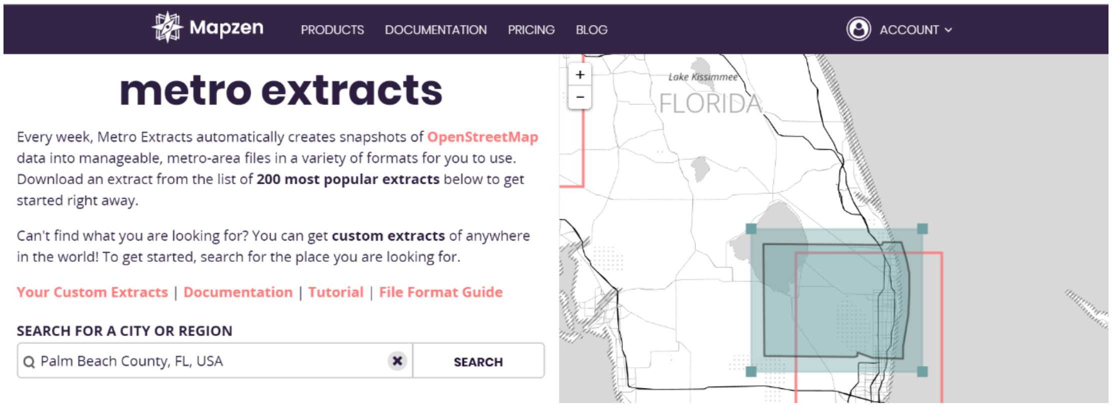
Open Street Data Map Case Study

Map Area:

Palm Beach County, FL

Open Street Map link: https://www.openstreetmap.org/relation/1210748

MapZen Screenshot: The highlighted area in blue is the extract taken for analysis.



metro extracts

Every week, Metro Extracts automatically creates snapshots of OpenStreetMap data into manageable, metro-area files in a variety of formats for you to use. Download an extract from the list of **200 most popular extracts** below to get started right away.

Can't find what you are looking for? You can get **custom extracts** of anywhere in the world! To get started, search for the place you are looking for.

Your Custom Extracts | Documentation | Tutorial | File Format Guide

SEARCH FOR A CITY OR REGION

Palm Beach County, FL, USA

I currently live in Palm Beach Gardens and I was interested to explore the data set in the county which I live in i.e. Palm Beach County.

Overview of the Data

Below is a statistics about the data in context:

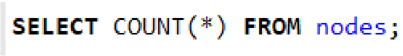
File Name	File Size
Palm Beach County.osm	192 MB
palmbeach.db	166 MB
nodes.csv	73.5 MB
nodes_tags.csv	1.81 MB
ways.csv	5.62 MB
ways_nodes.csv	25 MB
ways_tags.csv	16.1 MB

Number of unique users:

```
SELECT COUNT(DISTINCT(a.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) a;
```

701

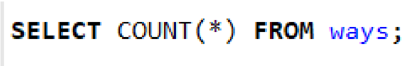
Number of Nodes:



```
SELECT COUNT(*) FROM nodes;
```

1846632

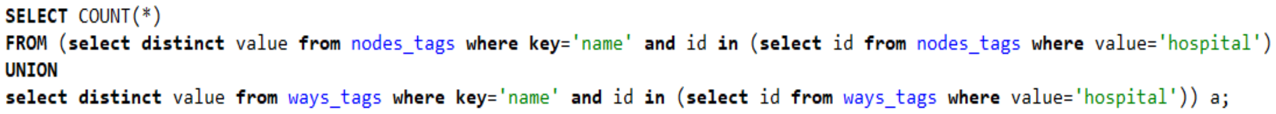
Number of Ways:



```
SELECT COUNT(*) FROM ways;
```

196734

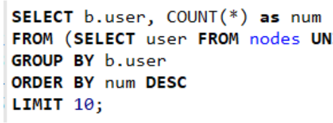
Number of Hospitals in Palm Beach County:



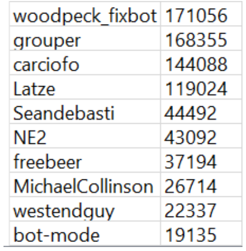
```
SELECT COUNT(*)
FROM (select distinct value from nodes_tags where key='name' and id in (select id from nodes_tags where value='hospital')
UNION
select distinct value from ways_tags where key='name' and id in (select id from ways_tags where value='hospital')) a;
```

24

Top 10 users who have contributed to Palm Beach County OSM:

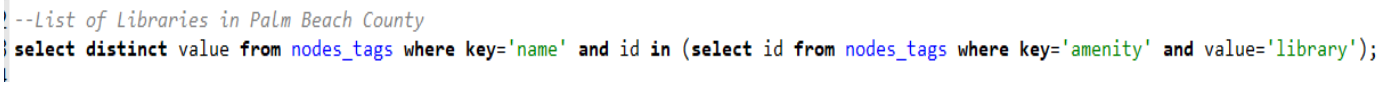


```
SELECT b.user, COUNT(*) as num
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) b
GROUP BY b.user
ORDER BY num DESC
LIMIT 10;
```

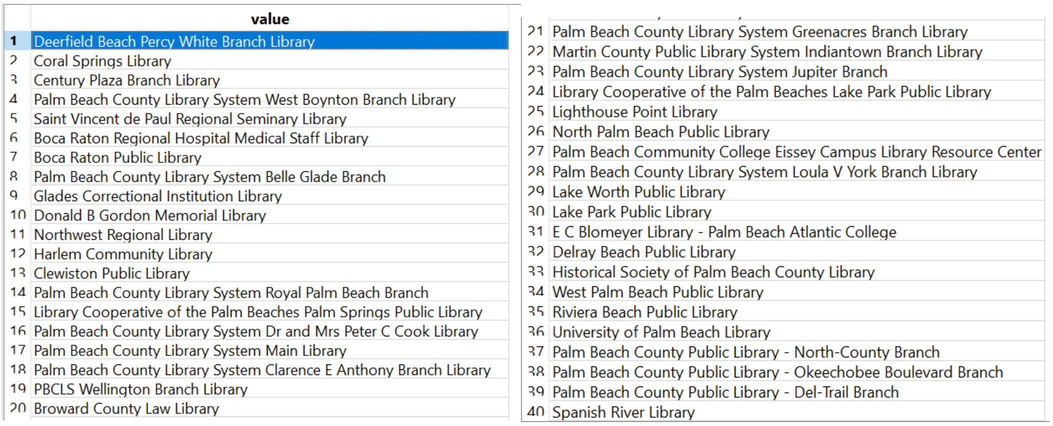


woodpeck_fixbot	171056
grouper	168355
carciofo	144088
Latze	119024
Seandebasti	44492
NE2	43092
freebeer	37194
MichaelCollinson	26714
westendguy	22337
bot-mode	19135

List of Libraries in Palm Beach County:



```
--List of Libraries in Palm Beach County
select distinct value from nodes_tags where key='name' and id in (select id from nodes_tags where key='amenity' and value='library');
```

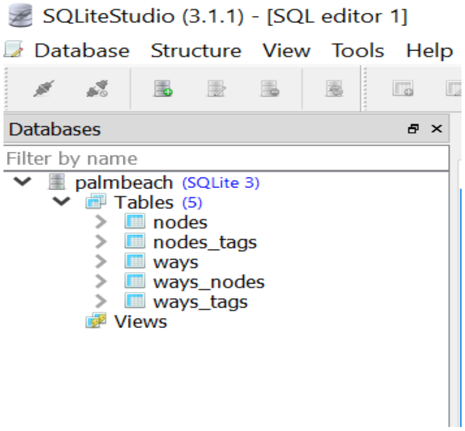


	value
1	Deerfield Beach Percy White Branch Library
2	Coral Springs Library
3	Century Plaza Branch Library
4	Palm Beach County Library System West Boynton Branch Library
5	Saint Vincent de Paul Regional Seminary Library
6	Boca Raton Regional Hospital Medical Staff Library
7	Boca Raton Public Library
8	Palm Beach County Library System Belle Glade Branch
9	Glades Correctional Institution Library
10	Donald B Gordon Memorial Library
11	Northwest Regional Library
12	Harlem Community Library
13	Clewiston Public Library
14	Palm Beach County Library System Royal Palm Beach Branch
15	Library Cooperative of the Palm Beaches Palm Springs Public Library
16	Palm Beach County Library System Dr and Mrs Peter C Cook Library
17	Palm Beach County Library System Main Library
18	Palm Beach County Library System Clarence E Anthony Branch Library
19	PBCLS Wellington Branch Library
20	Broward County Law Library
21	Palm Beach County Library System Greenacres Branch Library
22	Martin County Public Library System Indiantown Branch Library
23	Palm Beach County Library System Jupiter Branch
24	Library Cooperative of the Palm Beaches Lake Park Public Library
25	Lighthouse Point Library
26	North Palm Beach Public Library
27	Palm Beach Community College Eissey Campus Library Resource Center
28	Palm Beach County Library System Loula V York Branch Library
29	Lake Worth Public Library
30	Lake Park Public Library
31	E C Blomeyer Library - Palm Beach Atlantic College
32	Delray Beach Public Library
33	Historical Society of Palm Beach County Library
34	West Palm Beach Public Library
35	Riviera Beach Public Library
36	University of Palm Beach Library
37	Palm Beach County Public Library - North-County Branch
38	Palm Beach County Public Library - Okeechobee Boulevard Branch
39	Palm Beach County Public Library - Del-Trail Branch
40	Spanish River Library

Problems Encountered in the map extract:

After downloading the osm extract for Palm Beach County, I ran the extract against data.py file to create the following 5 files: *nodes.csv, nodes_tags.csv, ways.csv, ways_nodes.csv, ways_tags.csv*.

These files were later used to create palmbeach database with the tables as shown in the screenshot below. I used SQLite Studio for this purpose since its open source and very user friendly.

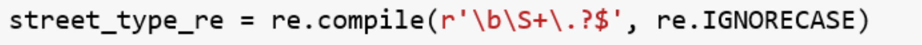


With the help of SQL queries and utilizing the audit.py, following were the main problems encountered:

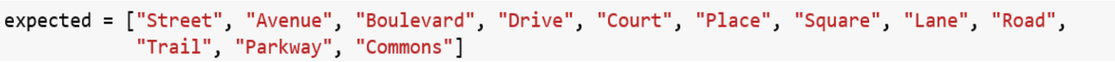
- ***Inconsistent Street Types:*** Inconsistences in street types like Street (St, St.), Avenue (Ave), Road (Rd, Rd.), Boulevard (Blvd, Blvd.), Drive (Dr, Dr.)
- ***Invalid Postcodes:*** Only those postcodes starting with 334 are considered valid Postcodes which fall within Palm Beach County. There are some invalid zip codes which start with FL (Eg: FL 33433) which needs to be updated by removing "FL" from them
- ***Inconsistent phone format:*** Phone numbers were present in multiple formats. Only phone numbers in xxx-xxx-xxxx format are considered valid. Some of the other formats were +1 xxx-xxx-xxx, (xxx)-xxx-xxxx, xxx xxx xxxx, etc.
- ***Data imported from GNIS seem to have old names*** and they have not been updated. For example: "Junior High School" is still widely used for many school names even when it was renamed as "Middle School"

Inconsistent Street Types

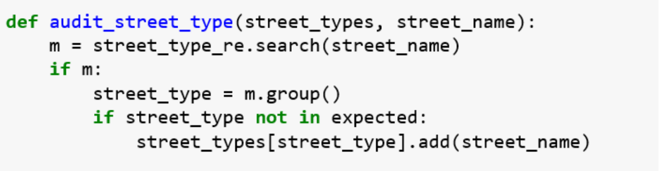
Utilizing the below code segment from audit.py, all the inconsistent street types were analyzed



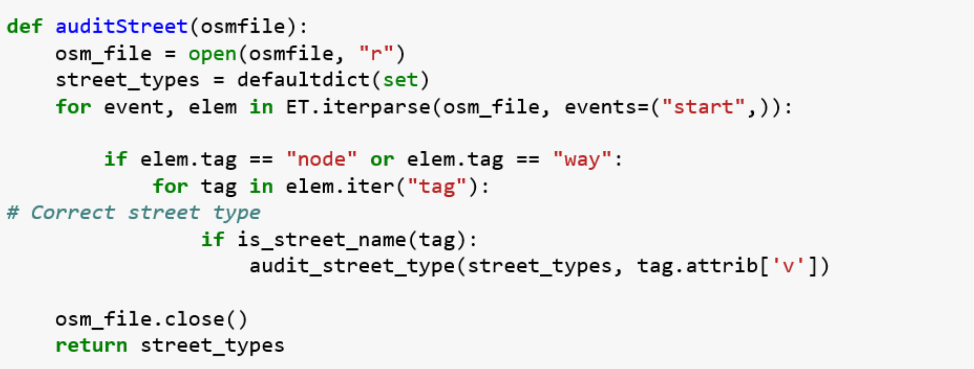
```
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)
```



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

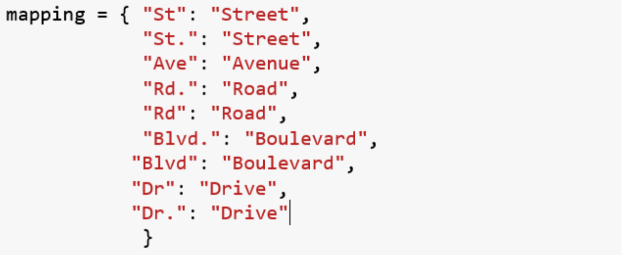


```
def audit_street_type(street_types, street_name):
    m = street_type_re.search(street_name)
    if m:
        street_type = m.group()
        if street_type not in expected:
            street_types[street_type].add(street_name)
```



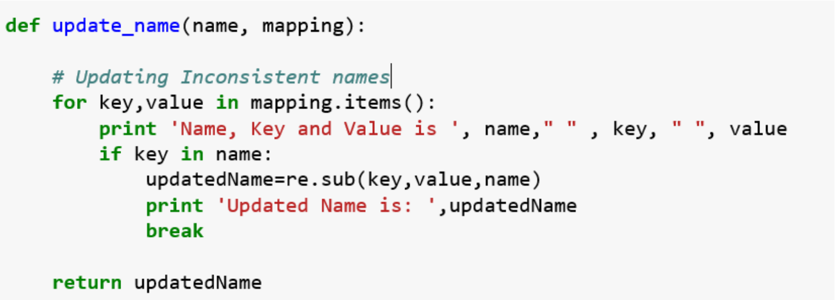
```
def auditStreet(osmfile):
    osm_file = open(osmfile, "r")
    street_types = defaultdict(set)
    for event, elem in ET.iterparse(osm_file, events=("start",)):

        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
# Correct street type
                if is_street_name(tag):
                    audit_street_type(street_types, tag.attrib['v'])

    osm_file.close()
    return street_types
```

Based on the above analysis, a mapping Dictionary as shown below was created to fix the inconsistencies.



```
mapping = { "St": "Street",
            "St.": "Street",
            "Ave": "Avenue",
            "Rd.": "Road",
            "Rd": "Road",
            "Blvd.": "Boulevard",
           "Blvd": "Boulevard",
           "Dr": "Drive",
           "Dr.": "Drive"
            }
```

Utilizing the above mapping dictionary, inconsistent street types are upated

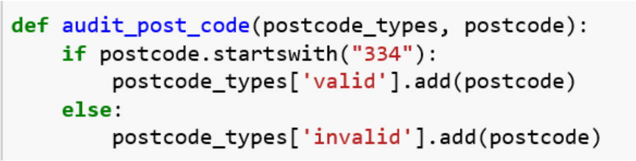


```
def update_name(name, mapping):

    # Updating Inconsistent names
    for key,value in mapping.items():
        print 'Name, Key and Value is ', name," " , key, " ", value
        if key in name:
            updatedName=re.sub(key,value,name)
            print 'Updated Name is: ',updatedName
            break

    return updatedName
```

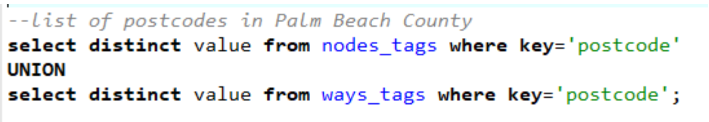
This updated all inconsistent addresses with street types like St, St., Ave, Rd., Rd, Blvd., Blvd, Dr, Dr. to their corresponding expected Street types.

Invalid Post Codes

Utilizing the below code snippet and SQL query, the formats of postcodes is the osm file were understood. From research it was found that all Palm beach County postcodes start with “334”. So, all other postcodes were considered invalid.

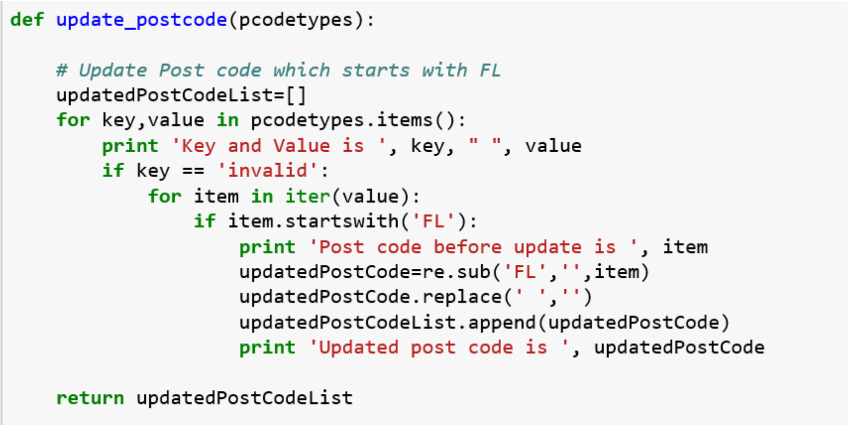


```
def audit_post_code(postcode_types, postcode):
    if postcode.startswith("334"):
        postcode_types['valid'].add(postcode)
    else:
        postcode_types['invalid'].add(postcode)
```



```
--list of postcodes in Palm Beach County
select distinct value from nodes_tags where key='postcode'
UNION
select distinct value from ways_tags where key='postcode';
```

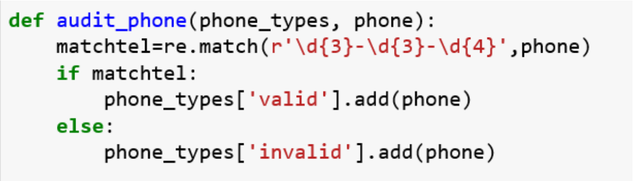
It was found out that there were multiple instances when postcode was starting with FL (Eg: FL 33433). The below code snippet was used to update these set of incorrect postcodes.



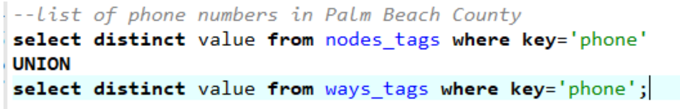
```
def update_postcode(pcodetypes):

    # Update Post code which starts with FL
    updatedPostCodeList=[]
    for key,value in pcodetypes.items():
        print 'Key and Value is ', key, " ", value
        if key == 'invalid':
            for item in iter(value):
                if item.startswith('FL'):
                    print 'Post code before update is ', item
                    updatedPostCode=re.sub('FL','',item)
                    updatedPostCode.replace(' ','')
                    updatedPostCodeList.append(updatedPostCode)
                    print 'Updated post code is ', updatedPostCode

    return updatedPostCodeList
```

Inconsistent Phone format

Utilizing below code snippet and SQL query, the formats of phone numbers in the osm file were understood. Phone numbers in xxx-xxx-xxxx are only considered as in the valid format.



```
def audit_phone(phone_types, phone):
    matchtel=re.match(r'\d{3}-\d{3}-\d{4}',phone)
    if matchtel:
        phone_types['valid'].add(phone)
    else:
        phone_types['invalid'].add(phone)
```

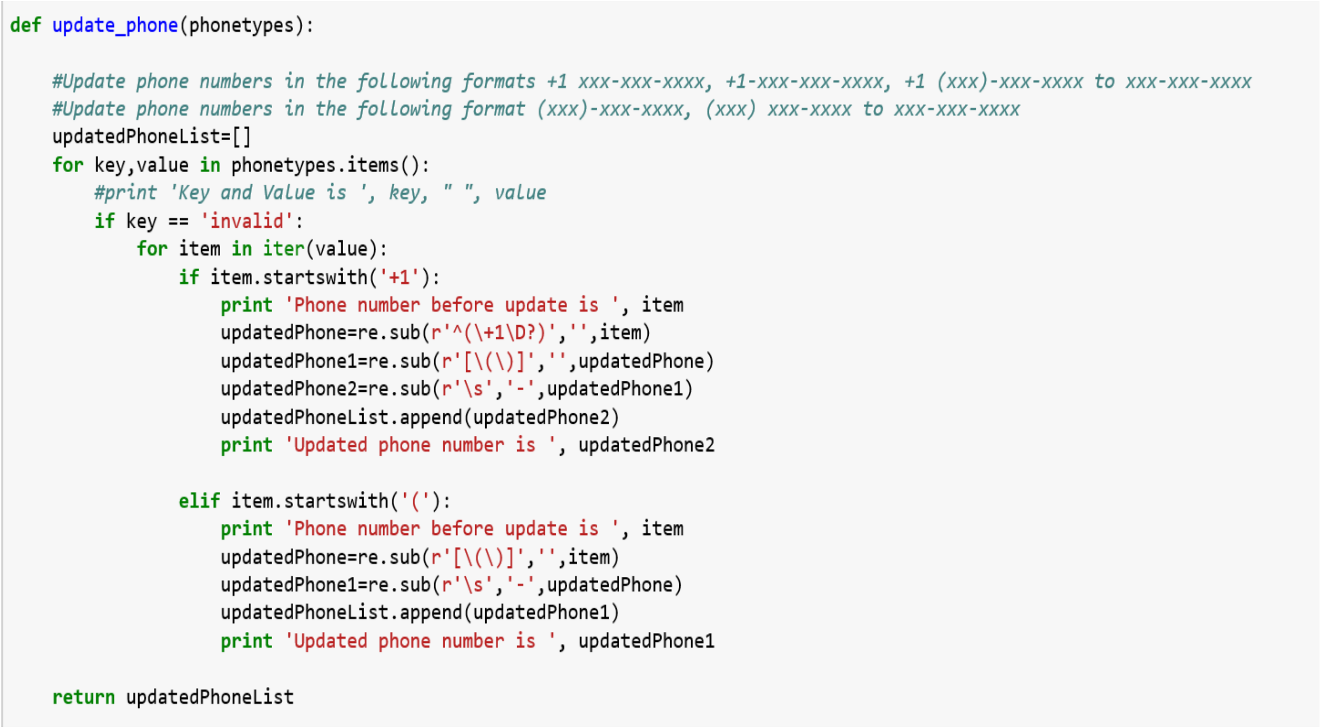


```
--list of phone numbers in Palm Beach County
select distinct value from nodes_tags where key='phone'
UNION
select distinct value from ways_tags where key='phone';
```

Some of the major formats generated by the above query included

- xxx-xxx-xxxx
- +1 xxx xxx xxxx
- (xxx) xxx xxxx
- (xxx)-xxx-xxxx

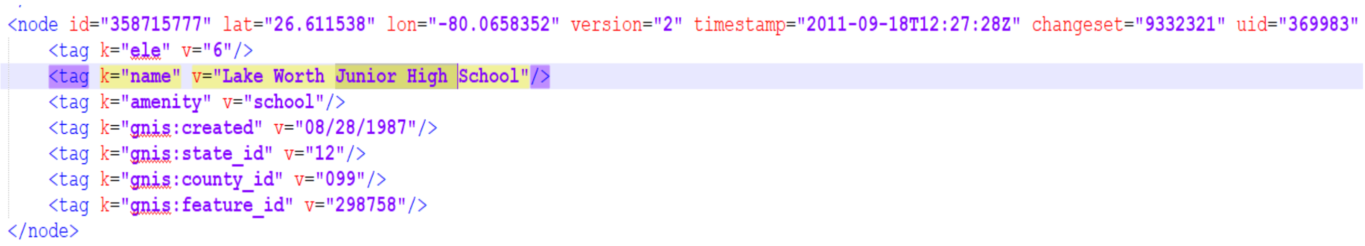
Utilizing the below code snippet some of the inconsistent formats were updated



```
def update_phone(phonetypes):

    #Update phone numbers in the following formats +1 xxx-xxx-xxxx, +1-xxx-xxx-xxxx, +1 (xxx)-xxx-xxxx to xxx-xxx-xxxx
    #Update phone numbers in the following format (xxx)-xxx-xxxx, (xxx) xxx-xxxx to xxx-xxx-xxxx
    updatedPhoneList=[]
    for key,value in phonetypes.items():
        #print 'Key and Value is ', key, " ", value
        if key == 'invalid':
            for item in iter(value):
                if item.startswith('+1'):
                    print 'Phone number before update is ', item
                    updatedPhone=re.sub(r'^(\+1\D?)','',item)
                    updatedPhone1=re.sub(r'[\(\)]','',updatedPhone)
                    updatedPhone2=re.sub(r'\s','-',updatedPhone1)
                    updatedPhoneList.append(updatedPhone2)
                    print 'Updated phone number is ', updatedPhone2

                elif item.startswith('('):
                    print 'Phone number before update is ', item
                    updatedPhone=re.sub(r'[\(\)]','',item)
                    updatedPhone1=re.sub(r'\s','-',updatedPhone)
                    updatedPhoneList.append(updatedPhone1)
                    print 'Updated phone number is ', updatedPhone1

    return updatedPhoneList
```

Old Names found in GNIS database

In the below osm snippet, it can be seen that the school name is given as “Lake Worth Junior High School”. This is an old format and the newer version is “Lake Worth Middle School”.



```
<node id="358715777" lat="26.611538" lon="-80.0658352" version="2" timestamp="2011-09-18T12:27:28Z" changeset="9332321" uid="369983"
    <tag k="ele" v="6"/>
    <tag k="name" v="Lake Worth Junior High School"/>
    <tag k="amenity" v="school"/>
    <tag k="gnis:created" v="08/28/1987"/>
    <tag k="gnis:state_id" v="12"/>
    <tag k="gnis:county_id" v="099"/>
    <tag k="gnis:feature_id" v="298758"/>
</node>
```

Additional Ideas

Data cleansing automation

Based on the previous analysis of old names found in GNIS database, it was found out that there are lots of old and stale data present in the OSM data set. We can think of a mechanism where bots can be used to validate this data very frequently so that the most valid and recent data is available at any point of time.

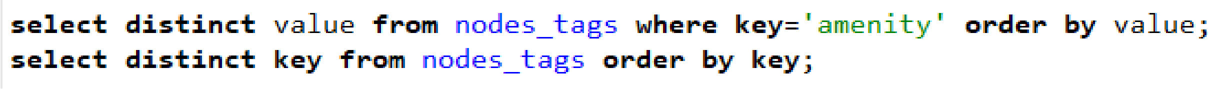
Currently, less than 2% of users in Palm Beach county are automated (user bot-mode is the most dominant automated user). This percentage share has to definitely improve so as to retrieve a high quality data from OSM data set.

The possible challenges associated to data cleansing could be deciding on the reliable data source which the bots need to rely on and the frequency of cleansing (since the data set is very huge).

Standardizing keys and values across data set

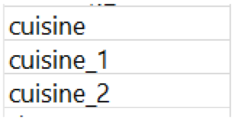
After exploring the existing data set for various keys and values being used, I found that there is lot of inconsistency on how this data is used across the data set.

I used the following queries to explore the various keys/values



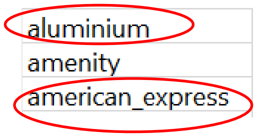
```
select distinct value from nodes_tags where key='amenity' order by value;
select distinct key from nodes_tags order by key;
```

For example, there are 3 different keys to define cuisine as shown below. This brings lots of inconsistencies in the final data



cuisine
cuisine_1
cuisine_2

Another example is:



aluminium
amenity
american_express

Key names like “Aluminium” or “American Express” should be avoided and they should be more standardized.

So by standardizing the data, the OSM data set becomes more reliable and more easy to understand for a user who is updating the data.

Conclusion

After exploring the data set, I could find that a major portion of the data for Palm Beach County is fed from TIGER data, produced by the US Census Bureau and GNIS database.

The analysis of the data provided me great insights in to the inconsistencies and invalid data that is present. It also gave me an opportunity to fix some main data prevalent issues which could seriously hamper data analysis. With more standardized data input formats, it would be possible to input a great amount of cleaned data to OpenStreetMap.org.