Open Street Map Data Case Study (SQL)

Map area

It was a hard task to choose the map which would have much data to correct. It was not the case for any Polish city, nor for Switzerland... So I chosen Miami, Florida, where I was on vacation in 2016.

Extract: https://mapzen.com/data/metro-extracts/metro/miami_florida/

Problems with data

After some investigation I've noticed following problems:

- many different street names abbreviations 'Ave', 'AVE' and 'Avenue' for "Avenue"
- there are some strange postal codes like '33312-1948', 'FL 33134' or even '361-0529â€Ž'
- state tags are inconsistent: 'FL', 'Fl' and 'Florida'
- as in the example SQL project, there are street names in second level "k" tags exported from Tiger GPS and divided into segments

I will clean the data in the same script which will be creating CSV files. I think it is more effective to clean majority of problems before loading data to database in order to have consistent data and remove rows, which are not useful.

Different street abbreviations

I have created a mapping, including all abreviations which I want to correct. I was not sure for some of them and also many of them seemed to be correct, so I only corrected these, where I was sure, that there was an abbreviation.

```
mapping = {"St": "Street", "St.": "Street", "Rd" : "Road", "Ave" : "Avenue", "AVE"
: "Avenue","Blvd" : "Boulevard", "Blvd." : "Boulevard", "Cir" : "Circle", "Ct" :
"Court", "Dr" : "Drive", "HWY" : "Highway", "N" : "North", "Pkwy" : "Parkway", "Trl"
: "Trail", "road" : "Road", "street" : "Street"}
```

Strange postal codes

Firstly, I remove FL from the beginning of the code, so that they alle are in the same format. It is done with the same function, which fixes postcodes with hyphen.

After doing some research here: https://fl.postcodebase.com/zip_code/33161-6695, I figured out, that the zip codes with hyphen are atually correct. The part after hyphen is an add on to the postal code. So I will separate them to be consistent with my database and save first part as a regular zip code and second as an add on with postcode type.

In the csv file it looks like this:

```
id,key,value,type
48850080,addition,3527,postcode
48850080,postcode,33487,addr
```

Lastly, there are two postcodes left, which are not correct: * '361-0529â€Ž' which was splitted in the function desribed above, so I will not write second part of it into database, because it cannot be converted to int. First part of this postal code is most likely not from Miami, but I'll leave it as it is. * '11890' which has correct format, but is most likely not from Miami. I will leave it as it is.

Inconsistent state tags

There are not that many different variations of state tags, but there are some data, which aren't state codes.

```
{'F': 1, 'FL': 813, 'FL.': 1, 'Fl': 11, 'Florida': 91, 'Inverrary Boulevard': 1,
'West Oakland Park Boulevard': 1, 'fl': 8, 'florida': 1}
```

I set every of this state codes as "FL". I am actually not sure, what to do with Boulevards, so I'll leave them as they are and have a look at these records in the database.

Checking data

Cities in database

I started with query from the example, to explore cities in the database. I noticed, that city names also need special cleaning - removing quotes.

```
SELECT tags.value, COUNT(*) as count FROM (SELECT * FROM node_tags UNION ALL SELECT
* FROM way_tags) tags WHERE tags.key LIKE '%city' GROUP BY tags.value ORDER BY
count DESC LIMIT 10;
Weston,18244
Miami,14291
Hialeah,1386
"Miami Beach",685
Homestead,477
"Coral Gables",459
Doral,370
"North Miami",366
"Miami Gardens",341
"North Miami Beach",315
```

I try the same for postal codes, because they should be more consistent that cities names.

```
SELECT tags.value, COUNT(*) as count  FROM (SELECT * FROM node_tags UNION ALL
SELECT * FROM way_tags) tags WHERE tags.key LIKE 'postcode' GROUP BY tags.value
ORDER BY count DESC LIMIT 10
```

```
33327,6770
33326,6469
33331,3117
33135,2150
33332,1883
33125,1801
33142,1782
33133,1411
33145,1283
33126,1209
```

It looks much better and there aren't that big differences between numbers. One city can obviously have different postal codes.

Data statistics

File sizes

```
miami_florida.osm  600MB
Miami DB           388.2MB
nodes.csv          222MB
nodes_tags.csv     11.5MB
ways.csv           20.3MB
ways_tags.csv      56.8MB
ways_nodes.csv     71.1MB
```

Number of nodes and nodes tags

```
sqlite> SELECT COUNT(*) FROM NODE;
2562651
sqlite> SELECT COUNT(*) FROM NODE_TAGS;
325862
```

Number of ways, ways nodes and ways tags

```
sqlite> SELECT COUNT(*) FROM WAY;
324409
sqlite> SELECT COUNT(*) FROM WAY_NODES;
2994488
sqlite> SELECT COUNT(*) FROM WAY_TAGS;
1652327
```

Number of contributors

```
sqlite> SELECT COUNT(DISTINCT(e.uid))  FROM (SELECT uid FROM node UNION ALL SELECT
uid FROM way) e;
1704
```

Number of active contributors (who contributed this year)

```
sqlite> SELECT COUNT(DISTINCT(e.uid))  FROM
(SELECT uid, timestamp FROM node UNION ALL SELECT uid, timestamp FROM way) e WHERE
e.timestamp LIKE '2017%';
313
```

It is actually surprising to see that nearly 20% of contributors were already active this year.

Additional Data Exploration

TOP 10 appearing amenities

```
sqlite> SELECT value, COUNT(*) as num FROM node_tags WHERE key='amenity' GROUP BY
value ORDER BY num DESC LIMIT 10;
school,1494
place_of_worship,566
restaurant,530
kindergarten,527
fire_station,293
fast_food,292
fountain,262
fuel,237
parking,224
police,186
```

There are really many schools. Let's have a look who inserted them.

```
SELECT user, COUNT(*) as count FROM node_tags INNER JOIN node on node.id =
node_tags.id where key = 'amenity' and value = 'school' GROUP BY user ORDER BY
count desc LIMIT  5;
iandees,1284
grouper,96
westendguy,17
wvdp,10
IvoSan,7
```

Most of them were inserted by one user and... on the same day.

```
sqlite> SELECT SUBSTR(Timestamp, 1, 10) as day, COUNT(*) as count FROM node_tags
INNER JOIN node on node.id = node_tags.id where key = 'amenity' and value =
'school' GROUP BY day ORDER BY count desc LIMIT  5;
2009-03-11,1284
2014-02-26,7
2012-01-22,4
2013-08-21,4
2011-05-05,3
```

I suppose that it was either a bot or a script, which made something like "initial load" for the most important schools.

Most popular fast foods

```
sqlite> SELECT value, COUNT(*) as number FROM node_tags WHERE ID in (SELECT id FROM
node_tags WHERE key = 'amenity' and value = 'fast_food')  and key = 'name' GROUP BY
value ORDER BY number DESC LIMIT 5;
Subway,59
"McDonald's",41
"Wendy's",20
"Burger King",12
"Papa John's",11
```

This is interesting, because I was sure, that McDonald's will be the most popular fast food, but it is actually Subway.

Additional ideas

Actually majority (57%) of the records are 2015 or older. I assume, that database would overwrite an record while updating it and not create a new record. Maybe ways names don't change so often, but let's have a look at nodes.

```
sqlite> SELECT SUBSTR(Timestamp, 1, 4) as year, COUNT(*) as count FROM node  GROUP
BY year ORDER BY count desc LIMIT  10;
2016|981531
2011|335451
2015|315292
2009|246390
2014|198557
2012|178768
2017|106666
2010|105782
2013|83314
2008|10900
```

There is more or less the same % of old records in nodes as in both nodes and ways.

I am aware, that amenities don't change every other day, but I suppose, that after including some stronger data validation (e.g. only ints with particular length as post codes, no quotes in city names etc.), regular map updates and entries would bring also better data quality. I would use some marketing strategies to keep users engaged:

- Gamification with awards, leaderboard etc. (which is already described in the example project).
- Maybe some social media push if there are any people willing to do this. Here the presence and the awareness of this project should grow with time so that more people could try to automatise whole map. Here is the difficulty, that it should be done regularly and there should be at least one person constantly doing this. However, it also won't ensure, that people will be more engaged.

- I would try to collaborate for some amenities to get discount codes when updating their amenities. Benefits would be similar as from social media push, but on addition, there could me some data exchange between amenities and openstreetmaps to see how the coupons are used, who uses them and these data could be also some base for future data scientist and analyst courses. I think that some schools and univerisities could be interested in this project. On the other hand, I suppose it would be somehow difficult to implement, because there should be a Marketing or Sales team working with amenities. I can also imagine, that not many amenities would see such cooperation as important.

Other idea would be:

- making an API export from TripAdvisor, Booking.com and/or Google Maps and try to insert these data into openstreet maps. I am not sure how many people would be willing to do this if only award would be visibility on the leaderboard, but if not many, I would suggest another Udacity project. What I mean by this is, in some other course e.g. python programming or backend developer to make a project which would export data from one of the services mentioned before and push them into openstreetmaps. For the test purposes I would use something like openstreetmaps staging, but after approving the code students could actually use it or at least give it to openstreetmaps. Maybe one day openstreetmaps would be able to completely automatise their website thank to Udacity students :) It is for sure a big project for Udacity and not even sure if it would make sense for students.

Conclusion

I am actually suprised, that there are particular records which are clean and some which are not. I would suppose, that cities names would be cleaner than fast foods, but it seems to be other way round. There were also many inconsistent postal codes, which I also didn't expect. To make this data even more evaluable, I would suggest to clean names of the cities. I think, there will also be some records, which will not be analysis-friendly, but the task here is to clean majority of them.

Another thig that I suggest, would be stronger verification during input of the data. For example that postal codes should be ONLY numbers, street names should be chosen from the list etc. Then the data would be much more easy to clean. However I am positively surprised, that it wasn't actually that hard to clean this data so that an analysis is possible.