Wrangle OpenStreetMap Data

For this project, I will analyze OpenStreetMap data in the United States. First, I will audit the dataset to find out if there are any problems that need to be fixed. Next, I will use SQL queries to obtain an overview of the dataset. Last, I will provide some ideas to further improve the dataset.

Data Source Information

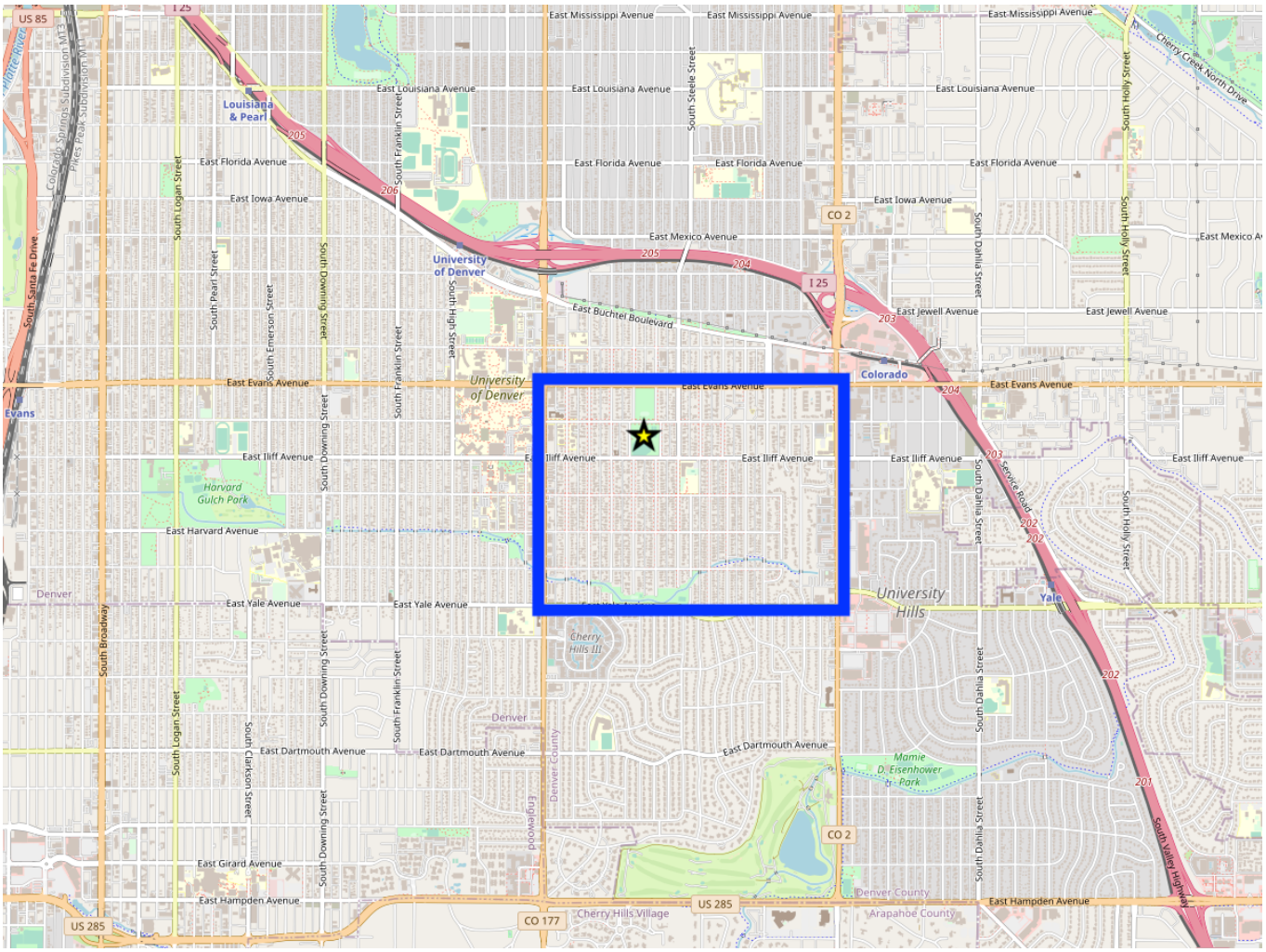
Using openstreetmap.org and the Overpass API, I extracted openstreetmap data from the geographical area surrounding the Chamberlin Observatory. This is a unique observatory that is located near Denver's city center. I love astronomy and the lawn area near the telescope is lovely in autumn.

The sample area is within the boundaries of the blue square. The observatory is marked with a yellow star.

map.osm URL: https://www.openstreetmap.org/#map=14/39.6739/-104.9538

sample.osm Coordinates: 39.6712/-104.9208

Location: Chamberlin Observatory, Denver, Colorado, USA



Problems Encountered

I audited and cleaned the sample.osm and map.osm street names using a modified version of the audit.py submission from the Case Study exercise:

Example from sample.osm converted to xml

<node id="2859304700" visible="true" version="3" changeset="86639096" timestamp="2020-06-15T01:38:53Z" user="bhietsch" uid="5390272" lat="39.6690197" lon="-104.9405544">
 <tag k="bus" v="yes"/>
 <tag k="highway" v="bus_stop"/>
 <tag k="local_ref" v="RTD 40 46"/>
 <tag k="name" v="S Colorado Blvd & E Yale Ave"/>
 <tag k="public_transport" v="platform"/>
 <tag k="source" v="Bing"/>
</node>

Cleaned sample.osm in node_tags.csv

id	key	value
2859304700	name	S Colorado Boulevard & E Yale Avenue

Cleaned map.osm excerpt from **audit.py** output:

South Broadway 130 => South Broadway 130
South Broadway Building 2 => South Broadway Building 2
East Colorado Avenue Apt 1 => East Colorado Avenue Apt 1
South Broadway Building 1 => South Broadway Building 1
South Acoma Street Building 1 => South Acoma Street Building 1
South Lincoln Street 1 => South Lincoln Street 1
South Cherokee Street A => South Cherokee Street A
South Broadway Road A => South Broadway Road A
South Delaware Street Unit A => South Delaware Street Unit A
South Lincoln Street Apt B => South Lincoln Street Apt B

Observation 1:
After running the audit on map.osm, I observed multiple addresses that included suite or unit numbers. The audit script handled the scenario, but this is problematic from an overall consistency perspective.

Observation 2:
In the sample.osm csv for nodes_tags, I confirmed that the addresses were properly cleaned for the addr:street and names keys. During the audit, I discovered that the alt_name key also contains addresses that could be cleaned. The alt_name values are inconsistent. Business names, street names, and random abbreviations are included. Due to the irregularities, cleaning alt_name is not useful at this time.

Here is an example of the sample.osm street names in alt_name:

key	value
alt_name	**E Yale Ave & S Colorado Blvd**
alt_name	University and Yale
alt_name	Yale and Clayton
alt_name	Yale and Clayton
alt_name	University and Yale
alt_name	University and Yale

Preparing the Data

Using the data.py and schema.py code, I cleaned and converted the full dataset, map.osm, to five .csv files. Then, I created a database in SQLite Studio with a table for each csv file. SQL queries were used to further explore the data.

Overview of the data

Statistical overview

File Sizes

File Name	Size
map.osm	95.5 mb
nodes.csv	37.2 mb
nodes_tags.csv	1.4 mb
ways.csv	3.1 mb
ways_nodes.csv	10.9 mb
ways_tags.csv	8.1 mb
sample.osm	8.4 mb

Total rows in each table:

```
SELECT 'nodes' AS table_name, COUNT(*) FROM nodes
UNION
SELECT 'nodes_tags' AS table_name, COUNT(*) FROM nodes_tags
UNION
SELECT 'ways' AS table_name, COUNT(*) FROM ways
UNION
SELECT 'ways_nodes' AS table_name, COUNT(*) FROM ways_nodes
UNION
SELECT 'ways_tags' AS table_name, COUNT(*) FROM ways_tags;
```

table_name	COUNT(*)
nodes	397403
nodes_tags	31947
ways	45362
ways_nodes	450413
ways_tags	232628

Number of unique users who contributed to the dataset

```
SELECT COUNT ( DISTINCT uid)
FROM nodes;
```

```
442
```

Top 10 users who contributed to the dataset

```
SELECT user, COUNT(*) AS count
FROM nodes
GROUP BY uid
ORDER BY count DESC
LIMIT 10;
```

User	Count
omgitsgela_imports	119640
Your Village Maps	81199
fishcharlie_imports	40909
mspyker_imports	28026
greenorangeparrot_imports	19822
luisluigi639_imports	17868
Van_imports	15705
chachafish	13854
omgitsgela	11223
samuelestabrook_imports	6078

Types of amenities

```
SELECT tags.value
FROM (SELECT * FROM nodes_tags
UNION ALL
SELECT * FROM ways_tags) tags
WHERE tags.key = 'amenity'
GROUP BY tags.value;
```

(54 total)

animal_boarding	dojo	post_office
arts_centre	drinking_water	pub

atm	fast_food	recycling
bank	fire_station	restaurant
bar	fountain	school
bbq	fuel	shelter
bench	hospital	studio
bicycle_parking	ice_cream	taxi
bicycle_rental	kindergarten	telephone
bicycle_repair_station	library	theatre
cafe	nightclub	toilets
car_rental	office	university
car_wash	parking	vending_machine
charging_station	parking_entrance	veterinary
cinema	parking_space	waste_basket
clinic	pharmacy	waste_disposal
dentist	place_of_worship	
doctors	police	
doctors dentist & orthodontist	post_box	

Problem Exploration

Phone Numbers

```
SELECT id, key, value
FROM nodes_tags
WHERE key = 'phone'
GROUP BY id
ORDER BY id DESC
LIMIT 10;
```

Id	Key	value
358921058	phone	720-747-2400
358921066	phone	720-424-0960
358921089	phone	+1-303-757-7727
358921093	phone	303-759-2076
358921108	phone	+1 303-777-3812
358921178	phone	303-733-2421
358921193	phone	(303) 744-1069
358949945	phone	303-761-8882
358950020	phone	(303) 758-5900
358950305	phone	303-777-7638

There are 425 phone numbers in the nodes_tags table. I included the first 10 that were returned in the query. We can see that the values have been entered in disparate formats. Additional cleaning could be performed to improve the consistency of these values.

Zip Codes

Let's count the unique zip codes, group by the zip code value, then sort in descending order.

```
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;
```

postcode	count
80210	9863
80222	4392
80231	689
80223	613
80113	540
80224	293
80237	123
80110	106
80209	105
80246	35
80247	31
80208	6
80206	3
80247-2121	2
80218	2
80210-2938	2
90222	1
80211	1
80210;80222	1
80205	1
80123	1
80113-1525	1
80022	1

We can see that there are 23 unique postcode tags, but 4 tags are incorrectly formatted. Additional cleaning could be performed to improve the consistency of these values. It is also peculiar that the postal codes vary widely for a small area of Denver.

Amenities

I scream, you scream, we all scream for data! There is an ice cream shop near the observatory.

```
SELECT id, key, value
FROM nodes_tags
WHERE key = 'amenity' AND value = 'ice_cream';
```

id	key	value
631845194	amenity	ice_cream
2784760103	amenity	ice_cream

I was wrong. There are TWO ice cream shops. How will we decide which one to visit?

```
SELECT *
FROM nodes_tags
WHERE key = 'brand' AND (id = '631845194' OR id = '2784760103');
```

id	key	value	type
631845194	brand	Ben & Jerry's	regular
2784760103	brand	Cold Stone Creamery	regular

Cold Stone and Ben & Jerrys. Tough decision, but I'm going to Ben & Jerry's. I hope Udacity reviewers like Cherry Garcia.

Other Ideas About the Dataset

During the audit, I noticed that the dataset was derived from multiple sources. Here are the top 10:

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags
UNION ALL
SELECT * FROM ways_tags) tags
WHERE tags.key = 'source'
GROUP BY tags.value
ORDER BY count DESC
LIMIT 10;
```

value	count
DRCOG PLANIMETRICS DATA	25210
Bing	9764
bing	1446
DigitalGlobe-Premium	934
yahoo	819
Yahoo	189
knowledge	114
Mapbox	97
EsriWorldImagery	59
solar	42

That is odd. Let's focus on the Bing entries.

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

values	count
Bing	9764
bing	1446
bing;DRCOG PLANIMETRICS DATA	1
bing;Bing	1
Bing;knowledge	1
Bing;gps	1
Bing;bing	1
Bing;aerial imagery;streetlevel imagery	1
Bing;Google	1
Bing;DRCOG PLANIMETRICS DATA	1
BING	1

Search engine data is gathered by multiple users using processes that may combine information from more than one source. Also, it appears that the data is not strictly text. Aerial imagery data is included. This is open-source data. It may be difficult to strictly enforce the formatting.

Suggestion:

Enforce a standard set of values for data source types

Benefits:

If the source values were consistent, openstreetmap users would be able to perform an analysis of the data quality by data source. This would be an excellent method for understanding which search engines or open source projects provide the best data for specific types of tags.

Anticipated Problems:

Data may not always be sourced from common or well-known websites like Google, Bing, or Yahoo. For example, “DRCOG PLANIMETRICS DATA” is a regional high-resolution aerial imagery project for Denver. This source would not be utilized for other locations outside of Colorado.

Conclusion

Overall, this dataset is surprisingly clean considering the volume of information and number of unique contributors. There are some areas of inconsistency, like zip codes, phone numbers, and data sources that would benefit from standardization.

There is an opportunity to add more data to explain the reason for varied zip codes. This is a relatively small area in Denver's city center. From a geographic perspective, all postal codes should be the same. Perhaps the zip codes are referencing corporate offices. It is not possible to confirm this with the current dataset.

This analysis was fun! My family was happy to visit Ben & Jerry's with me for a data-driven treat.

Sources

OpenStreetMap
https://www.openstreetmap.org/

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

SQL Schema Provided by Udacity
https://gist.github.com/swwelch/f1144229848b407e0a5d13fcb7fbbd6f

Sample Project Provided by Udacity
https://gist.github.com/carlward/54ec1c91b62a5f911c42#file-sample_project-md

Python.org: The ElementTree XML API
https://docs.python.org/3/library/xml.etree.elementtree.html#module-xml.etree.ElementTree

w3schools.com: Python String split() Method
https://www.w3schools.com/python/ref_string_split.asp

Python 3.0 Built-In Changes. Modified Udacity data.py code with Python 3.9 iterator
https://wiki.python.org/moin/Python3.0#Built-In_Changes

Denver Regional Council of Governments: Regional Planimetric Data Project
https://drcog.org/services-and-resources/data-maps-and-modeling/regional-planimetric-data-project