WEBSITE TRAFFIC ANALYSIS

INTRODUCTION:

To analyze website traffic data to understand user behaviour, popular pages and traffic sources , helping website owners to improve user experience. Website traffic analysis is a crucial aspect of data analytics, providing valuable insights into user behavior and interaction patterns on a website. Analyzing website traffic helps businesses and website owners understand their audience, optimize user experience, and make data-driven decisions to enhance their online presence.

DATASET:

DATA SOURCE:

https://www.kaggle.com/datasets/bobnau/daily-website-visitors

PREPROCESSING:

PROGRAM:

```
import numpy as np

import pandas as pd

import os

for dirname, _, filenames in os.walk('/kaggle/input'):

    for filename in filenames:

        print(os.path.join(dirname, filename))

# Import Libraries

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats import mode

data=pd.read_csv("/content/daily-website-visitors.csv")

data.head()
```

Row	Day	Day.Of.Week	Date	Page.Loads	Unique.Visits	First.Time.Visits	Returning.Visits
0	1	Sunday	9/14/2014	2,146	1,582	1,430	152
1	2	Monday	9/15/2014	3,621	2,528	2,297	231
2	3	Tuesday	9/16/2014	3,698	2,630	2,352	278
3	4	Wednesday	9/17/2014	3,667	2,614	2,327	287
4	5	Thursday	9/18/2014	3,316	3,366	2,130	236

# Data preprocessing

* 1.Convert Date into Datetime format.

* 2.Removing ',' from Page.Loads, Unique.Visits, First.Time.Visits, Returning.Visits.

* 3.Convert the above values into float.

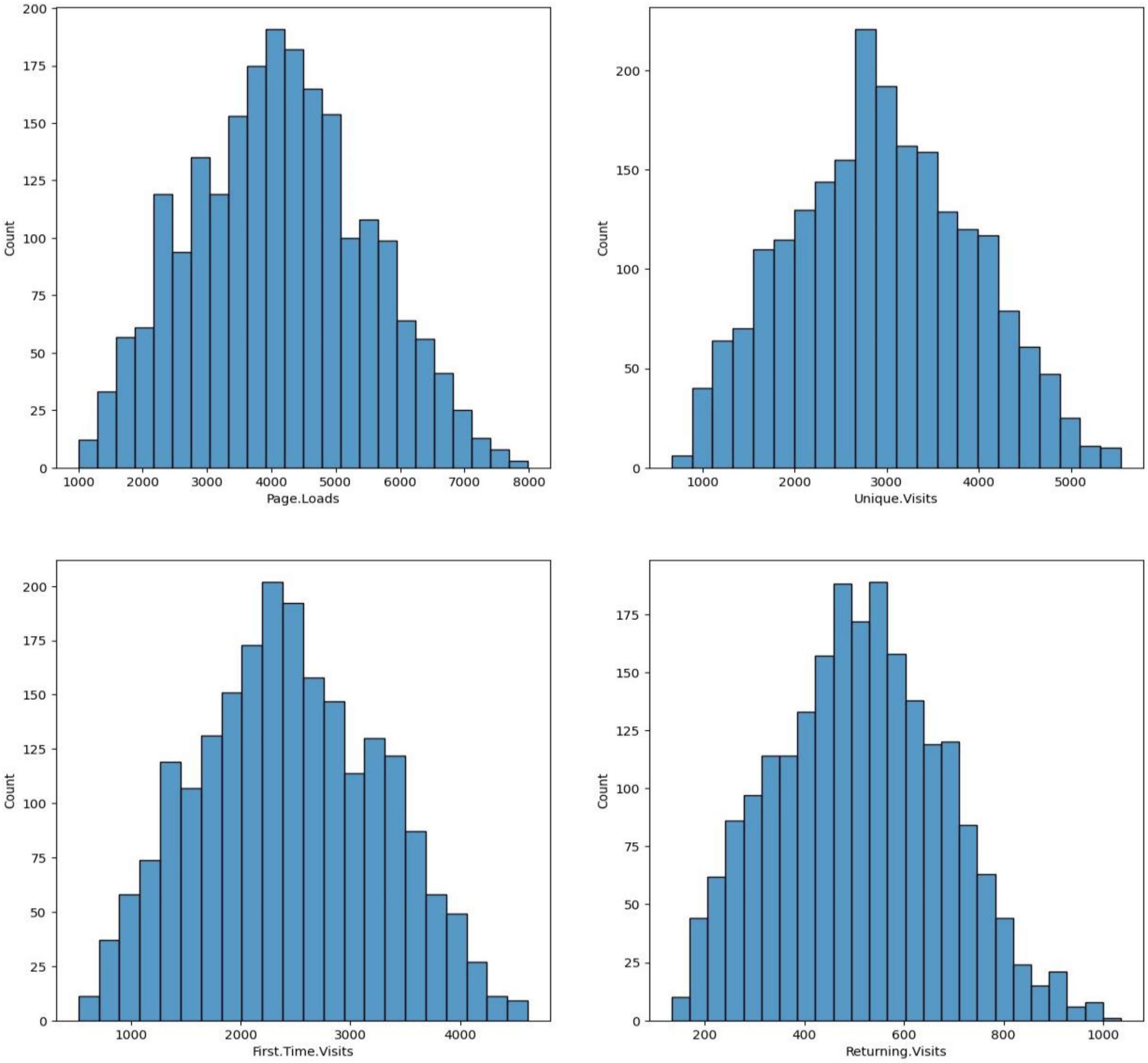
```
# Function to remove commas
def remove_commas(x):
    return float(x.replace(',', ''))

# 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))
data.head()
```

EXPLOATORY DATA ANALYSIS

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)
```



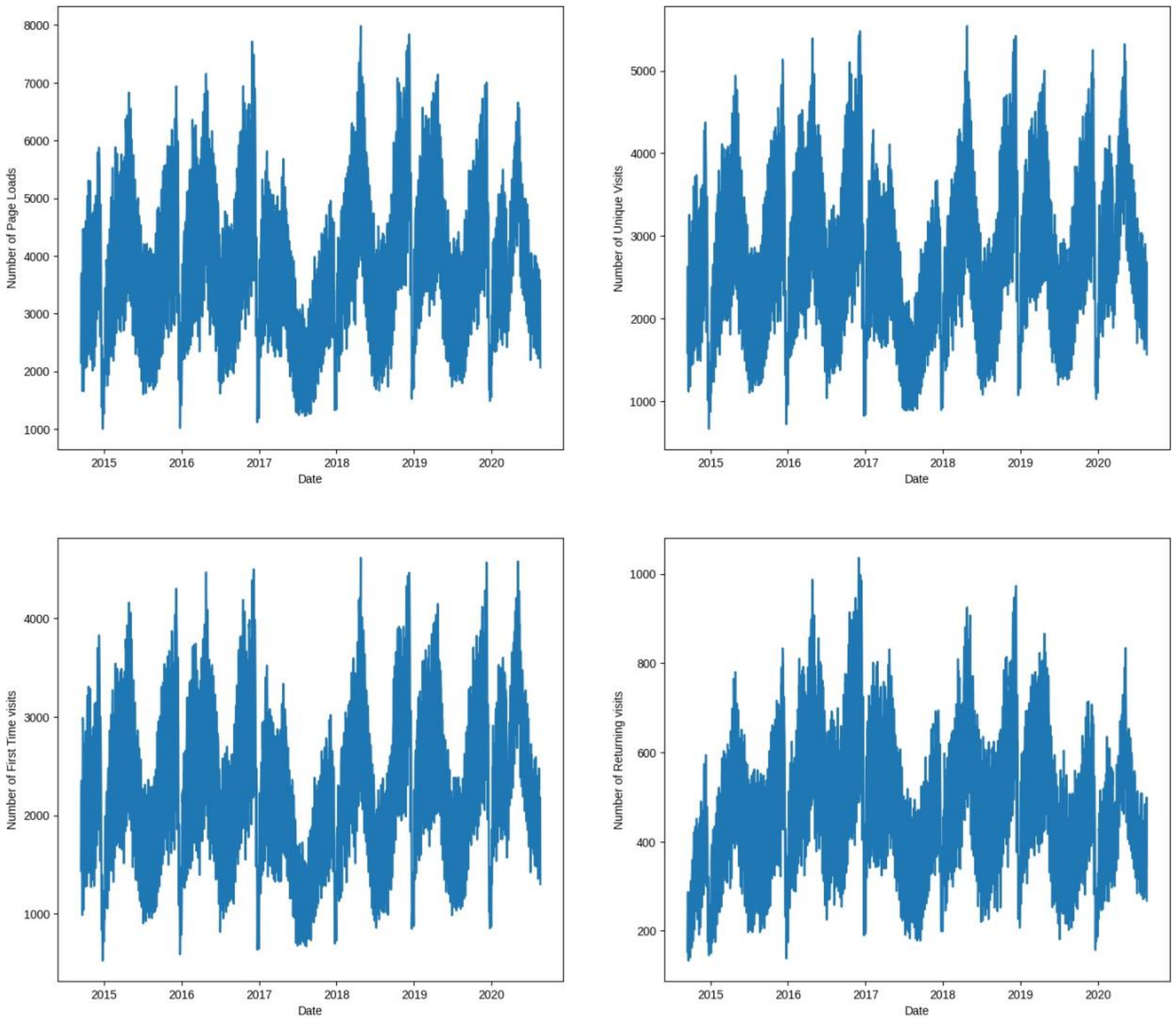
```
Def check_normality(data, col):

   # Compute mean

   mean = int(np.mean(data[col]))

   median = int(np.median(data[col]))
```

```
    mode_ = int(mode(data[col])[0][0])

    print("mean", ":", mean, "median", ":", median, "mode", ":", mode_)
    if mean == median == mode_:
        print("{} Distribution is Normal".format(col))
    elif mean > median and mean > mode_ and mode_ < median:
        print("{} Distribution is skewed towards right".format(col))
    else:
        print("{} Distribution is skewed towards left".format(col))
for col in cols_to_plot:
    check_normality(data, col)
```

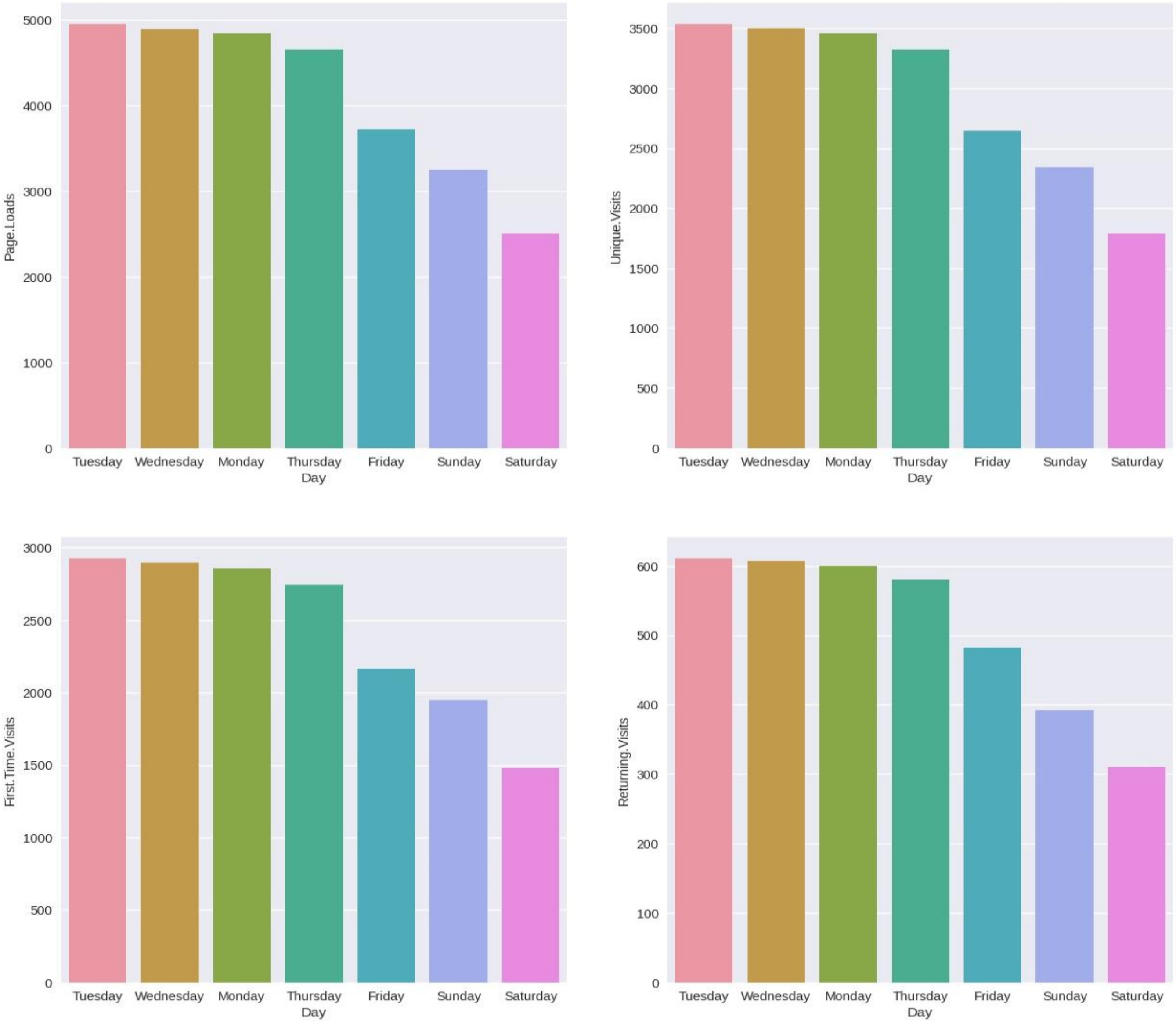
```
* mean : 4116 median : 4106 mode : 2948
* Page.Loads Distribution is skewed towards right
* mean : 2943 median : 2914 mode : 1197
* Unique.Visits Distribution is skewed towards right
* mean : 2431 median : 2400 mode : 3133
* First.Time.Visits Distribution is skewed towards left
* mean : 511 median : 509 mode : 552
* Returning.Visits Distribution is skewed towards left
```

```
# Perform the EDA
figure, ax = plt.subplots(2, 2, figsize=(17, 15))
plt.style.use('seaborn')
ax1 = ax[0]
ax2 = ax[1]

# Plot the Number of Page Loads with time
ax1[0].plot(data['Date'], data['Page.Loads'])
ax1[0].set_xlabel("Date")
ax1[0].set_ylabel("Number of Page Loads")
```

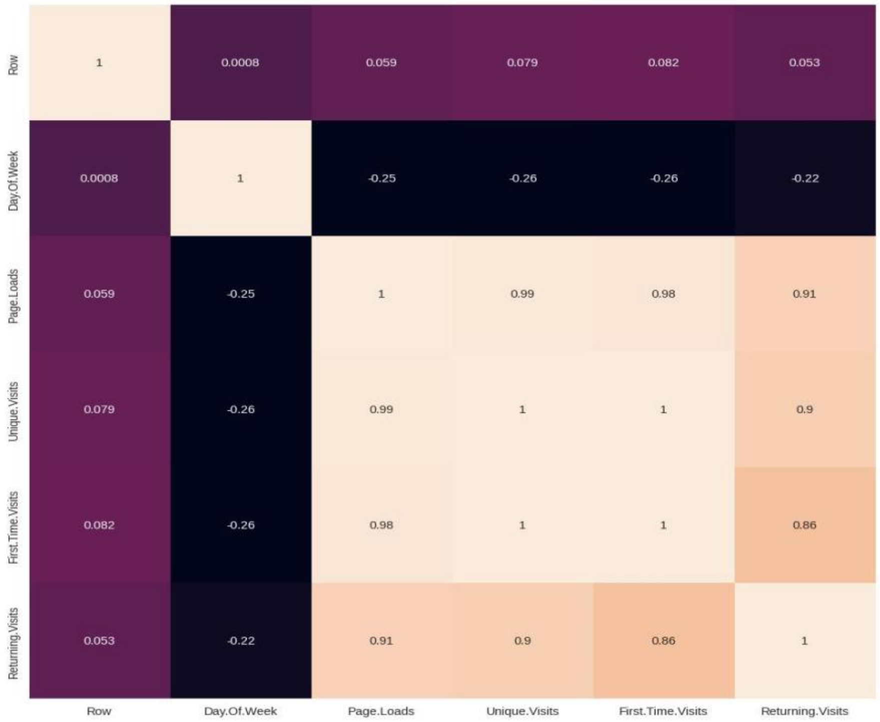
```
# Plot the Number of Unique Visits with time
ax1[1].plot(data['Date'], data['Unique.Visits'])
ax1[1].set_xlabel("Date")
ax1[1].set_ylabel("Number of Unique Visits")


# Plot the Number of First Time visits with time
ax2[0].plot(data['Date'], data['First.Time.Visits'])
ax2[0].set_xlabel("Date")
ax2[0].set_ylabel("Number of First Time visits")


# Plot the Number of Returning visits with time
ax2[1].plot(data['Date'], data['Returning.Visits'])
ax2[1].set_xlabel("Date")
ax2[1].set_ylabel("Number of Returning visits")
figure.show()
```



```
# Plot the Bargraph for every continuous variable across day

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.barplot(data=avg_day_data.sort_values(by=col, ascending=False), x='Day', y=col)
```



```
#Plot the correlation heatmap
Corr_matrix = data.corr()
Plt.figure(figsize=(12,12))
Sns.heatmap(corr_matrix, annot=True, cbar=False)
Plt.show()
```



	Row	Day.Of.Week	Page.Loads	Unique.Visits	First.Time.Visits	Returning.Visits
Row	1	0.0008	0.059	0.079	0.082	0.053
Day.Of.Week	0.0008	1	-0.25	-0.26	-0.26	-0.22
Page.Loads	0.059	-0.25	1	0.99	0.98	0.91
Unique.Visits	0.079	-0.26	0.99	1	1	0.9
First.Time.Visits	0.082	-0.26	0.98	1	1	0.86
Returning.Visits	0.053	-0.22	0.91	0.9	0.86	1

High positive correlation can be observed between the following features:

- page.Loads and Returning.Visits
- Returning.Visits and Unique.Visits
- Returning.Visits and First.Time.Visits