**Exp no: 1 Time Series data cleaning, loading,handling data and preprocessing techniques**

**Date: 24/1/25**

**Aim:**

To implement time series data data cleaning, loading,handling data and preprocessing techniques

### ****Objectives:****

The objective of this preprocessing pipeline for a time series sales dataset is to prepare the data for predictive modeling. The steps aim to handle missing values, remove anomalies, and ensure proper time-based ordering. The goal is to process the data so that models can accurately predict future sales trends, detect seasonality patterns, and generate forecasts. By addressing issues such as missing timestamps and ensuring the data is in a consistent format, we improve the quality of input for time series forecasting models.

**Background-Scope:**

In time series data, each observation is indexed by time, often involving sales data over days, months, or years. Missing values, outliers, and improper time ordering can introduce bias into the analysis. This preprocessing scope focuses on handling these issues by filling gaps, removing outliers, and ensuring that the data is correctly indexed. Additionally, transforming features like seasonal components and trend patterns can help improve forecast accuracy. Proper preprocessing lays the groundwork for time series forecasting models to detect trends, predict future sales, and guide business decisions.

### ****Steps for Time Series Sales Data Preprocessing****:

**Steps 1:** Import the necessary libraries # Data

import json

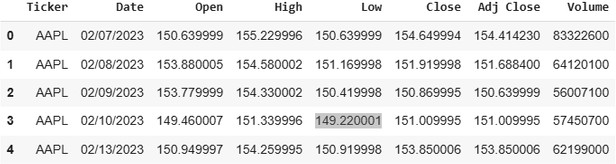
import numpy as np= import pandas as pd

from sklearn.preprocessing import LabelEncoder= from sklearn.preprocessing import StandardScaler # Visual

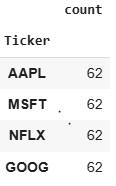
import matplotlib.pyplot as plt import seaborn as sns

import warnings warnings.filterwarnings("ignore")

**Steps 2:** Load the CSV (data) file.

df = pd.read\_csv("/content/stocks.csv") df.head()

**Steps 3:** Counting the no. of values df.Ticker.value\_counts()



**Steps 4:** Fixing the date value. df['Date'] = pd.to\_datetime(df['Date'])

**Steps 5:** Visualizing the stocks prices over time

fig, axs = plt.subplots(3, 2, figsize=(15, 16)) fig.suptitle('Data by ticker type')

cols = ['Close', 'Adj Close', 'Open', 'High', 'Low', 'Volume'] for i, col in enumerate(cols):

row = i // 2 col = i % 2

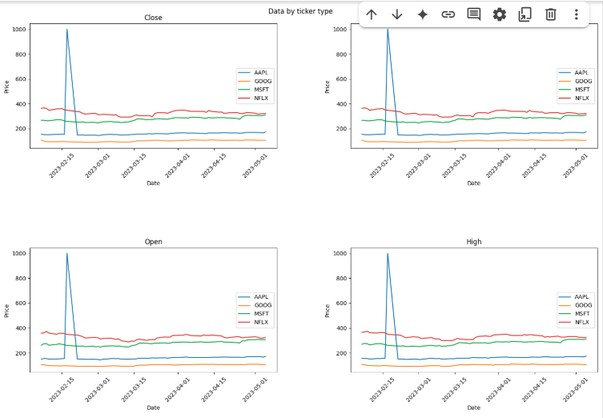
for ticker, data in df.groupby('Ticker'):

axs[row, col].plot(data['Date'], data[cols[i]], label=ticker) axs[row, col].set\_title(cols[i])

axs[row, col].set\_xlabel('Date') axs[row, col].set\_ylabel('Price') axs[row, col].legend(loc='right')

axs[row, col].tick\_params(axis='x', rotation=45)

plt.tight\_layout() plt.subplots\_adjust(wspace=0.3, hspace=0.8) plt.show()



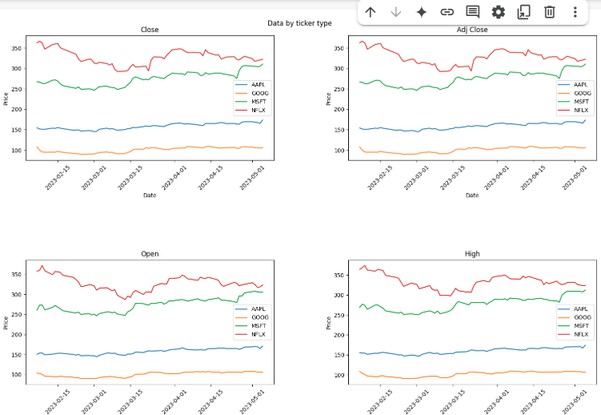
**Steps 6:** As we discovered outlier, we gonna minimize them using data cleaning techniques

Q1 = df['Close'].quantile(0.25) Q3 = df['Close'].quantile(0.75)

IQR = Q3 - Q1

df = df[(df['Close'] >= (Q1 - 1.5 \* IQR)) & (df['Close'] <= (Q3 + 1.5 \* IQR))]

**Steps 7:** Repeat the step 5.



Result:

Thus the program is implemented

221501024