# 1. Implement programs for time series data cleaning, loading and handling times series data and pre-processing techniques.

Implement programs for time series data cleaning, loading and handling times series data and pre-processing techniques.

### **AIM:**

To Implement programs for time series data cleaning, loading and handling times series data and pre-processing techniques.

#### **PROGRAM:**

def clean\_data(df):

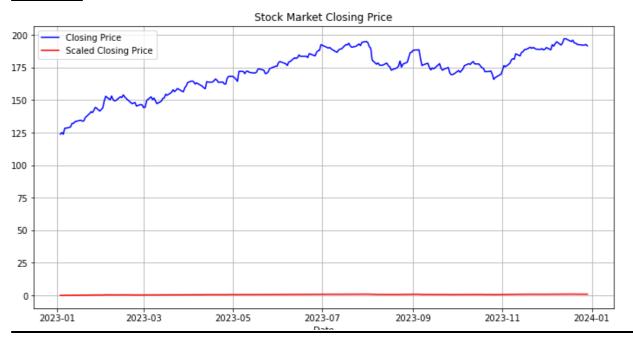
```
Import necessary libraries
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
# Function to load data from Yahoo Finance
def load_data(ticker, start_date, end_date):
  Loads stock data from Yahoo Finance for a given ticker symbol.
  Args:
     ticker (str): The ticker symbol of the stock (e.g., 'AAPL' for Apple).
    start_date (str): Start date in 'YYYY-MM-DD' format.
    end_date (str): End date in 'YYYY-MM-DD' format.
  Returns:
     pd.DataFrame: Stock data with date as the index.
  # Download stock data using yfinance
  df = yf.download(ticker, start=start_date, end=end_date)
  print("Data loaded successfully. First 5 rows:")
  print(df.head())
  return df
# Function to clean the data
```

Cleans the stock market dataset by handling missing values, outliers, and duplicates.

```
Args:
     df (pd.DataFrame): Input DataFrame.
  Returns:
    pd.DataFrame: Cleaned DataFrame.
  # Handle missing values (e.g., forward fill missing data)
  df.fillna(method='ffill', inplace=True)
  # Handle outliers using the IQR method for 'Close' price
  Q1 = df['Close'].quantile(0.25)
  Q3 = df['Close'].quantile(0.75)
  IQR = Q3 - Q1
  lower bound = Q1 - 1.5 * IQR
  upper_bound = Q3 + 1.5 * IQR
  df = df[(df['Close'] >= lower_bound) & (df['Close'] <= upper_bound)]
  # Remove duplicate rows
  df = df[\sim df.index.duplicated(keep='first')]
  print(f"Data cleaned. Remaining rows: {len(df)}")
  return df
# Function to preprocess the data (normalization)
def preprocess_data(df):
  Preprocesses the data by scaling the stock prices using MinMaxScaler.
  Args:
     df (pd.DataFrame): Input DataFrame.
  Returns:
    pd.DataFrame: DataFrame with an additional scaled 'Close' price.
    MinMaxScaler: Fitted scaler object.
  scaler = MinMaxScaler(feature_range=(0, 1))
  df['Scaled_Close'] = scaler.fit_transform(df[['Close']])
  print("Data preprocessing completed. Preview of scaled data:")
  print(df.head())
  return df. scaler
# Function to visualize the data
def visualize_data(df):
  Visualizes the stock data (e.g., Closing Price) over time.
  Args:
```

```
df (pd.DataFrame): Input DataFrame.
  plt.figure(figsize=(12, 6))
  plt.plot(df.index, df['Close'], label='Closing Price', color='blue')
  plt.plot(df.index, df['Scaled_Close'], label='Scaled Closing Price', color='red')
  plt.title('Stock Market Closing Price')
  plt.xlabel('Date')
  plt.ylabel('Price')
  plt.legend()
  plt.grid()
  plt.show()
# Main program
if _name_ == "_main_":
  # Set the ticker symbol and date range
  ticker = 'AAPL' # Example: Apple Inc.
  start_date = '2023-01-01'
  end_date = '2023-12-31'
  # Step 1: Load data from Yahoo Finance
  df = load_data(ticker, start_date, end_date)
  # Step 2: Clean data
  df = clean_data(df)
  # Step 3: Preprocess data
  df, scaler = preprocess_data(df)
  # Step 4: Visualize data
  visualize_data(df)
```

## **OUTPUT:**



# **RESULT:**

Thus, the program for Implement programs for time series data cleaning, loading and handling times series data and pre-processing techniques is executed successfully.