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 (if applicable).

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 necessary libraries
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


# Prompt user to upload the dataset
print("Please upload your dataset (CSV file).")
uploaded = files.upload()


# Get the uploaded filename
filename = list(uploaded.keys())[0]
print(f"Uploaded file: {filename}")


# Load dataset without parsing dates first
df = pd.read_csv(io.BytesIO(uploaded[filename]))


# Display column names
print("\nColumn names in the dataset:", df.columns.tolist())


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


# If a date column is provided, parse it as an index
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.")
```

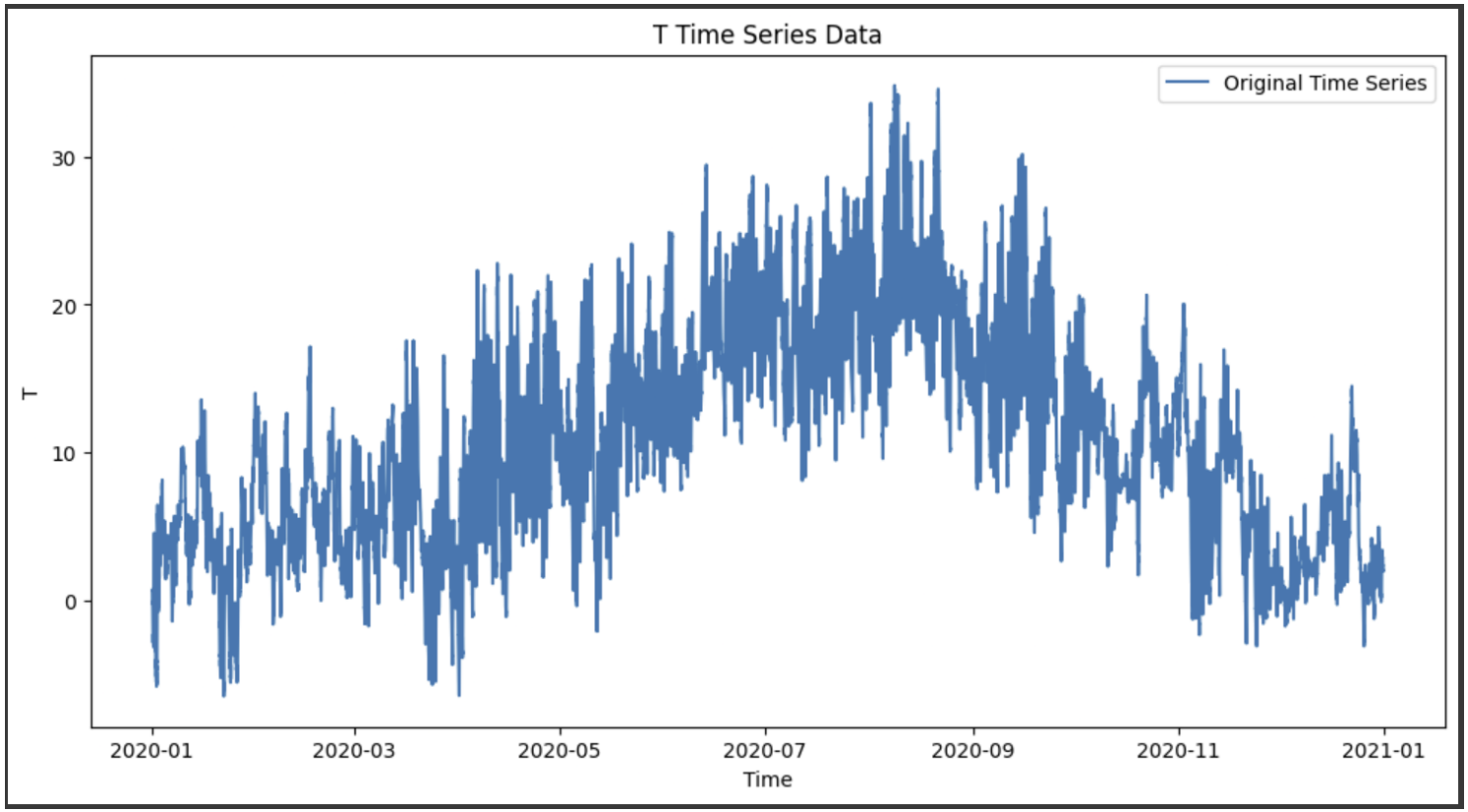
```
# Display first few rows
print("\nFirst few rows of the dataset:")
print(df.head())


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


# Check if column exists
if column_name not in df.columns:
  print(f"\nError: Column '{column_name}' not found in dataset.")
else:
  # Extract time series data
  ts = df[column_name]


  # Plot the original time series
  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()
```

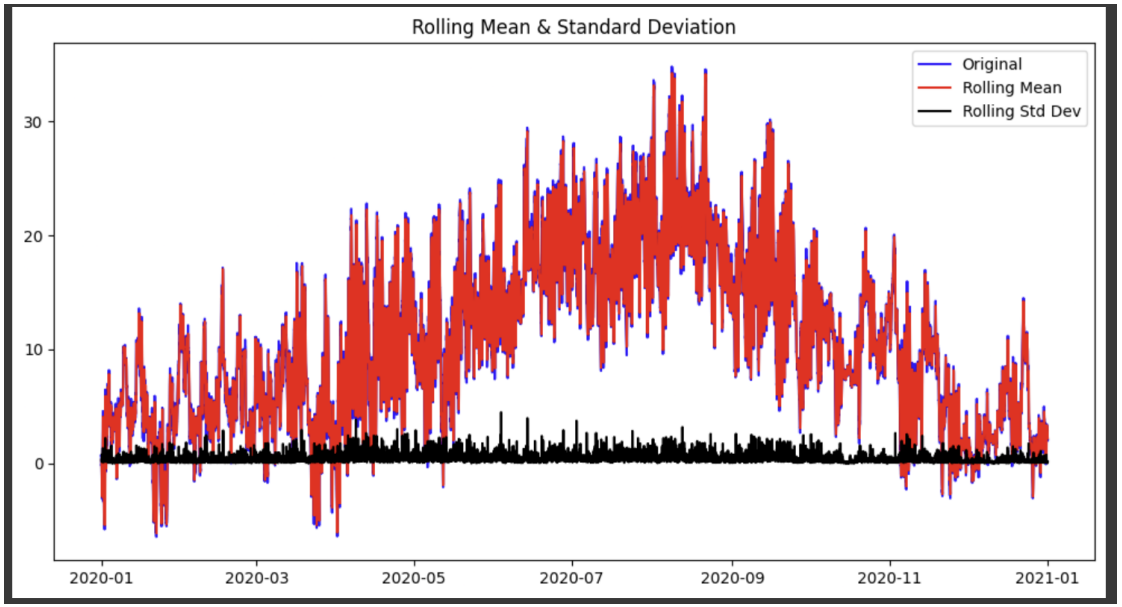
O/P:


T Time Series Data
Original Time Series
30
20
10
0
T
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01
Time

```
# Rolling statistics (Moving Average & Standard Deviation)
rolling_window = 12  # Choose a window size
rolmean = ts.rolling(window=rolling_window).mean()
rolstd = ts.rolling(window=rolling_window).std()


# Plot rolling statistics
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()
```

O/P:




```
# Perform Augmented Dickey-Fuller Test
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}")
```

```
    # Check stationarity
    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)")


# Run ADF test
adf_test(ts)
```

O/P:

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:

- 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.