Wrangling OpenStreetMap Data

Map Area

Portland, OR, United States

- https://www.openstreetmap.org/relation/186579#map=11/45.5428/-122.6544

I decided to explore the Portland area because it is a city I have never been to and wanted to know a little more about.

Problems Encountered in Map

Since the original map file was fairly large to work with (1.2 GB), I created a small (8.9 MB) and intermediate (123.3 MB) sample file to work with in identifying the data cleaning tasks needed to be handled. Here were the most common problems, which I will discuss in this order:

- Inconsistent street name endings ("St.", "St", "Street")
- Other items mixed in to street names ("Southeast Orient Drive #C113, "US 26 (OR)")
- Inconsistent zip codes ("97219", "Portland, OR 97209", "97124-5961")
- Various state name abbreviations ("OR", "Or", "WA", "wa"). And yes, some Washington addresses appear in the database, which will be discussed later.

Inconsistent Street Name Endings

In order to deal with this issue, I decided to standardize street names into their full form. After auditing the small and intermediate samples, I identified which forms of street endings should be recognized as correct by the program, such as "Street", "Avenue", "Court", "Drive", "Boulevard" and some others. If the end of a street name did not match one of these, then it would be marked as an issue. There were some general mappings that dealt with a majority of these issues ("St." corrected to "Street", "Ave." corrected to "Avenue", "Hwy" corrected to "Highway", etc.).

Other Items Mixed In

Issues that were not corrected by the general mappings were corrected in one of two ways. The first way was to manually change some full street names. “North Marine Srive” became “North Marine Drive.” “Southeast Hwy 212” became “Southeast Highway 212” (since Hwy was already being corrected to “Highway” at the end of a street name). The second way was to take any street name that ended with a “#” symbol and a series of characters, such as “Northeast 82nd Avenue #D” and to get rid of the “#” and the accompanying characters. This was done for a few cases.
I did my best to correct what I could. Some street names were likely incorrectly entered by the user, such as “Botticelli,” “Brighton” (which could have been Brighton Way or Brighton Court in Portland), and “4616.” These were left alone since I did not have the information to correct them.

The final function used to update street names looked like this:

```
def update_name(name):
    """Return a street name in its proper form according to the conventions listed above"""

    m = street_type_re.search(name)
    street_type = m.group()
    if name in specific_mappings:
        name = name.replace(name, specific_mappings[name])
    elif street_type in mapping:
        name = name.replace(street_type, mapping[street_type]).strip(".").strip(" ")
    elif street_type in special_cases:
        name_to_replace = " #" + street_type
        name = name.replace(name_to_replace, "")
    return name
```

Inconsistent Zip Codes

Standardizing zip codes proved to be a much easier task. I decided to have all zip codes in a five-digit format. This meant I only had to correct hyphenated zip codes to be just five digits and get rid of “Portland OR,” when it appeared before a zip code. My function for updating zip codes is below:

```
def update_zip(zip_code):
    if "-" in zip_code:
        zip_code = zip_code.split("-")[0]
    if "Portland, OR " in zip_code:
        zip_code = zip_code.split("Portland, OR ")[1]
    return zip_code
```

State Name Abbreviations

For state names, I decided to stick to the full state name, since that form is more readable. There were various abbreviations used in the map file. After the auditing steps, I identified all the abbreviations used and created a mapping dictionary to correct them to the full state name (the way I did for the general street name mapping). Here is the function used to update the state names:

```
def update_state_name(state_name):
    if state_name in state_name_mapping:
        state_name = state_name.replace(state_name, state_name_mapping[state_name])
    return state_name
```

Database Overview and Interesting Queries

After auditing and cleaning the data, I exported the different nodes and tags into csv files (see database_prep.py) that would be used to create SQL tables in the Portland.db database. Adapting code from https://www.daniweb.com/programming/software-development/code/216951/size-of-a-file-folder-directory-python,
I was able to programmatically find the size of the following files:

map.osm	1.2 GB
Portland.db	978.2 MB
nodes.csv	441.6 MB
nodes_tags.csv	9 MB
ways.csv	43.7 MB
ways_nodes.csv	129.5 MB
ways_tags.csv	1171.1 MB

Number of Nodes

SELECT COUNT(*) FROM nodes;

4,896,167

Number of Ways

SELECT COUNT(*) FROM ways;

660,515

Number of Unique Users

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

1,253

Number of Appearances for Each City

I wanted to find which cities appeared most in the database.

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

Here are the top ten cities in descending order:

Portland	222,202
Beaverton	29,160
Hillsboro	23,812
Tigard	22,071
Gresham	19,299
Aloha	15,431
Lake Oswego	11,352
Milwaukie	9,905
Happy Valley	8,458
Troutdale	5,446

After researching these cities, it appears that the counties most represented are Multnomah County (where Portland is located), Washington County, and Clackamas County.

Places of Worship by Religion

Being of Jewish descent, I was interested in seeing how big Jewish life is in Portland and how that compares to the prominence of other religions.

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='place_of_worship') i
   ON nodes_tags.id=i.id
WHERE nodes_tags.key='religion'
GROUP BY nodes_tags.value
ORDER BY num DESC;
```

Christian	374
Buddhist	5
Jewish	2
Muslim	2
Eckankar	1
Hindu	1
Sikh	1
Unitarian Universalist	1

It appears that Jewish life is not at all big in Portland.

Additional Ideas

The auditing process revealed that some addresses in the map file came from Washington. In order to investigate this further, I looked at the number of times each postal code appeared in the database:

```
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;
```

This query revealed that there were a good amount of zip codes from Washington (beginning with “986”). Some of these zips codes appeared more than one hundred times.

I then looked at the number of times each state appeared in the database:

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
```

```
        SELECT * FROM ways_tags) tags
WHERE tags.key = 'state'
GROUP BY tags.value
ORDER BY count DESC;
```

Oregon appeared 31,359 times in the database. Washington appeared 686 times.

Therefore, one way to improve the way this data is inputted on OpenStreetMap is to create the boundaries for an area based on zip codes, rather than latitude or longitude. An advantage of this approach would be the ability to verify an address is in the area a user says it is. However, this approach does not take into account that zip codes are not a perfect indicator of where someone lives. Some people who listed zip codes from Washington might actually live in Oregon but their zip code could have been from a town that is near the border of the two states. Although "972" appears to be the first three digits of all Portland zip codes, Wikipedia (https://en.wikipedia.org/wiki/List_of_ZIP_code_prefixes#Starts_with_9) indicates that a "986" zip code (which I believed to be in Washington) is actually from Portland as well.

Conclusions

This project was a good start in seeing the potential for analysis in the OpenStreetMap database. A lot can be done to work with the data and its quirks in Python and SQL. Still, some of the data (especially street names) were fairly messy and I wonder if there is a better way for users to input their data, in which the system forces the user to enter the data in a more standardized form. Nevertheless, the Portland area of the map was fairly large to work with, which left a lot of room for exploration on my end.