**221501111**

**EXPERIMENT 4 : STATIONARY CHECK OF IN TIME SERIES DATA**

AIM:

To implement a program that checks the stationarity of a time series dataset using statistical methods and visualization techniques.

ALGORITHM:

1. Import necessary libraries.
2. Prompt the user to upload a dataset file.
3. Load the dataset into a Pandas DataFrame.
4. Display column names and ask for the date column name.
5. Parse the date column and set it as an index .
6. Ask the user to select a target column for time series analysis.
7. Visualize the time series data using line plots.
8. Compute rolling mean and standard deviation to check trends
9. Perform the Augmented Dickey-Fuller (ADF) Testto check for stationarity.
10. Display results and interpretation.

CODE:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns+

from statsmodels.tsa.stattools import adfuller

from google.colab import files

import io

print("Please upload your dataset (CSV file).")

uploaded = files.upload()

filename = list(uploaded.keys())[0]

print(f"Uploaded file: {filename}")

df = pd.read\_csv(io.BytesIO(uploaded[filename]))

print("\nColumn names in the dataset:", df.columns.toli))st(

date\_column = input("\nEnter the column name for the date (or press Enter if no date column): ").strip()

if date\_column and date\_column in df.columns:

    df[date\_column] = pd.to\_datetime(df[date\_column])  # Convert to datetime format

    df.set\_index(date\_column, inplace=True)  # Set as index

    print(f"\n'{date\_column}' column set as index.")

else:

    print("\nNo date column provided or found. Using default index.")

print("\nFirst few rows of the dataset:")

print(df.head()

column\_name = input("\nEnter the column name for time-series analysis (e.g., Temperature): ").strip()

if column\_name not in df.columns:

    print(f"\nError: Column '{column\_name}' not found in dataset.")

else:

    ts = df[column\_name]

 plt.figure(figsize=(12,6))

 plt.plot(ts, label="Original Time Series")

 plt.title(f"{column\_name} Time Series Data")

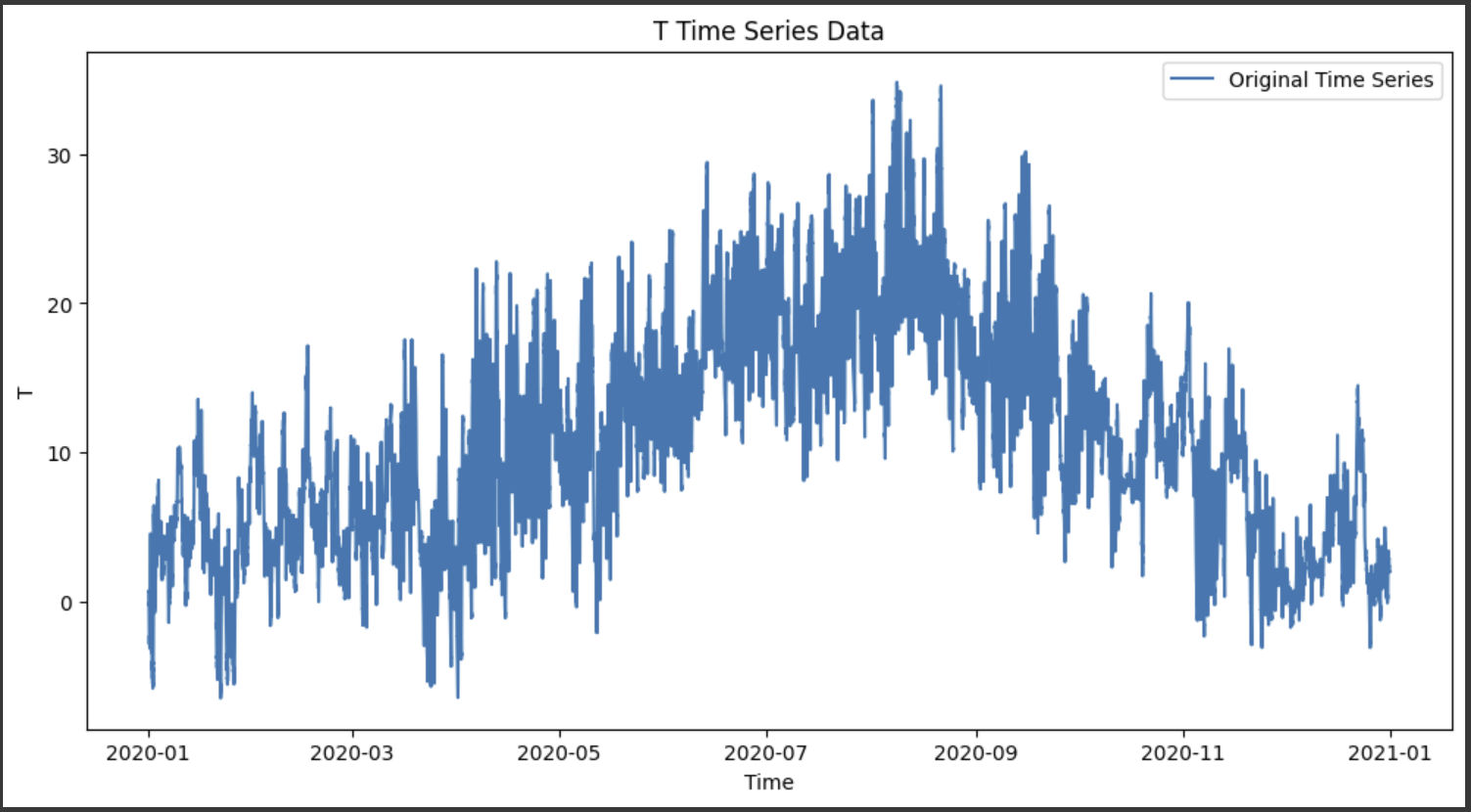
 plt.xlabel("Time")

 plt.ylabel(column\_name)

 plt.legend()

 plt.show()

**output:**



rolling\_window = 12  # Choose a window size

rolmean = ts.rolling(window=rolling\_window).mean()

rolstd = ts.rolling(window=rolling\_window).std()

    plt.figure(figsize=(12,6))

    plt.plot(ts, color="blue", label="Original")

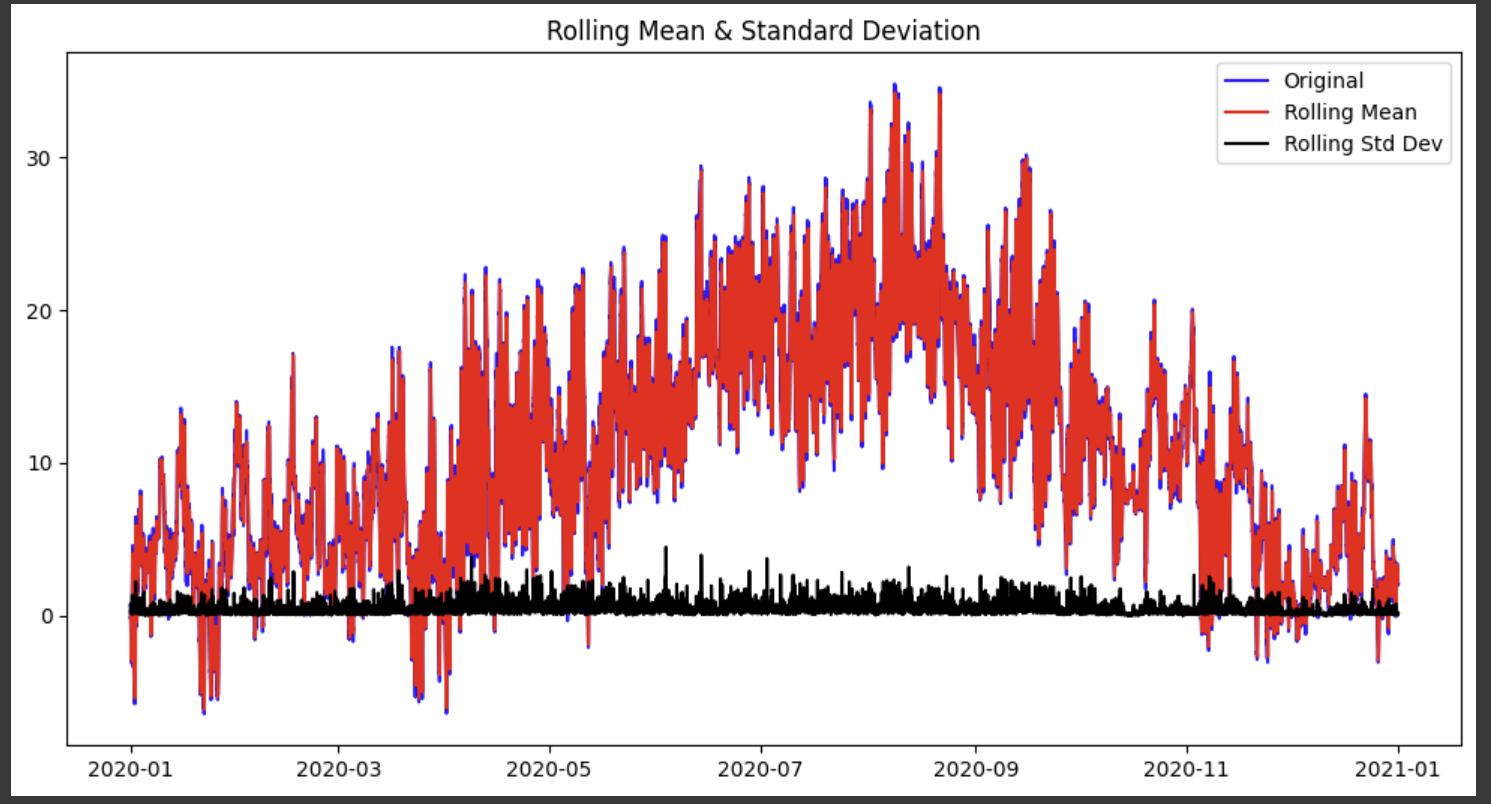
    plt.plot(rolmean, color="red", label="Rolling Mean")

    plt.plot(rolstd, color="black", label="Rolling Std Dev")

    plt.title("Rolling Mean & Standard Deviation")

    plt.legend()

    plt.show()

**output:**

def adf\_test(timeseries):

        print("\nResults of Augmented Dickey-Fuller Test:")

        adf\_result = adfuller(timeseries.dropna())  # Drop NaN values

        labels = ["Test Statistic", "p-value", "#Lags Used", "Number of Observations Used"]

        for value, label in zip(adf\_result[:4], labels):

            print(f"{label}: {value}")

        print("\nCritical Values:")

        for key, value in adf\_result[4].items():

            print(f"\t{key}: {value}")

        if adf\_result[1] <= 0.05:

            print("\nConclusion: The time series is STATIONARY (p-value <= 0.05)")

        else:

            print("\nConclusion: The time series is NON-STATIONARY (p-value > 0.05)")

   adf\_test(ts)

**OUTPUT:**

Results of Augmented Dickey-Fuller Test:

Test Statistic: -8.407443757648588

p-value: 2.1485277355859027e-13

#Lags Used: 58

Number of Observations Used: 52637

Critical Values:

1%: -3.43047423996295

5%: -2.8615949115726993

10%: -2.5667992276035014

Conclusion: The time series is STATIONARY (p-value <= 0.05)

RESULT:

Thus The program successfully analyzed the stationarity of the time series dataset. The ADF Test results were displayed, indicating whether the data is stationary. Rolling Mean and Standard Deviation plots were generated to visualize trends.