OpenStreetMap Case Study

Map Area

Roswell, GA, US

- http://www.openstreetmap.org/relation/119569 (http://www.openstreetmap.org/relation/119569)

I moved to this area 2 years ago, and chose to use this map for my project as a way to learn more about the area and improve the OpenStreetMap data

Problems found in audit

1. Inconsistent address abbreviations

('Glen Meadows Dr NW','Clubland Drive Northeast'

To clean this field, I scraped the https://pe.usps.com/text/pub28/28apc_002.htm (https://pe.usps.com/text/pub28/28apc_002.htm) website to get a listing of all expected street suffix abbreviations mapped to the full street suffix name. As well, I added mappings for cardinal directions (North, South, East, West, etc) and rather than using regular expressions to update the last word in the street name, I split the street name and iterated through each word (ex. Glen Meadows Dr NW => Glen Meadows Drive Northwest rather than Glen Meadows Dr Northwest)

In [ ]:
```
def update_addr(key, value, mapping, expected):
    if key == "street":
            value_split = value.split()
            i = 0
            new_value = ""
            while i < len(value_split):
                if value_split[i] != None:
                    word = value_split[i]
                    word = word.capitalize().replace(".", "")
                    word = update_word(word, mapping, expected)
                    new_value += word + " "
                i += 1
            return new_value
```

2. Misspelled city, unexpected city and state and zip in the city field

('Sandy Springa', 'Bismarck', 'GA 30350')

I corrected the misspelling of Sandy Springs programmatically. Upon further investigation of 'Bismarck's' node_tag ID, this value is associated with Ridey Taxi Service in Bismarck, ND 58502. Since this information does not belong in this dataset, I chose to delete it manually from the database. Investigation into the 'GA 30350' value showed that for id # 42882100 the city and postcode fields were switched. I chose to manually update this in the database as well.

```
In [ ]:     elif key == "city":
                    if value == "Sandy Springa":
                        value = "Sandy Springs"
                    return value
```

3. Inconsistent State abbreviations and capitalization, as well as inaccurate state's

('GA', 'Georgia', 'ND', 'ga')

I Chose to maintain a capitalized abbreviation (GA) and updated all fields to match. The unexpected 'ND' value was also taken care of in the deletion explained above.

```
In [ ]:     elif key == "state":
                    if value == "Georgia":
                        value = "GA"
                    return value.upper()
```

4. Inconsistent and incorrect zip codes, as well as city names in the zip code field

('58502', 'Atlanta,', '30092-4207', '1879')

The unexpected '58502' value was taken care of in the deletion explained above. I chose to standardize to just the 5 digit zip code. '1879' belongs to the id associated with 'Burger King # 4089' in Norcross, GA so the zip code should actually be '30092'. This was easily corrected in the database. The value 'Atlanta' was taken care of in the manual update to id # 42882100 described above

```
In [ ]:     elif key == "postcode":
                    return value[0:5]
                else:
                    return value
```

Database Queries

Size of files

In [3]:
```
files_list = ['roswell.db', 'roswell.osm', 'csv_files/nodes_tags.csv',
              'csv_files/nodes.csv', 'csv_files/ways.csv',
              'csv_files/ways_tags.csv', 'csv_files/ways_nodes.csv']
for each in files_list:
    print each, "....", round((os.path.getsize(each) * .000001), 1), "M
B"
```

```
roswell.db .... 75.0 MB
roswell.osm .... 126.3 MB
csv_files/nodes_tags.csv .... 1.6 MB
csv_files/nodes.csv .... 52.2 MB
csv_files/ways.csv .... 4.3 MB
csv_files/ways_tags.csv .... 9.2 MB
csv_files/ways_nodes.csv .... 14.3 MB
```

Number of Unique Users

In [4]:
```
query = "SELECT COUNT(DISTINCT(all_users.uid)) as Unique_Users FROM \
        (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) as all_us
ers"
cursor.execute(query)
print pd.read_sql_query(query, connection)
```

```
   Unique_Users
0           295
```

Number of Nodes and Ways

In [5]:
```
query = "SELECT (SELECT COUNT(*) FROM nodes) as Nodes, (SELECT COUNT(*)
 FROM ways) as Ways"
cursor.execute(query)
print pd.read_sql_query(query, connection)
```

```
    Nodes   Ways
0  539125  57442
```

Number of Cuisine Categories

In [6]:
```
query = "SELECT value, COUNT(*) as number FROM nodes_tags WHERE key = 'c
uisine' \
        GROUP BY value ORDER BY number DESC"
cursor.execute(query)
print pd.read_sql_query(query, connection)
```

```
             value  number
0            pizza       7
1          mexican       6
2      coffee_shop       5
3          italian       4
4         sandwich       4
5         american       3
6        bar&grill       3
7          chicken       3
8          seafood       3
9           burger       2
10            thai       2
11        American       1
12          Coffee       1
13        barbecue       1
14         chinese       1
15          indian       1
16        pancakes       1
17           sushi       1
18  sushi,_hibachi       1
19         tex-mex       1
20      vegetarian       1
```

Additional Statistics

Counties Represented

In [7]:
```
query = "SELECT DISTINCT(value) as County FROM nodes_tags WHERE type =
 'gnis' \
        and key = 'County' LIMIT 100"
cursor.execute(query)
print pd.read_sql_query(query, connection)
```

```
     County
0      Cobb
1    Fulton
2    DeKalb
3  Cherokee
4  Gwinnett
5   Forsyth
```

Top 10 Appearing Leisure Facilities

```
In [8]: query = "SELECT leisure.value as Leisure_Activity, COUNT(*) as Count FRO
        M \
                (SELECT value FROM nodes_tags WHERE key = 'leisure' UNION ALL \
                SELECT value FROM ways_tags WHERE key = 'leisure') as leisure\
                GROUP BY Leisure_Activity ORDER BY Count DESC LIMIT 10"
        cursor.execute(query)
        cursor.execute(query)
        print pd.read_sql_query(query, connection)
```

```
  Leisure_Activity  Count
0            pitch    284
1    swimming_pool     52
2             park     38
3      golf_course     16
4       playground     15
5    sports_centre      5
6          stadium      4
7            track      3
8     picnic_table      2
9     horse_riding      1
```

Top 10 Building Forms

```
In [9]: query = "SELECT value as Building_Form, COUNT(*) as count FROM ways_tags
         \
                WHERE key = 'BLDG_FORM' GROUP BY Building_Form ORDER BY count DE
        SC LIMIT 10"
        cursor.execute(query)
        cursor.execute(query)
        print pd.read_sql_query(query, connection)
```

```
  Building_Form  count
0  CONVENTIONAL   1543
1         RANCH    395
2   SPLIT-LEVEL    335
3      COLONIAL    217
4      BI-LEVEL    135
5        MODERN     58
6          CAPE     11
7       CLUSTER      5
8     TOWNHOUSE      5
9        DUPLEX      2
```

Top 10 Contributing Users

In [10]:
```
query = "SELECT user, COUNT(*) as contributions FROM \
        (SELECT user FROM nodes UNION ALL SELECT user FROM ways) as all_
users\
        GROUP BY user ORDER BY contributions DESC LIMIT 10"
cursor.execute(query)
print pd.read_sql_query(query, connection)
```

```
                           user  contributions
0  Saikrishna_FultonCountyImport         389241
1                         Liber          47006
2                woodpeck_fixbot          43415
3                Jack the Ripper          31721
4                         demory          12372
5                       greenv505           7154
6                    Lisa Jackson           6963
7                         afonit           5126
8                      mackerski           5064
9                jacobbraeutigam           3909
```

Suggestion for Improving the Data

In this project my focus was on cleaning the address data from the 'addr' type, however there is a lot more that can be done to improve address data to make querying and analysis easier. One suggestion could be to make sure every address field is available for each node. A full address should have the following fields: housename, housenumber, suite, street, city, state, postcode, county, and country. In the example query below you can see that address information for id 69515387 only has street, postcode and housenumber, while id 358781696 only has housenumber information. If I were to query for a count of addresses in a certain county, or for the number of address in a certain city, these would be excluded. The problem with this is it would be time consuming to find all the missing values and validate that they're correct, however simply adding the field with a placeholder value of "unavailable" would allow those addresses to be counted in queries. The second query below highlights how incomplete the address data is for nodes_tags (ex. While there are 7,083 nodes with type "addr", 7,053 have housenumber key values, while only 1,188 have state key values).

In [11]:
```
query = "SELECT * FROM nodes_tags WHERE type = 'addr' LIMIT 10"
cursor.execute(query)
print pd.read_sql_query(query, connection)
```

```
          id          key               value  type
0   69515387       street     Waterstone Way   addr
1   69515387     postcode              30076   addr
2   69515387  housenumber                200   addr
3  358781696  housenumber                755   addr
4  358781700       street               814   addr
5  358782756  housenumber                793   addr
6  358782756       street  Mimosa Boulevard   addr
7  358785462       street      School Drive   addr
8  358785462  housenumber                 86   addr
9  367912693        state                 GA   addr
```

In [12]:
```
query = "SELECT COUNT(DISTINCT(id)) as total FROM nodes_tags WHERE type
 = 'addr'"
cursor.execute(query)
print pd.read_sql_query(query, connection)

query = "SELECT key, COUNT(DISTINCT(id)) as count FROM nodes_tags WHERE
 type = 'addr' GROUP BY key ORDER BY count DESC"
cursor.execute(query)
print pd.read_sql_query(query, connection)
```

```
   total
0   7083
            key  count
0   housenumber   7053
1        street   7051
2      postcode   6224
3          city   5942
4         state   1188
5       country   1065
6        county   1059
7         suite     16
8     housename      4
9          unit      3
10     building      1
11         full      1
```