OpenStreetMap Data Case Study - SQL

Deepashree Jayaramu

Map Area

San Diego, California, USA

- https://mapzen.com/data/metro-extracts/metro/san-diego_california/

San Diego is the town I am currently living from past 8 years and it is become one of my favorite cities after my home town back in India. I have fallen in love with the weather and diversified ethnic community making it the city of my choice for this project as I can get to know more of its amenities and contribute to any improvements that might require on the Open Street Map data.

Problems Encountered in the Map

On auditing the data, it was found that there are inconsistencies in the way the street names are represented. Few examples of incorrect or non-standard formats are given below.

'101': set(['Murphy Canyon Road #101']),

'102': set(['Avocado Ave #102', 'Dewey Rd #102']),

'103': set(['Mission Boulevard #103']),

'104': set(['Cuyamaca St #104']),

'105': set(['Camino Del Rio S #105']),

'1500': set(['B Street Suite 1500']),

'3604': set(['Admiral Baker Rd #3604']),

'67': set(['Highway 67']),

'75': set(['Highway 75']),

'92131': set(['92131']),

'94': set(['Highway 94']),

There were many ‘Spanish’ street names starting with the street names as the first word of the complete name. Ex: ‘Avenido’ Venusto, ‘Calle’ Cabrillo, ‘Via’ Oceania, etc..

The code in the project captures and audits only the last part of the word in ‘street_name’. It was a challenge to address this issue while auditing.The following code be useful in auditing and identifying any wrong names in street names for cleaning them later.

```
street_type_re = re.compile(r’\b\S+\.?$’, re.IGNORECASE)

street_first_re = re.compile(r’^\S+\b’, re.IGNORECASE)

street_name = ‘street name’

m = street_type_re.search(street_name)

print m.group()

m = street_first_re.search(street_name)

print m.group()
```

Similarly, there were post codes that were in non-standard format as shown below.

The post codes were removed from data as there were unknown or lack of appropriate data.

Post codes:

```
'9839': set(['9839']), (4 digits)

'CA 91914': set(['CA 91914']),

'CA 92104': set(['CA 92104']),

'Scripps Ranch Blvd.': set(['Scripps Ranch Blvd.'])}
```

Overview of the data:

san-diego_california.osm	299 MB
OSMSD.db	85.5 MB
Sample.osm	30.3 MB
nodes.csv	26.7 MB
nodes_tags.csv	27.7 MB
ways.csv	5.84 MB
ways_tags.csv	10.2 MB
ways_nodes.csv	19.9 MB

Number of nodes

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

Output: 362698

Number of ways

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

Output: 100487

Number of unique users

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

Output: 1076 (Unique users)

Top 10 contributing users

```
sqlite> 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:

```
n76|331315
TheDutchMan13|25059
Sat|17574
Adam Geitgey|15156
Zian Choy|5359
Michael Verrier|4534
bdiscoe|3404
Brian@Brea|2543
```

```
ridixcr|1790
stevea|1590
```

Number of users appearing only once:

```
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:288 (users had only one post)

Additional data exploration and Improvements:

- Top 10 cuisines in San Diego.

```
Sqlite> SELECT value, COUNT(*) as num FROM nodes_tags WHERE key = 'cuisine'
GROUP BY value ORDER BY num DESC limit 10;
```

Output:

```
burger|3
pizza|3
coffee_shop|2
India|1
bagel|1
burger;breakfast|1
cafe/diner|1
chicken|1
dessert|1
ice_cream|1
```

Clearly, these values for the key, Cuisine needs cleaning as they are not the type of cuisines (except for one Value, India). They are more like Restaurant types.

- Top 10 values for Highways.

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

Output:

```
turning_circle|3893

traffic_signals|2110

crossing|1946

stop|867

bus_stop|123

turning_loop|31

street_lamp|28

motorway_junction|13

give_way|8

elevator|3
```

We see that the data is messy as we have 3 Elevators as values for the Key, Highway giving scope for cleaning.

Conclusion

The dataset is very messy in San Diego area of Open Street Map. Spanish names of streets and the non-standard format of the post codes were a challenge to audit and clean. There is a lot of scope for cleaning and improving the dataset.

Additional Suggestion and Ideas:

1. Having a standard formatted template to allow users to update the data can reduce the data inconsistency.
2. Data can be compared and updated to match with Google Maps by using similar APIs like their Geocoding and GeoDirection APIs.

By using such APIs, we can get many benefits like 3D view, hybrid maps and better zoom on OSM.
However, Google Map APIs are expensive when it comes to unlimited usage.
Open Street Map data are more customizable and well controlled.

References

- Forum links answered by @Myles , Myles Callen. Almost all the help that I could take right from creating the sample file of the map area to converting the .CSVs into data base tables.
- Live Help Chat room mentors had been extremely patient in answering my doubts and directing me appropriately to understand the project
- Used Github link to create the Project report: https://github.com/pratyush19/Udacity-Data-Analyst-Nanodegree/tree/master/P3-OpenStreetMap-Wrangling-with-SQL

Files

- Project Report OSM-SQL.pdf : Report of this project including the answers to Rubric questions, references, Map of the chosen region from Open Street Map website and file lists that are part of submission.
- data.py : build CSV files from OSM and also parse, clean and shape data.
- audit.py : audit street, post codes and update their names.
- Python codes : Jupiter notebook containing the python codes for:
    1. creating the sample OSM file
    2. parsing and identifying the unique tags
    3. list of Key values
- SQL : Jupiter notebook with codes for creating the databse tables from the .CSV files.
- Sample OSM file