Group Work

BSD 3203 Programming for Data Science.

BSc. Software Development.

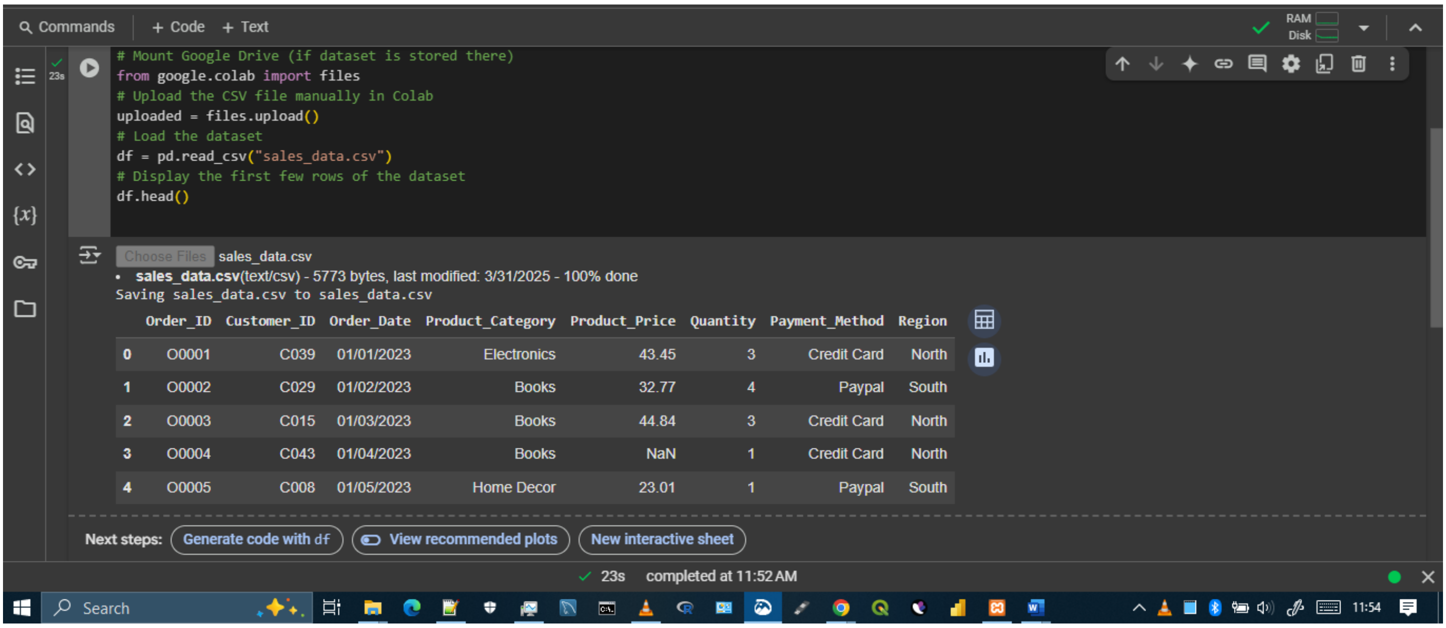
ASSIGNMENT 3

Group Members

1. 20/04860 Nzioka Emmanuel Munyao
2. 21/07757 Swalehe Hussein
3. 21/06201 Pauline Nafuna
4. 21/06600 Eunice Mwae
5. 21/06709 Charity Mutunga
6. 21/03241 G. Moses Githaiga
7. 21/06439 Marubi N Richard
8. 21/04983 Rotich Caren

NB: GOOGLE COLLAB NOTEBOOK WAS USED TO HANDLE THIS ASSIGNMENT AVAILABLE ON THIS LINK.

The first step is to import the necessary libraries as follows;



This Python script imports essential libraries for data analysis (pandas, numpy), visualization (matplotlib, seaborn), and statistical calculations (scipy.stats). It includes a function to upload a CSV file manually in Google Colab using files.upload(), allowing users to select and upload sales_data.csv. Once uploaded, the dataset is loaded into a Pandas DataFrame using pd.read_csv(), and the first few rows are displayed using df.head() to provide an initial preview of the dataset. This setup ensures that the data is ready for further exploratory data analysis (EDA).

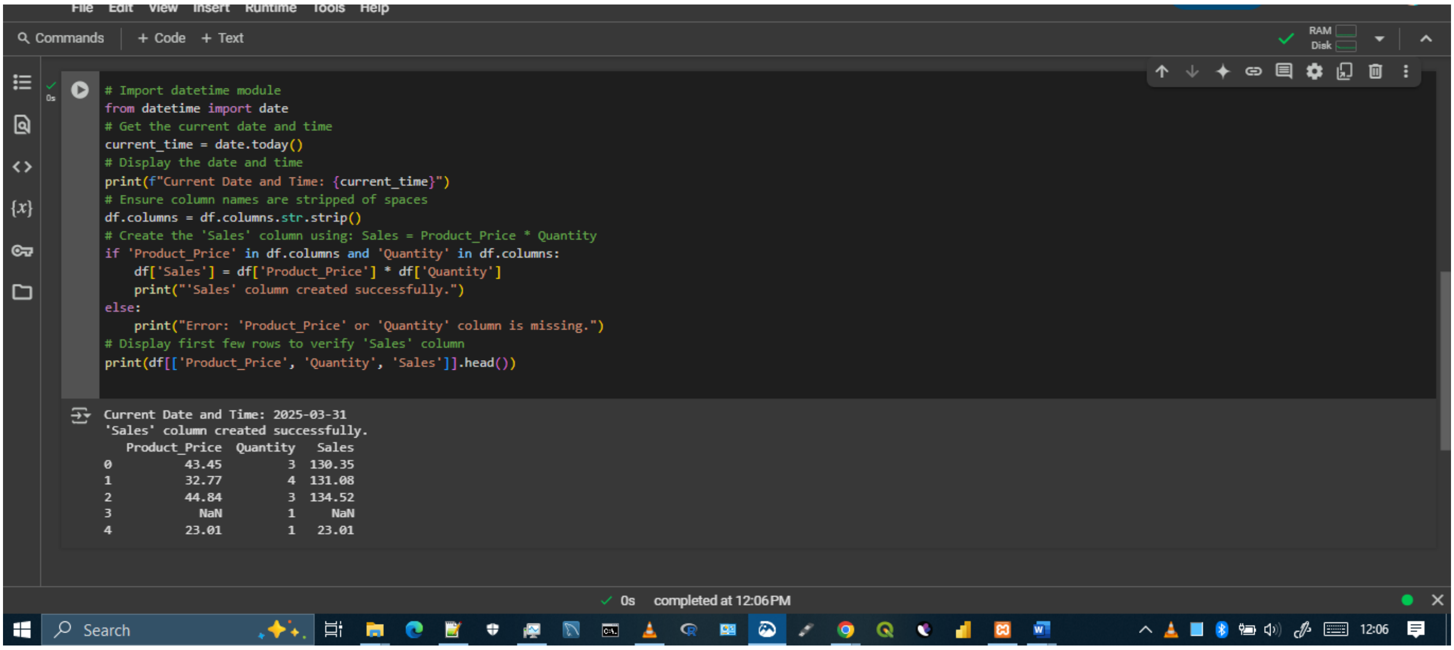
Code Used

```
# Import required libraries
import pandas as pd  # For data handling
import numpy as np  # For numerical computations
import matplotlib.pyplot as plt  # For visualization
import seaborn as sns  # Advanced visualization
import scipy.stats as stats  # For statistical calculations
# Mount Google Drive (if dataset is stored there)
from google.colab import files
# Upload the CSV file manually in Colab
uploaded = files.upload()
```

```
# Load the dataset
df = pd.read_csv("sales_data.csv")
# Display the first few rows of the dataset
df.head()
```

Question 1: Univariate Non-Graphical EDA

First create the sales column;



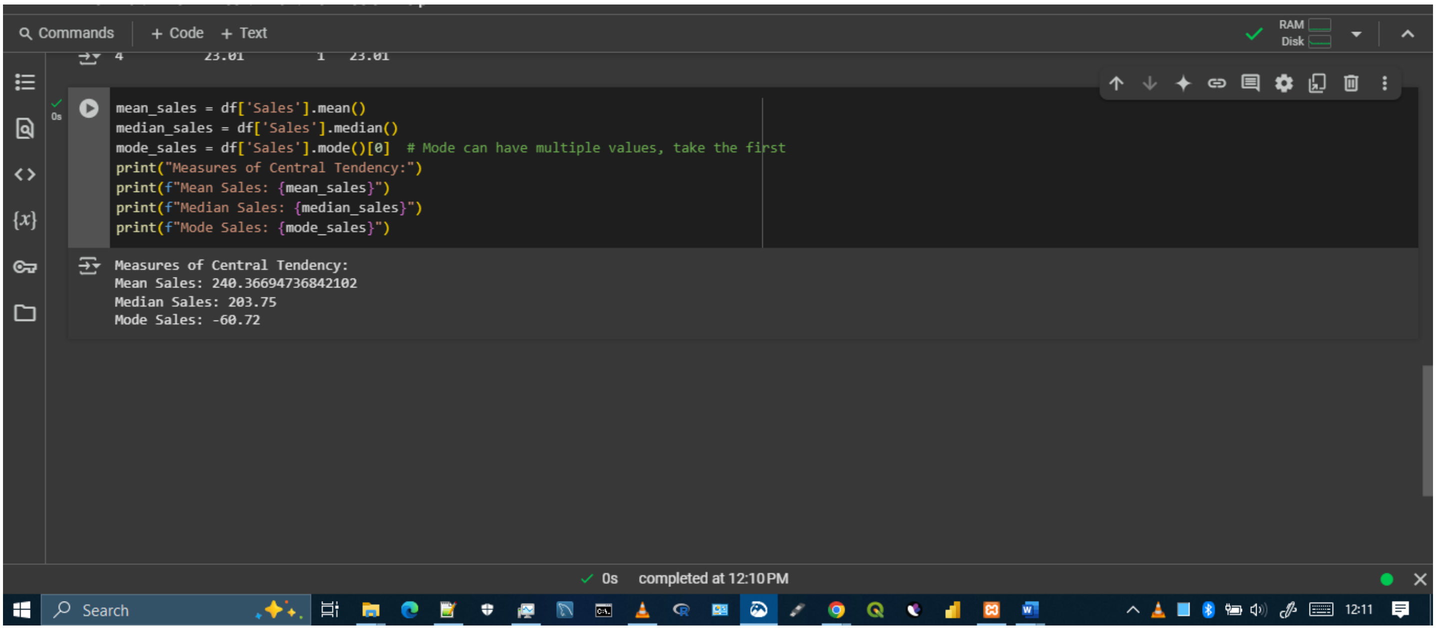
This part ensures that column names are stripped of any leading or trailing spaces using df.columns.str.strip() to avoid errors when accessing them. It then checks if the Product_Price and Quantity columns exist in the dataset. If both are present, it creates a new Sales column by multiplying Product_Price by Quantity and prints a success message. If either column is missing, it prints an error message. Finally, it displays the first few rows of Product_Price, Quantity, and the newly created Sales column to verify the computation.

Code Used

```
# Import datetime module
from datetime import date
# Get the current date and time
current_time = date.today()
# Display the date and time
print(f"Current Date and Time: {current_time}")
# Ensure column names are stripped of spaces
```

```
df.columns = df.columns.str.strip()
# Create the 'Sales' column using: Sales = Product_Price * Quantity
if 'Product_Price' in df.columns and 'Quantity' in df.columns:
    df['Sales'] = df['Product_Price'] * df['Quantity']
    print("'Sales' column created successfully.")
else:
    print("Error: 'Product_Price' or 'Quantity' column is missing.")
# Display first few rows to verify 'Sales' column
print(df[['Product_Price', 'Quantity', 'Sales']].head())
```

1a) Measures of Central Tendency

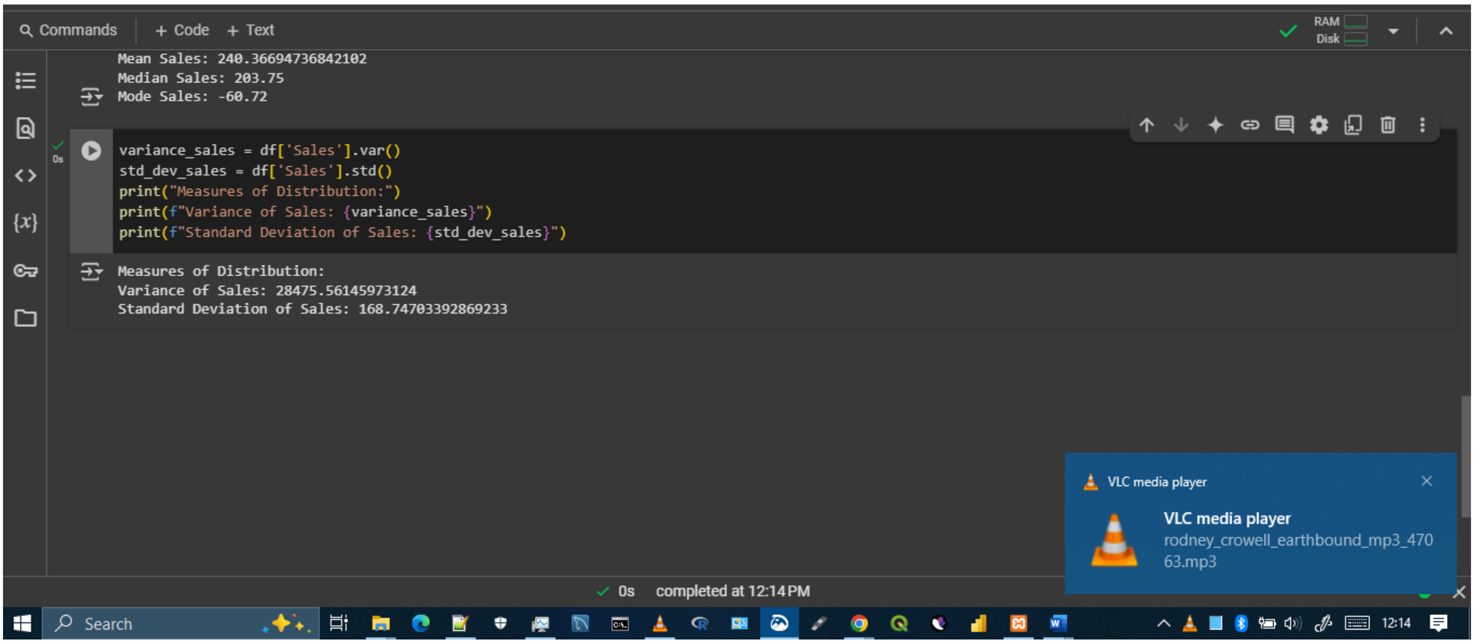


This part calculates and displays the measures of central tendency for the Sales column in the dataset. The mean sales (average) is approximately 240.37, indicating the typical sales value across transactions. The median sales (middle value when sorted) is 203.75, suggesting that half of the sales are below this value and half are above. The mode sales (most frequently occurring value) is -60.72, which is unusual, possibly indicating data entry errors or returns/refunds recorded as negative sales. This requires further investigation to ensure data accuracy.

Code Used

```
mean_sales = df['Sales'].mean()
median_sales = df['Sales'].median()
mode_sales = df['Sales'].mode()[0]  # Mode can have multiple values, take the first
print("Measures of Central Tendency:")
print(f"Mean Sales: {mean_sales}")
print(f"Median Sales: {median_sales}")
print(f"Mode Sales: {mode_sales}")
```

1b) Measures of Distribution

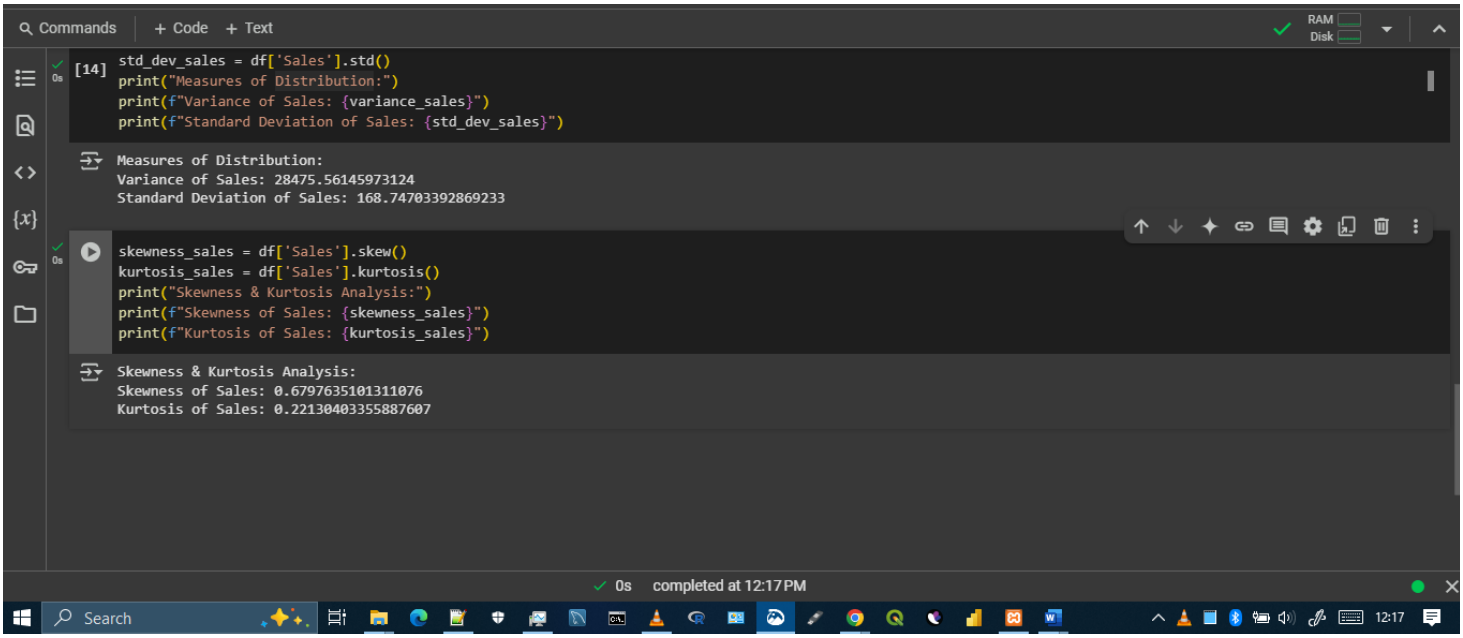


This part calculates and displays the **measures of distribution** for the Sales column, specifically **variance** and **standard deviation**. The **variance** is **28,475.56**, which indicates the average squared deviation from the mean, showing a high spread in sales values. The **standard deviation** is **168.75**, which measures the average deviation from the mean in the same unit as sales. A high standard deviation suggests that sales values vary significantly, indicating potential fluctuations in transaction amounts. This could be due to varying product prices, seasonal trends, or customer purchasing behavior.

Code Used

```
variance_sales = df['Sales'].var()
std_dev_sales = df['Sales'].std()
print("Measures of Distribution:")
print(f"Variance of Sales: {variance_sales}")
print(f"Standard Deviation of Sales: {std_dev_sales}")
```

1c) Skewness & Kurtosis



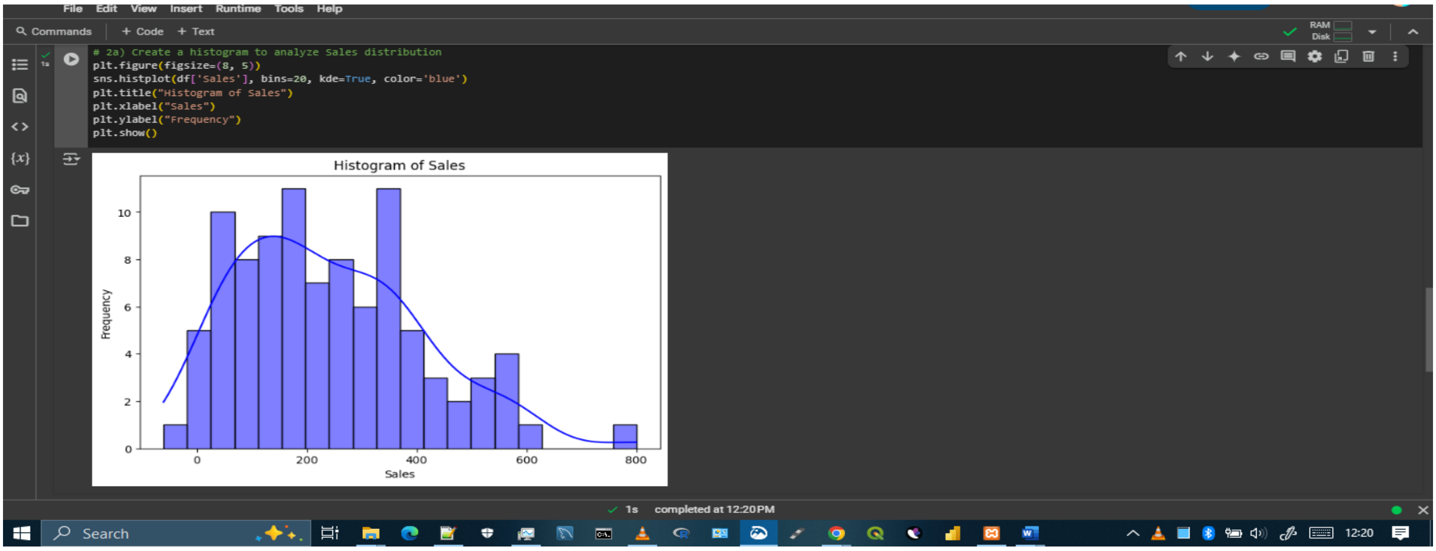
This part calculates skewness and kurtosis for the Sales column to analyze its distribution shape. The skewness value of 0.68 indicates a slightly right-skewed (positively skewed) distribution, meaning that most sales values are concentrated on the lower end, with a few higher sales values pulling the distribution to the right. The kurtosis value of 0.22 suggests a distribution that is approximately normal but slightly platykurtic (flatter than a normal distribution), meaning it has fewer extreme values or outliers than a typical bell-shaped curve. This analysis helps understand the data spread and potential anomalies in sales trends.

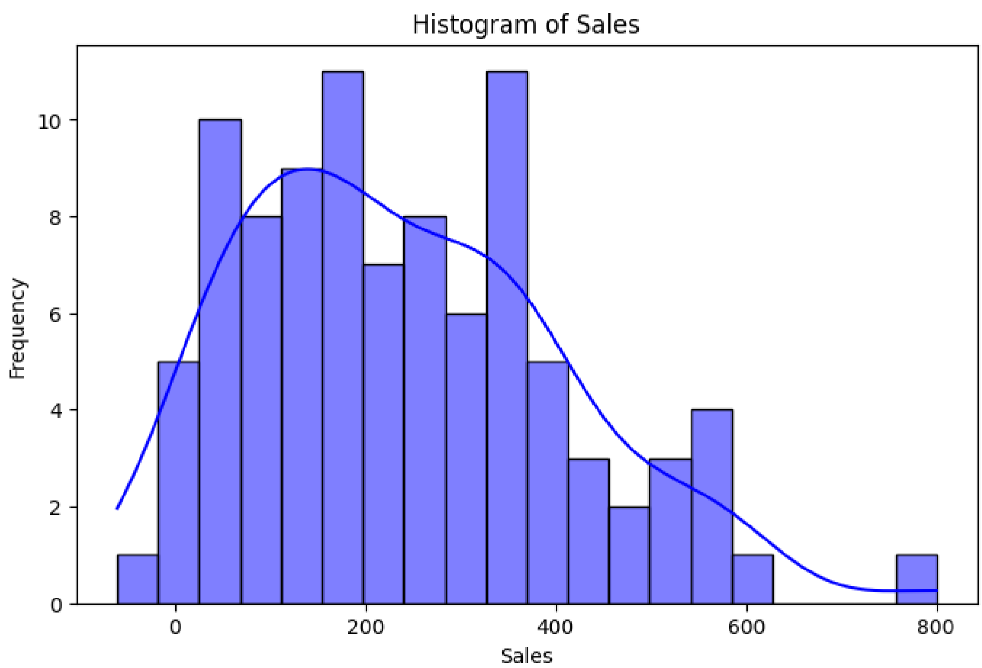
Code Used

```
skewness_sales = df['Sales'].skew()
kurtosis_sales = df['Sales'].kurtosis()
print("Skewness & Kurtosis Analysis:")
print(f"Skewness of Sales: {skewness_sales}")
print(f"Kurtosis of Sales: {kurtosis_sales}")
```

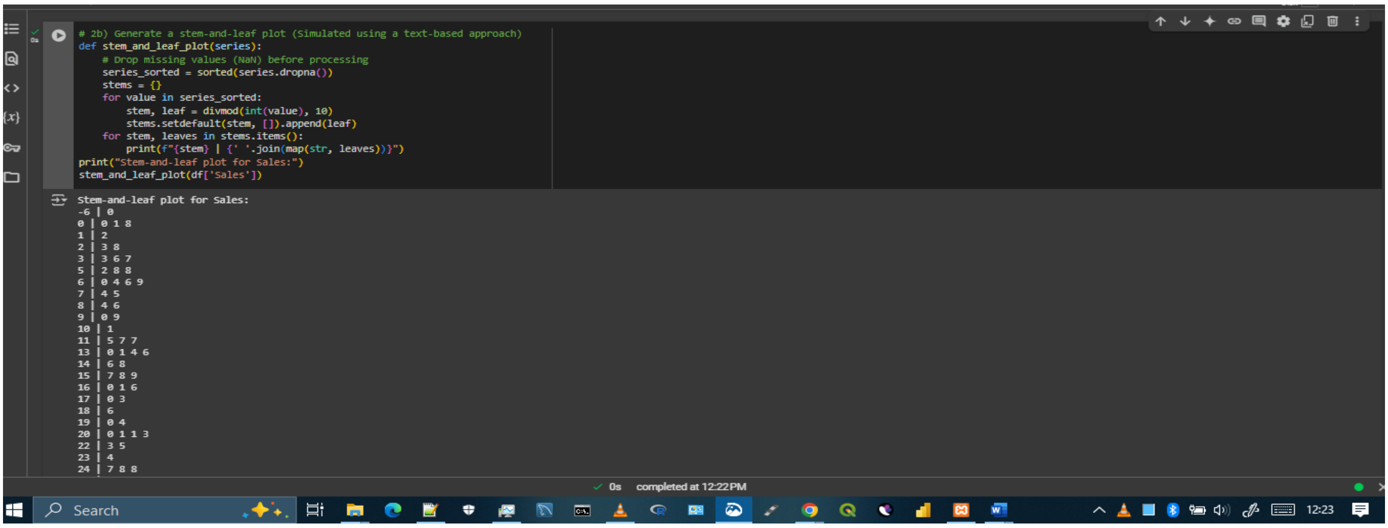
Question 2: Univariate Graphical EDA

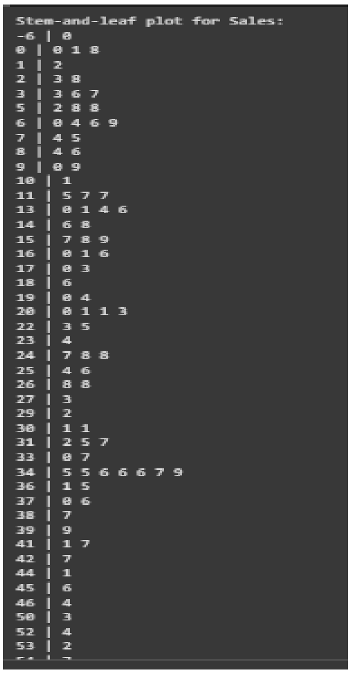
2a) Create a histogram to analyze Sales distribution

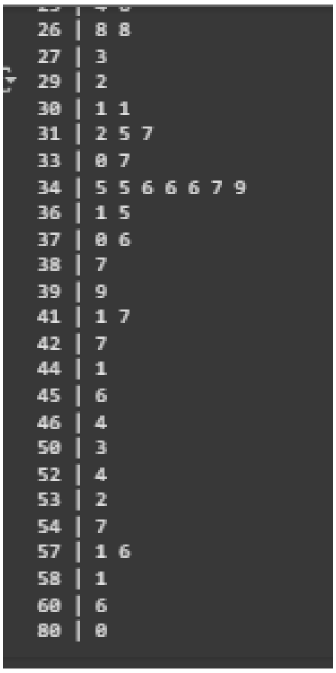




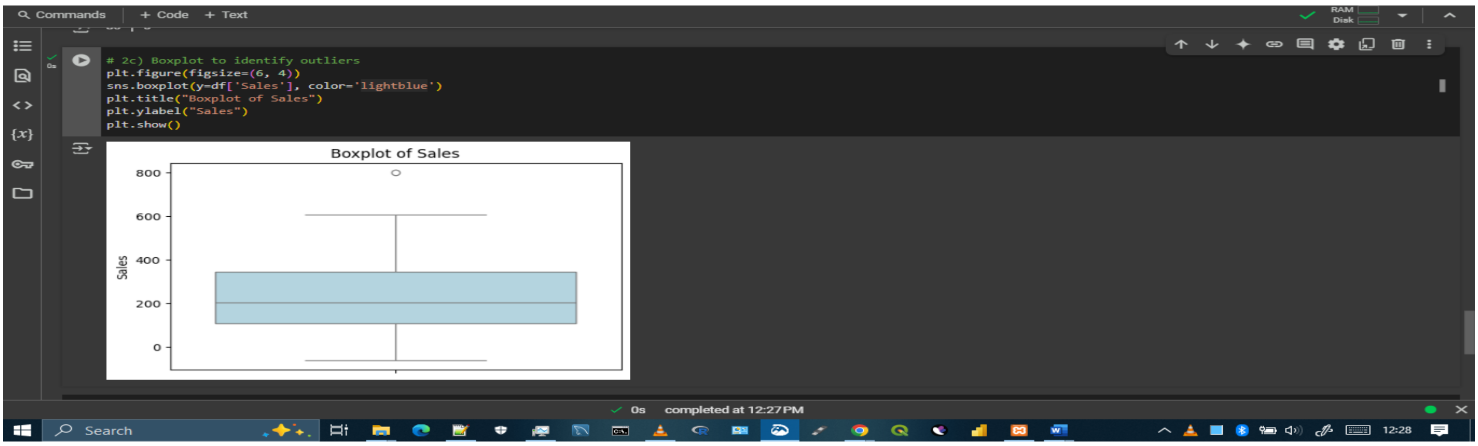
2b) Generate a stem-and-leaf plot (Simulated using a text-based approach)

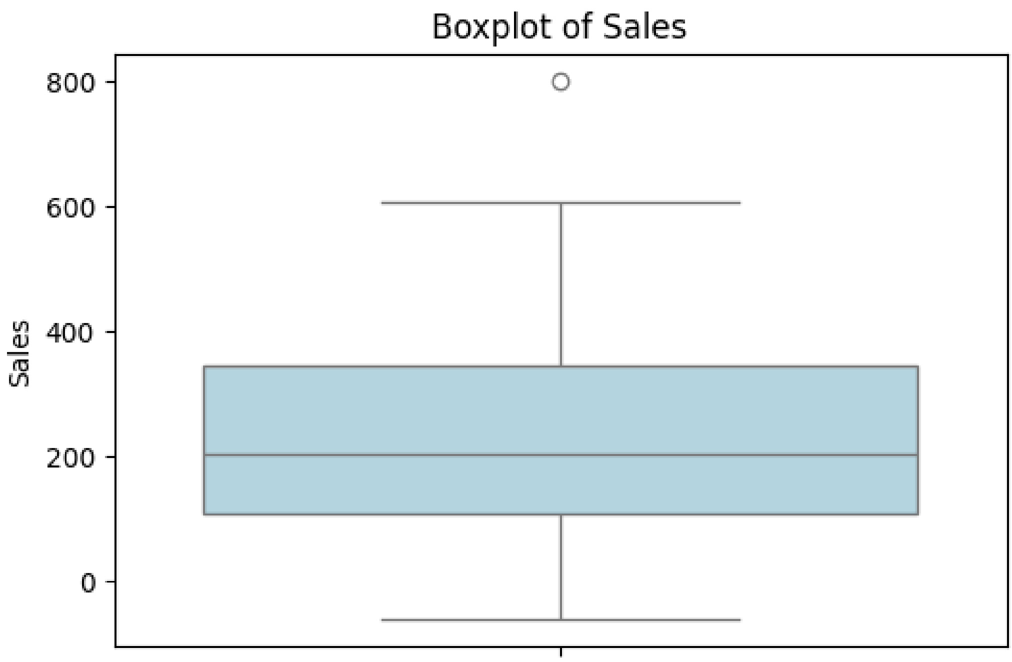




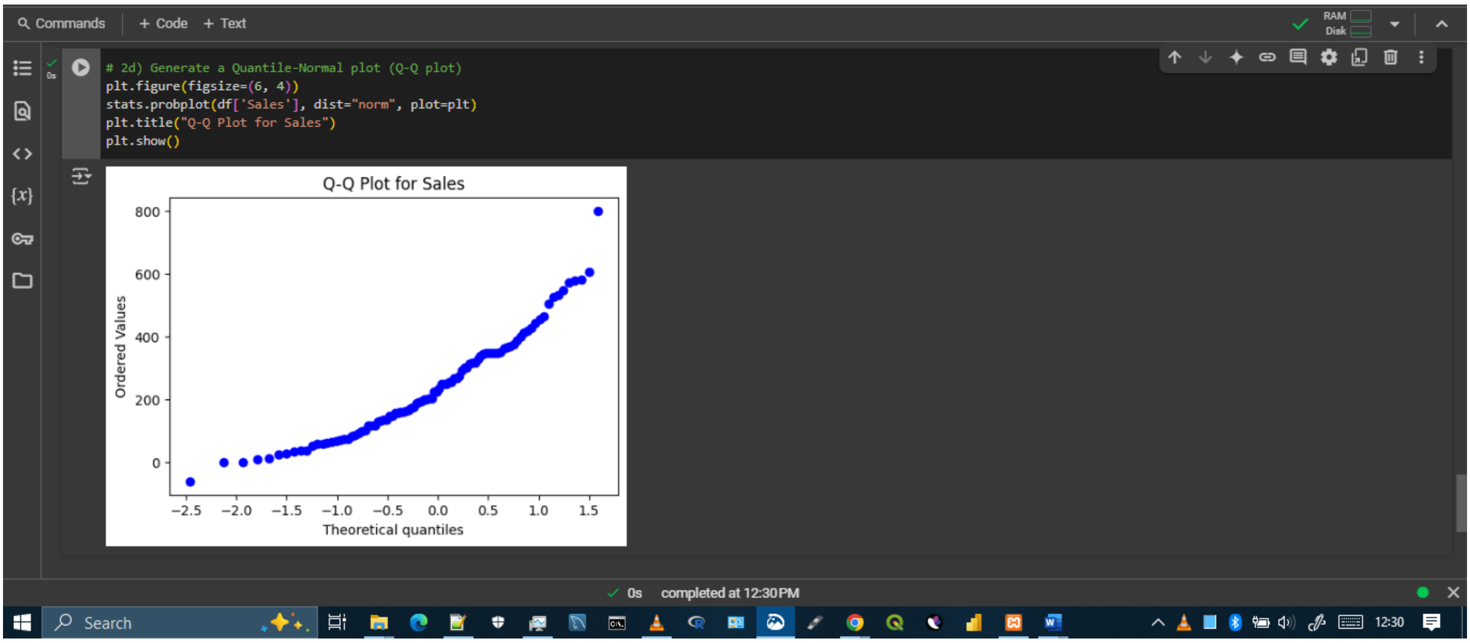


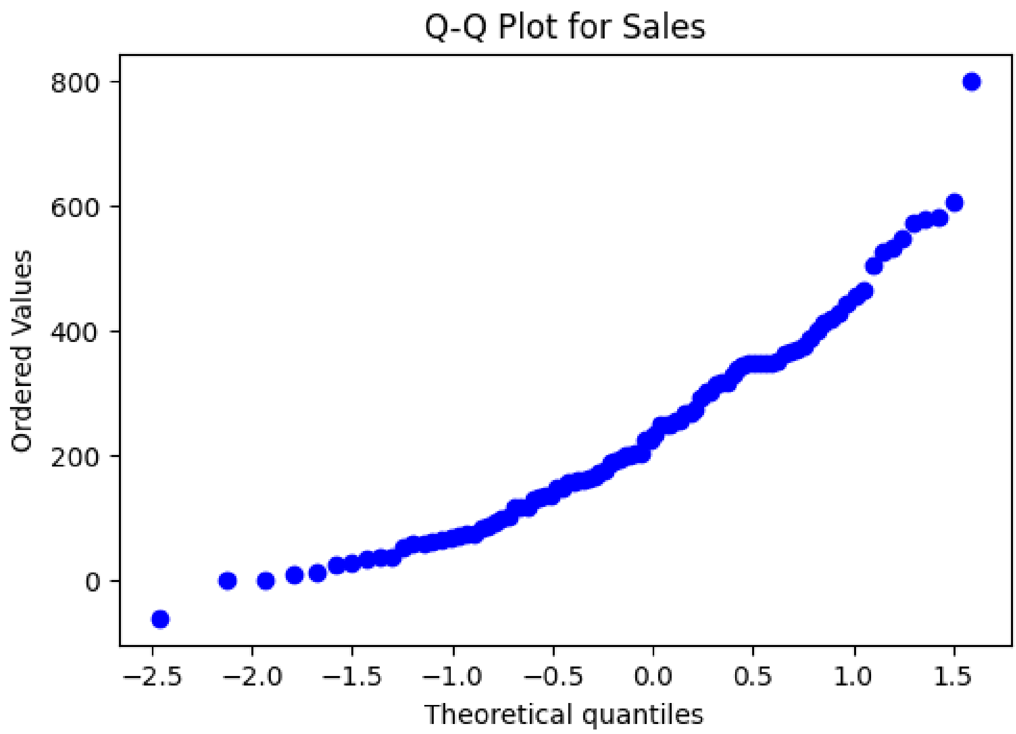
2c) Boxplot to identify outliers





2d) Generate a Quantile-Normal plot (Q-Q plot)





The provided visualizations and stem-and-leaf plot offer insights into the distribution of sales data. The histogram shows that most sales frequencies are concentrated around the lower values (600-800), suggesting a right-skewed distribution with fewer high-value sales. The boxplot confirms this skewness, with a long upper tail and potential outliers at higher sales values. The Q-Q plot further supports the non-normality of the data, as the points deviate significantly from the theoretical quantile line, especially at the higher end.

The stem-and-leaf plot provides detailed data points, revealing a wide range of sales values from -60 to 800. The majority of values are clustered at the lower end (e.g., 0-200), with fewer extreme values (e.g., above 500). The presence of negative values (e.g., -60) and very high values (e.g., 800) indicates potential data entry errors or exceptionally large transactions. Overall, the sales data is not normally distributed, exhibiting significant skewness and outliers, which may require transformation or non-parametric statistical methods for analysis.

Code Used for Number 2

```
# 2a) Create a histogram to analyze Sales distribution
plt.figure(figsize=(8, 5))
sns.histplot(df['Sales'], bins=20, kde=True, color='blue')
plt.title("Histogram of Sales")
plt.xlabel("Sales")
```

```
plt.ylabel("Frequency")
plt.show()

# 2b) Generate a stem-and-leaf plot (Simulated using a text-based approach)
def stem_and_leaf_plot(series):
    # Drop missing values (NaN) before processing
    series_sorted = sorted(series.dropna())
    stems = {}
    for value in series_sorted:
        stem, leaf = divmod(int(value), 10)
        stems.setdefault(stem, []).append(leaf)
    for stem, leaves in stems.items():
        print(f"{stem} | {' '.join(map(str, leaves))}")
print("Stem-and-leaf plot for Sales:")
stem_and_leaf_plot(df['Sales'])

# 2c) Boxplot to identify outliers
plt.figure(figsize=(6, 4))
sns.boxplot(y=df['Sales'], color='lightblue')
plt.title("Boxplot of Sales")
plt.ylabel("Sales")
plt.show()

# 2d) Generate a Quantile-Normal plot (Q-Q plot)
plt.figure(figsize=(6, 4))
stats.probplot(df['Sales'], dist="norm", plot=plt)
plt.title("Q-Q Plot for Sales")
plt.show()
```

Question 3: Multivariate EDA

3a) Cross-tabulation of Sales transactions by Region and Product Category

```
# 3a) Cross-tabulation of Sales transactions by Region and Product Category
cross_tab = pd.crosstab(df['Region'], df['Product_Category'])
print("Cross-tabulation of Sales by Region and Product Category:\n", cross_tab)
```

```
Cross-tabulation of Sales by Region and Product Category:
 Product_Category  Books  Clothing  Electronics  Home Decor
Region
East                  7         5            6           6
North                 5         6            7          11
South                 5         3            5           8
West                  7         5            5           4
```

```
# 3b) Calculate covariance and correlation between Sales and Quantity
```

0s completed at 12:52 PM

3b) Calculate covariance and correlation between Sales and Quantity

```
# 3b) Calculate covariance and correlation between Sales and Quantity
covariance = df[['Sales', 'Quantity']].cov()
correlation = df[['Sales', 'Quantity']].corr()

print("Covariance between Sales and Quantity:\n", covariance)
print("Correlation between Sales and Quantity:\n", correlation)
```

```
Covariance between Sales and Quantity:
                 Sales    Quantity
Sales     28475.56146  334.905660
Quantity    334.90566    6.539293
Correlation between Sales and Quantity:
              Sales  Quantity
Sales     1.000000  0.774528
Quantity  0.774528  1.000000
```

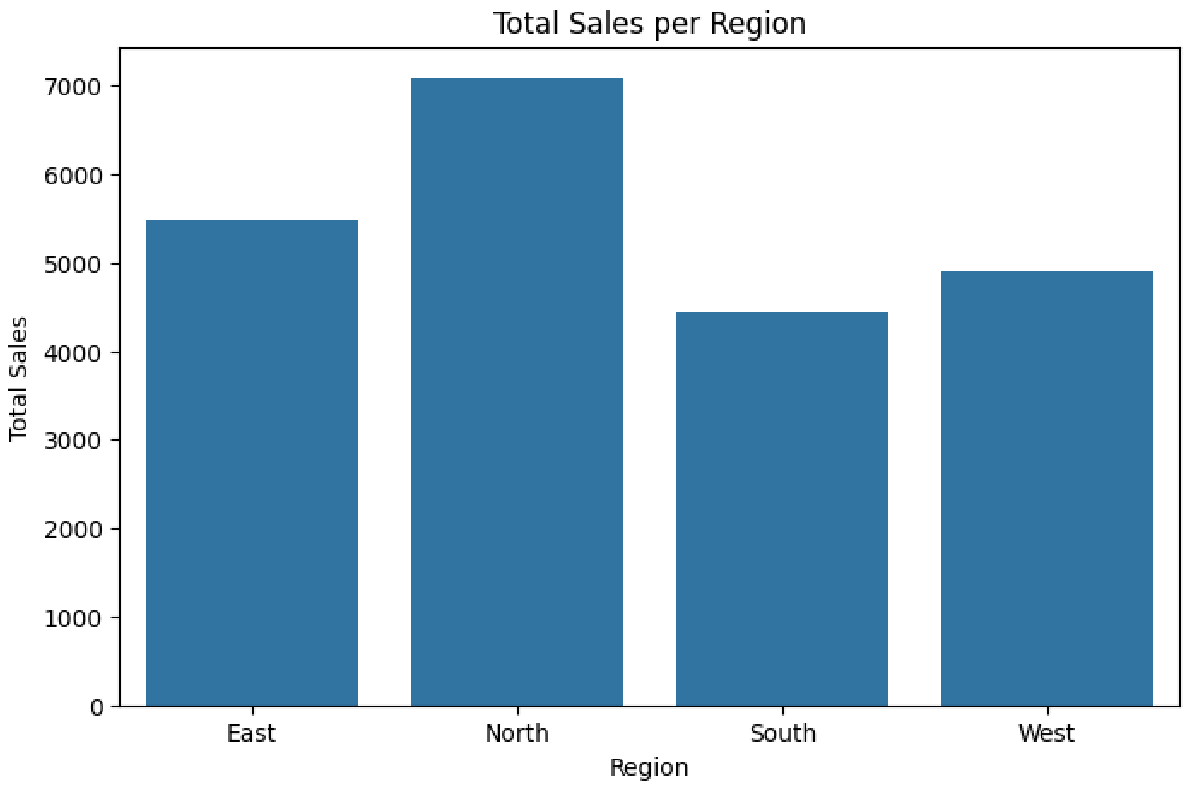
```
[26] # 3c) Visualization
```

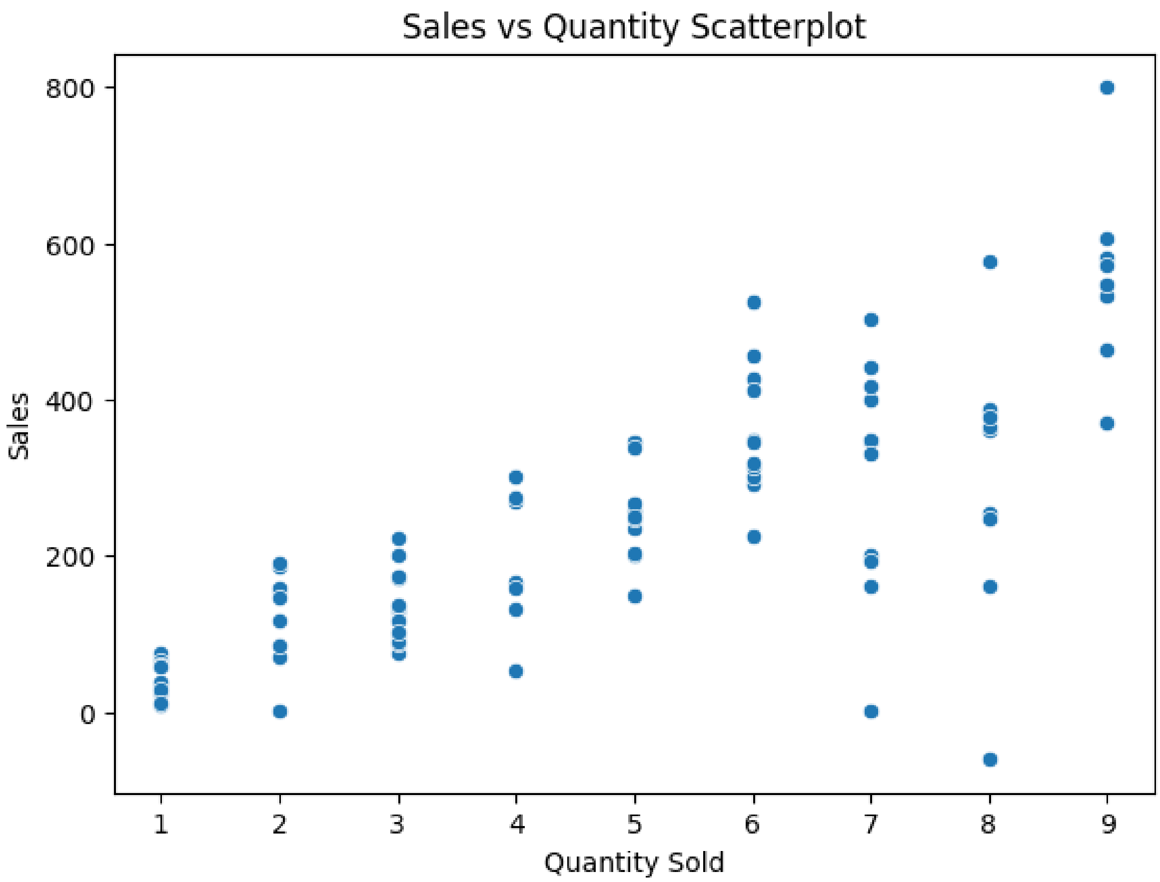
0s completed at 12:52 PM

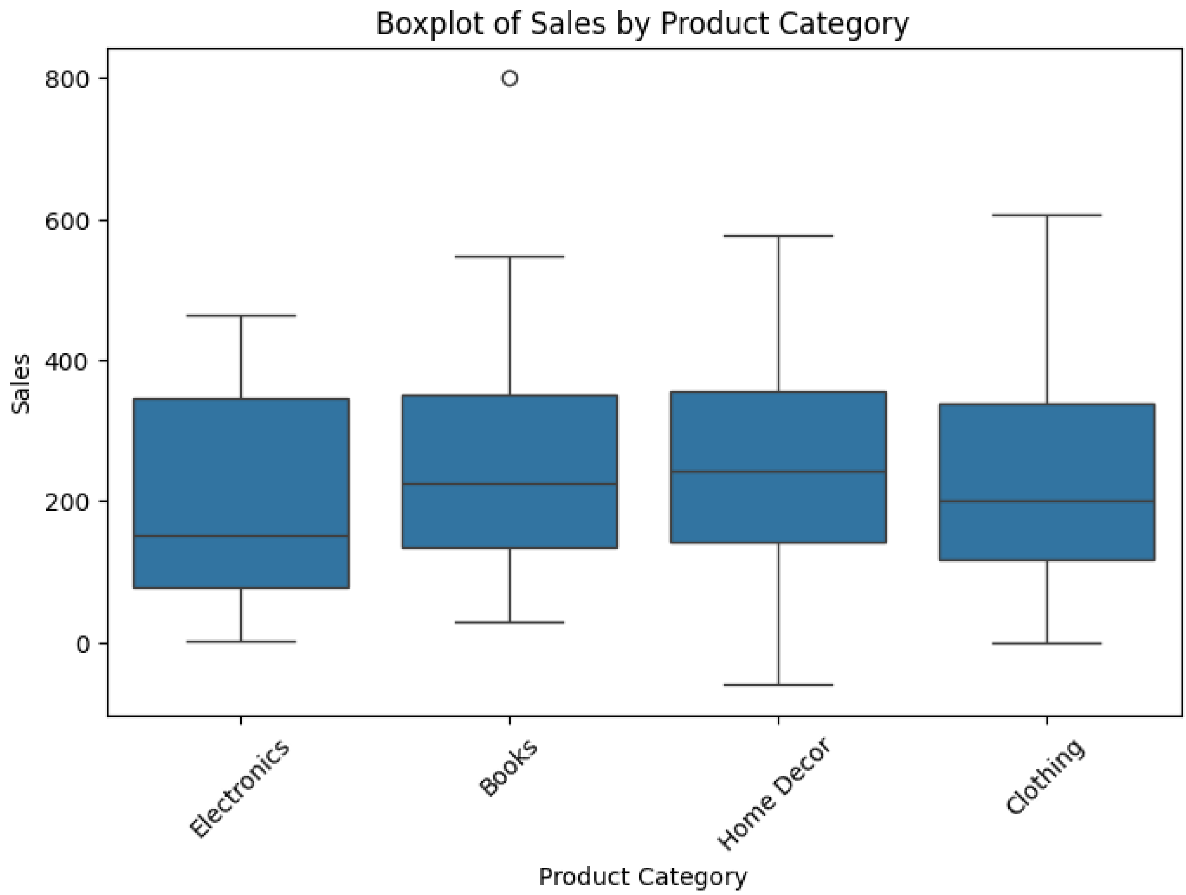
3c) Visualization

```
# 3c) Visualization

# Barplot of total Sales per Region
plt.figure(figsize=(8, 5))
sns.barplot(x=df.groupby("Region")["Sales"].sum().index, y=df.groupby("Region")["Sales"].sum().values)
plt.title("Total Sales per Region")
plt.xlabel("Region")
plt.ylabel("Total Sales")
plt.show()

# Scatterplot of Sales vs Quantity
plt.figure(figsize=(7, 5))
sns.scatterplot(x=df['Quantity'], y=df['Sales'])
plt.title("Sales vs Quantity Scatterplot")
plt.xlabel("Quantity Sold")
plt.ylabel("Sales")
plt.show()

# Side-by-side boxplots of Sales for each Product Category
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['Product_Category'], y=df['Sales'])
plt.title("Boxplot of Sales by Product Category")
plt.xlabel("Product Category")
plt.ylabel("Sales")
plt.xticks(rotation=45)  # Rotate x-axis labels if needed
plt.show()
```

0s completed at 12:59 PM


Total Sales per Region
7000
6000
5000
4000
3000
2000
1000
0
Total Sales
East
North
South
West
Region


Sales vs Quantity Scatterplot
800
600
400
200
0
Sales
1
2
3
4
5
6
7
8
9
Quantity Sold



The sales data analysis reveals key insights through visualizations and statistical measures. The bar chart shows the East region leading in total sales (≈7,000), followed by West (≈6,000), North (≈5,000), and South (≈4,000). A scatterplot and correlation matrix demonstrate a strong positive relationship (r=0.77) between sales and quantity sold, supported by a covariance of 334.91. Cross-tabulation data highlights product category performance by region: Home Decor dominates in the North (11 sales), Electronics maintains consistent performance across regions (5-7 sales), Books sell well in East/West (7 each), while Clothing underperforms in the South (only 3 sales). These findings suggest opportunities to replicate North's Home Decor success in other regions, address Clothing weaknesses in the South, and leverage the East's overall strong performance, while the sales-quantity correlation supports potential bundling strategies.

Code Used for Question Three

```
# 3a) Cross-tabulation of Sales transactions by Region and Product Category
cross_tab = pd.crosstab(df['Region'], df['Product_Category'])
print("Cross-tabulation of Sales by Region and Product Category:\n", cross_tab)
```

```
# 3b) Calculate covariance and correlation between Sales and Quantity
covariance = df[['Sales', 'Quantity']].cov()
correlation = df[['Sales', 'Quantity']].corr()

print("Covariance between Sales and Quantity:\n", covariance)
print("Correlation between Sales and Quantity:\n", correlation)
# 3c) Visualization

# Barplot of total Sales per Region
plt.figure(figsize=(8, 5))
sns.barplot(x=df.groupby("Region")["Sales"].sum().index,
y=df.groupby("Region")["Sales"].sum().values)
plt.title("Total Sales per Region")
plt.xlabel("Region")
plt.ylabel("Total Sales")
plt.show()

# Scatterplot of Sales vs Quantity
plt.figure(figsize=(7, 5))
sns.scatterplot(x=df['Quantity'], y=df['Sales'])
plt.title("Sales vs Quantity Scatterplot")
plt.xlabel("Quantity Sold")
plt.ylabel("Sales")
plt.show()

# Side-by-side boxplots of Sales for each Product Category
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['Product_Category'], y=df['Sales'])
plt.title("Boxplot of Sales by Product Category")
plt.xlabel("Product Category")
plt.ylabel("Sales")
plt.xticks(rotation=45)  # Rotate x-axis labels if needed
plt.show()
```