Wrangling OpenStreetMap data using SQL

By: Sunda Gerard

April 2018

Map Area

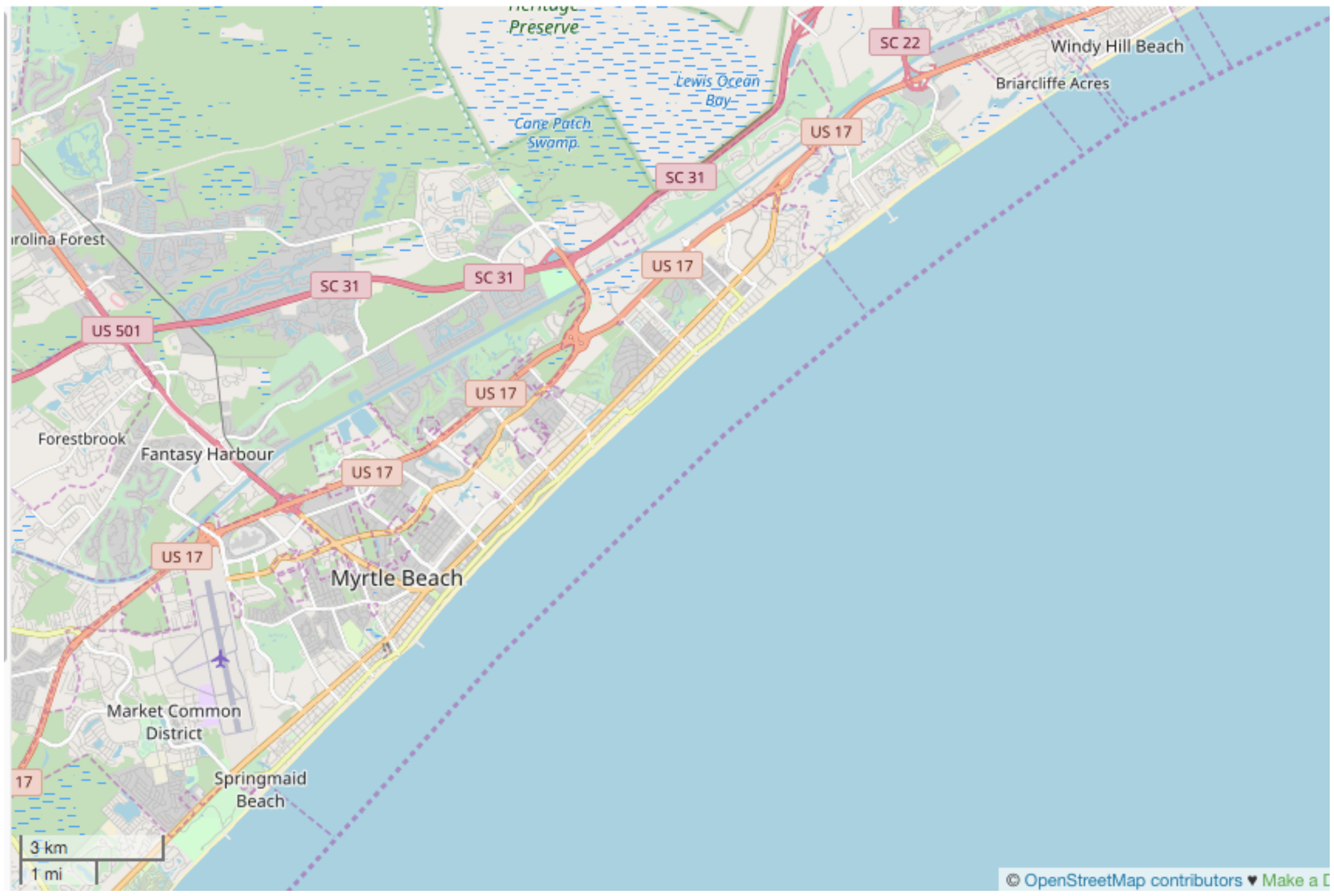
I chose the map area for Myrtle Beach, South Carolina. I was not able to choose my home area of Morgantown, West Virginia due to it being too sparsely populated and spread out for meeting requirements of the size and scope of the project. I had considered other locations, but chose Myrtle Beach due to my familiarity with the location due to vacationing there many times on beach trips.

Here is the link to the OpenStreetMap data on the Myrtle Beach, SC location: https://www.openstreetmap.org/#map=12/33.7192/-78.8818 (https://www.openstreetmap.org/#map=12/33.7192/-78.8818)

```
In [4]:
```

```python
from IPython.display import Image
Image(filename="MBmap.png", width=600, height=600)
```

```
Out[4]:
```



Here is the osm file = MyrtleBeach.osm

And a small sample osm file = MBsample.osm

Here are the coordinates for the map area: -78.9756 to -78.6676, 33.8008 to 33.6375.

Data was taken from OpenStreetMaps for the Myrtle Beach, SC location. The data was in an XML file format. I used iterative parsing to convert the data to CSV files. The following information was obtained from the mapparser.py file that outputted various information regarding the dataset using Python programming:

def test(): tags = count_tags('MyrtleBeach.osm') pprint.pprint(tags)

Returned the results:

{'bounds': 1, 'member': 8733, 'meta': 1, 'nd': 327539, 'node': 272653, 'note': 1 'osm': 1, 'relation': 218, 'tag': 90058, 'way': 26056}

Problems with the data

Problem Characters

During auditing of the dataset in Python using the tags.py file, the following data was collected on tags:

{'lower': 57713, 'lower_colon': 29676, 'other': 2669, 'problemchars': 0}

This is a good sign, as there are no problem characters indicated by the auditing.

Street Corrections

Based on our mapping in the audit.py file, there were only six street corrections that needed to be made:

707-Connector Rd => 707-Connector Road Cannon Rd => Cannon Road Highway 17 S => Highway 17 South Hwy. 17 N => Hwy. 17 North N Kings Hwy => N Kings Highway Kings Hwy => Kings Highway

We can conclude that the dataset is actually pretty clean as-is and did not need much updating. We are now ready to move the data into SQL and start running queries on the data to see what interesting things we can find!

Moving data to SQL

In preparing the SQL database, I parsed the contents of the OSM XML file into a CSV file using the Python programming language and using the file data.py. The data.py file outputs the data into five separate CSV files:

Way: Incorporates way attributes at the top level. The columns of searchable data within this table of the database are: id, user, uid, version, changeset, timestamp.

Way_nodes: These are the way child tag attributes. The columns of searchable data within this table of the database are: id, node_id, position.

Way_tags: These are way secondary tag attributes. The columns of searchable data within this table of the database are: id, key_value, type.

Node: Incorporates node attributes at the top level. The columns of searchable data within this table of the database are: id, lat, lon, user, uid, version, changeset, timestamp.

Node_tags: These are node secondary tag attributes. The columns of searchable data within this table of the database are: The columns of searchable data within this table of the database are: id, key_value, type.

Overview of the data

I used PostgreSQL to do analysis of the data and build the database. The following are the CSV files that were imported into PostgreSQL and their file sizes.

Ways.csv - 1.43 MB

Ways_nodes.csv - 7.28 MB

Ways_tags.csv - 2.37 MB

Nodes.csv - 21.1 MB

Nodes_tags.csv - 582 KB

```
SELECT COUNT (*) FROM node
```

Returned a count of "272653"

```
SELECT COUNT (*) FROM node_tags
```

Returned a count of "15972"

```
SELECT COUNT (*) FROM way
```

Returned a count of "26056"

```
SELECT COUNT (*) FROM way_nodes
```

Returned a count of "327539"

```
SELECT COUNT (*) FROM way_tags
```

Returned a count of "72692"

```
SELECT COUNT(*) AS num FROM node_tags WHERE node_tags.value = 'school';
```

There are 42 schools in the Myrtle Beach region

```
SELECT COUNT (DISTINCT(e.uid)) FROM (SELECT uid FROM node UNION ALL SELECT uid FROM way) e;
```

There are 309 unique users who have contributed to the database information

```
SELECT uid, COUNT (uid) as num FROM way GROUP BY uid ORDER BY num DESC LIMIT 10;
```

To find the top 10 contributors by user id:

```
23999 "13900" 1408522 "9564" 4176310 "4040" 175600 "3928" 38487 "2522" 451693 "2492" 2859402
"1416" 402624 "1138" 2735864 "958" 481564 "848"
```

After investigation into the user id data, the top 10 usernames in contributions to the ways data were in order: be9110, Omnific, TheMBDude, grossing, jumbanho, bot-mode, Sarr_Cat, bdiscoe, croniox88, and meflaboy.

```
SELECT value, COUNT(*) AS num FROM node_tags WHERE node_tags.key = 'amenity' GROUP BY value
ORDER BY num DESC LIMIT 10;
```

To find the top 10 amenities in the database:

```
"restaurant" "187" "place_of_worship" "115" "fast_food" "91" "fountain" "74" "fuel" "72" "school" "42"
"bank" "30" "toilets" "29" "fire_station" "27" "cafe" "19"
```

Restaurant is the top amenity with 187 instances, followed by place of worship with 115 instances. Interestingly, as well as unintentionally funny is that toilets are in the top 10 of amenities. I am curious why this isn't represented as restrooms or other tag that would be more appropriate. Also, I am not sure what exactly a "fountain" actually is supposed to represent, other than an opportunity to drink from a water fountain.

```
SELECT value, COUNT(*) AS num FROM node_tags WHERE node_tags.key = 'cuisine' GROUP BY value
ORDER BY num DESC LIMIT 10;
```

The top restaurant or cuisine listed is burger places, followed by pizza. It is obvious that all 187 restuarants found in the previous query are not represented in this category since the top restaurant only has 28 instances.

```
"burger" "28" "pizza" "9" "chicken" "7" "sandwich" "5" "coffee_shop" "5" "mexican" "5" "regional" "3"
"american" "3" "donut" "2" "sushi" "1"
```

Suggestions for improvement to the dataset

Due to the dataset being fairly clean, the only information that will need to be looked at would be any changing information or new information (such as new restaurants or businesses). It would be nice to get notificiations on OpenStreetMap when these take place, so that they could be verified and checked for acccuracy. I'm not sure how that could take place unless some kind of agreement could be in place. Unfortunately, since the database is open source and community driven, this is highly unlikely. Websites such as Wikipedia are community driven and backed by a staff of employees that verify data, so this may be a possibility at some point.

Another suggestion to improve the data would be to fill in the gaps in the dataset. An example of this would be the previous query on restuarants. Out of 187 restuarants, only a portion of those were actually described. Someone could put that information into the data to make it more complete. Also, even though restuarants come and go quite frequently in the Myrtle Beach area, it seems that there should be more than 187 restuarants. It seems likely that there are some restaurants in the area that are not represented at all in the database.

References

stackoverflow.com/questions/3348460/csv-file-written-with-python-has-blank-lines-between-each-row

discussions.udacity.com/t/exporting-to-csv-problem-extra-letter-b-in-output/223928/9

discussions.udacity.com/t/schemas-for-creating-database/170298/10

oslandia.com/en/2017/07/03/openstreetmap-data-analysis-how-to-parse-the-data-with-python/

gist.github.com

github.com

https://youtu.be/xlD8FIM5biA (https://youtu.be/xlD8FIM5biA)