In [1]:

```
import os
import sqlite3
import pandas as pd
os.chdir(r"C:\Users\randy\OneDrive\Desktop\Udacity\Wrangle OpenStreetMap Data")
conn = sqlite3.connect('sea_map.db')
```

Map area within project is the location of Seattle, WA, United States

Although I currently reside within a different city, Seattle, WA is where I grew up and spent my early adult years. I chose this location because of the personal nostalgia that it holds.

> https://www.openstreetmap.org/relation/237385
>
> https://overpass-api.de/api/map?bbox=-122.7715,47.4448,-121.9132,47.7698

Wrangling the data and problems encountered

Within the given focus, data auditing exposed several problems to be addressed as follows:

- Inconsistent street type names.
- Road and highway speed limit tags do not always list the given unit of measure as 'mph'.
- Postal codes are predominantly listed without the 4 digit zip extension, however some extensions do exist.
- Tags with a colon between values need to be split to identify the tag "type" and tag "key"

> ### Inconsistent street type names
>
> When considering the street type names, the data was audited and compared to a list of standard expected street types as to identify values which do not correspond.
> Upon identification, a dictionary of values was created to map and change these problem values to the expected dominant value for proper data unification.

> ### Speed limit tags within unit of measure
>
> Upon audit, several speed limit tags were found to be missing the given unit of measure. Given the location within the United States, the 'mph' suffix was added to all data points where missing

> ### Inconsistent postal code values
>
> While auditing the postal codes, uniform data was the primary concern. Postal codes were predominantly recoreded without the 4 digit extension. The inclusion of this extension would be ideal, however for the pupose of this project, the dominant abbreviated postal codes will suffice. As such, the postal extensions were removed from within the data points that included them.

> ### Isolation of tag "type" and tag "key"
>
> The tags with colons were identified through the use of a regular expression. Once located, the string was split at the first occurrance of a colon. With this, the tag type and key fields were identified.

Exploring the Data Sources

> **Files and their given sizes**

In [2]:

```
print ('sea_map.db     ',round(float(os.stat('sea_map.db').st_size / 2**20),2), 'MB')
print ('nodes.csv      ',round(float(os.stat('nodes.csv').st_size / 2**20),2), 'MB')
print ('nodes_tags.csv ',round(float(os.stat('nodes_tags.csv').st_size / 2**20),2), 'MB')
print ('ways.csv       ',round(float(os.stat('ways.csv').st_size / 2**20),2), 'MB')
print ('ways_tags.csv  ',round(float(os.stat('ways_tags.csv').st_size / 2**20),2), 'MB')
print ('ways_nodes.csv ',round(float(os.stat('ways_nodes.csv').st_size / 2**20),2), 'MB')
```

```
sea_map.db      244.95 MB
nodes.csv       40.67 MB
nodes_tags.csv  2.97 MB
ways.csv        3.82 MB
ways_tags.csv   8.91 MB
ways_nodes.csv  13.33 MB
```

> **The total number of unique users**

In [18]:

```
totalUsers = pd.read_sql('''SELECT COUNT(*) AS Total_Unique_Users
FROM (SELECT uid FROM ways UNION SELECT uid FROM nodes)''',conn).transpose()
print(totalUsers.to_string(header=False))
```

```
Total_Unique_Users  2697
```

> **The total number of tags within the given dataset**

In [4]:

```
totalCounts = pd.read_sql ("""
SELECT  (SELECT COUNT(*) FROM nodes) AS nodes,
        (SELECT COUNT(*)FROM nodes_tags) AS nodes_tags,
        (SELECT COUNT(*)FROM ways) AS ways,
        (SELECT COUNT(*)FROM ways_tags) AS ways_tags,
        (SELECT COUNT(*)FROM ways_nodes) AS ways_nodes
""",conn).transpose()
print(totalCounts.to_string(header=False))
```

```
nodes         968850
nodes_tags    159382
ways          126762
ways_tags     507402
ways_nodes   1115592
```

Exploring the Map Locations

Different location types are sampled and limited to their top results. This provides a greater overview of the data as it exists and any possible biases that may be present.

> **Establishments with the most individual locations**

In [5]:

```
establishments = pd.read_sql('''
SELECT value, Total
FROM (
SELECT value, COUNT(DISTINCT (ID)) AS Total
FROM nodes_tags WHERE key = 'name' GROUP BY value ORDER BY Total DESC LIMIT 10
)''',conn)
print(establishments.to_string(header=False, index=False))
```

```
         Starbucks 17
            Subway 11
Little Free Library  7
             Chase  7
   Bank of America  5
          Allstate  5
     The UPS Store  4
      Taco Del Mar  4
          T-Mobile  4
      Edward Jones  4
```

> **Top 5 Food and Drink individual values**

In [6]:

```
restaurants = pd.read_sql('''
SELECT value, Total
FROM (
SELECT value, COUNT(DISTINCT(id)) AS Total
FROM nodes_tags WHERE key = 'cuisine' GROUP BY value ORDER BY Total DESC LIMIT 5
)''',conn)
print(restaurants.to_string(header=False, index=False))
```

```
coffee_shop 39
      pizza 23
    mexican 19
   sandwich 18
    chinese 16
```

> **Top 5 HealthCare provider types**

In [7]:

```
health = pd.read_sql('''
SELECT value, Total
FROM (
SELECT value, COUNT(DISTINCT(id)) AS Total
FROM nodes_tags WHERE key = 'healthcare' GROUP BY value ORDER BY Total DESC LIMIT 5
)''',conn)
print(health.to_string(header=False, index=False))
```

```
    dentist 21
     doctor 14
     clinic 13
alternative 12
   pharmacy  9
```

> **Top 5 tourist attraction locations**

In [8]:

```
tours = pd.read_sql('''
SELECT value, Total
FROM (
SELECT value, COUNT(DISTINCT(id)) AS Total
FROM nodes_tags WHERE key = 'tourism' GROUP BY value ORDER BY Total DESC LIMIT 5
)''',conn)
print(tours.to_string(header=False, index=False))
```

```
    artwork 81
information 54
picnic_site 21
  viewpoint 12
 attraction  9
```

> **Top 5 public amenities**

In [9]:

```
amenities = pd.read_sql('''
```

```
SELECT value, Total
FROM (
SELECT value, COUNT(DISTINCT(id)) AS Total
FROM nodes_tags
WHERE key = 'amenity'
AND  NOT (value LIKE '%restaurant%' or value LIKE '%fast_food%' or value LIKE '%cafe%')
GROUP BY value
ORDER BY Total DESC
LIMIT 5
)''',conn)
print(amenities.to_string(header=False, index=False))
```

```
 bicycle_parking 384
           bench 344
    waste_basket 154
parking_entrance  62
       recycling  41
```

> **Breakdown of total establishments and amenities by sector**

In [10]:

```
totalEstablishmentCounts = pd.read_sql ("""
SELECT
(SELECT COUNT(DISTINCT(id)) FROM nodes_tags WHERE key = 'cuisine') AS Food_Drink,
(SELECT COUNT(DISTINCT(id)) FROM nodes_tags WHERE key = 'healthcare') AS HealthCare,
(SELECT COUNT(DISTINCT(id)) FROM nodes_tags WHERE key = 'tourism') AS Tourist,
(SELECT COUNT(DISTINCT(id)) FROM nodes_tags WHERE key = 'amenity'
AND  NOT
(value LIKE '%restaurant%' or value LIKE '%fast_food%' or value LIKE '%cafe%')) AS Public_Amenities
(SELECT COUNT(DISTINCT(id)) FROM nodes_tags WHERE key = 'name') AS Total_Bussiness_Establishments

""",conn).transpose()
print(totalEstablishmentCounts.to_string(header=False))
```

```
Food_Drink                       285
HealthCare                        77
Tourist                          191
Public_Amenities                1512
Total_Bussiness_Establishments  2433
```

Individual leaders from each sector

Individual leader from key tags is isolated and calculated as a percent of the whole, with the whole being the sum of its individualized key grouping.

```
Food_Drink:       coffee_shops:    14%
HealthCare:       doctor:          18%
Tourist:          art:             42%
Public_Amenities: bicycle_parking: 25%
```

```
The most interesting of these statistics appear to be that of bicycle_parking.  One
hypothesis is that Seattle is a very bike friendly city.  In line with this
hypothesis would be the thought that a lot of the user input to within the map
system may be derived from users on bicycles.  Given this information, it is also
possible that a large amount of the data from within the bike friendly commercial
corridors has been recieved.  This would also explain the heavy weighting within
the art sector, as this information may be more easily obtained by foot/bicycle.
In moving forward, it may be prudent to explore the expansion of data populated by
users traveling by vehicle.
```

Room for improvement

```
Although a great start, the data still looks to be somewhat incomplete.  When
calculating the total entries by street name it appears as if the data is weighted
towards the commercial corridors.  Also, without surprise, coffee shops are the
leading cuisine/dining option within Seattle.  What is surprising though is that
Starbucks is shown having 43% of all total coffee shop locations from within the
map.  A heavy weighting towards Starbucks is a given, however in knowing Seattle,
it could be easily assumed that several drive through and alternative coffee shops
are not included within this data set.  This small clue provides more proof of an
evolving but incomplete data set.
```

> **Street names with the most entries**

In [11]:

```
street = pd.read_sql('''
SELECT value, Total
FROM (
SELECT value, COUNT(DISTINCT(id)) AS Total
FROM nodes_tags WHERE key = 'street' GROUP BY value ORDER BY Total DESC LIMIT 10
)''',conn)
print(street.to_string(header=False, index=False))
```

```
Lake Washington Boulevard Northeast 87
             Rainier Avenue South 86
            Greenwood Avenue North 72
       California Avenue Southwest 67
                1st Avenue South 57
              Aurora Avenue North 54
          Roosevelt Way Northeast 52
                  Kirkland Avenue 51
             Northeast 68th Street 47
            Northeast 128th Street 44
```

> **Exploring the users**
>
> In an effort to identify where cooperation may be leveraged, users who identify as professional drivers are extracted from within the database

In [12]:

```
drivers = pd.read_sql('''SELECT user
FROM (SELECT user FROM ways UNION SELECT user FROM nodes )
WHERE user LIKE '%lyft%' or user LIKE '%driver%'  ''',conn)
print(drivers.to_string(header=False,index=False))
```

```
   A_Prokopova_lyft
    SeyfuLakew_lyft
      VeloBusDriver
       YTaulai_lyft
YuliyaShustava_lyft
       akakhno_lyft
    akuksouski_lyft
   amakarevich_lyft
     cpligovka_lyft
           cvo_lyft
         dlapo_lyft
        ggando_lyft
       imcnabb_lyft
    justin0206_lyft
           kli_lyft
      kmarkish_lyft
        metrodriver
```

```
  pmarkina_lyft
 rdanouski_lyft
skudrashou_lyft
technician_lyft
vrynkevich_lyft
      xliu_lyft
    zmoore_lyft
```

> **Ideas for improvement:**
> In keeping with the theme of the OpenStreetMap concept, open collaboration and integration within a select group of users provides the greatest opportunity to enhance the available data. When querying the users, it can be noted that some users are self identifying themselves as professional drivers within organizations such as Lyft. Although this group represents a small subset of the 2697 total users, it is very possible that others exist who are simpily not identifying as such. Given this information, and assuming a level of cooperation within these organizations, data can be leveraged from within individual stops and integrated to within the OpenStreetMap project. This data may be extracted through several methods, however the most seemless approach would be through mobile application access and permissions to within the origination/destination descriptions of each paying customer. The combination of this and GPS data would provide seemless automation to within the map environment.
>
> In accomplishing this, a very large hurdle to overcome would be that of aquiring the data feed from within these organizations. Although complex, I believe the benefits of this venture could be communicated and proposed in such a way where all parties involved would understand the value of a complete and dynamic mapping system. This mapping system has the potential to be much more dynamic than existing systems. Although still open source, it could be leveraged as a very valuable tool within any logistical organization.
>
> From a technical viewpoint, integrating the data from these sources would also be a challenge. Amongst other things, algorithms would need to be developed to interpret the data and compare it to any possible existing values for a given location. This would ensure that only unique values are injected within the database.

Conclusion

> As with many open source projects, the OpenStreetMap evolves on a daily basis. Although the data may be incomplete, its concept is solid. In teaming up with logistal organizations, the engagement and automation of user input can be greatly increased. In doing so, the map system has the potential to become a great resource with dynamic evolution on a second by second basis.