OpenStreetMap Data Case Study

Map Area

San-Jose, CA, USA

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

San Jose is the largest city within the Bay Area and earns the nickname "Capital of Silicon Valley". Many technology company headquarters are located in this area. I'm interested in finding unique features of the city map.

Assessing Quality of Map Data

Before analyzing the whole set of the database, I'd like to take a small sample of the datafile to assess the quality of the data. After processing the sample data, I found that the overall quality of dataset is fair, and in order to improve consistency and accuracy, some problematic data needs to be cleaned. Data quality problems are:

- Inconsistent postal codes format(CA 94035, 95014-1657, 95070)
- Inconsistent street type name, street name abbreviation (Rd, Road; Cir,Circle)
- Tags in second level of 'way' tags, data from Tiger GPS is divided into different categories for searching usage. (county, name_base, name_type)

```
        <tag k="name" v="Lafayette Street"/>
        <tag k="oneway" v="yes"/>
        <tag k="highway" v="primary"/>
        <tag k="maxspeed" v="40 mph"/>
        <tag k="tiger:county" v="Santa Clara, CA"/>
        <tag k="tiger:name_base" v="Lafayette"/>
        <tag k="tiger:name_type" v="St"/>
```

Data Cleaning

Before outputting the OSM XML data into CSV files, perform the data cleaning first to improve data quality. Data cleaning includes two steps: update postal code and update street name.

Update Postal Code

In the original datafile, postal code formats are different, some entries have state abbreviation heading, many entries have four-digit code extension. We'll standardize the data into five-digit format first.

```
postcode_re = re.compile(r'[0-9]{5}')

def update_postcode(postcode):
    m = postcode_re.search(postcode)
    try:
        postcode = m.group()
    except:
        pass
    return postcode
```

Update Street Name

Use regular expression to find street type first, if it is in abbreviation, change it into full street type. For Example: 'Blvd' is updated to 'Boulevard'.

```
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)
def update_name(name, mapping):
    m = street_type_re.search(name)
    better_name = name
    if m:
        if m.group() in mapping.keys():
            better_street_type = mapping[m.group()]
            better_name = street_type_re.sub(better_street_type, name)
    return better_name
    return name
```

After clean up the problematic data, the OSM XML file will be parsed into five CSV files follow the schema from schema.py. Then the data will be imported into SQLite3 database.

Statistical Overview of Dataset

This part contains basic statistics about the database, such as file size, number of nodes, ways, etc.

File Sizes

File Name	File Size
san-jose_california.osm	284.8 MB
san_jose_california.db	153 MB
nodes.csv	107.2 MB
nodes_tags.csv	2.6 MB
ways.csv	10.1 MB
ways_nodes.csv	35.5 MB
ways_tags.csv	21.3 MB

Number of Nodes

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

1285767

Number of Ways

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

171036

Number of Unique Users

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

1245

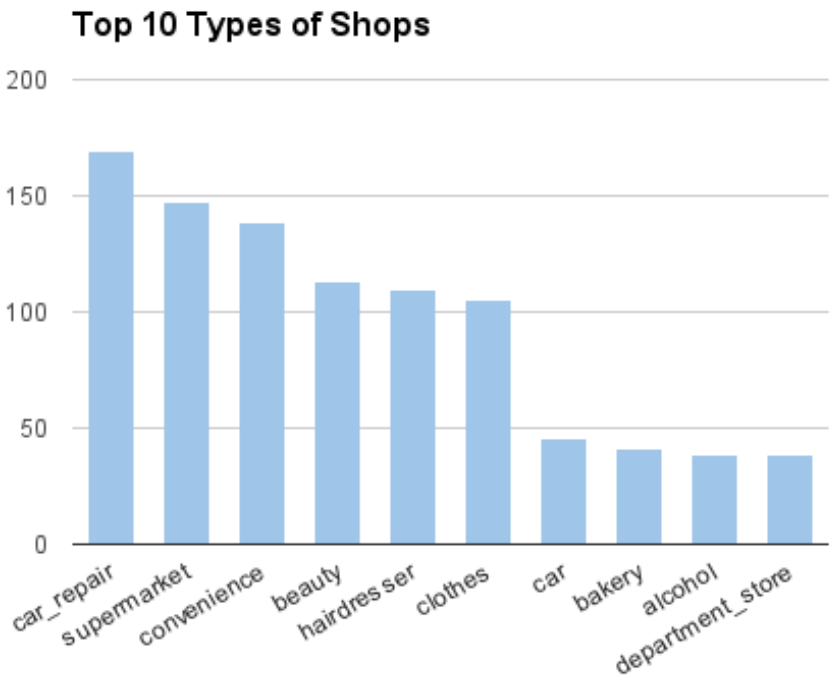
Number of Shops

```
SELECT COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key = 'shop';
```

Top 10 Types of Shops

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key = 'shop'
GROUP BY tags.value
ORDER BY count DESC
LIMIT 10;
```

Type of Shop	Count
car_repair	169
supermarket	147
convenience	139
beauty	113
hairdresser	110
clothes	105
car	46
bakery	41
alcohol	39
department_store	39



City Map Exploration

Some interesting findings have been discovered in this section.

Top 10 Cities by Count

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key = 'city'
GROUP BY tags.value
ORDER BY count DESC
LIMIT 10;
```

City	Count
Sunnyvale	3396
San Jose	733
Morgan Hill	373
Santa Clara	301
Saratoga	231
San José	115
Milpitas	94
Cupertino	56
Los Gatos	49
Campbell	47

Top 10 Postal Code Area by Count

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags
      UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key='postcode'
GROUP BY tags.value
ORDER BY count DESC
LIMIT 10;
```

Postal Code	Count
95014	9920
95037	386
95070	236
94087	211
94086	201
95051	164
95054	98
95125	92
95127	90
95035	89

Postal code 95014 refers to city of Cupertino, as city of Cupertino has appeared only 56 times under tag of city, so many tags are lack of city value. Similarly, for city of Sunnyvale, postal code ranges from 94085 to 94089, while under tag of city the appearance number is 3396. This is due to the incompleteness of data.

Top 10 Appearing Amenities

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key = 'amenity'
GROUP BY tags.value
ORDER BY count DESC
LIMIT 10;
```

Amenity	Count
parking	1808
restaurant	939
school	537
fast_food	475
place_of_worship	343
cafe	238
fuel	233
bench	198
toilets	183
bicycle_parking	182

Popular Fast Food

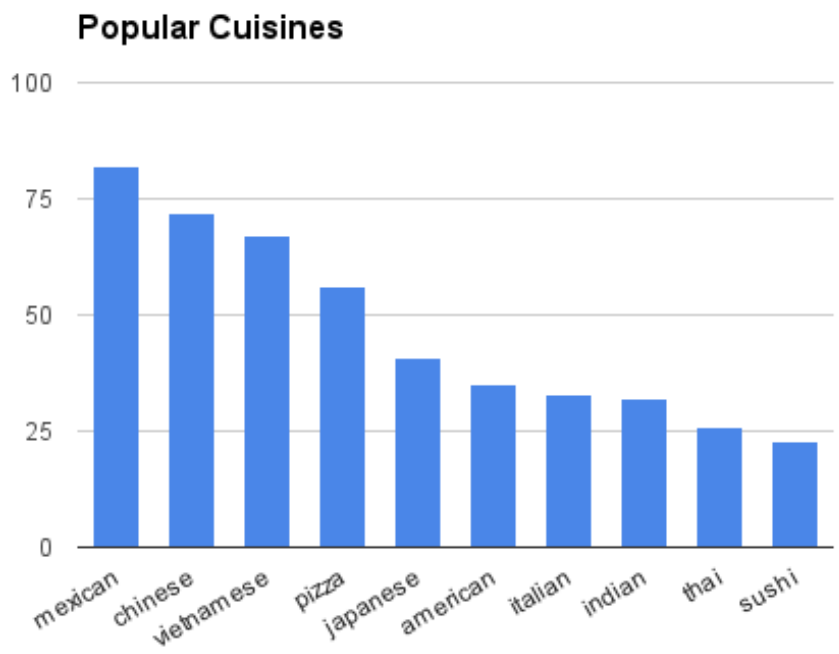
```
SELECT tags.value, COUNT(*) as num
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags JOIN
      (SELECT DISTINCT(id) FROM nodes_tags
      WHERE value='fast_food' UNION ALL
      SELECT  DISTINCT(id) FROM ways_tags
      WHERE value='fast_food') i
    ON tags.id=i.id
WHERE tags.key='cuisine'
GROUP BY tags.value
ORDER BY num DESC
LIMIT 10;
```

Cuisine	Count
burger	85
sandwich	70
mexican	43
pizza	29
ice_cream	23
chicken	16
vietnamese	16
chinese	12
american	7
indian	6

Popular Cuisines

```
SELECT tags.value, COUNT(*) as num
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags JOIN
      (SELECT DISTINCT(id) FROM nodes_tags
      WHERE value='restaurant' UNION ALL
      SELECT  DISTINCT(id) FROM ways_tags
      WHERE value='restaurant') i
    ON tags.id=i.id
WHERE tags.key='cuisine'
GROUP BY tags.value
ORDER BY num DESC
LIMIT 10;
```

Cuisine	Count
mexican	82
chinese	72
vietnamese	67
pizza	56
japanese	41
american	35
italian	33
indian	32
thai	26
sushi	23



Additional Ideas

The above results have revealed the incompleteness of data, for example, some road tags may only have postal code entries, or may only have city data without postal code. So, based on the given postal code data, it is possible to update the street city values.

```
Postal Code & City Mappings:

{ '95014':'Cupertino',
  '95037':'Morgan Hill',
  '95070':'Saratoga'
  ...
}
```

Benefits:

- As we have more data about city, we can implement further analyze for particular city, for example, the distribution of different types of shops or amenities in Sunnyvale.
- We can also calculate the distribution of certain types of shop or amenities in different cities. For example, how is parking place distributed in different area. Or which area has the most number of vietnamese restaurants.

Anticipated Problems

- Although we can update the city value based on the postal code entries, there still exists many tags that lack **both** city value and postal code.
- Mappings from postal code to city can be easily obtained, however, as cities may have more than one postal code, it is hard to do in the reverse way.

Conclusion

Through analysis of dataset, it is clear that postal code and city data is not consistent, which is due to the incompleteness of the map data. We can update the city tag by its postal code value. And other interesting study topics are: cuisine popularity in different cities, relationship between number of parking place and types of street, etc.