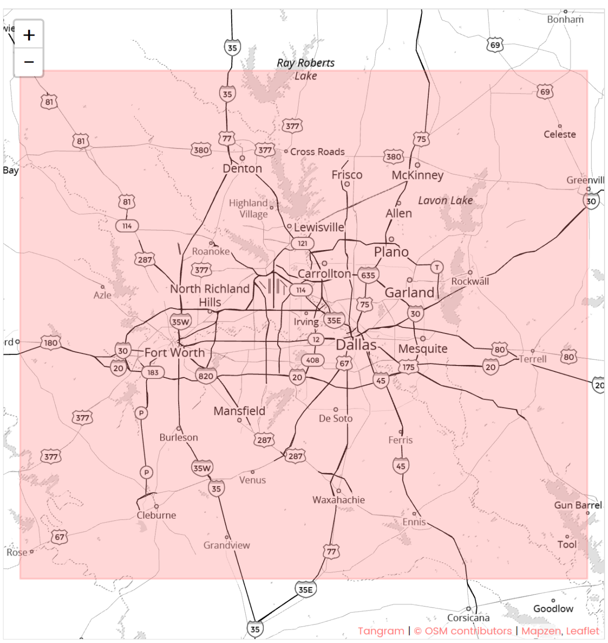
Dallas-Texas OpenStreetMap Data Case Study

```
import osm_functions as osmf
import osm_variables as osmv
import write_csvs as csv
import pandas as pd
```

Map Area

Dallas, Texas, United States.

The OpenStreetMap (OSM) data were downloaded from: https://mapzen.com/data/metro-extracts/metro/dallas_texas/ (https://mapzen.com/data/metro-extracts/metro/dallas_texas/), which provides chunks of OSM data clipped to the rectangular region surrounding Dallas - Fort Worth area as seen on the map below.



The boundaries of the subject area in this project are as follow:

```
pd.DataFrame(osmf.get_map_bounds(osmv.OSM_PATH), index=['min', 'max'])
```

	Latitude	Longitude
min	32.166	-97.789
max	33.431	-96.113

The Dallas metro extracts was chosen because my family is planning a trip to the area this summer. I think it would be interesting to see what the OpenStreetMap data of Dallas have to offer to first time visitors like me, and I also love the idea of contributing to its improvement on OpenStreetMap.org.

Problems Encountered in the Map

Prior to working with the entire dataset, an initial review was performed on several samples of the data. I noticed some problems with the samples as highlighted below:

Child Tag Keys

There's consistency issue within element child tag's key in the dataset, especially between regular data and Topologically Integrated Geographic Encoding and Referencing system (TIGER) data. For example, regular OSM data use "addr:street" as a key for street name, but TIGER based data use several tags to represent a street name, such as "tiger:name_base", "tiger:name_direction_prefix", and "tiger:name_type". In this project, I chose to focus on data cleaning for regular OSM data and to leave the TIGER based data as is.

Number of TIGER tags

```
sqlite> SELECT COUNT(*)
        FROM (
              SELECT * FROM nodes_tags UNION ALL
              SELECT * FROM ways_tags UNION ALL
              SELECT * FROM relations_tags) e
        WHERE e.type = 'tiger';
```

```
1296460
```

Number of total tags

```
sqlite> SELECT COUNT(*)
        FROM (
              SELECT * FROM nodes_tags UNION ALL
              SELECT * FROM ways_tags UNION ALL
              SELECT * FROM relations_tags) e;
```

```
3115530
```

Thus, 41.61% tags in the database are from TIGER, making the decision to leave them out in the cleaning process might seem unreasonable. However, considering the fact that these data were produced by US Census Bureau and prior editing had also been done to these data, I believe that the decision is reasonable. Here are some TIGER name_base tags pulled from the database

```
sqlite> SELECT value as name_base
        FROM ways_tags
        WHERE type = 'tiger' AND key='name_base'
        LIMIT 10;
```

```
name_base
----------------------
Dallas North Tollway
Hulen
Private Road 2415
Greenbrier
County Road 818
County Road 2424
Kirkwood
I-30
Private Road 2416
Private Road 2416
```

Street Name Values

After several reviews and audits using audit_street_name.py, I found the following problems with street name:

Problematic characters

- 'S. This could be the result of prior capitalization of the first letter of each words in street name. The fix to this problem is to replace 'S with 's. For example, **Green'S Court** should be updated to **Green's Court**.
- Comma character. Some street names contain a comma between street name and building number. The fix to this inconsistency is to remove comma from street name. For example: **Forest Central Drive, Suite 300** should be updated to **Forest Central Drive Suite 300**.
- Semicolon character. Some street names contain a semicolon such as in **East Harwood Road;Harwood Road**. I chose to remove the semicolon from street name and all characters that follow. The problematic street name in the example should be updated to **East Hardwood Road**.
- Ordinal number with capital letter. Some street names contain an ordinal number with capital letter. This could be another result of prior capitalization of the first letter of each words in street name. The fix to this problem is to replace capital letters in ordinal number with lower letters. For example, **West 12Th Street** should be updated to **West 12th Street**.

Building types

- Abbreviated suite. Some street names contain an abbreviation of the word suite, such as in **Avenue N Ste 3**. This street name should be updated to **Avenue N Suite 3**.
- Hash (#) character. Some street names have # character such as in **Ridge Road #110**. I chose to handle this issue by replacing # character with No. instead. The street name in the example should then be updated to **Ridge Road No.110**.

Abbreviated cardinal/ordinal points (direction)

Some street names have abbreviated point or points in them such as in **N Interstate 35 E**. This street name should be updated to **North Interstate 35 East**. The problem with updating points is that some letters might not correspond to a point at all such as in **West John W Carpenter Freeway**. The letter W in this street name should not be updated to West.

Abbreviated street types

Some street names have an abbreviated street type such as in **Avoca St.**. This street name should be updated to **Avoca Street**. In addition, I also found several special cases in street types such as **Hwy78**, which I chose to handle in the cleaning process of street types. Specific handlings of those special cases can be found in clean_street_name.py file under clean_type method.

Inconsistent highway name

Some highway names were inconsistently written. For example, **Texas Highway 78** is inconsistently written as **Hwy 78**, **State Highway 78**, or **TX 78**. All these names should be updated to the standard **Texas Highway 78**. I also found several special cases in highway naming and numbering such as in **S Interstate 35E**, which need special handling to update the number to 35 East. Specific handlings of those special cases can be found in clean_street_name.py file under clean_highway method.

Zip code Value

After several reviews and audits of samples data using audit_postcode.py, I found the following problems with zip codes:

- Non digit value. Some zip codes include an abbreviated state name such in **TX 75070**, some zip codes include '-' characters such as in **76209-1540**.
- Non 5-digit value. This is also the case with both zip codes in examples above.
- Non Dallas area zip codes. Some zip codes are not even from Dallas and it's the surrounding area. For example, **54231** is a zip code of an area in Kewaunee County in Wisconsin and it is not suppose to be in Dallas map. The fix to all these problems is to extract only the 5 digit zip code. The **TX 75070** should be updated to **75070**, and **76209-1540** to **76209**. I chose not to include zip codes that are not from surrounding Dallas area when writing the data into csv files for database import.

After importing the dataset with clean zip codes into a database, we can perform an aggregation below to see the result:

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

Here are the top ten result of zipcode sorted by count, edited for readability:

```
zipcode   count
--------  ------
75104      630
75093      325
75070      182
76013      182
75051      120
76210      119
75069       92
75019       89
75050       78
75080       73
```

City Name Value

After several reviews and audits of samples data using audit_city_name.py, I found the following problems with city names:

- Include state name. Some city names have state name included in them, such as in **Allen, Texas** or **Allen TX**. The fix to this problem is to remove state name from city name. Both **Allen, Texas** and **Allen TX** should be updated to **Allen**.
- Non alphabet. Some city names have non alphabet characters, such as in **Ft. Worth** or **4920**. I chose not to include city name like **4920** when writing the data into csv files for database import. Meanwhile, the **Ft. Worth** city name should be updated to **Fort Worth**.
- Abbreviated and misspelled name. Some city names have abbreviated word in it such as in **DFW**, and some others were misspelled such as in **De Soto**. I updated those names to **Dallas Fort Worth** and **DeSoto** respectively.

Below is an aggregation used to sort city name in the database which contains cleaned city names:

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

Here are the top ten city sorted by count, edited for readability:

```
city             count
---------------  ------
Frisco           49574
Plano             3059
Dallas             785
Cedar Hill         629
Fort Worth         504
Arlington          325
McKinney           299
Grand Prairie      255
Irving             146
Denton             138
```

Data Overview

File Sizes

```
Dallas_Texas.osm ......... 1370297.5 KB
Dallas (database) ......... 804670.5 KB
nodes.csv ................. 558753.3 KB
nodes_tags.csv ............. 12694.1 KB
relations.csv ................ 202.2 KB
relations_nodes.csv ........... 44.6 KB
relations_relations.csv ........ 4.2 KB
relations_tags.csv ........... 570.3 KB
relations_ways.csv ........... 832.4 KB
ways.csv ................... 42188.4 KB
ways_nodes.csv ............ 169299.1 KB
ways_tags.csv ............. 101753.0 KB
```

OSM Elements Counts

Elements counts in Dallas_Texas.osm

```
pd.DataFrame(osmf.get_element_count(osmv.OSM_PATH), index = ['counts'])
```

	node	relation	way
counts	6103171	3126	629047

Elements counts in Database

Node counts

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

6103171

Relation counts

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

3126

Way counts

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

629047

Notice that we have the same numbers of nodes, relations, and ways in the OSM file and in the database, which means we have successfully imported all elements into the database.

OSM User's Facts:

Number of unique users

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

2088

Top ten contributors

```
sqlite> SELECT e.uid, e.user, COUNT(*) as count
        FROM (
              SELECT uid, user FROM nodes UNION ALL
              SELECT uid, user FROM relations UNION ALL
              SELECT uid, user FROM ways
             ) e
        GROUP BY e.uid
        ORDER BY count DESC
        LIMIT 10;
```

The results are edited for readability:

```
uid       user                    count
--------  ----------------------  --------
4904442   Andrew Matheny_import   3374554
147510    woodpeck_fixbot         1081569
3265371   Andrew Matheny           225082
4018842   Stephen214               185248
2362216   TheDude05                133140
672878    TexasNHD                  88117
37392     25or6to4                  68580
36121     Chris Lawrence            60008
2012449   Dami_Tn                   54512
20587     balrog-kun                54405
```

Number of users appearing only once

```
sqlite> SELECT COUNT(*)
        FROM (
              SELECT e.uid, COUNT(*) as num
              FROM (
                    SELECT uid FROM nodes UNION ALL
                    SELECT uid FROM relations UNION ALL
                    SELECT uid FROM ways
                   ) e
              GROUP BY e.uid
              HAVING num=1
             ) u;
```

436

Skewed user contributions

Using similar aggregations as above, we can see that almost half of users contributed to less than 10 posts each and only 47 out of 2088 unique users contributed to more than ten thousand posts each.

Here are some user percentage statistics:

- Top user contribution (Andrew Matheny_import) - 50.10%
- Top 2 users contribution (Andrew Matheny_import & woodpect_fixbot) - 66.16%
- Top 10 users contribution - 76.06%

It is clear that user's contribution to OpenStreetMap is highly skewed.

Additional Data Exploration

Top 5 Religions

The rank is based on the number of OSM element tagged as place of worship:

```
sqlite> SELECT c.value as religion, COUNT(*) as count
        FROM (
              SELECT key, value FROM nodes_tags UNION ALL
              SELECT key, value FROM relations_tags UNION ALL
              SELECT key, value FROM ways_tags
             ) c
        WHERE c.key = 'religion'
        GROUP BY c.value
        ORDER BY count DESC
        LIMIT 5;
```

```
religion                  count
------------------------  ------
christian                 2832
muslim                    5
unitarian_universalist    4
hindu                     3
jewish                    3
```

Top 5 Most Popular Fast Food & Restaurant Chains

The rank is based on number of OSM element tagged either as fast food or restaurant.

```
sqlite> SELECT value AS name, COUNT(*) AS count
        FROM (
              SELECT value FROM nodes_tags
              WHERE key = 'name' AND id IN (
                    SELECT id FROM nodes_tags
                    WHERE key = 'amenity' AND (value = 'restaurant' OR value =
'fast_food'))
              UNION ALL
              SELECT value FROM ways_tags
              WHERE key = 'name' AND id IN (
                    SELECT id FROM ways_tags
                    WHERE key = 'amenity' AND (value = 'restaurant' OR value =
'fast_food'))
              UNION ALL
              SELECT value FROM relations_tags
              WHERE key = 'name' AND id IN (
                    SELECT id FROM relations_tags
                    WHERE key = 'amenity' AND (value = 'restaurant' OR value =
'fast_food')))
        GROUP BY name
        ORDER BY count DESC
        LIMIT 5;
```

```
name                  count
--------------------  -----
Whataburger           154
McDonald's            120
Chick-fil-A           74
Wendy's               64
Schlotzsky's Deli     61
```

Dallas Bike Routes

Number of bike routes

```
sqlite> SELECT COUNT(*) as count
        FROM (
              SELECT * FROM nodes_tags UNION ALL
              SELECT * FROM relations_tags UNION ALL
              SELECT * FROM ways_tags
             ) e
        WHERE e.key = 'route' and e.value = 'bicycle';
```

97

Bike routes with the most way connections

The rank is based on the number of ways element in the bike route relations

```
sqlite> SELECT relations_tags.value as route_name, e.id, e.way_counts
        FROM relations_tags
        JOIN (
              SELECT relations_ways.id, COUNT(*) as way_counts
              FROM relations_ways JOIN relations JOIN relations_tags
              ON relations_ways.id = relations.id AND relations.id = relations_tags.id
              WHERE relations_tags.key = 'route' AND relations_tags.value = 'bicycle'
              GROUP BY relations_ways.id) e
        ON relations_tags.id = e.id
        WHERE relations_tags.key = 'name'
        ORDER BY e.way_counts DESC
        LIMIT 5;
```

```
route_name            id          way_counts
--------------------  ----------  ----------
On-Street Route 45    1328724     246
On-Street Route 37    1310620     189
On-Street Route 100   1310535     118
On-Street Route 170   1310748     108
On-Street Route 23    1337848     108
```

Area of bike route with the most way connections

```
sqlite> SELECT Min(lat), Max(lat), Min(lon), Max(lon)
        FROM nodes
        WHERE id IN (
                     SELECT node_id FROM ways_nodes JOIN ways JOIN relations_ways
                     ON ways_nodes.id = ways.id AND ways.id = relations_ways.way_id
                     WHERE relations_ways.id = 1328724
                    );
```

```
Min(lat)    Max(lat)    Min(lon)    Max(lon)
----------  ----------  ----------  ----------
32.633195   32.9977948  -96.846283  -96.763855
```

Area of bike route with second most member way connections

```
sqlite> SELECT Min(lat), Max(lat), Min(lon), Max(lon)
        FROM nodes
        WHERE id IN (
                     SELECT node_id FROM ways_nodes JOIN ways JOIN relations_ways
                     ON ways_nodes.id = ways.id AND ways.id = relations_ways.way_id
                     WHERE relations_ways.id = 1310620
                    );
```

```
Min(lat)    Max(lat)    Min(lon)    Max(lon)
----------  ----------  ----------  ----------
32.6475542  33.0086425  -96.857216  -96.7834
```

Additional Ideas

Weekly Challenges

In order to motivate user's contribution, I think OpenStreetMap can create weekly challenges for its users to tag areas in the map based on specific theme for the week. For example, this week's theme could be place of worship, and next week's theme could be 'elementary school', and so on. As a reward for completing a challenge, users get points or badges. Furthermore, there can also be user ranking based on points or badges accumulation. At last, OpenStreetMap can let users share and showcase their OSM contribution on online social media, not only as a way to motivate current users but also to motivate new people to contribute to OpenStreetMap.

However, allowing too many new users to contribute to OpenStreetMap might increase user errors in OSM data which is not desirable. Therefore, a careful consideration is needed in an attempt to implement this suggestion.

More Cleaning

More data cleaning is needed to increase reliability of OSM data. Names of supermarkets and names of cafes in tag child elements are some examples of data that need cleaning.

While cleaning current data is easier to handle, keeping future data entries clean and consistent with cleaned data could be harder to do because the open source nature of OSM. Regular data cleaning could be an option to solve this issue.

The Use of OSM Element

After exploring the OSM database, I found some inconsistencies in the use of OSM element to represent a place. For example, a school can be tagged as a node, a way, or a relation, depending on how a user draw the place in the map.

```
sqlite> SELECT COUNT(*) FROM nodes_tags
        WHERE key = 'amenity' AND value = 'school';
```

```
614
```

```
sqlite> SELECT COUNT(*) FROM ways_tags
        WHERE key = 'amenity' AND value = 'school';
```

```
1330
```

```
sqlite> SELECT COUNT(*) FROM relations_tags
        WHERE key = 'amenity' AND value = 'school';
```

```
2
```

Therefore, in order to find all schools in the map, I have to queries all three tables (*nodes*, *ways*, and *relations*) as follow:

```
sqlite> SELECT COUNT(*) as count
        FROM (
              SELECT key, value FROM nodes_tags UNION ALL
              SELECT key, value FROM ways_tags UNION ALL
              SELECT key, value FROM relations_tags
             ) e
        WHERE e.key = 'amenity' AND e.value = 'school';
```

```
1946
```

However, since a *way*, per OSM definition, consists of *nodes*, and a *relation* consists of *ways* and/or *nodes*, it is not clear whether *nodes* in a school *way*, or *ways* in a school relations, are counted as different schools.

This confusion can be prevented by consistent use of OSM element to represent the same type of area. For example, a school should all be tagged as a *way* and a restaurant should also be tagged as a *way*. However, a stricter rule might discourage user contributions which is not desirable.

Conclusion

It is obvious after reviewing that the OSM data of Dallas area need further cleaning. Though most of the cleaning process can be done programmatically, manual cleaning might also be needed to handle special cases. Since clean data promote data consistency, this process is worth the effort though there is always a possibility of getting a less desirable side effect that a significant chunk of data could be trashed by a strict cleaning process. Additionally, data completeness is also a problem in OSM data that can only be improved by pulling location information from electronic devices instead of relying on users to input them manually.

References

- http://wiki.openstreetmap.org/wiki/OSM_XML (http://wiki.openstreetmap.org/wiki/OSM_XML)
- https://mapzen.com/data/metro-extracts/metro/dallas_texas/ (https://mapzen.com/data/metro-extracts/metro/dallas_texas/)
- https://discussions.udacity.com/t/help-cleaning-data/169833/81 (https://discussions.udacity.com/t/help-cleaning-data/169833/81)
- https://discussions.udacity.com/t/how-do-nodes-ways-and-relations-connected-to-each-other/245764 (https://discussions.udacity.com/t/how-do-nodes-ways-and-relations-connected-to-each-other/245764)