OpenStreetMap Data Case Study

Map Area

San Francisco, CA, United States

https://www.openstreetmap.org/relation/111968#map=11/37.8117/-122.3812
(https://www.openstreetmap.org/relation/111968#map=11/37.8117/-122.3812)

San Francisco has been my home for the past 5 years and I was interested to see what the Open Street Map data would reveal about the city. I used the existing Open Street Map Metro Extract for the city of San Francisco (not the San Francisco Bay Area, which is a separate extract).

XML Data Overview

This data set is very large so I used the 'get_element' function provided in the course materials to subsample the data. My script including this function is called `"create_subset.py'`. I performed my initial analyses with a 10% subset of the data.

Stats for Full XML Data:

```
file size................................ 1.41 GB
number of unique contributors ............. 2858
number of nodes........................... 6626853
number of relations.......................7615
number of tags............................2056740
number of ways............................826366
```

Stats for Subset of XML Data (10%):

```
file size................................ 142.8 MB
number of unique contributors ............. 1658
number of nodes........................... 662686
number of relations.......................762
number of tags............................205382
number of ways............................82636
```

XML Data Audit: Developing an 'Approved' List of Street Types

I chose to audit the street_type data for clarity and consistency. I ran the data against a script (`'determine_street_types.py'`) to generate a dictionary recording the number of occurances of each instance of street type. I wanted to extract the 'street_type' data as a dictionary rather than a list or set so that I could examine the number of times that potentially unusual street_types appeared. This was essential for appropriately developing an 'approved' list.

Based on this dictionary, I created a comprehensive list of 'approved' street types. I also created a dictionary matching common street type mistakes to the appropriate 'approved' street type (see below).

Systematic Inconsistencies in the Data

Examining the `'all_street_type_instances'` dictionary (not included) reveals that expected street types like 'Avenue' and 'Street' are very common. There are some frequently occurring inconsistencies (capitalization, abbreviations, and terminal '.') that can easily be corrected to produce a consistent data set. Most street type names are of the format 'Avenue' and 'Street', so I elected to format inconsistent street_types according to this convention. The incorrect to correct formats are mapped in the `'type_mapping_dict'` dictionary.

Non-Systematic Inconsistencies in the Data

I noticed a few unexpected 'street_types'. For example, there is one instance of street_type = 'Telegraph', when it is in fact a street name and not a type. I searched the data and found and element that looked like this:

```
{'k': 'addr:street', 'v': '3605 Telegraph'}
```

This element is formatted incorrectly with the addr:housenumber value included in the addr:street.

Similarly, there are three instances of:

```
{'k': 'addr:street', 'v': 'Fort Mason'}
```

This element refers to 'Fort Mason', and is incorrectly pulling 'Mason' as a street type.

These errors are non-systematic and can't be dealt with programmatically. These elements are rare, and I will deal omit them from my data by simply not including them in my approved formats list.

Unique Street_types Exceptions: 'Alameda' and 'Broadway'

I noticed a high number of street_types 'Alameda' in the subset data. Most such elements had the structures:

```
{'k': 'addr:street', 'v': 'Alameda'}
{'k': 'addr:street', 'v': 'The Alameda'}
```

In Spanish the word 'Alameda' means 'promenade', so it's possible that in this context 'Alameda' is a valid street_type. Additionally, I also noticed that the Metro Extract selection of San Francisco includes a small industrial area of Alameda island (which is within the limits of the City and County of San Francisco). Perhaps this data is labeled as 'Alameda' for street types. In either event, I elected to add 'Alameda' to the approved street_type list to retain this data for further exploration.

I also noticed a large instance of 'Broadway' as street_types in the subset data. It is common to usage colloqially to refer to a 'Broadway Street' as simply 'Broadway', and I elected to add 'Broadway' as a stand alone street_type in my approved list. These elements had the following structure:

```
{'k': 'addr:street', 'v': 'Broadway'}
```

List of Approved Street Types and Dictionary to Standardize Formats

Based on the above criteria, I created an 'approved_list' of street_type formats and a type_mapping_dict to map correct formats to incorrect formats.

```
approved_list =['Boulevard','Court','Bridgeway','Way','Circle','Alameda','Highway','Real','Embarcadero','P
ath','Lane','Center','Plaza','Drive','Place','Parkway','Gardens','Road','Square','Alley','Walk','Stree
t','Terrace','Broadway','Avenue']

type_mapping_dict = {'Plz.': 'Plaza', 'boulevard': 'Boulevard', 'Ave.': 'Avenue', 'avenue': 'Avenue', 'S
t.':'Street', 'plaza': 'Plaza', 'Plz': 'Plaza', 'drive': 'Drive', 'Rd': 'Road', 'street': 'Street', 'roa
d': 'Road', 'Dr.': 'Drive', 'Blvd': 'Boulevard', 'Ave': 'Avenue', 'St': 'Street', 'Dr': 'Drive', 'Rd.': 'R
oad', 'Blvd.': 'Boulevard'}
```

XML Data Audit: Other Parameters and Observations

I also audited the postal code data for consistency using the `'audit_post_codes.py'` script. This script takes in an OSM file, parses by XML tag, and saves unique postal code instances to a set. I elected to store the data as a set (rather than an enumerated dictionary) so that I could visually scan the data for obvious errors. If I had seen abnormalities I could enumerate the instances to investigate, but this turned out to be not necessary. The data was consistently formatted and all data looked reasonable (all codes began with '9' as expected and were 5 digits).

I audited street names in the data to make sure capitalizations of names was correct. This data was well formatted, but I did notice that many of the street names aren't streets in San Francisco, but rather thoroughfares across the bay in Berkley and Oakland (examples are Shattuck Avenue, Telegraph Avenue, and University Street). This observation is beyond the scope of this audit since I downloaded the data as an existing Metro Extract, but its something to explore when the data is in the SQL database.

Parsing and Cleaning the XML Data

After auditing the XML data I wrote a 'cleaning' function for modifying non-compliant street_types. The `'clean_street_name'` function lives as a standalone function in `'clean_street_name.py'` and within the `'XML_cleaning_write_to_csvs.py'` script. This is a modified version of the process_mapping scrip developed in the case study, and it cleans the street_type data, shapes the XML elements to python dictionaries, and saves the python data to CSV files corresponding to the desired data base tables. I ran the `'XML_cleaning_write_to_csvs.py'` script on the entire San Francisco OSM file (not the subset) to create CSVs for each table.

Stats for Cleaned Data:

```
san_francisco_california.osm ......... 1.41 GB
san_francisco_OSM_full.db ........... 768.9 MB
nodes.csv ............. 555.9 MB
nodes_tags.csv ........ 9.7 MB
ways.csv .............. 50.6 MB
ways_tags.csv ......... 60 MB
ways_nodes.cv ......... 189.2 MB
```

SQL Input and Data Overview

I loaded the CSV tables into the SQLite database editor DB Browser to build my SQL data base. I used DB Browser to run initial exploratory analyses but I elected to re-run analyses in my Jupyter Notebook. I also elected to load the SQL queries into pandas data frames to easily visualize analysis results. Below is a statistical overview of the data as defined in the specifications for this project as well as my own exploratory analysis.

In [17]:
```
import sqlite3
import pandas as pd
conn = sqlite3.connect('/Users/jeff/Desktop/Udacity_2017/DAND_Lecture4/L4_OSM_Project/san_francisco_OSM_full.db')
```

Nodes, Ways, and Unique Users

In [2]:
```
# The number of nodes in the San Francisco data set
print 'Number of Nodes'
pd.read_sql_query("""
SELECT COUNT(*) as nodes FROM nodes;
""", conn)
```

```
Number of Nodes
```

Out[2]:

	nodes
0	6626853

In [3]:
```
# The number of ways in the San Francisco data set
print 'Number of Ways'
pd.read_sql_query("""
SELECT COUNT(*) as ways FROM ways;
""", conn)
```

```
Number of Ways
```

Out[3]:

	ways
0	826366

In [4]:
```
# The number of distinct users in the San Francisco data set
print 'Number of Distinct Users'
pd.read_sql_query("""
SELECT COUNT(DISTINCT(users.uid)) as users
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) users;
""", conn)
```

```
Number of Distinct Users
```

Out[4]:

	users
0	2832

Using SQL to Explore Audit Findings

I took a closer look at some of the anomalous data identified during the XML audit.

Street Names Outside San Francisco Limits

First, I wanted to look at the 'cities' data. I recognized that a lot of the street names were not actually San Francisco streets, and I wanted to know more about the cities listed for the data. To my surprise, the San Francisco Metro Extract data includes many nodes for other Bay Area cities. I'm not sure if this is due to how the extract is generated or the actual mapping itself (latitude and longitude), but this is certainly an unexpected finding and suggests an aspect for improving the current state of the Open Street Map data.

In [5]:
```
pd.read_sql_query("""
SELECT nodes_tags.value as city, COUNT(*) as node_occurances
FROM nodes_tags
WHERE nodes_tags.key='city'
GROUP BY nodes_tags.value
ORDER BY node_occurances DESC
LIMIT 10;
""", conn)
```

Out[5]:

	city	node_occurances
0	San Francisco	9173
1	Berkeley	3523
2	Redwood City	3047
3	Oakland	369
4	Richmond	240
5	Union City	127
6	Palo Alto	97
7	El Cerrito	89
8	San Mateo	73
9	Pacifica	64

Occurance of 'Alameda' as a Street Type

I wanted to know if the incidence of 'Alameda' street types in the data had anything to do with the fact that part of Alameda Island is within the San Francisco City limits. This is an industrial area and I thought perhaps street types and names were poorly documented as simply 'Alameda'. I queried the cities where 'Alameda' street types occur and enumerated occurances. I found that the bulk of these incidents occur in Berkeley and Redwood City, not San Francisco, so my hypothesis is incorrect.

In [6]:
```
## I wanted to look into the spurious 'Alameda' tags.
pd.read_sql_query("""
SELECT nodes_tags.value as street_name, city.value as city, count(*) as occurances
FROM nodes_tags
JOIN (SELECT id, value FROM nodes_tags WHERE key='city') city
    ON nodes_tags.id=city.id
WHERE (nodes_tags.key = 'street' and nodes_tags.value LIKE '%Alameda%')
GROUP BY 1,2
ORDER BY occurances DESC

;
""", conn)
```

Out[6]:

	street_name	city	occurances
0	The Alameda	Berkeley	35
1	Alameda	Redwood City	12
2	Alameda Street	San Francisco	2
3	Alameda Avenue	Alameda	1
4	Alameda Avenue	Oakland	1

Occurrence of 'Broadway' as a Street Type

I was also curious about where the 'Broadway' street types were occurring. I ran the same query to pull out streets containing the word 'Broadway' in either the street name or street type. There are occurrences of 'Broadway', 'Broadway Street', and 'Broadway Avenue' in Redwood City. This is most likely an inconsistency in how people are documenting the name for a single street, and the data could be improved by addressing these issues.

In [7]:
```
## I wanted to look into the occurance of 'Broadway' as a street_type.
pd.read_sql_query("""
SELECT nodes_tags.value as street_name, city.value as city, count(*) as occurances
FROM nodes_tags
JOIN (SELECT id, value FROM nodes_tags WHERE key='city') city
    ON nodes_tags.id=city.id
WHERE (nodes_tags.key = 'street' and nodes_tags.value LIKE '%Broadway%')
GROUP BY 1,2
ORDER BY occurances DESC
;
""", conn)
```

Out[7]:

	street_name	city	occurances
0	Broadway	Redwood City	172
1	Broadway	Oakland	33
2	Broadway Street	Redwood City	23
3	Broadway	San Francisco	17
4	Broadway	Milbrae	6
5	Broadway	Burlingame	4
6	Broadway Street	San Francisco	2
7	North Broadway	Walnut Creek	2
8	Broadway Avenue	Redwood City	1

Further Explorations in SQL:

Sidewalks, Starbucks, and Leisure

Sidewalks

I love walking in San Francisco but sometimes sidewalks are inadequate. I noticed that 'sidewalk' is a key of the way_tags data, so I wanted to explore this. I enumerated the number of counts for a given way 'name' (usually a street) in which the way_tag.key was 'sidewalk'. While not a perfect estimate of sidewalk coverage, I used this approach to estimate the number of times sidewalks were tagged for a given street. I found that San Pablo Avenue (in Oakland) and Geary Boulevard (in San Francisco) had the most occurrences of a 'sidewalk' tag. This makes sense since both streets are very long thoroughfares and would likely include lots of different sidewalk 'types' (right, left, both, etc.) that would be recorded in the data.

In [18]:
```
# How many side_walk tags occur on a given way street name?

pd.read_sql_query("""
SELECT ways_tags.value as street, COUNT(*) as sidewalks
FROM ways_tags
JOIN (SELECT DISTINCT(id) FROM ways_tags WHERE key='sidewalk') i
    ON ways_tags.id=i.id
WHERE ways_tags.key = 'name'
GROUP BY street
ORDER BY sidewalks  DESC
LIMIT 5;
""", conn)
```

Out[18]:

	street	sidewalks
0	San Pablo Avenue	402
1	Geary Boulevard	217
2	Broadway	191
3	Market Street	185
4	International Boulevard	131

Starbucks

I noticed that the 'name' key in the nodes_tag table is associated with a business/place name. I enumerated the number of times each business/place name occured in the data. That's a lot of Starbucks and Subways!

In [9]:
```
# Enumerating the number of times a given busines name appears in the
# nodes_tags data
pd.read_sql_query("""
SELECT nodes_tags.value, count(*) as num
FROM nodes_tags
WHERE nodes_tags.key = 'name'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 10;
""", conn)
```

Out[9]:

	value	num
0	Starbucks	129
1	Subway	74
2	Wells Fargo	68
3	Walgreens	65
4	Bank of America	64
5	Chase	45
6	Shell	40
7	76	38
8	7-Eleven	34
9	Chevron	34

Leisure

Next I looked at the business/place names that are also associated with the key 'leisure'. Evidently, the working out is a 'leisure' activity.

In [19]:
```
# Enumerating business/place names associated with the key 'leisure'
pd.read_sql_query("""
SELECT nodes_tags.value, COUNT(*) as number
FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE key='leisure') i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key = 'name'
GROUP BY nodes_tags.value
ORDER BY number DESC
LIMIT 10;
""", conn)
```

Out[19]:

	value	number
0	24 Hour Fitness	5
1	29th & Diamond Open Space	2
2	Alvarado Park	2
3	Burn	2
4	Creekside Park	2
5	Fit Club Fitness	2
6	Fitness SF	2
7	Gym	2
8	Harding Park	2
9	Japanese Gardens	2

Visualizing SQL Queries in Pandas Data Frames

I saved my SQL queries as pandas data frames so I could visualize the data. I wanted to visualize three different elements of the San Francisco Metro Extract data: 1) most frequent cities in the nodes_tags data (also discussed above), 2) the most frequent amenity types, and 3) the most frequent fast food types. I love Mexican fast food, so the last query was close to my heart.

Cities in the San Francisco Metro Extract

As I had previously identified, San Francisco is the most freqent city in the data, followed by Berkeley and Redwood City. Visualization highlights how much of the data is not tagged as San Francisco and highlights that this is an area for improvement in the data quality.

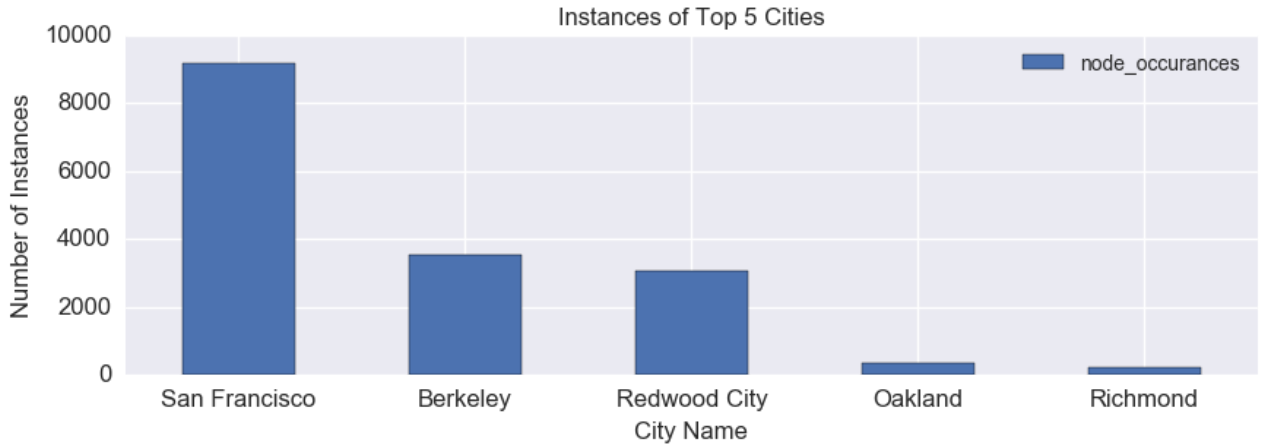
In [11]:
```
# Data frame from the SQL query enumerating 5 most frequent cities
top_5_cities_df = pd.read_sql_query("""
SELECT nodes_tags.value as city, COUNT(*) as node_occurances
FROM nodes_tags
WHERE nodes_tags.key='city'
GROUP BY nodes_tags.value
ORDER BY node_occurances DESC
LIMIT 5;
""", conn)
```

In [12]:
```
# Plotting the top 5 cities by instance number
import matplotlib.pyplot as plt
import seaborn as sns
%pylab inline

ax = top_5_cities_df.set_index('city').plot.bar(rot=0, title='Instances of Top 5 Cities', figsize=
(10,3), fontsize=12)
ax.set_xlabel("City Name",fontsize=12)
ax.set_ylabel("Number of Instances",fontsize=12)
plt.show()
```

```
/Users/jeff/anaconda/envs/DAND/lib/python2.7/site-packages/matplotlib/font_manager.py:273: UserWarni
ng: Matplotlib is building the font cache using fc-list. This may take a moment.
  warnings.warn('Matplotlib is building the font cache using fc-list. This may take a moment.')
```

```
Populating the interactive namespace from numpy and matplotlib
```

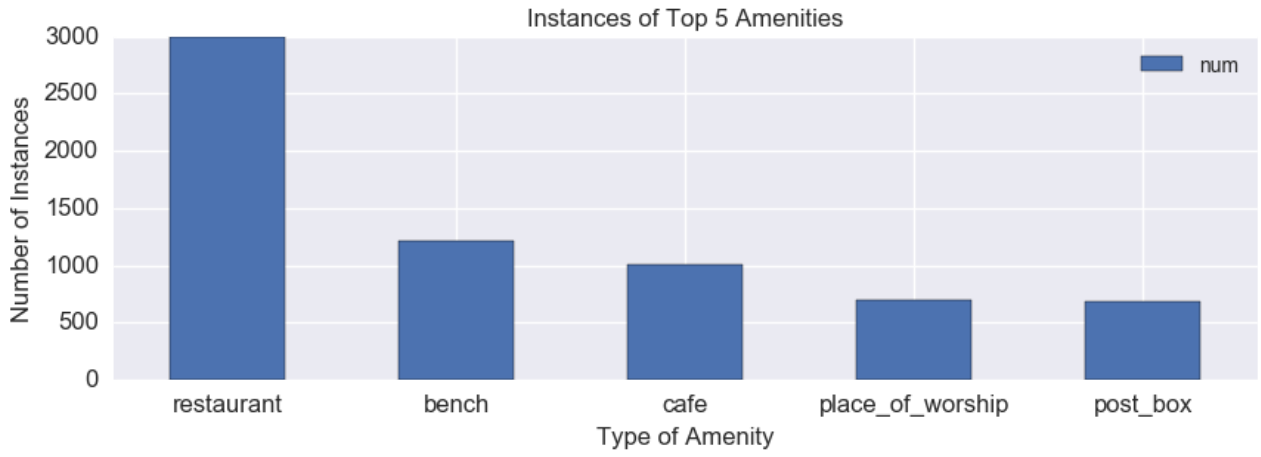


Amenities in the San Francisco Metro Extract

Not surprisingly, I found that the top amenity tag in the data is 'restaurant'. The Bay Area has tons of food establishments so this was expected. I was surprised, however, to see that the next most frequent amenity was 'bench'. Hummm, where are these?

In [13]:
```
# Data frame from the SQL query enumerating 5 most frequent amenities
amenities_df = pd.read_sql_query("""
SELECT value, COUNT(*) as num
FROM nodes_tags
WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 5;
""", conn)
```

In [14]:
```
# Plotting the 5 most frequent amenities by instance number
ax = amenities_df.set_index('value').plot.bar(rot=0, title='Instances of Top 5 Amenities', figsize=
(10,3), fontsize=12)
ax.set_xlabel("Type of Amenity",fontsize=12)
ax.set_ylabel("Number of Instances",fontsize=12)
plt.show()
```

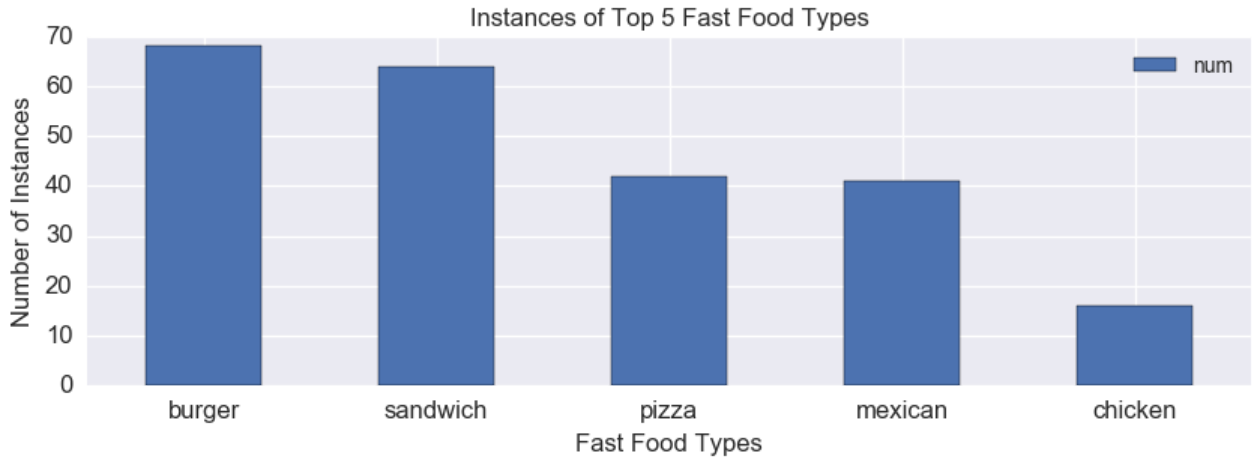


Fast Food Types in the San Francisco Metro Extract

My final visualization was to find the most frequent fast food types in the data. In my humble opinion burritos are San Francisco's greatest fast food, and I was curious if the data would confirm my biases. I found that 'burgers' are the most popular fast food type, with my preferred 'mexican' food lagging at fourth place.

In [15]:
```
fast_food_df = pd.read_sql_query("""
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='fast_food') i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 5;
""", conn)
```

In [16]:
```
# Plotting the 5 most fast food types by instance number
ax = fast_food_df.set_index('value').plot.bar(rot=0, title='Instances of Top 5 Fast Food Types', figsi
ze=(10,3), fontsize=12)
ax.set_xlabel("Fast Food Types",fontsize=12)
ax.set_ylabel("Number of Instances",fontsize=12)
plt.show()
```



Conclusions

My analyses identified two different aspects of the San Francisco Metro Extract data that could be improved. First, even though this data is for the City of San Francisco (and not the greater Bay Area, which is a separate extract) the data contains many other cities. Second, many street names are reported inconsistently, with one user reporting 'Broadway Street', another reporting 'Broadway Avenue', and yet another reporting simply 'Broadway'.

The first issue could be fixed by more strictly limiting the 'city' tag in the data. If a node isn't tagged as 'San Francisco' it could be omitted from the extract data. However, it's possible that the team at Open Street Map has made a deliberate decision to include all of this data as a way of ensuring full coverage. They may care less about inappropriately mapped tags than areas that receive no coverage. If this is their model for building the project, then additional stringencies wouldn't be necessary.

The second issue seems more pressing for improving the quality of the data. I examined several examples of annotation inconsistencies in street_type and it is likely that there are many other such inconsistencies in the data. To address the naming discrepancy one could parse the data looking for 'LIKE' items in street names, then compare the 'lat' and 'long' values in the nodes table. If elements have similar names ('Broadway Street' and 'Broadway Avenue') as well as similar geographic locations they should have the same street name. Rather than relying on user generated street names, this approach would require a having a list of official street names for each street within a given city.

There are certainty other improvements that could be made in the San Francisco Metro Extract data but these two issues are highly visible and would be nice improvements to the overall quality of the data in the Open Street Map project.

References

The sample project provided: https://gist.github.com/carlward/54ec1c91b62a5f911c42#file-sample_project-md (https://gist.github.com/carlward/54ec1c91b62a5f911c42#file-sample_project-md)

The Open Street Map Wiki: https://it.wikipedia.org/wiki/OpenStreetMap (https://it.wikipedia.org/wiki/OpenStreetMap)

> Particularly the 'Elements Page' http://wiki.openstreetmap.org/wiki/Elements (http://wiki.openstreetmap.org/wiki/Elements)

I got the DB Browser and a tutorial here: http://sqlitebrowser.org/ (http://sqlitebrowser.org/)