Data Analyst Nanodegree Assignment III—Wrangle OpenStreetMap Data

Map Area:

For this project, I chose to analyze data from Toronto, Canada. I chose this region because it is the setting of my favorite detective show, *The Murdoch Mysteries*.

Benefits:

The dataset does a good job of sorting out the various types of establishments and their counts. It is pretty easy to access the different types of nodes from the dataset just by running the queries. On a related note, the dataset is also a good indicator of how many different types of ethnic groups live in Toronto.

Problems:

One problem that I noticed arose when I used sqlite3 to examine place names. There are two entries for Yola's (shown below): They are "Yola's" and "Yola's Nails, Hair Design & Esthetics." It is unclear whether these are in fact two completely separate places or the same place entered twice by mistake.

```
Yola's|2
Yola's Nails, Hair Design & Esthetics|2
```

The same problem arises under yogurt shops. I found several misspellings and inconsistencies in their names, as shown below.

```
Yoguart's Froyo|2
Yogurt's|2
Yogurtys|2
Yogurtys Froyo|2
```

On a related note, some places are entered multiple times with the same count. For example, Sherway Gardens Road is entered three times, once as "Sherway Gardens; The Queensway," another time as "Sherway Gardens; The Queensway; Evans Avenue; Browns Line," and once more as "Sherway Gardens; The Queensway; Evans Avenue; Browns Line 1 km / Dundas Street."

```
Sherway Gardens Road; The Queensway|2
Sherway Gardens Road; The Queensway; Evans Avenue; Browns
Line|2
Sherway Gardens Road; The Queensway; Evans Avenue; Browns
Line 1 km / Dundas Street|2
```

The same issue arises in Sherwin Williams. In some instances, it is written as "Sherwin Williams" and in other instances it is written as "Sherwin-Williams." Other flaws include omissions of the word "Paints."

```
Sherwin Williams|2
Sherwin-Willams Paint|2
Sherwin-Williams|2
Sherwin-Williams Paint|2
Sherwin-Williams Pants|2
```

Relevant Demographic Data:

As I expected, the three most popular religions in Toronto are Christianity, Islam, and Judaism. The following snippet displays the religions represented in Toronto and the number of people who follow each religion (presumably in thousands).

```
christian|660
muslim|34
jewish|18
buddhist|10
hindu|6
taoist|6
eckankar|2
scientologist|2
sikh|2
zoroastrian|2
```

I also tried to determine the male-to-female ratio within Toronto, whereupon I ran into some problems. When I ran the query to determine the number of males, I got this result:

```
yes|32
no|2
```

But when I ran the query to determine the number of females, I was expecting:

```
yes|2
no|32
```

But instead I got:

```
yes|22
```

This could possibly be due to men misidentifying themselves, because there is no "no" line when I ran the query for females.

Different counts of the CSV:

After running the query to determine the size of each CSV, I discovered that there are 4,891,770 nodes, 716,252 ways, 4,650,570 nodes in tags, and only 1 tag in ways.

Queries:

I used the following queries to obtain the data outlined in this report:

- These queries find the sex of the people living in Toronto:
  - ```
    SELECT tags.value, COUNT(*) as count FROM (SELECT *
    FROM nodes_tags UNION ALL SELECT * FROM ways_tags)
    tags WHERE tags.key='male' GROUP BY tags.value ORDER
    BY count DESC;
    ```
  - ```
    SELECT tags.value, COUNT(*) as count FROM (SELECT *
    FROM nodes_tags UNION ALL SELECT * FROM ways_tags)
    tags WHERE tags.key='female' GROUP BY tags.value
    ORDER BY count DESC;
    ```
- This query finds the different religious groups in Toronto:
  - ```
    SELECT tags.value, COUNT(*) as count FROM (SELECT *
    FROM nodes_tags UNION ALL SELECT * FROM ways_tags)
    tags WHERE tags.key='religion' GROUP BY tags.value
    ORDER BY count DESC;
    ```
- These queries find the names and types of all proprietary establishments:
  - ```
    SELECT tags.value, COUNT(*) as count FROM (SELECT *
    FROM nodes_tags UNION ALL SELECT * FROM ways_tags)
    tags WHERE tags.key='name' GROUP BY tags.value ORDER
    BY count DESC;
    ```
  - ```
    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;
    ```
- Finally, this query finds all of the streets in Toronto:
  - ```
    SELECT tags.value, COUNT(*) as count FROM (SELECT *
    FROM nodes_tags UNION ALL SELECT * FROM ways_tags)
    tags WHERE tags.key='street' GROUP BY tags.value
    ORDER BY count DESC;
    ```

Cleaning the Dataset:

I used the above function from audit.py to try to clean the street names. The variable “street_re” is the object that ignores discrepancies in uppercase and lowercase and it also ignores special characters such as ampersand signs, periods, etc. The variable

“k” is the street name itself. These street names are then collected in the group produced by “k.group().” If a street name is not found, the function raises an exception. The variable “l” is the type of road in the mapping that corresponds to “k.” If the program finds a type of roadway that is not in “l,” it raises an exception. The variable “new_name” replaces the current name with the corrected name, and the return line produces the new street name.

```
def update_name(name, mapping):
k = street_re.search(name)
k = k.group()
if not k:
raise Exception(name)
l = mapping[k]
if not l:
raise Exception(k)
new_name = re.sub(k, l, name)
return new_name
```

I also made a small edit to the above function to address issues in the names of the amenities. I made a list of the types of amenities and used the above function, the only edit being substituting all instances of “mapping” with “amenity.”

```
amenity = { "School", "Office", "Station", "Employment",}

def update_amenity(name, amenity): #cleans amenity names
k = street_re.search(name)
k = k.group()
if not k:
raise Exception(name)
l = amenity[k]
if not l:
raise Exception(l)
new_name = re.sub(k, l, name)
return new_name
```

Suggested Fixes:

One way to fix the dataset would be to include places beginning with the same name under one umbrella. For example, instead of recording all instances of “Sherwin-Williams” separately, include them all under the “Sherwin-Williams” umbrella since they all have the same count (2).

Another way to fix the dataset would be to use Google Maps to try to sort the different types of establishments in Toronto. This would reduce the number of duplicate entries because they would improve the location services of the different types of establishments.

Anticipated Problems:

However, the fixes proposed above could have problems. For example, even Google Maps could fail because it would group establishments with the same name in

different locations together. From the data that I obtained, it is difficult to determine whether repeat instances of certain establishments appear because of accidental reentry, or because they are franchises, in which case they would have multiple locations.

Sources:

I obtained the queries from the sample project found here:
https://gist.github.com/carlward/54ec1c91b62a5f911c42#file-sample_project-md

I adapted my update function from the code found at this link:
https://github.com/tollek/udacity-data-science/blob/master/p3/case_study6/audit.py