P3: OpenStreetMap Data Case Study

Map Area

1. Schaumburg, IL, United States

Source of the map and OSM data file:

http://www.openstreetmap.org/#map=12/42.0152/-88.0300 (http://www.openstreetmap.org/#map=12/42.0152/-88.0300)

https://mapzen.com/data/metro-extracts/your-extracts/c24fa71992dc (https://mapzen.com/data/metro-extracts/your-extracts/c24fa71992dc)

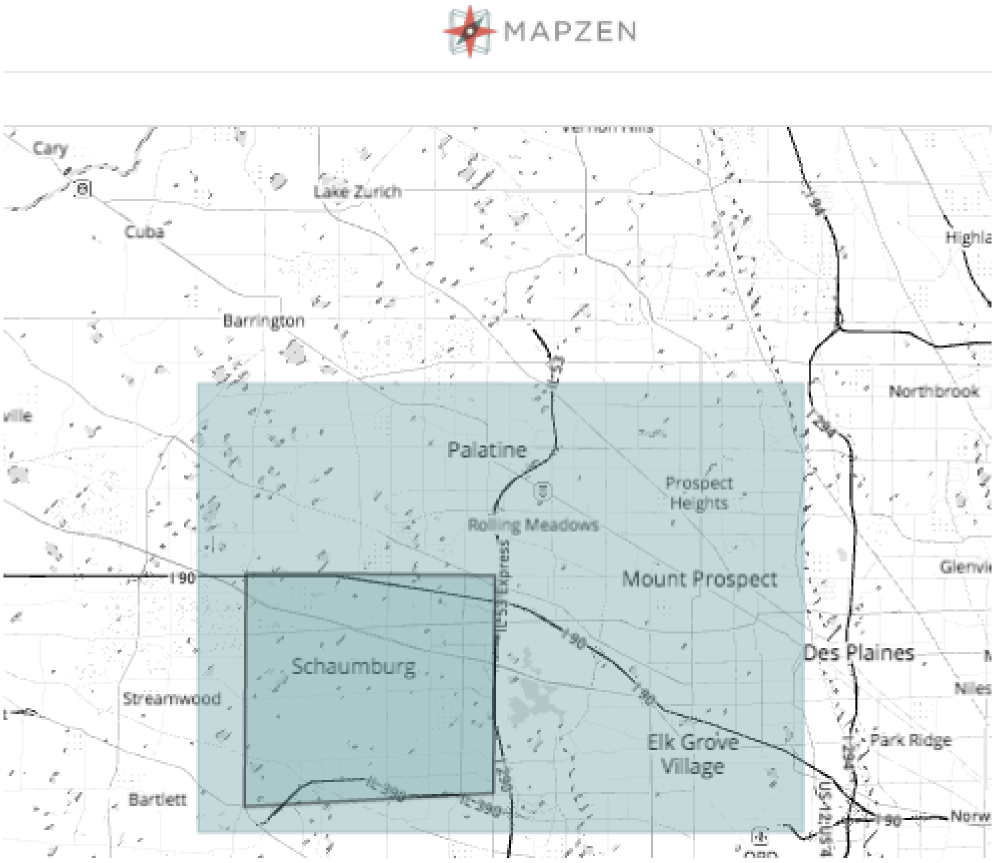
This is a map of Schaumburg and neighboring towns representing my data. I’m interested to see what database querying reveals.

(Image: ScreenShot_03.PNG)

In [1]:

```
from IPython.display import Image
Image(filename='Images/ScreenShot_03.PNG')
```

Out[1]:



Overview of the Data

The choice of the databases for this project: SQL.

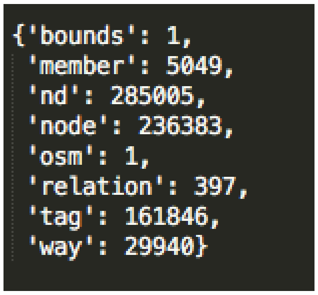
The extract was created on 2016 September 02, at 04:53 PM. OSM XML compressed file size is 3.7MB, uncompressed size 53.5MB.

To get the feeling on how much of which data I can expect to have in the map, I did the iterative parsing to process the map file and find out not only what tags are there, but also how many.

(Python file: Iterative_Parsing.py, Image: ScreenShot_09.PNG)

```
In [2]: Image(filename='Images/ScreenShot_09.PNG')
```

Out[2]:



```
{'bounds': 1,
 'member': 5049,
 'nd': 285005,
 'node': 236383,
 'osm': 1,
 'relation': 397,
 'tag': 161846,
 'way': 29940}
```

To see if there are any potential problems I explored the data a bit more. Before processing the data and adding it into the database, I checked the "k" value for each "tag".

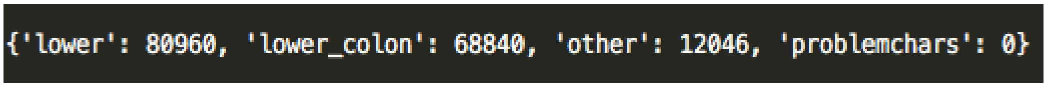
The count of each of four tag categories in a dictionary:

1. "lower", for tags that contain only lowercase letters and are valid,
2. "lower_colon", for otherwise valid tags with a colon in their names,
3. "problemchars", for tags with problematic characters, and
4. "other", for other tags that do not fall into the other three categories.

(Python file: Tag_Types.py, Image: ScreenShot_10.PNG)

```
In [3]: Image(filename='Images/ScreenShot_10.PNG')
```

Out[3]:



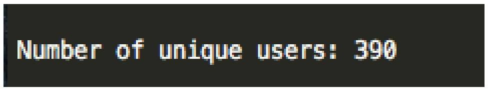
```
{'lower': 80960, 'lower_colon': 68840, 'other': 12046, 'problemchars': 0}
```

The number of unique users who have contributed to the map in this particular area is 390.

(Python file: Exploring_Users.py, Image: ScreenShot_11.PNG)

```
In [4]: Image(filename='Images/ScreenShot_11.PNG')
```

Out[4]:



```
Number of unique users: 390
```

Problems Encountered in the Map

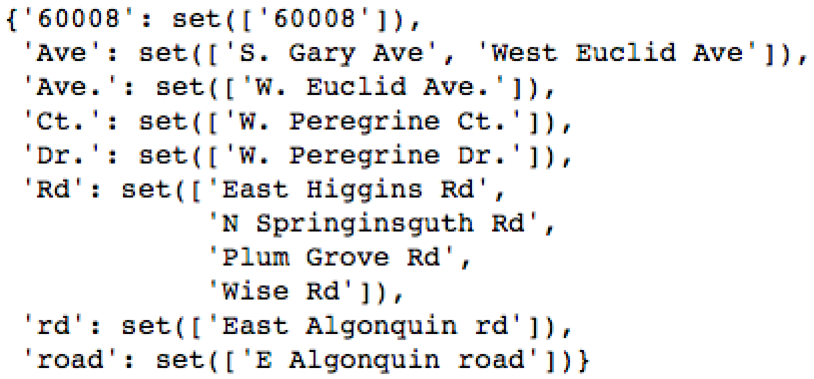
1. Abbreviated street names

I decided to explore the chosen Chicago west suburban area programmatically. After running audit_street_names.py and printing the results, one of the problems I have encountered was the inconsistency in naming the streets. See Image 01. Various abbreviations (with or without dots, upper/ lower case) of the names will be spelled out in full word with the first capital letter. Also, there are values which are assigned to a wrong map feature, e.g a zipcode as a street name. I am not planning to remove these values.

(Python file: audit_street_names.py, Image: ScreenShot_01.PNG)

```
In [5]: Image(filename='Images/ScreenShot_01.PNG')
```

Out[5]:



```
{'60008': set(['60008']),
 'Ave': set(['S. Gary Ave', 'West Euclid Ave']),
 'Ave.': set(['W. Euclid Ave.']),
 'Ct.': set(['W. Peregrine Ct.']),
 'Dr.': set(['W. Peregrine Dr.']),
 'Rd': set(['East Higgins Rd',
            'N Springinsguth Rd',
            'Plum Grove Rd',
            'Wise Rd']),
 'rd': set(['East Algonquin rd']),
 'road': set(['E Algonquin road'])}
```

2. Abbreviated street directions.

The other inconsistency is the street directions. Various abbreviations. All street directions will be spelled out in full word with the first capital letter, e.g. W -> West, E -> East. See image 01.

(Image: ScreenShot_01.PNG)

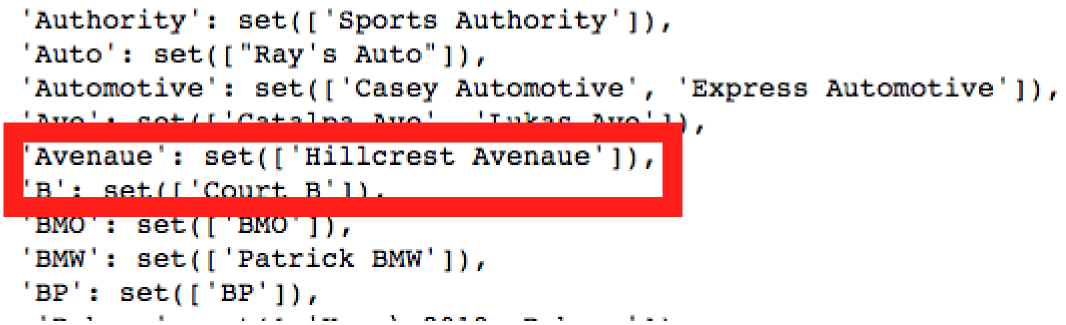
3. Spelling mistake

The spelling mistake was found after running the code with various attribute k values.

(Python file: audit_street_names.py, Image: ScreenShot_02.PNG)

```
In [6]: Image(filename='Images/ScreenShot_02.PNG')
```

Out[6]:



```
 'Authority': set(['Sports Authority']),
 'Auto': set(["Ray's Auto"]),
 'Automotive': set(['Casey Automotive', 'Express Automotive']),
 'Ave': set(['Catalpa Ave', 'Lukas Ave']),
 'Avenaue': set(['Hillcrest Avenaue']),
 'B': set(['Court B']),
 'BMO': set(['BMO']),
 'BMW': set(['Patrick BMW']),
 'BP': set(['BP']),
```

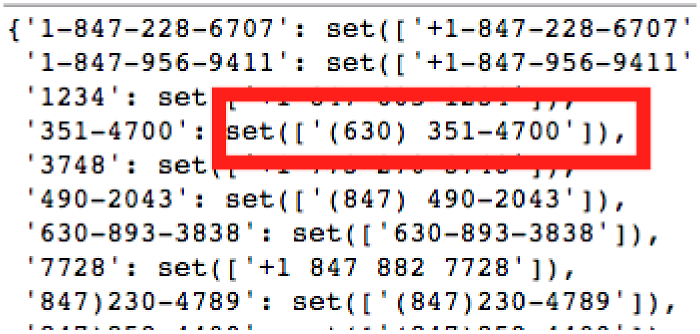
3. Inconsistent phone number formats

After running the code with attribute k = 'phone', I found a wide variety in how phone numbers are presented, with regard to spacing and punctuation. I decided to follow the NANP convention, and chose to use instantly recognizable and most common format of the number () -**.

(Python file: audit_street_names.py, Image: ScreenShot_07.PNG)

```
In [7]: Image(filename='Images/ScreenShot_07.PNG')
```

Out[7]:



```
{'1-847-228-6707': set(['+1-847-228-6707'
 '1-847-956-9411': set(['+1-847-956-9411'
 '1234': set
 '351-4700': set(['(630) 351-4700']),
 '3748': set
 '490-2043': set(['(847) 490-2043']),
 '630-893-3838': set(['630-893-3838']),
 '7728': set(['+1 847 882 7728']),
 '847)230-4789': set(['(847)230-4789']),
```

The Cleaning and Re-shaping Data

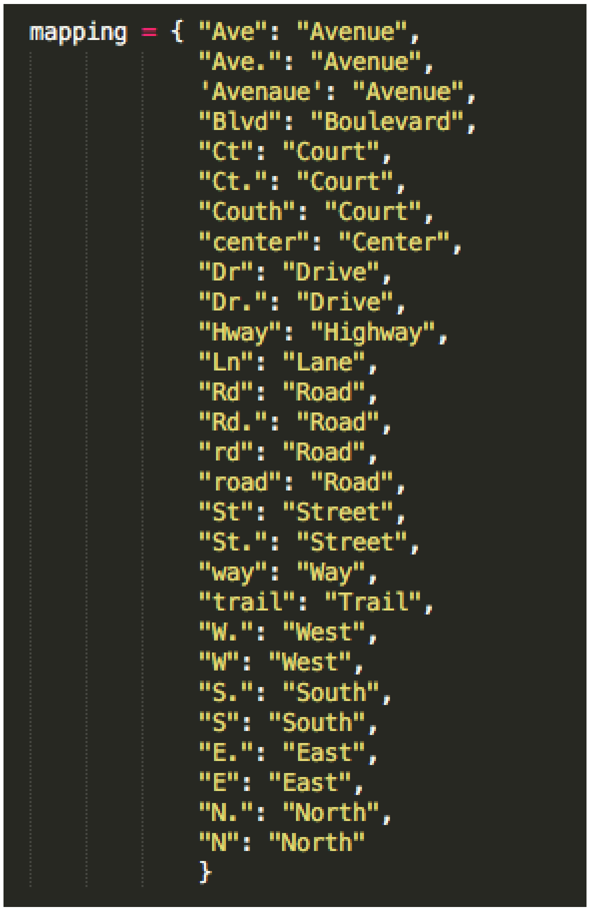
Based on my findings, I created a dictionary mapping the incorrect street names to correct values.

(Python file: data.py, Image: ScreenShot_12.PNG)

In [8]:

```
Image(filename='Images/ScreenShot_12.PNG')
```

Out[8]:



```
mapping = { "Ave": "Avenue",
            "Ave.": "Avenue",
            'Avenaue': "Avenue",
            "Blvd": "Boulevard",
            "Ct": "Court",
            "Ct.": "Court",
            "Couth": "Court",
            "center": "Center",
            "Dr": "Drive",
            "Dr.": "Drive",
            "Hway": "Highway",
            "Ln": "Lane",
            "Rd": "Road",
            "Rd.": "Road",
            "rd": "Road",
            "road": "Road",
            "St": "Street",
            "St.": "Street",
            "way": "Way",
            "trail": "Trail",
            "W.": "West",
            "W": "West",
            "S.": "South",
            "S": "South",
            "E.": "East",
            "E": "East",
            "N.": "North",
            "N": "North"
            }
```

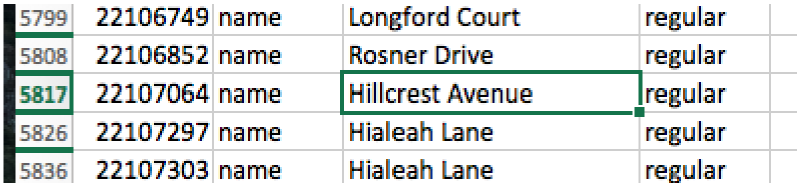
The street names were fixed, cleaned and updated using function update_name(name, mapping)

(Python file: data.py, ways_tags.csv, ScreenShot_15.PNG)

In [9]:

```
Image(filename='Images/ScreenShot_15.PNG')
```

Out[9]:



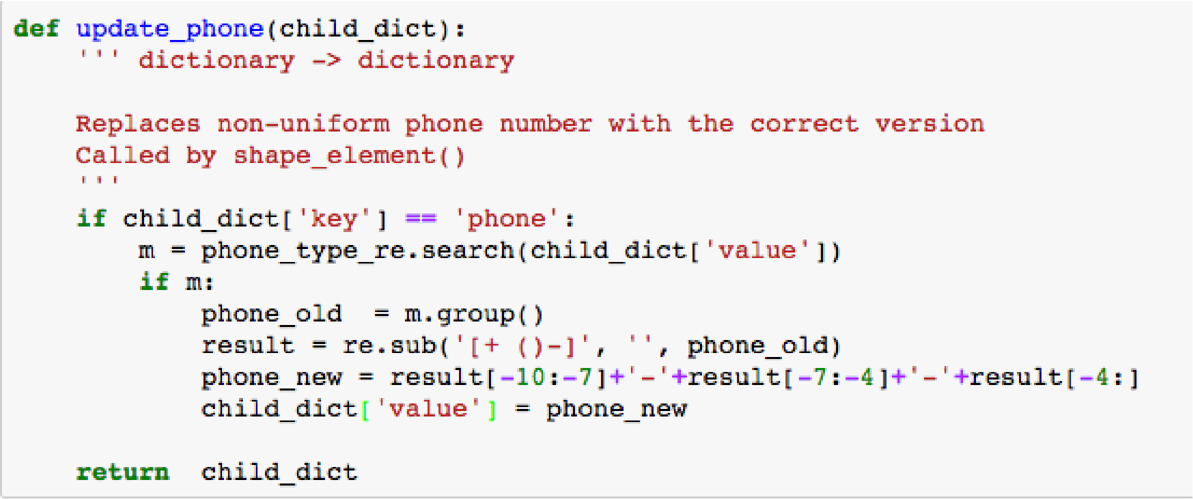
5799	22106749	name	Longford Court	regular
5808	22106852	name	Rosner Drive	regular
5817	22107064	name	Hillcrest Avenue	regular
5826	22107297	name	Hialeah Lane	regular
5836	22107303	name	Hialeah Lane	regular

With regular expressions pattern I checked for various formats of the phone number and made them uniform format with function update_phone(child_dict)

(Python file: data.py, nodes_tags.csv, Image: ScreenShot_13.PNG, ScreenShot_14.PNG)

In [10]:
```
Image(filename='Images/ScreenShot_14.PNG')
```

Out[10]:



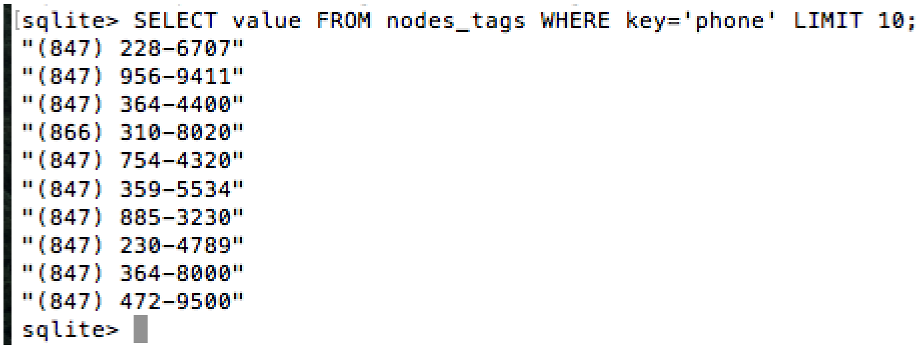
```
def update_phone(child_dict):
    ''' dictionary -> dictionary

    Replaces non-uniform phone number with the correct version
    Called by shape_element()
    '''
    if child_dict['key'] == 'phone':
        m = phone_type_re.search(child_dict['value'])
        if m:
            phone_old  = m.group()
            result = re.sub('[+ ()-]', '', phone_old)
            phone_new = result[-10:-7]+'-'+result[-7:-4]+'-'+result[-4:]
            child_dict['value'] = phone_new

    return  child_dict
```

In [11]:
```
Image(filename='Images/ScreenShot_13.PNG')
```

Out[11]:



```
[sqlite> SELECT value FROM nodes_tags WHERE key='phone' LIMIT 10;
"(847) 228-6707"
"(847) 956-9411"
"(847) 364-4400"
"(866) 310-8020"
"(847) 754-4320"
"(847) 359-5534"
"(847) 885-3230"
"(847) 230-4789"
"(847) 364-8000"
"(847) 472-9500"
sqlite> 
```

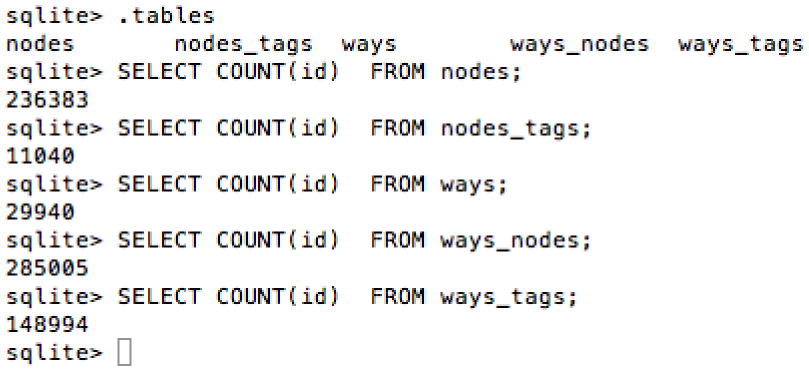
Overview of the Data in my Database

I created a database Schaumburg.db consisting of 5 tables.

(Image: ScreenShot_16.PNG)

In [12]:
```
Image(filename='Images/ScreenShot_16.PNG')
```

Out[12]:



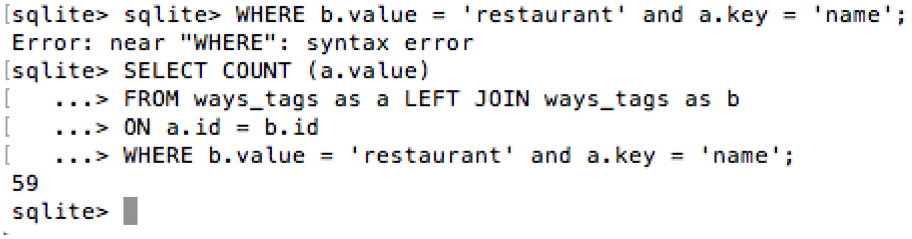
```
sqlite> .tables
nodes        nodes_tags  ways        ways_nodes  ways_tags
sqlite> SELECT COUNT(id)  FROM nodes;
236383
sqlite> SELECT COUNT(id)  FROM nodes_tags;
11040
sqlite> SELECT COUNT(id)  FROM ways;
29940
sqlite> SELECT COUNT(id)  FROM ways_nodes;
285005
sqlite> SELECT COUNT(id)  FROM ways_tags;
148994
sqlite> 
```

To get started, I was curious how many restaurants there are in my town and nearbay. Plenty of choices.

(Image: ScreenShot_18.PNG)

```
In [13]: Image(filename='Images/ScreenShot_18.PNG')
```

Out[13]:



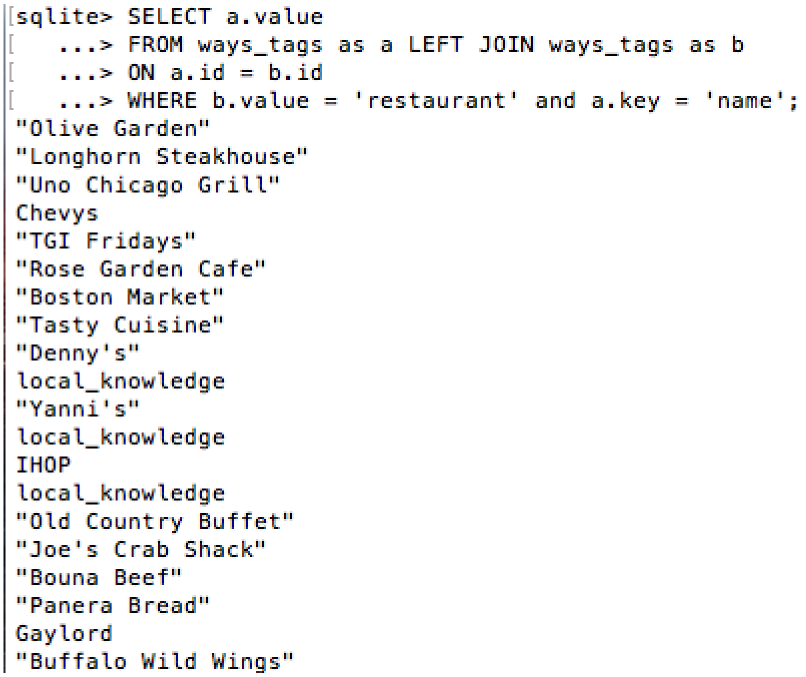
```
[sqlite> sqlite> WHERE b.value = 'restaurant' and a.key = 'name';
Error: near "WHERE": syntax error
[sqlite> SELECT COUNT (a.value)
[   ...> FROM ways_tags as a LEFT JOIN ways_tags as b
[   ...> ON a.id = b.id
[   ...> WHERE b.value = 'restaurant' and a.key = 'name';
59
sqlite> 
```

And here is beginning of the restaurant list:

(Image: ScreenShot_17.PNG)

```
In [14]: Image(filename='Images/ScreenShot_17.PNG')
```

Out[14]:



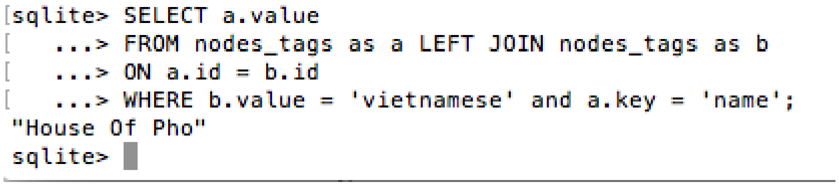
```
[sqlite> SELECT a.value
[   ...> FROM ways_tags as a LEFT JOIN ways_tags as b
[   ...> ON a.id = b.id
[   ...> WHERE b.value = 'restaurant' and a.key = 'name';
"Olive Garden"
"Longhorn Steakhouse"
"Uno Chicago Grill"
Chevys
"TGI Fridays"
"Rose Garden Cafe"
"Boston Market"
"Tasty Cuisine"
"Denny's"
local_knowledge
"Yanni's"
local_knowledge
IHOP
local_knowledge
"Old Country Buffet"
"Joe's Crab Shack"
"Bouna Beef"
"Panera Bread"
Gaylord
"Buffalo Wild Wings"
```

And only one restaurant which serves Vietnamese cuisine.

(Image: ScreenShot_19.PNG)

```
In [15]: Image(filename='Images/ScreenShot_19.PNG')
```

Out[15]:



```
[sqlite> SELECT a.value
[   ...> FROM nodes_tags as a LEFT JOIN nodes_tags as b
[   ...> ON a.id = b.id
[   ...> WHERE b.value = 'vietnamese' and a.key = 'name';
"House Of Pho"
sqlite> 
```

Some other observations:

Found secondary level tags of the node 'node' element, which k atribute holds 'created_by' value. Not sure if this information has any value for the map.

<tag k="created_by" v="Potlatch 0.10b"/>

Some key attributes with value = 'building' lists an address instead of the name:

```
sqlite> SELECT a.value FROM nodes_tags as a LEFT JOIN nodes_tags as b ON a.id = b.id WHERE a.key = 'name' and b.key = 'building' ORDER BY a.value;
```

```
999 East Touhy
```

Most of the schools have no addresses listed. School count:

```
sqlite> SELECT COUNT(a.value) FROM nodes_tags as a LEFT JOIN nodes_tags as b ON a.id = b.id WHERE b.value = 'school' and a.key = 'name' ORDER BY a.value;
```

```
95
```

School with the address:

```
sqlite> SELECT COUNT(a.value) FROM nodes_tags as a LEFT JOIN nodes_tags as b ON a.id = b.id WHERE b.value = 'school' and a.key = 'street' ORDER BY a.value;
```

```
1
```

I checked, how many Churches are in the area:

```
sqlite> SELECT COUNT(a.value) FROM nodes_tags as a LEFT JOIN nodes_tags as b ON a.id = b.id WHERE b.value = 'place_of_worship' and a.key = 'name' ORDER BY a.value;
```

```
81
```

And this query explains why the number of churches is so high. Because of double entries:

```
sqlite> SELECT a.value FROM nodes_tags as a LEFT JOIN nodes_tags as b ON a.id = b.id WHERE b.value = 'school' and a.key = 'name' ORDER BY a.value;
```

```
..

Robert Frost Elementary School

Robert Frost Elementary School

Saint John School

Saint Johns School

..
```

```
sqlite> SELECT a.value FROM nodes_tags as a LEFT JOIN nodes_tags as b ON a.id = b.id WHERE b.value = 'place_of_worship' and a.key = 'name' ORDER BY a.value;
```

```
First United Methodist Church

First United Methodist Church

Immanuel Lutheran Church

Immanuel Lutheran Church

Saint Johns Church

Saint Johns Church

Saint Martin's Episcopal Church

Saint Nicholas Episcopal Church

..
```

Full address as a house number value

```
sqlite> SELECT b.value FROM nodes_tags as a LEFT JOIN nodes_tags as b ON a.id = b.id WHERE b.key = 'housenumber' and a.key = 'name';
```

```
2310 South Elmhurst Road
```

Conclusion

After taken a look at the map, I realized that there could be done a lot of stuff to improve my neighborhood map. To list a few - almost all schools and churches have no listed address features. That alone would be a ton of work. I would need to do more research to get to know the process of getting involved. But, I think, there is a very active community to join and get a help.

Recources

https://mapzen.com/data/metro-extracts/your-extracts/c24fa71992dc (https://mapzen.com/data/metro-extracts/your-extracts/c24fa71992dc)

http://wiki.openstreetmap.org/wiki/OSM_XML (http://wiki.openstreetmap.org/wiki/OSM_XML)

http://wiki.openstreetmap.org/wiki/Map_Features (http://wiki.openstreetmap.org/wiki/Map_Features)

http://www.w3schools.com/sql/ (http://www.w3schools.com/sql/)

http://stackoverflow.com/questions/9751548/how-do-i-correctly-paste-multi-line-xml-snippet-to-github-wiki-when-using-markdo (http://stackoverflow.com/questions/9751548/how-do-i-correctly-paste-multi-line-xml-snippet-to-github-wiki-when-using-markdo)