OpenStreetMap Project
Data Wrangling with MongoDB

Chosen area: Curitiba, Brazil

Auditing:
File Audit – All.py is the code used to audit all OSM data
File Auditing Output Curitiba.txt has the output of the auditing code

Problems encountered

Post Codes
Post codes not following the gold standard of 5 numbers, followed by a dash and 3 numbers. For this region the first digit of the post code has to be 8, all entries matched the criteria. Examples:
In [24]: tagelementsvalues("curitiba.osm", "addr:postcode")
Examples: 80050360: 5, 637: 1, 80420: 1, 557: 1
Full Listing on Auditing Output Curitiba.txt

Phone Numbers
There was no uniformity on the phone numbers. Some presented country code, others area code and some none of them. As gold standard I have considered country code plus area code. Examples of lack of uniformity:
In [30]: tagelementsvalues("curitiba.osm", "phone")
(41) 3044-7778: 1
041 30265772: 1
+55 (41) 3276-0227: 1
32825700: 1
Full Listing on Auditing Output Curitiba.txt

Websites:
Many websites had the *http://* missing. Some examples:
In [34]: tagelementsvalues("curitiba.osm", "website")
http://www.hotelestreladosul.com.br/: 1
www.confidencecambio.com.br: 1
https://www.facebook.com/BarBaranUcrania: 1
Full Listing on Auditing Output Curitiba.txt

Street Names
I have found some over-abbreviated, others wrong. However I was surprised that on the overall there were not many problems (mainly R. for Rua). Note that I have considered Portuguese as the standard. Examples:

Al.: 1 (For Alameda)
Av: 4, Av.: 5 (For Avenida)
Aveninda: 1 (For Avenida)
R.: 39 (For Rua)
Rod.: 1 (For Rodovia)
Full Listing on Auditing Output Curitiba.txt

Reasons for the encountered problems
The causes for those problems are mainly the fact it is user entered data. To cover the lack of uniformity seen on most of the fields, one could force those specific fields pre-formatting (postcode, phone number, etc). For example, automatically convert entered post code to the local standard and notify the user if it does not have enough data.

Cleaning plan

Clean post codes

Criteria: Numbers have to follow the golden standard of 5 digits, followed by a dash and 3 digits (80000-000).

Method: Analyze digits first, then other characters like dots (.) and dashes (-) and convert the number to the golden format.

Exception: Numbers that have less than 8 digits are considered invalid and will have its data replaced by a default of 0000-000.

Cleaning Code:

```
def clean_postcode(entry): # Receives the postcode entry

    better_postcode = entry # Initializes better_postcode

    if entry[0] == ' ':
        better_postcode = entry[1:3] + entry[4:7] + "-" + entry[8:11]
        # Checks if the first digit is blank

    elif len(entry) > 8 and "-" == entry[6] and "." == entry[2]:
        better_postcode = entry[0:2] + entry[3:6] + "-" + entry[7:10]
        #Checks the length and other characters than numbers

    elif len(entry) > 7 and "-" != entry[5]:
        better_postcode = entry[0:5] + "-" + entry[5:9]
        #Checks if the postcode already follows the golden standard

    elif len(entry) < 9:
        better_postcode = '00000-000'
        #If not enough data, changes to all 0s value
```

```
return better_postcode # Returns the adjusted postcode to the main function
```

Postal Codes correction:

```
In [150]: clean_postcode('curitiba.osm')
80620010 => 80620-010
80050380 => 80050-380
557 => 00000-000
80050360 => 80050-360
82560040 => 82560-040
83430000 => 83430-000
80420011 => 80420-011
81050430 => 81050-430
80050470 => 80050-470
637 => 00000-000
83015140 => 83015-140
80410001 => 80410-001
81460275 => 81460-275
82820310 => 82820-310
82020470 => 82020-470
81730040 => 81730-040
80240041 => 80240-041
80410140 => 80410-140
83540000 => 83540-000
80540220 => 80540-220
82560440 => 82560-440
82510901 => 82510-901
80420 => 00000-000
80420000 => 80420-000
82.130-080 => 82130-080
81320000 => 81320-000
8300-502 => 00000-000
81050380 => 81050-380
Postcodes to be replaced: 28
```

Clean phone numbers

Criteria: Phone numbers have to be in the format of plus sign, followed by the country code, a space, area code between brackets, another space, prefix of 4 digits, a dash and finally a number of other 4 digits. The country code will always be 55 for Brazil, the area code will be 41 for Curitiba. If the data does not have the possibility of being right (9 digits), the entry will be converted to blank. Example of good entry: +55 (41) 1234-5678.

Method: Remove all characters other than numbers. Analyze its length to determine which data to recover to remount the phone number.

Exception: No exception defined at this moment.

Cleaning code

```
def clean_phone(entry): # Receives the phone number entry
    better_phone = entry # Initializes better_phone
    better_phone = re.sub(phonechars, '', entry)
    # Removes unwanted characters

    if len(better_phone) == 13:
        better_phone = '+55 (41) ' + better_phone[5:9] + '-' +
    better_phone[9:13] # Checks the length to adjust to golden format

    elif len(better_phone) == 12:
    better_phone = '+55 (41) ' + better_phone[4:8] + '-' + better_phone[8:12]

    elif len(better_phone) == 11:
    better_phone = '+55 (41) ' + better_phone[3:7] + '-' + better_phone[7:11]

    elif len(better_phone) == 10:
    better_phone = '+55 (41) ' + better_phone[2:6] + '-' + better_phone[6:10]

    elif len(better_phone) == 9:
    better_phone = ''

    elif len(better_phone) == 8: better_phone = '+55 (41) ' + better_phone[0:4]
+ '-' + better_phone[4:8]

return better_phone #Returns the adjusted phone to the main function
```

Phone numbers adjustment

```
In [6]: clean_phone('curitiba.osm')
(41) 3322-3844 => +55 (41) 3322-3844
(41) 3044-7778 => +55 (41) 3044-7778
41 33228229 => +55 (41) 3322-8229
(41) 3016-5970 => +55 (41) 3016-5970
+55 41 3363-7209 => +55 (41) 3363-7209
55 41 3319 1491 => +55 (41) 3319-1491
(41) 3286-1714 => +55 (41) 3286-1714
+55 41 3323 1031 => +55 (41) 3323-1031
+55 41 3324 2456 => +55 (41) 3324-2456
55 41 3262 3707 => +55 (41) 3262-3707
8854-6079 => +55 (41) 8854-6079
(41) 3205-3566 => +55 (41) 3205-3566
+55 41 9934 4676 => +55 (41) 9934-4676
41 3092-5996 => +55 (41) 3092-5996
41 3383-2364 => +55 (41) 3383-2364
+55 41 3079-2717 => +55 (41) 3079-2717
41 3283-5915 => +55 (41) 3283-5915
(41) 3322-2912 => +55 (41) 3322-2912
+55 41 3354-2448 => +55 (41) 3354-2448
+55 41 3354-3637 => +55 (41) 3354-3637
55 42 3246-2154 => +55 (41) 3246-2154
(41) 3053-8453 => +55 (41) 3053-8453
(41) 3262-7172 => +55 (41) 3262-7172
+55 (41) 3276-0227 => +55 (41) 3276-0227
```

+55 41 32238490 => +55 (41) 3223-8490
+55 41 30780358 => +55 (41) 3078-0358
55 41 3521-6046 => +55 (41) 3521-6046
+55 41 3556-3996 => +55 (41) 3556-3996
3252-6262 => +55 (41) 3252-6262
(41) 3264 6097 => +55 (41) 3264-6097
(41)3249-2695 => +55 (41) 3249-2695
+55 (41) 3636-1548 => +55 (41) 3636-1548
41 32961262 => +55 (41) 3296-1262
+55 41 3024-2757 => +55 (41) 3024-2757
+55 41 3362-1738 => +55 (41) 3362-1738
(41) 3082-9648 => +55 (41) 3082-9648
+554131524059 => +55 (41) 3152-4059
41 3372-2121 => +55 (41) 3372-2121
(41) 3257-9811 => +55 (41) 3257-9811
+55 41 3072-7000 => +55 (41) 3072-7000
(41) 4101-8609 => +55 (41) 4101-8609
+55 41 3222-4515 => +55 (41) 3222-4515
+55 41 3019-9580 => +55 (41) 3019-9580
+55 41 3249-4947 => +55 (41) 3249-4947
41 3028-3800 => +55 (41) 3028-3800
+55 41 3675-5600 => +55 (41) 3675-5600
+55 41 95183794 => +55 (41) 9518-3794
+55 42 3222 4222 => +55 (41) 3222-4222
41 3282-3159 => +55 (41) 3282-3159
+55413349-3698 => +55 (41) 3349-3698
+55 41 3098-8686 => +55 (41) 3098-8686
+55 4135246715 => +55 (41) 3524-6715
(41) 3264-9123 => +55 (41) 3264-9123
+55 41 3666-4383 => +55 (41) 3666-4383
(41) 3026-7473 => +55 (41) 3026-7473
(41) 3598-6636 => +55 (41) 3598-6636
(41) 3888-2239 => +55 (41) 3888-2239
3383-3015 => +55 (41) 3383-3015
(41) 3376-4098 => +55 (41) 3376-4098
55 41 30273566 => +55 (41) 3027-3566
(41) 3286-5153 => +55 (41) 3286-5153
+55 41 3212-5700 => +55 (41) 3212-5700
(41) 3377-1657 => +55 (41) 3377-1657
+55 41 3271-9000 => +55 (41) 3271-9000
41 3296 1253 => +55 (41) 3296-1253
(41) 3014-8014 => +55 (41) 3014-8014
(41) 3288-1112 => +55 (41) 3288-1112
41 3336-9222 => +55 (41) 3336-9222
55 41 88226709 => +55 (41) 8822-6709
55 41 8434-8005 => +55 (41) 8434-8005
+554132485329 => +55 (41) 3248-5329
3245-4545 => +55 (41) 3245-4545
41 3342-5453 => +55 (41) 3342-5453
(41) 3247-7904 => +55 (41) 3247-7904
++55 41 3238-2032 => +55 (41) 3238-2032
+55 41 3010-0044 => +55 (41) 3010-0044
41-30924155 => +55 (41) 3092-4155
(41) 3218-2408 => +55 (41) 3218-2408
41 3635-1201 => +55 (41) 3635-1201
4004-5700 => +55 (41) 4004-5700
+55 (41) 3304-4600 => +55 (41) 3304-4600

```
(41) 3264-1762 => +55 (41) 3264-1762
41 3336-0785 => +55 (41) 3336-0785
+55 41 3085 3453 => +55 (41) 3085-3453
(41) 3077-0408 => +55 (41) 3077-0408
(41) 3327-6573 => +55 (41) 3327-6573
+55 41 36421944 => +55 (41) 3642-1944
+55 41 3075-0100 => +55 (41) 3075-0100
3384-4344 => +55 (41) 3384-4344
(41) 3346-1989 => +55 (41) 3346-1989
41 3267 8827 => +55 (41) 3267-8827
(41) 3308-0370 => +55 (41) 3308-0370
+55 41 3304-2266 => +55 (41) 3304-2266
(41) 9615-9029 => +55 (41) 9615-9029
+55 41 21012000 => +55 (41) 2101-2000
+55 41 3675 5800 => +55 (41) 3675-5800
+55 (41) 3636-1060 => +55 (41) 3636-1060
+55 41 32480535 => +55 (41) 3248-0535
(41) 3357-6841 => +55 (41) 3357-6841
+55 41 3079-2504 => +55 (41) 3079-2504
55 41 30234041 => +55 (41) 3023-4041
+55 41 3073-0201 => +55 (41) 3073-0201
+55 41 3264-7234 => +55 (41) 3264-7234
(41) 3224-2527 => +55 (41) 3224-2527
+55 41 3347-7000 => +55 (41) 3347-7000
(41) 3262-8857 => +55 (41) 3262-8857
+55 41 95671321 => +55 (41) 9567-1321
+55 41 3256-4543 => +55 (41) 3256-4543
55 41 3339 9615 => +55 (41) 3339-9615
+55 (41) 3304-2222 => +55 (41) 3304-2222
+554196938680 => +55 (41) 9693-8680
+55 41 3154 7120 => +55 (41) 3154-7120
+55 41 3213 8700 => +55 (41) 3213-8700
+554132764551 => +55 (41) 3276-4551
41 30397856 => +55 (41) 3039-7856
(41) 3022-8324 => +55 (41) 3022-8324
Phones readjusted: 120
```

Clean website

Criteria: Check if the first digits of the website correspond to the golden standard of http:// or https://.

Method: Prepend websites prefixes for those who do not have it.

Exception: No exception defined at this moment.

Cleaning Code:

```
def clean_site(entry):
    better_site = entry # Initializes better_site
    if entry[0:7] != "http://" and entry[0:8] != "https://":
        better_site = "http://" + entry # Adjusts the entry
return better_site # Returns the adjusted site to the main function
```

```
Websites adjustment

In [99]: clean_site('curitiba.osm')
www.confidencecambio.com.br => http://www.confidencecambio.com.br
www.condor.com.br => http://www.condor.com.br
www.hungerbikes.com.br => http://www.hungerbikes.com.br
www.facebook.com/terrabrasilcafe =>
http://www.facebook.com/terrabrasilcafe
www.pracadojapao.wordpress.com => http://www.pracadojapao.wordpress.com
www.precisao.far.br => http://www.precisao.far.br
www.cloroquimica.com.br => http://www.cloroquimica.com.br
www.hotelsaara.com.br => http://www.hotelsaara.com.br
www.ciclesradar.com.br => http://www.ciclesradar.com.br
www.ciclessouza.com.br => http://www.ciclessouza.com.br
www.nayp.com.br => http://www.nayp.com.br
www.star12.com.br => http://www.star12.com.br
www.detran.pr.gov.br => http://www.detran.pr.gov.br
www.costafrio.com.br => http://www.costafrio.com.br
www.curitibagps.com.br => http://www.curitibagps.com.br
urbs.curitiba.pr.gov.br => http://urbs.curitiba.pr.gov.br
Websites to be replaced: 16
```

Clean Street Names

Criteria: Based on the results of the Audit, determine the mapping between the over-abbreviated and wrong street names and its correction. Replace using the map.

Method: Use re.sub to replace the wrong entries with its replacement.

Exception: Some entries although not following the golden standards represent unique entries, so its replacement is not worth nor a manual correction.

Cleaning code:

```
def clean_street(entry):

mapping = { "AVENIDA": "Avenida",
            "Av ": "Avenida ",
            "Avda.": "Avenida",
            "Av. ": "Avenida " ,
            "Aveninda": "Avenida",
            "Al. ": "Alameda ",
            "Br ": "BR ",
            "R. ": "Rua ",
            " Rua": "Rua",
            "Rod. ": "Rodovia "}
# Defines the mapping between wrong entries and its correction

    better_street = entry #Initializes better_street
    for item in mapping.keys():#Initiates the routine to use the mapping
```

```
        match = re.search(item, entry) #Searches for the keys on the entry
        if match:
            better_street = re.sub(item, mapping[item], entry)
            #Replaces the found key by its mapping
return better_street #Returns the better street to the main function
```

```
Street names correction

In [77]: clean_street('curitiba.osm')
Al. Dom Pedro II => Alameda Dom Pedro II
Rod. BR-376, Km 23,5 (sentido Joinville/ Curitiba => Rodovia BR-376, Km 23,5
(sentido Joinville/ Curitiba
Hospital Ônix Hospital Ônix 2321 Av. Vicente Machado => Hospital Ônix Hospital
Ônix 2321 Avenida Vicente Machado
Av das Torres => Avenida das Torres
Av Comendador Franco => Avenida Comendador Franco
Av Nossa Senhora de Lourdes => Avenida Nossa Senhora de Lourdes
Av. Comendador Franco => Avenida Comendador Franco
Av. Sete de Setembro => Avenida Sete de Setembro
Av. Cel. Francisco H. dos Santos => Avenida Cel. Francisco H. dos Santos
Br 376 => BR 376
Aveninda Marechal Floriano Peixoto => Avenida Marechal Floriano Peixoto
R. Sargento Lafayette => Rua Sargento Lafayette
R. Escola de Oficiais Especialistas => Rua Escola de Oficiais Especialistas
R. Sargento Milano => Rua Sargento Milano
R. Sgt. Roberto Maciel => Rua Sgt. Roberto Maciel
R. Sargento Erwin => Rua Sargento Erwin
Names to be replaced: 16
```

Data overview

File Sizes
curitiba.osm: 64,335KB
curitiba.osm.json: 72,348KB

Number of documents
In [**15**]: db.elements.find().count()
Out[**15**]: 332897

Number of nodes
In [**17**]: db.elements.find({'type': 'node'}).count()
Out[**17**]: 295895

Number of ways
In [**19**]: db.elements.find({'type': 'way'}).count()
Out[**19**]: 37002

Import to note that the number of nodes and ways obtained through MongoDB queries match the total amount obtained through audit code I created directly from Python,

showing both methods are consistent and rulling out any possible gap on the code and/or import into the database.

Number of unique users
```
> db.elements.distinct("created.user").length
386
```

Top 1 contributing user
```
> db.elements.aggregate([{"$group":{"_id":"$created.user",
"count":{"$sum":1}}},{'$sort':{'count':-1}},{'$limit':1}])
{ "_id" : "jump6024", "count" : 105799 }
```

Number of users appearing only once

```
> db.elements.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
{"$group":{"_id":"$count", "num_users":{"$sum":1}}}, {"$sort":{"_id":1}}, {"$limit":1}])
{ "_id" : 1, "num_users" : 65 }
```

Additional Ideas

This data could be used in conjunction with other sources (like Google Trends) as a support for GeoMarketing. It can represent a powerful tool to map potential consumer demands to best choose a place to open a new retail outlet, restaurant and other services. For instance, if a certain residential region of the map has a lack of service (restaurant, gas station, etc), the data on OSM could be used as an additional support when researching and determining the best location to invest. Therefore, a use like this helps in clarifying what a region may be needed as well as helps organizations and investors on their planning abilities.

Additional data exploration using MongoDB queries

Top amenities

The MongoDB query shows that the top amenity is parking, followed by fuel (gas station) and restaurant.

```
>
db.elements.aggregate([{"$match":{"amenity":{"$exists":1}}},{"$group":{"_id":"$amen
ity", "count":{"$sum":1}}},{"$sort":{"count":-1}}, {"$limit":10}])
{ "_id" : "parking", "count" : 248 }
{ "_id" : "fuel", "count" : 186 }
{ "_id" : "restaurant", "count" : 165 }
{ "_id" : "school", "count" : 130 }
{ "_id" : "place_of_worship", "count" : 96 }
```

{ "_id" : "pharmacy", "count" : 88 }
{ "_id" : "bank", "count" : 79 }
{ "_id" : "fast_food", "count" : 63 }
{ "_id" : "university", "count" : 57 }
{ "_id" : "hospital", "count" : 56 }

Top building type

The data shows the region is mainly residential (house and residential), followed by churches and commercial.

>
db.elements.aggregate([{"$match":{"building":{"$exists":1}}},{"$group":{"_id":"$building", "count":{"$sum":1}}},{"$sort":{"count":-1}}, {"$limit":10}])
{ "_id" : "yes", "count" : 4226 }
{ "_id" : "house", "count" : 522 }
{ "_id" : "residential", "count" : 268 }
{ "_id" : "church", "count" : 34 }
{ "_id" : "commercial", "count" : 33 }
{ "_id" : "industrial", "count" : 26 }
{ "_id" : "apartments", "count" : 22 }
{ "_id" : "school", "count" : 21 }
{ "_id" : "hospital", "count" : 15 }
{ "_id" : "stable", "count" : 13 }

Top brand

The region under analysis shows the top brands are related to gas station and cars/motorcycles stores.

db.elements.aggregate([{"$match":{"brand":{"$exists":1}}},{"$group":{"_id":"$brand", "count":{"$sum":1}}},{"$sort":{"count":-1}},
{"$limit":10}])
{ "_id" : "Ipiranga", "count" : 29 }
{ "_id" : "Multimarca", "count" : 14 }
{ "_id" : "BR", "count" : 13 }
{ "_id" : "Shell", "count" : 10 }
{ "_id" : "Branca", "count" : 4 }
{ "_id" : "Fiat", "count" : 2 }
{ "_id" : "Esso", "count" : 2 }
{ "_id" : "Petrobras", "count" : 2 }

{ "_id" : "Toyota", "count" : 2 }
{ "_id" : "Kawasaki", "count" : 1 }

Top land use

The land use matches with the other query showing the region is mainly residential. Interesting to note the amount of land use related to forest what could imply a very "green" city.

```
db.elements.aggregate([{"$match":{"landuse":{"$exists":1}}},{"$group":{"_id":"$landuse
", "count":{"$sum":1}}},{"$sort":{"count":-1}},
{"$limit":10}])
{ "_id" : "residential", "count" : 202 }
{ "_id" : "forest", "count" : 93 }
{ "_id" : "farm", "count" : 54 }
{ "_id" : "industrial", "count" : 48 }
{ "_id" : "cemetery", "count" : 30 }
{ "_id" : "grass", "count" : 27 }
{ "_id" : "reservoir", "count" : 25 }
{ "_id" : "village_green", "count" : 24 }
{ "_id" : "quarry", "count" : 15 }
{ "_id" : "recreation_ground", "count" : 12 }
```

Top max speed

The top max speed list shows the streets have mainly small max speeds (40 and 60 km/h). What brings our attention is that 50km/h exists but is not widely used (only 5 streets have this limit).

```
db.elements.aggregate([{"$match":{"maxspeed":{"$exists":1}}},{"$group":{"_id":"$maxs
peed", "count":{"$sum":1}}},{"$sort":{"count":-1}},
{"$limit":10}])
{ "_id" : "40", "count" : 809 }
{ "_id" : "60", "count" : 450 }
{ "_id" : "20", "count" : 192 }
{ "_id" : "80", "count" : 176 }
{ "_id" : "30", "count" : 165 }
{ "_id" : "70", "count" : 126 }
{ "_id" : "110", "count" : 74 }
{ "_id" : "50", "count" : 5 }
```

Top sports

The top sport is soccer followed by tennis.

```
db.elements.aggregate([{"$match":{"sport":{"$exists":1}}},{"$group":{"_id":"$sport",
"count":{"$sum":1}}},{"$sort":{"count":-1}},
{"$limit":10}])

{ "_id" : "soccer", "count" : 90 }
{ "_id" : "tennis", "count" : 40 }
{ "_id" : "swimming", "count" : 14 }
{ "_id" : "basketball", "count" : 14 }
{ "_id" : "volleyball", "count" : 9 }
{ "_id" : "multi", "count" : 3 }
{ "_id" : "athletics", "count" : 2 }
{ "_id" : "skateboard", "count" : 2 }
{ "_id" : "gymnastics", "count" : 2 }
{ "_id" : "skating", "count" : 1 }
```

Top surface type

The top surface type is mainly asphalt what can imply the city streets are well maintained.

```
db.elements.aggregate([{"$match":{"surface":{"$exists":1}}},{"$group":{"_id":"$surface",
"count":{"$sum":1}}},{"$sort":{"count":-1}},
{"$limit":10}])
{ "_id" : "asphalt", "count" : 4723 }
{ "_id" : "dirt", "count" : 743 }
{ "_id" : "unpaved", "count" : 621 }
{ "_id" : "paved", "count" : 399 }
{ "_id" : "ground", "count" : 243 }
{ "_id" : "compacted", "count" : 184 }
{ "_id" : "cobblestone", "count" : 102 }
{ "_id" : "gravel", "count" : 67 }
{ "_id" : "concrete", "count" : 28 }
{ "_id" : "fine_gravel", "count" : 27 }
```

Conclusion

It is interesting to see that MongoDB represents a very efficient way to store data in a database and perform queries. I could achieve somewhat similar results through Python coding but in the end, when comparing the amount of time I spent creating the code versus the time it took to enter data into MongoDB and perform the queries, it is clear that MongoDB would be the preferred method. It would make a huge difference when dealing with really big data.

Through the analysis of the data, the city of Curitiba does not seem to have increased number of dirty data and it may be because the city is not that big. Some insights from the analysis is that the city is mainly residential, and the same time it has an important number of streets that have as max speed between 40 and 60km/h, it also has an important number of forest areas, so one could use that to determine the quality of life. By finding out that the city has many parkings, gas stations and streets, we could imply that either the public transportation is not spread or the local culture gives more importance to private transport. The gathered data could be used to compare cities in the same country with similar size to provide additional insights about their differences.