

Project 3: OpenStreetMap Data Wrangling with SQL

Name: Olivia Jonah

Map Area: Freetown, Sierra Leone; I chose this because it is my hometown.

Location: Freetown, Sierra Leone

OpenStreetMap Url

MapZen URL

1. Data Audit

The XML file uses different types of tags so using ElementTree to parse the dataset and count number of the unique tags using mapparser.py .

```
{'bounds': 1,  
'member': 356,  
'nd': 1000732,  
'node': 820215,  
'osm': 1,  
'relation': 99,  
'tag': 199055,  
'way': 171156}
```

Patterns in the Tags

The "k" value of each tag contain different patterns. Using tags.py, I created 3 regular expressions to check for certain patterns in the tags. I have counted each of four tag categories.

```
{'lower': 195953, 'lower_colon': 754, 'other': 2346, 'problemchars': 2}
```

'lower': 195953- for tags that contain only lowercase letters and are valid,

'lower_colon': 754- for otherwise valid tags with a colon in their names,

'other': 2346- for tags with problematic characters, and

'problemchars': 2 - for other tags that do not fall into the other three categories}

2. Problems Encountered in the Map

Street address inconsistencies

The main problem we encountered in the dataset is the street name inconsistencies. Below is the old name corrected with the better name. Using audit.py , we updated the names.

Abbreviations / Incomplete Street Name

```
runfile('C:/Udacity/audit1.py', wdir='C:/Udacity')

{'Circular': set(['Circular Road']),
 'Cockle': set(['Cockle Bay Road']),
 'College': set(['College Road']),
 "Derick's": set(['Derick's Drive']),
 'Gordon': set(['Gordon Drive']),
 'Lab': set(['Lab Lane']),
 'Main': set(['Main Road Juba Hill']),
 'Mamba': set(['Mamba Ridge']),
 'New': set(['New Low Cost Housing']),
 'Off': set(['Off Cape Road']),
 'Pine': set(['Pine Street']),
 'Priscilla': set(['Priscilla Street']),
 'Regent': set(['Regent Road']),
 'Sackville': set(['Sackville Lane']),
 'Samercy': set(['Samercy Place']),
 'Spur': set(['Spur Loop', 'Spur Road']),
 'Wilberforce': set(['Wilberforce Street']),
 'Wilkinson': set(['Wilkinson Road',
 'Wilkinson Road, Brinsley Johnson Drive']),
 'Williams': set(['Williams Street'])}
```

Wilberforce Street => Wilberforce Street

Off Cape Road => Off Cape Road

Williams Street => Williams Street

College Road => College Road

Derick's Drive => Derick'S Drive

Regent Road => Regent Road

Main Road Juba Hill => Main Road Juba Hill

Lab Lane => Lab Lane

Cockle Bay Road => Cockle Bay Road

Sackville Lane => Sackville Lane

Gordon Drive => Gordon Drive

Samercy Place => Samercy Place

New Low Cost Housing => New Low Cost Housing

Mamba Ridge => Mamba Ridge

Pine Street => Pine Street

Wilkinson Road, Brinsley Johnson Drive => Wilkinson Road, Brinsley Johnson Drive

Wilkinson Road => Wilkinson Road







Spur Road => Spur Road

Spur Loop => Spur Loop

Priscilla Street => Priscilla Street

Circular Road => Circular Road

3. Data Overview

 nodes	1/5/2017 11:16 AM	Microsoft Excel Comma Separated Values File	67,490 KB
 nodes_tags	1/5/2017 11:16 AM	Microsoft Excel Comma Separated Values File	400 KB
 ways	1/5/2017 11:16 AM	Microsoft Excel Comma Separated Values File	10,294 KB
 ways_nodes	1/5/2017 11:16 AM	Microsoft Excel Comma Separated Values File	24,470 KB
 ways_tags	1/5/2017 11:16 AM	Microsoft Excel Comma Separated Values File	6,188 KB
 Freetown.osm	8/23/2016 8:48 AM	OSM File	176,022 KB

Number of Nodes

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

Number of nodes: 820215

Number of ways

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

Number of ways: 171156

Number of Unique users:

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

Number of unique users: 669

Top contributing Users

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

vichada|208691

RAytoun|40532

Russ|31740

Susan O|30330

CloCkWeRX|27690

latte_oz|24141

uliwanne|23801

acrossthesound1|22577

seichter|19835

cityeditor1000|18556

Numbers of users contributing only once

```
sqlite> SELECT COUNT(*) FROM (SELECT a.user, COUNT(*) as num FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) a GROUP BY a.user HAVING num=1) b;
```

Number of users contributing once: 46

4. Additional Data Exploration

Common amenities

```
sqlite> SELECT value, COUNT(*) as num FROM nodes_tags WHERE  
key='amenity' GROUP BY value ORDER BY num DESC LIMIT 10;
```

Output:

```
school|95  
place_of_worship|79  
drinking_water|48  
restaurant|36  
bank|32  
fuel|31  
hospital|24  
clinic|14  
police|12  
embassy|9
```

Biggest Religion

```
sqlite> 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;
```

Output: christian | 38

Popular Cuisine

```
sqlite> SELECT nodes_tags.value, COUNT(*) as num FROM nodes_tags  
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE  
value='restaurant') a ON nodes_tags.id=a.id WHERE  
nodes_tags.key='cuisine' GROUP BY nodes_tags.value ORDER BY num  
DESC;
```

Output:

chinese|1

indian|1

seafood|1

5. Conclusion

The database was created with a clean data by updating the street names, curbing abbreviations, extra spaces etc. I noticed that the contributing users are so many so my thought to be able to get clean data is :- there has to be a standard for inputting data, throw in some exception handlers that will send out error if a certain filed/ values does not meet data type specifications. This I think will curb abbreviations, extra space, typos and some character types also maybe a monitor system to check everybody's contribution regularly be put in place. Openstreet Map should set a regular updating system to get current information of cities.

Files:

Quiz/ : scripts completed in lesson Case Study OpenStreetMap

README.md : this file freetown.osm :

sample data of the OSM file

sample.py : extract sample data from the OSM file

tags.py : count multiple patterns in the tags

mapparser.py : find unique tags in the data

audit1.py : audit street, city and update their names

data.py : build CSV files from OSM and also parse, clean and shape data

schema.py: to enable import schema

database.py : create database of the CSV files

query.py : different queries about the database using SQL

report.pdf : pdf of this document