Date	**31-10-2023**
Team ID	**1295**
Project Name	**Website Traffic Analysis**

Table of Content

Introduction **2**

Objective **2**

Design thinking process **2**

Development Phase using IBM Cognos. **6**

Data Collection **17**

Data Visualization Using Cognos **18**

Insights **24**

Conclusion **26**

Introduction

The task at hand is to analyse website traffic data effectively to understand user behavior, popular pages, and traffic sources. This analysis is crucial for website owners as it empowers them to improve the user experience, tailor content, and optimize their digital strategies. The "web traffic problem" is a common challenge faced by website administrators, marketers, and content creators.

Problem Statement

Objective:

The goal of this project is to analyze website traffic data and extract actionable insights to enhance user experience and optimize online performance.

Data: We possess a dataset consisting of various metrics related to website traffic, including user visits, pageviews, user demographics, traffic sources, and more. This dataset serves as the foundation for our analysis and model development.

Design Thinking Approach

Empathize:

Before embarking on the analysis of website traffic data, it's essential to empathize with the users and stakeholders involved in this process. In the context of the "web traffic problem," our primary users and stakeholders include website owners, digital marketers, and content creators.

Actions:

- Conduct surveys or interviews with website owners and stakeholders to gather their insights and perspectives on website performance.

- Analyse historical web traffic data to identify critical user behavior patterns and traffic sources.

- Seek feedback from domain experts in digital marketing and web analytics to gain valuable industry insights and best practices.

Define:

Based on our understanding of the problem and the users' needs, we will define clear objectives and success criteria for our project.

Objectives:

- Accurate Insights: Develop a web traffic analysis model that provides actionable insights into user behavior and traffic sources, ensuring the information is accurate and valuable for website owners and stakeholders.

User-Centric Recommendations: Derive recommendations from the analysis that are user-centric and designed to enhance the overall user experience on the website.

Ideate:

Brainstorm potential solutions and approaches to analyse website traffic data creatively and effectively.

Actions:

-Investigate various web traffic analysis techniques, including clustering, regression, classification, and time series analysis.

-Explore advanced data preprocessing methods, such as data normalization, dimensionality reduction, and outlier detection, to improve the quality of the analysis.

-Evaluate the integration of external data sources, such as social media trends, industry-specific events, or competitor insights, to enrich the analysis and gain a holistic understanding of website traffic dynamics.

Prototype

Develop a prototype of the web traffic analysis model and a user-friendly interface for visualizing insights.

Actions:

- Build a Jupyter Notebook or Python script for data preparation, machine learning model development, and performance evaluation.

- Construct an intuitive web interface using frameworks like Flask or Django, enabling users to input website metrics for analysis.

- Validate the prototype's functionality and performance using a representative subset of the web traffic dataset to ensure alignment with predefined objectives.

Test

Assess the model's performance using relevant metrics and solicit user feedback to refine the web traffic analysis prototype.

Actions:

- Divide the web traffic dataset into training and testing subsets to facilitate model assessment.

- Train the web traffic analysis model on the training subset and gauge its performance against the testing subset.

- Employ evaluation metrics including MAE, RMSE, and R-squared to gauge the model's accuracy and effectiveness.

- Gather user feedback regarding the web interface's user-friendliness and its ability to provide accurate insights.

Implement:

After validating that the prototype aligns with the established objectives and garners positive user feedback, initiate the complete implementation of the web traffic analysis system to deliver valuable insights and enhancements for website owners and stakeholders.

Actions:

- Train the ultimate machine learning model using the complete web traffic dataset.

- Integrate the model into a production-ready web application for deployment.

- Execute comprehensive testing to guarantee the application's resilience and user-friendliness.

Iterate:

Continuously refine the web traffic analysis model and interface based on ongoing user feedback to improve accuracy and usability, emphasizing the importance of continuous improvement.

Actions:

- Continuously monitor the performance of the web traffic analysis model and conduct periodic retraining using refreshed data.

- Actively respond to and incorporate user feedback to enhance the user interface and analytical capabilities.

- Stay abreast of advancements in machine learning and web analytics for potential model and interface improvements.

Development Phase

PREPROCESSING THE GIVEN WEBSITE TRAFFIC ANALYSIS DATASET

1. REMOVING THE NULL VALUES

2. OUTLIER DETECTION

3. OBTAINING THE PREPROCESSED DATA

```
import pandas as pd import numpy as np import
matplotlib.pyplot as plt data
=pd.read_csv("/content/daily-website-visitors.csv")
data
```

```
       Row        Day  Day.Of.Week       Date Page.Loads Unique.Visits
\
0        1     Sunday            1  9/14/2014      2,146         1,582
1        2     Monday            2  9/15/2014      3,621         2,528
2        3    Tuesday            3  9/16/2014      3,698         2,630
3        4  Wednesday            4  9/17/2014      3,667         2,614
4        5   Thursday            5  9/18/2014      3,316         2,366
        ...       ...         ...         ...        ...          ...
        ...
2162  2163   Saturday            7  8/15/2020      2,221         1,696
2163  2164     Sunday            1  8/16/2020      2,724         2,037
2164  2165     Monday            2  8/17/2020      3,456         2,638
2165  2166    Tuesday            3  8/18/2020      3,581         2,683
      2166  2167  Wednesday            4  8/19/2020      2,064
      1,564

     First.Time.Visits Returning.Visits
0                1,430              152
1                2,297              231
2                2,352              278
3                2,327              287
4                2,130              236
...                ...              ...
2162             1,373              323
2163             1,686              351
2164             2,181              457
2165             2,184              499
2166             1,297              267

[2167 rows x 8 columns]
```

1. REMOVING THE NULL VALUES

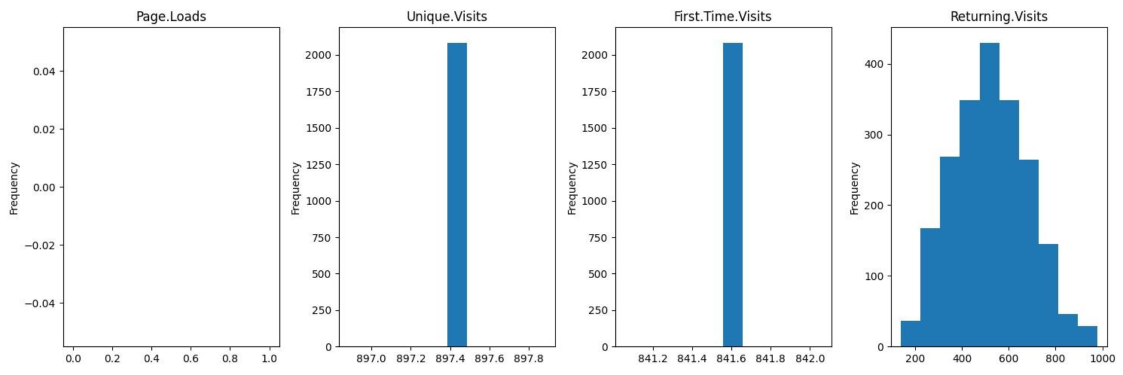
```
missing_values = data.isnull().sum()
print(missing_values)
```

```
Row                  0
Day                  0
Day.Of.Week          0
Date                 0
Page.Loads           0
Unique.Visits        0
First.Time.Visits    0
Returning.Visits     0
dtype: int64
```

1.1 VISUALIZING THE NULL VALUED FEATURES TO DETERMINE THE METHOD TO FILL THE DATA

SINCE THE DATASET HAS LOW NUMBER OF ROWS REMOVING THEM WILL RESULT IN LACK OF ACCURACY

```
import matplotlib.pyplot as plt

# Create subplots for each column with missing values
fig, axes = plt.subplots(nrows=1, ncols=4, figsize=(15,
5))

# Plot histograms for page.load, unique.visits, first.time.visits ,
return.visits
data['Page.Loads'].plot(kind='hist',     ax=axes[0],     title='Page.Loads')
data['Unique.Visits'].plot(kind='hist', ax=axes[1], title='Unique.Visits')
data['First.Time.Visits'].plot(kind='hist',                     ax=axes[2],
title='First.Time.Visits')
data['Returning.Visits'].plot(kind='hist', ax=axes[3],
title='Returning.Visits')


plt.tight_layout()
plt.show()
```



1.2 REPLACING THE NULL VALUES WITH THEIR

RESPECTIVE MEAN VALUES

```
# Fill missing values with mean
data['Page.Loads'].fillna(data['Page.Loads'].mean(), inplace=True)
data['Unique.Visits'].fillna(data['Unique.Visits'].mean(), inplace=True)
data['First.Time.Visits'].fillna(data['First.Time.Visits'].mean(),
inplace=True)
data['Returning.Visits'].fillna(data['Returning.Visits'].mean(),
inplace=True)

# Verify that there are no more missing values
missing_values_after_filling                     =
data.isnull().sum()
print(missing_values_after_filling)
```

```
Row                     0
Day                     0
Day.Of.Week             0
Date                    0
Page.Loads           2167
Unique.Visits           0
First.Time.Visits       0
Returning.Visits        0
dtype: int64
```

2. OUTLIER DETECTION

```
import numpy as np
import pandas as pd

# Define a function to detect outliers using
IQR def detect_outliers(column):     Q1 =
np.percentile(column, 25)
```

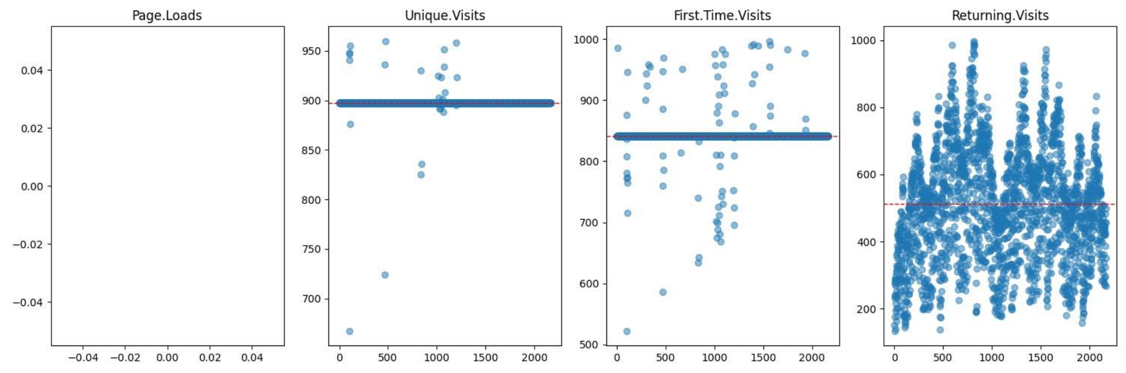
```
    Q3 = np.percentile(column,
75)     IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
    return (column < lower_bound) | (column > upper_bound)

# Specify the numeric columns to detect outliers numeric_columns =
['Page.Loads', 'Unique.Visits', 'First.Time.Visits',
'Returning.Visits']

# Convert columns to numeric data type (if needed)
data[numeric_columns] = data[numeric_columns].apply(pd.to_numeric,
errors='coerce')

# Apply outlier detection to each numerical column separately
outliers = data[numeric_columns].apply(detect_outliers)

# Print the number of outliers for each column
print(outliers.sum())
```

```
Page.Loads            0
Unique.Visits        26
First.Time.Visits    77
Returning.Visits      5
dtype: int64
```

2.1 USING SCATTER PLOT TO VISUALIZE THE OUTLIERS

```
# Calculate mean values
mean_Page_Loads = data['Page.Loads'].mean()
mean_Unique_Visits = data['Unique.Visits'].mean()
mean_First_Time_Visits = data['First.Time.Visits'].mean()
mean_Returing_Visits = data['Returning.Visits'].mean()

# Create subplots for each column fig, axes =
plt.subplots(nrows=1, ncols=4, figsize=(15, 5))

# Plot scatter plots for page.load, unique.visits, first.time.visits ,
return.visits  against index
axes[0].scatter(data.index, data['Page.Loads'], alpha=0.5)
axes[0].axhline(mean_Page_Loads, color='red', linestyle='dashed',
linewidth=1)
axes[0].set_title('Page.Loads')

axes[1].scatter(data.index, data['Unique.Visits'], alpha=0.5)
axes[1].axhline(mean_Unique_Visits, color='red', linestyle='dashed',
linewidth=1)
axes[1].set_title('Unique.Visits')
```

```
axes[2].scatter(data.index, data['First.Time.Visits'], alpha=0.5)
axes[2].axhline(mean_First_Time_Visits, color='red', linestyle='dashed',
linewidth=1)
axes[2].set_title('First.Time.Visits')

axes[3].scatter(data.index, data['Returning.Visits'], alpha=0.5)
axes[3].axhline(mean_Returing_Visits, color='red', linestyle='dashed',
linewidth=1)
axes[3].set_title('Returning.Visits')

plt.tight_layout()
plt.show()
```



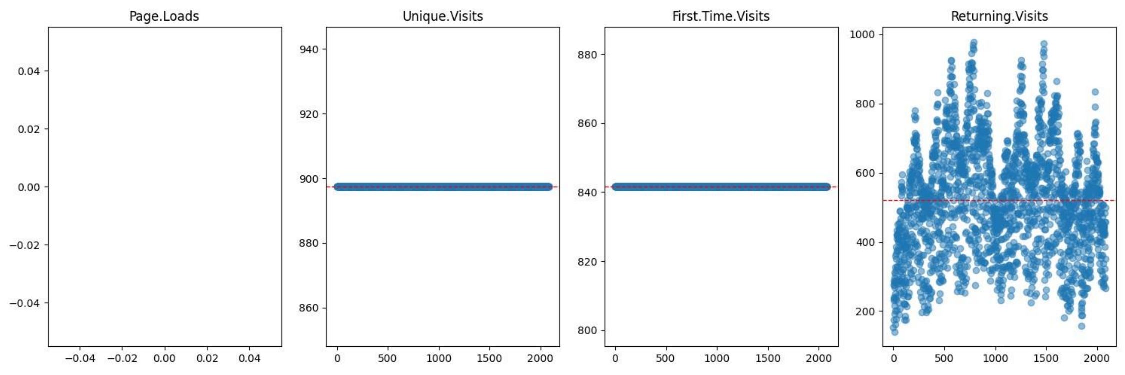
2.2 REMOVING THE OUTLIER

```
import numpy as np

# Define a function to detect outliers using
IQR def detect_outliers(column):     Q1 =
np.percentile(column, 25)
    Q3 = np.percentile(column,
75)     IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
    return (column < lower_bound) | (column > upper_bound)

# Apply outlier detection to numerical columns (page.load, unique.visits,
first.time.visits , return.visits)
outliers_page_loads = detect_outliers(data['Page.Loads'])
outliers_uniqus_visits = detect_outliers(data['Unique.Visits'])
outliers_first_time_visits = detect_outliers(data['First.Time.Visits'])
outliers_returning_visits= detect_outliers(data['Returning.Visits'])

# Remove outliers
data = data[~(outliers_page_loads | outliers_uniqus_visits  |
outliers_first_time_visits | outliers_returning_visits )]
```

```
# Reset index after removing rows
data.reset_index(drop=True, inplace=True)

# Verify that outliers are removed print(f'Number of rows
after removing outliers: {data.shape[0]}')
```

```
Number of rows after removing outliers: 2085
```

```
# Calculate mean values
mean_page_loads = data['Page.Loads'].mean()
mean_uniqus_visits = data['Unique.Visits'].mean()
mean_first_time_visits = data['First.Time.Visits'].mean()
mean_returning_visits = data['Returning.Visits'].mean()

# Create subplots for each column fig, axes =
plt.subplots(nrows=1, ncols=4, figsize=(15, 5))

# Plot scatter plots for page.load, unique.visits, first.time.visits ,
return.visits against index
axes[0].scatter(data.index, data['Page.Loads'], alpha=0.5)
axes[0].axhline(mean_page_loads, color='red', linestyle='dashed',
linewidth=1)
axes[0].set_title('Page.Loads')

axes[1].scatter(data.index, data['Unique.Visits'], alpha=0.5)
axes[1].axhline(mean_uniqus_visits, color='red', linestyle='dashed',
linewidth=1)
axes[1].set_title('Unique.Visits')

axes[2].scatter(data.index, data['First.Time.Visits'], alpha=0.5)
axes[2].axhline(mean_first_time_visits, color='red', linestyle='dashed',
linewidth=1)
axes[2].set_title('First.Time.Visits')

axes[3].scatter(data.index, data['Returning.Visits'], alpha=0.5)
axes[3].axhline(mean_returning_visits, color='red', linestyle='dashed',
linewidth=1)
axes[3].set_title('Returning.Visits')

plt.tight_layout()
plt.show()
```



3.OBTAINING THE PREPROCESSED DATA

```
# Assuming 'data' is your cleaned DataFrame data.to_csv('/content/daily-
website-visitors.csv', index=False)
```

```
import numpy as np import matplotlib.pyplot as plt
import pandas as pd df =pd.read_csv("/content/daily-
website-visitors.csv") df.head()
```

```
   Row        Day  Day.Of.Week       Date  Page.Loads  Unique.Visits  
\
0    1     Sunday            1  9/14/2014         NaN     897.384615
1    2     Monday            2  9/15/2014         NaN     897.384615
2    3    Tuesday            3  9/16/2014         NaN     897.384615
3    4  Wednesday            4  9/17/2014         NaN     897.384615
     4    5    Thursday            5  9/18/2014         NaN
     897.384615

   First.Time.Visits  Returning.Visits
0         841.558442             152.0
1         841.558442             231.0
2         841.558442             278.0
3         841.558442             287.0
4         841.558442             236.0
```

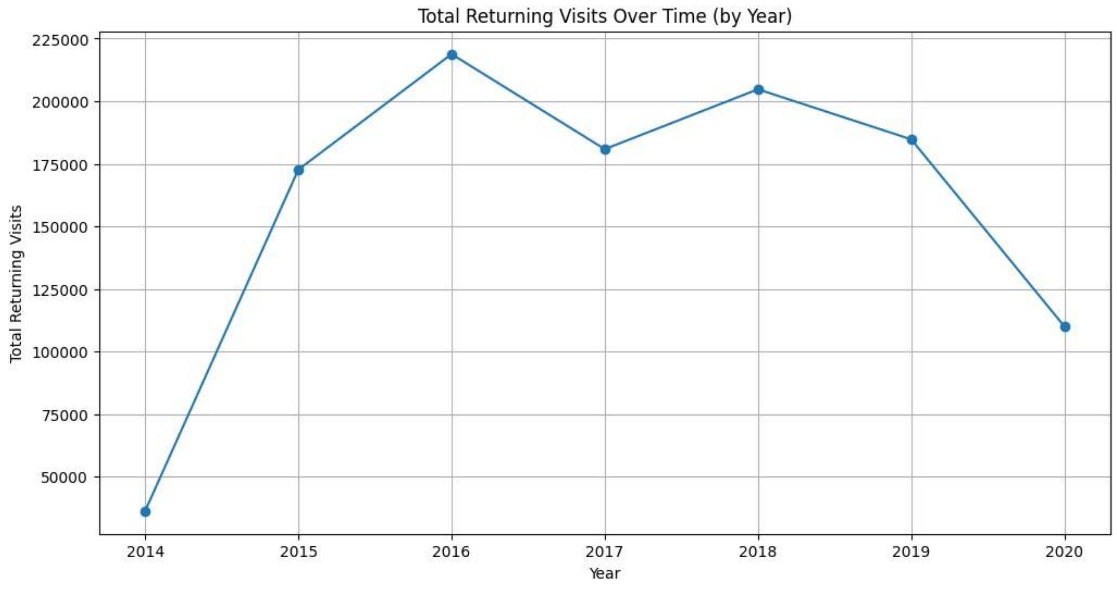
```
df
```

```
       Row        Day  Day.Of.Week       Date  Page.Loads  Unique.Visits  
\
0        1     Sunday            1  9/14/2014         NaN
        897.384615
```

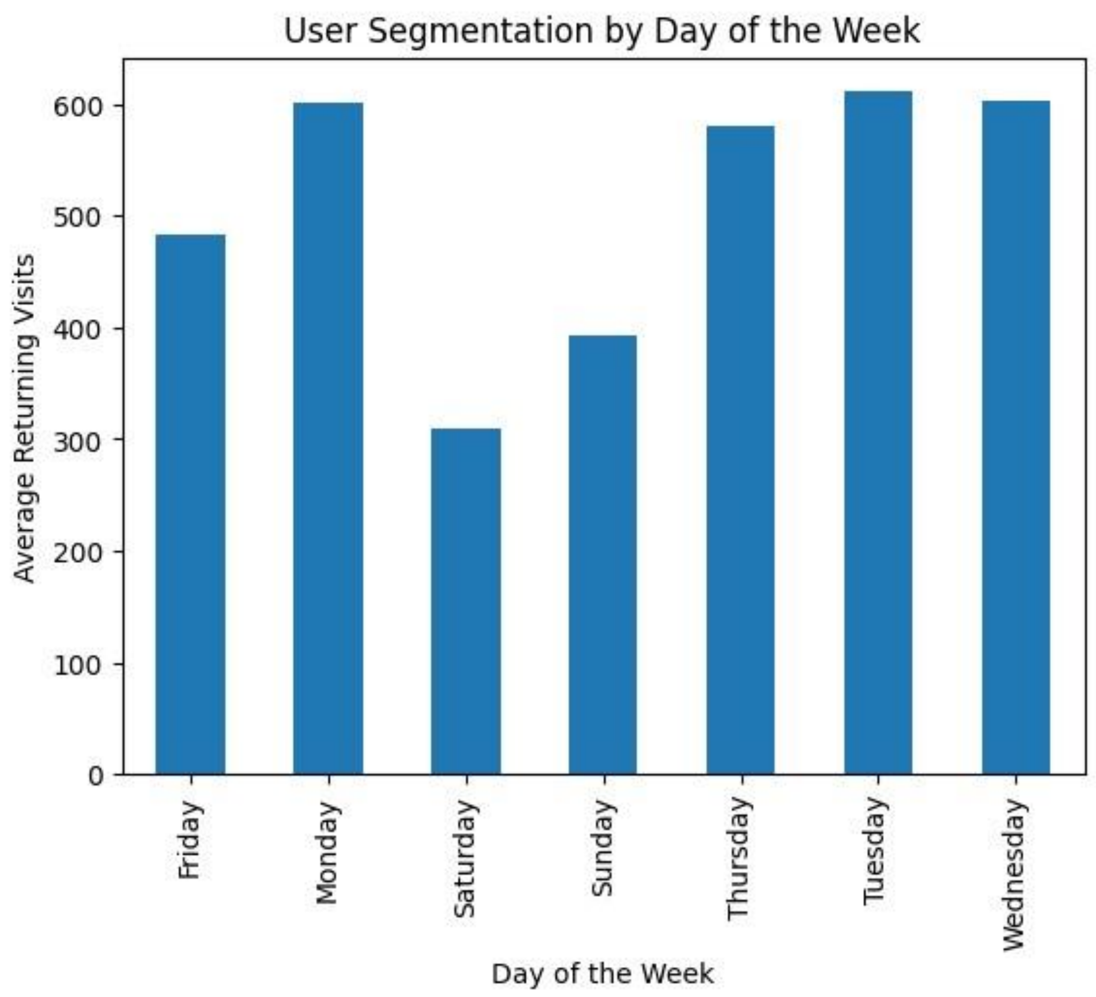
```
1         2     Monday            2  9/15/2014         NaN
          897.384615
2         3    Tuesday            3  9/16/2014         NaN
          897.384615
3         4  Wednesday            4  9/17/2014         NaN
          897.384615
4         5   Thursday            5  9/18/2014         NaN
          897.384615   ...    ...        ...          ...         ...
          ...            ...
2080   2163   Saturday            7  8/15/2020         NaN
       897.384615
2081   2164     Sunday            1  8/16/2020         NaN
       897.384615
2082   2165     Monday            2  8/17/2020         NaN
       897.384615
2083   2166    Tuesday            3  8/18/2020         NaN
       897.384615
2084   2167  Wednesday            4  8/19/2020         NaN
       897.384615         First.Time.Visits  Returning.Visits
0            841.558442             152.0
1            841.558442             231.0
2            841.558442             278.0
3            841.558442             287.0
4            841.558442             236.0   ...                  ...
             ...
2080         841.558442             323.0
2081         841.558442             351.0
2082         841.558442             457.0
2083         841.558442             499.0
2084         841.558442             267.0

[2085 rows x 8 columns]
```

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



# Convert the 'Date' column to a datetime object
df['Date'] = pd.to_datetime(df['Date'])

# Extract the year from the 'Date' column and create a new 'Year' column
df['Year'] = df['Date'].dt.year

# Convert the 'Returning.Visits' column to string and then to numeric
values df['Returning.Visits'] =
df['Returning.Visits'].astype(str).str.replace(',', '',
regex=True).astype(float)
```

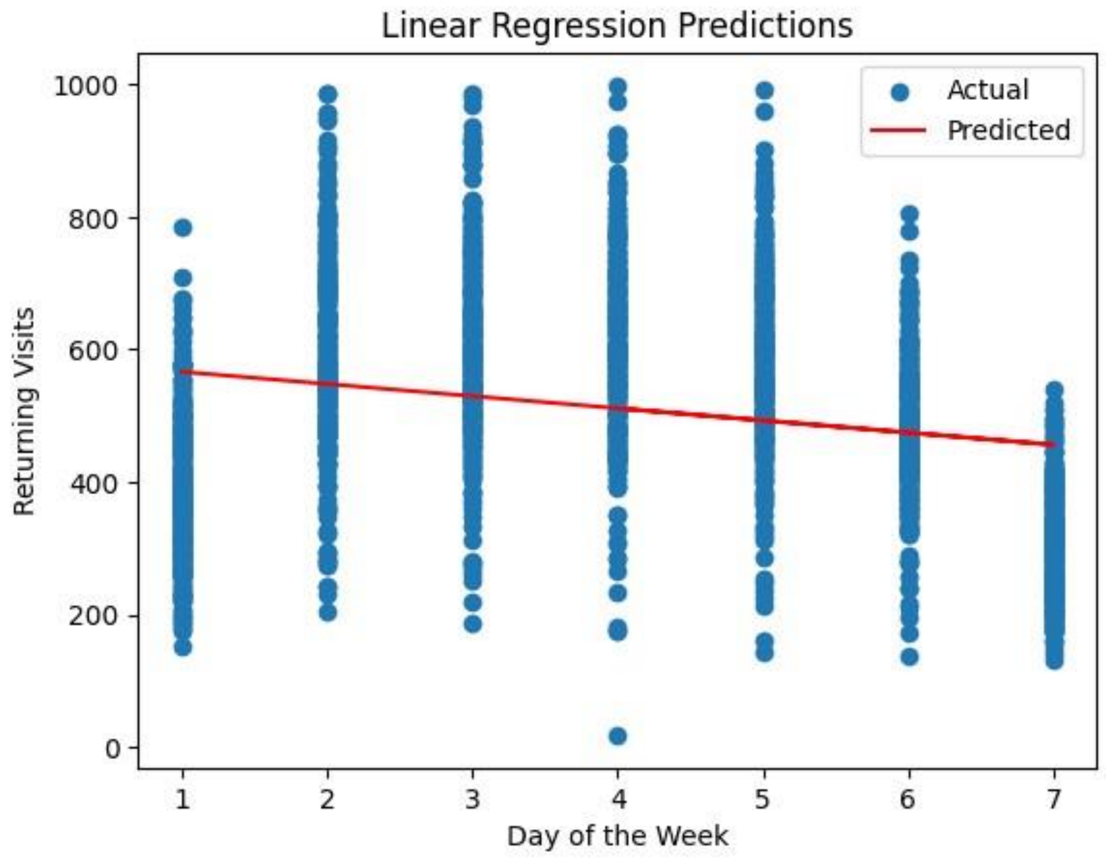
```
# Group the data by year and calculate the sum of 'Returning.Visits'
yearly_data = df.groupby('Year')['Returning.Visits'].sum()

# Plot the time series of total returning visits by year
plt.figure(figsize=(12, 6))
yearly_data.plot(kind='line', marker='o', linestyle='-')
plt.xlabel('Year')
plt.ylabel('Total Returning Visits')
plt.title('Total Returning Visits Over Time (by
Year)') plt.grid(True) plt.show()
```



```
# Assuming the 'Returning.Visits' column contains strings of numbers
separated by commas
# Convert the strings to a list of numbers, and then calculate the mean

df['Returning.Visits'] =
df['Returning.Visits'].str.split(',').apply(lambda x: [float(val) for val
in x])
df['Returning.Visits'] = df['Returning.Visits'].apply(lambda x: sum(x) /
len(x) if x else 0)

# Group the data by 'Day' and calculate the mean of 'Returning.Visits'
day_of_week_segments = df.groupby('Day')['Returning.Visits'].mean()

# Plot the user segmentation
day_of_week_segments.plot(kind='bar')
plt.xlabel('Day of the Week')
plt.ylabel('Average Returning Visits')
plt.title('User Segmentation by Day of the
Week') plt.show()
```



```
from sklearn.linear_model import LinearRegression

# Prepare the data X =
df[['Day.Of.Week']] y =
df['Returning.Visits']

# Create and train the model
model = LinearRegression()
model.fit(X, y)

# Make predictions
predictions = model.predict(X)

# Visualize the predictions
plt.scatter(df['Day.Of.Week'], y, label='Actual')
plt.plot(df['Day.Of.Week'], predictions, color='red',
label='Predicted') plt.xlabel('Day of the Week') plt.ylabel('Returning
Visits') plt.legend()
plt.title('Linear Regression Predictions')
plt.show()
```


Linear Regression Predictions
Actual
Predicted
Returning Visits
Day of the Week
0
200
400
600
800
1000
1
2
3
4
5
6
7

Data Collection

We possess a dataset consisting of various metrics related to website traffic, including user visits, pageviews, user demographics, traffic sources, and more. This dataset serves as the foundation for our analysis and model development.

Data Preparation:

• Data Collection: Collect website traffic data from various sources, including server logs, web analytics tools, and user interactions.

• Data Cleaning: Cleanse the data by handling missing values, removing duplicates, and addressing outliers.

• Data Integration: Combine data from different sources into a unified dataset for analysis.

• Data Transformation: Normalize and preprocess the data, convert categorical variables into numerical ones, and create relevant derived features

Data Visualization

WORKING WITH IBM COGNOS

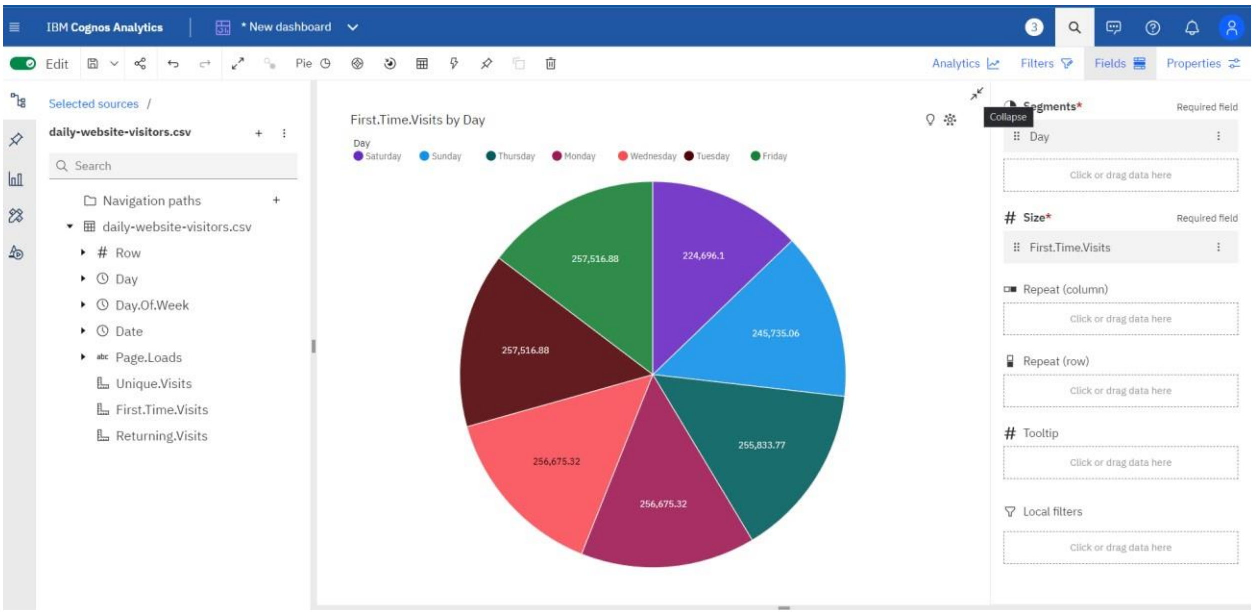
INSIGHTS GATHERED FROM IBM COGNOS

First.Time.Visits has a moderate upward trend.

Add insight to favorites

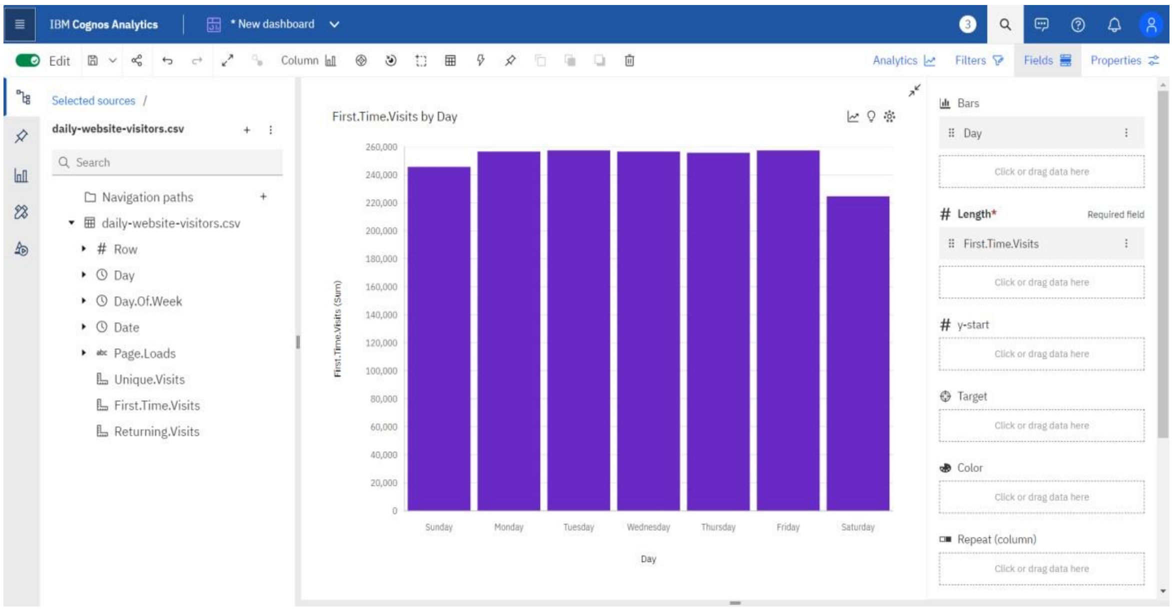
- Based on the current forecasting, First.Time.Visits may reach over 239 thousand by Day Monday+1.
- Friday (14.7 %), Tuesday (14.7 %), Monday (14.6 %), Wednesday (14.6 %), and Thursday (14.6 %) are the most frequently occurring categories of Day with a combined count of 1526 items with First.Time.Visits values (73.2 % of the total) .
- Over all days, the average of First.Time.Visits is 841.6.
- The total number of results for First.Time.Visits, across all days, is over two thousand.

```
First.Time.Visits ranges from nearly 225 thousand, when Day is Saturday,
to nearly 258 thousand, when Day is Tuesday.
```



First.Time.Visits has a moderate upward trend.

- Based on the current forecasting, First.Time.Visits may reach over 239 thousand by Day Monday+1.
- Over all days, the sum of First.Time.Visits is almost 1.8 million.
- First.Time.Visits ranges from nearly 225 thousand, when Day is Saturday, to nearly 258 thousand, when Day is Tuesday.
- For First.Time.Visits, the most significant values of Day are Friday, Tuesday, Monday, Wednesday, and Thursday, whose respective First.Time.Visits values add up to almost 1.3 million, or 73.2 % of the total.



-

Returning.Visits by Day

Based on the current forecasting, Returning.Visits may reach 183.4 by Day Monday+1.

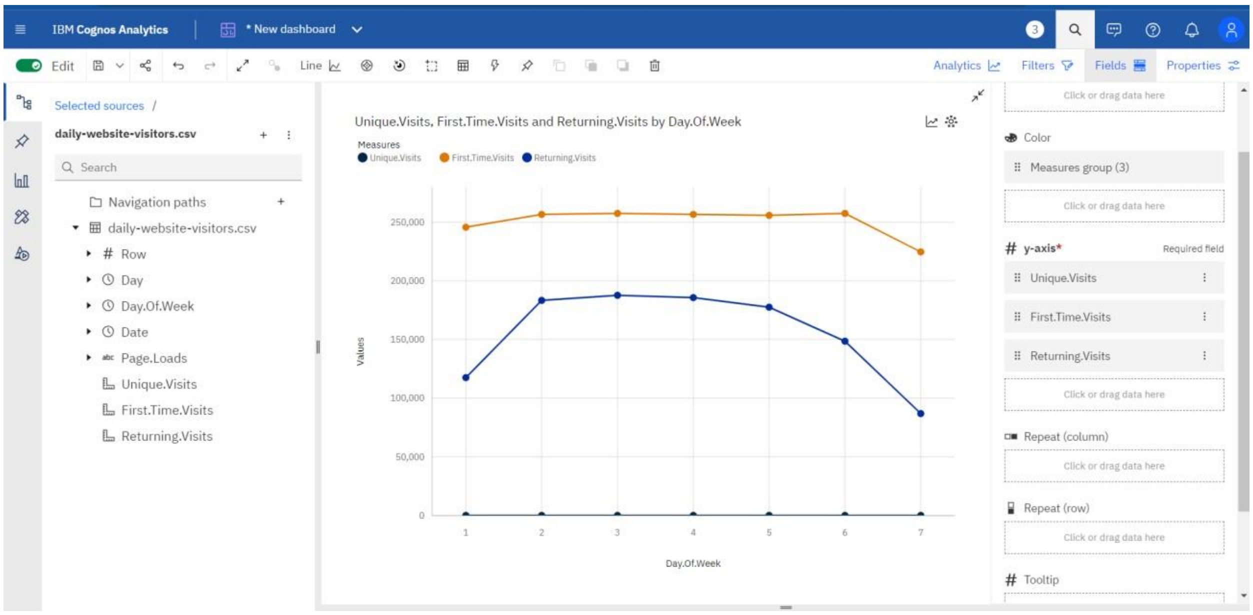
Add insight to favorites

The total number of results for Day, across all Returning.Visits, is over two thousand.

Based on the current forecasting, Unique.Visits may reach 1 by Day.Of.Week 9.

- 3 (14.7 %), 6 (14.7 %), 2 (14.6 %), 4 (14.6 %), and 5 (14.6 %) are the most frequently occurring categories of Day.Of.Week with a combined count of 1526 items with First.Time.Visits values (73.2 % of the total) .

- 3 (14.7 %), 6 (14.7 %), 2 (14.6 %), 4 (14.6 %), and 5 (14.6 %) are the most frequently occurring categories of Day.Of.Week with a combined count of 1526 items with Returning.Visits values (73.2 % of the total) .
- Over all values of Day.Of.Week, the average of First.Time.Visits is 841.6.
- Over all values of Day.Of.Week, the average of Returning.Visits is 521.4.
- The total number of results for First.Time.Visits, across all Day.Of.Week, is over two thousand.
- The total number of results for Returning.Visits, across all Day.Of.Week, is over two thousand.
- The total number of results for Unique.Visits, across all Day.Of.Week, is over two thousand.
- First.Time.Visits ranges from nearly 225 thousand, when Day.Of.Week is 7, to nearly 258 thousand, when Day.Of.Week is 3.
- Returning.Visits ranges from nearly 87 thousand, when Day.Of.Week is 7, to almost 188 thousand, when Day.Of.Week is 3.

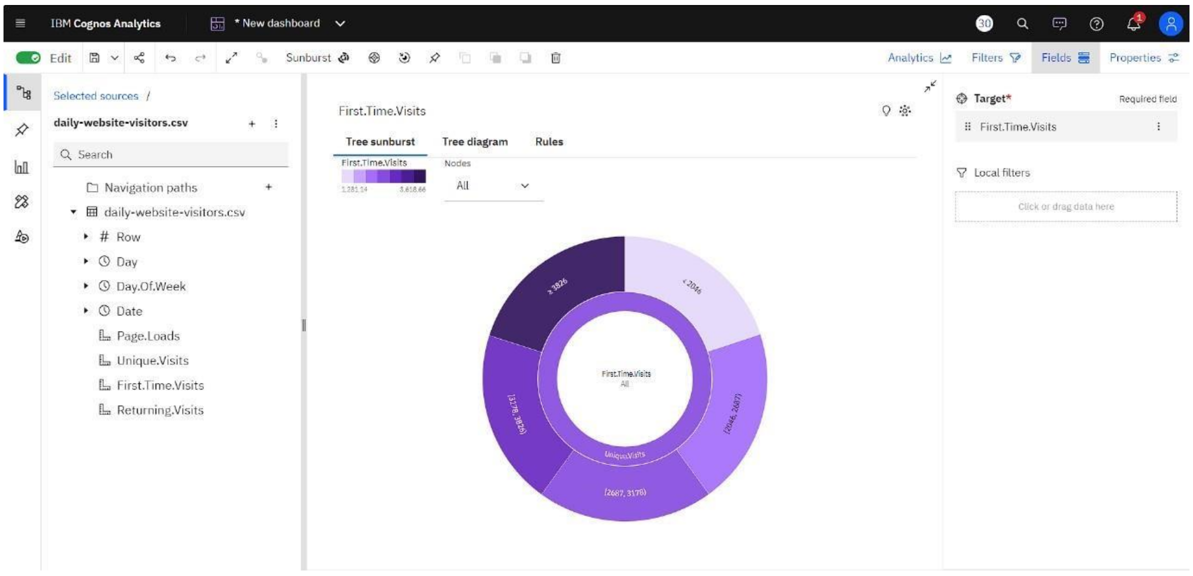


WORKING WITH IBM COGNOS

INSIGHTS GATHERED FROM IBM COGNOS

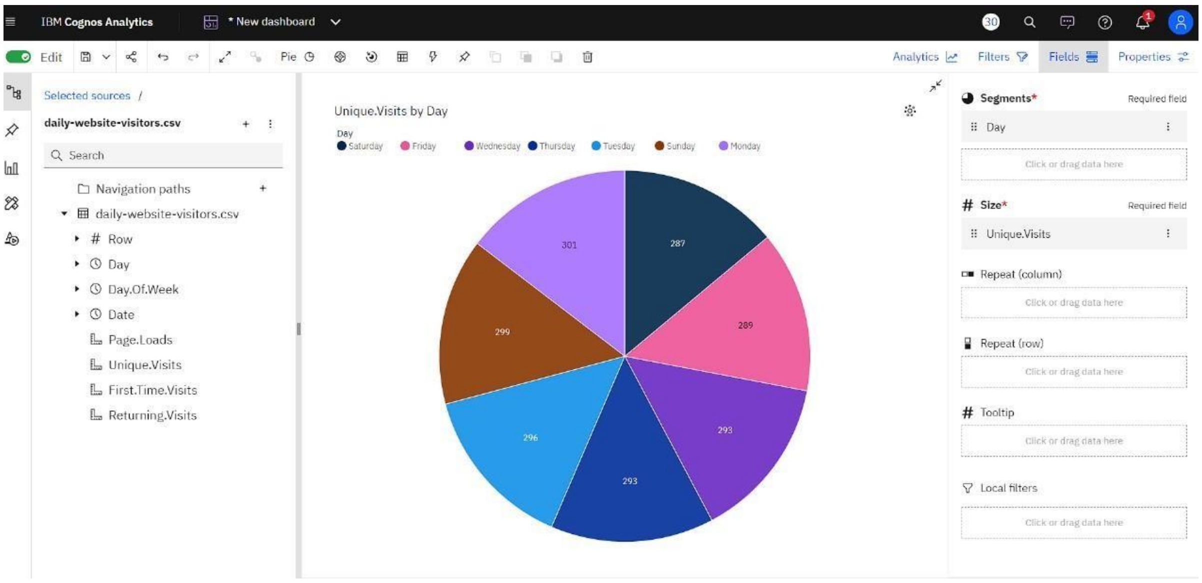
- Unique.Visits has a strong downward trend.
- Add insight to favorites

- Based on the current forecasting, Unique.Visits may reach 281.5 by Day Monday+1.
- Add insight to favorites
- Monday (14.3 %), Sunday (14.3 %), Wednesday (14.3 %), and Tuesday (14.3 %) are the most frequently occurring categories of Day with a combined count of 1240 items with Unique.Visits values (57.2 % of the total).
- Add insight to favorites
- The total number of results for Unique.Visits, across all days, is over two thousand.



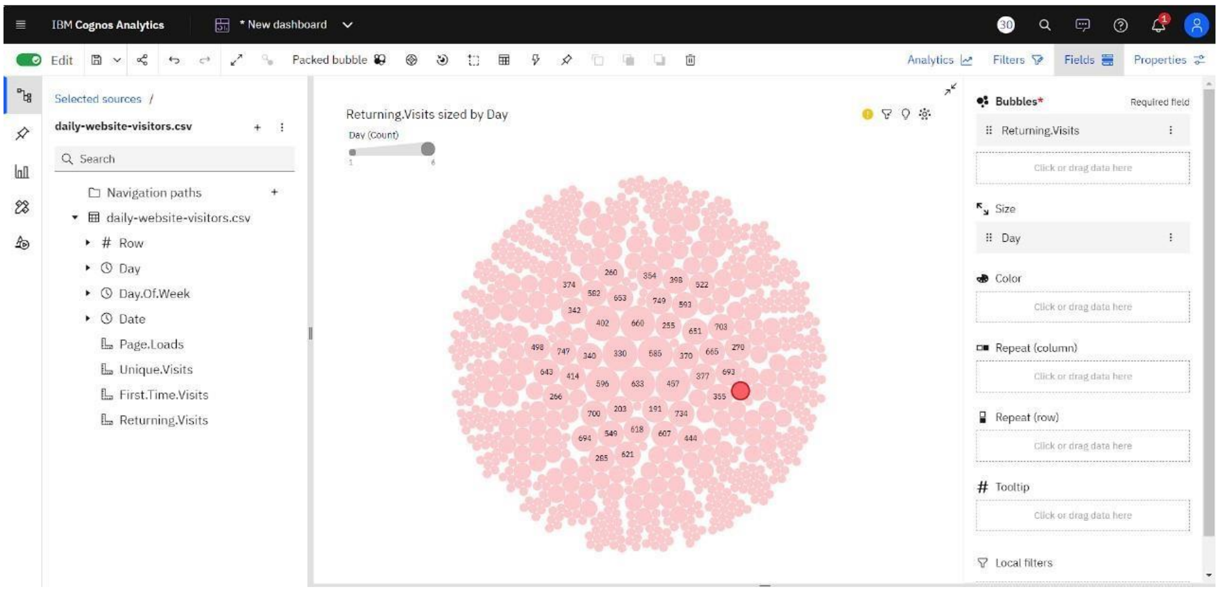
lowest average & highest average & forecasting

- Day Saturday has the lowest average First.Time.Visits at almost 1500, followed by Sunday at nearly 2 thousand.
- Add insight to favorites
- Day Tuesday has the highest average First.Time.Visits at 2928.23, followed by Wednesday at 2895.49.
- Add insight to favorites
- Based on the current forecasting, First.Time.Visits may reach almost 1500 by Day Monday+1.

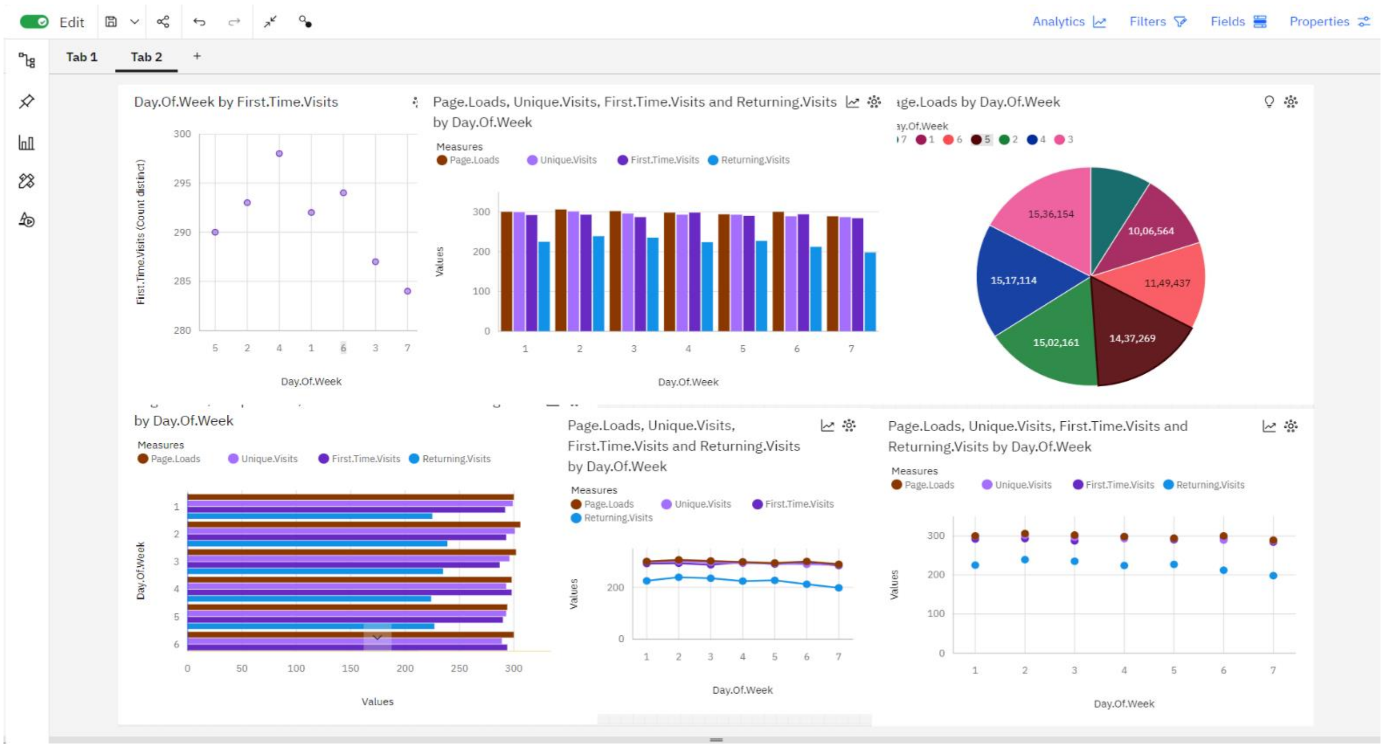


All Returning.Visits

The total number of results for Day, across all Returning.Visits, is over two thousand.



Final Dashboard


Edit
Tab 1
Tab 2
Analytics
Filters
Fields
Properties
Day.Of.Week by First.Time.Visits
First.Time.Visits (Count distinct)
Day.Of.Week
Page.Loads, Unique.Visits, First.Time.Visits and Returning.Visits by Day.Of.Week
Measures
Page.Loads
Unique.Visits
First.Time.Visits
Returning.Visits
Values
Day.Of.Week
ge.Loads by Day.Of.Week
15,36,154
10,06,564
11,49,437
14,37,269
15,02,161
15,17,114
by Day.Of.Week
Page.Loads, Unique.Visits, First.Time.Visits and Returning.Visits by Day.Of.Week
Page.Loads, Unique.Visits, First.Time.Visits and Returning.Visits by Day.Of.Week

Insights

Day.Of.Week by First.Time.Visits

1) Based on the current forecasting, First.Time.Visits may reach 291.5 by Day.Of.Week 9.

2) 1 (14.3 %), 2 (14.3 %), 3 (14.3 %), and 4 (14.3 %) are the most frequently occurring categories of Day.Of.Week with a combined count of 1240 items with First.Time.Visits values (57.2 % of the total).

3) The total number of results for First.Time.Visits, across all Day.Of.Week, is over two thousand.

Page.Loads, Unique.Visits, First.Time.Visits and Returning.Visits by Day.Of.Week

- Based on the current forecasting, Page.Loads may reach 285.1 by Day.Of.Week 9.
- Add insight to favorites
- The total number of results for First.Time.Visits, across all Day.Of.Week, is over two thousand.
- Add insight to favorites
- The total number of results for Page.Loads, across all Day.Of.Week, is over two thousand.
- Add insight to favorites
- The total number of results for Returning.Visits, across all Day.Of.Week, is over two thousand.
- Add insight to favorites
- The total number of results for Unique.Visits, across all Day.Of.Week, is over two thousand.

Page.Loads by Day.Of.Week

Page.Loads is unusually low when Day.Of.Week is 7.

Based on the current forecasting, Page.Loads may reach over 675 thousand by Day.Of.Week 9.

1 (14.3 %), 2 (14.3 %), 3 (14.3 %), and 4 (14.3 %) are the most frequently occurring categories of Day.Of.Week with a combined count of 1240 items with Page.Loads values (57.2 % of the total).

1 (14.3 %), 2 (14.3 %), 3 (14.3 %), and 4 (14.3 %) are the most frequently occurring categories of Day.Of.Week with a combined count of 1240 items with Returning.Visits values (57.2 % of the total).

Across all values of Day.Of.Week, the average of Page.Loads is over four thousand.

Over all values of Day.Of.Week, the average of Returning.Visits is 511.8.

The total number of results for Page.Loads, across all Day.Of.Week, is over two thousand.

The total number of results for Returning.Visits, across all Day.Of.Week, is over two thousand.

Page.Loads ranges from nearly 773 thousand, when Day.Of.Week is 7, to over 1.5 million, when Day.Of.Week is 3.

Returning.Visits ranges from almost 96 thousand, when Day.Of.Week is 7, to over 189 thousand, when Day.Of.Week is 3.

Page.Loads, Unique.Visits, First.Time.Visits and Returning.Visits by Day.Of.Week

Based on the current forecasting, Page.Loads may reach 285.1 by Day.Of.Week 9.

The total number of results for First.Time.Visits, across all Day.Of.Week, is over two thousand.

The total number of results for Page.Loads, across all Day.Of.Week, is over two thousand.

The total number of results for Returning.Visits, across all Day.Of.Week, is over two thousand.

The total number of results for Unique.Visits, across all Day.Of.Week, is over two thousand.

Page.Loads, Unique.Visits, First.Time.Visits and Returning.Visits by Day.Of.Week

Based on the current forecasting, Page.Loads may reach 285.1 by Day.Of.Week 9.

The total number of results for First.Time.Visits, across all Day.Of.Week, is over two thousand.

The total number of results for Page.Loads, across all Day.Of.Week, is over two thousand.

The total number of results for Returning.Visits, across all Day.Of.Week, is over two thousand.

The total number of results for Unique.Visits, across all Day.Of.Week, is over two thousand.

Page.Loads, Unique.Visits, First.Time.Visits and Returning.Visits by Day.Of.Week

Based on the current forecasting, Page.Loads may reach 285.1 by Day.Of.Week 9.

The total number of results for First.Time.Visits, across all Day.Of.Week, is over two thousand.

The total number of results for Page.Loads, across all Day.Of.Week, is over two thousand.

The total number of results for Returning.Visits, across all Day.Of.Week, is over two thousand.

The total number of results for Unique.Visits, across all Day.Of.Week, is over two thousand.

Conclusion:

In conclusion, through the utilization of predictive models, comprehensive traffic analysis, and insightful data visualizations, we have gained valuable insights from the website traffic data. These insights enable us to make informed decisions to enhance the user experience, refine content strategies, and optimize online performance, ultimately driving improved outcomes and engagement.