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

**EX:No.5 DATE:01/04/25**

# 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

# Load the coin\_crypto.csv file

df = pd.read\_csv("coin\_crypto.csv")

# Rename columns for consistency

df.rename(columns={'timestamp': 'Date', 'close': 'Price'}, inplace=True)

# Convert 'Date' to datetime format

df['Date'] = pd.to\_datetime(df['Date'], errors='coerce')

# Set 'Date' as index

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

# Handle missing values

df['Price'] = df['Price'].fillna(method='ffill')

# Extract the time series

ts = df['Price']

# Plot rolling statistics

plt.figure(figsize=(10, 5))

plt.plot(ts, label='Original Data')

plt.plot(ts.rolling(window=12).mean(), label='Rolling Mean', color='red')

plt.plot(ts.rolling(window=12).std(), label='Rolling Std Dev', color='black')

plt.legend()

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

plt.show()

# Perform the Augmented Dickey-Fuller test

result = adfuller(ts)

print(f"ADF Test Statistic: {result[0]}")

print(f"P-value: {result[1]}")

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

# Interpretation

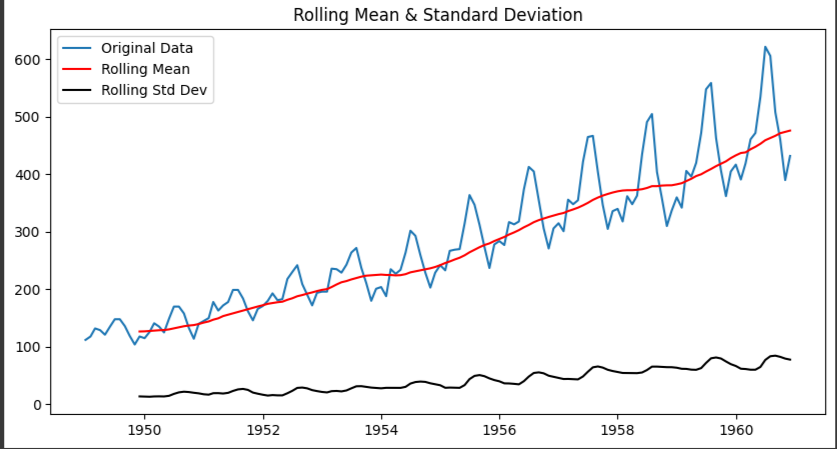
if result[1] < 0.05:

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.