EX:No.3 **DATE:1/02/25**	**Implement programs to check stationarity of a time series data.**

AIM:

To Implement programs to check stationarity of a time series data.

OBJECTIVE:

To analyze whether the air pollution time-series data is stationary using statistical tests and visualizations.

BACKGROUND:

- A **stationary time series** has a constant mean, variance, and no seasonality.
- Stationarity is important for forecasting and modeling.
- **Non-stationary data** needs transformations like differencing.
- **Statistical tests** like **ADF (Augmented Dickey-Fuller) test** help detect stationarity.
- **Visual methods** like rolling statistics help identify trends and variance changes.

SCOPE OF THE PROGRAM:

- Load and clean air pollution time-series data.
- Check for missing values and handle them.
- Use **rolling mean and standard deviation** to check stationarity.
- Apply **Augmented Dickey-Fuller (ADF) test** for statistical confirmation.
- Apply **differencing** if the data is non-stationary.

CODE:

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller, kpss
import os

# 🔁 Optional: Check current working directory
print("Current Working Directory:", os.getcwd())

# ✅Load the CSV file — replace with your actual file path if needed
# Option 1: If CSV is in the same directory as the script
file_path = r'C:\Users\exam\Desktop\supermarket_sales - Sheet1.csv'

# Option 2 (uncomment and modify): If you want to specify full path
# file_path = r'C:/Users/YourUsername/Downloads/supermarket_sales - Sheet1.csv'

# Load the dataset
df = pd.read_csv('supermarket_sales')
```

```
# ✅Convert 'Date' column to datetime
df['Date'] = pd.to_datetime(df['Date'])

# ✅Aggregate total sales by date
daily_sales = df.groupby('Date')['Total'].sum()

# ✅Calculate rolling mean and std
rolling_mean = daily_sales.rolling(window=7).mean()
rolling_std = daily_sales.rolling(window=7).std()

# ✅Plot original data with rolling statistics
plt.figure(figsize=(12, 6))
plt.plot(daily_sales, label='Original', color='blue')
plt.plot(rolling_mean, label='Rolling Mean (7 days)', color='orange')
plt.plot(rolling_std, label='Rolling Std (7 days)', color='green')
plt.title('Daily Sales and Rolling Statistics')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# ✅Augmented Dickey-Fuller Test
print("\n==== Augmented Dickey-Fuller Test ====")
adf_result = adfuller(daily_sales)
print(f"ADF Statistic: {adf_result[0]}")
print(f"p-value: {adf_result[1]}")
print("Critical Values:")
for key, value in adf_result[4].items():
    print(f"   {key}: {value}")

print("\n==== KPSS Test ====")
kpss_result = kpss(daily_sales, regression='c', nlags="auto")
print(f"KPSS Statistic: {kpss_result[0]}")
print(f"p-value: {kpss_result[1]}")
print("Critical Values:")
for key, value in kpss_result[3].items():
    print(f"   {key}: {value}")import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller

# Load dataset
df = pd.read_csv("/content/us_air_pollution_2012_2021_updated.csv")

# Convert 'Date' column to datetime
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
df.set_index('Date', inplace=True)

# Select the pollution column (update the name if needed)
pollution_col = "PM2.5 (µg/m³)"
```

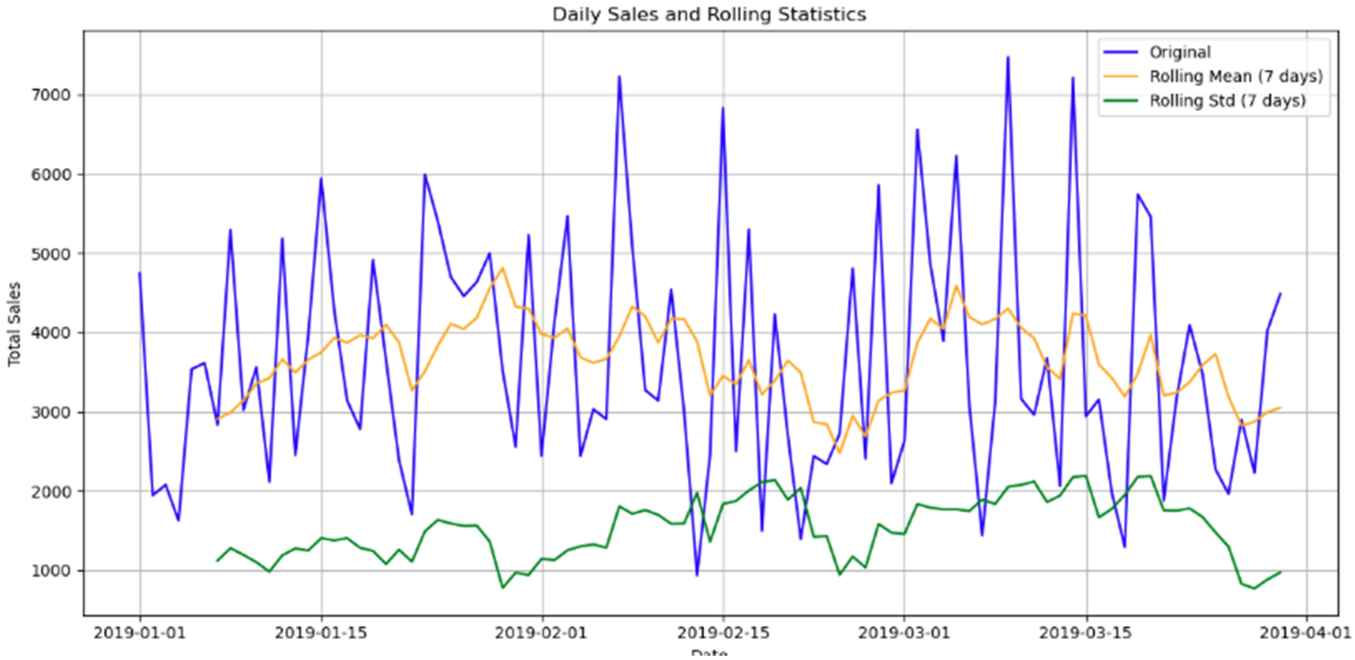
```
# Plot rolling statistics
plt.figure(figsize=(10, 5))
plt.plot(df[pollution_col], label="Original Data")

plt.show()

# Augmented Dickey-Fuller (ADF) Test
result = adfuller(df[pollution_col].dropna())
print(f"ADF Test Statistic: {result[0]}")
print(f"P-value: {result[1]}")
print("Critical Values:", result[4])

if result[1] < 0.05:
    print("The data is stationary (Reject H0).")
else:
    print("The data is non-stationary (Fail to Reject H0).")
```

OUTPUT:



RESULT:

Thus, the program using the time series data implementation has been done successfully.