Project: OpenStreetMap Data Wrangling with SQL

Map Area: I chose Pune city as live here and I'll be able to relate quickly with the information in the dataset.

- **Location:** Pune, India
- OpenStreetMap
- https://overpass-api.de/api/map?bbox=73.7744,18.4676,73.9340,18.5732

1. Data Audit

Looking at osm/xml file, I found that it uses different types of tags. So, I parse the Pune dataset using ElementTree and count number of unique tags.

mapparser.py is used to count the number of unique tags:

- 'bounds': 1,
- 'member': 12827,
- 'meta': 1,
- 'nd': 576964,
- 'node': 481644,
- 'note': 1,
- 'osm': 1,
- 'relation': 975,
- 'tag': 113041,
- 'way': 89552

Patterns in the tags

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

'lower': 108075, for tags that contain only lower case letters and valid

'lower_colon': 4851, for valid tags with colon in their names

'problemchars': 1 for tags with problematic characters

'other': 114, for tags that don't fall into above 3 categories

2. Problems Encountered in the map

Street address inconsistency

One of the problems encountered with the dataset is the street name inconsistencies. Below is the old names corrected with more acceptable names.

- Lowercase

  road – Road

  aundh road => Aundh Road
- Misspelling

  Rd – Road

  MIT College Rd => MIT College Road
- Hindi or Marathi names

  Path – Road

  Marg – Road

  Sadhu Vaswani Path => Sadhu Vaswani Road

  Chhatrapati Rajaram Marg => Chhatrapati Rajaram Road

City Name inconsistency

Using audit.py, we update the names

- Lower case

  pune – Pune

  nagar - Nagar

  Gate no 9, marketyard,pune => Gate No 9, Marketyard,Pune

  Kalyani nagar => Kalyani Nagar

3. Data Overview

I have used data.py to export the dataset into different csv files.

Overview of the csv files and osm file used are as below.

Pune.osm	104,912KB
Nodes.csv	39,967 KB
nodes_tags.csv	356 KB
ways.csv	5,419 KB
ways_nodes.csv	14,142 KB
ways_tags.csv	3,433 KB

I used database.py to create sqlite database, read data from csv files and insert into the respective tables.

Used query.py to execute different select queries to the database (Pune.db) and read the data.

Number of nodes:

Query - > SELECT COUNT(*) FROM nodes

Output: 481644

Number of ways:

Query -> SELECT COUNT(*) FROM ways

Output: 89552

Number of Unique Users:

Query -> SELECT COUNT(DISTINCT(e.uid)) \

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

Output: 518

Top Contributing Users:

Query-> 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

Output:

Singleton	50932
harishk	30185
kranthikumar	28163
vamshiN	25934
chaitanya110	21431
sramesh	21199
sathishshetty	21057
pallavi1	20868
sampath reddy	20734
Apreethi	20213

Number of users contributing only once:

Query-> 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
```

Output: 141

4. Additional Data Exploration

Common Amenities

Query->
```
SELECT value, COUNT(*) as num \
   FROM nodes_tags \
   WHERE key="amenity" \
   GROUP BY value \
   ORDER BY num DESC \
   LIMIT 10
```

Output:

restaurant	194
bank	130
atm	104
place_of_worship	85
café	70
fast_food	58
hospital	32
school	27
police	25
pharmacy	22

Biggest Religion:

Query-> 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:

```
hindu           57
```

Popular Cuisines:

Query ->
```
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
```

Output:

```
indian         37
vegetarian     11
pizza           8
regional        7
international   4
chinese         3
italian         3
barbecue        2
burger          2
Multi-Cuisine   1
```

5. Conclusion

The data of Pune is of reasonable good quality but the typo errors and users input in native languages are observed. For requirement of this project, we have cleaned a reasonable amount of data. But, there are improvement needed in the dataset. The dataset must have additional information such as tourist attractions, popular places etc.

Additional Suggestions

Control typo errors:

- parser can be built which can parse every word entered by user.
- Apply some rules to input data which users can follow while entering data. This will help in restricting users input data in the native language.
- Develop bot or script for cleaning the data on frequent interval.

More Information:

- Users generally search map to see the common attractions in the city e.g. historical places etc. So the users must be motivated to provide this information as well in the map.

Files:

• Report.pdf	: Report of this project
• mappparser.py	: find unique tags in the data
• tags.py	: count multiple patterns in the tags
• audit.py	: audit street, city and update their names
• data.py	: build csv files from OSM and also parse, clean and shape data
• database.py	: create database of the csv files
• query.py	: different queries about the database using SQL
• Pune_sample.osm	: sample data of the osm file for submission

6. References

- https://www.openstreetmap.org/export#map=13/18.5204/73.8542
- https://gist.github.com/carlward/54ec1c91b62a5f911c42#file-sample_project-md
- https://github.com/pratyush19/Udacity-Data-Analyst-Nanodegree/blob/master/P3-OpenStreetMap-Wrangling-with-SQL
- https://www.tutorialspoint.com/sqlite/sqlite_python.htm
- https://stackoverflow.com