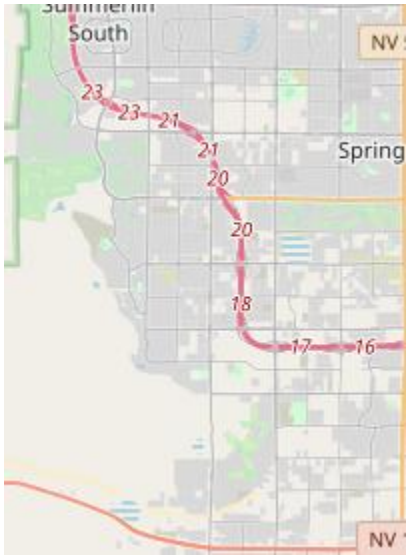
Source Map

https://www.openstreetmap.org/#map=11/36.1612/-115.3510



The map boundaries of the chosen selection are as follows:

36.1595

-115.3729

35.9791

-115.2425

Reasons for Choosing This Location

I have been living in Las Vegas a little over 13 years. For the last 12 years, I have lived in the Southwest area of the Las Vegas Valley. This general area is represented by the map selection I used for this exercise.

I chose this location because I am very familiar with it and have been able to travel around this area extensively. I wanted to bring life experiences and knowledge of the location to the table so that I could add another layer of auditing verification to my subsequent explorations. In addition, I also sought to gain a greater understanding of the data behind the location in which I live.

Audit Process and Reasoning

I began with a simple query into the size of the source file. I used the built-in 'getsize()' function to determine the size. The OSM source file came in at 62711991 bytes, which translates to roughly 62.71 MB. This confirmed that the source file met the 50 MB minimum of the project requirement.

I then created a couple of simple functions to parse through the file using the 'iterparse()' method in order to determine the number of unique users that have contributed to this map selection on OpenStreetMap. I queried both the 'uid' and 'user' tags to add an additional layer of verification by comparing two different values. It was determined that a total of 676 unique users contributed to the map data.

Next, I created a function to determine the type and total count of different tag elements in the source file. I then performed a deeper and more detailed audit of the 'k' tags. This allowed me to identify different areas that I wanted to explore once the data was cleaned up.

Overview Statistics

After the JSON conversion, I explored the updated data via a series of pipeline functions and more simplified built-in method queries (specifically, the .find().count() method). The pipeline function structure worked well for this exercise as it allowed me to query several different categories (tags) in the area using only minor modifications to the code. This approach led to code that was easy to read and reuse as necessary. I have included some of the more interesting findings below, followed by samples of the code and I used for the queries:

- Total number of documents in the source file: 317,101
- Total number of 'node' tags: 290,183
- Total number of 'way' tags: 26,918
- 'TheDutchMan13' was the top map contributor, with 23,434 total contributions
- The top 4 brands represented in the map selection area were Chevron, Starbucks, CVS Pharmacy, and Subway, all with 9 total locations
- The top 3 leisure facilities were labeled as 'pitch', 'park', and 'garden'
- The top 3 healthcare facility labels found were pharmacy, clinic, and hospital

```
def top_amenities():
# Top 10 appearing amenities
 match = {"$match":{"brand":{"$exists":1}}}
            group = {"$group":{"_id":"$brand", "count":{"$sum":1}}}
            sort = {"$sort":{"count":-1}}
            limit = {"$limit":10}
           pipeline = [match, group, sort, limit]
           return pipeline


def aggregate(db, pipeline):
           result = db.vegas.aggregate(pipeline)
           return result


def test(pipeline_function):
           db = get_db(db_name)
           pipeline = pipeline_function
           cursor = aggregate(db, pipeline)
           import pprint
           for document in cursor:
           pprint.pprint(document)


print("Total number of documents in source file: ", db.vegas.find().count())
print("Total number of node tags: ", db.vegas.find({'type':'node'}).count())
print("Total number of way tags: ", db.vegas.find({'type':'way'}).count())
```

Problems Encountered in the Map

Luckily, I did not find many obvious problems during my initial queries and research of the source data. There were no problem characters encountered in the data based on two different queries performed. I audited the different types of 'k' tags and arranged them by count in descending order. While some potential problems appeared to exist on the surface of the findings, such as the multiple different 'tiger' tags, I chose not to explore those further as

they were not a part of the ultimate inquiries of the project. An examination of the openstreetmap wiki page also shows that the tags were a part of an import project to get the data into OSM. They were not likely to affect my searches. However, because there were so many instances of these tags, I did address them in the JSON conversion just in case as I wanted to ensure uniformity regardless of relevance.

Other potential problems included the address data. Because a good portion of this area is new construction built within the last 15 years, much of the address data was accurate. I did not find any major mistakes or anomalies in the data. One potential issue was the audit of the 'city' field. For one, there were only 293 total entries for 'city' tags. This may mean that many addresses would not be included if one were to search for something based on city. In addition, 'Spring Valley', and 'Las Vegas, NV' were also represented in a city tag search. While those only represent 4 of the total files with a city tag, they do have the potential to through off subsequent searches. My final audit of the tags also produced only 287 instances of the city tag, which means that 6 of the total entries were not represented. I chose not to explore or clean those up further because of the small number of entries, but felt it worth mentioning because of the inconsistency of total values.

The other prominent inconsistency I found in the address themed data was the relationship between abbreviations of street names. Many of the entries were consistent and to be expected, but a few of the abbreviations needed to be cleaned up. I chose to take care of those while converting the source map to JSON data. Any other potential problems or concerns will be addressed later in my report.

Improvement Process

As mentioned earlier, the biggest problem I found in the source file was the inconsistency in the address data. The process I used to clean this up is as follows:

- Create functions to audit data to identify street names and types that fall outside of expected results
- Comb through results to identify unexpected entries and determine whether they needed further "clean up" or were accurate in name
- Created a list of all anomalies with a plan of action for each item
- Amended "expected" list with an update of street names and labels unique to this area that were accurate in spelling
- Created a mapping function to update abbreviated street names and directional labels with the appropriate complete word
- Created an audit function to test examples and assert better names where issues exist
- Created an update function to address tiger tags and zip code length in JSON conversion – although not included in audits, wanted to ensure uniformity of data

Benefits and Anticipated Problems of Improvement Implementation

I feel as though consistency and uniformity of tag info is probably the most important aspect of the improvement process, as that is the starting point for almost all general inquiries into the source data. As my earlier initial inquiries revealed, there were dozens of different types of tags involved. A more thorough investigation and audit of the tags would definitely be warranted if one needed to gain a complete understanding of the map involved. Real-world applications of this project would definitely need to include more legwork in the discovery process. As this was a learning exercise, I chose to keep it more basic to improve my ability to focus on the 'bones' of the project.

Beyond the general tag data, I feel that address consistency is the next most important aspect of this and any map. This is the information used by most in the world to find specific locations. Creation of the functions allows for a simplification of the code, which makes subsequent updates or inquiries easier to implement. In addition, the entire process allows the auditor to gain a greater understanding of the area and how addresses work in a specific locale.

Creation and conversion of source data into a JSON document allows for the data to be more efficiently organized and presented in a way that makes subsequent audits and inquiries much easier and time effective using MongoDB. One can simply implement one line of code using built-in functionality versus having to create entire functions just to find one piece of data.

Additional potential problems in the final data are indicated as follows:

- There were a relatively small number of 'city' labels found when compared to the total number of documents. Further investigation would be needed to determine the reason for the discrepancy.
- The number of different types of healthcare facilities found – I know from personal experience that there are many more dentists and optometrists (and potentially clinics) in the area than are listed here. Is this really an accurate finding and representation of the area? Would one need to find another way to inquire about these types of businesses?
- The different label tags for leisure facilities.
    - Does the 'pitch' label represent soccer fields or something else?
    - 'Park' label – does this refer only to public and private parts or something else (parking lot, etc.)?
    - 'Garden' label – does this refer only to public spaces or does it include private locations as well? Is a garden really a leisure facility?
    - 'Centre' label – would this non-traditional spelling affect an unexpecting auditor's ability to search by this field?

Resources for Project

The following is a list of resources I used to complete this project. The list includes a mix of website tutorials, forums, blog posts, youtube videos, and github repositories.

- https://wordpress.com/support/markdown-quick-reference/
- https://github.com/aerdem4/kaggle-quora-dup/issues/3
- https://nbviewer.jupyter.org/github/tybyers/MongoDB_finalproject/blob/master/exploring_data.ipynb
- https://github.com/SheenaYu/Wrangle-OpenStreetMap-data-in-DFW-Area
- https://github.com/baocongchen/Wrangle-OpenStreetMap-Data
- https://stackoverflow.com/questions/30418481/error-dict-object-has-no-attribute-iteritems
- https://github.com/tensorflow/models/issues/3173
- https://www.google.com/search?q=w+buckskin+las+vegas&rlz=1C1CHBF_enUS905US907&oq=w+bucksk&aqs=chrome.0.69i59l2j69i57j0i22i30l7.2641j0j7&sourceid=chrome&ie=UTF-8
- https://www.geeksforgeeks.org/how-to-import-json-file-in-mongodb-using-python/
- https://stackoverflow.com/questions/31055637/getting-mongoimport-is-not-recognized-as-an-internal-or-external-command-ope
- https://docs.mongodb.com/guides/server/import/
- https://stackoverflow.com/questions/33684316/cannot-import-example-dataset-the-system-cannot-find-the-specified-file
- https://github.com/SheenaYu/Wrangle-OpenStreetMap-data-in-DFW-Area/blob/master/mongodb_query.py
- https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
- https://nbviewer.jupyter.org/github/tybyers/MongoDB_finalproject/blob/master/exploring_data.ipynb
- https://github.com/ipython/ipython/issues/8478/
- https://wiki.openstreetmap.org/wiki/Key:tiger: