P4OpenStreetMap_Wrangling

March 30, 2019

```
In [ ]: # import required libraries

        import xml.etree.cElementTree as ET
        from collections import defaultdict
        import pprint
        import re
        import csv
        import codecs
        import cerberus
        import schema
        import os
        import sqlite3
        import sys
        import string
        import pymysql
```

1 OpenStreetMap Data Wrangling with Python and SQLite

Author: Sara Uenoyama
Date: August 4th, 2018

1.1 0. Introduction

As a final project of Udacity Data Analyst Nanodegree Program 4, Data Wrangling, I would like to audit and explore the OpenStreetMap Data of Singapore. After spending half a year as an exchange student, I am feeling more familiar with this lion city, and also would like to contribute to its map data through this project.

1.2 1. Map Area

The area of the map we are going to investigate here is the whole land of Singapore.

"singapore.osm" is the data I downloaded from the OpenStreetMap website: https://www.openstreetmap.org/search?query=singapore#map=11/1.2904/103.8517

I am originally from Japan, but after spending half a year and visiting several times, Singapore became second home country to me. That's why I'm interested in seeing what database querying reveals, and I'd also like to take an opportunity to contribute to its improvement on OpenStreetMap.org.

1.3 1-2. Overview the data

Folloing the approach from the Udacity course on Data Wrangling, I would go through the input data step-by-step.

1. Audit the data: identify errors/missing or generally "dirty" data in the original XML file
2. Create a data cleaning plan based on the audit

- Identify the causes of any inconsistent/incorrect data
- Develop a set of corrective cleaning actions and test on a small sample of the XML data

3. Implement the data cleaning plan: run cleaning scripts and transfer the cleaned data to .csv files
4. Manually correct as necessary: import the data from .csv files to a SQL database and perform SQL queries on the data to identify any further inconsistencies that would necessitate returning to step 1.

1.3.1 1-2-1. Tags within the data

First, to get an overview of the data, we would find tag names and the numbers of each tag. After running 1_mapparser.py, I received the following results:

```
{'bounds': 1,
'member': 127066,
'meta': 1,
'nd': 1193086,
'node': 944343,
'note': 1,
'osm': 1,
'relation': 3030,
'tag': 523838,
'way': 151035}
```

1.3.2 1-2-2. Problematic characters

Next, to explore the data, I created 3 regular expressions to find tags only with lower cases, tags with a colon in their names and tags with problematic characters.

Using 2_tags.py, I found the results below:

- 'lower' : 419805, that is 419805 tags are only with lower case
- 'lower_colon' : 102938, that is 102938 tags have a colon in their names
- 'problemchars' : 3, that is 3 tag is with problematic characters
- 'other' : 1092, that is 1092 tags do not fall into any category above

1.4 2. Problems Encountered in the Map

1.5 2-1. Inconsistent street names

I have found many inconsistencies with its street names when we run the inital 3_audit.py. So I updated its mapping variable and corrected the inconsistent street names as below:

- Abbreviations
  - Rd -> Road
  - rd -> Road
  - jln -> Jalan : street
  - Jln -> Jalan : street
  - Lor -> Lorong : path
  - Blk -> Block
  - blk -> Block
  - Cl -> Close
  - Dr -> Drive
  - Bkt -> Bukit : hill
  - Bt -> Bukit : hill
  - Upp -> Upper
- Incorrect street names
  - Jalan Yayang Layang -> Jalan Layang Layang
- LowerCase
  - jalan kubor -> Jalan Kubor
  - 41 lorong 16 geylang-> 41 Lorong 16 Geylang
  - jln afifi -> Jalan Afifi
  - lornie rd -> Lornie Road
  - Gloucester road -> Gloucester Road
  - Nanson road -> Nanson Road
  - Yuan Ching road -> Yuan Ching Road
  - chinese garden road -> Chinese Garden Road
  - 1801 Ang Mo kio avenue 1 -> 1801 Ang Mo Kio Avenue 1
  - lorong 2 -> Lorong 2
  - serangoon avenue -> Serangoon Avenue
  - kalidassa avenue -> Kalidassa Avenue
  - 2 kensington park drive -> 2 Kensington Park Drive
  - ah soo garden -> Ah Soo Garden
- Misspelling
  - Aenue -> Avenue
  - Avneue -> Avenue
  - Ajunied -> Aljunied
  - Aliasr -> Alias
  - Gelyang -> Geylang
  - Ridgwood -> Ridgewood
  - Roadc -> Road
- UpperCase Words
  - AveNue -> Avenue

Also, I revised the expected variable of 3_audit.py, so that it covers the following Malay words for street name:

- Other languages names
  - Jalan -> road
  - Lorong -> path
  - Lengkong -> curve

1.6 2-2. Abbreviations

This is the updated mapping variable. I have created this dictionary to translate the abbreviations into those proper long expression.

mapping =
{'St': 'Street',
'St.': 'Street',
'Ave': 'Avenue',
'Rd.': 'Road',
'Rd' : 'Road',
'rd' : 'Road',
'jln' : 'Jalan',
'Jln' : 'Jalan',
'Lor' : 'Lorong : path',
'Blk' : 'Block',
'blk' : 'Block',
'Cl' : 'Close',
'Dr' : 'Drive',
'Bkt' : 'Bukit',
'Bt' : 'Bukit',
'Upp' : 'Upper'}

Abbreviation is corrected by replacing the abbreviated words with correct long expressions.

1.7 2-2. Inconsistent postal codes

I run the following codes on terminal with sqlite3 to show postal codes.

```
In [ ]: %load_ext sql
```

```
In [ ]: %%sql
        SELECT tags.value, COUNT(*) as count
        FROM (SELECT * FROM nodes_tags
                  UNION
              SELECT * FROM ways_tags) tags
        WHERE tags.key='postcode'
        GROUP BY tags.value
        ORDER BY count DESC;
```

As a result:
039594 | 4 120106 | 2 129588 | 2 179030 | 2 187965 | 2 278115 | 2 449269 | 2 469001 | 2 569830 | 2 588177 | 2 598749 | 2 640498 | 2 640638 | 2 688692 | 2 689810 | 2 038988 | 1 039799 | 1 039803 | 1 049864 | 1 059389 | 1 118177 | 1 119080 | 1 119082 | 1 119219 | 1 119613 | 1 120206 | 1 120301 | 1 120302 | 1 120310 | 1 120331 | 1 120340 | 1 120350 | 1 120357 | 1 120367 | 1 120381 | 1 120414 | 1 120417 | 1 120426 | 1 120428 | 1 120446 | 1

120451 | 1 120455 | 1 120466 | 1 120506 | 1 120512 | 1 120516 | 1 120605 | 1 120609 | 1 120708 | 1 120721 | 1
122209 | 1 122311 | 1 123312 | 1 127158 | 1 127999 | 1 128355 | 1 129580 | 1 129956 | 1 130005 | 1 130017 | 1
130027 | 1 130028 | 1 135 | 1 138588 | 1 138617 | 1 138648 | 1 138669 | 1 138673 | 1 138839 | 1 140045 | 1
140132 | 1 140149 | 1 140158 | 1 140167 | 1 141019 | 1 141055 | 1 141085 | 1 141087 | 1 144091 | 1 148951 | 1
149061 | 1 149544 | 1 149732 | 1 150053 | 1 150079 | 1 158750 | 1 159012 | 1 159924 | 1 159945 | 1 159950 | 1
159956 | 1 159960 | 1 169663 | 1 178895 | 1 179024 | 1 179031 | 1 179103 | 1 179369 | 1 179429 | 1 188329 | 1
188505 | 1 188537 | 1 188592 | 1 188980 | 1 189560 | 1 189619 | 1 189673 | 1 189699 | 1 198497 | 1 198501 | 1
199149 | 1 199207 | 1 199323 | 1 199588 | 1 207229 | 1 207551 | 1 207569 | 1 207630 | 1 208524 | 1 208532 | 1
208786 | 1 209210 | 1 209664 | 1 209924 | 1 217562 | 1 217567 | 1 218578 | 1 219458 | 1 228091 | 1 228210 | 1
228211 | 1 228517 | 1 229813 | 1 237983 | 1 237994 | 1 238372 | 1 238853 | 1 238857 | 1 238870 | 1 238874 | 1
238884 | 1 238895 | 1 238896 | 1 247911 | 1 247913 | 1 247964 | 1 247966 | 1 248322 | 1 249679 | 1 259281 | 1
259366 | 1 259595 | 1 259772 | 1 259954 | 1 269707 | 1 270008 | 1 275764 | 1 276954 | 1 277116 | 1 278621 | 1
288162 | 1 288683 | 1 298186 | 1 307470 | 1 307506 | 1 308232 | 1 308433 | 1 310120 | 1 310190 | 1 310520 | 1
319191 | 1 319258 | 1 319757 | 1 320009 | 1 320097 | 1 320099 | 1 322101 | 1 324108 | 1 327874 | 1 328836 | 1
329901 | 1 330073 | 1 330095 | 1 339096 | 1 339914 | 1 347694 | 1 349323 | 1 350131 | 1 350143 | 1 357844 | 1
368125 | 1 370033 | 1 370086 | 1 380018 | 1 380045 | 1 387416 | 1 387440 | 1 387494 | 1 389200 | 1 389226 | 1
389752 | 1 390041 | 1 390051 | 1 398738 | 1 398824 | 1 399043 | 1 399096 | 1 399121 | 1 399719 | 1 399772 | 1
400343 | 1 400411 | 1 408654 | 1 409009 | 1 409075 | 1 409179 | 1 410101 | 1 410111 | 1 417943 | 1 418472 | 1
418730 | 1 419529 | 1 427726 | 1 428996 | 1 429538 | 1 438859 | 1 439012 | 1 440083 | 1 448880 | 1 449282 | 1
460212 | 1 460510 | 1 465492 | 1 467352 | 1 468980 | 1 468982 | 1 469032 | 1 469626 | 1 469680 | 1 469718 | 1
470104 | 1 470109 | 1 470602 | 1 470613 | 1 470618 | 1 470632 | 1 479220 | 1 479266 | 1 510721 | 1 510737 | 1
510752 | 1 510756 | 1 510764 | 1 510767 | 1 510772 | 1 520728 | 1 520733 | 1 520829 | 1 520871 | 1 520877 | 1
528933 | 1 530106 | 1 530110 | 1 530211 | 1 530356 | 1 530558 | 1 534057 | 1 538692 | 1 538830 | 1 538884 | 1
539775 | 1 539975 | 1 541162 | 1 541279 | 1 541281 | 1 541305 | 1 541331 | 1 541336 | 1 542331 | 1 543301 | 1
543334 | 1 544305 | 1 544338 | 1 544690 | 1 545078 | 1 545079 | 1 547809 | 1 550153 | 1 550241 | 1 550261 | 1
556111 | 1 556114 | 1 556123 | 1 560202 | 1 560233 | 1 560532 | 1 563226 | 1 567749 | 1 569061 | 1 569250 | 1
569277 | 1 569405 | 1 569663 | 1 569843 | 1 569920 | 1 569933 | 1 570123 | 1 570164 | 1 570175 | 1 570282 | 1
573911 | 1 573916 | 1 579782 | 1 579827 | 1 587976 | 1 588179 | 1 588192 | 1 588216 | 1 588996 | 1 590010 | 1
596468 | 1 597610 | 1 598381 | 1 598390 | 1 598436 | 1 598480 | 1 598490 | 1 600031 | 1 600038 | 1 600130 | 1
600135 | 1 608549 | 1 608567 | 1 609606 | 1 609961 | 1 609967 | 1 610140 | 1 610151 | 1 610161 | 1 610176 | 1
610181 | 1 610399 | 1 611337 | 1 618656 | 1 640526 | 1 640762 | 1 640815 | 1 640827 | 1 640831 | 1 640836 | 1
640962 | 1 640966 | 1 641601 | 1 642659 | 1 648346 | 1 648886 | 1 648967 | 1 649040 | 1 649846 | 1 650108 | 1
650117 | 1 650120 | 1 650140 | 1 650148 | 1 650156 | 1 650165 | 1 650184 | 1 650188 | 1 650209 | 1 650220 | 1
650227 | 1 650233 | 1 650237 | 1 650241 | 1 650243 | 1 650252 | 1 650264 | 1 650274 | 1 650292 | 1 650307 | 1
650322 | 1 650323 | 1 650352 | 1 650362 | 1 650366 | 1 650385 | 1 650413 | 1 650431 | 1 650434 | 1 650435 | 1
650447 | 1 650468 | 1 650530 | 1 650537 | 1 650620 | 1 651194 | 1 651443 | 1 651453 | 1 652438 | 1 652461 | 1
652463 | 1 653293 | 1 653449 | 1 653450 | 1 654440 | 1 656290 | 1 658079 | 1 658713 | 1 659003 | 1 659163 | 1
659204 | 1 659289 | 1 659440 | 1 659528 | 1 659578 | 1 659920 | 1 670143 | 1 670184 | 1 670257 | 1 670445 | 1
670628 | 1 677670 | 1 677737 | 1 677742 | 1 677899 | 1 679938 | 1 680116 | 1 680253 | 1 688688 | 1 688690 | 1
688691 | 1 688695 | 1 688848 | 1 688892 | 1 689379 | 1 689814 | 1 698918 | 1 719164 | 1 730900 | 1 732570 | 1
737888 | 1 760103 | 1 760323 | 1 760362 | 1 760866 | 1 768675 | 1 768857 | 1 769028 | 1 769092 | 1 787113 | 1
790452 | 1 790454 | 1 791408 | 1 791418 | 1 791435 | 1 791445 | 1 792411 | 1 793405 | 1 797650 | 1 798725 | 1
798727 | 1 798731 | 1 798755 | 1 798762 | 1 798781 | 1 799059 | 1 799067 | 1 799070 | 1 799072 | 1 799106 | 1
799296 | 1 799507 | 1 805073 | 1 805302 | 1 805418 | 1 805419 | 1 805434 | 1 805444 | 1 805467 | 1 805483 | 1
805493 | 1 805499 | 1 805502 | 1 805505 | 1 805514 | 1 805523 | 1 805658 | 1 805660 | 1 805672 | 1 805684 | 1
805697 | 1 805708 | 1 805716 | 1 805729 | 1 805731 | 1 805733 | 1 805741 | 1 805746 | 1 805758 | 1 805764 | 1
805774 | 1 805794 | 1 805824 | 1 805911 | 1 805917 | 1 805923 | 1 805935 | 1 805980 | 1 805993 | 1 806014 | 1
806036 | 1 806048 | 1 806049 | 1 806052 | 1 806075 | 1 806092 | 1 806097 | 1 806118 | 1 806128 | 1 806759 | 1

```
806767 | 1 806795 | 1 806797 | 1 806818 | 1 806845 | 1 806873 | 1 806876 | 1 806886 | 1 806892 | 1 806895 | 1
807001 | 1 807008 | 1 807248 | 1 807252 | 1 807272 | 1 807288 | 1 807294 | 1 807295 | 1 807300 | 1 807312 | 1
807313 | 1 807325 | 1 807349 | 1 807380 | 1 807384 | 1 807396 | 1 807398 | 1 807402 | 1 807404 | 1 807420 | 1
807438 | 1 807451 | 1 807452 | 1 807458 | 1 807512 | 1 807519 | 1 807543 | 1 807550 | 1 807558 | 1 807611 | 1
807612 | 1 807616 | 1 807654 | 1 807659 | 1 807669 | 1 807695 | 1 807697 | 1 807714 | 1 807736 | 1 807747 | 1
807748 | 1 807775 | 1 807792 | 1 807807 | 1 807810 | 1 807813 | 1 807827 | 1 807836 | 1 807839 | 1 807870 | 1
807885 | 1 807888 | 1 807889 | 1 807898 | 1 807904 | 1 807925 | 1 807936 | 1 807940 | 1 807954 | 1 807977 | 1
807989 | 1 808001 | 1 808002 | 1 808015 | 1 808026 | 1 808034 | 1 808042 | 1 808076 | 1 808141 | 1 808144 | 1
808148 | 1 808195 | 1 808216 | 1 808223 | 1 808227 | 1 808267 | 1 808270 | 1 808287 | 1 808288 | 1 808294 | 1
808306 | 1 808317 | 1 808331 | 1 808340 | 1 809428 | 1 809441 | 1 809456 | 1 809461 | 1 809478 | 1 809504 | 1
809513 | 1 809518 | 1 809552 | 1 809766 | 1 809769 | 1 809777 | 1 809781 | 1 809802 | 1 809843 | 1 809855 | 1
809858 | 1 809860 | 1 809864 | 1 809868 | 1 809888 | 1 809893 | 1 809910 | 1 809914 | 1 809946 | 1 809947 | 1
809948 | 1 809951 | 1 809952 | 1 809975 | 1 809984 | 1 820050 | 1 820053 | 1 820061 | 1 820070 | 1 820077 | 1
820104 | 1 820132 | 1 820218 | 1 820228 | 1 820266 | 1 820271 | 1 820312 | 1 820324 | 1 820326 | 1 820420 | 1
820651 | 1 820659 | 1 820660 | 1 820671 | 1 821207 | 1 821212 | 1 821232 | 1 821259 | 1 821316 | 1 821321 | 1
821408 | 1 821622 | 1 821632 | 1 821637 | 1 821661 | 1 821663 | 1 821673 | 1 821676 | 1 822224 | 1 822258 | 1
822270 | 1 822411 | 1 822415 | 1 822422 | 1 822423 | 1 822621 | 1 822641 | 1 822667 | 1 822672 | 1 822678 | 1
822683 | 1 823209 | 1 823256 | 1 823267 | 1 823617 | 1 823670 | 1 823683 | 1 824274 | 1 828716 | 1 828730 | 1
828812 | 1  | 1 Bukit Batok Street 25 | 1 S 642683 | 1 Singapore 408564 | 1
```

Postal codes in Singapore are ruled as 6 digits and made up with the 2 digits sector code and the 4 digits delivery point. There are 81 sectors in total.
Judging from this fact most of the postal codes are correct, but there are a few inconsistencies:

- 3 digits code: 135
- larger number of secotor code than 81: 820050, 820053, 820061 etc.

1.7.1 3 digits code

I tried to run some sql codes to get more info about this 3 digits postal code.

```
In [ ]: %%sql
        SELECT *
        FROM nodes_tags
        WHERE key = 'postcode' AND value = '135';
```

As a result:

```
1318498347 | postcode | 135 | addr
```

With the id 1318498347, I could get even more info by running the codes below.

```
In [ ]: %%sql
        SELECT * FROM nodes_tags WHERE id = '1318498347';
```

```
1318498347 | postcode | 135 | addr
1318498347 | street | Jln Pelatina | addr
1318498347 | building | yes | regular
```

As results only show the location is on 'Jalan Pelantina', I needed more information to get its specific postal code.

```
In [ ]: %%sql
        SELECT * FROM nodes WHERE id = '1318498347';
```

1318498347 | 1.3504999 | 103.8347008 | oeoeoe | 178837 | 1 | 8394625 | 2011-06-10T07:12:04Z
latitudes = '1.3504999', longitude = '103.8347008'
So, the postal code at the plot of (latitudes = '1.3504999', longitude = '103.8347008') is '577269' and I corrected the code with following query.

```
In [ ]: %%sql
        UPDATE nodes_tags SET value= 577269
        WHERE value="135" and key ="postcode" and type='addr';
```

1.7.2 Larger number than 82

I searched more detailed story about postal codes in Singapore, then I found interesting fact behind. They use from 01 to 82 to represent each sector, skipping 74. 74 is considered as very unlucky number because it sounds like "will certainly die" or "will die in anger" () in Chinese. So as long as there is no postal codes starting from 74, which is not there within the results above, the postal code data is correct.

1.8 3. Sort cities by count, descending

```
In [ ]: %%sql
        SELECT tags.value, COUNT(*) as count
        FROM (SELECT * FROM nodes_tags UNION ALL
              SELECT * FROM ways_tags) tags
        WHERE tags.key LIKE '%city'
        GROUP BY tags.value
        ORDER BY count DESC;
```

Results,
Singapore | 1322
2 | 4
3 | 2
4 | 2
#01-62 | 1
1 | 1
15 | 1
16 | 1
262 | 1
30 | 1
5 | 1
593 | 1
Ang Mo Kio | 1
Pasir Gudang | 1
Sembawang | 1
singapore | 1
The first letter of 'singapore' is corrected as 'Singapore' with the following query.

```
In [ ]: %%sql
        UPDATE ways_tags
        SET value= 'Singapore'
        WHERE value='singapore' and key LIKE '%city';
```

1.9 4. Data Overview

1.10 4-1. File sizes:

- singapore.osm: 255.1 MB
- nodes.csv: 78.4 MB
- nodes_tags.csv: 3.5 MB
- ways.csv: 9.1 MB
- ways_nodes.csv: 28.9 MB
- ways_tags.csv: 14.5 MB
- singapore.db: 116.6 MB

1.11 4-2. Open database

In terminal:
sqlite3 singapore.db

1.12 4-3. Numer of nodes

```
In [ ]: %%sql
        SELECT COUNT (*) FROM nodes;
```

It resulted in 94435 nodes.

1.13 4-4. Numer of ways

```
In [ ]: %%sql
        SELECT COUNT (*) FROM ways;
```

It resulted in 15103 ways.

1.14 4-5. Numer of unique users

```
In [ ]: %%sql
        SELECT COUNT (DISTINCT(e.uid))
        FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
```

It resulted in 797 unique users.

1.15 4-6. Top contributing users

```
In [ ]: %%sql
        SELECT e.user, COUNT(*) as num
        FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
        GROUP BY e.user
        ORDER BY num DESC
        LIMIT 10;
```

As a result:
JaLooNz | 24793
happy-camper | 5686
cboothroyd | 3909
Luis36995 | 3698
ridixcr | 3489
CapAhab | 3132
Evandering | 3037
geoJenn | 2791
Lilles | 2634
KartoSliwka | 2426

1.16 4-5. Numer of users contributing only once

```
In [ ]: %%sql
        SELECT COUNT(*)
        FROM
            (SELECT e.user, COUNT(*) as num
             FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
             GROUP BY e.user
             HAVING num=1) u;
```

It resulted in 261 users who contributed only once.

1.17 5. Additional Ideas

1.18 5-1. Common ammenities:

```
In [ ]: %%sql
        SELECT value, COUNT(*) as num
        FROM nodes_tags
        WHERE key = 'amenity'
        GROUP BY value
        ORDER BY num DESC
        LIMIT 10;
```

Here are the results:
restaurant | 122
atm | 62
parking | 37
parking_entrance | 37
cafe | 30
fast_food | 27
taxi | 22
shelter | 16
bench | 15
toilets | 14

The most common ammenities are restaurants, followed by atms and parkings. It was a little surprise to me that there are less cafes than parkings, since driving cars are comparatively costly in Singapore.

1.19 6. Conclusion

The OpenStreetMapa data of Singapore area is fairly correct but better understandings towards its mixed culture to revise further, since both Malay / Chinese culture and languages are affecting the map data.

1.20 7. Additional Suggetion and Ideas

I found several entries with foreign languages such as probably Tamil (i.e.) and Korean (i.e.), asI looked through the singapore osm data. Controlling languages or at least letters used for map data will benefit users' readability. However, it might be hard to cover all the languages used to update osm data.

1.21 Files

- P4OpenStreetMap_Wrangling: this file
- README.pdf: pdf version of this file
- singapore.osm: available to download from this website https://www.openstreetmap.org/search?query=singapore#map=11/1.2904/103.8517
- 1_mapparser.py: find unique tags in the data
- 2_tags.py: find errors in the data
- 3_audit.py: audit atreet, city and update their names
- 4_create_db.ipynb: create csv files from osm data and create database using those csv files
- 4_create_db.py: python version of 4_create_db.ipynb
- schema.py: supplied schema to create database

1.22 References

1. carlward/sample_project.md
2. mabelvj/data-analyst-nanodegree