assign1_q1

August 17, 2019

1 Assignment 1: Question 1

1.1 The Question

The crucial task before applying any machine learning algorithms is to understand the given data, i.e., a thorough data analysis cum data visualization is always necessary. As the part of this assignment, you are given a dataset, from which the following informations are to be extracted. Dataset : stackOverflow.csv Information to be extracted out: 1. Find out the no. of questions asked with respect to the given Tags. 2. Find out the most commonly used tags and what is the trend in Data Science Tags. 3. The average time is taken to answer a question. 4. Numbers of views related to the number of Answers. 5. Tags get highest/lowest rating in Questions. 6. Tags get highest/lowest rating in Answers. 7. Find out the most Active/Inactive in answering the questions. 8. Which tags draws the highest/lowest views?

1.2 The Solution

1.2.1 Importing Libraries

```
In [28]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         from collections import Counter
```

1.2.2 Loading the data into a DataFrame

```
In [29]: df = pd.read_csv('stackOverflow.csv')
```

1.2.3 Part 1 and 2

Extract qids amd tags

```
In [30]: df_filter = df[['qid', 'tags']]
         df_filter = df_filter.drop_duplicates()
```

Getting the values

```
In [31]: df_tags = df_filter['tags'].values
         df_qid = df_filter['qid'].values
```

Separating the tags to a single list

```
In [32]: bag_list = []
         for tag_list in df_tags :
             tag_list_array = tag_list.split(',')
             for tags in tag_list_array:
                 bag_list.append(tags)
```

Counting the tags individually using Counter and putting in a DF

```
In [33]: element_count = Counter(bag_list)
         df_filter_res = pd.DataFrame(list(element_count.items()), columns=['tag', 'count'])
         df_filter_res = df_filter_res.sort_values(by=['count'], ascending=False)
         df_filter_res.to_csv('tags_freqency.csv', index=False)
         df_filter_res
```

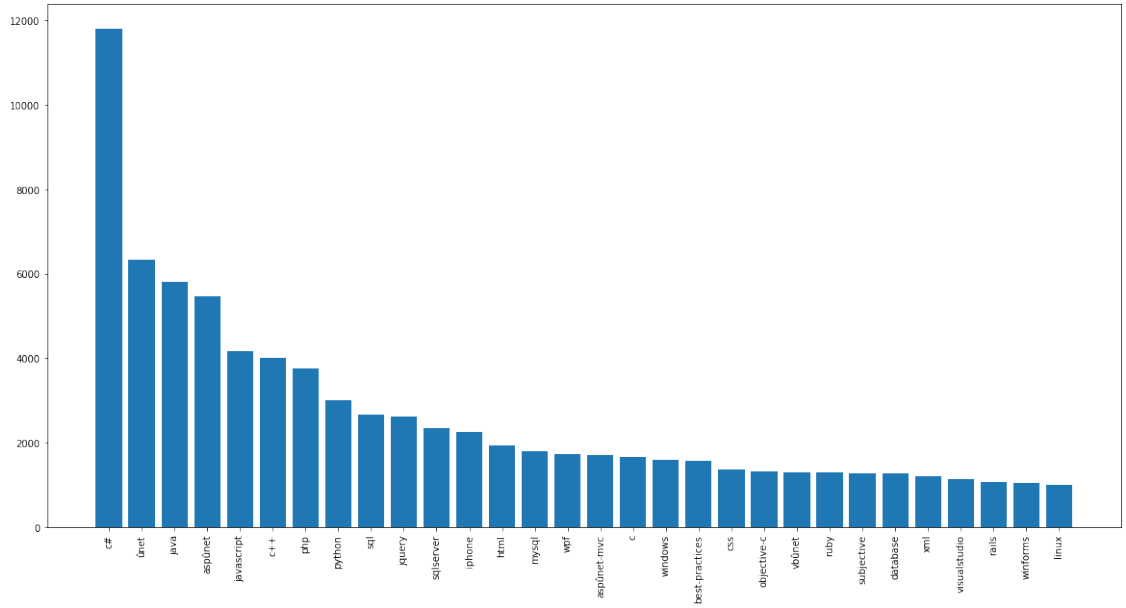
Out[33]:

	tag	count
97	c#	11793
25	ûnet	6328
90	java	5800
14	aspûnet	5459
21	javascript	4168
43	c++	4011
0	php	3771
26	python	3004
172	sql	2663
20	jquery	2618
136	sqlserver	2353
194	iphone	2246
101	html	1944
199	mysql	1809
191	wpf	1740
139	aspûnet-mvc	1720
121	c	1673
225	windows	1596
63	best-practices	1563
33	css	1359
9	objective-c	1315
123	vbûnet	1303
147	ruby	1301
6	subjective	1277
169	database	1270
183	xml	1218
32	visualstudio	1140
293	rails	1073
23	winforms	1057
120	linux	1000
...	...	...
9846	rehydrate	1

```
9845          double-dollar      1
9844         generic-method      1
9843        class-hierarchy      1
9842           inner-class      1
9865                   w32      1
9866            auto-value      1
9867   encryption-symmetric      1
9881           tesselation      1
9891          thoughtworks      1
9890                 cruise      1
9888   customtypedescriptor      1
4726          getobjectdata      1
4728             set-theory      1
9885              paralysis      1
9884            paul-graham      1
9883                web-gui      1
9882               diffview      1
9880        hexagonal-tiles      1
4753              data-grid      1
9879                 aforge      1
9878                   cite      1
4729       pullingmyhairout      1
4734       login-throttling      1
4751            xsdclassgen      1
9872             gtkbuilder      1
9871    discrete-structures      1
9870                   stax      1
4752           xsdobjectgen      1
14247         custom-fields      1

[14248 rows x 2 columns]
```

Number of Questions v/s Tags (Top 30)

In [34]:
```
%matplotlib inline

plt.rcParams['figure.figsize'] = [20,10]
plt.xticks(rotation='vertical')
plt.bar(df_filter_res['tag'].values[:30], df_filter_res['count'].values[:30])
plt.show()
```



1.2.4 Part 3

Calculating the answer times for each answer

```
In [35]: df['ans_time'] = df['at'] - df['qt']
```

The DF with answer times

```
In [36]: df
```

```
Out[36]:          Unnamed: 0     qid          i  qs          qt  \
          0                1  563355   62701.0   0  1235000081
          1                2  563355   62701.0   0  1235000081
          2                3  563356   15842.0  10  1235000140
          3                4  563356   15842.0  10  1235000140
          4                5  563356   15842.0  10  1235000140
          5                6  563356   15842.0  10  1235000140
          6                7  563356   15842.0  10  1235000140
          7                8  563356   15842.0  10  1235000140
          8                9  563356   15842.0  10  1235000140
          9               10  563356   15842.0  10  1235000140
          10              11  563356   15842.0  10  1235000140
          11              12  563356   15842.0  10  1235000140
          12              13  563356   15842.0  10  1235000140
          13              14  563356   15842.0  10  1235000140
          14              15  563356   15842.0  10  1235000140
          15              16  563356   15842.0  10  1235000140
          16              17  563356   15842.0  10  1235000140
```

```
17              18  563356   15842.0  10  1235000140
18              19  563365   68122.0   0  1235000369
19              20  563365   68122.0   0  1235000369
20              21  563365   68122.0   0  1235000369
21              22  563366    4556.0   0  1235000377
22              23  563367   32728.0   0  1235000414
23              24  563367   32728.0   0  1235000414
24              25  563368   64895.0   0  1235000427
25              26  563369   33775.0   5  1235000481
26              27  563369   33775.0   5  1235000481
27              28  563369   33775.0   5  1235000481
28              29  563375   27580.0   0  1235000572
29              30  563383   26414.0   2  1235000804
...            ...     ...       ...  ..         ...
263510      263511  961367  118747.0   1  1244358062
263511      263512  961368   83102.0   0  1244358258
263512      263513  961368   83102.0   0  1244358258
263513      263514  961375  117069.0   0  1244358461
263514      263515  961375  117069.0   0  1244358461
263515      263516  961375  117069.0   0  1244358461
263516      263517  961375  117069.0   0  1244358461
263517      263518  961383   84583.0   0  1244358739
263518      263519  961383   84583.0   0  1244358739
263519      263520  961383   84583.0   0  1244358739
263520      263521  961389   86751.0   0  1244359049
263521      263522  961389   86751.0   0  1244359049
263522      263523  961392  110233.0   0  1244359159
263523      263524  961392  110233.0   0  1244359159
263524      263525  961392  110233.0   0  1244359159
263525      263526  961392  110233.0   0  1244359159
263526      263527  961406   90537.0   2  1244359773
263527      263528  961406   90537.0   2  1244359773
263528      263529  961406   90537.0   2  1244359773
263529      263530  961406   90537.0   2  1244359773
263530      263531  961406   90537.0   2  1244359773
263531      263532  961406   90537.0   2  1244359773
263532      263533  961406   90537.0   2  1244359773
263533      263534  961406   90537.0   2  1244359773
263534      263535  961406   90537.0   2  1244359773
263535      263536  961412  117069.0   0  1244359898
263536      263537  961412  117069.0   0  1244359898
263537      263538  961416   63225.0   2  1244360143
263538      263539  961416   63225.0   2  1244360143
263539      263540  961459   33584.0   0  1244361919

                                                tags   qvc  qac     aid  \
0                          php,error,gd,image-processing   220    2  563372
1                          php,error,gd,image-processing   220    2  563374
```

2	lisp,scheme,subjective,clojure	1047	16	563358
3	lisp,scheme,subjective,clojure	1047	16	563413
4	lisp,scheme,subjective,clojure	1047	16	563454
5	lisp,scheme,subjective,clojure	1047	16	563472
6	lisp,scheme,subjective,clojure	1047	16	563484
7	lisp,scheme,subjective,clojure	1047	16	563635
8	lisp,scheme,subjective,clojure	1047	16	563642
9	lisp,scheme,subjective,clojure	1047	16	564028
10	lisp,scheme,subjective,clojure	1047	16	564747
11	lisp,scheme,subjective,clojure	1047	16	568102
12	lisp,scheme,subjective,clojure	1047	16	568213
13	lisp,scheme,subjective,clojure	1047	16	568221
14	lisp,scheme,subjective,clojure	1047	16	568975
15	lisp,scheme,subjective,clojure	1047	16	569399
16	lisp,scheme,subjective,clojure	1047	16	931176
17	lisp,scheme,subjective,clojure	1047	16	931214
18	cocoa-touch,objective-c,design-patterns	108	3	563376
19	cocoa-touch,objective-c,design-patterns	108	3	563379
20	cocoa-touch,objective-c,design-patterns	108	3	564758
21	core-animation	179	1	563486
22	django,django-models	247	2	565729
23	django,django-models	247	2	565801
24	aspûnet	48	1	563389
25	scala,pattern-matching,oop,object-oriented-des...	276	3	563790
26	scala,pattern-matching,oop,object-oriented-des...	276	3	564247
27	scala,pattern-matching,oop,object-oriented-des...	276	3	564498
28	jquery,javascript,dom	627	1	563400
29	winforms,gridview,ûnet	215	1	563466
...	...	...	...	...
263510	programming	76	7	961400
263511	bitmap,3d	25	2	961427
263512	bitmap,3d	25	2	961430
263513	c#,lists	42	4	961378
263514	c#,lists	42	4	961381
263515	c#,lists	42	4	961384
263516	c#,lists	42	4	961403
263517	c#,httpwebrequest,httpwebresponse	31	3	961410
263518	c#,httpwebrequest,httpwebresponse	31	3	961418
263519	c#,httpwebrequest,httpwebresponse	31	3	961441
263520	sql,transactions	23	2	961426
263521	sql,transactions	23	2	961435
263522	network-programming	37	4	961397
263523	network-programming	37	4	961399
263524	network-programming	37	4	961404
263525	network-programming	37	4	961408
263526	beginner,self-learning	91	9	961411
263527	beginner,self-learning	91	9	961414
263528	beginner,self-learning	91	9	961420

263529	beginner,self-learning	91	9	961423
263530	beginner,self-learning	91	9	961429
263531	beginner,self-learning	91	9	961431
263532	beginner,self-learning	91	9	961442
263533	beginner,self-learning	91	9	961451
263534	beginner,self-learning	91	9	961452
263535	c#,string-manipulation	26	2	961415
263536	c#,string-manipulation	26	2	961421
263537	ûnet,msil,history	42	2	961428
263538	ûnet,msil,history	42	2	961443
263539	css	5	1	961462

	j	as	at	ans_time
0	67183.0	2	1235000501	420
1	66554.0	0	1235000551	470
2	15842.0	3	1235000177	37
3	893.0	18	1235001545	1405
4	11649.0	4	1235002457	2317
5	50742.0	6	1235002809	2669
6	8899.0	1	1235003266	3126
7	60190.0	12	1235007817	7677
8	65235.0	1	1235007913	7773
9	32797.0	8	1235020626	20486
10	19619.0	1	1235040652	40512
11	10728.0	3	1235098147	98007
12	68757.0	5	1235101761	101621
13	26177.0	0	1235102039	101899
14	31141.0	0	1235124207	124067
15	21734.0	3	1235132888	132748
16	114979.0	0	1243739978	8739838
17	82368.0	1	1243741787	8741647
18	68122.0	0	1235000573	204
19	66344.0	2	1235000607	238
20	28768.0	0	1235040919	40550
21	4556.0	1	1235003300	2923
22	6760.0	3	1235056128	55714
23	5616.0	1	1235056872	56458
24	47365.0	1	1235001121	694
25	9815.0	6	1235012496	12015
26	62237.0	0	1235027318	26837
27	51431.0	7	1235035072	34591
28	27580.0	0	1235001327	755
29	60824.0	1	1235002683	1879
...	...	..	...	...
263510	109122.0	0	1244359545	1483
263511	3829.0	0	1244360467	2209
263512	82294.0	0	1244360622	2364
263513	25300.0	3	1244358638	177

```
263514   87053.0   0  1244358726       265
263515   75500.0   0  1244358782       321
263516    3829.0   1  1244359658      1197
263517    3546.0   0  1244359851      1112
263518   22656.0   1  1244360200      1461
263519   29407.0   0  1244360919      2180
263520  109122.0   1  1244360455      1406
263521   27535.0   1  1244360774      1725
263522   10307.0   0  1244359518       359
263523   56541.0   2  1244359538       379
263524   65696.0   0  1244359677       518
263525  116346.0   1  1244359810       651
263526  104097.0   0  1244359853        80
263527   60664.0   7  1244359924       151
263528  116371.0   0  1244360249       476
263529    2525.0   2  1244360314       541
263530   42086.0   0  1244360565       792
263531   16076.0   2  1244360651       878
263532  117069.0   0  1244360992      1219
263533  108130.0   0  1244361527      1754
263534   66353.0   0  1244361541      1768
263535   22656.0   2  1244360077       179
263536    3712.0   1  1244360250       352
263537   22656.0   2  1244360525       382
263538  110945.0   0  1244361172      1029
263539   23590.0   0  1244362060       141

[263540 rows x 13 columns]
```

Calculating the average answer time for each question after dropping anomalous data points(negative answer time)

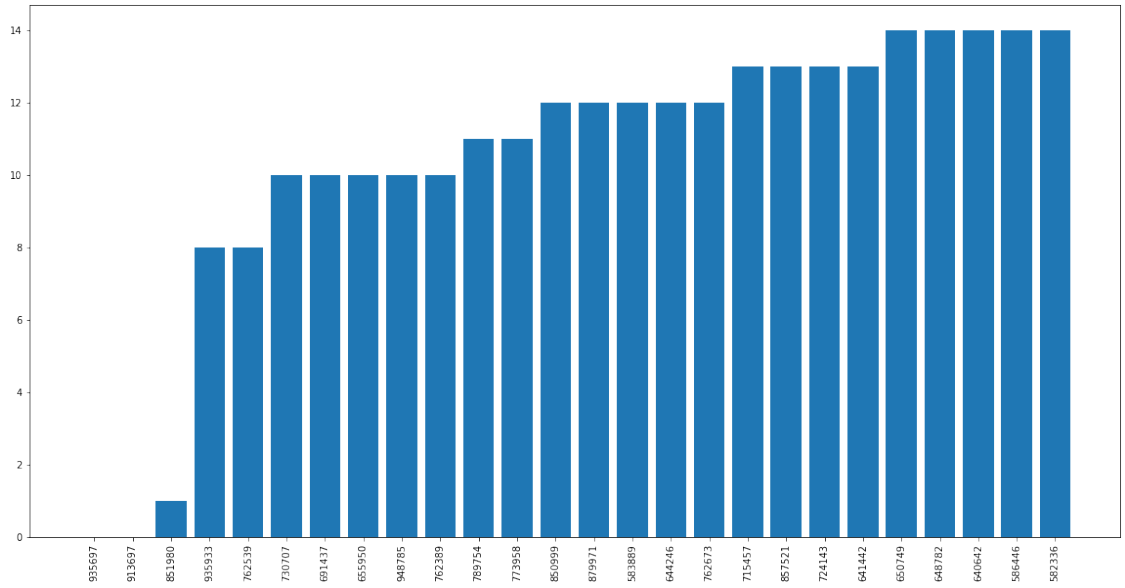
```
In [37]: df_time_f = df[['qid', 'ans_time']]
         df_time_error = df_time_f[df_time_f['ans_time']<0]
         df_time_f = df_time_f[df_time_f['ans_time']>=0]
         df_time_f.groupby(['qid']).mean()
         df_time_f = df_time_f.sort_values(by=['ans_time'])
         df_time_f

Out[37]:           qid  ans_time
         247908  935697         0
         233976  913697         0
         233977  913697         0
         194270  851980         1
         194271  851980         1
         194272  851980         1
         194273  851980         1
         248054  935933         8
```

135463	762539	8
114036	730707	10
87485	691437	10
63582	655950	10
255971	948785	10
135336	762389	10
153523	789754	11
143024	773958	11
193587	850999	12
212260	879971	12
14013	583889	12
55668	644246	12
135530	762673	12
103712	715457	13
197846	857521	13
109524	724143	13
53635	641442	13
60070	650749	14
58727	648782	14
53060	640642	14
15777	586446	14
12973	582336	14
...	...	...
6836	573627	8635186
10978	579511	8663962
1775	565893	8664083
14648	584886	8672930
6184	572611	8682458
6185	572611	8682703
8763	576456	8686782
14029	583897	8696282
6602	573235	8696745
16	563356	8739838
17	563356	8741647
12338	581448	8749437
11619	580385	8764355
2491	566872	8798315
10750	579196	8798896
11464	580216	8825547
9786	577824	8862761
9122	576908	8870648
1863	566066	8876885
6591	573225	8921649
6592	573225	8922559
7245	574195	8929826
2390	566746	8952135
6589	573215	8971741
6377	572938	8985521

```
3023    567674   8988592
4851    570302   8998296
2767    567304   9053854
74      563457   9137689
802     564608   9273540

[263523 rows x 2 columns]
```

Average answer time v/s Question ID (Best 30)

```
In [38]: %matplotlib inline

         plt.rcParams['figure.figsize'] = [20,10]
         plt.xticks(rotation='vertical')
         x=df_time_f['qid'].values[:30]
         y=[]
         for i in x:
             i = str(i)
             y.append(i)
         plt.bar(y, df_time_f['ans_time'].values[:30])
         plt.show()
```



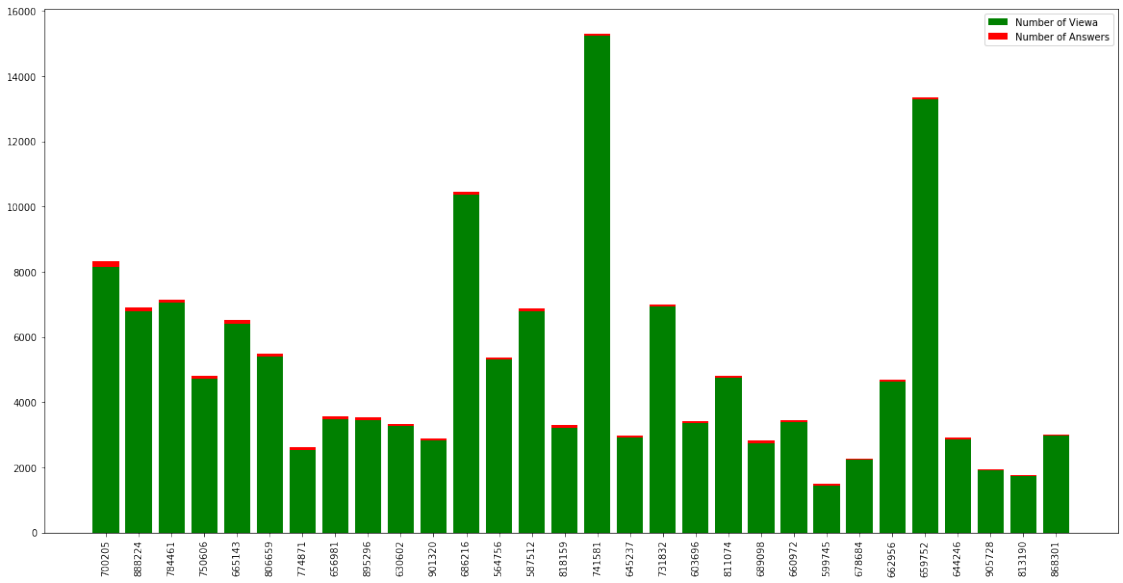
1.2.5 Part 4

Extracting and sorting by no. of answers

```
In [39]: df_qac_qvc = df[['qid', 'qvc', 'qac']]
         df_qac_qvc = df_qac_qvc.drop_duplicates()
         df_qac_qvc = df_qac_qvc.sort_values(by=['qac'], ascending=False)
```

Number of views AND Number of answers v/s Question ID (Best 30(Number of answers))

```
In [40]: %matplotlib inline

         plt.rcParams['figure.figsize'] = [20,10]
         plt.xticks(rotation='vertical')
         x=df_qac_qvc['qid'].values[:30]
         y=[]
         for i in x:
             i = str(i)
             y.append(i)
         p1 = plt.bar(y, df_qac_qvc['qvc'].values[:30], color='g')
         p2 = plt.bar(y, df_qac_qvc['qac'].values[:30], bottom=df_qac_qvc['qvc'].values[:30], c

         label = ['Number of Viewa', 'Number of Answers']
         plt.legend(labels=label)

         plt.show()
```



1.2.6 Part 5

Extracting Scores under individual tags

```
In [41]: df_q_score = df[['qid', 'tags', 'qs']]
         df_q_score = df_q_score.drop_duplicates()
         xt = df_q_score['tags'].values
         x1 = df_q_score['qid'].values
         x2 = df_q_score['qs'].values
         #df_q_score_sep = pd.DataFrame(columns=['tag', 'qid', 'qs'])
```

```
rows = []
for i in range(len(x1)):
    k = xt[i].split(',')
    for j in k:
        #df_q_score_sep = df_q_score_sep.append({'tag':j, 'qid':x1[i], 'qs':x2[i]}, i
        l1 = [j, x1[i], x2[i]]
        rows.append(l1)

df_q_score_sep = pd.DataFrame(rows, columns=['tag', 'qid', 'qs'])
df_q_score_sep
```

Out[41]:

	tag	qid	qs
0	php	563355	0
1	error	563355	0
2	gd	563355	0
3	image-processing	563355	0
4	lisp	563356	10
5	scheme	563356	10
6	subjective	563356	10
7	clojure	563356	10
8	cocoa-touch	563365	0
9	objective-c	563365	0
10	design-patterns	563365	0
11	core-animation	563366	0
12	django	563367	0
13	django-models	563367	0
14	aspûnet	563368	0
15	scala	563369	5
16	pattern-matching	563369	5
17	oop	563369	5
18	object-oriented-design	563369	5
19	design-principles	563369	5
20	jquery	563375	0
21	javascript	563375	0
22	dom	563375	0
23	winforms	563383	2
24	gridview	563383	2
25	ûnet	563383	2
26	python	563384	2
27	http	563384	2
28	vim	563385	2
29	netrw	563385	2
...	...	...	..
254190	html	961333	0
254191	hidden	961333	0
254192	cross-domain	961334	0
254193	webbrowser	961334	0
254194	same-origin-policy	961334	0

```
254195                  msvcrt  961337   0
254196                     x64  961337   0
254197                    java  961340   0
254198                    jdbc  961340   0
254199               beginner  961344   1
254200                  python  961344   1
254201             programming  961367   1
254202                  bitmap  961368   0
254203                      3d  961368   0
254204                      c#  961375   0
254205                   lists  961375   0
254206                      c#  961383   0
254207          httpwebrequest  961383   0
254208         httpwebresponse  961383   0
254209                     sql  961389   0
254210            transactions  961389   0
254211     network-programming  961392   0
254212               beginner  961406   2
254213          self-learning  961406   2
254214                      c#  961412   0
254215     string-manipulation  961412   0
254216                   ûnet  961416   2
254217                   msil  961416   2
254218                history  961416   2
254219                    css  961459   0

[254220 rows x 3 columns]
```

Calculating the Highest and Lowest scores per tag and respective Question IDs

```
In [42]: df_min_qs = df_q_score_sep.groupby(['tag']).min()
         df_max_qs = df_q_score_sep.groupby(['tag']).max()
         df_max_qs = df_max_qs.rename(columns={'qs':'qs_max', 'qid':'qid_max'})
         df_min_qs = df_min_qs.rename(columns={'qs':'qs_min', 'qid':'qid_min'})
         df_qs_merge = pd.concat([df_max_qs, df_min_qs], axis=1)
         df_qs_merge = df_qs_merge.sort_values(by=['qs_max'], ascending=False)
         df_qs_merge = df_qs_merge.reset_index(level=0)
         df_qs_merge
```

Out[42]:

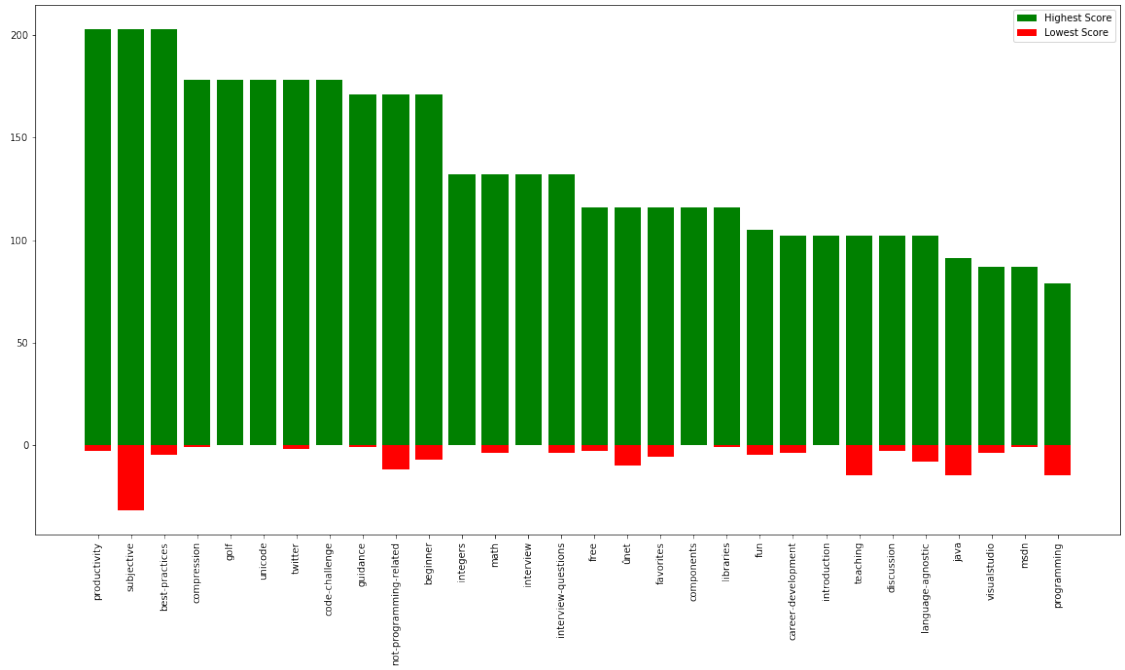
	tag	qid_max	qs_max	qid_min	qs_min
0	productivity	959024	203	580448	-3
1	subjective	961169	203	563356	-32
2	best-practices	960622	203	563493	-5
3	compression	960386	178	573210	-1
4	golf	938022	178	838004	0
5	unicode	956867	178	571174	0
6	twitter	954236	178	580369	-2
7	code-challenge	954926	178	585721	0

8	guidance	891094	171	587512	-1
9	not-programming-related	959288	171	563401	-12
10	beginner	961406	171	563904	-7
11	integers	879375	132	579310	0
12	math	960648	132	566460	-4
13	interview	947748	132	568428	0
14	interview-questions	958322	132	564184	-4
15	free	959289	116	570317	-3
16	ûnet	961416	116	563383	-10
17	favorites	945537	116	626859	-6
18	components	953357	116	567254	0
19	libraries	960357	116	564872	-1
20	fun	954644	105	566315	-5
21	career-development	958707	102	564688	-4
22	introduction	876202	102	577382	0
23	teaching	950619	102	576136	-15
24	discussion	958258	102	564756	-3
25	language-agnostic	961169	102	563891	-8
26	java	961340	91	563531	-15
27	visualstudio	961060	87	563394	-4
28	msdn	950850	87	584162	-1
29	programming	961367	79	564367	-15
...	...	...	...	...	...
14218	entertainment	658828	-3	658828	-3
14219	one	912865	-3	912865	-3
14220	flamebait	806573	-3	806573	-3
14221	credit	841727	-3	841727	-3
14222	domainname	955709	-3	955709	-3
14223	google-desktop-search	835565	-3	835565	-3
14224	board	849580	-3	849580	-3
14225	llvmclang	815998	-3	815998	-3
14226	totals	603857	-3	603857	-3
14227	maczoop	811106	-3	811106	-3
14228	card	841727	-3	841727	-3
14229	stdcall	695734	-3	695734	-3
14230	tom-tom	912865	-3	912865	-3
14231	indie	674922	-4	674922	-4
14232	programming-by-contract	598978	-4	598978	-4
14233	stackoverflowerror	951635	-4	951635	-4
14234	ddos	938588	-4	938588	-4
14235	wisdom-of-crowds	674922	-4	674922	-4
14236	django-custom-tag	594259	-4	594259	-4
14237	go	739681	-4	739681	-4
14238	dancer	597428	-5	597428	-5
14239	kthnxbye	897173	-5	897173	-5
14240	human	597428	-5	597428	-5
14241	geekdom	631062	-5	631062	-5
14242	sqlxmlreader	710240	-5	710240	-5

```
14243            online-video   716313      -5   716313      -5
14244                    misv   714522      -6   714522      -6
14245                 spyware   805715      -6   805715      -6
14246          hide-component   805715      -6   805715      -6
14247             personality   651459     -15   651459     -15

[14248 rows x 5 columns]
```

Highest Score AND Lowest Score v/s Tag (Best 30(Highest Score))

```
In [43]: %matplotlib inline

         plt.rcParams['figure.figsize'] = [20,10]
         plt.xticks(rotation='vertical')
         p1 = plt.bar(df_qs_merge['tag'].values[:30], df_qs_merge['qs_max'].values[:30], color=
         p2 = plt.bar(df_qs_merge['tag'].values[:30], df_qs_merge['qs_min'].values[:30], color=

         label = ['Highest Score', 'Lowest Score']
         plt.legend(labels=label)

         plt.show()
```



1.2.7 Part 6

Extracting Scores under individual tags

In [44]:
```
df_a_score = df[['aid', 'tags', 'as']]
df_a_score = df_a_score.drop_duplicates()
xt = df_a_score['tags'].values
x1 = df_a_score['aid'].values
x2 = df_a_score['as'].values
#df_a_score_sep = pd.DataFrame(columns=['tag', 'aid', 'as'])
rows = []
for i in range(len(x1)):
    k = xt[i].split(',')
    for j in k:
        #df_a_score_sep = df_a_score_sep.append({'tag':j, 'aid':x1[i], 'as':x2[i]}, i
        l1 = [j, x1[i], x2[i]]
        rows.append(l1)

df_a_score_sep = pd.DataFrame(rows, columns=['tag', 'aid', 'as'])
df_a_score_sep
```

Out[44]:

	tag	aid	as
0	php	563372	2
1	error	563372	2
2	gd	563372	2
3	image-processing	563372	2
4	php	563374	0
5	error	563374	0
6	gd	563374	0
7	image-processing	563374	0
8	lisp	563358	3
9	scheme	563358	3
10	subjective	563358	3
11	clojure	563358	3
12	lisp	563413	18
13	scheme	563413	18
14	subjective	563413	18
15	clojure	563413	18
16	lisp	563454	4
17	scheme	563454	4
18	subjective	563454	4
19	clojure	563454	4
20	lisp	563472	6
21	scheme	563472	6
22	subjective	563472	6
23	clojure	563472	6
24	lisp	563484	1
25	scheme	563484	1
26	subjective	563484	1
27	clojure	563484	1
28	lisp	563635	12
29	scheme	563635	12

```
...                     ...     ...  ..
814598  network-programming  961408   1
814599             beginner  961411   0
814600        self-learning  961411   0
814601             beginner  961414   7
814602        self-learning  961414   7
814603             beginner  961420   0
814604        self-learning  961420   0
814605             beginner  961423   2
814606        self-learning  961423   2
814607             beginner  961429   0
814608        self-learning  961429   0
814609             beginner  961431   2
814610        self-learning  961431   2
814611             beginner  961442   0
814612        self-learning  961442   0
814613             beginner  961451   0
814614        self-learning  961451   0
814615             beginner  961452   0
814616        self-learning  961452   0
814617                  c#  961415   2
814618  string-manipulation  961415   2
814619                  c#  961421   1
814620  string-manipulation  961421   1
814621                ûnet  961428   2
814622                msil  961428   2
814623             history  961428   2
814624                ûnet  961443   0
814625                msil  961443   0
814626             history  961443   0
814627                 css  961462   0

[814628 rows x 3 columns]
```

Calculating the Highest and Lowest scores per tag and respective Answer IDs

```
In [45]: df_min_as = df_a_score_sep.groupby(['tag']).min()
         df_max_as = df_a_score_sep.groupby(['tag']).max()
         df_max_as = df_max_as.rename(columns={'as':'as_max', 'aid':'aid_max'})
         df_min_as = df_min_as.rename(columns={'as':'as_min', 'aid':'aid_min'})
         df_as_merge = pd.concat([df_max_as, df_min_as], axis=1)
         df_as_merge = df_as_merge.sort_values(by=['as_max'], ascending=False)
         df_as_merge = df_as_merge.reset_index(level=0)
         df_as_merge
```

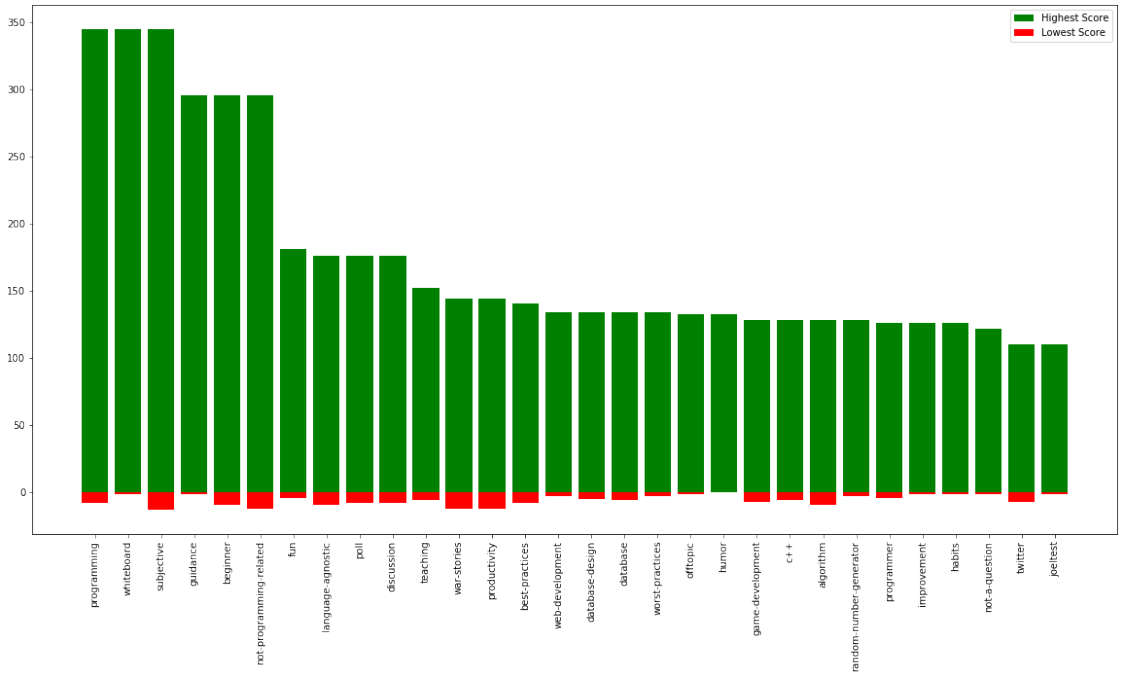
```
Out[45]:                         tag  aid_max  as_max  aid_min  as_min
         0                programming   961405     345   564381      -8
         1                 whiteboard   953002     345   688123      -1
```

2	subjective	961405	345	563358	-13
3	guidance	952601	296	587517	-1
4	beginner	961452	296	563917	-9
5	not-programming-related	960508	296	563422	-12
6	fun	961143	181	566330	-4
7	language-agnostic	961208	176	563898	-9
8	poll	959493	176	577864	-8
9	discussion	959112	176	564764	-8
10	teaching	956226	152	576138	-6
11	war-stories	898373	144	741583	-12
12	productivity	959091	144	580505	-12
13	best-practices	961293	141	563503	-8
14	web-development	961308	134	563590	-3
15	database-design	961231	134	563846	-5
16	database	960938	134	563846	-6
17	worst-practices	956712	134	571269	-3
18	offtopic	961112	133	569894	-1
19	humor	958234	133	574791	0
20	game-development	961117	128	564150	-7
21	c++	961448	128	563444	-6
22	algorithm	961282	128	564572	-9
23	random-number-generator	939504	128	576338	-3
24	programmer	942674	126	584237	-4
25	improvement	939694	126	583681	-1
26	habits	913239	126	806670	-1
27	not-a-question	961117	122	600160	-1
28	twitter	954586	110	580378	-7
29	joeltest	961405	110	567848	-1
...	...	...	...	...	...
14218	searching-xml	591192	0	591192	0
14219	jboss-tools	750758	0	750758	0
14220	tablelayoutpanel	927220	0	678936	0
14221	berkeley	959662	0	959662	0
14222	video-embedding	839005	0	838952	0
14223	cproj	909211	0	909196	0
14224	video-capturing	690045	0	688189	0
14225	into	831154	0	831131	0
14226	invalidauthenticitytoken	941661	0	941660	0
14227	cpu-cores	855085	0	654692	0
14228	invalidation	654747	0	654304	0
14229	inversion	942546	0	906101	0
14230	secure-gateway	734923	0	699028	0
14231	belongs-on-uservoice	952117	0	931143	0
14232	investment-protection	645751	0	645751	0
14233	exi	679743	0	679743	0
14234	tablecell-collection	647192	0	647192	0
14235	intruments	749699	0	684198	0
14236	table-udf	882493	0	786720	0

```
14237                commons-pool   940272      -1   940272      -1
14238              live-meetings   885638      -1   885638      -1
14239               createobject   826381      -1   826381      -1
14240  sql-server-data-services   592772      -1   592772      -1
14241                        wsx   910420      -1   910420      -1
14242                     votive   910420      -1   910420      -1
14243                       fail   758161      -1   758099      -2
14244                      gmake   871965      -2   871965      -2
14245             sofaq-official   838635      -2   838635      -2
14246         large-file-support   908406      -2   908128      -3
14247                       ipsec   703306      -2   703306      -2

[14248 rows x 5 columns]
```

Highest Score AND Lowest Score v/s Tag (Best 30(Highest Score))

```
In [46]: %matplotlib inline

         plt.rcParams['figure.figsize'] = [20,10]
         plt.xticks(rotation='vertical')
         p1 = plt.bar(df_as_merge['tag'].values[:30], df_as_merge['as_max'].values[:30], color=
         p2 = plt.bar(df_as_merge['tag'].values[:30], df_as_merge['as_min'].values[:30], color=

         label = ['Highest Score', 'Lowest Score']
         plt.legend(labels=label)

         plt.show()
```



1.2.8 Part 7

Finding the activity order of users by number of answers posted

```
In [47]: df_user = df[['aid','j']]
         df_user = df_user.groupby(['j']).count().reset_index(level=0).sort_values(by=['aid'],
         df_user
```

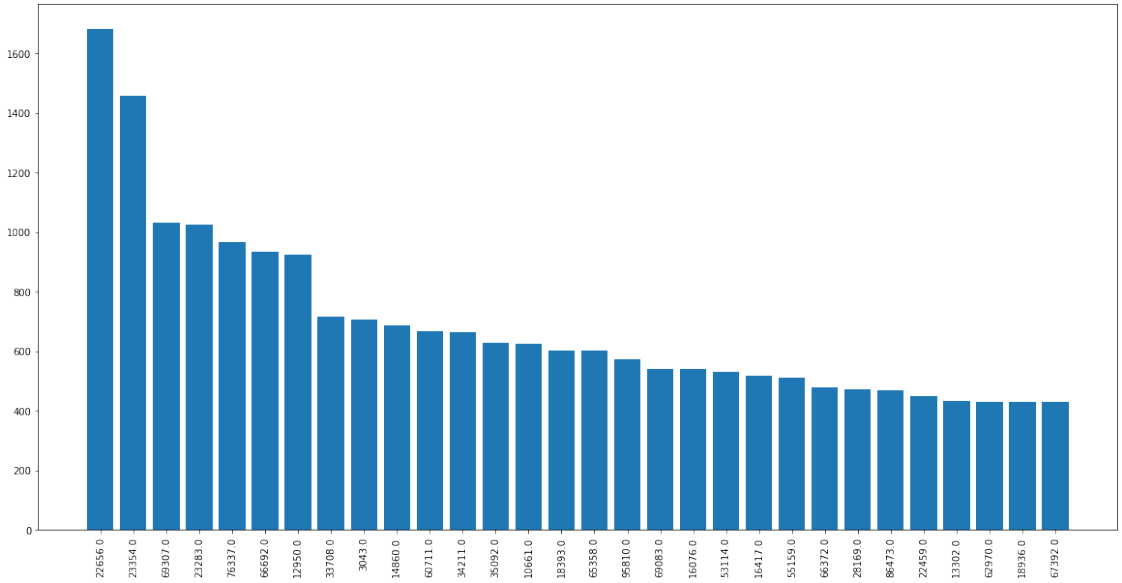
Out[47]:

	j	aid
5353	22656.0	1683
5477	23354.0	1457
12458	69307.0	1033
5466	23283.0	1024
14452	76337.0	967
11876	66692.0	935
3599	12950.0	926
7083	33708.0	716
1236	3043.0	708
3940	14860.0	688
10826	60711.0	669
7148	34211.0	663
7253	35092.0	630
3115	10661.0	624
4583	18393.0	604
11645	65358.0	602
20283	95810.0	575
12405	69083.0	542
4163	16076.0	541
9695	53114.0	530
4225	16417.0	517
10012	55159.0	511
11815	66372.0	479
6265	28169.0	474
17546	86473.0	471
5323	22459.0	449
3668	13302.0	433
11227	62970.0	432
4682	18936.0	431
12007	67392.0	430
...	...	...
5529	23650.0	1
17156	85273.0	1
17157	85274.0	1
17158	85277.0	1
5545	23724.0	1
17128	85196.0	1

```
17127   85195.0     1
17126   85193.0     1
17089   85074.0     1
17091   85078.0     1
17092   85083.0     1
17094   85087.0     1
5580    23904.0     1
5575    23873.0     1
17100   85110.0     1
5573    23864.0     1
17106   85141.0     1
17107   85143.0     1
5563    23824.0     1
17110   85154.0     1
17112   85160.0     1
17115   85166.0     1
5555    23785.0     1
17117   85173.0     1
5552    23766.0     1
17120   85179.0     1
17122   85188.0     1
17124   85191.0     1
17125   85192.0     1
26750  118753.0     1

[26751 rows x 2 columns]
```

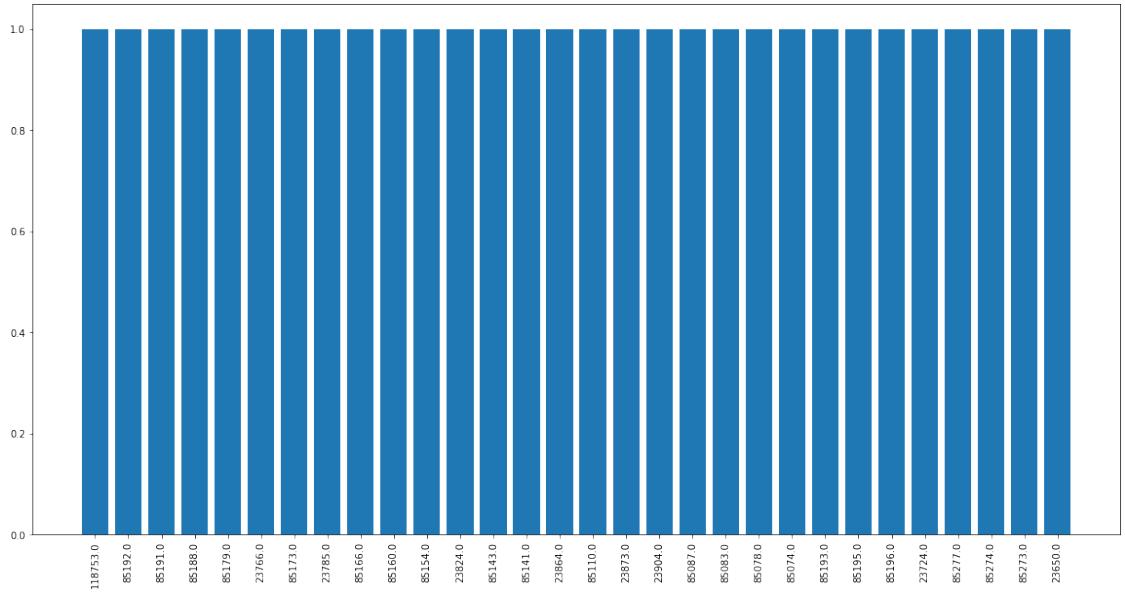
Number of answers posted v/s User ID (Best 30)

In [48]:
```
%matplotlib inline

plt.rcParams['figure.figsize'] = [20,10]
plt.xticks(rotation='vertical')
x=df_user['j'].values[:30]
y=[]
for i in x:
    i = str(i)
    y.append(i)
plt.bar(y, df_user['aid'].values[:30])
plt.show()
```



Number of answers posted v/s User ID (Worst 30)

```
In [49]: %matplotlib inline

         plt.rcParams['figure.figsize'] = [20,10]
         plt.xticks(rotation='vertical')
         x=df_user['j'].values[-30:][::-1]
         y=[]
         for i in x:
             i = str(i)
             y.append(i)
         plt.bar(y, df_user['aid'].values[-30:][::-1])
         plt.show()
```



1.2.9 Part 8

Extracting the number of views wrt individual tags

```
In [50]: df_q_score = df[[ 'tags', 'qid', 'qvc']]
         df_q_score = df_q_score.drop_duplicates()
         xt = df_q_score['tags'].values
         x2 = df_q_score['qvc'].values
         #df_q_score_sep = pd.DataFrame(columns=['tag', 'qid', 'qs'])
         rows = []
         for i in range(len(x2)):
             k = xt[i].split(',')
             for j in k:
                 #df_q_score_sep = df_q_score_sep.append({'tag':j, 'qid':x1[i], 'qs':x2[i]}, ig
                 l1 = [j, x2[i]]
                 rows.append(l1)

         df_q_score_sep = pd.DataFrame(rows, columns=['tag', 'qvc'])
         df_q_score_sep
```

Out[50]:

	tag	qvc
0	php	220
1	error	220
2	gd	220
3	image-processing	220
4	lisp	1047
5	scheme	1047
6	subjective	1047

7	clojure	1047
8	cocoa-touch	108
9	objective-c	108
10	design-patterns	108
11	core-animation	179
12	django	247
13	django-models	247
14	aspûnet	48
15	scala	276
16	pattern-matching	276
17	oop	276
18	object-oriented-design	276
19	design-principles	276
20	jquery	627
21	javascript	627
22	dom	627
23	winforms	215
24	gridview	215
25	ûnet	215
26	python	204
27	http	204
28	vim	61
29	netrw	61
...	...	...
254190	html	20
254191	hidden	20
254192	cross-domain	12
254193	webbrowser	12
254194	same-origin-policy	12
254195	msvcrt	13
254196	x64	13
254197	java	20
254198	jdbc	20
254199	beginner	46
254200	python	46
254201	programming	76
254202	bitmap	25
254203	3d	25
254204	c#	42
254205	lists	42
254206	c#	31
254207	httpwebrequest	31
254208	httpwebresponse	31
254209	sql	23
254210	transactions	23
254211	network-programming	37
254212	beginner	91
254213	self-learning	91

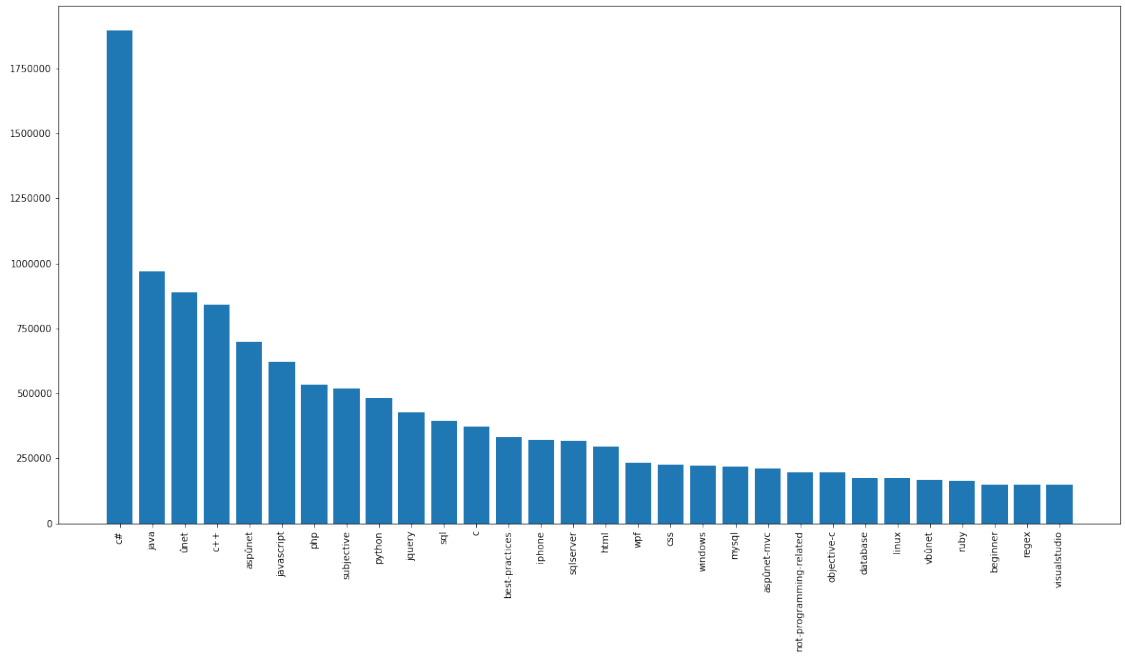
```
254214                      c#    26
254215    string-manipulation    26
254216                   ûnet    42
254217                   msil    42
254218                history    42
254219                    css     5

[254220 rows x 2 columns]
```

Adding all the views tag-wise and sorting them

```
In [51]: df_q_score_sep = df_q_score_sep.groupby(['tag']).sum().reset_index(level=0).sort_value
```

Numbers of views v/s Tag (Best 30)

```
In [52]: %matplotlib inline

         plt.rcParams['figure.figsize'] = [20,10]
         plt.xticks(rotation='vertical')
         plt.bar(df_q_score_sep['tag'].values[:30], df_q_score_sep['qvc'].values[:30])
         plt.show()
```



Numbers of views v/s Tag (Worst 30)

```
In [53]: %matplotlib inline
```

```
plt.rcParams['figure.figsize'] = [20,10]
plt.xticks(rotation='vertical')
plt.bar(df_q_score_sep['tag'].values[-30:][::-1], df_q_score_sep['qvc'].values[-30:][
plt.show()
```