EDA ON SALES ANALYSIS

```
#  Import Necessary Python Libraries:
print('Import the essential libraries for data manipulation, analysis,
and visualization')
```

```
Import the essential libraries for data manipulation, analysis, and
visualization
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt # visualizing data
%matplotlib inline
import seaborn as sns
```

Load the Dataset

```
df=pd.read_csv(r'C:\Users\Vipin\Downloads\End To End Power BI Project\
Python\E Commerce\supermarket_sales.csv')
df
```

```
      Invoice ID Branch       City Customer type  Gender  \
0    750-67-8428      A     Yangon        Member  Female
1    226-31-3081      C  Naypyitaw        Normal  Female
2    631-41-3108      A     Yangon        Normal    Male
3    123-19-1176      A     Yangon        Member    Male
4    373-73-7910      A     Yangon        Normal    Male
..           ...    ...        ...           ...     ...
995  233-67-5758      C  Naypyitaw        Normal    Male
996  303-96-2227      B   Mandalay        Normal  Female
997  727-02-1313      A     Yangon        Member    Male
998  347-56-2442      A     Yangon        Normal    Male
999  849-09-3807      A     Yangon        Member  Female

               Product line  Unit price  Quantity   Tax 5%      Total
\
0         Health and beauty       74.69         7  26.1415   548.9715

1    Electronic accessories       15.28         5   3.8200    80.2200
```

```
2        Home and lifestyle       46.33         7  16.2155   340.5255

3         Health and beauty       58.22         8  23.2880   489.0480

4         Sports and travel       86.31         7  30.2085   634.3785

..                      ...         ...       ...      ...        ...

995       Health and beauty       40.35         1   2.0175    42.3675

996      Home and lifestyle       97.38        10  48.6900  1022.4900

997      Food and beverages       31.84         1   1.5920    33.4320

998      Home and lifestyle       65.82         1   3.2910    69.1110

999     Fashion accessories       88.34         7  30.9190   649.2990


         Date   Time      Payment    cogs  gross margin percentage  \
0    05-01-19  13:08      Ewallet  522.83                 4.761905
1    08-03-19  10:29         Cash   76.40                 4.761905
2    03-03-19  13:23  Credit card  324.31                 4.761905
3    27-01-19  20:33      Ewallet  465.76                 4.761905
4    08-02-19  10:37      Ewallet  604.17                 4.761905
..        ...    ...          ...     ...                      ...
995  29-01-19  13:46      Ewallet   40.35                 4.761905
996  02-03-19  17:16      Ewallet  973.80                 4.761905
997  09-02-19  13:22         Cash   31.84                 4.761905
998  22-02-19  15:33         Cash   65.82                 4.761905
999  18-02-19  13:28         Cash  618.38                 4.761905

     gross income  Rating
0         26.1415     9.1
1          3.8200     9.6
2         16.2155     7.4
3         23.2880     8.4
4         30.2085     5.3
..            ...     ...
995        2.0175     6.2
996       48.6900     4.4
997        1.5920     7.7
998        3.2910     4.1
999       30.9190     6.6

[1000 rows x 17 columns]
```

Display Basic Information

```
# Display the first few rows of the dataframe
df.head()

    Invoice ID Branch       City Customer type  Gender  \
0  750-67-8428      A     Yangon        Member  Female
1  226-31-3081      C  Naypyitaw        Normal  Female
2  631-41-3108      A     Yangon        Normal    Male
3  123-19-1176      A     Yangon        Member    Male
4  373-73-7910      A     Yangon        Normal    Male

             Product line  Unit price  Quantity   Tax 5%     Total
Date  \
0       Health and beauty       74.69         7  26.1415  548.9715
05-01-19
1  Electronic accessories       15.28         5   3.8200   80.2200
08-03-19
2      Home and lifestyle       46.33         7  16.2155  340.5255
03-03-19
3       Health and beauty       58.22         8  23.2880  489.0480
27-01-19
4       Sports and travel       86.31         7  30.2085  634.3785
08-02-19

    Time      Payment    cogs  gross margin percentage  gross income
Rating
0  13:08      Ewallet  522.83                 4.761905       26.1415
9.1
1  10:29         Cash   76.40                 4.761905        3.8200
9.6
2  13:23  Credit card  324.31                 4.761905       16.2155
7.4
3  20:33      Ewallet  465.76                 4.761905       23.2880
8.4
4  10:37      Ewallet  604.17                 4.761905       30.2085
5.3

# Display the last few rows of the dataframe
df.tail()

      Invoice ID Branch       City Customer type  Gender
Product line  \
995  233-67-5758      C  Naypyitaw        Normal    Male    Health and
beauty
996  303-96-2227      B   Mandalay        Normal  Female   Home and
lifestyle
997  727-02-1313      A     Yangon        Member    Male   Food and
beverages
998  347-56-2442      A     Yangon        Normal    Male   Home and
```

```
lifestyle
999  849-09-3807      A     Yangon        Member  Female  Fashion
accessories

     Unit price  Quantity   Tax 5%      Total      Date   Time
Payment  \
995       40.35         1   2.0175    42.3675  29-01-19  13:46
Ewallet
996       97.38        10  48.6900  1022.4900  02-03-19  17:16
Ewallet
997       31.84         1   1.5920    33.4320  09-02-19  13:22
Cash
998       65.82         1   3.2910    69.1110  22-02-19  15:33
Cash
999       88.34         7  30.9190   649.2990  18-02-19  13:28
Cash

       cogs  gross margin percentage  gross income  Rating
995   40.35                 4.761905        2.0175     6.2
996  973.80                 4.761905       48.6900     4.4
997   31.84                 4.761905        1.5920     7.7
998   65.82                 4.761905        3.2910     4.1
999  618.38                 4.761905       30.9190     6.6
```

```
#checking shape of Iris dataset
df.shape
```

```
(1000, 17)
```

```
#checking size of Iris dataset
df.size
```

```
17000
```

```
#checking what are the variables here
df.columns
```

```
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')
```

```
# Get a concise summary of the dataframe
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

# Basic descriptive statistics
df.describe()

        Unit price     Quantity       Tax 5%        Total        cogs
\
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.00000

mean     55.672130     5.510000    15.379369   322.966749   307.58738

std      26.494628     2.923431    11.708825   245.885335   234.17651

min      10.080000     1.000000     0.508500    10.678500    10.17000

25%      32.875000     3.000000     5.924875   124.422375   118.49750

50%      55.230000     5.000000    12.088000   253.848000   241.76000

75%      77.935000     8.000000    22.445250   471.350250   448.90500

max      99.960000    10.000000    49.650000  1042.650000   993.00000


       gross margin percentage  gross income      Rating
count             1.000000e+03   1000.000000  1000.00000
mean              4.761905e+00     15.379369     6.97270
std               6.131498e-14     11.708825     1.71858
min               4.761905e+00      0.508500     4.00000
25%               4.761905e+00      5.924875     5.50000
50%               4.761905e+00     12.088000     7.00000
```

```
75%               4.761905e+00     22.445250     8.50000
max               4.761905e+00     49.650000    10.00000
```

Data Cleaning and Preprocessing

```
# Check for missing values
df.isnull().sum()
```

```
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 for duplicate values
df.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
995    False
996    False
997    False
998    False
999    False
Length: 1000, dtype: bool
```

```
# Check for duplicate values
df[df.duplicated()]
```

```
Empty DataFrame
Columns: [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]
Index: []

# drop duplicate values if available
df.drop_duplicates(inplace=True)

# check datatypes
df.dtypes

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

# Convert 'Date' column to datetime format
df['Date']=pd.to_datetime(df['Date'])

# Convert 'Time' column to datetime format (hours and minutes)
df['Time']=pd.to_datetime(df['Time'])

# check datatypes
df.dtypes

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                           datetime64[ns]
Payment                                object
cogs                                  float64
gross margin percentage               float64
gross income                          float64
Rating                                float64
dtype: object
```

```
# Extracting year, month,month_name and day from the 'Date' column
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
df['month_name'] = df['Date'].dt.month_name()
```

```
df.head(2)
```

```
    Invoice ID Branch       City Customer type  Gender  \
0  750-67-8428      A     Yangon        Member  Female
1  226-31-3081      C  Naypyitaw        Normal  Female

             Product line  Unit price  Quantity   Tax 5%
Total  ...  \
0       Health and beauty       74.69         7  26.1415
548.9715  ...
1  Electronic accessories       15.28         5   3.8200
80.2200  ...

                 Time  Payment    cogs  gross margin percentage  gross
income  \
0 2024-09-27 13:08:00  Ewallet  522.83                 4.761905
26.1415
1 2024-09-27 10:29:00     Cash   76.40                 4.761905
3.8200

   Rating  Year  Month  Day  month_name
0     9.1  2019      5    1         May
1     9.6  2019      8    3      August

[2 rows x 21 columns]
```

Exploratory Data Analysis (EDA)

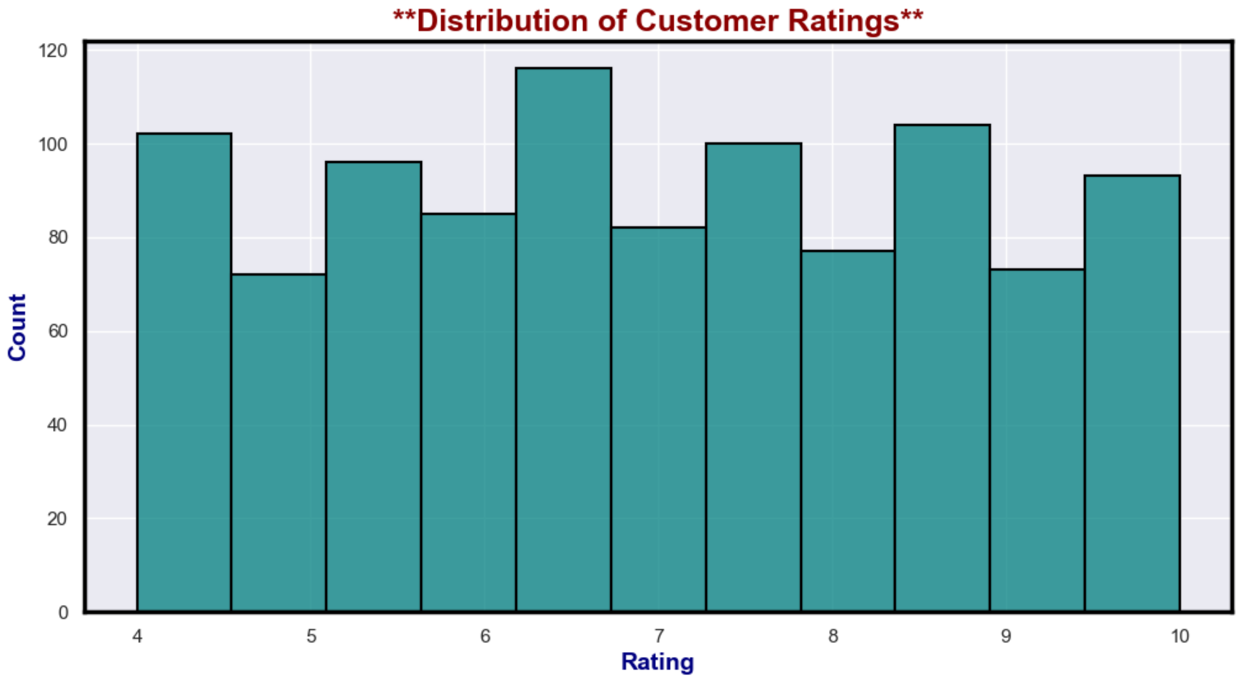
```
df.head(3)
```

```
    Invoice ID Branch       City Customer type  Gender  \
0  750-67-8428      A     Yangon        Member  Female   
1  226-31-3081      C  Naypyitaw        Normal  Female   
2  631-41-3108      A     Yangon        Normal    Male   

             Product line  Unit price  Quantity   Tax 5%
Total  ...  \
0       Health and beauty       74.69         7  26.1415
548.9715  ...   
1  Electronic accessories       15.28         5   3.8200
80.2200  ...   
2      Home and lifestyle       46.33         7  16.2155
340.5255  ...   

                 Time      Payment    cogs  gross margin percentage  \
0 2024-09-27 13:08:00      Ewallet  522.83                 4.761905   
1 2024-09-27 10:29:00         Cash   76.40                 4.761905   
2 2024-09-27 13:23:00  Credit card  324.31                 4.761905   

   gross income  Rating  Year  Month  Day  month_name  
0       26.1415     9.1  2019      5    1         May  
1        3.8200     9.6  2019      8    3      August  
2       16.2155     7.4  2019      3    3       March  

[3 rows x 21 columns]
```

Distribution of Rating

```
plt.figure(figsize=(12, 6))


sns.histplot(data=df, x='Rating', kde=False, color='teal',
edgecolor='black', linewidth=1.5)


plt.title('**Distribution of Customer Ratings**', fontsize=18,
weight='bold', color='darkred')
```

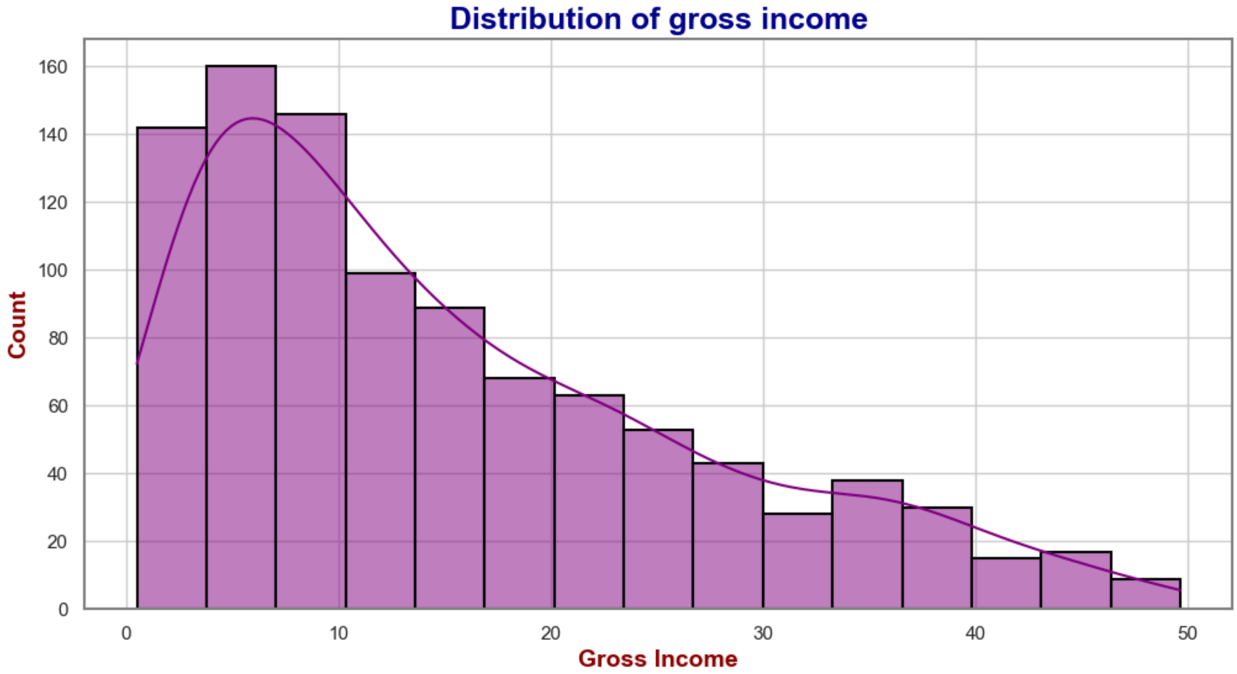
```
plt.xlabel('Rating', fontsize=14, weight='bold', color='navy')
plt.ylabel('Count', fontsize=14, weight='bold', color='navy')


plt.gca().spines['top'].set_color('black')
plt.gca().spines['right'].set_color('black')
plt.gca().spines['left'].set_color('black')
plt.gca().spines['bottom'].set_color('black')
plt.gca().spines['top'].set_linewidth(2.5)
plt.gca().spines['right'].set_linewidth(2.5)
plt.gca().spines['left'].set_linewidth(2.5)
plt.gca().spines['bottom'].set_linewidth(2.5)


plt.show()
```



Distribution of Gross Income

```
plt.figure(figsize=(12, 6))

sns.histplot(data=df, x='gross income', kde=True, color='purple',
edgecolor='black', linewidth=1.5)
```

```
plt.title('Distribution of gross income', fontsize=18,
fontweight='bold', color='darkblue')


plt.xlabel('Gross Income', fontsize=14, fontweight='bold',
color='darkred')
plt.ylabel('Count', fontsize=14, fontweight='bold', color='darkred')


plt.gca().spines['top'].set_visible(True)
plt.gca().spines['top'].set_color('gray')
plt.gca().spines['top'].set_linewidth(1.5)

plt.gca().spines['right'].set_visible(True)
plt.gca().spines['right'].set_color('gray')
plt.gca().spines['right'].set_linewidth(1.5)

plt.gca().spines['left'].set_visible(True)
plt.gca().spines['left'].set_color('gray')
plt.gca().spines['left'].set_linewidth(1.5)

plt.gca().spines['bottom'].set_visible(True)
plt.gca().spines['bottom'].set_color('gray')
plt.gca().spines['bottom'].set_linewidth(1.5)


plt.show()
```



Correlation Heatmap

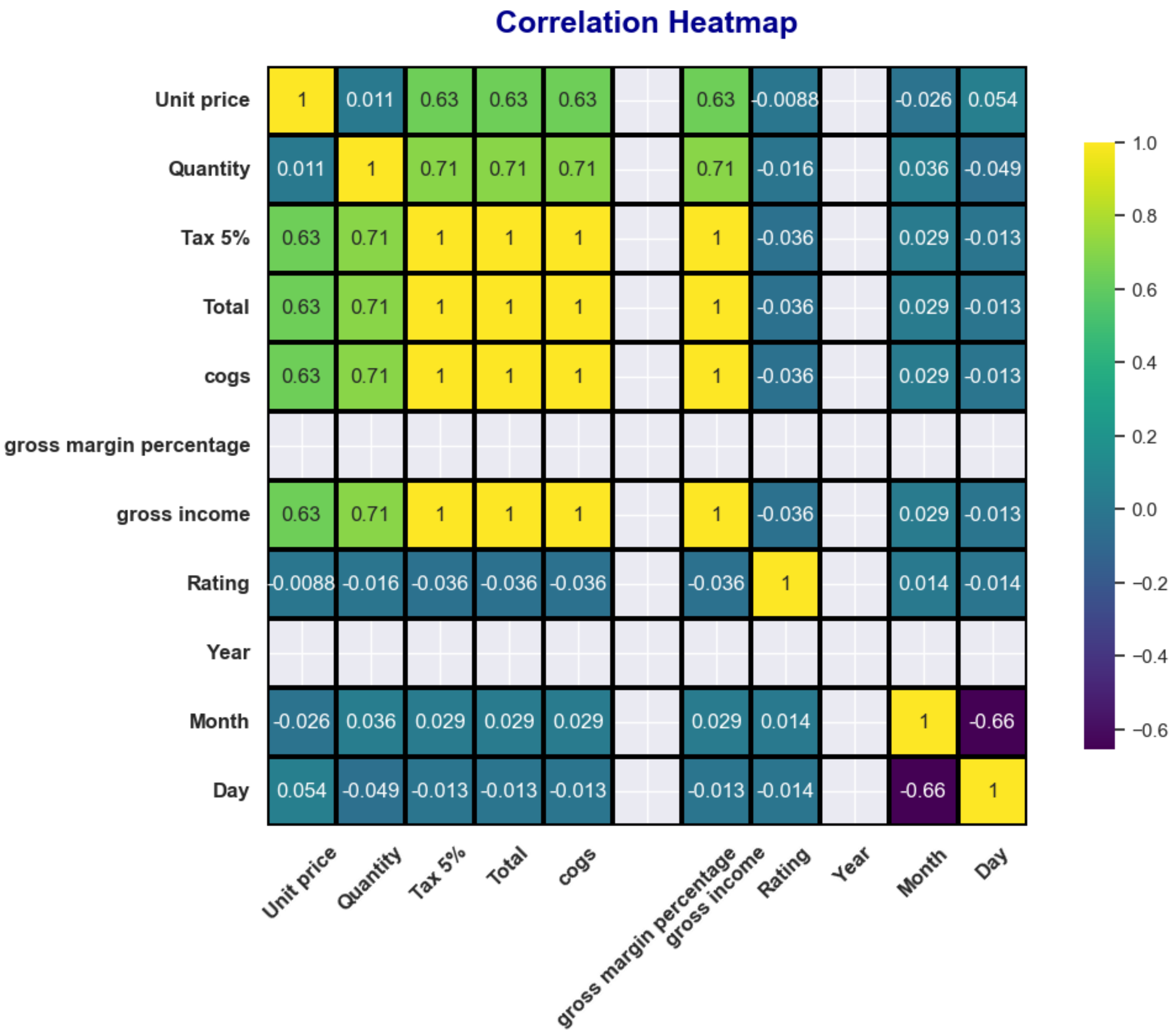
```
df.corr()
```

```
C:\Users\Vipin\AppData\Local\Temp\ipykernel_15636\1134722465.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only
valid columns or specify the value of numeric_only to silence this
warning.
  df.corr()

                         Unit price  Quantity    Tax 5%     Total
cogs  \
Unit price                 1.000000  0.010778  0.633962  0.633962
0.633962
Quantity                   0.010778  1.000000  0.705510  0.705510
0.705510
Tax 5%                     0.633962  0.705510  1.000000  1.000000
1.000000
Total                      0.633962  0.705510  1.000000  1.000000
1.000000
cogs                       0.633962  0.705510  1.000000  1.000000
1.000000
gross margin percentage         NaN       NaN       NaN       NaN
NaN
gross income               0.633962  0.705510  1.000000  1.000000
1.000000
Rating                    -0.008778 -0.015815 -0.036442 -0.036442 -
0.036442
Year                            NaN       NaN       NaN       NaN
NaN
Month                     -0.026155  0.036188  0.028576  0.028576
0.028576
Day                        0.053549 -0.048506 -0.012581 -0.012581 -
0.012581

                         gross margin percentage  gross income
Rating  \
Unit price                                   NaN      0.633962 -
0.008778
Quantity                                     NaN      0.705510 -
0.015815
Tax 5%                                       NaN      1.000000 -
0.036442
Total                                        NaN      1.000000 -
0.036442
cogs                                         NaN      1.000000 -
0.036442
gross margin percentage                      NaN           NaN
NaN
```

```
gross income                                 NaN      1.000000 -
0.036442
Rating                                       NaN     -0.036442
1.000000
Year                                         NaN           NaN
NaN
Month                                        NaN      0.028576
0.014373
Day                                          NaN     -0.012581 -
0.013752

                         Year     Month       Day
Unit price                NaN -0.026155  0.053549
Quantity                  NaN  0.036188 -0.048506
Tax 5%                    NaN  0.028576 -0.012581
Total                     NaN  0.028576 -0.012581
cogs                      NaN  0.028576 -0.012581
gross margin percentage   NaN       NaN       NaN
gross income              NaN  0.028576 -0.012581
Rating                    NaN  0.014373 -0.013752
Year                      NaN       NaN       NaN
Month                     NaN  1.000000 -0.656430
Day                       NaN -0.656430  1.000000
```

```
plt.figure(figsize=(12, 8))


sns.heatmap(df.corr(), annot=True, cmap='viridis', linewidths=1.5,
linecolor='black', cbar=True,
            cbar_kws={"shrink": .8, "orientation": "vertical"},
square=True)


plt.title('Correlation Heatmap', fontsize=18, fontweight='bold',
color='darkblue', pad=20)


plt.xticks(fontsize=12, rotation=45, fontweight='bold')
plt.yticks(fontsize=12, rotation=0, fontweight='bold')


plt.show()
```

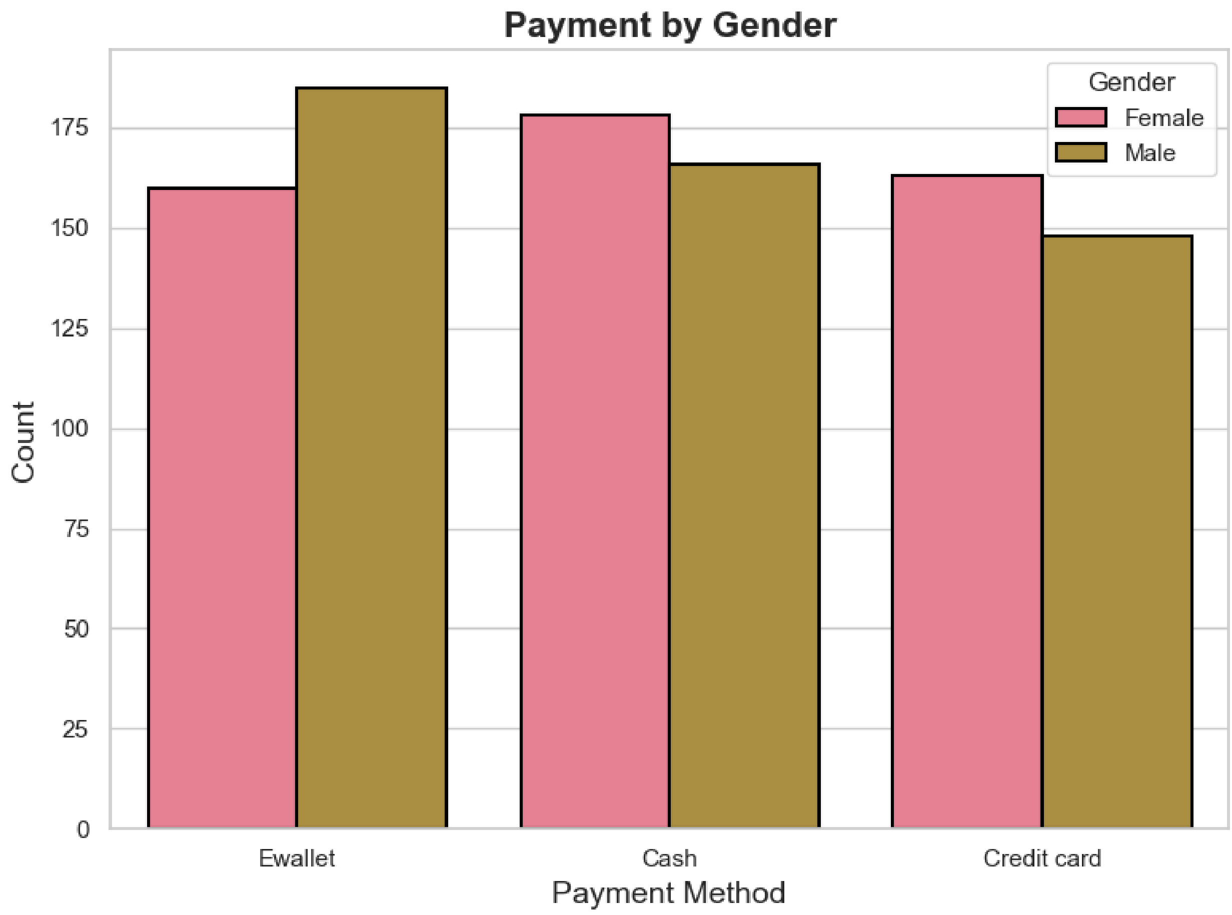
```
C:\Users\Vipin\AppData\Local\Temp\ipykernel_15636\881416650.py:4:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only
valid columns or specify the value of numeric_only to silence this
warning.
  sns.heatmap(df.corr(), annot=True, cmap='viridis', linewidths=1.5,
linecolor='black', cbar=True,
```



Payment by Gender

```
plt.figure(figsize=(8, 6))
sns.set_palette("husl")
ax = sns.countplot(x='Payment', data=df, hue='Gender',
edgecolor='black')


for patch in ax.patches:
    patch.set_edgecolor('black')
    patch.set_linewidth(1.5)
```

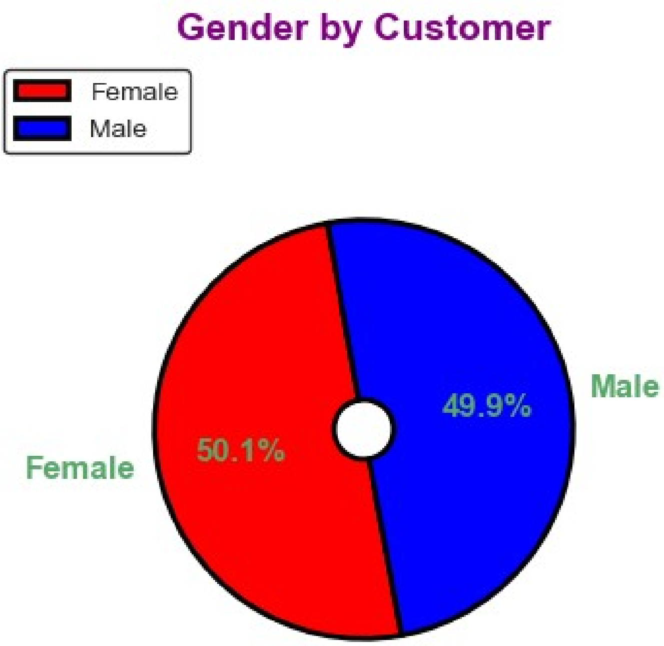
```
plt.title('Payment by Gender', fontsize=16, fontweight='bold')
plt.xlabel('Payment Method', fontsize=14)
plt.ylabel('Count', fontsize=14)


plt.legend(title='Gender', loc='upper right')


plt.tight_layout()
plt.show()
```



Gender by Customer

```
df['Gender'].value_counts()
```

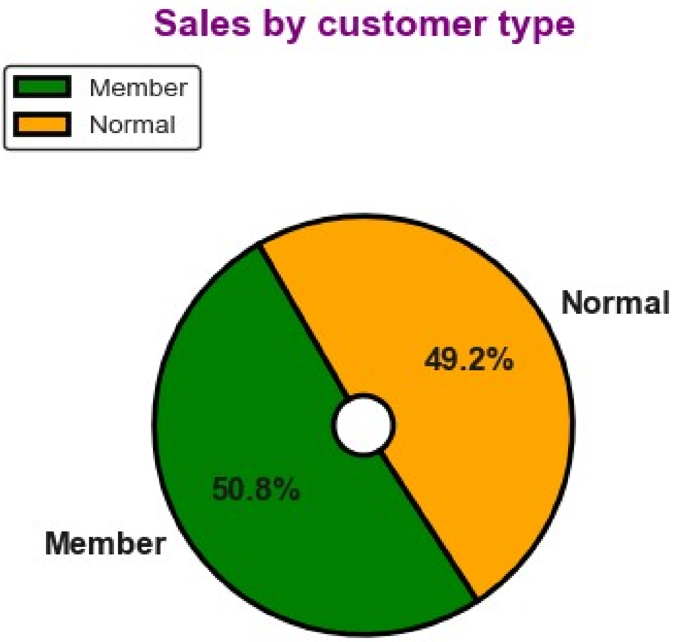
```
Female    501
Male      499
Name: Gender, dtype: int64


textprops = {'fontsize': 12, 'color': 'g', 'weight': 'bold'}


plt.figure(figsize=(6, 5))


a = df['Gender'].value_counts()

wedgeprops = {'width': 0.6, 'edgecolor': 'black', 'linewidth': 2}


plt.pie(a, labels=a.index,
        colors=['red', 'blue'],startangle=100,
        autopct='%0.1f%%',
        radius=0.7,
        wedgeprops=wedgeprops,
        textprops=textprops)


plt.title('Gender by Customer', fontsize=14, weight='bold',
color='purple')


plt.legend(loc='upper left',edgecolor='black', fontsize=10)


plt.show()
```



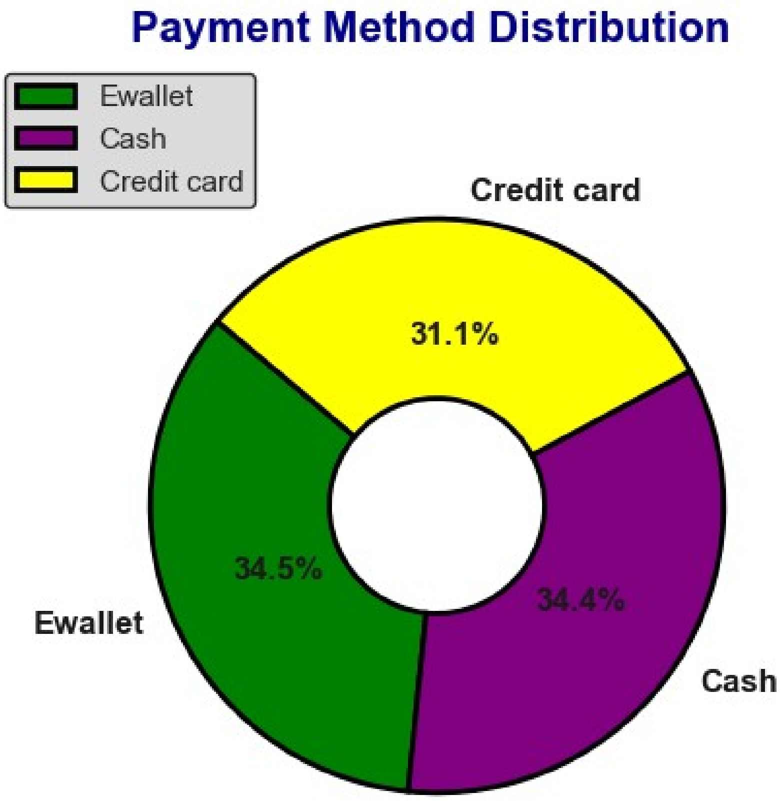
Sales by Customer Type

```
textprops = {'fontsize': 12, 'color': 'k', 'weight': 'bold'}


plt.figure(figsize=(6, 5))


sales_Customer_type=df.groupby('Customer type')['Total'].sum()

wedgeprops = {'width': 0.6, 'edgecolor': 'black', 'linewidth': 2}


plt.pie(sales_Customer_type, labels=sales_Customer_type.index,
        colors=['green', 'orange'], startangle=120,
        autopct='%0.1f%%',
        radius=0.7,
        wedgeprops=wedgeprops,
        textprops=textprops)
```

```
plt.title('Sales by customer type', fontsize=14, weight='bold',
color='purple')


plt.legend(loc='upper left',edgecolor='black', fontsize=10)


plt.show()
```

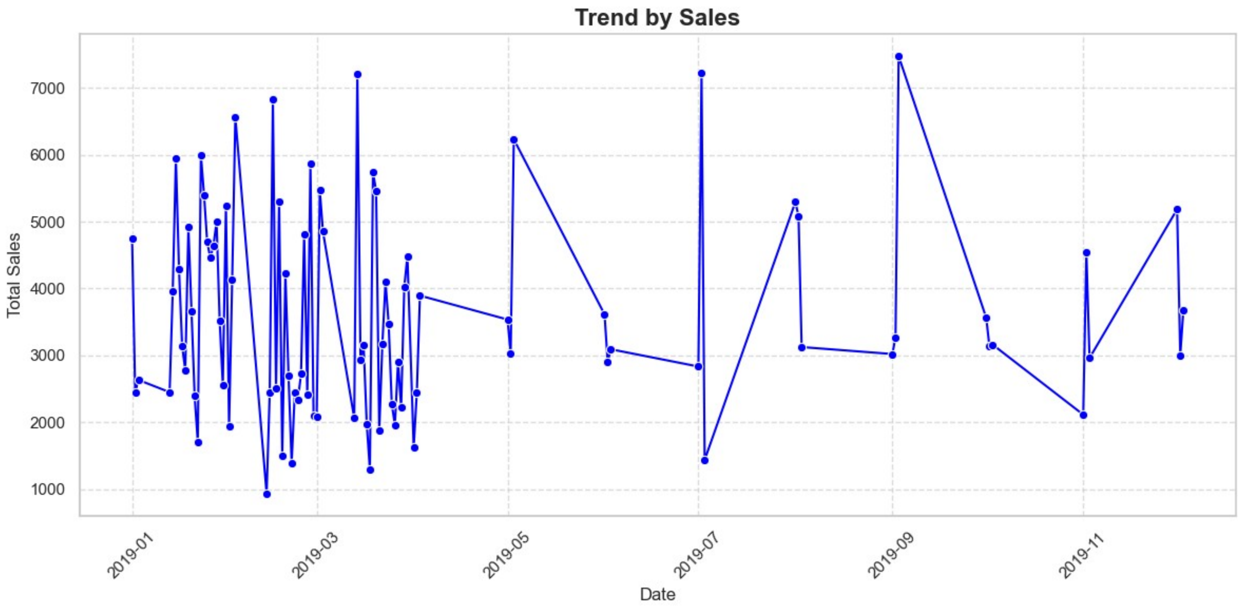


```
plt.figure(figsize=(8, 6))


textprops = {'fontsize': 12, 'color': 'k', 'weight': 'bold'}
titleprops = {'fontsize': 16, 'weight': 'bold', 'color': 'darkblue'}


a = df['Payment'].value_counts()


plt.pie(a,
        labels=a.index,
        colors=['green', 'purple', 'yellow', 'orange', 'blue'],
        autopct='%0.1f%%',
        textprops=textprops,
        radius=0.8,
        wedgeprops={'width': 0.5, 'edgecolor': 'black', 'linewidth':
2},
        startangle=140)


plt.title('Payment Method Distribution ', **titleprops)


plt.legend(loc='upper left', frameon=True, facecolor='lightgrey',
edgecolor='black')


plt.show()
```



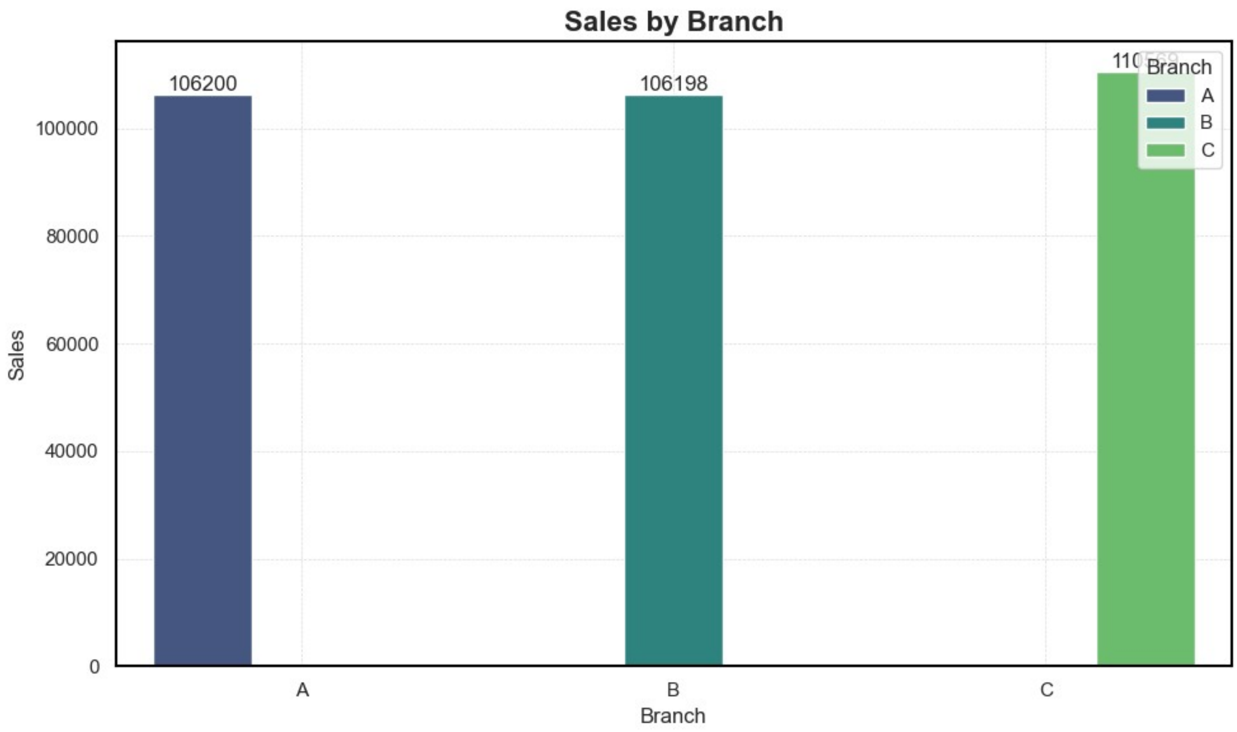
Trend by Sales

```python
a = df.groupby(['Date'], as_index=False)['Total'].sum()


a = a.sort_values('Date')


sns.set(style="whitegrid")


plt.figure(figsize=(12, 6))
sns.lineplot(x='Date', y='Total', data=a, marker='o', color='blue')


plt.gca().spines['top'].set_visible(True)
plt.gca().spines['right'].set_visible(True)
plt.gca().spines['left'].set_linewidth(1.5)
plt.gca().spines['bottom'].set_linewidth(1.5)


plt.xticks(rotation=45)


plt.title("Trend by Sales", fontsize=16, fontweight='bold')
plt.xlabel("Date", fontsize=12)
plt.ylabel("Total Sales", fontsize=12)


plt.grid(True, linestyle='--', alpha=0.7)


plt.tight_layout()
plt.show()
```



Sales by Branch

```
sales_branch = df.groupby(['Branch'], as_index=False)['Total'].sum()

plt.figure(figsize=(10, 6))
ax = sns.barplot(x='Branch', y='Total', hue='Branch', 
data=sales_branch, palette='viridis')


plt.ylabel('Sales', fontsize=12)
plt.title('Sales by Branch', fontsize=16, fontweight='bold')
plt.grid(True, linestyle='--', linewidth=0.5, alpha=0.7)


for spine in ax.spines.values():
    spine.set_edgecolor('black')
    spine.set_linewidth(1.5)


for bar in ax.containers:
    ax.bar_label(bar)


plt.tight_layout()
plt.show()
```



Sales by City

```
df['City'].unique()
```

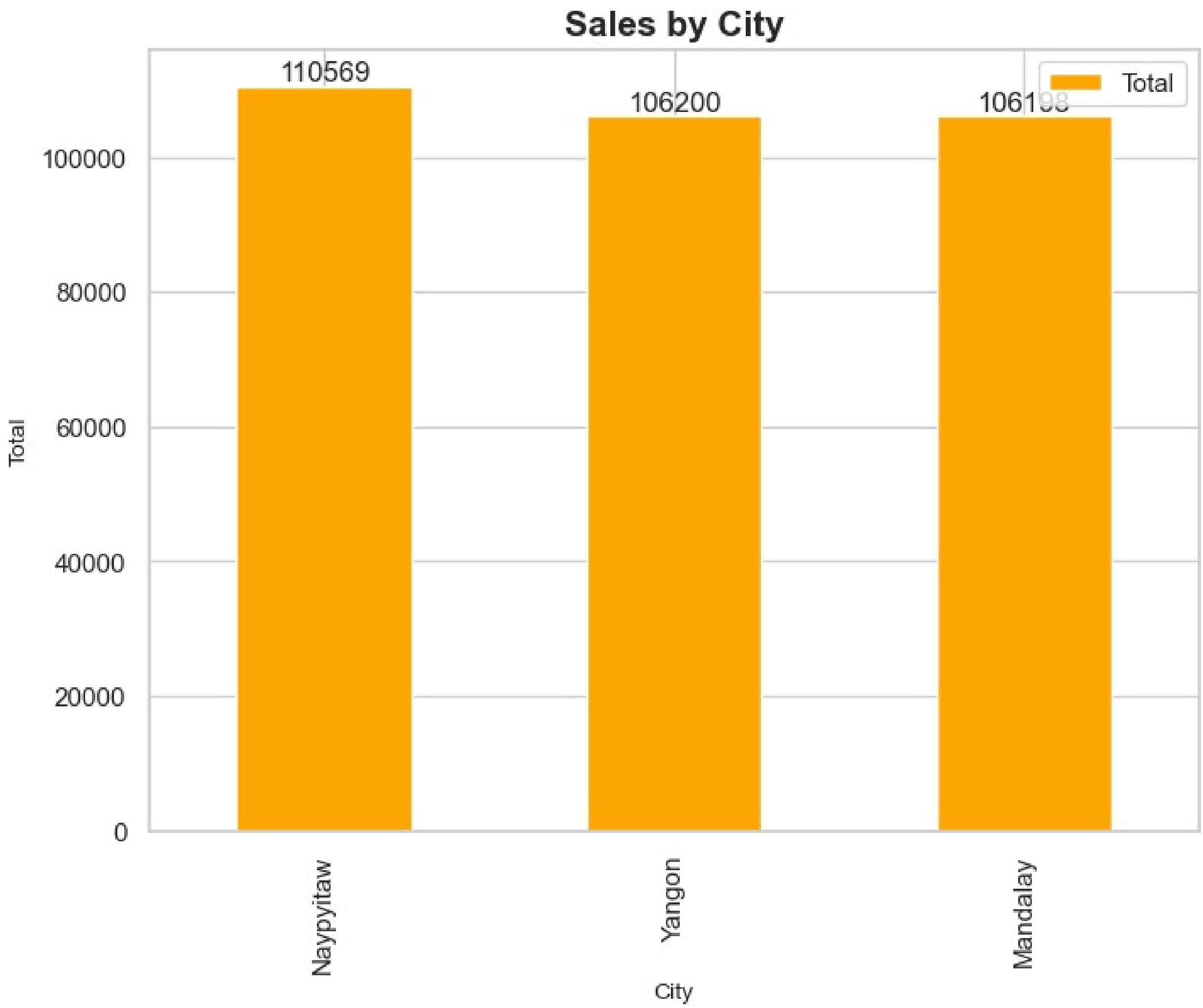
```
array(['Yangon', 'Naypyitaw', 'Mandalay'], dtype=object)
```

```
city_by_total=df.groupby(['City'], as_index=False)
['Total'].sum().sort_values(by='Total', ascending=False)
city_by_total
```

```
        City        Total
1  Naypyitaw  110568.7065
2     Yangon  106200.3705
0   Mandalay  106197.6720
```

```
city_by_total = df.groupby(['City'], as_index=False)
['Total'].sum().sort_values(by='Total', ascending=False)


colors = ['orange']

ax = city_by_total.plot(kind='bar', x='City', color=colors,
figsize=(8, 6))
```

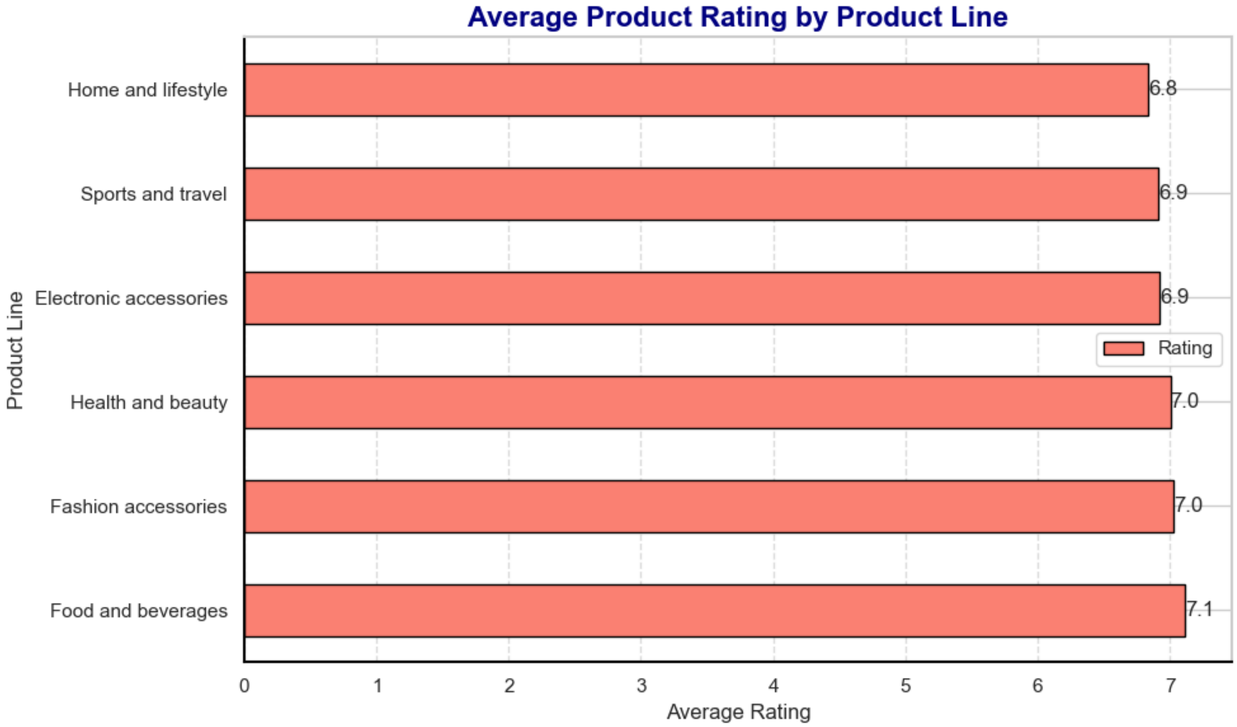
```python
for bars in ax.containers:
    ax.bar_label(bars)

ax.set_xlabel('City',fontsize=10)
ax.set_ylabel('Total',fontsize=10)
ax.set_title('Sales by City',fontsize=15,fontweight='bold')


plt.show()
```



Average Product Rating by Product

```python
avg_product_rating = df.groupby(['Product line'], as_index=False)
```

```
['Rating'].mean().sort_values(by='Rating', ascending=False)

colors = [ 'salmon', 'violet', 'peachpuff', 'lightcoral']  # Added
more colors

ax = avg_product_rating.plot(kind='barh', x='Product line',
color=colors, figsize=(10, 6), edgecolor='black')


for bars in ax.containers:
    ax.bar_label(bars, fmt='%.1f')

ax.set_xlabel('Average Rating', fontsize=12)
ax.set_ylabel('Product Line', fontsize=12)
ax.set_title('Average Product Rating by Product Line', fontsize=16,
fontweight='bold', color='navy')


ax.grid(axis='x', linestyle='--', alpha=0.7)


plt.gca().spines['top'].set_linewidth(1.5)
plt.gca().spines['right'].set_linewidth(1.5)
plt.gca().spines['left'].set_linewidth(1.5)
plt.gca().spines['bottom'].set_linewidth(1.5)


plt.gca().spines['left'].set_color('black')
plt.gca().spines['bottom'].set_color('black')

plt.tight_layout()


plt.show()
```

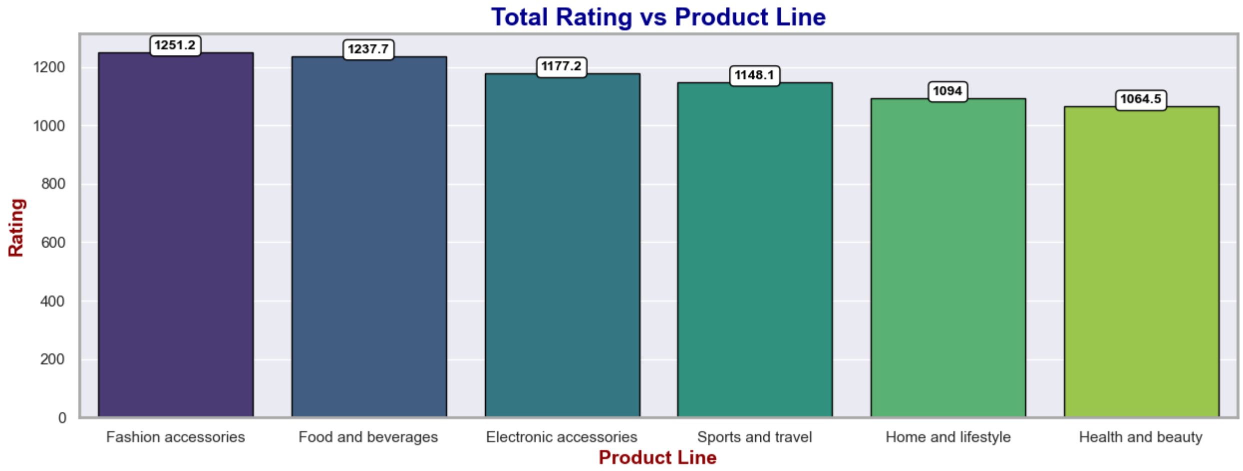


Total Rating vs Product

```
productline_by_rating = df.groupby(['Product line'], as_index=False)
['Rating'].sum().sort_values(by='Rating', ascending=False)
productline_by_rating
```

```
             Product line  Rating
1     Fashion accessories  1251.2
2      Food and beverages  1237.7
0  Electronic accessories  1177.2
5       Sports and travel  1148.1
4      Home and lifestyle  1094.0
3       Health and beauty  1064.5
```

```
sns.set(rc={'figure.figsize': (15, 5)})


a = df.groupby(['Product line'], as_index=False)
['Rating'].sum().sort_values(by='Rating', ascending=False)


ax = sns.barplot(x='Product line', y='Rating', data=a,
palette='viridis', edgecolor='black')


for i in ax.containers:
    ax.bar_label(i, label_type='edge', fontsize=10, color='black',
weight='bold', bbox=dict(facecolor='white', edgecolor='black',
boxstyle='round,pad=0.3'))


plt.title('Total Rating vs Product Line', fontsize=18,
fontweight='bold', color='darkblue')
plt.xlabel('Product Line', fontsize=14, fontweight='bold',
color='darkred')
plt.ylabel('Rating', fontsize=14, fontweight='bold', color='darkred')


plt.gca().spines['top'].set_linewidth(2)
plt.gca().spines['bottom'].set_linewidth(2)
plt.gca().spines['left'].set_linewidth(2)
plt.gca().spines['right'].set_linewidth(2)
plt.gca().spines['top'].set_color('darkgray')
plt.gca().spines['bottom'].set_color('darkgray')
plt.gca().spines['left'].set_color('darkgray')
plt.gca().spines['right'].set_color('darkgray')


plt.show()
```



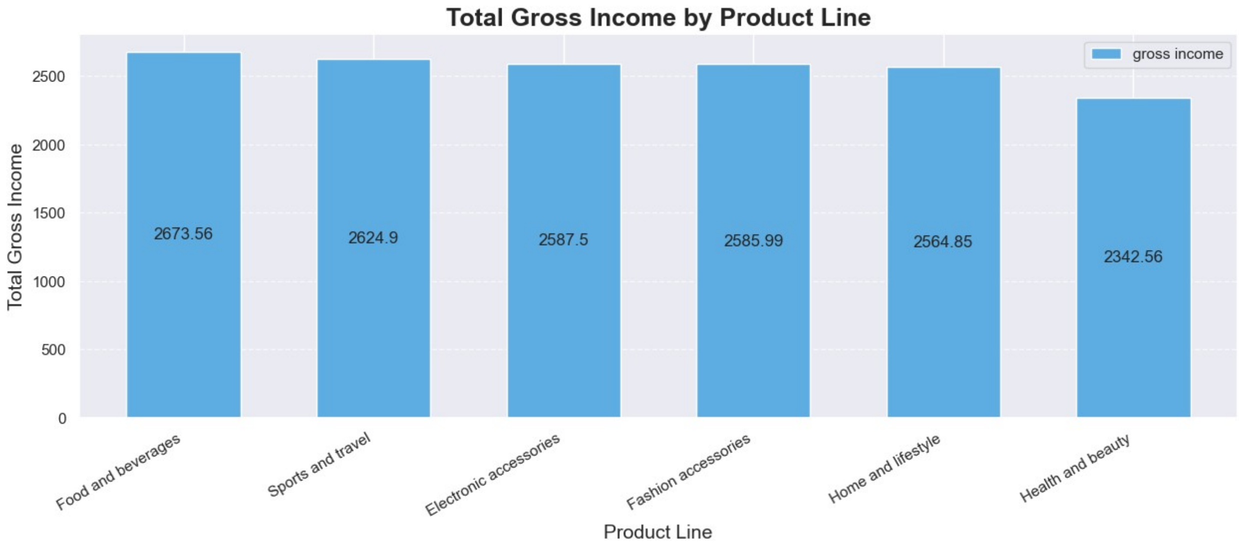
Total Gross Income by Product

```
gross_income_by_products= df.groupby(['Product line'], as_index=False)
['gross income'].sum().sort_values(by='gross income', ascending=False)
gross_income_by_products
```

```
             Product line  gross income
2      Food and beverages     2673.5640
5       Sports and travel     2624.8965
0  Electronic accessories     2587.5015
1     Fashion accessories     2585.9950
4      Home and lifestyle     2564.8530
3       Health and beauty     2342.5590
```

```
a = df.groupby(['Product line'], as_index=False)['gross
income'].sum().sort_values(by='gross income', ascending=False)


plt.figure(figsize=(12, 7))

ax = a.plot(kind='bar', color=['#5DADE2', '#48C9B0', '#F4D03F',
'#E74C3C', '#AF7AC5', '#F39C12'],
            x='Product line', width=0.6)

for container in ax.containers:
    ax.bar_label(container, label_type='center', fontsize=12)

plt.title('Total Gross Income by Product Line', fontsize=18,
fontweight='bold')
plt.xlabel('Product Line', fontsize=14)
plt.ylabel('Total Gross Income', fontsize=14)


plt.xticks(rotation=30, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)


plt.show()
```

```
<Figure size 1200x700 with 0 Axes>
```



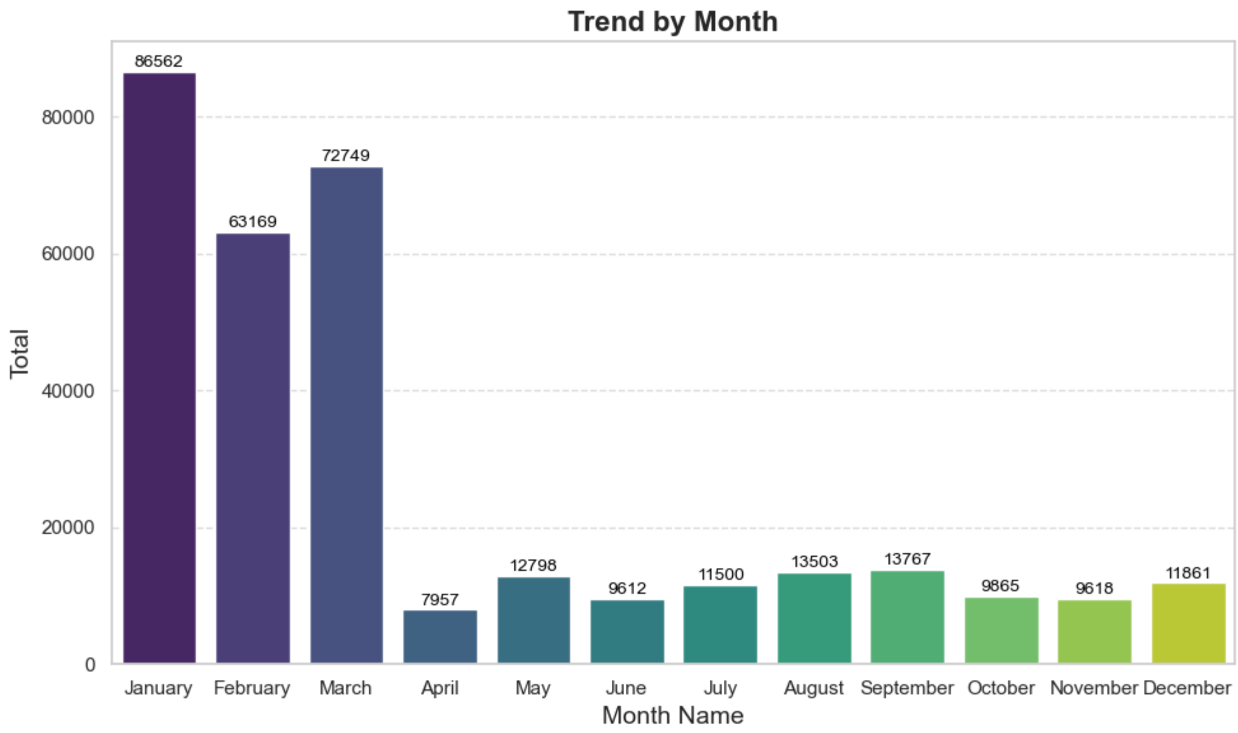
Trend by Month

```
month_order = ['January', 'February', 'March', 'April', 'May', 'June',

               'July', 'August', 'September', 'October', 'November',
'December']


df['month_name'] = pd.Categorical(df['month_name'],
categories=month_order, ordered=True)


a = df.groupby(['month_name'], as_index=False)
['Total'].sum().sort_values(by='month_name')


a['Total'] = a['Total'].round(0).astype(int)


print(a)
```

```
  month_name  Total
0    January  86563
1   February  63170
2      March  72749
3      April   7958
4        May  12799
5       June   9612
6       July  11501
```

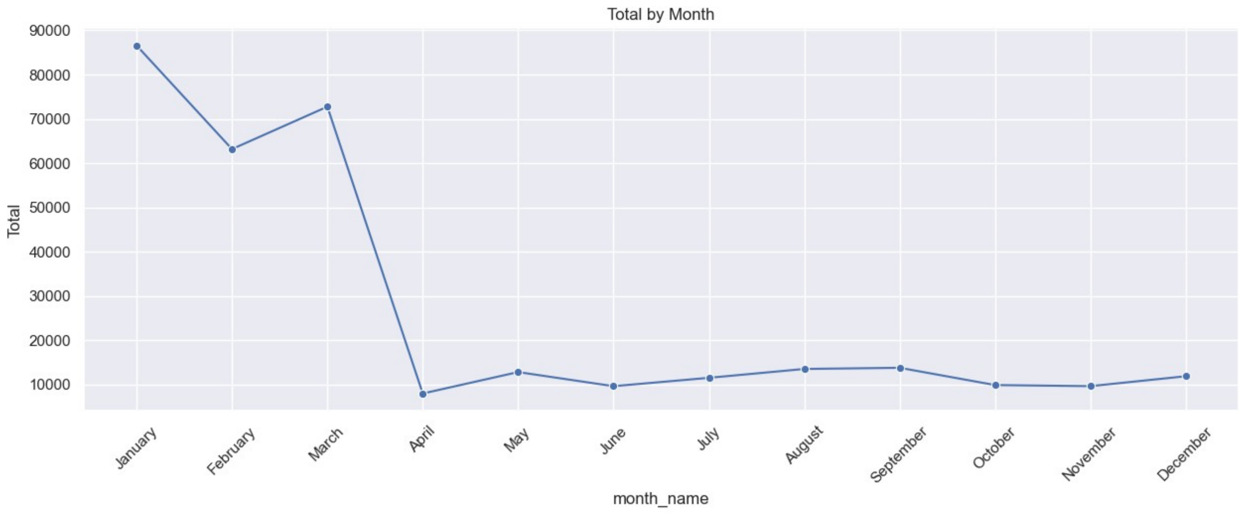
```
7      August  13504
8   September  13767
9     October   9865
10   November   9618
11   December  11861
```

```
a = df.groupby(['month_name'], as_index=False)
['Total'].sum().sort_values(by='Total', ascending=False)

plt.figure(figsize=(10, 6))
bar_plot = sns.barplot(x='month_name', y='Total', data=a,
palette='viridis')

plt.title('Trend by Month', fontsize=16, fontweight='bold')
plt.xlabel('Month Name', fontsize=14)
plt.ylabel('Total', fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.7)

for patch in bar_plot.patches:
    bar_plot.annotate(f'{int(patch.get_height())}',  # Change here to
use int
                      (patch.get_x() + patch.get_width() / 2.,
                       patch.get_height()),
                      ha='center', va='center',
                      fontsize=10, color='black',
                      xytext=(0, 5),
                      textcoords='offset points')

plt.tight_layout()
plt.show()
```



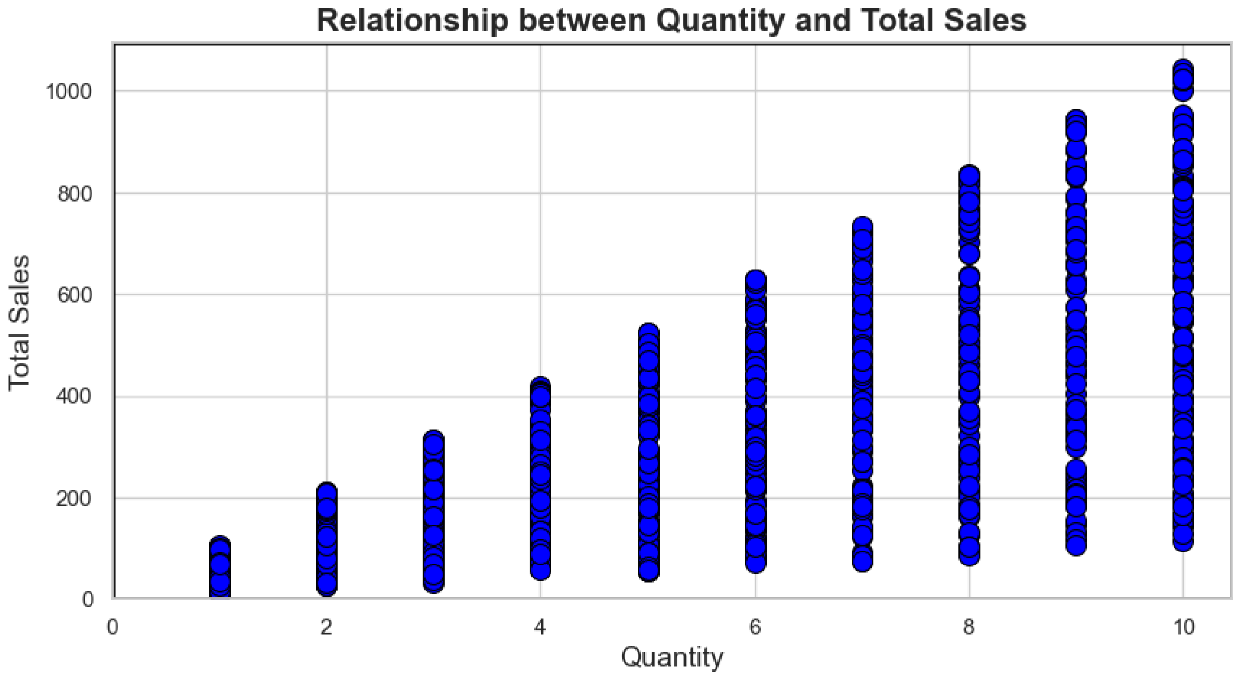
Trend by Month Using Line Graph

```
a = df.groupby(['month_name'], as_index=False)['Total'].sum()


month_order = ["January", "February", "March", "April", "May", "June",
               "July", "August", "September", "October", "November",
"December"]


a['month_name'] = pd.Categorical(a['month_name'],
categories=month_order, ordered=True)
a = a.sort_values('month_name')


sns.lineplot(x='month_name', y='Total', data=a, marker='o')
plt.xticks(rotation=45)
plt.title("Trend by  by Month")
plt.show()
```



Relationship between Quantity and Sales

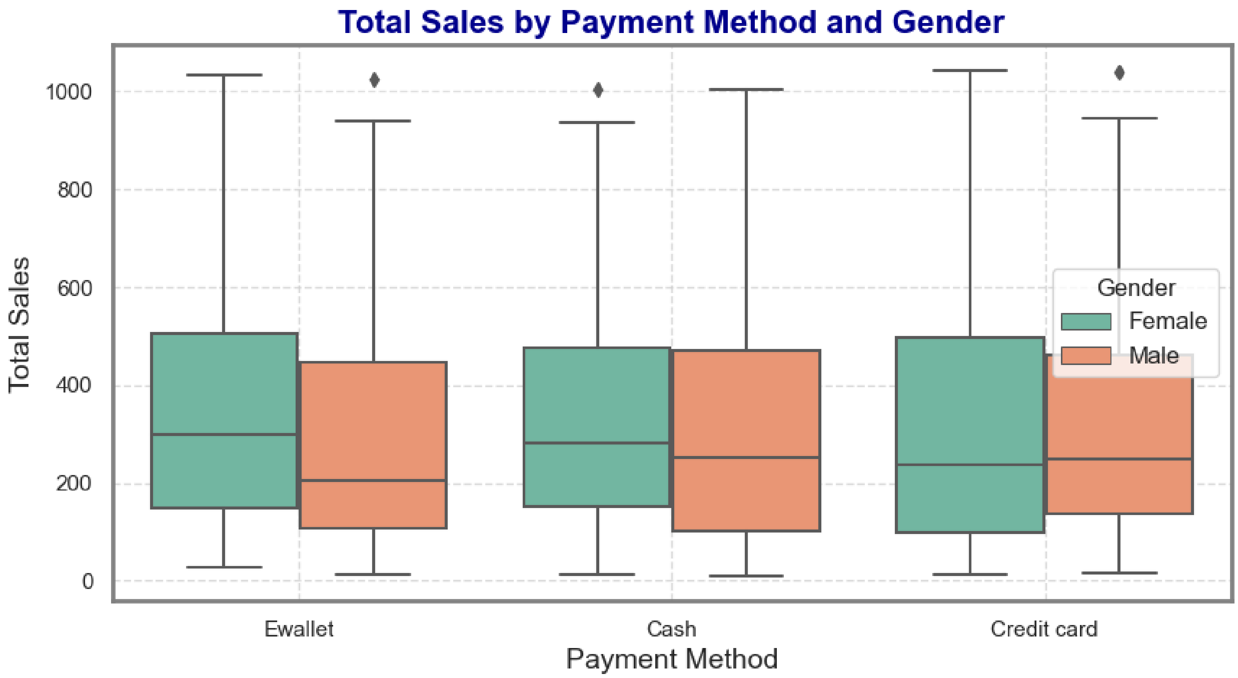
```
plt.figure(figsize=(10, 5))


sns.scatterplot(data=df, x='Quantity', y='Total', color='blue',
edgecolor='black', s=100)


plt.title('Relationship between Quantity and Total Sales',
fontsize=16, fontweight='bold')


plt.xlabel('Quantity', fontsize=14)
plt.ylabel('Total Sales', fontsize=14)
plt.xlim(left=0)
plt.ylim(bottom=0)


plt.gca().add_patch(plt.Rectangle((0, 0), plt.xlim()[1], plt.ylim()
[1], fill=False, edgecolor='black', linewidth=2))


plt.show()
```



Sales by Payment Method and Gender

```
plt.figure(figsize=(10, 5))


sns.boxplot(data=df, x='Payment', y='Total', hue='Gender',
palette='Set2')


plt.title('Total Sales by Payment Method and Gender', fontsize=16,
fontweight='bold', color='darkblue')
plt.xlabel('Payment Method', fontsize=14)
plt.ylabel('Total Sales', fontsize=14)


plt.gca().spines['top'].set_linewidth(2)
plt.gca().spines['bottom'].set_linewidth(2)
plt.gca().spines['right'].set_linewidth(2)
plt.gca().spines['left'].set_linewidth(2)


plt.gca().spines['top'].set_color('gray')
plt.gca().spines['bottom'].set_color('gray')
plt.gca().spines['right'].set_color('gray')
```

```python
plt.gca().spines['left'].set_color('gray')


plt.legend(title='Gender', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```


Total Sales by Payment Method and Gender
Total Sales
Ewallet
Cash
Credit card
Payment Method
Gender
🟩 Female
🟧 Male