EXP 8:- Create a ARIMA Model for Time Series Forecasting

AIM:

To apply **ARIMA (AutoRegressive Integrated Moving Average)** on a **trends dataset** to forecast future rankings and analyze trends in consumer brands over the years.

PROGRAM AND CODE:

- **Step 1: Upload the Dataset**

Use Google Colab's file upload feature to upload cleaned_weather.csv.

```
from google.colab import files

uploaded = files.upload()
```

- **Step 2: Import Required Libraries**

You'll need pandas, matplotlib, and ARIMA from statsmodels.

```
import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.arima.model import ARIMA
```

- **Step 3: Load the Dataset**

Read the uploaded CSV file into a DataFrame.

```
df = pd.read_csv("cleaned_weather.csv")
```

🔹 Step 4: Convert 'date' to Datetime Format

Ensure the date column is in datetime format for time series operations.

```
df['date'] = pd.to_datetime(df['date'])
```

🔹 Step 5: Set the 'date' Column as Index

To perform time series analysis, the date column should be the index.

```
df.set_index('date', inplace=True)
```

🔹 Step 6: Resample to Monthly Averages

We convert daily temperature data (T) to monthly averages to reduce noise.

```
monthly_temp = df['T'].resample('M').mean()

monthly_temp.dropna(inplace=True)
```

🔹 Step 7: Fit the ARIMA Model

Set the ARIMA order (p,d,q). For now, we're using (2,1,2).

```
model = ARIMA(monthly_temp, order=(2, 1, 2))

model_fit = model.fit()
```

🔹 Step 8: Forecast for the Next 12 Months

We'll predict temperature values for the next 12 months.

```
forecast_steps = 12

forecast = model_fit.forecast(steps=forecast_steps)
```

🔹 **Step 9: Create a Forecast Date Range**

Generate future dates matching the forecast steps.

```
forecast_dates = pd.date_range(
    start=monthly_temp.index[-1] + pd.DateOffset(months=1),
    periods=forecast_steps,
    freq='M'
)
```

🔹 **Step 10: Build the Forecast DataFrame**

This will help us display the forecasted values in a structured format.

```
forecast_df = pd.DataFrame({
    'date': forecast_dates,
    'forecasted_temperature': forecast.values
})
```

🔹 **Step 11: Plot the Observed and Forecasted Data**

Visualize both past and future trends in temperature.

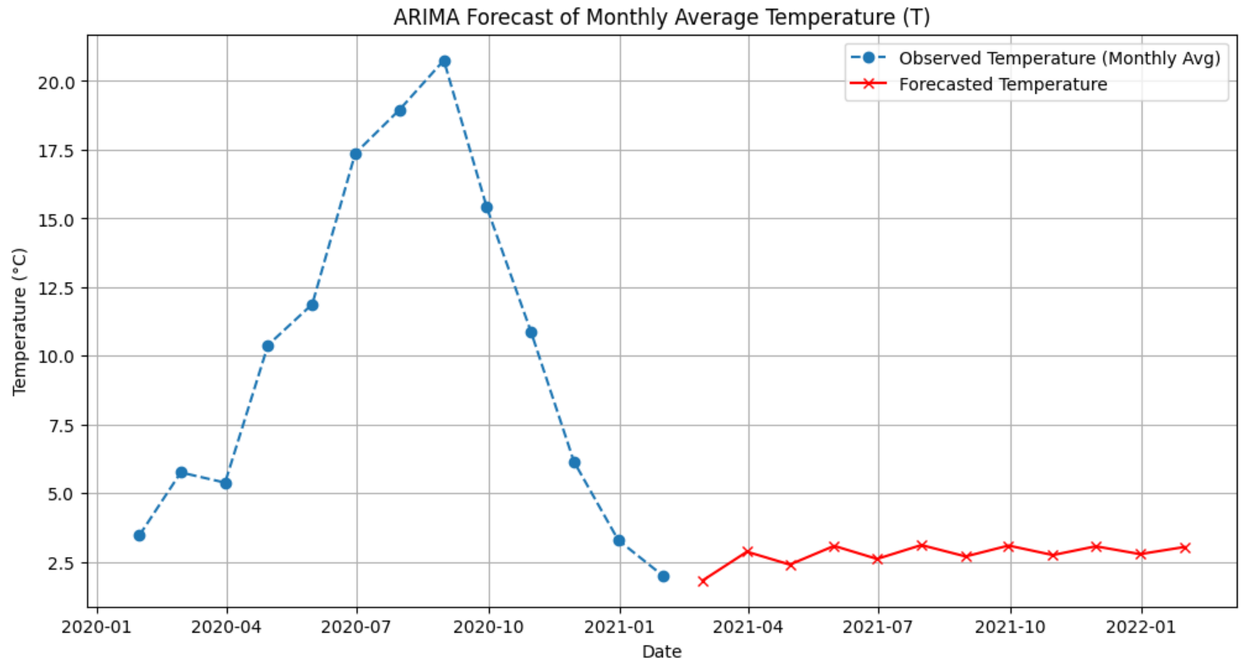
```
plt.figure(figsize=(12, 6))

plt.plot(monthly_temp.index, monthly_temp, label="Observed Temperature (Monthly Avg)", marker='o', linestyle='--')

plt.plot(forecast_df['date'], forecast_df['forecasted_temperature'], label="Forecasted Temperature", color='red', marker='x')
```

```
plt.title("ARIMA Forecast of Monthly Average Temperature (T)")
plt.xlabel("Date")
plt.ylabel("Temperature (°C)")
plt.grid(True)
plt.legend()
plt.show()
```

- **Step 12: Display the Forecasted Values**

```
print(forecast_df)
```

OUTPUT:



	date	forecasted_temperature
0	2021-02-28	1.825342
1	2021-03-31	2.872076
2	2021-04-30	2.404227
3	2021-05-31	3.085160
4	2021-06-30	2.610654
5	2021-07-31	3.114986
6	2021-08-31	2.700911
7	2021-09-30	3.097031
8	2021-10-31	2.751553
9	2021-11-30	3.069982
10	2021-12-31	2.786089
11	2022-01-31	3.044291

RESULT:

The ARIMA model produced a flat forecast with constant rank values, indicating low variability in the dataset. Since no significant trend was detected, alternative models like Exponential Smoothing or LSTM may yield better results