Data Wrangling with OpenStreetMap and MongoDB

Introduction

I have chosen Syracuse, New York and use data munging techniques, to assess the quality of the data for validity, accuracy, completeness, consistency and uniformity and to clean the OpenStreetMap data using MongoDB as the data schema. This following link is the map position of Syracuse in the OpenStreetMap http://www.openstreetmap.org/relation/174916#map=12/43.0353/-76.1394. Syracuse is a city in Onondaga County, New York, in the United States. It is the largest U.S. city and is the fourth most populous metropolitan city in the state of New York. OpenStreetMap is a map of the world, created by people and free to use under an open license. (http://www.openstreetmap.org)

OSM XML (http://wiki.openstreetmap.org/wiki/OSM_XML)is list of instances of data primatives (nodes, ways, and relations) found within a given bounds - nodes represent dimensionless points on the map - ways contain node references to form either a polyline or polygon on the map - nodes and ways both contain children tag elements that represent key value pairs of descriptive information about a given node or way.

Data Overwiew

In order to have a general overview of Syracuse OSM-XML data, Iterative parse through ElementTree is done to create a dictionary of all the tags available within the OSM XML file and the number of unique element types is count. Below is the total count of element types

```
{'bounds': 1,
 'member': 5556,
 'nd': 328273,
 'node': 279462,
 'osm': 1,
 'relation': 795,
 'tag': 215643,
 'way': 35500}
```

In order to see if there are any potential problems in the dataset, a count is conducted of each of three tag categories in a dictionary: "lower", for tags that contain only lowercase letters and are valid, "lower_colon", for otherwise valid tags with a colon in their names, "problemchars", for tags with problematic characters and "other", for other tags that do not fall into the other three categories. Here are the results;

```
{'lower': 99834, 'lower_colon': 108213, 'other': 7595, 'problemchars': 1}
```

The number of unique users that contributed to the map of Syracuse are found out to be 248.

Data Auditing (Problems with the Dataset)

The main problem encountered in this dataset (Syracuse, NY) are problems arising from inconsistent street name abbreviation and postcode (zip code) usage.

1. Street type issues

In order to deal with the street name inconsistency, Audit function is assigned to create a dictionary of street types, a string to audit, a regex to match against that string, and the list of expected street types. The function will search the string for the regex. If there is a match and the match is not in our list of expected street types, add the match as a key to the dictionary and add the string to the set. The is_street_name function looks at the attribute k if k="addre:street" audit function will return the list that match previous two functions. After that, we would print the output of the audit. With the list of all the abbreviated street types we can understand and fill-up our "mapping" dictionary to change to the correct form.

Fixing street names is done with a better name by mapping with list given below.

```
expected = ["Avenue", "Boulevard", "Commons", "Court", "Drive", "Lane",
                "Parkway", "Place", "Road", "Square", "Street", "Trail"]


mapping = {'Ave'  : 'Avenue',
            'Blvd' : 'Boulevard',
            'Dr'   : 'Drive',
            'Ln'   : 'Lane',
            'Pkwy' : 'Parkway',
            'Rd'   : 'Road',
            'St.'   : 'Street',
            'Rd.'   : 'Road',
            'St'   : 'Street',
            'street' :"Street",
            'Ct'   : "Court",
            'Cir'  : "Circle",
            'Cr'   : "Court",
            'ave'  : 'Avenue',
            'Hwg'  : 'Highway',
            'Hwy'  : 'Highway',
            'N'   : "North",
            'S'   : "South",
            'E'   : "East",
            'W'   : "West",
            'Sq'   : "Square"}
```

Below is a sample of the Street Audit out put for the Syracuse dataset (before and after fixitaion). only one street name appreviation was found, that is `'St': set(['James St'])`. In this case, Street is abbreviated as St and should be fixed to ‘James Street’.

Before cleaning;

```
'Courts': set(['Presidential Courts']),
 'East': set(['Erie Boulevard East', 'Lawrence Road East']),
 'Path': set(['Bronco Path']),
 'Plaza': set(['Presidential Plaza', 'Shop City Plaza']),
 'Rowe': set(['Basile Rowe']),
 'Run': set(['Trotters Ridge Run']),
 'St': set(['James St']),
```

After cleaning;

```
New York 31 => New York 31
State Route 31 => State Route 31
State Highway 31 => State Highway 31
Shop City Plaza => Shop City Plaza
Presidential Plaza => Presidential Plaza
James St => James Street
```

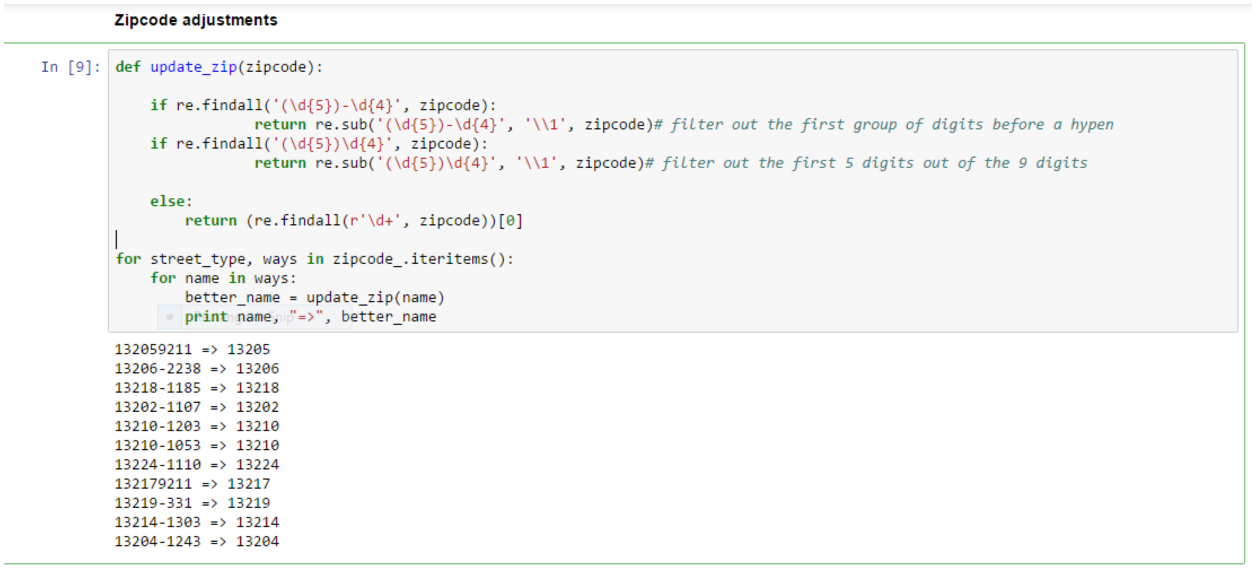
2. Post code problems

Post codes which were not of a valid format (‘dddd’) were considered as invalid by assigning a regex, re.match(r'^\d{5}$', zipcode) and results were fixed according the type of error the invalid postcodes were holding. In the Syracuse dataset, some zip codes were of 9 digits and some 9 digists but separated by a hypen after the 5th digit.

Before cleaning;

```
{None: set(['13202-1107',
            '13204-1243',
            '132059211',
            '13206-2238',
            '13210-1053',
            '13210-1203',
            '13214-1303',
            '132179211',
            '13218-1185',
            '13219-331',
            '13224-1110'])}
```

Code and after cleaning



Zipcode adjustments

```
In [9]: def update_zip(zipcode):

            if re.findall('(\d{5})-\d{4}', zipcode):
                        return re.sub('(\d{5})-\d{4}', '\\1', zipcode)# filter out the first group of digits before a hypen
            if re.findall('(\d{5})\d{4}', zipcode):
                        return re.sub('(\d{5})\d{4}', '\\1', zipcode)# filter out the first 5 digits out of the 9 digits

            else:
                return (re.findall(r'\d+', zipcode))[0]

        for street_type, ways in zipcode_.iteritems():
            for name in ways:
                better_name = update_zip(name)
                print name, "=>", better_name
```

```
132059211 => 13205
13206-2238 => 13206
13218-1185 => 13218
13202-1107 => 13202
13210-1203 => 13210
13210-1053 => 13210
13224-1110 => 13224
132179211 => 13217
13219-331 => 13219
13214-1303 => 13214
13204-1243 => 13204
```

Data formatting to json to fit into MongoDB

In order to load the XML data into MongoDB, data ise transformed into json structure by taking into consideration the following points in the 'shape_element' function.

only 2 types of top level tags: "node" and "way" are processed. All attributes of "node" and "way" are turned into regular key/value pairs, except:

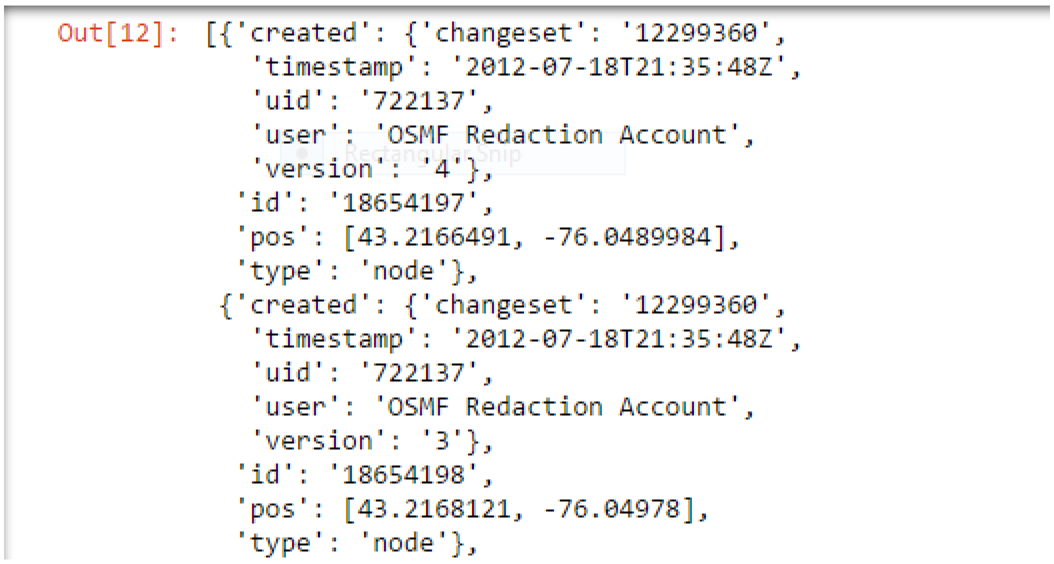
- the following attributes are added under a key created: version, changeset, timestamp, user, uid
- attributes in the CREATED array are added under a key "created"
- values of pos array are set to floats.
- attributes for latitude and longitude are added to a "pos" array,

second level tag "k" value contains problematic characters, are ignored

second level tag "k" value starts with "addr:", is added to a dictionary "address"

second level tag "k" value does not start with "addr:", but contains ":", are converted to an updated key

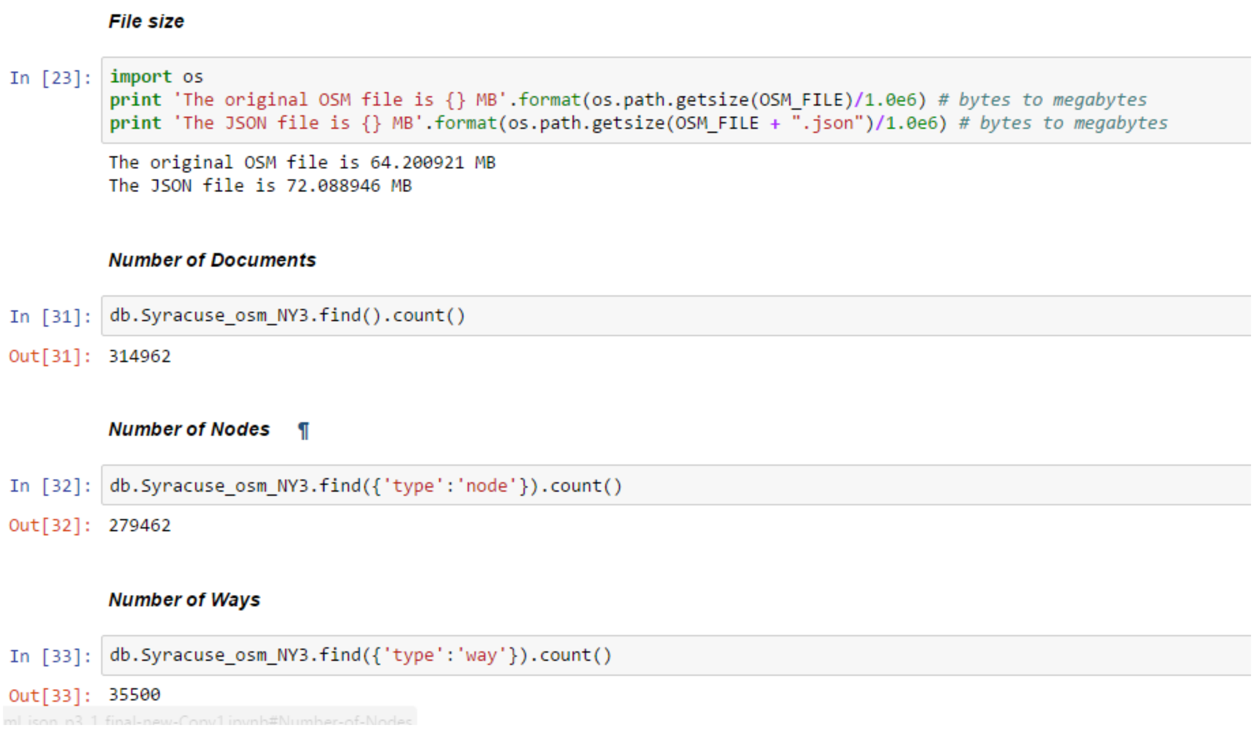
Syracuse data Json format



```
Out[12]: [{'created': {'changeset': '12299360',
             'timestamp': '2012-07-18T21:35:48Z',
             'uid': '722137',
             'user': 'OSMF Redaction Account',
             'version': '4'},
            'id': '18654197',
            'pos': [43.2166491, -76.0489984],
            'type': 'node'},
           {'created': {'changeset': '12299360',
             'timestamp': '2012-07-18T21:35:48Z',
             'uid': '722137',
             'user': 'OSMF Redaction Account',
             'version': '3'},
            'id': '18654198',
            'pos': [43.2168121, -76.04978],
            'type': 'node'},
```

Syracuse MongoDB data investigation

The open street map file for Syracuse, NY is a size of about 72MB json format and has 314,962 documents, 279,460 number of Nodes and 35,483 number of Ways. So far, 234 users has contributed to the Syracuse open street map.



File size

```
In [23]: import os
         print 'The original OSM file is {} MB'.format(os.path.getsize(OSM_FILE)/1.0e6) # bytes to megabytes
         print 'The JSON file is {} MB'.format(os.path.getsize(OSM_FILE + ".json")/1.0e6) # bytes to megabytes

         The original OSM file is 64.200921 MB
         The JSON file is 72.088946 MB
```

Number of Documents

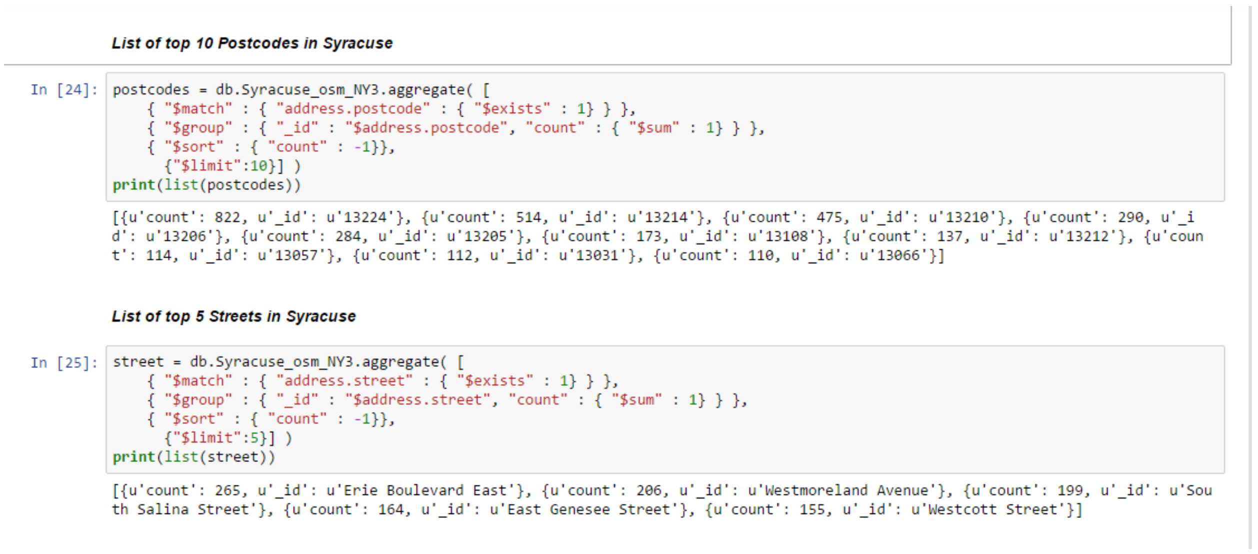
```
In [31]: db.Syracuse_osm_NY3.find().count()
Out[31]: 314962
```

Number of Nodes ¶

```
In [32]: db.Syracuse_osm_NY3.find({'type':'node'}).count()
Out[32]: 279462
```

Number of Ways

```
In [33]: db.Syracuse_osm_NY3.find({'type':'way'}).count()
Out[33]: 35500
```



List of top 10 Postcodes in Syracuse

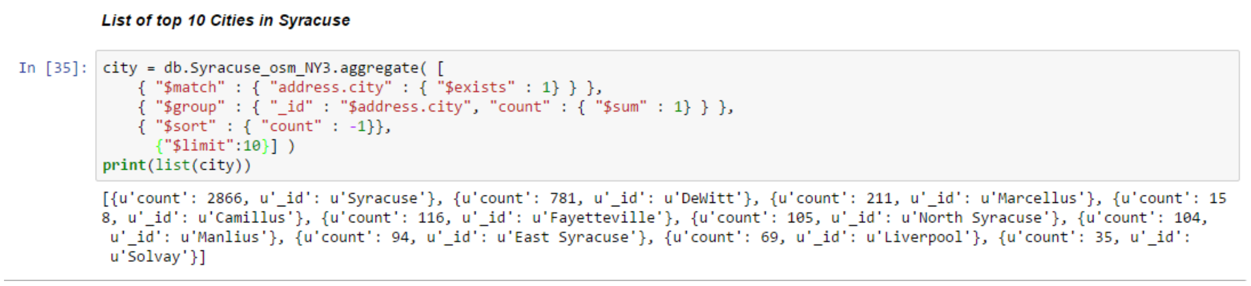
```
In [24]: postcodes = db.Syracuse_osm_NY3.aggregate( [
             { "$match" : { "address.postcode" : { "$exists" : 1} } },
             { "$group" : { "_id" : "$address.postcode", "count" : { "$sum" : 1} } },
             { "$sort" : { "count" : -1}},
               {"$limit":10}] )
         print(list(postcodes))

         [{u'count': 822, u'_id': u'13224'}, {u'count': 514, u'_id': u'13214'}, {u'count': 475, u'_id': u'13210'}, {u'count': 290, u'_i
         d': u'13206'}, {u'count': 284, u'_id': u'13205'}, {u'count': 173, u'_id': u'13108'}, {u'count': 137, u'_id': u'13212'}, {u'coun
         t': 114, u'_id': u'13057'}, {u'count': 112, u'_id': u'13031'}, {u'count': 110, u'_id': u'13066'}]
```

List of top 5 Streets in Syracuse

```
In [25]: street = db.Syracuse_osm_NY3.aggregate( [
             { "$match" : { "address.street" : { "$exists" : 1} } },
             { "$group" : { "_id" : "$address.street", "count" : { "$sum" : 1} } },
             { "$sort" : { "count" : -1}},
               {"$limit":5}] )
         print(list(street))

         [{u'count': 265, u'_id': u'Erie Boulevard East'}, {u'count': 206, u'_id': u'Westmoreland Avenue'}, {u'count': 199, u'_id': u'Sou
         th Salina Street'}, {u'count': 164, u'_id': u'East Genesee Street'}, {u'count': 155, u'_id': u'Westcott Street'}]
```

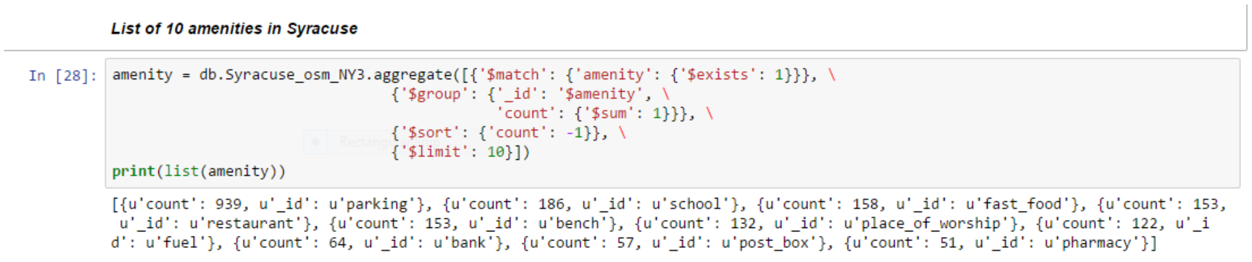


List of top 10 Cities in Syracuse

```
In [35]: city = db.Syracuse_osm_NY3.aggregate( [
             { "$match" : { "address.city" : { "$exists" : 1} } },
             { "$group" : { "_id" : "$address.city", "count" : { "$sum" : 1} } },
             { "$sort" : { "count" : -1}},
               {"$limit":10}] )
         print(list(city))

         [{u'count': 2866, u'_id': u'Syracuse'}, {u'count': 781, u'_id': u'DeWitt'}, {u'count': 211, u'_id': u'Marcellus'}, {u'count': 15
         8, u'_id': u'Camillus'}, {u'count': 116, u'_id': u'Fayetteville'}, {u'count': 105, u'_id': u'North Syracuse'}, {u'count': 104,
          u'_id': u'Manlius'}, {u'count': 94, u'_id': u'East Syracuse'}, {u'count': 69, u'_id': u'Liverpool'}, {u'count': 35, u'_id':
          u'Solvay'}]
```

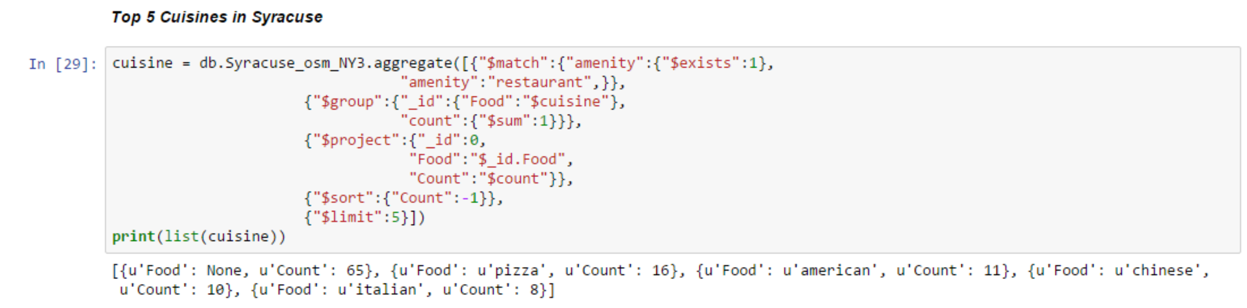
I found it very challenging to see many cities other than Syracuse in this dataset. I wish I know this place in the New York to come up with a good answer. However, I checked in the internet to find out what could be the reason behind. I realized further investigation is required which I can't do in my situation. In the following site https://www.google.com/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=google%20map%20syracuse%20ny , it is depicted that the openstreetmap boundary for Syracuse also includes cities in the boundary like DeWitt, East Syracuse and Salvoy. However, these cities can be considered as part of Syracuse city. I have the feeling that including cities like Fayetteville, Liverpool, Nedrow and Minoa, which are in the County of Onondaga County, where Syracuse is one of the cities in this county, in the map of Syracuse can be misleading. This is only my opinion, I hope other contributors will come up with a better understanding and explanation.



List of 10 amenities in Syracuse

```
In [28]: amenity = db.Syracuse_osm_NY3.aggregate([{'$match': {'amenity': {'$exists': 1}}}, \
                                          {'$group': {'_id': '$amenity', \
                                                      'count': {'$sum': 1}}}, \
                                          {'$sort': {'count': -1}}, \
                                          {'$limit': 10}])
         print(list(amenity))

         [{u'count': 939, u'_id': u'parking'}, {u'count': 186, u'_id': u'school'}, {u'count': 158, u'_id': u'fast_food'}, {u'count': 153,
          u'_id': u'restaurant'}, {u'count': 153, u'_id': u'bench'}, {u'count': 132, u'_id': u'place_of_worship'}, {u'count': 122, u'_i
         d': u'fuel'}, {u'count': 64, u'_id': u'bank'}, {u'count': 57, u'_id': u'post_box'}, {u'count': 51, u'_id': u'pharmacy'}]
```

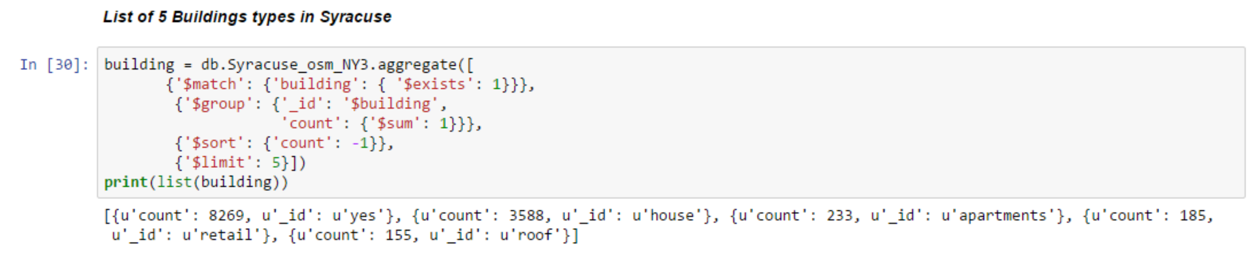


Top 5 Cuisines in Syracuse

```
In [29]: cuisine = db.Syracuse_osm_NY3.aggregate([{"$match":{"amenity":{"$exists":1},
                                          "amenity":"restaurant",}},
                               {"$group":{"_id":{"Food":"$cuisine"},
                                          "count":{"$sum":1}}},
                               {"$project":{"_id":0,
                                           "Food":"$_id.Food",
                                           "Count":"$count"}},
                               {"$sort":{"Count":-1}},
                               {"$limit":5}])
         print(list(cuisine))

         [{u'Food': None, u'Count': 65}, {u'Food': u'pizza', u'Count': 16}, {u'Food': u'american', u'Count': 11}, {u'Food': u'chinese',
          u'Count': 10}, {u'Food': u'italian', u'Count': 8}]
```

It is very interesting to notice that, disregarding the 325 unclassified cusines, not only American cuisines but also Chinese and Italian cuisines are very popular in Syracuse. Pizza seems to be very popular too. From the results we can understand that, there is still a need for Syracuse map to be cleaned and updated. Then only we will be able to do a good analysis.



List of 5 Buildings types in Syracuse

```
In [30]: building = db.Syracuse_osm_NY3.aggregate([
                {'$match': {'building': { '$exists': 1}}},
                 {'$group': {'_id': '$building',
                             'count': {'$sum': 1}}},
                 {'$sort': {'count': -1}},
                 {'$limit': 5}])
         print(list(building))

         [{u'count': 8269, u'_id': u'yes'}, {u'count': 3588, u'_id': u'house'}, {u'count': 233, u'_id': u'apartments'}, {u'count': 185,
          u'_id': u'retail'}, {u'count': 155, u'_id': u'roof'}]
```

The same goes for the 8269 buildings which their type is not classified. This shows there is a need for improvement. Majority of the buildings in Syracuse seems to be houses, followed by apartments.

Conclusion and Suggestion

Eventhough, OpenStreetMap is an open source and is subject to errors caused by humans, the dataset is fairly well handled and well organized. Taking the number of users into account, it would have been great if some sort of structured input is prepared for all users to track down any activity done to minimize more errors.

The benefits of using OpenStreetMap is that, it allow users to apply whatever knowledge they have to make decisions on correcting faulty errors. A user who made adjustments to a known place (home town, a place where the person know details of the place very well, is by far better than the user who try to adjust based on figures generated by a computer. The fact that it is open gives the user an opportunity to work on known places. For instance, with regard to deciding which cities truly belong to Syracuse, I found it very difficult to take decisions or at least to make suggestion to a problem I noticed since I don't know the city which I choose to analyze. It would have been much easier to do known places. Unfortunately my city data size is below the criteria.

Since something is better than nothing, I would also suggest if people are made to give attention to the places where their dataset is never been accessed or only few users have so far contributed. One possible way of doing this could be generating of statistics for less frequently used datasets and inviting users (attracting users) who already contributed a lot and being belong to the place somehow or another.

It is extremely challenging to rely on findings sourced from a system where anyone can have access and make changes. On the other hand, it is difficult to put a restriction and to bring a vital and sound solution to a very big datasets like OpenStreetMap where no one knows everything. There should be a mechanism to bring these extreme realities into a balanced state. I suggest users have access to a log of history where any change made is tracked down and shared. It is a collective work, and only voluntary contributors can bring the concern less, and the maximum achievement.

References:

1. https://www.youtube.com/watch?v=ZdDOauFIDkw
2. https://docs.python.org/3/howto/regex.html#performing-matches
3. http://napitupulu-jon.appspot.com/posts/wrangling-openstreetmap.html
4. https://jameskao.me/everything-you-need-to-know-about-facebooks-flux-graphql-and-relay/
5. https://github.com/vaikunthkannan/datawrangling_mongodb/blob/master/audit.py
6. http://stackoverflow.com/questions/6497293/remove-in-zipcodes/6497322#6497322
7. https://www.google.com/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=syracuse%2C%20ny