Project Summary

What is your name?

Benjamin Lee

What E-mail address do you use to sign in to Udacity?

benjwlee@gmail.com

What area of the world you used for your project? Post a link to the map position and write a short description. Note that the osm file of the map should be at least 50MB.

URL: https://www.openstreetmap.org/export#map=12/25.0327/121.5131

This is where I grew up. It's fun to go through this place and found out all the changes.

Is there a list of Web sites, books, forums, blog posts, github repositories etc that you referred to or used in this submission (Add N/A if you did not use such resources)?

Use this place to list the citations.

1. http://docs.mongodb.org/manual/core/aggregation-introduction/
2. http://stackoverflow.com/ I used this site quite extensively to resolve many of my problems in programming.
3. http://stackoverflow.com/questions/14568283/mongodb-aggregation-count-array-set-size has given me specific instruction how to count array size
4. http://discussions.udacity.com/c/nd002-2014-12-03/data-wrangling was used to consult doubts of if Unicode was the source of my problems.

Please carefully read the following statement and include it in your email:

*"I hereby confirm that this submission is my work. I have cited above the origins of **any** parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc. By including this in my email, I understand that I will be expected to explain my work in a video call with a Udacity coach before I can receive my verified certificate."*

Is there any other important information that you would want your project evaluator to know?

Use this space to communicate with your project evaluator. Is there anything you would like to communicate? Feedback or suggestions?

I have chosen Taipei, my childhood town as my subject of study.

Problems Encountered in the Map

The Unicode in map gave me some early doubts but was resolved later on. By issuing:

```
> db.mapT.aggregate([
... {"$match":{"amenity":{"$exists":1},"amenity":"restaurant"}},
... {"$group":{"_id":"$cuisine","count":{"$sum":1}}},
... {"$sort":{"count":-1}},
... {"$limit":10},])
{ "_id" : null, "count" : 2023 }
{ "_id" : "chinese", "count" : 245 }
{ "_id" : "japanese", "count" : 105 }
{ "_id" : "italian", "count" : 50 }
{ "_id" : "regional", "count" : 45 }
{ "_id" : "burger", "count" : 23 }
{ "_id" : "pizza", "count" : 19 }
{ "_id" : "asian", "count" : 15 }
{ "_id" : "american", "count" : 14 }
{ "_id" : "sushi", "count" : 11 }
```

It appears to have many rows failed due to one reason or another. My starting guess was the null value due to Unicode. With more understanding, I later on by adding this code:

```
> db.mapT.aggregate([
... {"$match":{"amenity":{"$exists":1},"amenity":"restaurant"}},
... {"$match":{"cuisine":{"$exists":1}}},
... {"$group":{"_id":"$cuisine","count":{"$sum":1}}},
... {"$sort":{"count":-1}},
... {"$limit":10},])
```

{ "_id" : "chinese", "count" : 245 }
{ "_id" : "japanese", "count" : 105 }
{ "_id" : "italian", "count" : 50 }
{ "_id" : "regional", "count" : 45 }
{ "_id" : "burger", "count" : 23 }
{ "_id" : "pizza", "count" : 19 }
{ "_id" : "asian", "count" : 15 }
{ "_id" : "american", "count" : 14 }
{ "_id" : "sushi", "count" : 11 }
{ "_id" : "vegetarian", "count" : 10 }
I removed entries without value needed in order to avoid 'null' as part of the represented values.

In order to count how many language is present for a given cuisine, I have reprogrammed 6_12's script. I have chosen to put different languages into an array in order to count. The original name of the entity is saved in a different variable: nameDef, short for 'Default Name'. I used regular expression to discover the language modifier in the format of name:en, name:zh, moved 'en', 'zh' into an array. For the original name in the field name, I created 'default' as its variation. Although there are some variation of each languages such as zh can be zh-py. It is the same language although on a different system. I left this alone and counted each as a different language.

This is an example of my concern:

<tag k="amenity" v="fast_food"/>
<tag k="cuisine" v="burger"/>
<tag k="name" v="茉莉漢堡"/> ← default name
<tag k="name:en" v="Mary's Hamburger"/> ← English name
<tag k="name:zh" v="茉莉漢堡"/> ← Chinese name
<tag k="wheelchair" v="limited"/>

The calculation is done for a subset of tags. We accumulated a set of languages for these tags. Cuisine of an amenity can be filtered through and answer question #6.

```
addpat = re.compile(r'addr:(\w+)(:\w*)*')
namepat = re.compile(r'name:(\w+)(:\w*)*')

        else:
          match=re.search(namepat,k) ← name:(\w+)(:\w*)* pattern
          if match:
            #no additional sections, if do, ignore data
            if match.group(2) == None:
              #create address structure as needed
              if 'name' not in node.keys():← if not present already, create the array first
                node['name']=[]
              #append language to name array if not already
              if not match.group(1) in node['name']: ←If not registered already
                node['name'].extend([match.group(1)]) ←Add language presented to array
          #all other values, this is not enumerated
          else:
            #the name does not fit into name:en pattern
            if k == 'name': ←Similar behavior if there's only 'name' in the tag
              #default name stored
              node['nameDef']=v
              if 'name' not in node.keys():
                node['name']=[]
              #add to array
              node['name'].extend(['default'])
            else:
```

```
        node[k]=v
      #trying to catch all known tags
      if k not in nottags:
        nottags[k]=1
```

Map.py is to create the collection mapT to pack the name array formation part.

Data Overview

File size

File	Size
mapTaipei.osm	204 MB (214,725,416 bytes)
mapTaipei.osm.json	304 MB (319,782,801 bytes)

Sort postcodes by count, descending #1

```
db.mapT.aggregate([
{"$match":{"address.postcode":{"$exists":1}}},
{"$group":{"_id":"$address.postcode","count":{"$sum":1}}},
{"$sort":{"count":-1}},
{"$limit":10}])
```

Here are the top ten postal codes around Taipei:

```
{ "_id" : "220", "count" : 97 }
{ "_id" : "238", "count" : 95 }
{ "_id" : "114", "count" : 88 }
{ "_id" : "300", "count" : 79 }
{ "_id" : "106", "count" : 78 }
{ "_id" : "302", "count" : 59 }
{ "_id" : "11463", "count" : 52 }
{ "_id" : "224", "count" : 46 }
{ "_id" : "104", "count" : 37 }
{ "_id" : "100", "count" : 33 }
```

Sort cities by count, descending #2

```
db.mapT.aggregate([
{"$match":{"address.city":{"$exists":1}}},
{"$group":{"_id":"$address.city","count":{"$sum":1}}},
{"$sort":{"count":-1}},
{"$limit":10},])
```

```
And, the top 10 cities around city of Taipei:
{ "_id" : "台北市", "count" : 447 } ← Unicode works just fine
{ "_id" : "臺北市", "count" : 365 }
{ "_id" : "新北市", "count" : 336 }
{ "_id" : "新北市板橋區", "count" : 144 }
{ "_id" : "桃園市", "count" : 131 }
{ "_id" : "新竹市", "count" : 105 }
{ "_id" : "新竹縣竹北市", "count" : 67 }
{ "_id" : "新北市瑞芳區", "count" : 53 }
{ "_id" : "Taipei", "count" : 48 }
{ "_id" : "新竹市東區", "count" : 42 }
```

Number of documents

```
db.mapT.find().count()
1072571
```

Number of nodes

```
db.mapT.find({"type":"node"}).count()
974462
```

Number of ways

```
db.mapT.find({"type":"way"}).count()
98081
```

Number of unique users

```
db.mapT.distinct("created.user").length
1251
```

Number of Amenity

```
db.mapT.distinct("amenity").length
145
```

Number of Type of nodes

```
db.mapT.distinct("type").length
24
```

Top 1 contributing user

```
db.mapT.aggregate([
{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
{"$sort":{"count":-1}},
{"$limit":1}])
```

{ "_id" : "Supaplex", "count": 183985 }

Number of users appearing only once (having 1 post)

```
db.mapT.aggregate([
{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
{"$group":{"_id":"$count", "num_users":{"$sum":1}}},
{"$sort":{"_id":1}},
{"$limit":1}])
```

{ "_id" : 1 "num_users" : 193 }

Additional data exploration using Python parser and MongoDB queries

Top 10 appearing amenities #3

```
db.mapT.aggregate([
{"$match":{"amenity":{"$exists":1}}},
{"$group":{"_id":"$amenity","count":{"$sum":1}}},
{"$sort":{"count":-1}},
{"$limit":10},])
Top 10 ranked amenities around city of Taipei:
{ "_id" : "restaurant", "count" : 2810 }
{ "_id" : "parking", "count" : 1683 }
{ "_id" : "place_of_worship", "count" : 928 }
{ "_id" : "school", "count" : 854 }
{ "_id" : "cafe", "count" : 572 }
{ "_id" : "bank", "count" : 521 }
{ "_id" : "police", "count" : 464 }
{ "_id" : "shelter", "count" : 437 }
{ "_id" : "toilets", "count" : 375 }
```

```
{ "_id" : "fast_food", "count" : 359 }
```

Biggest religion (no surprise here) #4

```
      db.mapT.aggregate([
      {"$match":{"amenity":{"$exists":1},"amenity":"place_of_worship"}},
      {"$match":{"religion":{"$exists":1}}},
      {"$group":{"_id":"$religion", "count":{"$sum":1}}},
      {"$sort":{"count":-1}},
      {"$limit":10},])
Top ten religions around city of Taipei:
{ "_id" : "taoist", "count" : 189 }
{ "_id" : "buddhist", "count" : 172 }
{ "_id" : "christian", "count" : 68 }
{ "_id" : "muslim", "count" : 3 }
{ "_id" : "民間信仰", "count" : 2 }
{ "_id" : "catholic", "count" : 1 }
{ "_id" : "hindu", "count" : 1 }
{ "_id" : "Taoism", "count" : 1 }
{ "_id" : "buddhist;taoist;folk", "count" : 1 }
{ "_id" : "jain", "count" : 1 }
```

Most popular cuisines #5

```
      db.mapT.aggregate([
      {"$match":{"amenity":{"$exists":1},"amenity":"restaurant"}},
      {"$group":{"_id":"$cuisine","count":{"$sum":1}}},
      {"$sort":{"count":-1}},
      {"$limit":10},])
Top ten cuisines around city of Taipei:
{ "_id" : "chinese", "count" : 245 }
{ "_id" : "japanese", "count" : 105 }
{ "_id" : "italian", "count" : 50 }
{ "_id" : "regional", "count" : 45 }
{ "_id" : "burger", "count" : 23 }
{ "_id" : "pizza", "count" : 19 }
{ "_id" : "asian", "count" : 15 }
{ "_id" : "american", "count" : 14 }
{ "_id" : "sushi", "count" : 11 }
{ "_id" : "vegetarian", "count" : 10 }
```

#Most Languages for a Cuisine #6

By using the field NameDef, I am able to calculate the size of ‘name’ field. Again, this is where all the languages are stored so the result is the count of how many languages are stored for the amenity.

```
      db.mapNT.aggregate([
      {$match:{"cuisine":{"$exists":1}}},
      {$project:{_id:0,id:1,nameDef:1,count:{$size:{"$ifNull":["$name",[]]}}}},
      {$sort:{"count":-1}},
      {$limit:10},])
Top ten cuisines have largest language count:
{ "nameDef" : "星巴克", "id" : "736470054", "count" : 3 }
{ "nameDef" : "Creative Pasta 創義麵景美店", "id" : "2177250580", "count" : 3 }
{ "nameDef" : "餐莊快餐", "id" : "2473232871", "count" : 3 }
{ "nameDef" : "茉莉漢堡", "id" : "485688385", "count" : 3 }
{ "nameDef" : "閾靈咖啡館 (The Fairy Cafe)", "id" : "2090614029", "count" : 3 }
```

```
{ "nameDef" : "吉野家", "id" : "2320119243", "count" : 3 }
{ "nameDef" : "薄多義(BITE 2 EAT)", "id" : "2724850255", "count" : 3 }
{ "nameDef" : "長興小舖", "id" : "2473232956", "count" : 3 }
{ "nameDef" : "摩斯漢堡", "id" : "2124843059", "count" : 3 }
{ "nameDef" : "Rakuzan Ramen (樂山溫泉拉麵)", "id" : "2071356199", "count" : 3 }

There are 25 cuisines have 3 languages listed, but only 10 are printed.

As these selections have at most 3 languages marked, I can use the following to discover
the rest of the documents that fits the same criteria.
db.mapNT.find({"$and":[{"cuisine":{$exists:true}},{"name":{$size:3}}]})

MongoDB shell version: 2.6.7
connecting to: test
cuisine,name,id,amenity,languages
burger,茉莉漢堡,485688385,fast_food,default,en,zh
coffee_shop,星巴克,736470054,cafe,default,en,zh
japanese,Rakuzan Ramen (樂山溫泉拉麵),2071356199,restaurant,default,en,zh
south_african_bistro,閾靈咖啡館 (The Fairy Cafe),2090614029,cafe,default,en,zh
burger,摩斯漢堡,2124843059,fast_food,default,en,zh
italian,Creative Pasta 創義麵景美店,2177250580,restaurant,default,en,zh
japanese,吉野家,2320119243,fast_food,default,en,zh
chinese,餐莊快餐,2473232871,restaurant,default,en,zh
chinese,長興小舖,2473232956,restaurant,default,en,zh
pizza,薄多義(BITE 2 EAT),2724850255,restaurant,default,en,zh
sushi,鮨彩壽司,2726475748,restaurant,default,en,zh
Thai,非常泰,2726475749,restaurant,default,en,zh
Steak,法樂琪,2726475934,restaurant,default,en,zh
Coffee_shop,跳舞香水,2726480264,restaurant,default,en,zh
Chinese,鼎泰豐,2726480268,restaurant,default,en,zh
mexican;american,環球蔬食樂 (Global Veggie Joy),2810326373,restaurant,default,en,zh
pizza,達美樂披薩,2908557823,fast_food,default,en,zh
coffee_shop,星巴克,2909069806,cafe,default,en,zh
pizza,芝加哥比薩,2935820265,restaurant,default,en,ja
italian,貝里斯義大利麵 (Belize Pasta),2940699052,restaurant,default,en,zh
coffee_shop,窩著咖啡,2949072184,cafe,default,en,zh
burger,麥當勞 (McDonald's),3278145373,fast_food,default,en,zh
pizza,Domino's Pizza,3374413330,fast_food,default,en,zh
pizza,Pizza Hut,3374413331,fast_food,default,en,zh
burger,McDonald's,328305856,fast_food,default,en,zh
bye
```

Conclusion

I have decided by dumping all unknown tags will help me to understand the map better. It really is up to the pursuer determine which direction he/she wants to go.

Other idea, but lack of time to implement, for example, is to calculate distance of certain location, such as where are all the restaurants and movie theaters around my house in a 5 miles radius. The distance calculation between two points can be calculated through two pairs of Longitude and Latitude. However I am not aware how aggregate can be coached to do such a task. The osm file for Taipei city has 3 million lines of text. I have acquired a simple distance calculation

program from: http://www.johndcook.com/blog/python_longitude_latitude/. Where 10 million calculations takes around 2.828 seconds on my laptop. In other words, I anticipate the calculation alone should take less than a second to accomplish even at the map's scope. Given if the search is limited to a reasonable small group, such calculated search should cost no more than a slow load and would be acceptable for real time consumer use.

By adapt this mechanism to our project, can we integrate this function inside of our aggregate function so I can ask questions such as:

1. How many movie theaters has more than 2 restaurants in its 2 miles radius.
2. What is the fastest way, including walking, going from where I am to a restaurant along orange line subway stops, real time?
3. Combining information such as weather, hot events and public transportation capacity to estimate impact to local traffic. Modeling can be used for prediction.
4. Etc.

Working with mongoDB aggregate has me feel a lot to be desired.

Other idea such as hooking up a voting system (by machine, or by people) to confirm update of a targeted location can have interesting repercussion. I can ask questions such as:

1. How long shall I wait in line for tickets to the latest and hottest movie in town in all the theaters along the green subway line?
2. Audience's real time thumbing up and down about a movie in town.
3. Latest and more precise real time reflect of weather in smaller areas, as I assume, lots of sensors can be integrated on a personal wearable/residential in the future.
4. Etc.

Alas, this is a good class. Makes me think. And that's what I am here for.

*I hereby confirm that this submission is my work. I have cited above the origins of **any** parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc. By including this in my email, I understand that I will be expected to explain my work in a video call with a Udacity coach before I can receive my verified certificate.*