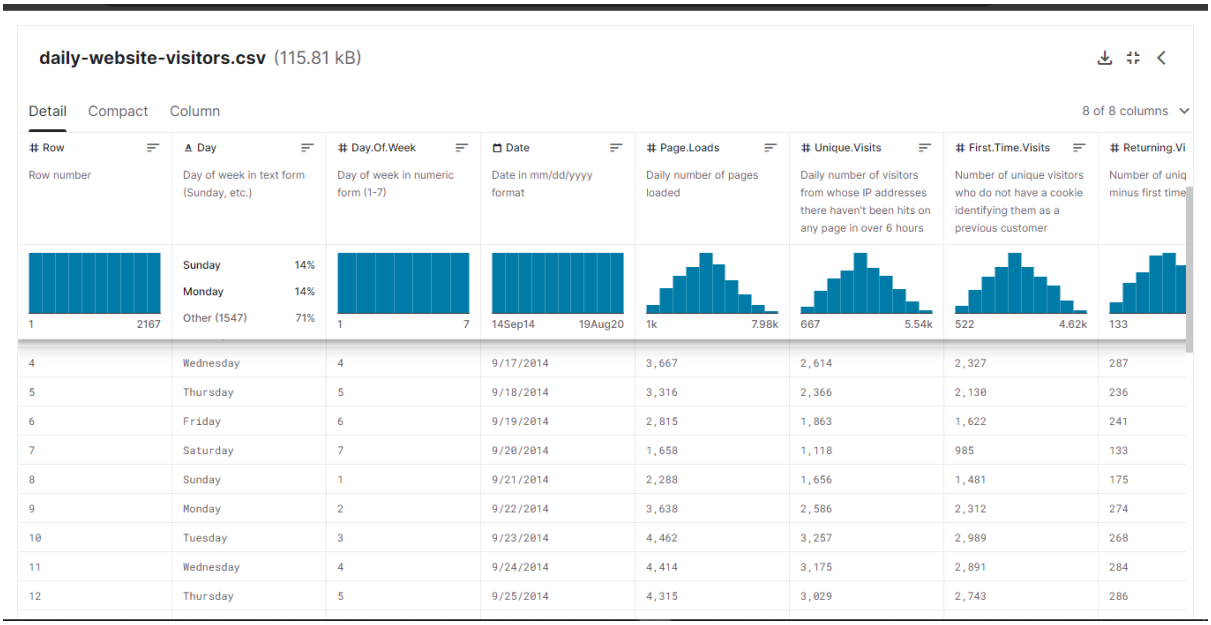
WEBSITE TRAFFIC ANALYSIS

PHASE 3: Development Part 1

Start building the website traffic analysis by loading and pre-processing the dataset.

GIVEN DATA SET:

The given dataset contains the variables are daily counts of page loads, unique visitors, first-time visitors, and returning visitors to an academic teaching notes website. There are 2167 rows of data spanning the date range from September 14, 2014, to August 19, 2020.



daily-website-visitors.csv (115.81 kB)

Detail Compact Column — 8 of 8 columns

Row	Day	Day.Of.Week	Date	Page.Loads	Unique.Visits	First.Time.Visits	Returning.Vi
Row number	Day of week in text form (Sunday, etc.)	Day of week in numeric form (1-7)	Date in mm/dd/yyyy format	Daily number of pages loaded	Daily number of visitors from whose IP addresses there haven't been hits on any page in over 6 hours	Number of unique visitors who do not have a cookie identifying them as a previous customer	Number of uniq minus first time
1 – 2167	Sunday 14%, Monday 14%, Other (1547) 71%	1 – 7	14Sep14 – 19Aug20	1k – 7.98k	667 – 5.54k	522 – 4.62k	133
4	Wednesday	4	9/17/2014	3,667	2,614	2,327	287
5	Thursday	5	9/18/2014	3,316	2,366	2,130	236
6	Friday	6	9/19/2014	2,815	1,863	1,622	241
7	Saturday	7	9/20/2014	1,658	1,118	985	133
8	Sunday	1	9/21/2014	2,288	1,656	1,481	175
9	Monday	2	9/22/2014	3,638	2,586	2,312	274
10	Tuesday	3	9/23/2014	4,462	3,257	2,989	268
11	Wednesday	4	9/24/2014	4,414	3,175	2,891	284
12	Thursday	5	9/25/2014	4,315	3,029	2,743	286

NECESSARY STEP TO FOLLOW:

1.IMPORT NECESSARY LIBRARIES:

Start by importing the necessary libraries:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns from scipy.stats import mode

2.LOAD THE DATASET:

Load your dataset into a Pandas DataFrame. You can typically find Website traffic datasets in CSV format, but you can adapt this code to other formats as needed.

Program:

```
# Replace 'your_dataset.csv' with the actual file path of your dataset
df = pd.read_csv('your_dataset.csv')
```

3.EXPLORE THE DATASET:

Once you've loaded the dataset, it's essential to explore it to understand its structure and contents. Use functions like df.head() to view the first few rows, df.info() to get an overview of the dataset, and df.describe() for statistical summary.

Program:

```
# View the first few rows
print(df.head())
# Get an overview of the dataset
print(df.info())
# Statistical summary of the dataset
print(df.describe())
```

4.DATA PREPROCESSING:

Data preprocessing is crucial to handle missing values, outliers, and ensure data consistency. Some common preprocessing steps include:

Handling missing values:

Use df.dropna() or df.fillna() to drop or fill missing values.

Handling duplicates:

Use df.drop_duplicates() to remove duplicate entries if necessary.

Data type conversion:

Use df.astype() to convert data types if needed. Outlier detection: Use statistical methods or visualization techniques to identify and handle outliers.

Data normalization or scaling:

Use techniques like Min-Max scaling or Standardization to scale numerical data if required.

‘‘‘

```
# Handle missing values
df = df.dropna()  # Drop rows with missing values
# Handle duplicates
df = df.drop_duplicates()  # Drop duplicate entries
# Data type conversion
# Example: Convert a column to datetime
df['timestamp'] = pd.to_datetime(df['timestamp'])
# Outlier detection and handling
# Implement outlier detection techniques such as z-score or IQR
# Data normalization or scaling
```

Implement scaling techniques as per the requirements""

By following these steps, you can effectively load and preprocess your website traffic dataset. Make sure to tailor the preprocessing steps based on the specific characteristics of your dataset and the requirements of your analysis objectives.

PROGRAM:

```
# Apply the preprocessing functions data['Date'] = pd.to_datetime(data['Date'])
data['Page.Loads'] = data['Page.Loads'].apply(lambda x : remove_commas(x))
data['Unique.Visits'] = data['Unique.Visits'].apply(lambda x : remove_commas(x))
data['First.Time.Visits'] = data['First.Time.Visits'].apply(lambda x : remove_commas(x))
data['Returning.Visits'] = data['Returning.Visits'].apply(lambda x : remove_commas(x))
```

5.EXPLORATORY DATA ANALYSIS (EDA):

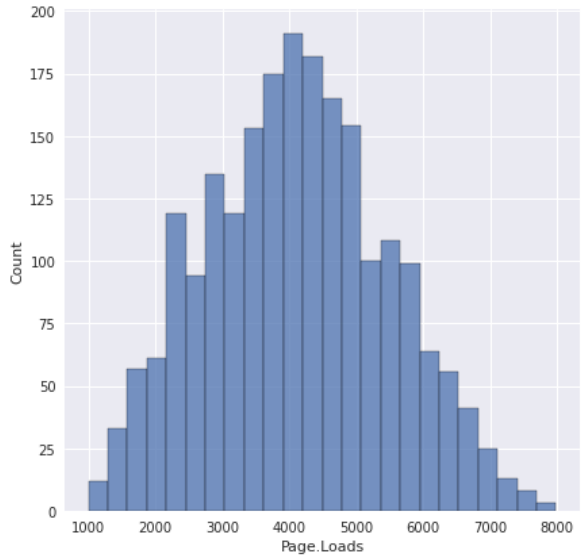
Depending on your dataset, you may need to create new features or transform existing ones. This can involve one-hot encoding categorical variables, handling date/time data, or scaling numerical features.

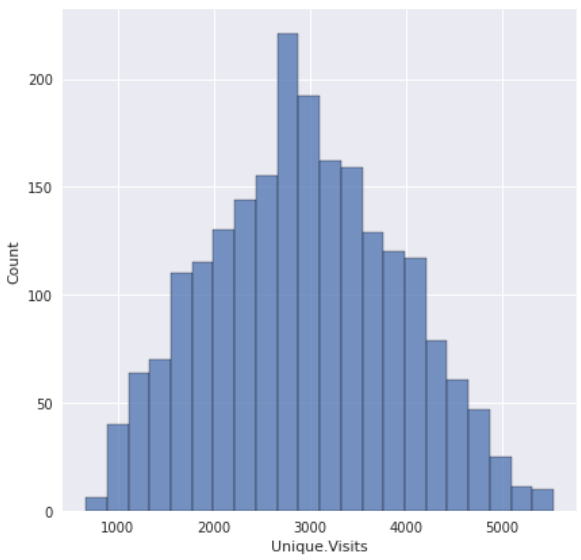
High positive correlation can be observed between the following features:

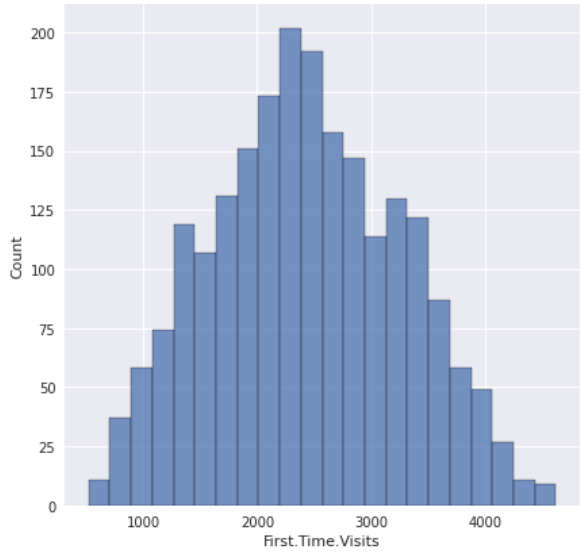
1. Page.Loads and Returning.Visits
2. Returning.Visits and Unique.Visits
3. Returning.Visits and First.Time.Visits

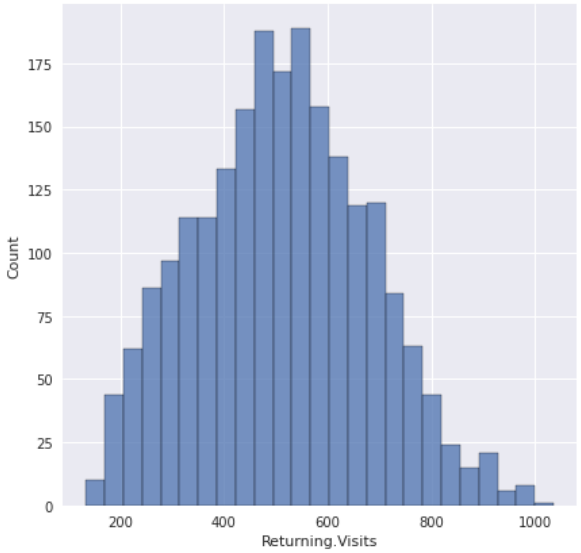
PROGRAM:EX

```
# Frequency distribution of each continuous column
cols_to_plot = ['Page.Loads', 'Unique.Visits', 'First.Time.Visits', 'Returning.Visits']
plt.figure(figsize=(15, 15))
for i, col in enumerate(cols_to_plot):
    plt.subplot(2, 2, i+1)
    sns.histplot(data=data, x=col)
```









CONCLUSION:

- In the quest to build a website traffic analysis model, we have embarked on a critical journey that begins with loading and preprocessing the dataset.We have traversed through essential steps, starting with importing the necessary libraries to facilitate data manipulation and analysis.
- Understanding the data's structure, characteristics, and any potential issues through exploratory data analysis (EDA) is essential for informed decision-making.
- Data preprocessing emerged as a pivotal aspect of this process. It involves cleaning, transforming, and refining the dataset to ensure that it aligns with the requirements of machine learning algorithms.
- With these foundational steps completed, our dataset is now primed for the subsequent stages of building and training a website traffic prediction model.