OpenStreetMap Data Case Study

Map Area

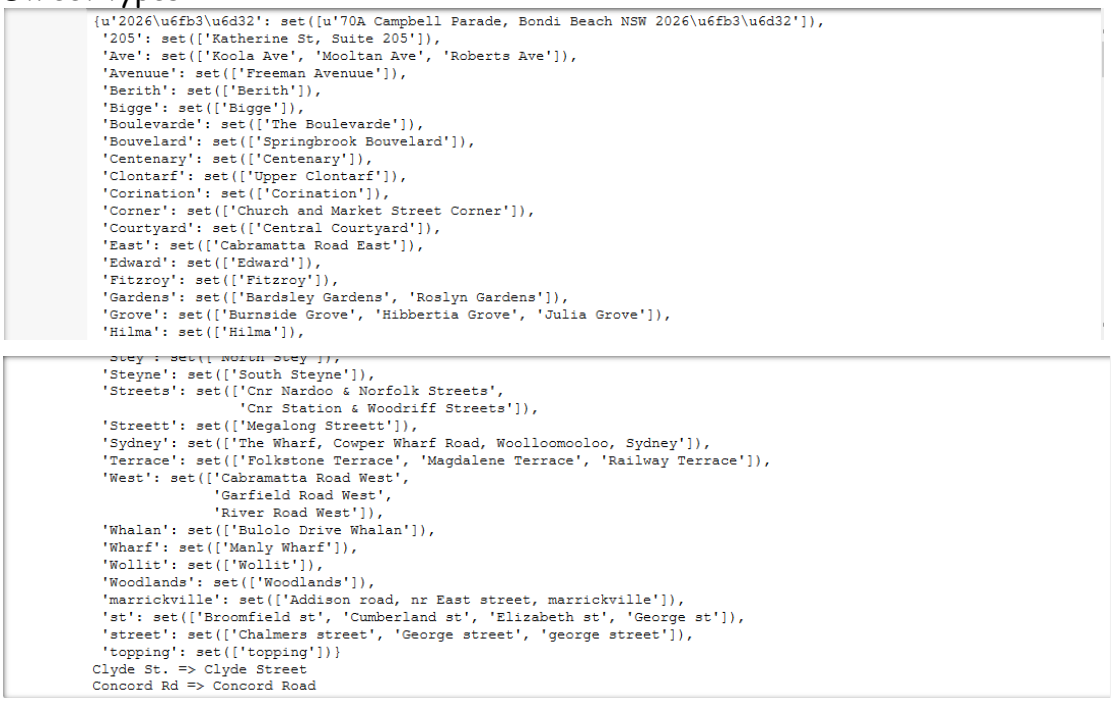
Sydney,NSW,Australia
https://www.openstreetmap.org/relation/5729534
https://mapzen.com/data/metro-extracts/metro/sydney_australia/

This project sets to explore part of the open street map data of the city of Sydney. At first, all data of the city was included. But every time I ran the codes, the computer would break down due to the large size of the original file "Sydney.osm" (328mb). So I had to run the provided codes to cut down half of the data and create another file "Sydney_big_sample.osm"(165mb), on which the following report is based.

Problems Encountered in the Map

I altered the above codes and created a small sample of the data "sample.osm" (11mb). And I discovered that there were some problems associated with the data. Then I wrote some codes to deal with those problems. And I ran those codes against the whole file and found out some new problems.
The following problems exist in the whole dataset:

1. Street types:



```
{u'2026\u6fb3\u6d32': set([u'70A Campbell Parade, Bondi Beach NSW 2026\u6fb3\u6d32']),
 '205': set(['Katherine St, Suite 205']),
 'Ave': set(['Koola Ave', 'Mooltan Ave', 'Roberts Ave']),
 'Avenuue': set(['Freeman Avenuue']),
 'Berith': set(['Berith']),
 'Bigge': set(['Bigge']),
 'Boulevarde': set(['The Boulevarde']),
 'Bouvelard': set(['Springbrook Bouvelard']),
 'Centenary': set(['Centenary']),
 'Clontarf': set(['Upper Clontarf']),
 'Corination': set(['Corination']),
 'Corner': set(['Church and Market Street Corner']),
 'Courtyard': set(['Central Courtyard']),
 'East': set(['Cabramatta Road East']),
 'Edward': set(['Edward']),
 'Fitzroy': set(['Fitzroy']),
 'Gardens': set(['Bardsley Gardens', 'Roslyn Gardens']),
 'Grove': set(['Burnside Grove', 'Hibbertia Grove', 'Julia Grove']),
 'Hilma': set(['Hilma']),
 'Steyne': set(['South Steyne']),
 'Streets': set(['Cnr Nardoo & Norfolk Streets',
                 'Cnr Station & Woodriff Streets']),
 'Streett': set(['Megalong Streett']),
 'Sydney': set(['The Wharf, Cowper Wharf Road, Woolloomooloo, Sydney']),
 'Terrace': set(['Folkstone Terrace', 'Magdalene Terrace', 'Railway Terrace']),
 'West': set(['Cabramatta Road West',
              'Garfield Road West',
              'River Road West']),
 'Whalan': set(['Bulolo Drive Whalan']),
 'Wharf': set(['Manly Wharf']),
 'Wollit': set(['Wollit']),
 'Woodlands': set(['Woodlands']),
 'marrickville': set(['Addison road, nr East street, marrickville']),
 'st': set(['Broomfield st', 'Cumberland st', 'Elizabeth st', 'George st']),
 'street': set(['Chalmers street', 'George street', 'george street']),
 'topping': set(['topping'])}
Clyde St. => Clyde Street
Concord Rd => Concord Road
```

First, there were abbreviated street names like "Ave" and wrong names like "Streett" due to typing mistakes. Also, some of the places' names like "Addison

road, nr East street, marrickville" were too specific.
I wrote the following codes to solve these problems:

```
mapping = { "St": "Street",
            "St.": "Street",
            "st": "Street",
            "Ave": "Avenue",
            "Avenuue": "Avenue",
            "Rd": "Road",
            "street": "Street",
            "Pl": "Place"
            }
big_names = [" Woolloomooloo", " Sydney", " marrickville"]
```

```
def audit_street_type(name):
    m = street_type_re.search(name)
    if m:
        street_type = m.group()
        if street_type in mapping.keys():
            name = name.replace(street_type,mapping[street_type])
        for big in big_names:
            if big in name:
                name = name.replace(big,"")
    return name
```

As there were only two places that included the names of the county and the city, I created a list that had the names of the two county and that of the city. Then I iterated the whole dataset to find out the two places and changed their names

2. Postal codes

The data of postal code were relatively clean, except that a few of them had "NSW", the abbreviated name of the state where Sydney is located.

```
'NSW 2026', '2010', 'NSW 2022',
```

I wrote the following code to leave it out:

```
def audit_postalcode(code):
    if 'NSW' in code:
        code = code.replace("NSW","").strip()
    return code
```

3. House numbers

```
['138', '81', '509', '384', '1', '75', '40/44', '48', '5-11', '557', '744', '665', '2-6', '70', '30-36', '416,418',
'24', '82', '45', '1-25', '36', '60', '75', '371', '10', '15', '75', '78', '161', '330', '99', '2', '4', '24a', '8
6', '145', '65', '54', '137', '242', '470', '4-12', '576', '60-64', '110-114', '87', '255', '502-504', '63', '60',
'17', '162', '2', '1', '27', '310', '52', '150', '747', '494', '2-6', '175', '477', '340', '651', '367', '64', '13
6', '191', '318', '1', '262', '4', '63', '505-525', '51', '1', '29-35', '7', '311', '24', '8-10', '390-392 Pittwate
r Road', '30', '287', '287', '2', '1', '37-39', '151', '315', '147-151', 'Gate A', '14', '123', '131', '14', '18',
'28', '34', '40B', '6', '107', '386', '67', '2', '6', '3/54', '1', '101', 'Unit 4 / 53', '371', '116', '12', '23',
'831', '653', '24', '2-4', '74', '53', '133', '2', '65', '61', '67', '69', '16', '28', '12', '14', '8', '22', '34',
'32A', '36A', '90', '34', '27', '29', '15', '35', '33', '63', '202', '74', '50', '74', '118', '224', '260', '74',
'74', '72', '7 - 9', '135', '15', '243', '440', '71', '87', '188', '122', '7', '1', '49', '22', '30', '7', '133',
'758', '633', '3/100', '37', '1206', '156', '143', '70', '596', '41', '55', '32', '68', '4', '2', '2', '34', '38',
'10', '5', '3', '11', '15', '19', '23', '25', '27', '29', '7', '53', '209', '337', '71', '50', '375', '2', '36', '9
3', '12', '17', '22', '12', '38', '42', '6', '30', '46', '14', '2', '33', 'Shop 106/7/8 Westfield Hornsby', '92',
'20', '163', '13', '2', '5', '9', '57', '47', '51', '55', '11', '9', '2', '4', '6', '77', '81', '9', '6', '9', '1
1', '2', '4', '8', '50', '54', '370', '32', '36', '10', '14', '18', '22', '26', '30', '2', '23', '27', '31', '53-5
5', '208', '46-50', '275', '401', '418', '421', '425', '433', '441', '450', '463', '485', '488', '492', '497', '50
7', '517', '526', '534', '542', '554', '558', '571', '579', '602', '116', '120', '127', '131', '138', '142', '145',
'160', '190', '24', '155', '168', '176', '220', '234', '280', '287', '312', '320', '333', '363', '395', '403', '45
0', '462', '476', '531', '591', '603', '636', '647', '695', '708', '730', '762', '768', '7', '151', '168', '24', '5
```

There were a few problems concerned with the data. First, some house numbers had specific address like "Westfield Hornsby". Second, some house numbers had the capitalized "A" or "B", some had lowercase letters. Lastly, some had abbreviated "Lvl" while there were more house numbers containing "Level".
Here is the solution:

```
CHARNUM = re.compile(r'^[0-9]|[0-9]|[0-9]|[a-b]')  #change the capitalization of XXXa or XXXb, without changing the "a"
# Make sure the fields order in the csvs matches the column order in the sql table schema
```

```python
def audit_housenumber(number):
    if 'Lvl' in number:
        number = number.replace("Lvl","level")
    for big in big_numbers:
        if big in number:
            number = number.replace(big,"")
    if re.search(CHARNUM,number):
        number = number.replace("a","A")
        number = number.replace("b","B")
    return number
```

File size

sydney_sample_big.osm 165mb
sydney_sample_big.db 115mb
nodes.csv 60.6mb
nodes_tags.csv 2.96mb
ways.csv 5.92mb
ways_nodes.csv 20.8mb
ways_tags.csv 11.1mb

1. Number of nodes

```
1  SELECT COUNT(*) FROM nodes
```

	COUNT(*)
1	764249

2. Number of ways

```
1  SELECT COUNT(*) FROM ways
```

	COUNT(*)
1	104401

3. Number of unique contributors

```
SELECT COUNT(DISTINCT uid) FROM (SELECT uid FROM ways UNION SELECT uid FROM nodes)
```

	COUNT(DISTINCT uid)
1	2067

4. Top 10 contributing users

```
1  SELECT e.user,COUNT(*) as num FROM (SELECT user FROM ways UNION ALL SELECT user FROM nodes) e
2  GROUP BY e.user
3  ORDER BY num DESC
4  LIMIT 10
```

	user	num
1	balcoath	58690
2	inas	44603
3	TheSwavu	37159
4	aharvey	33171
5	ChopStiR	30216
6	ozhiker2	25920
7	Leon K	23292
8	cleary	20765
9	Rhubarb	20392
10	AntBurnett	18778

5. Number of users appearing only once (having 1 post)

```
1  SELECT COUNT(*)
2  FROM (SELECT user ,COUNT(*) as num
3        FROM (SELECT user FROM ways UNION ALL SELECT user FROM nodes) e
4        GROUP BY e.user
5        HAVING num = 1) u
```

	COUNT(*)
1	459

Data Review and Additional Ideas

I explored three areas of this data set---house number, postal code and street type. The data of postal code are the cleanest ones, perhaps because every contributor is very familiar with the format of postal code in Sydney. The problems in the data of house number are very tricky as I only have stayed in Sydney for a couple of weeks and could only deal with some very simple problems like different capitalization. As for the data of street type, the data are relatively dirty. There are abbreviated problems, typing mistakes and problems of wired street names.

Suggestion 1:

To solve the above problems, the website had better offer a reference of formats for users who would like to contribute. The reference could look like:

Postal code: Only 4-digit-numbers are included. E.g. 2292(√)　NSW,2292(×)

Street name: DO NOT use abbreviated street names. DO NOT capitalize the first letter of street type. COUNTY or CITY names should not be included. E.g. Hilma street(√) Hilma Street(×) Hilma street, Sydney(×)

...

Suggestion 2:

Also, the website could ask the contributors to recheck what they have written before they hand in.

Advantage of my two suggestions:

My two suggestions could make the formats of different fields look more uniform.

Problems:

However, there would be some users that do not follow the reference. Perhaps the website could hire someone to check the data every week. This could also help solve the problem of wired street names. Also, the reference and the reconfirming process may seem complicated and discourage users to contribute.

Additional data exploration

1. Top 10 appearing amenities

```
SELECT value, count(*) as num
FROM nodes_tags
WHERE key = 'amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 10
```

	value	num
1	bench	715
2	restaurant	489
3	cafe	462
4	drinking_water	427
5	parking	375
6	toilets	339
7	fast_food	306
8	bicycle_parking	299
9	place_of_worship	206
10	post_box	199

2. Biggest religion

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
     JOIN (SELECT DISTINCT (id) FROM nodes_tags WHERE value = "place_of_worship")i ON i.id = nodes_tags.id
WHERE nodes_tags.key = 'religion'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 1
```

	value	num
1	christian	175

3. Top 10 popular cuisine

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
     JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value = 'restaurant') i
     ON i.id = nodes_tags.id
WHERE key = 'cuisine'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 10
```

	value	num
1	thai	34
2	chinese	29
3	italian	28
4	pizza	24
5	japanese	18
6	indian	16
7	korean	9
8	vietnamese	8
9	greek	6
10	asian	5

Conclusion

Though the data of this project are incomplete, they are remarkably detailed, including specific information like the website of a particular shop. This should be attributed to 2067 users who have made efforts to improve the map. And through this project, I now know the process of cleaning data, building data and querying data. Thank you, Udacity!