**Implement programs to check stationarity of a time series data**

**AIM:** The aim of this exercise is to assess the stationarity of a time series dataset using statistical tests (e.g., ADF test) and visualizations, and to transform non-stationary data if necessary.

**STEP 1:** Load necessary libraries.

!pip install pandas matplotlib statsmodels

import pandas as pd

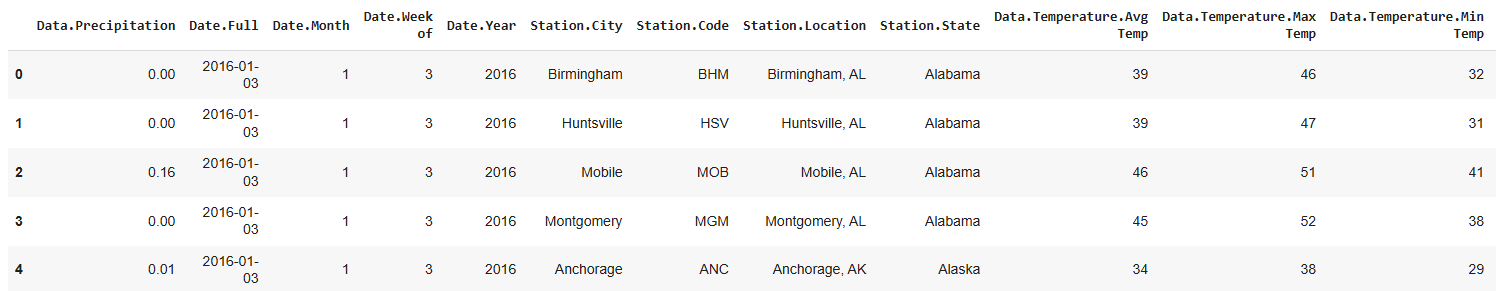
**STEP 2:**Load the Weather dataset.

# Load the dataset

df = pd.read\_csv('/content/weather.csv')

# Display the first few rows of the dataset

df.head()



# Convert the 'Date.Full' column to datetime format

df['Date.Full'] = pd.to\_datetime(df['Date.Full'])

# Verify the conversion

Df.dtypes

**STEP 3:**Preprocess the weather dataset.

# Select relevant columns for time series analysis

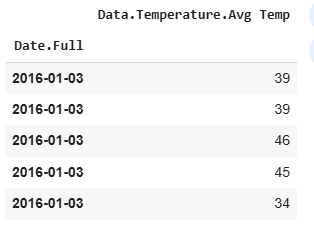
df = df[['Date.Full', 'Data.Temperature.Avg Temp']]

# Set 'Date.Full' as the index (time series index)

df.set\_index('Date.Full', inplace=True)

# Check the first few rows of the DataFrame to confirm

df.head()



**STEP 4:**Perform the Augmented Dickey-Fuller (ADF) test.

from statsmodels.tsa.stattools import adfuller

result = adfuller(df['Data.Temperature.Avg Temp'].dropna())

print("ADF Statistic:", result[0])

print("p-value:", result[1])

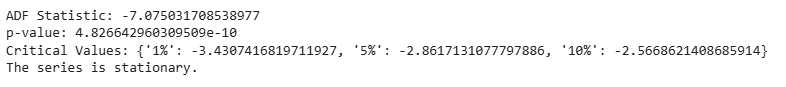
print("Critical Values:", result[4])

if result[1] <= 0.05:

print("The series is stationary.")

else:

print("The series is non-stationary.")



**STEP 5:**Plot the rolling mean and rolling standard deviation of the original series.

window = 12

rolling\_mean = df['Data.Temperature.Avg Temp'].rolling(window=window).mean()

rolling\_std = df['Data.Temperature.Avg Temp'].rolling(window=window).std()

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

plt.plot(df['Data.Temperature.Avg Temp'], label='Original Average Temperature', color='blue', alpha=0.6)

plt.plot(rolling\_mean, label='Rolling Mean', color='red', linestyle='--')

plt.plot(rolling\_std, label='Rolling Std Dev', color='green', linestyle='--')

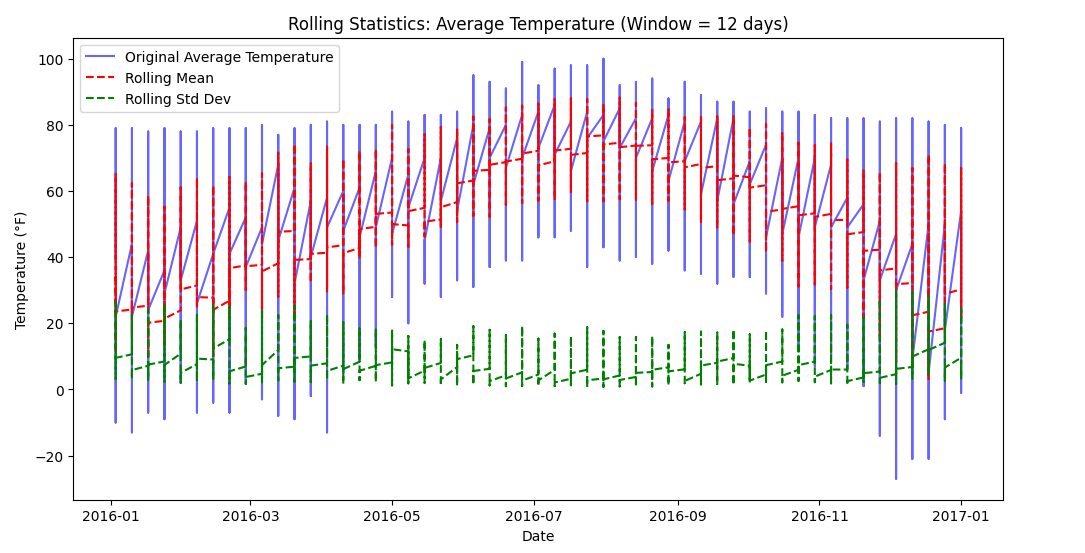
plt.title(f'Rolling Statistics: Average Temperature (Window = {window} days)')

plt.xlabel('Date')

plt.ylabel('Temperature (°F)')

plt.legend()

plt.show()



**STEP 6:** Plot ACF and PACF for the original time series and for the differenced time series.

from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf

# Plot ACF and PACF for the original time series

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

# ACF plot

plt.subplot(2, 1, 1)

plot\_acf(df['Data.Temperature.Avg Temp'].dropna(), lags=40, ax=plt.gca())

plt.title('ACF - Original Series')

# PACF plot

plt.subplot(2, 1, 2)

plot\_pacf(df['Data.Temperature.Avg Temp'].dropna(), lags=40, ax=plt.gca())

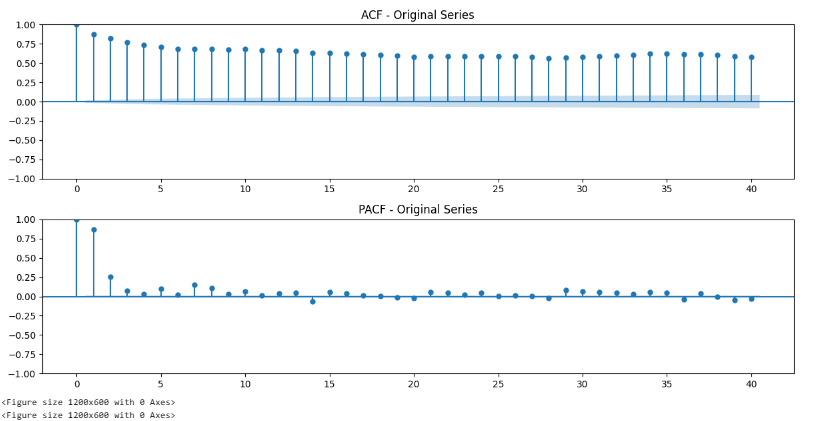
plt.title('PACF - Original Series')

plt.tight\_layout()

plt.show()

# Plot ACF and PACF for the differenced time series

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



**STEP 7:**calculate the difference between consecutive values and checks if the transformed data is stationary using the Augmented Dickey-Fuller (ADF) test and visualizes the results.

# Apply differencing

df['Differenced'] = df['Data.Temperature.Avg Temp'].diff()

# Drop NaN values after differencing

df\_differenced = df['Differenced'].dropna()

# Plotting the differenced series

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

plt.plot(df['Differenced'], label='Differenced Average Temperature', color='red')

plt.title('Differenced Average Temperature Over Time')

plt.xlabel('Date')

plt.ylabel('Differenced Temperature')

plt.legend()

plt.show()

# Perform ADF test for the differenced series

from statsmodels.tsa.stattools import adfuller

result\_diff = adfuller(df['Differenced'].dropna())

# Print ADF test result

print(f"ADF Statistic: {result\_diff[0]:.2f}")

print(f"p-value: {result\_diff[1]:.4f}")

# Plotting ADF result on the differenced time series

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

plt.plot(df['Differenced'], label='Differenced Average Temperature', color='red')

plt.title('Differenced Average Temperature with ADF Test Result')

# Add ADF test result annotation

plt.annotate(f'ADF Statistic: {result\_diff[0]:.2f}\np-value: {result\_diff[1]:.4f}',

xy=(0.5, 0.9), xycoords='axes fraction', ha='center', fontsize=12, color='red')

plt.xlabel('Date')

plt.ylabel('Differenced Temperature')

plt.legend()

plt.show()

