**EX:No.2 221501013**

**31/01/25**

**IMPLEMENTING DIFFERENT VISUALIZATION TECHNIQUE USING TIME SERIES DATA**

**AIM:**

To implementing different visualization technique using time series dataset.

**PROCESS:**

**#Importing libraries**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from statsmodels.tsa.seasonal import seasonal\_decompose

**# Generate Synthetic Dataset**

np.random.seed(42)

n = 200

dates = pd.date\_range(start='2022-01-01', periods=n)

close\_prices = np.random.normal(loc=150, scale=10, size=n) # Normal distribution

outliers = np.random.choice(n, size=5, replace=False)

close\_prices[outliers] += np.random.normal(loc=50, scale=5, size=5) # Inject outliers

**# Create DataFrame**

data = pd.DataFrame({'Date': dates, 'Close': close\_prices})

**# Box Plot to Check Outliers**

plt.figure(figsize=(8, 5))

sns.boxplot(x=data['Close'])

plt.title('Box Plot of Synthetic Close Prices')

plt.grid(True)

plt.show()

**# Scatter Plot to Check Distribution**

plt.figure(figsize=(10, 6))

plt.scatter(data['Date'], data['Close'], color='blue', alpha=0.6, label='Close Prices')

plt.xlabel('Date')

plt.ylabel('Close Prices')

plt.title('Scatter Plot of Synthetic Close Prices')

plt.legend()

plt.grid(True)

plt.show()

**# Reverse the order to maintain chronological order**

data = data.iloc[::-1].reset\_index(drop=True)

**# Handling Missing Values**

data.dropna(inplace=True) # Drop rows with missing values

data['Close'].fillna(data['Close'].mean(), inplace=True) # Fill NaNs in 'Close'

**# Handling Outliers using IQR**

Q1 = data['Close'].quantile(0.25)

Q3 = data['Close'].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

data = data[(data['Close'] >= lower\_bound) & (data['Close'] <= upper\_bound)]

**# Extract Close Prices**

close\_prices = data['Close'].values

data['Normalized\_Close'] = close\_prices / np.max(close\_prices) # Normalize data

def plot\_time\_series(data, title='Time Series Data', xlabel='Time', ylabel='Value'):

plt.figure(figsize=(12, 6))

plt.plot(data, label='Close Prices', color='blue')

plt.xlabel(xlabel)

plt.ylabel(ylabel)

plt.title(title)

plt.legend()

plt.grid(True)

plt.show()

**# Simple Line Plot of Closing Prices**

plot\_time\_series(data['Close'], title='Synthetic Stock Close Prices')

**# Moving Average (Smoothing)**

window\_size = 50

data['Moving\_Avg'] = data['Close'].rolling(window=window\_size).mean()

plt.figure(figsize=(12, 6))

plt.plot(data['Close'], label='Close Prices', alpha=0.6)

plt.plot(data['Moving\_Avg'], label=f'{window\_size}-Day Moving Average', color='red')

plt.xlabel('Time')

plt.ylabel('Close Prices')

plt.title('Synthetic Stock Close Prices with Moving Average')

plt.legend()

plt.grid(True)

plt.show()

**# Seasonal Decomposition**

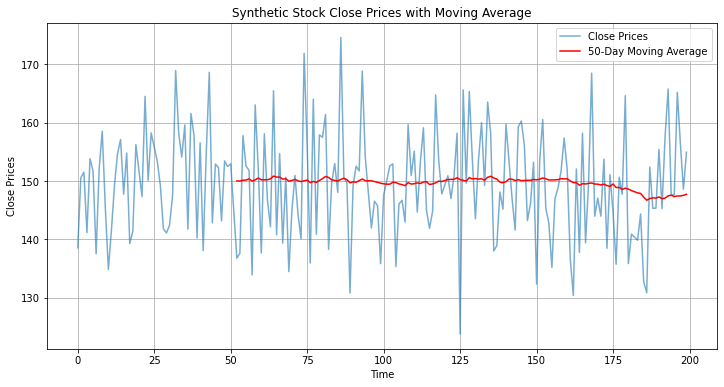
result = seasonal\_decompose(data['Close'], model='additive', period=30)

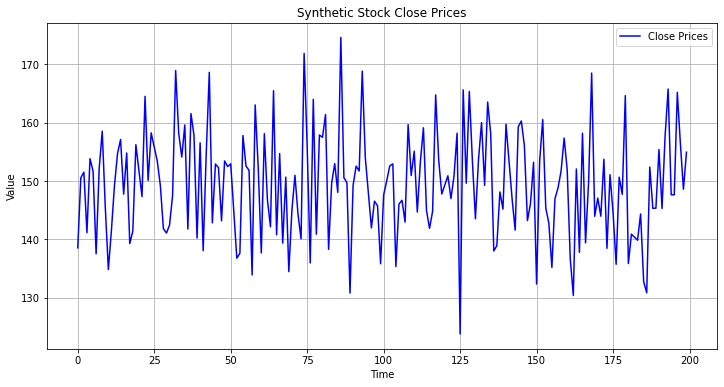
plt.figure(figsize=(12, 8))

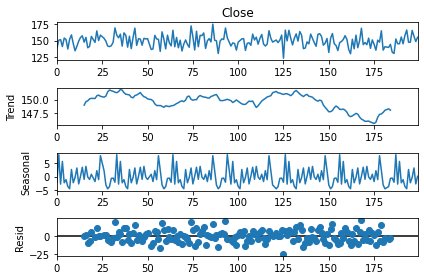
result.plot()

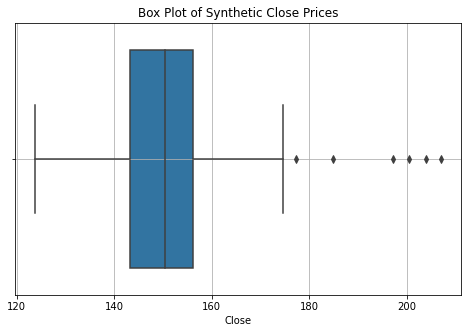
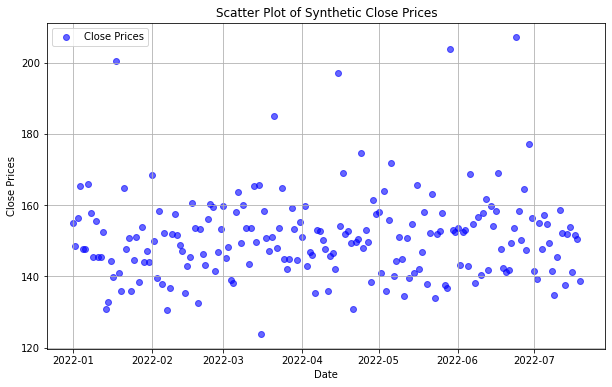
plt.show()

**OUTPUT:**



****

****



**RESULT:**

The implementing different visualization technique using time series dataset is successfully implemented.