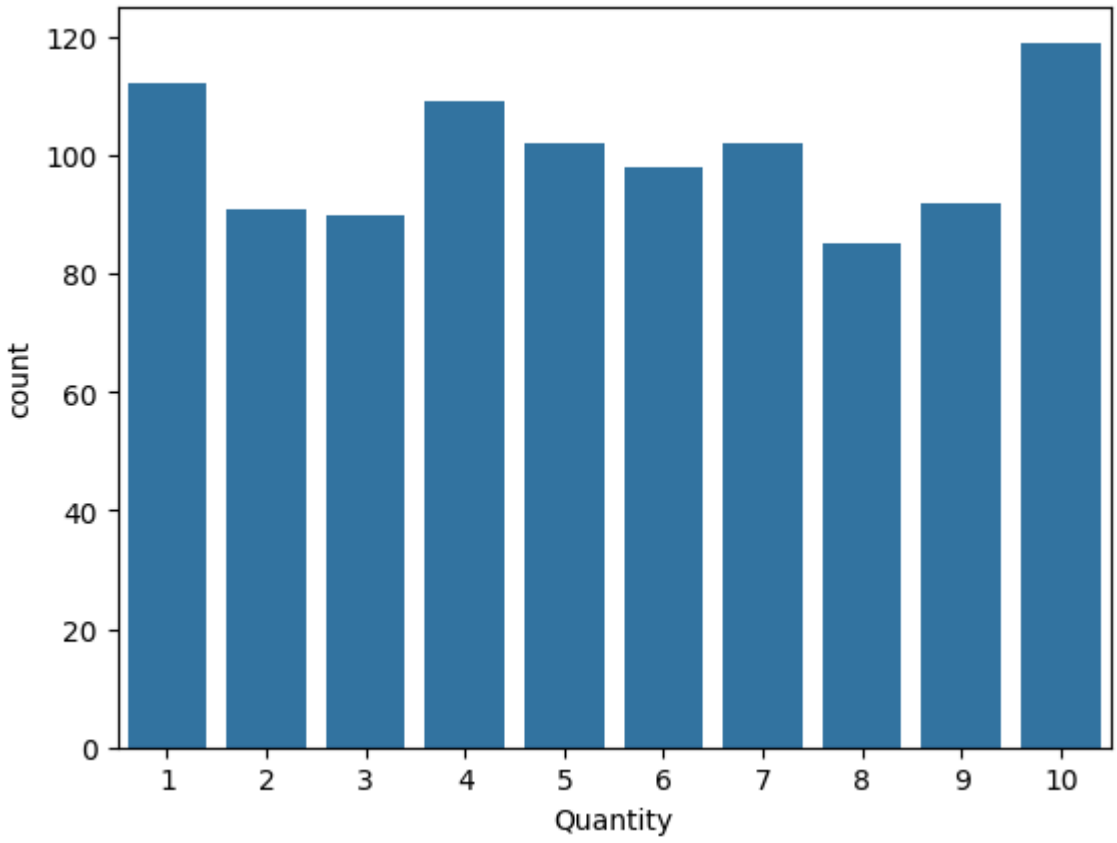


In [23]:
```
sns.countplot(x='Product line',data=df)
plt.tight_layout()
plt.xticks(rotation=45,ha='right')
df['Product line'].value_counts()
plt.show()
```



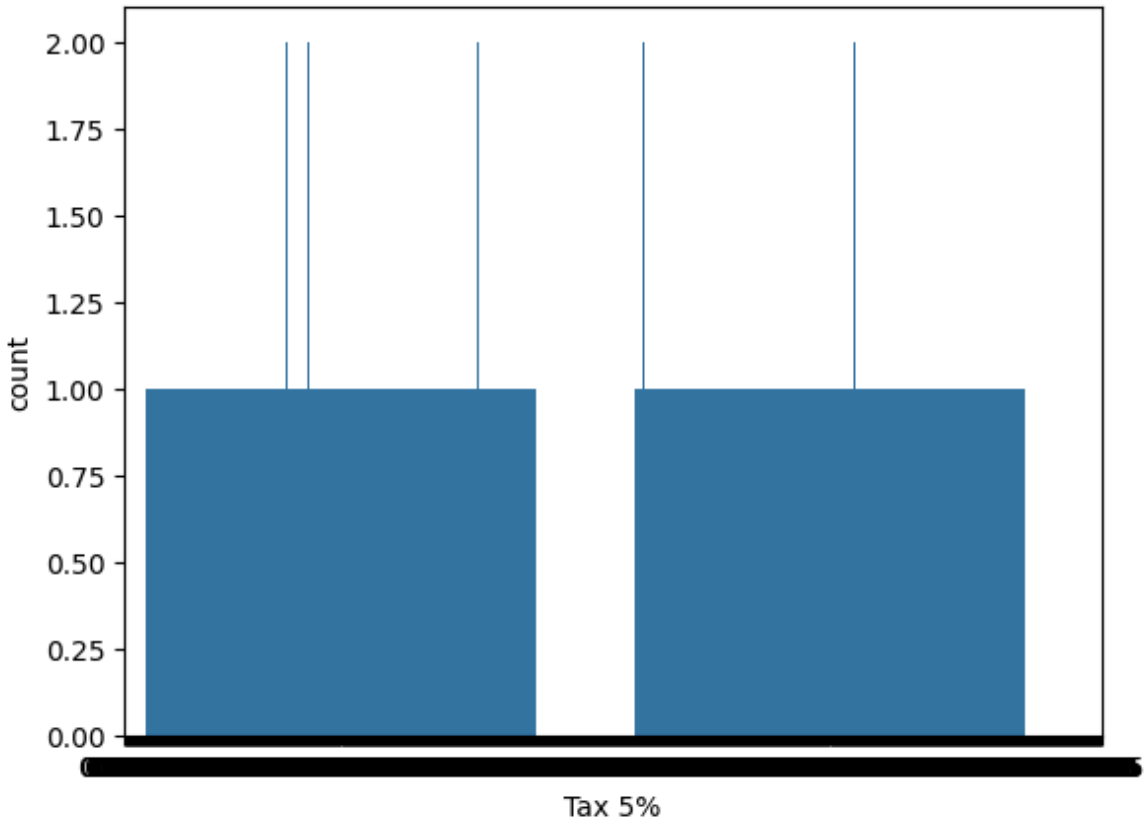
In [24]:
```
sns.countplot(x='Quantity',data=df)
```

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



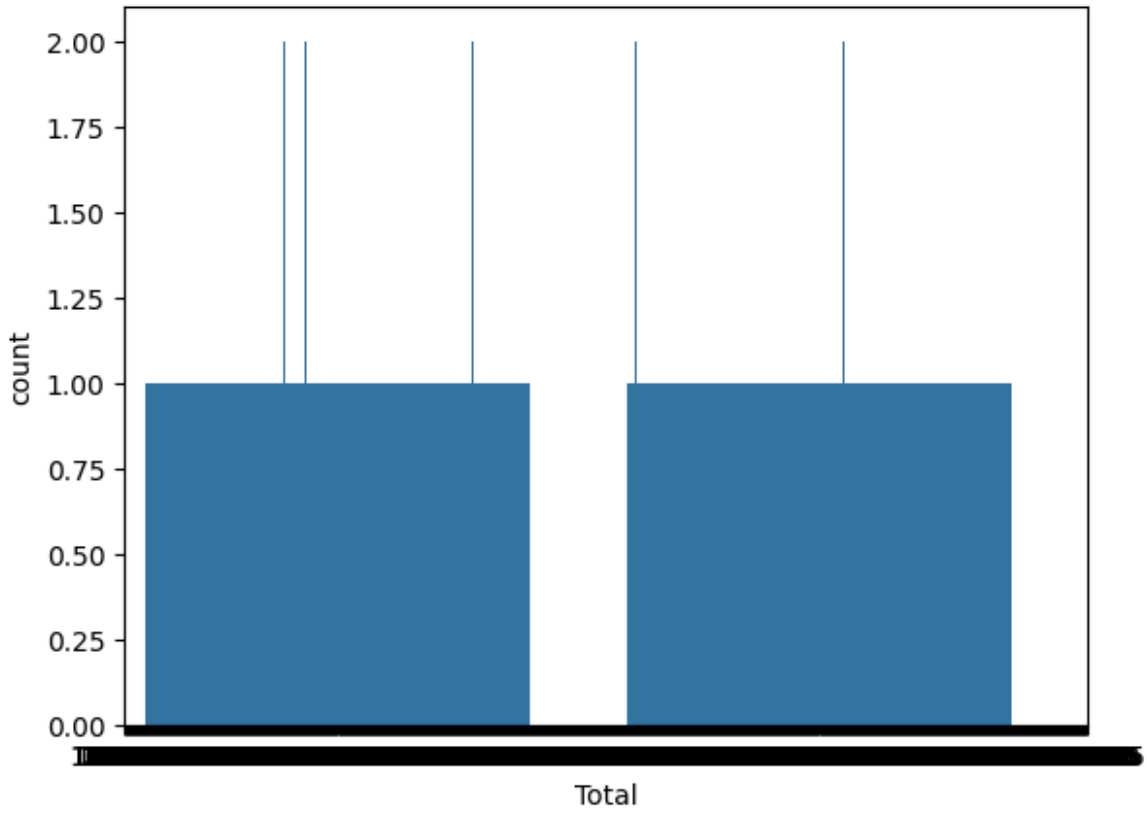
In [25]:
```
sns.countplot(x='Tax 5%',data=df)
```

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



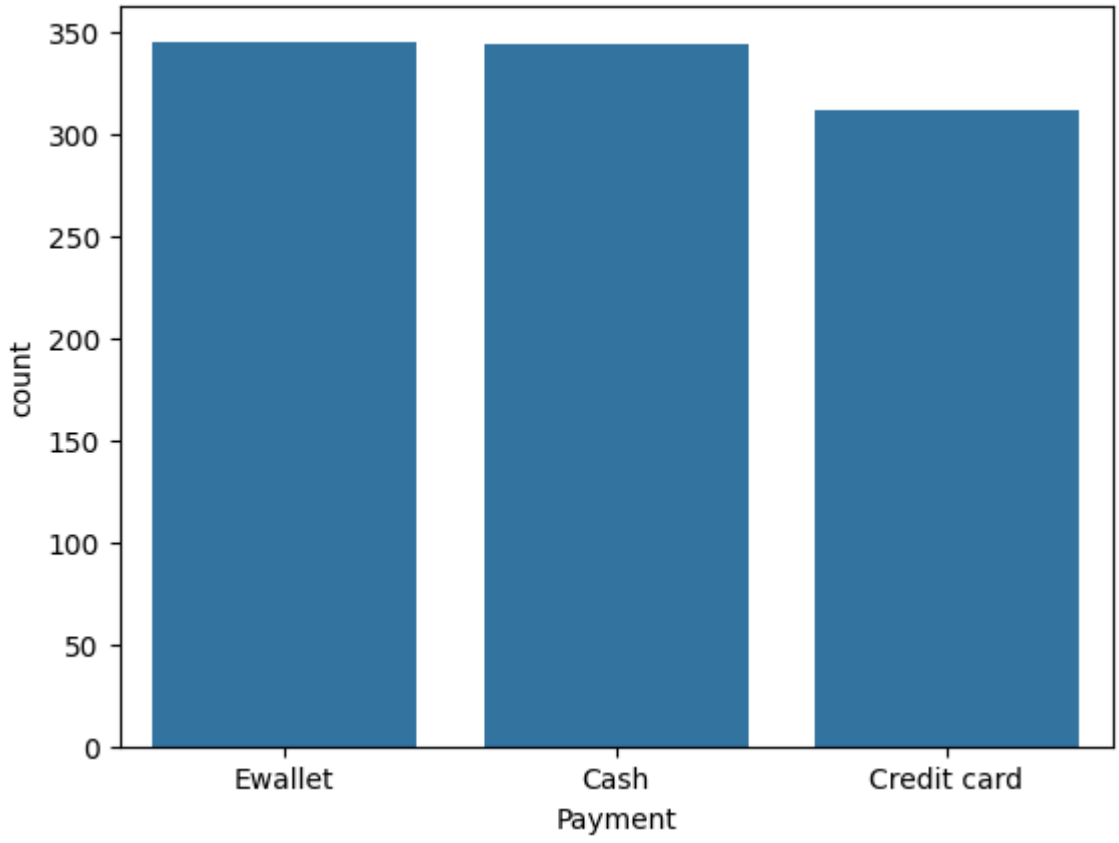
In [26]:
```
sns.countplot(x='Total',data=df)
```

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



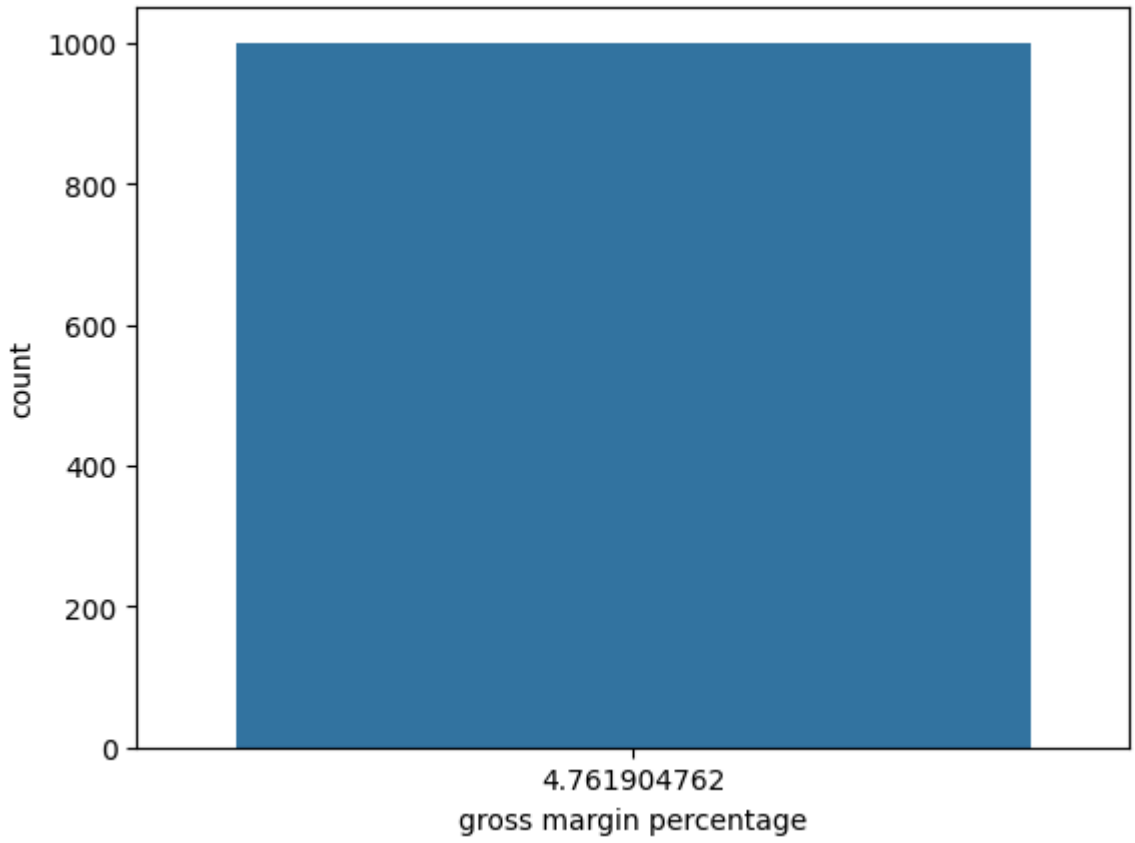
In [27]:
```
sns.countplot(x='Payment', data=df)
```

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



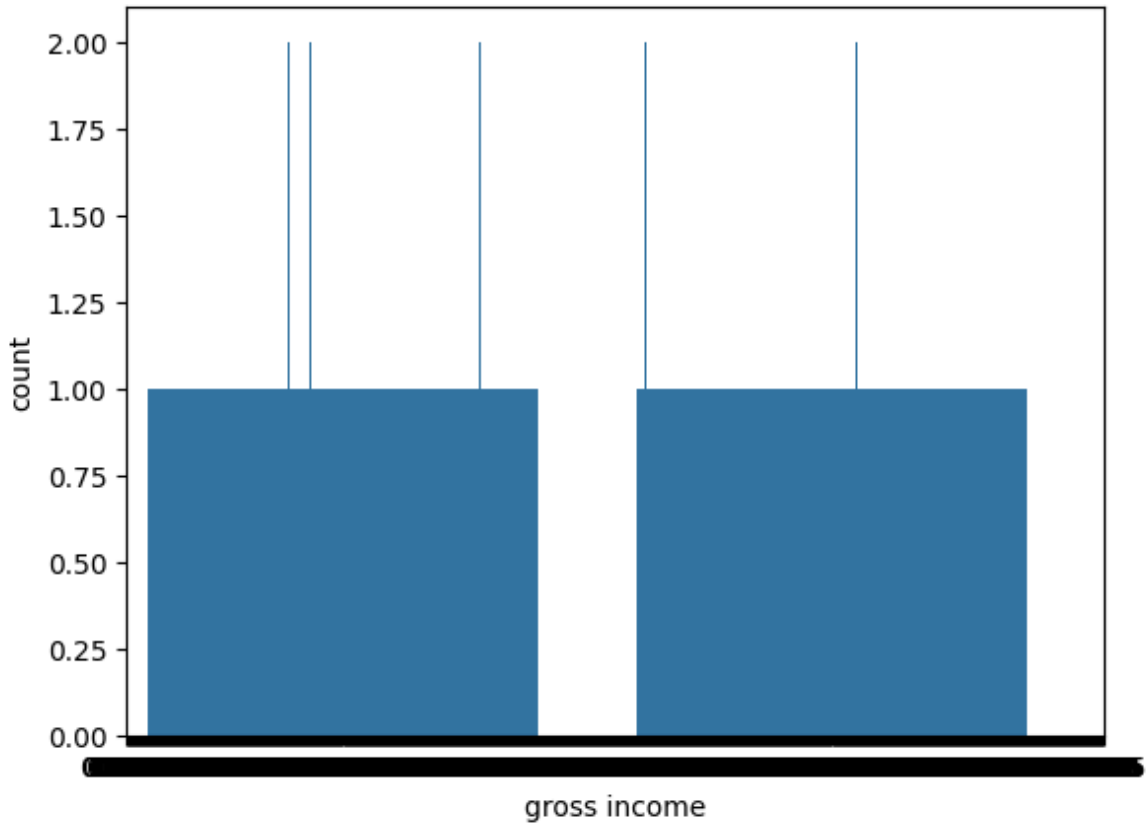
In [28]:
```
sns.countplot(x='gross margin percentage',data=df)
```

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



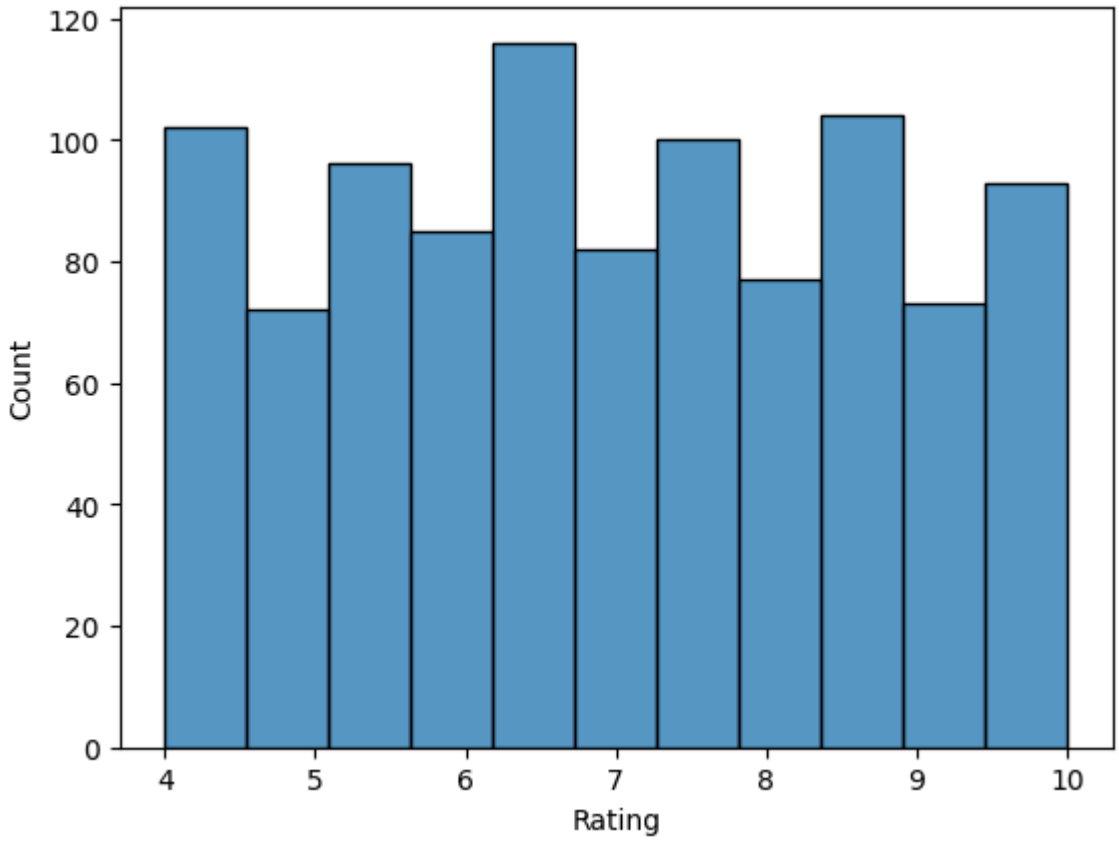
In [29]:
```
sns.countplot(x='gross income', data=df)
```

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



In [30]:
```
sns.histplot(x='Rating', data=df)
```

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



In [1]:
```
#counvarting data column into data
df['Date']=pd.to_datetime(df['Date'])
```

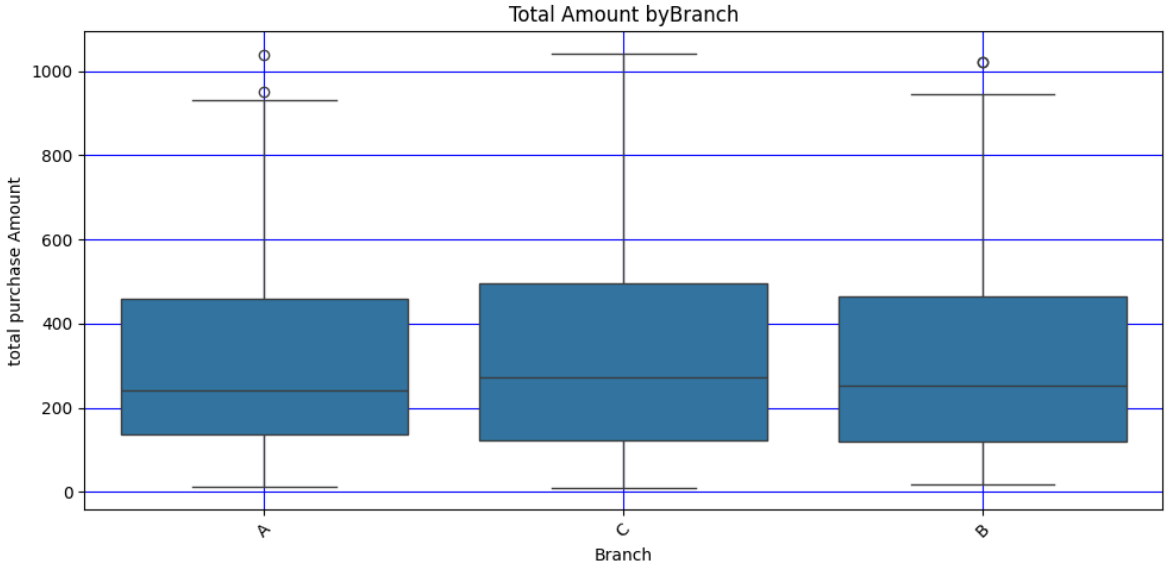
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 2
      1 #counvarting data column into data
----> 2 df['Date']=pd.to_datetime(df['Date'])

NameError: name 'pd' is not defined
```

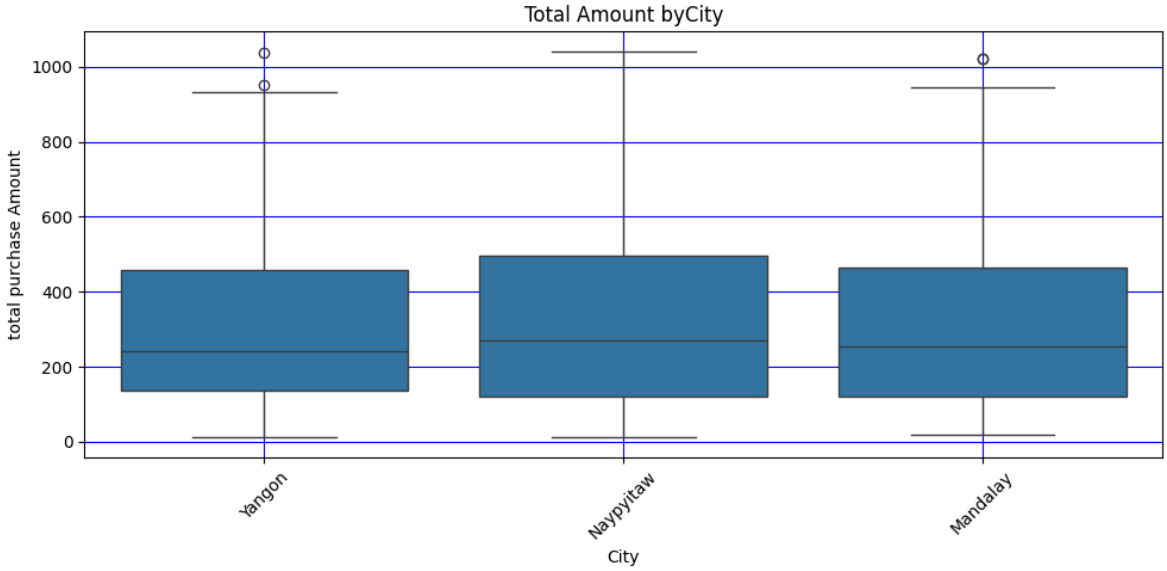
In [ ]:
```
df['Date'].isnull().sum()
```

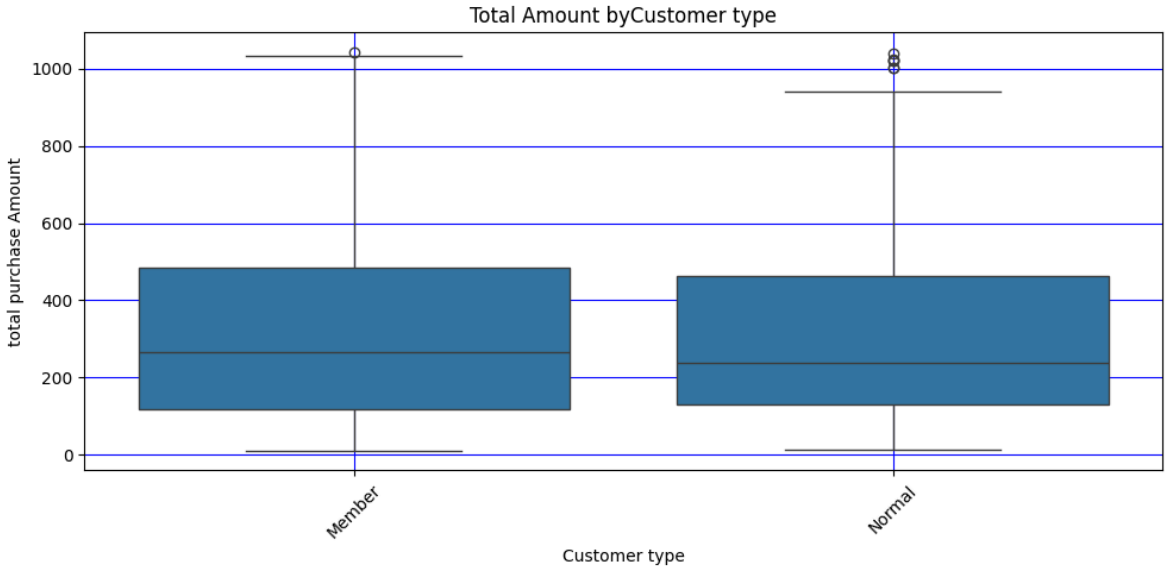
In [ ]:
```
df['Date'].fillna(method='ffill',inplace=True)
```

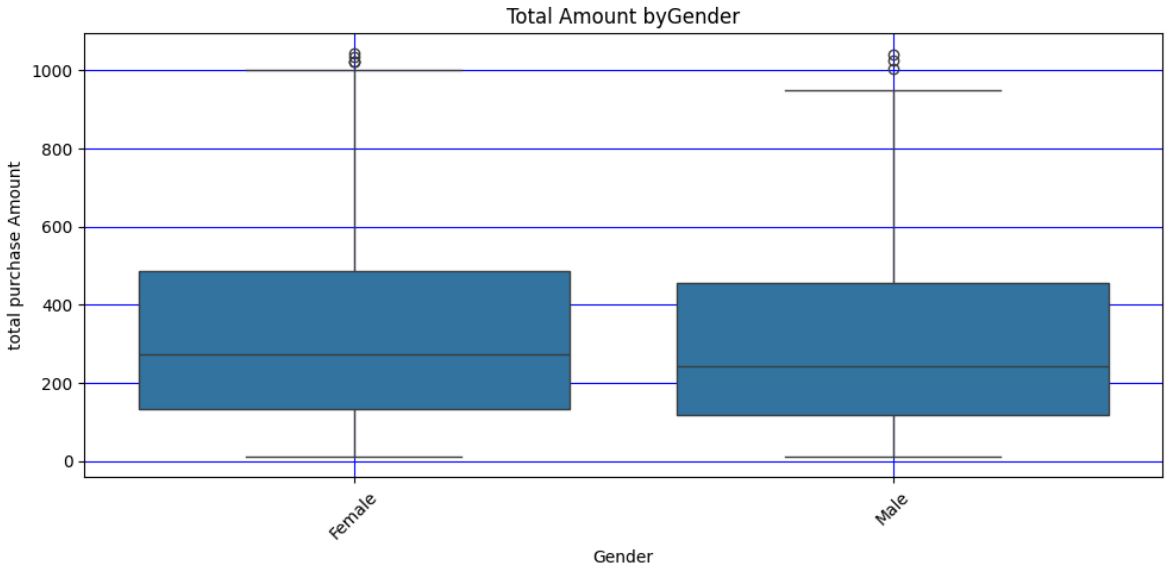
In [34]:
```
#list of categorical columns for bivariate analysis
categorical_cols=['Branch', 'City', 'Customer type', 'Gender','Payment',
       'Product line']
print(categorical_cols)
#criating box plot with loop for bivariate analysis
#total vs categorical
# plot boxplots for 'total' vs categorical 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()
```

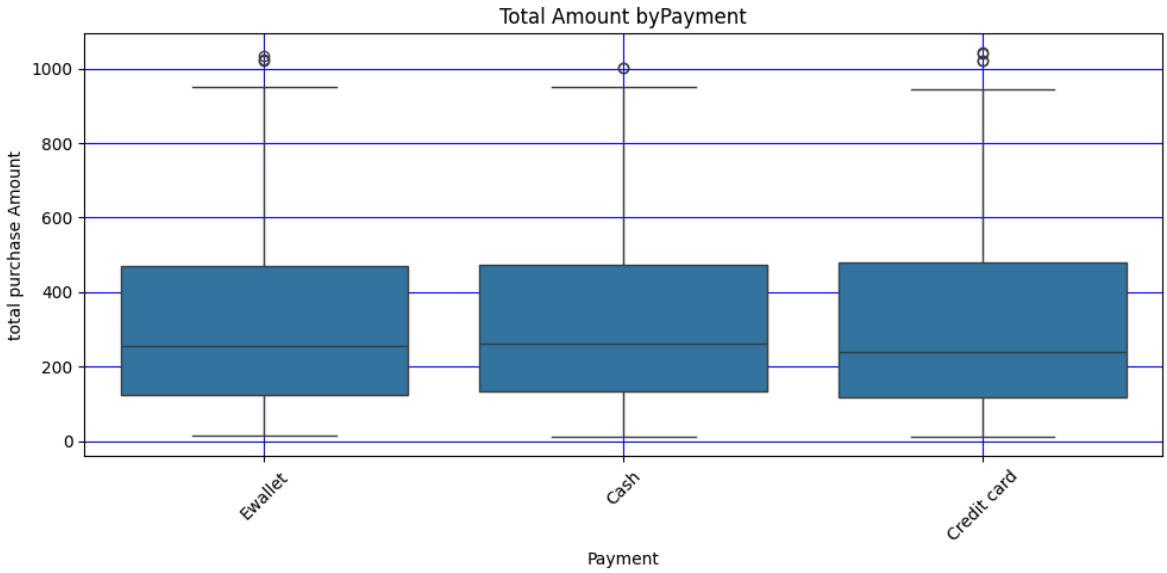
```
['Branch', 'City', 'Customer type', 'Gender', 'Payment', 'Product line']
```

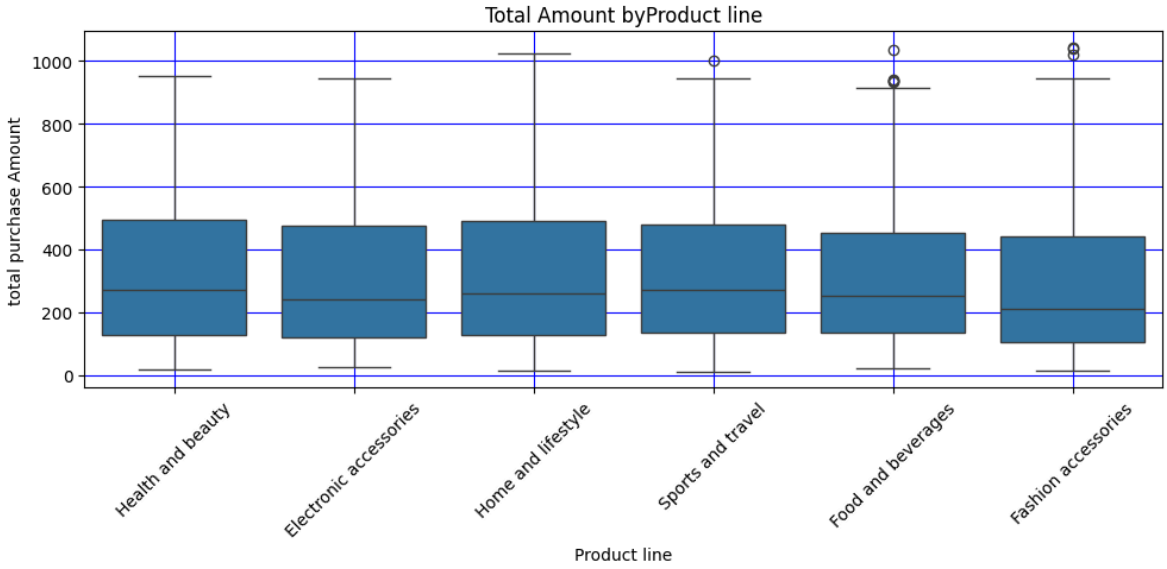






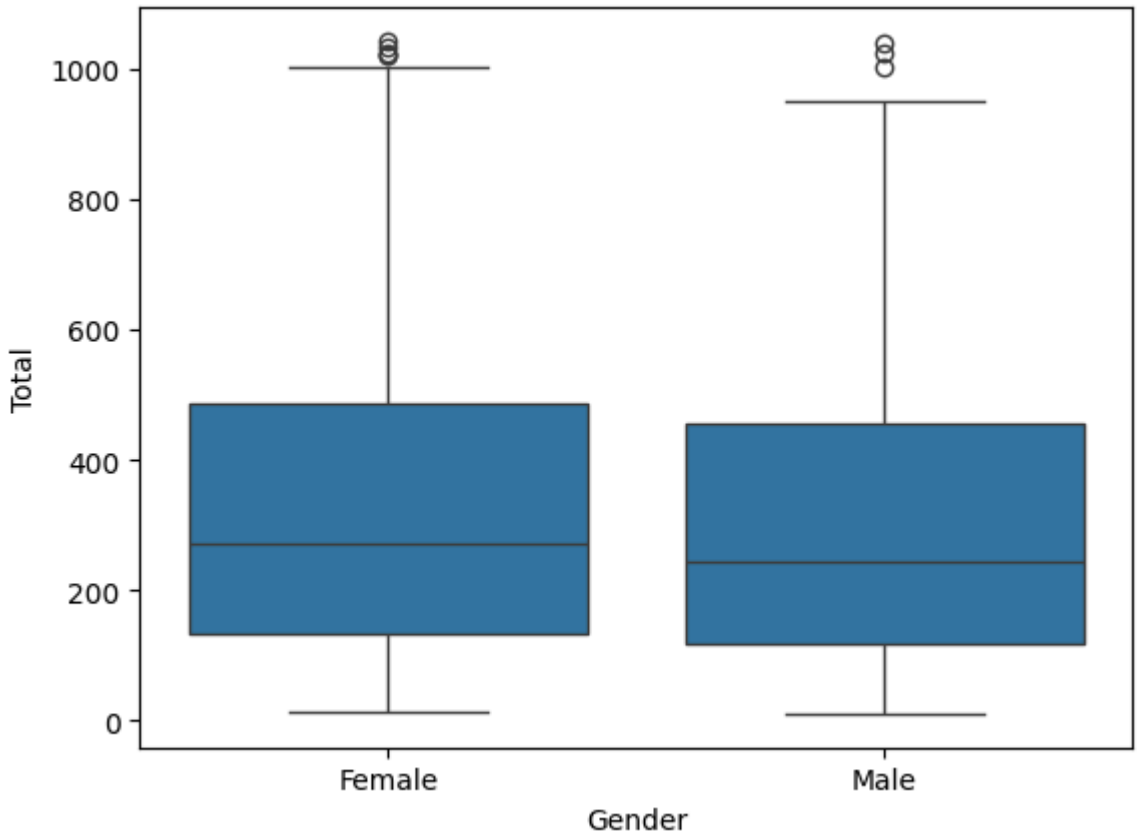






In [35]:
```
#box plot total,pl g,m,p
sns.boxplot(x='Gender',y='Total',data=df)
```

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



In [36]:
```
#Identify numarical columns
df=pd.read_csv('supermarket_sales - Sheet1.csv')
df
```

Out[36]:

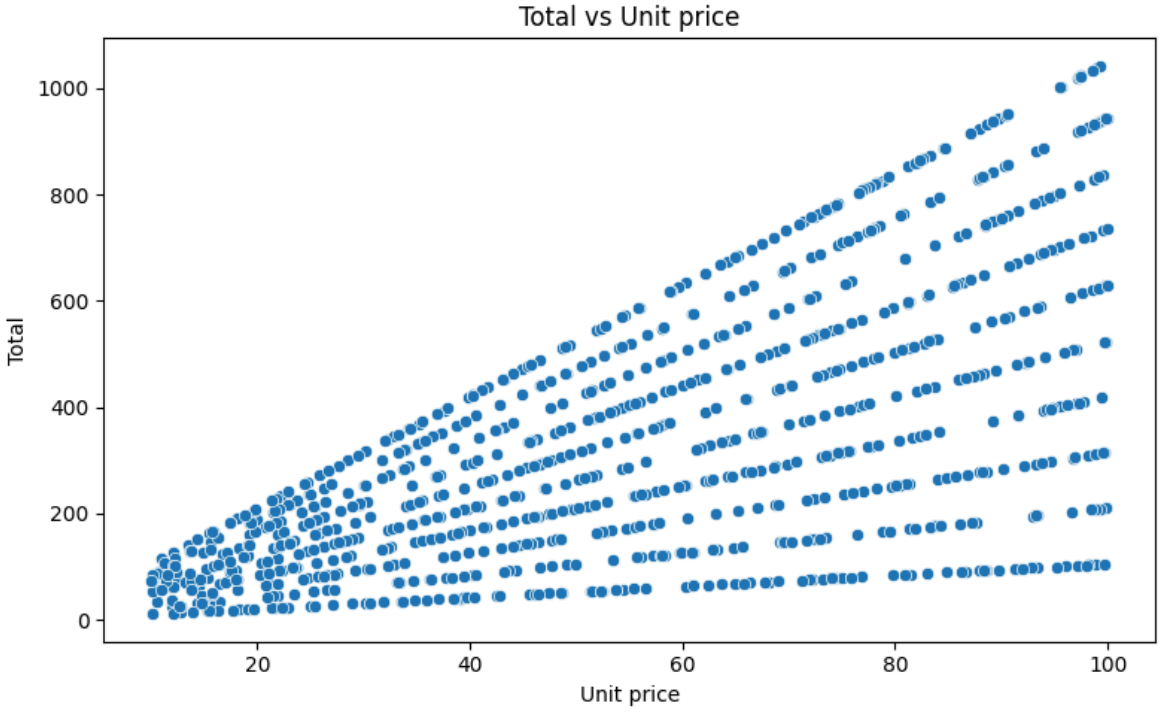
	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.14
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.82
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.21
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.28
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.20
...	...	...	...	...	...	...	...	...	
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.01
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.69
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.59
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.29
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.91

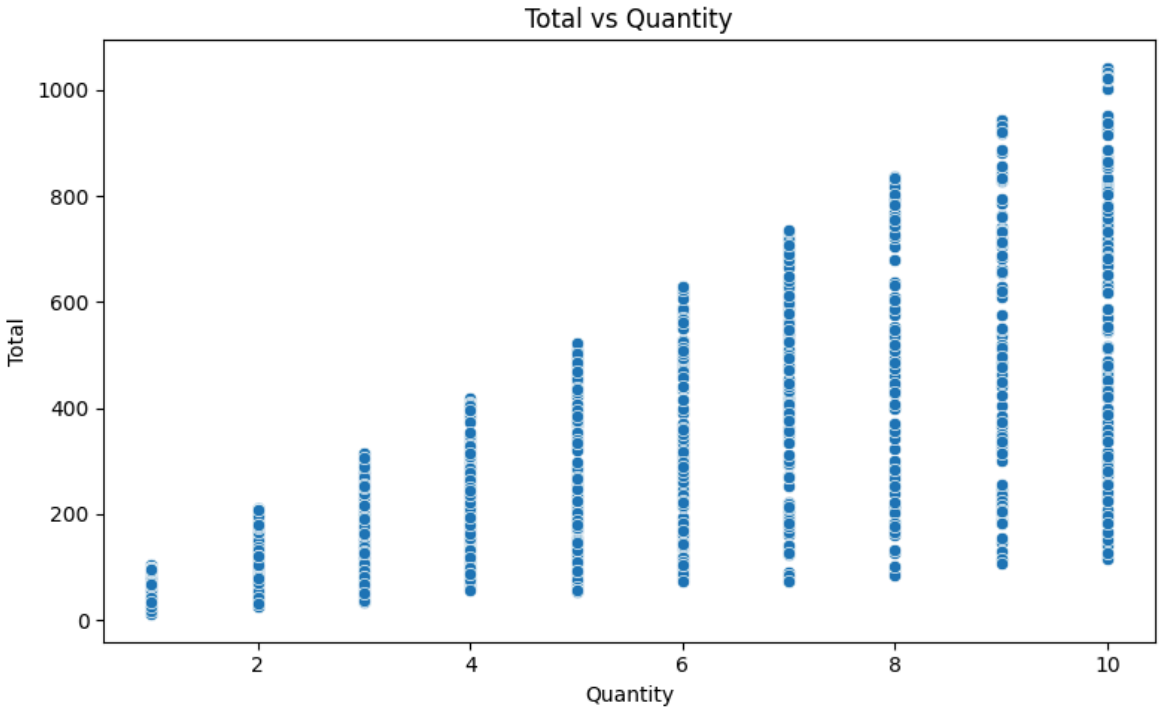
1000 rows × 17 columns

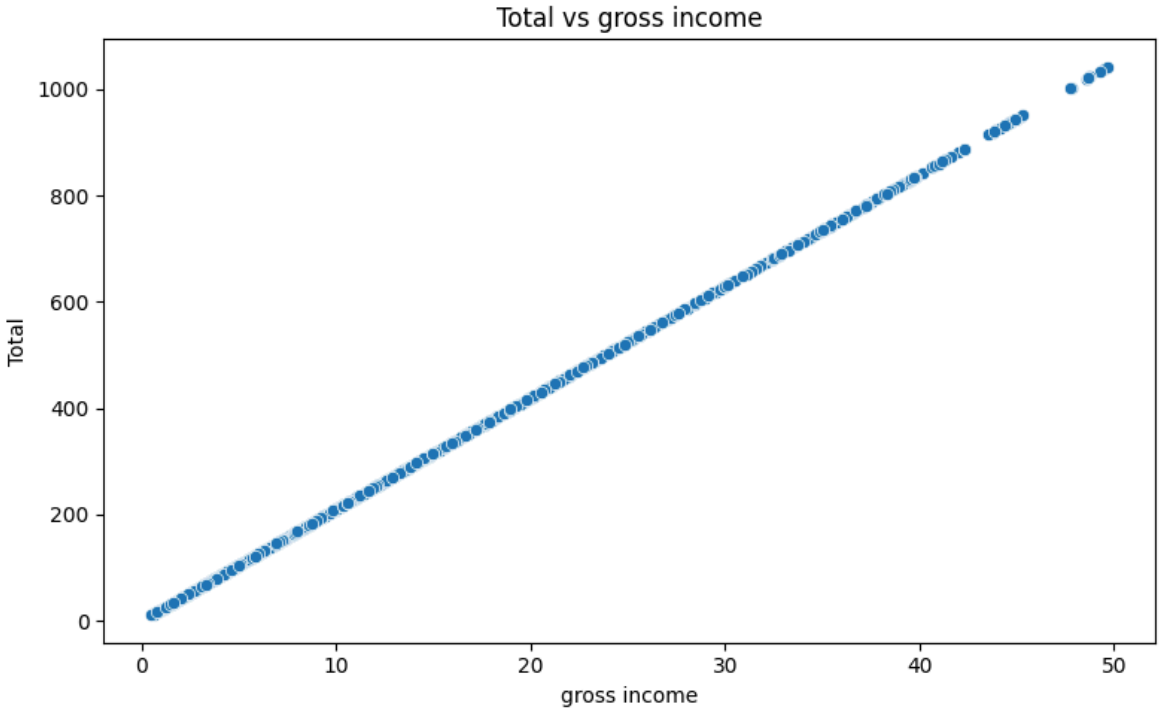
In [41]:
```
numerical_cols=df.select_dtypes(include=['float64','int64']).columns
print(numerical_cols)
scatter_pairs=[
    ('Unit price','Total'),
     ('Quantity','Total'),
     ('gross income','Total'),
     ('Rating','Total')
 ]
print(scatter_pairs)
for x,y in scatter_pairs:
     plt.figure(figsize=(8,5))
     sns.scatterplot(data=df,x=x,y=y)
```

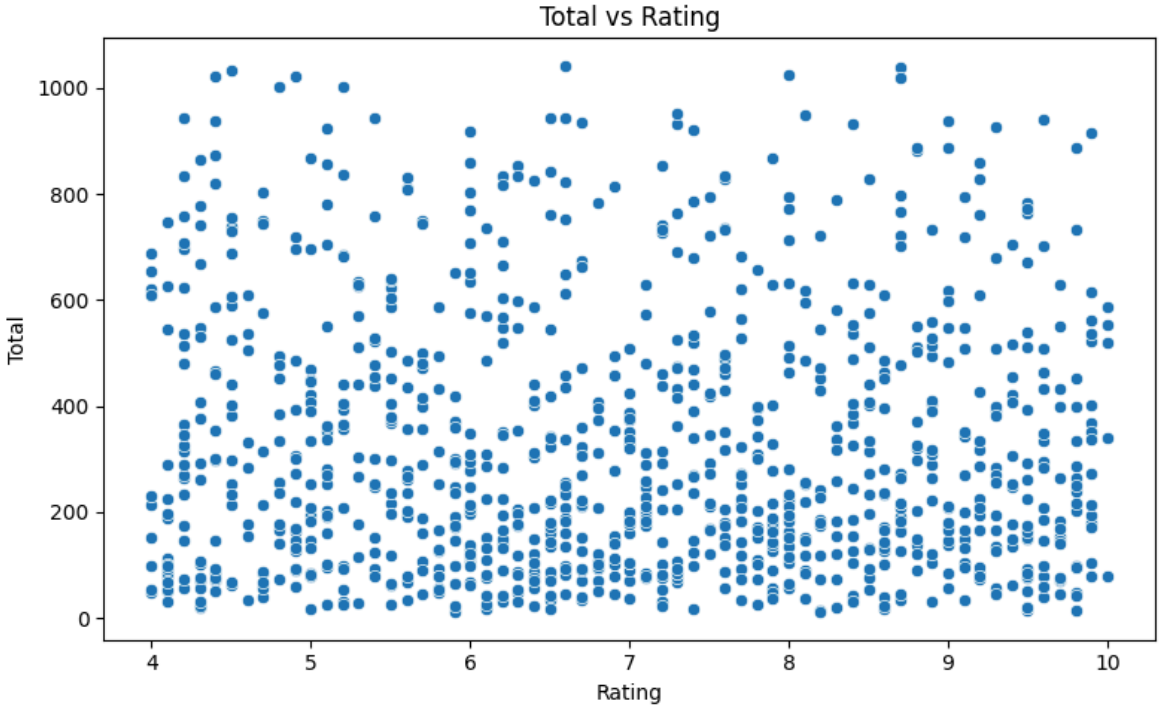
```
    plt.title(f'{y} vs {x}')
    plt.tight_layout()
```

```
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')]
```









In [42]:
```
#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
```

In [43]:
```
df.dtypes
```

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 ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.14
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.82
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.21
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.28
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.20
...	...	...	...	...	...	...	...	...	
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.01
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.69
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.59
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.29
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.91

1000 rows × 21 columns

In [51]:
```
#covert 'Time' to hour only
df['Hour']=pd.to_datetime(df['Time'],format='%H,:%M,%S',errors='coerce').apply(l
```

In [52]:
```
df
```

Out[52]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.14
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.82
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.21
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.28
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.20
...	...	...	...	...	...	...	...	...	
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.01
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.69
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.59
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.29
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.91

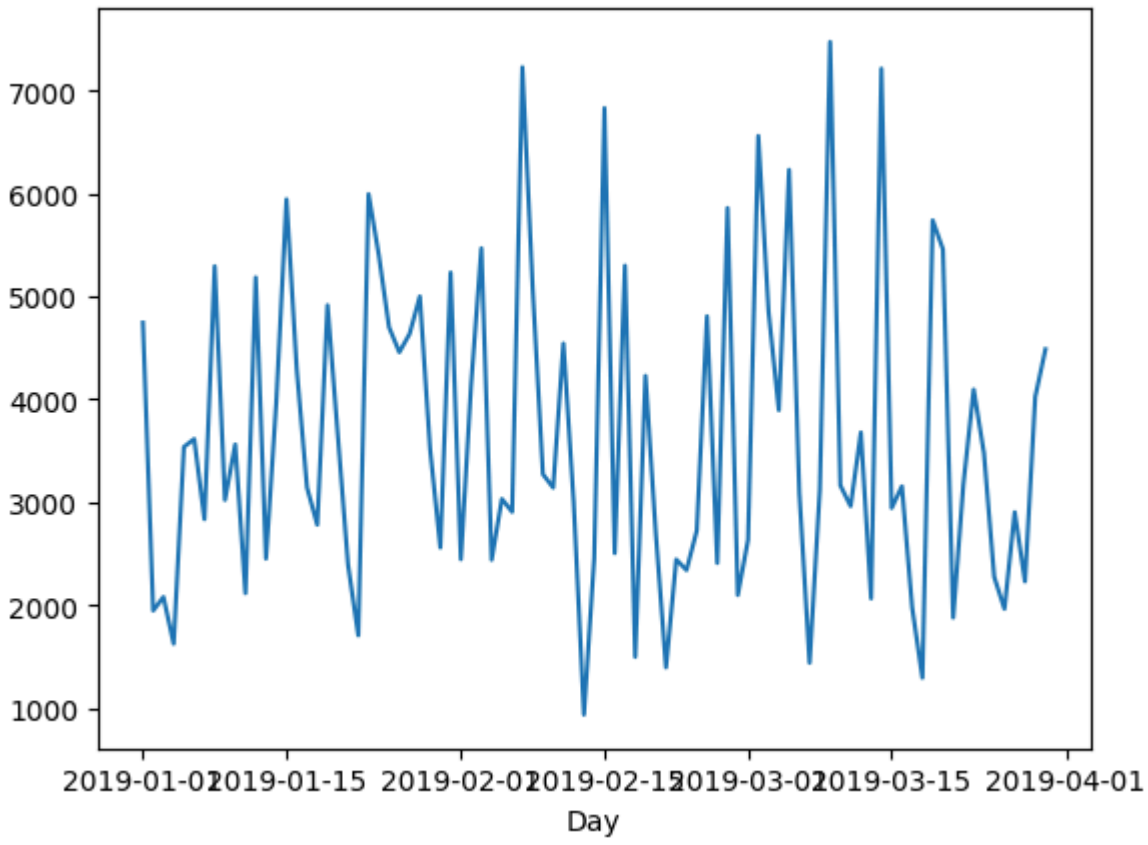
1000 rows × 21 columns

In [53]:
```
#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)
```

```
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
Name: Total, dtype: float64
```

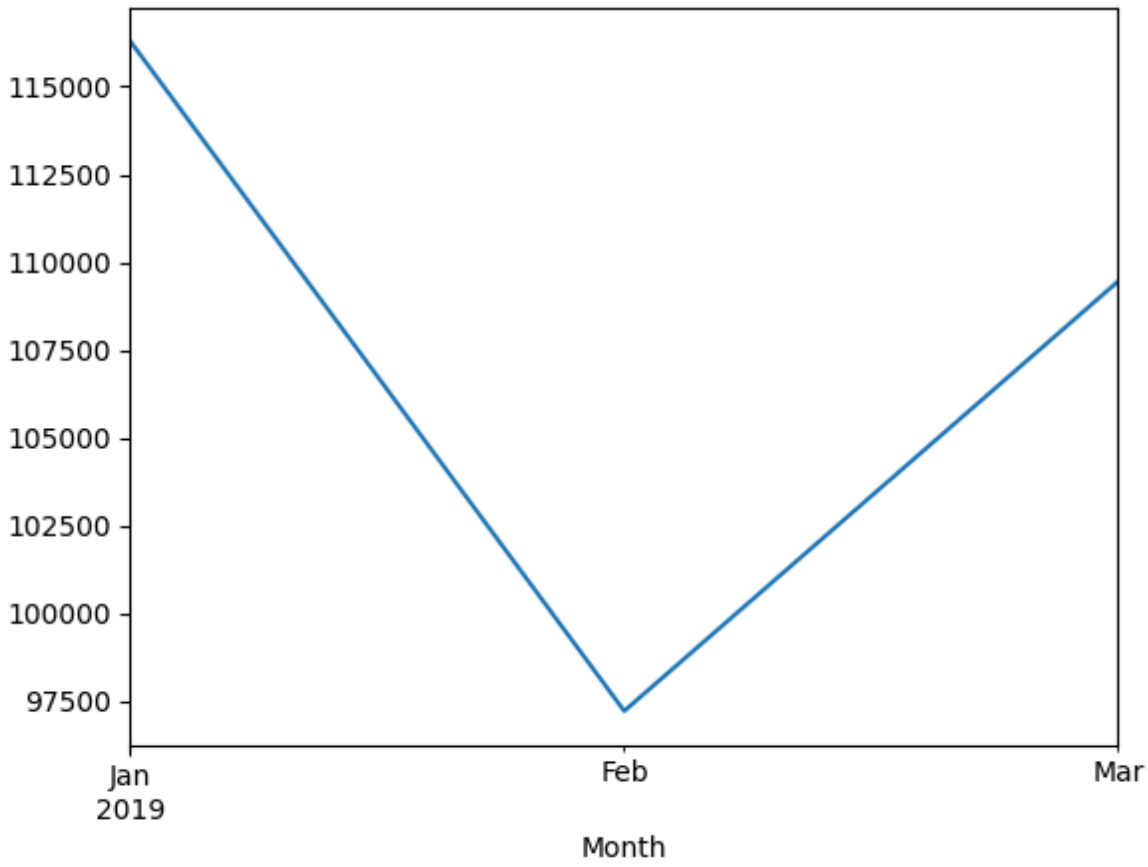
In [54]:
```
Total_sum.plot()
```

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



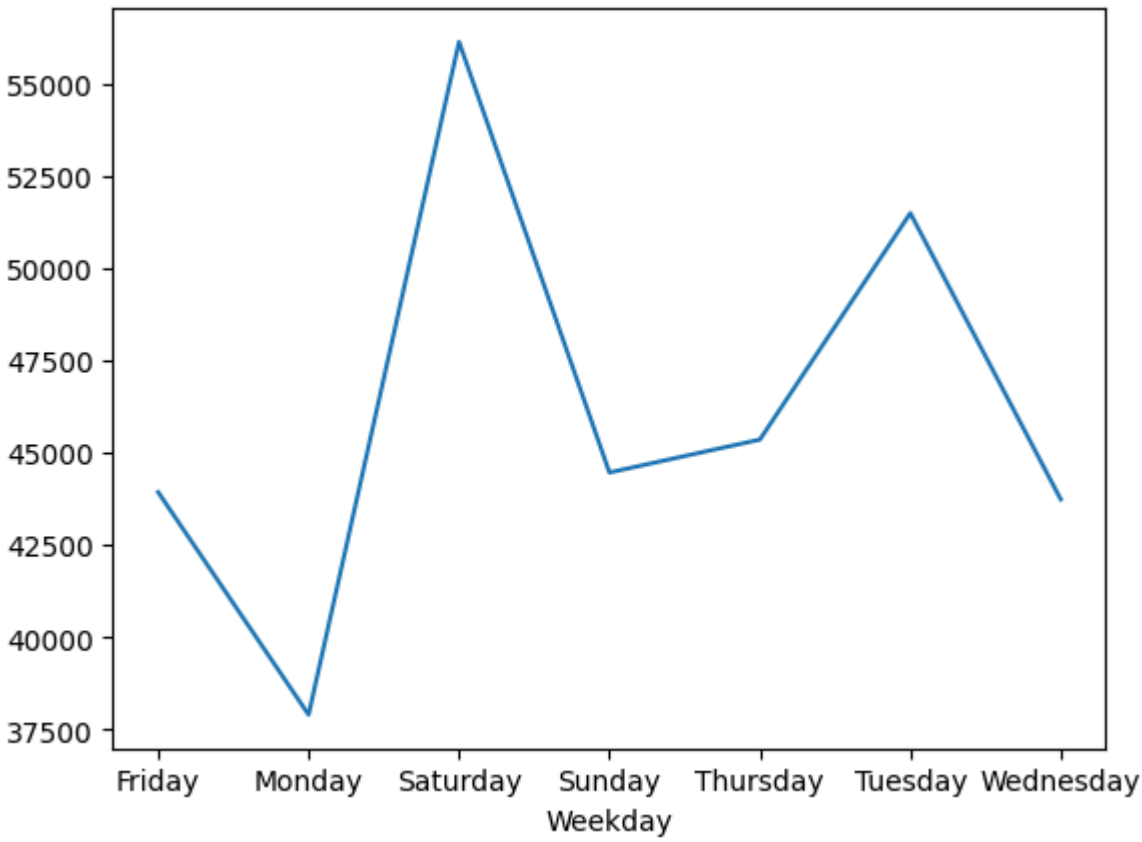
In [55]:
```
Total_monthwise.plot()
```

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



In [56]:
```
Day_wise_salse.plot()
```

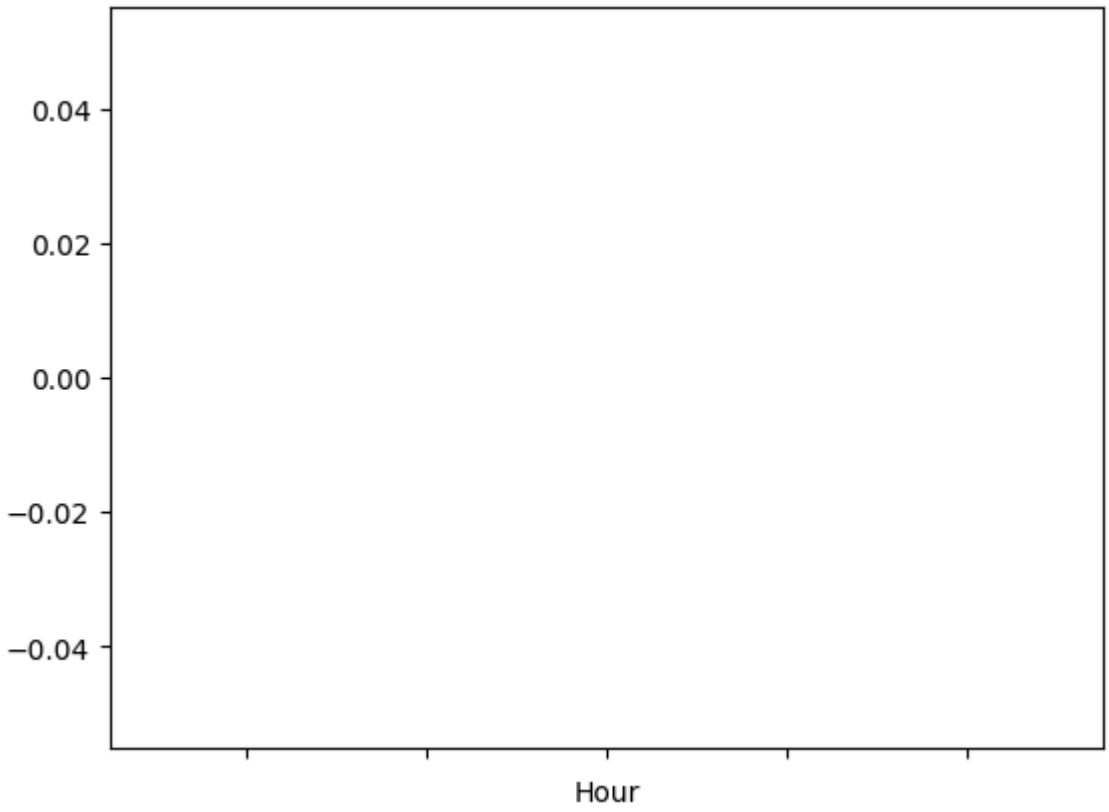
Out[56]: <Axes: xlabel='Weekday'>



In [57]:
```
hourly_sales=df.groupby('Hour')['Total'].sum()
print(hourly_sales)
hourly_sales.plot()
```

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

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



In [ ]: