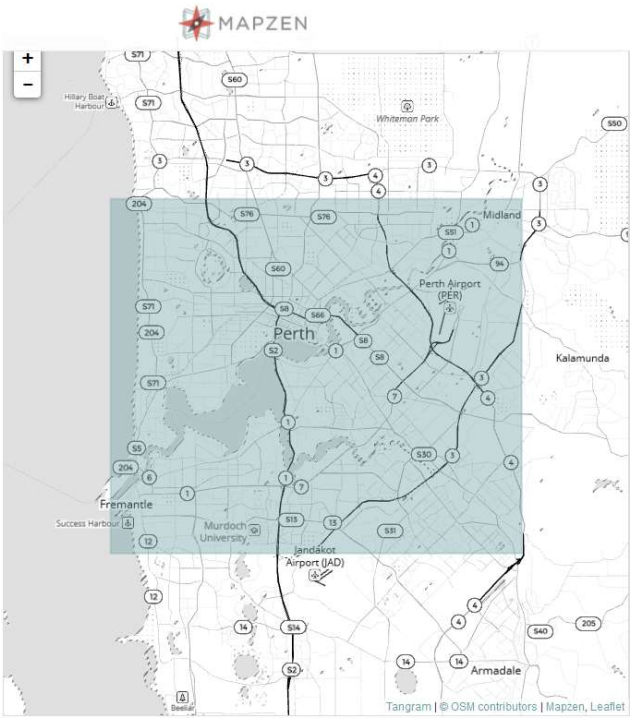
P3: Wrangle OpenStreetMap Data (SQL) Project

Project Objective

In this project, the quality of OpenStreetMap (OSM) XML data is audited and the problematic data is ironed out programmatically with Python. After the data cleaning, they are converted into .csv files according to its element tags, and stored in SQL database in order to perform some basic database queries.

Map Area

The OSM data chosen for this project is a custom extracted map of City of Perth, Western Australia from Map Zen.



Problems Encountered in the Map

The downloaded .osm file was run against audit.py file in order to spot problematic street names and street types in both node and way elements which as shown below:

- Street name with state code as suffix (`'Beaufort St WA'`)
- Missing capitalisation in name (`'Waterford plaza', 'hay street'`)
- Street name ended with name of suburb (`'Bulwer Street and Lord Street, Highgate'`)
- Abbreviated street types (`Ave`, `Cres`, `St`)
- Misspelled street types (`'Road,', 'Terriace', 'Boulevarde'`)

- House number entered into street name (`'170 Burswood Road', '27 / 168 Guildford Road'`)
- One building with two street names (`'Bulwer Street and Lord Street, Highgate', 'Corner Cape and Stoneham Streets'`)
- Use of apostrophe in street name (`'King's Park Road'`)
- Street name without street type (`'Bulwer', 'Cross', 'Beaufort'`)

The provisional data.py file used in the course was altered to accommodate this data clean-up.

The structure for these street names will be looked as multiple names and single name, in which the multiple names contains [‘And’, ‘and’, ‘&’] and single name does not. The multiple names will then be splitted into two single street names and each single street name will updated using name_checklist function.

Street name without street type

Some of the name was entered with missing Street suffix:

```
'Bulwer': set(['Bulwer']),
'Cross': set(['Cross']),
'Beaufort': set(['Beaufort'])
```

These was attempted to be fixed by running an if-loop to look for dictionary key that is the same as its value, then have ‘Street’ added after the name. However there are also these keys and values in street_types dictionary calling abort on the initial idea:

```
'Broadway': set(['Broadway']),
'Fairway': set(['Fairway', 'Meuse Fairway'])
```

The following code was then added into update_name function to do the job:

```
if len(name.split()) == 1 and name[-3:] != 'way':
        name = name + ' Street'
```

Fixes to problematic names

These are improved names after running problematic street/road names against update_name function in Appendix section:

```
Corner Cape and Stoneham Streets => ['Cape Street', 'Stoneham Street']
Barrow Cres => Barrow Crescent
Cross => Cross Street
hay street => Hay Street
Bulwer Street and Lord Street, Highgate => ['Bulwer Street', 'Lord Street']
Beaufort St WA => Beaufort Street
27 / 168 Guildford Road => Guildford Road
170 Burswood Road => Burswood Road
King's Park Road => Kings Park Road
Oxford => Oxford Street
Beasley Road, => Beasley Road
Beaufort => Beaufort Street
Fitzgerald St => Fitzgerald Street
Foundry St => Foundry Street
Newcastle St => Newcastle Street
Wordsworth Ave => Wordsworth Avenue
St Georges Terriace => St Georges Terrace
Bulwer => Bulwer Street
Waterford plaza => Waterford Plaza
Hardwick Boulevarde => Hardwick Boulevard
```

Statistical Data Overview

File sizes

```
city_of_perth.osm ............................... 107.27 MB
city_of_perth_sample.osm ........................ 2.69 MB
nodes.csv ....................................... 42.93 MB
nodes_tags.csv .................................. 1.58 MB
OSMData.db ...................................... 57.34 MB
P3_Final Project Report.pdf ..................... 0.64 MB
P3_Wrangle_OSM_Data_SQL_Final.py ................ 0.03 MB
ways.csv ........................................ 3.73 MB
ways_nodes.csv .................................. 13.38 MB
ways_tags.csv ................................... 4.68 MB
```

Number of nodes

```
sqlite> SELECT count(*) FROM nodes;
536489
```

Number of ways

```
sqlite> SELECT count(*) FROM ways;
65564
```

Number of unique users

```
sqlite> SELECT count(DISTINCT uid)
   ...> FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways);
663
```

Additional Statistics Overview

Top 10 contributing users

```
sqlite> SELECT user, COUNT(*) as num
   ...> FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways)
   ...> GROUP BY user
   ...> ORDER BY num DESC
   ...> LIMIT 10;
```

```
aaronsta|141127
SDavies|106813
sb9576|65764
andy_88|48109
browny_au|31519
Sam Wilson|24718
TRS-80|19982
drgis|11480
andrewpmk|10111
Andrew Gregory|9398
```

Top 10 amenities

```
sqlite> SELECT value, COUNT(*) as num
   ...> FROM nodes_tags
   ...> WHERE key = 'amenity'
   ...> GROUP BY value
   ...> ORDER BY num DESC
   ...> LIMIT 10;
```

```
bench|205
drinking_water|205
restaurant|197
cafe|178
post_box|164
fast_food|156
parking|109
bicycle_parking|99
telephone|97
toilets|81
```

Top 10 cuisines

```
sqlite> SELECT value, COUNT(*) as num
   ...> FROM nodes_tags
   ...> WHERE key = 'cuisine'
   ...> GROUP BY value
   ...> ORDER BY num DESC
   ...> LIMIT 10;
```

```
chinese|32
coffee_shop|26
pizza|23
indian|21
italian|19
fish_and_chips|16
japanese|14
sandwich|14
chicken|9
```

Additional Improvement Idea

I have noticed a misspelled road name in the downloaded osm data. Rokeby Road was misspelled as Rockeby Road (I am sure there is no Rockeby Road in Western Australia). That was just one that I came across, there could be more. The misspelling could probably be cross checked against a third party tools like Google Maps in data cleaning process.

Another dataset improvement could be done by location-based augmented reality game like Pokemon GO. Game users were to submit image and details of a location to game developer Niantic and had the submitted data updated onto OSM by Niantic. This improvement idea will have both benefit and problem in implementation.

Having users of popular games to update OSM dataset will speed up completion of map tremendously. However introducing human factor into data collection process will be followed by massive data cleaning task in need.

Conclusion

Some of the street names entered in the map are bound for human error and they have been well cleaned for the purposes of this project. It has been noticed that impressive effort by contributing users has been exerted in updating custom extracted map of City of Perth. This is shown by how accurate the map is especially for Perth CBD area. Amenities like drinking water fountains and toilets around the town which were overlooked on a daily basis are well documented.

Appendix

```
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)
problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\.\t\r\n]')
remove_numsym = re.compile(r'[a-zA-Z]+\s\w*')

expected = ["Street", "Avenue", "Boulvard", "Drive", "Court", "Place", "Square",
            "Lane", "Trail", "Parkway", "Commons", "Terrace"]

def name_checklist(name):

    if name[-2:] == 'WA':
        name = name.replace('WA', "")

    if len(name.split()) == 1 and name[-3:] != 'way':
        name = name + ' Street'

    if ',' in name:  # to remove ', suburb name' in street name
        fixed_name = name.split(',', 1)[0]
        name = fixed_name

    name = string.capwords(name)
    st_name = name.rsplit(None, 1)[-1]

    if st_name in mapping.keys():
    # replace abbreviated street type to mapped street type
        st_name_aud = mapping[st_name]
        name = name.replace(st_name, st_name_aud)

    elif st_name not in expected or st_name not in mapping.values():
        # suspected abbreviated street type could be misspelled instead
        d = enchant.Dict("en_AU")
        if d.check(st_name) == False: # if street type is misspelled
            suggest_list = d.suggest(st_name) # find suggested words

            # find suggested word that matches expected list
            replace_as = list(set(suggest_list).intersection(expected))
            name = name.replace(st_name, replace_as[0])
    else:
        return name

    if hasNums(name) == True:
        # remove numbers and symbols from street name
        fixed_name = remove_numsym.search(name).group(0)
        name = fixed_name

    # find any symbols such as comma or period
    problem = problemchars.findall(name)
    if problem and (set(name.split()).intersection(['Road', 'Street'])):
        name = name.replace(problem[0],"")

    return name
```

```
def update_name(name, mapping):

    and_patterns = ['And', 'and', '&']

    if set(name.split()) & set(and_patterns):
        pattern = list(set(name.split()).intersection(and_patterns))[0]
        multiple_names = re.split(pattern, name)

        for i, item in enumerate(multiple_names):
            fixed_item = and_name_checklist(item)
            multiple_names[i] = fixed_item

        name = multiple_names
    else:
        name = name_checklist(name)

    return name
```