**DATA ANALYTICS**

**WEBSITE TRAFFIC ANALYSIS WITH COGNOS ANALYTICS**

**P.Bakkiyam**

**420721104032**

**PHASE 5**

**WEBSITE TRAFFIC ANALYSIS**

**ABSTRACT:**

**In the real of modern business and digital presence, website traffic analysis stands as a crucial practice for gaining valuable insights into online user interactions. This analysis equips organizations with the knowledge needed to optimize their web strategies and enhance user experiences. This comprehensive project focuses on website traffic analysis using IBM Cognos Analytics, a robust platform for data visualization and reporting. We employ Cognos Analytics to create interactive dashboards and detailed reports that illuminate critical insights, including popular pages, traffic sources, and user engagement metrics. In addition, we seamlessly integrate Python for more advanced analytics, such as time series analysis, user segmentation, and machine learning-based predictions. This holistic approach empowers data-driven decision-making, allowing us to optimize web performance, enhance the user experience, and achieve our project objectives.**

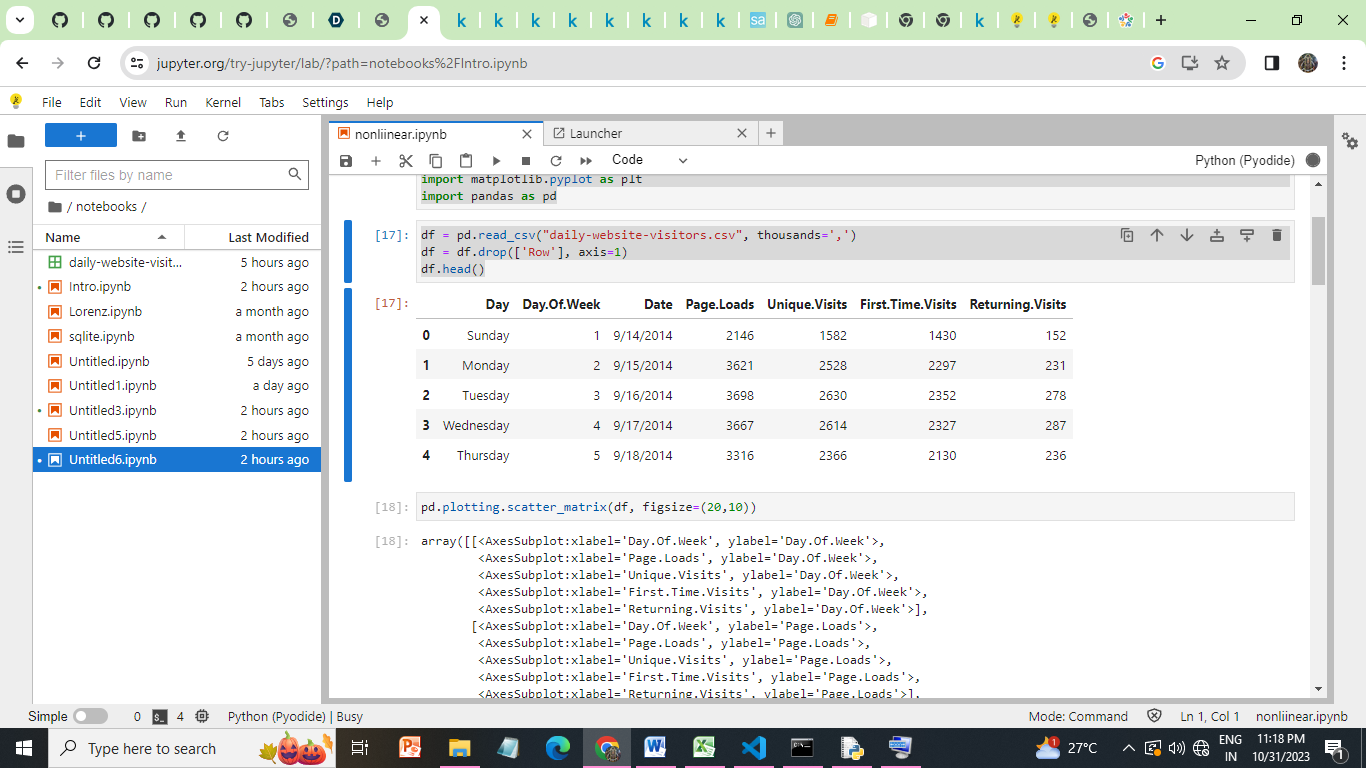
**INTRODUCTION:**

**In the context of our project, we are leveraging IBM Cognos Analytics to conduct website traffic analysis, a critical aspect of web performance assessment. By harnessing the capabilities of Cognos Analytics, we aim to create a dynamic and user-friendly interface, presenting vital insights regarding website traffic, user engagement, popular pages, and traffic sources. This project seeks to provide actionable data-driven insights, ultimately enabling more informed decision-making for optimizing web performance and enhancing the user experience.**

**DATA COLLECTION:**

**GIVEN DATASET:**

**LINK:**[**https://www.kaggle.com/datasets/bobnau/daily-website-visitors**](https://www.kaggle.com/datasets/bobnau/daily-website-visitors)



**TECHNOLOGIES STACK :**

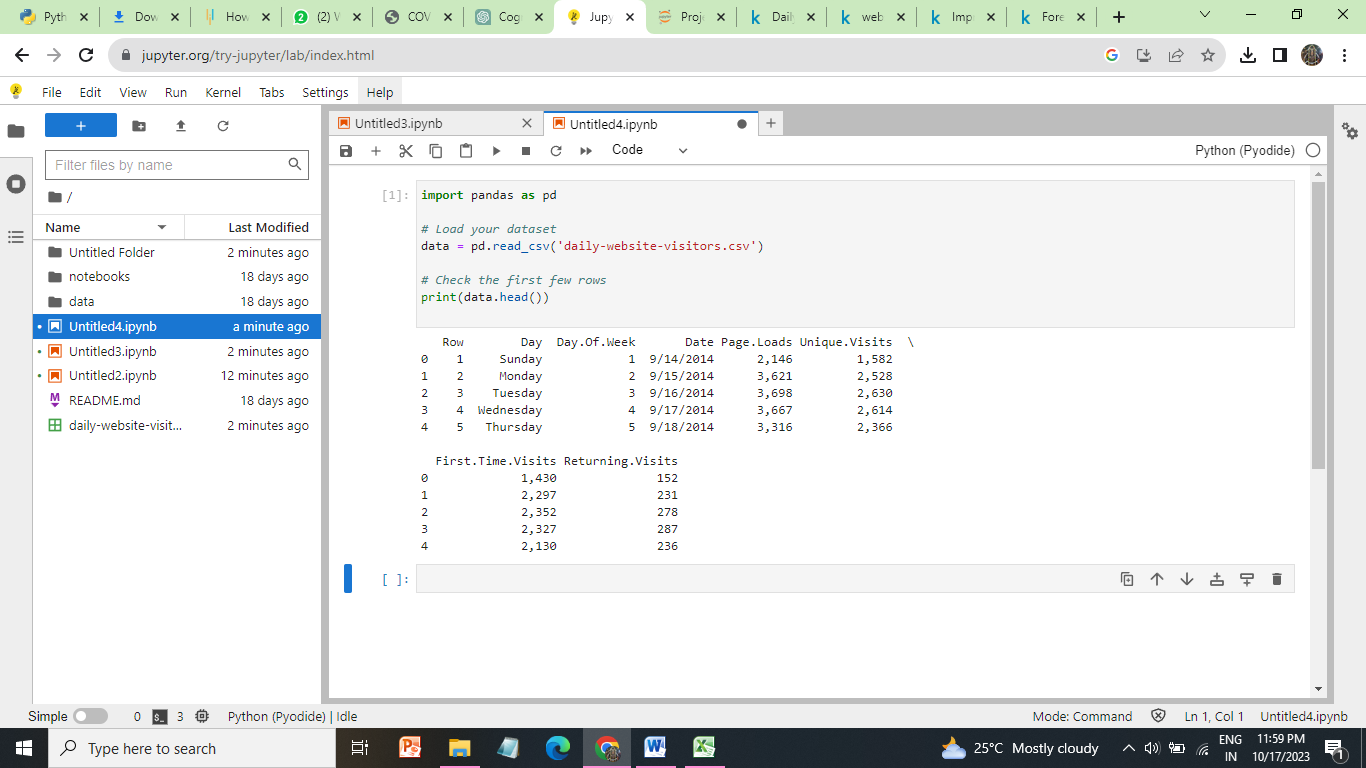
1. **IBM Cognos Analytics:** The core platform for data visualization and reporting, Cognos Analytics enables us to create interactive dashboards and detailed reports to present our findings effectively.
2. **Python**: We leverage Python for advanced data analysis and modeling. Key libraries like Pandas are used for data manipulation and preparation, while Matplotlib facilitates the creation of custom visualizations.
3. **Web Analytics Data:** This project relies on data generated by web analytics tools such as Google Analytics or Adobe Analytics, which capture and store user interactions on the website.
4. **Data Sources:** Data sources are essential for extracting, transforming, and loading data into our analysis environment. Common data sources include databases, data warehouses, spreadsheets, and flat files.
5. **Time Series Analysis Tools**: Time series analysis requires specialized tools and libraries for modeling and forecasting trends. We may use libraries like Statsmodels or Prophet.
6. **User Segmentation Tools:** For user segmentation, we might utilize clustering algorithms from libraries like Scikit-Learn to group users based on their behavior and characteristics.
7. **Machine Learning Libraries:** If we employ machine learning for predictive analytics, libraries such as Scikit-Learn or TensorFlow may be used to build and train models.
8. **Data Visualization Tools:** In addition to Matplotlib, data visualization tools like Seaborn or Plotly may be employed to create insightful charts and graphs.
9. **Data Preprocessing Tools:** Data preprocessing is a crucial step, and we use tools like Scikit-Learn or Pandas to clean and transform data.
10. **Report Generation:** IBM Cognos Analytics provides in-built report generation capabilities, enabling the creation of detailed reports for sharing insights with stakeholders.

**LOADING AND PREPROCESSING THE DATASET**

**DATA INGESTION:**

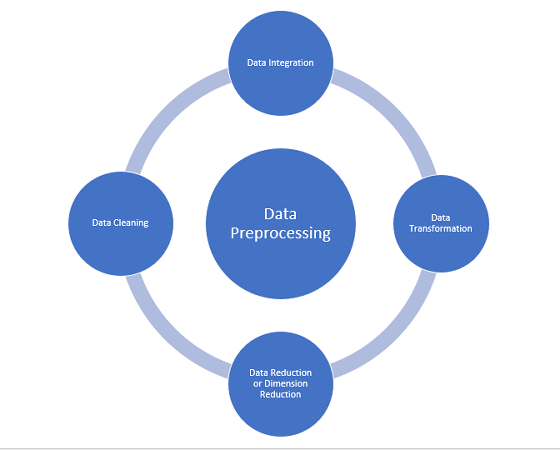
Data ingestion into Python Jupyter is a critical first step for preprocessing and cleansing website traffic data, enabling you to leverage Python's data manipulation libraries to prepare the data for analysis. Subsequently, using IBM Cognos for data visualization, you can create interactive dashboards that provide a comprehensive overview of key website performance metrics, empowering stakeholders to make data-driven decisions and effectively track progress toward business objectives, all within a seamless and integrated workflow.

**DATA INGESTING INTO PYTHON JUPYTER**

****

**DATA PREPROCESSING**

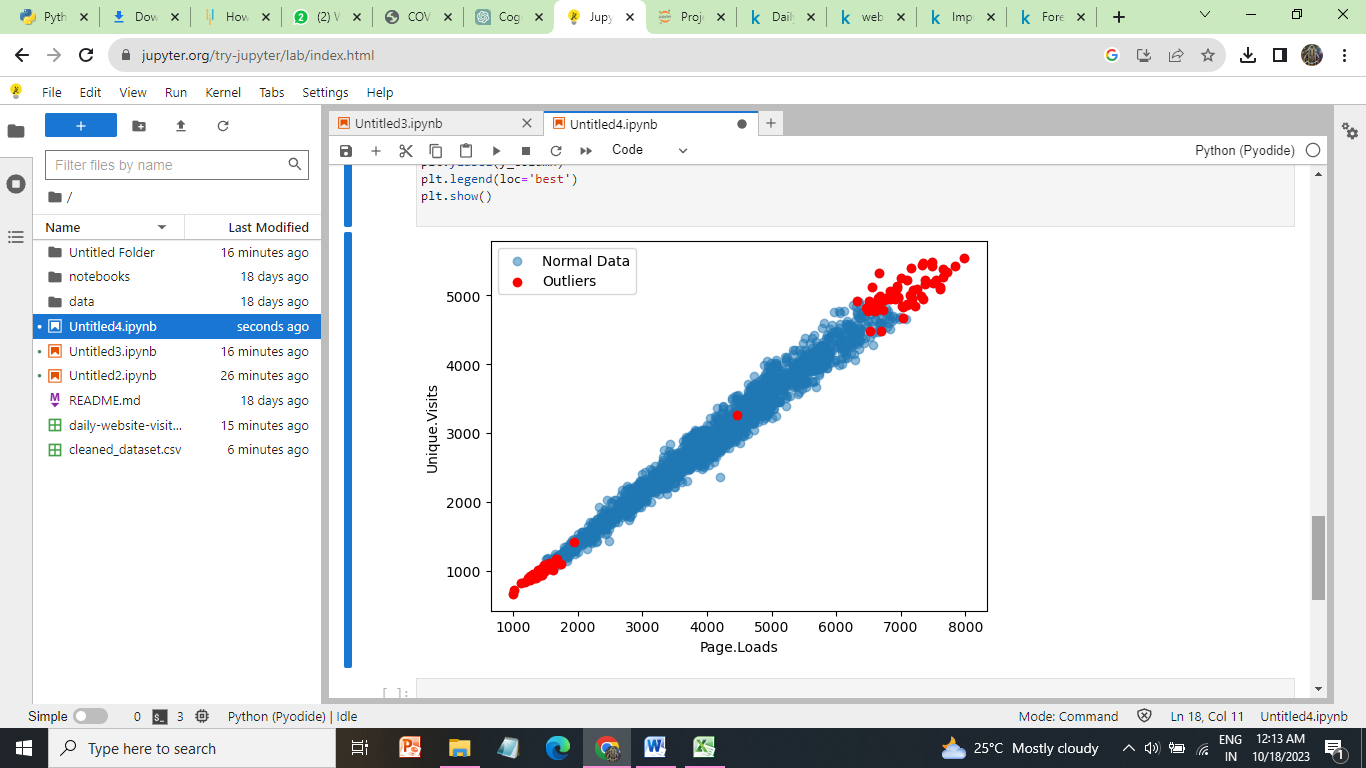
Data preprocessing is a step in the data mining and data analysis process that takes raw data and transforms it into a format that can be understood and analyzed by computers and machine learning.



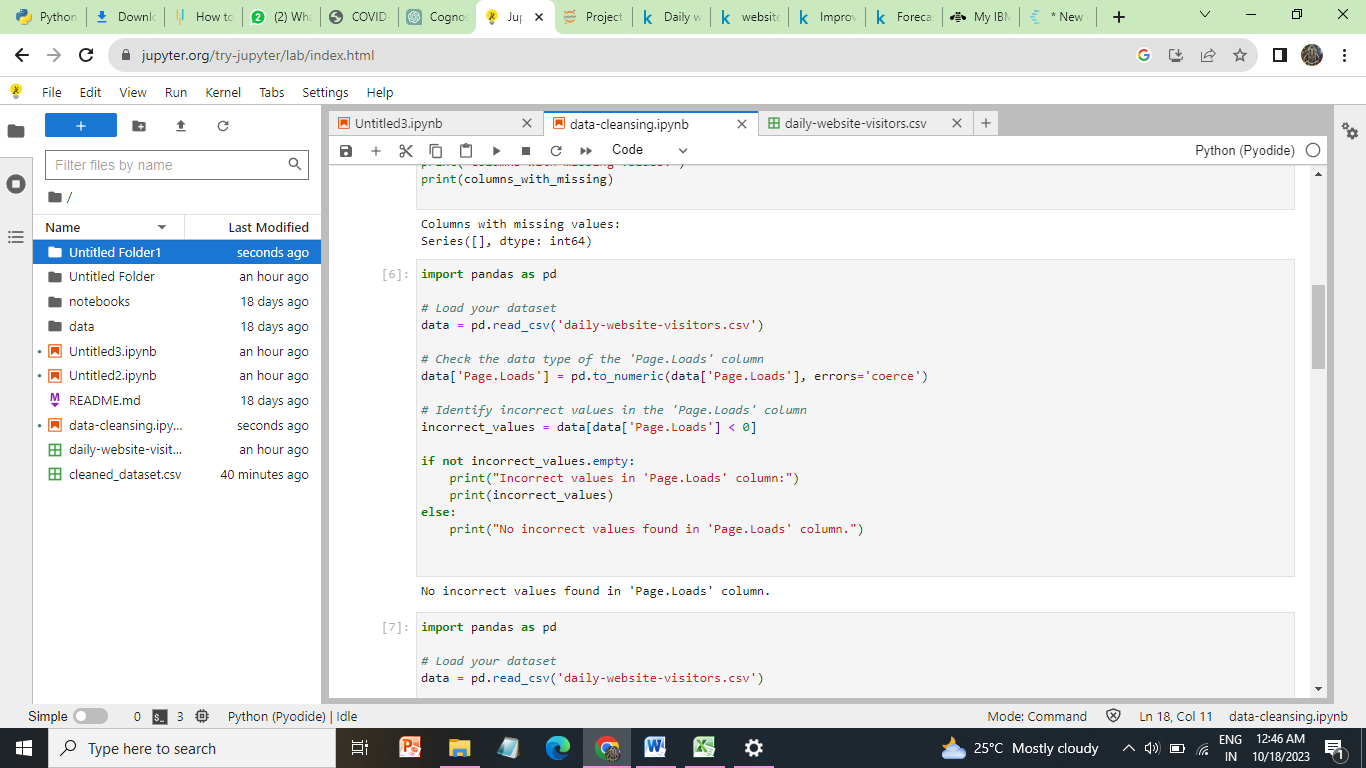
**DATA CLEANSING:**

[Data cleaning](https://monkeylearn.com/data-cleaning/) is the process of adding missing data and correcting, repairing, or removing incorrect or irrelevant data from a data set. Dating cleaning [is the most important step](https://monkeylearn.com/blog/data-cleaning-steps/) of preprocessing because it will ensure that your data is ready to go for your downstream needs.

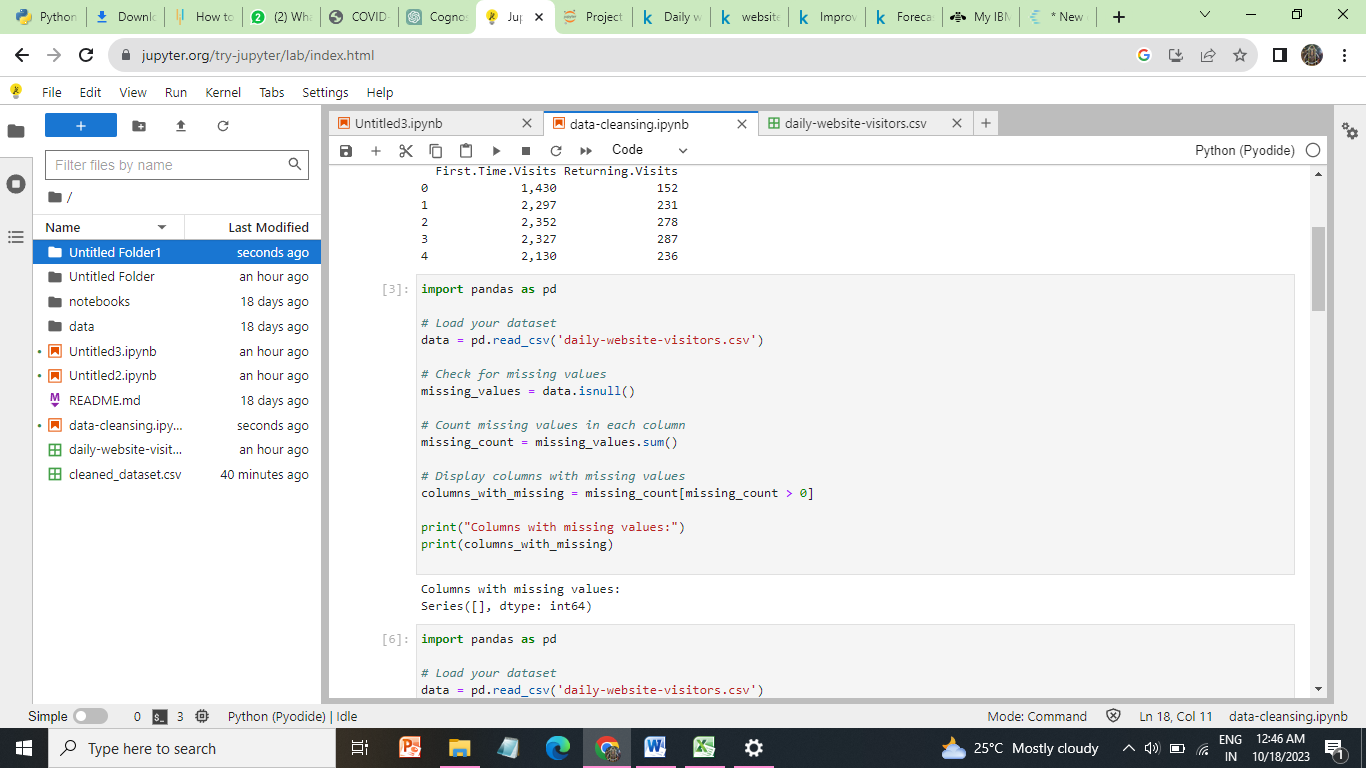
**OUTLIERS IN PAGELOADS:**

****

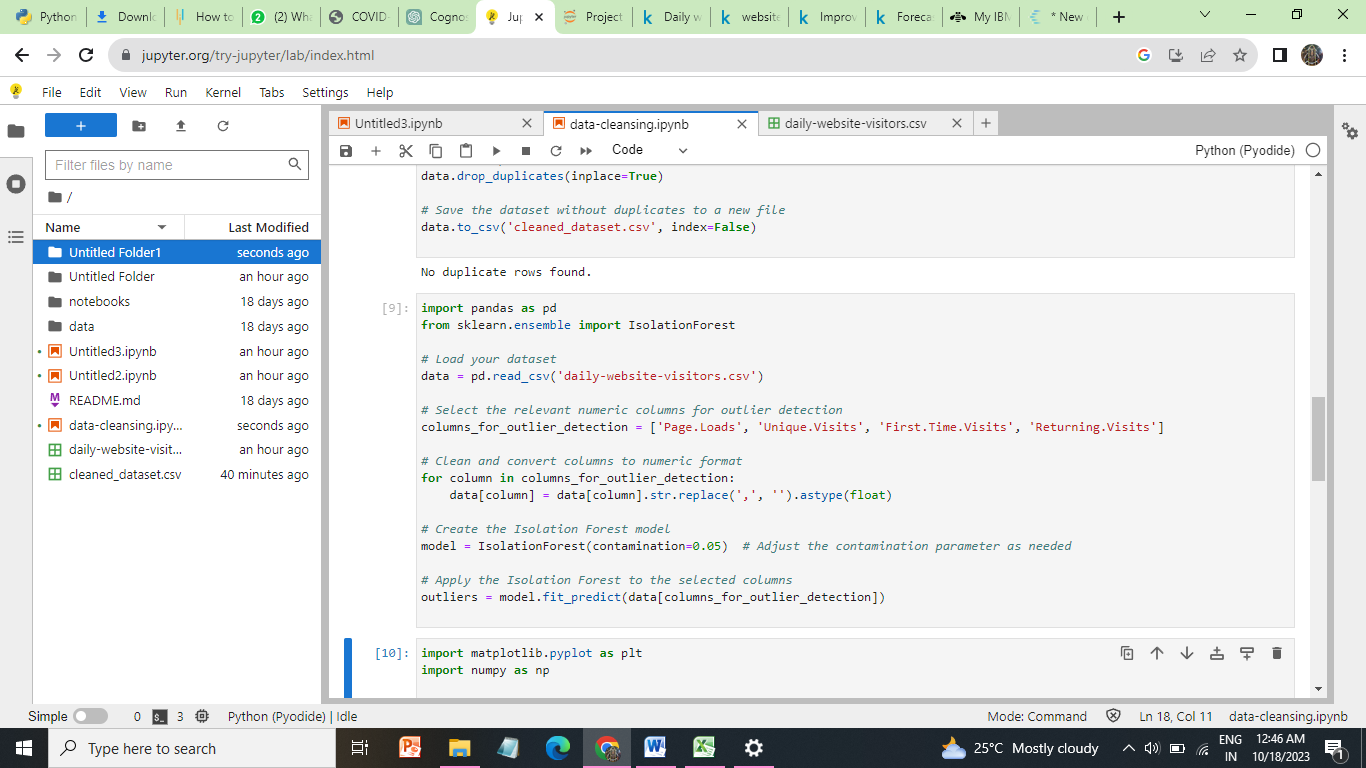
**INCORRECT AND MISSING VALUE:**

****

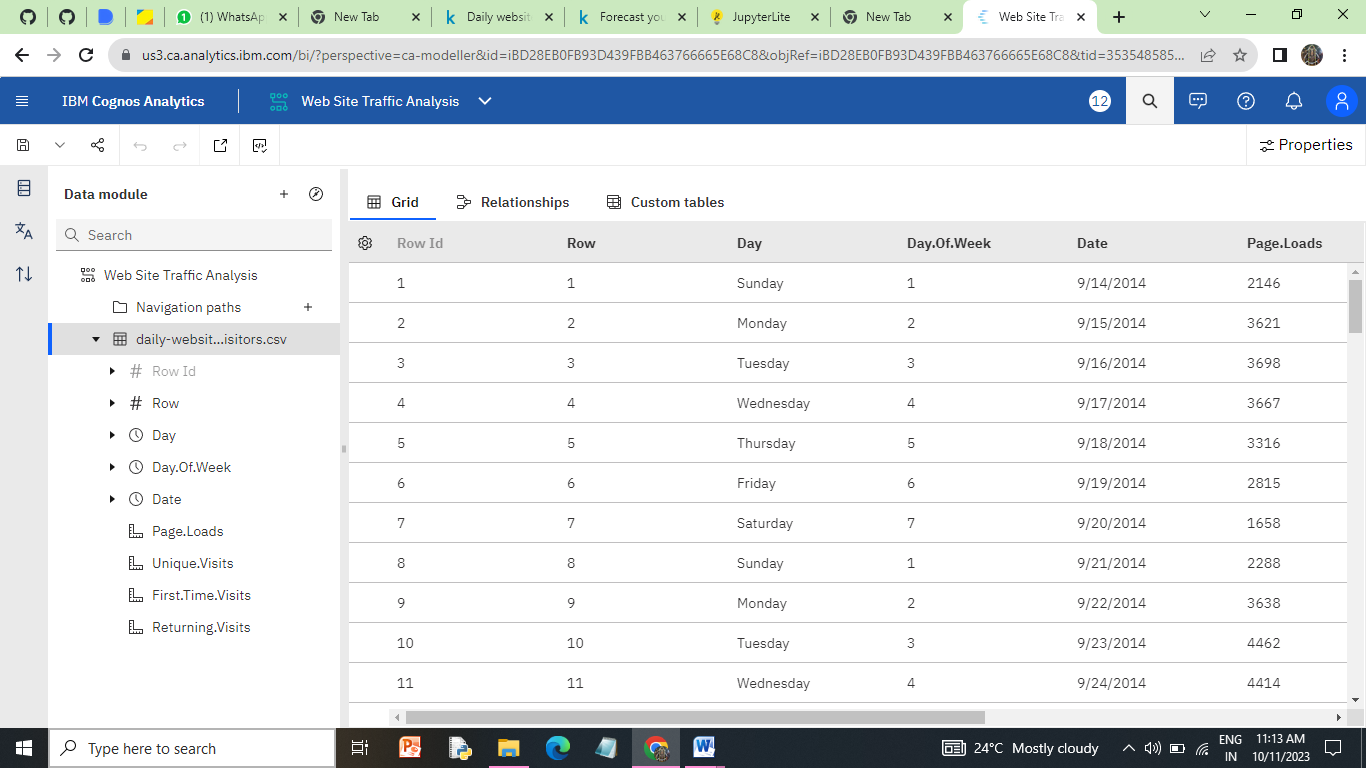
**MISSING VALUES**

****

**DUPLICATE VALUES:**

****

**DATA INGESTION IN IBM COGNOS:**

****

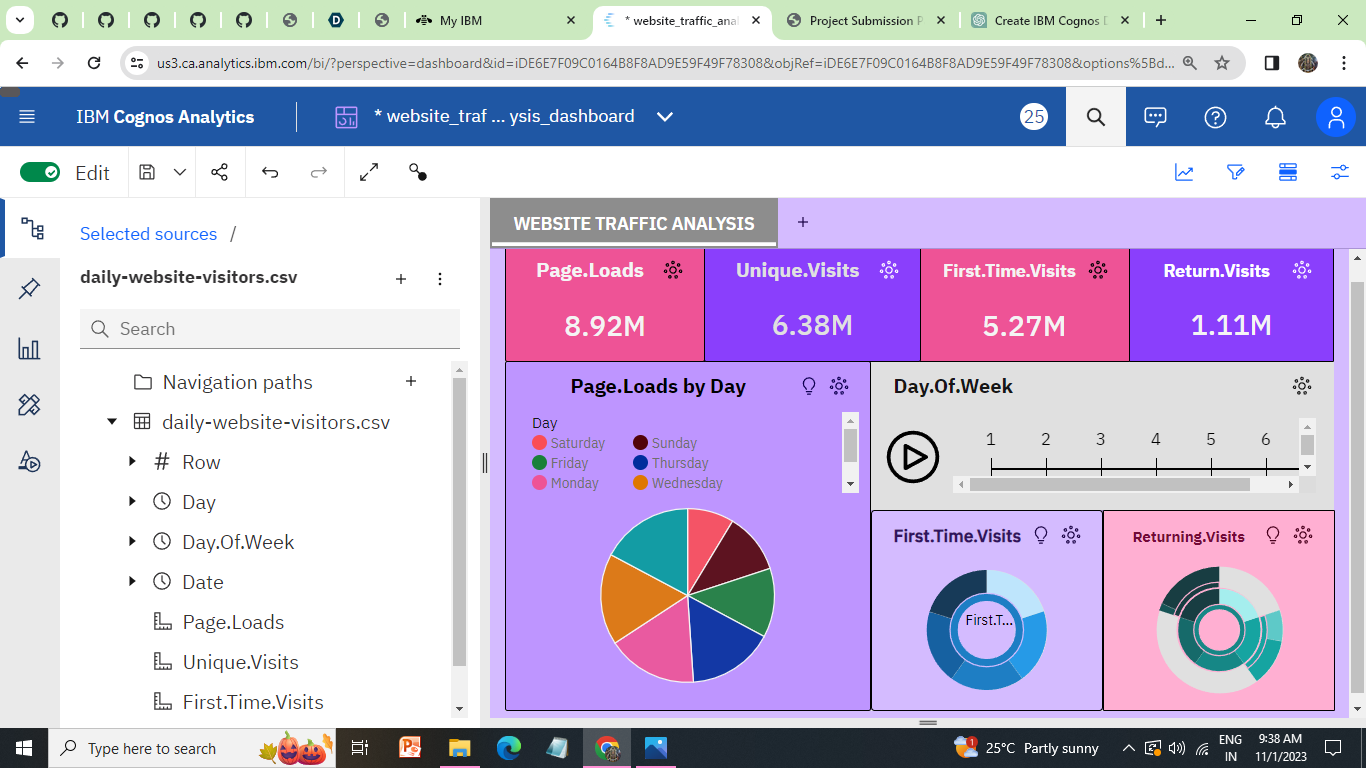
**WE HAVE ALSO ADDED THE JUPYTER NOTBOOK OF DATA PREPROCESSING IN OUR REPOSITORY.**

**IMPLEMENTATION:**

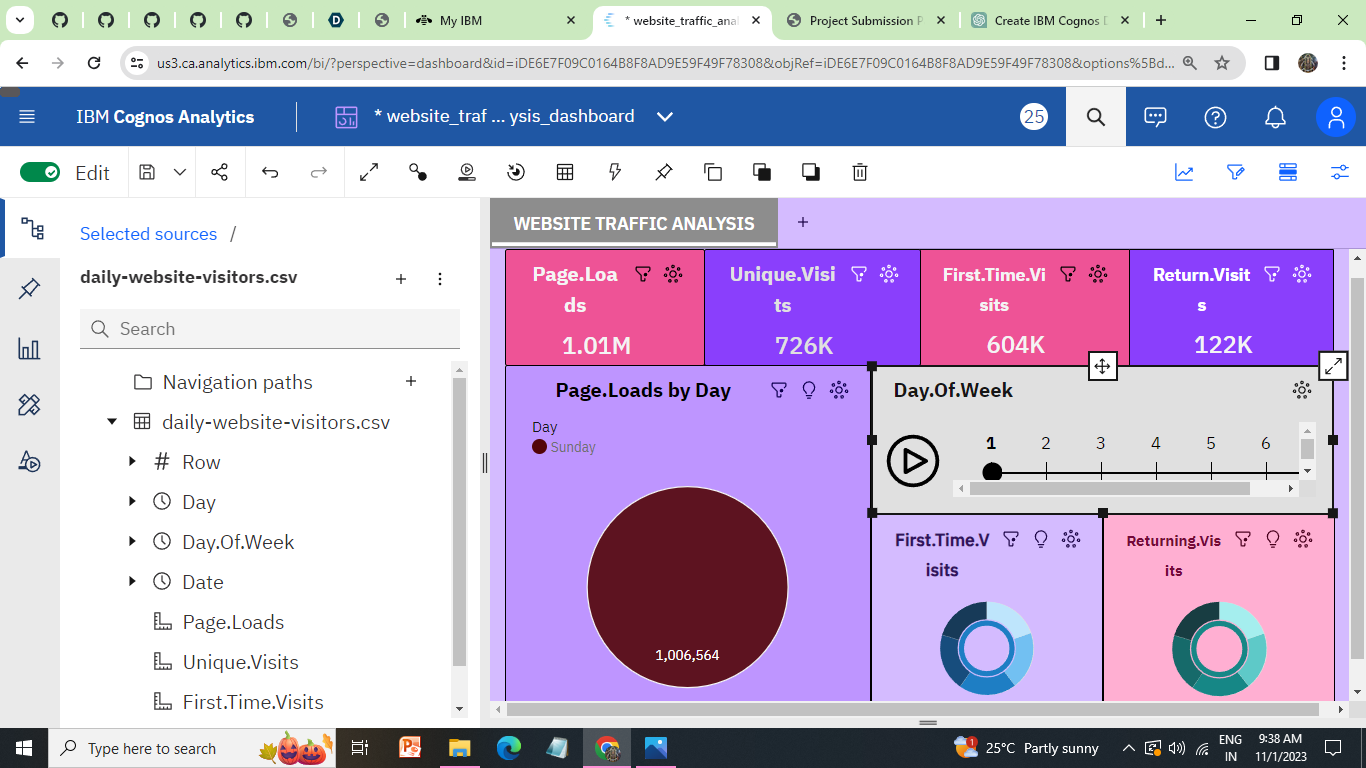
**Dashboard and Report Creation:**

In IBM Cognos, we can create interactive dashboards and detailed reports that offer a comprehensive view of our data. Dashboards provide at-a-glance insights with interactive visualizations, allowing users to monitor key metrics and drill down into details. Reports, on the other hand, offer in-depth analysis and documentation with tables, charts, and graphs, providing a more comprehensive understanding of our data. By combining both dashboards and reports, we can effectively cater to different user needs, from quick data exploration in dashboards to detailed analysis in reports, all within the same analytics platform.

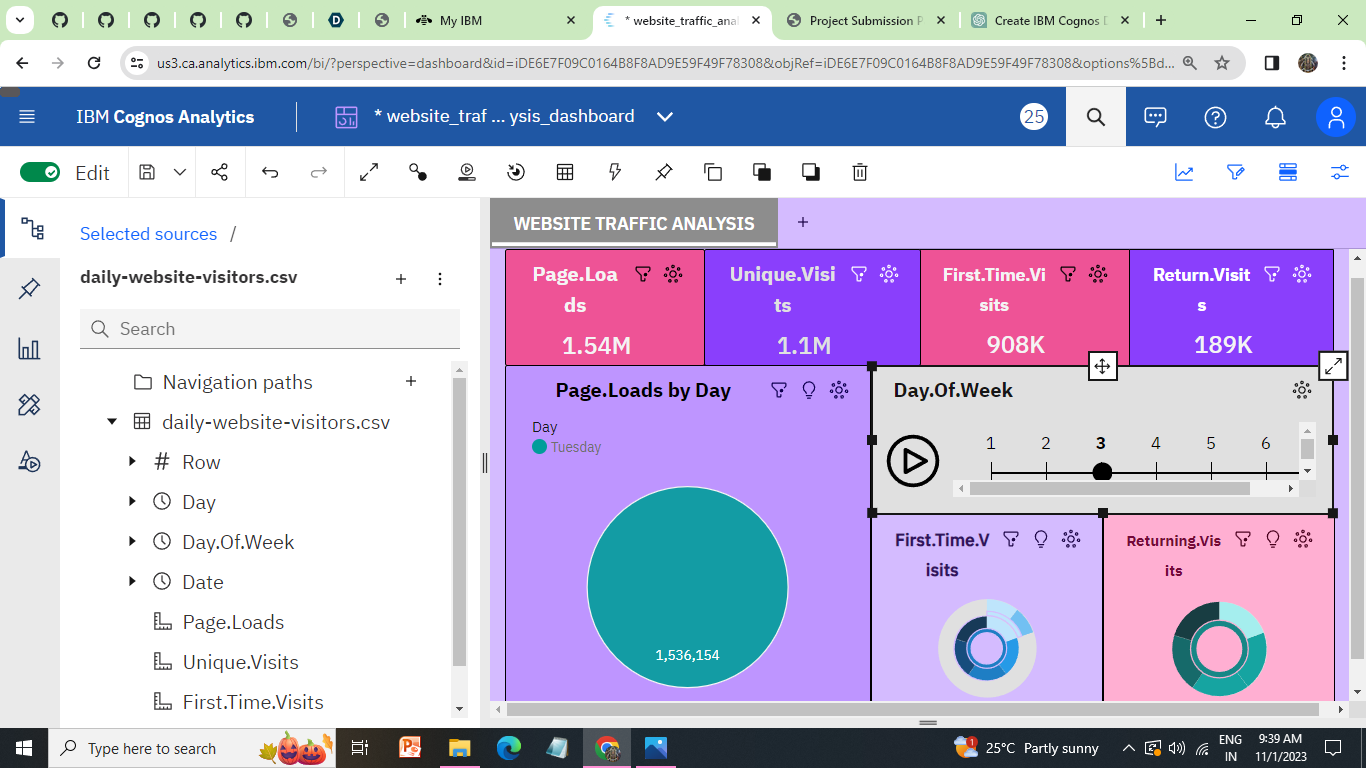
**Interactive Dashboard:**

****

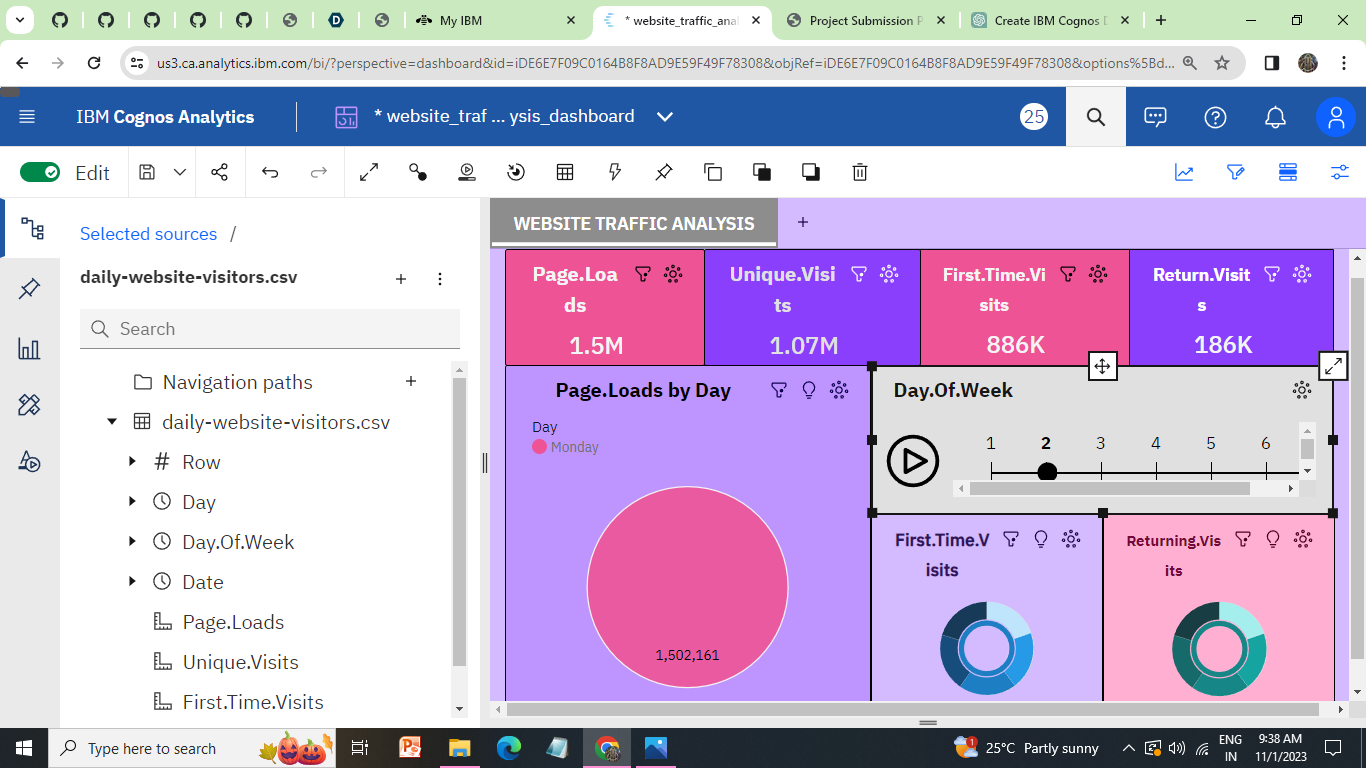
**SUNDAY**

****

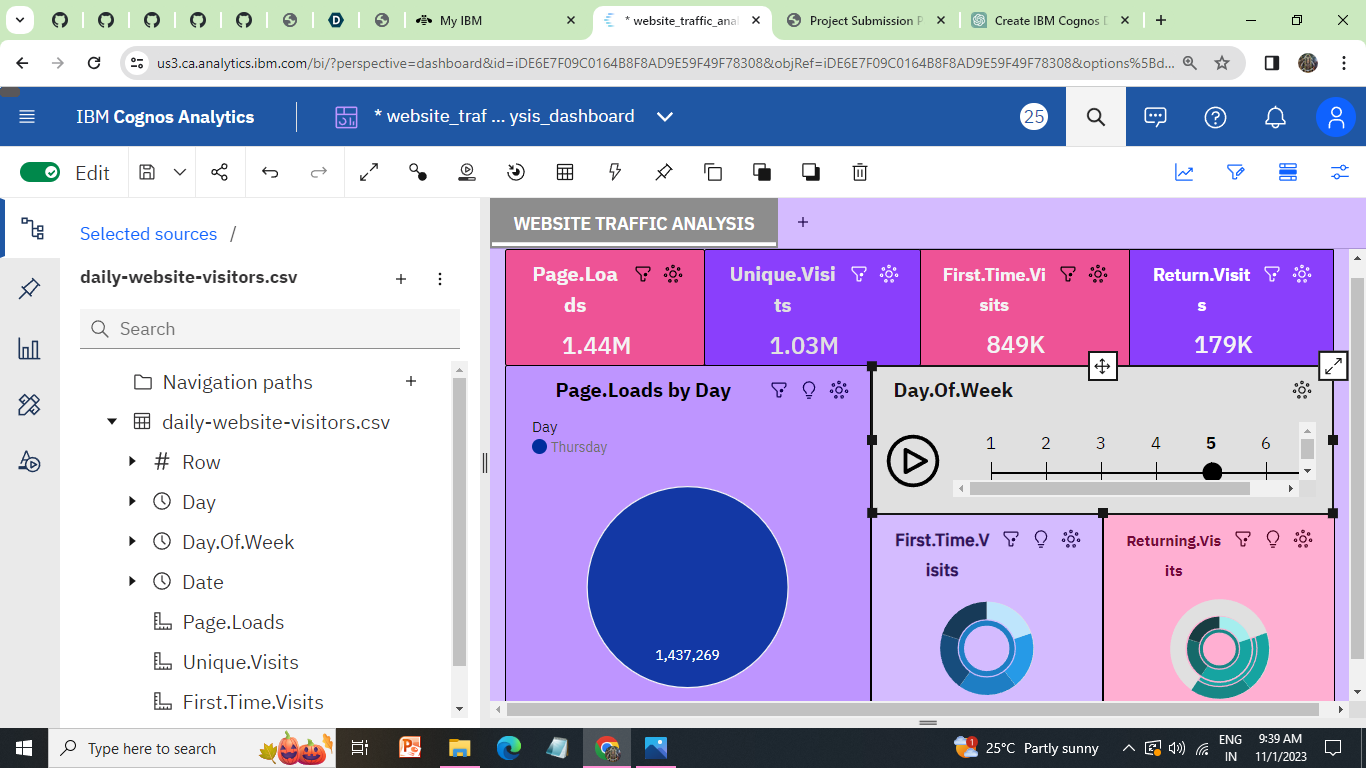
**MONDAY**

****

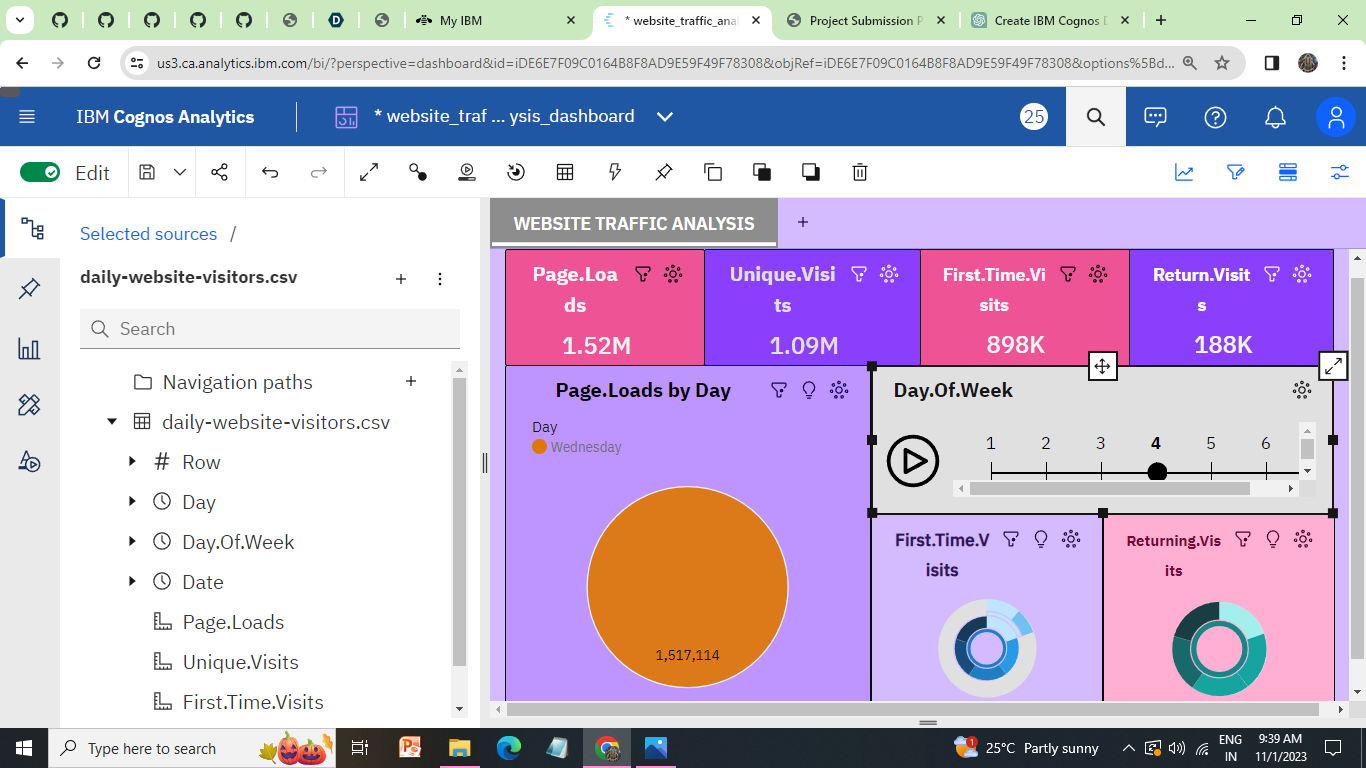
**TUESDAY**

****

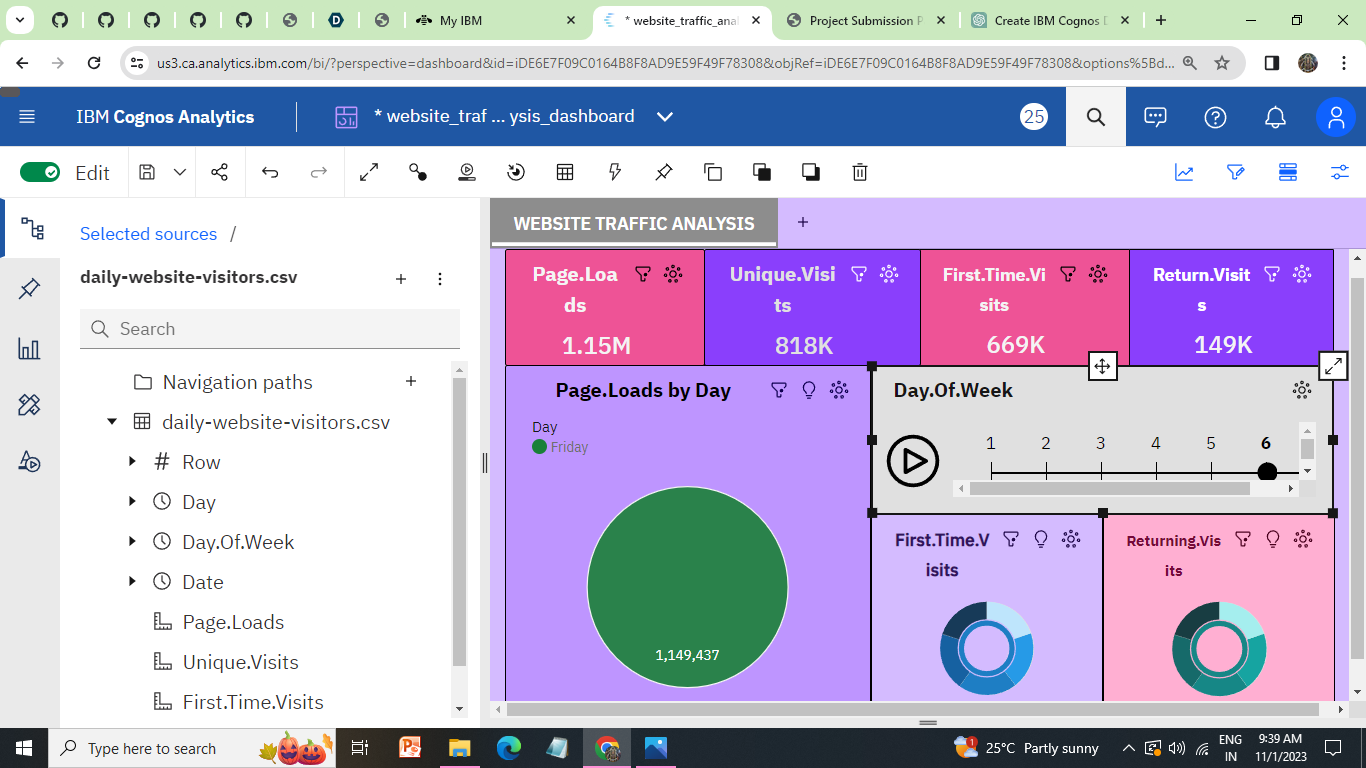
**WEDNESDAY**

****

**THURSDAY:**

****

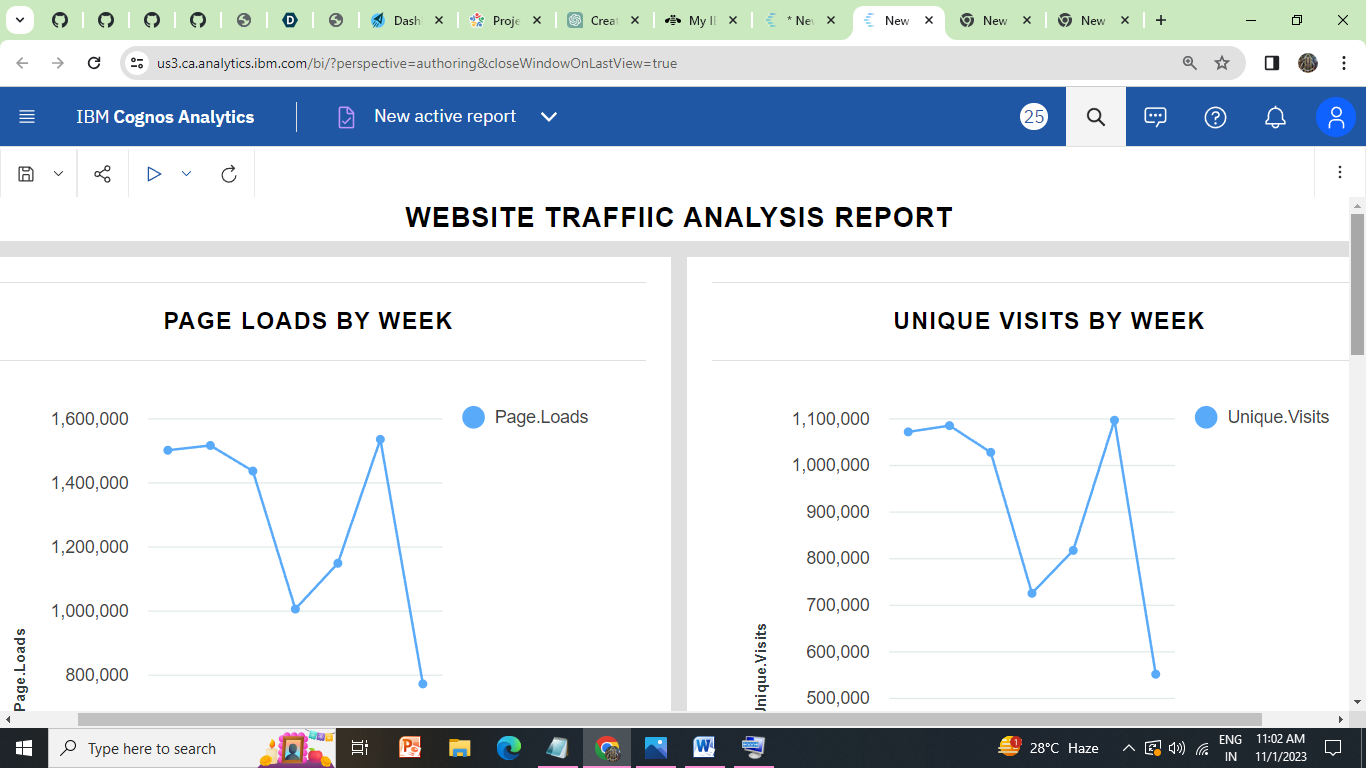
**FRIDAY:**

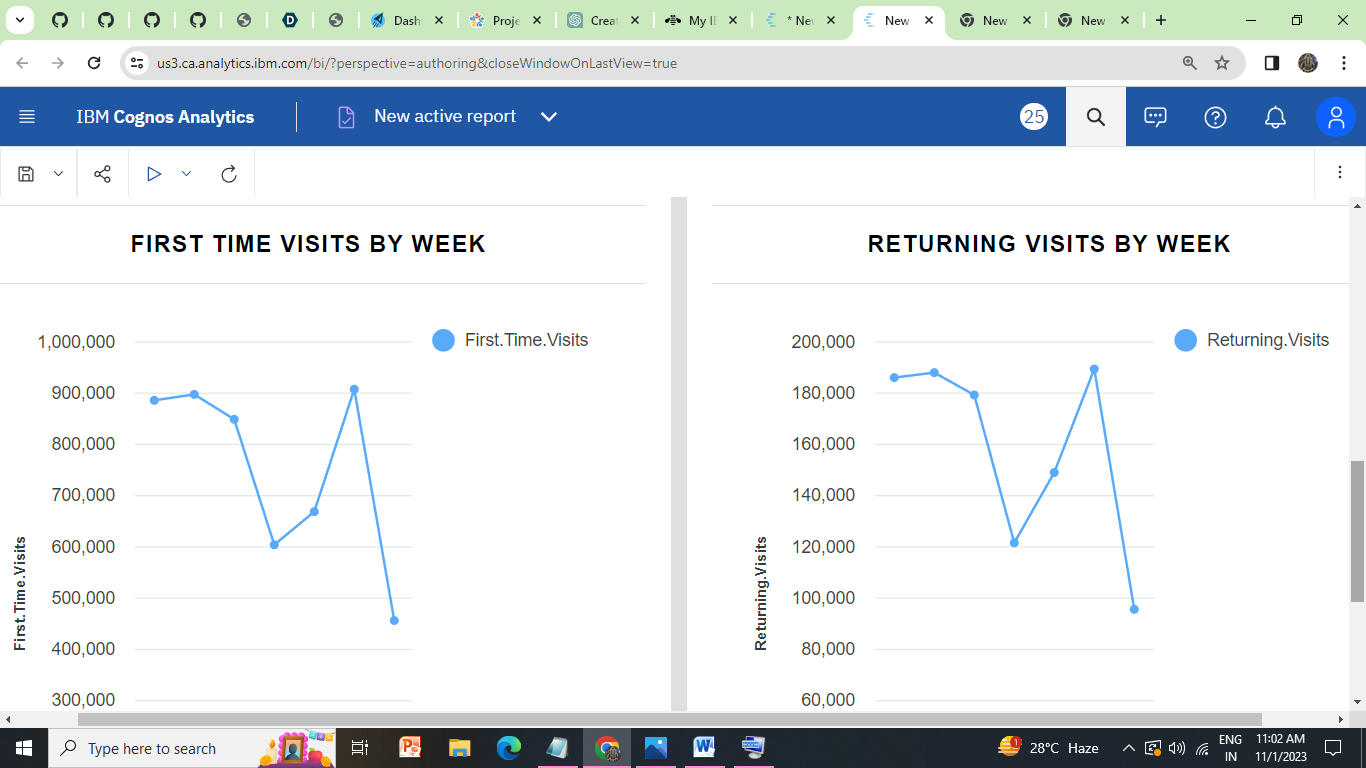
****

**SATURDAY**

****

**Report:**

****

****

**Analysis with python:**

Analysis with Python refers to the process of using the Python programming language and various data analysis libraries and tools to explore, manipulate, visualize, and derive insights from data. Python provides a versatile environment for tasks such as data cleaning, statistical analysis, machine learning, time series forecasting, and data visualization, making it a popular choice for data scientists, analysts, and researchers across various domains. It allows users to uncover patterns, trends, and relationships within datasets, enabling data-driven decision-making and problem-solving.

**Types :**

1. Linear and non linear relationship
2. Linear Regression
3. User\_segmentation Analysis
4. Time\_series\_Decompostion

**1.Linear and non linear relationship**

Analyzing linear relationships can help us understand how an increase in marketing spending correlates with linear growth in website traffic. Non-linear relationships can be used to understand complex interactions, such as how website load times affect bounce rates in a non-linear way.

**#Import Libraries**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.ensemble import RandomForestRegressor**

**from sklearn.metrics import mean\_squared\_error**

**from sklearn.cluster import KMeans**

**from sklearn import neighbors**

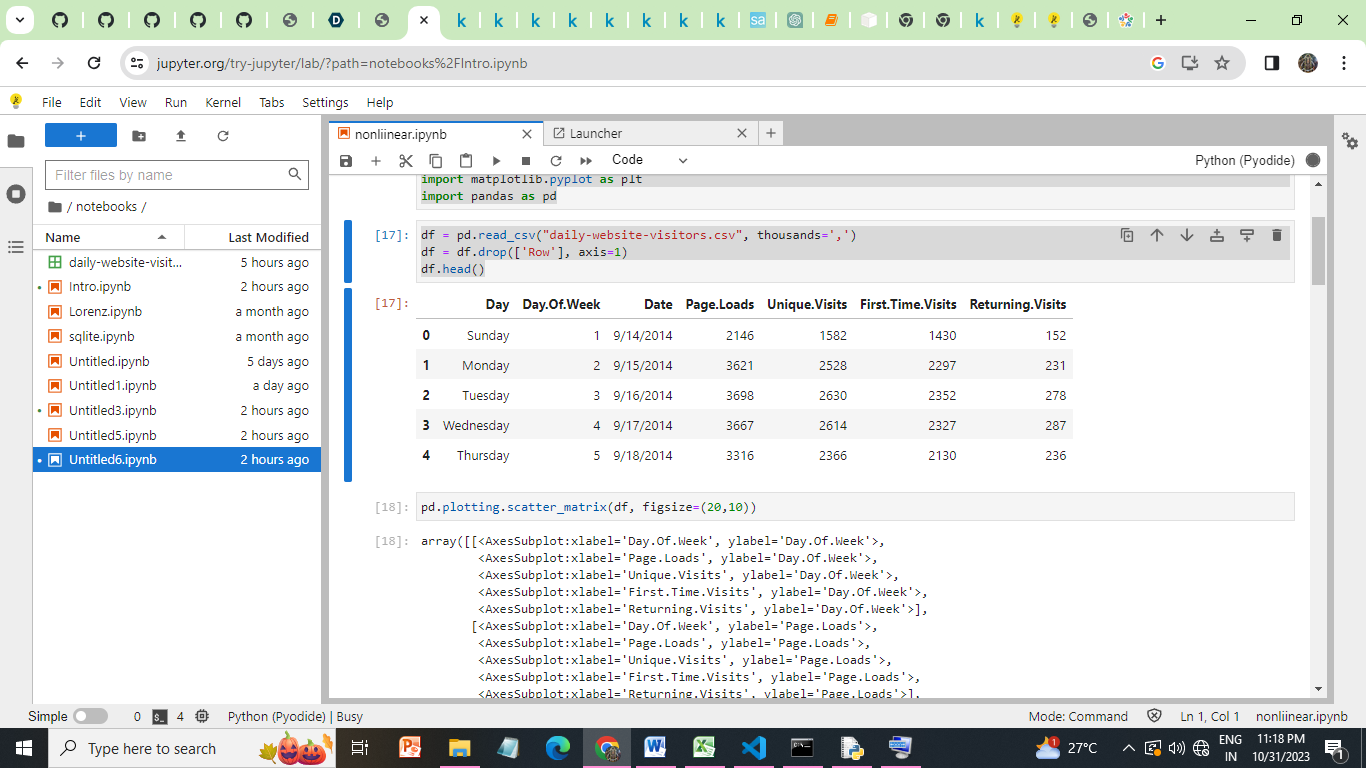
**#load the Dataset**

**df = pd.read\_csv("daily-website-visitors.csv", thousands=',')**

**df = df.drop(['Row'], axis=1)**

**df.head()**

**output:**

****

**pd.plotting.scatter\_matrix(df, figsize=(20,10))**

array([[<AxesSubplot:xlabel='Day.Of.Week', ylabel='Day.Of.Week'>,

<AxesSubplot:xlabel='Page.Loads', ylabel='Day.Of.Week'>,

<AxesSubplot:xlabel='Unique.Visits', ylabel='Day.Of.Week'>,

<AxesSubplot:xlabel='First.Time.Visits', ylabel='Day.Of.Week'>,

<AxesSubplot:xlabel='Returning.Visits', ylabel='Day.Of.Week'>],

[<AxesSubplot:xlabel='Day.Of.Week', ylabel='Page.Loads'>,

<AxesSubplot:xlabel='Page.Loads', ylabel='Page.Loads'>,

<AxesSubplot:xlabel='Unique.Visits', ylabel='Page.Loads'>,

<AxesSubplot:xlabel='First.Time.Visits', ylabel='Page.Loads'>,

<AxesSubplot:xlabel='Returning.Visits', ylabel='Page.Loads'>],

[<AxesSubplot:xlabel='Day.Of.Week', ylabel='Unique.Visits'>,

<AxesSubplot:xlabel='Page.Loads', ylabel='Unique.Visits'>,

<AxesSubplot:xlabel='Unique.Visits', ylabel='Unique.Visits'>,

<AxesSubplot:xlabel='First.Time.Visits', ylabel='Unique.Visits'>,

<AxesSubplot:xlabel='Returning.Visits', ylabel='Unique.Visits'>],

[<AxesSubplot:xlabel='Day.Of.Week', ylabel='First.Time.Visits'>,

<AxesSubplot:xlabel='Page.Loads', ylabel='First.Time.Visits'>,

<AxesSubplot:xlabel='Unique.Visits', ylabel='First.Time.Visits'>,

<AxesSubplot:xlabel='First.Time.Visits', ylabel='First.Time.Visits'>,

<AxesSubplot:xlabel='Returning.Visits', ylabel='First.Time.Visits'>],

[<AxesSubplot:xlabel='Day.Of.Week', ylabel='Returning.Visits'>,

<AxesSubplot:xlabel='Page.Loads', ylabel='Returning.Visits'>,

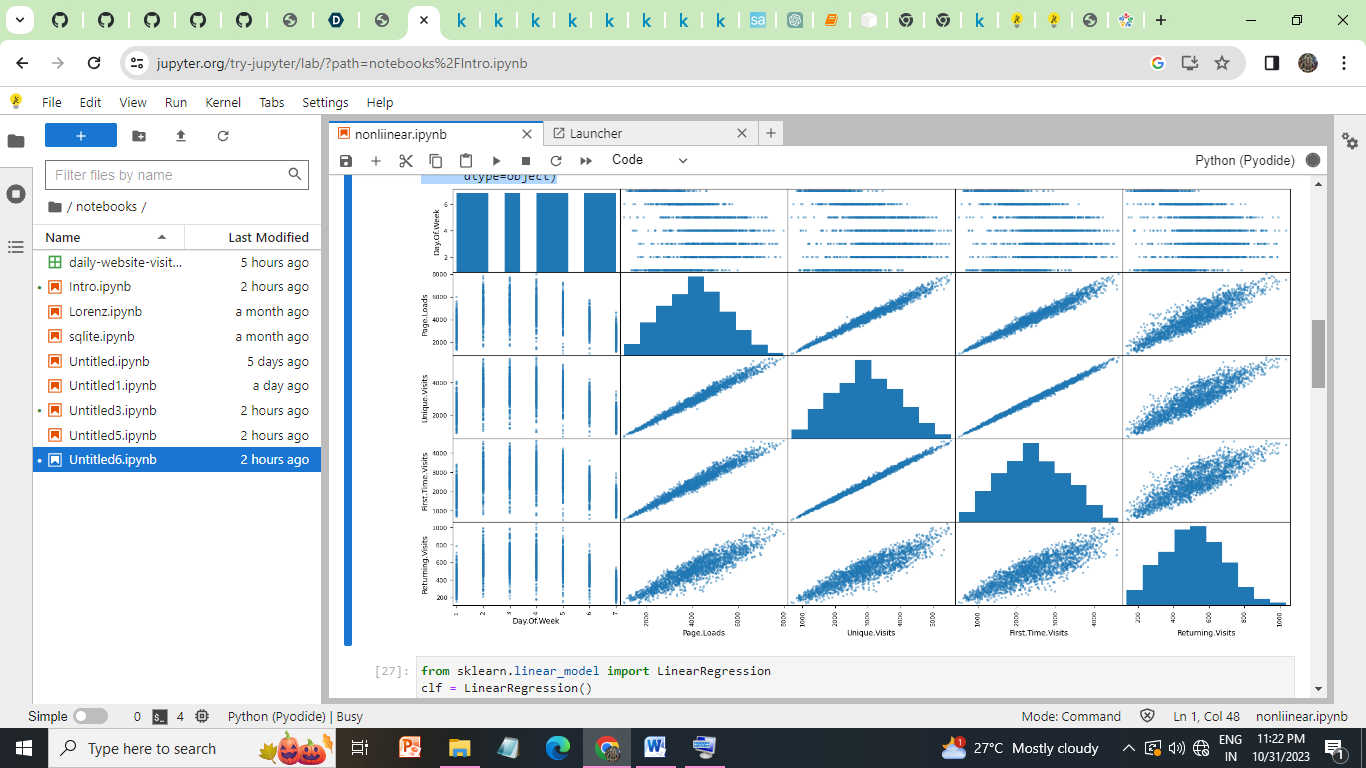
<AxesSubplot:xlabel='Unique.Visits', ylabel='Returning.Visits'>,

<AxesSubplot:xlabel='First.Time.Visits', ylabel='Returning.Visits'>,

<AxesSubplot:xlabel='Returning.Visits', ylabel='Returning.Visits'>]],

dtype=object)

**OUTPUT:**

****

**2.LINEAR REGRESSION:**

Linear regression is a statistical technique that can be used to model and analyze the relationship between a dependent variable (e.g., website traffic) and one or more independent variables (e.g., advertising spend, time spent on the website). This can help in predicting future trends and making data-driven decisions in website development.

**CODE:**

**from sklearn.linear\_model import LinearRegression**

**clf = LinearRegression()**

**clf.fit(x\_train,y\_train)**

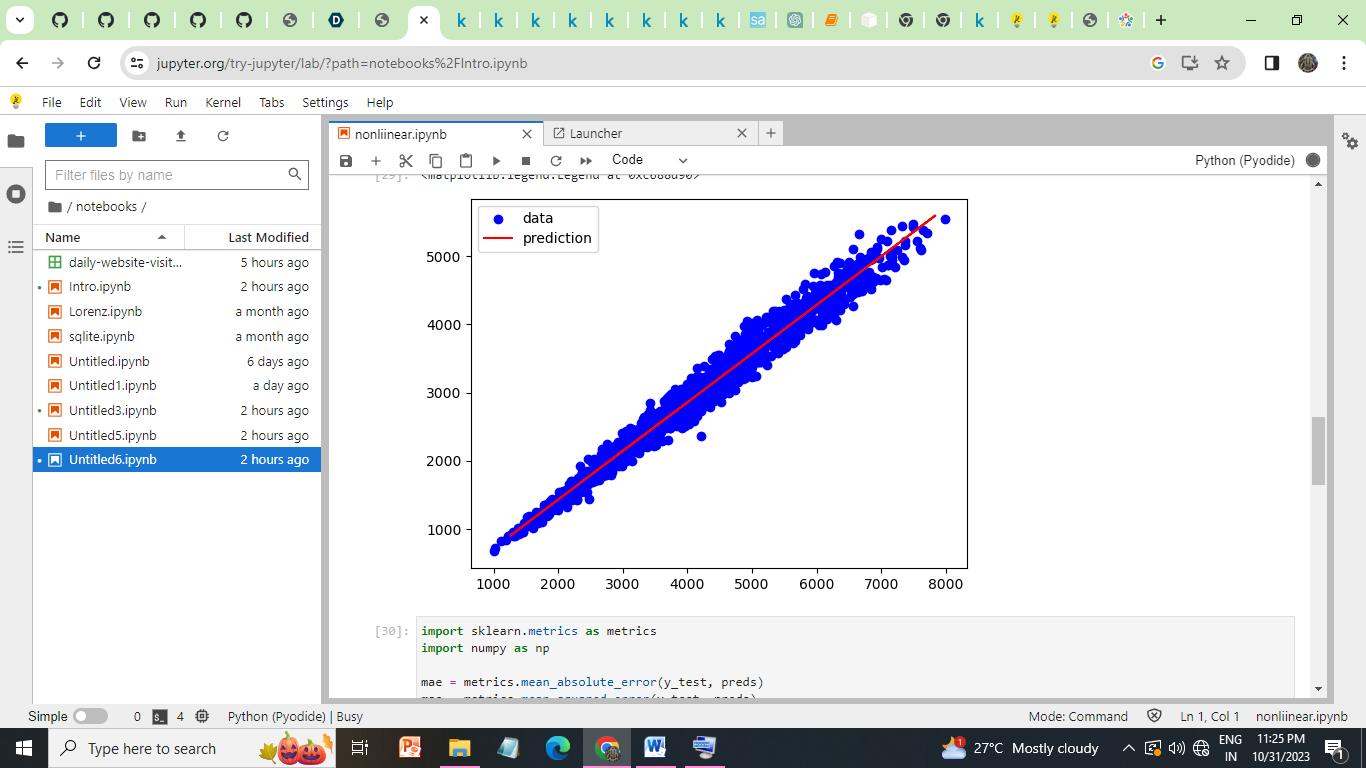
**preds = clf.predict(x\_test)**

**plt.scatter(x\_train, y\_train, color="blue", label="data")**

**plt.plot(x\_test, preds, color="red", label="prediction")**

**plt.legend()**

**OUTPUT**

**: **

**METRICS FOR LINEAR REGRESSION:**

**import sklearn.metrics as metrics**

**import numpy as np**

**mae = metrics.mean\_absolute\_error(y\_test, preds)**

**mse = metrics.mean\_squared\_error(y\_test, preds)**

**rmse = np.sqrt(mse) # or mse\*\*(0.5)**

**r2 = metrics.r2\_score(y\_test,preds)**

**print("Results of Linear Regression:")**

**print("MAE:",mae)**

**print("MSE:", mse)**

**print("RMSE:", rmse)**

**print("R-Squared:", r2)**

**OUTPUT:**

Results of Linear Regression:

MAE: 113.01405534987238

MSE: 20710.290923291977

RMSE: 143.91070468624625

R-Squared: 0.977890439598449

**#TimeSeriesForPageLoads**

**import matplotlib.pyplot as plt**

**df['Date'] = pd.to\_datetime(df['Date']) # Convert Date to a datetime object**

**plt.figure(figsize=(12, 6))**

**plt.plot(df['Date'], df['Page.Loads'], label='Page Loads')**

**plt.xlabel('Date')**

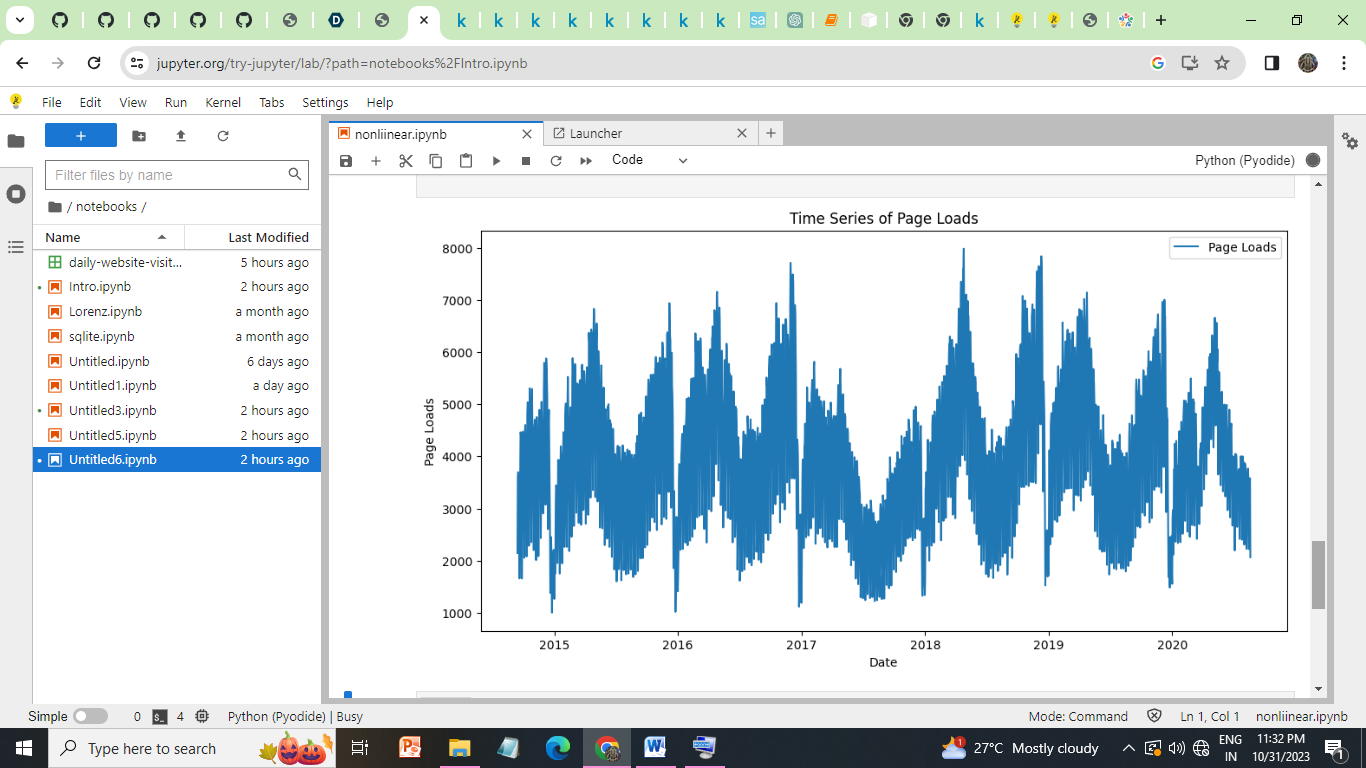
**plt.ylabel('Page Loads')**

**plt.title('Time Series of Page Loads')**

**plt.legend()**

**plt.show()**

**Output:**

****

**3.USER SEGMENTATION ANALYSIS:**

User segmentation involves categorizing website visitors into different groups based on shared characteristics or behaviors. Python can be used to analyze user data and segment users for personalized content, marketing campaigns, or user experience improvements. For example,we can segment users by location, demographics, or behavior (e.g., frequent visitors, first-time visitors) and tailor our website accordingly.

**CODE:**

**import seaborn as sns**

**from sklearn.preprocessing import StandardScaler**

**from scipy.cluster.hierarchy import dendrogram, linkage, fcluster**

**from scipy.spatial.distance import pdist**

**# Load our dataset into a DataFrame**

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

**# Data Preprocessing**

**selected\_features = ['Page.Loads', 'Unique.Visits', 'First.Time.Visits', 'Returning.Visits']**

**# Remove any commas and convert columns to float**

**for feature in selected\_features:**

**df[feature] = df[feature].str.replace(',', '').astype(float)**

**# Standardize the selected features**

**scaler = StandardScaler()**

**X = scaler.fit\_transform(df[selected\_features])**

**# Calculate the linkage matrix for hierarchical clustering**

**linkage\_matrix = linkage(X, method='ward', metric='euclidean')**

**# Determine the optimal number of clusters using the dendrogram**

**plt.figure(figsize=(12, 6))**

**dendrogram(linkage\_matrix)**

**plt.title('User Segmentation Analysis')**

**plt.xlabel('users')**

**plt.ylabel('Distance')**

**plt.show()**

**# Based on the dendrogram, choose the optimal number of clusters (e.g., 3)**

**num\_clusters = 3**

**clusters = fcluster(linkage\_matrix, t=num\_clusters, criterion='maxclust')**

**# Add the cluster labels to the DataFrame**

**df['Cluster'] = clusters**

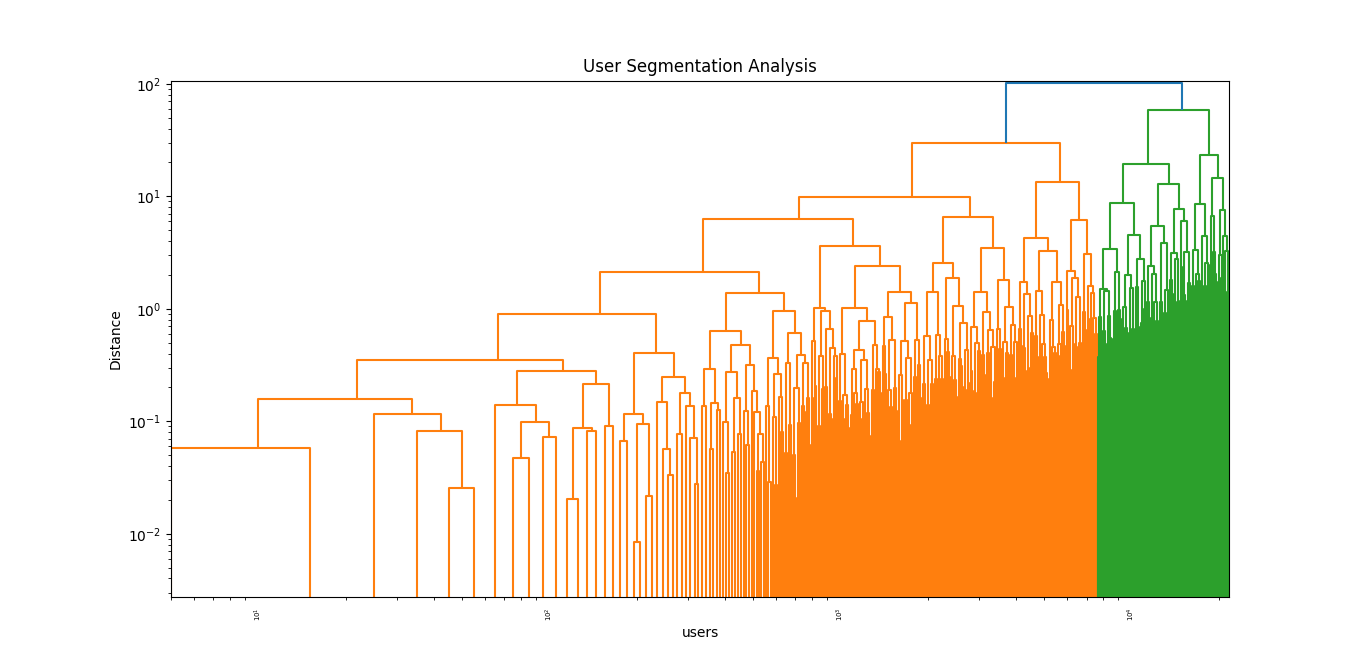
**# Visualize the user segments**

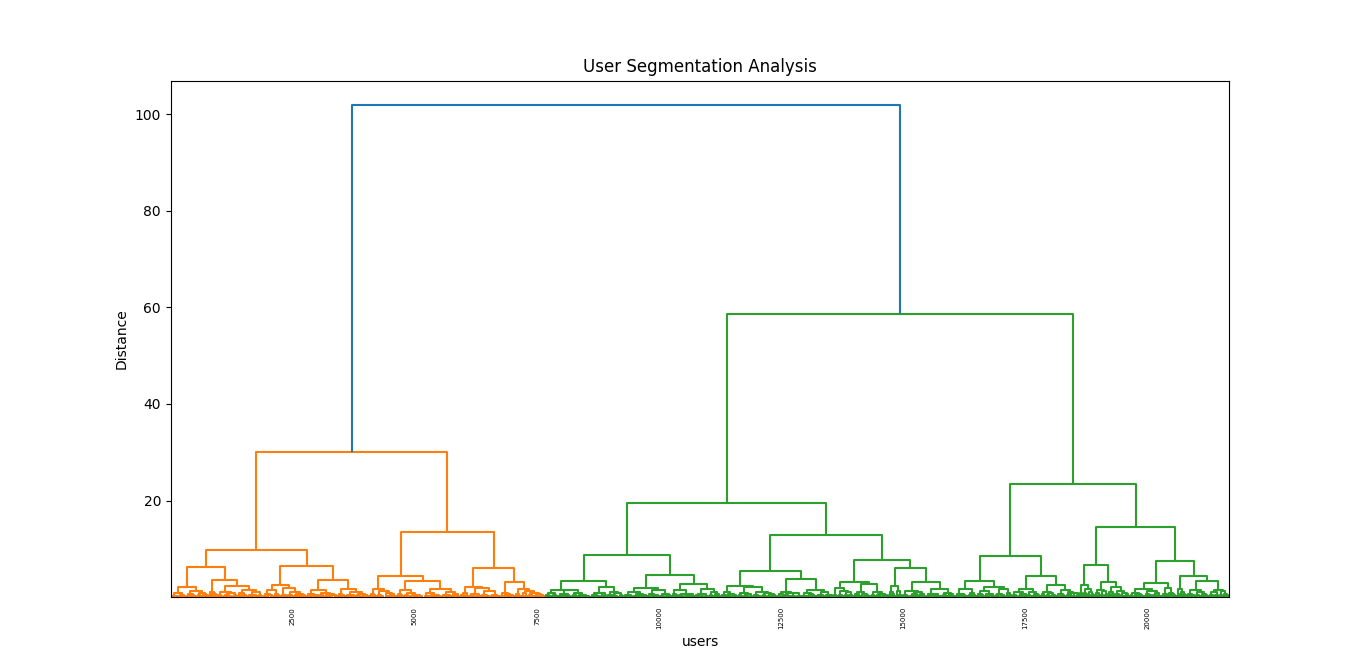
**sns.pairplot(df, hue='Cluster', vars=selected\_features, diag\_kind='kde')**

**plt.suptitle('User Segmentation Analysis')**

**plt.show()**

**OUTPUT:**

****

****

**4.TIME SERIES DECOMPOSITION:**

Time series decomposition is a technique used to break down time-based data into its constituent components, typically including trend, seasonality, and noise. In website development, we can apply time series decomposition to understand patterns in website traffic over time. This can help in identifying recurring trends, season-specific content, and optimizing the timing of promotions or updates to our website.

**import pandas as pd**

**import matplotlib.pyplot as plt**

**from statsmodels.tsa.seasonal import seasonal\_decompose**

**# Load our dataset**

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

**# Clean and preprocess the "Page.Loads" column**

**df['Page.Loads'] = df['Page.Loads'].str.replace(',', '').astype(float)**

**# Convert the "Date" column to datetime and set it as the index**

**df['Date'] = pd.to\_datetime(df['Date'])**

**df.set\_index('Date', inplace=True)**

**# Specify the frequency (assuming daily data)**

**df.index.freq = 'D'**

**# Perform seasonal decomposition using statsmodels**

**result = seasonal\_decompose(df['Page.Loads'], model='additive')**

**# Plot the components**

**plt.figure(figsize=(12, 8))**

**plt.subplot(411)**

**plt.plot(result.observed, label='Observed')**

**plt.legend()**

**plt.subplot(412)**

**plt.plot(result.trend, label='Trend')**

**plt.legend()**

**plt.subplot(413)**

**plt.plot(result.seasonal, label='Seasonal')**

**plt.legend()**

**plt.subplot(414)**

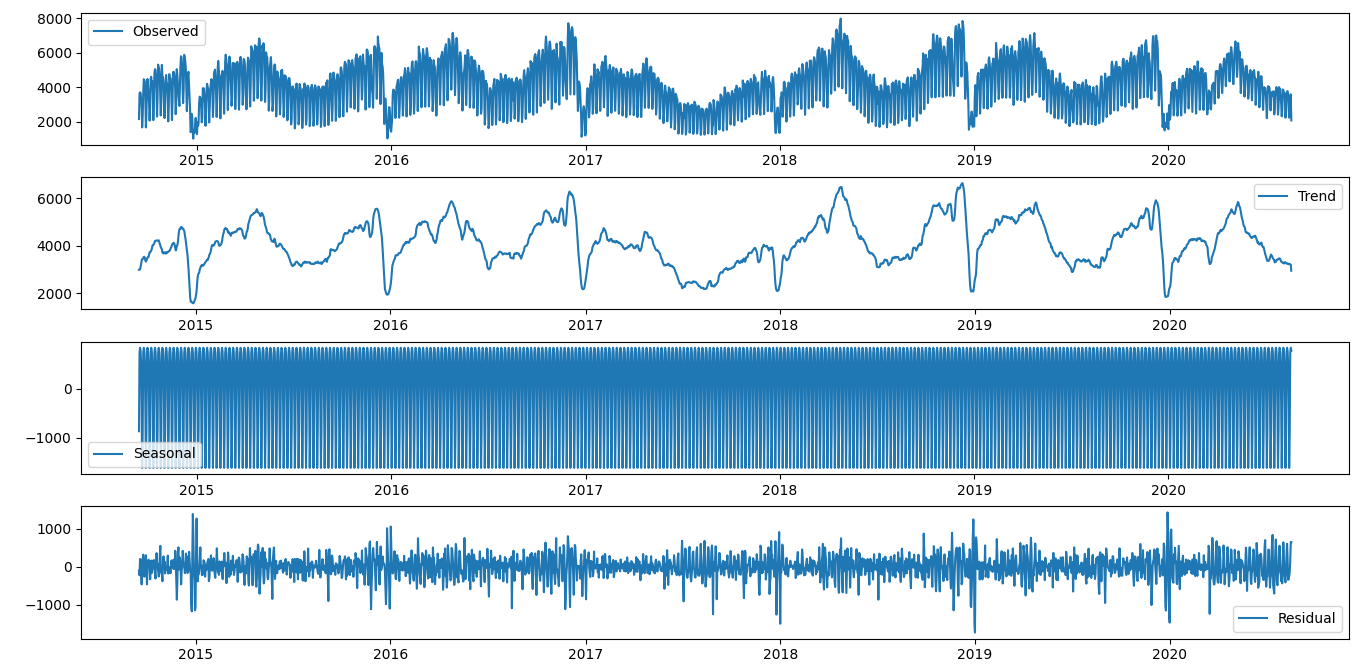
**plt.plot(result.resid, label='Residual')**

**plt.legend()**

**plt.tight\_layout()**

**plt.show()**

**OUTPUT:**

****

**DATA SPLITS:**

**from sklearn.model\_selection import train\_test\_split**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**import seaborn as sns**

**# Load your dataset**

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

**# Split the data into training and testing sets**

**Y = df['Unique.Visits']**

**X = df[['Page.Loads']]**

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2)**

**# Count the data points in each set**

**train\_count = y\_train.count()**

**test\_count = y\_test.count()**

**# Create a bar plot**

**plt.bar(x=['Train', 'Test'], height=[train\_count, test\_count])**

**# Add labels and title**

**plt.xlabel('Dataset')**

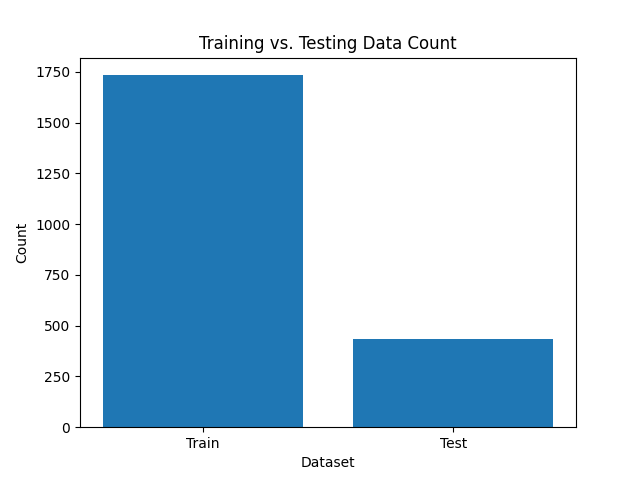
**plt.ylabel('Count')**

**plt.title('Training vs. Testing Data Count')**

**# Show the plot**

**plt.show()**

**OUTPUT:**

****

**Making Machine Learning Model**

**1)Split the Data into Training and Testing Sets:**

We can use a portion of our data for training the model and reserve the rest for testing. The typical split is around 70-80% for training and 20-30% for testing, but we can adjust this based on our data and requirements.

**2)Train the ARIMA Model:**

Train the ARIMA model using the training data. we can use the training data to find the best model parameters

**3)Make Predictions on the Testing Set:**

Use the trained model to make predictions on the testing set.

**4)Evaluate the Model:**

Evaluate the model's performance by comparing the predicted values with the actual values in the testing set. WE can use metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and visualizations to assess the quality of the forecasts.

**OVERALL :**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**# Load our dataset**

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

**# Clean and preprocess the "Page.Loads" column**

**df['Page.Loads'] = df['Page.Loads'].str.replace(',', '').astype(float)**

**# Convert the "Date" column to datetime and set it as the index**

**df['Date'] = pd.to\_datetime(df['Date'])**

**df.set\_index('Date', inplace=True)**

**# Specify the frequency (assuming daily data)**

**df.index.freq = 'D'**

**# Split the data into training and testing sets**

**train\_size = int(len(df) \* 0.8) # Adjust the split ratio as needed**

**train, test = df[:train\_size], df[train\_size:]**

**from statsmodels.tsa.arima.model import ARIMA**

**# Fit an ARIMA model to the training data**

**model = ARIMA(train['Page.Loads'], order=(1, 1, 1))**

**model\_fit = model.fit()**

**forecast\_steps = len(test)**

**forecast = model\_fit.forecast(steps=forecast\_steps)**

**from sklearn.metrics import mean\_squared\_error**

**mse = mean\_squared\_error(test['Page.Loads'], forecast)**

**rmse = mse \*\* 0.5**

**print(f"Mean Squared Error: {mse}")**

**print(f"Root Mean Squared Error: {rmse}")**

**# Plot observed vs. predicted values**

**plt.figure(figsize=(12, 6))**

**plt.plot(test.index, test['Page.Loads'], label='Observed')**

**plt.plot(test.index, forecast, label='Forecast', linestyle='--')**

**plt.xlabel('Date')**

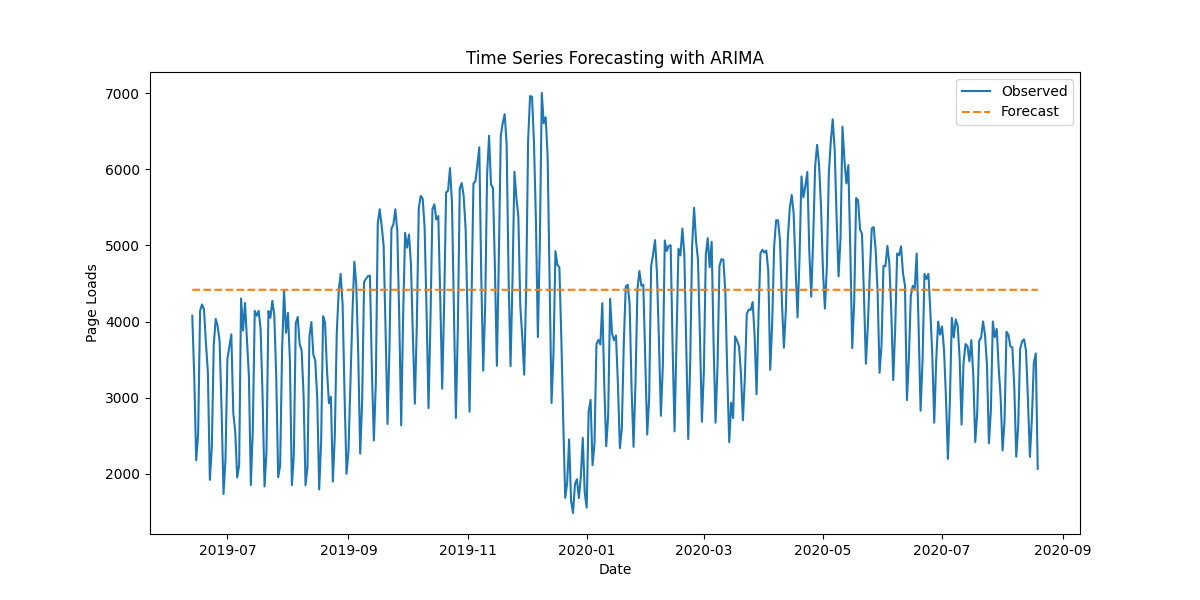
**plt.ylabel('Page Loads')**

**plt.title('Time Series Forecasting with ARIMA')**

**plt.legend()**

**plt.show()**

**OUTPUT:**

****

**All Python Files ,Jupyter Files ,Dashboard Link And Report Link Are Provided In My Github.**

**Enhance the user experience:**

1. **Time-on-Page and Bounce Rates:**
   * **Time-on-Page: This metric reveals how long users spend on a particular page. Longer times indicate user engagement, while shorter times may suggest that users are not finding what they expected.**
   * **Bounce Rates: Bounce rate measures the percentage of visitors who leave a site after viewing only one page. High bounce rates often indicate dissatisfaction or unmet expectations.**

**Optimization Implications:**

* + **If time-on-page is low or bounce rates are high, website owners can investigate and identify the specific pages or content that need improvement.**
  + **Content and design adjustments can be made to provide more engaging, informative, and relevant information to align with user expectations.**
  + **Ensuring that the content matches the user's intent when arriving on a page can reduce bounce rates and increase the time users spend on the site.**

1. **Tracking Search Queries:**
   * **Analyzing the search queries made by users on our website's internal search function can provide valuable insights into what users are looking for.**

**Optimization Implications:**

* + **Understanding user search queries can help website owners create content or features that directly address user needs and preferences.**
  + **Keyword analysis can reveal trending or popular search terms, which can inform content creation and SEO strategies.**
  + **Owners can enhance the search function to better match user queries with relevant content, reducing friction and improving user satisfaction.**

1. **Click-Through Rates (CTR):**
   * **CTR measures the percentage of users who click on a specific link, advertisement, or call-to-action element. It's often used in email marketing, PPC campaigns, and on-site promotions.**

**Optimization Implications:**

* + **Low CTR indicates that users are not engaging with a particular link or element as expected.**
  + **Website owners can experiment with different calls to action, button placements, or link text to improve CTR.**
  + **Analyzing the CTR can help optimize landing pages and ensure that users are directed to the most relevant content or offers.**

**Tips For Improving Traffic :**

* Optimising all of our content for SEO, including picture filenames, is critical for ranking.
* Google is more likely to propose image filenames that are easy to remember.
* The filename should be descriptive, include relevant keywords, and use hyphens to separate words rather than spaces.
* Keep our filenames to a maximum of five to six words.
* Remember that this is simply one part of our total SEO strategy that we should address in the future. DO NOT be concerned about returning to re-upload and rename all of your photographs!

**SEO and Bring More Visitors to our Website:**

Search engine optimisation (SEO) is a theory and collection of recommended practises for assisting our website's indexing by search engines. These tactics are designed to ensure that our website ranks high in search engine results pages whenever someone searches for a keyword or keyword phrase related to the items offer or the themes of the web pages you create.

Search engines employ'spiders,' which scan through the pages of our website, creating an index (similar to the index of a book) that describes what our website is about and what features are available to assist consumers discover what they are looking for.



**Future Directions:**

In the ever-evolving landscape of website traffic analysis, future directions involve harnessing emerging technologies such as advanced artificial intelligence, 5G connectivity, and privacy-preserving techniques. These directions emphasize the importance of real-time insights, adapting to voice and augmented reality interfaces, and ensuring data security through blockchain. Additionally, the focus is on understanding user emotions, embracing quantum computing, and maintaining sustainability while catering to a global and multilingual audience.

**Benefits :**

1. **Improved User Experience:** Understanding user behavior allows website owners to optimize content and design, providing a more user-friendly and engaging experience.
2. **Enhanced Content Strategy:** Insights help tailor content to meet user preferences and interests, increasing user satisfaction.
3. **Better Conversion Rates:** By addressing bottlenecks and issues in the conversion funnel, websites can improve conversion rates and drive revenue.
4. **Data-Driven Decision-Making:** Website owners can make informed decisions about design, content, and marketing strategies, leading to more effective campaigns and improved performance.
5. **Optimized Mobile Experience:** Analysis reveals user device preferences, enabling owners to optimize the mobile experience for smartphones and tablets.
6. **Personalization:** Tailoring content and recommendations based on user behavior leads to a more customized experience, increasing engagement.
7. **Continuous Improvement:** Regular analysis and iteration ensure the website remains competitive and responsive to changing user behavior and market trends.
8. **Issue Resolution:** Anomaly detection can help identify and resolve technical issues swiftly, ensuring a smooth user experience.
9. **User Feedback Integration:** Insights from user feedback can lead to targeted improvements that directly address user concerns and preferences.
10. **Competitive Advantage:** Websites that actively analyze user behavior and make data-driven improvements are better positioned to outperform competitors and maintain user loyalty.

**Conclusion:**

**"In conclusion, our website traffic analysis project has yielded valuable insights for improving user experience. By analyzing metrics and user behavior, we're committed to making data-driven enhancements, optimizing content, and delivering a more engaging online journey. We understand the importance of addressing bounce rates and optimizing content to align with user intent. Our ongoing commitment to continuous improvement ensures a user experience that exceeds expectations."**