OpenStreetMap Project

Map Area: Singapore

References:
http://regexone.com/lesson/matching_characters
https://docs.mongodb.org/manual/reference/operator/query/in/
http://www.regular-expressions.info/anchors.html

Python files used:
audit-data.py *[for initial investigation and cleaning of data]*
data.py [*incorporating code to clean data and transform OSM file to JSON]*
testquery.py *[codes for querying MongoDB]*

Overview:

1. Problems Encountered in the Map

I first used a subset of the Singapore data set (sample.osm) and ran it against the *audit-data.py* file**,** and discovered the following problems.

- **Many expected street types were flagged as new types because they were numbered e.g. "Changi South *Street 1*".**

  This was not a problem in itself because as far as I could see, the naming convention for numbered streets was consistent i.e. all numbered streets were Changi South Street 1 and there were no 1st Changi South Street or First Changi South Street.

  However, abbreviated street-types appearing before digits e.g. "Bukit Batok Ave. 5" would need to be treated and amended to "Bukit Batok Avenue 5"

  Solution:
  I updated the the street_type_re object and the update_name function to capture these cases. The output after the treatment is shown below:

```
Macintosh:Wrangle-OSM-Data SherryT$ python audit-data.py
set(['81100', '05901', '29461', '29466', '80300', '80200', '80050
ifferent>', '79200', '81750', '80000', '80250', '80400', 'Bukit Ba
, '81300', '81310'])
Bukit Timah Rd => Bukit Timah Road
Sengkan West Ave => Sengkan West Avenue
Bukit Batok East Ave 5 => Bukit Batok East Avenue 5
Gopeng St => Gopeng Street
Admiralty St => Admiralty Street
Macintosh:Wrangle-OSM-Data SherryT$
```

- **Some new street-types were seen such as , "View", "Way", "Quay", "Hill", "Rise".**

  Solution:
  These were added to the 'expected' array for completeness, although it is unlikely that these could be abbreviated further and I saw no evidence of this.

- **Some of Singapore's street names have Malay origins with different naming conventions**

  e.g. 'Lorong 27 Geylang' and 'Jalan Besar'. 'Lorong' and 'Jalan' are Malay for street/road. The naming convention for these Malay street names will be retained as reordering them to fit the English naming convention would make the road names nonsensical (e.g. Besar Jalan is not grammatically correct).

  As far as I could see, the Malay road names were consistently labelled with Lorong and Jalan appearing at the start of the street address. However a potential problem is abbreviation of “Lorong” to “Lor” or “Jalan” to “Jln”.

  Solution:
  I wrote an additional regex object “malay_type_re” to process such addresses and updated the update_name function to replace “Lor” and “Jln” abbreviations. The output after treatment is as follows:

```
Bukit Timah Rd => Bukit Timah Road
Lor Telok => Lorong Telok
Gopeng St => Gopeng Street
Admiralty St => Admiralty Street
Sengkan West Ave => Sengkan West Avenue
Bukit Batok East Ave 5 => Bukit Batok East Avenue 5
Macintosh:Wrangle-OSM-Data SherryT$
```

- **There was a street entry that said “'656289”.**

  This is likely to be a postal code wrongly entered into the 'street' field. Querying specifically for this entry in MongoDB revealed that the entry for 'postcode' had been swopped with the entry for 'street ' As there was only 1 instance of this, I will not be addressing it in this project. Instead, I have proposed how this could be addressed in Section 3: Additional Ideas – Conclusion.

```
{u'_id': ObjectId('563377ee8d8ca4aa458f7990'),
 u'address': {u'postcode': u'Bukit Batok Street 25', u'street': u'656289'},
 u'building': u'residential',
 u'created': {u'changeset': u'11497251',
              u'timestamp': u'2012-05-04T13:14:55Z',
              u'uid': u'673743',
              u'user': u'hiddenAce',
              u'version': u'2'},
 u'id': u'162094514',
 u'name': u'288G',
 u'node_refs': [u'1740286654',
                u'1740286655',
                u'1740286661',
```

- **A number of incorrect post codes were detected (5 digits instead of the standard 6).**

  Querying with MongoDB revealed that most of these postcodes were associated with addresses in Batam and Johor Bahru. Further querying to look at the number of counts by city revealed that quite a large number of data sets were not even in Singapore. (mostly in Johor Bahru).

```
u'address': {u'city': u'Johor Bahru',
             u'postcode': u'81100',
             u'street': u'Jalan Kemboja'},
u'building': u'house',
u'created': {u'changeset': u'34095778',
             u'timestamp': u'2015-09-18T05:07:10Z',
             u'uid': u'2007873',
             u'user': u'berjaya',
             u'version': u'2'},
u'id': u'371095876',
u'node_refs': [u'3747098985',
               u'3747098986',
               u'3747098987',
               u'3747098988',
               u'3747098985'],
u'type': u'way'},
{u'_id': ObjectId('563230c6ba933dddaf94257e'),
 u'address': {u'city': u'Johor Bahru',
             u'postcode': u'81100',
             u'street': u'Jalan Kemboja'},
u'building': u'house',
u'created': {u'changeset': u'34095778',
             u'timestamp': u'2015-09-18T05:07:12Z',
             u'uid': u'2007873',
```

```
Macintosh:Wrangle-OSM-Data SherryT$ python testquery.py
[{u'_id': u'Singapore', u'count': 759},
 {u'_id': u'Johor Bahru', u'count': 419},
 {u'_id': u'SKUDAI', u'count': 5},
 {u'_id': u'Sembawang', u'count': 2},
 {u'_id': u'Danga Bay', u'count': 1},
 {u'_id': u'Permas Jaya', u'count': 1},
 {u'_id': u'Batam', u'count': 1},
 {u'_id': u'JB', u'count': 1},
 {u'_id': u'Taman Century', u'count': 1},
 {u'_id': u'Masai', u'count': 1},
 {u'_id': u'Woodlands Spectrum II', u'count': 1},
 {u'_id': u'Batam Kota', u'count': 1}]
Macintosh:Wrangle-OSM-Data SherryT$
```

Solution:

Additional code was included to check the values of “addr:city” during the transformation of the OSM file in *data.py* so that these data points could be excluded.

- **There were 4 instances of postcode reflected as “<different>”.**

  Querying revealed that they were all located at the same street “Tuas Avenue 11”. Further querying was done to see if there were other entries located at “Tuas Avenue 11” with the postcode field filled in (so that I could use the info and complete the 4 entries with missing post codes) . Unfortunately, no other entries could be found. Since there are only 4 instances of this, this problem will not be further addressed in this project, but a future solution could be to manually check the postcodes against another map reference and then to update the entries.

As a final check, the full Singapore OSM data set was run against the *audit-data.py* file to pick up any other inconsistencies. The following was found:

A) Misspelled street types
B) New Street Type names
C) Postal Codes preceded by 'S' or 'Singapore'

The 'expected' array and 'mapping' dictionary were updated to clean instances of (A) and (B).

For (C), the code was updated to strip the redundant text 'S' and 'Singapore' with the following results

```
Macintosh:Wrangle-OSM-Data SherryT$ python audit-data.py
set(['79000', '80739', '80100', '437 437', '80800', '135', 'Johor Bahru', '80150
', '79250', '29463', '29461', '29466', '29464', '59765', '81900', '80200', '<dif
ferent>', 'S118556', '81000', '29426', '29427', '81620', '81210', '80000', '8131
0', '80300', '81700', '2424', '05901', '81800', '74', 'Singapore 408564', '81550
', '80250', '80050', '81200', '29456', '31', '80350', '81100', '80463', '81110',
 '79200', 'S120517', '81750', '80464', '80400', 'Bukit Batok Street 25', '81300'
, '29433', '29432'])
S118556 => 118556
Singapore 408564 =>  408564
S120517 => 120517
```

The full Singapore OSM data was then processed with the *data.py* file and imported to MongoDB for querying

2. Data Overview

This section contains basic statistics about the dataset and the MongoDB queries used to gather them.

File Sizes

singapore.osm 194.7MB
singapore.osm.json .... 219.4 MB

Number of documents

```
> db.SG.find().count()
999493
```

Number of nodes

```
> db.SG.find({"type":"node"}).count()
868489
```

Number of ways

```
> db.SG.find({"type":"way"}).count()
130961
```

Number of unique users

```
> db.SG.distinct("created.user").length
1108
```

Top 1 contributing user

```
db.SG.aggregate([{'$group':{'_id':'$created.user', 'count':{'$sum':1}}},
{'$sort':{'count':-1}},{'$limit':1}]  )
{ "_id" : "JaLooNz", "count" : 291164 }
```

Number of users appearing only once (having 1 post)

```
> db.SG.aggregate([{'$group':{'_id':'$created.user', 'count':{'$sum':1}}},
{'$group':{'_id':'$count', 'freq':{'$sum':1}}}, {'$sort':{'_id':+1}},
{'$limit':1}]
... )
{ "_id" : 1, "freq" : 209 }
```

“freq” represents the number of users having “_id” number of appearances

3. Additional Ideas

Contributor statistics

Users generally seem relatively active in their contributions. Out of 1108 distinct users, approximately 80% of users contributed more than once .

Additional data exploration using MongoDB queries

Top 10 appearing amenities

```
      > db.SG.aggregate([ {'$match':{'amenity':{'$exists':1}}},{'$group':
      {'_id':'$amenity', 'count':{'$sum':1}}},{'$sort':{'count':-1}},
      {'$limit':10}])
{ "_id" : "parking", "count" : 1821 }
{ "_id" : "restaurant", "count" : 996 }
{ "_id" : "place_of_worship", "count" : 800 }
{ "_id" : "school", "count" : 644 }
{ "_id" : "swimming_pool", "count" : 302 }
{ "_id" : "fuel", "count" : 258 }
{ "_id" : "fast_food", "count" : 205 }
{ "_id" : "cafe", "count" : 182 }
{ "_id" : "shelter", "count" : 176 }
{ "_id" : "toilets", "count" : 170 }
```

Singapore has high car ownership, but having parking at the top of the list is somewhat surprising because most carparks are multi-story carparks

Top 5 fast food chains

```
>> db.SG.aggregate( [ {'$match':{'amenity':'fast_food', 'name':{'$exists':1}}},
{'$group':{'_id':'$name', 'count':{'$sum':1}}},{'$sort':{'count':-1}},
{'$limit':5}])
{ "_id" : "McDonald's", "count" : 62 }
{ "_id" : "KFC", "count" : 33 }
{ "_id" : "Burger King", "count" : 13 }
{ "_id" : "Subway", "count" : 9 }
{ "_id" : "Popeye", "count" : 4 }
>
```

Having McDonald's at the top of the list is expected as you can see a McDonald's almost everywhere you go, compared to other fast food chains.

Top 5 cafes

```
> db.SG.aggregate([ {'$match':{'amenity':'cafe', 'name':{'$exists':1}}},
{'$group':{'_id':'$name', 'count':{'$sum':1}}},{'$sort':{'count':-1}},
{'$limit':5}])
{ "_id" : "Starbucks", "count" : 22 }
{ "_id" : "The Coffee Bean & Tea Leaf", "count" : 8 }
{ "_id" : "Toast Box", "count" : 4 }
{ "_id" : "Starbucks Coffee", "count" : 3 }
{ "_id" : "Old Town White Coffee", "count" : 2 }
```

Again, no surprise here as Starbucks is ubiquitous. There may be a need to clean the data further to standardize the names appearing in the data i.e. Starbucks vs Starbucks Coffee

Top 5 religions

```
> db.SG.aggregate([ {'$match':{'amenity':"place_of_worship", 'religion':
{'$exists':1}}},{'$group':{'_id':'$religion', 'count':{'$sum':1}}},{'$sort':
{'count':-1}},{'$limit':5}])
{ "_id" : "muslim", "count" : 440 }
{ "_id" : "christian", "count" : 176 }
{ "_id" : "buddhist", "count" : 62 }
{ "_id" : "hindu", "count" : 16 }
{ "_id" : "taoist", "count" : 7 }
```

Having Muslim at the top was extremely surprising because the Malays/Muslims are not a majority group in Singapore. The map contributions could have been skewed. Alternatively, it could be that there is less variation among the Malays when it comes to religion (i.e. most Malays are Muslims). On the other hand, the majority Chinese ethnic group has a wide variation in terms of religions practiced and are quite evenly split among Christian, Buddhism and Taoism. This would result in fewer places of worship for each of those religions due to the effect of dilution.

Additionally, Muslims and Christians generally have the practice of visiting mosques or churches on a weekly basis (Friday prayers for Muslims and Sunday worship for Christians). Buddhists and Taoists do not have such practices and some have altars in their own homes. There would therefore be a lesser need for Buddhist and Taoist temples to be built, and therefore a lower count as shown in the query.

Top 5 cuisines

```
> db.SG.aggregate([ {'$match':{'amenity':'restaurant', 'cuisine':
{'$exists':1}}},{'$group':{'_id':'$cuisine', 'count':{'$sum':1}}},{'$sort':
{'count':-1}},{'$limit':5}])
{ "_id" : "chinese", "count" : 86 }
{ "_id" : "korean", "count" : 31 }
{ "_id" : "italian", "count" : 28 }
{ "_id" : "japanese", "count" : 27 }
{ "_id" : "pizza", "count" : 22 }
```

As expected, Singapore with its majority Chinese population has Chinese as its top cuisine.

4. Conclusion

While the data has been cleaned sufficiently to enable some insights to be gleaned from the MongoDB queries, further improvements can be made to enhance the data quality.

In this project, the focus was on cleaning addresses. The cleaning could be extended to the names of amenities (e.g. As was pointed out earlier, Starbucks has some inconsistent entries in 'names').

It was also difficult to clean some data that were obviously erroneous, such as data being entered in the wrong field (e.g. 'postcode' entry swopped with 'street' entry) or missing postal codes. Data fields could be cross-checked against each other or against a trusted reference to rectify this. However, it could be more challenging to address erroneous data as these occur quite randomly. The nature of the data error could be quite varied as well. For instance, the approach to dealing with a missing postcode would be to check an external reference using other address fields, whereas. On the other hand, swopped data fields may merely require internal cross-checking and updating without having to consult external sources. Due to the varied nature of erroneous data, it could be challenging to predict where these errors are and the best strategy to deal with them without manual eyeballing.

As there are many applications in Singapore giving information on best places to park, bus routes, listings of amenities etc, it might be worth exploring whether the databases contained in those applications could be mapped and matched to Openstreet map data in an automated way rather than relying on user contributions.