Project 3: OpenStreetMap Data Wrangling with SQL

Name: Sicong Chen

Map Area:

- Location: San Francisco, California
- OpenStreetMap URL (https://www.openstreetmap.org/relation/111968)
- MapZen URL (https://mapzen.com/data/metro-extracts/metro/san-francisco_california/)

I went to college at Berkeley and stayed aroud the bay area for a couple of years. San Francisco is the city I was familiar with and I miss it a lot. I am very interested in looking at the data of this city and hopefully contribute to it myself.

1. Tag Overview

Tags in the Data

First of all I want to know what tags are used in the dataset. `mapparser.py` is used to count the numbers of each unique tags.

- 'bounds': 1,
- 'member': 54957,
- 'nd': 7774168,
- 'node': 6559150,
- 'osm': 1,
- 'relation': 6227,
- 'tag': 2124121,
- 'way': 807513}`

  The OSM file is about 1.4G and has more than 19 million top level tags.

Patterns in the Tags

Using `tags.py`, I created 3 regular expressions for certain tags, and returns the number of each type.

The following reveals count of each tag categories.

- `"lower" : 1409185`, for tags that contain only lowercase letters and are valid,
- `"lower_colon" : 689009`, for otherwise valid tags with a colon in their names,
- `"problemchars" : 130`, for tags that contain problematic charactoers,
- `"other" : 25797`, for other tags that do not fall into the other three categories.

2. Problems Encountered in the Map

Abbreviated Street Names

One of the problems we encountered in the dataset is the street name inconsistencies. Here are some examples that we want to fix:

- **Abbreviations -> Corrected Names**
  - Woodside Plz -> Woodside Plaza
  - Tehama Ave -> Tehama Avenue
  - Peralta St-> Peralta Street
  - Redwood Hwy -> Redwood Hwy
  - California Dr -> California Drive
  - Newark Blvd -> Newark Boulevard
  - etc.

We create a mapping dictionary that includes all these abbreviations (i.e. 'Rd':'Road') and use 'audit.fix_street' to map abbreviated street names to full street names.

Long Post Code

Post code should be 5-digit long. We run the audit.audit_postcode to return a list of long post code and observe the following different formats of long post code:

- **Long Post Code**
    - 'CA 94544' - Include State
    - '941234' - Incorrect 6 digit zipcode
    - '94402-3025' - Zipcode with 4 digit area code

Phone Number Format Inconsistent

We also observed that there are several different formats for phone numbers recorded in the dataset. Here's some example by running audit.audit_phone:

- **Sample Phone Numbers**
    - +1 510 528 8888
    - (415) 550-5534
    - +1-510-524-7031
    - +1 4152529888
    - etc

To standardize the format, we want it all to be in XXX-XXX-XXXX or 1-XXX-XXX-XXXX. We run the audit.fix_phonenumber to fix to standardize the phone number formatting.

3. Data Overview

This part we first use the 'data.py' to create csv files that preparing for the dagabase. Then create the database and explore it with SQL queries.

File sizes:

- `san-francisco_california.osm: 1.4 G`
- `nodes_csv: 550 MB`
- `nodes_tags.csv: 9.5 MB`
- `ways_csv: 49.2 MB`
- `ways_nodes.csv: 186.9 MB`
- `ways_tags.csv: 62.6 MB`
- `sfosm.db: 746.3 MB`

Number of nodes:

```
sqlite> SELECT COUNT(*) FROM node
```

Output:

```
6559145
```

Number of ways:

```
sqlite> SELECT COUNT(*) FROM way
```

Output:

```
807514
```

Number of unique users:

```
sqlite> SELECT COUNT(DISTINCT(sub.uid))
FROM (SELECT uid FROM node UNION ALL SELECT uid FROM way) as sub;
```

Output:

```
2740
```

Top contributing users:

```
sqlite> SELECT sub.user, COUNT(*) as num
FROM (SELECT user FROM node UNION ALL SELECT user FROM way) as sub
GROUP BY sub.user
ORDER BY num DESC
LIMIT 10;
```

Output:

```
andygol|1497774
ediyes|888405
Luis36995|663476
dannykath|540943
RichRico|404889
Rub21|383614
calfarome|186305
oldtopos|166631
KindredCoda|151266
karitotp|135711
```

The top contributing user contributed to more than a million nodes!!

Number of users contributing only once:

```
sqlite> SELECT COUNT(*)
FROM
    (SELECT sub.user, COUNT(*) as num
     FROM (SELECT user FROM node UNION ALL SELECT user FROM way) as sub
     GROUP BY sub.user
     HAVING num=1) as subsub;
```

Output:

```
682
```

4. Additional Data Exploration

Common ammenities:

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

Output:

```
restaurant|2925
bench|1183
cafe|989
place_of_worship|694
post_box|685
school|583
fast_food|580
bicycle_parking|566
drinking_water|519
toilets|410
```

The most common amenenty is restraunt - there are about three thousand restaurants in the area!

Religions:

```
sqlite> SELECT node_tags.value, COUNT(*) as num
FROM node_tags
    JOIN (SELECT DISTINCT(id) FROM node_tags WHERE value='place_of_worship') as a
ON node_tags.id=a.id
WHERE node_tags.key='religion'
GROUP BY node_tags.value
ORDER BY num DESC
LIMIT 5;
```

Output:

```
christian|632
buddhist|15
jewish|10
muslim|4
unitarian_universalist|2
```

Popular cuisines

```
sqlite> SELECT node_tags.value, COUNT(*) as num
FROM node_tags
    JOIN (SELECT DISTINCT(id) FROM node_tags WHERE value='restaurant') as sub
    ON node_tags.id=sub.id
WHERE node_tags.key='cuisine'
GROUP BY node_tags.value
ORDER BY num DESC LIMIT 10;
```

Output:

```
mexican|194
chinese|164
pizza|151
japanese|141
italian|128
thai|107
american|98
vietnamese|70
indian|58
sushi|56
```

5. Conclusion

The San Francisco dataset is a relative big one with more than 19 million top level tags. Through the data cleaning process we can see that the biggest problem is the inconsistency of information format. We standardized the address and phone number format by eliminatiing abbreviations and lowercased addresses, and reformatting phone numbers. And by putting these information into database, we can get much information including amenities, religions, popular cuisines of the area, and how much users have contribute to the dataset. There is certainly more to be discover. San Francisco is a city that I am familiar with. Next time I want to try look at a compelet strange city and try understanding it beginning from the its data.

Additional Suggestion and Ideas

The cleaning process I conducted on this dataset is far from enough. There are still a lot inconsistencies including address, post codes, city names and etc. If there is a standardized way for us to record each type of information, the information itself will be much more efficient. For example, if we import data from Google Map/ Yelp, as the data format is standardized on these third party apps, the imported data maybe better formatted and more complete. By implementing this method, some benefits as well as problems may occur:

- Benefits:
  - Better formatted information - As there is a more standardized way of address, phonumber information on google map
  - Faster updates - For example, if a restaurant moved and thus need to update its address, it is more likely their first place to update thier information will be yelp. So if we pull information directly from yelp, it will be more efficient.
- Problems:
  - Conflicted information - Import information from various sources may result in conflicts. The name/address for the same location may vary due to how it was recorded on different database.
  - Possible costs - Including other parties to improve the database may add extra cost.
  - Less motivation for contributers - The more standardized formatting means stricker rules for users to contribute extra information. This may make users less motivated to contribute.

Files

- `README.md` : this file
- `sample.osm`: sample data of the OSM file
- `audit.py` : audit street, city and update their names
- `data.py` : parse the data and build 5 seperate CSV files from OSM
- `mapparser.py` : find unique tags in the data
- `report.pdf` : pdf of this document
- `sample.py` : extract sample data from the OSM file
- `tags.py` : count number of top level tags

Reference

- https://discussions.udacity.com/t/validation-error-osm/247882/4 (https://discussions.udacity.com/t/validation-error-osm/247882/4)
- https://classroom.udacity.com/nanodegrees/nd002/parts/0021345404/modules/316820862075463/lessons/3168208620239847/concepts/77135319070923 (https://classroom.udacity.com/nanodegrees/nd002/parts/0021345404/modules/316820862075463/lessons/3168208620239847/concepts/77135319070923)
- https://gist.github.com/carlward/54ec1c91b62a5f911c42#file-sample_project-md (https://gist.github.com/carlward/54ec1c91b62a5f911c42#file-sample_project-md)