# ***Website Traffic Analysis***

PHASE _4

Data Analytics with IBM Cognos

## Problem Definition:

The project involves analyzing website traffic data to gain insights into user behavior, popular pages, and traffic sources. The goal is to help website owners enhance the user experience by understanding how visitors interact with the site. This project encompasses defining the analysis objectives, collecting website traffic data, using IBM Cognos for data visualization, and integrating Python code for advanced analysis.

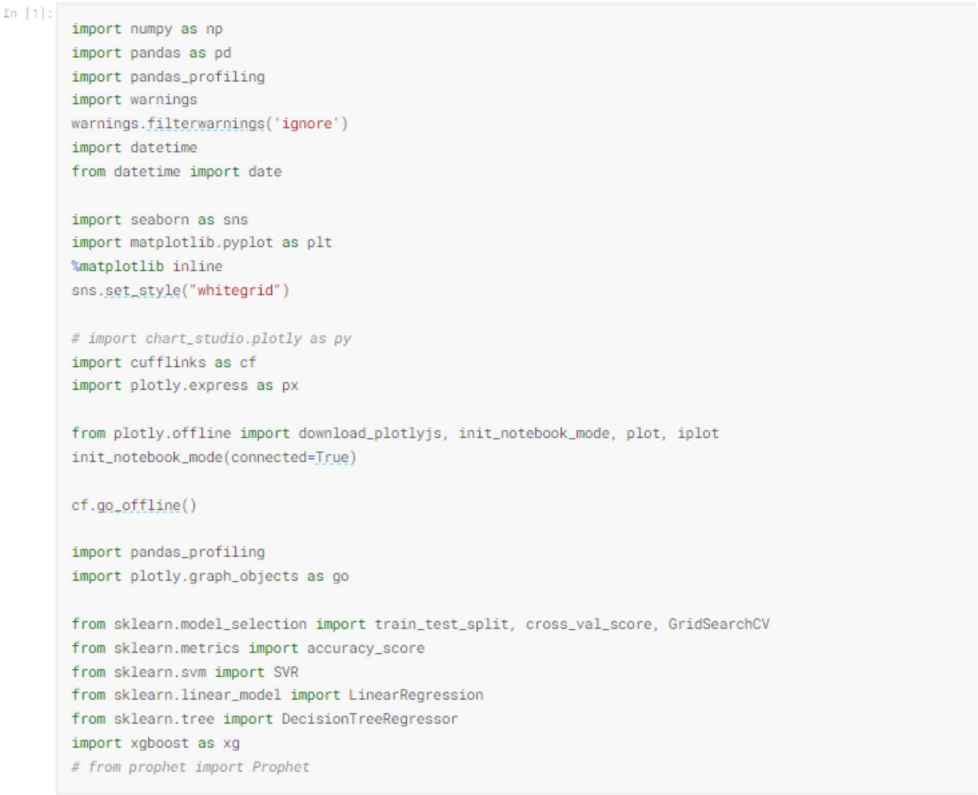
## Data Cleansing:

- Data Cleansing is the process of removing inconsistencies and incorrect values in the dataset.
- It also involves handling missing values either by removing them or assigning the average values.
- It helps to improve the efficiency of the model.

## Importing Necessary Libraries and Initializing Plotly & Cufflinks:

- Import various Python Libraries such as Numpy, Pandas ,Seaborn ,Matplotlib , Plotly Express ,Cufflinks and others. These libraries are used for data manipulation, visualization, and machine leaning.

- Set up the notebook environment for Plotly and Cufflinks to enable interactive plotting in Jupyter notebooks.

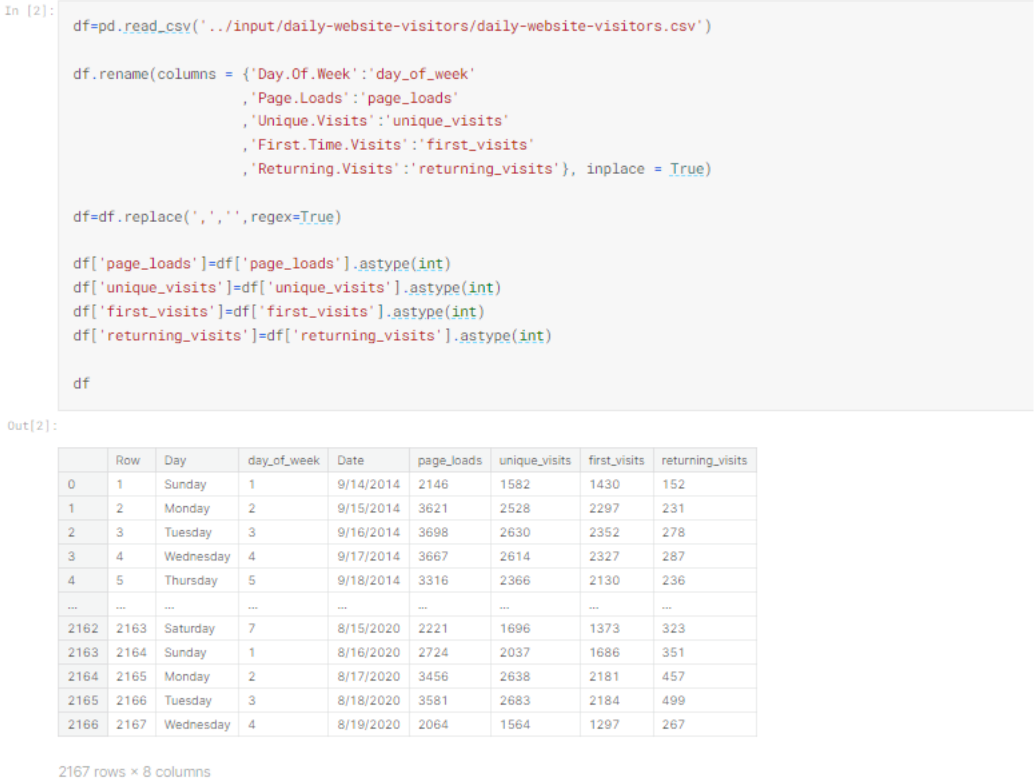


```
In [1]:
import numpy as np
import pandas as pd
import pandas_profiling
import warnings
warnings.filterwarnings('ignore')
import datetime
from datetime import date

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set_style("whitegrid")

# import chart_studio.plotly as py
import cufflinks as cf
import plotly.express as px

from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)

cf.go_offline()

import pandas_profiling
import plotly.graph_objects as go

from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
import xgboost as xg
# from prophet import Prophet
```

## Reading Data & Data Preprocessing:

- Read a CSV file containing website traffic data into a Pandas DataFrame. Ensure that the file path is correctly specified.

- Rename columns in the DataFrame for easier reference and remove commas from numeric columns. The columns 'page_loads', 'unique_visits', 'first_visits', and 'returning_visits' are converted to integer data types.



In [2]:

```
df=pd.read_csv('../input/daily-website-visitors/daily-website-visitors.csv')

df.rename(columns = {'Day.Of.Week':'day_of_week'
                    ,'Page.Loads':'page_loads'
                    ,'Unique.Visits':'unique_visits'
                    ,'First.Time.Visits':'first_visits'
                    ,'Returning.Visits':'returning_visits'}, inplace = True)

df=df.replace(',','',regex=True)

df['page_loads']=df['page_loads'].astype(int)
df['unique_visits']=df['unique_visits'].astype(int)
df['first_visits']=df['first_visits'].astype(int)
df['returning_visits']=df['returning_visits'].astype(int)

df
```

Out[2]:

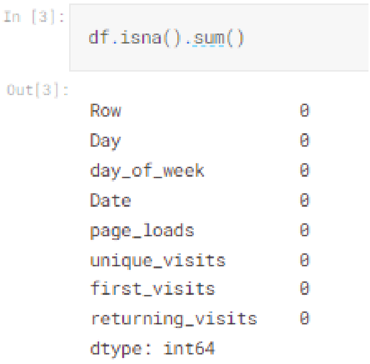
| | Row | Day | day_of_week | Date | page_loads | unique_visits | first_visits | returning_visits |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | Sunday | 1 | 9/14/2014 | 2146 | 1582 | 1430 | 152 |
| 1 | 2 | Monday | 2 | 9/15/2014 | 3621 | 2528 | 2297 | 231 |
| 2 | 3 | Tuesday | 3 | 9/16/2014 | 3698 | 2630 | 2352 | 278 |
| 3 | 4 | Wednesday | 4 | 9/17/2014 | 3667 | 2614 | 2327 | 287 |
| 4 | 5 | Thursday | 5 | 9/18/2014 | 3316 | 2366 | 2130 | 236 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2162 | 2163 | Saturday | 7 | 8/15/2020 | 2221 | 1696 | 1373 | 323 |
| 2163 | 2164 | Sunday | 1 | 8/16/2020 | 2724 | 2037 | 1686 | 351 |
| 2164 | 2165 | Monday | 2 | 8/17/2020 | 3456 | 2638 | 2181 | 457 |
| 2165 | 2166 | Tuesday | 3 | 8/18/2020 | 3581 | 2683 | 2184 | 499 |
| 2166 | 2167 | Wednesday | 4 | 8/19/2020 | 2064 | 1564 | 1297 | 267 |

2167 rows × 8 columns

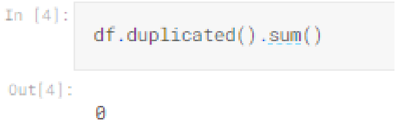
## Data Quality Check & Data Information:

- Check for missing values using `df.isna().sum()` and duplicate rows using `df.duplicated().sum()` in the DataFrame.

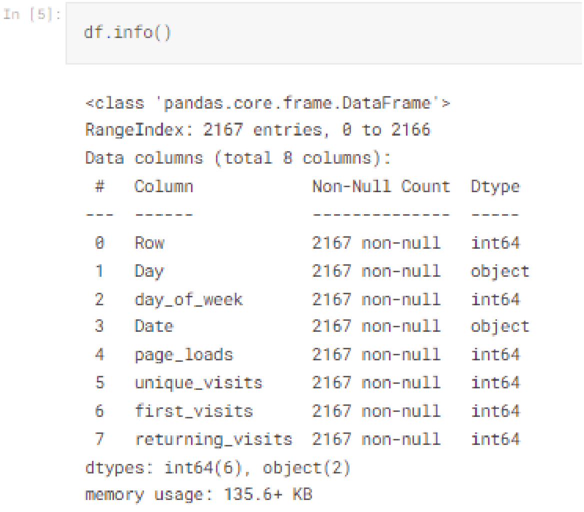
- Display information about the DataFrame using `df.info()`. This provides details on column data types, non-null counts, and memory usage.



```
In [3]: df.isna().sum()

Out[3]:
Row                 0
Day                 0
day_of_week         0
Date                0
page_loads          0
unique_visits       0
first_visits        0
returning_visits    0
dtype: int64
```



```
In [4]: df.duplicated().sum()

Out[4]:
0
```

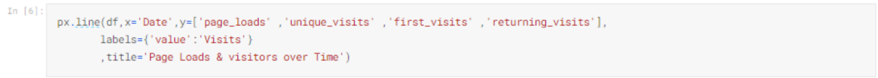


```
In [5]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2167 entries, 0 to 2166
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Row               2167 non-null   int64 
 1   Day               2167 non-null   object
 2   day_of_week       2167 non-null   int64 
 3   Date              2167 non-null   object
 4   page_loads        2167 non-null   int64 
 5   unique_visits     2167 non-null   int64 
 6   first_visits      2167 non-null   int64 
 7   returning_visits  2167 non-null   int64 
dtypes: int64(6), object(2)
memory usage: 135.6+ KB
```

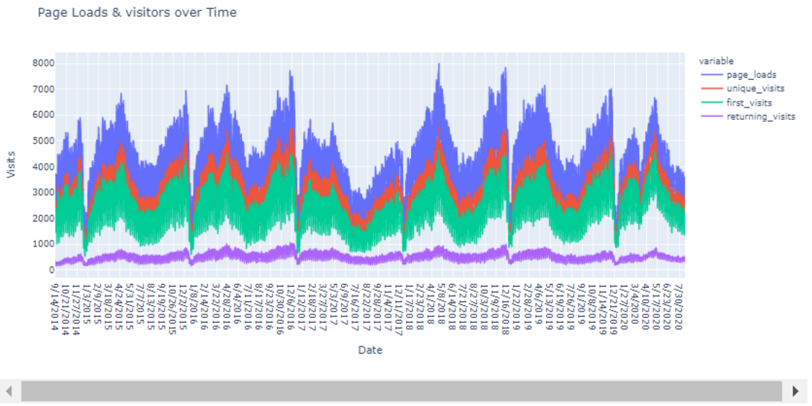
## Data Visualization:

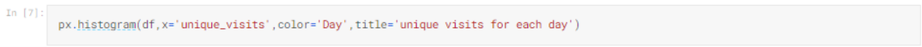
Create various data visualizations using Plotly Express and Matplotlib:

- A line plot (`fig`) showing page loads and visitors over time.
- A histogram of unique visits for each day.
- A bar chart of the sum of unique visits for each day.
- A histogram showing the sum of unique visits for each day over time.
- A bar chart showing the sum of various types of visits (page loads, unique visits, first-time visits, returning visits) for each day.
- A correlation heatmap using Seaborn, illustrating the correlation between the numeric variables.
- A scatter matrix using Plotly Express, showing scatter plots and histograms for the numeric variables.

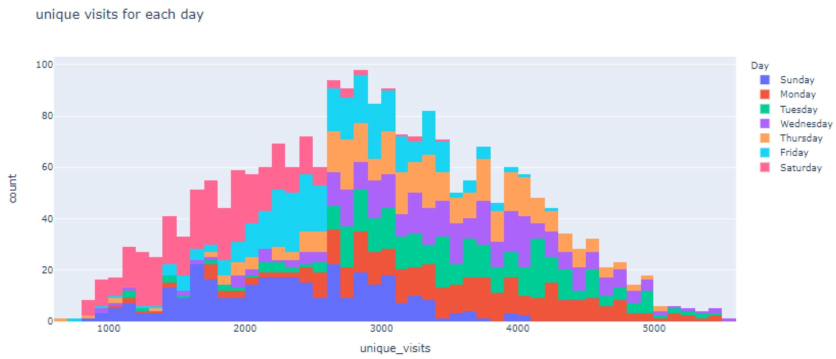


```
In [6]:
px.line(df,x='Date',y=['page_loads' ,'unique_visits' ,'first_visits' ,'returning_visits'],
        labels={'value':'Visits'}
        ,title='Page Loads & visitors over Time')
```



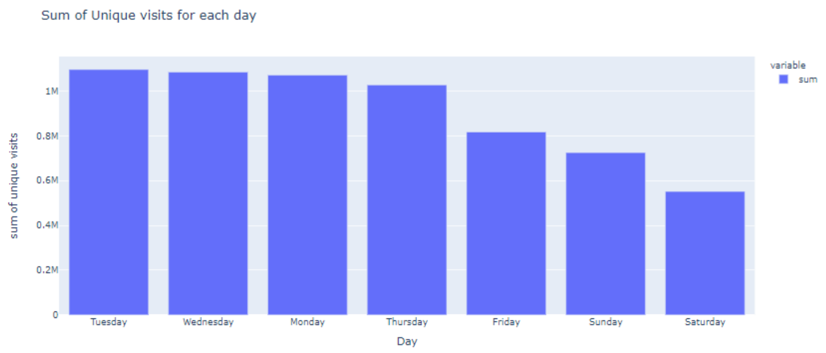


```
In [7]:
px.histogram(df,x='unique_visits',color='Day',title='unique visits for each day')
```



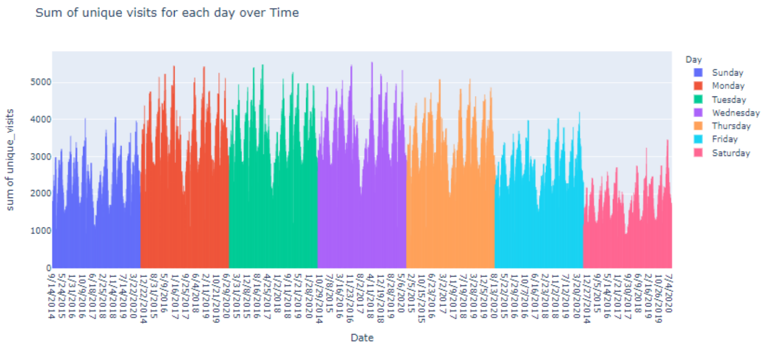
In [8]:

```
day_imp=df.groupby(['Day'])['unique_visits'].agg(['sum']).sort_values(by='sum',ascending=False)
px.bar(day_imp,labels={'value':'sum of unique visits'},title='Sum of Unique visits for each day')
```



In [9]:

```
px.histogram(df,x='Date',y='unique_visits',color='Day',title='Sum of unique visits for each day over Time')
```



In [10]:

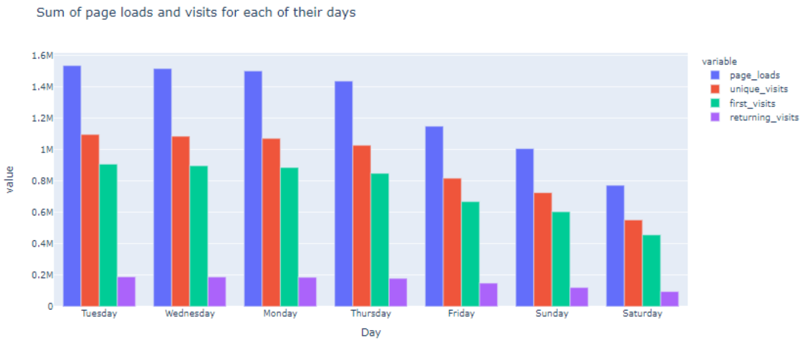
```
sums=df.groupby(['Day'])[['page_loads' ,'unique_visits' ,'first_visits' ,'returning_visits']].sum().sort_values(
    by='unique_visits',ascending=False)
sums
```

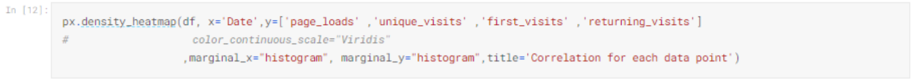
Out[10]:

| | page_loads | unique_visits | first_visits | returning_visits |
|---|---|---|---|---|
| Day | | | | |
| Tuesday | 1536154 | 1097181 | 907752 | 189429 |
| Wednesday | 1517114 | 1085624 | 897602 | 188022 |
| Monday | 1502161 | 1072112 | 886036 | 186076 |
| Thursday | 1437269 | 1028214 | 848921 | 179293 |
| Friday | 1149437 | 817852 | 668805 | 149047 |
| Sunday | 1006564 | 725794 | 604198 | 121596 |
| Saturday | 772817 | 552105 | 456449 | 95656 |

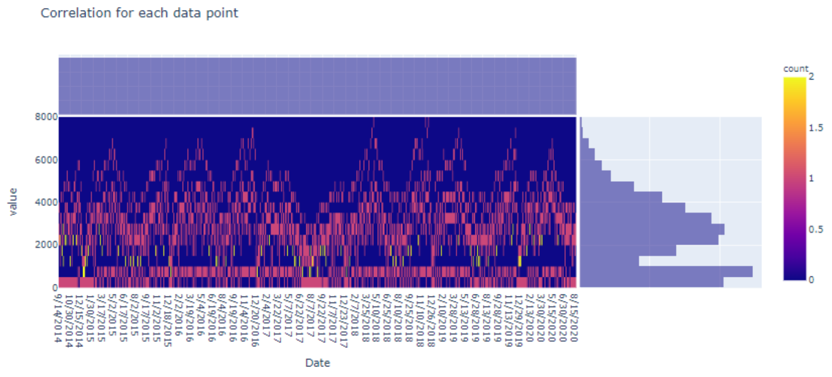
In [11]:

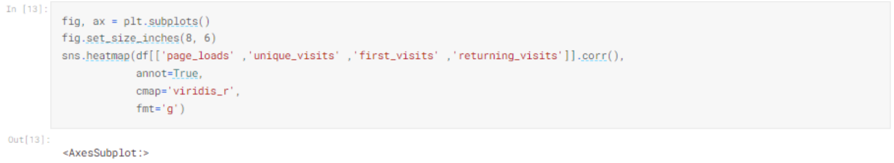
```
px.bar(sums,barmode='group',title='Sum of page loads and visits for each of their days')
```





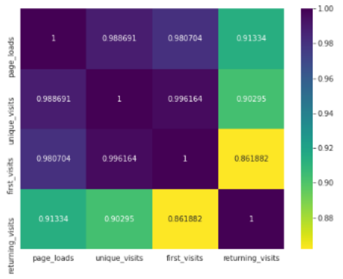
In [12]:
```
px.density_heatmap(df, x='Date',y=['page_loads' ,'unique_visits' ,'first_visits' ,'returning_visits']
#                   color_continuous_scale="Viridis"
                   ,marginal_x="histogram", marginal_y="histogram",title='Correlation for each data point')
```

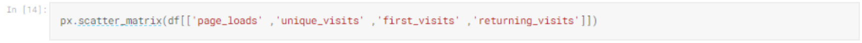

Correlation for each data point



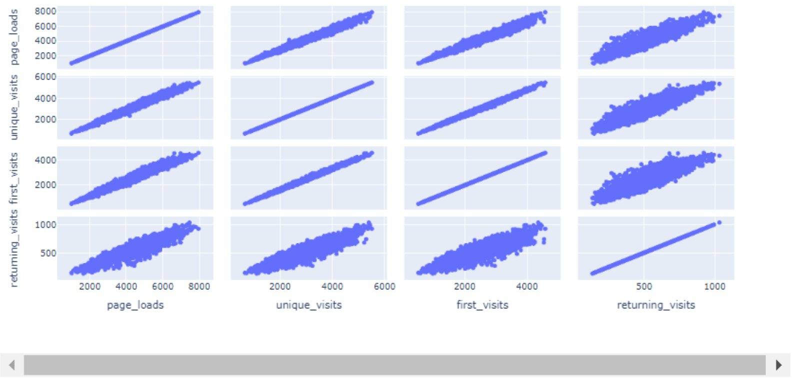
In [13]:
```
fig, ax = plt.subplots()
fig.set_size_inches(8, 6)
sns.heatmap(df[['page_loads' ,'unique_visits' ,'first_visits' ,'returning_visits']].corr(),
            annot=True,
            cmap='viridis_r',
            fmt='g')
```

Out[13]:
```
<AxesSubplot:>
```



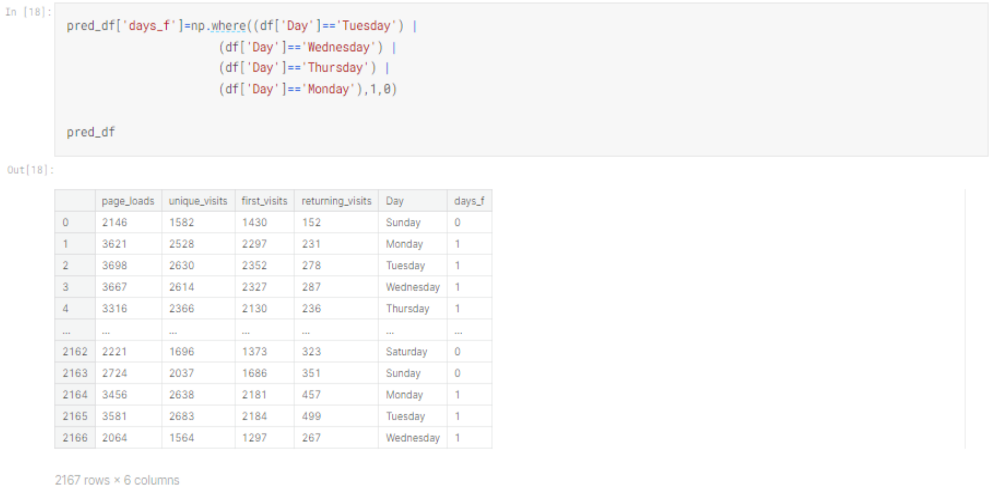


In [14]:
```
px.scatter_matrix(df[['page_loads' ,'unique_visits' ,'first_visits' ,'returning_visits']])
```



## Feature Engineering:

- Create a new DataFrame `pred_df` as a copy of the original data, and add a new binary feature 'days_f' based on the day of the week. This feature is set to 1 for Monday, Tuesday, Wednesday, and Thursday, and 0 otherwise.



```
In [18]:
pred_df['days_f']=np.where((df['Day']=='Tuesday') |
                    (df['Day']=='Wednesday') |
                    (df['Day']=='Thursday') |
                    (df['Day']=='Monday'),1,0)

pred_df
```

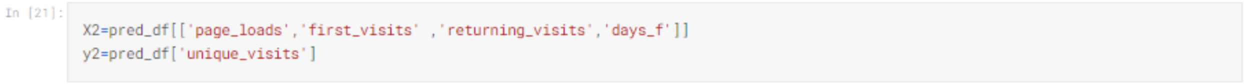
Out[18]:

| | page_loads | unique_visits | first_visits | returning_visits | Day | days_f |
|---|---|---|---|---|---|---|
| 0 | 2146 | 1582 | 1430 | 152 | Sunday | 0 |
| 1 | 3621 | 2528 | 2297 | 231 | Monday | 1 |
| 2 | 3698 | 2630 | 2352 | 278 | Tuesday | 1 |
| 3 | 3667 | 2614 | 2327 | 287 | Wednesday | 1 |
| 4 | 3316 | 2366 | 2130 | 236 | Thursday | 1 |
| ... | ... | ... | ... | ... | ... | ... |
| 2162 | 2221 | 1696 | 1373 | 323 | Saturday | 0 |
| 2163 | 2724 | 2037 | 1686 | 351 | Sunday | 0 |
| 2164 | 3456 | 2638 | 2181 | 457 | Monday | 1 |
| 2165 | 3581 | 2683 | 2184 | 499 | Tuesday | 1 |
| 2166 | 2064 | 1564 | 1297 | 267 | Wednesday | 1 |

2167 rows × 6 columns

## Data Modeling Preparation:

- Define the feature matrix `X` and the target variable `y` for machine learning modeling. The features include 'page_loads', 'first_visits', 'returning_visits', and 'days_f'. The target variable is 'unique_visits'.



```
In [21]:
X2=pred_df[['page_loads','first_visits' ,'returning_visits','days_f']]
y2=pred_df['unique_visits']
```

## Conclusion:

In this website traffic analysis project, we began by importing and manipulating website traffic data using Python. We performed essential data preprocessing tasks, such as renaming columns, removing commas, and checking for missing values and duplicates. We then delved into exploratory data analysis, creating a range of visualizations to gain insights into website visitor behavior. This included visualizations like line plots, histograms, and correlation heatmaps to understand the relationship between variables. Furthermore, we introduced feature engineering by categorizing days of the week as a binary feature, 'days_f,' to aid future machine learning tasks. This comprehensive analysis lays the groundwork for making informed decisions and predictions regarding website traffic trends and visitor patterns.

In the next phase of this project, we can build predictive models, such as time series forecasting or regression, to anticipate website traffic and optimize user experiences. These models can leverage the insights gained from our exploratory analysis and feature engineering to make data-driven decisions aimed at improving the website's performance and user engagement. Overall, this project equips us with the knowledge and tools to make strategic improvements to our website based on a thorough understanding of its traffic patterns and trends.

| | |
|---|---|
| AFSAL P A | (au720921243004) |
| ASWIN GEORGE | (au720921243015) |
| HARI KRISHNA K R | (au720921243026) |
| ROSHAN ROY | (au720921243047) |

JCT COLLEGE OF ENGINEERING AND TECHNOLOGY