WEBSITE TRAFFIC ANALYSIS
Development part - 1

Date	**26-10-2023**
Team ID	**497**
Project Name	**6112 - Website Traffic Analysis**

Table of Contents:

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.

Data Pre-processing:

Data Preprocessing is a 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.

1. **Data collection:**
   - Collect the data from web analytics tools.
2. **Data cleaning:**
   - **Handle missing data:** Remove or impute missing values, as missing data can lead to inaccurate insights.
   - **Remove duplicates:** Check for and remove duplicate records, especially if the data source may generate duplicate entries.

- **Data validation:** Validate the data to ensure it conforms to expected formats and ranges.

3. **Data Transformation:**
    - **Categorization:** Group data into meaningful categories, such as segmenting users by demographics or behaviour.
4. **Feature Engineering:**
    - **Create new features:** Generate additional features that may be useful for analysis, extracting the day, date, etc.,
5. **Data scaling:**
    - Scale or normalise numeric feature if necessary to ensure that they have a similar impact during analysis.
6. **Data Quality Assurance:**
    - Continuously monitor data quality and take corrective actions when anomalies or issues arise.
7. **Documentation:**
    - Keep a detailed record of the preprocessing steps and transformation applied to the data for transparency and reproducibility.

Data Visualization:

```
import pandas as pd
x=pd.read_csv("/content/daily-website-visitors.csv")
```

```
x
```

Row	Day	Day.Of.Week	Date	Page.Loads	Unique.Visits	First.Time.Visits	Returning.Visits	
0	1	Sunday	1	9/14/2014	2,146	1,582	1,430	152
1	2	Monday	2	9/15/2014	3,621	2,528	2,297	231
2	3	Tuesday	3	9/16/2014	3,698	2,630	2,352	278
3	4	Wednesday	4	9/17/2014	3,667	2,614	2,327	287
4	5	Thursday	5	9/18/2014	3,316	2,366	2,130	236
...	...	...	...	...	...	...	...	...
2162	2163	Saturday	7	8/15/2020	2,221	1,696	1,373	323

Row	Day	Day.Of.Week	Date	Page.Loads	Unique.Visits	First.Time.Visits	Returning.Visits	
2163	2164	Sunday	1	8/16/2020	2,724	2,037	1,686	351
2164	2165	Monday	2	8/17/2020	3,456	2,638	2,181	457
2165	2166	Tuesday	3	8/18/2020	3,581	2,683	2,184	499
2166	2167	Wednesday	4	8/19/2020	2,064	1,564	1,297	267

2167 rows × 8 columns

```
x.isnull().sum()
```

```
Row 0
Day 0
Day.Of.Week 0
Date 0
Page.Loads 0
Unique.Visits 0
First.Time.Visits 0
Returning.Visits 0
dtype: int64
```

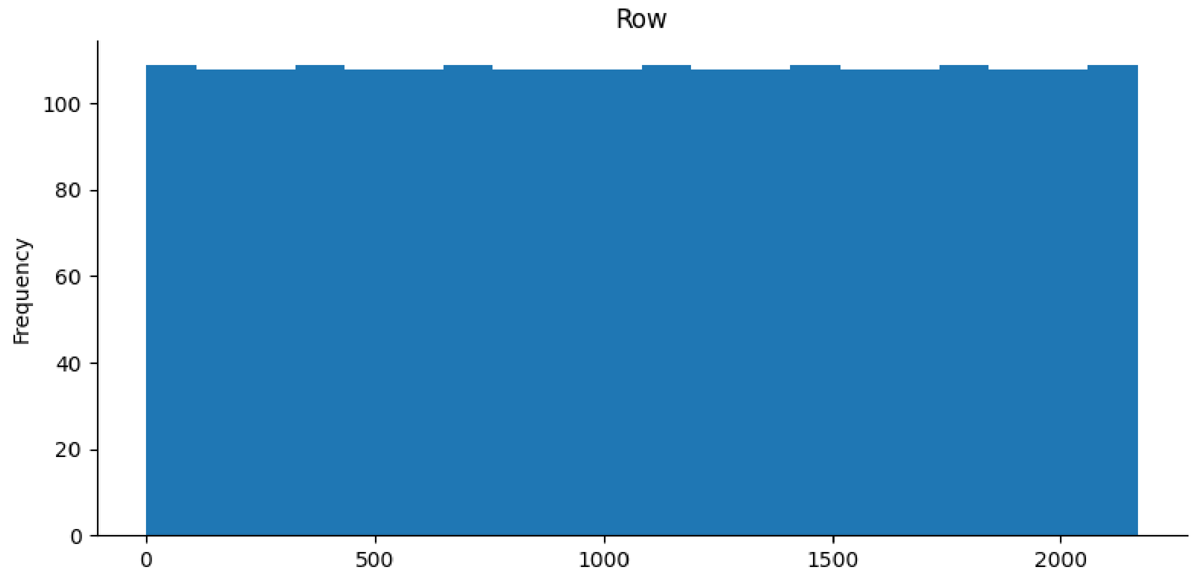
```
x
```

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

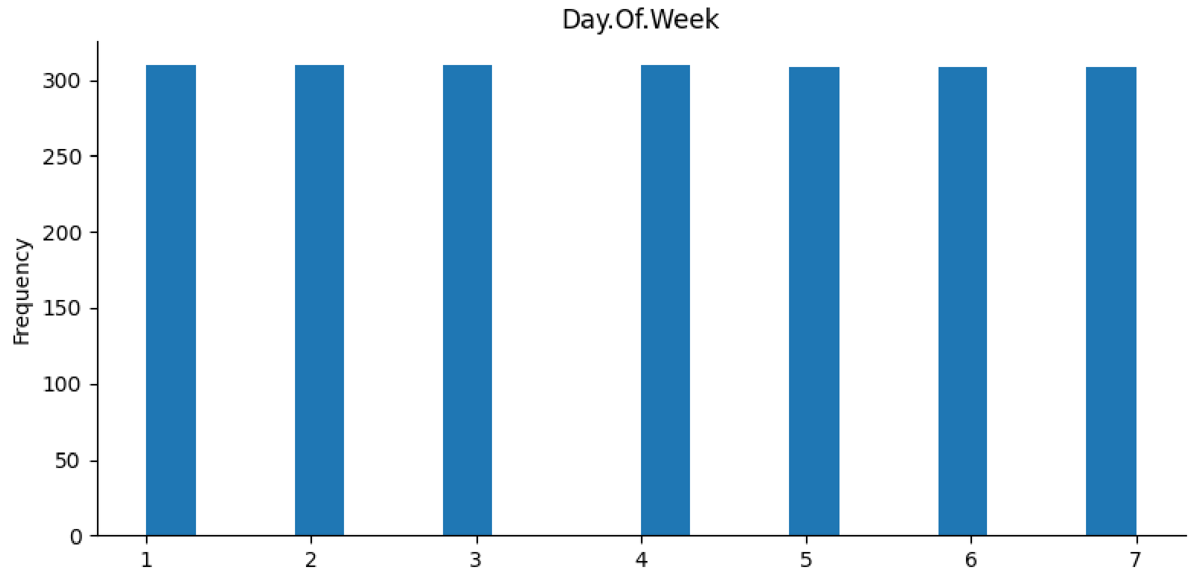
Row	Day	Day.Of.Week	Date	Page.Loads	Unique.Visits	First.Time.Visits	Returning.Visits	
4	5	Thursday	5	9/18/2014	3,316	2,366	2,130	236
...	...	...	...	...	...	...	...	...
2162	2163	Saturday	7	8/15/2020	2,221	1,696	1,373	323
2163	2164	Sunday	1	8/16/2020	2,724	2,037	1,686	351
2164	2165	Monday	2	8/17/2020	3,456	2,638	2,181	457
2165	2166	Tuesday	3	8/18/2020	3,581	2,683	2,184	499
2166	2167	Wednesday	4	8/19/2020	2,064	1,564	1,297	267

2167 rows × 8 columns

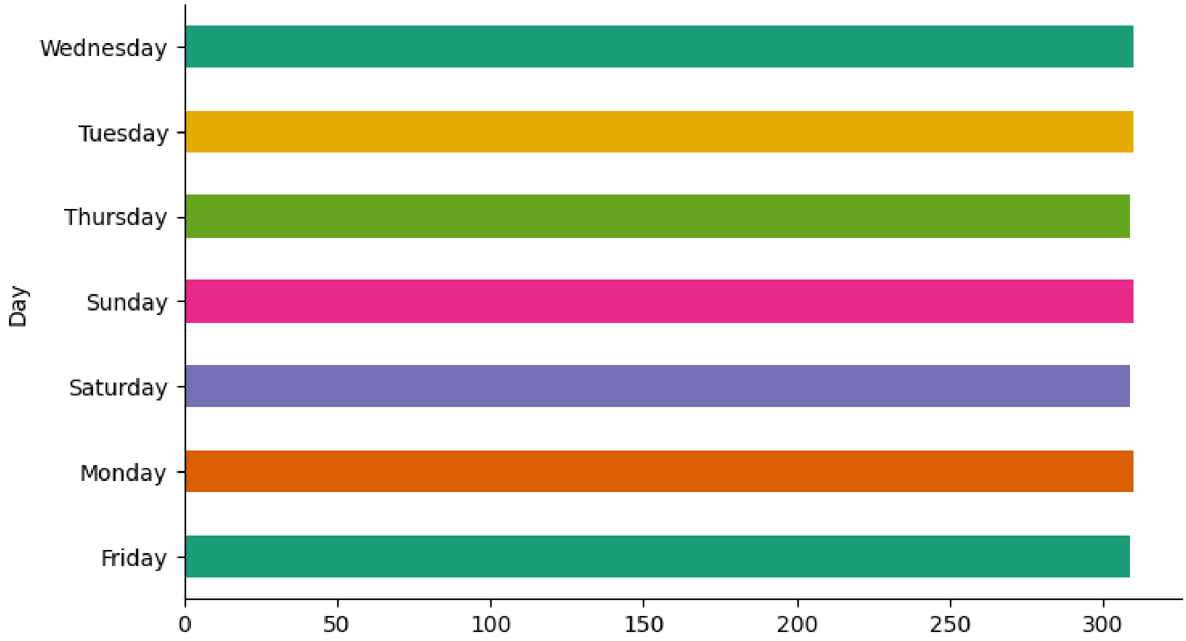
```
import numpy as np
from google.colab import autoviz

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
```



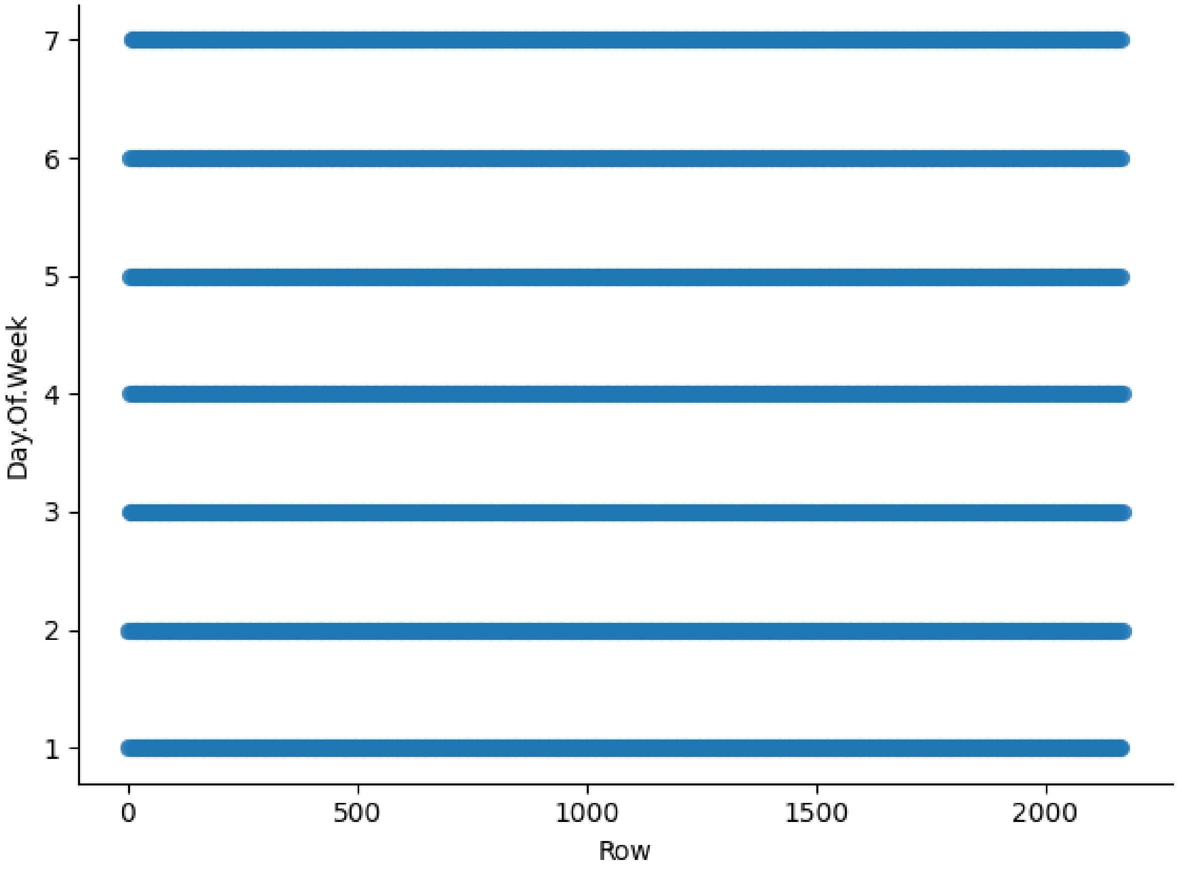
```
import numpy as np
from google.colab import autoviz

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
```

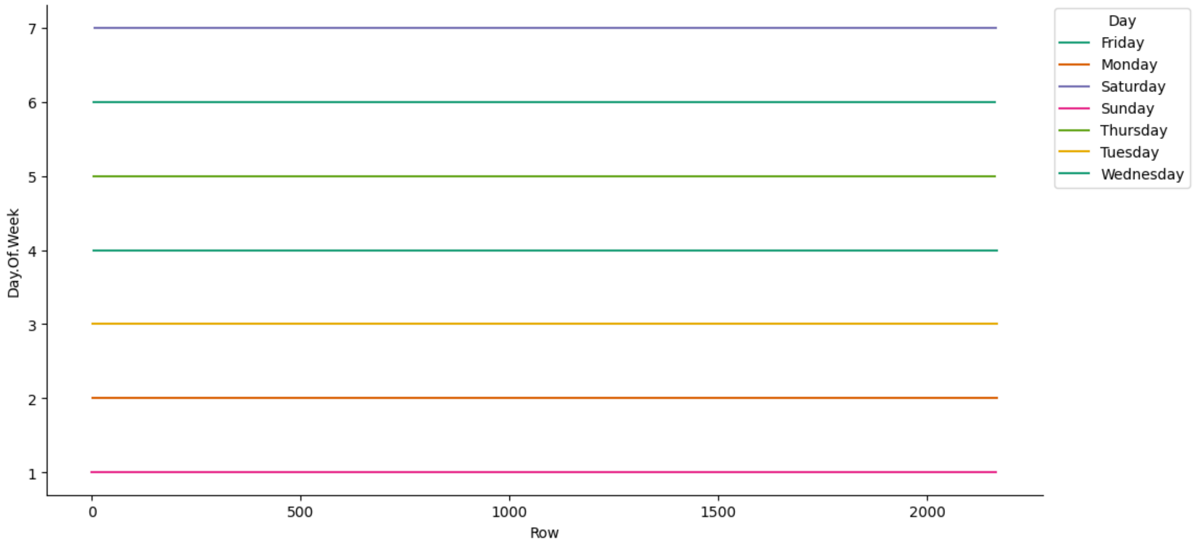


```
import numpy as np
from google.colab import autoviz

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
```



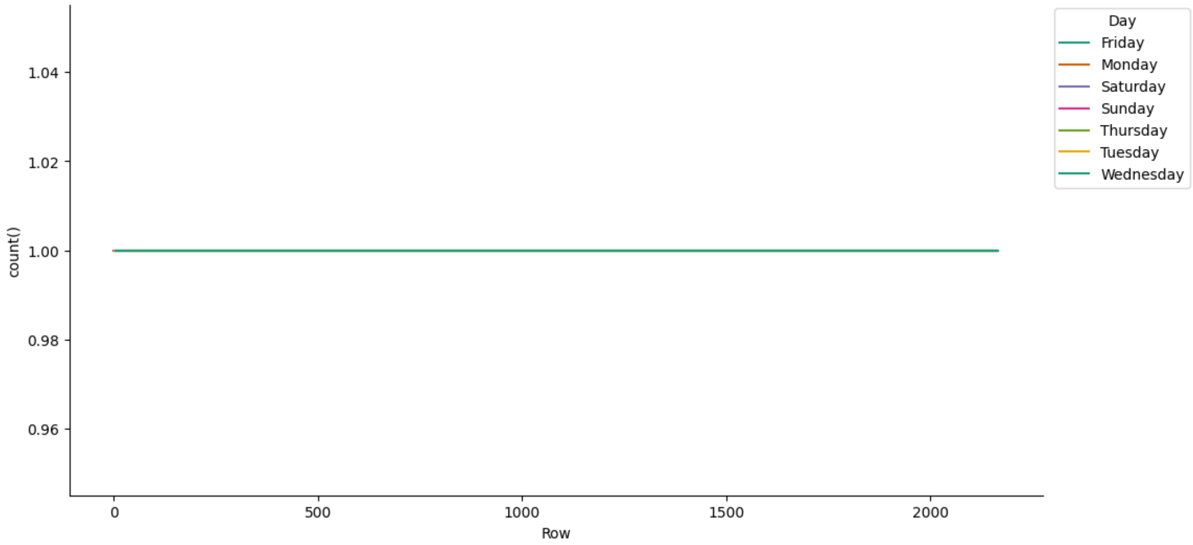
```
import numpy as np
from google.colab import autoviz

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
```

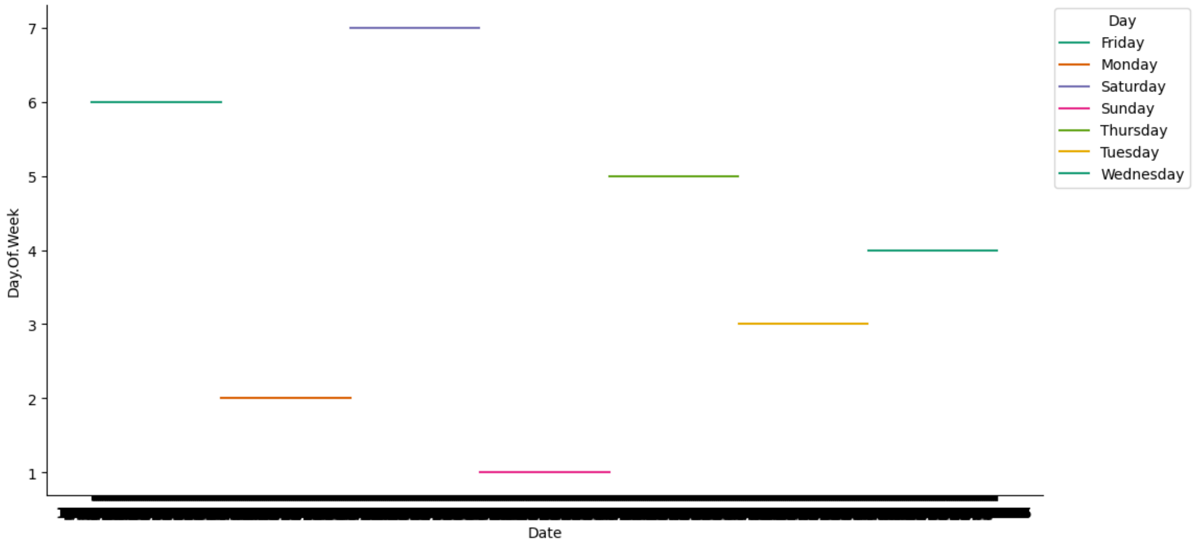


```
import numpy as np
from google.colab import autoviz

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)])
```

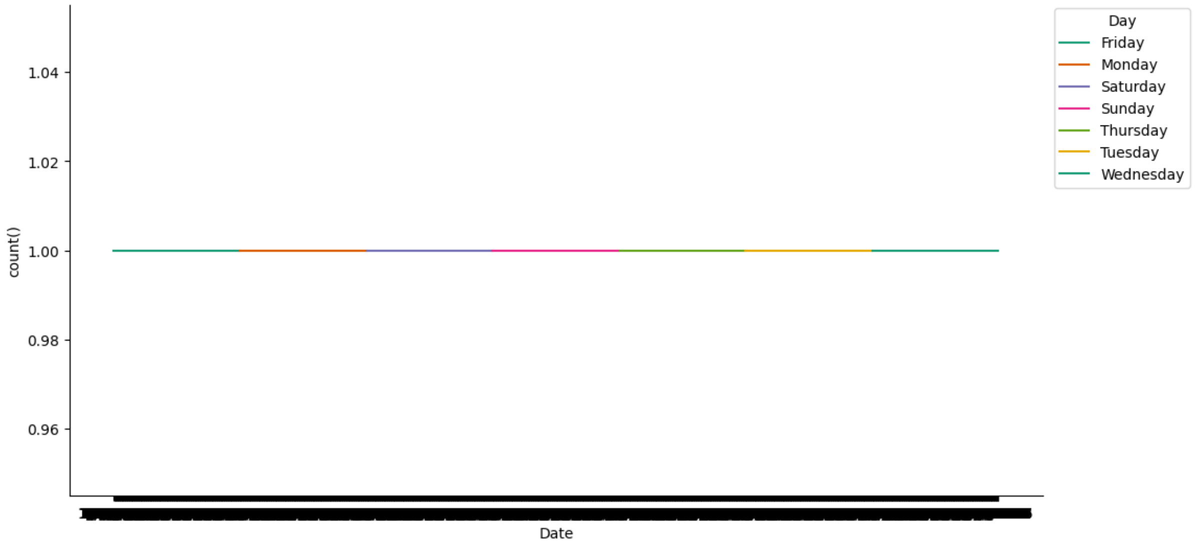
```
  fig, ax = plt.subplots(figsize=figsize,
layout='constrained')
  df = df.sort_values(timelike_colname, ascending=True)
  if series_colname:
    for i, (series_name, series) in
enumerate(df.groupby(series_colname)):
      _plot_series(series, series_name, i)
    fig.legend(title=series_colname, bbox_to_anchor=(1,
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
```



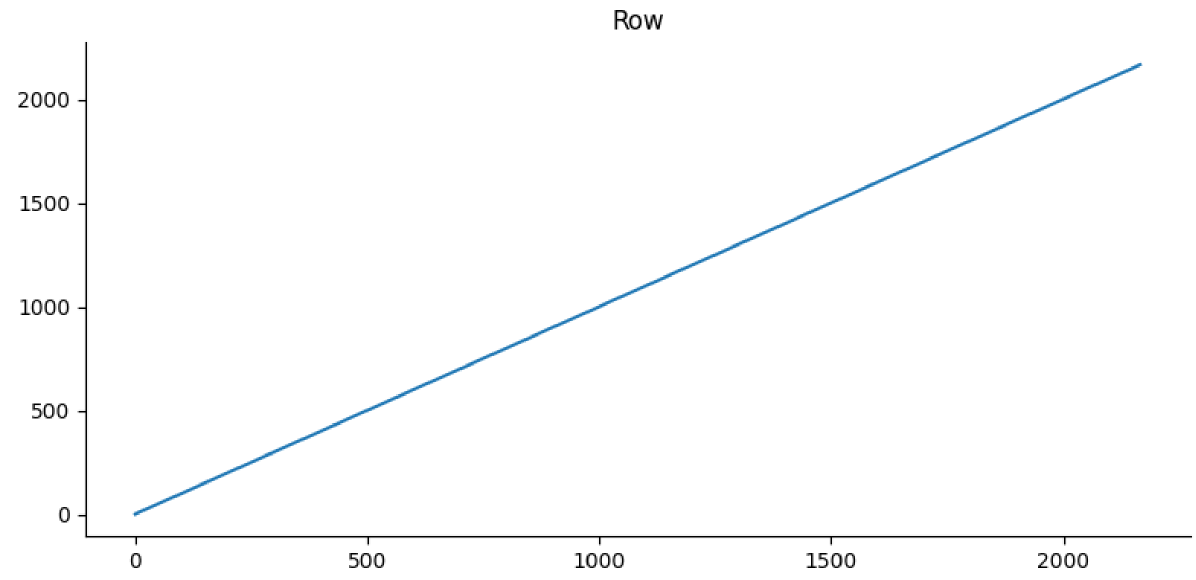
```
import numpy as np
from google.colab import autoviz

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
```



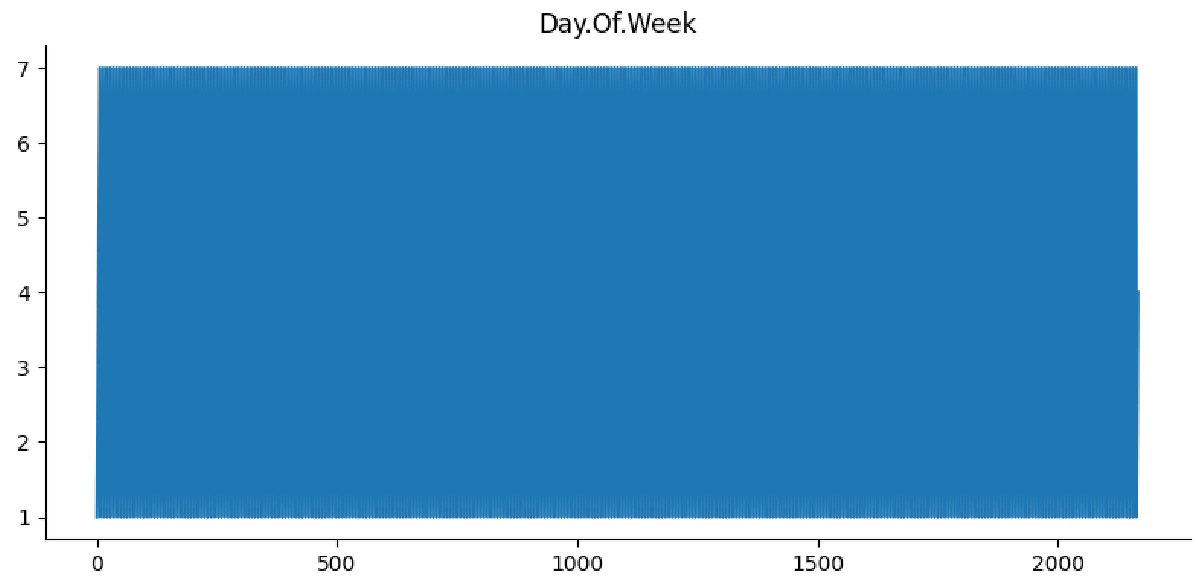
```
import numpy as np
from google.colab import autoviz

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
```



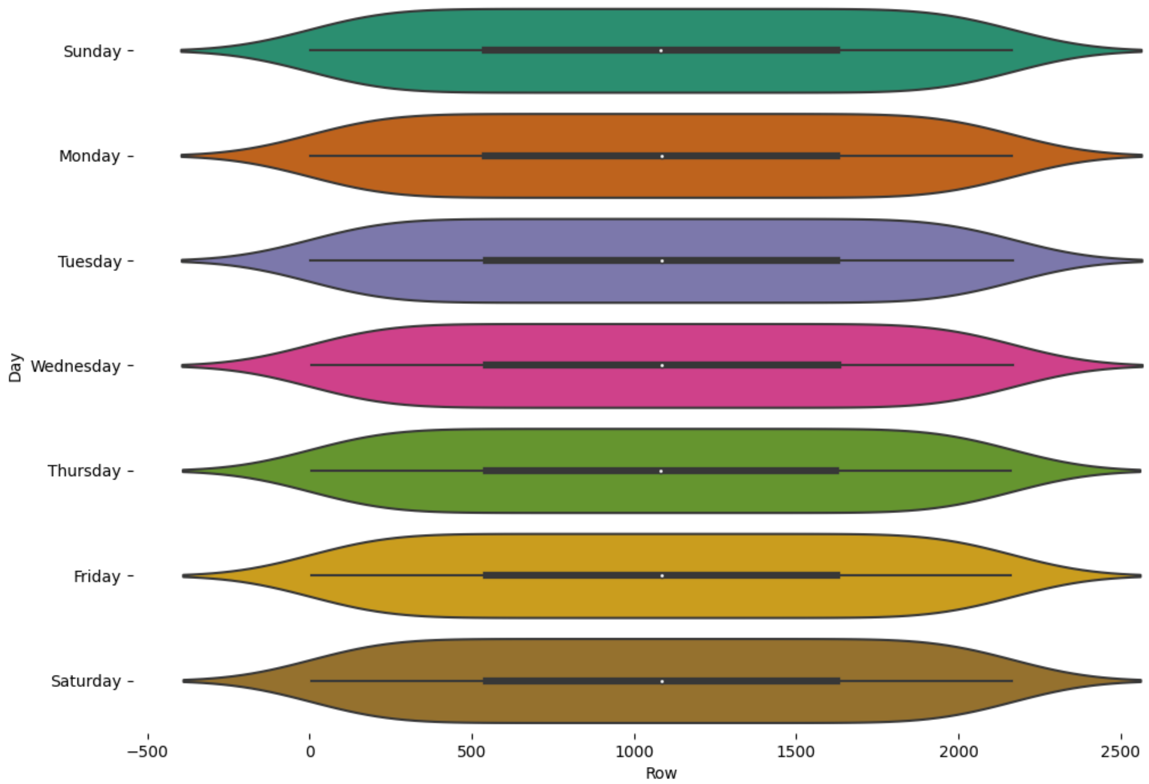
```
import numpy as np
from google.colab import autoviz

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
```



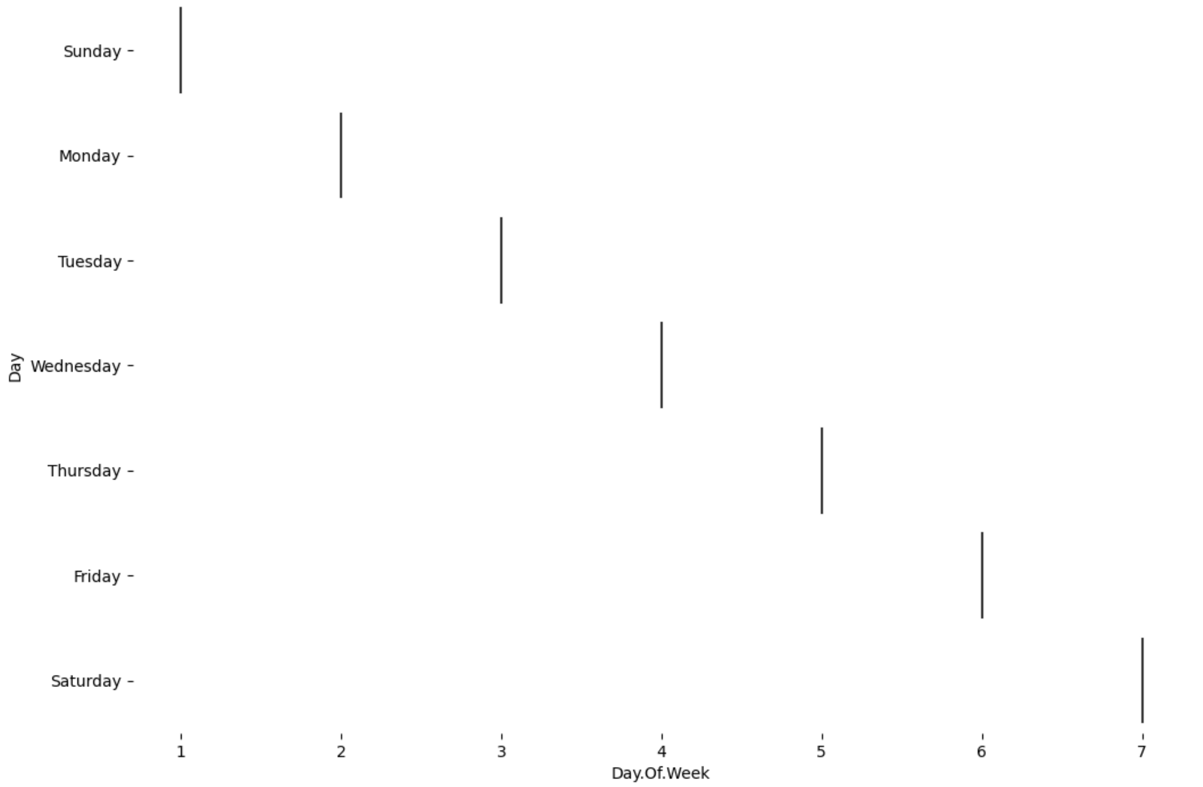
```
import numpy as np
from google.colab import autoviz

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
```



Conclusion:

In conclusion, Website traffic analysis is a critical component of managing and optimizing an online presence. It involves several fundamental aspects of website traffic analysis, including data collection, data preprocessing and initial steps towards understanding user behaviour and performance metrics.