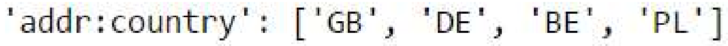
# Data Wrangling with MongoDB

By: Yee Jie Thay  
Map Area: Central London, United Kingdom  
Map Link: https://mapzen.com/data/metro-extracts-alt/odes/extracts/b19f74202960

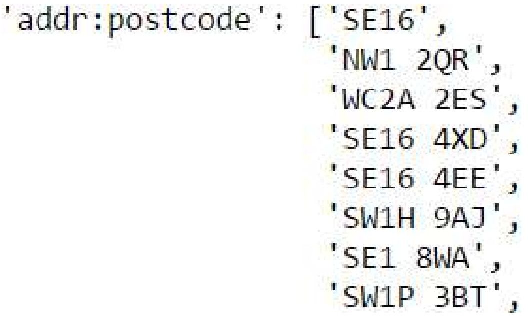
## Problems Encountered:

1. First and foremost, I noticed that there are information that are wrong or duplicated.
    a. This is a Central London map and hence any addresses that have a country code other than "GB" should be excluded
        i. We will look to exclude any addresses that has addr:country =/= "GB"

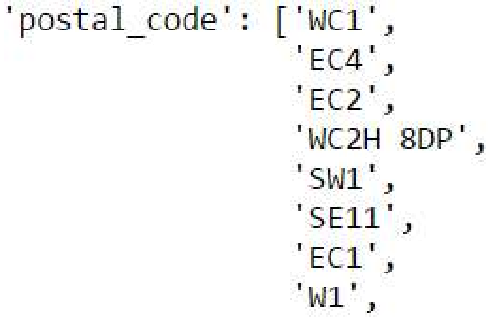


```
'addr:country': ['GB', 'DE', 'BE', 'PL']
```

2. Also there are 2 fields that have similar data
    a. We will merge the 2 fields into a single field "addr:postcode"



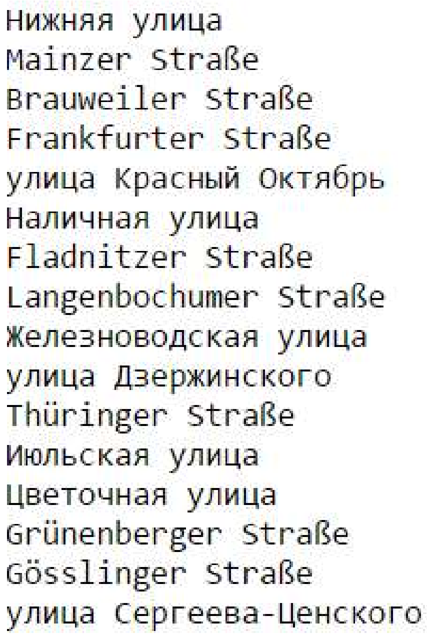
```
'addr:postcode': ['SE16',
                  'NW1 2QR',
                  'WC2A 2ES',
                  'SE16 4XD',
                  'SE16 4EE',
                  'SW1H 9AJ',
                  'SE1 8WA',
                  'SW1P 3BT',
```



```
'postal_code': ['WC1',
                'EC4',
                'EC2',
                'WC2H 8DP',
                'SW1',
                'SE11',
                'EC1',
                'W1',
```

3. Lastly, there are some street names that need to be standardized
    a. There are 2 types of standardization
        i. The last word of the street name that mean the same thing
            1. "Stamford street" can be standardized to "Stamford Street"
            2. "Soho St." can be standaridized into "Soho Street"
            3. "Shoreditch High St" can be standardized into "Shoreditch High Street"
        ii. Phrases in street name that mean the same thing
            1. "St Katharine's Way" can be standardized to "St. Katharine's Way"
            2. "Saint Katherines Way" can be standardized to "St. Katharines Way"
            3. "St. Katharines Way"" can be standardized to "St. Katharine's

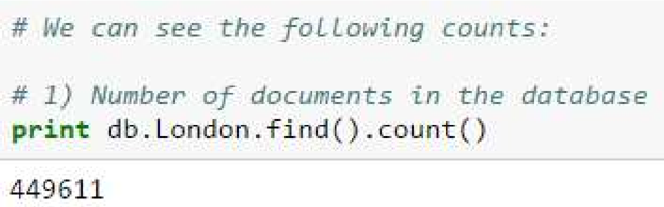
4. Also, I note that even after the removal of the non-GB countries, there are certain street names that are clearly not English as seen to the right
    a. For example, u'Stra\xdfe' and u'\u0443\u043b\u0438\u0446\u0430'
    b. I will delete entries that have these phrases as part of my data cleaning process



```
Нижняя улица
Mainzer Straße
Brauweiler Straße
Frankfurter Straße
улица Красный Октябрь
Наличная улица
Fladnitzer Straße
Langenbochumer Straße
Железноводская улица
улица Дзержинского
Thüringer Straße
Июльская улица
Цветочная улица
Grünenberger Straße
Gösslinger Straße
улица Сергеева-Ценского
```

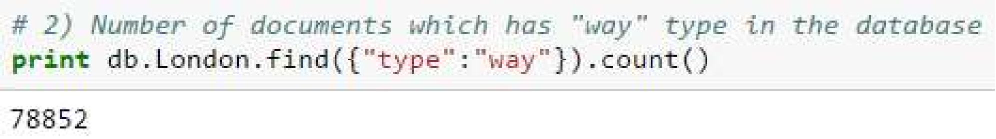
## Data Overview:

Using the following functions we can compile kind of information about the data we have on Central London.



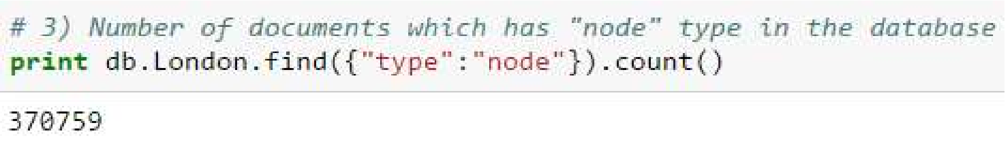
```
# We can see the following counts:

# 1) Number of documents in the database
print db.London.find().count()
```

```
449611
```



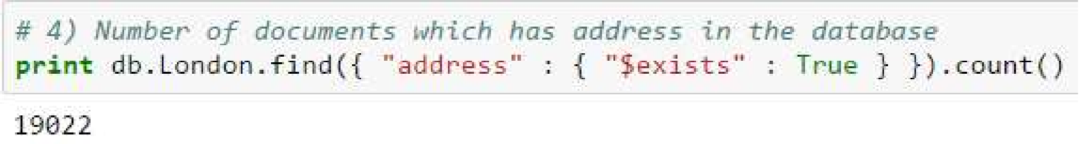
```
# 2) Number of documents which has "way" type in the database
print db.London.find({"type":"way"}).count()
```

```
78852
```



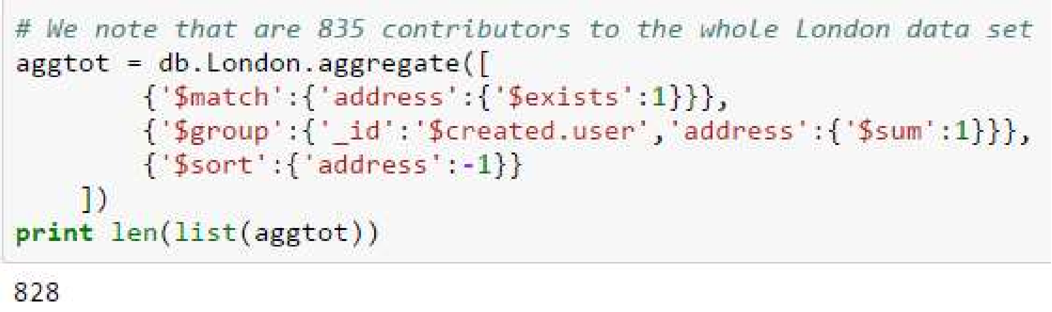
```
# 3) Number of documents which has "node" type in the database
print db.London.find({"type":"node"}).count()
```

```
370759
```



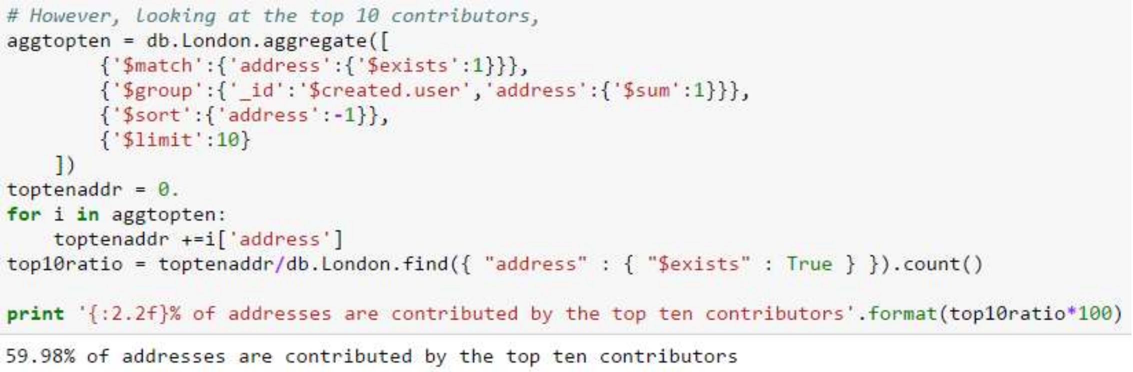
```
# 4) Number of documents which has address in the database
print db.London.find({ "address" : { "$exists" : True } }).count()
```

```
19022
```



```
# We note that are 835 contributors to the whole London data set
aggtot = db.London.aggregate([
        {'$match':{'address':{'$exists':1}}},
        {'$group':{'_id':'$created.user','address':{'$sum':1}}},
        {'$sort':{'address':-1}}
    ])
print len(list(aggtot))
```

```
828
```

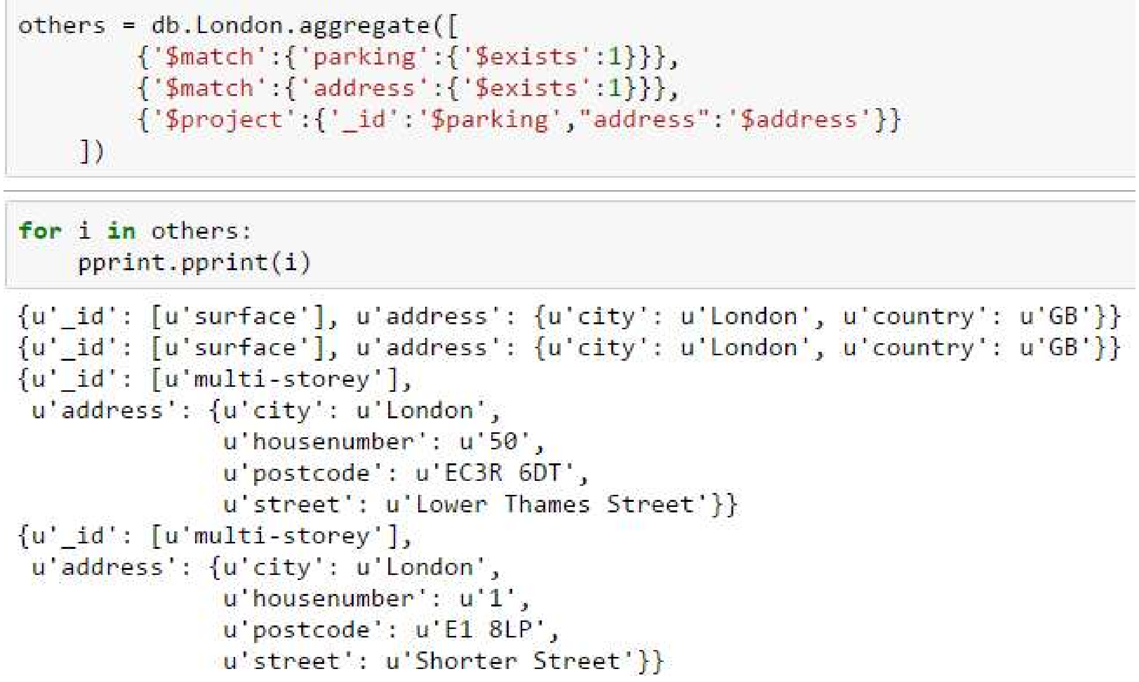


```
# However, looking at the top 10 contributors,
aggtopten = db.London.aggregate([
        {'$match':{'address':{'$exists':1}}},
        {'$group':{'_id':'$created.user','address':{'$sum':1}}},
        {'$sort':{'address':-1}},
        {'$limit':10}
    ])
toptenaddr = 0.
for i in aggtopten:
    toptenaddr +=i['address']
top10ratio = toptenaddr/db.London.find({ "address" : { "$exists" : True } }).count()

print '{:2.2f}% of addresses are contributed by the top ten contributors'.format(top10ratio*100)
```

```
59.98% of addresses are contributed by the top ten contributors
```

## Other ideas about the datasets:

1. Cross referencing
    a. Lat Long vs Address
        i. We can look to cross reference latitude longitude field with the address
        ii. This allows us to have an idea if the house numbers are correct for the data as typically, evens are on a side and odds on the oppsite side of the road in London
2. Map user contributions to address
    a. We can have a good idea of what areas certain users tend to contribute towards.
    b. With that information, we can infer user behaviour and also be able to have a sense of where and what are the areas these users belong to
    c. It can be useful when we approach gamification so as to encourage user participation
        i. We should be able to incentivise users to cross check the data if we are able to have a good sense of where they are based and hence lower the hurdle of someone to step out and cross check existing data.
3. We can flag out data that data that has been input in the wee hours of the day.
    a. Assuming, that most entries are done with some sort of human intervention
    b. These data would be more likely to be corrupted and thus should
4. Inclusion of photos of landmark:
    a. Will be incredibly useful for photo verification of locations
    b. This will tie in with the whole idea of gamification where geotagged photos by users can be included in the data set to allow people to contribute in a simply fashion (e.g. geotagged photo of the street name)
5. Generate a parking map based off the data
    a. Using the data, we should be able to create a reliable parking map



```
others = db.London.aggregate([
        {'$match':{'parking':{'$exists':1}}},
        {'$match':{'address':{'$exists':1}}},
        {'$project':{'_id':'$parking',"address":'$address'}}
    ])
```

```
for i in others:
    pprint.pprint(i)
```

```
{u'_id': [u'surface'], u'address': {u'city': u'London', u'country': u'GB'}}
{u'_id': [u'surface'], u'address': {u'city': u'London', u'country': u'GB'}}
{u'_id': [u'multi-storey'],
 u'address': {u'city': u'London',
              u'housenumber': u'50',
              u'postcode': u'EC3R 6DT',
              u'street': u'Lower Thames Street'}}
{u'_id': [u'multi-storey'],
 u'address': {u'city': u'London',
              u'housenumber': u'1',
              u'postcode': u'E1 8LP',
              u'street': u'Shorter Street'}}
```

## Conclusions:

I note that there are still a lot of work to be done before the data can be used in any meaningful way. If we are able to gamify the whole process, we should be able to get much more accurate data from OpenStreetMap.org. This can be extended to a very practical use in terms of parking. Any driver in London will know that it is incredibly difficult to find any sort of parking within London. If we can combine OpenStreetMap data and some gamification process where users can verify each others' entries, we will be able to come up with an app which will be incredibly useful for users