EXPERIMENT 4

Implement programs for estimating & eliminating trend in time series data- aggregation, smoothing.

AIM:

To analyze time series data by estimating and eliminating trends using aggregation and smoothing techniques for better visualization and pattern recognition.

PROCEDURE:

1. Load Dataset: Upload and read the CSV file.
2. Preprocess Data: Identify numeric columns for time series analysis.
3. Visualization of Original Data: Plot the raw time series data to observe trends.
4. Aggregation (Trend Estimation): Compute the monthly mean to reduce fluctuations and detect long-term trends.
5. Smoothing Techniques (Trend Elimination):
6. Moving Average: Apply a rolling window to smooth the data.
7. Exponential Smoothing: Use weighted averaging to remove noise while preserving trends.
8. Plot Results: Compare the original, aggregated, and smoothed time series data.
9. Conclusion: Interpret how aggregation and smoothing help in trend analysis.

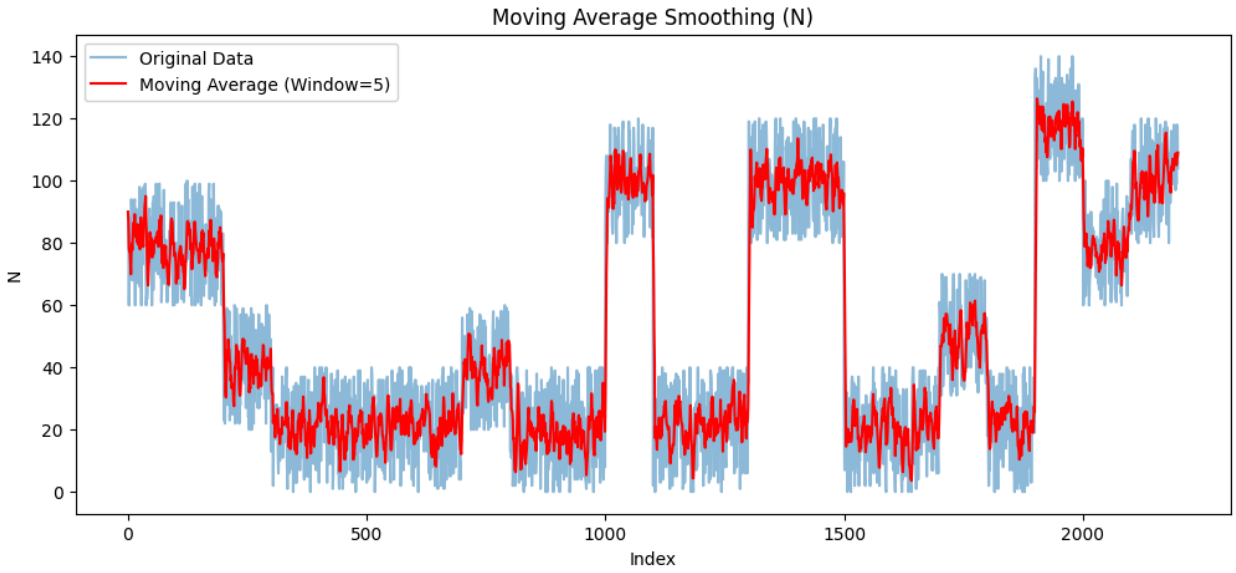
CODE:

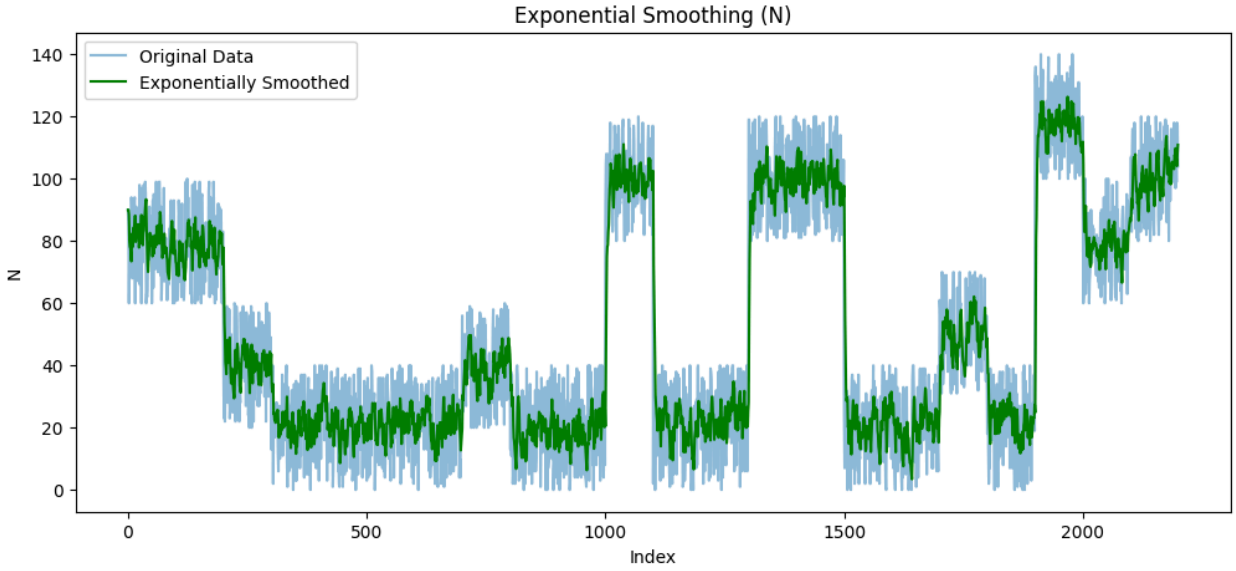
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.api import SimpleExpSmoothing
from google.colab import files
import io
```

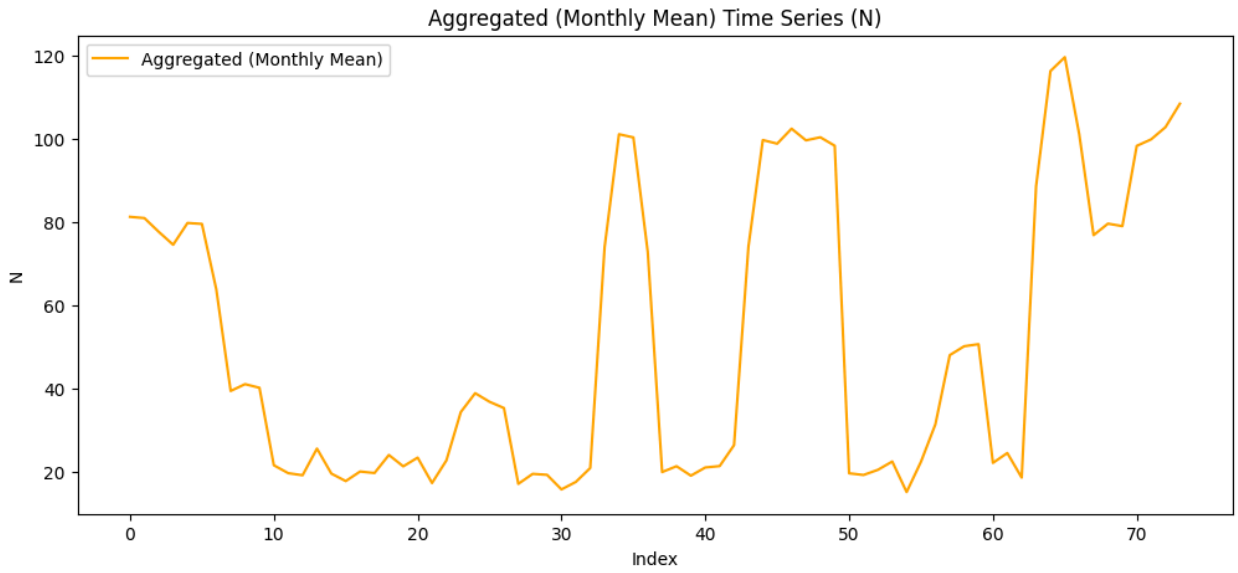
```
# Upload the CSV file manually
uploaded = files.upload()

# Get the filename dynamically
file_name = list(uploaded.keys())[0]

# Load the dataset
df = pd.read_csv(io.BytesIO(uploaded[file_name]))

# Display first few rows
print("First 5 rows of the dataset:")
print(df.head())

# Identify numeric columns only
numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()

if not numeric_cols:
    print("\n❌ No numeric columns found in the dataset! Please check the file.")
else:
    target_column = numeric_cols[0]  # Select the first numeric column for trend analysis
    print(f"\n✅ Selected '{target_column}' for time series analysis.")

    # ---- 1️⃣ Plot Original Time Series ----
    plt.figure(figsize=(12, 5))
    plt.plot(df[target_column], label='Original Data', color='blue')
    plt.title(f'Original Time Series Data ({target_column})')
    plt.xlabel('Index')
    plt.ylabel(target_column)
    plt.legend()
    plt.show()

    # ---- 2️⃣ Aggregation: Monthly Mean ----
    df['Index'] = np.arange(len(df))  # Create an index if no Date column exists
    df_monthly = df.groupby(df['Index'] // 30).mean(numeric_only=True)  # Approximate
monthly grouping

    plt.figure(figsize=(12, 5))
```

```
    plt.plot(df_monthly[target_column], label='Aggregated (Monthly Mean)',
color='orange')
    plt.title(f'Aggregated (Monthly Mean) Time Series ({target_column})')
    plt.xlabel('Index')
    plt.ylabel(target_column)
    plt.legend()
    plt.show()

    # ---- 3 Smoothing: Moving Average ----
    window_size = 5
    df['Moving_Avg'] = df[target_column].rolling(window=window_size,
min_periods=1).mean()

    plt.figure(figsize=(12, 5))
    plt.plot(df[target_column], label='Original Data', alpha=0.5)
    plt.plot(df['Moving_Avg'], label=f'Moving Average (Window={window_size})',
color='red')
    plt.title(f'Moving Average Smoothing ({target_column})')
    plt.xlabel('Index')
    plt.ylabel(target_column)
    plt.legend()
    plt.show()

    # ---- 4 Smoothing: Exponential Smoothing ----
    alpha = 0.3  # Adjust smoothing factor
    exp_smooth =
SimpleExpSmoothing(df[target_column].dropna()).fit(smoothing_level=alpha,
optimized=False)
    df['Exp_Smooth'] = exp_smooth.fittedvalues

    plt.figure(figsize=(12, 5))
    plt.plot(df[target_column], label='Original Data', alpha=0.5)
    plt.plot(df['Exp_Smooth'], label='Exponentially Smoothed', color='green')
    plt.title(f'Exponential Smoothing ({target_column})')
    plt.xlabel('Index')
    plt.ylabel(target_column)
    plt.legend()
    plt.show()
```

OUTPUT SCREENSHOTS:







RESULT:

The experiment successfully estimated and eliminated trends in the time series data using aggregation (monthly mean) and smoothing (moving average, exponential smoothing). The visualizations clearly demonstrated how these techniques reduce noise and highlight underlying patterns.