Google Analytics Segment Export Analysis

Checking the working directory

In [1]:
```
import os
os.getcwd()
```

Out[1]:
```
'/Users/tanvishthakker/Desktop/BlueOwl'
```

Importing Pandas library for Data Manipulation and Analysis while Numpy for numerical operations

In [2]:
```
import pandas as pd
import numpy as np
```

Reading csv into a data frame and skipping the unwanted Google Analytics information rows

In [3]:
```
testFile = pd.read_csv("GA.csv",skiprows=6)
```

In [4]:
```
testFile
```

Out[4]:

	Day Index	Users	Page Views
0	01/01/2019	38	229
1	02/01/2019	98	521
2	03/01/2019	82	467
3	04/01/2019	88	572
4	05/01/2019	50	426
5	06/01/2019	52	392
6	07/01/2019	128	827
7	08/01/2019	111	631
8	09/01/2019	121	812
9	10/01/2019	117	785
10	11/01/2019	102	584
11	12/01/2019	64	532
12	13/01/2019	55	402
13	14/01/2019	137	998
14	15/01/2019	92	626
15	16/01/2019	72	562
16	17/01/2019	114	905
17	18/01/2019	68	474
18	19/01/2019	47	322
19	20/01/2019	36	200
20	21/01/2019	52	240
21	22/01/2019	105	673
22	23/01/2019	110	717
23	24/01/2019	113	771
24	25/01/2019	118	637
25	26/01/2019	63	364
26	27/01/2019	70	316
27	28/01/2019	106	650
28	29/01/2019	148	872
29	30/01/2019	159	954
30	31/01/2019	92	529
31	NaN	2,808	17,990

Ignoring the 31st row i.e. the total row in Google Analytics export

In [5]:
```
df = testFile.iloc[0:31,0:3]
df
```

Out[5]:

	Day Index	Users	Page Views
0	01/01/2019	38	229
1	02/01/2019	98	521
2	03/01/2019	82	467
3	04/01/2019	88	572
4	05/01/2019	50	426
5	06/01/2019	52	392
6	07/01/2019	128	827
7	08/01/2019	111	631
8	09/01/2019	121	812
9	10/01/2019	117	785
10	11/01/2019	102	584
11	12/01/2019	64	532
12	13/01/2019	55	402
13	14/01/2019	137	998
14	15/01/2019	92	626
15	16/01/2019	72	562
16	17/01/2019	114	905
17	18/01/2019	68	474
18	19/01/2019	47	322
19	20/01/2019	36	200
20	21/01/2019	52	240
21	22/01/2019	105	673
22	23/01/2019	110	717
23	24/01/2019	113	771
24	25/01/2019	118	637
25	26/01/2019	63	364
26	27/01/2019	70	316
27	28/01/2019	106	650
28	29/01/2019	148	872
29	30/01/2019	159	954
30	31/01/2019	92	529

Coverting 'Day Index' column to Datetime format for Date and Week manipulation

In [6]:
```
df['Day Index'] = pd.to_datetime(df['Day Index'],format="%d/%m/%Y")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31 entries, 0 to 30
Data columns (total 3 columns):
Day Index     31 non-null datetime64[ns]
Users         31 non-null object
Page Views    31 non-null object
dtypes: datetime64[ns](1), object(2)
memory usage: 824.0+ bytes
```

Assigning week numbers based on dates

Note: January 1st was a Tuesday and January 6th was a Sunday

In [7]:
```
df['Week_Number'] = df['Day Index'].dt.week
df
```

Out[7]:

	Day Index	Users	Page Views	Week_Number
0	2019-01-01	38	229	1
1	2019-01-02	98	521	1
2	2019-01-03	82	467	1
3	2019-01-04	88	572	1
4	2019-01-05	50	426	1
5	2019-01-06	52	392	1
6	2019-01-07	128	827	2
7	2019-01-08	111	631	2
8	2019-01-09	121	812	2
9	2019-01-10	117	785	2
10	2019-01-11	102	584	2
11	2019-01-12	64	532	2
12	2019-01-13	55	402	2
13	2019-01-14	137	998	3
14	2019-01-15	92	626	3
15	2019-01-16	72	562	3
16	2019-01-17	114	905	3
17	2019-01-18	68	474	3
18	2019-01-19	47	322	3
19	2019-01-20	36	200	3
20	2019-01-21	52	240	4
21	2019-01-22	105	673	4
22	2019-01-23	110	717	4
23	2019-01-24	113	771	4
24	2019-01-25	118	637	4
25	2019-01-26	63	364	4
26	2019-01-27	70	316	4
27	2019-01-28	106	650	5
28	2019-01-29	148	872	5
29	2019-01-30	159	954	5
30	2019-01-31	92	529	5

Converting 'Page Views' and 'Users' to Integer type data format while converting 'Week_Number' into string object for easy Grouping

In [8]:
```
df['Page Views'] = df['Page Views'].astype('int64')
df['Users'] = df['Users'].astype('int64')
df['Week_Number'] = df['Week_Number'].apply(str)
```

Checking for format transformations

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31 entries, 0 to 30
Data columns (total 4 columns):
Day Index      31 non-null datetime64[ns]
Users          31 non-null int64
Page Views     31 non-null int64
Week_Number    31 non-null object
dtypes: datetime64[ns](1), int64(2), object(1)
memory usage: 1.0+ KB
```

Grouping 'Week_Number' while simultaneously adding 'Page Views' and 'Users' for that week

In [10]:
```
df1 = df.groupby(['Week_Number']).sum()
df1
```

Out[10]:

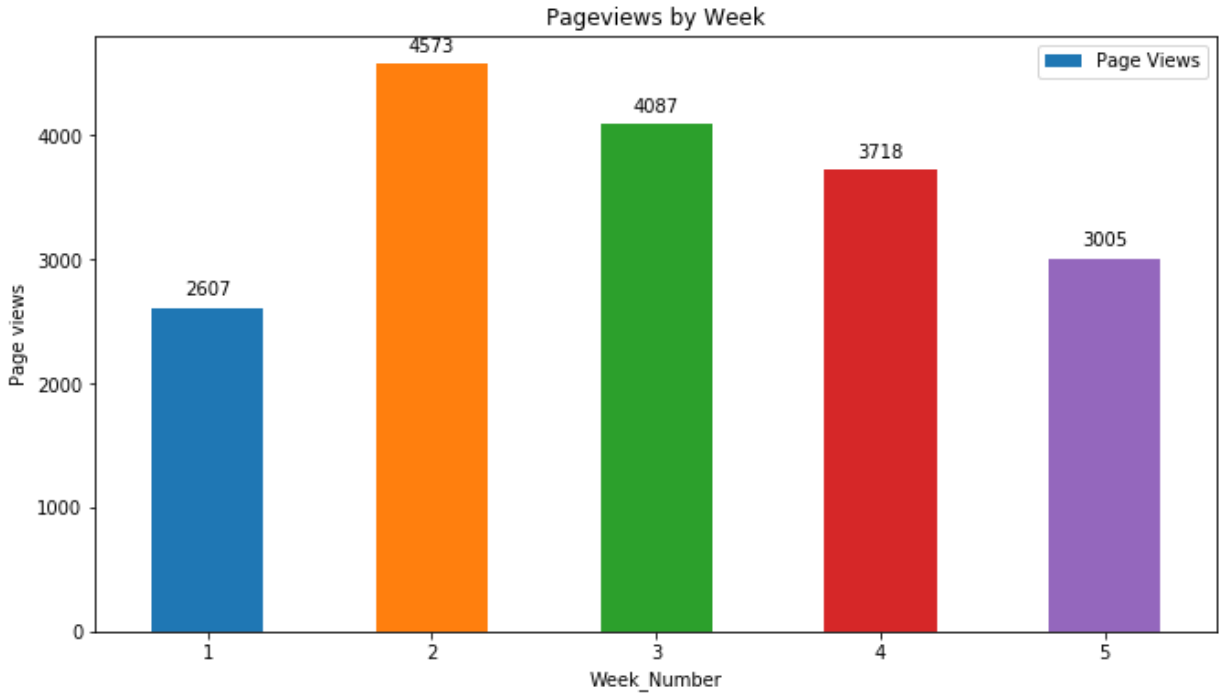
Week_Number	Users	Page Views
1	408	2607
2	698	4573
3	566	4087
4	631	3718
5	505	3005

Plotting and Labeling 'Week_Number' and 'Page Views'

In [14]:
```
ax = df1.reset_index().plot(x='Week_Number', y='Page Views',kind ='bar
',figsize = (11,6), title = 'Pageviews by Week')

for p in ax.patches:                    # For Bar Plot data labels
    ax.annotate(np.round(p.get_height(),decimals=2), (p.get_x()+p.get_
width()/2, p.get_height()), ha='center', va='center', xytext=(0, 10),
textcoords='offset points')
for tick in ax.get_xticklabels():  # To rotate X-axis data labels
    tick.set_rotation(0)
ax.set_ylabel('Page views')
```

Out[14]:
```
Text(0, 0.5, 'Page views')
```



Calculating Pageviews per User for each day

In [12]:
```
df['Pageviews per User'] = df['Page Views']/df['Users']
df
```

Out[12]:

	Day Index	Users	Page Views	Week_Number	Pageviews per User
0	2019-01-01	38	229	1	6.026316
1	2019-01-02	98	521	1	5.316327
2	2019-01-03	82	467	1	5.695122
3	2019-01-04	88	572	1	6.500000
4	2019-01-05	50	426	1	8.520000
5	2019-01-06	52	392	1	7.538462
6	2019-01-07	128	827	2	6.460938
7	2019-01-08	111	631	2	5.684685
8	2019-01-09	121	812	2	6.710744
9	2019-01-10	117	785	2	6.709402
10	2019-01-11	102	584	2	5.725490
11	2019-01-12	64	532	2	8.312500
12	2019-01-13	55	402	2	7.309091
13	2019-01-14	137	998	3	7.284672
14	2019-01-15	92	626	3	6.804348
15	2019-01-16	72	562	3	7.805556
16	2019-01-17	114	905	3	7.938596
17	2019-01-18	68	474	3	6.970588
18	2019-01-19	47	322	3	6.851064
19	2019-01-20	36	200	3	5.555556
20	2019-01-21	52	240	4	4.615385
21	2019-01-22	105	673	4	6.409524
22	2019-01-23	110	717	4	6.518182
23	2019-01-24	113	771	4	6.823009
24	2019-01-25	118	637	4	5.398305
25	2019-01-26	63	364	4	5.777778
26	2019-01-27	70	316	4	4.514286
27	2019-01-28	106	650	5	6.132075
28	2019-01-29	148	872	5	5.891892
29	2019-01-30	159	954	5	6.000000
30	2019-01-31	92	529	5	5.750000

Plotting Pageviews per User for each day in the month of January

In [13]:
```
bx = df.plot(x='Day Index', y='Pageviews per User',kind='bar',figsize
= (16,8), title= 'Pageviews/User by Day')
for p in bx.patches:
    bx.annotate(np.round(p.get_height(),decimals=2), (p.get_x()+p.get_
width()/2, p.get_height()), ha='center', va='center', xytext=(0, 10),
textcoords='offset points')
bx.set_ylabel('Pageviews per User')
```

Out[13]: Text(0, 0.5, 'Pageviews per User')