EX:No.8 DATE:13/4/2025	**Create an ARIMA model for time series forecasting**

AIM:

To Implement program to create an ARIMA model for time series forecasting

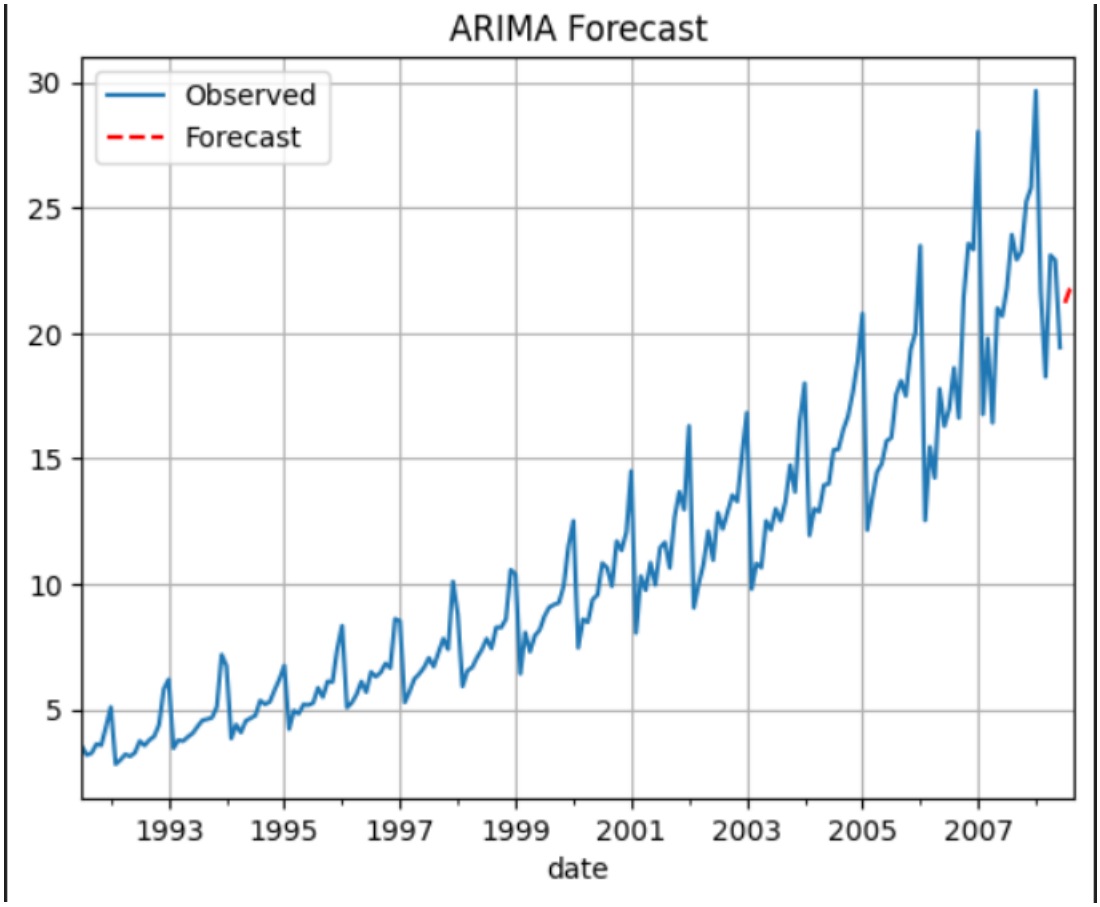
ALGORITHM:

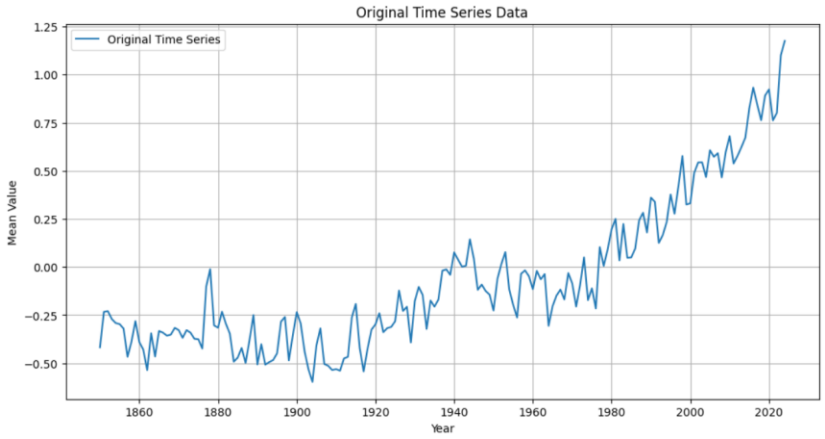
- **Step 1: Load and Visualize the Data**
- Import time series data.
- Plot the time series to understand patterns (trend, seasonality, noise).
- **Step 2: Check for Stationarity**
- Use **ADF test (Augmented Dickey-Fuller)**.
- If p-value > 0.05 → data is non-stationary → differencing needed.
- **Step 3: Make the Series Stationary**
- Apply differencing (data.diff()) until stationarity is achieved.
- Re-check with ADF test after each differencing.
- **Step 4: Plot ACF and PACF**
- **ACF** (AutoCorrelation Function): Suggests value of **q** (MA term).
- **PACF** (Partial ACF): Suggests value of **p** (AR term).
- **Step 5: Build ARIMA Model**
- Define order as **ARIMA(p, d, q)**:
  - p → AR order (from PACF)
  - d → differencing level
  - q → MA order (from ACF)
- Fit the model on training data.
- **Step 6: Forecast**
- Use .forecast() or .predict() to generate future values.
- Inverse differencing if needed to convert back to original scale.
- **Step 7: Evaluate the Model**
- Compare forecast vs actual using MAE, RMSE, or MAPE.
- Plot forecast vs actual for visual verification.

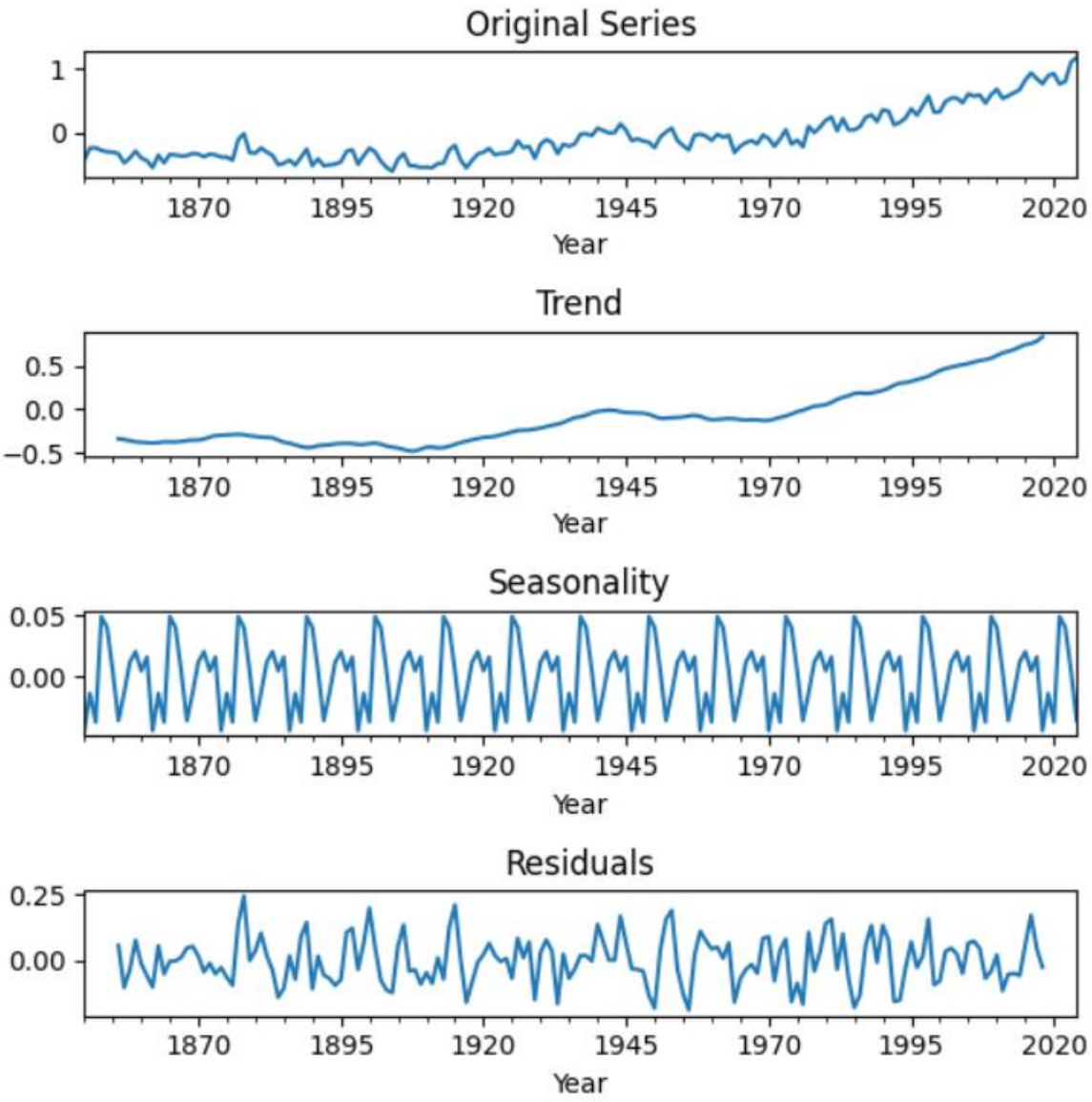
CODE:

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
import warnings
warnings.filterwarnings("ignore")

# Load data
df = pd.read_csv('sale.csv', parse_dates=['date'], index_col='date')
ts = df['value']

# Plot data
ts.plot(title='Time Series Data')
plt.xlabel('Date')
plt.ylabel('Value')
plt.grid(True)
plt.show()

# Check for stationarity
adf_result = adfuller(ts)
print("ADF Statistic:", adf_result[0])
print("p-value:", adf_result[1])

# Fit ARIMA model (choose order based on ACF/PACF for bigger datasets; here assume (1,1,1))
model = ARIMA(ts, order=(1, 1, 1))
model_fit = model.fit()

print(model_fit.summary())

# Forecast next 3 steps
forecast = model_fit.forecast(steps=3)
print("Forecasted values:")
print(forecast)

# Plot forecast
ts.plot(label='Observed')
forecast.plot(label='Forecast', style='--', color='red')
plt.legend()
plt.grid(True)
plt.title('ARIMA Forecast')
plt.show()
```

OUTPUT:


ARIMA Forecast
Observed
Forecast
30
25
20
15
10
5
1993
1995
1997
1999
2001
2003
2005
2007
date


Original Time Series Data
Original Time Series
Mean Value
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
1860
1880
1900
1920
1940
1960
1980
2000
2020
Year



RESULT:

Thus the program has been completed and verified successfully.