Name: Andrew Bauman

Email: baumanab@gmail.com

URL: https://www.openstreetmap.org/export#map=12/47.7405/-122.3359

I chose this area because it is where I have lived and worked since 2008. I spent nearly 2 years walking and riding the bus, so I learned the area very well.

Other Important Information:

The course was good overall, especially the first part. However, it started to break down around lesson 4 in terms of continuity. The final project is interesting but I don't think it was well designed and it doesn't utilize major aspects of the course. For example, it doesn't use MongoDB, this is a huge gap. This is disappointing. I will say that most of the tutors and coaches are top notch. I wish I could pay a monthly fee just to have coaching and tutoring available for every day learning projects. Also, at the original course was supposed to involve bioinformatics, which would have been more interesting for me since I have a proteomics researcher/protein scientist. There is also a lot of convenience and other code to help the exercises work. The tutors did a good job explaining this but I think overall the Udacity instructors should focus a bit more on the practical implementation, testing, and debugging of the code.

Also, for a course involving MongoDB, we barely used it. This is also disappointing. Had I known this I would have taken a different course.

Problems Encountered in my Map:

Actually there were not that many major problems. It was the typical things encountered in the example.osm in the lessons. There were non-standard abbreviations for string names, inappropriate data types (strings for numbers), missing data, problem characters or names in the addresses (very few of these). The biggest problem I encountered was in update name. This happened both in the reshape (data.py) and in audit.py itself. If you try to run update_name on something that is not in the mapping then you are trying to do a replace with 'None.' Which clearly doesn't work. Anyway, none of it this was hard to figure out. I would just run the code, see what exceptions popped up or what the data looked like, then add print statements or collect into in lists/dicts, to get a better idea of what was going on. Usually I found the problem then rewrote the code accordingly. Basically when I found an output I didn't like I determined what part of my code produced the output or if it was from the original data, then rewrote the code to fix it. Standard stuff. My coach was a big help with this.

There was also the memory problem. The original data.py from the exercise was a resource hog. With Carl's suggestions and running some tests, I focused on the reshape and got rid of extraneous code. I also implemented an element clear to free up ram. Now the code hums right along, no problems.

After getting this code into the DB I found some other problems as well, usually involving strings separated by delimiters and would be best turned into arrays. I decided not to tackle those at this point. Now that I have a process and fast and easy to understand code I feel confident I can face most challenges. I picked one (the street naming and data shape) and went with it. I intentionally left directional (NW, SE, etc.) as is in the beginning of street names. This is how we write them in Seattle. No one would ever direct you to Northeast 85th St. It would be NE 85th Street.

I should mention that one major thing I learned was how to really leverage the update_name function. Once I had data.py working well I focused only on changing update_name to make changes to during the reshaping. If I were to fix other problems I would use this approach, but with separate functions. That makes it easy to track changes and debug.

Here are some things I would consider fixing.

Turn Sources such as "source": "King County GIS;data.seattle.gov;bing;knowledge" into arrays

Change the type of opening hours and turn them into a datetime array.

Turn directions such as "exit_to": "WA 523 (Northeast 145th Street); 5th Avenue Northeast" into arrays.

Overview of the data:

size of the file: 117mb

Added Stats and queries:

Collection stats: db.command('collStats','myseattle')

{'avgObjSize': 257.08978839327483,
'count': 581456,
'indexSizes': {'_id_': 16997904},
'lastExtentSize': 50798592,

'nindexes': 1,
'ns':'osm.myseattle',
'numExtents': 12,
'ok': 1.0,
'paddingFactor': 1.0,
'size': 149486400,
'storageSize': 174735360,
'systemFlags': 1,
'totalIndexSize': 16997904,
'userFlags': 0}

Number of Unique Users:

```
pipeline = [{'$group':{'_id': '$created.user',
                       'count':{'$sum':1}}},
            {'$group':{'_id': '$created.user',
                       'count':{'$sum':1}}}]
```

{u'ok': 1.0, u'result': [{u'_id': None, u'count': 213}]}

Who are the people (adding info) to your neighborhood?

```
pipeline = [{'$group':{'_id': '$created.user',
                       'count':{'$sum':1}}},
            {'$sort': {'count': -1}},
            {'$limit': 10}]
```

Top 10 Contributors:

[{u'_id': u'SeattleImport', u'count': 308039},
{u'_id': u'Glassman', u'count': 176165},
{u'_id': u'Foundatron', u'count': 62301},
{u'_id': u'zephyr', u'count': 6506},

{u'_id': u'compdude', u'count': 4698},
{u'_id': u'STBrenden', u'count': 3131},
{u'_id': u'Paul Johnson', u'count': 1982},
{u'_id': u'WernerP', u'count': 1721},
{u'_id': u'ewedistrict', u'count': 1339},
{u'_id': u'Yeliena', u'count': 1260}]}

ewedistrict (University District) is very clever

No(des) Way(s)!

```
pipeline = [{'$group':{'_id': '$type',
                       'count':{'$sum':1}}}]
```

[{u'_id': u'way', u'count': **57064**},
{u'_id': u'node', u'count': **524392**}]}

Total Number of Amenities: 1382

```
pipeline = [{'$match':{'amenity':{'$exists':True}}},
            {'$group':{'_id': '$amenity',
                       'count':{'$sum':1}}},
            {'$group':{'_id': '$amenity',
                       'count':{'$sum':'$count'}}}]
```

Oh, and 532 of them are bicycle parking, WTH!?

Number of Unique Amenity Types: 55

```
pipeline = [{'$group':{'_id': '$amenity',
                       'count':{'$sum':1}}},
            {'$group':{'_id': '$amenity',
                       'count':{'$sum':1}}}]
```

{u'ok': 1.0, u'result': [{u'_id': None, u'count': 55}]}

Top 20 Amenities:

```
pipeline = [{'$match':{'amenity':{'$exists':True}}},
            {'$group':{'_id': '$amenity',
                    'count':{'$sum':1}}},
            {'$sort': {'count': -1}},
            {'$limit': 20}]
```

```
{u'_id': u'bicycle_parking', u'count': 532},
        {u'_id': u'parking', u'count': 161},
        {u'_id': u'restaurant', u'count': 141},
        {u'_id': u'place_of_worship', u'count': 88},
        {u'_id': u'school', u'count': 54},
        {u'_id': u'cafe', u'count': 54},
        {u'_id': u'fast_food', u'count': 42},
        {u'_id': u'pub', u'count': 32},
        {u'_id': u'fuel', u'count': 25},
        {u'_id': u'post_box', u'count': 24},
        {u'_id': u'bench', u'count': 23},
        {u'_id': u'toilets', u'count': 23},
        {u'_id': u'bank', u'count': 20},
        {u'_id': u'bar', u'count': 19},
        {u'_id': u'vending_machine', u'count': 15},
        {u'_id': u'atm', u'count': 15},
        {u'_id': u'pharmacy', u'count': 12},
        {u'_id': u'post_office', u'count': 10},
        {u'_id': u'dentist', u'count': 7},
        {u'_id': u'drinking_water', u'count': 6}]}
```

Drinking water #20! We know how to class it up in Seattle

Incidentally I queried the notes for this data set, one of the pharmacies is actually a medical marijuana dispensary.

```
pipeline = [{'$match':{'note':{'$exists':True}}},
            {'$match':{'amenity': 'pharmacy'}},
            {'$group':{'_id': '$amenity',
                    'count':{'$sum':1},
                    'name': {'$push': '$name'},
                    'note':{'$push': '$note'}}}]

u'result': [{u'_id': u'pharmacy',
           u'count': 1,
           u'name': [u'Herb(n)Care'],
           u'note': [u'medical marijuana dispensary']}]}
```

Total Number of Shops: 280

```
pipeline = [{'$match':{'shop':{'$exists':True}}},
            {'$group':{'_id': '$shop',
                    'count':{'$sum':1}}},
            {'$group':{'_id': '$shop',
                    'count':{'$sum':'$count'}}}]
```

Number of Unique Shop Types: 91

```
pipeline = [{'$match':{'shop':{'$exists':True}}},
            {'$group':{'_id': '$shop',
                    'count':{'$sum':1}}},
            {'$group':{'_id': '$shop',
                    'count':{'$sum':1}}}]

{u'ok': 1.0, u'result': [{u'_id': None, u'count': 91}]}
```

Top 20 Shops

```
pipeline = [{'$match':{'shop':{'$exists':True}}},
            {'$group':{'_id': '$shop',
                    'count':{'$sum':1}}},
            {'$sort': {'count': -1}},
            {'$limit': 20}]
```

```
{u'_id': u'convenience', u'count': 24},
        {u'_id': u'supermarket', u'count': 17},
        {u'_id': u'hairdresser', u'count': 14},
        {u'_id': u'clothes', u'count': 12},
        {u'_id': u'furniture', u'count': 12},
        {u'_id': u'dry_cleaning', u'count': 10},
        {u'_id': u'books', u'count': 9},
        {u'_id': u'car_repair', u'count': 9},
        {u'_id': u'beauty', u'count': 7},
        {u'_id': u'pet', u'count': 6},
        {u'_id': u'doityourself', u'count': 5},
        {u'_id': u'optician', u'count': 4},
        {u'_id': u'vacant', u'count': 4},
        {u'_id': u'bicycle', u'count': 4},
        {u'_id': u'music', u'count': 4},
        {u'_id': u'gift', u'count': 4},
        {u'_id': u'department_store', u'count': 4},
        {u'_id': u'second_hand', u'count': 4},
        {u'_id': u'bakery', u'count': 4},
        {u'_id': u'toys', u'count': 4}]}
```

#1 hit, convenience. Give me convenience or give me death!

Another useful note:

```
pipeline = [{'$match':{'note':{'$exists':True}}},
            {'$match':{'name': 'Hubbard Homestead Park'}},
            {'$group':{'_id': '$leisure',
                       'name': {'$push': '$name'},
                       'note':{'$push': '$note'}}}]

u'result': [{u'_id': u'park',
             u'name': [u'Hubbard Homestead Park'],
             u'note': [u"this park actually exists--it's just so new that it's not in the sat imagery."]}]}
```

#1 Cuisine: Coffee

Only in Seattle would Coffee be considered a cuisine.

```
pipeline = [{'$match':{'cuisine':{'$exists':True}}},
            {'$group':{'_id': '$cuisine',
                       'count':{'$sum':1},
                       'name': {'$push': '$name'}}},
            {'$sort': {'count': -1}},
            {'$limit': 1}]

u'result': [{u'_id': u'coffee_shop',
             u'count': 26,
             u'name': [u'Starbucks',
                       u'Herkimer Coffee',
                       u'Caffe Fiore',
                       u'Cupcake Royale',
                       u'Starbucks',
                       u'Starbucks',
```

u'Emerald City Orchids',
u'Greenwood Chocolati Cafe',
u'Grumpy Ds',
u'Wayward Coffeehouse',
u'Cafe Bambino',
u'Anchored Ship',
u'Ballard Coffee Works',
u'Starbucks',
u'Java Bean Coffee',
u'Caffe Vita',
u'The Old Peculiar',
u"Tony's Coffee",
u'Zoka Coffee',
u'Home Esspresso Repair and Cafe',
u'Jewel Box Cafe',
u'Bauhaus',
u'Starbucks Drive Thru',
u'Starbucks',
u'Firehouse Coffee',
u'Diva Espresso']}]}

Other ideas about the datasets:

I considered several ways of analyzing and improving the data, as stated below.

Timestamp:

Convert the time stamp from a string to a datetime object when cleaning/reshaping the data. This would be done by adding code here (pseudo-code in bold):

```
for cursor in element.iter(element.tag):
        for label in cursor.attrib:
            val = cursor.attrib[label]
            if label in CREATED:
                created[label] = val

            if label == 'lat' or label == 'lon':
                pos.append(float(val))

            if label == 'timestamp':
                convert label to datetime object
```

To implement the improvement you would first start by auditing that field to get an idea of how uniform it is and if there are any characters or other issues in making the conversion. You would then clean the field and write the converter (currently beyond the scope of my knowledge but I'm teaching myself how to do this now). Problems would be as mentioned problem characters or non-uniformity of the field. Since this field is not user generated I don't actually anticipate major problems. The benefit would be the ability to easily make datetime calculations using this field.

Improving the notes for Bike Racks:

As I discovered in my analysis there are over 500 entries related to bike bicycle parking. These are bike racks. Here is what a typical bike rack note looks like.

"note": "17TH AVE NW 0550 BLOCK W SIDE (21) 21 FT N/O NW MARKET ST"

This could be improved in much the same way as the street names were improved, using essentially the auditing, cleaning, and reshaping process. This would help make the note more human readable. For example the entry above could read.

"note": "17TH AVE Northwest 0550 BLOCK West SIDE (21) 21First North of Northwest MARKET Street"

The only problem I see in implementing this is that it would take extra processing time and memory as well as make the DB larger. I doubt this would be a serious problem and

could be managed by carefully designed code. If this caused any downstream performance query problems that could be managed by indexing (beyond my knowledge scope but that is where I would start.)

Relevance: This one is a stretch and really beyond my current knowledge scope but I put it here to demonstrate my thought process on the data set and how issues of sparseness and outdated entries might be handled.

An important part of data analysis is consideration of the relevance of the data. How relevant is the OSM data for Seattle? That is, how well does the OSM data snapshot I chose for my analysis align with the actual state of that area? Seattle is an expanding city and has been undergoing major changes in its overall structure and particularly its transportation infrastructure. We have also had major changes to our laws in regards to liquor, medical marijuana, and recreational marijuana, which has dramatically changed the types of shops on major routes in North Seattle. How well is the OSM data keeping up with these changes? As a resident of this area it is very obvious to me that it is not keeping up with these changes in many ways, and is poorly aligned with the actual state of the area, especially in terms of amenities. A great example of this is that most of the medical marijuana stores and private liquor stores are not represented in the data. This is largely because they are very new as the laws enabling their existence have only been in their current state for a short period. If someone not from the area was using this data set they might not realize how sparse the set is or how poorly aligned it is to the actual state of the area. This might invalidate some of their analyses no matter how competently conducted. I would improve this data set by assigning a score to gauge relevance. This could help guide researchers as to how confident they can be in the accuracy of the data as gauged by time.

I would do this by use of the time stamp and the position fields.

- Lay a grid onto the area of study by starting from the area boundaries using vertical and horizontal lines at regular intervals from the N/S and E/W boundary lines. This would essentially create a set of boxes at regular intervals.
- Assign each box a number and a letter based on its N/S and E/W position (exactly like a regular, paper, road map). Add this as a new field 'subarea' using an update query ($set)
- Use an update query to add a new field ($set): 'freshness'
    - This would consist of the difference of the most recent updates for a subarea from the current date and time ($subtract) or a date and time of the researchers choosing.
    - Comparison operators could then be used to assign a 'freshness' score.
- The 'freshness' score would help gauge the relevance of the area.

- Some of the details of doing this or its validity are beyond the scope of my current knowledge. However, I do know that in preparing the data that the date time field should be converted from a string to a datetime object. This might require additional cleaning.
- Implementing this might cause performance issues depending on how often you were going to update the freshness field.
- The best way to calculate freshness is not clear to me (most recently updated entry in an area, average age of the data in the area?). This is an important choice and some trial and error as well as validation would be involved.