OpenStreetMap Data Wrangling

Map Area

Delaware Beaches, DE, United States

Map

- https://www.openstreetmap.org/export#map=11/38.6381/-75.0675

How to download this area

- http://overpass-api.de/query_form.html

Use this query

- (node(38.4740,-75.3951,38.8012,-74.7400);<;);out meta;

This map is of the part of Delaware along the Atlantic Ocean. Sea level rise from ice melt will likely erase this area in the next 50 years so this map won't be valid then.

The project uses python 2.7 and sqlite3. Queries below and from the "sql" file called make_temp_tables_and_import_csvs.sql were done on the command line.

Problems Encountered in the Map

- Street name abbreviations are "cleaned up" and changed to be spelled out as whole words.
- I really want to show how popular beach fries are as a whole cuisine here, but the fries stands are lumped under fast-food as an amenity instead of restaurants with a specific cuisine
- Nothing to "fix" with the zipcodes, they're all uniform and correct. There's some zip code data laid out though below since that seems like a common thing to work on.
- Going from xml to sqlite took eleventy billion years so that's a big portion of this readme.

Running an audit on street names found some abbreviations. A sample of results is below. The most common abbreviations found were Ln, St, and Blvd.

```
'Alley': set(['Hudson Alley',
              'Neils Alley',
              'Pond Alley',
              'Shovelhead Alley',
              'Truitt Alley']),
 'Bluff': set(['Ocean Bluff']),
 'Blvd.': set(['Edwards Blvd.']),
 'Boardwalk': set(['North Boardwalk', 'South Boardwalk']),
 'Branch': set(['Short Branch']),
 'Circle': set(['Beacon Circle',
                'Bridle Reach Circle',
                'Campbell Circle',
                'Captains Circle',
                'Comanche Circle',
                'Coverdale Circle',
                'Crist Circle',
                'Doury Circle',
                'Dune Circle',
                'East Atlantic Circle',
                'Halls Heritage Circle',
                'Harbour Circle',
                'Houston Circle',
                'Indian Meadows Circle',
                'Kyle Circle',
                'Lakewood Circle',
                'Long Neck Circle',
                'Magnolia Circle',
                'Marina Bay Circle',
                'Mill Chase Circle',
```

```
                'Nash Circle',
                'Patrick Henry Circle',
                'Pine Run Circle',
                'Samuel Adams Circle',
                'Springwood Circle',
                'The Circle',
                'Villa Circle',
                'Vireo Circle',
                'Woodlake Circle']),
 'Corner': set(['Caitlans Corner']),
 'Cout': set(['Creek View Cout']),
 'Cove': set(['Breezy Cove',
              'Mimosa Cove',
              'Misty Cove',
              'Mooring Cove',
              'Paradise Cove',
              'Peaceful Cove',
              'Rock Cove',
              'Sandcastle Cove',
              'Tern Cove']),
 'Crossing': set(['Moores Crossing', 'Pugs Crossing']),
 'DE-1': set(['DE-1']),
 'Dunbarton': set(['Dunbarton']),
 'Entrance': set(['Entrance']),
 'Esplanade': set(['Peninsula Esplanade']),
 'Extended': set(['Rehoboth Avenue Extended',
                  'Skipjack Extended',
                  'South Galley Extended',
                  'South Oak Drive Extended',
                  'South Shore Drive Extended',
                  'Union Street Extended',
                  'Washington Street Extended']),
 'Falls': set(['Berry Bramble Falls', 'North Berry Bramble Falls']),
 'Haven': set(['Hazelnut Haven']),
 'Highway': set(['Coastal Highway',
                 'John J Williams Highway',
                 'Kings Highway',
                 'Lewes Georgetown Highway',
                 'Millsboro Highway',
                 'Milton Ellendale Highway',
                 'North Dupont Highway',
                 'Wilson Highway']),
 'Hills': set(['Buttercream Hills',
               'North Cotton Patch Hills',
               'South Cotton Patch Hills']),
 'Hwy': set(['John J Williams Hwy']),
 'Knoll': set(['Honeysuckle Knoll']),
 'Landing': set(['Herring Landing',
                 'Pembroke Landing',
                 'Roanoke Rapids Landing']),
 'Ln': set(['Branch View Ln',
```

```
            'Cedar Ln',
            'No Name Ln',
            'Pine Ln',
            'Salty Dog Ln']),
 'Loop': set(['Alderwood Loop',
              'Bethany Loop',
              'Draper Loop',
              'Sawyer Loop',
              'Seaview Loop']),
 'Mews': set(['Rehoboth Mews']),
 'One': set(['Highway One']),
 'Pass': set(['Tecumseh Pass']),
 'Path': set(['Brookstone Path',
              'Fowlers Path',
              'Kayakers Path',
              'Sunburst Path',
              'Swirling Waters Path']),
 'Plaza': set(['Georgetown Plaza', 'Ocean One Plaza']),
 'Point': set(['Fishers Point',
               'Heron Point',
               'Pier Point',
               'Pilot Point',
               'Racoons Point']),
 'Ranch': set(['Dakotas Ranch']),
 'Reach': set(['Harvest Run Reach']),
 'Row': set(['King Street Row', 'Pusey Row']),
 'Run': set(['Deer Run',
             'Fox Run',
             'Meadow Run',
             'Mermaid Run',
             'Millers Run',
             'Pine Run']),
 'St': set(['Goff St', 'N Race St']),
 'Strip': set(['Delberts Strip']),
 'View': set(['Endless View']),
 'Village': set(['Cannery Village']),
 'Walk': set(['Navigators Walk', 'Wilson Walk'])
```

here's part of the clean_streets_audit.py file that finds and cleans abbreviations out of street names. Before and after versions of the OSM files are available in zip files in this repository.

```
def update_name(name, mapping):

    for key, value in mapping.iteritems():
        if re.search(key, name):
            name = re.sub(street_type_re, value, name)
```

```
    return name
```

Once I finally got CSVs created, they had blank lines after every line so I deleted them all and adjusted the csv writer lines like so:

```
131 codecs.open(NODES_PATH, 'wb') as nodes_file
(the 'wb' was originally 'w' but windows gets weird with that for some reason)
```

While trying to insert form the CSVs into the sqlite3 tables, I got a datatype mismatch error. My pile of code didn't take my encoding as utf-8 advice when I tried to add that to lines 131 and friends in the delaware_beach.py file and had a bunch of errors so I made temp tables then forced everything into the real tables sideways:

```
CREATE TABLE nodes (
    id INTEGER PRIMARY KEY NOT NULL,
    lat REAL,
    lon REAL,
    user TEXT,
    uid INTEGER,
    version INTEGER,
    changeset INTEGER,
    timestamp TEXT
);

CREATE TABLE nodes_temp (
    id TEXT,
    lat TEXT,
    lon TEXT,
    user TEXT,
    uid TEXT,
    version TEXT,
    changeset TEXT,
    timestamp TEXT
);

.mode csv
.import nodes.csv nodes_temp

INSERT INTO nodes
SELECT CAST(id AS INTEGER),
    CAST(lat AS REAL),
    CAST(lon AS REAL),
    CAST(user AS TEXT),
```

```
    CAST(uid AS INTEGER),
    CAST(version AS INTEGER),
    CAST(changeset AS INTEGER),
    CAST(timestamp AS TEXT)
FROM nodes_temp;

SELECT * FROM nodes
LIMIT 10;

DROP TABLE nodes_temp;
```

This seems less impressive in hindsight, but took me days to figure out. I did a version of this for each table.

Postal Codes

Here are the postal codes that exist on this map. Nothing to see here. It's a small state.

- Postal codes when grouped together with this aggregator:

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags
          UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key='postcode'
GROUP BY tags.value
ORDER BY count DESC;
```

- All query results pretttied up with tabs to make them human friendly

```
value  count
19966   8257
19958   2008
19930   1138
19939    706
19971    701
19947    681
19968    358
19970    237
19975    188
19967     91
19951     87
19945     30
19944      8
```

If these postal codes did have 9 digits, I'd hit them with one of these:

```
UPDATE nodes_tags
SET value=SUBSTR(value,0,5)
WHERE key='postcode' AND length(value)>5;
```

Sort cities by count, descending

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key = 'city'
GROUP BY tags.value
ORDER BY count DESC;
```

```
City,                   Count

Millsboro,              8173
Lewes,                  1995
Rehoboth Beach,          777
Bethany Beach,           751
Dagsboro,                706
Georgetown,              681
South Bethany,           379
Milton,                  357
Millville,               247
Selbyville,              188
Dewey Beach,             108
Long Neck,                89
Harbeson,                 87
Ocean View,               86
Frankford,                31
Fenwick Island,            7
milton,                    3
Clarksville,               1
North Bethany Beach,       1
```

Clarksville looks like an error, but it does exist even though only 3 people live there and nobody really knows where it is.

North Bethany Beach appears to be a mistake, but unfortunately does exist as lump of billion dollar ocean-front homes that are located on a strip of land that dissapears 3 times a year but nobody cares because

the people that own these houses can afford helicopters full of dirt to be flown in and houses to be rebuilt every couple years.

Milton and milton are the same place so I'll fix that capitalization issue here:

```
UPDATE ways_tags
SET value='Milton'
WHERE value='milton';
```

and now after running:

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key = 'city'
GROUP BY tags.value
ORDER BY count DESC;
```

we get:

Millsboro, 8173 Lewes, 1995 Rehoboth Beach, 777 Bethany Beach, 751 Dagsboro, 706 Georgetown, 681 South Bethany, 379 Milton, 360 Millville, 247 Selbyville, 188 Dewey Beach, 108
Long Neck, 89 Harbeson, 87 Ocean View, 86 Frankford, 31 Fenwick Island, 7 Clarksville, 1 North Bethany Beach, 1

Data Overview

File sizes

```
delaware_beach.osm .... 108.72 MB
rehoboth_map.db ....... 137.28 MB
nodes.csv ..............  41.72 MB
nodes_tags.csv ........   0.55 MB
ways.csv ..............   2.71 MB
ways_tags.csv .........  15.24 MB
ways_nodes.cv .........   6.14 MB
```

Number of nodes

```
sqlite> SELECT COUNT(*) FROM nodes;
```

512720

Number of ways

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

46830

Number of unique users

```
SELECT COUNT(DISTINCT(e.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
```

351

Top 10 contributing users

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

User Count

```
tlt83                286794
RyanSta80             91129
ElliottPlack          70451
bmillman85            13574
RoadGeek_MD99         11997
"danny fish"           5622
ceyockey               5309
dchiles                5158
JmanVT                 4873
losthiker              4355
```

Number of users appearing only once (having 1 post)

```
SELECT COUNT(*)
FROM
    (SELECT e.user, COUNT(*) as num
     FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
     GROUP BY e.user
     HAVING num=1)  u;
```

59

Additional Data Exploration

Top 10 appearing amenities (or a brief explanation why there's nowhere to park)

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

Amenity Count

restaurant	92
fountain	91
bench	53
fast_food	27
bicycle_parking	16
toilets	16
ice_cream	10
cafe	7
parking	7
bicycle_repair_station	6

Most popular cuisines

How much of the fast_food is actually beach fries? Again, this had to be dragged from

the names because potatoes aren't real food.

```
SELECT value, COUNT(*) as num
FROM nodes_tags
WHERE value LIKE '%FRIES' OR value='Gus & Gus';
```

8

Other food that exists here. Of 92 restaurants, only 43 have a specific type of cuisine. Ice cream is somehow both an amenity and a type of cuisine.

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='restaurant') i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY num DESC;
```

```
pizza                  11
chinese                 8
seafood                 6
american                4
ice_cream               4
mexican                 4
italian                 2
bistro                  1
caribbean               1
diner                   1
french                  1
pizza;american          1
steak_house             1
sushi                   1
thai                    1
wings                   1
```

Conclusion

Come for the beach fries, stay for the mansions. Or the other way around. Something.