OpenStreetMap Project

Data Wrangling with MongoDB

Kening Ren

Map Area: San Jose, California, United States

Contents:

1, Why to choose San Jose, GA?

It is in the heart of Silicon Valley. A lot of high-tech companies and other businesses which support innovation and entrepreneurship. It is alway amazing to have a chance to study this area.

2, Where to download the map data for San Jose, GA?

Download the preselected metro area from Map Zen. Go to www.mapzen.com (https://mapzen.com/data/metro-extracts). And search "San Jose, CA". And download its "OSM XML" file. Its size is 16.9 MB. After unzip it, it is 248MB. So it meets project requirements on data size. The file name is "san-jose_california.osm"

3, Problems Encountered in the Map

Since the downloaded San Jose map data is too large, it is very slow to work on it in a PC. So after downloading the San Jose map from MapZen, I run a Python code "generate_small_osm.py" to extract a small-sample-size map from the downloaded the San Jose map. This Python code is provided by Udancity. It picks one out of every ten elements in the "OSM XML" file, therefore it reduces the number of elements by 90%. With this small sample map data, I can study and identify the data wrangling I need to perform. The file name of the small sample data file is "san-jose_california_sample.osm"

3.1, The element "phone" has different formats

For example, we have the following formats of phone number:
408-225-010
+1 408 236 7395
+1-408-730-5044
+1.408.245.3686
(1)(408) 766-7000
+14089829040
+1 408-778-6409
4087395533

When generating the json map file, all different formats of phone number are converted to, for example, 4087395533-like number.

3.2, The element "postcode" has different formats

For example, we have the following formats of the postcode:
95014-2218
95050

When generating the json map file, all different formats of postcode are converted to, for example, 95050-like number.

3.3, Tag "Amenity" has different names

For example, we have the following names of Tag "Amenity":
amenity
amenity:history

When generating the json map file, all these names are converted to "amenity".

4, Port JSON file into MongoDB

The Python code "data.py" extracts the information from San Jose OSM XML file "san-jose_california.osm" and converts it to JSON file "san-jose_california.osm.json".

Then it is added into the MongoDB. A db "project" is created and a collection "sj" is created to take in the json data. The following are the commands:

To run the MongoDB server in Windows 7:

c:\Program Files\MongoDB\Server\3.2\bin\mongod.exe

To run the MongoDB client in Windows 7:

c:\Program Files\MongoDB\Server\3.2\bin\mongo.exe

To add the new db "project" in MongoDB, in MongoDB client console, run:

use project

Then add the JSON File "san-jose_california.osm.json" into "project" in MongoDB. It uses Collection "sj":

```
c:\Program Files\MongoDB\Server\3.2\bin\mongoimport.exe -d project -c sj --file
C:\Users\kenren\Documents\doc\My_Learning\Data_Scientis\Udacity_DataAnalyst\DataWrangling_Mong
jose_california.osm\san-jose_california.osm.json
```

5, Study the data

To study the JSON data, I choose two ways:

- CLI command in MongoDB client
  In the report, it is abreviated as "CLI"
- Python code to access MongoDB
  In the report, it is abreviated as "CODE"

  Here is the framework of the Python code:

In [1]:
```
from pymongo import MongoClient
import pprint

def get_db(db_name):
    client = MongoClient('localhost:27017')
    db = client[db_name]
    return db

def make_pipeline():
    # complete the aggregation pipeline
    pipeline = [ ]
    # Access command is added here in pipeline. Here are two examples.
    #pipeline.append({'$group':{'_id': '$created.user'}})
    #pipeline.append({'$group':{'_id':1, 'count':{'$sum':1}}})
    return pipeline

def aggregate(db, pipeline):
    return [doc for doc in db.sj.aggregate(pipeline)]

if __name__ == '__main__':
    db = get_db('project')
    pipeline = make_pipeline()
    result = aggregate(db, pipeline)
    #pprint.pprint(result)
```

5.1, File sizes

The original data is:
san-jose_california.osm ..... 254,756 KB
san-jose_california.osm.json ..... 402,215 KB

The small sample data is:
san-jose_california_sample.osm ..... 25,774 KB
san-jose_california_sample.osm.json ... 39,167 KB

5.2, Data overview

5.2.1, Total number of elements

> db.sj.find().count()
1344358

5.2.2, Number of nodes

(CLI)> db.sj.find({"element_type":"node"}).count()
1192305

5.2.3, Number of ways

(CLI)> db.sj.find({"element_type":"node"}).count()
150928

5.2.4, Number of relations

(CLI)> db.sj.find({"type":"node"}).count()
1088

5.2.5, Number of unique users

(CLI)> db.sj.aggregate([{'$group':{'_id':'$created.user'}},{'$group':{'_id':'Total numer of unique users','count':{'$sum':1}}}])
{ "_id" : "Total numer of unique users", "count" : 1053 }

5.2.6, Top 10 users

Using Python code "top10_users.py". Below is the "pipeline". It is the input to db.sj.aggregate().

(CODE)
pipeline.append({'$group':{'_id': '$created.user', 'count' :{'$sum':1}}})
pipeline.append({'$sort':{'count':-1}})
pipeline.append({'$limit': 10})

result:
[{u'_id': u'nmixter', u'count': 278452},
{u'_id': u'mk408', u'count': 153980},
{u'_id': u'Bike Mapper', u'count': 80949},
{u'_id': u'n76_cupertino_import', u'count': 73967},
{u'_id': u'n76', u'count': 57834},
{u'_id': u'samely', u'count': 53967},
{u'_id': u'RichRico', u'count': 47920},
{u'_id': u'erjiang_imports', u'count': 39358},
{u'_id': u'dannykath', u'count': 37104},
{u'_id': u'matthieun', u'count': 35727}]

Useing Python code "number_top10_users.py" to calcualte the total number of records the top 10 users created. Below is the "pipeline". It is the input to db.sj.aggregate().

(CODE)
```
pipeline.append({'$group':{'_id': '$created.user', 'count' :{'$sum':1}}})
pipeline.append({'$sort':{'count':-1}})
pipeline.append({'$limit': 10})
pipeline.append({'$group':{'_id':'Total number of top 10 users', 'count':{'$sum':'$count'}}})
```

Result: [{u'_id': u'Total number of top 10 users', u'count': 859258}]

In the result above, we can see the user "nmixter" created 278452 records. It is about 278452/1344358 = 20.7%.
The top 10 users created about **63.9%** of total records.

5.2.7, Find the locations added by some of the Top 10 users in San Jose map

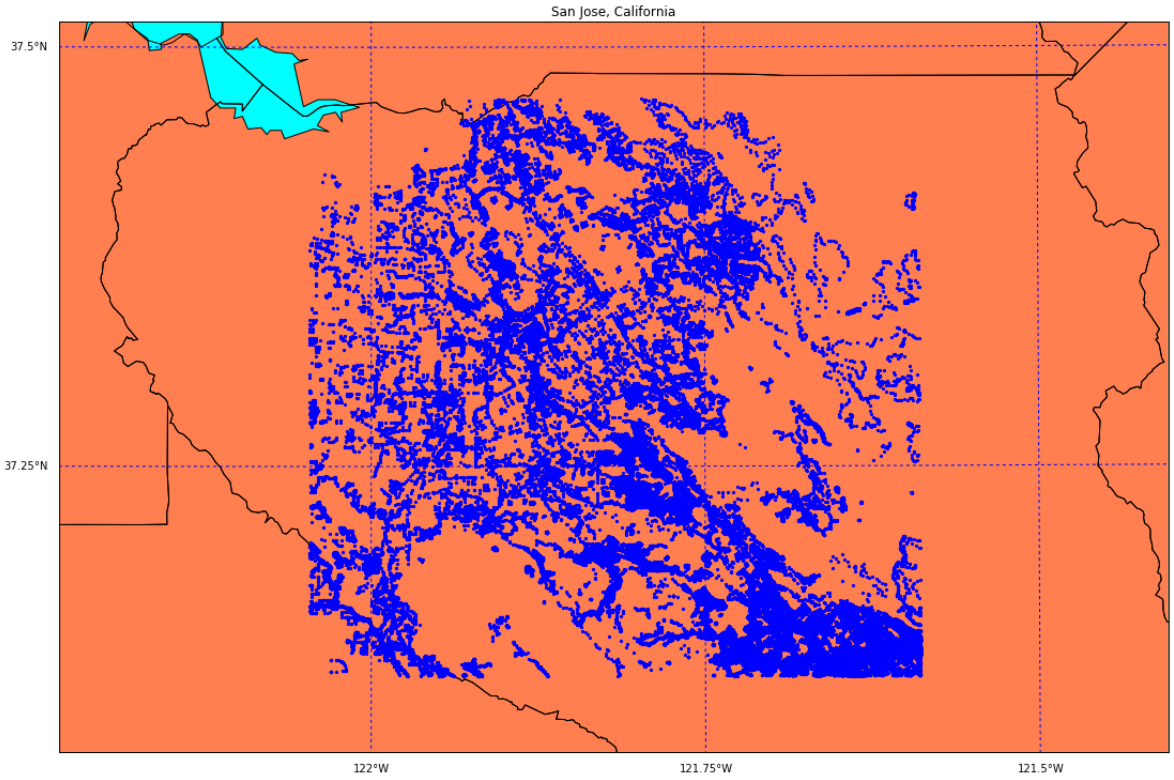
Python code "map_top10_users.py" does the following:

- find the top 10 users from San Jose map data in MongoDB.
- draw the map of San Jose using Stereographic Projection.
- for some of the top 10 users, extract the records with the location in latitude and longitude this user added into the San Jose map data.
- draw the records as dots in the map.

The records added by the Top user "nmixter" are shown in the following map as dots

In [2]:
```
from IPython.display import Image
Image(filename='map_user_nmixter.png')
```

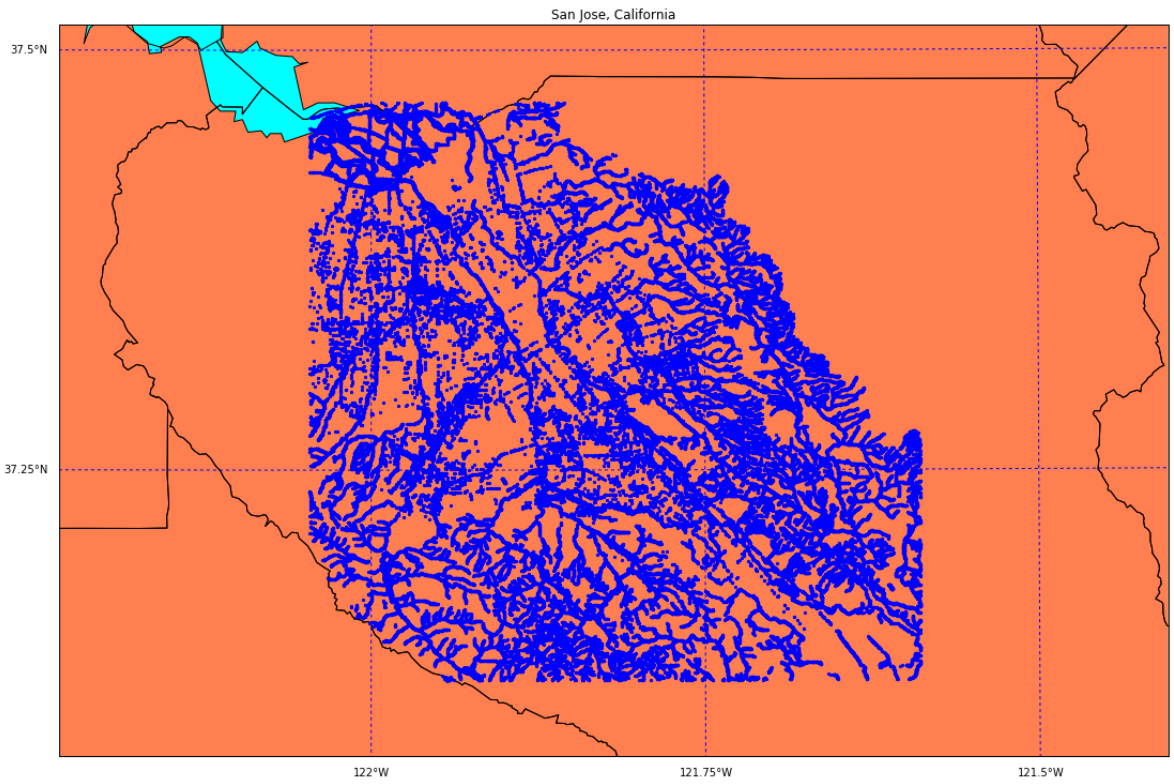
Out[2]:



The records added by the second top user "mk408" are shown in the following map as dots

In [3]:
```
from IPython.display import Image
Image(filename='map_user_mk408.png')
```

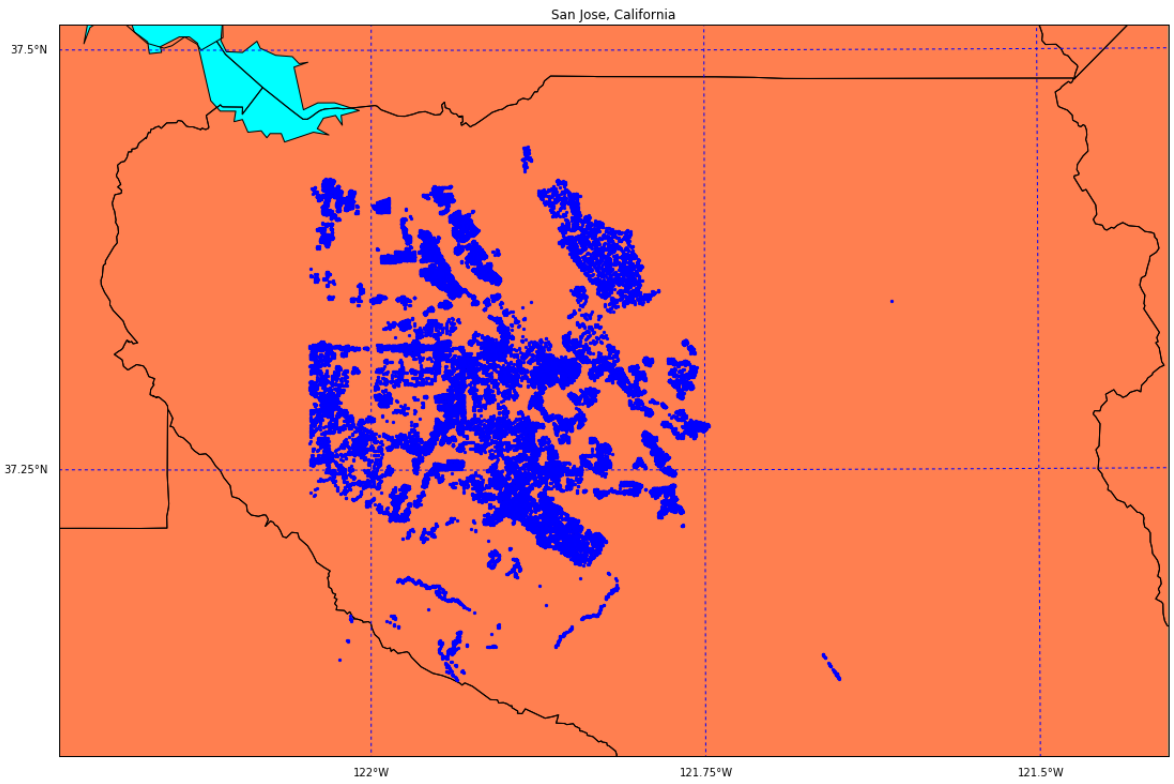
Out[3]:



The records added by the second top user "mk408" are shown in the following map as dots

In [6]:
```
from IPython.display import Image
Image(filename='map_user_Bike_Mapper.png')
```

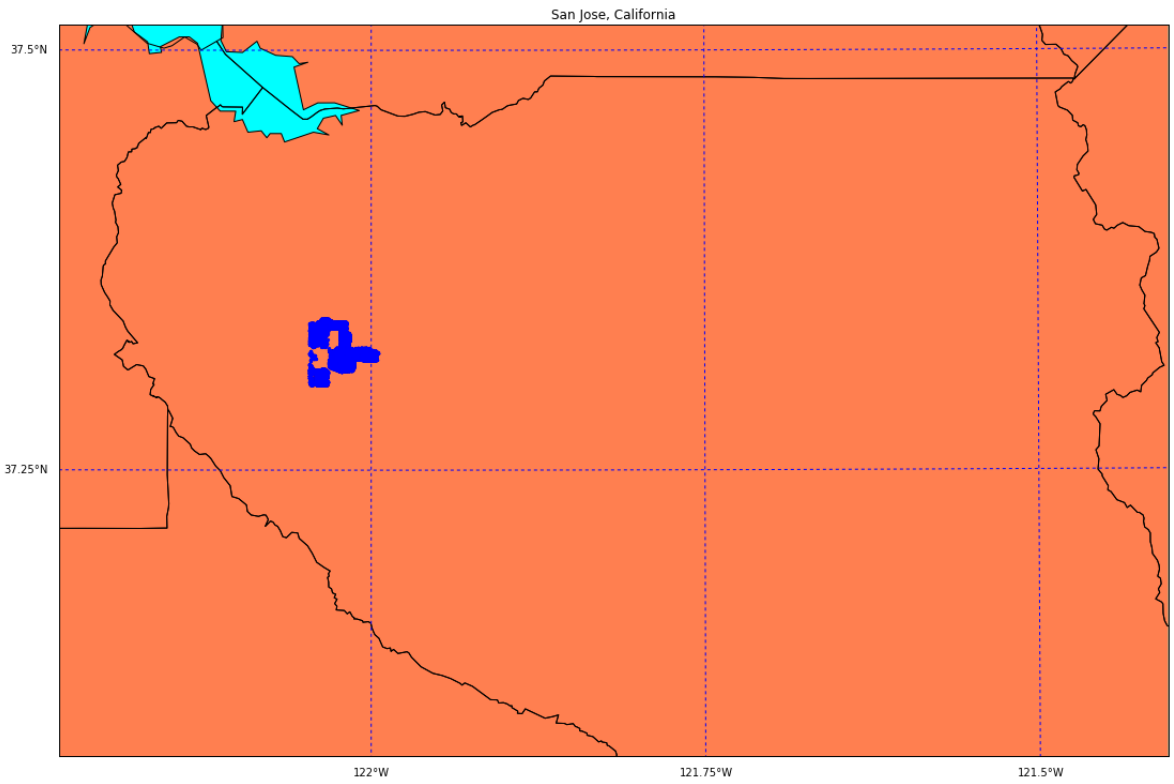
Out[6]:



The records added by the second top user "n76_cupertino_import" are shown in the following map as dots

In [5]:
```
from IPython.display import Image
Image(filename='map_user_n76_cupertino_import.png')
```

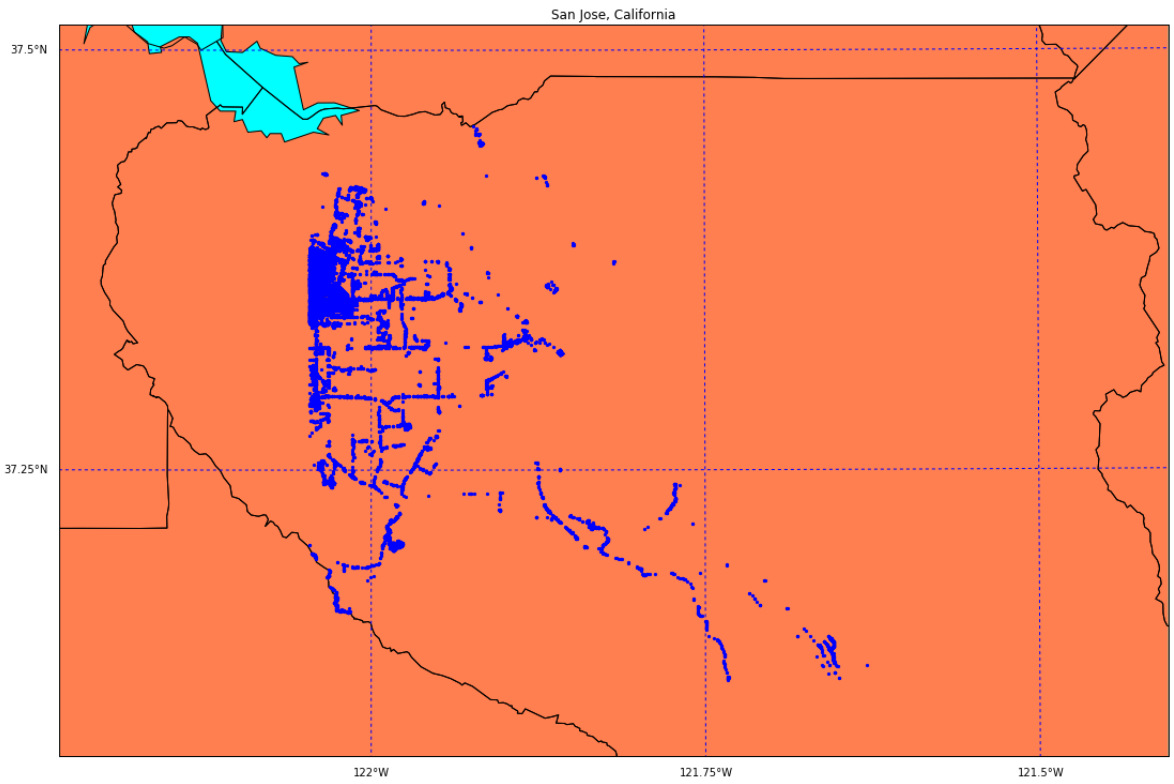
Out[5]:



The records added by the second top user "n76" are shown in the following map as dots

In [7]:
```
from IPython.display import Image
Image(filename='map_user_n76.png')
```

Out[7]:



Summary of the work style of some of the top 10 users:

- The top user " ***nmixter*** " and the third top user " ***Bike mapper*** " seemed to work in the same way. They picked up an area and fill in the records, then they moved on to another area. And they covered almost all areas. The fourth top user " ***n76_cupertino_import*** " is doing the same. But he only focused on one small area, which he was interested.
- The second top user " ***mk408*** " seemed to work along the road. He added records for each building along a road. After he finished one road, he would begin to work on another road. Until he finished all roads. The fifth top user " ***n76*** " was doing in the same way. But he only did some roads.
- It appears on the map that the fourth top user " ***n76_cupertino_import*** " covered smaller area than that of the fifth top user " ***n76*** ". But " ***n76_cupertino_import*** " added 67312 dots. It is more than 48988 dots added by " ***n76*** ".

5.3, More data exploration

5.3.1, Study "amenity"

We lists the top 40 "amenity", by running the python code "amenity.py". Below is the "pipeline". It is the input to db.sj.aggregate().
(CODE)

```
pipeline.append({'$match':{'amenity':{'$exists':1}}})
pipeline.append({'$group':{'_id': '$amenity', 'count' :{'$sum':1}}})
pipeline.append({'$sort':{'count':-1}})
pipeline.append({'$limit': 40})
```

The results are:

[{u'_id': u'parking', u'count': 1655},
{u'_id': u'restaurant', u'count': 850},
{u'_id': u'school', u'count': 550},
{u'_id': u'fast_food', u'count': 426},
{u'_id': u'place_of_worship', u'count': 359},
{u'_id': u'fuel', u'count': 222},
{u'_id': u'cafe', u'count': 217},
{u'_id': u'bench', u'count': 174},
{u'_id': u'bank', u'count': 173},
{u'_id': u'bicycle_parking', u'count': 168},
{u'_id': u'toilets', u'count': 166},
{u'_id': u'post_box', u'count': 74},
{u'_id': u'dentist', u'count': 74},
{u'_id': u'drinking_water', u'count': 63},
{u'_id': u'bar', u'count': 60},
{u'_id': u'pharmacy', u'count': 59},
{u'_id': u'fire_station', u'count': 58},
{u'_id': u'swimming_pool', u'count': 51},
{u'_id': u'bbq', u'count': 50},
{u'_id': u'post_office', u'count': 49},
{u'_id': u'doctors', u'count': 48},
{u'_id': u'emergency_phone', u'count': 45},
{u'_id': u'library', u'count': 44},
{u'_id': u'parking_space', u'count': 43},
{u'_id': u'atm', u'count': 43},
{u'_id': u'public_building', u'count': 38},
{u'_id': u'parking_entrance', u'count': 36},
{u'_id': u'fountain', u'count': 32},
{u'_id': u'vending_machine', u'count': 31},
{u'_id': u'car_wash', u'count': 29},
{u'_id': u'hospital', u'count': 27},
{u'_id': u'telephone', u'count': 25},
{u'_id': u'pub', u'count': 24},
{u'_id': u'bicycle_rental', u'count': 18},
{u'_id': u'cinema', u'count': 17},
{u'_id': u'college', u'count': 17},
{u'_id': u'veterinary', u'count': 16},
{u'_id': u'police', u'count': 16},
{u'_id': u'social_facility', u'count': 16},
{u'_id': u'theatre', u'count': 15}]

Summary :

- In general, the records of "amenty" for different facilities are in good shape. Users did not pick up any name they want to use. They did follow the rule of naming, if there is one. Some minor fix is needed such as "parking" and "parking_space" should both be named as "parking".
- There are about 1700 parking lots.
- For dining business, there are restaurant, fast food, cafe, bar, pub. Totally there are about 1570 dining places.
- For education, there are 550 schools, 17 colleges and 13 universities.
- For medicine, there are 74 dentists, 59 pharmacies, 48 clinics, 27 hospitals, 16 veterinary clinics.

5.3.2, Study the cuisine of the dining business

We lists the top 25 "cuisine", by running the python code "cuisine.py". Below is the "pipeline". It is the input to db.sj.aggregate().

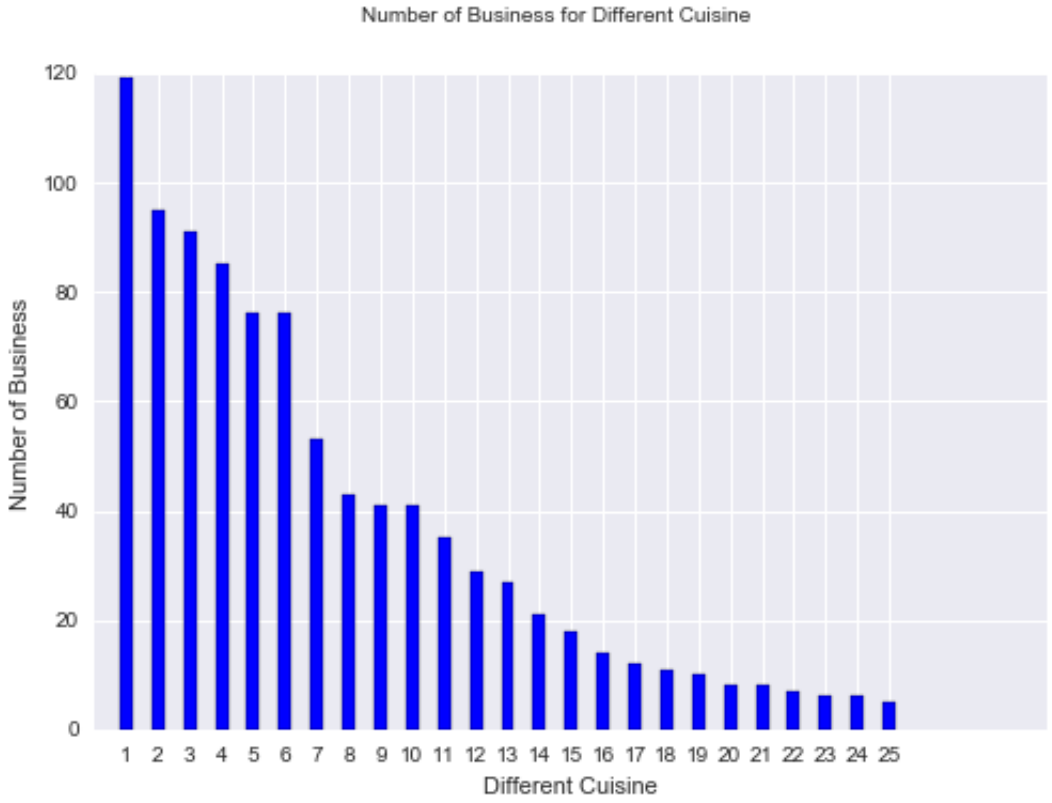
(CODE) pipeline.append({'$match':{'amenity':{'$exists':1}}})
pipeline.append({'$group':{'_id': '$amenity', 'count' :{'$sum':1}}})
pipeline.append({'$sort':{'count':-1}})
pipeline.append({'$limit': 40})

The results are:
[{u'_id': u'mexican', u'count': 119},
{u'_id': u'burger', u'count': 95},
{u'_id': u'sandwich', u'count': 91},
{u'_id': u'vietnamese', u'count': 85},
{u'_id': u'chinese', u'count': 76},
{u'_id': u'pizza', u'count': 76},
{u'_id': u'coffee_shop', u'count': 53},
{u'_id': u'japanese', u'count': 43},
{u'_id': u'indian', u'count': 41},
{u'_id': u'american', u'count': 41},
{u'_id': u'italian', u'count': 35},
{u'_id': u'ice_cream', u'count': 29},
{u'_id': u'thai', u'count': 27},
{u'_id': u'sushi', u'count': 21},
{u'_id': u'chicken', u'count': 18},
{u'_id': u'mediterranean', u'count': 14},
{u'_id': u'regional', u'count': 12},
{u'_id': u'steak_house', u'count': 11},
{u'_id': u'greek', u'count': 10},
{u'_id': u'korean', u'count': 8},
{u'_id': u'asian', u'count': 8},
{u'_id': u'barbecue', u'count': 7},
{u'_id': u'bagel', u'count': 6},
{u'_id': u'seafood', u'count': 6},
{u'_id': u'hawaiian', u'count': 5}]

Then I draw a bar plot based on the result above.

In [3]:
```
from IPython.display import Image
Image(filename='cuisine.png')
```

Out[3]:



Here is how cuisine is numbered:
1 = mexican, 2 = burger, 3 = sandwich, 4 = vietnamese, 5 = chinese, 6 = pizza, 7 = coffee_shop, 8 = japanese, 9 = indian
10 = american, 11 = italian, 12 = ice_cream, 13 = thai, 14 = sushi, 15 = chicken, 16 = mediterranean, 17 = regional
18 = steak_house, 19 = greek, 20 = korean, 21 = asian, 22 = barbecue, 23 = bagel, 24 = seafood, 25 = hawaiian

Summary :

- For me, burger, sandwich, pizza, coffee shop, american, chicken, steak house, barbecue, seafood and bagel all belong to American food. So there are about 440 American style dining places.
- Apparently, Mexican food is the quite popular. There are 119 Mexican dining places. Not a surprise in California.
- Then it is Vietnamese food. There are 85 diniing places.
- After that, there is Chinese (76), Japanese (43 + Sushi 21, total: 64), Indian (41), and Thai (27). So Asian food is the most popular cuisine after American.
- It is surprising that European food still holds a place: Italian (35), Mediterranean (14), and Greek (10).

6, Conclusion

6.1, What to improve? and How?

- San Jose's map data is quite complete. Top 10 users added 63.9% of the records. I hope more users could jump in and contribute more. Maybe OpenStreetMap should find some ways to encourage more users to do this. Such as publishing the name of Top 10 users, giving out prizes to users who contribute a lot, running a campaign to call people to add data for their neighbourhood.
- To me, the data is quite clean. I notices some discrepancies in the formats of the data. I hope that OpenStreetMap publishes some clear and succinct rules about naming and formatting of data. OpenStreetMap should run some software to filter through the data and fix all the errors and bad data.
- Some other data should be added, such as the static traffic data. User can add the traffic status at different roads, which shows, on average, at what time which road at which direction will have the traffic jam and how bad is the traffic jam.

6.2, New Opportunities to Use the Map Data

The government agency can use the map data to better plan the city development. For example, find whether the public parking lots are in the right places, and find where to set up the charging station for the electric cars.
Business can use the map data to find the new opportunities. For example, the location to open a new business.