Contents

1. Problems Encountered in the Map

In order to have an idea of the possible problems I could find, I first downloaded a sample of CABA City, which included two known neighbordhoods. Three problems were found:

- Different formats in house numbers (e.g.: 800;3502 / 9678,9680,9684 / 2401.00)
- Wrong data in house numbers (e.g.:+54 (11) 4857-2978 / Brick Hotel / C1033AAB)
- Different formats in phone numbers (e.g.: +54 810 888223 / (54-11) 5273 - 1200 / +541148023432)

Different formats in house numbers

For many cases, the house number was a concatenation of multiple numbers. It is not an error because it is something frequent with large buildings or stores that they have more than one housenumber. But two different characters were found to separate those numbers: comma and semicolon. Both were taken into account to create an array in MongoDB which holds all the housenombres for every document.

It also was detected that for some few cases the house number was in decimal format (e.g.: 2401.00, 2402.00). For these cases the data was converted into integer before uploading it into MongoDB.

Wrong data in house numbers

Some data was populated wrongly into housenumber (postal codes, phones, names, etc). Before uploading it into MongoDB, some regex where performed to ensure the data was correct. Only were allowed numbers and the following characters: '.',',' and ';'.

Different formats in phone numbers

Running a regex with the "phone" word included to filter the k values, six fields related to phone numbers were found (contact:phone, payment:telephone_cards, phone, phone_1, phone_2).

Plenty of different formats were found for the combitation: country_code + area_code + phone_number.

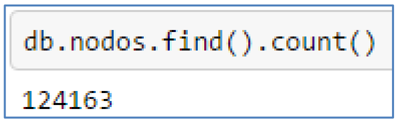
Before uploading the fields into MongoDB I converted the different formats into a standard one, which only includes the phone number without the country code and area code, due to the fact that it is the same in all analyzed regions.

2. Data Overview

File Size

caba&surrounds.osm _ _ _ _ _ _ _ _ _ 113MB

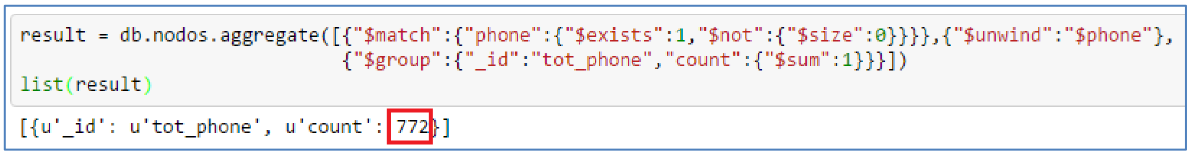
Number of documents



```
db.nodos.find().count()
```

```
124163
```

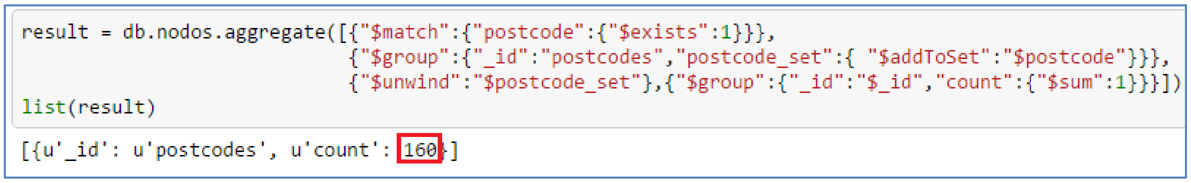
Number of phones



```
result = db.nodos.aggregate([{"$match":{"phone":{"$exists":1,"$not":{"$size":0}}}},{"$unwind":"$phone"},
                             {"$group":{"_id":"tot_phone","count":{"$sum":1}}}])
list(result)
```

```
[{u'_id': u'tot_phone', u'count': 772}]
```

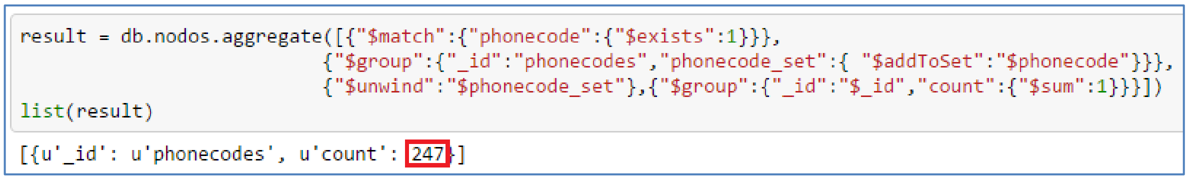
Number of postcodes



```
result = db.nodos.aggregate([{"$match":{"postcode":{"$exists":1}}},
                             {"$group":{"_id":"postcodes","postcode_set":{ "$addToSet":"$postcode"}}},
                             {"$unwind":"$postcode_set"},{"$group":{"_id":"$_id","count":{"$sum":1}}}])
list(result)
```

```
[{u'_id': u'postcodes', u'count': 160}]
```

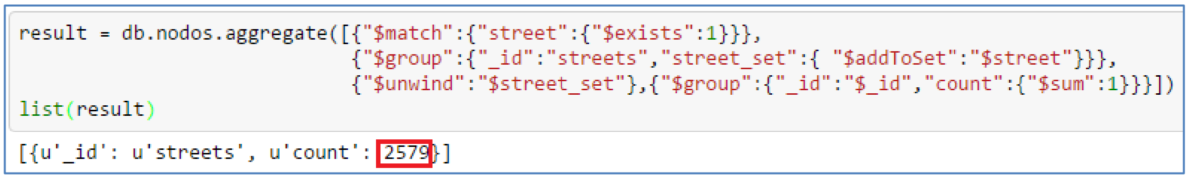
Number of phone codes



```
result = db.nodos.aggregate([{"$match":{"phonecode":{"$exists":1}}},
                            {"$group":{"_id":"phonecodes","phonecode_set":{ "$addToSet":"$phonecode"}}},
                            {"$unwind":"$phonecode_set"},{"$group":{"_id":"$_id","count":{"$sum":1}}}])
list(result)
```

```
[{u'_id': u'phonecodes', u'count': 247}]
```

Number of streets



```
result = db.nodos.aggregate([{"$match":{"street":{"$exists":1}}},
                             {"$group":{"_id":"streets","street_set":{ "$addToSet":"$street"}}},
                             {"$unwind":"$street_set"},{"$group":{"_id":"$_id","count":{"$sum":1}}}])
list(result)
```

```
[{u'_id': u'streets', u'count': 2579}]
```

3. Additional Ideas

Data validation and Measures of Data Quality

I detected many fields like phone and postcode number in many different formats, and even wrong data like Stores names in the postcode field. In the context of OpenStreetMap, if there were a validation of the data when it is populated, many of these mistakes could be avoided. In this sense, I think some Measures of Data Quality concepts could be applied to the data population itself. And it could exist some special users with authorization to define the data rules, and they could set this rules for the particular countries and regions.

As an example, the following two measures could be taken into account:

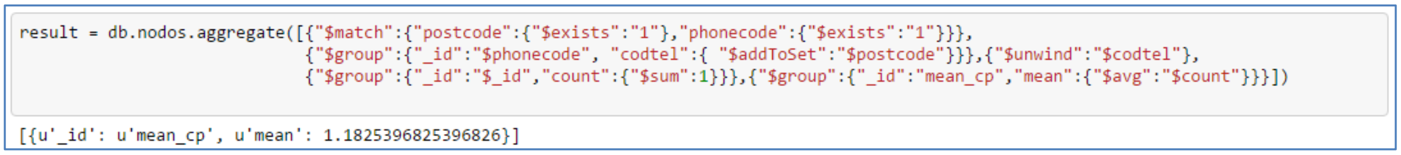
Validity: A particular pattern could be defined for the input, and it could be validated through a regex, or even with a data type.

Auditing Accuracy: Depending on the region, there could be lists of valid data, which are usually provided by official agencies, to compare with the data which is being populated. It could apply to streets, postcode, etc.

Additional data exploration using MongoDB queries

Relation between Phone Code and PostCode Number

In Buenos Aires the Phone Code is composed by the first four numbers of the phone. The first one is always 4, therefore only the three last numbers are taken into account (e.g.: for 4833-1212 the code is 833), and generally people who lives in the same neighborhood has a similar code (the code begin with 833 for instance). As the PostCode number also has a relation with the neighborhood, I wanted to determine if one phone code was related to one or more PostCode numbers. To determine that I ran the following query:

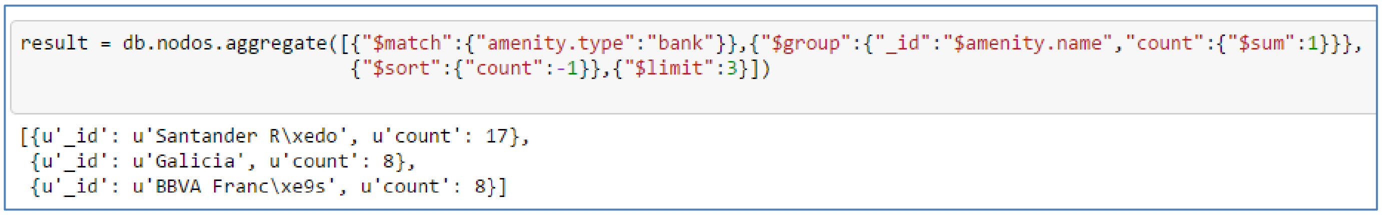


```
result = db.nodos.aggregate([{"$match":{"postcode":{"$exists":"1"},"phonecode":{"$exists":"1"}}},
                             {"$group":{"_id":"$phonecode", "codtel":{ "$addToSet":"$postcode"}}},{"$unwind":"$codtel"},
                             {"$group":{"_id":"$_id","count":{"$sum":1}}},{"$group":{"_id":"mean_cp","mean":{"$avg":"$count"}}}])

[{u'_id': u'mean_cp', u'mean': 1.1825396825396826}]
```

The results seems to show a correlation between the phone code and the postcode area, due to the fact that both are related to neighborhoods. Multiple phone codes could be related to a particular postcode area, but a particular phone code tends to be circumscribed to one postcode area.

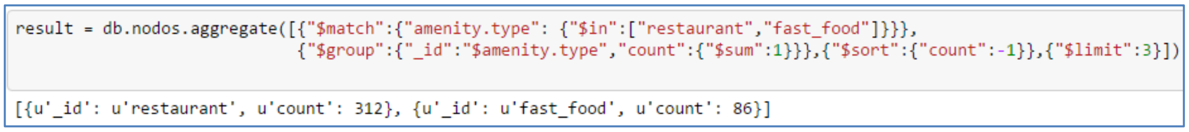
Three Most Popular Banks



```
result = db.nodos.aggregate([{"$match":{"amenity.type":"bank"}},{"$group":{"_id":"$amenity.name","count":{"$sum":1}}},
                             {"$sort":{"count":-1}},{"$limit":3}])
```

```
[{u'_id': u'Santander R\xedo', u'count': 17},
 {u'_id': u'Galicia', u'count': 8},
 {u'_id': u'BBVA Franc\xe9s', u'count': 8}]
```

Restaurants vs FastFood



```
result = db.nodos.aggregate([{"$match":{"amenity.type": {"$in":["restaurant","fast_food"]}}},
                             {"$group":{"_id":"$amenity.type","count":{"$sum":1}}},{"$sort":{"count":-1}},{"$limit":3}])
```

```
[{u'_id': u'restaurant', u'count': 312}, {u'_id': u'fast_food', u'count': 86}]
```