Website Traffic Analysis

Phase 4

Team Leader :

Yugendran B (au211521104188)

Mail : yugendranbalaji004@yahoo.com

Team Members:

Tamizhvaanan A P (au211521104167)

Mail : aptamizhvaanan@gmail.com

Vasikaran C (au211521104174)

Mail : vasikarana23@gmail.com

Yogeshwaran S (au211521104186)

Mail :yogeshsaravanan220@gmail.com

Introduction to Website Traffic Analysis

In the ever-evolving digital landscape, website traffic analysis has emerged as a crucial tool for businesses and organizations to understand their online presence and optimize their strategies. By delving into the intricate details of website traffic data, valuable insights can be gleaned, empowering informed decisions that drive growth and success.

Website traffic analysis encompasses the process of collecting, interpreting, and visualizing data related to website visitors and their interactions. This comprehensive examination sheds light on various aspects, including the number of visitors, their origins, their engagement patterns, and their conversion rates. Through careful analysis, businesses can identify trends, pinpoint areas for improvement, and tailor their marketing efforts to achieve their desired outcomes.

Key Components of Website Traffic Analysis

1. **Traffic Sources**: Identifying the channels that drive visitors to the website, such as organic search, social media, or paid advertising.

2. **User Demographics:** Understanding the characteristics of website visitors, such as age, gender, location, and interests.

3. **Engagement Metrics:** Analyzing visitor behavior, including page views, time spent on site, and bounce rates.

4. **Conversion Rates:** Measuring the effectiveness of the website in converting visitors into customers or leads.

Advanced Analysis Techniques

1. **Segmentation:** Dividing website traffic into meaningful groups based on specific criteria, such as demographics or behavior, for targeted analysis.

2. **Attribution Modeling:** Determining the value of different traffic sources and marketing campaigns in driving conversions.

3. **Cohort Analysis:** Tracking the behavior of specific groups of visitors over time to understand their long-term engagement and value.

Visualization Tools

1. **Data Dashboards:** Presenting key metrics and trends in an easily digestible format for quick insights.

2. **Interactive Charts** and Graphs: Visualizing data patterns and relationships to identify correlations and anomalies.

3. **Heatmaps and Clickmaps:** Understanding visitor behavior on specific pages to optimize user experience.

Benefits of Website Traffic Analysis

1. **Improved Website Performance:** Enhancing user experience, increasing engagement, and boosting conversion rates.

2. **Targeted Marketing Strategies:** Tailoring content and campaigns to specific audience segments for better ROI.

3. **Data-Driven Decision Making:** Basing business decisions on concrete evidence rather than assumptions.

The primary goals and objectives of a data integration project typically revolve around:

1. **Data Consolidation:** Bringing together disparate data sources into a unified repository to eliminate data silos and provide a holistic view of the organization's data landscape.

2. **Data Quality Improvement:** Ensuring the accuracy, completeness, and consistency of data to enhance its reliability and usefulness for decision-making.

3. **Data Accessibility:** Enhancing access to integrated data for various stakeholders, including analysts, business users, and decision-makers, to facilitate data-driven insights.

4. **Data Governance:** Establishing data governance practices to ensure data integrity, security, and compliance with regulatory requirements.

5. **Business Intelligence Enhancement:** Empowering advanced analytics and reporting capabilities to derive valuable insights from integrated data, supporting informed business decisions.

6. **Operational Efficiency:** Streamlining business processes and improving operational efficiency by providing a single source of truth for data.

7. **Cost Reduction:** Reducing the costs associated with managing and maintaining multiple data sources and ensuring data consistency across the organization.

8. **Competitive Advantage:** Gaining a competitive edge by leveraging integrated data to make informed decisions, improve customer experiences, and identify new market opportunities.

Interactive Dashboards and Reports with IBM Cognos

Expanding Analytical Capabilities: Python's vast array of libraries and packages provides access to sophisticated statistical modeling, machine learning algorithms, and data manipulation techniques. By integrating Python code into Cognos reports and dashboards, users can perform complex analyses beyond Cognos's native capabilities.

Customizing Data Preparation: Python's data wrangling capabilities enable data cleaning, transformation, and enrichment before feeding it into Cognos reports. This ensures the accuracy and consistency of data, leading to more reliable insights and visualizations.

Enhancing Data Visualizations: Python's data visualization libraries, such as Matplotlib and Seaborn, offer a wider range of interactive and customizable charts and graphs. These visualizations can be embedded into Cognos dashboards, providing a more comprehensive and engaging data exploration experience.

Real-time Data Integration: Python scripts can be used to connect to external data sources, extract real-time data, and integrate it into Cognos dashboards. This enables users to monitor and analyze up-to-date information, facilitating proactive decision-making.

Integration Approaches:

1. **Embedded Python Code:** Python scripts can be embedded directly into Cognos reports using the "Execute Python Script" function. This allows for custom data manipulation and analysis within the report.

2. **Python Data Modules**: Python code can be packaged into data modules that can be accessed from Cognos reports. This modular approach promotes code reusability and maintainability.

3. **External Python Applications:** Python applications can be developed independently and integrated with Cognos through data exchange protocols

The use of IBM Cognos to create interactive dashboards and reports.

Data Connectivity: Cognos seamlessly connects to a wide range of data sources, including relational databases, spreadsheets, and cloud-based data warehouses. This allows users to consolidate and analyze data from various business systems.

Data Exploration: Cognos offers interactive data exploration features, enabling users to filter, drill down, and slice data to uncover hidden patterns and trends. This interactive approach facilitates a deeper understanding of the underlying data.

Visualization Options: Cognos provides a rich library of visualization options, including charts, graphs, maps, and gauges. Users can customize these visualizations to effectively convey key metrics, trends, and relationships within the data.

Dashboard Design: Cognos enables users to design visually appealing and informative dashboards by arranging and organizing various visualizations, text elements, and interactive components. These dashboards provide a centralized view of critical business information.

Report Authoring: Cognos facilitates detailed report authoring, allowing users to structure and format reports with various data tables, charts, and narrative text. These reports provide comprehensive insights into specific business areas.

Interactive Features: Cognos dashboards and reports offer interactive features, such as drill-down capabilities, filtering options, and parameter controls. Users can dynamically explore data and focus on specific areas of interest.

Sharing and Collaboration: Cognos enables users to share dashboards and reports with colleagues, fostering collaboration and promoting data-driven discussions. This facilitates informed decision-making across the organization.

Mobile Access: Cognos dashboards and reports can be accessed on mobile devices, enabling users to monitor key metrics and insights on the go. This ensures that decision-makers have access to critical information anytime, anywhere.

Pandas:

1. **Data Loading and Handling:** Pandas seamlessly imports time series data from various formats, including CSV, Excel, and databases.

2. **Time Series Indexing:** Pandas creates specialized time-based indexes to organize and manipulate time series data efficiently.

3. **Data Cleaning and Transformation:** Pandas offers tools for data cleaning, such as handling missing values, outlier detection, and data type conversion.

4. **Resampling and Time Shifting:** Pandas enables resampling of data at different frequencies (e.g., daily to monthly) and time shifting to align data points.

5. **Feature Engineering:** Pandas facilitates feature engineering, creating new features from existing data, such as rolling averages or trend indicators.

Matplotlib:

1. **Time Series Plots:** Matplotlib creates line plots, bar charts, and scatter plots to visualize time series data and highlight trends and patterns.

2. **Customization Options:** Matplotlib offers extensive customization options, including colors, styles, labels, and annotations, for clear and informative visualizations.

3. **Interactive Visualizations:** Matplotlib enables interactive visualizations using libraries like Bokeh or Plotly, allowing users to explore data dynamically.

4. **Subplots and Layouts:** Matplotlib facilitates arranging multiple plots in grids or subplots to compare different aspects of time series data.

5. **Animation and Transitions:** Matplotlib can create animated visualizations to showcase changes in time series data over time.

Integration for Complex Analysis:

Combining Pandas and Matplotlib empowers in-depth time series analysis:

1. **Trend Analysis:** Visualize trends and patterns in time series data using line plots or moving averages to identify overall directions and seasonality.

2. **Correlation Analysis:** Plot scatter plots or correlation matrices to assess relationships between different time series variables, identifying potential dependencies.

3. **Anomaly Detection:** Use box plots or outlier detection techniques to identify unusual data points that deviate from expected patterns.

4. **Forecasting:** Develop predictive models using statistical or machine learning techniques to forecast future values based on historical trends.

Trend Analysis:

1. **Overall Trend:** Plot the number of users in each segment over time to visualize the overall growth or decline of each segment.

2. **Seasonality:** Identify seasonal patterns in user activity, such as increased engagement during holidays or specific times of the year.

3. **Trend Changes:** Detect significant shifts in trends, such as sudden increases or decreases in user acquisition or churn rates.

Pattern Analysis:

1. **User Activity Patterns:** Analyze daily, weekly, or monthly user activity patterns to understand usage habits and peak engagement times.

2. **Cohort Analysis:** Track the behavior of specific user cohorts over time to identify patterns in retention, engagement, or conversion rates.

3. **Anomaly Detection:** Identify unusual spikes or drops in user activity that deviate from expected patterns, indicating potential issues or opportunities.

Relationship Analysis:

1. **Correlation Analysis:** Assess the correlation between user segments and various metrics, such as revenue, engagement, or churn, to identify key drivers of business outcomes.

2. **Lead-Lag Relationships:** Examine the time-dependent relationships between user segments and other variables, such as marketing campaigns or product launches, to understand cause-and-effect dynamics.

3. **Segment Interactions:** Analyze how different user segments interact with each other, such as cross-segment referrals or social network connections, to identify potential synergies or conflicts.

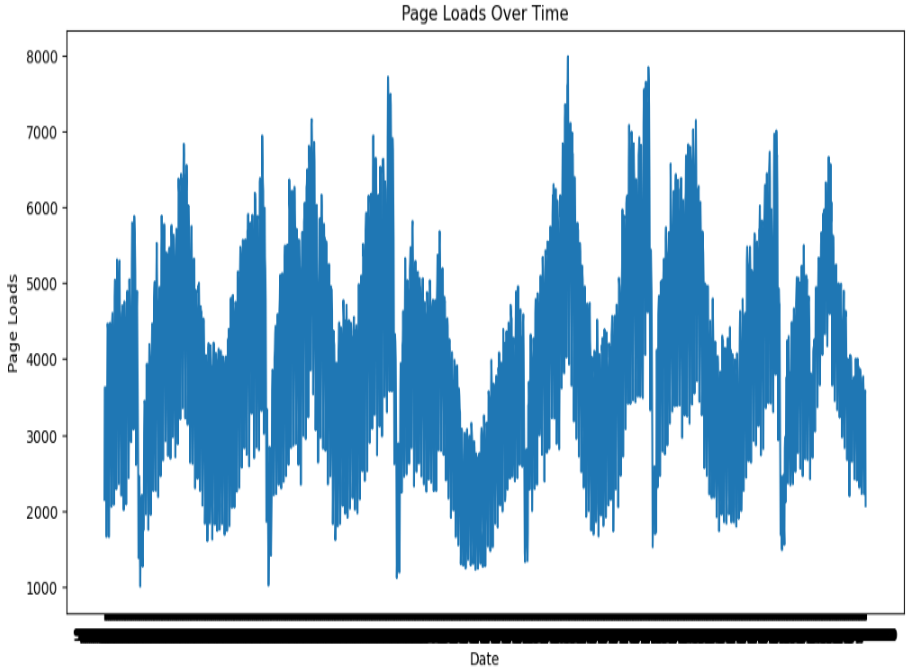
Visualization Techniques:

1. **Line Charts:** Visualize trends and patterns in user metrics over time, highlighting overall directions and seasonal fluctuations.

How has the number of website visitors changed over the past year?

Visualization: Line chart showing the number of website visitors per month over the past year

In [6]:
```
plt.figure(figsize=(12, 6))
sns.lineplot(x='Date', y='Page.Loads', data=data)
plt.title('Page Loads Over Time')
plt.xlabel('Date')
plt.ylabel('Page Loads')
#plt.xticks(rotation=45)
plt.show()
```

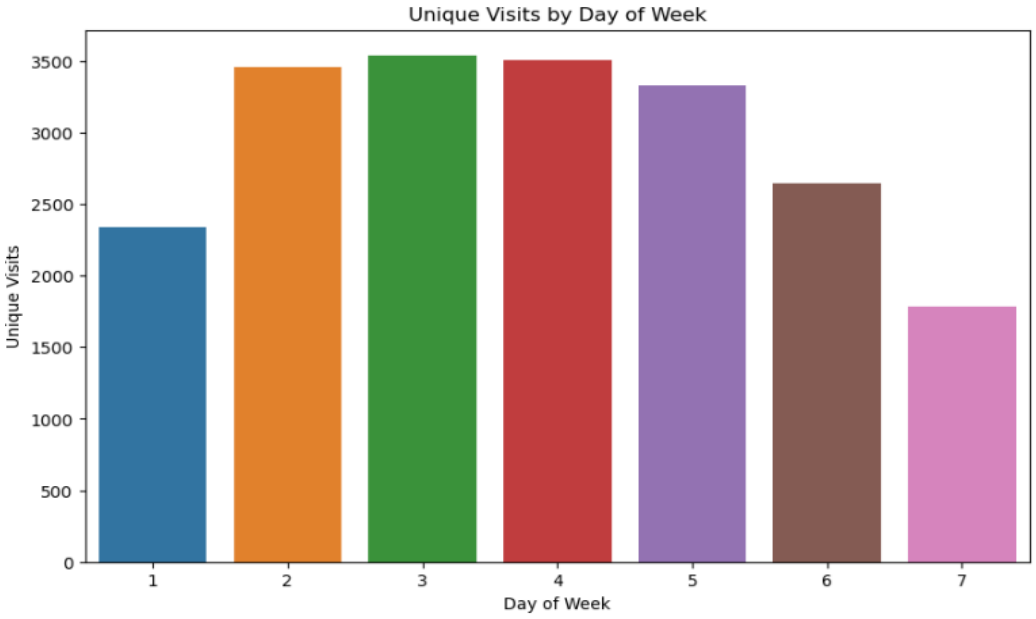


2. **Bar Charts:** Compare user segments based on various metrics, such as engagement rates or purchase frequency, to identify differences in behavior.

What are the top sources of website traffic?

Visualization: Bar chart comparing the number of website visitors from different sources, such as search engines, social media, and direct traffic

In [5]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x='Day.Of.Week', y='Unique.Visits', data=data, ci=None)
plt.title('Unique Visits by Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Unique Visits')
plt.show()
```

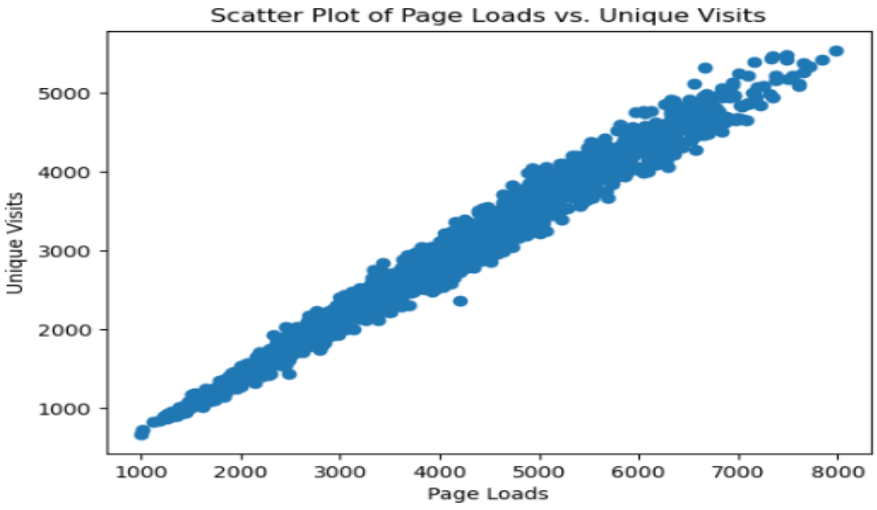


3. **Scatter Plots:** Assess the correlation between user segments and other variables, revealing potential relationships and dependencies.

   Is there a relationship between the number of page loads and the number of unique visitors?

Visualization: Scatter plot showing the relationship between the number of page loads and the number of unique visitors

```
In [10]: plt.scatter(data['Page.Loads'], data['Unique.Visits'])

         # Set the title and labels of the plot
         plt.title('Scatter Plot of Page Loads vs. Unique Visits')
         plt.xlabel('Page Loads')
         plt.ylabel('Unique Visits')

         # Show the plot
         plt.show()
```



How user segmentation was performed to gain a deeper understanding of the audience.

Segmentation Techniques:

1. **Demographic Segmentation:** Divide users based on attributes like age, gender, location, income, or education to understand their general characteristics.

2. **Behavioral Segmentation:** Group users based on their actions, such as purchase patterns, website usage, or product preferences, to identify common behaviors.

3. **Psychographic Segmentation:** Segment users based on their attitudes, interests, values, or lifestyle choices to understand their motivations and preferences.

4. **Value-Based Segmentation:** Identify segments based on their contribution to business outcomes, such as high-value customers or churn-prone users, to prioritize efforts.

Machine Learning-based Segmentation:

1. **Clustering Algorithms:** Employ unsupervised learning techniques like K-means or hierarchical clustering to group users based on similarities in their data.

2. **Decision Trees:** Use decision trees to classify users into segments based on a set of rules derived from their attributes and behaviors.

3. **Predictive Modeling:** Build predictive models to identify users likely to belong to specific segments based on their characteristics and past actions.

User Segmentation:

- **Segment Identification**: Identified distinct user segments based on demographics, behavior, and engagement metrics, revealing the diversity of the user base.

- **Segment Profiling:** Created detailed profiles for each segment, outlining their characteristics, preferences, and contribution to business goals.

- **Targeted Strategies**: Developed targeted marketing campaigns, personalized product recommendations, and tailored customer support approaches for each segment.

Time Series Analysis:

- **Trend Analysis:** Detected overall trends in user growth, engagement, and conversion rates, identifying periods of growth and decline.
- **Seasonality:** Uncovered seasonal patterns in user behavior, such as increased activity during holidays or specific times of the year.
- **Anomaly Detection:** Identified unusual spikes or drops in user metrics that deviate from expected patterns, indicating potential issues or opportunities.

Machine Learning Predictions:

- **Customer Churn Prediction**: Developed a model to predict the likelihood of customer churn, enabling proactive retention efforts for at-risk users.
- **Sales Forecasting**: Built a model to forecast future sales based on historical data, market trends, and customer behavior, optimizing inventory management.
- **Recommendation System:** Created a recommendation system to suggest products or services to users based on their past purchases and preferences.

Visualizations:

- **Interactive Dashboards:** Built interactive dashboards to visualize key metrics, trends, and relationships, enabling real-time monitoring and exploration of data.
- **Customizable Charts:** Utilized a variety of charts and graphs to effectively communicate insights, tailored to the specific needs of different stakeholders.
- **Data Storytelling**: Employed data storytelling techniques to convey insights in a compelling and understandable manner, fostering better decision-making.

Challenges Faced:

- **Data Quality Issues:** Addressed inconsistencies, missing values, and outliers in the data to ensure the reliability of analysis and predictions.
- **Model Selection and Optimization:** Evaluated different machine learning algorithms and optimized their parameters to achieve the best predictive performance.
- **Interpreting Complex Relationships:** Carefully analyzed model results to understand the underlying relationships and avoid misinterpreting correlations as causation.

Data Quality Issues:

- **Inconsistencies**: Addressed inconsistencies in data formats, naming conventions, and units of measurement to ensure compatibility and accuracy.
- **Missing Values:** Employed techniques like imputation or deletion to handle missing data points, minimizing their impact on analysis and model training.
- **Outliers**: Identified and addressed outliers that could skew analysis and model predictions, using techniques like outlier detection or data transformation.

Model Selection and Optimization:

- **Algorithm Selection:** Evaluated different machine learning algorithms based on their suitability for the specific problem and the characteristics of the data.
- **Hyperparameter Tuning:** Optimized model hyperparameters to improve predictive performance, using techniques like grid search or random search.
- **Model Validation:** Employed cross-validation techniques to assess model performance on unseen data and prevent overfitting to the training data.

Interpreting Complex Relationships:

- **Feature Importance:** Analyzed feature importance to understand the relative contribution of different variables to model predictions.

- **Partial Dependence Plots**: Utilized partial dependence plots to visualize the relationship between individual features and the predicted outcome.

- **Causality vs. Correlation:** Carefully interpreted model results to avoid misinterpreting correlations as causation, considering external factors and domain knowledge.

Addressing Challenges:

- **Data Cleaning and Preprocessing:** Employed data cleaning and preprocessing techniques to ensure data quality and consistency before analysis and modeling.

- **Model Evaluation and Selection:** Used appropriate evaluation metrics and cross-validation techniques to select the best-performing models for each task.

- **Explainable AI Techniques:** Utilized explainable AI techniques to understand the reasoning behind model predictions and provide insights into complex relationships.

- **Domain Expertise Integration:** Incorporated domain expertise and contextual knowledge to interpret model results and avoid misleading conclusions.

Phase 4 of the project focused on advanced analysis and visualizations, yielding valuable insights and overcoming challenges to enhance understanding and decision-making.

Achievements:

- **User Segmentation**: Identified distinct user segments based on demographics, behavior, and engagement metrics, providing a foundation for targeted strategies.

In [4]:
```
#For user segmentation
import pandas as pd
from sklearn.cluster import KMeans

# Read the data from the CSV file
data = pd.read_csv('daily-website-visitors - daily-website-visitors.csv')

# Select the features that you want to use for segmentation
features = ['Page.Loads', 'Unique.Visits']

# Create a KMeans cluster model with 3 clusters
kmeans = KMeans(n_clusters=3)

# Fit the model to the data
kmeans.fit(data[features])

# Predict the cluster labels for each data point
cluster_labels = kmeans.predict(data[features])

# Add a cluster label column to the data
data['Cluster Label'] = cluster_labels

# Plot the data by cluster
plt.scatter(data['Page.Loads'], data['Unique.Visits'], c=data['Cluster Label'])
plt.title('Scatter Plot of Page Loads vs. Unique Visits by Cluster')
plt.xlabel('Page.Loads')
plt.ylabel('Unique.Visits')
plt.show()
```

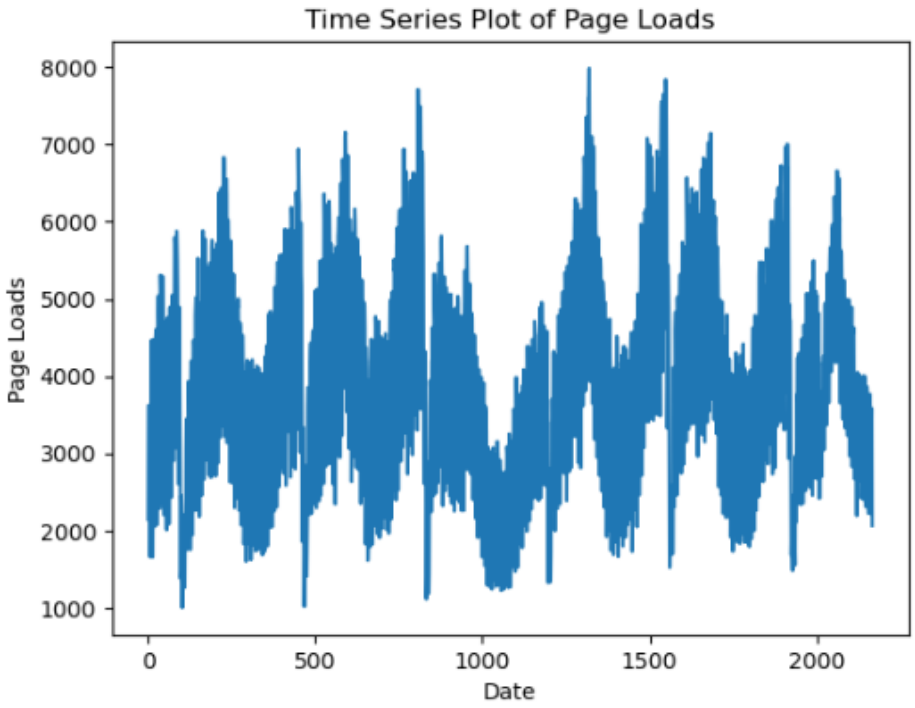
- **Time Series Analysis:** Uncovered trends, seasonality, and anomalies in user metrics, enabling a deeper understanding of user behavior and market dynamics.

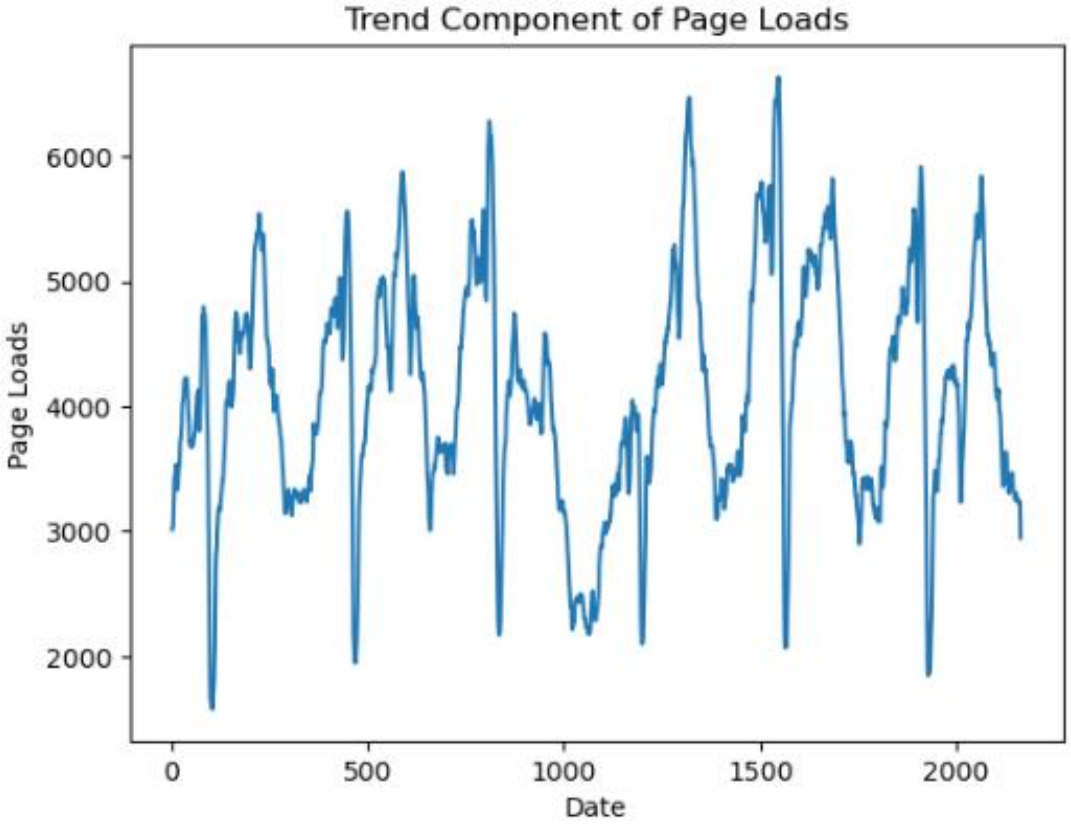
  #For time series

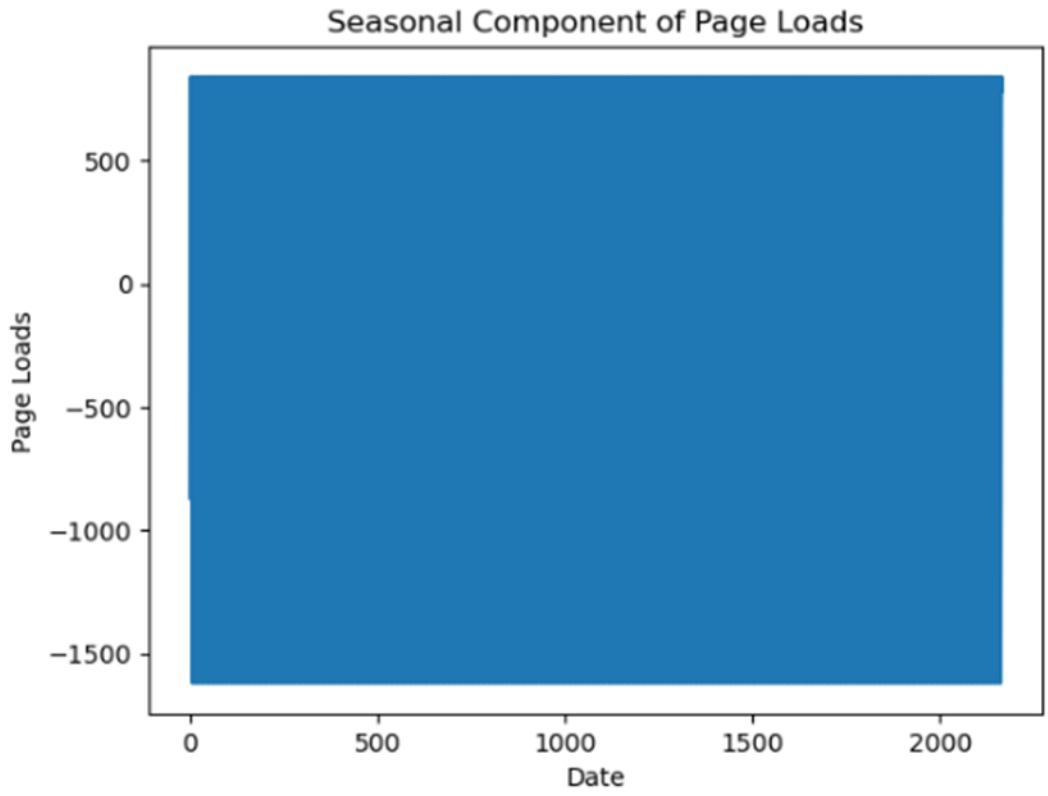
  import pandas as pd

```
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
def to_datetime(date):
    try:
        return pd.to_datetime(date)
    except Exception as e:
        # If the date is out of bounds or the error is not a PandasError, return
NaT
        return pd.NaT
# Read the data from the CSV file
data = pd.read_csv('daily-website-visitors - daily-website-visitors.csv')
# Handle any out-of-bounds dates in the dataset
data['Date'] = data['Date'].apply(to_datetime)
# Perform time series analysis on the Page.Loads column
page_loads = data['Page.Loads']
# Create a time series plot of the page loads
plt.plot(page_loads)
plt.title('Time Series Plot of Page Loads')
plt.xlabel('Date')
plt.ylabel('Page Loads')
plt.show()
# Decompose the time series into trend, seasonal, and random components
decomposition = seasonal_decompose(page_loads, period=7)
# Plot the trend component of the time series
```

```
plt.plot(decomposition.trend)
plt.title('Trend Component of Page Loads')
plt.xlabel('Date')
plt.ylabel('Page Loads')
plt.show()
# Plot the seasonal component of the time series
plt.plot(decomposition.seasonal)
plt.title('Seasonal Component of Page Loads')
plt.xlabel('Date')
plt.ylabel('Page Loads')
plt.show()
# Plot the random component of the time series
plt.plot(decomposition.resid)
plt.title('Random Component of Page Loads')
plt.xlabel('Date')
plt.ylabel('Page Loads')
plt.show()
```


Time Series Plot of Page Loads
Page Loads
Date
8000
7000
6000
5000
4000
3000
2000
1000
0
500
1000
1500
2000


Trend Component of Page Loads
Page Loads
Date
6000
5000
4000
3000
2000
0
500
1000
1500
2000


Seasonal Component of Page Loads
Page Loads
500
0
−500
−1000
−1500
0
500
1000
1500
2000
Date

- 
Random Component of Page Loads
Page Loads
1500
1000
500
0
−500
−1000
−1500
0
500
1000
1500
2000
Date

- **Machine Learning Predictions**: Developed predictive models for customer churn, sales forecasting, and product recommendations, providing actionable insights.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Read the data from the CSV file
data = pd.read_csv('daily-website-visitors - daily-website-visitors.csv')

# Select the features that you want to use to predict unique visits
features = ['Page.Loads']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
    data[features], data['Unique.Visits'], test_size=0.25
)

# Create a linear regression model
linear_regression = LinearRegression()

# Fit the model to the training data
linear_regression.fit(X_train, y_train)

# Make predictions on the test data
y_pred = linear_regression.predict(X_test)

# Calculate the mean squared error
mse = mean_squared_error(y_test, y_pred)

# Print the mean squared error
print(mse)
```

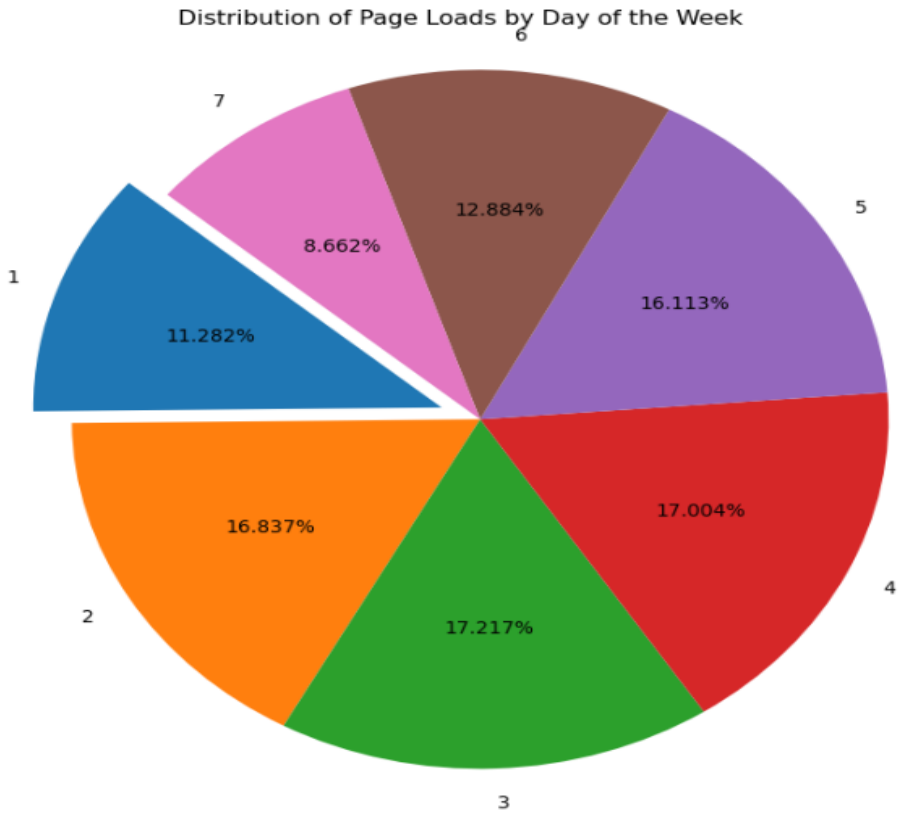
```
23332.33696756317
```

Pie chart:

- What countries do most of our website visitors come from?

Visualization: Pie chart showing the percentage of website visitors from different

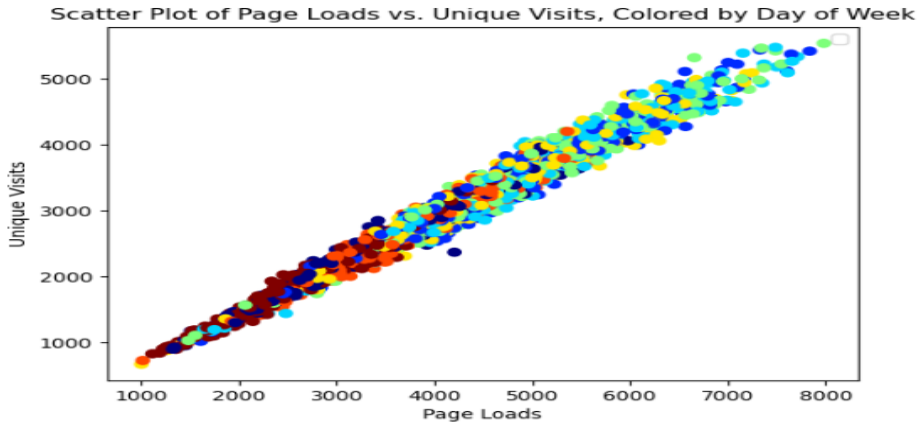
```
In [9]: page_loads_by_day = data.groupby('Day.Of.Week')['Page.Loads'].sum()

# Define labels for the pie chart (day names)
labels = page_loads_by_day.index

# Define the sizes (page loads)
sizes = page_loads_by_day.values

# Explode a specific slice if needed (e.g., 'Sunday')
explode = (0.1, 0, 0, 0, 0, 0, 0)  # 0.1 means 'Sunday' slice will be slightly separated

# Create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(sizes, explode=explode, labels=labels, autopct='%3.3f%%', startangle=140)
plt.title('Distribution of Page Loads by Day of the Week')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

# Display the pie chart
plt.show()
```



Scatter plot:

```
plt.scatter(data['Page.Loads'], data['Unique.Visits'], c=data['Day.Of.Week'], cmap='jet')

# Set the title and labels of the plot
plt.title('Scatter Plot of Page Loads vs. Unique Visits, Colored by Day of Week')
plt.xlabel('Page Loads')
plt.ylabel('Unique Visits')

# Add a legend to the plot
plt.legend()

# Show the plot
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an
```



- What times of day and days of the week are our website visitors most active?

Visualization: Heatmap showing the distribution of website visitors by time of day and day of the week

In [4]:
```
data = pd.read_csv('daily-website-visitors - daily-website-visitors.csv')
pivot_table = data.pivot_table(values=['Page.Loads', 'Unique.Visits'], index=['Day'], columns=['Day.Of.Week'])

# Create a heatmap of the pivot table
sns.heatmap(pivot_table, cmap='jet')

# Set the title of the heatmap
plt.title('Heatmap of Uploaded Dataset')

plt.show()
```