Website Traffic Analysis – Phase 4

Development Part 2

TEAM MEMBERS :

KAMARAJ	**2021115047**
KARTHIKA	**2021115049**
KARTHIKEYAN	**2021115050**
KAVIYA	**2021115051**
JEEVESH	**2021115312**

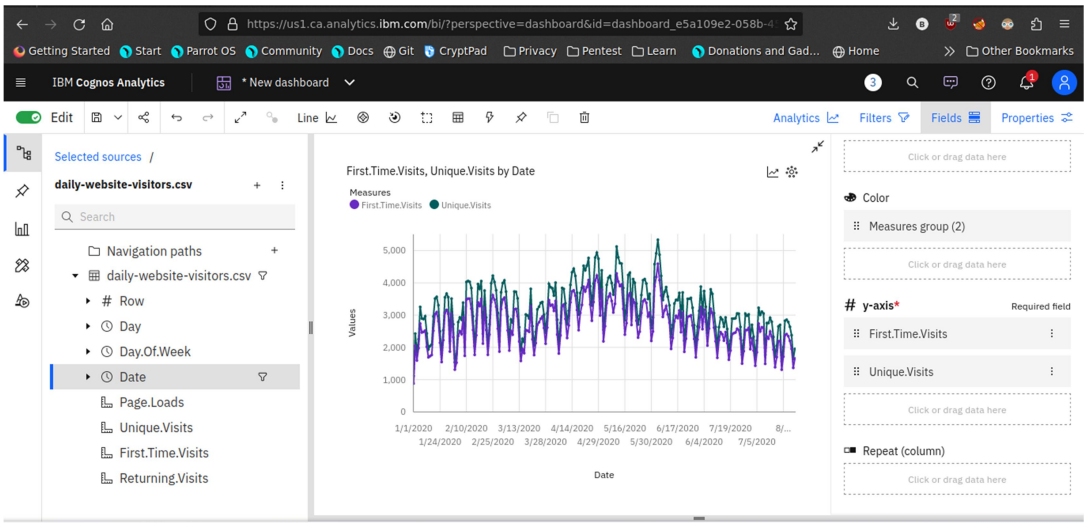
OBJECTIVES:

In this phase defines the visualization using IBM cognos, Integrating python code for advanced analysis and creating interactive dashboard. Use Python libraries and machine learning models for Complex analysis.

Visualization in IBM Cognos:

1.)Line plot:

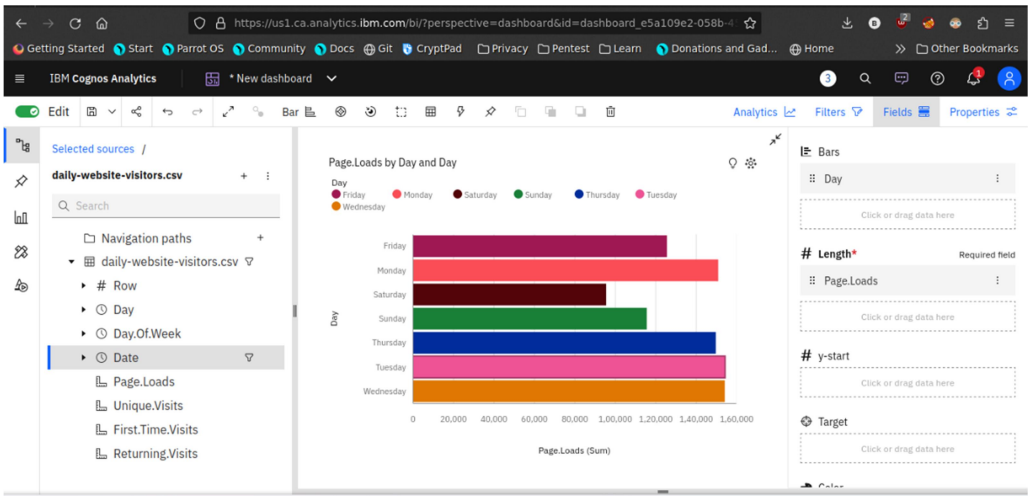
In this line plot X_axis are dates and Y_axis are First time visits and Unique visits



2.)Barchart:

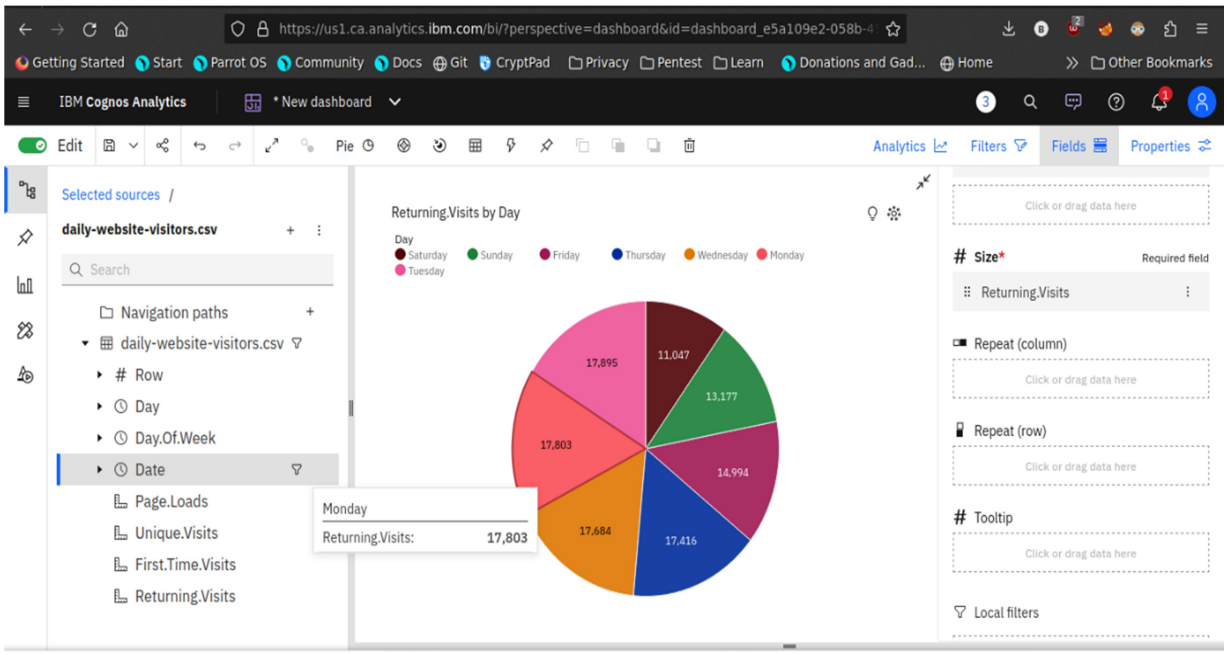
In this Bar chart the bars represent the ‘days in week’ and length defines ‘Page.Loads’

It helpful to visualize the maximum pageloads occurs on a day



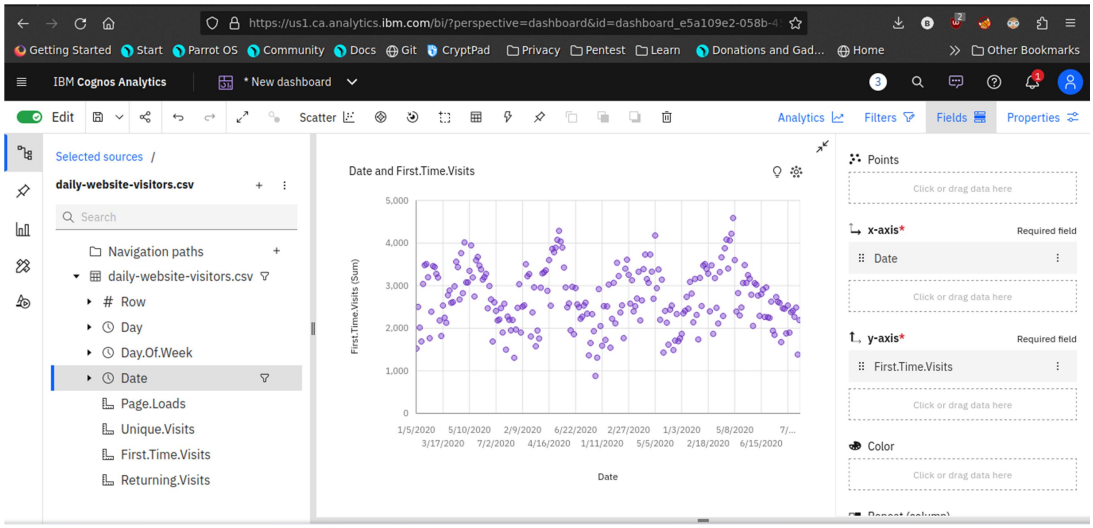
3.)Piechart:

This is same as a bar chart. it helpful to analyze the Returing visits occurs on a particular day

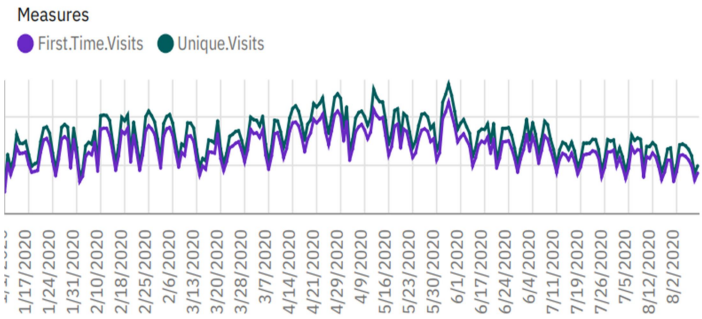


4.)Scatter plot:

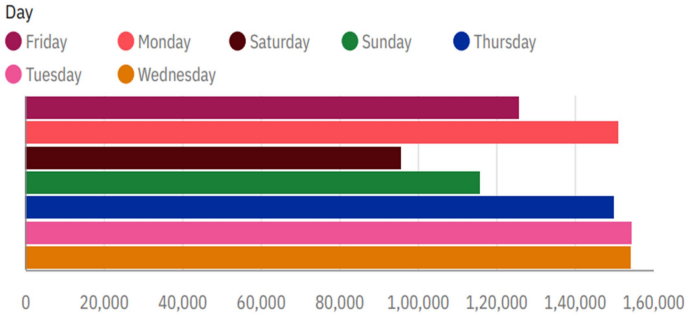
It is used to display the relationship between two variables and observe the nature of the relationship. The relationships observed can either be positive or negative, non-linear or linear



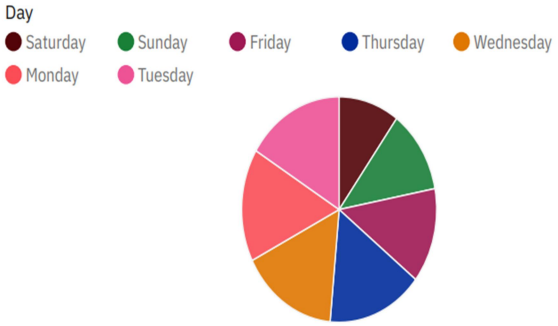
First.Time.Visits, Unique.Visits by Date



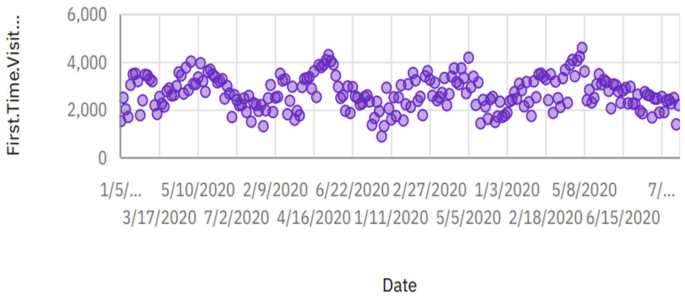
Page.Loads by Day and Day



Returning.Visits by Day



Date and First.Time.Visits



Now the visualization phase where over. lets start analyze the dataset using Python libraries use machine learning models for predictive analysis.

In [83]:
```
import numpy as np
import pandas as pd
from pydantic_settings import BaseSettings
import warnings
from warnings import warn
import datetime
from datetime import date

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

import plotly.express as px
from sklearn.linear_model import LinearRegression
```

In [10]:
```
import pandas as pd
df=pd.read_csv("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[10]:

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

2167 rows × 8 columns

In [11]:
```
df.isna().sum()
```

Out[11]:
```
Row                0
Day                0
day_of_week        0
Date               0
page_loads         0
unique_visits      0
first_visits       0
returning_visits   0
dtype: int64
```

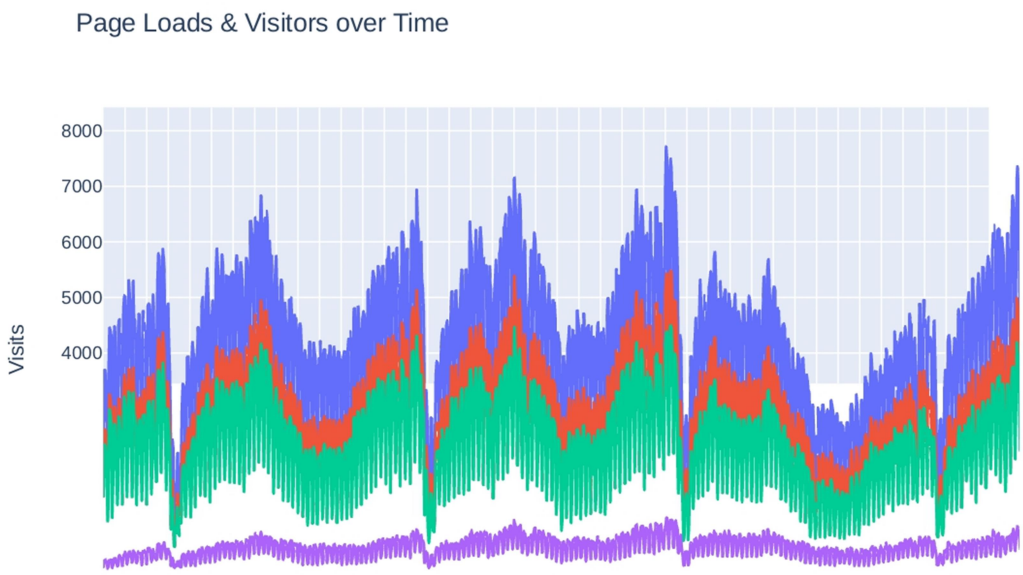
In [12]:
```
df.duplicated().sum()
```

Out[12]:
```
0
```

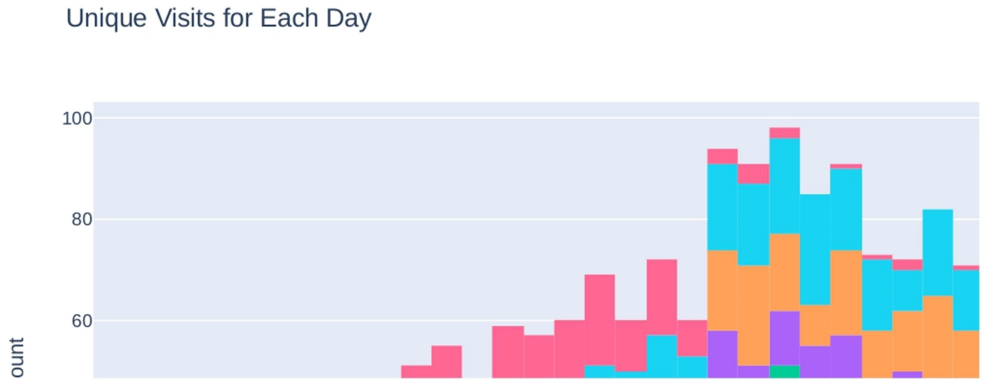
In [13]:
```
df.info()
```

```
<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
```

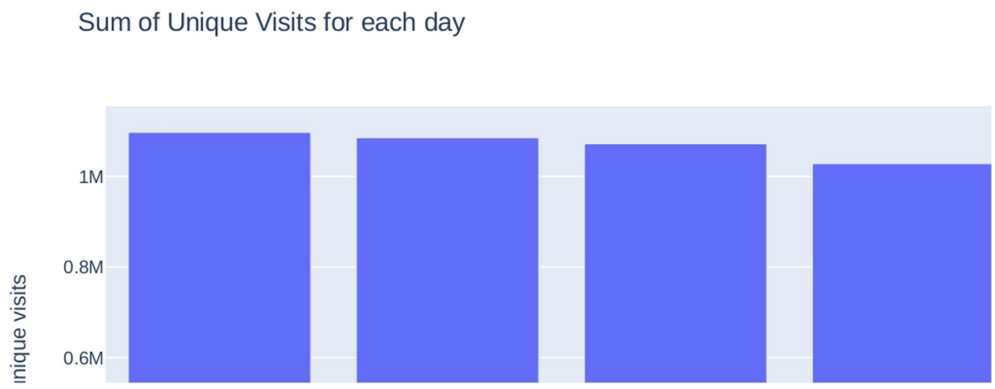
In [16]:
```
px.line(df, x='Date', y=['page_loads', 'unique_visits', 'first_visits', 'returning_vi
labels={'value': 'Visits'},
title='Page Loads & Visitors over Time')
```



In [17]:
```
px.histogram(df,x='unique_visits',color='Day',title='Unique Visits for Each Day')
```



In [18]:
```
day_imp=df.groupby(['Day'])['unique_visits'].agg(['sum']).sort_values(by='sum',ascend
px.bar(day_imp,labels={'value':'sum of unique visits'},title='Sum of Unique Visits fo
```



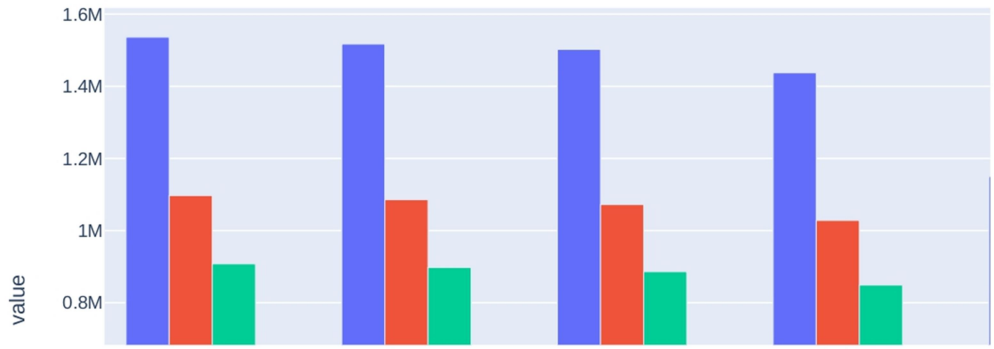
In [19]:
```
px.histogram(df,x='Date',y='unique_visits',color='Day',title='Sum of Unique vists for
```

In [20]:
```
sums=df.groupby(['Day'])[['page_loads' ,'unique_visits' ,'first_visits' ,'returning_v
    by='unique_visits',ascending=False)
sums
```

Out[20]:

Day	page_loads	unique_visits	first_visits	returning_visits
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 [21]:
```
px.bar(sums,barmode='group')
```



In [19]:
```
pip install seaborn
```

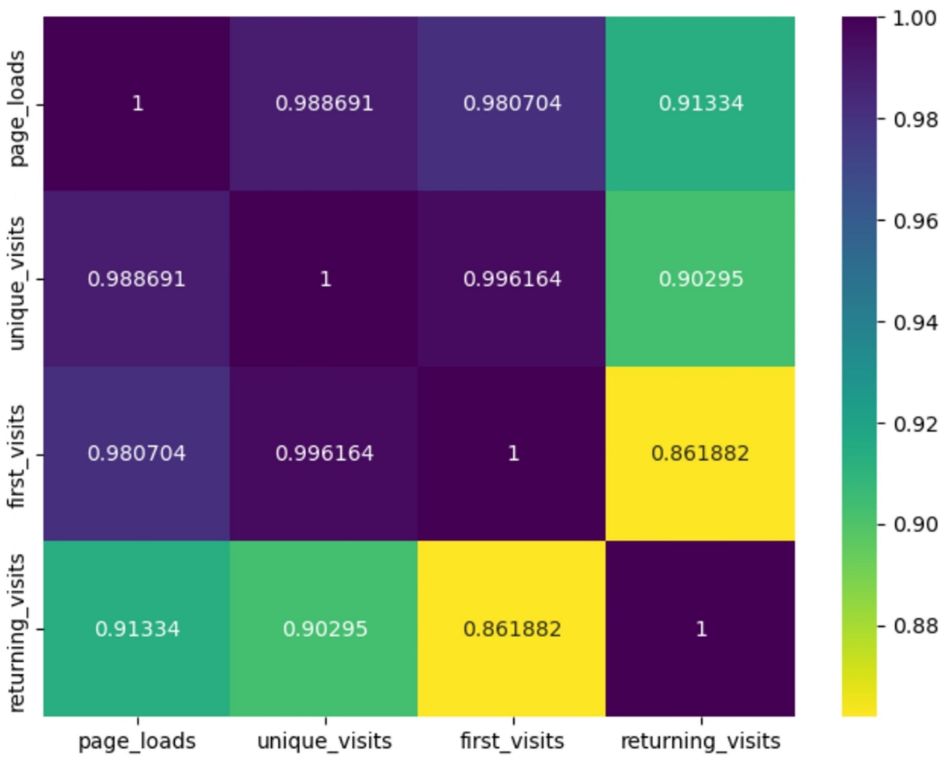
```
Requirement already satisfied: seaborn in /home/badhrinathan/.local/lib/python3.9/site
-packages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in /home/badhrinathan/.local/lib/p
ython3.9/site-packages (from seaborn) (1.24.3)
Requirement already satisfied: pandas>=0.25 in /home/badhrinathan/.local/lib/python3.9
/site-packages (from seaborn) (2.0.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in /usr/lib/python3/dist-packag
es (from seaborn) (3.3.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /home/badhrinathan/.local/lib
/python3.9/site-packages (from pandas>=0.25->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/lib/python3/dist-packages (from pa
ndas>=0.25->seaborn) (2021.1)
Requirement already satisfied: tzdata>=2022.1 in /home/badhrinathan/.local/lib/python
3.9/site-packages (from pandas>=0.25->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (from python
-dateutil>=2.8.2->pandas>=0.25->seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [24]:
```
import matplotlib.pyplot as plt
```

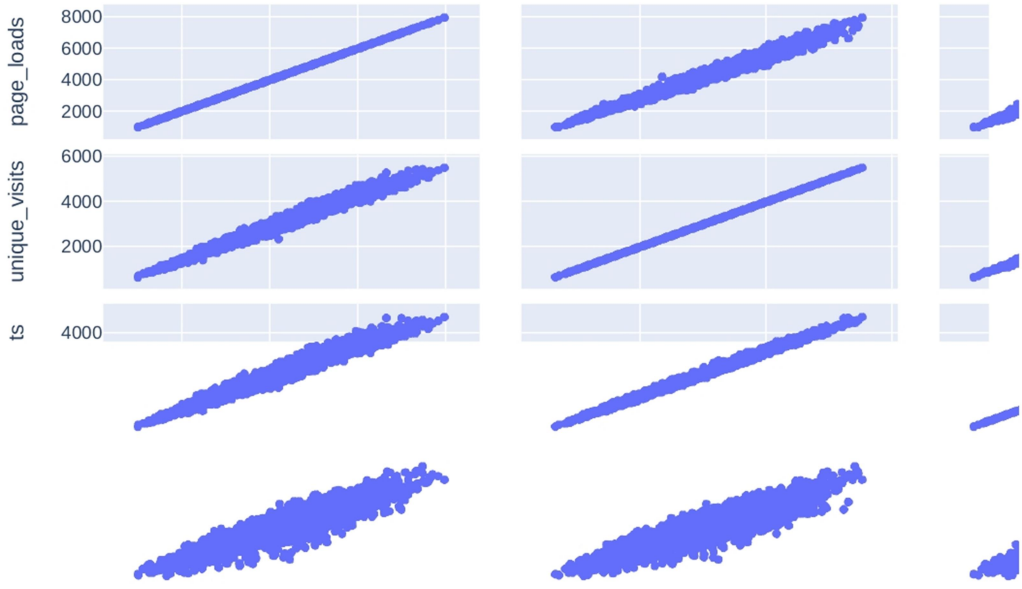
In [25]:
```
import seaborn as sns
```

In [26]:
```
fig, ax = plt.subplots()
fig.set_size_inches(8, 6)
sns.heatmap(df[['page_loads' ,'unique_visits' ,'first_visits' ,'returning_visits']].c
            annot=True,
            cmap='viridis_r',
            fmt='g')
```

Out[26]:
```
<AxesSubplot:>
```



In [27]:
```
px.scatter_matrix(df[['page_loads' ,'unique_visits' ,'first_visits' ,'returning_visit
```



In [14]:
```
import plotly.express as px
import numpy as np
```

In [15]:
```
df['days_f']=np.where((df['Day']=='Tuesday') |
                      (df['Day']=='Wednesday') |
                      (df['Day']=='Thursday') |
                      (df['Day']=='Monday'),1,0)
df
```

Out[15]:

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

2167 rows × 9 columns

In [28]:
```
X2=df[['page_loads','first_visits' ,'returning_visits','days_f']]
y2=df['unique_visits']
```

In [59]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X2, y2, test_size=0.2)
```

In [60]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[60]:
```
▾ LinearRegression
LinearRegression()
```

In [61]:
```
regressor.coef_
```

Out[61]:
```
array([-3.88376916e-15,  1.00000000e+00,  1.00000000e+00, -3.99733415e-14])
```

In [62]:
```
regressor.intercept_
```

Out[62]:
```
1.4097167877480388e-11
```

In [63]:
```
y_pred=regressor.predict(X_test)
X_pred=regressor.predict(X_train)
```

In [64]:
```
y_pred
```

Out[64]:
```
array([1011., 4340., 3552., 2293., 3783., 2919., 2031., 3505., 3621.,
       4823., 4474., 2857., 4382., 3493., 4399., 2818., 2238., 4322.,
       4724., 1981., 2325., 2702., 3830., 1999., 4674., 2622., 2898.,
       2604., 2566., 2856., 3767., 4345., 3611., 1281., 2477., 2442.,
       3008., 2621., 4866., 2681., 1791., 3075., 1658., 1928., 2591.,
       1864., 3019., 2643., 2685., 2664., 3925.,  948., 1005., 3215.,
       3194., 3517., 3441., 2397., 2269., 3085., 1981., 3557., 3838.,
       3457., 4960., 3287., 3808., 3689., 2496., 4119., 3469., 1976.,
       2870., 3996., 1407., 3711., 2913., 1177., 3178., 3784., 1502.,
       2199., 3770., 1609., 2658.,  876., 2274., 3604.,  923., 3247.,
       1312., 3601., 2912., 2394., 1376., 3967., 4975., 1223., 4102.,
       3331., 2704., 3780., 2816., 2892., 1977., 3838., 2277., 1577.,
       4520., 2850., 3064., 2366., 4176., 4159., 3827., 1755., 3381.,
       1054., 1567., 2211., 1947., 3011., 4216., 3002., 2508., 3021.,
       2805., 3244., 2860., 3219., 3904., 3236., 2153., 3345., 3793.,
       3812., 2800., 1788., 3372., 2410., 1099., 2979., 3119., 2906.,
       2169., 1777., 1155., 3787., 1015., 2743., 2515., 2145., 1564.,
        930., 3387., 4962., 4983., 3535., 1423., 1319., 1690., 4968.,
       1987., 2026., 1834., 3877., 2520., 3447., 3766., 2568., 1783.,
       2476., 3681., 2640., 3279., 2760., 4377., 3232., 3406., 3761.,
       3194., 2438., 2952., 3613., 5181., 3908., 2530., 2618., 3266.,
       2907., 2078., 4019., 2423., 3324., 3500., 2062., 3277., 3474.,
       4092., 4105., 4132., 2598., 1884., 4547., 3768., 4776., 3932.,
       2208., 3350., 2122.,  667., 2346., 3946., 4672., 1307., 3499.,
       2808., 2143., 1664., 2457., 3665., 2209., 2784., 3454., 2770.,
       3635., 2049., 1876., 1548., 2595., 1319., 4331., 1911., 1410.,
       3047., 2364., 4596., 3791., 4086., 4512., 2299., 4207., 4735.,
       1358., 3444., 3486., 4160., 4184., 3966., 4115., 1380., 3951.,
       2140., 2111., 2127., 2620., 1957., 1168., 3039., 3760., 4226.,
       2265., 3369., 3147., 2239., 1112., 3297., 3189., 1316., 2614.,
       1276., 3943., 3139., 3894., 3007., 2207., 1478., 2954., 1447.,
       2268., 1823., 4073., 3430., 1538., 2015., 2468., 2613., 2351.,
       3145., 4016., 2783., 2518., 2002., 3964., 1902., 2422., 1039.,
       1765., 2911., 2628., 2087., 3533., 4533., 2032., 2853., 2704.,
       2289., 2947., 2791., 1678., 2381., 4006.,  934., 2540., 4020.,
       2724., 3164., 3035., 2539., 1626., 1790., 4031., 3346., 3056.,
       3307., 2281., 4240., 2804., 2231., 3339., 1993., 2838., 2084.,
       1618., 4092., 1649., 3259., 3144., 1192., 2860., 2750., 4127.,
       3682., 3707., 3668., 2769., 3804., 4202., 4134., 4542., 4038.,
       1721., 3988., 2962., 3493., 3241., 5124., 2579., 2827., 1710.,
       3364., 2264., 1913., 2275., 1677., 4112., 1472., 2864., 2715.,
       3437., 3500., 2716., 2976., 4086., 3284., 4917., 2400., 3733.,
       3348., 2625., 2105., 1517., 3357., 3245., 3593., 2762., 2291.,
       1901., 3390., 4520., 3265., 2881., 3321., 1370., 3307., 3597.,
       1277., 3669., 3309., 4266., 2970., 2971., 3762., 3409., 2825.,
       1430., 1211., 4042., 3003., 3388., 3175., 3703., 4106., 4674.,
       2946., 3236., 4738., 4888.,  892., 2476., 3582., 1760., 2818.,
       3099., 4059., 3082., 2845., 1867., 3954., 5267., 3949., 2743.,
       3832., 2539.])
```

In [65]:
```
X_pred
```

Out[65]:
```
array([4178., 3934., 3341., ..., 3340., 3708., 2546.])
```

In [78]:
```
X_pred
```

Out[78]:
```
array([4178., 3934., 3341., ..., 3340., 3708., 2546.])
```

In [72]:
```
X_train
```

Out[72]:

	page_loads	first_visits	returning_visits	days_f
415	5624	3504	674	1
388	5400	3350	584	1
723	4619	2727	614	1
455	4132	2548	516	0
983	4154	2400	596	1
...	...	...	...	...
1524	5066	2863	634	0
1978	5004	3188	635	1
621	4793	2692	648	0
1345	5499	3005	703	1
623	3687	2067	479	0

1733 rows × 4 columns

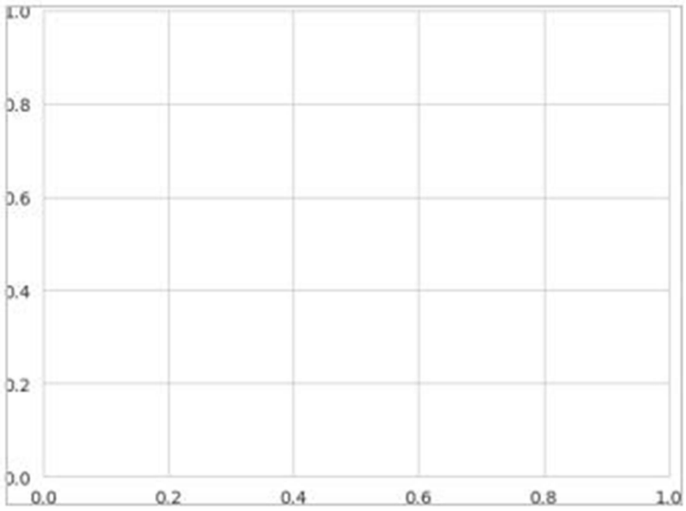
In [73]:
```
y_train
```

Out[73]:
```
415     4178
388     3934
723     3341
455     3064
983     2996
        ...
1524    3497
1978    3823
621     3340
1345    3708
623     2546
Name: unique_visits, Length: 1733, dtype: int64
```

In [75]:
```
y_pred
```

Out[75]:
```
array([1011., 4340., 3552., 2293., 3783., 2919., 2031., 3505., 3621.,
       4823., 4474., 2857., 4382., 3493., 4399., 2818., 2238., 4322.,
       4724., 1981., 2325., 2702., 3830., 1999., 4674., 2622., 2898.,
       2604., 2566., 2856., 3767., 4345., 3611., 1281., 2477., 2442.,
       3008., 2621., 4866., 2681., 1791., 3075., 1658., 1928., 2591.,
       1864., 3019., 2643., 2685., 2664., 3925.,  948., 1005., 3215.,
       3194., 3517., 3441., 2397., 2269., 3085., 1981., 3557., 3838.,
       3457., 4960., 3287., 3808., 3689., 2496., 4119., 3469., 1976.,
       2870., 3996., 1407., 3711., 2913., 1177., 3178., 3784., 1502.,
       2199., 3770., 1609., 2658.,  876., 2274., 3604.,  923., 3247.,
       1312., 3601., 2912., 2394., 1376., 3967., 4975., 1223., 4102.,
       3331., 2704., 3780., 2816., 2892., 1977., 3838., 2277., 1577.,
       4520., 2850., 3064., 2366., 4176., 4159., 3827., 1755., 3381.,
       1054., 1567., 2211., 1947., 3011., 4216., 3002., 2508., 3021.,
       2805., 3244., 2860., 3219., 3904., 3236., 2153., 3345., 3793.,
       3812., 2800., 1788., 3372., 2410., 1099., 2979., 3119., 2906.,
       2169., 1777., 1155., 3787., 1015., 2743., 2515., 2145., 1564.,
        930., 3387., 4962., 4983., 3535., 1423., 1319., 1690., 4968.,
       1987., 2026., 1834., 3877., 2520., 3447., 3766., 2568., 1783.,
       2476., 3681., 2640., 3279., 2760., 4377., 3232., 3406., 3761.,
       3194., 2438., 2952., 3613., 5181., 3908., 2530., 2618., 3266.,
       2907., 2078., 4019., 2423., 3324., 3500., 2062., 3277., 3474.,
       4092., 4105., 4132., 2598., 1884., 4547., 3768., 4776., 3932.,
       2208., 3350., 2122.,  667., 2346., 3946., 4672., 1307., 3499.,
       2808., 2143., 1664., 2457., 3665., 2209., 2784., 3454., 2770.,
       3635., 2049., 1876., 1548., 2595., 1319., 4331., 1911., 1410.,
       3047., 2364., 4596., 3791., 4086., 4512., 2299., 4207., 4735.,
       1358., 3444., 3486., 4160., 4184., 3966., 4115., 1380., 3951.,
       2140., 2111., 2127., 2620., 1957., 1168., 3039., 3760., 4226.,
       2265., 3369., 3147., 2239., 1112., 3297., 3189., 1316., 2614.,
       1276., 3943., 3139., 3894., 3007., 2207., 1478., 2954., 1447.,
       2268., 1823., 4073., 3430., 1538., 2015., 2468., 2613., 2351.,
       3145., 4016., 2783., 2518., 2002., 3964., 1902., 2422., 1039.,
       1765., 2911., 2628., 2087., 3533., 4533., 2032., 2853., 2704.,
       2289., 2947., 2791., 1678., 2381., 4006.,  934., 2540., 4020.,
       2724., 3164., 3035., 2539., 1626., 1790., 4031., 3346., 3056.,
       3307., 2281., 4240., 2804., 2231., 3339., 1993., 2838., 2084.,
       1618., 4092., 1649., 3259., 3144., 1192., 2860., 2750., 4127.,
       3682., 3707., 3668., 2769., 3804., 4202., 4134., 4542., 4038.,
       1721., 3988., 2962., 3493., 3241., 5124., 2579., 2827., 1710.,
       3364., 2264., 1913., 2275., 1677., 4112., 1472., 2864., 2715.,
       3437., 3500., 2716., 2976., 4086., 3284., 4917., 2400., 3733.,
       3348., 2625., 2105., 1517., 3357., 3245., 3593., 2762., 2291.,
       1901., 3390., 4520., 3265., 2881., 3321., 1370., 3307., 3597.,
       1277., 3669., 3309., 4266., 2970., 2971., 3762., 3409., 2825.,
       1430., 1211., 4042., 3003., 3388., 3175., 3703., 4106., 4674.,
       2946., 3236., 4738., 4888.,  892., 2476., 3582., 1760., 2818.,
       3099., 4059., 3082., 2845., 1867., 3954., 5267., 3949., 2743.,
       3832., 2539.])
```

In [85]:
```
import matplotlib.pyplot as plt
plt.scatter(X_train, y_train, color='red')
plt.plot(X_train, X_pred, color='blue')
plt.title("Linear Regression")
plt.show()
```




```
In [88]: from sklearn.metrics import r2_score
         r2_score(y_test,y_pred)

Out[88]: 0.780523547
```

The accuracy for the above machine learning model is

0.780523547

Conclusion

This technology project Involves loading and preprocessing a dataset, followed by various analysis and visualizations using IBM Cognos. helpful to make an data-driven decisions and responsive the web Traffic trends