OPEN STREET MAP DATA WRANGLING WITH SQL

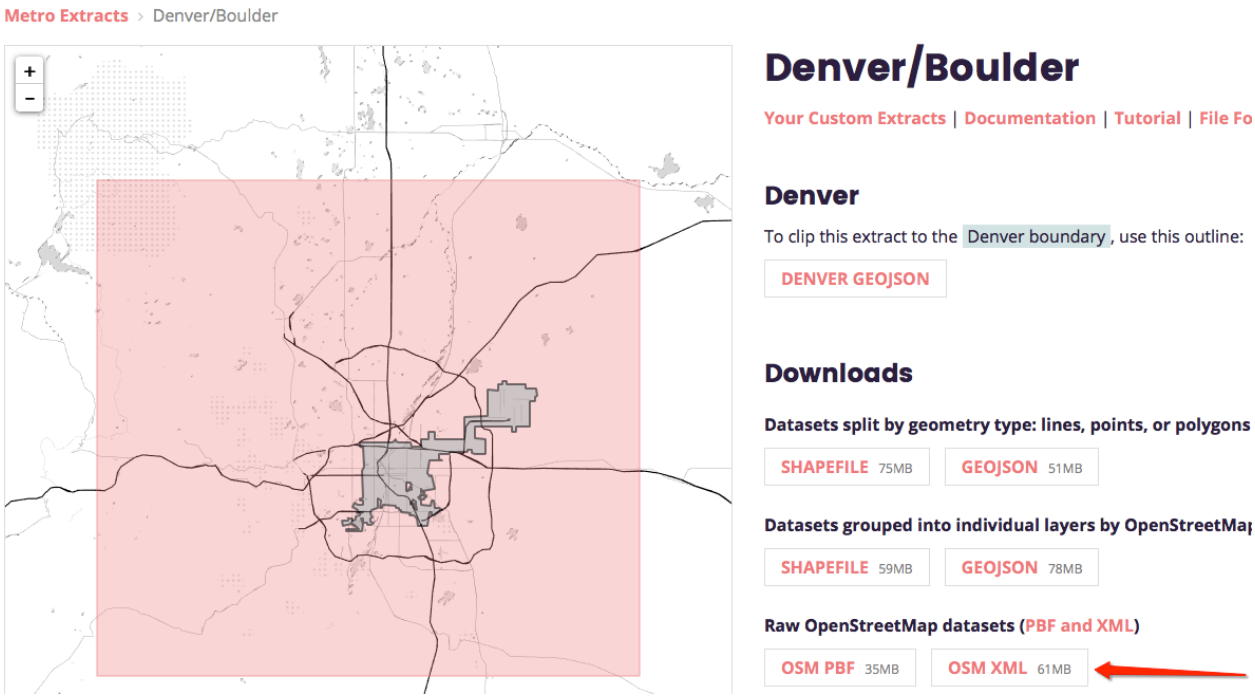
Awad Bin-Jawed

Based on the demonstration of Blueprint for cleaning data on Chicago, I will utilize this cycle for the city of **Denver, CO, USA**. This project will be an iterative process, so hopefully I will get a desired result as accurate as possible after an extensive cycle of repeating operations. This is why we say this whole cycle is convergent.

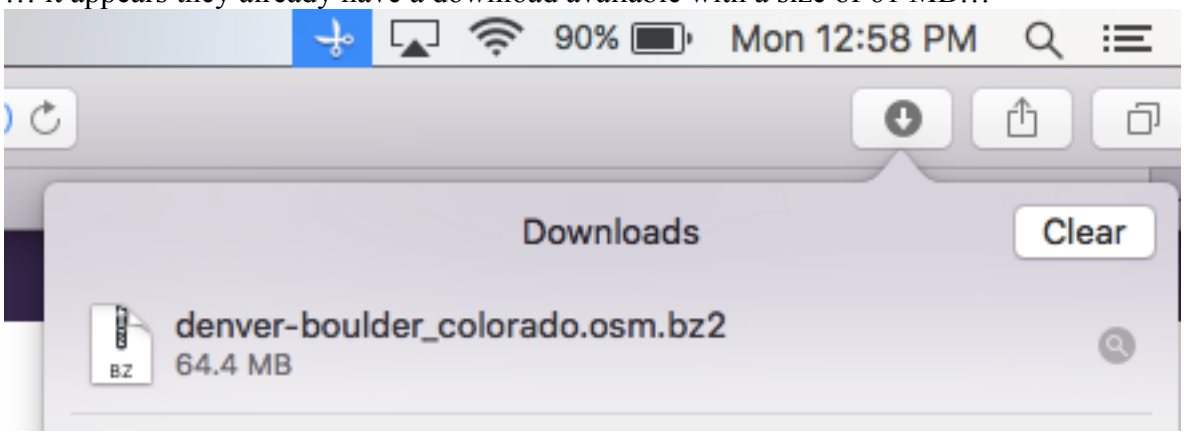
There was already an OSM XML file of Denver.

I will need to audit the street types and standardize the data; this way, I will know exactly what to extract.

Mapzen URL https://mapzen.com/data/metro-extracts/metro/denver-boulder_colorado/85928879/Denver/



… it appears they already have a download available with a size of 61 MB…



… turns out it is actually about 64.4 MB

For simplicity purposes, I will change the name of the OSM XML file to denver.osm

DATA AUDITING

For scripting and environments, I opted for the Atom Text Editor.

As for executing the **denver.osm** and **sample.osm** files, I used the Anaconda IPython due to their individual sizes each exceeding 25 MB.

From Lesson 5 (Data Quality) of the Udacity dashboard, video 8, I utilized "blueprint," which was the provided code as a template for auditing the data.

I saved the necessary codes in .py format as **blueprint.py** with some minor adjustments:

blueprint.py

```
import xml.etree.cElementTree as ET
from collections import defaultdict
import re

osm_file = open("denver.osm", "r")

street_type_re = re.compile(r'\S+\.?$', re.IGNORECASE)
street_types = defaultdict(int)

def audit_street_type(street_types, street_name):
    m = street_type_re.search(street_name)
    if m:
        street_type = m.group()

        street_types[street_type] += 1

def print_sorted_dict(d):
    keys = d.keys()
    keys = sorted(keys, key=lambda s: s.lower())
    for k in keys:
        v = d[k]
        print "%s: %d" % (k, v)

def is_street_name(elem):
    return (elem.tag == "tag") and (elem.attrib['k'] == "addr:street")

def audit():
    for event, elem in ET.iterparse(osm_file):
        if is_street_name(elem):
            audit_street_type(street_types, elem.attrib['v'])
    print_sorted_dict(street_types)

if __name__ == '__main__':
    audit()
```

This **blueprint.py** code was saved in the .py format in Atom.
The codes were executed in Anaconda IPython.

After executing the following code…

```
import xml.etree.cElementTree as ET
from collections import defaultdict
import re

osm_file = open("denver.osm", "r")

street_type_re = re.compile(r'\S+\.?$', re.IGNORECASE)
street_types = defaultdict(int)

def audit_street_type(street_types, street_name):
    m = street_type_re.search(street_name)
    if m:
        street_type = m.group()

        street_types[street_type] += 1

def print_sorted_dict(d):
    keys = d.keys()
    keys = sorted(keys, key=lambda s: s.lower())
    for k in keys:
        v = d[k]
        print "%s: %d" % (k, v)

def is_street_name(elem):
    return (elem.tag == "tag") and (elem.attrib['k'] == "addr:street")

def audit():
    for event, elem in ET.iterparse(osm_file):
        if is_street_name(elem):
            audit_street_type(street_types, elem.attrib['v'])
    print_sorted_dict(street_types)

if __name__ == '__main__':
    audit()
```

…I got the counts of variables:

```
#1: 1
#100: 2
#103: 1
#104: 1
#107: 1
#110: 1
#115: 1
#130: 1
#140: 1
#169: 1
#200: 3
#201: 1
#220: 1
#220H: 1
#300: 4
#320: 1
#380: 1
#4: 1
#400: 1
#500: 1
#500a: 1
#600: 1
#6300: 1
#8: 1
#800: 1
#850: 1
#9k: 1
#A: 2
#B: 1
#D: 1
#E: 1
(CO): 1
-76: 1
100: 2
106: 1
110: 1
119: 7
120: 1
126: 7
13: 1
154: 1
1606: 1
186: 1
19: 1
2: 68
200: 1
200c: 1
2132: 1
285: 6
287: 76
300: 2
314: 2
4: 1
40: 4
400: 2
41: 1
421: 1
45: 1
52: 8
59: 4
5B: 2
6001: 1
65: 2
6523: 1
68th: 1
7: 4
700: 1
72: 6
73: 9
7331: 1
74: 42
77: 1
8: 1
80208: 1
80305: 1
83: 18
85: 1
86: 5
8765: 1
88th: 1
900: 1
93: 2
A: 1
Ace: 2
Appia: 3
Arapahoe: 2
Av: 2
Ave: 98
Ave.: 6
ave.: 1
Avenue: 10216
Avenue): 4
B: 2
B-7: 1
Baselin: 2
Baseline: 66
Bikeway: 1
Blf: 1
Blvd: 103
Blvd.: 1
Boulder: 1
Boulevard: 1900
Broadway: 378
Bypass: 2
Camground: 1
Campground: 18
Caria: 4
Centennial: 3
Center: 26
Cherryvale: 1
Cir: 27
circle: 1
Circle: 1371
Co-119: 1
CO-119: 1
Colfax: 38
Colorado: 1
Court: 1617
Crescent: 9
ct: 1
Ct: 22
dr: 2
Dr: 205
drive: 1
Drive: 2555
E: 1
E1: 1
East: 21
Elm: 18
Etna: 2
Grant: 1
H: 1
Highway: 10
Hwy: 1
K: 1
Lamar: 1
Lane: 398
lane: 1
Lincoln: 1
Ln: 2
Locust: 1
Loop: 6
Main: 10
Mainstreet: 57
Mall: 18
Mine: 1
MUP: 1
Newland: 1
North: 16
Osage: 3
Parkway: 329
Pennsylvania: 1
Perdido: 1
Pkwy: 14
Pky: 1
Pl: 62
Place: 1753
Plaza: 5
Point: 65
R-250: 1
Ramp: 1
Raod: 1
Rd: 81
rd: 1
Rd.: 16
road: 1
Road: 1456
Row: 2
Run: 3
South: 26
Speer: 1
Sreet: 3
St: 449
st: 5
St.: 10
STreet: 1
Street: 22442
Strret: 1
Tennyson: 1
Terrace: 1
Thornton,: 1
Trail: 193
trail: 1
US-85: 1
Ute: 1
Varra: 2
Wadsworth: 1
Walnut: 1
Way: 1017
WB: 1
West: 67
Wewatta: 1
Woodfern: 1
```

Now we need to decide what type of cleaning is necessary:

Abbreviations:

Av needs to be converted to Avenue
Ave needs to be converted to Avenue
Ave. needs to be converted to Avenue
Avenue) needs to be converted to have the right parenthesis “)” removed
Baselin needs to be converted to Baseline
Blf needs to be converted to Boulevard
Blvd needs to be converted to Boulevard
Blvd. needs to be converted to Boulevard
Cir. needs to be converted to Circle
Ct needs to be converted to Court
Dr needs to be converted to Drive
Hwy needs to be converted to Highway
Ln needs to be converted to Parkway
Pkwy needs to be converted to Parkway
Pky needs to be converted to Parkway
Rd needs to be converted to Road
Rd. needs to be converted to be Road
St needs to be converted to Street
St. needs to be converted to Street
Strret needs to be converted Street

The imported print statement from pprint will print the dictionary of keys (street types) and their associated values (street names):

```
# Print the results
pprint.pprint(dict(st_types))
```

We will have to clean the attributes and store them in memory for future use:

STREET NAMES OF NODES AND WAYS

```
# -*- coding: utf-8 -*-


import xml.etree.cElementTree as ET
from collections import defaultdict
import re
import pprint

OSM_FILE = "/Users/awadbin-jawed/Desktop/Udacity/Files/denver.osm"
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)

expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place",
            "Square", "Lane", "Road", "Trail", "Parkway", "Commons", "Alley",
            "Bridge", "Highway", "Circle", "Terrace", "Way"]


def audit_street_type(street_types, street_name):
m = street_type_re.search(street_name)
    if m:
        street_type = m.group()
        if street_type not in expected:
            street_types[street_type].add(street_name)

def is_street_name(elem):
return (elem.attrib['k'] == "addr:street")

def audit(osmfile):
    osm_file = open(osmfile, "r")
    street_types = defaultdict(set)
    for event, elem in ET.iterparse(osm_file, events=("start",)):
        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
                if is_street_name(tag):
                    audit_street_type(street_types, tag.attrib['v'])
    osm_file.close()
    return street_types

st_types = audit(OSM_FILE)

pprint.pprint(dict(st_types))
```

POSTCODE INCONSISTENCY

```
# -*- coding: utf-8 -*-


import xml.etree.cElementTree as ET
import re
import pprint

OSM_FILE = "/Users/awadbin-jawed/Desktop/Udacity/Files/denver.osm"
postcode_re = re.compile(r'^\d{5}( [\-]?\d{4})?$')


def audit_postcode(postcodes,value):
    m = postcode_re.search(value)
    if not m:
        postcodes.add(value)

def is_postcode(elem):
    return (elem.attrib['k'] == "addr:postcode" or elem.attrib['k'] == "postal_code")

def audit(osmfile):
    osm_file = open(osmfile, "r")
    postcodes = set()
    for event, elem in ET.iterparse(osm_file, events=("start",)):
        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
                if is_postcode(tag):
                    audit_postcode(postcodes,tag.attrib['v'])
    osm_file.close()
    return postcodes

postcodes = audit(OSM_FILE)
print "unexpected postcode values:"
pprint.pprint(postcodes)
```

```
            'W Quincy Ave',
            'W. 10th Ave',
            'W. 25th Ave',
            'Washington Ave',
            'West 136th Ave',
            'Yarmouth Ave',
            'w 67th Ave',
            'w. 10th Ave']),
'Ave.': set(['11200 W. 64th Ave.', 'E. College Ave.', 'W. Alameda Ave.']),
'Avenue)': set(['East Bromley Lane (152nd Avenue)']),
'B': set(['4th Street Suite B',
          'Colorado Blvd #B',
          'South Pratt Parkway Unit B']),
'B-7': set(['East 120th Avenue Unit B-7']),
'Baselin': set(['Baselin']),
'Baseline': set(['Baseline']),
'Bikeway': set(['US 36 Bikeway']),
'Blf': set(['W 85th Blf']),
'Blvd': set(['745 Colorado Blvd',
             'Airport Blvd',
             'Brighton Blvd',
             'Colorado Blvd',
             'East Academy Blvd',
             'Eldorado Blvd',
             'Federal Blvd',
             'Foothills Canyon Blvd',
             'Green Valley Ranch Blvd',
             'Ken Prat Blvd',
             'Ken Pratt Blvd',
             'Martin Luther King Blvd',
             'Martin Luther King Jr Blvd',
             'McCaslin Blvd',
             'N Federal Blvd',
             'S Lowell Blvd',
             'S University  Blvd',
             'S University Blvd',
             'S Wadsworth Blvd',
             'Sheridan Blvd',
             'South Colorado Blvd',
             'South University Blvd',
             'Wadsworth Blvd',
             'Wadworth Blvd']),
'Blvd.': set(['Sheridan Blvd.']),

'314': set(['County Road 314']),
'320': set(['E Arapahoe Rd Tower 1 #320']),
'380': set(['South Colorado Boulevard #380']),
'4': set(['10143 w Chatfield, Ste #4', 'County Road 4']),
'40': set(['US Highway 40', 'US Hwy 40']),
'400': set(['E 45th Ave, Ste 400',
            'Pace Street #400',
            'Wadsworth Boulevard Ste 400']),
'41': set(['County Road 41']),
'421': set(['Cook St Suite 421']),
'45': set(['County Road 45']),
'500': set(['Weaver Park Rd #500']),
'500a': set(['W 92nd Ave #500a']),
'52': set(['I-76 & CO Hwy 52', 'State Highway 52']),
'59': set(['Weld County Road 59']),
'5B': set(['Sheridan Boulevard 5B', 'Sheridan Boulevard Unit 5B']),
'600': set(['South Main Street #600']),
'6001': set(['6001']),
'6300': set(['East 19th Avenue  #6300']),
'65': set(['County Road 65']),
'6523': set(['6523']),
'68th': set(['68th']),
'7': set(['County Road 7',
          'SH 7',
          'State Highway 7',
          'West State Highway 7']),
'700': set(['Broadway, Ste 700']),
'72': set(['CO 72', 'County Road 72', 'Park 72']),
'73': set(['CO Highway 73', 'County Highway 73', 'County Road 73', 'Hwy 73']),
'7331': set(['P.O. Box 7331']),
'74': set(['Colorado 74', 'County Road 74', 'Highway 74', 'Hwy 74']),
'76': set(['Frontage Road I -76']),
'77': set(['County Road 77']),
'8': set(['49th St #8', 'CO HWY 8']),
'800': set(['W 92nd Ave #800']),
'80208': set(['2148 S. High Street, Denver, Colorado, 80208']),
'80305': set(['1190 South Lashley Lane Boulder, CO 80305']),
'83': set(['Colorado Highway 83',
           'North State Highway 83',
           'State Highway 83']),
'85': set(['US 85']),
'850': set(['Santa Fe Drive, Unit #850']),
'86': set(['State Highway 86']),
```

```
{'1': set(['Bradburn Boulevard #1']),
 '100': set(['15th Street Suite 100',
             'Sheridan Boulevard #100',
             'South Ridgeline Blvd, Suite 100',
             'Wadsworth Parkway #100']),
 '103': set(['15th Street #103']),
 '104': set(['Cherry Creek S Dr #104']),
 '106': set(['Landmark Way, Suite 106']),
 '107': set(['Dorchester St #107']),
 '110': set(['E. Mineral Avenue, Suite 110', 'W Cross Dr #110']),
 '115': set(['East Iliff Avenue #115']),
 '119': set(['Highway 119', 'State Highway 119']),
 '120': set(['West 120th Avenue, Ste 120']),
 '126': set(['County Rd 126']),
 '13': set(['County road 13']),
 '130': set(['E Caley Ave #130']),
 '140': set(['East Arapahoe Road, #140']),
 '154': set(['West 136th Avenue Unit 154']),
 '1606': set(['1606']),
 '169': set(['S Broadway #169']),
 '186': set(['County Road 186']),
 '19': set(['South Rock Creek Parkway, Building 19']),
 '2': set(['Colorado SH 2',
           'Colorado SR 2',
           'Delaware St Unit 2',
           'SR 2',
           'State Highway 2']),
 '200': set(['Ken Pratt Blvd #200',
             'Pace Street #200',
             'S Bellaire St #200',
             'Wadsworth Boulevard, Ste. 200']),
 '200c': set(['West 25th Avenue Suit 200c']),
 '201': set(['Ken Pratt Boulevard #201']),
 '2132': set(['W Belleview Ave Ste 2132']),
 '220': set(['South Colorado Boulevard, #220']),
 '220H': set(['30th St #220H']),
 '285': set(['Highway 285', 'US Highway 285']),
 '287': set(['Highway 287', 'South Highway 287', 'US Highway 287']),
 '300': set(['Commerce Center Cir #300',
             'Crown Crest Blvd., Suite 300',
             'Delaware St #300',
             'Wadsworth Boulevard, Ste 300',
             'Wadsworth Parkway #300']),
 '8765': set(['8765']),
 '88th': set(['E 88th']),
 '900': set(['West 120th Avenue, Suite 900']),
 '93': set(['Hwy 72 & Hwy 93', 'State Highway 93']),
 '9k': set(['S Alton Way #9k']),
 'A': set(['Sherman Dr #A', 'South Road A', 'W Ellsworth Ave #A']),
 'Ace': set(['4775 West 121st Ace', '5005 West 120th Ace']),
 'Appia': set(['Via Appia']),
 'Arapahoe': set(['Arapahoe']),
 'Av': set(['W 56th Av', 'W Belleview Av']),
 'Ave': set(['Arapahoe Ave',
             'Cottonwood Ave',
             'Dawn Ave',
             'Dorado Ave',
             'E 120th Ave',
             'E 1st Ave',
             'E 6th Ave',
             'E 72nd Ave',
             'E 84th Ave',
             'E 88th Ave',
             'E Arizona Ave',
             'E Caley Ave',
             'E Evans Ave',
             'E Fair Ave',
             'E Louisiana Ave',
             'E Maplewood Ave',
             'East Mississippi Ave',
             'Evans Ave',
             'W 10th Ave',
             'W 120th Ave',
             'W 44th Ave',
             'W 64th Ave',
             'W 84th Ave',
             'W 92nd Ave',
             'W 96th Ave',
             'W 98th Ave',
             'W Floyd Ave',
             'W Hampden Ave',
             'W Ken Caryl Ave',
             'W Plymouth Ave',
             'W Quincy Ave',
             'W. 25th Ave',
             'Washington Ave',
             'West 136th Ave',
```

```
'Ave.': set(['11200 W. 64th Ave.', 'E. College Ave.']),
'Avenue)': set(['East Bromley Lane (152nd Avenue)']),
'B': set(['4th Street Suite B',
          'Colorado Blvd #B',
          'South Pratt Parkway Unit B']),
'B-7': set(['East 120th Avenue Unit B-7']),
'Baselin': set(['Baselin']),
'Baseline': set(['Baseline']),
'Bikeway': set(['US 36 Bikeway']),
'Blf': set(['W 85th Blf']),
'Blvd': set(['745 Colorado Blvd',
             'Airport Blvd',
             'Brighton Blvd',
             'Colorado Blvd',
             'East Academy Blvd',
             'Eldorado Blvd',
             'Federal Blvd',
             'Foothills Canyon Blvd',
             'Green Valley Ranch Blvd',
             'Ken Prat Blvd',
             'Ken Pratt Blvd',
             'Martin Luther King Blvd',
             'Martin Luther King Jr Blvd',
             'McCaslin Blvd',
             'N Federal Blvd',
             'N Sheridan Blvd',
             'S Lowell Blvd',
             'S University  Blvd',
             'S University Blvd',
             'S Wadsworth Blvd',
             'Sheridan Blvd',
             'South Colorado Blvd',
             'South University Blvd',
             'Wadsworth Blvd',
             'Wadworth Blvd']),
'Boulder': set(['Boulder']),
'Broadway': set(['Broadway',
                 'N Broadway',
                 'North Broadway',
                 'S Broadway',
                 'S. Broadway',
                 'South Broadway']),
'Bypass': set(['Wadsworth Bypass']),
'Pky': set(['DTC Pky']),
'Pl': set(['E Lake Pl',
           'E Maplewood Pl',
           'E Orchard Pl',
           'Pinon Pl',
           'W 16th Pl',
           'W 88th Pl',
           'W Lehigh Pl']),
'Plaza': set(['SouthPark Plaza', 'Wewatta Plaza', 'Wynkoop Plaza']),
'Point': set(['Buchanan Point',
              'High Lonesome Point',
              'Lodgewood Point',
              'Lone Eagle Point',
              'Pawnee Point',
              'Red Feather Point',
              'Sawtooth Point',
              'Sleeping Owl Point']),
'R-250': set(['17th Street R-250']),
'Ramp': set(['Park Avenue West Ramp']),
'Raod': set(['South Parker Raod']),
'Rd': set(['Coalton Rd',
           'Depot Hill Rd',
           'E Arapahoe Rd',
           'E Arapaphoe Rd',
           'E Araphahoe Rd',
           'E Araphoe Rd',
           'E County Line Rd',
           'E. Baseline Rd',
           'E. Rim Rd',
           'East Arapahoe Rd',
           'Elk Creek Rd',
           'Lagae Rd',
           'Lookout Mountain Rd',
           'Mt. Vernon Rd',
           'North Watkins Rd',
           'Pine Valley Rd',
           'S Golden Rd',
           'S Mt Vernon Country Club Rd',
           'S Parker Rd',
           'S. Golden Rd',
           'S. River Rd',
           'South Parker Rd',
           'West Ridge Rd']),
'CO)': set(['US 6 (CO)']),
'CO-119': set(['CO-119']),
'Camground': set(['West Magnolia Camground']),
'Campground': set(['West Magnolia Campground']),
'Caria': set(['Caria']),
'Centennial': set(['Centennial']),
'Center': set(['Garden Center']),
'Cherryvale': set(['Cherryvale']),
'Cir': set(['E Flatiron Cir',
            'Highlands Cir',
            'Rock Creek Cir',
            'S Lake Cir']),
'Co-119': set(['Co-119']),
'Colfax': set(['East Colfax']),
'Colorado': set(['12354 West Alameda Parkway Lakewood Colorado']),
'Crescent': set(['Front View Crescent', 'Interlocken Crescent']),
'Ct': set(['Autumn Ct',
           'E Arapahoe Ct',
           'North 57th Ct',
           'Pear Lake Ct',
           'Rodeo Ct',
           'S Niagra Ct',
           'South 57th Ct']),
'D': set(['E 3rd Ave #D']),
'Dr': set(['Apache Dr',
           'Bonanza Dr',
           'Broadlands Dr',
           'Cabela Dr',
           'Community Center Dr',
           'Community Circle Dr',
           'Conifer Town Denter Dr',
           'Cowley Dr',
           'Delwood Dr',
           'E 35th Dr',
           'Eisenhower Dr',
           'Indian Peaks Dr',
           'Lakeview Dr',
           'Leetsdale Dr',
           'Longs Peak Dr',
           'Melody Dr',
           'Reliance Dr',
           'S Santa Fe Dr',
           'Zotos Dr']),
'E': set(['1100 Ken Pratt Blvd Unit E', 'Main Street #E']),
'E1': set(['University Blvd, Suite E1']),
'East': set(['Inverness Court East',
             'Inverness Drive East',
             'South Chapparal Circle East',
             'Sterling Circle East',
             'Village Center Drive East']),
'Elm': set(['East Elm']),
'Etna': set(['Etna']),
'Grant': set(['Grant']),
'H': set(['Wilderness Place, Unit H']),
'Hwy': set(['S Valley Hwy']),
'K': set(['Norwood Drive Unit K']),
'Lamar': set(['Lamar']),
'Lincoln': set(['Lincoln']),
'Ln': set(['Lee Ln', 'Tuscany Ln']),
'Locust': set(['Yellow Locust']),
'Loop': set(['Eagle River Loop', 'Quay Loop']),
'MUP': set(['128th Ave MUP']),
'Main': set(['Main']),
'Mainstreet': set(['East Mainstreet']),
'Mall': set(['10th Street Mall',
             'Central Campus Mall',
             'Lawrence Street Mall',
             'Pearl Street Mall']),
'Mine': set(['Coal Mine']),
'Newland': set(['Newland']),
'North': set(['57th Court North',
              'Central Court North',
              'East Speer Boulevard North',
              'Nautilus Court North',
              'Parkside Drive North',
              'West 6th Avenue Frontage Road North']),
'Osage': set(['Osage']),
'Pennsylvania': set(['Pennsylvania']),
'Perdido': set(['Camino Perdido']),
'Pkwy': set(['Castle Pines Pkwy',
             'E Alameda Pkwy',
             'Evergreen Pkwy',
             'Orchard Pkwy',
             'S Aurora Pkwy',
             'S Kipling Pkwy',
             'S Monaco Pkwy',
'Rd.': set(['402 Marshall Rd.', 'E. Rim Rd.', 'S. Jordan Rd.']),
'Run': set(['Raven Run', 'Stage Run']),
'STreet': set(['Main STreet']),
'South': set(['57th Court South',
              'Buchtel Boulevard South',
              'East Cherry Creek Drive South',
              'Inverness Drive South',
              'Inverness Way South',
              'Nautilus Court South',
              'West 6th Avenue Frontage Road South']),
'Speer': set(['Speer']),
'Sreet': set(['South Fairplay Sreet']),
'St': set(['12th St',
           '14th St',
           '3495 S. Downing St',
           '9th St',
           'Broadway St',
           'Commons St',
           'Dahlia St',
           'Dayton St',
           'Dorchester St',
           'Downing St',
           'E Simpson St',
           'East Evans St',
           'First St',
           'Glencoe St',
           'Gold Hill St',
           'Grant St',
           'Grove St',
           'Jefferson St',
           'Main St',
           'N Richfield St',
           'N Salida St',
           'N Washington St',
           'Oak St',
           'S Akron St',
           'S Broadway St',
           'S Clayton St',
           'S Columbine St',
           'S Dallas St',
           'S Delaware St',
           'S Elizabeth St',
           'S Franklin St',
'St.': set(['11265 Decatur St.',
            'California St.',
            'E Simpson St.',
            'Little Raven St.',
            'Main St.',
            'Pearl St.',
            'Purcell St.',
            'Robert St.',
            'S. Bannock St.']),
'Strret': set(['South Holly Strret']),
'Tennyson': set(['Tennyson']),
'Thornton,': set(['4710 Washington St, Thornton,']),
'US-85': set(['US-85']),
'Ute': set(['West Ute']),
'Varra': set(['Via Varra']),
'WB': set(['E Orchard Rd Bike Lane WB']),
'Wadsworth': set(['Wadsworth']),
'Walnut': set(['Walnut']),
'West': set(['Castlegate Drive West',
             'Inverness Drive West',
             'Park Avenue West',
             'South Carr Avenue West',
             'Sterling Circle West']),
'Wewatta': set(['Wewatta']),
'Woodfern': set(['Woodfern']),
'ave.': set(['Pennsylvania ave.']),
'circle': set(['Oxford circle']),
'ct': set(['S Newcombe ct']),
'dr': set(['20 amaranth dr', 'W friend dr']),
'drive': set(['City View drive']),
'lane': set(['Lindenwood lane']),
'rd': set(['E. Arapahoe rd']),
'road': set(['south hover road']),
'st': set(['28th st', 'S gaylord st', 'South Carr st']),
'trail': set(['shoshone trail'])}
```

DATA CLEANING:

The following print statement:

```
for st_type, ways in st_types.iteritems():
    for name in ways:
        better_name = update_name(name, mapping)
        if name != better_name:
            print name, "=>", better_name
            name = better_name
```

This will print the street names after the update of the original, unclean street names:

```
Elk Creek Rd => Elk Creek Road
North Watkins Rd => North Watkins Road
S Mt Vernon Country Club Rd => S Mt Vernon Country Club Road
E. Rim Rd => E. Rim Road
Lagae Rd => Lagae Road
E Arapahoe Rd => E Arapahoe Road
East Arapahoe Rd => East Arapahoe Road
S. River Rd => S. River Road
Mt. Vernon Rd => Mt. Vernon Road
E County Line Rd => E County Line Road
E. Baseline Rd => E. Baseline Road
E Araphahoe Rd => E Araphahoe Road
Lookout Mountain Rd => Lookout Mountain Road
S Parker Rd => S Parker Road
E Arapaphoe Rd => E Arapaphoe Road
West Ridge Rd => West Ridge Road
S. Golden Rd => S. Golden Road
Depot Hill Rd => Depot Hill Road
Coalton Rd => Coalton Road
Pine Valley Rd => Pine Valley Road
S Golden Rd => S Golden Road
E Araphoe Rd => E Araphoe Road
South Holly Strret => South Holly Street
S Aurora Pkwy => S Aurora Parkway
Castle Pines Pkwy => Castle Pines Parkway
E Alameda Pkwy => E Alameda Parkway
W Alameda Pkwy => W Alameda Parkway
S Platte River Pkwy => S Platte River Parkway
Orchard Pkwy => Orchard Parkway
S Kipling Pkwy => S Kipling Parkway
Wadsworth Pkwy => Wadsworth Parkway
Evergreen Pkwy => Evergreen Parkway
S Monaco Pkwy => S Monaco Parkway
402 Marshall Rd. => 402 Marshall Road
S. Jordan Rd. => S. Jordan Road
E. Rim Rd. => E. Rim Road
East Bromley Lane (152nd Avenue) => East Bromley Lane (152nd Avenue
E. College Ave. => E. College Avenue
11200 W. 64th Ave. => 11200 W. 64th Avenue
W. Alameda Ave. => W. Alameda Avenue
Sheridan Blvd. => Sheridan Boulevard
DTC Pky => DTC Parkway
W 85th Blf => W 85th Boulevard
```

```
Baselin => Baseline
Purcell St. => Purcell Street
Pearl St. => Pearl Street
California St. => California Street
S. Bannock St. => S. Bannock Street
Main St. => Main Street
Little Raven St. => Little Raven Street
E Simpson St. => E Simpson Street
Robert St. => Robert Street
N Salida St => N Salida Street
W Mulberry St => W Mulberry Street
Dorchester St => Dorchester Street
S Quebec St => S Quebec Street
S Peoria St => S Peoria Street
Downing St => Downing Street
South Grant St => South Grant Street
S Franklin St => S Franklin Street
Wright St => Wright Street
Washington St => Washington Street
Yukon St => Yukon Street
S Broadway St => S Broadway Street
Main St => Main Street
S Sherman St => S Sherman Street
S Abilene St => S Abilene Street
S Niagra St => S Niagra Street
S Clayton St => S Clayton Street
Oak St => Oak Street
S Gaylord St => S Gaylord Street
E Simpson St => E Simpson Street
S Pontiac St => S Pontiac Street
Dayton St => Dayton Street
First St => First Street
Lanewood St => Lanewood Street
S Dallas St => S Dallas Street
Gold Hill St => Gold Hill Street
12th St => 12th Street
N Richfield St => N Richfield Street
Commons St => Commons Street
Glencoe St => Glencoe Street
S Akron St => S Akron Street
S Poplar St => S Poplar Street
East Evans St => East Evans Street
Wewatta St => Wewatta Street
```

```
3495 S. Downing St => 3495 S. Downing Street
S Delaware St => S Delaware Street
South Sunset St => South Sunset Street
Grant St => Grant Street
Broadway St => Broadway Street
N Washington St => N Washington Street
14th St => 14th Street
S York St => S York Street
S Pearl St => S Pearl Street
S Logan St => S Logan Street
S Elizabeth St => S Elizabeth Street
Grove St => Grove Street
Jefferson St => Jefferson Street
9th St => 9th Street
S Josephine St => S Josephine Street
S Columbine St => S Columbine Street
Wynkoop St => Wynkoop Street
S Newport St => S Newport Street
W Quincy Ave => W Quincy Avenue
Evans Ave => Evans Avenue
E Arizona Ave => E Arizona Avenue
w 67th Ave => w 67th Avenue
w. 10th Ave => w. 10th Avenue
W Hampden Ave => W Hampden Avenue
Arapahoe Ave => Arapahoe Avenue
W Floyd Ave => W Floyd Avenue
Lincoln Ave => Lincoln Avenue
E Fair Ave => E Fair Avenue
W 92nd Ave => W 92nd Avenue
E 88th Ave => E 88th Avenue
East Mississippi Ave => East Mississippi Avenue
W 84th Ave => W 84th Avenue
W 72nd Ave => W 72nd Avenue
E 1st Ave => E 1st Avenue
E Louisiana Ave => E Louisiana Avenue
W. 10th Ave => W. 10th Avenue
W 44th Ave => W 44th Avenue
Dorado Ave => Dorado Avenue
E 120th Ave => E 120th Avenue
Cottonwood Ave => Cottonwood Avenue
E 84th Ave => E 84th Avenue
Dawn Ave => Dawn Avenue
W 96th Ave => W 96th Avenue
```

```
W 10th Ave => W 10th Avenue
W Ken Caryl Ave => W Ken Caryl Avenue
East Hampden Ave => East Hampden Avenue
W Plymouth Ave => W Plymouth Avenue
Washington Ave => Washington Avenue
W 98th Ave => W 98th Avenue
W. 25th Ave => W. 25th Avenue
Yarmouth Ave => Yarmouth Avenue
E Caley Ave => E Caley Avenue
E Maplewood Ave => E Maplewood Avenue
W 64th Ave => W 64th Avenue
E 72nd Ave => E 72nd Avenue
West 136th Ave => West 136th Avenue
W 120th Ave => W 120th Avenue
E Evans Ave => E Evans Avenue
North 57th Ct => North 57th Court
Pear Lake Ct => Pear Lake Court
E Arapahoe Ct => E Arapahoe Court
Autumn Ct => Autumn Court
Rodeo Ct => Rodeo Court
S Niagra Ct => S Niagra Court
South 57th Ct => South 57th Court
Tuscany Ln => Tuscany Lane
Lee Ln => Lee Lane
S Valley Hwy => S Valley Highway
Indian Peaks Dr => Indian Peaks Drive
Community Circle Dr => Community Circle Drive
Bonanza Dr => Bonanza Drive
Conifer Town Denter Dr => Conifer Town Denter Drive
E 35th Dr => E 35th Drive
Community Center Dr => Community Center Drive
Reliance Dr => Reliance Drive
Zotos Dr => Zotos Drive
Broadlands Dr => Broadlands Drive
Cabela Dr => Cabela Drive
Eisenhower Dr => Eisenhower Drive
Lakeview Dr => Lakeview Drive
Leetsdale Dr => Leetsdale Drive
Cowley Dr => Cowley Drive
Longs Peak Dr => Longs Peak Drive
Melody Dr => Melody Drive
Delwood Dr => Delwood Drive
S Santa Fe Dr => S Santa Fe Drive
```

```
Apache Dr => Apache Drive
W 56th Av => W 56th Avenue
W Belleview Av => W Belleview Avenue
Wadworth Blvd => Wadworth Boulevard
S Lowell Blvd => S Lowell Boulevard
Green Valley Ranch Blvd => Green Valley Ranch Boulevard
S Wadsworth Blvd => S Wadsworth Boulevard
Eldorado Blvd => Eldorado Boulevard
Wadsworth Blvd => Wadsworth Boulevard
Federal Blvd => Federal Boulevard
S University Blvd => S University Boulevard
Martin Luther King Blvd => Martin Luther King Boulevard
Sheridan Blvd => Sheridan Boulevard
Brighton Blvd => Brighton Boulevard
Foothills Canyon Blvd => Foothills Canyon Boulevard
N Federal Blvd => N Federal Boulevard
McCaslin Blvd => McCaslin Boulevard
Martin Luther King Jr Blvd => Martin Luther King Jr Boulevard
Ken Prat Blvd => Ken Prat Boulevard
Airport Blvd => Airport Boulevard
S University  Blvd => S University  Boulevard
745 Colorado Blvd => 745 Colorado Boulevard
South Colorado Blvd => South Colorado Boulevard
Colorado Blvd => Colorado Boulevard
East Academy Blvd => East Academy Boulevard
Ken Pratt Blvd => Ken Pratt Boulevard
South University Blvd => South University Boulevard
```

DATA OVERVIEW

Step One: Create the .csv files

After auditing, the next step is to prepare the data for insertion to a SQL database. We parse the elements in the OSM XML file, which will transform them from .doc format to tabular format. The process involved using the iterparse through the elements in the XML and shaping the elements into several structures. The schema was to ensure the data is in the correct format. Now we write each data structure to the appropriate .csv files.

```
# -*- coding: utf-8 -*-


import xml.etree.cElementTree as ET

OSM_FILE = "/Users/awadbin-jawed/Desktop/Udacity/Files/denver.osm"
NEW_OSM_FILE = "/Users/awadbin-jawed/Desktop/Udacity/Files/denver_clean_node_tag_keys.osm"

def convert_key(letter1,letter2,key):
    l1 = key.find(letter1)
    better_key = key[:l1]+letter2+key[l1+1:]
    return better_key

def get_element(osm_file, tags=('node', 'way', 'relation')):

    context = iter(ET.iterparse(osm_file, events=('start', 'end')))
    _, root = next(context)
    for event, elem in context:
        if event == 'end' and elem.tag in tags:
            yield elem
            root.clear()

def clean_node_tag_keys(filename,newfilename):
    with open(filename, "rb") as infile, open(newfilename, "wb") as outfile:
        outfile.write('<?xml version="1.0" encoding="UTF-8"?>\n')
        outfile.write('<osm>\n  ')

        for elem in get_element(infile):
            if elem.tag == "node":
                if elem.find("tag") != -1:
                    for tag in elem.iter("tag"):
                        if tag.attrib['k'] == 'FIXME' or tag.attrib['k'] == 'fixme' :
                            elem.remove(tag)
                            if tag.attrib['v'] == 'continue' or tag.attrib['v'] == 'continues'
                                ET.SubElement(elem,'tag', {'k':'noexit', 'v':'yes'})
                            elif tag.attrib['v'] == 'address & hours':
                                ET.SubElement(elem,'tag', {'k':'addr:city', 'v':Denver})
                                ET.SubElement(elem,'tag', {'k':'addr:country', 'v':'US'})
                                ET.SubElement(elem,'tag', {'k':'addr:state', 'v':CO})
                        elif ' ' in tag.attrib['k']:
```

```
                    tag.attrib['k'] = convert_key(' ','_',tag.attrib['k']))
                elif '.' in tag.attrib['k']:
                    tag.attrib['k'] = convert_key('.',':',tag.attrib['k']))
                elif 'alt_name:' in tag.attrib['k']:
                    tag.attrib['k'] = convert_key(':','_',tag.attrib['k'])
            outfile.write(ET.tostring(elem, encoding='utf-8'))
        outfile.write('</osm>')

clean_node_tag_keys(OSM_FILE,NEW_OSM_FILE)
```

We will have the following input from the data.py and schema.py codes:

```
# -*- coding: utf-8 -*-


import csv
import codecs
import pprint
import re
import xml.etree.cElementTree as ET
import cerberus

import sys
sys.path.append("/Users/awadbin-jawed/Desktop/Udacity/Files")

import schema

SCHEMA_PATH = "/Users/awadbin-jawed/Desktop/Udacity/Files/schema.py"

OSM_PATH = "/Users/awadbin-jawed/Desktop/Udacity/Files/denver.osm"

NODES_PATH = "/Users/awadbin-jawed/Desktop/Udacity/Files/nodes.csv"
NODE_TAGS_PATH = "/Users/awadbin-jawed/Desktop/Udacity/Files/nodes_tags.csv"
WAYS_PATH = "/Users/awadbin-jawed/Desktop/Udacity/Files/ways.csv"
WAY_NODES_PATH = "/Users/awadbin-jawed/Desktop/Udacity/Fils/ways_nodes.csv"
WAY_TAGS_PATH = "/Users/awadbin-jawed/Desktop/Udacity/Files/ways_tags.csv"

LOWER_COLON = re.compile(r'^([a-z]|_)+:([a-z]|_)+')
PROBLEMCHARS = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')

SCHEMA = schema.schema

NODE_FIELDS = ['id', 'lat', 'lon', 'user', 'uid', 'version', 'changeset', 'timestamp']
NODE_TAGS_FIELDS = ['id', 'key', 'value', 'type']
WAY_FIELDS = ['id', 'user', 'uid', 'version', 'changeset', 'timestamp']
WAY_TAGS_FIELDS = ['id', 'key', 'value', 'type']
WAY_NODES_FIELDS = ['id', 'node_id', 'position']


def shape_element(element, node_attr_fields=NODE_FIELDS, way_attr_fields=WAY_FIELDS,
                  problem_chars=PROBLEMCHARS, default_tag_type='regular'):
```

```
    node_attribs = {}
    way_attribs = {}
    way_nodes = []
    tags = []

    # YOUR CODE HERE
    if element.tag == 'node':
        return {'node': node_attribs, 'node_tags': tags}
    elif element.tag == 'way':
        return {'way': way_attribs, 'way_nodes': way_nodes, 'way_tags': tags}


# ================================================== #
#               Helper Functions                     #
# ================================================== #
def get_element(osm_file, tags=('node', 'way', 'relation')):

    context = ET.iterparse(osm_file, events=('start', 'end'))
    _, root = next(context)
    for event, elem in context:
        if event == 'end' and elem.tag in tags:
            yield elem
            root.clear()


def validate_element(element, validator, schema=SCHEMA):
    if validator.validate(element, schema) is not True:
        field, errors = next(validator.errors.iteritems())
        message_string = "\nElement of type '{0}' has the following errors:\n{1}"
        error_string = pprint.pformat(errors)

        raise Exception(message_string.format(field, error_string))


class UnicodeDictWriter(csv.DictWriter, object):

    def writerow(self, row):
        super(UnicodeDictWriter, self).writerow({
            k: (v.encode('utf-8') if isinstance(v, unicode) else v) for k, v in row.iteritems()
        })

    def writerows(self, rows):
        for row in rows:
            self.writerow(row)


# ================================================== #
#               Main Function                        #
# ================================================== #
def process_map(file_in, validate):

    with codecs.open(NODES_PATH, 'w') as nodes_file, \
         codecs.open(NODE_TAGS_PATH, 'w') as nodes_tags_file, \
```

```
        codecs.open(WAYS_PATH, 'w') as ways_file, \
        codecs.open(WAY_NODES_PATH, 'w') as way_nodes_file, \
        codecs.open(WAY_TAGS_PATH, 'w') as way_tags_file:

        nodes_writer = UnicodeDictWriter(nodes_file, NODE_FIELDS)
        node_tags_writer = UnicodeDictWriter(nodes_tags_file, NODE_TAGS_FIELDS)
        ways_writer = UnicodeDictWriter(ways_file, WAY_FIELDS)
        way_nodes_writer = UnicodeDictWriter(way_nodes_file, WAY_NODES_FIELDS)
        way_tags_writer = UnicodeDictWriter(way_tags_file, WAY_TAGS_FIELDS)

        nodes_writer.writeheader()
        node_tags_writer.writeheader()
        ways_writer.writeheader()
        way_nodes_writer.writeheader()
        way_tags_writer.writeheader()

        validator = cerberus.Validator()

        for element in get_element(file_in, tags=('node', 'way')):
            el = shape_element(element)
            if el:
                if validate is True:
                    validate_element(el, validator)

                if element.tag == 'node':
                    nodes_writer.writerow(el['node'])
                    node_tags_writer.writerows(el['node_tags'])
                elif element.tag == 'way':
                    ways_writer.writerow(el['way'])
                    way_nodes_writer.writerows(el['way_nodes'])
                    way_tags_writer.writerows(el['way_tags'])


if __name__ == '__main__':
    process_map(OSM_PATH, validate=True)
```

Five .csv files, which will be imported into SQL:

NODES_PATH = “nodes.csv”
NODE_TAGS_PATH = “nodes_tags.csv”
WAYS_PATH = “ways.csv”
WAY_NODES_PATH = “ways_nodes.csv”
WAY_TAGS_PATH = “ways_tags.csv”

Once I have created the SQLite database of the 5 .csv files, I will then need to create queries for parts of the database that interest me.

Step Two: Upload the .csv files to SQL

We now want to load the csv files into a database:

```
import sqlite3
import csv
from pprint import pprint


sqlite_file = 'denver.db'

# Connect to the database
conn = sqlite3.connect(sqlite_file)


# Get a cursor object
cur = conn.cursor()


cur.execute('''DROP TABLE IF EXISTS nodes_tags''')
conn.commit()


# Create the table, specifying the column names and data types:
cur.execute('''
    CREATE TABLE nodes_tags(id INTEGER, key TEXT, value TEXT,type TEXT)
''')
# commit the changes
conn.commit()


# Read in the csv file as a dictionary, format the
# data as a list of tuples:
with open('nodes_tags.csv','rb') as fin:
    dr = csv.DictReader(fin) # comma is default delimiter
    to_db = [(i['id'], i['key'], i['value'].decode("utf-8"), i['type']) for i in dr]


# insert the formatted data
cur.executemany("INSERT INTO nodes_tags(id, key, value,type) VALUES (?, ?, ?, ?);", to_db)
# commit the changes
conn.commit()


cur.execute('SELECT * FROM nodes_tags')
all_rows = cur.fetchall()
print('1):')
pprint(all_rows)


conn.close()
```

Problems Encountered:

Misspelled/Abbreviated variables:

I added “What is Expected” and updated the inconsistencies:

```
expected = ["Street", "Avenue", "Boulevard", "Drive", "Court",
            "Place", "Square", "Lane", "Road", "Trail", "Parkway", "Commons"]
```

In addition, I added the mapping:

```
mapping = { "Av": "Avenue",
            "Ave": "Avenue",
            "Ave.": "Avenue",
            "Avenue)": "Avenue"
            "Baselin": "Baseline",
            "Blf": "Boulevard",
            "Blvd": "Boulevard",
            "Blvd.": "Boulevard",
            "Cir.": "Circle",
            "Ct": "Court",
            "Dr": "Drive",
            "Hwy": "Highway",
            "Ln": "Lane",
            "Pkwy": "Parkway",
            "Pky": "Parkway",
            "Rd": "Road",
            "Rd.": "Road",
            "St": "Street",
            "St.": "Street"
            "Strret": "Street"
            "Thornton,": "Thornton"
            }
```

Accessing the file in the directory:

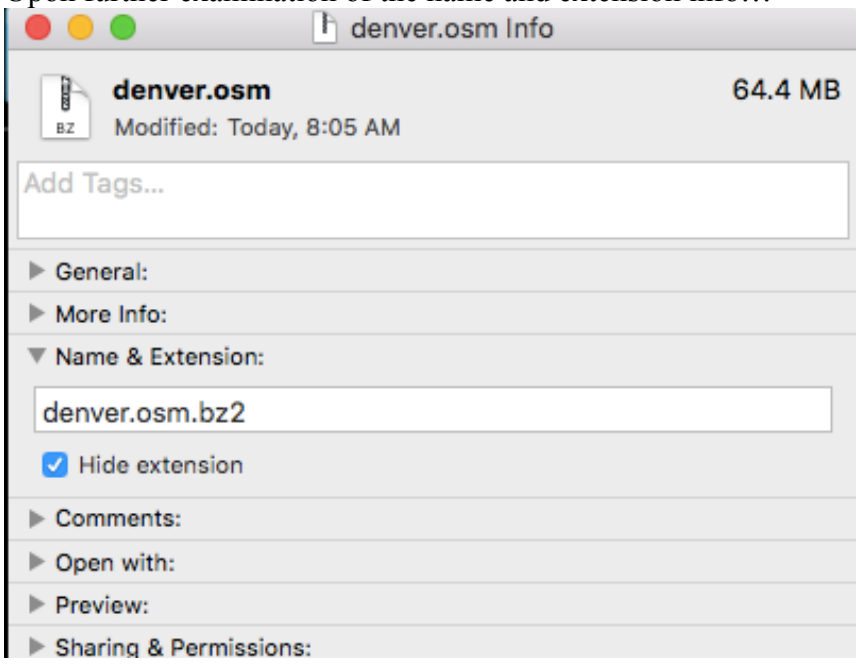
I kept getting an error stating that there was no such file or directory as denver.osm

```
In [9]: import xml.etree.cElementTree as ET
   ...: from collections import defaultdict
   ...: import re
   ...:
   ...: osm_file = open("denver.osm", "r")
   ...:

IOErrorTraceback (most recent call last)
<ipython-input-9-c68874416e98> in <module>()
      3 import re
      4
----> 5 osm_file = open("denver.osm", "r")
      6

IOError: [Errno 2] No such file or directory: 'denver.osm'
```

Upon further examination of the name and extension info…



… but after double-clicking on the file, it expanded and it expanded for the regular .osm file.

Anaconda Navigator:

The Anaconda Navigator, one random day, decided to keep crashing, so I decided to get a new, clean install. This resolved the issue of the navigator crashing, but the new install has iPython version 4. The codes in Udacity are catered towards the Python Version 2 environment. I kept getting the “Syntax Error” message.

In order to avoid the Syntax Errors, I resolved this issue by opening the MacOS terminal and installing the python version 2:

```
conda create -n python2 python=2.7 anaconda
```

If I need to ever activate this environment for future purposes, I simply need to enter:

```
source activate python2
```

Resources:

Sample SQL Report
https://gist.github.com/carlward/54ec1c91b62a5f911c42#file-sample_project-md

Blueprint for auditing
https://classroom.udacity.com/nanodegrees/nd002/parts/0021345404/modules/316820862075460/lessons/699689362/concepts/7796785460923

Clarification on the .osm.bz2 file versus the regular .osm file (post 7/10 by Myles Callan)
https://discussions.udacity.com/t/i-am-clueless-how-to-start-openstrretmaps-project/196647/7

Parsing
https://www.w3schools.com/xml/xpath_syntax.asp

Element Tree
https://docs.python.org/2/library/xml.etree.elementtree.html#module-xml.etree.ElementTree

Using CSV Module
https://docs.python.org/2/library/csv.html#csv.reader

Using the functions **audit.py** and **data.py** for better auditing, cleaning, and CSV creation
https://discussions.udacity.com/t/p3-project-combining-auditing-cleaning-and-csv-creation/231037

https://classroom.udacity.com/nanodegrees/nd002/parts/0021345404/modules/316820862075461/lessons/5436095827/concepts/54908788190923#

Using the function **mapparser.py** to get the tags
https://discussions.udacity.com/t/understanding-counter/286168/43

Using **schema.py** function for the last part of the project
https://discussions.udacity.com/t/osm-data-project-getting-started-running-locally/232476/2

Using tags.py to create expressions that scan for patterns in tags
https://discussions.udacity.com/t/cant-proceed-with-project-on-openstreet-data-wrangling-any-help/201104/53

Creating a CSV file to import to SQLite
https://discussions.udacity.com/t/final-project-importing-cerberus-and-schema/177231/10

https://gist.github.com/swwelch/f1144229848b407e0a5d13fcb7fbbd6f

To add what is expected in the dictionary of Street types (post 10/30 by Bhavya Ptn90)
https://discussions.udacity.com/t/final-project-openstreet-map/182754/10

Fixing the abbreviations of the street names
https://discussions.udacity.com/t/update-street-name/258913/2?u=awad_330179501771723

https://discussions.udacity.com/t/update-street-name/258913/2

Cleaning the data (pprint)
https://discussions.udacity.com/t/help-cleaning-data/169833

Using get_element( )
https://discussions.udacity.com/t/p3-openstreetmap-overview/172045/2

https://discussions.udacity.com/t/using-get-element/198489

https://stackoverflow.com/questions/3095434/inserting-newlines-in-xml-file-generated-via-xml-etree-elementtree-in-python

Installing Python Version 2.7
https://stackoverflow.com/questions/44348022/how-to-use-python-2-7-packages-in-anaconda-python-2-7-environment

Using data.py and schema.py codes
https://classroom.udacity.com/nanodegrees/nd002/parts/0021345404/modules/316820862075461/lessons/5436095827/concepts/54908788190923

Creating and Querying a Database with tables
https://discussions.udacity.com/t/creating-sql-database/316237

https://discussions.udacity.com/t/cant-proceed-with-project-on-openstreet-data-wrangling-any-help/201104/74

Grouping node key tags into categories
https://discussions.udacity.com/t/add-cleaning-functions-to-shape-element/348188/15

Uploading csv and creating .db file
https://discussions.udacity.com/t/creating-db-file-from-csv-files-with-non-ascii-unicode-characters/174958/7