Problems Encountered in the Dataset:

When I first started working with this dataset, the first problem I encountered was how I wanted to set up my data. I am fairly familiar with Pandas, and I usually like to upload my data into a Pandas DF, and make queries from there. However, I had a hard time getting a XML file into the dataframe. In the past I have used import from csv functions, and soon realized that there was not an equivalent XML function that I could get to work properly in a pandas DF.

Following that I needed to learn how I can access the <tag> elements of the <node> and <way> tags. At first I thought that running a for loop over the nodes would capture the info of the <tags> so that I could later pull the information as needed, for example to look at different postal codes in the dataset. However, as I continued to query the data I soon realized this was not the case, and that somehow python seemed to skip over those tags when creating arrays or dictionaries of the <node> tags. It wasn't until I started utilizing BeautifulSoup, that I was able to really pull the information on the <tags>.

Next I looked at street names, and how they were written in the OSM file. I used code from the class lessons in Udacity and found that many streets had different abbreviations and names. For example Avenue vs Ave., Street vs St. , Boulevard Vs. Blvd etc. On top of that I could see that there were some naming conventions that did not fall into any standard category. For example results came back that a street had been named 40, and another named s. Most likely these were entry errors, where someone entered the data into the wrong section.

After looking into street names, I decided I wanted to audit zip codes for accuracy and uniformity. I thought about auditing a few different sections of the OSM dataset, but this seemed to make the most sense to look for entry inaccuracies. Within the zip codes I could see that there were zip

codes that did not belong to the city of Reno. Within the data there were 6 different zip codes that did not belong to the city of Reno. These codes were 89434, 89431, 89432, 89435, 89436, and 96118, which is a California zip code, so really off there. Once I discovered these zip codes, I then cleaned and the dataset of these zip codes.

I thought that after that I was near the finish line of completing this project, but then realized that converting the dataset into multiple csv files was quite complicated. One issues I came across was that the <tag> of <nodes> and <way> are not always within there tags. For example, you can have 10 <node> tags with <tags> embedded within them, and 100 without. With so many missing elements it made transferring the data to csv tables difficult.

Lastly running the data through Sqlite3 DB, was more complicated than it should be, I did not like the Sqlite3 DB CMD prompt, and found it confusing and hard to work with. I then found that there is a DB browser that connects to Sqlite3, so that you do not have to use the CMD interface. I found this to be much simpler. I was still able to use SQL commands to create, drop, and query my tables, but with a better interface.

Overview of the Data:

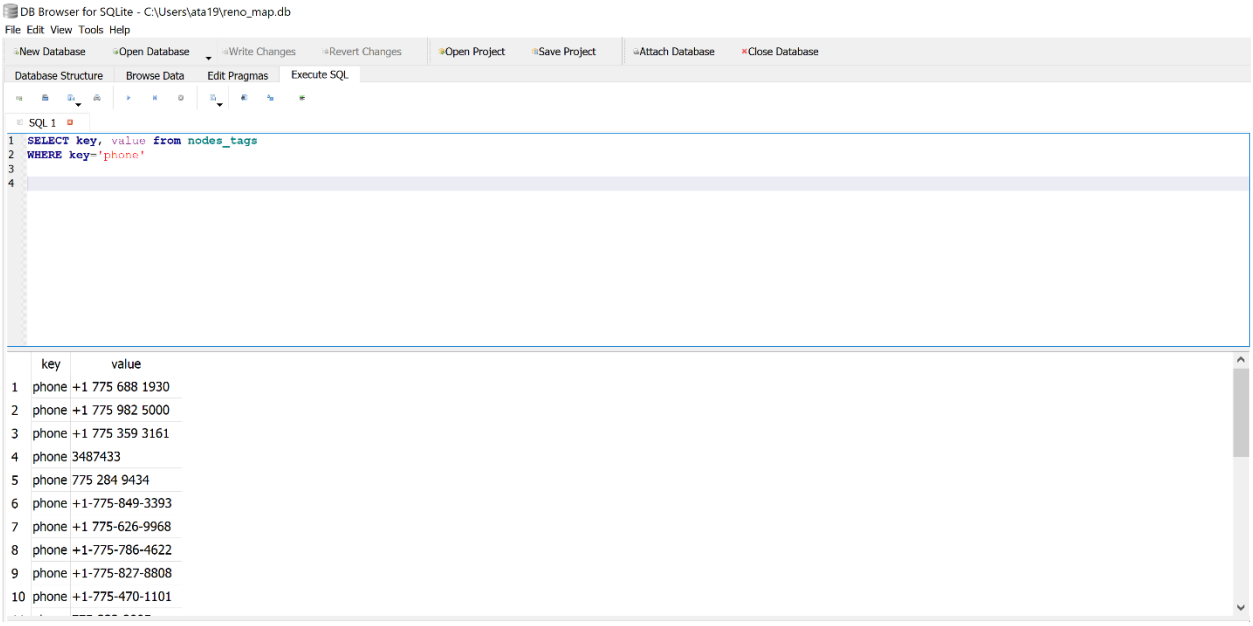
This data set is of the city of Reno Nevada. I have lived here in Reno for over 20 years, so I thought I'd have a better understanding of the data if I already knew the area. The file size for my Sample2.osm files is around 25 MB, the original file of all the data of Reno was 245 MB.

Other Ideas about the Data Set:

I felt overall this data set was harder to get data, sense we are only pulling about a 1/10th sample size to work with. I originally wanted to query things like how many schools are in Reno, or how many Starbuck's there are. I felt these results would be inaccurate, as we are not pulling all the data. So to say in my sample there were only 5 Starbucks's within the city, would be false. I also felt there was not a lot of options on things to clean/query. For both node and way tags, there latitudes, longitudes, changesets, timestamps, users, etc. But this is not really information about the city itself, and not something that you can clean/query.

I feel that for its time, OSM is a pretty cool feature, but you can tell that map apps like google or apple maps a lot better and more up to date and accurate. With this data being made of human updates, there is a lot of things that are inaccurate or not uniform.

One recommendation My suggestion for this website openstreetmap.com, would be to use an html validator on the client, similar to what we use when filling out a form online. For example some websites will only allow phone numbers in a certain way, or the website will not allow a email address that does not contain a @ symbol. These forms are great for creating uniform data, that is held to the same standard. Going back to the phone number example, some people may enter a phone number xxx-xxx-xxxx, while others, 1+xxx-xxx-xxxx, and others xxxxxxxxxx. When we have only one way of entering the data, and only one format, you will not get data that looks like the following:

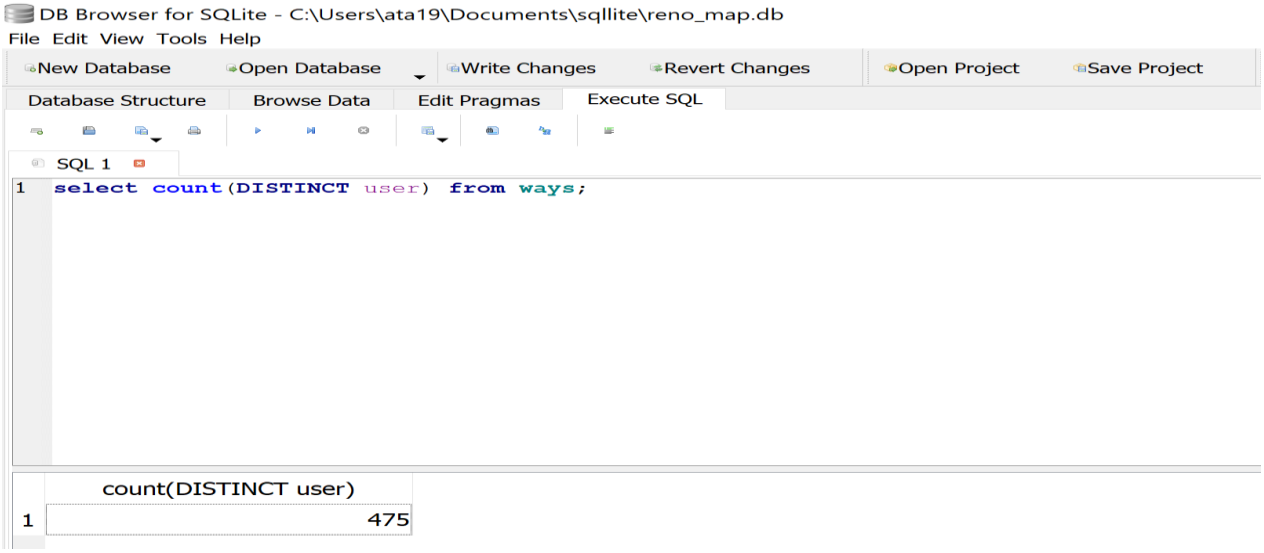


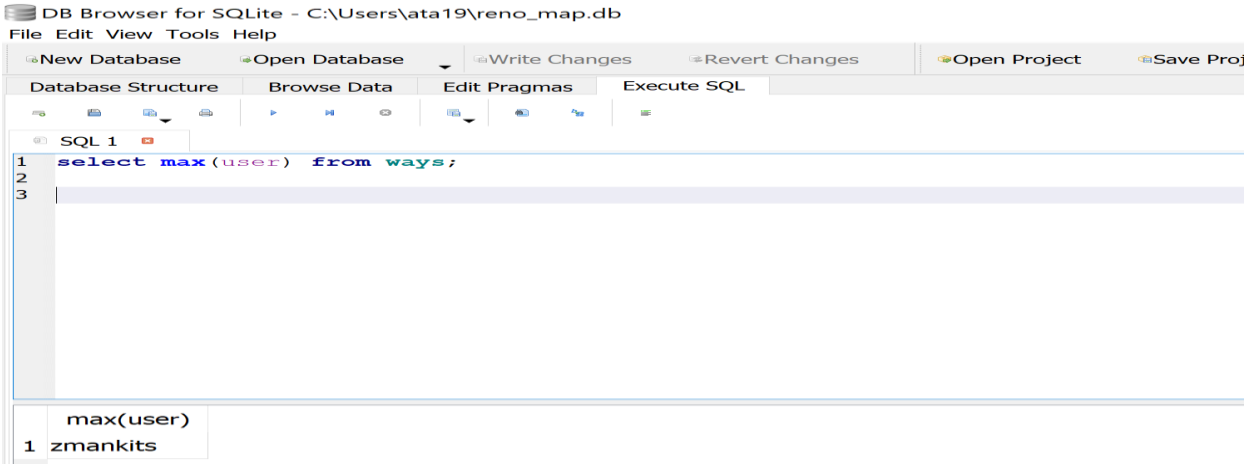
I have not worked with form validation, with any addition scripts, there is always a chance that this could cause a problem. But I believe any problems produced from a validator would be worth it to have cleaner data to use and wrangle.

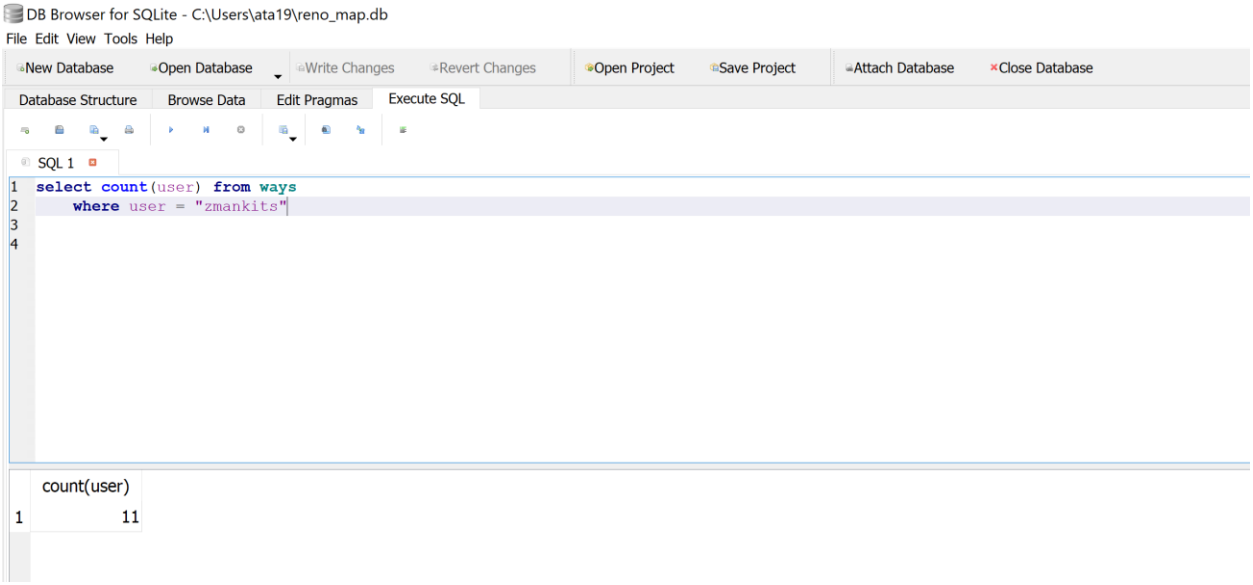
DB SQL Statistics:

```
select count(user)
from nodes1;
```

	count(user)
1	122110






DB Browser for SQLite - C:\Users\ata19\reno_map.db
File Edit View Tools Help
New Database
Open Database
Write Changes
Revert Changes
Open Project
Save Project
Attach Database
Close Database
Database Structure
Browse Data
Edit Pragmas
Execute SQL
SQL 1
1 select count(user) from ways
2 where user = "zmankits"
3
4
count(user)
1 11