# OpenStreetMap Data Case Study

## Map Area

Denver & Boulder, CO, United States

* <https://mapzen.com/data/metro-extracts/metro/denver-boulder_colorado/>

This map is of my hometown Louisville, CO (right outside Denver) as well as two cities Boulder and Denver that I have lived in. I’m interested in the opportunity to contribute to its improvement on OpenStreetMap.org.

## Explore the OSM Data

After initially downloading the Denver/Boulder OSM file, I found it cumbersome to work with such a large data set to explore, so I wrote the **‘sample\_output\_osm.py’** script to output a sample file of the data that was 1/100th the size. I named the file **denver\_sample.osm**, but to clarify the sample data includes samples from both Boulder, Denver, and the metro areas. The abbreviated file name was chosen for ease of typing.

Next, I started to explore the data and here’s where I focused in the following order:

* Understanding the ‘Elements’ in the OSM File
* Understanding the ‘Tags’ in the OSM File
* Exploring and Auditing Street Names
* Exploring and Auditing Postal Codes
* Understanding Users

### Understanding the ‘Elements’ in the OSM File

A key data concept in OSM is **Elements**.

Elements are the basic components of OpenStreetMap's conceptual data model of the physical world. The core Elements of OSM are:

(\*\*\*Definitions and Examples taken directly from the OSM Documentation\*\*\*)

#### **Node:** Consists of a single point in space defined by its latitude, longitude and node id.

Example:

<node id="25496583" lat="51.5173639" lon="-0.140043" version="1" changeset="203496" user="80n" uid="1238" visible="true" timestamp="2007-01-28T11:40:26Z">

<tag k="highway" v="traffic\_signals"/>

</node>

#### **Way**: An ordered list of nodes which normally also has at least one tag or is included within a **Relation**.

A way can have between 2 and 2,000 nodes, although it's possible that faulty ways with zero or a single node exist.

Example:

<way id="5090250" visible="true" timestamp="2009-01-19T19:07:25Z" version="8" changeset="816806" user="Blumpsy" uid="64226">

<nd ref="822403"/>

<nd ref="21533912"/>

<nd ref="821601"/>

<nd ref="21533910"/>

<nd ref="135791608"/>

<nd ref="333725784"/>

<nd ref="333725781"/>

<nd ref="333725774"/>

<nd ref="333725776"/>

<nd ref="823771"/>

<tag k="highway" v="residential"/>

<tag k="name" v="Clipstone Street"/>

<tag k="oneway" v="yes"/>

</way>

#### **Relations**: Consists of one or more tags as well as an ordered list of more nodes, ways, and/or relations as members, which is used to define logical or geographic relationships between other elements.

#### **Tags**: All of the elements above can have one or more associated Tags which describe specific features of the map elements.

Tags consists of two items, a **key** and a **value**.

Example:

<tag k="highway" v="residential"/>

<tag k="name" v="Clipstone Street"/>

<tag k="oneway" v="yes"/>

Now to explore what Elements we have in the Denver/Boulder OSM data and to see if we have any extra tags, I've written the **‘explore\_elements.py'** script.

The **'explore\_elements.py'** script will count the different elements we will be seeing in the OSM File, here is the output:

{'bounds': 1,

'member': 39294,

'nd': 4740351,

'node': 4109077,

'osm': 1,

'relation': 2137,

'tag': 2230326,

'way': 460309}

It's important to note that in addition to our 4 core OSM Elements (Nodes, Ways, Relations, Tags) we have Members and Nd's in our output.

To clarify those elements:

Nd: Are tags in Ways that reference certain Nodes.

Member: Describes a particular feature within a Relation Element.

The 'bounds' and 'osm' Elements are headers for the osm file.

### Understanding the ‘Tags’ in the OSM File

As we can see Tags are one of our core Elements and from looking at the OSM file they appear to store a lot of the information about the map. In order to get more information about tags, I wrote a script **'explore\_tag\_types.py'** to inspect and count the 'tag' elements to see what we’re dealing with.

The output is too large to place in this document but it’s interesting to note the range of tags that we will be dealing with if you run the program.

Also from the results we see that the highest counts appear to be streets, postcodes, and city values.

Let’s dive in to a few of these tags to see if we can clean some of this data.

### Exploring and Auditing Street Names

From looking at the tags, one of the most important tags is street. However, from a quick look at the data it looks like streets come in all sorts of abbreviations and formats.

We’re going to want to clean this.

To do this I created the **"explore\_street\_names.py**", which takes an expected list of street formats (see below) and compares the values in the Street Name tags.

expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Square", "Circle", "Lane", "Road", "Trail", "Parkway", "Way"]

The output of the “explore\_street\_names.py” shows the Street Names that don’t adhere to the expected format:

{'2': set(['Colorado SH 2']),

'287': set(['US Highway 287']),

'74': set(['Highway 74']),

'800': set(['W 92nd Ave #800']),

'Ave': set(['E Maplewood Ave']),

'Baseline': set(['Baseline']),

'Blvd': set(['Colorado Blvd', u'Pe\xf1a Blvd']),

'Broadway': set(['Broadway']),

'Colfax': set(['East Colfax']),

'Ct': set(['North 57th Ct']),

'Dr': set(['Community Circle Dr']),

'East': set(['Sterling Circle East']),

'Mall': set(['Central Campus Mall', 'Pearl Street Mall']),

'Perdido': set(['Camino Perdido']),

'Pl': set(['W 16th Pl']),

'Rd': set(['E Arapahoe Rd']),

'St': set(['S Columbine St', 'S Elizabeth St', 'S Pearl St', 'Wright St']),

'West': set(['Park Avenue West', 'South Carr Avenue West'])}

As we can see ‘Street’ is often times abbreviated as ‘St’, Avenue as ‘Ave’, etc.

Let’s keep this in mind when we clean and output the data as we will want to make sure to audit this.

### Exploring and Auditing Postal Codes

Another tag that seems to be prevalent is ‘postcodes’, let’s explore these tags and see if there’s some clean-up we can do.

To do this I wrote the **‘explore\_post\_codes.py’** to output postal codes that don’t fit into the 5-digit format. Here are the results:

['80214-1801', 'Golden, CO 80401']

As we can see it looks like the only post codes that are flagged were post codes that have the city included as well as the additional 4 digits added on at the end.

We’ll keep this in mind when we clean and output the data. Now onto users!

### Understanding Users

What’s fascinating about the Open Street Map data is that a lot of the data is volunteer generated. I’m a little curious to see what the user data looks like and see if there are lot of people contributing little amounts or if there are super users contributing large amounts.

To explore this, I wrote the script **‘explore\_users.py’** which outputs the UID’s with their contribution count.

The output is too big to show here, but it looks like we have a mix of users contributing only a few times and a few users contributing a massive amount to the data set. In our deriving insights section, later in the project, we’ll actually use SQL queries to find the most active users as well as see how many of the users have just been active once.

## Output the Data into CSV Format

Now, we need to clean and output the data to CSV’s which we can then use to import into an sqlite3 database.

For this I wrote the **“output\_csv.py”**, which takes in the OSM file and outputs CSV's for nodes, ways, node\_tags, way\_tags, way\_nodes. It also uses the scripts **‘update\_street\_names.py‘** and **‘update\_post\_codes.py’** to clean the street and postal code data which we talked about in the explore section above.

Once we have the CSV’s, we can import them into a SQLite3 database. I wrote the instructions **“csv\_to\_sqlite3.txt”**, which details this process.

Once the data is in our database, it’s time to run some queries to derive insights!

## Derive Insights through Queries

The fun part begins! We’ll be performing queries to derive insights from the data. Below are sections where we derive insights from the data.

### File Sizes

denver\_sample.osm – 9.2 MB

nodes.csv – 3.5 MB

nodes\_tags.csv – 128 KB

ways.csv – 282 KB

ways\_tags.csv – 647 KB

ways\_nodes.csv – 1.1 MB

### Number of Nodes

> SELECT COUNT(\*) FROM nodes;

*41,091*

### Number of Ways

> SELECT COUNT(\*) FROM ways;

*4,603*

### Number of Unique Users

> SELECT COUNT(DISTINCT(e.uid))

> FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) AS e;

*814*

### Top 10 Contributing Users

> SELECT e.user, COUNT(\*) AS num

> FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) AS e

> GROUP BY e.user

> ORDER BY num DESC

> LIMIT 10;

*1) "Your Village Maps" - 7076*

*2) chachafish - 6896*

*3) woodpeck\_fixbot - 3433*

*4) GPS\_dr - 3208*

*5) jjyach - 2607*

*6) DavidJDBA - 1846*

*7) Stevestr - 1718*

*8) CornCO - 1287*

*9) russdeffner - 1250*

*10) Berjoh - 865*

### Number of Users Appearing Only Once

> SELECT COUNT(\*)

> FROM

(SELECT e.user, COUNT(\*) AS num

FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) AS e

GROUP BY e.user

HAVING num=1) AS u;

*284*

### Top 10 Appearing Amenities

> SELECT value, COUNT(\*) AS num

> FROM nodes\_tags

> WHERE key='amenity'

> GROUP BY value

> ORDER BY num DESC

> LIMIT 10;

*1) restaurant - 14*

*2) bench - 12*

*3) bicycle\_parking - 9*

*3) cafe - 9*

*5) fast\_food - 7*

*6) place\_of\_worship - 6*

*7) fuel - 5*

*7) school - 5*

*9) atm - 4*

*9) toilets - 4*

### Biggest Religion

> SELECT nodes\_tags.value, COUNT(\*) AS num

> FROM nodes\_tags

> JOIN (SELECT DISTINCT(id) FROM nodes\_tags WHERE value='place\_of\_worship') > AS id

> ON nodes\_tags.id = id.id

> WHERE nodes\_tags.key = 'religion'

> GROUP BY nodes\_tags.value

> ORDER BY num DESC

> LIMIT 1;

*Christian - 6*

### Most Popular Cuisines

> SELECT nodes\_tags.value, COUNT(\*) AS num

> FROM nodes\_tags

> JOIN (SELECT DISTINCT(id) FROM nodes\_tags WHERE value='restaurant') AS id

> ON nodes\_tags.id=id.id

> WHERE nodes\_tags.key='cuisine'

> GROUP BY nodes\_tags.value

> ORDER BY num DESC;

* *American - 1*
* *Chicken - 1*
* *ice\_cream - 1*
* *mexican - 1*
* *pizza - 1*
* *sandwich - 1*
* *steak\_house - 1*

## Conclusion & Improvement

From our exploration of the data it’s obvious that the Denver & Boulder OSM areas are incomplete.

### How do we improve this data?

**Gamification!** Gamification is a proven way to incentivize users to behave correctly.

For the Open Street Map, we can incentivize users to contribute more, by giving them points and showing a leader board of top contributors.

In gamification of this data set there are two areas we need to focus on:

1. Quantity
2. Completeness

First, by incentivizing users to contribute more often by giving them points, badges, or rankings, we’re adding more information to our data set so we can get an even more accurate representation of the world.

Second, by incentivizing users to not only add data but make sure the data is complete, formatted correctly, and adheres to standards we can spend much less time in the Data Wrangling portion part of projects.

One potential issue with this though, is if completeness isn’t taken into effect users who want to up their score could just keep adding data points that are incomplete to climb the leaderboard.

Even with that issue in mind the gamification process could leader to much higher quality of data for more analysis!