In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Conclusion

By making use of the BigQuery and programming in Python, I summarized the finding in this cell. Due to the time limit, I did not comment the code below. The important tables and visualizations can be found in the following section.

- Top five regions with the most users in 2015 is India, Athens, Italy, Denmark and LA
- Most of user profile does not contain the information of ages.
- The averge active duration(from creation to the last edit) is around 235 days, however, it has a large variations, popular posts can last much longer than normal ones.
- Through 12 months of 2015, people tend to produce less posts in January and Feburary(might due to holidays, more research needs).
- The **score, view_count, favoriate_num** are quite correalted with each other
- Surprisingly, the **comments_num** does not increase with number of **score, view_count** and **favoriate_num**(can be interpreted that a good post does not need too much explain)

Load and preprocess data retireved by BigQuery

In [2]:
```
posts = pd.read_csv('post_info.csv')
dates = pd.read_csv('date_info.csv')
location =  pd.read_csv('location_info.csv')
location = location.rename(columns={'owner_user_id':"userID"})

print(posts.columns)
print(dates.columns)
print(location.columns)

posts.head()
```

```
Index(['userID', 'post_cnt', 'avg_far', 'avg_com', 'avg_ans', 'avg_scr',
       'avg_view'],
      dtype='object')
Index(['creation_date', 'last_activity_date'], dtype='object')
Index(['userID', 'location'], dtype='object')
```

Out[2]:

	userID	post_cnt	avg_far	avg_com	avg_ans	avg_scr	avg_view
0	4304022	21	3.0	1.809524	1.571429	2.142857	1956.571429
1	1531040	23	0.0	1.565217	1.470588	-0.043478	89.058824
2	3395829	27	1.0	1.740741	1.250000	1.185185	392.125000
3	4235960	122	0.5	2.762295	1.681818	0.786885	209.136364
4	4801826	9	1.0	0.222222	0.800000	0.000000	112.600000

In [9]:
```
posts.describe()
```

Out[9]:

	userID	post_cnt	avg_far	avg_com	avg_ans	avg_scr	av
count	1.600000e+04	16000.000000	16000.000000	16000.000000	16000.000000	16000.000000	16000
mean	3.104228e+06	22.366875	0.710942	1.919714	1.406654	0.724918	282
std	1.633933e+06	57.621019	1.160697	1.183553	0.866429	1.989294	992
min	1.363000e+03	1.000000	0.000000	0.000000	0.125000	-8.000000	12
25%	1.643535e+06	6.000000	0.000000	1.176471	0.928571	0.200000	87
50%	3.349431e+06	11.000000	0.666667	1.714286	1.272727	0.500000	145
75%	4.569840e+06	22.000000	1.000000	2.400000	1.625000	0.944444	262
max	6.306298e+06	2372.000000	37.500000	28.000000	13.000000	115.000000	62835

In [3]:
```
locationinfo = location.groupby(['location']).count().reset_index()
locationinfo = locationinfo.rename(columns={'userID':"user num"})
locationinfo.sort_values(by=['user num'],ascending=False).head()
```

Out[3]:

	location	user num
888	Pune, Maharashtra, India	271
51	Athens	233
520	Italy	222
317	Denmark	218
616	Los Angeles	197

Explore the active date of posts

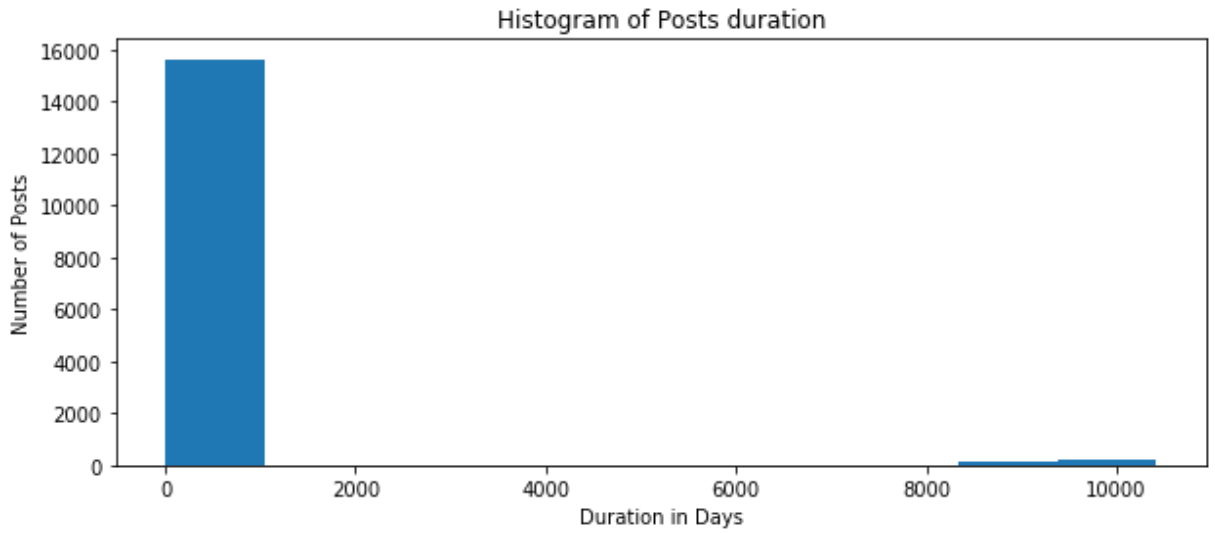
In [4]:
```
datelst = dates['creation_date']
creatdate = [i.split( )[0] for i in datelst]
dates['createMonth'] =  [i.replace('-', '', 2)[:-2] for i in creatdate]

datelst = dates['creation_date']
creatdate = [i.split( )[0] for i in datelst]
dates['create'] =  [int(i.replace('-', '', 2)) for i in creatdate]

datelst = dates['last_activity_date']
lasteditdate = [i.split( )[0] for i in datelst]
dates['lastedit'] =  [int(i.replace('-', '', 2)) for i in lasteditdate]

dates['dur'] = dates['lastedit'] - dates['create']

dates.head()
```

Out[4]:

	creation_date	last_activity_date	createMonth	create	lastedit	dur
0	2015-12-01 08:55:53.743 UTC	2015-12-04 11:52:25.647 UTC	201512	20151201	20151204	3
1	2015-02-04 12:38:37.197 UTC	2015-02-04 12:38:37.197 UTC	201502	20150204	20150204	0
2	2015-12-02 22:50:38.273 UTC	2015-12-17 08:06:23.237 UTC	201512	20151202	20151217	15
3	2015-01-08 09:17:39.027 UTC	2015-01-10 20:15:07.260 UTC	201501	20150108	20150110	2
4	2015-08-18 05:42:56.653 UTC	2015-08-18 09:31:58.830 UTC	201508	20150818	20150818	0

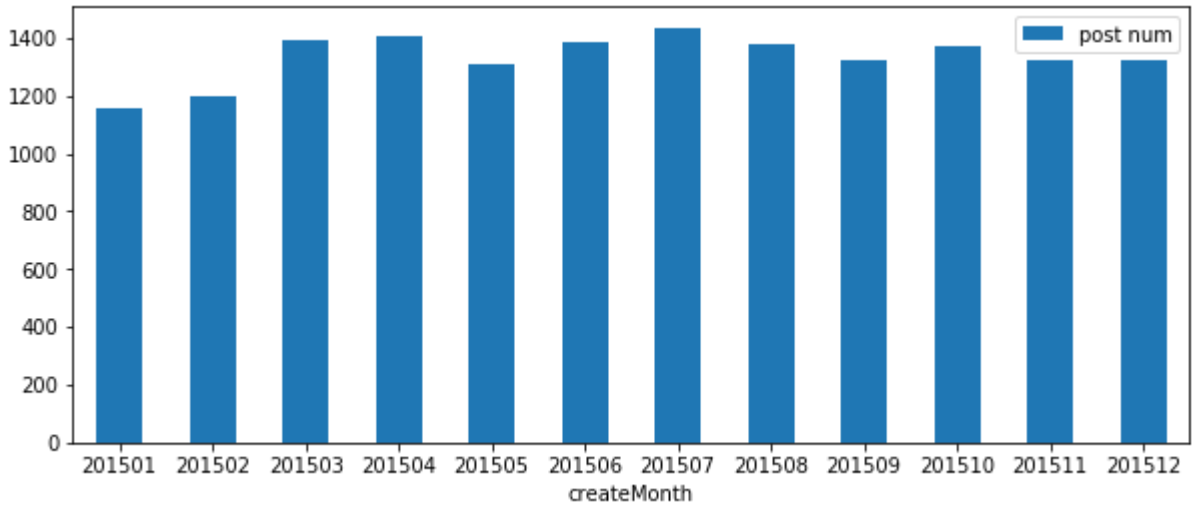
In [5]:
```
dates['dur'].plot(kind='hist', figsize=(10, 4))
count, bin_edges = np.histogram(dates['dur'], 30)
print(count)
print(bin_edges)

plt.title('Histogram of Posts duration') # add a title to the histogram
plt.ylabel('Number of Posts') # add y-label
plt.xlabel('Duration in Days') # add x-label

plt.show()

print('The average duration date:', dates['dur'].mean())
```

```
[15458   107    54     6     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0    44    86   114    82    49]
[    0.    347.3   694.6  1041.9  1389.2  1736.5  2083.8  2431.1  2778.4
  3125.7  3473.   3820.3  4167.6  4514.9  4862.2  5209.5  5556.8  5904.1
  6251.4  6598.7  6946.   7293.3  7640.6  7987.9  8335.2  8682.5  9029.8
  9377.1  9724.4 10071.7 10419. ]
```




```
The average duration date: 234.7088125
```

In [6]:
```
monthposts = dates.groupby(['createMonth']).count().reset_index()
monthposts = monthposts.rename(columns={'creation_date':"post num"})

ax = monthposts.plot.bar(x='createMonth', y='post num', rot=0, figsize=(10, 4))
```



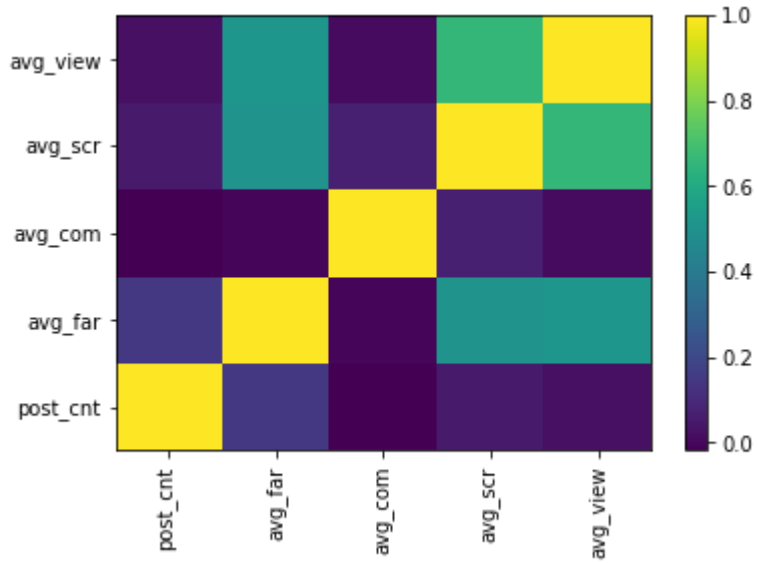
Correlation and Heatmap

In [7]:
```
correlation = posts[['post_cnt','avg_far','avg_com','avg_scr','avg_view']].corr(
correlation
```

Out[7]:

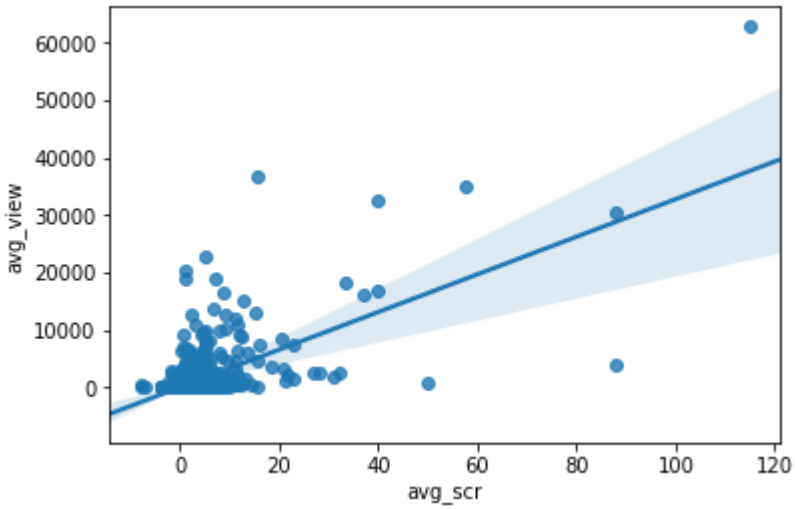
	post_cnt	avg_far	avg_com	avg_scr	avg_view
post_cnt	1.000000	0.144992	-0.019277	0.052184	0.024247
avg_far	0.144992	1.000000	-0.004022	0.502795	0.514834
avg_com	-0.019277	-0.004022	1.000000	0.070266	0.011941
avg_scr	0.052184	0.502795	0.070266	1.000000	0.656394
avg_view	0.024247	0.514834	0.011941	0.656394	1.000000

In [8]:
```
grouped_pivot = correlation

fig, ax = plt.subplots()
im = ax.pcolor(grouped_pivot)

#label names
row_labels = grouped_pivot.columns
col_labels = grouped_pivot.index

#move ticks and labels to the center
ax.set_xticks(np.arange(grouped_pivot.shape[1]) + 0.5, minor=False)
ax.set_yticks(np.arange(grouped_pivot.shape[0]) + 0.5, minor=False)

#insert labels
ax.set_xticklabels(row_labels, minor=False)
ax.set_yticklabels(col_labels, minor=False)

#rotate label if too long
plt.xticks(rotation=90)

fig.colorbar(im)
plt.show()
```



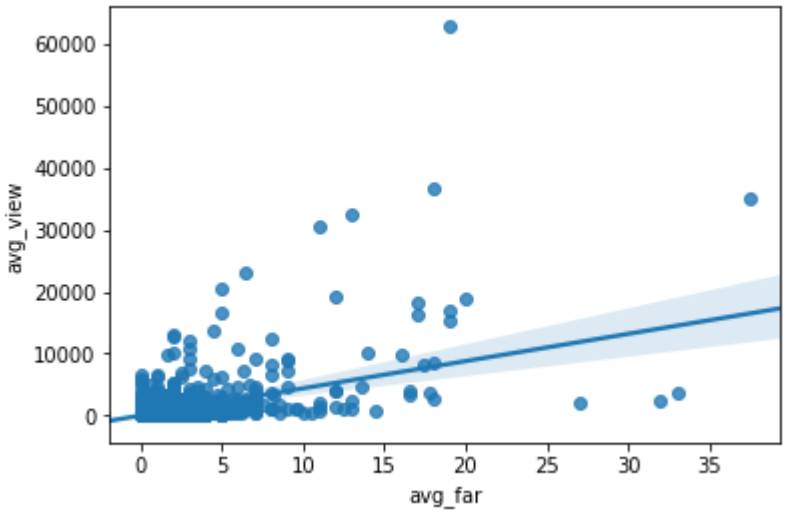
In [11]:
```
sns.regplot(x="avg_scr", y="avg_view", data=posts)
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x2577bb889e8>



In [12]:
```
sns.regplot(x="avg_far", y="avg_view", data=posts)
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x2577afdada0>



In [13]:
```
sns.regplot(x="avg_com", y="avg_view", data=posts)
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25779a67080>
```