**5. Implement programs for estimating & eliminating trend in time series data aggregation, smoothing**

| **EX.N0 : 5** | **Implement programs for estimating & eliminating trend in time series data**  **aggregation, smoothing.** |
| --- | --- |
| **DATE : 29/03/2025** |

**AIM:**

To Implement programs for estimating & eliminating trend in time series data- aggregation, Smoothing.

**PROGRAM:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.api import SimpleExpSmoothing

file\_path = r""C:\Users\heman\OneDrive\Desktop\abdul\TSA\EX 5\Crude Oil Prices Daily.xlsx") (1)\FINAL\_USO.csv" # Update path

df = pd.read\_csv(file\_path, parse\_dates=["Date"], index\_col="Date")

df.columns = df.columns.str.strip()

print("Columns in dataset:", df.columns) if "Adj Close" not in df.columns:

print("Error: 'Adj Close' column not found!")

print("Available columns:", df.columns)

exit()

target = "Adj Close"

features = [

'SP\_open', 'SP\_high', 'SP\_low', 'SP\_close', 'SP\_Ajclose', 'SP\_volume',

'DJ\_open', 'DJ\_high', 'DJ\_low', 'DJ\_close', 'DJ\_Ajclose', 'DJ\_volume',

'EG\_open', 'EG\_high', 'EG\_low', 'EG\_close', 'EG\_Ajclose', 'EG\_volume', 'EU\_Price', 'EU\_open', 'EU\_high', 'EU\_low', 'EU\_Trend',

'OF\_Price', 'OF\_Open', 'OF\_High', 'OF\_Low', 'OF\_Volume', 'OF\_Trend', 'OS\_Price', 'OS\_Open', 'OS\_High', 'OS\_Low', 'OS\_Trend',

'SF\_Price', 'SF\_Open', 'SF\_High', 'SF\_Low', 'SF\_Volume', 'SF\_Trend',

'USB\_Price', 'USB\_Open', 'USB\_High', 'USB\_Low', 'USB\_Trend',

'PLT\_Price', 'PLT\_Open', 'PLT\_High', 'PLT\_Low', 'PLT\_Trend',

'PLD\_Price', 'PLD\_Open', 'PLD\_High', 'PLD\_Low', 'PLD\_Trend',

'RHO\_PRICE', 'USDI\_Price', 'USDI\_Open', 'USDI\_High', 'USDI\_Low', 'USDI\_Volume', 'USDI\_Trend',

'GDX\_Open', 'GDX\_High', 'GDX\_Low', 'GDX\_Close', 'GDX\_Adj Close', 'GDX\_Volume', 'USO\_Open', 'USO\_High', 'USO\_Low', 'USO\_Close', 'USO\_Adj Close', 'USO\_Volume' ]

df = df[[target] + features].dropna()

df["SMA\_10"] = df[target].rolling(window=10).mean()

df["SMA\_20"] = df[target].rolling(window=20).mean() weights = np.arange(1, 11)

df["WMA\_10"] = df[target].rolling(window=10).apply(lambda x: np.dot(x, weights) / weights.sum(), raw=True) exp\_model = SimpleExpSmoothing(df[target]).fit(smoothing\_level=0.2, optimized=False) df["Exp\_Smooth"] = exp\_model.fittedvalues

df["Detrended\_SMA"] = df[target] - df["SMA\_10"]

df["Detrended\_WMA"] = df[target] - df["WMA\_10"]

df["Detrended\_Exp"] = df[target] - df["Exp\_Smooth"] plt.figure(figsize=(12, 5))

plt.plot(df[target], label="Original Price", color="blue", alpha=0.5)

plt.plot(df["SMA\_10"], label="SMA (10 days)", color="red")

plt.plot(df["Exp\_Smooth"], label="Exponential Smoothing", color="black") plt.title("Gold Price Trend Estimation")

plt.xlabel("Date")

plt.ylabel("Price")

plt.legend()

plt.show()

plt.figure(figsize=(12, 5))

plt.plot(df["Detrended\_SMA"], label="Detrended (SMA)", color="red") plt.plot(df["Detrended\_WMA"], label="Detrended (WMA)", color="green") plt.plot(df["Detrended\_Exp"], label="Detrended (Exp Smoothing)", color="black") plt.axhline(y=0, color='gray', linestyle='--')

plt.title("Detrended Gold Price Data")

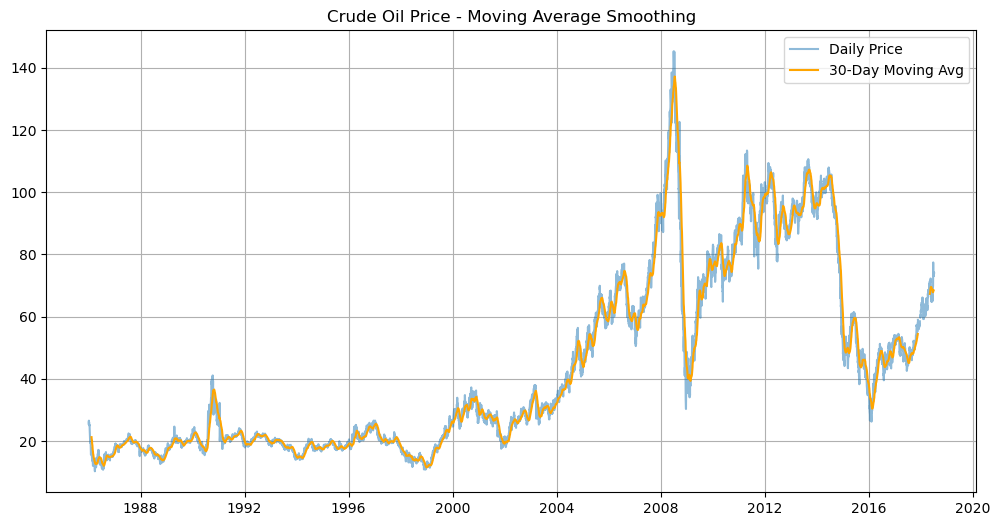
plt.xlabel("Date")

plt.ylabel("Price Difference")

plt.legend()

plt.show()

**OUTPUT:**

**RESULT:**

Thus, the program for Implement programs for estimating & eliminating trend in time series data- aggregation, smoothing is executed successfully.