SQL Project - Udacity:
OpenStreetMap Data Case Study

Map Area – City of Toronto, ON, Canada
https://www.openstreetmap.org/relation/324211

Toronto is my hometown, so I'm interested in seeing what I can find out from the database, but first, I would like to see how clean the data is, and see what I can do to audit the data.

Data Cleaning Summary:
After first downloading a small sample size of the Toronto area (see Toronto-test.py), I audited the three 'kvalues' below and cleaned some common issues in these data:

1. Street Name (kvalue = addr:street)
2. Postal Code (kvalue = addr:postcode)
3. City (kvalue = addr:city)

1. Street Names (Over abbreviated)

First, I ran the file against audit_kvalue.py for "addr:street" to get a sense of all the street names. I noticed some over abbreviated street names, and started adding those to the the list of names (mapping{}) that I'd like to update in improve_data.py. I took the code that was done for "Quiz: Improving Street Names" and adapted it to fit the Toronto dataset.

```
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)
m = street_type_re.search(street_name)
if m:
   street_type = m.group()
   if street_type not in expected:
      street_types[street_type].add(street_name)
```

2. Postal Codes (Inconsistent formatting)

Next, I did an audit on all the postal codes by running audit_kvalue.py for "addr:postcode". I noticed that they are not all in the same format. The correct format should be "A1A 1A1". Common problems I saw include, missing space in the middle and lower case instead of uppercase. So I added in codes that address that in the improve_data.py. (for code see post_code.py)

```
postcode = re.compile(r'[A-z]\d[A-z]\s?\d[A-z]\d')
m = postcode.match(post_code)
if m:
   if " " not in post_code:
      post_code = post_code [:3]+" "+post_code[3:]
   post_code = post_code.upper()
```

However, even after the clean, there are still odd ones that did not get fixed (e.g. missing part of the code, or having more than one post code listed.

3. City (Inconsistent Naming Convention)

Lastly, I ran the audit_kvalues.py file for “addr:city”. I noticed that sometimes the names start with “City/Town of/Township of”, and sometimes it doesn’t. I decided to delete all the “City/Town/Township of" to be consistent.
Additionally, there were also a few odd ones that I adjusted for:

- City names with “,”
- City names with “;”
- Mis-spelled city names (Torontoitalian and Etoicoke).

```
cityof = re.compile('of',re.IGNORECASE )
first= re.compile(';')
second = re.compile(',')
township = re.compile('Township')
bracket = re.compile(r'\(')
    m = cityof.search(city)
    n = first.search(city)
    o = second.search(city)
    t = township.search(city)
    b = bracket.search(city)
# if the city name starts with township, then only take the part after "township of"
    if t:
        city = city[12:].title()
# if the city name has bracket "(", then only take what's inside the bracket

    elif b:
        city = city.split("(")[1][:-1].title()
# if the city name has ";", then only take the part
    elif n:
        city = city.split(";")[1][8:].title()
# if the city name has "," in it, then only take the first part before the ","
    elif o:
        city = city.split(",")[0].title()
# if the city name has "city/town of" in it, then only take the part after

    elif m:
        city = city[8:].title()
# correct mis-spelled
    elif city == "Etoicoke":
        city = "Etobicoke"
# correct incorrect city torontoitalian
    elif city =="Torontoitalian":
```

```
city = “Toronto"
```

I understand that there are additional cleaning that could be done, but for the purpose of this exercise, I stopped after these three.
I then ran the improve_data.py to make all the changes I have identified, ran prepare_csv.py to prepare the individual csv files for each table, and create_db.py to create ‘database.db’ to use for querying.

<u>Data Analysis:</u>

To get some statistics on the Toronto dataset, I performed the following:

- Size of the file (see filesize.py): 552 MB
    - Confirms file is >50 MB
- Number of unique users (see users.py): 2,125
- Number of nodes: 2,113,891

```
SELECT COUNT(*)
FROM nodes
```

- Number of ways: 361,254

```
SELECT COUNT(*)
FROM ways
```

- Number of Highway nodes: 53,280

```
SELECT COUNT(DISTINCT(id))
FROM nodes_tags
WHERE key = "highway";
```

- Top 10 types of way tag:

```
SELECT type, COUNT(*)
FROM ways_tags
GROUP by type
ORDER by COUNT(*) DESC
LIMIT 10;

[(u'regular', 1018694),
 (u'addr', 167065),
 (u'geobase', 28592),
 (u'statscan', 20461),
 (u'building', 15086),
 (u'turn', 4620),
 (u'opendata', 3964),
 (u'lanes', 2976),
 (u'is_in', 2149),
 (u'destination', 2057)]
```

The most popular tag type is “Regular”. Makes sense, since regular is the default value for “type” if there isn’t one. Second most popular tag type is addr, which I remember include tag keys like postcode, street, and city. Let’s take a look at what are the top 10 keys for addr:

```
SELECT key, count(*)
FROM ways_tags
WHERE type = 'addr'
GROUP by KEY
ORDER by COUNT(*) DESC
LIMIT 10;

[(u'interpolation', 97147),
 (u'street', 15552),
 (u'housenumber', 15192),
 (u'city', 13892),
 (u'country', 9635),
 (u'province', 8852),
 (u'postcode', 3977),
 (u'state', 2450),
 (u'inclusion', 177),
 (u'housename', 129)]
```

We can see that “interpolation” is the most population tag key for the addr type, and then it’s street and housenumber. Postcode is 7th from the list. The % of address with house number with postcode is 3977/15192 = 26.19%. That’s quite low. Though I imagine, we can simply look up the missing postal code for complete addresses.

- Top 10 users:

```
SELECT e.user, COUNT(*)
FROM (SELECT user FROM nodes
    UNION ALL
    SELECT user FROM ways) e
GROUP by e.user
ORDER by COUNT(*) DESC
LIMIT 10;

[(u'andrewpmk', 1643330),
 (u'Kevo', 177484),
 (u'Victor Bielawski', 112402),
 (u'Bootprint', 66605),
 (u'Mojgan Jadidi', 43356),
 (u'MikeyCarter', 27032),
 (u'andrewpmk_imports', 26849),
 (u'TristanA', 25206),
 (u'Nate_Wessel', 22088),
 (u'geobase_stevens', 18914)]
```

So it looks like 2 of the top 10 users might be the same person (‘andrewpmk’ and ‘andrewpmk_imports’). It’s interesting that the top contributor ‘andrewpmk’ contributed almost more than 10x as the second most contributor. I wonder if ‘andrewpmk’ is some kind of professional openstreetmap-er. I did a bit of searching on google but couldn’t find much more.

- Number of restaurants and what key they are under: 2335 under amenity and 1 under vacant

```
SELECT KEY, COUNT(DISTINCT(ID))
from nodes_tags
WHERE value = 'restaurant'
GROUP by key;

[(u'amenity', 2335), (u'vacant', 1)]
```

Not exactly sure what ‘vacant’ means, if it’s a mistake in the dataset or if it means 1 out of the 2336 restaurants is currently closed (which is unlikely).

<u>Additional Ideas:</u>

- Further Cleaning:

The City of Toronto data can be cleaned a lot more, there are many fields that I have not touched (phone number, splitting out the unit/suite number in some street names, etc.).
Benefit: Once the data is cleaned more, we can then try to gain additional insights into the dataset.
Anticipated Problems: There are some problems that can’t be solved by more cleaning, including missing data, and data that just doesn’t make sense.

- Data Consistency Guideline: Create a guideline for users on inputting data. This can be done by having mandatory tags for specific types of nodes, and have specific format for tag values to follow.

Benefit: It can also be helpful to reduce the amount of possible tag keys. I see that some of the tags occurs very rarely, it could be beneficial to set up pre-determined tags (bucketing) to allow more meaningful data analysis, to be rid of the noise.

Anticipated Problems: The more strict the guidelines, the more consistent the data will be, but at the same time, we might lose some degree of accuracy when we try to conform all the data, so we need to aim to strike a balance.

- Improve Data Missing:

Imputing missing values from other values within the same node. For example, we can fill in the postal code by looking up an address.
Benefit: The benefit of having more postal code filled in is that we can conduct analysis by smaller regions than City, by looking at similar postal codes.
Anticipated Problems: This exercise will depend highly on the accuracy of pre-existing data, and any inaccurate data will lead to just more inaccurate, inferred data. So prior to improving data missing, we should first do what we can to ensure data accuracy and correctness.

Additional Data Exploration:

There are many areas that can be further explored. For example, we can further explore all the restaurant nodes by looking at all the tag keys associated with restaurant nodes:

```
SELECT key, COUNT(*)
FROM nodes_tags JOIN
(SELECT DISTINCT(id) from nodes_tags where value = 'restaurant') as u
WHERE u.id = nodes_tags.id
GROUP by key
ORDER by COUNT(*) DESC

[(u'amenity', 2336),
 (u'name', 2310),
 (u'cuisine', 1033),
 (u'street', 854),
 (u'housenumber', 835),
 (u'city', 703),
 (u'phone', 553),
 (u'province', 542),
 (u'website', 524),
 (u'country', 473),
 (u'postcode', 339),
 (u'wheelchair', 216),
 (u'opening_hours', 195),
……
)
```

We can then look into specific tag keys for additional insights. Example analysis include, number of restaurant by type of cuisine, number of restaurants with wheelchair access, most common opening hours, street with the most restaurants, etc.