2.Visualizing time series data

AIM:

To implement programs for visualizing time series data.

PROCEDURE:

Loading Data:We load the CSV file using pd.read_csv("supermarket_sales.csv").

Converting Date Column:The 'Date' column is converted to datetime format using pd.to_datetime(), so it can be used properly as the x-axis in the plot.

Setting Date as Index:We set the Date column as the index to make it easier for time series plotting and analysis.

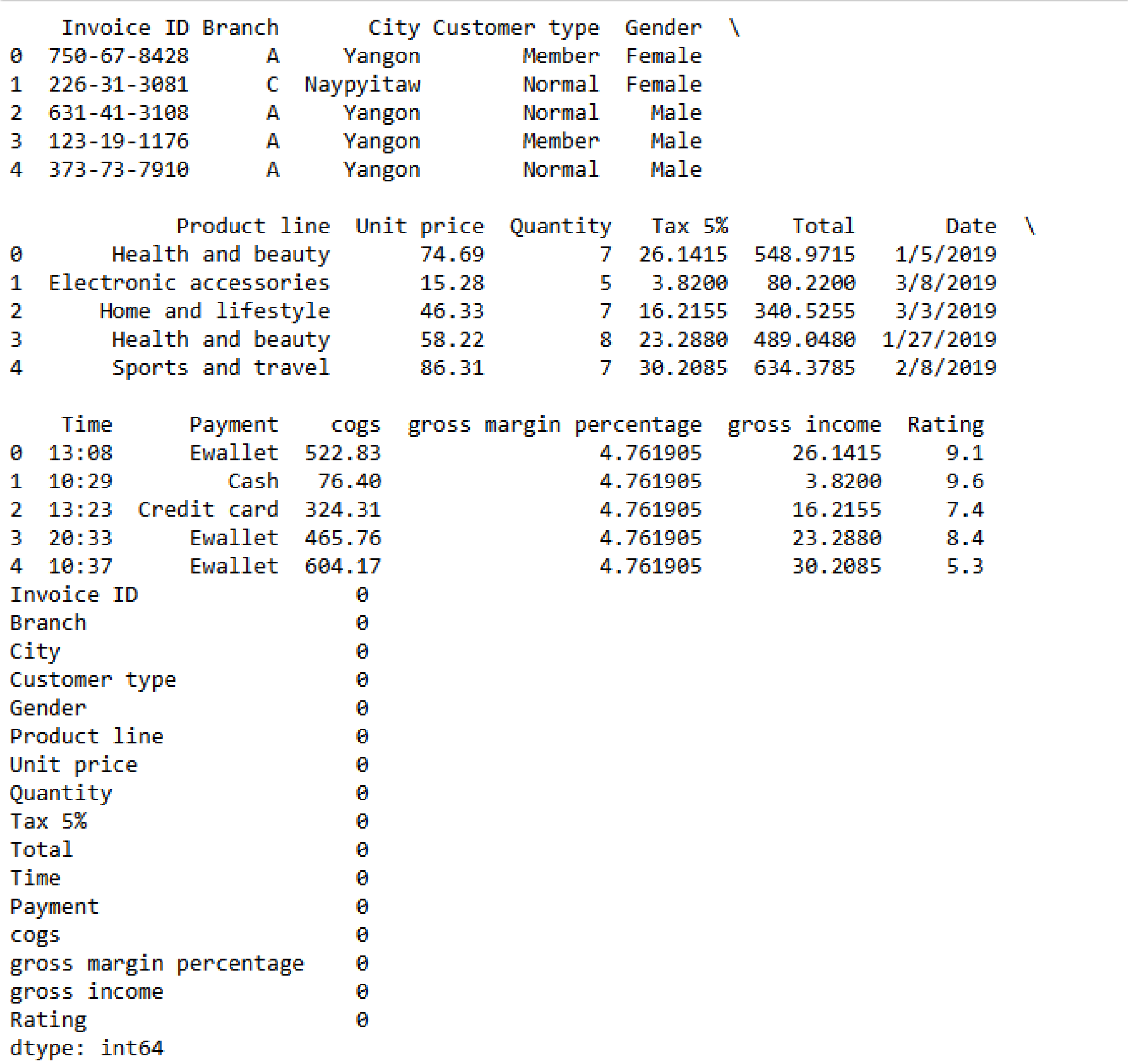
Plotting Sales:A line plot is generated using matplotlib to visualize sales over time. The Date will be plotted on the x-axis, and Sales will be plotted on the y-axis.

Customizing the Plot:We rotate the x-axis labels (plt.xticks(rotation=45)) for better readability. Adding plt.grid(True) makes the graph easier to interpret with gridlines.

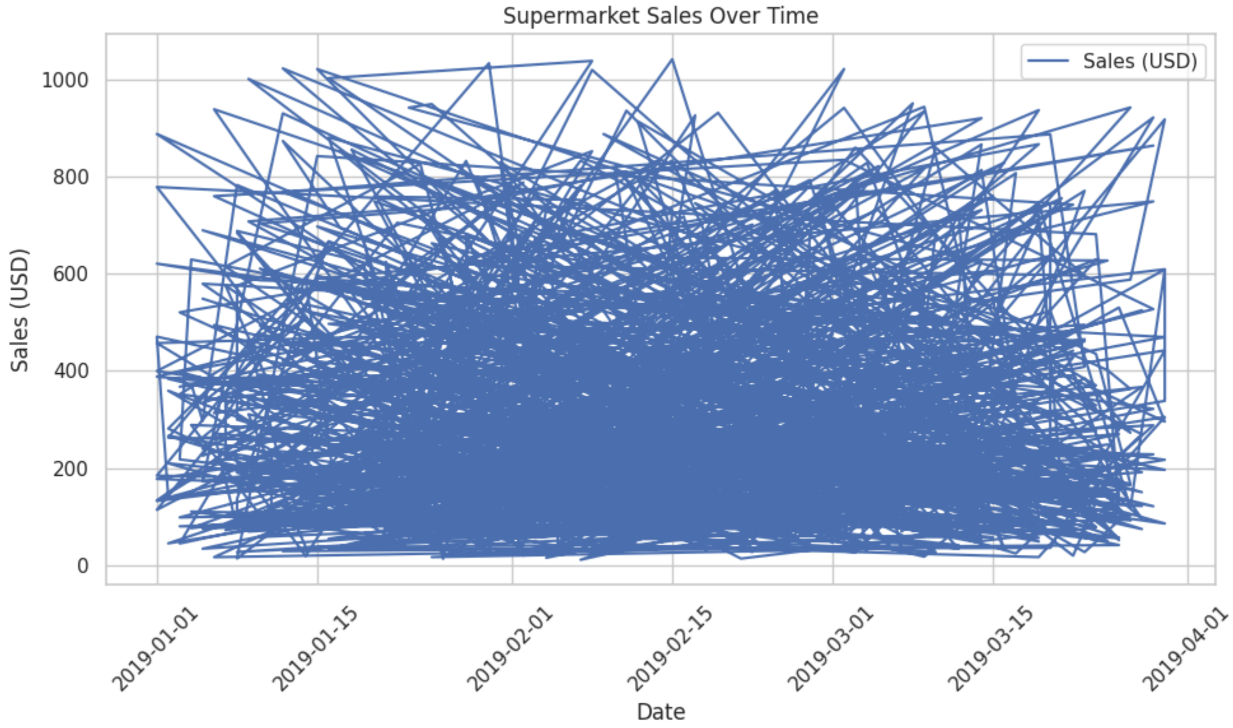
CODE:

```
 # Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load in the CSV file (adjust the file path accordingly)
sales_data = pd.read_csv(r"C:\Users\Lenovo\Downloads\supermarket_sales -
Sheet1.csv")

# Check the first few rows of the dataset
print(sales_data.head())

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

```
# Set the 'Date' column as the index for time series analysis
sales_data.set_index('Date', inplace=True)

# Check for missing values
print(sales_data.isnull().sum())

# Create a line plot of sales data over time
plt.figure(figsize=(10, 6))
# Access the correct sales column, likely 'Total' or 'sales'
plt.plot(sales_data.index, sales_data['Total'], label='Sales (USD)', color='b')
plt.xlabel('Date')
plt.ylabel('Sales (USD)')
plt.title('Supermarket Sales Over Time')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend()
plt.show()
```

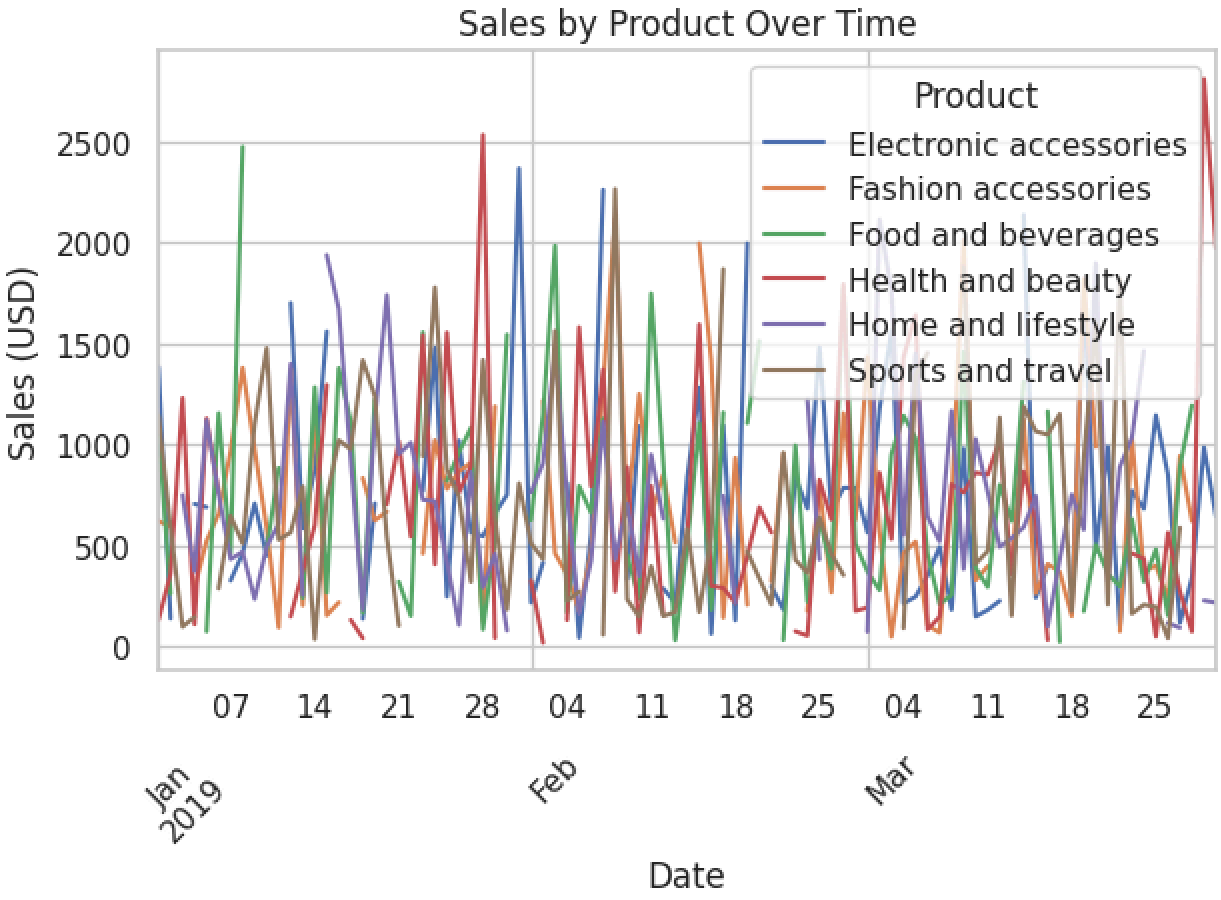


```
    Invoice ID Branch       City Customer type  Gender  \
0  750-67-8428      A     Yangon        Member  Female   
1  226-31-3081      C  Naypyitaw        Normal  Female   
2  631-41-3108      A     Yangon        Normal    Male   
3  123-19-1176      A     Yangon        Member    Male   
4  373-73-7910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

             Product line  Unit price  Quantity   Tax 5%     Total       Date  \
0       Health and beauty       74.69         7  26.1415  548.9715   1/5/2019   
1  Electronic accessories       15.28         5   3.8200   80.2200   3/8/2019   
2      Home and lifestyle       46.33         7  16.2155  340.5255   3/3/2019   
3       Health and beauty       58.22         8  23.2880  489.0480  1/27/2019   
4       Sports and travel       86.31         7  30.2085  634.3785   2/8/2019   

    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  
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
Time                       0
Payment                    0
cogs                       0
gross margin percentage    0
gross income               0
Rating                     0
dtype: int64
```



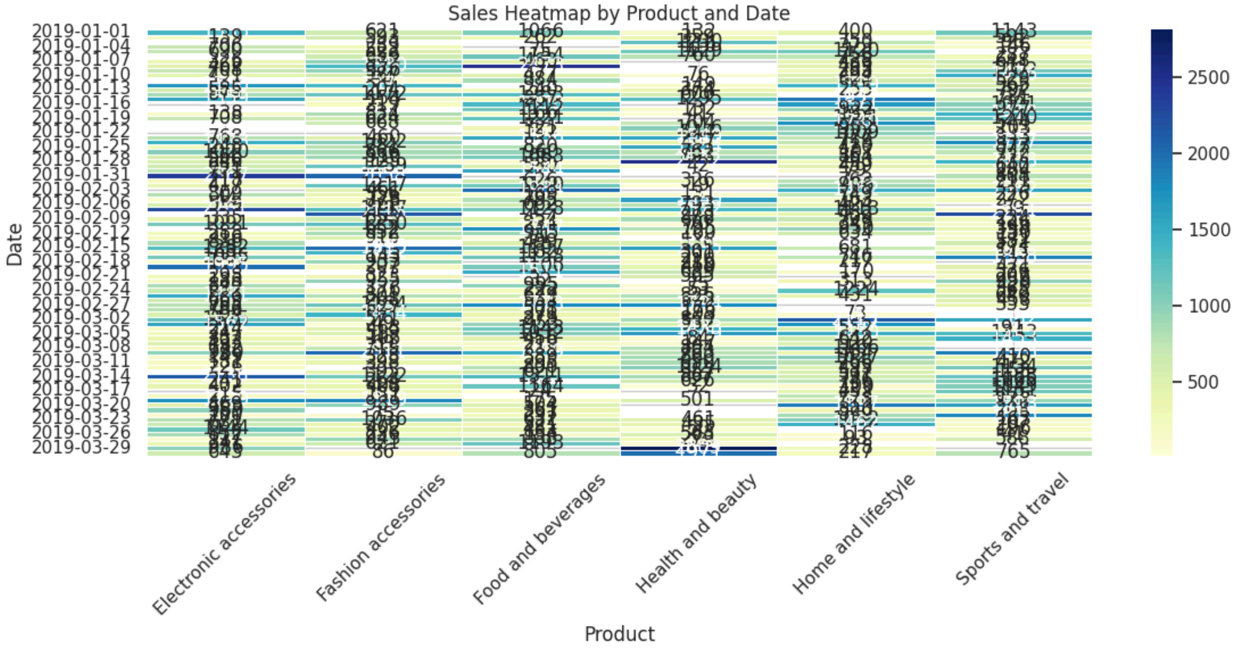
Line Plot with Multiple Series

```
sales_by_product = sales_data.groupby([sales_data.index, 'Product line'])['Total'].sum().unstack()

# Plot the sales for each product over time
plt.figure(figsize=(12, 6))
sales_by_product.plot(kind='line')
plt.title('Sales by Product Over Time')
plt.xlabel('Date')
plt.ylabel('Sales (USD)')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend(title='Product')
plt.show()
```



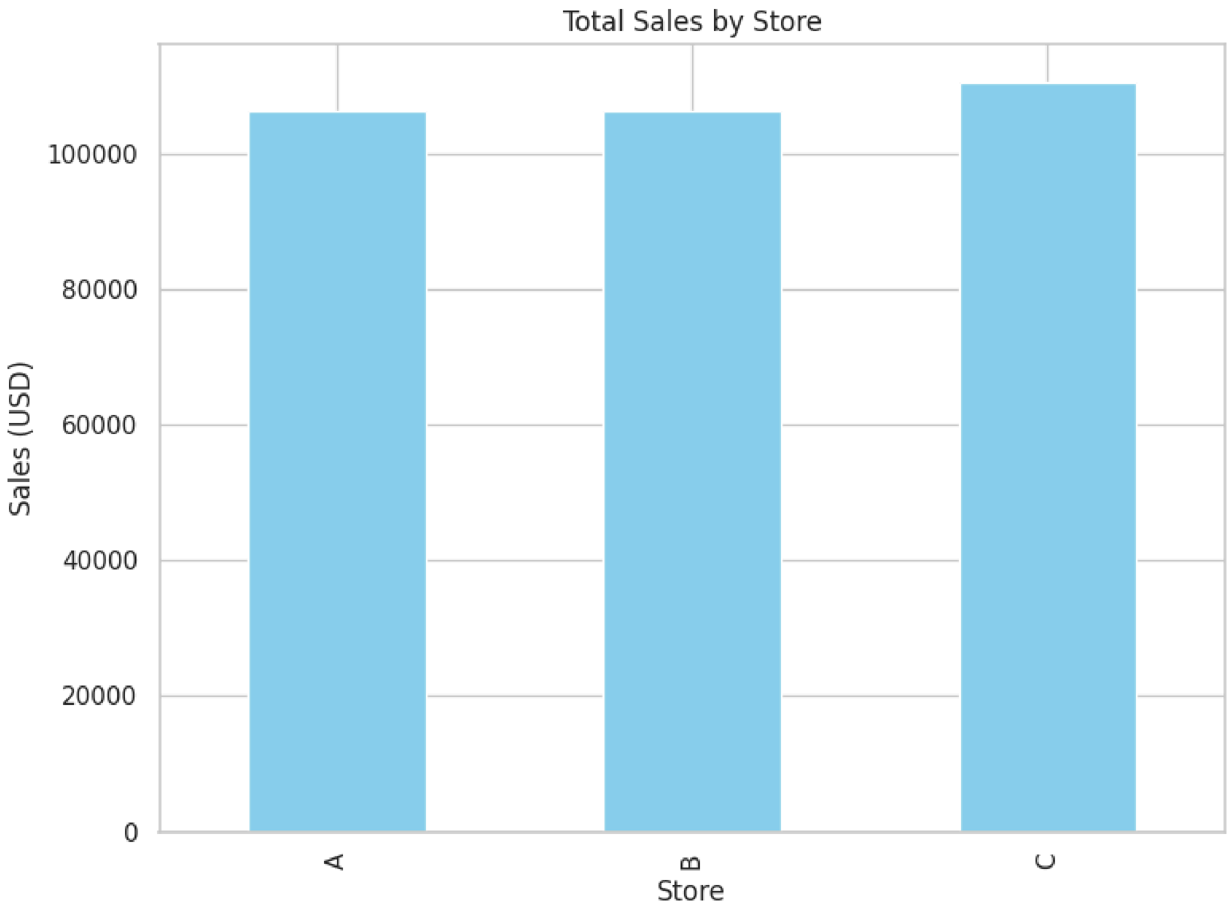
Heatmap

```
# Plot the heatmap
plt.figure(figsize=(12, 6))
sns.heatmap(sales_pivot, cmap='YlGnBu', annot=True, fmt='.0f', linewidths=.5)
plt.title('Sales Heatmap by Product and Date')
plt.xlabel('Product')
plt.ylabel('Date')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



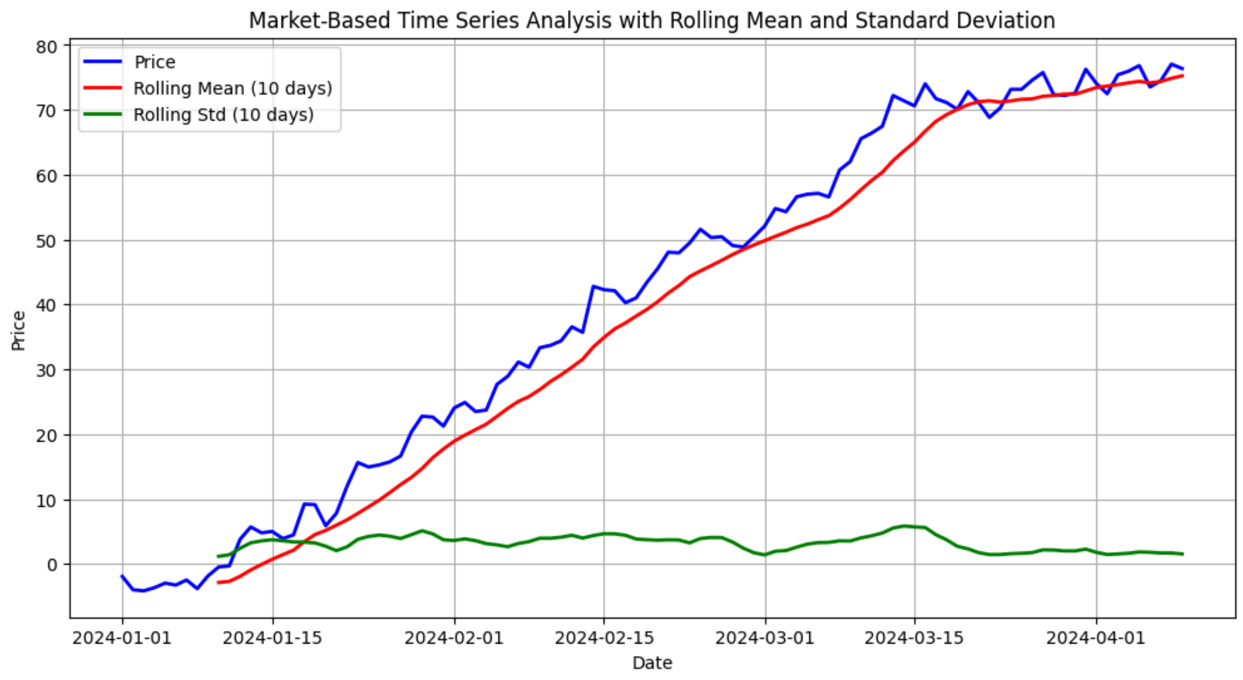
Bar Plot (Grouped by Category)

```
# Group by store and sum sales
sales_by_store = sales_data.groupby('Branch')['Total'].sum()

# Plot the bar plot
plt.figure(figsize=(8, 6))
sales_by_store.plot(kind='bar', color='skyblue')
plt.title('Total Sales by Store')
plt.xlabel('Store')
plt.ylabel('Sales (USD)')
plt.tight_layout()
plt.show()
```



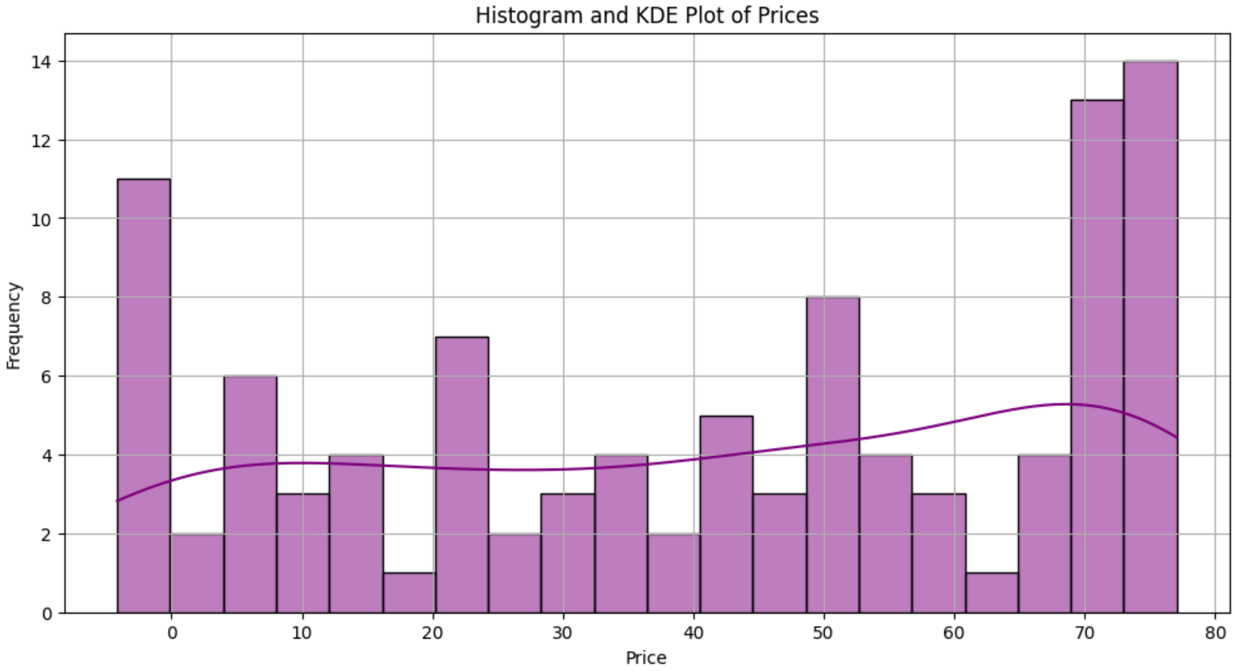
Rolling Mean and Standard Deviation Plot code

```
#plot the data
plt.figure(figsize=(12, 6))
sns.lineplot(x='Date', y='Price', data=data, color='blue', linewidth=2, label='Price')
plt.plot(data['Date'], rolling_mean, color='red', linewidth=2, label='Rolling Mean (10
days)')
plt.plot(data['Date'], rolling_std, color='green', linewidth=2, label='Rolling Std (10 days)')
plt.title('Market-Based Time Series Analysis with Rolling Mean and Standard Deviation')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.show()
```



Histogram

```
# Plot Histogram and KDE Plot
plt.figure(figsize=(12, 6))
Sns. histplot(data['Price'], kde=True, color='purple', bins=20)
plt.title('Histogram and KDE Plot of Prices')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



RESULT:

The visualizing time series data program was executed successfully.