# P3. OpenStreetMap Case Study - Milwaukee

## Map Area

Milwaukee, WI, United States

- https://www.openstreetmap.org/relation/251075 (https://www.openstreetmap.org/relation/251075)
- https://s3.amazonaws.com/metro-extracts.mapzen.com/milwaukee_wisconsin.osm.bz2 (https://s3.amazonaws.com/metro-extracts.mapzen.com/milwaukee_wisconsin.osm.bz2)

I selected Milwaukee because it is has been recommended to me by some friends recently as an interesting and underrated city to visit. I decided to use it for this project to see what the city has to offer, in preparation for a future visit.

## Problems Encountered

Through analysis of a small sample of the map, the data appeared to be very clean. However, on my first scan of the entire xml document, I came across a few issues with the data:

- Inconsistent street name abbreviations (ex: "Street" vs "St" vs "St."; or "Ave" vs "Ave.")
- Street names containing postal code information
- Inconsistent numeric postal code formats ("XXXXX" vs "XXXXX-XXXX")
- Postal codes containing non-numeric information (ex. "WI" as value of postal code)

### Inconsistent street naming conventions

There are a number of inconsistencies in street names. Some street types are abbreviated while others aren't (ex: "Street" vs. "St"). To solve this, abbreviated street types will be mapped to their unabbreviated forms. Street names with unusual street types (for example, numeric street types) will be mapped on a case-by-case basis.

There is one street name which contained only a postal code. To avoid a chance of error by including this point, this node will be omitted when importing to our database.

In [1]:
```
mapping_street_name = {'Ave': 'Avenue',     'Ave.': 'Avenue',
                       'Blvd': 'Boulevard', 'Ct': 'Court',
                       'Dr': 'Drive',       'Pkwy': 'Parkway',
                       'Rd': 'Road',        'Rd.': 'Road',
                       'St': 'Street',      'St.': 'Street'}
changes_street_name = {'HWY 164': 'Highway 164',
                       'N6620 HWY 164': 'N6620 Highway 164'}
skip_street_name = ['53076']

def update_street_name(name, mapping):
    m = street_type_re.search(name)
    if m:
        if (m.group() == '164') and (name in mapping.keys()):
            name = mapping[name]
        else:
            name = street_type_re.sub(mapping[m.group()], name)
        return name
```

### Inconsistent numeric postal code formats

For the database, we will try to keep the numeric format of postal codes consistent for all points. Therefore, postal codes found that do not match the common "XXXXX" numeric format are mapped to the correct format if possible or skipped if this conversion is not obvious.

In [2]:
```
mapping_postcode = {'53202-2001': '53202', '53203-3002': '53203',
                    '53212-3839': '53212', '53212-4099': '53212',
                    '53403-9998': '53403', '53217-5399': '53217',
                    '"Milwaukee WI, 53222"': '53222'}
skip_postcode = ['WI', '1729']

def update_postcode(name, mapping):
    m = postcode_re.search(name)
    if not m:
        if name in mapping.keys():
            name = mapping[name]
            return name
```

## Run Database Queries

The database consists of 5 tables (nodes, nodes_tags, ways, ways_tags, and ways_nodes), created from CSV data which was generated in the previous section.

### Sort cities by count, descending

In [ ]:
```
query = '''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
            LIMIT 8;'''
```

Results (edited for readability):

```
Milwaukee       2233
Racine          638
Mount Pleasant  238
Burlington      41
Sturtevant      35
Caledonia       32
Waukesha        32
MIlwaukee       23
```

While a large portion of the nodes are located in Milwaukee, there are a few which contain cities in the Milwaukee metropolitan area. Some cities (like Racine and Mount Pleasant) are actually a fairly long distance away from Milwaukee (25 miles between city centers for Milwaukee -> Racine).

Postal code data seems to be okay, with all listed postal codes starting with "53XXX". This indicates all postal codes are within some vicinity of Milwaukee

## Data Overview

### File sizes

```
milwaukee_wisconsin.osm..... 188.8 MB
milwaukee_osm.db............ 256.2 MB
nodes.csv...................  70.5 MB
nodes_tags.csv..............   2.4 MB
ways.csv....................   5.7 MB
ways_tags.csv...............  15.5 MB
ways_nodes.csv..............  24.6 MB
```

### Number of Nodes and Ways

In [ ]:
```
query = "SELECT COUNT(*) FROM nodes;"
query = "SELECT COUNT(*) FROM ways;"
```

Number of Nodes: 840764 (vs. 840763 from iterparse method)

Number of Ways: 94039 (vs. 94038 from iterparse method)

### Number of Unique Users

```
In [ ]: query = """SELECT COUNT(DISTINCT(e.uid))
                   FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;"""
```

641

### Top 10 contributing users

```
In [ ]: query = '''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;'''
```

```
woodpeck_fixbot 178980
shuui           136230
ItalianMustache 118347
reschultzed     61072
bbauter         29048
Gary Cox        26439
hogrod          25616
iandees         25396
Mulad           22393
TIGERcnl        21652
```

### Number of users appearing only once (having 1 post)

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

113

## Data Exploration

### Number of coffee shop / cafe locations, grouped by company (very important to me!)

It should be noted that Starbucks appears multiple times in this list (as well as some other cafes, such as Colectivo). This database could benefit from some data cleaning to standardize the names of these companies to a single spelling, to avoid redundant list items like those found below.

```
In [ ]: query = '''SELECT nodes_tags.value, COUNT(*) as num
                    FROM nodes_tags, (SELECT *
                                      FROM nodes_tags
                                      WHERE key="amenity" AND value = "cafe") as caf
                    WHERE nodes_tags.id = cafe.id AND nodes_tags.key="name"
                    GROUP BY nodes_tags.value
                    ORDER BY num DESC
                    LIMIT 10'''
```

```
Starbucks                9   (*)
Starbucks Coffee         3   (*)
Stone Creek Coffee       3
Colectivo Coffee         2
Dunkin' Donuts           2
Starbuck's               2   (*)
2894 On Main             1
600 East Cafe            1
8th Note Coffee House    1
Alderaan Coffee          1
```

**Most popular cuisines in the city**

Being the foodie that I am, I'm always interested in seeing what types of cuisines are offered in any new city I visit.

```
In [ ]: query = '''SELECT nodes_tags.value, COUNT(*) as num
                    FROM nodes_tags JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE
                                    ON nodes_tags.id = i.id
                    WHERE nodes_tags.key = 'cuisine'
                    GROUP BY nodes_tags.value
                    ORDER BY num DESC
                    LIMIT 25;'''
```

```
Cuisine         Count
-------         -----
american        26
pizza           25
italian         16
sandwich        15
chinese         13
burger          10
american_new    7
~~~(Some data omitted)~~~
korean          2   (!!!)
new_american    2
new_orleans     2   (!!!)
world           2
```

This is great! Although I wish there were more, there are 2 Korean restaurants in the city and surrounding area! I'm curious to find out the names of these restaurants so I can look into them further. Having grown up eating cajun food at least a few times a week, I'm also interested to know more about the New Orleans restaurants available in the city.

In [ ]:
```
# Name of all Korean & New Orleans restaurants
query = '''SELECT restaurant.value, nodes_tags.value
           FROM nodes_tags, (SELECT *
                             FROM nodes_tags
                             WHERE key="cuisine" AND
                                 (value = "korean" OR (value = "new_orleans
                             as restaurant
           WHERE nodes_tags.id = restaurant.id AND nodes_tags.key="name"'''
```

```
Cuisine     Restaurant Name
-------     ---------------
korean      Seoul Korean Restaurant
korean      Stone Bowl Grill
new_orleans Evolution Gastro Pong
new_orleans The Brass Alley
```

Both of the Korean restaurants offer Korean BBQ!! And have pretty good reviews on Yelp! The menus at each of the New Orleans restaurants also look delicious, with Evolution seeming to have a pretty good atmosphere.

## Conclusion

In general, the data did require some cleaning for consistency, but appeared to be accurate with respects to all nodes being within the Milwaukee metropolitan area. I would suggest further cleaning in the names of amenities to allow more accurate grouping with SQL queries (for example: "StarBucks" vs. "Starbucks", or "Pick N Save" vs. "Pick n Save").

This task, however, is quite labor intensive. While it is something that could be done to an extent programmatically using regular expressions, it will most likely also require a significant amount of manual user input to be performed completely. The most effective way of keeping this data clean could be in restricting user inputs to the OSM database to use standardized names. This could be useful for not only amenity names, but also other tags and city names (Milwaukee appeared at least twice in the list of cities, so having a standardized city name would most likely prevent this).

It would also be useful to have some built in error check when inputting OSM data to ensure that all nodes are within the city area (e.g. within XX miles of the city center). This could be done by checking longitude & latitude on every node input by the user and ensuring they're within some threshold set for the city. While not found to be an issue in this dataset, it could allow the site to offer two separate maps for Milwaukee and other major cities - one for the city itself, and one for the entire metropolitan area

## References

Sample Project - https://gist.github.com/carlward/54ec1c91b62a5f911c42#file-sample_project-md (https://gist.github.com/carlward/54ec1c91b62a5f911c42#file-sample_project-md)