EX:No.2 221501057

25/01/25

Program to analyze and visualize stock trends using time series plots, moving averages, volume analysis, and daily returns.

Aim:

Write a program to analyze and visualize stock trends using time series plots, moving averages, volume analysis, and daily returns.

Algorithm:

Step 1: Import Required Libraries

- Import pandas for data handling,numpy for numerical calculations,matplotlib.pyplot and seaborn for data visualization,Google Colab's files module to upload the dataset manually (if needed).

Step 2: Upload and Load the Dataset

- Upload the dataset using files.upload().
- Read the dataset using pd.read_csv().

Step 3: Data Preprocessing

- Generate a time series by creating a sale_date column using pd.date_range().
- Rename columns if necessary for better readability.
- Compute daily returns using pct_change() to analyze stock trends.

Step 4: Set Up Visualization Styling

- Use sns.set_style("whitegrid") to apply a clean grid layout.

Step 5: Generate Visualizations

Histogram of Sale Prices – To understand price distribution.
Scatter Plot of Sale Prices Over Time – To analyze price trends over time.
Scatter Plot of Daily Returns vs Sale Price – To evaluate returns correlation with price.
Box Plot of Sale Prices – To detect outliers and spread.
Correlation Heatmap – To check relationships between numerical variables.

Step 6: Display Results and Interpret Insights

- Analyze the trend patterns, price fluctuations, and correlation between different attributes in the dataset.

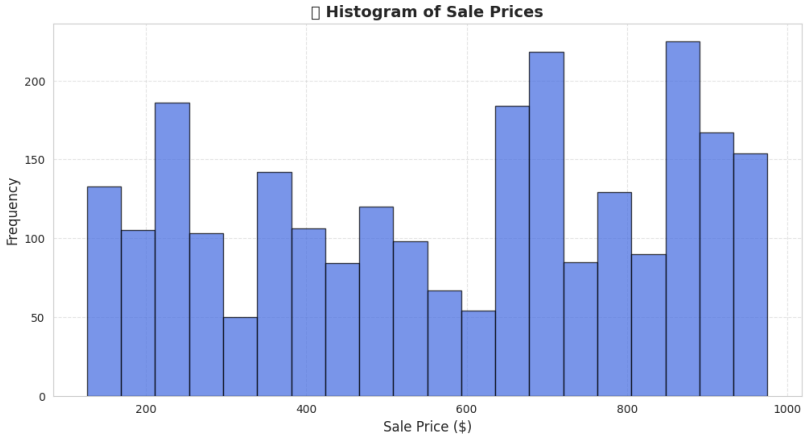
Code:

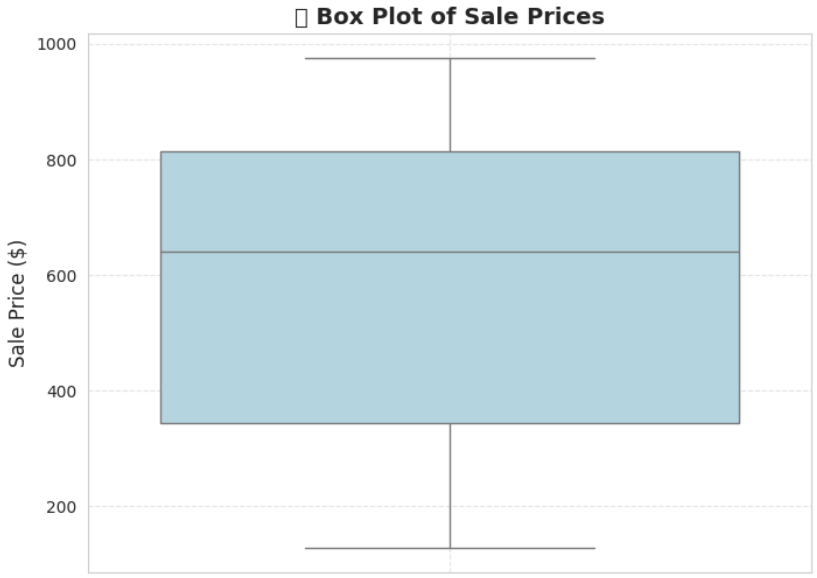
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from google.colab import files

uploaded = files.upload()
```

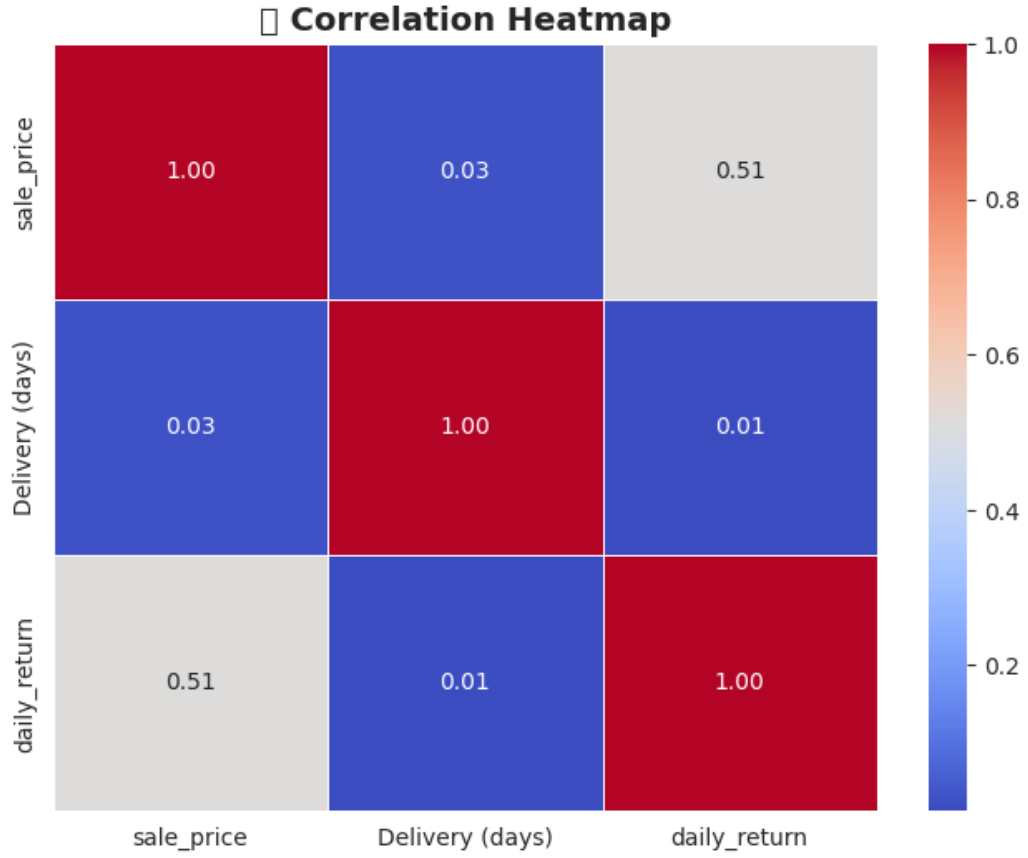
```
file_name = list(uploaded.keys())[0]
df = pd.read_csv(file_name)
df["sale_date"] = pd.date_range(start="2023-01-01", periods=len(df), freq="D")
df.rename(columns={"Price ($)": "sale_price"}, inplace=True)
df["daily_return"] = df["sale_price"].pct_change()
sns.set_style("whitegrid")
plt.figure(figsize=(12, 6))
plt.hist(df["sale_price"], bins=20, color="royalblue", alpha=0.7, edgecolor="black")
plt.xlabel("Sale Price ($)", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.title("📊 Histogram of Sale Prices", fontsize=14, fontweight="bold")
plt.grid(True, linestyle="--", alpha=0.5)
plt.show()
plt.figure(figsize=(12, 6))
plt.scatter(df["sale_date"], df["sale_price"], alpha=0.6, color="purple")
plt.xlabel("Date", fontsize=12)
plt.ylabel("Sale Price ($)", fontsize=12)
plt.title("Scatter Plot of Sale Prices Over Time", fontsize=14, fontweight="bold")
plt.xticks(rotation=45)
plt.grid(True, linestyle="--", alpha=0.5)
plt.show()
plt.figure(figsize=(12, 6))
plt.scatter(df["sale_price"], df["daily_return"], alpha=0.6, color="green")
plt.xlabel("Sale Price ($)", fontsize=12)
plt.ylabel("Daily Return (%)", fontsize=12)
plt.title("Scatter Plot of Daily Returns vs Sale Price", fontsize=14, fontweight="bold")
plt.grid(True, linestyle="--", alpha=0.5)
plt.show()
plt.figure(figsize=(8, 6))
sns.boxplot(y=df["sale_price"], color="lightblue")
plt.ylabel("Sale Price ($)", fontsize=12)
plt.title("Box Plot of Sale Prices", fontsize=14, fontweight="bold")
plt.grid(True, linestyle="--", alpha=0.5)
plt.show()
```

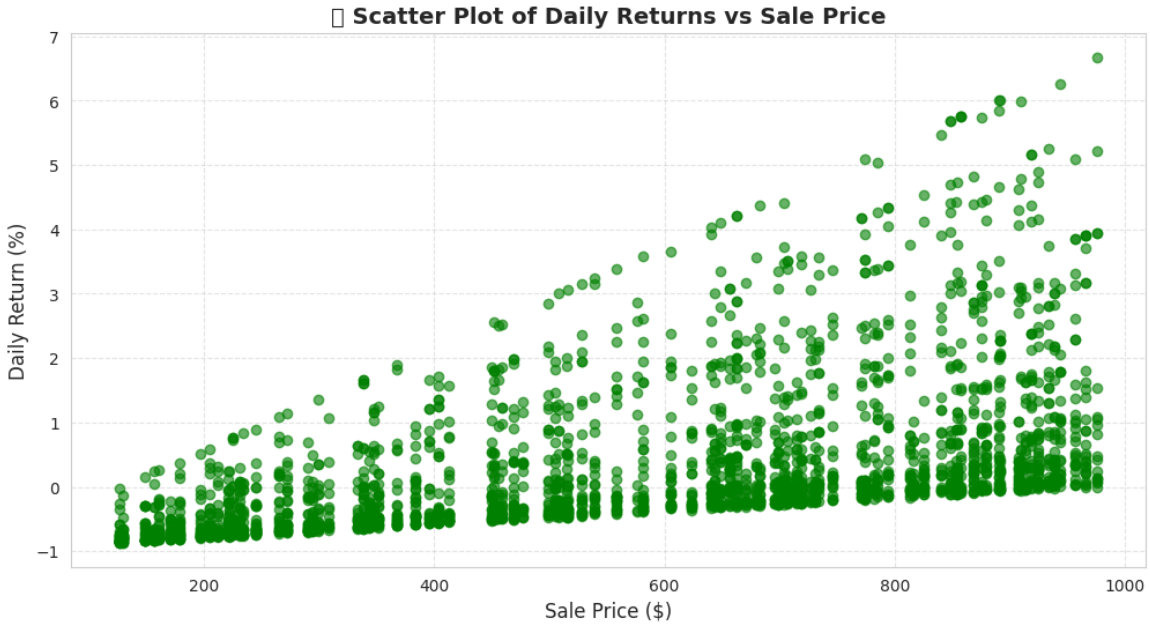
```
plt.figure(figsize=(8, 6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Correlation Heatmap", fontsize=14, fontweight="bold")
plt.show()
```

Output:




Box Plot of Sale Prices
Sale Price ($)
1000
800
600
400
200


Correlation Heatmap
sale_price
Delivery (days)
daily_return
1.00
0.03
0.51
0.03
1.00
0.01
0.51
0.01
1.00
sale_price
Delivery (days)
daily_return
1.0
0.8
0.6
0.4
0.2



Result:

Thus, the program using the time series data implementation has been done successfully.