Francis Samkoh

WGU Data Analyst - NNANODEGREE PROGRAM
Data Wrangling with SQL Project
Map Area: Silver Spring Maryland USA
https://www.openstreetmap.org/relation/133501

Project Overview

You will choose any area of the world in https://www.openstreetmap.org and use data munging techniques, such as assessing the quality of the data for validity, accuracy, completeness, consistency and uniformity, to clean the OpenStreetMap data for a part of the world that you care about. Finally, you will choose either MongoDB or SQL as the data schema to complete your project.

This area is in Montgomery County in Maryland USA with bounds as follows

- Minimum latitude =38.9631000
- Minimum longitude =-77.1197000
- Maximum latitude =39.0543000
- Maximum longitude =-76.9161000

This area had a great part in my life. It was the first place I settled in when I came to the US before recently moving to Texas. I would like to see what there is to clean in this city's data and explore the data for some insight.

Auditing

I ran the Silver Spring openstreetmap file against Tag.py file to get a count of the unique top-level elements and got the following results:

- bounds: 1
- member: 18834
- meta: 1
- nd: 743745
- node: 625049
- note: 1
- osm: 1
- relation: 838
- tag: 467776
- way: 84964

I ran the Silver Spring openstreetmap file against Patern_Count.py file to find multiple patterns including any problems in the 'k' value for each tag, (where "lower" is for valid lower case letters, "lower_colon" is for valid values with colon in the value, and problem is for tags with problematic characters, and "other" is for tags that don't fall in the above three mentioned) and found the following:

- lower: 225255
- lower_colon: 238967
- other: 3553
- problem: 1

Problems Encountered in the map

Used the Sample.py codes to create a sample file for Silver Spring openstreetmap. I Noticed some irregular street name when I ran silver Spring map file against the Audit_Street.py file (#101: 1 and 20740: 1 amongst

others). So, I ran the Silver Spring map against the Audit_Street.py file and even found more irregular street names and so many abbreviations. So, I re-ran the Silver Spring Map against Audit_Street_Detail.py and printing out only the very odd street names that I did not understand. Some were apartment building numbers, and some were zip codes, and these were not included in the cleaning. Other problems included the following:

- Mis-spelled street name (Aveenue: 1)
- Over abbreviation of street name (Ave, Dr, Ct, Ln, Rd)
- Lower case (ave)

I used the Clean_Street.py code to clean the street names and write the new clean osm file to Clean_SilverSpringMap.xml

Data Overview

I ran the Clean_SilverSpringMap.xml through the SQL_Export.py codes to create the .csv files. Created a database with the required tables using the Database_Create.py codes. I then used the Query.py code to query the database for insight

Number of nodes

```
def number_of_nodes():
   result = cur.execute('SELECT COUNT(*) FROM nodes')
   return result.fetchone()[0]
```

- Number of nodes ---------- 625049

Number of ways

```
def number_of_ways():
   result = cur.execute('SELECT COUNT(*) FROM ways')
   return result.fetchone()[0]
```

- Number of ways ---------- 84964

Number of unique users

```
def number_of_Unique_users():
   result = cur.execute('SELECT COUNT(distinct(uid)) FROM (SELECT uid FROM nodes UNION ALL
SELECT uid FROM ways)')
   return result.fetchone()[0]
```

- Number of unique users ---------- 992

Biggest religion

```
def Biggest_religion():
   for row in cur.execute('SELECT nodes_tags.value, COUNT(*) as num FROM nodes_tags \
                        JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value="place_of_worship") i \
                        ON nodes_tags.id=i.id \
                        WHERE nodes_tags.key="religion" \
                        GROUP BY nodes_tags.value \
                        ORDER BY num DESC\
                        LIMIT 1'):
      return row
```

- Biggest religion ---------- christian (98)

Number of users contributing once

```
def number_of_users_contributing_once():
    result = cur.execute('SELECT COUNT(*) \
            FROM \
               (SELECT e.user, COUNT(*) as num \
                FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e \
                GROUP BY e.user \
                HAVING num=1) u')
    return result.fetchone()[0]
```

- Number of users contributing once -- 217

Top Contributing users

```
def top_contributing_users():
    users = []
    for row in cur.execute('SELECT e.user, COUNT(*) as num \
            FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e \
            GROUP BY e.user \
            ORDER BY num DESC \
            LIMIT 10'):
        users.append(row)
    return users
```

```
Gregory_M-NCPPC_Import --- 182804
aude --- 123565
woodpeck_fixbot ----------------- 28981
kriscarle ---------------------------- 26246
abflts ------------------------------- 19865
S_H ---------------------------------- 18541
ecaldwell ---------------------------- 18205
RoadGeek_MD99 ---------------- 17130
bikepathmapper ------------------- 16542
xzonjali ----------------------------- 14744
```

Popular Amenities

```
def popular_amenities():
    amenities = []
    for row in cur.execute('SELECT value, COUNT(*) as num \
                        FROM nodes_tags \
                        WHERE key="amenity" \
                        GROUP BY value \
                        ORDER BY num DESC \
                        LIMIT 10'):
        amenities.append(row)
    return amenities
```

```
restaurant --------------------------- 214
school ------------------------------- 157
bench -------------------------------- 115
place_of_worship ----------------- 111
fast_food ---------------------------- 93
bicycle_parking ------------------- 86
cafe --------------------------------- 66
```

bank ---------- 63
bicycle_rental ---------- 57
drinking_water ---------- 40

Popular Cuisines

```
def popular_cuisines():
    cuisines = []
    for row in cur.execute('SELECT nodes_tags.value, COUNT(*) as num \
            FROM nodes_tags \
               JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value="restaurant") i \
               ON nodes_tags.id=i.id \
            WHERE nodes_tags.key="cuisine" \
            GROUP BY nodes_tags.value \
            ORDER BY num DESC\
            LIMIT 10'):
        cuisines.append(row)
    return cuisines
```

american ---------- 13
chinese ---------- 11
pizza ---------- 10
italian ---------- 7
seafood ---------- 6
french ---------- 5
asian ---------- 4
greek ---------- 4
sandwich ---------- 4
thai ---------- 4

Popular Cities

```
def popular_cities():
    cities = []
    for row in cur.execute("""SELECT tags.value, COUNT(*) as count
                    FROM (SELECT * FROM nodes_tags
                      UNION ALL
                      SELECT * FROM ways_tags) tags
                    WHERE tags.key = 'city'
                    GROUP BY tags.value
                    ORDER By count DESC
                    LIMIT 10"""):
        cities.append(row)
    return cities
```

Hyattsville ---------- 8399
College Park ---------- 4215
Silver Spring ---------- 2191
Beltsville ---------- 1676
University Park ---------- 928
Takoma Park ---------- 600
Bethesda ---------- 534
Riverdale ---------- 529

Washington ---------- 276
Chevy Chase ---------- 60

File sizes

SilverSpringMap.xml ---------- 146.254 MB
Clean_SilverSpringMap.xml ----- 148.034 MB
SilverSpring_sample.xml --------- 1.49 MB
nodes.csv ---------- 53.344 MB
nodes_tags.csv ---------- 1.463 MB
ways.csv ---------- 5.405 MB
ways_nodes.csv ---------- 17.816 MB
ways_tags.csv ---------- 13.981 MB
SilverSpring.db ---------- 102.588 MB

Additional Ideas

A considerable effort has been put in by various users to contribute data to the Silver Spring OSM area. For instance, I used the Audit_Postcode.py to search for bad postal code entries and found none. I think OSM should restrict data that can be entered to ensure accurate data entry. For instance, they could implement a dropdown of all the various street types such as Avenue, Street, Lane amongst others so that when one start typing a street type, some options will pop up for them to select from the list. It will help with consistency of street types. Such restrictions could be applied to other sections of OSM data entry.

Conclusion

I audited and programmatically, cleaned the streets of Silver Spring OSM. I then converted the data from XML to tabular format and exported it to a SQL database which allowed me to have a statistical overview of the dataset through querying. There is a lot of room for improvement on the quality and consistency of the dataset.