Experiment 4: Website Traffic Trend and Overlap Analysis

Aim : To analyze web traffic patterns and competitor overlap for three websites.

✅ Objectives

- Simulate traffic data for 3 competitor websites from Jan 1 to Mar 31, 2025.
- Plot:
  - Raw daily traffic trends.
  - Weekend-only behavior patterns.
- Calculate correlation between sites to estimate audience overlap.
- Display the results through a simple web dashboard using Flask.

Pseudo code

```
from flask import Flask
import pandas as pd
import numpy as np

app = Flask(__name__)

# 1. Generate dummy data from Jan 1 to Mar 31, 2025
def generate_data():
    dates = pd.date_range(start='2025-01-01', end='2025-03-31')
    np.random.seed(42)
    data = {
        'Date': dates,
        'Site_A_Visits': np.random.poisson(1200, len(dates)),
        'Site_B_Visits': np.random.poisson(1500, len(dates)),
        'Site_C_Visits': np.random.poisson(1000, len(dates)),
    }
    return pd.DataFrame(data)

# 2. Plot raw traffic for all dates
def plot_raw_traffic(df):
    # For each site, plot Date vs Visits
    # Set title: "Raw Daily Traffic"
    # Add legend
    pass  # Replace with matplotlib code in real program
```

```
# 3. Plot weekend-only traffic trend
def plot_weekend_trend(df):
    df['DayOfWeek'] = df['Date'].dt.dayofweek
    weekend_df = df[df['DayOfWeek'] >= 5]  # Keep Saturday and Sunday only

    # For each site, plot Date vs Visits from weekend_df
    # Set title: "Weekend Traffic Trend"
    # Add legend
    pass

# 4. Create correlation matrix
def get_correlation_matrix(df):
    # Return correlation between Site_A_Visits, Site_B_Visits, Site_C_Visits
    corr = df[['Site_A_Visits', 'Site_B_Visits', 'Site_C_Visits']].corr()
    return corr

@app.route('/')
def index():
    df = generate_data()
    df['Date'] = pd.to_datetime(df['Date'])

    # Step 1: Plot raw traffic
    plot_raw_traffic(df)

    # Step 2: Plot weekend traffic only
    plot_weekend_trend(df.copy())

    # Step 3: Show correlation matrix
    corr_matrix = get_correlation_matrix(df)
    print(corr_matrix)

    return "Dashboard with Traffic Analysis (Raw Traffic, Weekend Trend, Correlation Matr

if __name__ == '__main__':
    app.run(debug=True)
```

Python Code

```
from flask import Flask
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import io
import base64

app = Flask(__name__)
```

```
# Generate dummy traffic data from Jan 1 to Mar 31, 2025 (starts on a Wednesday)
def generate_data():
    dates = pd.date_range(start='2025-01-01', end='2025-03-31', freq='D')
    np.random.seed(42)
    data = {
        'Date': dates,
        'Site_A_Visits': np.random.poisson(lam=1200, size=len(dates)),
        'Site_B_Visits': np.random.poisson(lam=1500, size=len(dates)),
        'Site_C_Visits': np.random.poisson(lam=1000, size=len(dates)),
    }
    return pd.DataFrame(data)

# Plot 1: Raw daily traffic
def plot_raw_traffic(df):
    plt.figure(figsize=(10, 5))
    for site in ['Site_A_Visits', 'Site_B_Visits', 'Site_C_Visits']:
        plt.plot(df['Date'], df[site], label=site)
    plt.title('Daily Traffic (Raw Data)')
    plt.xlabel('Date')
    plt.ylabel('Visits')
    plt.legend()
    plt.tight_layout()

    img = io.BytesIO()
    plt.savefig(img, format='png')
    img.seek(0)
    raw_img = base64.b64encode(img.read()).decode('utf8')
    plt.close()
    return raw_img

# Plot 2: Weekend trend (only Saturdays and Sundays)
def plot_weekend_trend(df):
    df['Date'] = pd.to_datetime(df['Date'])
    df['DayOfWeek'] = df['Date'].dt.dayofweek
    weekend_df = df[df['DayOfWeek'] >= 5]  # 5 = Saturday, 6 = Sunday

    plt.figure(figsize=(10, 5))
    for site in ['Site_A_Visits', 'Site_B_Visits', 'Site_C_Visits']:
        plt.plot(weekend_df['Date'], weekend_df[site], marker='o', label=f'{site} (Weeken
    plt.title('Weekend Traffic Trend (Saturdays & Sundays Only)')
    plt.xlabel('Date')
    plt.ylabel('Visits')
    plt.legend()
    plt.tight_layout()

    img = io.BytesIO()
    plt.savefig(img, format='png')
```

```
    img.seek(0)
    weekend_img = base64.b64encode(img.read()).decode('utf8')
    plt.close()
    return weekend_img

# Plot 3: Correlation matrix
def get_correlation_html(df):
    corr = df[['Site_A_Visits', 'Site_B_Visits', 'Site_C_Visits']].corr()
    return corr.round(2).to_html(classes="table table-bordered", border=0)

@app.route('/')
def index():
    df = generate_data()
    raw_img = plot_raw_traffic(df)
    weekend_img = plot_weekend_trend(df.copy())
    correlation_html = get_correlation_html(df)

    html = f"""
    <!DOCTYPE html>
    <html>
    <head>
        <title>CI Lab Dashboard</title>
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/cs
    </head>
    <body class="p-4">
        <h2>📊 Competitive Intelligence: Website Traffic Analysis (Jan–Mar 2025)</h2>

        <h4>1️⃣ Raw Daily Traffic</h4>
        <img src="{raw_img}" class="img-fluid my-3"/>

        <h4>2️⃣ Weekend Traffic Trend</h4>
        <img src="{weekend_img}" class="img-fluid my-3"/>

        <h4>3️⃣ Site Overlap – Correlation Matrix</h4>
        <div class="table-responsive">
            {correlation_html}
        </div>
    </body>
    </html>
    """
    return html

if __name__ == '__main__':
    app.run(debug=True)
```