Project: Wrangle OpenStreetMap Data: **San-Jose**

Author: Ashish Sharma
Date: Jan 31, 2018

Objective:

Audit, clean the OSM dataset, re-shape it to an appropriate data model and convert into JSON format and analyze insight within the data.

Map Area:

I am using the OpenStreetMap data of **San Jose, CA** area downloaded from **https://s3.amazonaws.com/metro-extracts.mapzen.com/san-jose_california.osm.bz2**, on Jan 7, 2018.

Introduction:

City of San Jose, is the third most populous city in California and the tenth most populous in United States.
It is an economic, cultural, and political center of Silicon Valley and the largest city in Northern California.
Therefore, I find it interesting to investigate and gain insights about this cosmopolitan city.

Data Overview:

The downloaded map is in OpenStreetMap XML format, description of which can be found at
https://wiki.openstreetmap.org/wiki/OS_XML

Key elements of the OSM file:

A **node** is one of the core elements in the OpenStreetMap data model. It consists of a single point in space defined by its latitude, longitude and node id.

A **way** is an ordered list of between 2 and 2,000 nodes that define a polyline. Ways are used to represent linear features such as rivers and roads.

A **relation** is a multi-purpose data structure that documents a relationship between two or more data elements (nodes, ways, and/or other relations).

Examples include:

A **route relation**, which lists the ways that form a major (numbered) highway, a cycle route, or a bus route.
A **turn restriction** that says you can't turn from one way into another way.
A **multipolygon** that describes an area (whose boundary is the 'outer way') with holes (the 'inner ways').

All types of data element (nodes, ways and relations), as well as change sets, can have tags.
Tags describe the meaning of the specific element to which they are attached.

List of Files:

Filename	Description
modules.py	Contains modules referred by other source file and location and name for data files.
data_audit.py	Contains functions to parse, investigate data set and identify problems with the data
postal_codes.py	Contains functions to validate, audit, process and update address:postcode tag.
phone_nums.py	Contains functions to validate, audit, process and update phone number value, phone tag
street_types.py	Contains functions to validate, audit, process and update address:street and address:full tags
data_explore.py	Contains functions to gain insights using MongoDB.
data_wrangle.py	Contains functions to process the osm xml and reshape it into json format
san-jose_california.osm	Original downloaded OSM data file.
MapArea.txt	Specifies of the openstreet map used for investigation.

Step1 - Data Audit source codes: data_audit.py, postal_codes.py, phone_nums.py, street_types.py

Original File Size: 388 MB

```
'Original OSM XML File : data/san-jose_california.osm, Size : 388 MB'
```

- ### Coordinates Box Bounds:

  Maps stretches from Felton, CA 95018, USA to Livermore, CA 94550, USA

  Reference: https://www.latlong.net/Show-Latitude-Longitude.html

```
Map Coordinates:
bounds {'minlat': '37.125', 'maxlon': '-121.589', 'minlon': '-122.046', 'maxlat': '37.469'}
```

- ### Top Level Elements with occurrence counts:
  - There are 1872967 nodes defined in this data set.
  - There are 247157 ways defined in this data set which uses 2186910 nodes references
  - There are 2636 relations constituting 22054 members (nodes and ways)
  - There are 774266 tags defined describing more details for above elements

```
'Unique Top Level Elements:'
{'bounds': 1,
 'member': 22054,
 'nd': 2186910,
 'node': 1872967,
 'osm': 1,
 'relation': 2636,
 'tag': 774266,
 'way': 247157}
```

1. ### Identify problematic characters within attribute named 'k':

   A **tag** consists of two items, a key and a value.
   Tags describe specific features of map elements (nodes, ways, or relations) or changesets.
   Both items are free format text fields, but often represent numeric or other structured items
   To extract information from this tag, we need to ensure it does not have characters issues.

   No problematic characters identified in the data set:

```
Character types in k tag:
{'lower': 2939, 'lower_colon': 234566, 'other': 536761, 'problemchars': 0}
```

   There are ~22K addr_housenumber tags, followed by same number of street addresses.

```
Different Address types within k attribute
{'addr:city': 6109,
 'addr:city_1': 1,
 'addr:country': 905,
 'addr:county': 738,
 'addr:flats': 20,
 'addr:floor': 4,
 'addr:full': 36,
 'addr:housename': 112,
 'addr:housenumber': 22447,
 'addr:housenumber_1': 4,
 'addr:housenumber_2': 4,
 'addr:housenumber_3': 2,
 'addr:housenumber_4': 1,
 'addr:housenumber_5': 1,
 'addr:housenumber_6': 1,
 'addr:interpolation': 112,
 'addr:place': 1,
 'addr:postcode': 13173,
 'addr:state': 2184,
 'addr:street': 22066,
 'addr:street:source': 2,
 'addr:street_1': 1,
 'addr:suite': 3,
 'addr:unit': 214,
 'address': 4}
```

2. ### Manually review random sample for MongoDB conversion:

   In order to further validate the inner tags for MongoDB keys, sample data was dumped in json format

```
'Sample JSON File : data/san-jose_california_tag_sample.json, Size : 253381 bytes'
```

Step 2- Problems Encountered in the map: source codes: postal_codes.py, phone_nums.py, street_types.py

This dumped **data/san-jose_california_tag_sample.json** file was used to study the tag attributes to identify potential problems in the dataset.

3. A subset of issues that was found and were explicitly cleaned were:

1. **Abbreviated street names**: the **addr:street** and **addr:full** tags showed that street types were often abbreviated (St. for Street, etc.).
   Data is cleaned to replace abbreviations with the full street type using below mapping:

```
MAP_STREET_TYPE = \
{           'Ave'  : 'Avenue',
            'Blvd' : 'Boulevard',
            'Blvd.' : 'Boulevard',
            'Dr'   : 'Drive',
            'Ln'   : 'Lane',
            'Pkwy' : 'Parkway',
            'Rd'   : 'Road',
            'Rd.'   : 'Road',
            'St'   : 'Street',
            'street' :"Street",
            'Ct'     : "Court",
            'Cir'    : "Circle",
            'Cr'     : "Court",
            'ave'   : 'Avenue',
            'Hwg'  : 'Highway',
            'Hwy'  : 'Highway',
            'Sq'    : "Square",
            'Ter'   : "Terrace",
            'Expy'  : "Expressway",
            'Ste'   : "Suite"
}
```

2. **Phone numbers**: the phone tag contains the phone numbers and were present in all different formats. Data is transformed into standard format.

```
4089882555 => +14089882555
+1 408 971 3977 => +14089713977
(408) 354-7365 => +14083547365
+1 408 988 1883 => +14089881883
+1-408-971-1160 => +14089711160
+1.408.736.6688 => +14087366688
+1-408-988-5407 => +14089885407
408.266.6342 => +14082666342
+1 408-500-3000 \u200e => +14085003000
```

3. **Postal Code:** The "**addr:postcode**" tag contained non digits values and prefixed with state codes in certain code e.g. CA 95014. This which was cleaned.

4. **Cuisine:** It was identified that cuisine tag can have multiple values separated by delimiter - either **';'** or **','**
   Data was converted into a list before dumping into json file. It was further standardized with user defined mapping for categorization.

5. **Few restaurants and Cafés** names were also standardized as detected during manual review for sample data.

6. **Removed tags** which marked as '**disused:**' from final json.

```
<tag k="disused:name" v="Johnsville Mobile Home Park"/>
<tag k="disused:landuse" v="residential"/>
<tag k="disused:phone" v="+1 408 789 0037"/>
<tag k="disused:amenity" v="restaurant"/>
<tag k="disused:cuisine" v="italian"/>
<tag k="disused:website:official" v="http://www.pastapomodoro.com/"/>
<tag k="disused:aeroway" v="helipad"/>
<tag k="disused:amenity" v="restaurant"/>
<tag k="disused:shop" v="party"/>
<tag k="disused:amenity" v="restaurant"/>
<tag k="disused:cuisine" v="brazilian"/>
<tag k="disused:cuisine" v="pizza"/>
<tag k="disused:amenity" v="restaurant"/>
<tag k="disused:cuisine" v="italian"/>
```

Step 3 – Data wrangling source code: data_wrangle.py
Transforming OSM xml into json and further import into MongoDB

4. The dataset was transformed into json format using shape_element.py:

```
ashish@ubuntu:~/DataAnalyst/Project_Wrangle_OSM$ python data_wrangle.py
data_wrangle.py script Started..
'Final clean JSON File : data/san-jose_california_clean.json, Size : 446 MB'
data_wrangle.py script Ended.Duration:  228.909 s
```

5. The final json file is imported into MongodB using below command:

mongoimport –db osm –collection sanjose –drop –file data/san-jose_california_clean.json

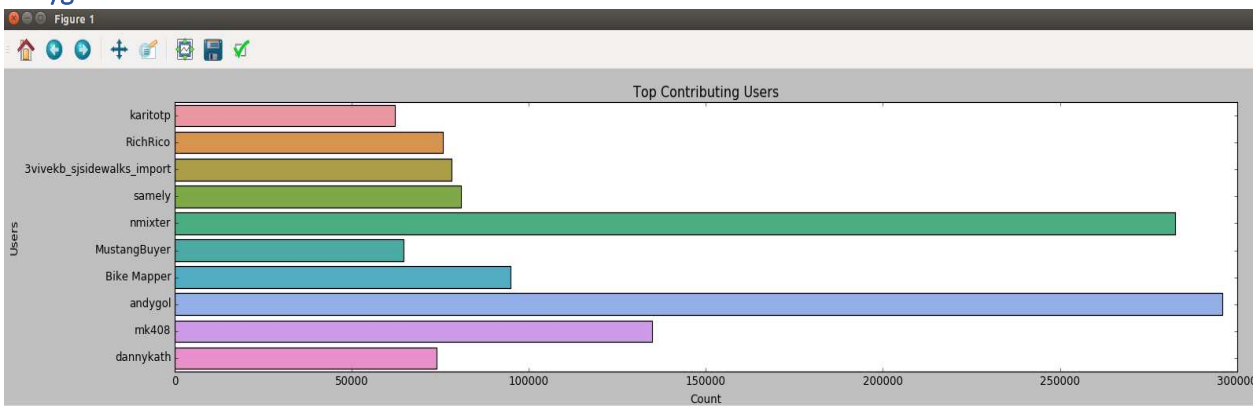
```
ashish@ubuntu:~/DataAnalyst/Project_Wrangle_OSM$ mongoimport --db osm --collection sanjose --drop --file data/san-jose_california_clean.json
2018-02-05T10:24:50.750-0800    connected to: localhost
2018-02-05T10:24:50.750-0800    dropping: osm.sanjose
2018-02-05T10:24:53.744-0800    [........................] osm.sanjose  7.04MB/446MB (1.6%)
2018-02-05T10:24:56.744-0800    [........................] osm.sanjose  18.0MB/446MB (4.0%)
2018-02-05T10:24:59.744-0800    [#.......................] osm.sanjose  27.6MB/446MB (6.2%)
2018-02-05T10:25:02.746-0800    [#.......................] osm.sanjose  36.6MB/446MB (8.2%)
2018-02-05T10:25:05.744-0800    [##......................] osm.sanjose  45.5MB/446MB (10.2%)
2018-02-05T10:25:08.744-0800    [##......................] osm.sanjose  55.4MB/446MB (12.4%)
2018-02-05T10:25:11.744-0800    [###.....................] osm.sanjose  64.6MB/446MB (14.5%)
2018-02-05T10:25:14.745-0800    [####....................] osm.sanjose  75.3MB/446MB (16.9%)
2018-02-05T10:25:17.747-0800    [####....................] osm.sanjose  86.1MB/446MB (19.3%)
2018-02-05T10:25:20.744-0800    [#####...................] osm.sanjose  97.6MB/446MB (21.9%)
2018-02-05T10:25:23.744-0800    [#####...................] osm.sanjose  108MB/446MB (24.1%)
2018-02-05T10:25:26.750-0800    [######..................] osm.sanjose  118MB/446MB (26.3%)
2018-02-05T10:25:29.744-0800    [######..................] osm.sanjose  129MB/446MB (28.8%)
2018-02-05T10:25:32.746-0800    [#######.................] osm.sanjose  139MB/446MB (31.1%)
2018-02-05T10:25:35.746-0800    [#######.................] osm.sanjose  148MB/446MB (33.3%)
2018-02-05T10:25:38.745-0800    [########................] osm.sanjose  159MB/446MB (35.5%)
2018-02-05T10:25:41.744-0800    [########................] osm.sanjose  167MB/446MB (37.4%)
2018-02-05T10:25:44.745-0800    [########................] osm.sanjose  175MB/446MB (39.2%)
2018-02-05T10:25:47.744-0800    [#########...............] osm.sanjose  184MB/446MB (41.3%)
2018-02-05T10:25:50.744-0800    [#########...............] osm.sanjose  193MB/446MB (43.3%)
2018-02-05T10:25:53.744-0800    [#########...............] osm.sanjose  202MB/446MB (45.3%)
2018-02-05T10:25:56.748-0800    [##########..............] osm.sanjose  213MB/446MB (47.6%)
2018-02-05T10:25:59.749-0800    [##########..............] osm.sanjose  223MB/446MB (49.9%)
2018-02-05T10:26:02.745-0800    [###########.............] osm.sanjose  232MB/446MB (52.0%)
2018-02-05T10:26:05.744-0800    [###########.............] osm.sanjose  241MB/446MB (53.9%)
2018-02-05T10:26:08.744-0800    [############............] osm.sanjose  252MB/446MB (56.5%)
2018-02-05T10:26:11.748-0800    [############............] osm.sanjose  261MB/446MB (58.4%)
2018-02-05T10:26:14.745-0800    [#############...........] osm.sanjose  271MB/446MB (60.7%)
2018-02-05T10:26:17.744-0800    [#############...........] osm.sanjose  282MB/446MB (63.1%)
2018-02-05T10:26:20.747-0800    [##############..........] osm.sanjose  295MB/446MB (66.0%)
2018-02-05T10:26:23.744-0800    [##############..........] osm.sanjose  310MB/446MB (69.4%)
2018-02-05T10:26:26.744-0800    [###############.........] osm.sanjose  323MB/446MB (72.4%)
2018-02-05T10:26:29.744-0800    [###############.........] osm.sanjose  334MB/446MB (74.9%)
2018-02-05T10:26:32.745-0800    [################........] osm.sanjose  347MB/446MB (77.6%)
2018-02-05T10:26:35.749-0800    [################........] osm.sanjose  357MB/446MB (80.0%)
2018-02-05T10:26:38.744-0800    [#################.......] osm.sanjose  375MB/446MB (84.1%)
2018-02-05T10:26:41.744-0800    [#################.......] osm.sanjose  395MB/446MB (88.5%)
2018-02-05T10:26:44.746-0800    [##################......] osm.sanjose  414MB/446MB (92.7%)
2018-02-05T10:26:47.744-0800    [##################......] osm.sanjose  434MB/446MB (97.1%)
2018-02-05T10:26:49.770-0800    [########################] osm.sanjose  446MB/446MB (100.0%)
2018-02-05T10:26:49.771-0800    imported 2120124 documents
```

The number of nodes and ways in MongoDB matches with the original OSM dataset.

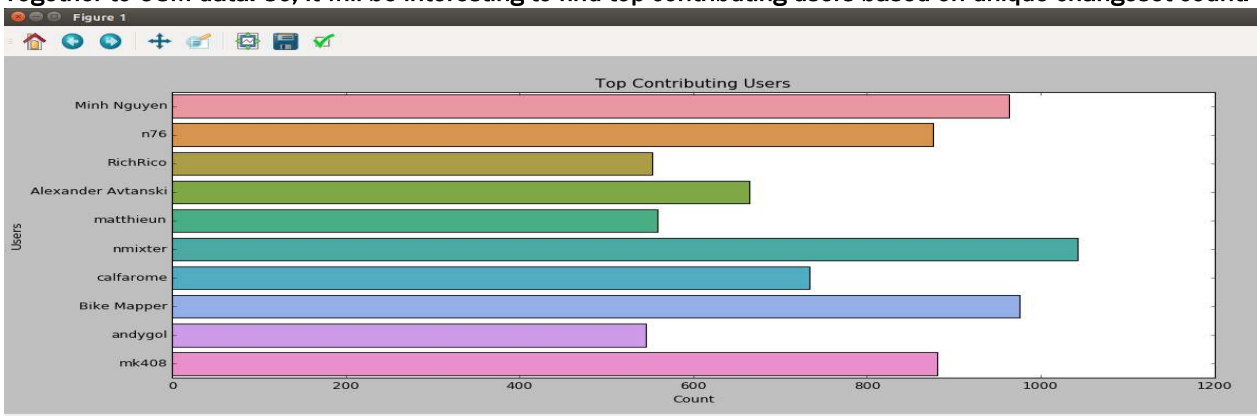
```
ashish@ubuntu:~/DataAnalyst/Project_Wrangle_OSM$ python data_explore.py
Number of documents =  2120124
Number of Nodes =  1872967
Number of Ways =  247157
Number of unique users =  1476
```

Step 4 – Overview of the data source code: data_explore.py
Data Insights using MongoDB.

Top contributing users. Total unique users is 1476
andygol and nmixter have more than 280K entries each

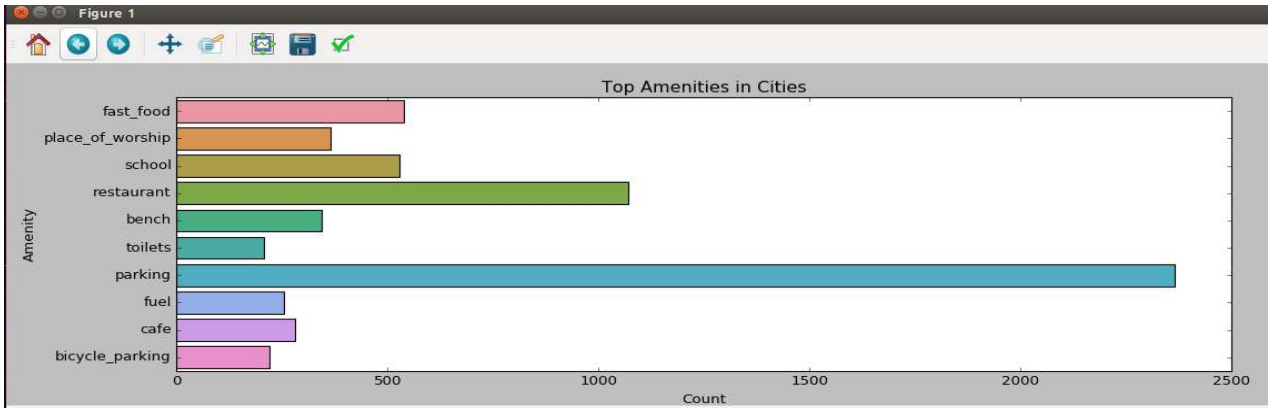


However, lot of entries are under same changeset.
A changeset is a collection of related changes (new objects, object modifications, or object deletions) applied Together to OSM data. So, it will be interesting to find top contributing users based on unique changeset count:-



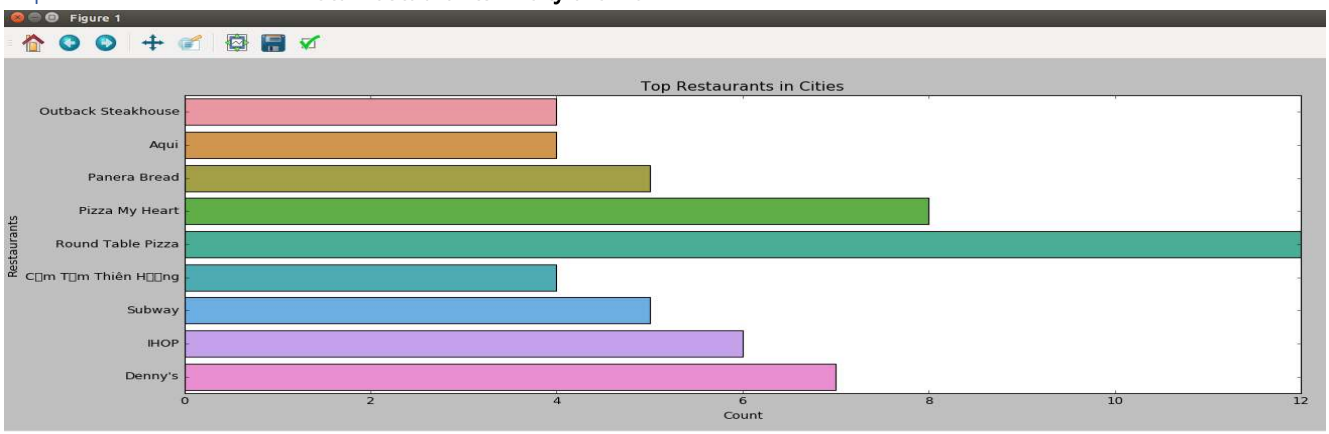
Interestingly, nmixter tops the list with others too making substantial contribution. Andygol has the least contribution.

Top Amenities in City



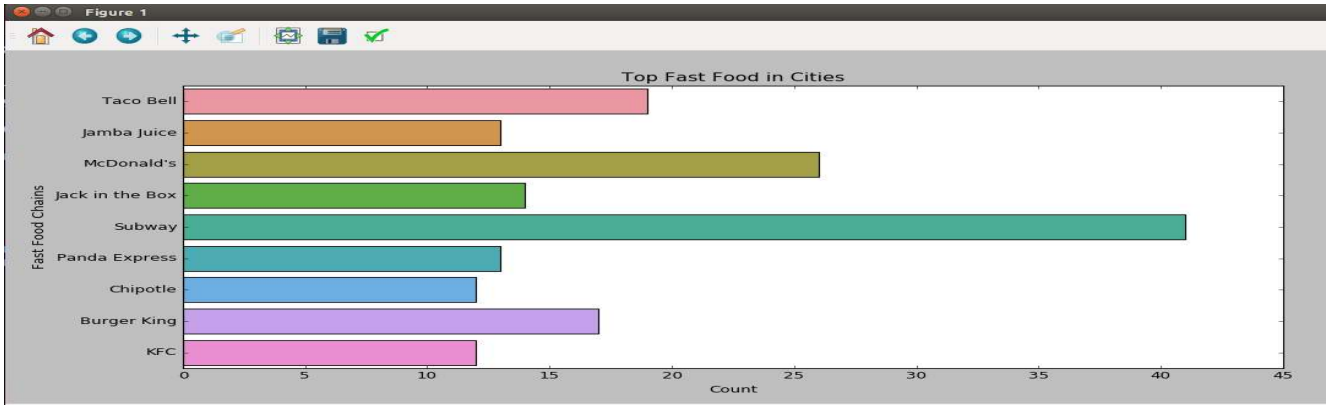
As expected, parking leads the chart, followed by restaurants, fast foods and cafes. Restaurants and fast food as they together are 2nd top most amenity after parking. However, there are twice as much restaurants when compared to fast foods

Top Restaurants in Cities **Total restaurants in city are 1071**



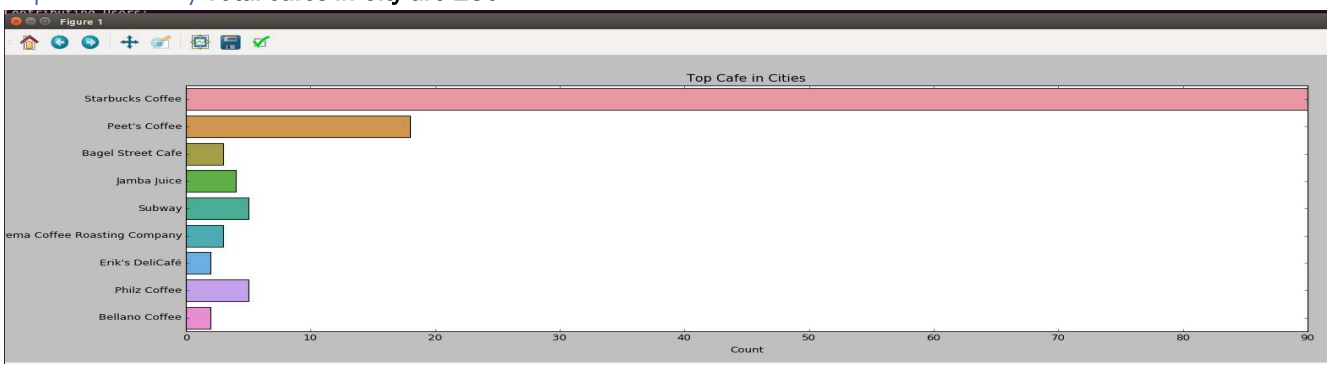
Pizza restaurants rules the chart with Round Table Pizza with 12 locations followed by Pizza My Heart, 8 locations

Top Fast Food Chains in city **Total Fast foods in City are 539**



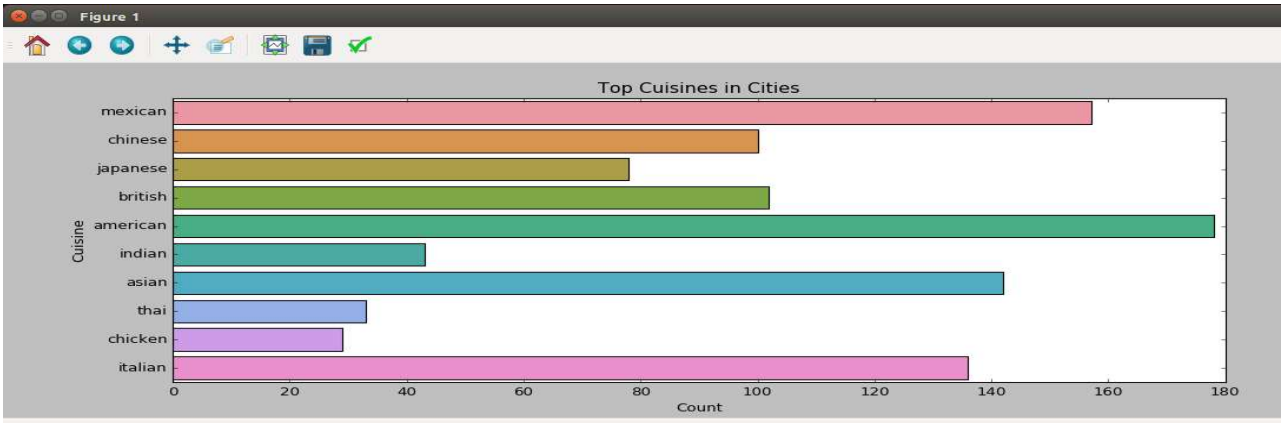
Subway restaurants seems quite famous with 41 locations. Subway is also featured under restaurants.

1. Top Cafes in City **Total cafes in City are 280**



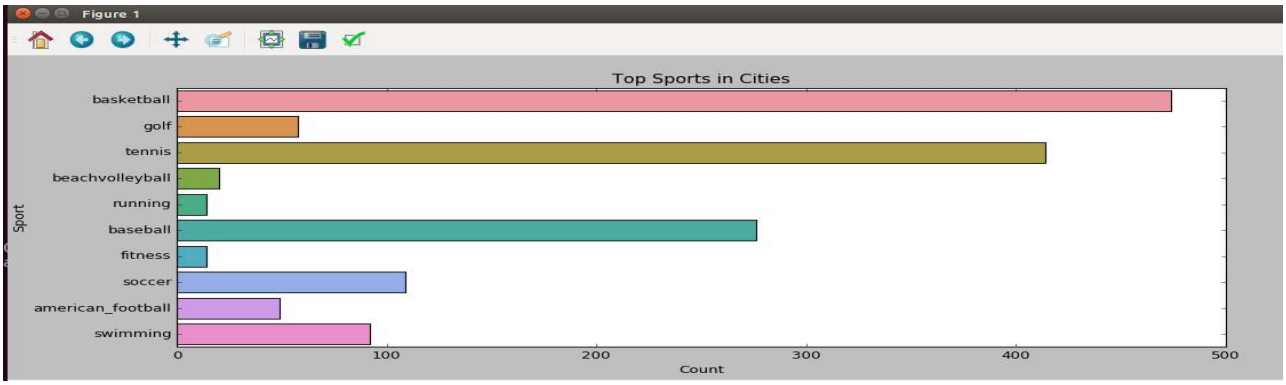
Starbucks leads the chart by quite a big margin. Infact, there are twice as many Starbucks café as all other café's together. Every 1 in 3 cafes is Starbucks. (90/280 locations)

Top Cuisines in city



American cuisine is expected to have more presence. It seems city likes Italian and Mexican also a lot.
Now it makes sense when Round Table Pizza and Subway lead the chart in restaurants and fast foods categories

Top Sports in City



Basketball and tennis are popular amongst citizen.

Step 5 – Conclusions

After going through the process, I found openstreet data can provide quite good insights for the business if inconsistencies in data are handled systematically. There should be system audit process before entries are persisted into the open street map.

Openstreetmap is open source project and we should have validation system to ensure entries from users. e.g. User andygol and nmixter has approximately 280,000 entries in openstereet map. Inconsistently and improper entries from these two users could have led to multiple fold errors.

I would recommend validation of addresses while entry is made e.g. using a global address information system to keep it consistent across various user entries like programmatically get that data from the Google Maps API. **We should use standardized, system driven abbreviation for street names, cuisines, key-value pairs.**

References

- http://wiki.openstreetmap.org/wiki/OSM_XML
- https://docs.mongodb.org/manual
- https://s3.amazonaws.com/metro-extracts.mapzen.com/san-jose_california.osm.bz2