In [335]:
```
import pandas as pd
import numpy as np
```

In [336]:
```
# Most seen delegates again, but they don't have to be officially listed a
results_path = 'WCA_export_Results.tsv'
results_df = pd.read_table(results_path)[['competitionId', 'personId']]
results_df.head()
```

Out[336]:

	competitionId	personId
0	LyonOpen2007	2007AMAN01
1	LyonOpen2007	2004ROUA01
2	LyonOpen2007	2005SIMO01
3	LyonOpen2007	2007MALL01
4	LyonOpen2007	2007DESM01

In [337]:
```
comp_path = 'WCA_export_Competitions.tsv'
comp_df = pd.read_table(comp_path)
fmc_only_comps = comp_df[comp_df['eventSpecs'] == '333fm'][['id']]
fmc_only_comp_ids = fmc_only_comps['id'].values

regular_comps = comp_df[~comp_df['id'].isin(fmc_only_comp_ids)][['id', 'wc
regular_comp_results = results_df[~results_df['competitionId'].isin(fmc_on
```

In [338]:
```
regular_comp_results = regular_comp_results.drop_duplicates()
```

In [339]:
```
def split_delegates(x):
    delegates = x.split('] ')
    delegate_names = []
    for d in delegates:
        delegate = d[2:d.index('}')]
        delegate_names.append(delegate)
    return delegate_names
regular_comps['delegates'] = regular_comps['wcaDelegate'].apply(split_dele
regular_comps = regular_comps.drop(columns=['wcaDelegate'])
```

In [340]:
```
# To find delegates who mapped to multiple ids (see below cell)
# for d, i in zip(regular_comps['delegates'], regular_comps['id']):
#     if 'Ray Li' in d:
#         print('found')
#         print(i)
```

In [341]:
```
delegate_names = list(set(regular_comps['delegates'].agg(sum)))
delegate_names = pd.DataFrame(delegate_names)
delegate_names = delegate_names.rename(columns={0:'name'})

persons_path = 'WCA_export_Persons.tsv'
persons_df = pd.read_table(persons_path)[['id', 'name']]
```

In [342]:
```
delegates_df = delegate_names.merge(persons_df, how='left', left_on='name'
delegates_df = delegates_df.drop_duplicates() # Reto Bubendorf appeared tw
# delegates_df[delegates_df['id'].isnull()] # Delegates who have never com
delegates_df = delegates_df.drop(axis=0, labels=[25,50,89,248]) # Drop the
```

In [343]:
```
# Fix delegates who when merged, mapped to multiple ids
delegates_to_fix = delegates_df[['name']].groupby('name').filter(lambda x:
delegates_to_fix['id'] = 0
delegates_to_fix.loc[delegates_to_fix['name'] == 'Gábor Szabó', 'id'] = '2
delegates_to_fix.loc[delegates_to_fix['name'] == 'Chenxi Shan (单晨曦)', 'id
delegates_to_fix.loc[delegates_to_fix['name'] == 'Zachary White', 'id'] =
delegates_to_fix.loc[delegates_to_fix['name'] == 'Ayush Kumar', 'id'] = '2
delegates_to_fix.loc[delegates_to_fix['name'] == 'Zheng Li (李政)', 'id'] =
delegates_to_fix.loc[delegates_to_fix['name'] == 'Artem Melikian (Артем Ме
delegates_to_fix.loc[delegates_to_fix['name'] == 'Evan Liu', 'id'] = '2009
delegates_to_fix.loc[delegates_to_fix['name'] == 'Caleb Hoover', 'id'] = '
delegates_to_fix.loc[delegates_to_fix['name'] == 'Jason White', 'id'] = '2
delegates_to_fix.loc[delegates_to_fix['name'] == 'Felix Lee', 'id'] = '200
delegates_to_fix.loc[delegates_to_fix['name'] == 'Ray Li', 'id'] = '2010LI
delegates_to_fix.loc[delegates_to_fix['name'] == 'Adam Walker', 'id'] = '2
delegates_to_fix_names = delegates_to_fix['name']
```

In [344]:
```
delegates_df = delegates_df.drop(delegates_df[delegates_df.name.isin(deleg
```

In [345]:
```
final_delegates = pd.concat([delegates_df, delegates_to_fix])
final_delegate_ids = list(final_delegates['id'])
```

In [346]:
```
def intersect(a, b):
    # Thanks https://www.saltycrane.com/blog/2008/01/how-to-find-intersect
    """ return the intersection of two lists """
    return list(set(a) & set(b))

def find_delegates(x):
    # Per competition, find the delegates that COMPETED there
    competitors = x['personId']
    return intersect(competitors, final_delegate_ids)

comp_to_delegates = regular_comp_results.groupby('competitionId').apply(fi
comp_to_delegates = comp_to_delegates.rename(columns={0:"delegateIds"})
comp_to_delegates.head()
```

Out[346]:

	competitionId	delegateIds
0	100Merito2018	[2014SANT16, 2007CINO01]
1	12SidesofSilesia2018	[2012TRZA01]
2	150thCubeMeetinginBrest2017	[2012TERE01, 2014PYAT01]
3	1AVG2013	[2003BRUC01, 2010BICL01, 2003VAND01, 2006BUUS01]
4	1BodyCubing2017	[2009BLAI01, 2005HAYE01]

In [351]:
```
person_to_delegates = regular_comp_results.merge(comp_to_delegates, how='i
person_to_delegates = person_to_delegates[['personId', 'delegateIds']]
person_to_delegates = person_to_delegates.groupby('personId').agg(sum)
```

In [352]:
```
person_to_delegates['delegateIds'] = person_to_delegates['delegateIds'].ap
person_to_delegates = person_to_delegates.reset_index()
```

In [353]:
```
person_to_delegates['numDelegates'] = person_to_delegates['delegateIds'].a
```

In [355]:
```
final_df = person_to_delegates.sort_values(by='numDelegates', ascending=Fa
```

In [360]:
```
final_df.merge(persons_df, how='inner', left_on='personId', right_on='id')
```

Out[360]:

	name	id	numDelegates	delegateIds
0	Mats Valk	2007VALK01	240	{2003BRUC01, 2009LIUE01, 2009JARU02, 2013BULM0...
1	Ron van Bruchem	2003BRUC01	230	{2003BRUC01, 2009LIUE01, 2009JARU02, 2007CINO0...
2	Shelley Chang	2004CHAN04	224	{2003BRUC01, 2009LIUE01, 2009JARU02, 2013BULM0...
3	Jeremy Fleischman	2005FLEI01	223	{2003BRUC01, 2009LIUE01, 2009JARU02, 2013BULM0...
4	Feliks Zemdegs	2009ZEMD01	222	{2003BRUC01, 2009LIUE01, 2009JARU02, 2013BULM0...
5	Hanneke Rijks	2008RIJK01	219	{2003BRUC01, 2009LIUE01, 2009JARU02, 2007CINO0...
6	Jasmine Lee	2003LEEJ01	217	{2003BRUC01, 2009LIUE01, 2009JARU02, 2013BULM0...
7	Tim Reynolds	2005REYN01	216	{2003BRUC01, 2009LIUE01, 2009JARU02, 2013BULM0...
8	Chris Hardwick	2003HARD01	213	{2003BRUC01, 2009LIUE01, 2009JARU02, 2007CINO0...
9	Sinpei Araki (荒木慎平)	2006ARAK01	212	{2003BRUC01, 2009LIUE01, 2009JARU02, 2013BULM0...
10	Lars Vandenbergh	2003VAND01	207	{2003BRUC01, 2009LIUE01, 2009JARU02, 2007CINO0...
11	Anthony Brooks	2008SEAR01	205	{2003BRUC01, 2009LIUE01, 2009JARU02, 2013BULM0...
12	Anthony Brooks	2008SEAR01	205	{2003BRUC01, 2009LIUE01, 2009JARU02, 2013BULM0...
13	Lucas Garron	2006GARR01	205	{2003BRUC01, 2009LIUE01, 2009JARU02, 2013BULM0...
14	Kevin Hays	2009HAYS01	204	{2003BRUC01, 2009LIUE01, 2009JARU02, 2013BULM0...
15	Takao Hashimoto (橋本貴夫)	2007HASH01	204	{2003BRUC01, 2009LIUE01, 2009JARU02, 2013BULM0...
16	István Kocza	2005KOCZ01	203	{2003BRUC01, 2009LIUE01, 2009JARU02, 2007CINO0...
17	Breandan Vallance	2007VALL01	203	{2003BRUC01, 2009LIUE01, 2009JARU02, 2007CINO0...
18	Sébastien Auroux	2008AURO01	202	{2003BRUC01, 2009LIUE01, 2009JARU02, 2007CINO0...
19	Oscar Roth Andersen	2008ANDE02	200	{2003BRUC01, 2009LIUE01, 2009JARU02, 2007CINO0...
20	Cornelius Dieckmann	2009DIEC01	197	{2003BRUC01, 2009LIUE01, 2009JARU02, 2013BULM0...
21	Michał Pleskowicz	2009PLES01	196	{2003BRUC01, 2009LIUE01, 2009JARU02, 2007CINO0...

	name	id	numDelegates	delegateIds
22	Anti Ingel	2009INGE01	193	{2003BRUC01, 2009LIUE01, 2009JARU02, 2007CINO0...
23	Anders Larsson	2003LARS01	191	{2003BRUC01, 2009LIUE01, 2009JARU02, 2007CINO0...
24	Helmut Heilig	2010HEIL02	191	{2003BRUC01, 2009LIUE01, 2009JARU02, 2013BULM0...
25	Jayden McNeill	2012MCNE01	190	{2003BRUC01, 2009LIUE01, 2007CINO01, 2009PARE0...
26	Daniel Sheppard	2009SHEP01	190	{2003BRUC01, 2009LIUE01, 2009JARU02, 2007CINO0...
27	Antoine Cantin	2010CANT02	190	{2003BRUC01, 2009LIUE01, 2013BULM01, 2007CINO0...
28	Joey Gouly	2007GOUL01	188	{2003BRUC01, 2009LIUE01, 2009JARU02, 2007CINO0...
29	Jules Desjardin	2010DESJ01	188	{2003BRUC01, 2009LIUE01, 2009JARU02, 2007CINO0...
...	...	...	...	...
115309	Keshvi Shah	2017SHAH06	0	{}
115310	Siyuan Lü (吕思远)	2016LUSI02	0	{}
115311	Shousen Lu (路守森)	2016LUSH03	0	{}
115312	Rui Lu (卢瑞)	2016LURU02	0	{}
115313	Asher Ang	2018ANGA04	0	{}
115314	Marco Alejandro Martinez Lozano	2016LOZA03	0	{}
115315	Israel Andrade	2018ANDR14	0	{}
115316	Diego Manuel Anaya Gonzalez	2017GONZ05	0	{}
115317	Ariel Dean Daroch Gonzalez	2017GONZ04	0	{}
115318	Irvin Daniel Benavides Arzate	2015ARZA01	0	{}
115319	Heber Arturo Castro Luevano	2016LUEV01	0	{}
115320	Guanhan Lu (卢冠翰)	2016LUGU01	0	{}
115321	Haozhe Lü (吕浩哲)	2016LUHA01	0	{}
115322	Roberto Adbetzait Flores Luis	2016LUIS01	0	{}
115323	Phutthithon Tobias Anestad	2018ANES01	0	{}
115324	Thanathon Martin Anestad	2018ANES02	0	{}
115325	Fangxin Gong (巩方新)	2017GONG09	0	{}
115326	Zhenqing Luo (罗振庆)	2016LUOZ01	0	{}

	name	id	numDelegates	delegateIds
115327	Shuhan Wang (王姝涵)	2018WANG30	0	{}
115328	Erick Marcial Carmona Bellido	2016BELL05	0	{}
115329	Sabrina Alexandra Cedeño Pujota	2018PUJO01	0	{}
115330	Chengju Luo (罗丞局)	2016LUOC01	0	{}
115331	Pablo Torres Sainz	2018SAIN02	0	{}
115332	Siqian Wang (王思谦)	2018WANG36	0	{}
115333	Hengxin Jing (靖恒欣)	2013JING01	0	{}
115334	Xiaozhen Wang (王晓祯)	2018WANG38	0	{}
115335	Yinong Luo (罗旖浓)	2016LUOY05	0	{}
115336	Yiqin Jiang (江怡勤)	2013JIAN09	0	{}
115337	Yuxin Luo (罗雨欣)	2016LUOY07	0	{}
115338	Ethan Scott Lee De Sheen	2018SHEE01	0	{}

115339 rows × 4 columns

```
In [ ]:
```