**Phase - 5**

**PROJECT DOCUMENTATION & SUBMISSION**

**WEBSITE TRAFFIC ANALYSIS**

|  |  |
| --- | --- |
| **Date** | **31-10-2023** |
| **Team ID** | **497** |
| **Project Name** | **6112 - Website Traffic Analysis** |

**Table of Contents:**

|  |  |
| --- | --- |
| 1. | Introduction |
| 2. | Problem Statement |
| 3. | Project Overview |
| 4. | Literature survey |
| 5. | Design Thinking Process |
| 6. | Development Phase |
| 7. | Data Collection Process |
| 8. | Data Visualization using IBM Cognos |
| 9. | Data Visualization using Python |
| 10. | Python Code Integration |
| 11. | Insights |
| 12. | Conclusion |

**1. Introduction:**

Web traffic analysis is the process of collecting, analyzing, and interpreting data about the visitors to a website. It provides valuable insights into how users interact with a website, what content they engage with, and how they navigate through it. This information is crucial for website owners, marketers, and web developers to make informed decisions and optimize the user experience.

**2. Problem Statement:**

In today’s digital landscape, businesses and website owners face the challenge of effectively understanding and leveraging their web traffic data to optimize their online presence and achieve their goals. The problem lies in the complexity of modern websites, the vast amount of data generated by users, and the need to turn this data into actionable insights. Key issues include identifying traffic sources, improving user engagement, reducing bounce rates, and enhancing conversion rates. Additionally, ensuring compliance with data privacy regulations adds a layer of complexity to web traffic analysis. To succeed in the online marketplace, businesses need a robust and efficient web traffic analysis solution that addresses these challenges, enabling them to make data-driven decisions, enhance user experiences, and meet their objectives.

**3. Project Overview:**

Objective: The project aims to analyse Website traffic data to improve user experience, enhance website performance, and optimize marketing strategies.

**4. Literature Survey:**

**Case 1:** **Machine Learning Based Web-Traffic Analysis for Detection of Fraudulent Resources Consumption Attack in Cloud**

The solution for detecting FRC attacks on cloud-based web servers involves a combination of techniques including quantile-based division, discrete wavelet transforms, and artificial neural network modelling. The web-pages are first divided into a fixed number of quantiles based on their popularity index, and the number of requests per hour for each of these quantiles is computed. Discrete Wavelet Transform is then applied to these quantiles to remove any high-frequency anomaly and smoothen the time series data. The n-tuple data from these quantiles along with their label (attack or normal) is used to train an Artificial Neural Network model. The trained model achieved an accuracy of 98.51% in detecting FRC attacks with a low percent of 5%. The proposed scheme is based on the fact that during an FRC attack, these different quantiles will have a disproportionate amount of change in the request pattern (violating Zipf's Law).

**Case 2: A Study of Web Traffic Analysis**

The solution for the problem statement - The authors aim to address the problem of effectively managing web server performance by studying web traffic analysis. They discuss the importance of understanding web user usage patterns and how it can help optimize web server usage for future growth. They also explore the different methods and tools of analysing web traffic data and suggest how web security can be improved by traffic analysis. The authors propose a hybrid approach that is efficient for mining and predicting web server traffic, and they also introduce layered queuing models (LQMs) that estimate client response time at a web server. Overall, the PDF provides insights and solutions for effectively managing web server performance through web traffic analysis.

**Case 3: A Review of Network Traffic Analysis and Prediction Techniques**

The review paper provides a comprehensive overview of various techniques proposed in previous studies for network traffic analysis and prediction. The paper discusses different approaches, including data mining techniques, neural network and component analysis, and linear and nonlinear time series models. The authors also group the research works surveyed in the paper based on the datasets used, implemented algorithms, and metrics used to evaluate the results. The review paper aims to provide insight to new researchers and network administrators on the different techniques available for effective network traffic analysis and prediction. The techniques discussed in the paper can be used to address different problems related to network traffic, such as intrusion detection, network performance optimization, and security management.

**Case 4: A Web Traffic Analysis Attack Using Only Timing Information**

The solution proposed in the paper is a novel approach to detecting encrypted web traffic that uses packet timing information alone. The paper describes an attack against encrypted web traffic that is highly effective against both wired and wireless traffic, achieving mean success rates in excess of 90%. The attack is also effective against traffic streams, where the packet boundaries between fetches are unknown. The paper suggests that this timing-only attack highlights deficiencies in existing defences and suggests areas where VPN designers can focus further attention to improve their defences against this type of attack. The paper also proposes potential countermeasures that could be developed to defend against this type of attack.

**Case 5: The Continued Evolution of Web Traffic**

We seek a method for segmenting web traffic into sections larger than request interactions, where these sections are related to the HTTP activity on the device. The method must be able to be applied to past and current traces, and require only anonymized TCP/IP headers as inputs. Connections durations have also increased substantially. This is due to better use of connections by the browsers. The introduction of speculative connections features added to modern browsers. Finally, the number of servers supporting web activity has consistently increased.

**5. Design Thinking process:**

**Empathize:**

* Start by gathering data on website’s current traffic patterns. Use tools like Google Analytics to collect information on user demographics, and traffic sources.
* Create user survey to better understand the different types of visitors coming to the site and their main points and expectations.

**Define:**

* After gaining empathy, you define the problem. You synthesize the information gathered in the empathize stage to create a clear and concise problem statement.
* This statement serves as a point of references for the rest of the process, ensuring a well-defined problem to solve.

**Ideate:**

* This stage encourages creative thinking to generate a wide range of possible solutions. It involves brainstorming, sketching, and thinking outside the box. The focus is on quantity and diversity of the ideas, without judgement or critique at this stage.

**Prototype:**

* Once you have a set of promising ideas, you create low-fidelity prototypes or representations of those ideas. These prototypes can be paper sketches, wireframes, or even simple models. Prototyping allows you to quickly test and refine ideas without significant investment.

**Test:**

* Testing involves gathering feedback from users through the evaluation of prototypes. This helps to understand what works and what doesn’t, and it often leads to revisions and further iterations of the prototype. The goal is to refine the solution based on real user feedback.

**6. Development Phase:**

**Planning:**

* Defining your goals, target audience, content strategy, technical requirements, budget, and resources.

**Research:**

* Analyzing keywords, market, user behaviour, competitors, content, backlinks, and technical SEO to inform your website development and SEO strategy.

**Data Preprocessing:**

* Data preprocessing in website traffic analysis involves cleaning and preparing the data collected from website analytics tools for effective analysis.

**Analysis and visualization:**

* In this process, analyse the data to understand the user behaviour, identifying trends, and making data-driven decisions. And visualize the finding using tools like Python for custom visualization and platforms like IBM Cognos for comprehensive reporting.

**Insights and Recommendations:**

* Interpret the findings and provide actionable recommendations for the betterment.

**Implementation:**

* It involves putting into action the insights and strategies derived from the analysis and planning phases.

**Testing and Validation:**

* These are the essential step in the development and implementation of various projects and systems to ensure they meet their objectives and quality standards.

**7. Data Collection Process:**

Data Collection is crucial step in website traffic analysis. It involves cleaning and transforming the raw data collected from website analytics tools to make it suitable for analysis.

**Data Collection:**

* Collect the data from web analytics tools.

**Data Cleaning:**

* Handle missing data.
* Remove duplicates.
* Data validation.

**Data Transformation:**

* Group the data into meaningful categories.

**Feature Engineering:**

* Create new features that are useful for analysis.

**Data Scaling:**

* Scale or normalize numeric feature if necessary to ensure that they have a similar impact during analysis.

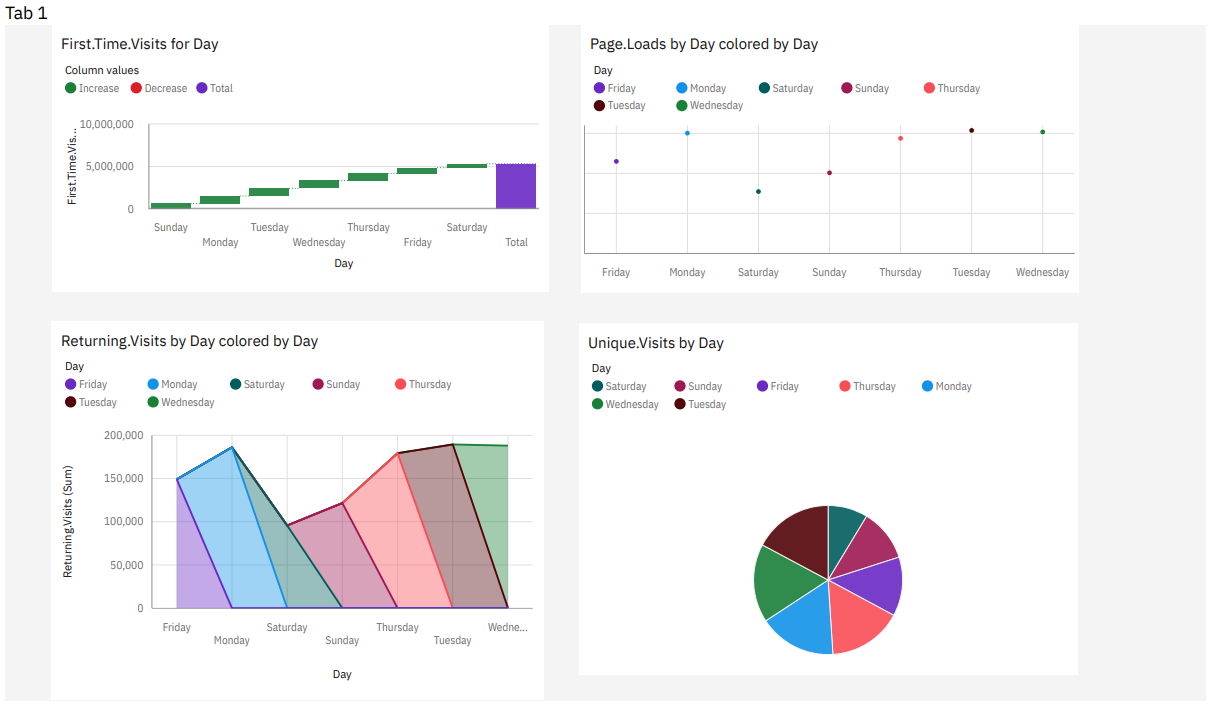
**Data Quality Assurance:**

* Continuously monitor data quality and take corrective actions when anomalies or issues arise.

**Documentation:**

* Keep a detailed record of the preprocessing steps and transforming applied to the data for transparency and reproducibility.

**8. Data Visualization using IBM Cognos:**

****

**First. Time. Visits for Day:**

Report:

* First. Time. Visits is unusually low when Day is Saturday.
* Based on the current forecasting, First. Time. Visits may reach over 395 thousand by Day Monday+1.
* Over all days, the sum of First. Time. Visits is almost 5.3 million.
* First. Time. Visits ranges from over 456 thousand, when Day is Saturday, to nearly 908 thousand, when Day is Tuesday.
* For First. Time. Visits, the most significant values of Day are Tuesday, Wednesday, Monday, Thursday, and Friday, whose respective First. Time. Visits values add up to over 4.2 million, or 79.9 % of the total.

**Page. Loads by Day colored by Day:**

Report:

* Page. Loads is unusually low when Day is Saturday.
* Based on the current forecasting, Page. Loads may reach over 675 thousand by Day Monday+1.
* Across all days and days, the sum of Page. Loads is over 8.9 million.
* The summed values of Page. Loads range from nearly 773 thousand to over 1.5 million.
* For Page. Loads, the most significant values of Day are Tuesday, Wednesday, Monday, Thursday, and Friday, whose respective Page. Loads values add up to over 7.1 million, or 80.1 % of the total.

**Returning. Visits by Day colored by Day:**

Report:

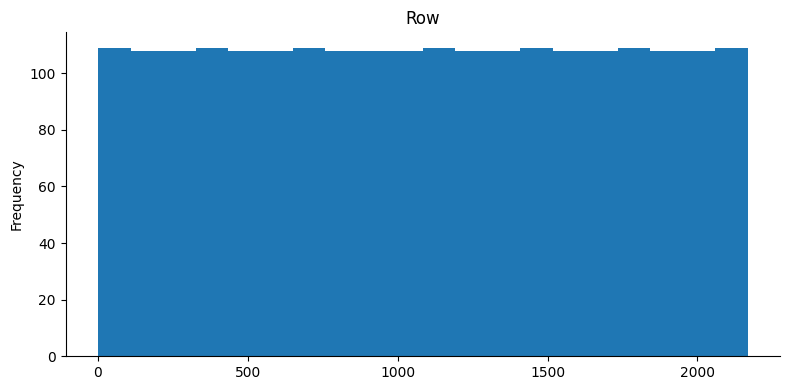
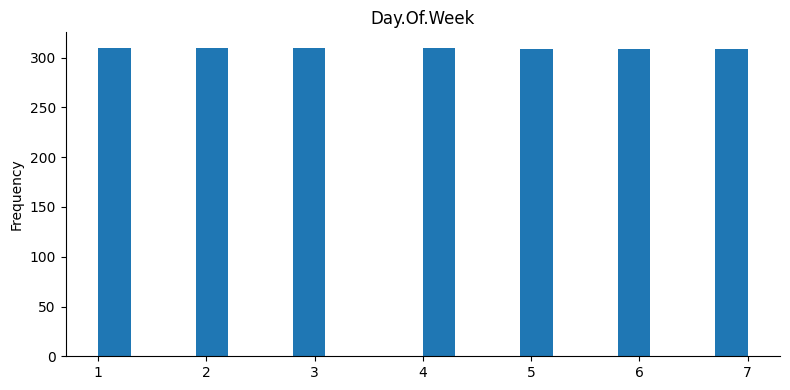
* Returning. Visits is unusually low when Day is Saturday.
* Based on the current forecasting, Returning. Visits may reach over 87 thousand by Day Monday+1.
* Across all days and days, the sum of Returning. Visits is over 1.1 million.
* The summed values of Returning. Visits range from almost 96 thousand to over 189 thousand.
* For Returning. Visits, the most significant values of Day are Tuesday, Wednesday, Monday, Thursday, and Friday, whose respective Returning. Visits values add up to almost 892 thousand, or 80.4 % of the total.

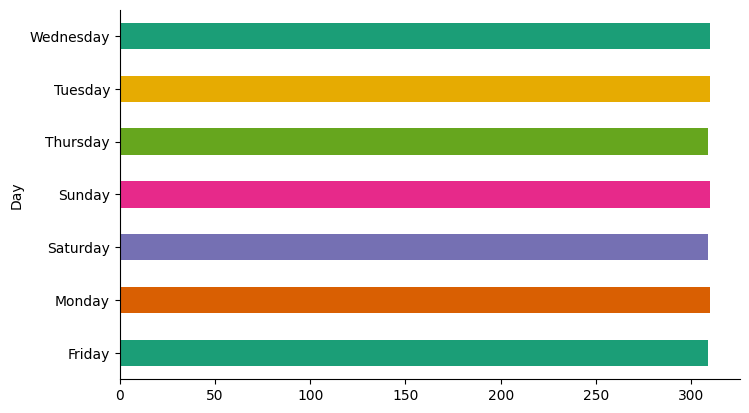
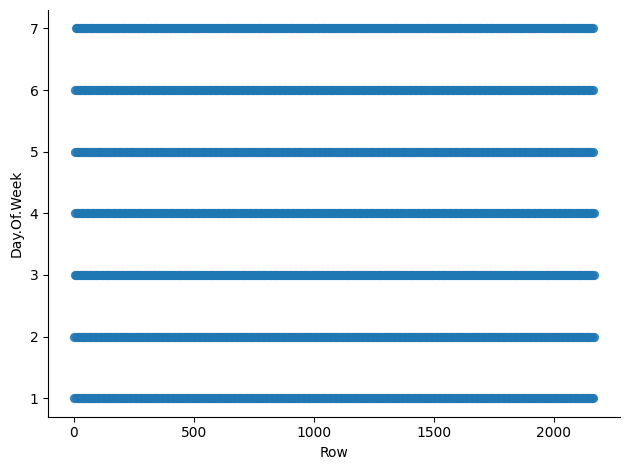
**Unique. Visits by Day:**

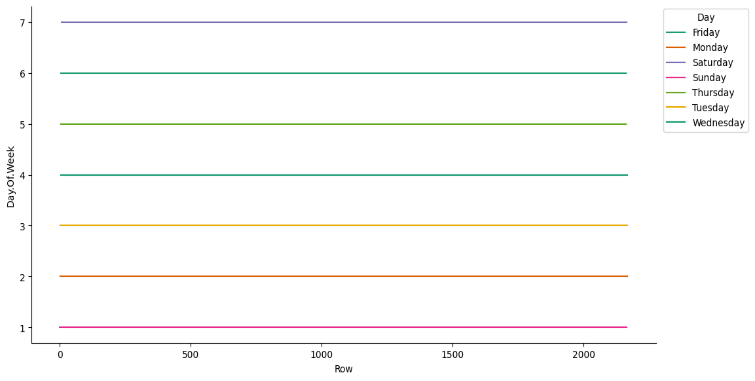
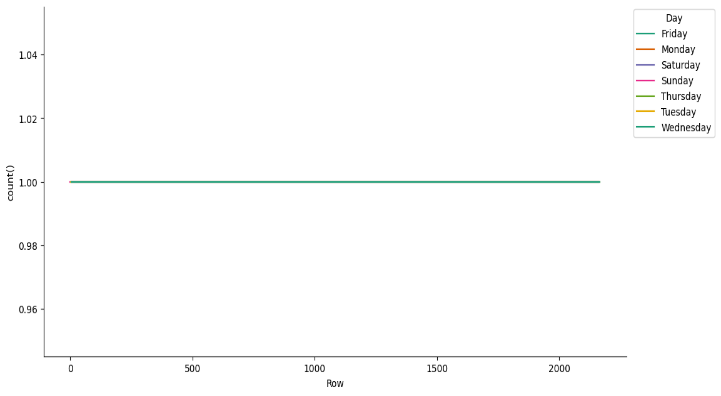
Report:

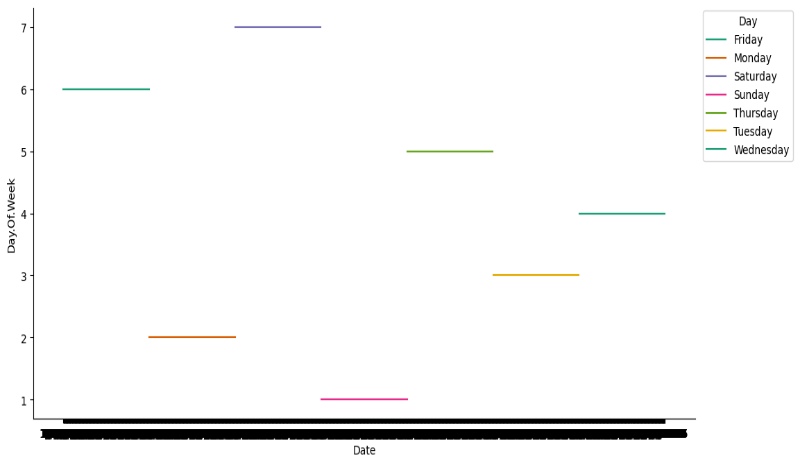
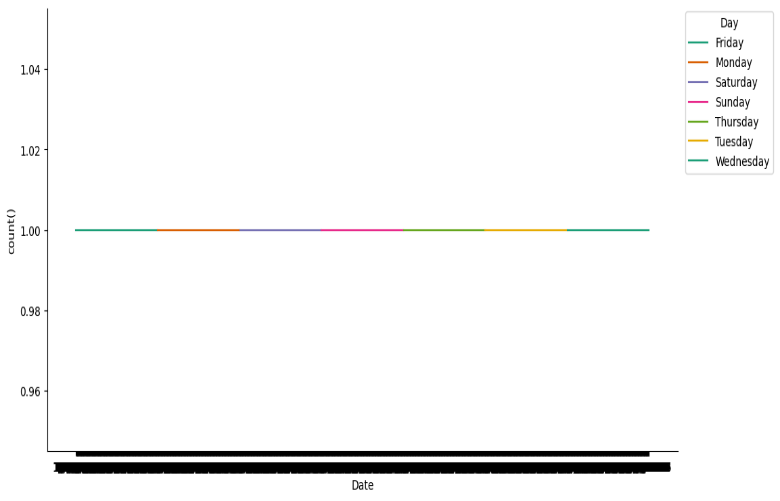
* Unique. Visits is unusually low when Day is Saturday.
* Based on the current forecasting, Unique. Visits may reach almost 481 thousand by Day Monday+1.
* Over all days, the sum of Unique. Visits is almost 6.4 million.
* Unique. Visits ranges from over 552 thousand, when Day is Saturday, to nearly 1.1 million, when Day is Tuesday.
* For Unique. Visits, the most significant values of Day are Tuesday, Wednesday, Monday, Thursday, and Friday, whose respective Unique. Visits values add up to over 5.1 million, or 80 % of the total.

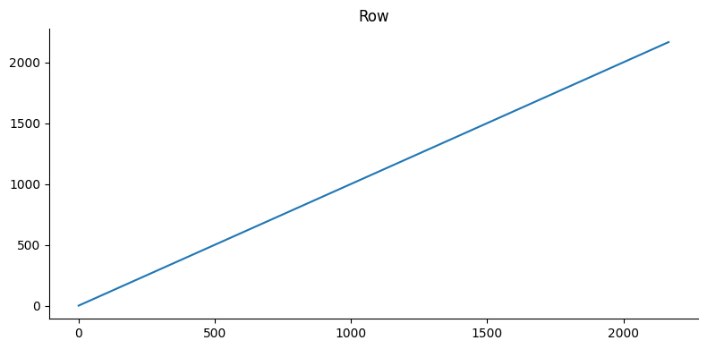
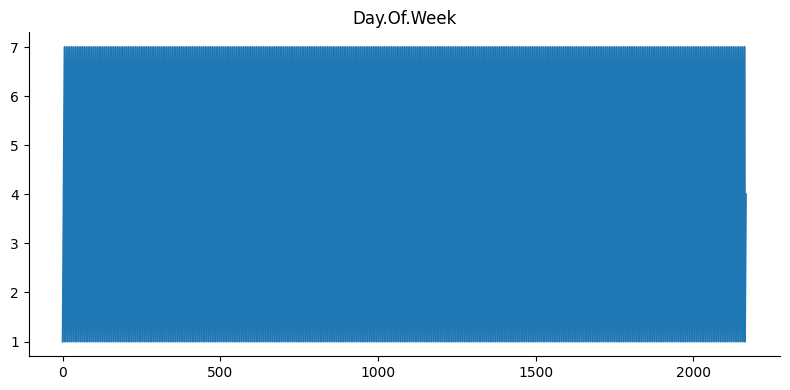
**9. Data Visualization using Python:**

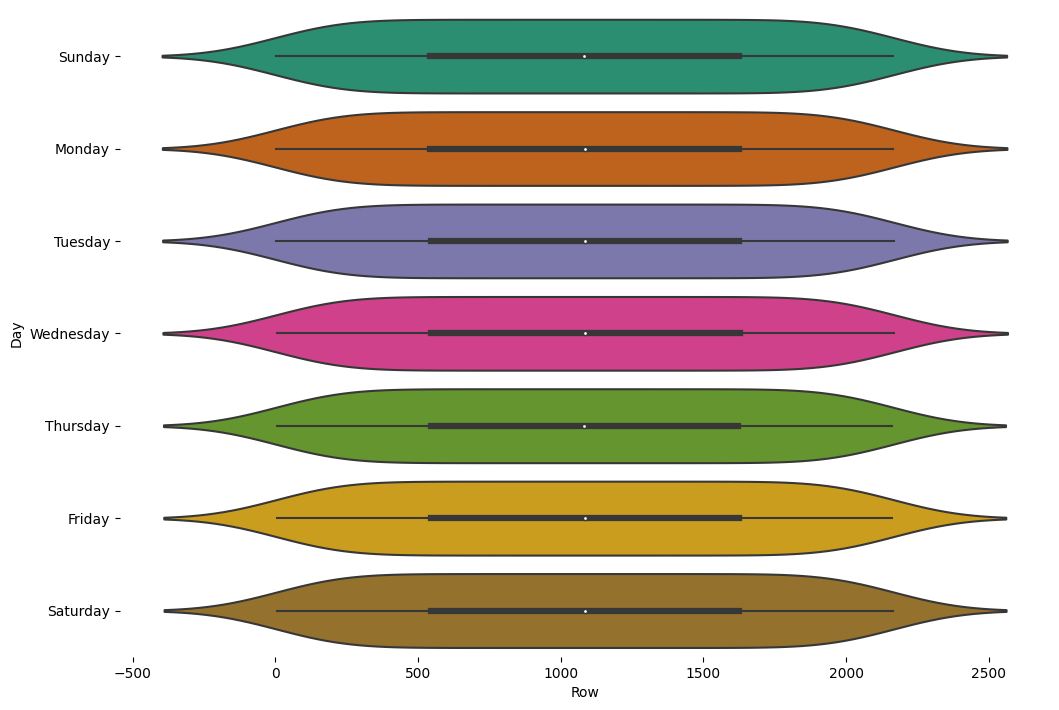


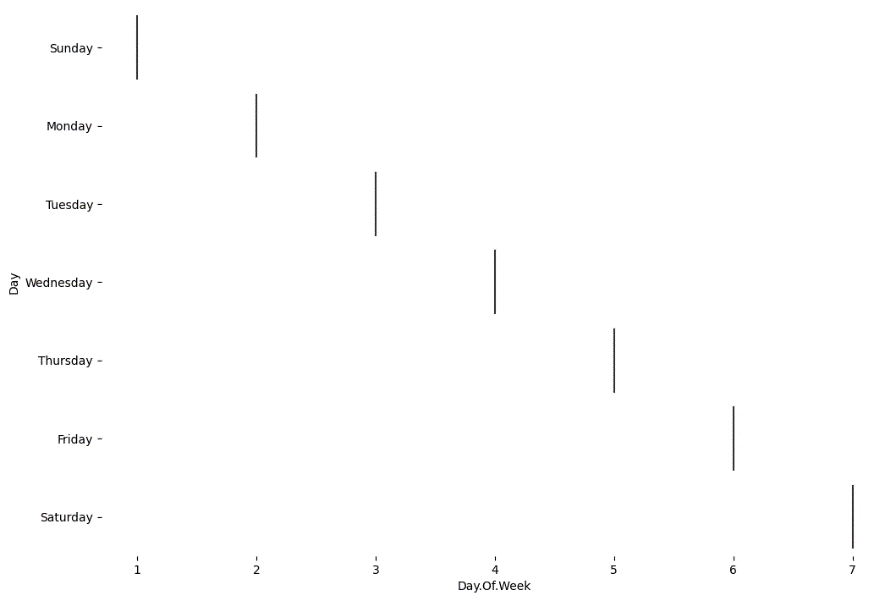












**10. Python Code Integration:**

import pandas as pd

x=pd.read\_csv("/content/daily-website-visitors.csv")

x

x.isnull().sum()

x

import numpy as np

from google.colab import autoviz

def histogram(df, colname, num\_bins=20, figscale=1):

  from matplotlib import pyplot as plt

  df[colname].plot(kind='hist', bins=num\_bins, title=colname, figsize=(8\*figscale, 4\*figscale))

  plt.gca().spines[['top', 'right',]].set\_visible(False)

  plt.tight\_layout()

  return autoviz.MplChart.from\_current\_mpl\_state()

chart = histogram(x, \*['Row'], \*\*{})

chart

import numpy as np

from google.colab import autoviz

def histogram(df, colname, num\_bins=20, figscale=1):

  from matplotlib import pyplot as plt

  df[colname].plot(kind='hist', bins=num\_bins, title=colname, figsize=(8\*figscale, 4\*figscale))

  plt.gca().spines[['top', 'right',]].set\_visible(False)

  plt.tight\_layout()

  return autoviz.MplChart.from\_current\_mpl\_state()

chart = histogram(x, \*['Day.Of.Week'], \*\*{})

chart

import numpy as np

from google.colab import autoviz

def categorical\_histogram(df, colname, figscale=1, mpl\_palette\_name='Dark2'):

  from matplotlib import pyplot as plt

  import seaborn as sns

  df.groupby(colname).size().plot(kind='barh', color=sns.palettes.mpl\_palette(mpl\_palette\_name), figsize=(8\*figscale, 4.8\*figscale))

  plt.gca().spines[['top', 'right',]].set\_visible(False)

  return autoviz.MplChart.from\_current\_mpl\_state()

chart = categorical\_histogram(x, \*['Day'], \*\*{})

chart

import numpy as np

from google.colab import autoviz

def scatter\_plot(df, x\_colname, y\_colname, figscale=1, alpha=.8):

  from matplotlib import pyplot as plt

  plt.figure(figsize=( 6 \* figscale, 6 \* figscale))

  df.plot(kind='scatter', x=x\_colname, y=y\_colname, s=(32 \* figscale), alpha=alpha)

  plt.gca().spines[['top', 'right',]].set\_visible(False)

  plt.tight\_layout()

  return autoviz.MplChart.from\_current\_mpl\_state()

chart = scatter\_plot(x, \*['Row', 'Day.Of.Week'], \*\*{})

chart

import numpy as np

from google.colab import autoviz

def time\_series\_multiline(df, timelike\_colname, value\_colname, series\_colname, figscale=1, mpl\_palette\_name='Dark2'):

  from matplotlib import pyplot as plt

  import seaborn as sns

  figsize = (10 \* figscale, 5.2 \* figscale)

  palette = list(sns.palettes.mpl\_palette(mpl\_palette\_name))

  def \_plot\_series(series, series\_name, series\_index=0):

    if value\_colname == 'count()':

      counted = (series[timelike\_colname]

                 .value\_counts()

                 .reset\_index(name='counts')

                 .rename({'index': timelike\_colname}, axis=1)

                 .sort\_values(timelike\_colname, ascending=True))

      xs = counted[timelike\_colname]

      ys = counted['counts']

    else:

      xs = series[timelike\_colname]

      ys = series[value\_colname]

    plt.plot(xs, ys, label=series\_name, color=palette[series\_index % len(palette)])

  fig, ax = plt.subplots(figsize=figsize, layout='constrained')

  df = df.sort\_values(timelike\_colname, ascending=True)

  if series\_colname:

    for i, (series\_name, series) in enumerate(df.groupby(series\_colname)):

      \_plot\_series(series, series\_name, i)

    fig.legend(title=series\_colname, bbox\_to\_anchor=(1, 1), loc='upper left')

  else:

    \_plot\_series(df, '')

  sns.despine(fig=fig, ax=ax)

  plt.xlabel(timelike\_colname)

  plt.ylabel(value\_colname)

  return autoviz.MplChart.from\_current\_mpl\_state()

chart = time\_series\_multiline(x, \*['Row', 'Day.Of.Week', 'Day'], \*\*{})

chart

import numpy as np

from google.colab import autoviz

def time\_series\_multiline(df, timelike\_colname, value\_colname, series\_colname, figscale=1, mpl\_palette\_name='Dark2'):

  from matplotlib import pyplot as plt

  import seaborn as sns

  figsize = (10 \* figscale, 5.2 \* figscale)

  palette = list(sns.palettes.mpl\_palette(mpl\_palette\_name))

  def \_plot\_series(series, series\_name, series\_index=0):

    if value\_colname == 'count()':

      counted = (series[timelike\_colname]

                 .value\_counts()

                 .reset\_index(name='counts')

                 .rename({'index': timelike\_colname}, axis=1)

                 .sort\_values(timelike\_colname, ascending=True))

      xs = counted[timelike\_colname]

      ys = counted['counts']

    else:

      xs = series[timelike\_colname]

      ys = series[value\_colname]

    plt.plot(xs, ys, label=series\_name, color=palette[series\_index % len(palette)])

  fig, ax = plt.subplots(figsize=figsize, layout='constrained')

  df = df.sort\_values(timelike\_colname, ascending=True)

  if series\_colname:

    for i, (series\_name, series) in enumerate(df.groupby(series\_colname)):

      \_plot\_series(series, series\_name, i)

    fig.legend(title=series\_colname, bbox\_to\_anchor=(1, 1), loc='upper left')

  else:

    \_plot\_series(df, '')

  sns.despine(fig=fig, ax=ax)

  plt.xlabel(timelike\_colname)

  plt.ylabel(value\_colname)

  return autoviz.MplChart.from\_current\_mpl\_state()

chart = time\_series\_multiline(x, \*['Row', 'count()', 'Day'], \*\*{})

chart

import numpy as np

from google.colab import autoviz

def time\_series\_multiline(df, timelike\_colname, value\_colname, series\_colname, figscale=1, mpl\_palette\_name='Dark2'):

  from matplotlib import pyplot as plt

  import seaborn as sns

  figsize = (10 \* figscale, 5.2 \* figscale)

  palette = list(sns.palettes.mpl\_palette(mpl\_palette\_name))

  def \_plot\_series(series, series\_name, series\_index=0):

    if value\_colname == 'count()':

      counted = (series[timelike\_colname]

                 .value\_counts()

                 .reset\_index(name='counts')

                 .rename({'index': timelike\_colname}, axis=1)

                 .sort\_values(timelike\_colname, ascending=True))

      xs = counted[timelike\_colname]

      ys = counted['counts']

    else:

      xs = series[timelike\_colname]

      ys = series[value\_colname]

    plt.plot(xs, ys, label=series\_name, color=palette[series\_index % len(palette)])

  fig, ax = plt.subplots(figsize=figsize, layout='constrained')

  df = df.sort\_values(timelike\_colname, ascending=True)

  if series\_colname:

    for i, (series\_name, series) in enumerate(df.groupby(series\_colname)):

      \_plot\_series(series, series\_name, i)

    fig.legend(title=series\_colname, bbox\_to\_anchor=(1, 1), loc='upper left')

  else:

    \_plot\_series(df, '')

  sns.despine(fig=fig, ax=ax)

  plt.xlabel(timelike\_colname)

  plt.ylabel(value\_colname)

  return autoviz.MplChart.from\_current\_mpl\_state()

chart = time\_series\_multiline(x, \*['Date', 'Day.Of.Week', 'Day'], \*\*{})

chart

import numpy as np

from google.colab import autoviz

def time\_series\_multiline(df, timelike\_colname, value\_colname, series\_colname, figscale=1, mpl\_palette\_name='Dark2'):

  from matplotlib import pyplot as plt

  import seaborn as sns

  figsize = (10 \* figscale, 5.2 \* figscale)

  palette = list(sns.palettes.mpl\_palette(mpl\_palette\_name))

  def \_plot\_series(series, series\_name, series\_index=0):

    if value\_colname == 'count()':

      counted = (series[timelike\_colname]

                 .value\_counts()

                 .reset\_index(name='counts')

                 .rename({'index': timelike\_colname}, axis=1)

                 .sort\_values(timelike\_colname, ascending=True))

      xs = counted[timelike\_colname]

      ys = counted['counts']

    else:

      xs = series[timelike\_colname]

      ys = series[value\_colname]

    plt.plot(xs, ys, label=series\_name, color=palette[series\_index % len(palette)])

  fig, ax = plt.subplots(figsize=figsize, layout='constrained')

  df = df.sort\_values(timelike\_colname, ascending=True)

  if series\_colname:

    for i, (series\_name, series) in enumerate(df.groupby(series\_colname)):

      \_plot\_series(series, series\_name, i)

    fig.legend(title=series\_colname, bbox\_to\_anchor=(1, 1), loc='upper left')

  else:

    \_plot\_series(df, '')

  sns.despine(fig=fig, ax=ax)

  plt.xlabel(timelike\_colname)

  plt.ylabel(value\_colname)

  return autoviz.MplChart.from\_current\_mpl\_state()

chart = time\_series\_multiline(x, \*['Date', 'count()', 'Day'], \*\*{})

chart

import numpy as np

from google.colab import autoviz

def value\_plot(df, y, figscale=1):

  from matplotlib import pyplot as plt

  df[y].plot(kind='line', figsize=(8 \* figscale, 4 \* figscale), title=y)

  plt.gca().spines[['top', 'right']].set\_visible(False)

  plt.tight\_layout()

  return autoviz.MplChart.from\_current\_mpl\_state()

chart = value\_plot(x, \*['Row'], \*\*{})

chart

import numpy as np

from google.colab import autoviz

def value\_plot(df, y, figscale=1):

  from matplotlib import pyplot as plt

  df[y].plot(kind='line', figsize=(8 \* figscale, 4 \* figscale), title=y)

  plt.gca().spines[['top', 'right']].set\_visible(False)

  plt.tight\_layout()

  return autoviz.MplChart.from\_current\_mpl\_state()

chart = value\_plot(x, \*['Day.Of.Week'], \*\*{})

chart

import numpy as np

from google.colab import autoviz

def violin\_plot(df, value\_colname, facet\_colname, figscale=1, mpl\_palette\_name='Dark2', \*\*kwargs):

  from matplotlib import pyplot as plt

  import seaborn as sns

  figsize = (12 \* figscale, 1.2 \* figscale \* len(df[facet\_colname].unique()))

  plt.figure(figsize=figsize)

  sns.violinplot(df, x=value\_colname, y=facet\_colname, palette=mpl\_palette\_name, \*\*kwargs)

  sns.despine(top=True, right=True, bottom=True, left=True)

  return autoviz.MplChart.from\_current\_mpl\_state()

chart = violin\_plot(x, \*['Row', 'Day'], \*\*{'inner': 'box'})

chart

import numpy as np

from google.colab import autoviz

def violin\_plot(df, value\_colname, facet\_colname, figscale=1, mpl\_palette\_name='Dark2', \*\*kwargs):

  from matplotlib import pyplot as plt

  import seaborn as sns

  figsize = (12 \* figscale, 1.2 \* figscale \* len(df[facet\_colname].unique()))

  plt.figure(figsize=figsize)

  sns.violinplot(df, x=value\_colname, y=facet\_colname, palette=mpl\_palette\_name, \*\*kwargs)

  sns.despine(top=True, right=True, bottom=True, left=True)

  return autoviz.MplChart.from\_current\_mpl\_state()

chart = violin\_plot(x, \*['Day.Of.Week', 'Day'], \*\*{'inner': 'box'})

chart

**11. Insights:**

The analysis highlights that specific website metrics, such as First. Time. Visits, Page. Loads, Returning. Visits, and Unique. Visits, exhibit variations on Saturdays. To enhance user experience, website owners can tailor their content and features to better align with user behaviour on Saturdays, potentially addressing any identified issues causing lower engagement. Furthermore, preparing engaging content and features in anticipation of the forecasted surge on Monday+1 can help maintain and further boost user engagement. By understanding and acting on these insights, website owners can optimize their website's performance, making it more user-friendly and responsive to user preferences, ultimately leading to a more satisfying and engaging user experience.

**12. Conclusion:**

In conclusion, website traffic analysis is a dynamic and essential practice for website owners and digital marketers. It provides valuable insights into user behaviour, performance, and the effectiveness of online strategies. The data-driven decisions made possible by this analysis can greatly enhance the user experience, optimize marketing efforts, and achieve business objectives. Continuous monitoring, adaptation, and adherence to data security and privacy regulations are crucial for success. By following a well-structured and iterative approach to website traffic analysis, businesses can stay attuned to evolving user behaviour and maintain a strong and user-friendly online presence, ensuring long-term success in the digital landscape.