Load Libraries

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

Load Data

In [ ]:
```
# load csv file into pandas data frame
df = pd.read_csv('supermarket_sales.csv')
```

Basic Data Exploration

In [ ]:
```
# check first few rows to understand its structure
display(df.head())
print(df.info())
```

	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	26.1415
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085

```
<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
None
```

Summary Statistics

In [ ]:
```python
numerical_columns = []
# Get numerical columns mean
for col in df.columns:
    try:
        print(col,'mean: ',round(df[col].mean(),2))
        numerical_columns.append(col)
    except:
        pass
```

```
Unit price mean:  55.67
Quantity mean:  5.51
Tax 5% mean:  15.38
Total mean:  322.97
cogs mean:  307.59
gross margin percentage mean:  4.76
gross income mean:  15.38
Rating mean:  6.97
```

In [ ]:
```python
# Get numerical columns median
for col in numerical_columns:
    print(col,'median: ',round(df[col].median(),2))
```

```
Unit price median:  55.23
Quantity median:  5.0
Tax 5% median:  12.09
Total median:  253.85
cogs median:  241.76
gross margin percentage median:  4.76
gross income median:  12.09
Rating median:  7.0
```

In [ ]:
```python
# Get standard deviation of numerical columns
for col in numerical_columns:
    print(col,'Std: ',round(df[col].std(),4))
```

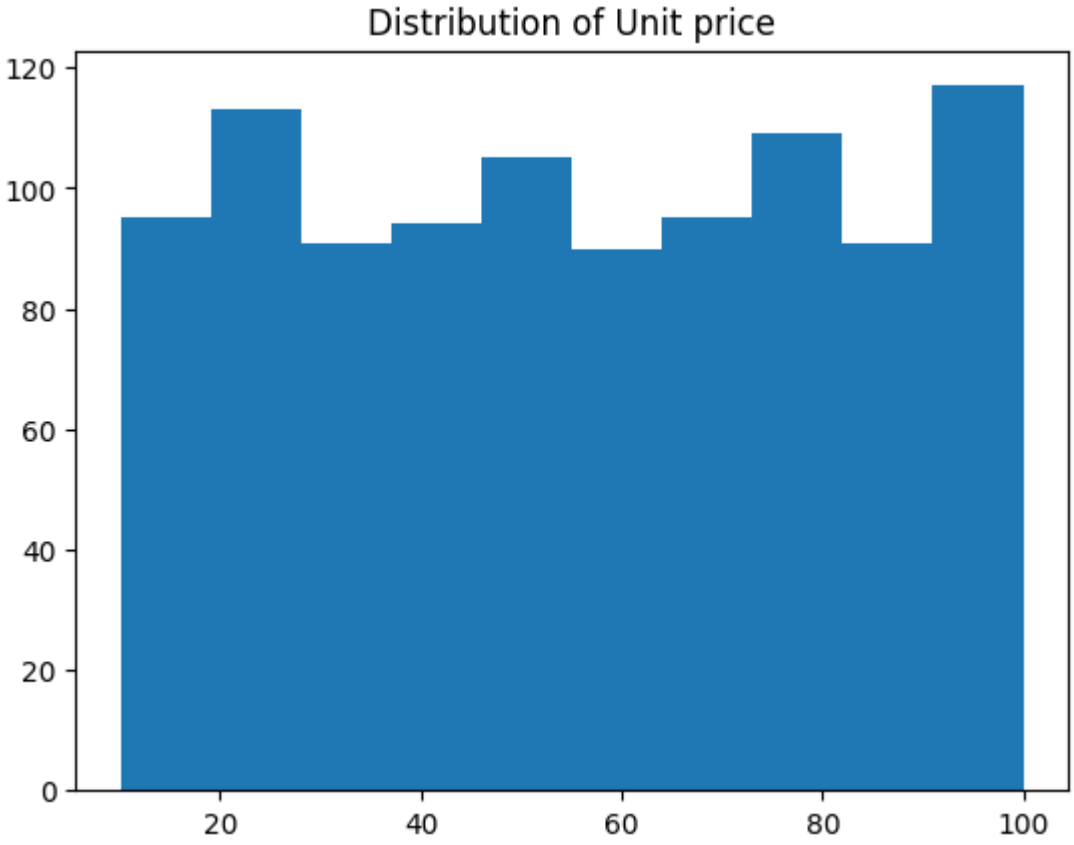
```
Unit price Std:  26.4946
Quantity Std:  2.9234
Tax 5% Std:  11.7088
Total Std:  245.8853
cogs Std:  234.1765
gross margin percentage Std:  0.0
gross income Std:  11.7088
Rating Std:  1.7186
```

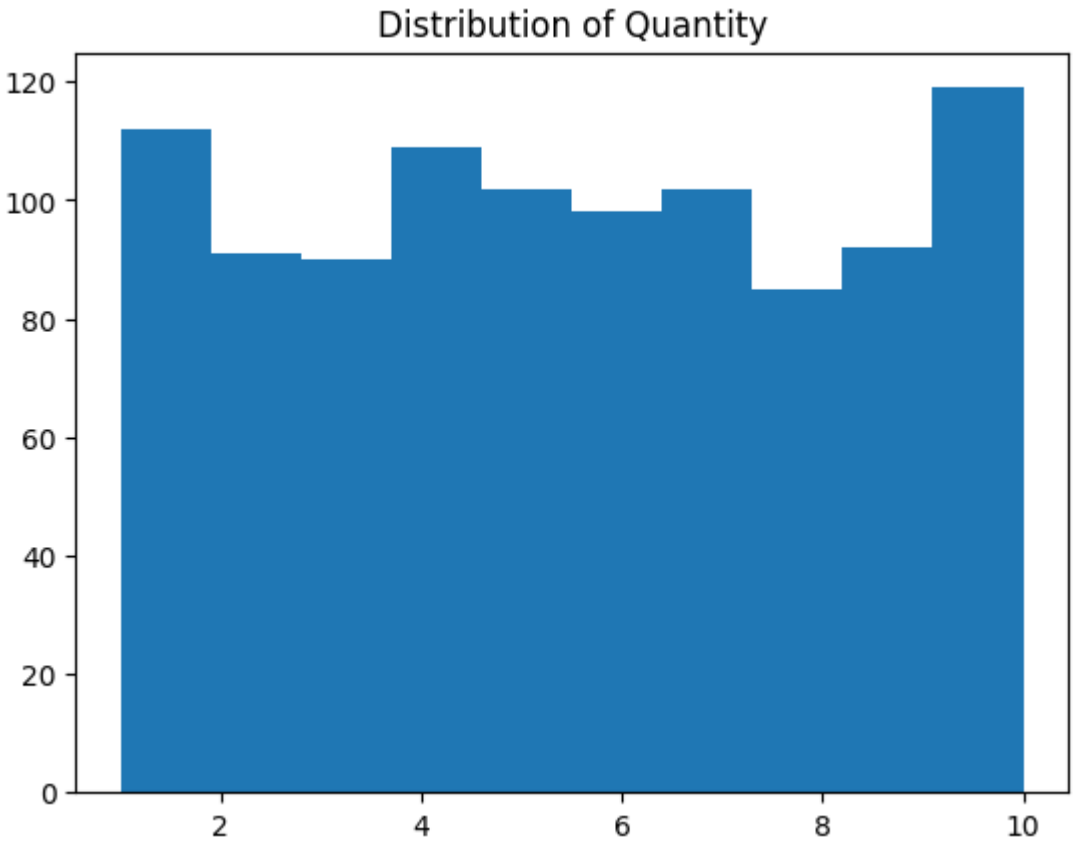
In [ ]:
```python
# get min of numerical columns
for col in numerical_columns:
    print(col,'min',df[col].min())

# get max of numerical columns
    print(col,'max',df[col].max(),'\n')
```

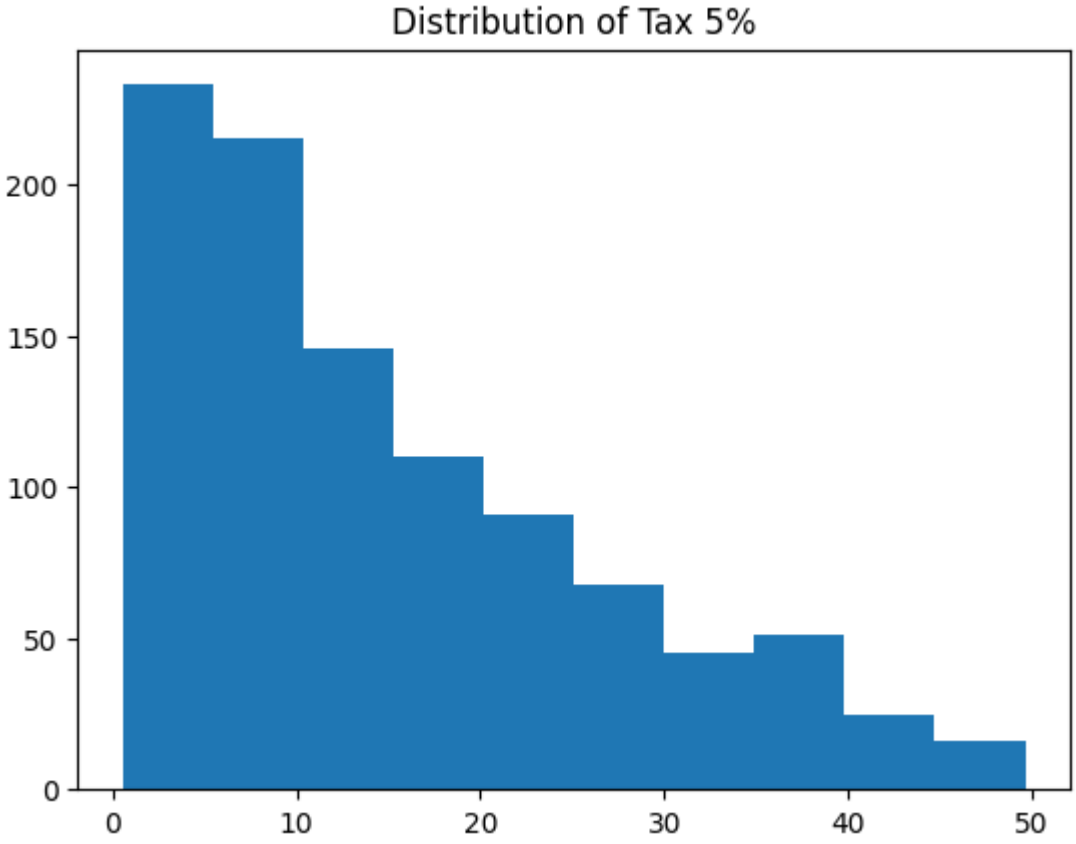
```
Unit price min 10.08
Unit price max 99.96

Quantity min 1
Quantity max 10

Tax 5% min 0.5085
Tax 5% max 49.65

Total min 10.6785
Total max 1042.65

cogs min 10.17
cogs max 993.0

gross margin percentage min 4.761904762
gross margin percentage max 4.761904762

gross income min 0.5085
gross income max 49.65

Rating min 4.0
Rating max 10.0
```

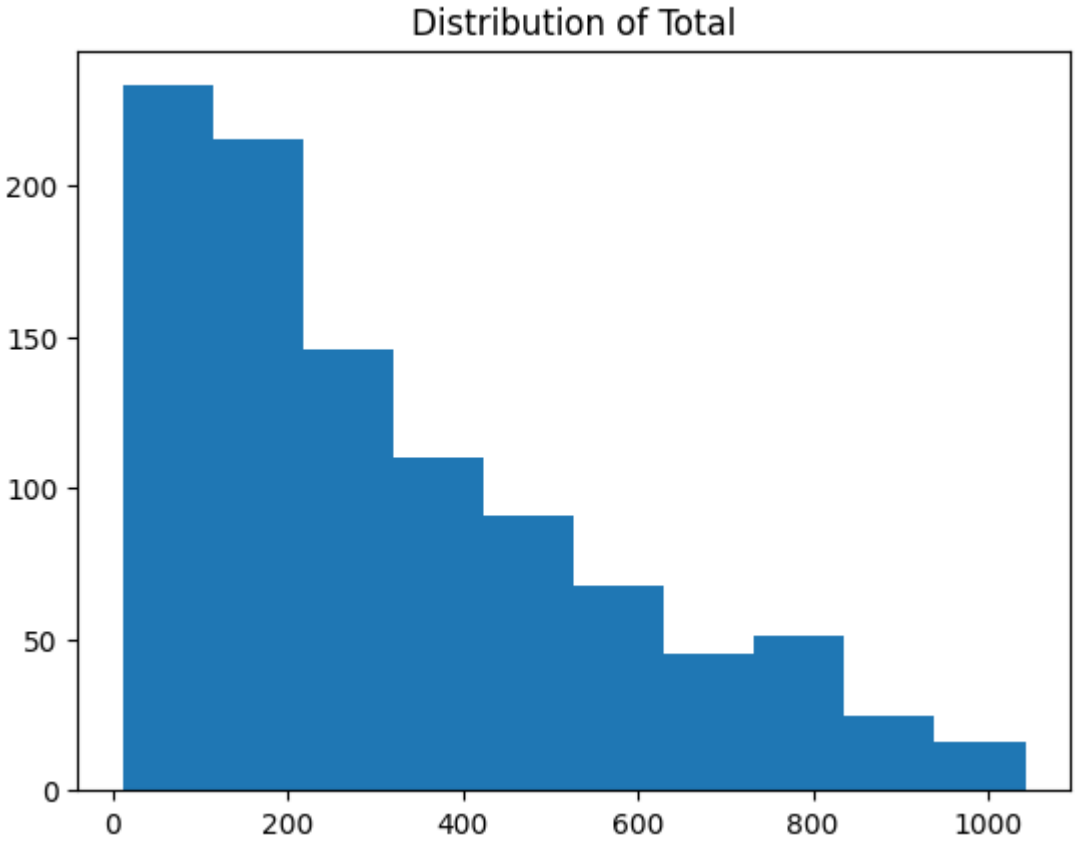
In [ ]:
```python
# Loop through column names
for col in numerical_columns:
    # Get distribution of numerical columns
```

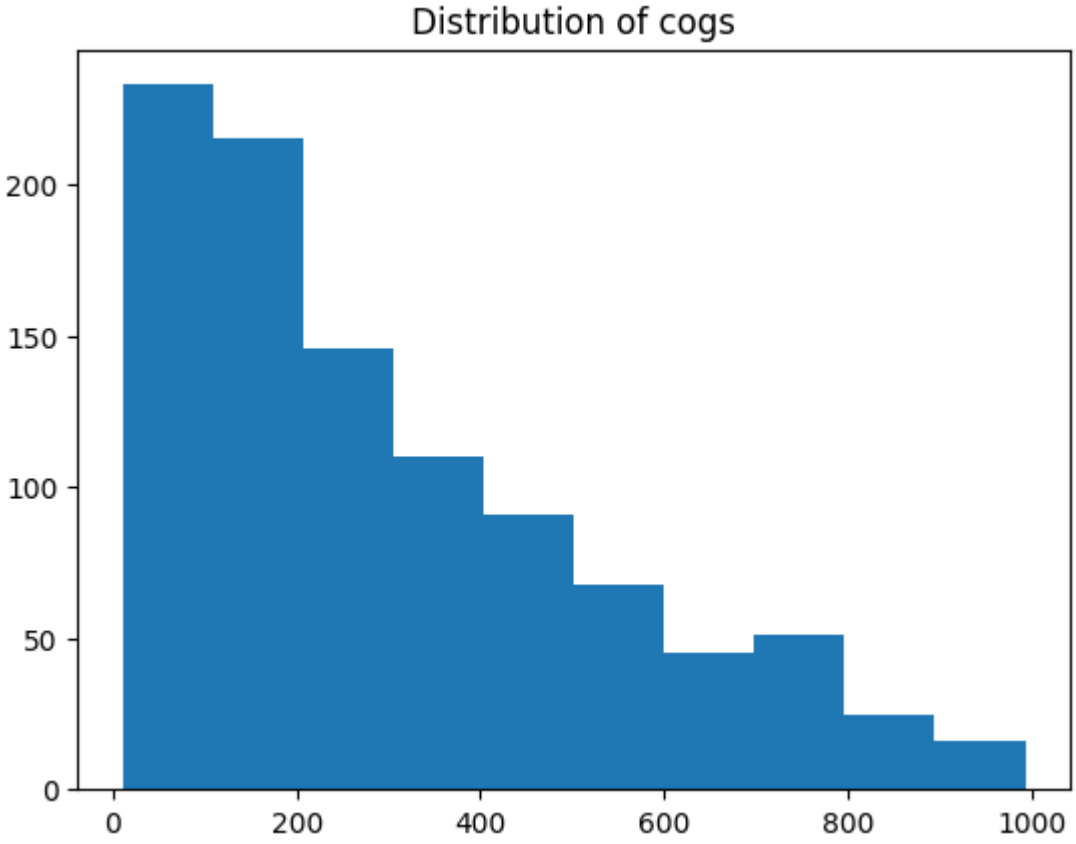
```
    plt.hist(df[col])
    plt.title(f'Distribution of {col}')

    plt.show()
```

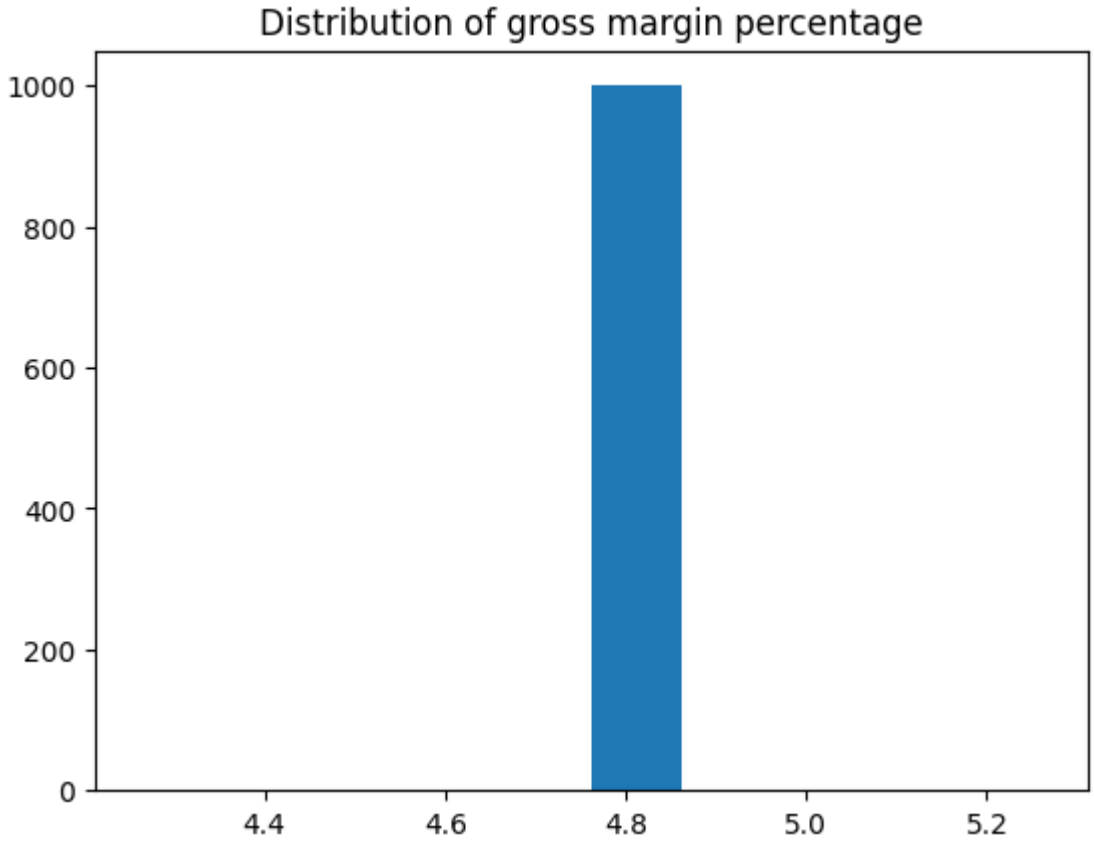


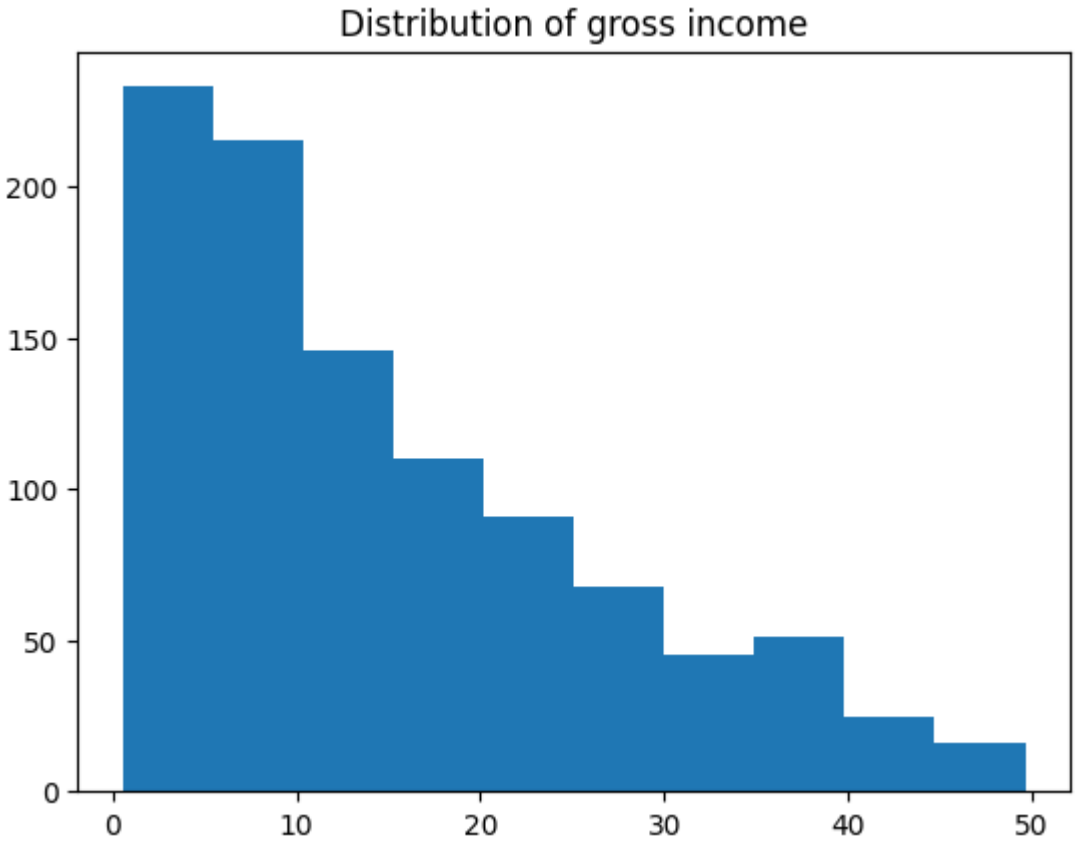


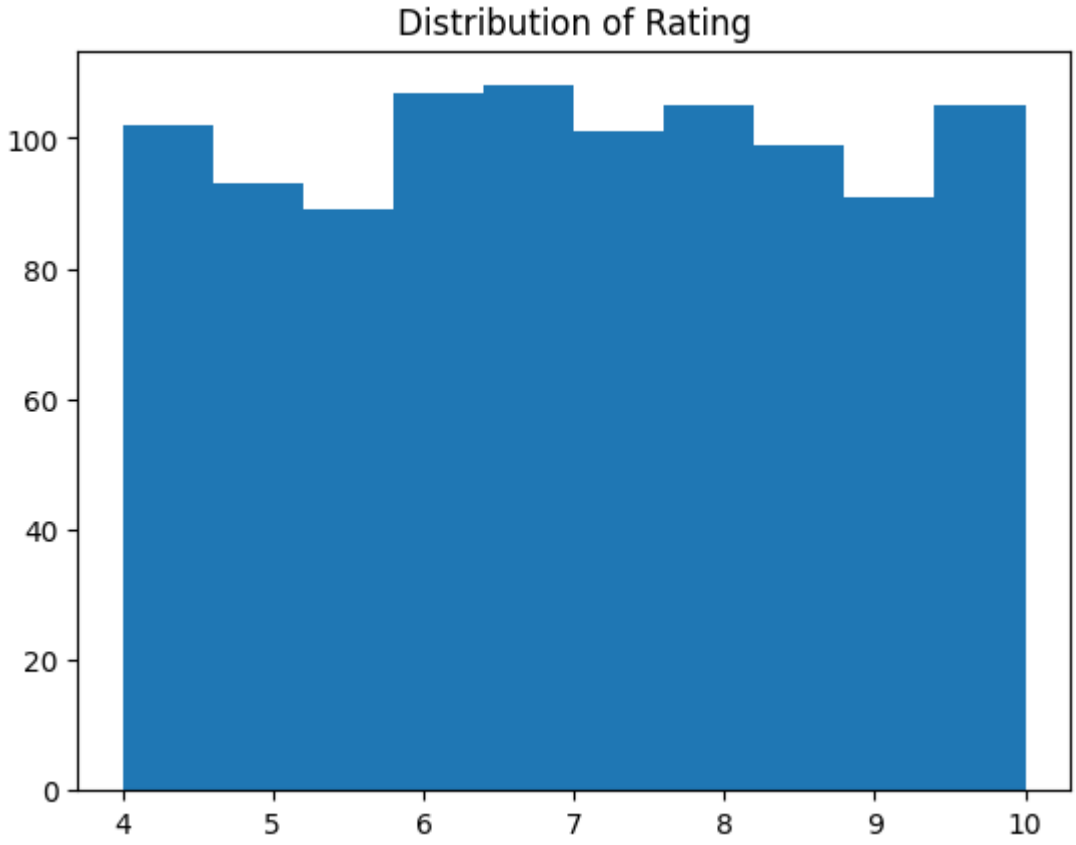

Distribution of Tax 5%
200
150
100
50
0
0
10
20
30
40
50


Distribution of Total
200
150
100
50
0
0
200
400
600
800
1000


Distribution of cogs
200
150
100
50
0
0
200
400
600
800
1000


Distribution of gross margin percentage
1000
800
600
400
200
0
4.4
4.6
4.8
5.0
5.2


Distribution of gross income
200
150
100
50
0
0
10
20
30
40
50


Distribution of Rating
100
80
60
40
20
0
4
5
6
7
8
9
10

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

Out[ ]:

	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

In [ ]:
```
# check for null values
df.isnull().value_counts()
```

Out[ ]:
```
Invoice ID  Branch  City   Customer type  Gender  Product line  Unit price
Quantity  Tax 5%  Total  Date   Time   Payment  cogs   gross margin percent
age  gross income  Rating
False       False   False  False          False   False         False
False     False   False  False  False  False    False  False
False         False     1000
Name: count, dtype: int64
```

Sales Analysis

In [ ]:
```
# Calculate total sales and average sales

# Total Sales
total_sales = df['Total'].sum()
print(f"Total sales: {round(total_sales,2)}") # 322966.75

# Average Sales
print(f"Average sales: {round(total_sales/len(df),2)}") # 322.97
```

```
Total sales: 322966.75
Average sales: 322.97
```

In [ ]:
```
# Best selling products
best_sellers = df.groupby('Product line')['Total'].sum().reset_index()

# display best_sellers data frame
display(best_sellers)
```

	Product line	Total
0	Electronic accessories	54337.5315
1	Fashion accessories	54305.8950
2	Food and beverages	56144.8440
3	Health and beauty	49193.7390
4	Home and lifestyle	53861.9130
5	Sports and travel	55122.8265

In [ ]:
```
df['Time'][0]
```

Out[ ]:
```
'13:08'
```

In [ ]:
```
# Reformat date and time columns to date time with feature engineering

# Reformat Date column
df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y')

# Reformat time column
df['Time'] = pd.to_datetime(df["Time"])

# Create a copy of data frame since we are introducing new columns
df2 = df

# Create new column with day of the week
df2['Day_of_week'] = df['Date'].dt.day_name()

# Create new column with the hour of purchase
df2['Hour_of_day'] = df['Time'].dt.hour
```

Out[ ]:

	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

In [ ]:
```
# Determine the busiest shoping days and hours

# Create busiest shopping days data frame
busiest_days = df['Day_of_week'].value_counts().reset_index()

# Display busiest_days
display(busiest_days)

# Create busiest shopping hours data frame
busiest_hours = df['Hour_of_day'].value_counts().reset_index()

# Display busiest_hours
display(busiest_hours)
```

	Day_of_week	count
0	Saturday	164
1	Tuesday	158
2	Wednesday	143
3	Friday	139
4	Thursday	138
5	Sunday	133
6	Monday	125

	Hour_of_day	count
0	19	113
1	13	103
2	15	102
3	10	101
4	18	93
5	11	90
6	12	89
7	14	83
8	16	77
9	20	75
10	17	74

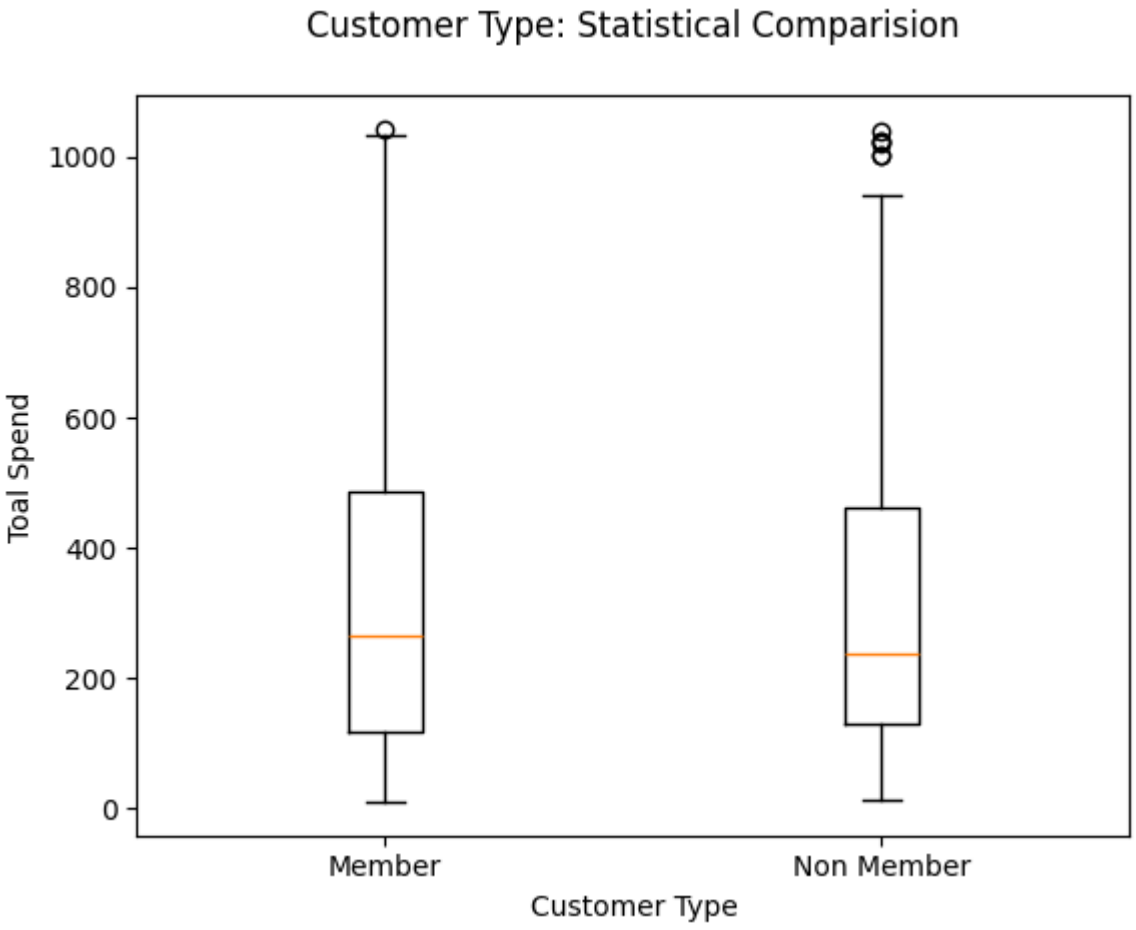
Customer Behavior Analysis

In [ ]:
```
# Analyze customer types distribution
customer_type = df['Customer type'].value_counts().reset_index()

# Display customer type analysis
display(customer_type)

# Analyze customer types payment methods
payment_method_by_customer = df.groupby('Customer type')['Payment'].value_cc

# Display payment method analysis
display(payment_method_by_customer)
```

	Customer type	count
0	Member	501
1	Normal	499

	Customer type	Payment	count
0	Member	Credit card	172
1	Member	Cash	168
2	Member	Ewallet	161
3	Normal	Ewallet	184
4	Normal	Cash	176
5	Normal	Credit card	139

In [ ]:
```
# Create member data frame
member_data = df[df['Customer type']=='Member']['Total']

# Create non member data frame
nonmember_data = df[df['Customer type']=='Normal']['Total']

# Create boxplot
plt.boxplot([member_data,nonmember_data])

# Set x-axis labels
plt.xticks([1,2], ['Member', 'Non Member'])

# Add title and and labels
plt.title('Customer Type: Statistical Comparision\n')
plt.xlabel('Customer Type')
plt.ylabel('Toal Spend')

plt.show()
```



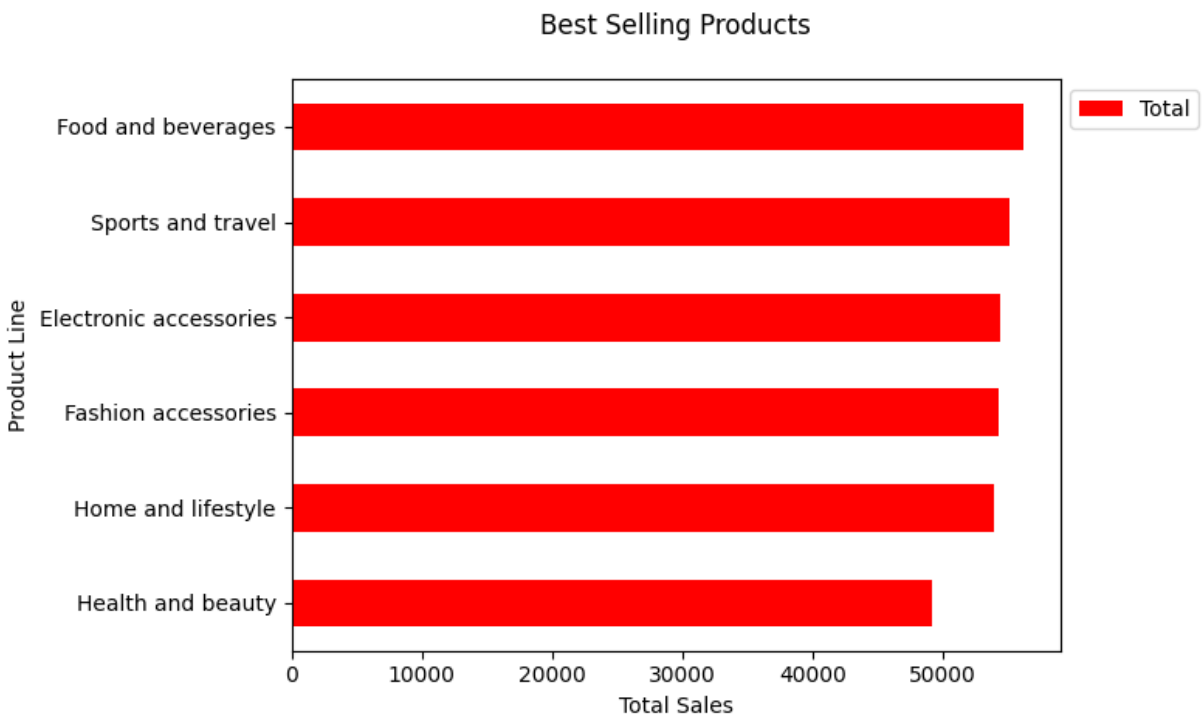
In [ ]:
```
# Product line by membertype
display(df.groupby('Customer type')['Product line'].value_counts())

# Product line by gender
display(df.groupby(['Gender','Product line'])['Total'].sum())
```

```
Customer type  Product line
Member         Food and beverages        94
               Sports and travel         87
               Fashion accessories       86
               Home and lifestyle        83
               Electronic accessories    78
               Health and beauty         73
Normal         Electronic accessories    92
               Fashion accessories       92
               Food and beverages        80
               Health and beauty         79
               Sports and travel         79
               Home and lifestyle        77
Name: count, dtype: int64
Gender  Product line
Female  Electronic accessories    27102.0225
        Fashion accessories       30437.4000
        Food and beverages        33170.9175
        Health and beauty         18560.9865
        Home and lifestyle        30036.8775
        Sports and travel         28574.7210
Male    Electronic accessories    27235.5090
        Fashion accessories       23868.4950
        Food and beverages        22973.9265
        Health and beauty         30632.7525
        Home and lifestyle        23825.0355
        Sports and travel         26548.1055
Name: Total, dtype: float64
```

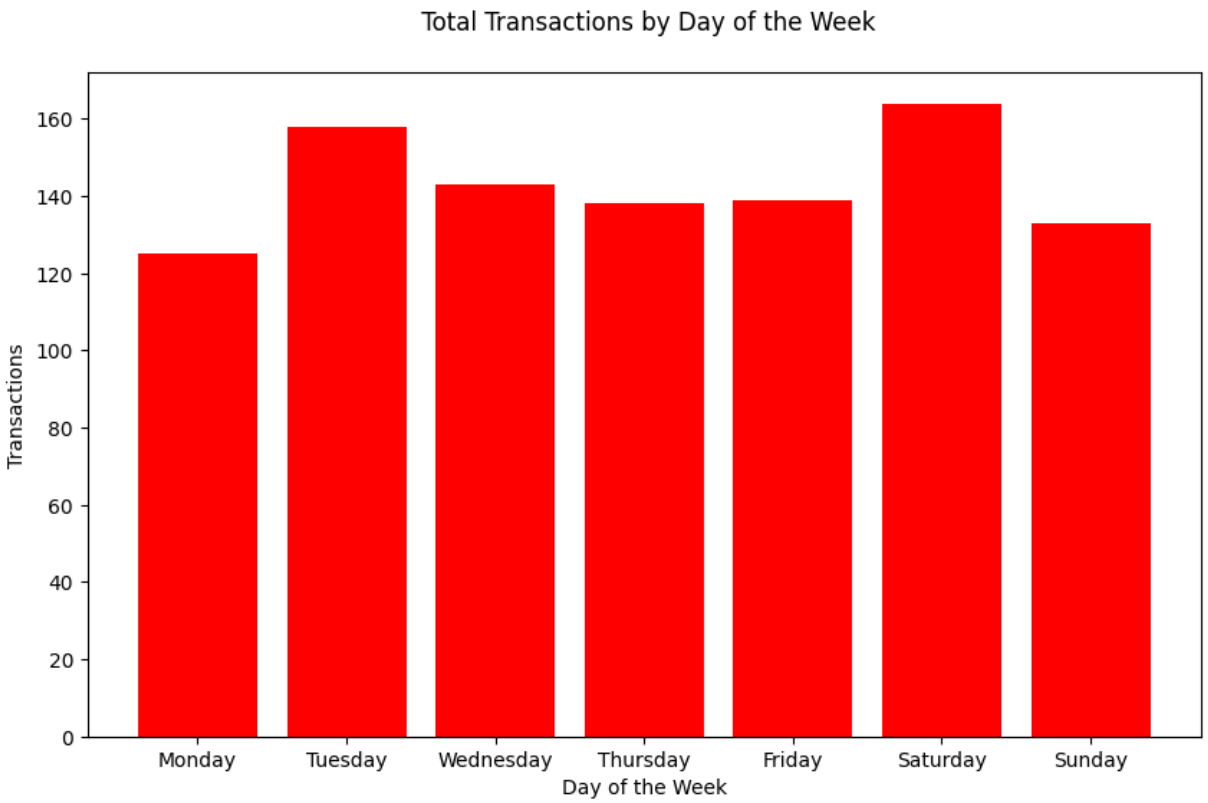
Visualization

In [ ]:

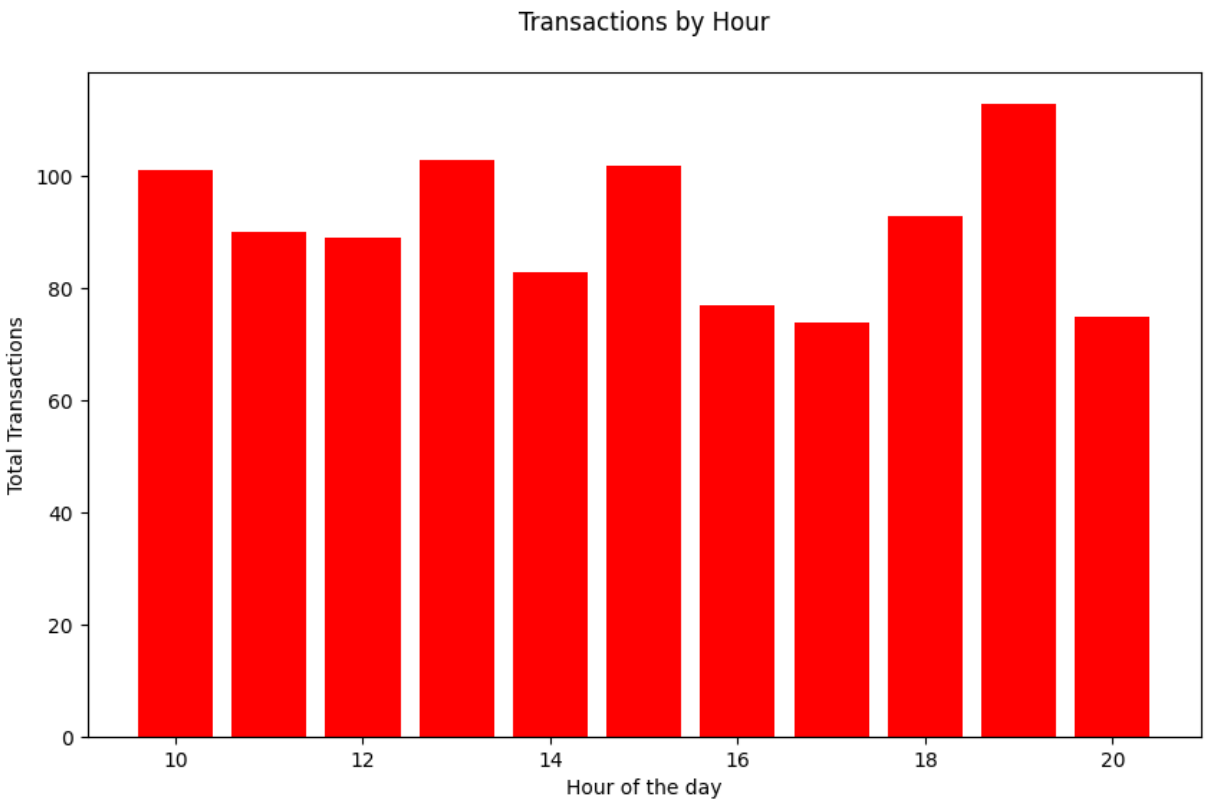
```
# Sort best_sellers by total
sorted_best_sellers = best_sellers.sort_values(by='Total')

# Create bar chart to display best sellers total spend
sorted_best_sellers.plot(kind='barh',x='Product line',y='Total',color='red')

# Add labels and title
plt.title('Best Selling Products\n')
plt.xlabel('Total Sales')
plt.ylabel('Product Line')
plt.legend(loc='upper left', bbox_to_anchor=(1,1))

plt.show()
```



In [ ]:
```
# Convert 'Day_of_week' to a categorical type with ordered categories
busiest_days['Day_of_week'] = pd.Categorical(busiest_days['Day_of_week'], ca
    'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sur
], ordered=True)

# Sort by 'Day_of_week'
sorted_day_of_week = busiest_days.sort_values('Day_of_week')

# Resize plot
plt.figure(figsize=(10,6))

# Create the bar plot
plt.bar(sorted_day_of_week['Day_of_week'], sorted_day_of_week['count'],color

# Add titles and labels
plt.title(' Total Transactions by Day of the Week\n')
plt.xlabel('Day of the Week')
plt.ylabel('Transactions')

plt.show()
```



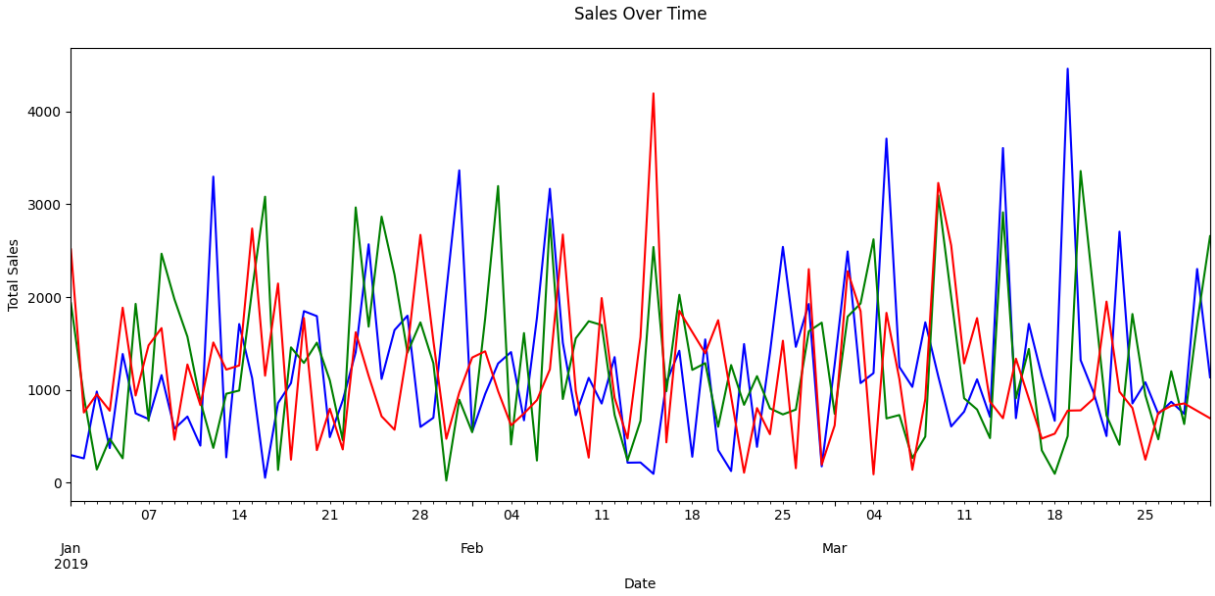
In [ ]:
```
# Resize figure
plt.figure(figsize=(10,6))

# Create bar chart showing transactions by hour of the day
plt.bar(busiest_hours['Hour_of_day'],busiest_hours['count'],color='red')

# Add title and labels
plt.title('Transactions by Hour\n')
plt.xlabel('Hour of the day')
plt.ylabel('Total Transactions')

plt.show()
```



In [ ]:
```
# Create data frame grouping data by Date and sum of sales
total_by_date = df.groupby('Date')['Total'].sum()

# Resize figure
plt.figure(figsize=(10,4))

# Create line chart for sales over time
total_by_date.plot(kind='line',x='Date',y='Total',color='red')

# Add title and labels
plt.title('Sales Over Time\n')
plt.xlabel('Date')
plt.ylabel('Total Sales')

plt.show()
```

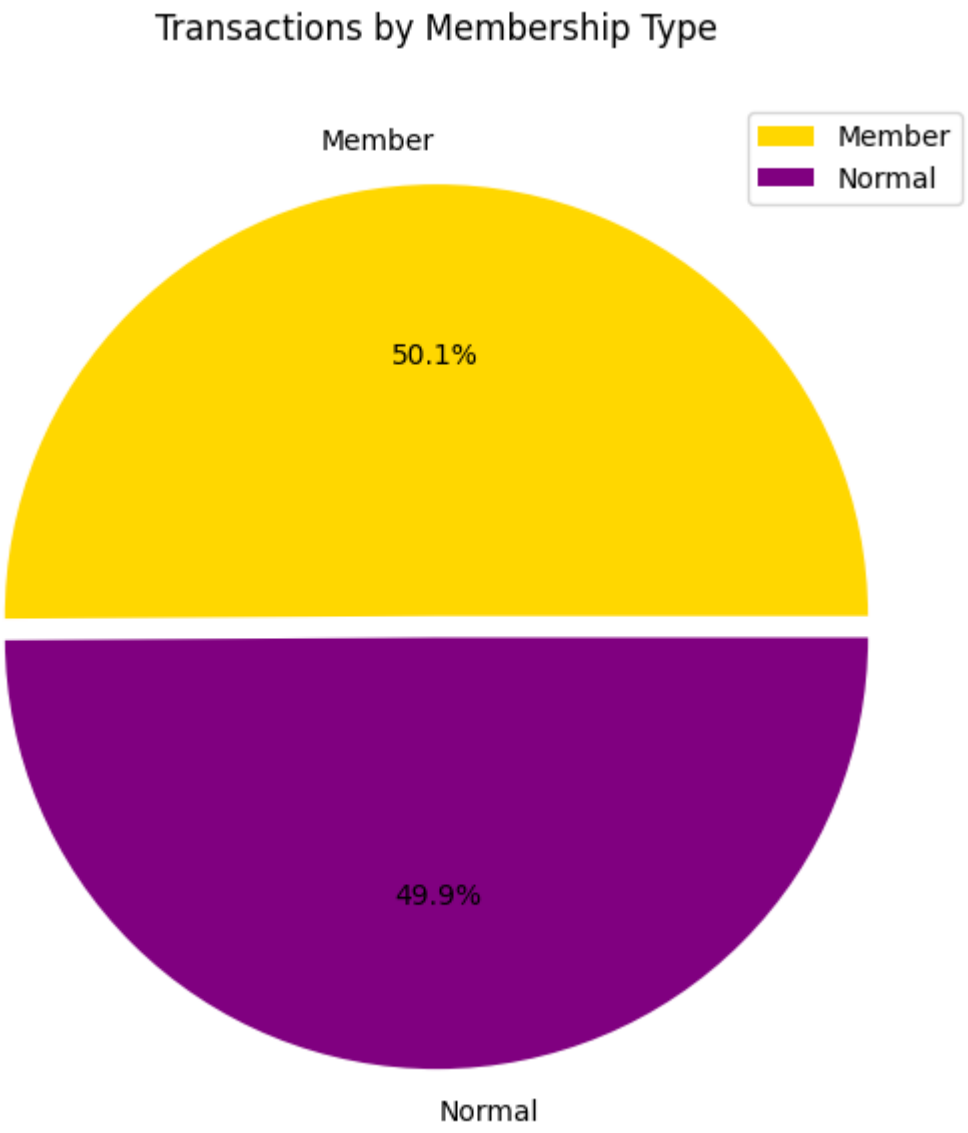
In [ ]:

```
# Extract payment type unique values
payment_types = df['Payment'].unique()

# create empty listto store data frames
pt_data_frames = []

#loop through payment type names
for i in payment_types:
    payment_type_df = df[df['Payment']==i]

    pt_data_frames.append(payment_type_df)

# Resize figure
plt.figure(figsize=(15,6))

# Create line chart for sales over time
#total_by_date.plot(kind='line',x='Date',y='Total',color='red')
pt_data_frames[0].groupby('Date')['Total'].sum().plot(kind='line',x='Date',y
pt_data_frames[1].groupby('Date')['Total'].sum().plot(kind='line',x='Date',y
pt_data_frames[2].groupby('Date')['Total'].sum().plot(kind='line',x='Date',y

# Add title and labels
plt.title('Sales Over Time\n')
plt.xlabel('Date')
plt.ylabel('Total Sales')

plt.show()
```



In [ ]:

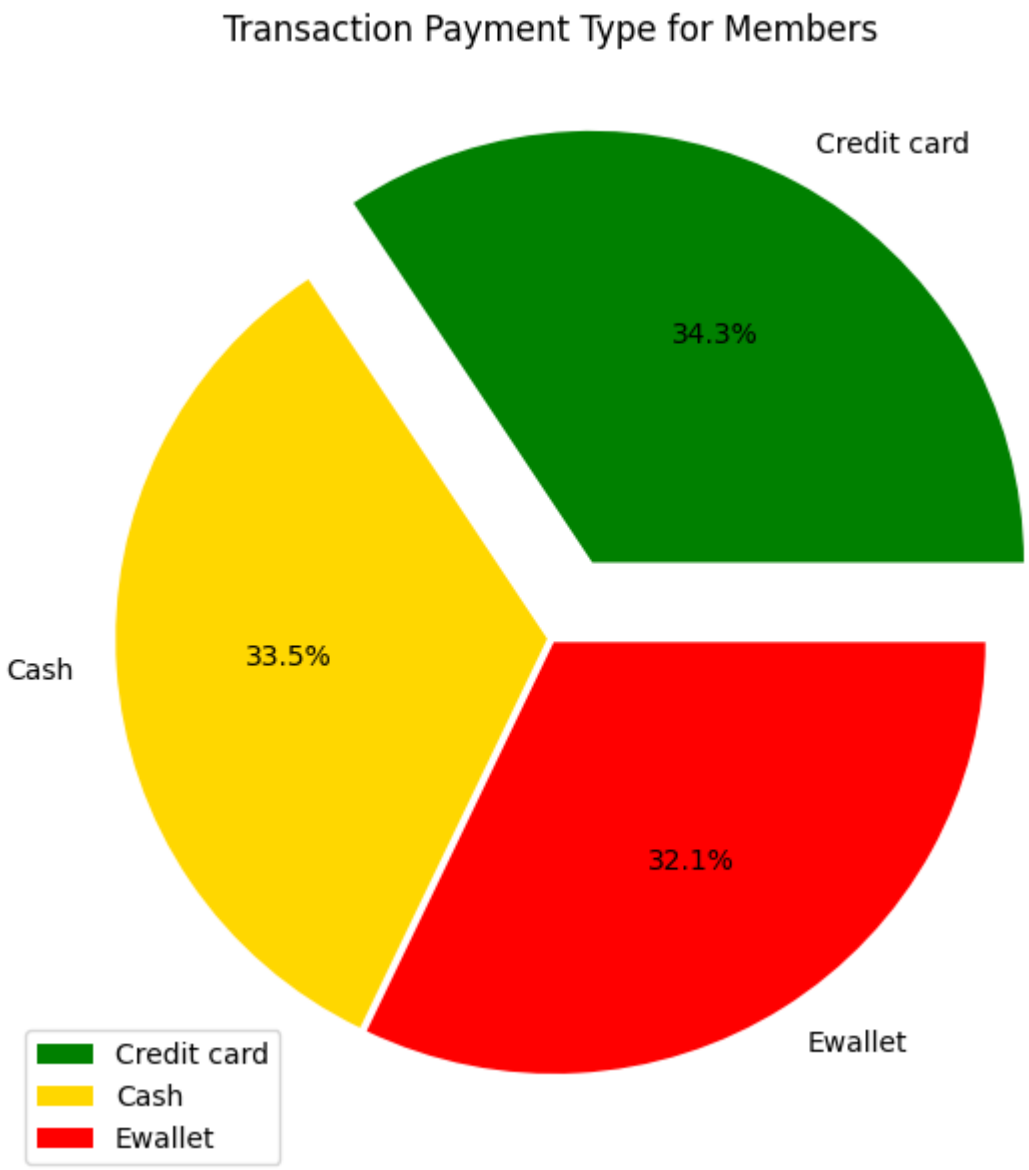
```
# Get unique customer types for the labels of the pie chart
labels = df['Customer type'].unique()

# Get the count of each unique customer type for the sizes of the pie chart
sizes = df['Customer type'].value_counts().values

# Define the colors to be used for each customer type
colors = ['gold','purple']

# Set the figure size for the pie chart
plt.figure(figsize=(7,7))

# Create the pie chart
# 'sizes' specifies the sizes of the pie slices
# 'explode' offsets the first slice (for better visibility)
# 'labels' specifies the labels of the slices
# 'colors' specifies the colors of the slices
# 'autopct' specifies the format of the percentage to display on each slice
plt.pie(sizes, explode=(.05,0), labels=labels, colors=colors, autopct='%1.1f

# Add a title to the pie chart
plt.title('Transactions by Membership Type\n')

# Add a legend to the pie chart, placing it at the 'best' location to avoid
plt.legend(labels, loc='best')

# Display the pie chart
plt.show()
```



In [ ]:

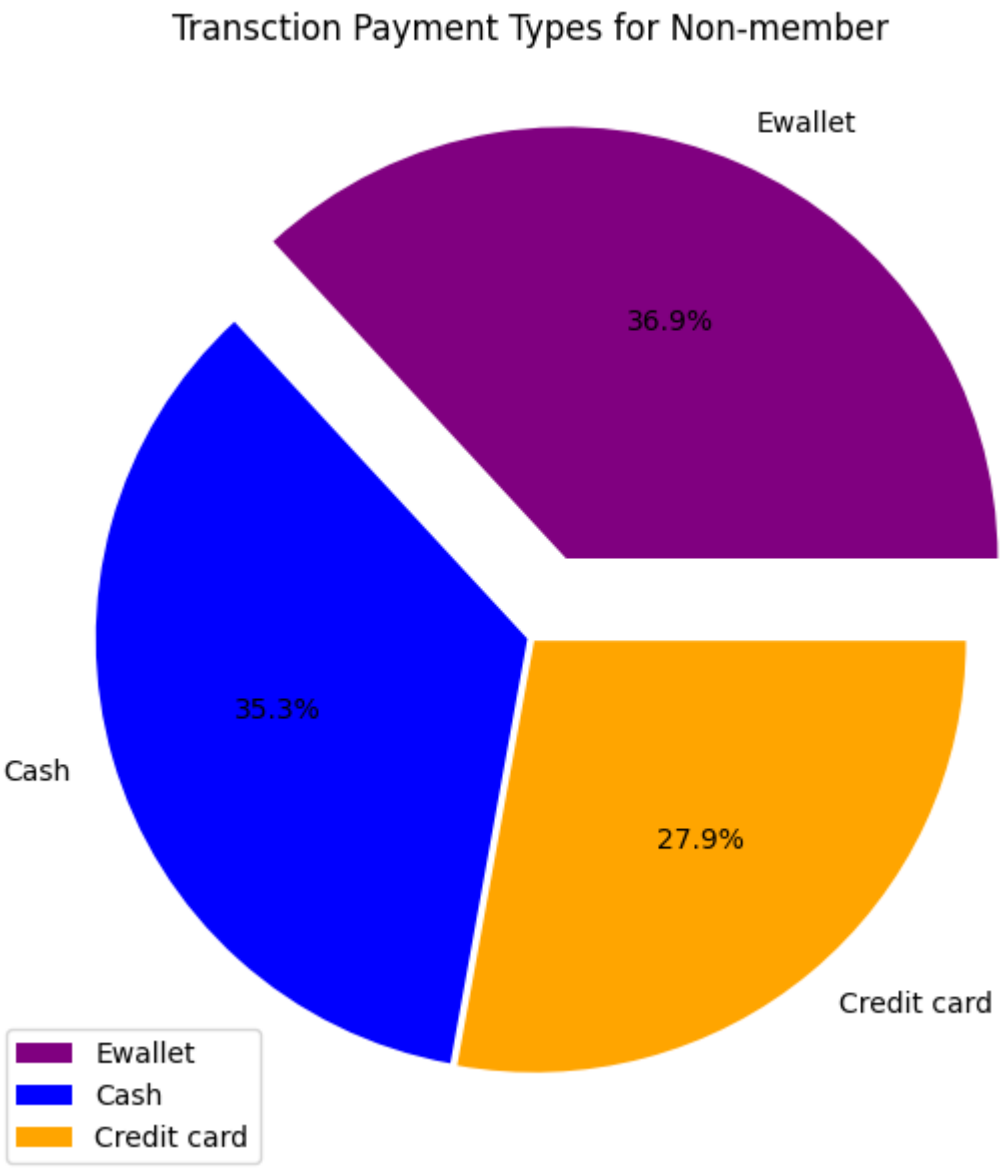
```
# Filter the DataFrame to only include rows where 'Customer type' is 'Member
# Then, count the occurrences of each 'Payment' type and reset the index
df_member = df[df['Customer type']=='Member']['Payment'].value_counts().rese

# Get unique payment types for the labels of the pie chart
labels = df_member['Payment'].unique()

# Get the count of each unique payment type for the sizes of the pie chart
sizes = df_member['count'].values

# Define the colors to be used for each payment type
colors = ['green','gold','red']

# Set the figure size for the pie chart
plt.figure(figsize=(7,7))

# Create the pie chart
# 'sizes' specifies the sizes of the pie slices
# 'explode' offsets the first slice (for better visibility)
```

```
# 'labels' specifies the labels of the slices
# 'colors' specifies the colors of the slices
# 'autopct' specifies the format of the percentage to display on each slice
plt.pie(sizes, explode=(0.2,0.01,0.01), labels=labels, colors=colors, autopc

# Add a title to the pie chart
plt.title('Transaction Payment Type for Members\n')

# Add a legend to the pie chart, placing it at the 'lower left' to avoid ove
plt.legend(labels, loc='lower left')

# Display the pie chart
plt.show()
```



In [ ]:
```
# Filter the DataFrame to only include rows where 'Customer type' is 'Normal
# Then, count the occurrences of each 'Payment' type and reset the index
df_nonmember = df[df['Customer type']=='Normal']['Payment'].value_counts().r

# Get unique payment types for the labels of the pie chart
labels = df_nonmember['Payment'].unique()
```

```
# Get the count of each unique payment type for the sizes of the pie chart
sizes = df_nonmember['count'].values

# Define the colors to be used for each payment type
colors = ['purple','blue','orange']

# Set the figure size for the pie chart
plt.figure(figsize=(7,7))

# Create the pie chart
# 'sizes' specifies the sizes of the pie slices
# 'explode' offsets the first slice (for better visibility)
# 'labels' specifies the labels of the slices
# 'colors' specifies the colors of the slices
# 'autopct' specifies the format of the percentage to display on each slice
plt.pie(sizes, explode=(0.2,0.01,0.01), labels=labels, colors=colors, autopc

# Add a title to the pie chart
plt.title('Transction Payment Types for Non-member\n')

# Add a legend to the pie chart, placing it at the 'lower left' to avoid ove
plt.legend(labels, loc='lower left')

# Display the pie chart
plt.show()
```



Insights and Recommendations

Insights

1. Customer membership type is about even with 50.1% Member and 49.9% Non Member
2. Members use payment types about 1/3 for eacy type where as Non-members use Ewallet about 37% over Credit Cards at about 28%.
3. There is peak spending at the beginning and middle of the months
4. Valentines day Feb 14th saw the largest spending of all dates

Recommendations

1. Run a campaign to increase membership
2. Non members are using ewallets which has less transaction fees, we should run a campaign to increase member ewallet transactions

3. We can run the campaigns during the beginning and middle of the month near peak spending times
4. Offer a yearly valentines promotion to increase membership and ewallet use.