EX 7 IMPLEMENT PROGRAM FOR DECOMPOSING TIME SERIES DATA INTO TREND AND SEASONALITY

AIM:

To decompose the Weather time series dataset into trend, seasonality, and residual components using the multiplicative model.

ALGORITHM:

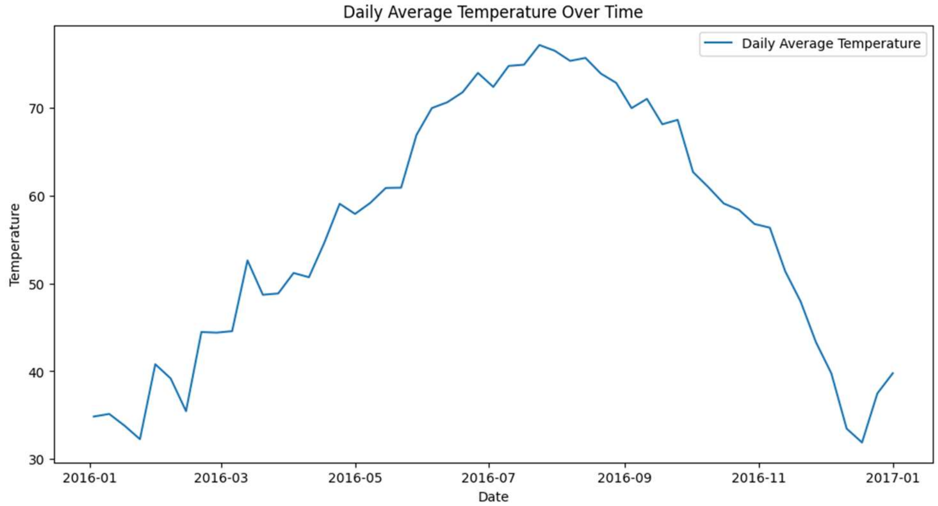
1. Load the weather.csv dataset, which contains daily weather data from multiple stations.
2. Preprocess the data by converting the Date.Full column to a datetime object and setting it as the index.
3. Group the data by the date and calculate the daily average temperature.
4. Handle missing values by setting the data to a daily frequency and filling gaps using linear interpolation.
5. Use the seasonal_decompose() function to decompose the daily temperature data into trend, seasonal, and residual components, assuming weekly or yearly seasonality based on the dataset's time span.
6. Visualize the decomposed components to analyze and understand the seasonal trends and fluctuations in the temperature data.

PROGRAM:

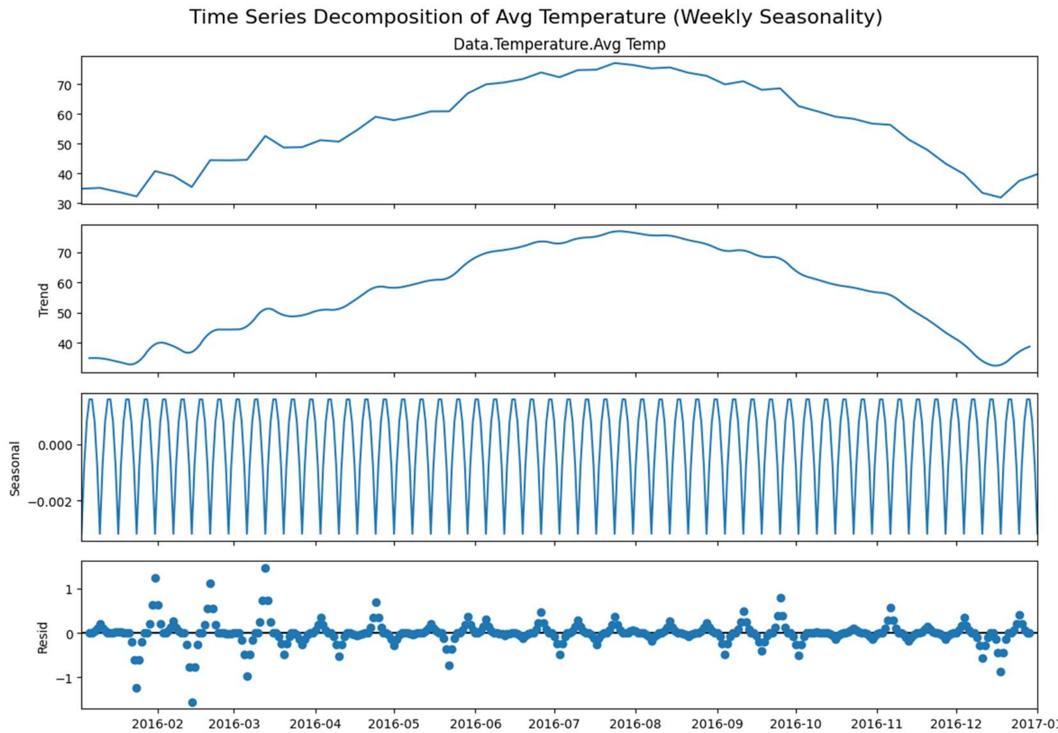
```
# Step 1: Import the necessary libraries

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.seasonal import seasonal_decompose


# Step 2: Load the dataset

df = pd.read_csv('/content/weather.csv')  # adjust path if needed

df.head()
```

	Data.Precipitation	Date.Full	Date.Month	Date.Week of	Date.Year	Station.City	Station.Code	Station.Location	Station.State	Data.Temperature.Avg Temp	Data.Temperature.Max Temp	Data.Temperature.Min Temp
0	0.00	2016-01-03	1	3	2016	Birmingham	BHM	Birmingham, AL	Alabama	39	46	32
1	0.00	2016-01-03	1	3	2016	Huntsville	HSV	Huntsville, AL	Alabama	39	47	31
2	0.16	2016-01-03	1	3	2016	Mobile	MOB	Mobile, AL	Alabama	46	51	41
3	0.00	2016-01-03	1	3	2016	Montgomery	MGM	Montgomery, AL	Alabama	45	52	38
4	0.01	2016-01-03	1	3	2016	Anchorage	ANC	Anchorage, AK	Alaska	34	38	29

```
# Step 3: Convert 'Date.Full' to datetime
df['Date.Full'] = pd.to_datetime(df['Date.Full'])


# Step 4: Group by Date and take the mean of 'Avg Temp' for each day
daily_avg = df.groupby('Date.Full')['Data.Temperature.Avg Temp'].mean()
# Convert to a time series with daily frequency
daily_avg = daily_avg.asfreq('D')
# Fill missing values using forward fill or interpolation
daily_avg = daily_avg.interpolate(method='linear')  # smoother than forward fill


# Step 5: Plot the Time Series to Visualize (check if still NaNs!)
plt.figure(figsize=(12, 6))
plt.plot(daily_avg, label='Daily Average Temperature')
plt.title('Daily Average Temperature Over Time')
plt.xlabel('Date')
plt.ylabel('Temperature')
plt.legend()
plt.show()
```




```
# Step 6 : Decompose the Time Series

result = seasonal_decompose(daily_avg, model='additive', period=7)

fig = result.plot()

fig.set_size_inches(12, 8)  # make it larger so text won't overlap

plt.tight_layout()

plt.suptitle('Time Series Decomposition of Avg Temperature (Weekly
Seasonality)', fontsize=16, y=1.02)

plt.show()
```



RESULT:

The time series decomposition successfully identified the trend, seasonal, and residual components in the daily average temperature data. The trend component shows the long-term temperature changes, the seasonal component highlights periodic fluctuations, and the residual component captures the random noise in the data.