Data Wrangling project for OpenStreetMap data of Bozeman, MT

This project is a part of Udacity's Data Analyst Nanodegree program. It utilizes a set of skills from XML for data extraction, SQL for database creating and querying, as well as Python for data visualization.

The overall objective is to extract data from a map of your choosing, and then process, clean, commit and explore the data.

List of Tasks for Project

Data Extraction

- [x] **Choose Map Area**

The map extract that I chose from Mapzen was of the Bozeman, MT area in the United States. It is a small city in the a very sparsely populated state. The initial extract of the metropolitan area didn't yield enough data for the basic size requirements. As a result, the User can extract more data by expanding map area to include the metropolitan area as well as the surrounding peri-urbanization.

- [x] **Utilize the Overpass API to extract map**

Most of this data collection was working with the Mapzen task outlined in references. User has to register with Mapzen, choose the parameters and then Mapzen sends the raw data file to process.

Reference:
https://www.openstreetmap.org/#map=7/45.132/-110.253
http://wiki.openstreetmap.org/wiki/Overpass_API

Data Processing

- [x] **Transfer the Dataset**
- [x] **Clean the Data Set**

Cleaning of Street Names

There were several functions that helped clean up the data set. For example, in the

- *py.project_openstreetmap_preparing_db.py*

file, there are several cElementTree functions that check if the phone numbers were put in correctly, the street names fit the schema and if the elements could be changed into dicts. When applicable, many tags were cleaned to fit the schema for consistency. See below for some of the street name changes.

> 'US Highway 10 West' -> 'US Highway 10 West'
> 'Bullrush Ave' -> 'Bullrush Avenue'
> '40 Spanish Peak Dr' -> '40 Spanish Peak Drive'
> 'Gallatin Park Dr' -> 'Gallatin Park Drive'
> 'Main ST' -> 'Main Street'
> 'Davis Ln' -> 'Davis Lane'

The additional functions used to clean the data are commented in the python file for each step along the process.

Write Data Set to DB

Created a sample file so as to explore the data without having to run queries on the whole map. File(s) below:

- *py.project_openstreetmap_create_sample_file.py*

The bulk of the work was done from this process. In the files below, the following steps were generally followed

- the program can extract data from OSM files,
- queried basic details about the types of data,
- inserted the data into schema,
- cleaned the data to match the schema,
- wrote new CSVs with the extracted data from the OSM,
- wrote the data from the CSVs onto a SQL database

File(s) below:

- *py.project_openstreetmap_preparing_db.py*
- *py.project_openstreetmap_inserting_db.py*
- *py.project_openstreetmap_data_case_study.py*

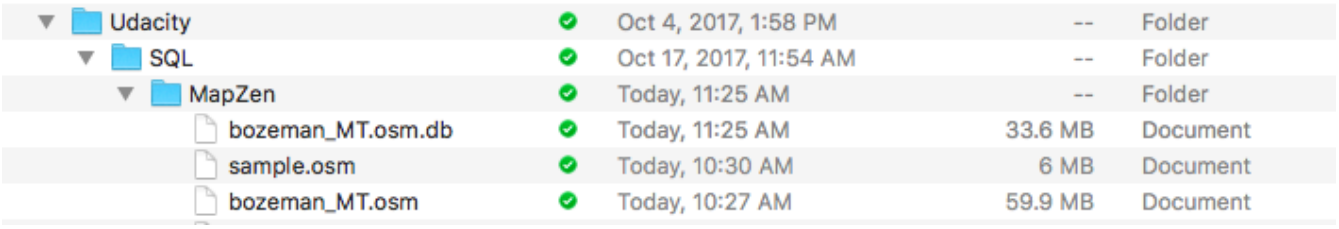
Reference:
http://stackoverflow.com/questions/3095434/inserting-newlines-in-xml-file-generated-via-xml-etree-elementtree-in-python

Data Exploring

Address basic questions

Size of the files

As pictured below, the uncompressed file for the map excerpt is larger than 50Mb



Number of unique users

```
unique_users = cur.execute("""SELECT COUNT(DISTINCT(e.uid))
                FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;""
pprint (unique_users)
```

```
[(235,)]
```

Number of nodes and ways

```
cur.execute('SELECT COUNT (*) FROM nodes')
cur.fetchall()
```

```
[(293237,)]
```

```
cur.execute("SELECT COUNT (*) FROM ways")
cur.fetchall()
```

```
[(17080,)]
```

Number of chosen type of nodes, like cafes, shops etc.

Coded below is a count of a chose nodes_tag for amount of cafes in the Bozeman area.

```
cur.execute("""SELECT COUNT (*) FROM nodes_tags WHERE value = 'cafe'""")
cur.fetchall()
```

```
[(13,)]
```

- **Populate and design visualization**

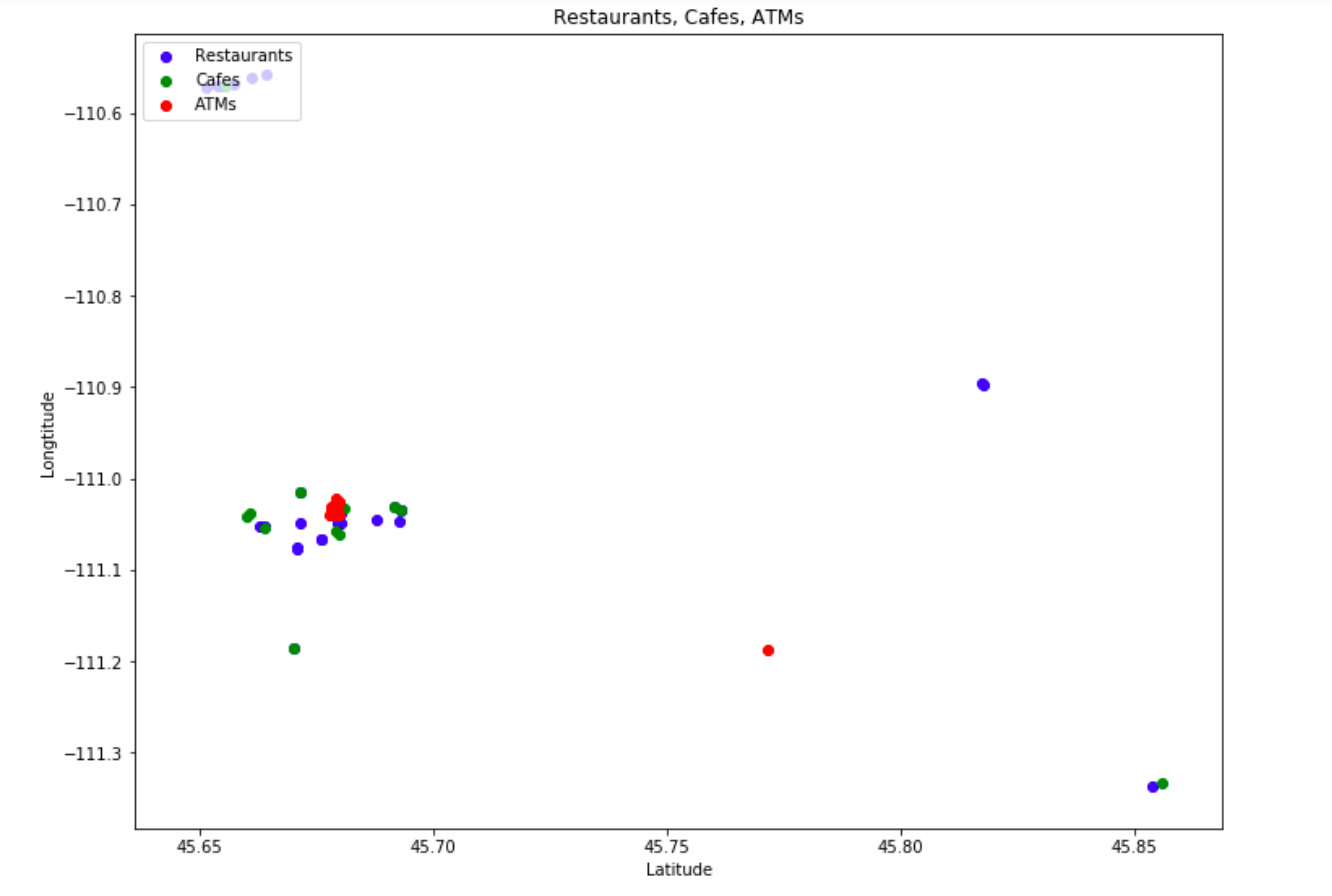
In this section, there is a mix of SQL functions to organize data and Matplotlib to visual the data. There is a simple plot of a subset of data shown below with the code. File below:

- *py.project_openstreetmap_data_chart.py*

When the user chooses the venues in Bozeman and separate them by type, they can lay them out on map set out with the longitude and latitude as axes. Please see visuzalation code and plot below:

```
plt.scatter([x[2] for x in cuisine_loc], [y[3] for y in cuisine_loc], c='blue',
plt.scatter([x[2] for x in cafe], [y[3] for y in cafe], c='green', label="Cafes"
plt.scatter([x[2] for x in atm], [y[3] for y in atm], c='red', label="ATMs")
plt.xlabel('Latitude')
plt.ylabel('Longtitude')
plt.title('Restaurants, Cafes, ATMs')
plt.legend(loc=2)
```

!



For this plot, you can see the coordinates for different Restaurants, Cafes, and ATMs in the Bozeman, MT area.

Reference:
https://github.com/mudspringhiker/openstreetmap_datawrangling

☑️ Suggestions for Improvement of Data

It was unclear to me whether some of the tag values could be mutually exclusive or not. The main example is how most all 'cafes' could be considered 'cuisine', but not vice versa. This gave me some reservations as to how to compare venues.

One suggestion for improvement to this end would be to have multi-value field for some of these tags.

Benefits:

- more comprehensive data points
    - more robust narrative
- lends towards a more flexible database

    - will matter in the future should there be a new type of data that calls for multiple values
- allows the Data Analyst to have to think about the dataset in new ways

Anticipated Problems:

- no clear primary designation
- duplicate points
    - presents problems for data visualization that have to plot multiple points on the same plane
- creates a vulnerability for database
    - once a database has the ability for multiple value inputs, a bot can crash the database by entering values ad-infinitum

Also, it would be a big help to be able to predict the size of the map excerpt before extracting. While most of the initial code checks were done with the sample file, once I brought in osm files that were 390+mb, I came across a myriad of debugging issues as it pertains to the my writing of csvs, running my kernal on the my Jupyter notebook, and troubleshooting. As a result, I ran my checks on files of different sizes and didn't have the benefit of knowing how large each file would before I would run it.

Benefits:

- makes project planning easier
    - estimates of file sizes will guide the actual area of the map extracted
- prevents mistakes like those that I was encountering

Anticipated Problems:

- bloats the web app
    - I am not sure how this would be file estimation would be done, but I imagine it would require one more program as part of the total application
- makes the data analyst less careful in their project planning

One more conceptual suggestion - I wonder if there is a more public discussion around 'Staff Mappers' and their role in the community of mappers. To my understanding there are enterprise departments that use this public database, tidy up the data and contribute back to the db. I am generally for this. However, I was unable to differentiate between amateur users and staffed users and this can be a bit controversial. In a world where gerrymandering, transnational security, and humanitarian crises are all addressed using mapping data (no pun intended), I suggest there be a formal designation of the types of mappers there are in the

community within the data submissions. In general, this in line with the maxim of "more available information the better".

The rest of the data was pretty straight forward and I was glad to have the hints to use sample data.

Conclusion

This project was informative from a data extraction and database writing perspective. In this respect the project directly reflects the Data Wrangling course from Udacity. Whether it was cleaning out the data files, writing onto files, or inserting a schema, this project tests a lot of these skills. There is still a lot of work that can be done on data visualization prospective, but that has been tested earlier in the courses and will continued to be tested.

The extensive code review and versioning has been crucial throughout this process as well.

Complete Set of References

Udacity Data Wrangling Course

Other's Sample Projects
https://github.com/mudspringhiker/openstreetmap_datawrangling
https://gist.github.com/carlward/54ec1c91b62a5f911c42
https://github.com/jkarakas/Wrangle-OpenStreetMaps-Data-with-SQL

SQLite documentation
https://docs.python.org/3.6/library/sqlite3.html

XML Extraction documentation
http://stackoverflow.com/questions/3095434/inserting-newlines-in-xml-file-generated-via-xml-etree-elementtree-in-python

Python3 issues
https://stackoverflow.com/questions/8515053/csv-error-iterator-should-return-strings-not-bytes

Mapzen data
https://www.openstreetmap.org/#map=7/45.132/-110.253
http://wiki.openstreetmap.org/wiki/Overpass_API