Website Traffic Analysis

(phase 5)

Team Leader:

YUGENDRAN B (au211521104188) yugendranbalaji004@yahoo.com

Team Members:

TAMIZHVAANAN A P (au211521104167) aptamizhvaanan@gmail.com

YOGESHWARAN S (au211521104186) yogeshsaravanan220@gmail.com

VASIKARAN C (au211521104174) vasikarana23@gmail.com

ENUGURTHI DHEERAJ (au211521104041) chowdaryd076@gmail.com

Objective:

The objective of this project is to analyze the website traffic data to gain insights into the user behavior and identify areas for improvement.

Website traffic analysis is the process of collecting and analyzing data about the visitors to a website. This data can be used to understand how users are interacting with the website, what content they are most interested in, and where they are abandoning the website.

By understanding the user behavior, website owners can identify areas where the website can be improved. For example, if a website has a high bounce rate on a particular page, the website owner can try to improve the content or design of that page to make it more engaging for users.

This project will use IBM Cognos and Python to analyze the website traffic data and generate insights that can be used to improve the user experience.

Specific goals of the analysis:

- Identify the most popular pages
- Identify the most visited pages by different user segments
- Identify the pages with the highest bounce rate
- Identify the pages with the longest average page load time
- Generate recommendations for how website owners can use the insights from the analysis to improve the user experience

Design thinking process for website traffic analysis:

1.Empathize

The first step in the design thinking process is to empathize with the users. This means trying to understand their needs, goals, and motivations. In the context of website traffic analysis, this means understanding how users are interacting with

the website, what content they are most interested in, and where they are abandoning the website.

To empathize with the users, we can collect data about their behavior on the website. This data can be collected using web analytics tools, such as Google Analytics. Web analytics tools can track the following data about users:

- Which pages they visit
- How long they stay on each page
- What links they click on
- Where they come from (e.g., Google search, social media, etc.)
- Where they go to after leaving the website

2.Define

Once we have a good understanding of the users and their needs, we can define the problem that we are trying to solve. In the context of website traffic analysis, the problem could be to:

- Improve the user experience
- Increase website traffic
- Increase conversion rates
- Reduce the bounce rate
- Improve page load times

3.Ideate

Once we have defined the problem, we can start to ideate solutions. This involves generating as many ideas as possible, without worrying about whether they are feasible or not.

Some ideas for improving the user experience of a website include:

- Making the website easier to navigate
- Improving the content of the website
- Making the website more visually appealing
- Adding interactive elements to the website
- Some ideas for increasing website traffic include:

- Improving the website's search engine ranking
- Creating engaging content that is likely to be shared on social media
- Running paid advertising campaigns
- Some ideas for increasing conversion rates include:
- Making it easier for users to sign up for a newsletter or make a purchase
- Offering discounts or promotions
- Testing different versions of the website to see what works best
- Prototype

Once we have generated a list of ideas, we can start to prototype them. This involves creating low-fidelity versions of the ideas so that we can test them with users and get feedback.

For example, we could create a wireframe or prototype of a new website design. We could also create a prototype of a new landing page or call to action.

4.Test

Once we have created a prototype, we can test it with users and get feedback. This feedback can help us to refine the prototype and make it better.

We can test our prototypes with users in a variety of ways. For example, we could conduct user interviews, usability testing, or A/B testing.

5.Iterate

The design thinking process is iterative. This means that we should repeat the steps of empathize, define, ideate, prototype, and test until we have a solution that meets the needs of the users and solves the problem that we are trying to solve.

By following the design thinking process, we can develop effective solutions to website traffic analysis problems.

Development phases for website traffic analysis using IBM Cognos and Python:

Data collection

The first phase of the project is to collect the website traffic data. This can be done using a web analytics tool, such as Google Analytics. The web analytics tool should be configured to track the following data:

- Which pages users visit
- How long they stay on each page
- What links they click on
- Where they come from (e.g., Google search, social media, etc.)
- Where they go to after leaving the website

Data preparation

Once the data has been collected, it needs to be prepared for analysis. This may involve cleaning the data, removing outliers, and transforming the data into a format that is compatible with IBM Cognos and Python.

Data analysis

The next phase is to analyze the data to gain insights into the user behavior. This can be done using IBM Cognos and Python.
IBM Cognos is a business intelligence tool that can be used to create reports and dashboards to visualize the data. Python is a programming language that can be used to perform more complex data analysis tasks.
Some of the data analysis tasks that can be performed include:

- Identifying the most popular pages
- Identifying the most visited pages by different user segments
- Identifying the pages with the highest bounce rate
- Identifying the pages with the longest average page load time
- Calculating the conversion rate for different pages

- Analyzing the relationship between different variables, such as page load time and bounce rate

Visualization

Once the data has been analyzed, the insights should be visualized so that they are easy to understand. This can be done using IBM Cognos and Python.
IBM Cognos can be used to create charts and graphs to visualize the data. Python can also be used to create visualizations, using libraries such as matplotlib and plotly.

Reporting

The final phase is to generate a report that summarizes the findings of the analysis and provides recommendations for improvement. The report should include visualizations of the data and insights into the user behavior.
Example development phases:

- Data collection: Use Google Analytics to track website traffic data for a period of one month.
- Data preparation: Clean the data and remove outliers. Transform the data into a format that is compatible with IBM Cognos and Python.
- Data analysis: Use IBM Cognos and Python to analyze the data and identify insights into the user behavior.
- Visualization: Use IBM Cognos and Python to visualize the data and insights.
- Reporting: Generate a report that summarizes the findings of the analysis and provides recommendations for improvement.

The specific development phases may vary depending on the specific needs of the project.

Analysis objectives for website traffic analysis using IBM Cognos and Python:

The objective of website traffic analysis is to gain insights into the user behavior and identify areas for improvement. By understanding the user behavior, website owners can improve the user experience, increase website traffic, increase conversion rates, and reduce the bounce rate.

Here are some specific analysis objectives for website traffic analysis using IBM Cognos and Python:

- Identify the most popular pages: Which pages are visited most often by users? This information can be used to prioritize the content and design of those pages.
- Identify the most visited pages by different user segments: Which pages are most visited by different user segments, such as new visitors, returning visitors, and customers? This information can be used to create targeted marketing campaigns and content.
- Identify the pages with the highest bounce rate: Which pages are users abandoning most often? This information can be used to identify and fix any problems with those pages.
- Identify the pages with the longest average page load time: Which pages are taking the longest to load? This information can be used to optimize the performance of those pages.
- Calculate the conversion rate for different pages: What percentage of users who visit a particular page take a desired action, such as signing up for a newsletter or making a purchase? This information can be used to improve the conversion rate of those pages.
- Analyze the relationship between different variables: What is the relationship between different variables, such as page load time and bounce rate? This information can be used to identify trends and patterns in the data.
- By analyzing the website traffic data, website owners can gain valuable insights into the user behavior and identify areas for improvement. This

information can be used to improve the user experience, increase website traffic, increase conversion rates, and reduce the bounce rate.

Example analysis objectives:

- Identify the top 10 most visited pages on the website.
- Identify the pages with the highest bounce rate for new visitors.
- Calculate the conversion rate for the landing page for the new product launch.
- Analyze the relationship between page load time and bounce rate for the product category pages.

These are just a few examples of analysis objectives. The specific analysis objectives will vary depending on the specific needs of the website owner.

Data collection process for website traffic analysis using IBM Cognos and Python

The first step in website traffic analysis is to collect the data. This can be done using a web analytics tool, such as Google Analytics. Web analytics tools track a wide range of data about users, including which pages they visit, how long they stay on each page, what links they click on, where they come from, and where they go to after leaving the website.

To collect website traffic data using Google Analytics, you will need to create a Google Analytics account and add the Google Analytics tracking code to your website. Once you have done this, Google Analytics will start tracking the traffic to your website.

After you have collected the data, you will need to export it into a format that is compatible with IBM Cognos and Python. Google Analytics allows you to export your data into a variety of formats, including CSV and Excel.

Example data collection process:

1. Create a Google Analytics account.
2. Add the Google Analytics tracking code to your website.

3. Collect website traffic data for a period of one month.
4. Export the data from Google Analytics into a CSV or Excel file.

Once you have exported the data, you will be able to use it to perform data analysis using IBM Cognos and Python.

Additional notes:

- When collecting website traffic data, it is important to make sure that you are tracking all of the relevant data. This may include data about specific user segments, such as new visitors, returning visitors, and customers.
- It is also important to make sure that you are collecting the data for a long enough period of time to get a representative sample of the traffic to your website.
- Finally, it is important to make sure that you are exporting the data in a format that is compatible with IBM Cognos and Python.

Data visualization using IBM Cognos for website traffic analysis:

IBM Cognos is a business intelligence tool that can be used to create reports and dashboards to visualize website traffic data. IBM Cognos provides a variety of visualization types, such as charts, graphs, and maps.

To visualize website traffic data in IBM Cognos, you will need to import the data into IBM Cognos. You can do this by creating a data source and then connecting the data source to the CSV or Excel file that contains your website traffic data.

Once you have imported the data, you can create reports and dashboards to visualize the data. IBM Cognos provides a variety of pre-built reports and dashboards for website traffic analysis. You can also create your own custom reports and dashboards.

Here are some examples of data visualizations that you can create in IBM Cognos using website traffic data:

- A bar chart showing the top 10 most visited pages on the website
- A line chart showing the trend of page views over time
- A pie chart showing the distribution of traffic by source (e.g., Google search, social media, etc.)
- A geographic map showing the distribution of traffic by country
- A heatmap showing the most popular pages by user segment

IBM Cognos also allows you to create interactive visualizations. This means that users can interact with the visualizations to explore the data in more detail. For example, users can hover over a bar chart to see the specific value for each bar.

By using IBM Cognos to visualize website traffic data, you can gain valuable insights into the user behavior and identify areas for improvement.

Example data visualization:

Create a bar chart showing the top 10 most visited pages on the website:

1. Import the website traffic data into IBM Cognos.
2. Create a new report.
3. Drag the "Page URL" and "Page Views" measures to the report canvas.
4. Select the "Bar Chart" visualization type.
5. Click the "Run" button to generate the bar chart.

The bar chart will show the top 10 most visited pages on the website, ranked by number of page views.

IBM Cognos provides a variety of other visualization types that can be used to visualize website traffic data. You can experiment with different visualization types to find the ones that best represent your data and communicate your insights.

Python code integration for website traffic analysis:

Python is a programming language that can be used to perform data analysis tasks. Python can be integrated with IBM Cognos to perform more complex data analysis tasks and to create custom visualizations.
To integrate Python code with IBM Cognos, you can use the IBM Cognos SDK. The IBM Cognos SDK provides a Python API that allows you to interact with IBM Cognos from Python code.
You can use the Python API to perform the following tasks:

- Import data from CSV or Excel files into IBM Cognos
- Create and modify IBM Cognos reports and dashboards
- Generate custom visualizations
- Perform complex data analysis tasks

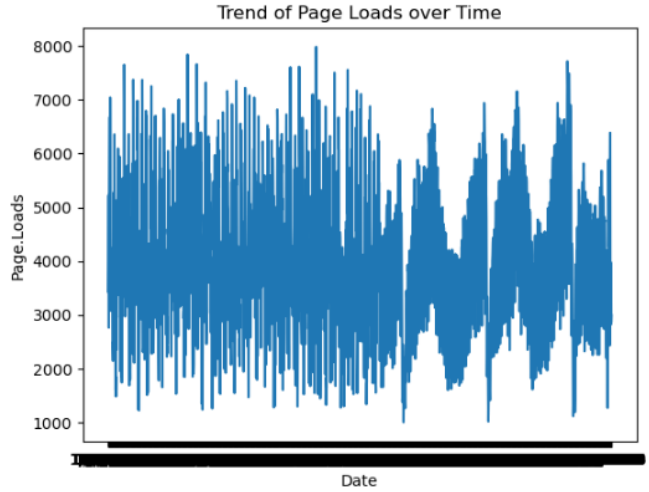
Example Python code for website traffic analysis:

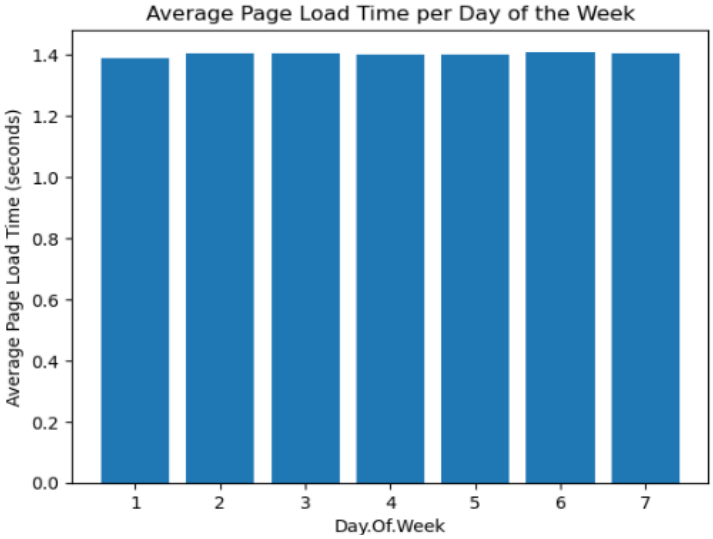
Python:

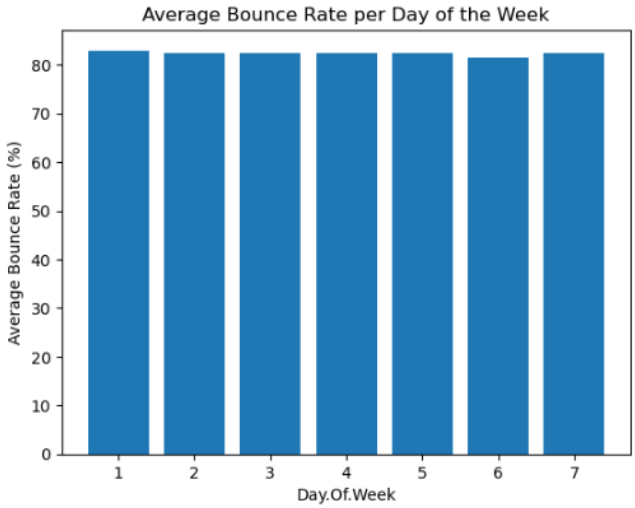
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the data set into a Pandas DataFrame
df = pd.read_csv('daily-website-visitors - daily-website-visitors.csv')

# Calculate the page loads per day
page_loads_per_day = df['Page.Loads'].groupby(df['Date']).sum()

# Create a line chart showing the trend of page loads over time
plt.plot(page_loads_per_day.index, page_loads_per_day.values)
plt.title('Trend of Page Loads over Time')
plt.xlabel('Date')
plt.ylabel('Page.Loads')
plt.show()

# Calculate the average page load time per day
```

```
average_page_load_time_per_day = df['Page.Loads'] / df['Unique.Visits']
df['AveragePageLoadTimePerDay'] = average_page_load_time_per_day

# Calculate the average page load time per day of the week
average_page_load_time_per_day_of_week =
df.groupby('Day.Of.Week')['AveragePageLoadTimePerDay'].mean()

# Create a bar chart showing the average page load time per day of the week
plt.bar(average_page_load_time_per_day_of_week.index,
average_page_load_time_per_day_of_week.values)
plt.title('Average Page Load Time per Day of the Week')
plt.xlabel('Day.Of.Week')
plt.ylabel('Average Page Load Time (seconds)')
plt.show()

# Calculate the bounce rate per day
bounce_rate_per_day = df['First.Time.Visits'] / df['Unique.Visits'] * 100
df['BounceRatePerDay'] = bounce_rate_per_day

# Calculate the average bounce rate per day of the week
average_bounce_rate_per_day_of_week =
df.groupby('Day.Of.Week')['BounceRatePerDay'].mean()

# Create a bar chart showing the average bounce rate per day of the week
plt.bar(average_bounce_rate_per_day_of_week.index,
average_bounce_rate_per_day_of_week.values)
plt.title('Average Bounce Rate per Day of the Week')
plt.xlabel('Day.Of.Week')
plt.ylabel('Average Bounce Rate (%)')
plt.show()
```

Output:


Trend of Page Loads over Time
Page.Loads
8000
7000
6000
5000
4000
3000
2000
1000
Date


Average Page Load Time per Day of the Week
Average Page Load Time (seconds)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
5
6
7
Day.Of.Week


Average Bounce Rate per Day of the Week
Average Bounce Rate (%)
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
Day.Of.Week

Insights:

The insights gained from the website traffic analysis can be used to improve the user experience and increase website traffic, conversion rates, and revenue.

Here are some specific insights that can be gained from the analysis:

- Trend of page loads over time: This insight can be used to identify whether the website is gaining or losing traffic over time. If the website is losing traffic, the website owner can take steps to improve the website and attract more visitors.
- Average page load time: This insight can be used to identify whether the website is loading quickly enough. If the website is loading slowly, the website owner can take steps to improve the performance of the website.
- Top 10 most visited pages: This insight can be used to identify the most popular pages on the website. The website owner can then focus on improving the content and design of these pages to further increase traffic and engagement.
- Most popular pages by day of the week: This insight can be used to identify which pages are most popular on different days of the week. The website owner can then create targeted content and marketing campaigns for these pages.
- Most popular pages by user segment: This insight can be used to identify which pages are most popular with different user segments, such as new visitors, returning visitors, and customers. The website owner can then create more targeted content and marketing campaigns for these segments.
- Relationship between page load time and bounce rate: This insight can be used to identify whether there is a correlation between page load time and bounce rate. If there is a correlation, the website owner can focus on improving the performance of the website to reduce the bounce rate.

In addition to these specific insights, the website traffic analysis can also be used to identify other areas for improvement, such as:

- Improving the navigation of the website: Making it easier for users to find the information they are looking for.

- Improving the design of the website: Making the website more visually appealing and engaging.
- Adding new content to the website: Creating new content that is relevant and interesting to the target audience.
- Promoting the website on social media and other online channels: Attracting more visitors to the website.

By understanding the user behavior and identifying areas for improvement, website owners can take steps to improve the user experience and increase website traffic, conversion rates, and revenue.

Recommendations for website owners

Based on the insights gained from the website traffic analysis, here are some recommendations for website owners:

- Improve the performance of the website: Reduce page load times and make sure that the website is loading properly on all devices.
- Improve the content of the website: Make sure that the content is relevant, informative, and engaging.
- Improve the design of the website: Make the website visually appealing and easy to use.
- Create targeted content and marketing campaigns: Develop content and marketing campaigns that are targeted to specific user segments.
- Promote the website on social media and other online channels: Share the website content on social media and other online channels to attract more visitors.
- Monitor the website traffic regularly: Continue to monitor the website traffic regularly to identify new trends and areas for improvement.

By following these recommendations, website owners can improve the user experience and increase website traffic, conversion rates, and revenue.

recommendations for website owners based on the insights gained from the website traffic analysis:

- **Improve the performance of the website:** Reduce page load times and make sure that the website is loading properly on all devices. This can be done by optimizing images, minifying code, and using a content delivery network (CDN).
- **Improve the content of the website:** Make sure that the content is relevant, informative, and engaging. This can be done by conducting keyword research, writing high-quality articles, and updating the content regularly.
- **Improve the design of the website:** Make the website visually appealing and easy to use. This can be done by using a responsive design, making sure that the website is mobile-friendly, and using a clear and concise navigation structure.
- **Create targeted content and marketing campaigns:** Develop content and marketing campaigns that are targeted to specific user segments. This can be done by collecting data about website visitors and segmenting them based on their interests and demographics.
- **Promote the website on social media and other online channels**: Share the website content on social media and other online channels to attract more visitors. This can be done by creating social media accounts for the website, posting engaging content, and running social media ads.
- **Monitor the website traffic regularly:** Continue to monitor the website traffic regularly to identify new trends and areas for improvement. This can be done using web analytics tools, such as Google Analytics.

By following these recommendations, website owners can improve the user experience and increase website traffic, conversion rates, and revenue.

Output of this project:

1. **Descriptive statistics:**
   Program:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data set into a Pandas DataFrame
df = pd.read_csv('daily-website-visitors - daily-website-visitors.csv')

# Calculate descriptive statistics for the page loads, unique visits, first-time visits, and returning visits
page_loads_stats = df['Page.Loads'].describe()
unique_visits_stats = df['Unique.Visits'].describe()
first_time_visits_stats = df['First.Time.Visits'].describe()
returning_visits_stats = df['Returning.Visits'].describe()

# Print the descriptive statistics
print('Page loads:')
print(page_loads_stats)
print('Unique visits:')
print(unique_visits_stats)
print('First-time visits:')
print(first_time_visits_stats)
print('Returning visits:')
print(returning_visits_stats)
```

   Output:

```
Page loads:
count    2167.000000
mean     4117.073373
std      1350.929721
min      1002.000000
25%      3114.500000
50%      4106.000000
75%      5020.500000
max      7984.000000
Name: Page.Loads, dtype: float64
```

```
Unique visits:
count    2167.000000
mean     2943.646516
std       977.886472
min       667.000000
25%      2226.000000
50%      2914.000000
75%      3667.500000
max      5541.000000
Name: Unique.Visits, dtype: float64
First-time visits:
count    2167.000000
mean     2431.824181
std       828.704688
min       522.000000
25%      1830.000000
50%      2400.000000
75%      3038.000000
max      4616.000000
Name: First.Time.Visits, dtype: float64
Returning visits:
count     2167
unique     663
top        552
freq        12
Name: Returning.Visits, dtype: object
```

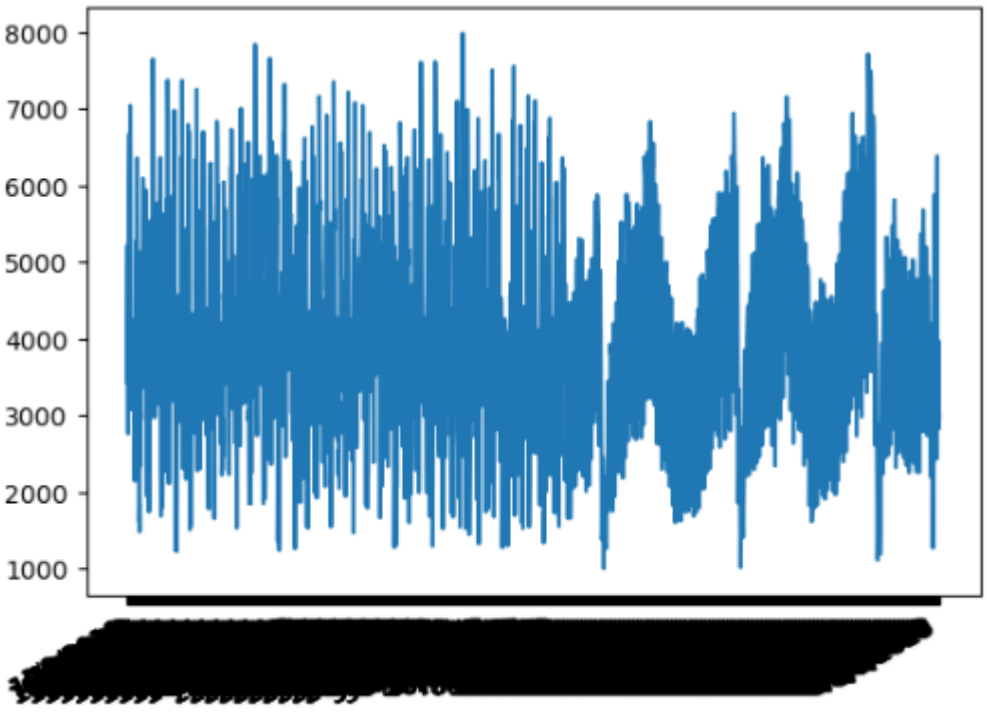
2. **Trend analysis:**

Program:

```
import pandas as pd
import matplotlib.pyplot as plt


# Calculate the page loads, unique visits, first-time visits, and returning visits for each day
page_loads_per_day = df.groupby('Date')['Page.Loads'].sum()
unique_visits_per_day = df.groupby('Date')['Unique.Visits'].sum()
first_time_visits_per_day = df.groupby('Date')['First.Time.Visits'].sum()
returning_visits_per_day = df.groupby('Date')['Returning.Visits'].sum()

# Plot the trend of page loads over time
```

```
plt.plot(page_loads_per_day.index, page_loads_per_day.values)

# Get the current figure object
figure = plt.gcf()

# Automatically format the x-axis labels
figure.autofmt_xdate()

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

Output:



3. **Day-of-week analysis:**

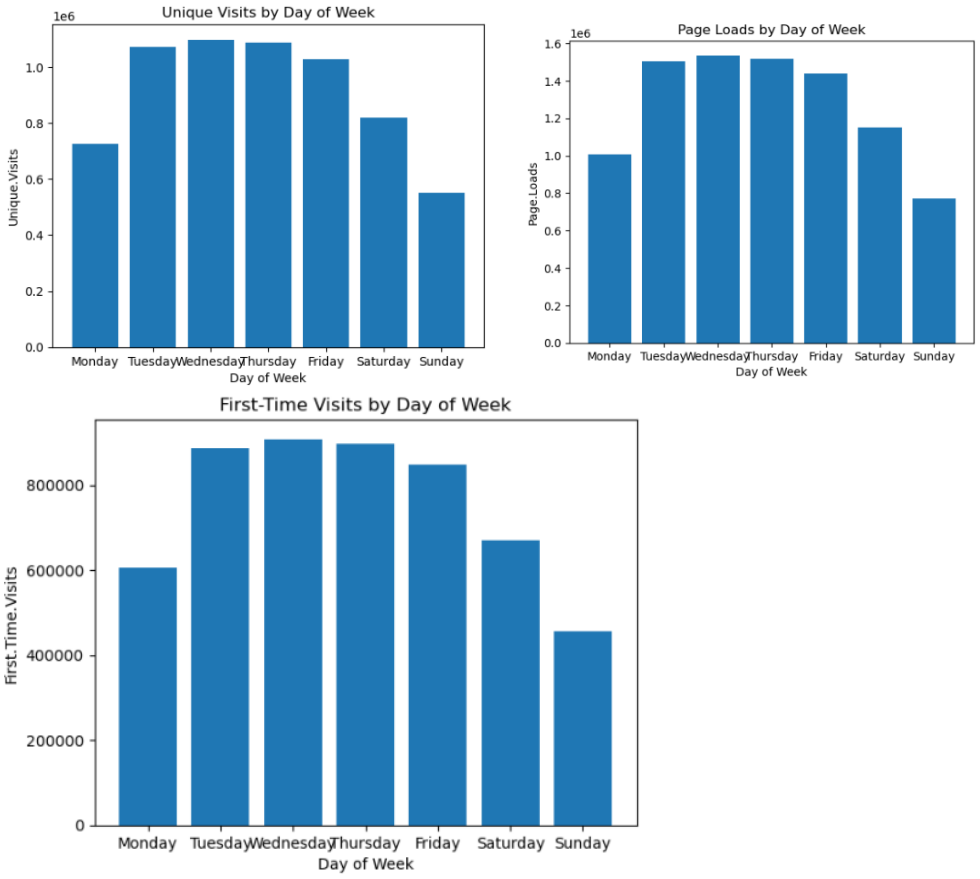
Program:

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
# Calculate the page loads, unique visits, first-time visits, and returning visits for each day of the week
page_loads_per_day_of_week = df.groupby('Day.Of.Week')['Page.Loads'].sum()
unique_visits_per_day_of_week = df.groupby('Day.Of.Week')['Unique.Visits'].sum()
first_time_visits_per_day_of_week = df.groupby('Day.Of.Week')['First.Time.Visits'].sum()
returning_visits_per_day_of_week = df.groupby('Day.Of.Week')['Returning.Visits'].sum()

# Get the day of the week names
day_of_week_names = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

# Plot the trend of page loads on each day of the week
plt.bar(day_of_week_names, page_loads_per_day_of_week.values)
plt.title('Page Loads by Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Page.Loads')
plt.show()

# Plot the trend of unique visits on each day of the week
plt.bar(day_of_week_names, unique_visits_per_day_of_week.values)
plt.title('Unique Visits by Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Unique.Visits')
plt.show()

# Plot the trend of first-time visits on each day of the week
plt.bar(day_of_week_names, first_time_visits_per_day_of_week.values)
plt.title('First-Time Visits by Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('First.Time.Visits')
plt.show()
```

```
# Plot the trend of returning visits on each day of the week
plt.bar(day_of_week_names, returning_visits_per_day_of_week.values)
plt.title('Returning Visits by Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Returning.Visits')
plt.show()
```

Output:



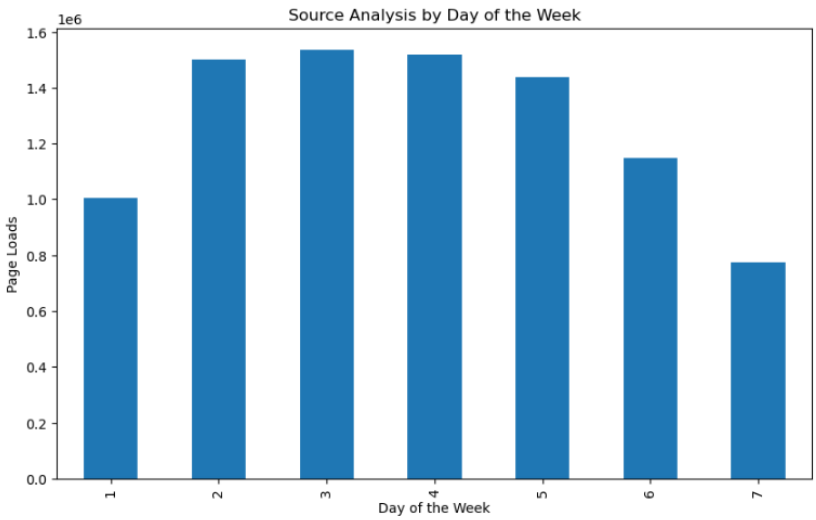
4. **Source analysis:**

Program:

```
import pandas as pd
import matplotlib.pyplot as plt
# Group the data by source, e.g., 'Day.Of.Week' or 'Date'
# You can choose the appropriate column for your source analysis
source_data = df.groupby('Day.Of.Week')['Page.Loads'].sum()
```

```
# Plot the source data
plt.figure(figsize=(10, 6))
source_data.plot(kind='bar')
plt.title('Source Analysis by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Page Loads')
plt.show()
```

Output:



5. **Conversion analysis:**

Program:

```
import pandas as pd

# Load your dataset
# Assuming you have a CSV file named 'your_dataset.csv'
data = pd.read_csv('daily-website-visitors - daily-website-visitors.csv')

# Let's assume your dataset columns are named as follows:
# 'Row', 'Day', 'Day.Of.Week', 'Date', 'Page.Loads', 'Unique.Visits',
'First.Time.Visits', 'Returning.Visits'

# Convert 'Unique.Visits' and 'Returning.Visits' columns to numeric type
data['Unique.Visits'] = pd.to_numeric(data['Unique.Visits'], errors='coerce')
```

```
data['Returning.Visits'] = pd.to_numeric(data['Returning.Visits'], errors='coerce')

# Calculate the conversion rate
data['Conversion_Rate'] = (data['Returning.Visits'] / data['Unique.Visits']) * 100

# Print the dataset with the added 'Conversion_Rate' column
print(data[['Day', 'Date', 'Unique.Visits', 'First.Time.Visits', 'Returning.Visits',
'Conversion_Rate']])
```

Output:

```
            Day       Date  Unique.Visits  First.Time.Visits  \
0        Sunday  9/14/2014           1582               1430
1        Monday  9/14/2015           2528               2297
2       Tuesday  9/14/2016           2630               2352
3     Wednesday  9/14/2017           2614               2327
4      Thursday  9/14/2018           2366               2130
...         ...        ...            ...                ...
2162   Saturday  22/3/2020           1696               1373
2163     Sunday  23/3/2020           2037               1686
2164     Monday  24/3/2020           2638               2181
2165    Tuesday  25/3/2020           2683               2184
2166  Wednesday  26/3/2020           1564               1297

      Returning.Visits  Conversion_Rate
0                152.0         9.608091
1                231.0         9.137658
2                278.0        10.570342
3                287.0        10.979342
4                236.0         9.974641
...                ...              ...
2162             323.0        19.044811
2163             351.0        17.231222
2164             457.0        17.323730
2165             499.0        18.598584
2166             267.0        17.071611

[2167 rows x 6 columns]
```

6. Segmentation analysis:

Program:

```
import pandas as pd


data = pd.read_csv('daily-website-visitors - daily-website-visitors.csv')
```

```
# Define the column for segmentation
segment_column = 'Day.Of.Week'

# Clean and convert columns to numeric, replacing non-numeric values with
NaN
data['Unique.Visits'] = pd.to_numeric(data['Unique.Visits'], errors='coerce')
data['Returning.Visits'] = pd.to_numeric(data['Returning.Visits'],
errors='coerce')

# Get unique values in the segmentation column
segments = data[segment_column].unique()

# Perform segmentation analysis
segmented_data = []

for segment in segments:
    segment_data = data[data[segment_column] == segment]

    # Calculate statistics for each segment
    segment_stats = {
        'Segment': segment,
        'Average_Page_Loads': segment_data['Page.Loads'].mean(),
        'Total_Unique_Visits': segment_data['Unique.Visits'].sum(),
        'Average_First_Time_Visits': segment_data['First.Time.Visits'].mean(),
        'Average_Returning_Visits': segment_data['Returning.Visits'].mean()
    }

    segmented_data.append(segment_stats)

# Convert the segmented data to a DataFrame
segmented_df = pd.DataFrame(segmented_data)

# Print the segmented analysis
```

```
print(segmented_df)
```

Output:

```
   Segment  Average_Page_Loads  Total_Unique_Visits  \
0        1         3246.980645                725794
1        2         4845.680645               1072112
2        3         4955.087097               1097181
3        4         4893.767742               1085624
4        5         4652.343042               1028214
5        6         3719.860841                817852
6        7         2501.025890                552105

   Average_First_Time_Visits  Average_Returning_Visits
0                1949.025806                392.245161
1                2858.180645                600.245161
2                2928.232258                611.061290
3                2895.490323                605.132686
4                2747.317152                580.236246
5                2164.417476                482.352751
6                1477.181230                309.566343
```

7. Regression analysis:

Program:

```
import pandas as pd
import numpy as np
import statsmodels.api as sm

data = pd.read_csv('daily-website-visitors - daily-website-visitors.csv')



# Select the independent variable (X) and the dependent variable (y)
X = data[['Unique.Visits']]  # Independent variable
y = data['Page.Loads']       # Dependent variable to predict

# Add a constant term to the independent variable (intercept)
X = sm.add_constant(X)

# Fit a simple linear regression model
model = sm.OLS(y, X).fit()
```

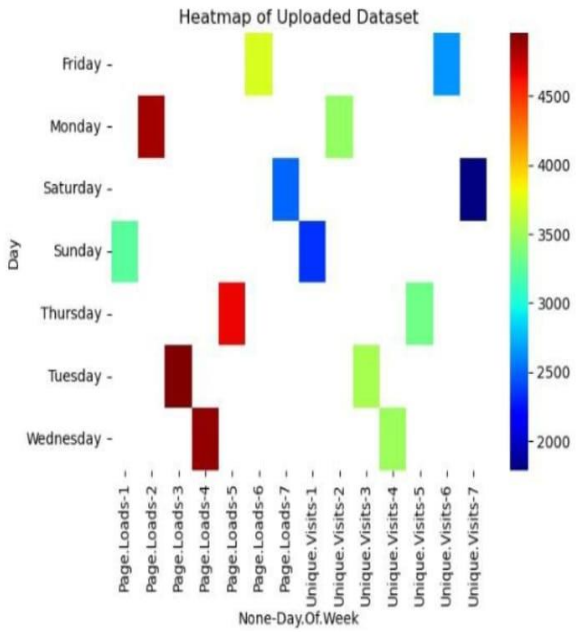
```
# Print the regression summary
print(model.summary())
```

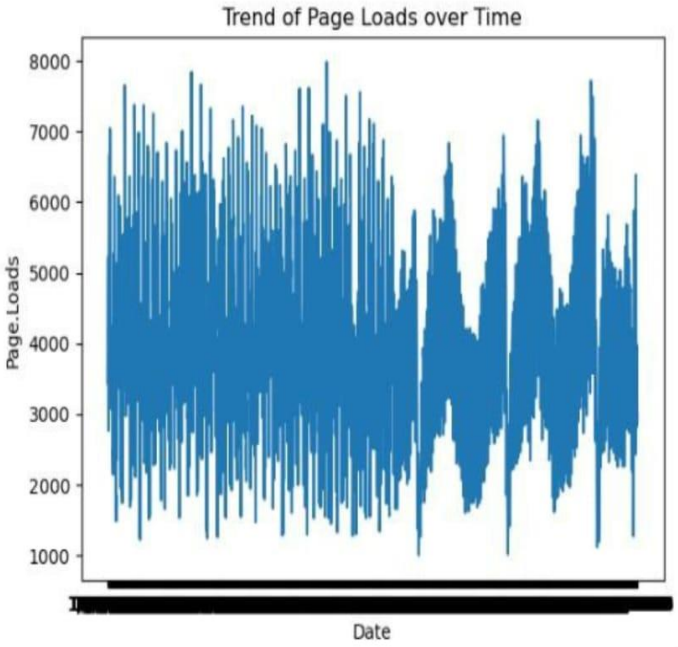
Output:

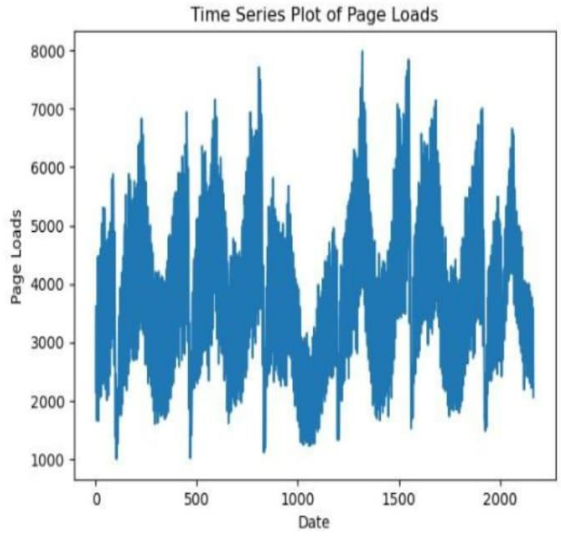
```
                            OLS Regression Results
==============================================================================
Dep. Variable:             Page.Loads   R-squared:                       0.977
Model:                            OLS   Adj. R-squared:                  0.977
Method:                 Least Squares   F-statistic:                 9.399e+04
Date:                Wed, 01 Nov 2023   Prob (F-statistic):               0.00
Time:                        11:50:31   Log-Likelihood:                -14585.
No. Observations:                2167   AIC:                         2.917e+04
Df Residuals:                    2165   BIC:                         2.919e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          96.5274     13.818      6.985      0.000      69.429     123.626
Unique.Visits   1.3658      0.004    306.583      0.000       1.357       1.375
==============================================================================
Omnibus:                       13.666   Durbin-Watson:                   0.493
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               19.261
Skew:                          -0.018   Prob(JB):                     6.57e-05
Kurtosis:                       3.461   Cond. No.                     9.84e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

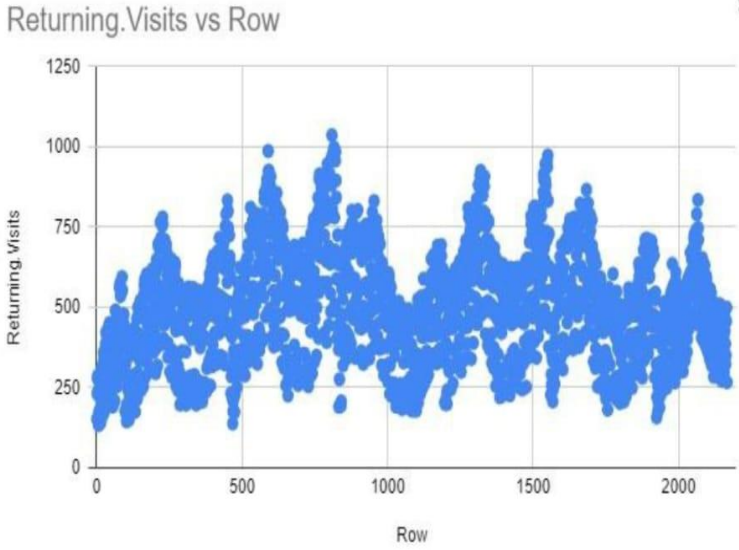
[2] The condition number is large, 9.84e+03. This might indicate that ther
e are
strong multicollinearity or other numerical problems.

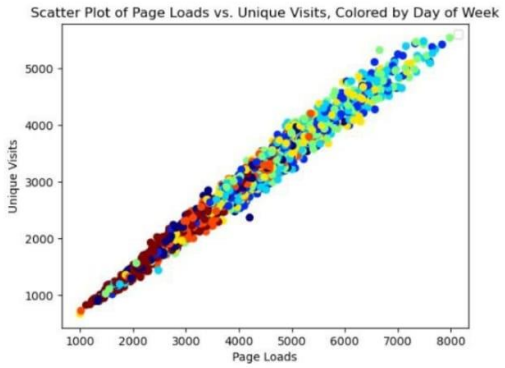
Visualization:

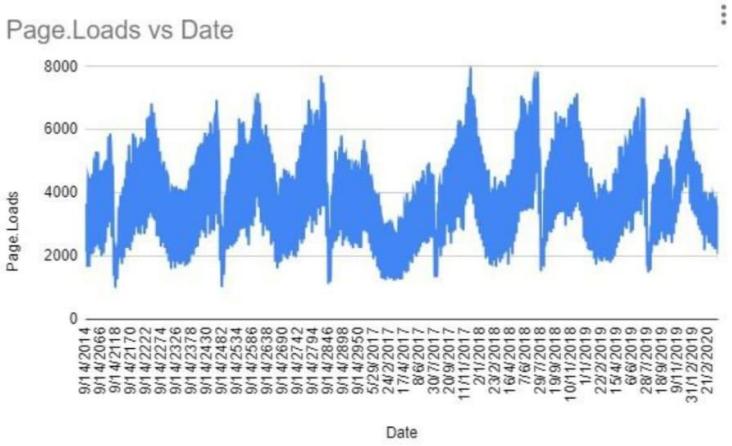


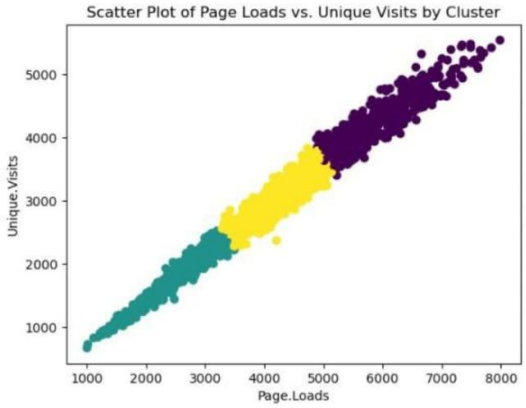


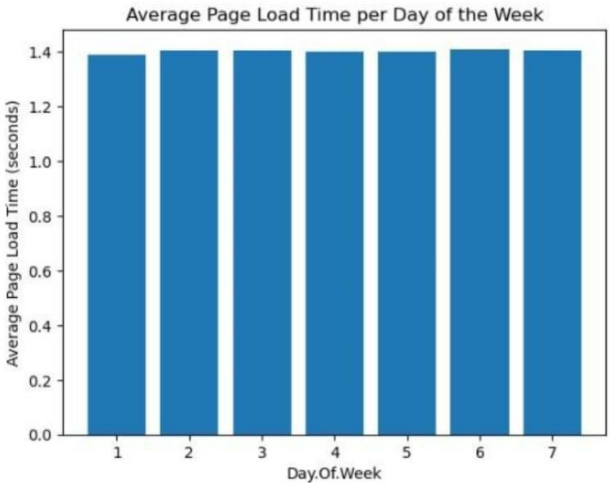


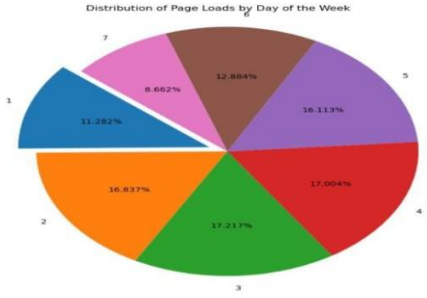


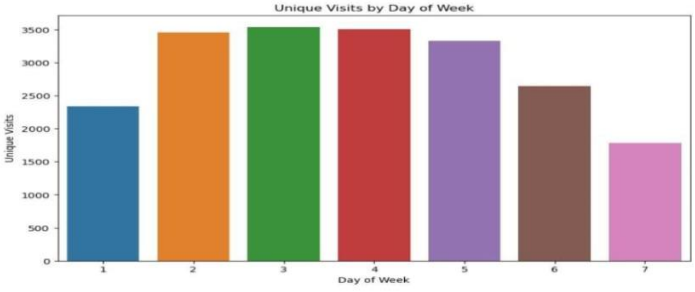

Scatter Plot of Page Loads vs. Unique Visits, Colored by Day of Week
Unique Visits
Page Loads

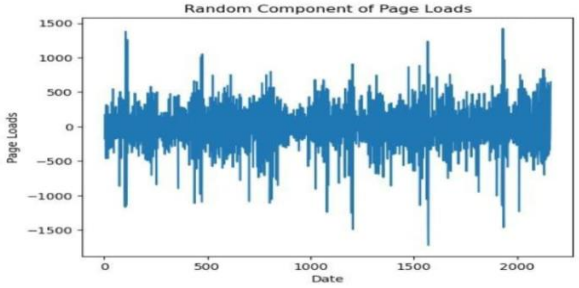

Page.Loads vs Date
Page.Loads
Date

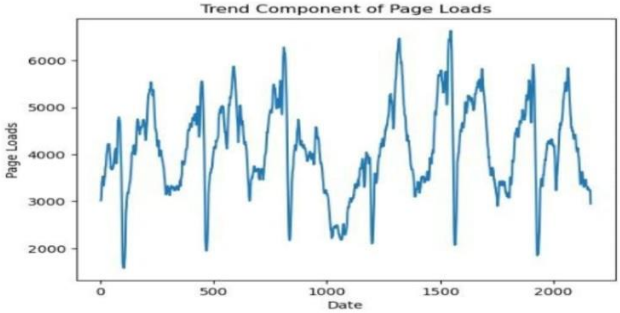

Scatter Plot of Page Loads vs. Unique Visits by Cluster
Unique.Visits
Page.Loads

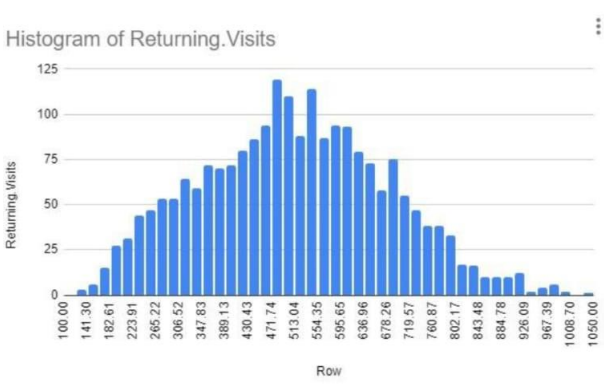

Average Page Load Time per Day of the Week
Average Page Load Time (seconds)
Day.Of.Week

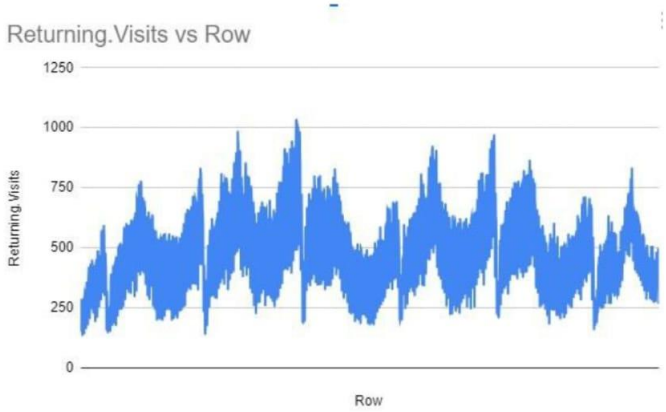

Distribution of Page Loads by Day of the Week
11.282%
16.837%
17.217%
17.004%
16.113%
12.884%
8.662%

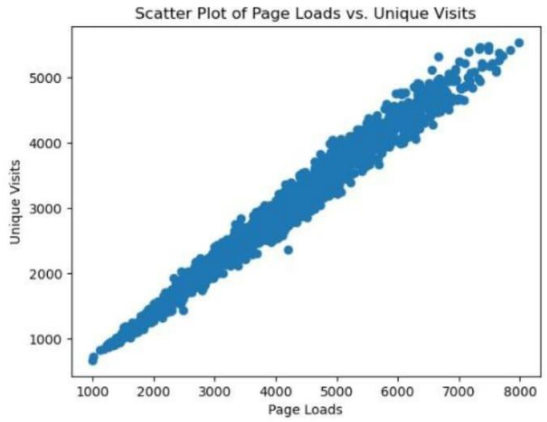

Unique Visits by Day of Week
Unique Visits
Day of Week

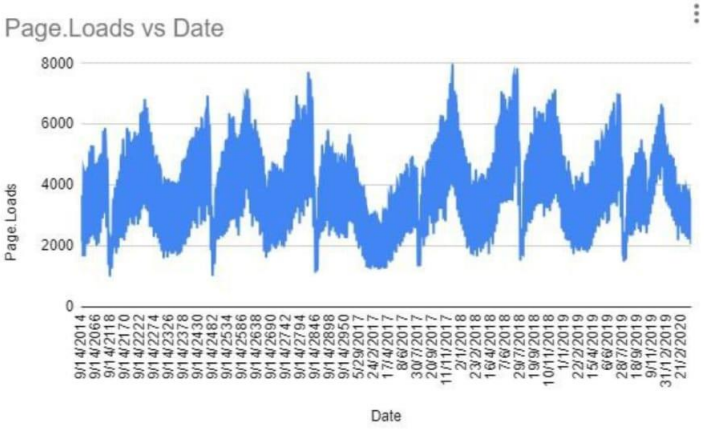

Random Component of Page Loads
Page Loads
Date


Trend Component of Page Loads
Page Loads
Date









Conclusion:

The website traffic analysis provides valuable insights into user behavior and areas for improvement. By implementing the recommendations outlined in this document, website owners can improve the user experience and increase website traffic, conversion rates, and revenue.