**OpenStreetMap Project**

**Map Area**

Edmond, OK, United States

* <https://www.openstreetmap.org/relation/184313#map=11/35.6238/-97.3347>

For this project, I have chosen Edmond, a small town North of Oklahoma City where I lived for 3 years when I just moved to the United States.

**Problems Encountered in the Map**

Data auditing:

After downloading osm file I created a sample part of the original map file by using suggested code and reviewed it in notepad. Some of the problems I have noticed were as follows:

1. Some of the tags contained ‘tiger’-type tags with information about area streets names and postal codes:

<tag k="name" v="North Portland Avenue" />

<tag k="tiger:name\_base" v="Portland" />

<tag k="tiger:name\_direction\_prefix" v="N" />

<tag k="tiger:name\_type" v="Ave" />

<tag k="tiger:zip\_left" v="73003" />

<tag k="tiger:zip\_right" v="73134" />

The other tags just had ‘addr:street’ and ‘addr:postcode’ names as keys, thus suggesting several sources of information

1. Some street names were abbreviated:

<tag k="addr:street" v="NW 155TH STREET"

<tag k="addr:street" v="SOnny Blues Pl" />

1. Some zip codes were in different format included dash, had several values listed with ‘;’:

<tag k="addr:postcode" v="73034-8584" />

Data cleaning:

1. In order to combine ‘tiger’-type information with the other street names I have created new tags ‘addr:street’ and ‘addr:postcode’ for the way and node elements that had ‘tiger’ data tags and copied ‘name’ and ‘tiger:zip\_left’ values into them (there were also ‘tiger:zip\_right’ values, but I chose ‘left’ to be consistent). Then I wrote all the modifications into Edmond.xml file and continued to work with it.
2. After collecting all the street names as values with the key ‘addr:street’ I have used python code to change the abbreviated names to the full name using mapping dictionary:

def clean\_street\_name(name, mapping):

better\_name = []

for split\_name in name.split(' '):

if split\_name in mapping.keys():

split\_name = mapping[split\_name]

better\_name.append(split\_name.title())

better\_name = ' '.join(better\_name)

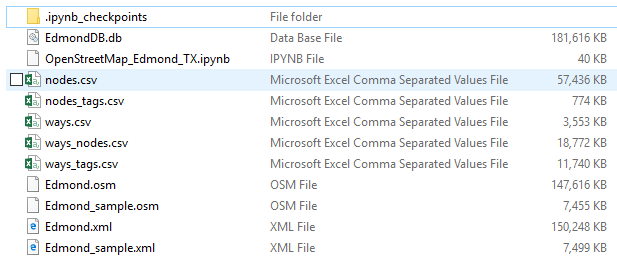
return better\_name

1. To create consistent zip codes I just took the first 5 digits of the ‘addr:postcode’ values

After writing cleaned data into 5 csv files and downloading them into DB Browser (SQlite) software into a new database numerous other problems related to ways and nodes tags keys and values data were discovered. Some of them were manually corrected.

**Data Overview**

* Files sizes are as shown below:



Edmond.xml file is about 150 MB in size.

* Number of unique users contributing to this dataset is 515.

The top 10 contributors are:

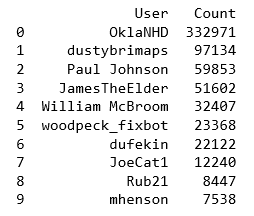
QUERY = '''

SELECT e.user as User, COUNT(\*) as Count

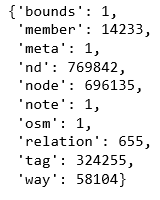
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e

GROUP BY e.user ORDER BY COUNT(\*) DESC;

'''



* Number of different tags occurring in the file is:



Some additional information about town extracted from database using SQLite queries:

* Count of top 5 amenities:

QUERY = '''

SELECT e.value as Amenity, COUNT(e.value) as Count

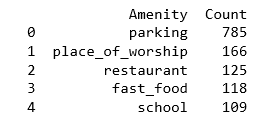
FROM (SELECT key, value FROM nodes\_tags WHERE nodes\_tags.key='amenity'

UNION ALL SELECT key, value FROM ways\_tags

WHERE ways\_tags.key='amenity') e

GROUP BY e.value ORDER BY COUNT(e.value) DESC;

'''



* Amount of church types:

QUERY = '''

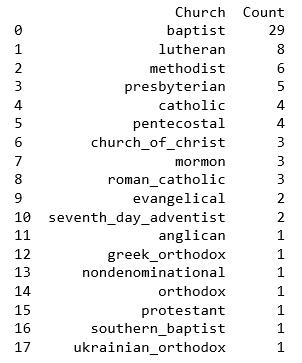
SELECT e.value as Church, COUNT(e.value) as Count

FROM (SELECT value FROM nodes\_tags WHERE key='denomination'

UNION ALL SELECT value FROM ways\_tags WHERE ways\_tags.key = 'denomination') e

GROUP BY e.value ORDER BY COUNT(e.value) DESC, e.value;

'''



* Top 10 popular cuisines:

QUERY = '''

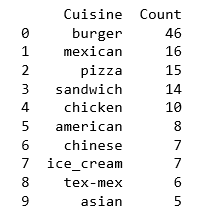
SELECT e.value as Cuisine, COUNT(e.value) as Count

FROM (SELECT value FROM nodes\_tags WHERE key='cuisine'

UNION ALL SELECT value FROM ways\_tags WHERE ways\_tags.key = 'cuisine') e

GROUP BY e.value ORDER BY COUNT(e.value) DESC, e.value;

'''



* Number of Schools:

QUERY1 = '''

SELECT COUNT(DISTINCT(e.value)) as Elementary

FROM (SELECT value FROM nodes\_tags WHERE key='name' AND value LIKE '%Elementary School'

UNION ALL SELECT value FROM ways\_tags WHERE key = 'name' AND value LIKE '%Elementary School') e

ORDER BY e.value;

'''

QUERY2 = '''

SELECT COUNT(DISTINCT(e.value)) as Middle

FROM (SELECT value FROM nodes\_tags WHERE key='name' AND value LIKE '%Middle School'

UNION ALL SELECT value FROM ways\_tags WHERE key = 'name' AND value LIKE '%Middle School') e

ORDER BY e.value;

'''

QUERY3 = '''

SELECT COUNT(DISTINCT(e.value)) as High

FROM (SELECT value FROM nodes\_tags WHERE key='name' AND value LIKE '%High School'

UNION ALL SELECT value FROM ways\_tags WHERE key = 'name' AND value LIKE '%High School') e

ORDER BY e.value;

'''



* Popular spots in town:

QUERY = '''

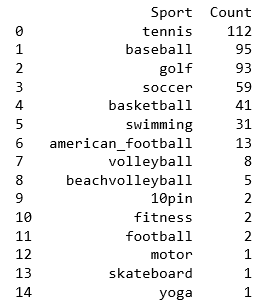
SELECT e.value as Sport, COUNT(e.value) as Count

FROM (SELECT value FROM nodes\_tags WHERE key='sport'

UNION ALL SELECT value FROM ways\_tags WHERE ways\_tags.key = 'sport') e

GROUP BY e.value ORDER BY COUNT(e.value) DESC, e.value;

'''



* Count of eating places with their names and type (the top 30 shown):

QUERY = '''

SELECT COUNT(e.Name) as Number, e.Name, e.Type FROM

(SELECT c.value as Type, b.value as Name, a.value as Cuisine

FROM nodes\_tags as a, nodes\_tags as b, nodes\_tags as c

WHERE a.id = b.id AND a.id = c.id AND a.key = 'cuisine' AND a.type = 'regular' AND b.key = 'name' AND b.type = 'regular' AND c.key = 'amenity' AND c.type = 'regular'

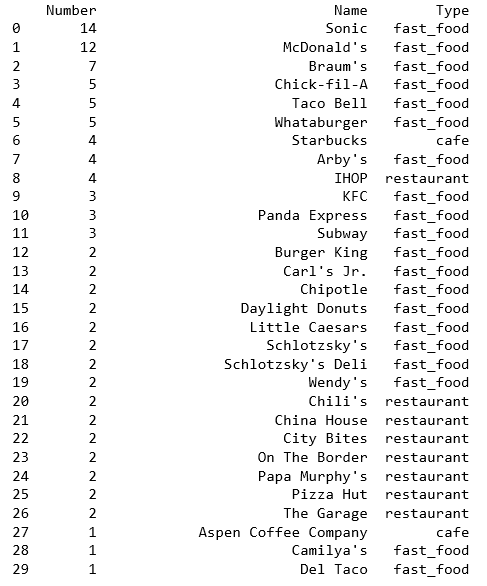
UNION ALL SELECT c.value as Type, b.value as Name, a.value as Cuisine

FROM ways\_tags as a, ways\_tags as b, ways\_tags as c

WHERE a.id = b.id AND a.id = c.id AND a.key = 'cuisine' AND a.type = 'regular' AND b.key = 'name' AND b.type = 'regular' AND c.key = 'amenity' AND c.type = 'regular') e

GROUP BY e.Name ORDER BY Count(e.Name) DESC, e.Type, e.Name;

'''



**Other ideas**

There are a lot of confusing and inconsistent data in this dataset especially concerning key-value pairs in nodes and ways tags which could be cleaned with the code, e.g. phone numbers, amenities, restaurants, etc. Although manual cleaning from users familiar with the area would be necessary. Also removing repeated values from ‘tiger’ elements would help with cleaning as well.