OpenStreetMap Data Wrangling with SQL

August 27, 2017

0.1 Map Area

Pune, Maharashtra, India

- https://www.openstreetmap.org/node/16174445
- https://mapzen.com/data/metro-extracts/metro/pune_india/

I have chosen this city because it is one of the places that I want to visit.

1 Problems Encountered in the Map

Below are the inconsistencies I found in the dataset.

- Abbreviations
  - Rd -> Road
- Spelling mistakes
  - raod -> Road
- Lower Case
  - road -> Road
- Multiple keys for pin codes
  - addr:postcode and postal_code
- Pin code had spaces
  - 411 004 -> 411004

I changed all the abbreviations to the full representations of the street types. I corrected the spelling mistakes and the lower case to upper case. I then removed the spaces from the pin codes. We have 6 digit pin codes here in India.

2 Preparing SQL Database

After fixing the prooblems encountered, the next step is to prepare the data to be inserted into a SQL database. To do so I converted the data from xml format to csv files. These csv files can be easily inserted into the tables.

Finally, I built the SQL database and created tables in this database and then inserted the data from the csv files. I used sqlite3 shell for this purpose.

3 Querying the database

3.1 File Sizes

- pune_india.osm: 293 MB
- nodes.csv: 114 MB
- nodes_tags.csv: 511 KB
- ways.csv: 15.9 MB
- ways_nodes.csv: 40.7 MB
- ways_tags.csv: 9.32 MB
- pune.db: 160 MB

3.2 Element Count

{’bounds’: 1, ’member’: 8026, ’nd’: 1699628, ’node’: 1416405, ’osm’: 1, ’relation’: 2173, ’tag’: 305753, ’way’: 270179}

3.3 Character count

{’lower’: 299033, ’lower_colon’: 6529, ’other’: 191, ’problemchars’: 0}

3.4 Number of nodes & ways

```
In [2]: import sqlite3
        import pprint
        conn = sqlite3.connect("pune.db")
        cursor = conn.cursor()

        cursor.execute("select count(id) from nodes;")
        print 'There are {} nodes in database.'.format(cursor.fetchall()[0][0])
        cursor.execute("select count(id) from ways;")
        print 'There are {} ways in database.'.format(cursor.fetchall()[0][0])
```

```
There are 1416405 nodes in database.
There are 270179 ways in database.
```

3.5 Number of unique users

```
In [3]: cursor.execute("select count(distinct(uid)) \
        from (select uid from nodes union select uid from ways);")
        print 'There are {} unique users in \
        database.'.format(cursor.fetchall()[0][0])
```

```
There are 673 unique users in database.
```

4 Additional Exploration

4.1 Top 5 pin codes

```
In [4]: cursor.execute("select value, count(*) as count \
        from nodes_tags \
        where key = 'postcode' \
        group by value \
        order by count desc limit 5;")
        pprint.pprint(cursor.fetchall())
```

```
[(u'411038', 34),
 (u'411004', 31),
 (u'411048', 28),
 (u'411021', 20),
 (u'411041', 19)]
```

4.2 Top 10 contributing users

```
In [5]: cursor.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;")
        pprint.pprint(cursor.fetchall())
```

```
[(u'singleton', 96751),
 (u'harishvarma', 60144),
 (u'jasvinderkaur', 57697),
 (u'sramesh', 57627),
 (u'praveeng', 56788),
 (u'shiva05', 51899),
 (u'anushapyata', 49530),
 (u'kranthikumar', 47445),
 (u'harishk', 43181),
 (u'saikumar', 40332)]
```

4.3 Number of users with a single entry

```
In [6]: cursor.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;")
        print 'There are {} users in database with only \
        one entry.'.format(cursor.fetchall()[0][0])
```

```
There are 180 users in database with only one entry.
```

4.4 Common amenities

```
In [7]: cursor.execute("SELECT value, COUNT(*) as num \
        FROM nodes_tags WHERE key='amenity' GROUP BY value \
        ORDER BY num DESC LIMIT 10;")
        pprint.pprint(cursor.fetchall())
```

```
[(u'restaurant', 235),
 (u'bank', 152),
 (u'atm', 129),
 (u'place_of_worship', 113),
 (u'cafe', 74),
 (u'fast_food', 69),
 (u'hospital', 45),
 (u'fuel', 43),
 (u'school', 41),
 (u'pharmacy', 30)]
```

4.5 Top 5 religions based on places of worship

```
In [8]: cursor.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 5;")
        pprint.pprint(cursor.fetchall())
```

```
[(u'hindu', 77), (u'muslim', 9), (u'christian', 4), (u'sikh', 1)]
```

4.6 Top 10 cuisines

```
In [9]: cursor.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;")
        pprint.pprint(cursor.fetchall())
```

```
[(u'indian', 46),
 (u'vegetarian', 13),
 (u'pizza', 10),
 (u'regional', 8),
 (u'international', 5),
 (u'barbecue', 3),
 (u'chinese', 3),
 (u'italian', 3),
 (u'burger', 2),
 (u'Multi-Cuisine', 1)]
```

5 Conclusion

In my analysis, the Pune OpenStreetMap data is in good quality but still contains some typos formatting errors. I have parsed this data to correct the street names and reformat the pin codes. But, there are still lots of inconsistencies in the dataset that arise due to human input. We have cleaned and modified the street names and pin codes to remove the inconsistencies. We then transformed the XML to CSV and inserted the data into SQL tables. We found some interesting information by querying the database.

5.1 Suggestions for improvement

- We need to have a check in place that would minimize the human error and typos by comparing the new additions with already existing keywords.
- We can attract more contributors by having a ranking system in place for improving the maps.

5.1.1 Benefits

- There will be lesser typos and a uniform format throughout the dataset.
- A ranking system will help in retaining high performers.

5.1.2 Anticipated problems

- The security check would need to be updated whenever new features are added to the dataset. The check for one field can not be used on all the other fields.
- Having a ranking system might create competition which can sometime be counter productive.