**IBM DATA ANALYTICS COGNOS**

**PROJECT IV**

|  |
| --- |
| **NAME:JAYANTHI M**  **REG NO:42042104030**  **ROLL NO:14** |

**WEBSITE TRAFFIC ANALYSIS**

**Problem statement :**

\*In this part you will continue building your project.

\*Continue building the analysis by creating visualizations using IBM Cognos and integrating Python code for advanced analysis.

\*Use IBM Cognos to create interactive dashboards and reports that display insights such as popular pages, traffic sources, and user engagement metrics.

\*Use Python libraries like Pandas and Matplotlib to perform more complex analyses on the data, such as time series analysis, user segmentation, or machine learning-based predictions.

**Problem Solution :**

**Attribute Information**

* + **Row:** Number of rows starting from 0
  + **Day:** Day of the week (Monday through Sunday)
  + **Day.Of.Week:** Numeric form of Day (Sunday = 1, Monday = 2, ... , Friday = 6, Saturday = 7)
  + **Date:** Date of website visits
  + **Page.Loads:** Number of webpages loaded for visit
  + **Unique.Visits:** A visit is classified as "unique" if a hit from the same IP address has not come within the last 6 hours. The count of unique visitors is the sum of the counts of returning and first-time visitors by definition.
  + **First.Time.Visits:** Number of visitors that are not identified by cookies
  + **Returning.Visits:** Returning visitors are identified by cookies if those are accepted.

**Necessary step to follow:**

1.Import Libraries:

Start by importing the necessary libraries.

**Program:**

**import** pandas **as** pd

**import** numpy **as** np

**import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

**from** sklearn.metrics **import** accuracy\_score

**%matplotlib** inline

2.Load the Dataset:

Load your into a Pandas DataFrame.You can typically find house price datasets in csv format,but you can adapt this code to other formats as needed.

**Program:**

df = pd.read\_csv('daily-website-visitors.csv')

**Let's Visualize daily website visiters data and get some insight on our dataset!**

Python has lots of visualization tools from Pandas, Matplotlib and Seaborn Libraries. We'll use couples of them to visualize data. You'll find data visualization quite interesting and v interesting fact is without calculating/applying any algorthim we will be able to make intuition on Data.

**Relationship between Day.Of.Week and Page.Loads using scatter plot:**

plt.figure(figsize=(14,8))

ax = df[df.Day=='Sunday'].plot.scatter(x='Day.Of.Week', y='Page.Loads', label='Sunday')

df[df.Day=='Monday'].plot.scatter(x='Day.Of.Week', y='Page.Loads', color='red', label='Monday', ax=ax)

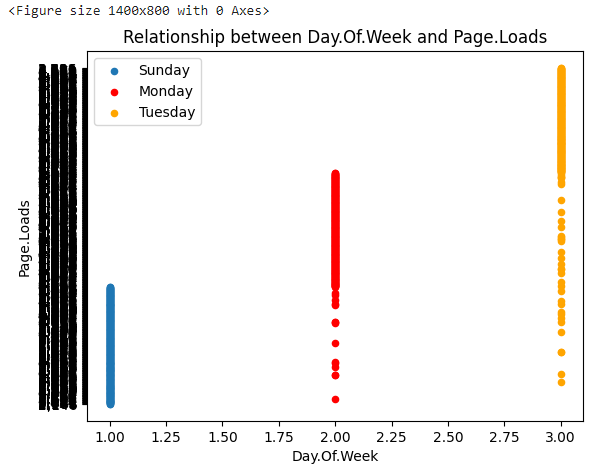
df[df.Day=='Tuesday'].plot.scatter(x='Day.Of.Week', y='Page.Loads', color='orange', label='Tuesday', ax=ax)

ax.set\_xlabel("Day.Of.Week")

ax.set\_ylabel("Page.Loads")

ax.set\_title("Relationship between Day.Of.Week and Page.Loads")

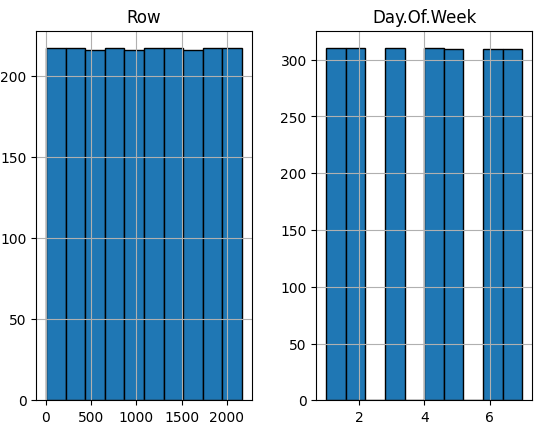
plt.show()

****

**Histogram:**

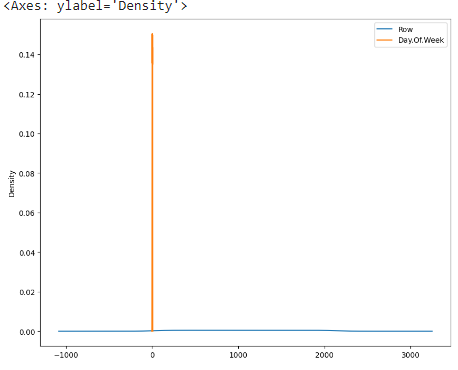
df**.**hist(edgecolor**=**'black', linewidth**=**1)

plt**.**show()

****

#### Take a look at density:

df**.**plot(kind **=** "density", figsize**=**(10,8))

**

### Seaborn:

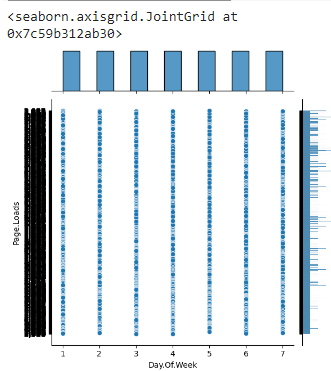
Seaborn has many nice functions for plotting. You can create jointplot, hisplot, pairplot, boxplot, violinplot(it's damn cool), heatmap polt etc etc. I recommend seaborn personally cause it has many cool ploting techniques!

**Program:**

import seaborn as sns

df = pd.read\_csv('daily-website-visitors.csv')

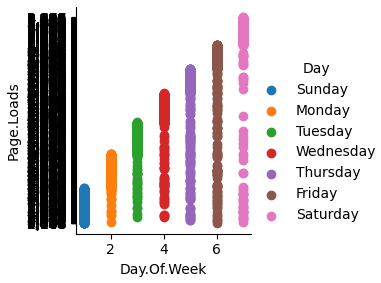
sns.jointplot(x='Day.Of.Week',y='Page.Loads', data = df)



#Seaborn **Scatterplot** by Day onDay.Of.Week vs Page.Loads

g = sns.FacetGrid(df, hue='Day')

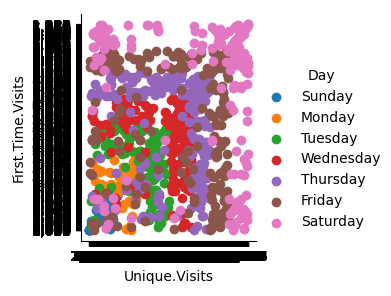
g = g.map(plt.scatter, 'Day.Of.Week','Page.Loads').add\_legend()



# seaborn scatterplot by Day onUnique.Visits vs First.Time.Visits

g = sns.FacetGrid(df, hue='Day')

g = g.map(plt.scatter, 'Unique.Visits', 'First.Time.Visits').add\_legend()

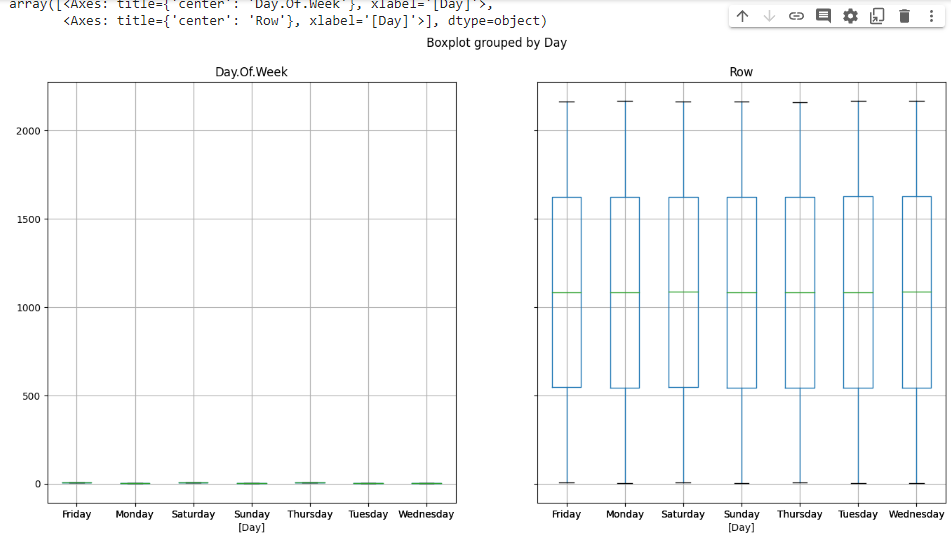
****

**Box Plot:**

***Box Plot*** is the visual representation of the depicting groups of numerical data through their quartiles. Boxplot is also used for detect the outlier in data set. It captures the summary of the data efficiently with a simple box and whiskers and allows us to compare easily across groups.

**Program:**

df.boxplot(by = 'Day', figsize = (16,8))



**Voilin Plot:**

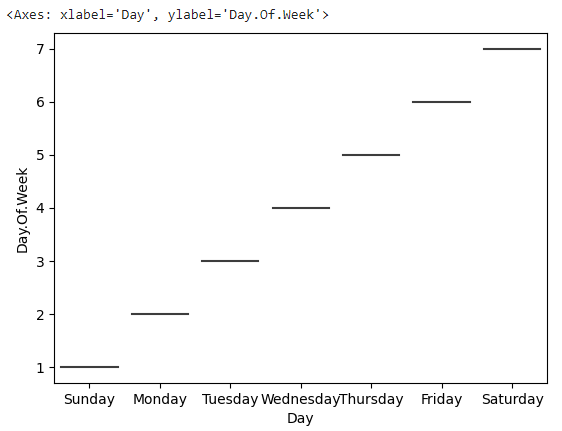
A violin plot plays a similar role as a box-and-whisker plot. It shows the distribution of data points after grouping by one (or more) variables. Unlike a box plot, each violin is drawn using a kernel density estimate of the underlying distribution.

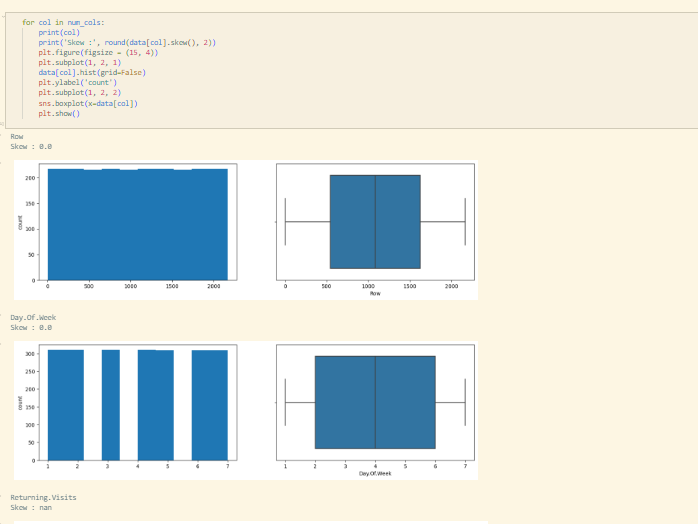
**Program:**

plt.figure(figsize=(14,10))

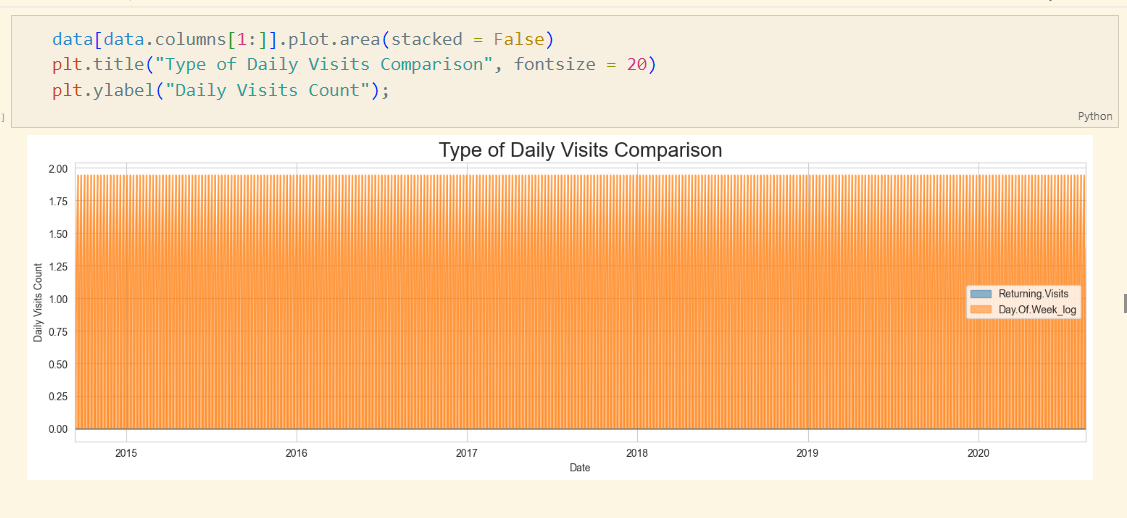
plt.subplot(2,2,1)

sns.violinplot(x='Day', y='Day.Of.Week', data=df, size=5)

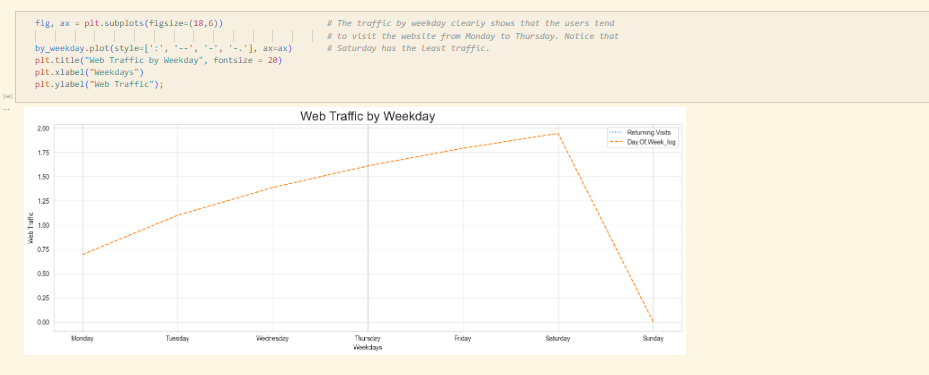




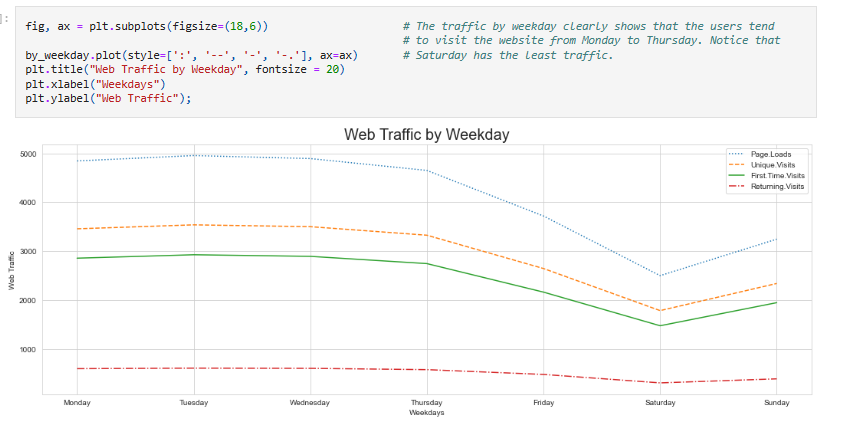
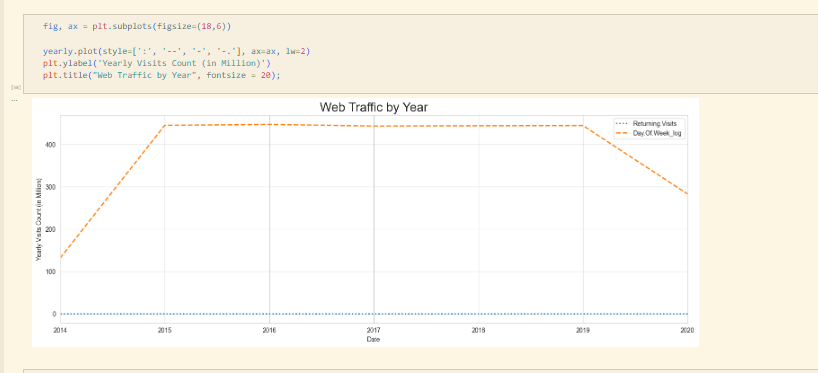
**Types of Daily Visits Comparison :**



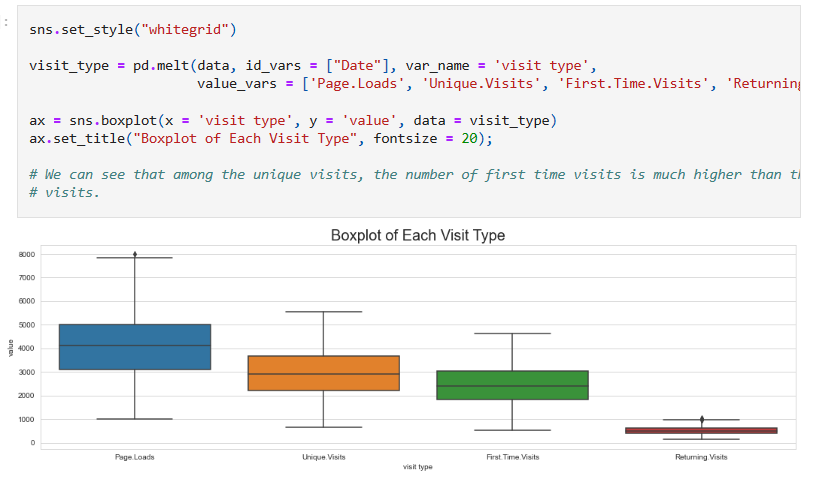
**Web Traffic by Wednesday:**



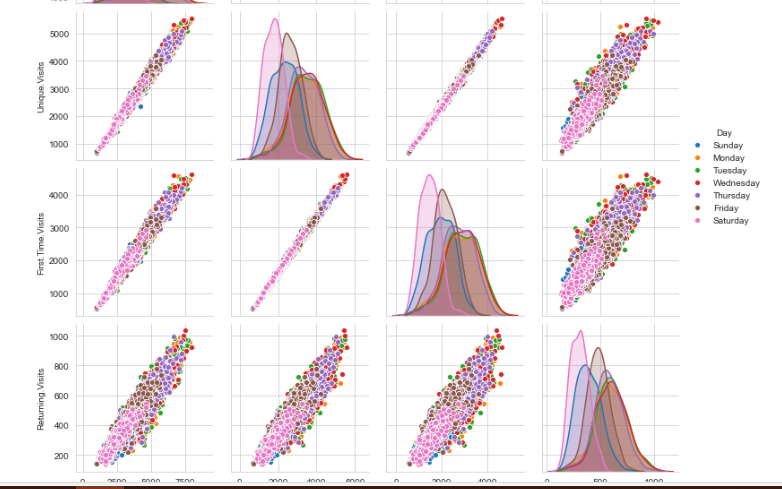
**Web traffic by year**

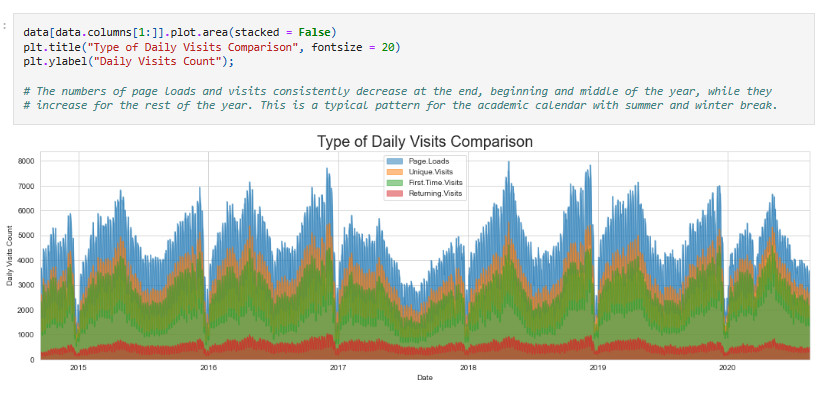


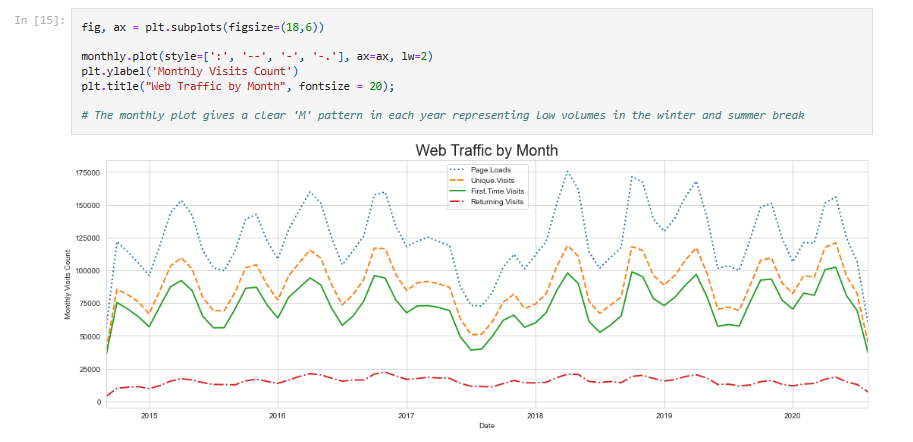
**Web traffic by month** 

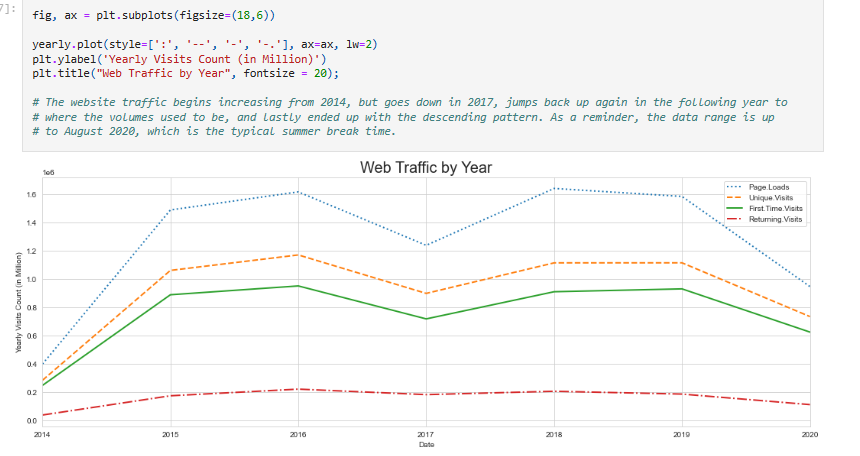


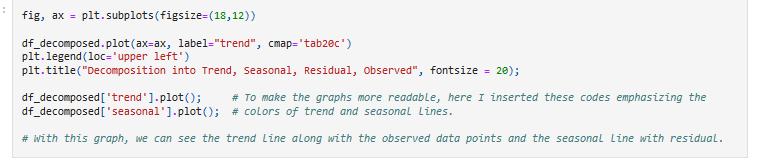


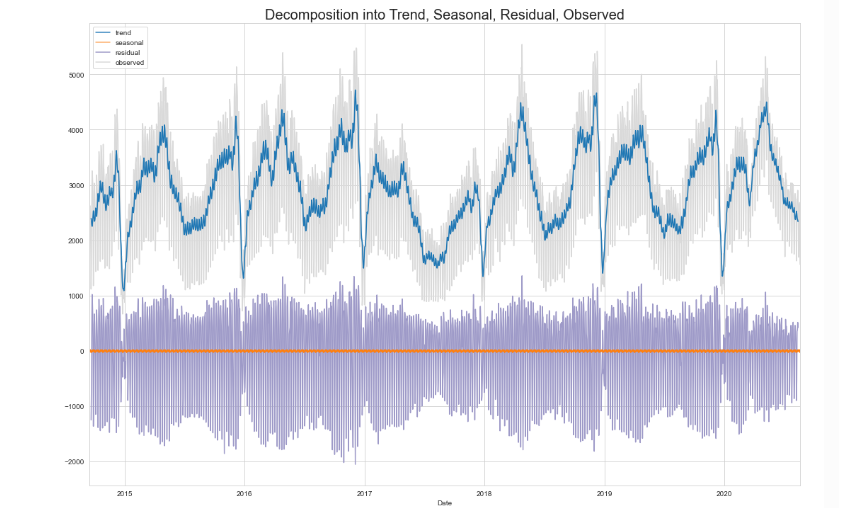


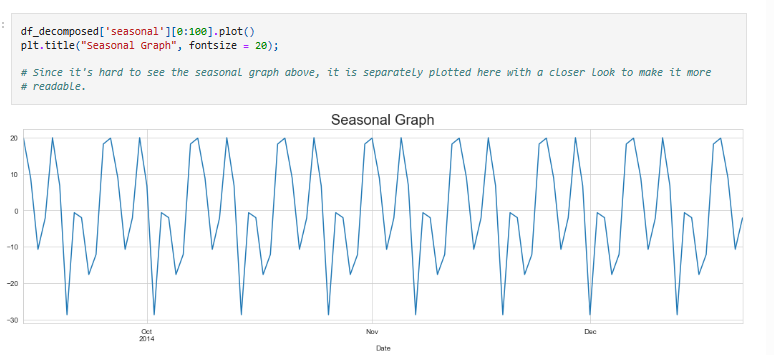


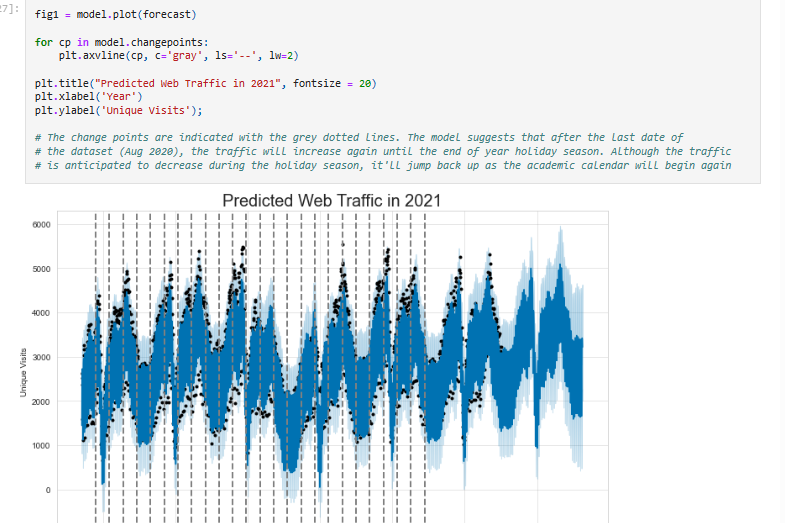
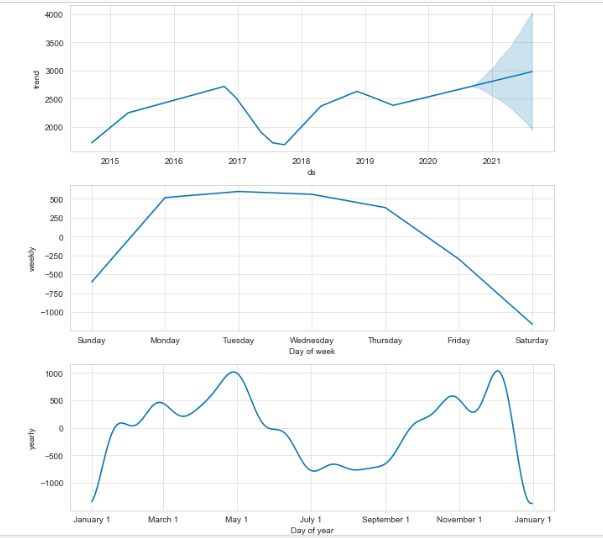






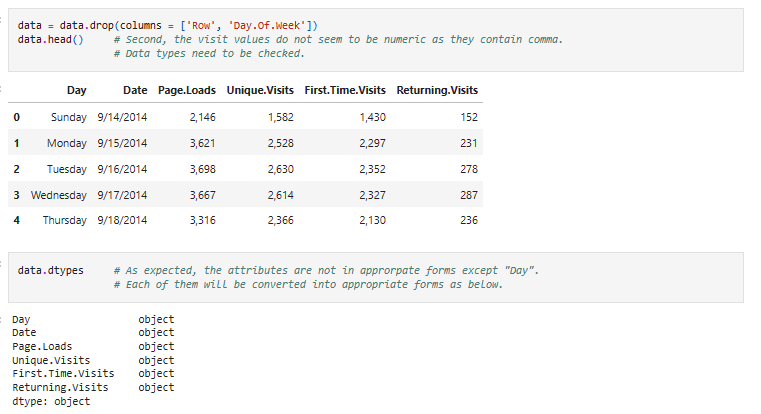


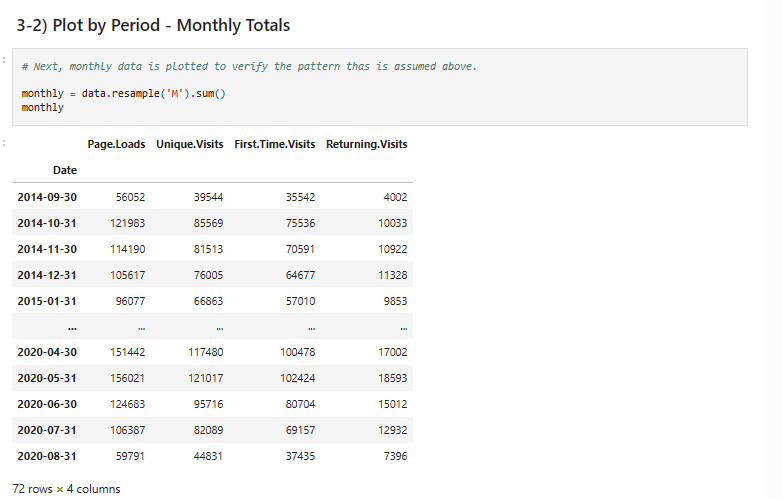
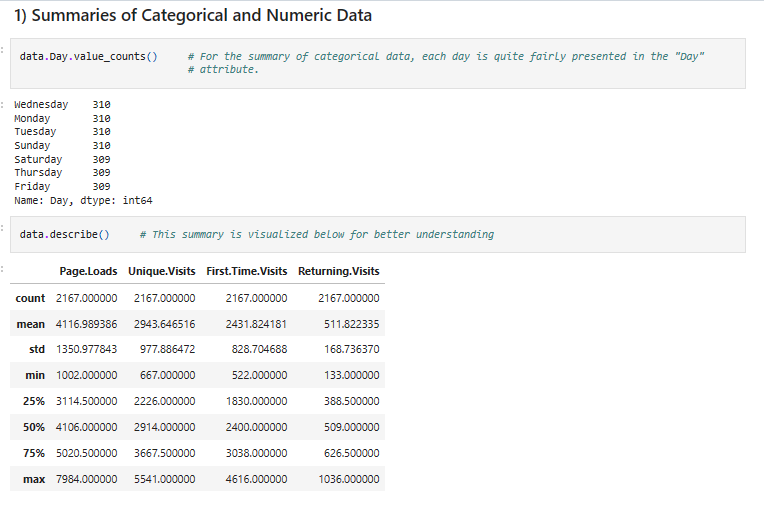


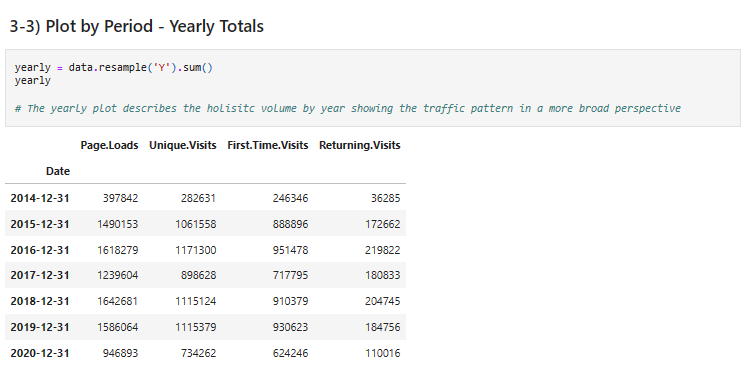
fig2 **=** model**.**plot\_components(forecast

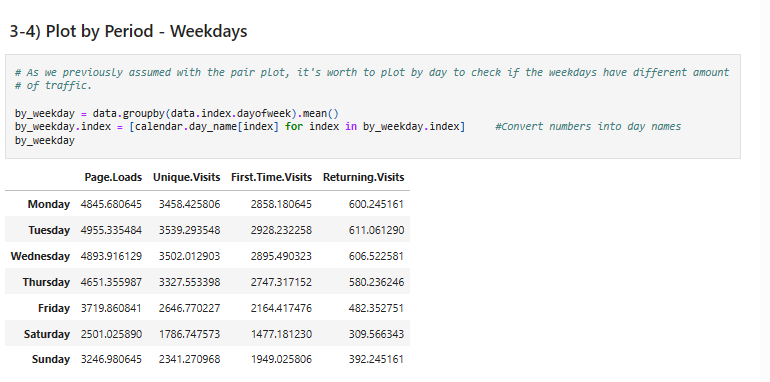
Use Python libraries like Pandas and Matplotlib to perform more complex analyses on the data, such as time series analysis, user segmentation, or machine learning-based predictions.



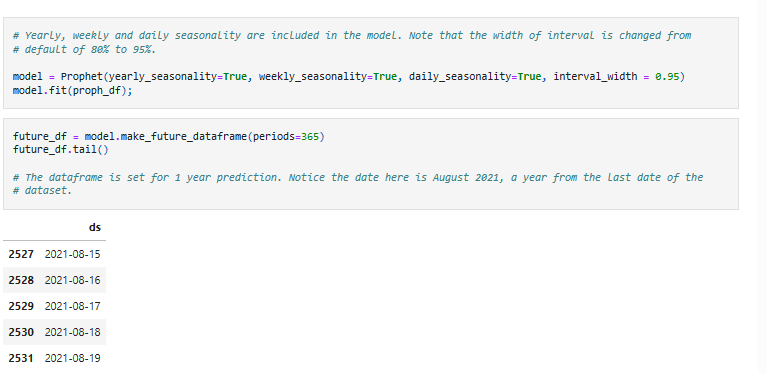












**Conclusion:**

In conclusion, the findings and recommendations contained within this report have the potential to drive positive change and meaningful progress in our [organization, project, or field]. We look forward to the collaborative efforts that will shape the future in line with the insights presented here.