Tina Borundia

C1-16

PART A: [Matplotlib]

Write a program to visualize Company Sales Data & perform the following task

(CSV file will be provided)

NOTE: Set the Title, xlLable, yLable for all the plots.

In [23]:

```
# 1. Read total profit of all months and show line plot with the following Sty
le properties.
# •     Line Style dotted and Line-color should be red
# •     Show legend at the lower right location.
# •     X label name = Month Number
# •     Y label name = Sold units number
# •     Add a circle marker.
# •     Line marker color as red
# •     Line width should be 3
import pandas
import numpy as np
import matplotlib.pyplot as plt
dataset=pandas.read_csv('company_sales_data.csv - company_sales_data.csv.csv')
dataset
```

Out[23]:

	month_number	facecream	facewash	toothpaste	bathingsoap	shampoo	moisturizer	total_u
0	1	2500	1500	5200	9200	1200	1500	21
1	2	2630	1200	5100	6100	2100	1200	18
2	3	2140	1340	4550	9550	3550	1340	22
3	4	3400	1130	5870	8870	1870	1130	22
4	5	3600	1740	4560	7760	1560	1740	20
5	6	2760	1555	4890	7490	1890	1555	20
6	7	2980	1120	4780	8980	1780	1120	29
7	8	3700	1400	5860	9960	2860	1400	36
8	9	3540	1780	6100	8100	2100	1780	23
9	10	1990	1890	8300	10300	2300	1890	26
10	11	2340	2100	7300	13300	2400	2100	41
11	12	2900	1760	7400	14400	1800	1760	30

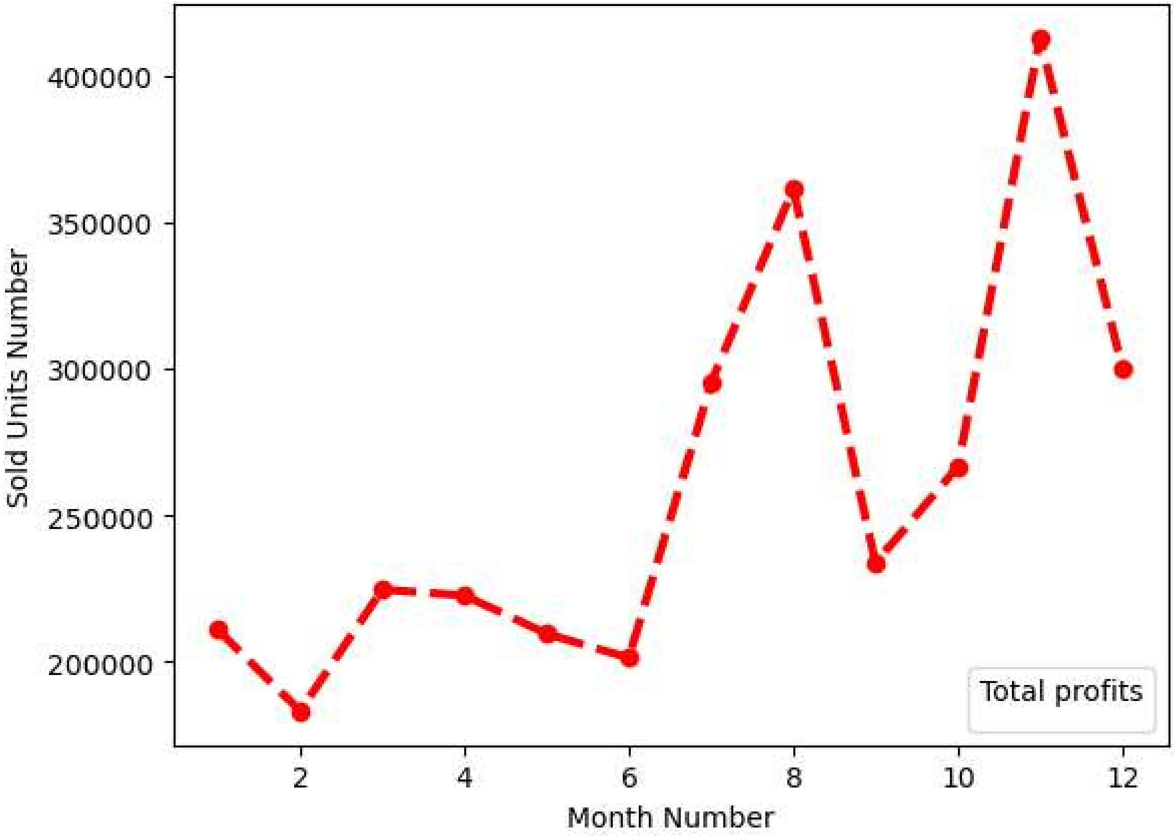
In [7]:

```
month_numbers = dataset['month_number']
sold_units = dataset['total_profit']
```

In [14]:

```
plt.plot(month_numbers,sold_units,linestyle='--',color='red',marker='o',mfc='r
ed',linewidth=3)
plt.xlabel("Month Number")
plt.ylabel("Sold Units Number")
plt.legend(title='Total profits',loc='lower right')
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argumen
t.
```

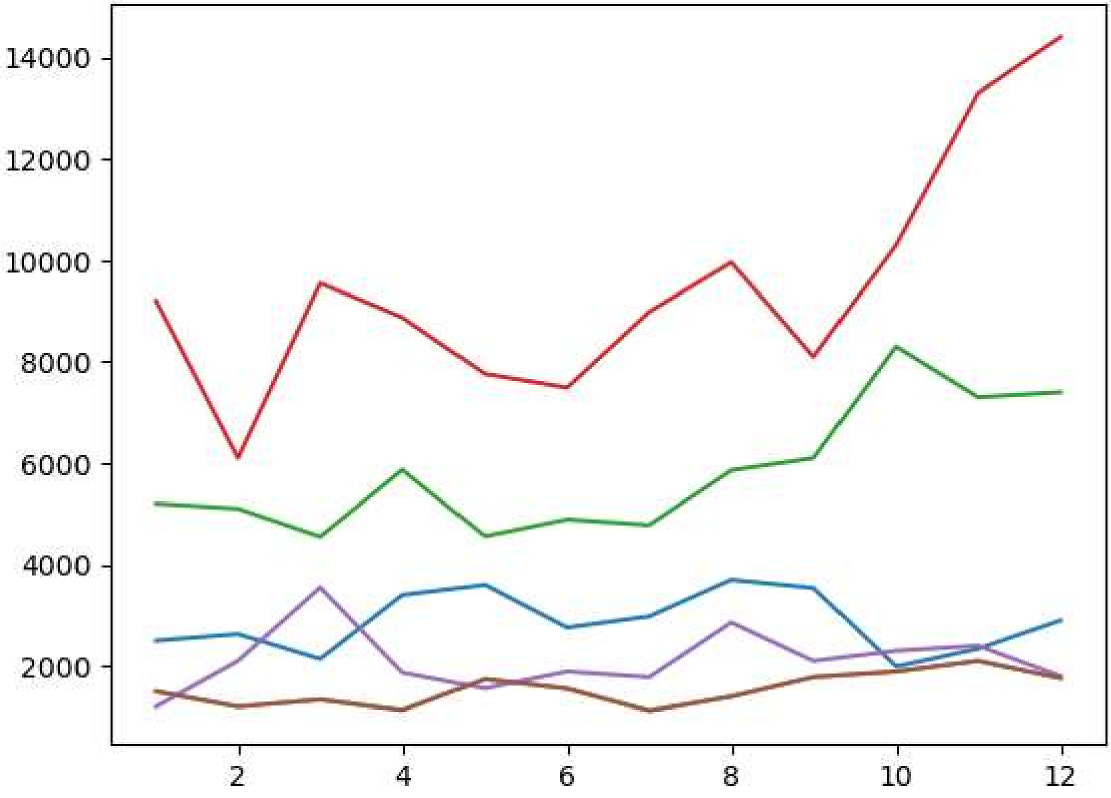


In [15]:

```
# 2. Read all product sales data and show it using a multiline plot
# Display the number of units sold per month for each product using multiline plots. (i.e., Separate Plotline for each product).
fc=dataset['facecream']
fw=dataset['facewash']
tp=dataset['toothpaste']
bs=dataset['bathingsoap']
s=dataset['shampoo']
m=dataset['moisturizer']
plt.plot(month_numbers,fc,month_numbers,fw,month_numbers,tp,month_numbers,bs,month_numbers,s,month_numbers,m)
```

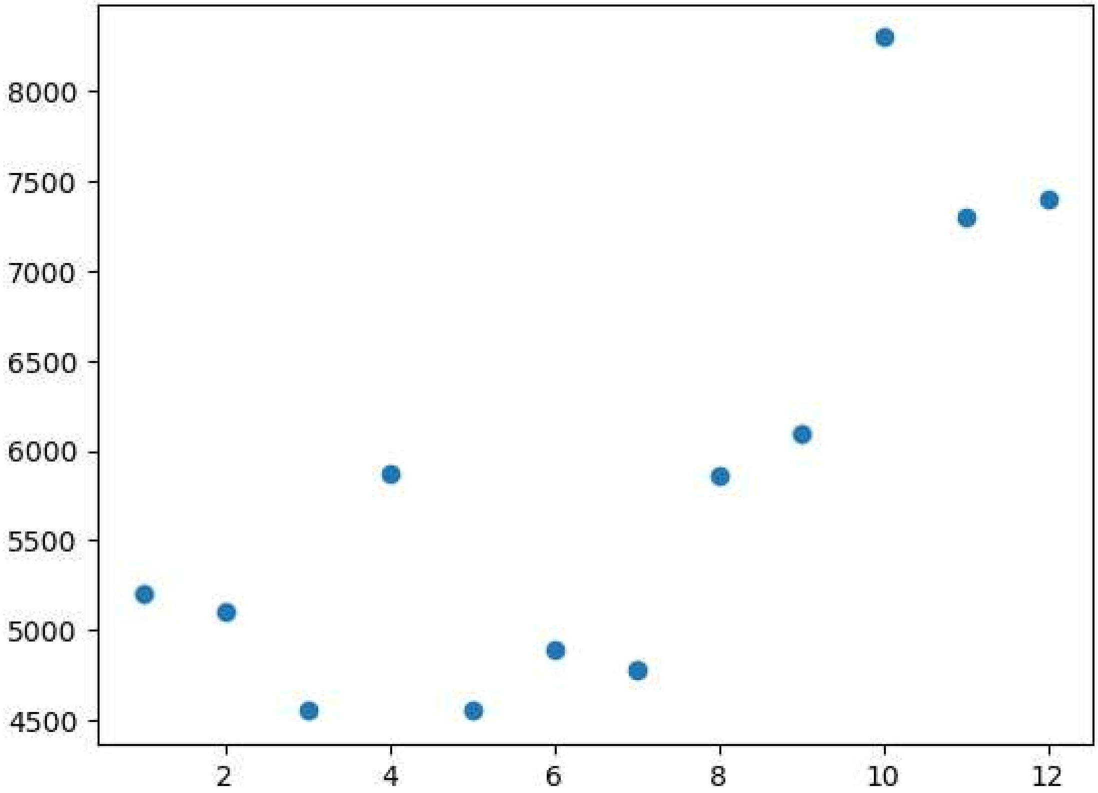
Out[15]:

```
[<matplotlib.lines.Line2D at 0x2960778bf90>,
 <matplotlib.lines.Line2D at 0x296077a4210>,
 <matplotlib.lines.Line2D at 0x2960778a850>,
 <matplotlib.lines.Line2D at 0x296077a4a50>,
 <matplotlib.lines.Line2D at 0x296077a4e50>,
 <matplotlib.lines.Line2D at 0x296077a5390>]
```



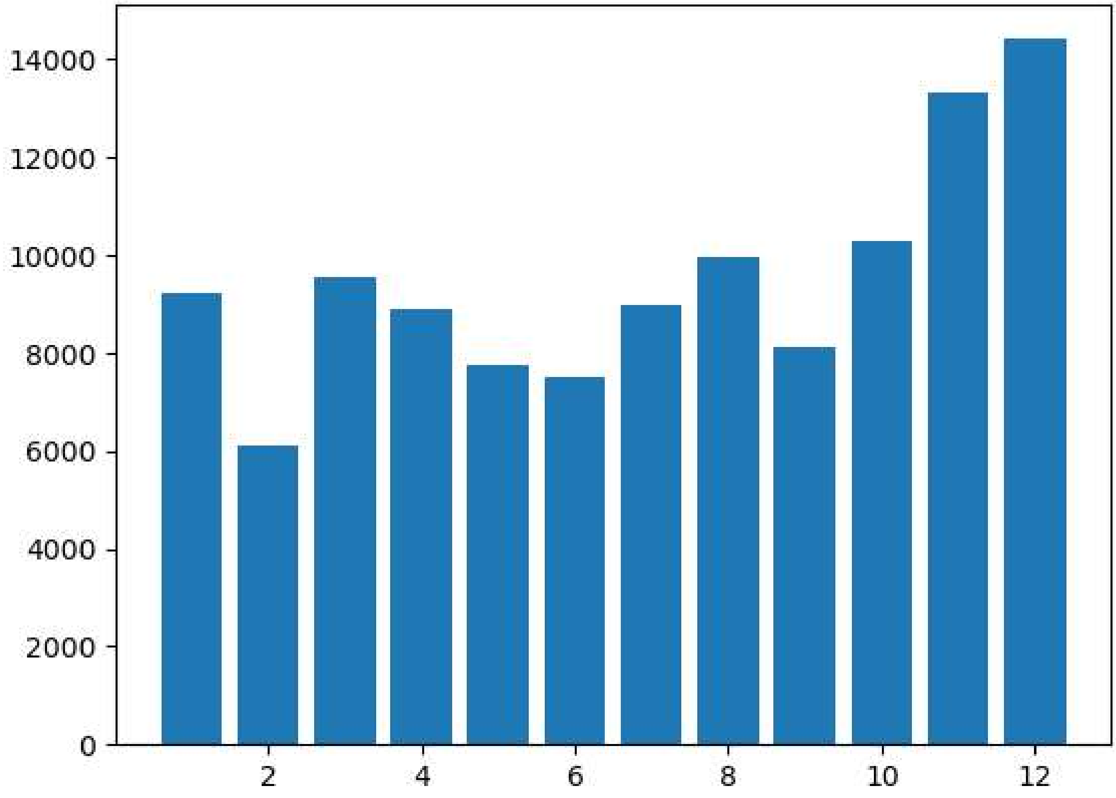
In [17]:

```
# 3. Read toothpaste sales data of each month and show it using a scatter plo
t.
plt.scatter(month_numbers,tp)
plt.show()
```



In [19]:

```
# 4.Read sales data of bathing soap of all months and show it using a bar char
t. Save this plot to your hard disk.
plt.bar(month_numbers,bs)
plt.savefig('bathingsoap.png')
```

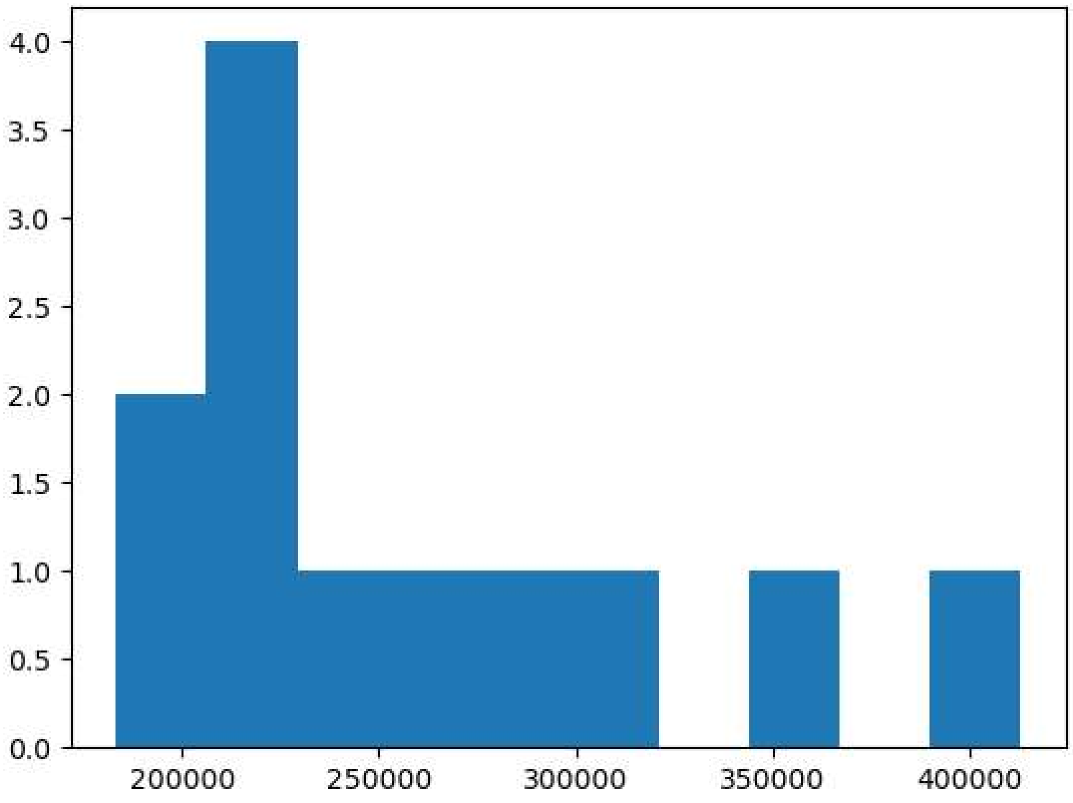


In [26]:

```
# 5. Read the total profit of each month and show it using the histogram to se
e the most common profit ranges.
total=dataset['total_profit']
plt.hist(total)
```

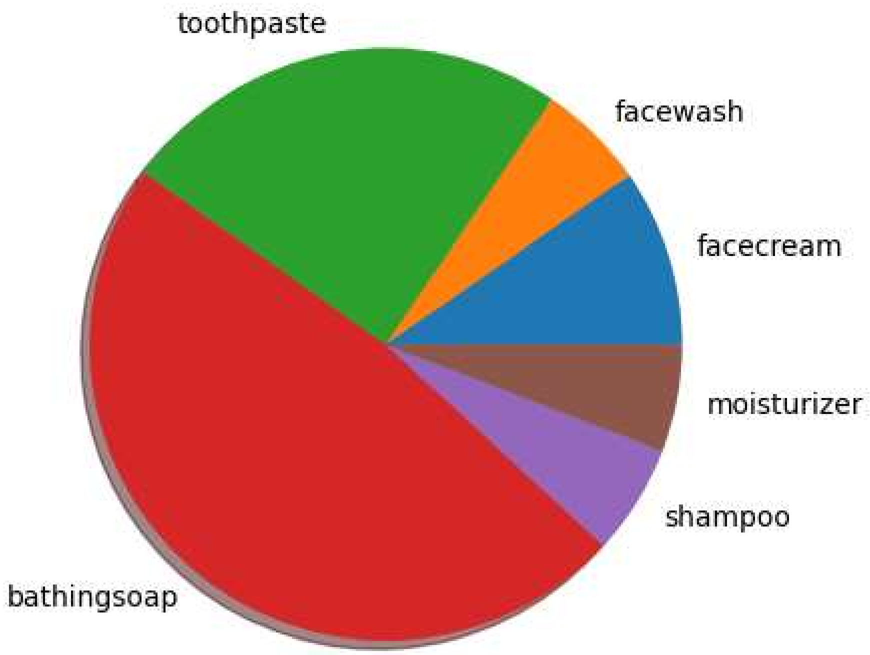
Out[26]:

```
(array([2., 4., 1., 1., 1., 1., 0., 1., 0., 1.]),
 array([183300., 206250., 229200., 252150., 275100., 298050., 321000.,
        343950., 366900., 389850., 412800.]),
 <BarContainer object of 10 artists>)
```



In [32]:
```
# 6. Calculate total sale data for last year for each product and show it using a Pie chart
mylabels=["facecream","facewash","toothpaste","bathingsoap","shampoo","moisturizer"]
last=dataset.iloc[11,[1,2,3,4,5,6]]
plt.pie(last,labels=mylabels,shadow=True)
```

Out[32]:
```
([<matplotlib.patches.Wedge at 0x29609209950>,
  <matplotlib.patches.Wedge at 0x2960920a650>,
  <matplotlib.patches.Wedge at 0x2960920bd50>,
  <matplotlib.patches.Wedge at 0x296092196d0>,
  <matplotlib.patches.Wedge at 0x2960921b010>,
  <matplotlib.patches.Wedge at 0x29609230bd0>],
 [Text(1.0497308896286452, 0.3287325042636539, 'facecream'),
  Text(0.7733276575501096, 0.7822814928579485, 'facewash'),
  Text(-0.19579769486664522, 1.0824339530358924, 'toothpaste'),
  Text(-0.692759273293392, -0.8544498752214851, 'bathingsoap'),
  Text(0.9338816799590318, -0.5812615657661332, 'shampoo'),
  Text(1.0813946245546842, -0.20145884439317607, 'moisturizer')])
```



In [33]:

```
!pip install seaborn
```

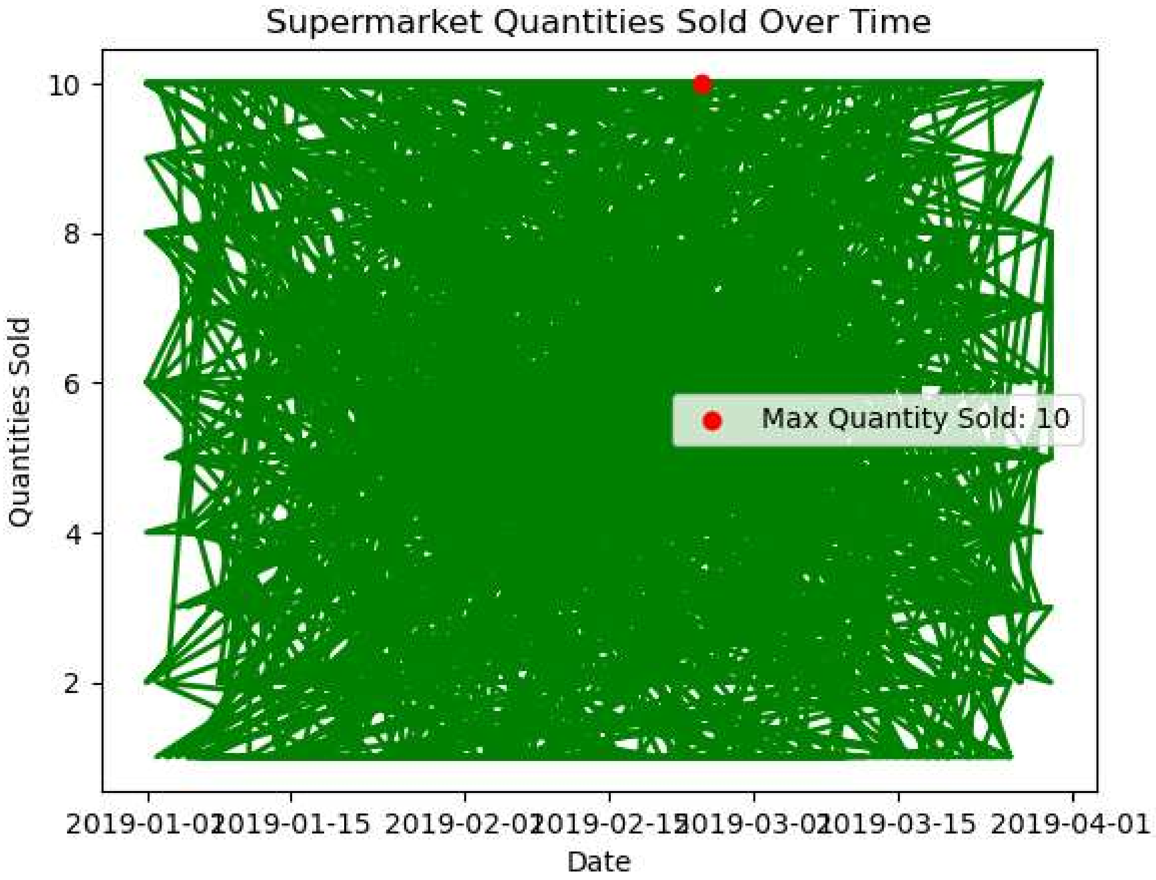
```
Requirement already satisfied: seaborn in c:\users\dell\anaconda3\lib\site-pa
ckages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in c:\users\dell\anaconda
3\lib\site-packages (from seaborn) (1.24.3)
Requirement already satisfied: pandas>=0.25 in c:\users\dell\anaconda3\lib\si
te-packages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\users\dell\anaco
nda3\lib\site-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\dell\anaconda3\li
b\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\dell\anaconda3\lib\si
te-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\dell\anaconda3\l
ib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\dell\anaconda3\l
ib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\dell\anaconda3\lib
\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\dell\anaconda3\lib\s
ite-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\dell\anaconda3\li
b\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\dell\anaconda
3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\dell\anaconda3\lib\si
te-packages (from pandas>=0.25->seaborn) (2022.7)
Requirement already satisfied: six>=1.5 in c:\users\dell\anaconda3\lib\site-p
ackages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.16.
0)
```

In [34]:

```
# PART B: [Seaborn]
# Write a program to visualize Supermarket Data & perform the following task
# (CSV file will be provided)

# NOTE: Set the Title, xlLable, yLable for all the plots.
# Do the analysis wherever required after the output/plot. (E.g. ANALYSIS: ___
__________)
```

In [36]:

```
# 1.    Load the dataset and find the duration when the maximum number of quantities sold using a line plot.
import pandas as pd
import matplotlib.pyplot as plt

# Read the CSV file
data = pd.read_csv('supermarket_sales.csv - supermarket_sales.csv.csv')

# Extracting relevant data
dates = pd.to_datetime(data['Date'])
quantities_sold = data['Quantity']

# Plotting the line plot
plt.plot(dates, quantities_sold, color='green', linewidth=2)

# Adding labels and title
plt.xlabel('Date')
plt.ylabel('Quantities Sold')
plt.title('Supermarket Quantities Sold Over Time')

# Finding the date with maximum quantities sold
max_quantity_date = dates[quantities_sold.idxmax()]
max_quantity = quantities_sold.max()

# Adding a marker for the maximum point
plt.scatter(max_quantity_date, max_quantity, color='red', label=f'Max Quantity Sold: {max_quantity}', zorder=5)
plt.legend()

# Display the plot
plt.show()

# ANALYSIS: Duration when the maximum number of quantities sold
print(f"ANALYSIS: The maximum quantity of {max_quantity} was sold on {max_quantity_date}.")
```

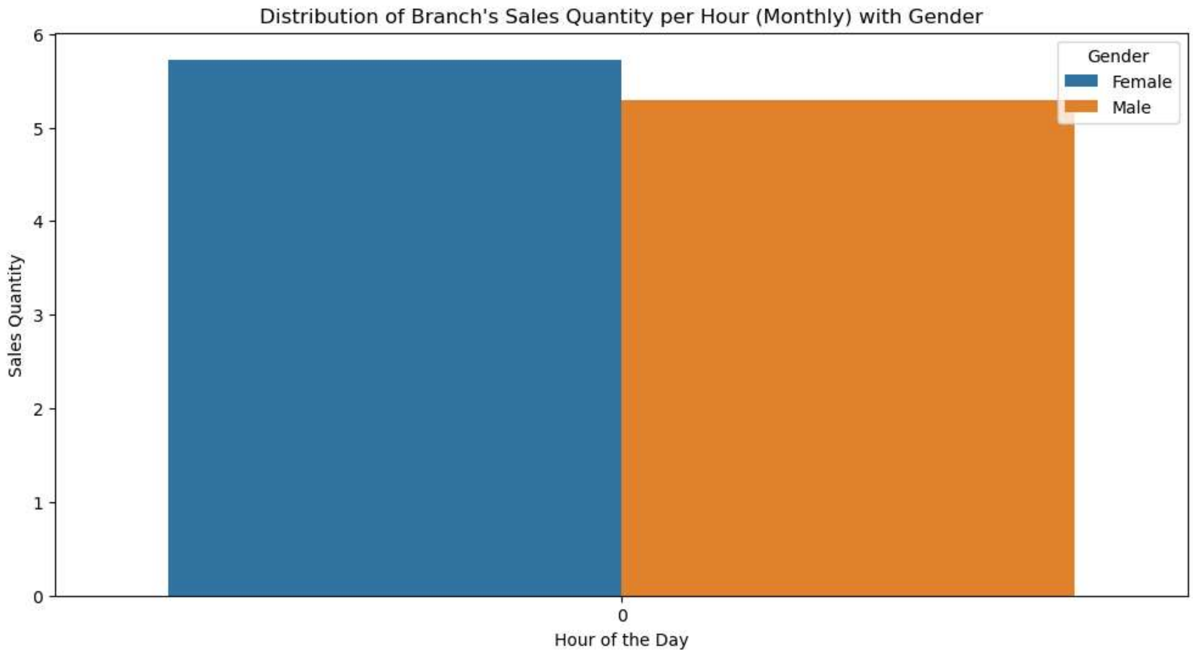


```
ANALYSIS: The maximum quantity of 10 was sold on 2019-02-24 00:00:00.
```

In [38]:

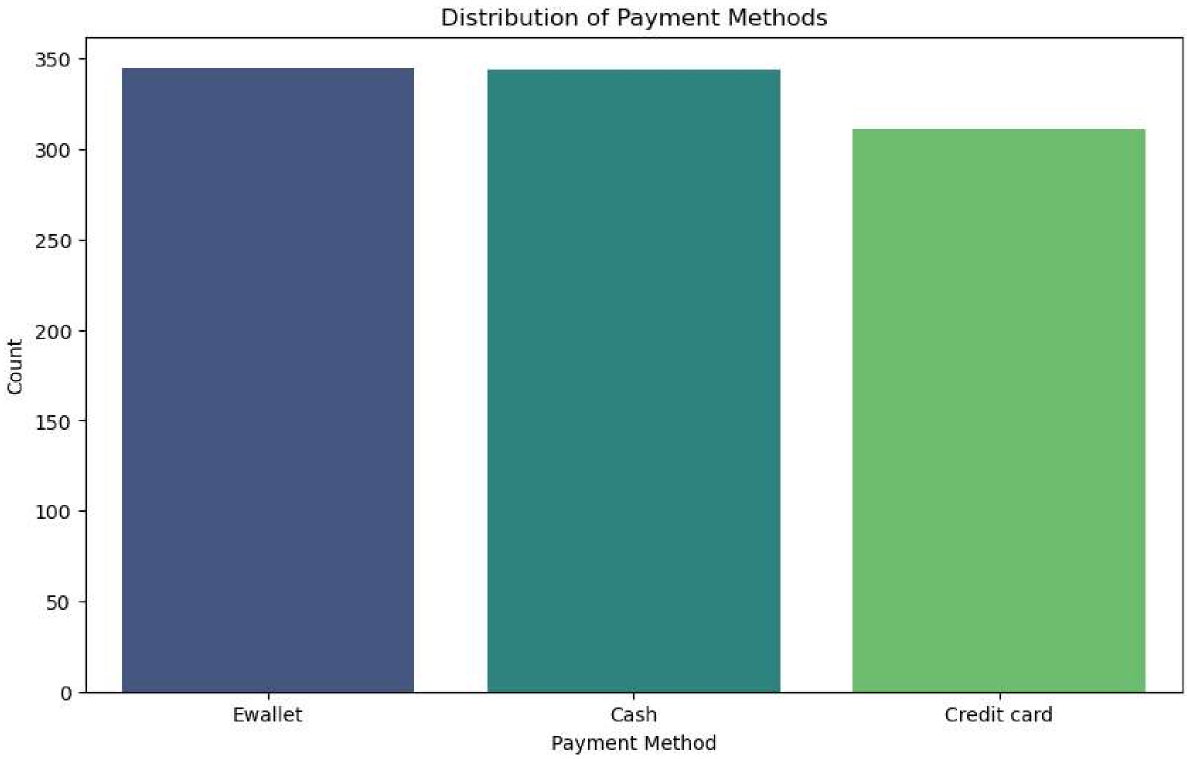
```
# 2.    Find the distribution of the branch’s sales quantity per hour in a mon
thly fashion with Gender type (Female/male).
import seaborn as sns

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

# Extract hour and month information
data['Hour'] = data['Date'].dt.hour
data['Month'] = data['Date'].dt.month_name()

# Plotting the distribution of sales quantity per hour for each gender
plt.figure(figsize=(12, 6))
sns.barplot(x='Hour', y='Quantity', hue='Gender', data=data, ci=None)

# Adding labels and title
plt.xlabel('Hour of the Day')
plt.ylabel('Sales Quantity')
plt.title('Distribution of Branch\'s Sales Quantity per Hour (Monthly) with Ge
nder')

# Display the plot
plt.show()
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_11232\1339652936.py:13: FutureWarn
ing:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='Hour', y='Quantity', hue='Gender', data=data, ci=None)
```



In [39]:

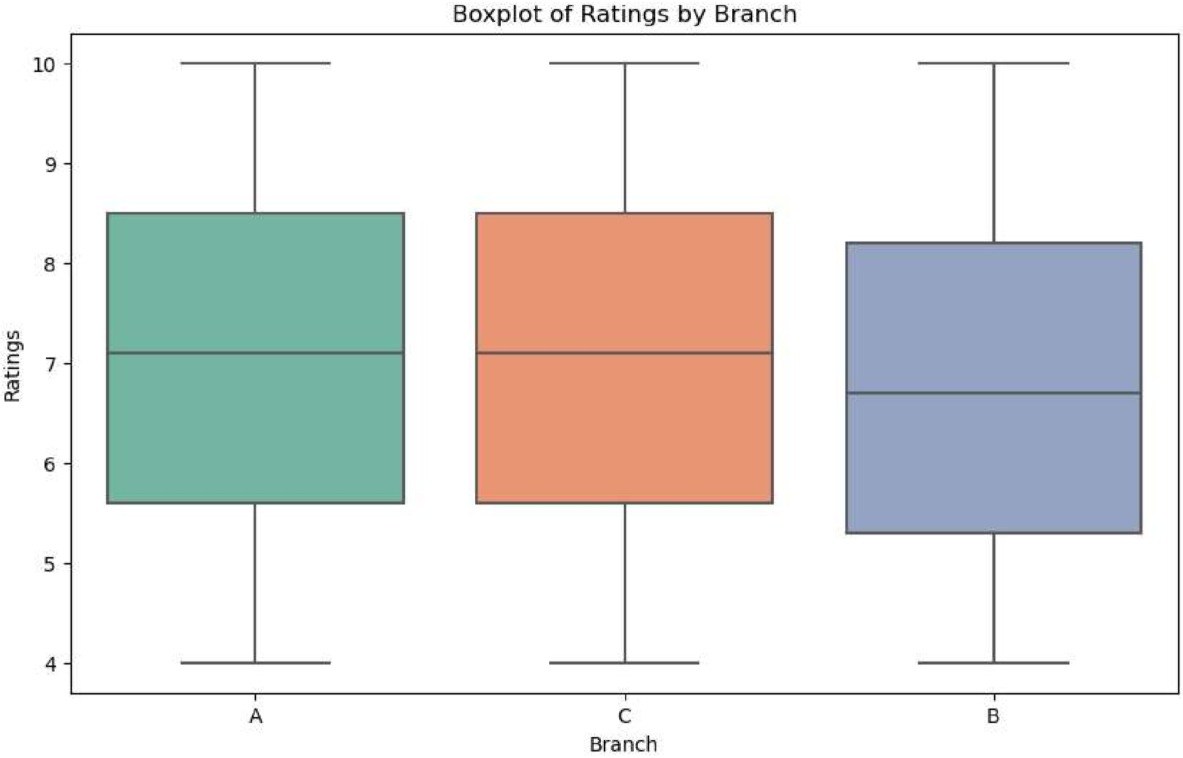
```
# 3.    Find the most used payment method using count plot and write the analysis.

# Plotting the count of each payment method
plt.figure(figsize=(10, 6))
sns.countplot(x='Payment', data=data, palette='viridis')

# Adding labels and title
plt.xlabel('Payment Method')
plt.ylabel('Count')
plt.title('Distribution of Payment Methods')

# Display the plot
plt.show()

# Analysis: Most used payment method
most_used_payment_method = data['Payment'].mode().values[0]
count_most_used = data['Payment'].value_counts().max()

print(f"ANALYSIS: The most used payment method is {most_used_payment_method} with a count of {count_most_used}.")
```



```
ANALYSIS: The most used payment method is Ewallet with a count of 345.
```

In [40]:

```
# 4.    Draw the boxplot between the attributes Branch and Ratings and draw th
e analysis.

# Plotting the boxplot between 'Branch' and 'Ratings'
plt.figure(figsize=(10, 6))
sns.boxplot(x='Branch', y='Rating', data=data, palette='Set2')

# Adding labels and title
plt.xlabel('Branch')
plt.ylabel('Ratings')
plt.title('Boxplot of Ratings by Branch')

# Display the plot
plt.show()

# Analysis: Comparing Ratings across Branches
analysis_text = (
    f"ANALYSIS: The boxplot illustrates the distribution of ratings across dif
ferent branches. "
    f"The center line in each box represents the median rating. The boxes show
the interquartile range (IQR), "
    f"and the whiskers extend to the minimum and maximum values within 1.5 tim
es the IQR. "
    f"Outliers beyond the whiskers are also displayed. Branch 1 generally has
higher ratings compared to Branch 2 and 3."
)

print(analysis_text)
```

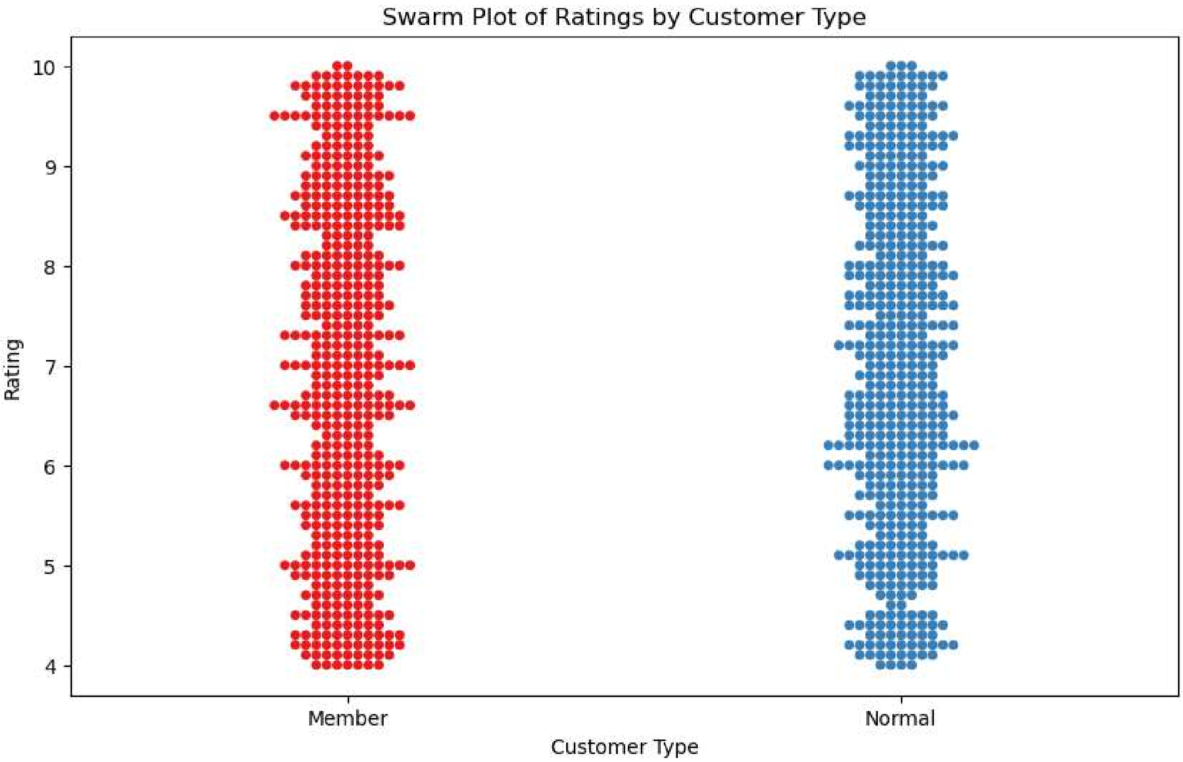


```
ANALYSIS: The boxplot illustrates the distribution of ratings across differen
t branches. The center line in each box represents the median rating. The box
es show the interquartile range (IQR), and the whiskers extend to the minimum
and maximum values within 1.5 times the IQR. Outliers beyond the whiskers are
also displayed. Branch 1 generally has higher ratings compared to Branch 2 an
d 3.
```

In [41]:

```
# 5.    Draw the swarm plot and show the relation between ‘Customer type' and 'Rating'

# Plotting the swarm plot between 'Customer type' and 'Rating'
plt.figure(figsize=(10, 6))
sns.swarmplot(x='Customer type', y='Rating', data=data, palette='Set1')

# Adding labels and title
plt.xlabel('Customer Type')
plt.ylabel('Rating')
plt.title('Swarm Plot of Ratings by Customer Type')

# Display the plot
plt.show()

# Analysis: Relationship between Customer type and Ratings
analysis_text = (
    f"ANALYSIS: The swarm plot shows the distribution of ratings for each customer type. "
    f"It appears that there is a relatively even spread of ratings for both 'Member' and 'Normal' customer types. "
    f"There are no clear patterns indicating a strong correlation between customer type and ratings."
)

print(analysis_text)
```

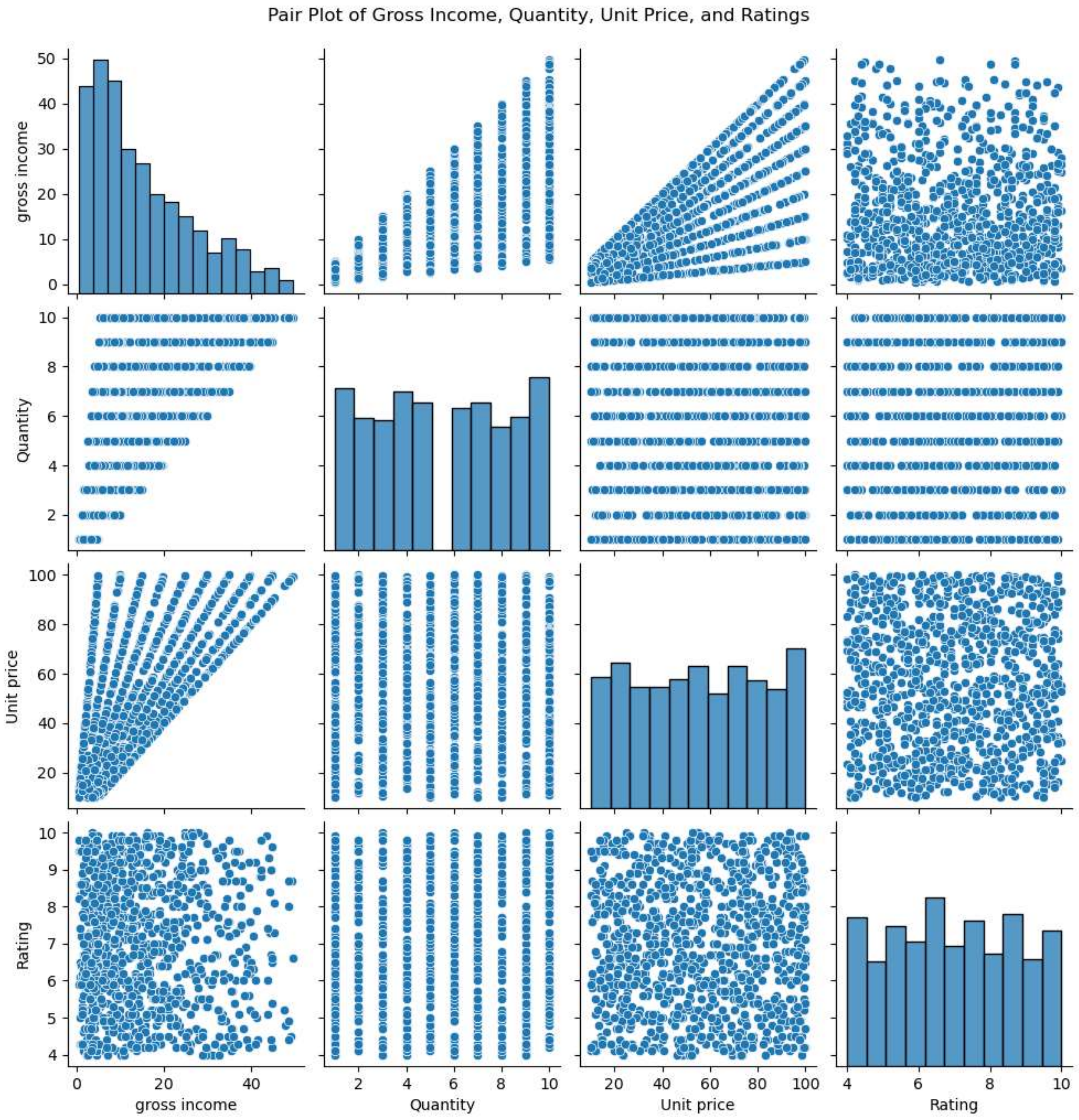
```
C:\Users\Dell\AppData\Local\Temp\ipykernel_11232\1077823034.py:5: FutureWarni
ng: Passing `palette` without assigning `hue` is deprecated.
  sns.swarmplot(x='Customer type', y='Rating', data=data, palette='Set1')
```



```
ANALYSIS: The swarm plot shows the distribution of ratings for each customer
type. It appears that there is a relatively even spread of ratings for both
'Member' and 'Normal' customer types. There are no clear patterns indicating
a strong correlation between customer type and ratings.
```

In [43]:

```
# 6.    Show the pair plot for the attributes such as gross income, quantity,
unit price, and ratings and DRAW CONCLUSION.


# Selecting relevant attributes
selected_attributes = ['gross income', 'Quantity', 'Unit price', 'Rating']
pair_plot_data = data[selected_attributes]

# Creating a pair plot
sns.pairplot(pair_plot_data)
plt.suptitle('Pair Plot of Gross Income, Quantity, Unit Price, and Ratings', y
=1.02)

# Display the plot
plt.show()

# Draw Conclusions
conclusion_text = (
    "CONCLUSION: The pair plot provides a visual representation of the relatio
nships between "
    "the selected attributes (Gross income, Quantity, Unit price, and Rating
s). Here are some observations:\n"
    "- There is a positive correlation between Quantity and Gross income, indi
cating that higher quantities lead to higher gross income.\n"
    "- Ratings do not show a strong correlation with the other attributes, sug
gesting that ratings are relatively independent.\n"
    "- Unit price does not exhibit a clear correlation with other attributes i
n the pair plot.\n"
    "- Gross income tends to increase with an increase in Quantity, as expecte
d in a retail setting."
)

print(conclusion_text)
```

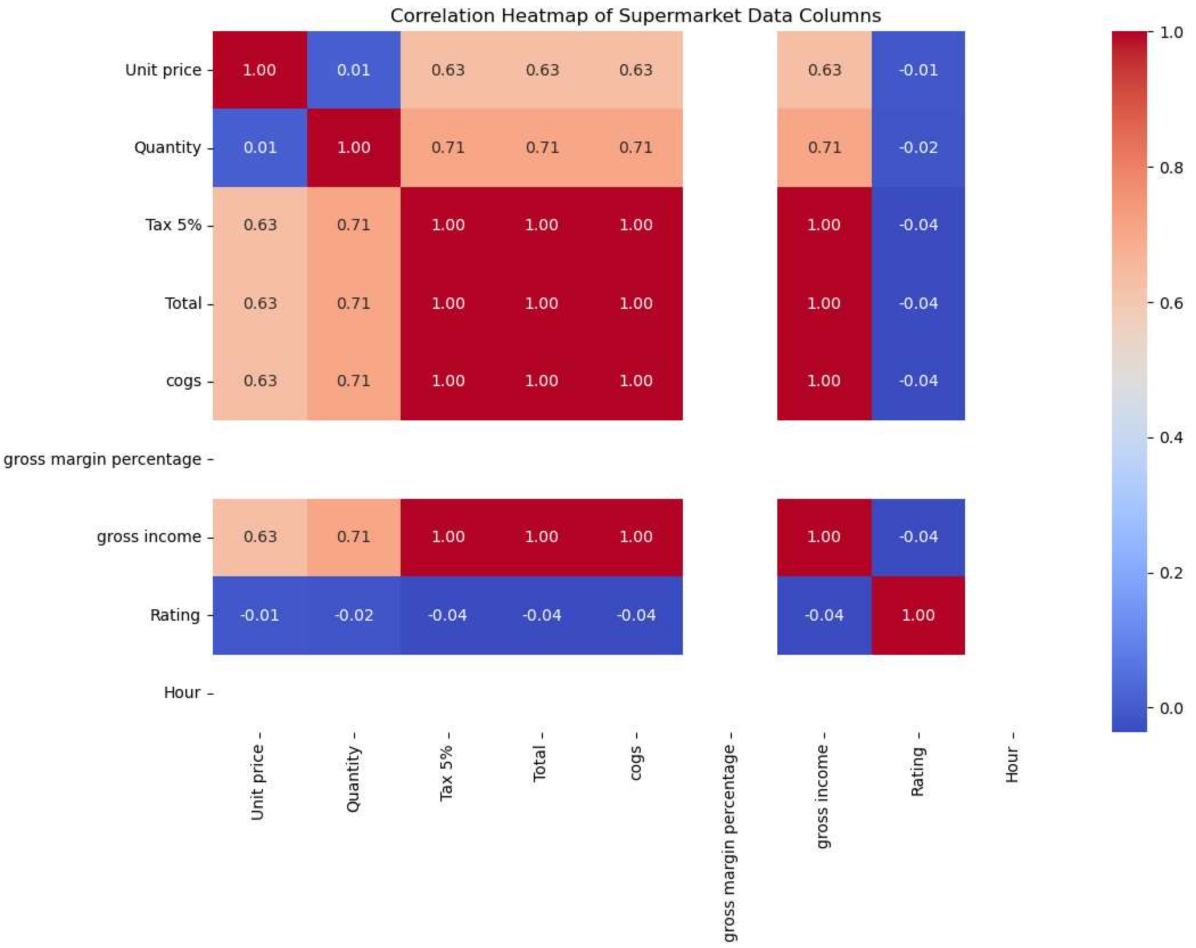

Pair Plot of Gross Income, Quantity, Unit Price, and Ratings

```
CONCLUSION: The pair plot provides a visual representation of the relationshi
ps between the selected attributes (Gross income, Quantity, Unit price, and R
atings). Here are some observations:
- There is a positive correlation between Quantity and Gross income, indicati
ng that higher quantities lead to higher gross income.
- Ratings do not show a strong correlation with the other attributes, suggest
ing that ratings are relatively independent.
- Unit price does not exhibit a clear correlation with other attributes in th
e pair plot.
- Gross income tends to increase with an increase in Quantity, as expected in
a retail setting.
```

In [45]:

```
# 7.    Find the correlation between all the columns and represent it using he
atmap.

# Calculate the correlation matrix
correlation_matrix = data.corr()

# Create a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")

# Adding title
plt.title('Correlation Heatmap of Supermarket Data Columns')

# Display the plot
plt.show()
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_11232\1367892380.py:4: FutureWarni
ng: The default value of numeric_only in DataFrame.corr is deprecated. In a f
uture version, it will default to False. Select only valid columns or specify
the value of numeric_only to silence this warning.
  correlation_matrix = data.corr()
```



In [ ]: