Sam Miller’s OpenStreetMap Case Study:

Map Area:

Potomac, MD United States

- https://mapzen.com/data/metro-extracts/your-extracts/f1cd461cb6f4
- I lived in this town when I was a child and wanted to review the quality of the OpenStreetMaps data.

Problems Encountered In the Map:

After thoroughly reviewing the data from my extract in SQLite3 I found a few problems. Most of these problems were based upon data sets not being consistent. I also found a few data points that were completely incorrect. These problems are as follows:

1. Inconsistent Pharmacy Names:
    - SQL Query:
        - SELECT id, key, value
        - from nodes_tags, (select id as the_id from nodes_tags where value = 'pharmacy') as subq
        - where id = the_id and key = 'name';
    - Example of Inconsistent Values:

Id	**Key**	**Value**
1595591459	name	Giant Pharmacy
1603830741	name	CVS Pharmacy
1668199591	name	CVS
3676838644	name	Giant
693960686	name	Rite-Aid
582788382	name	Rite Aid
808007344	name	CVS/Pharmacy
2433173604	name	CVS/pharmacy

    - Cleaning Method:
        - I decided the best solution would be to only the have pharmacy name and to remove the word pharmacy from all values. I also removed all characters that were not letters.
    - Cleaning Code Function:

```
def fix_pharmacy(value):
   # remove non-letters and pharmacy from names of pharmacies
   value = value.replace('/', '')
   value = value.replace('-', '')
   value = value.replace('Pharmacy', '')
   value = value.replace('pharmacy', '')
   value = value.replace(' ', '')
   return value
```

2. Inconsistent and Inaccurate County Names which should not include State Name:
    - SQL Query:

        - select id, key, value, type
        - from ways_tags, (select id as the_id from ways_tags where key = 'county') as subq
        - where id = the_id and key = 'county';
    - Example of Inconsistent Values:

Id	Key	Value	Type
5968752	county	Montgomery, MD	tiger
435217294	county	District of Columbia, DC:Montgomery, MD	tiger
435134463	county	District of Columbia, DC	tiger
26709153	county	District of Columbia, DC:Montgomery, MD	tiger
8797811	county	Arlington, VA:Fairfax, VA	tiger
8815742	county	Fairfax, VA:Loudoun, VA	tiger
8840643	county	Fairfax, VA:Loudoun, VA	tiger

- Cleaning Method:
    - Since I did not find a way to distinguish the correct county when multiple counties were listed for one location I decided to leave those entries unchanged.
    - When there was only one county listed in the value I removed the state abbreviation from the value.
- Cleaning Code Function:

```
def fix_county(value):
    # if only one county listed, remove state abbreviation
    if value.find(':') == -1 and value.find(';') == -1 :
        value = value.split(',', 1)[0]
    return value
```

3. Inconsistent and Incorrect Postcode Values:
    - SQL Query:
        - select id, key, value
        - from nodes_tags, (select id as the_id from nodes_tags where key = 'postcode') as subq
        - where id = the_id and key = 'postcode';
    - Example of Inconsistent Values:

Id	Key	Value
759552711	postcode	20036-5305
2393440167	postcode	2011
Correct Value:	Epiphany Open Pit Beef & Subs	4128 Georgia Ave NW, Washington, DC 20011
4332706293	postcode	2005

Correct Value:	7-Eleven	1400 Rhode Island Ave NW, Washington, DC 20005
4362220268	postcode	20850

- Cleaning Method:
  - I first corrected all values that were incorrect.
  - I then made all values only 5 digits.
- Cleaning Code Function:

```
def fix_postcode(value):
   # modify all postcode numbers so they are simply 5 digits
   # 2011 should be 20011
   # https://www.yelp.com/biz/epiphany-open-pit-beef-and-subs-
washington
   if value == '2011':
      value = '20011'
   # 2005 should be 20005
   # https://www.google.com/maps/place/7-Eleven/@38.9084384,-
77.0322289,21z/data=!4m18!1m12!4m11!1m3!2m2!1d-
77.0321551!2d38.9085401!1m6!1m2!1s0x89b7b7ea61f0d259:0x8fc5b8b
0c95d49af!2s7-Eleven+Washington,+DC!2m2!1d-
77.0321722!2d38.9085452!3m4!1s0x89b7b7ea61f0d259:0x8fc5b8b0c95
d49af!8m2!3d38.9085452!4d-77.0321722
   if value == '2005':
      value = '20005'
   if len(value)!=5:
      value = value[0:5]
   return value
```

4. Inconsistent and Incorrect Phone Values:
   - SQL Query:
     - select id, key, value
     - from nodes_tags
     - where key = 'phone';
   - Example Inconsistent and Incorrect Phone Values:

Id	**Key**	**Value**
4085798290	phone	+13192881
358254855	phone	649 3555
4217721745	phone	(202) 640-4361
4270800540	phone	+1 202 298 8888
4312175837	phone	+1 703 4343286
4353432310	phone	202-986-1200
4353442989	phone	202.588.5766
4208482894	phone	3016571114
4097006388	phone	+1 301 428 8128
4089935022	phone	+1 202-939-4750
3919278833	phone	703-709-4114 or 703 794-2103
3586834753	phone	+1 866-RIDMTA
3378443530	phone	703-476-4500 x3
3373827507	phone	New Customer: (855) 837-8090 Current Customer: (703) 471-6336

2470456488	phone	Susanna Farm Nursery: (301) 972-7513

- Cleaning Method:
  - I first corrected all incorrect phone values.
  - I then made all phone values 10 digits long with no other characters.
- Cleaning Code Function:

  ```
  def fix_phone(value):
      # modify all phone numbers so they are simply 10 digits
      # +1 866-RIDMTA should be 1-866-RIDE-MTA
      # https://mta.maryland.gov/ride-mta-to-african-american-festival
      if value == '+1 866-RIDMTA':
          value = '8667433682'
      # +13192881 should be 2023192881
      # https://www.yelp.com/biz/jolie-jewelry-washington
      if value == '+13192881':
          value =  '2023192881'
      # 649 3555 should be 3016493555
      # https://standrewapostle.org/School-WP/contact/
      if value == '649 3555':
          value =  '3016493555'
      value = value.replace('+1 ', '')
      value = value.replace('-', '')
      value = value.replace(' ', '')
      value = value.replace('.', '')
      value = value.replace('(', '')
      value = value.replace(')', '')
      value = value.replace('+1', '')
      value = value.replace('New Customer: ', '')
      value = value.replace('Susanna Farm Nursery: ', '')
      value = value.replace('tel:', '')
      if len(value)>10:
          value = value[0:10]
      return value
  ```

5. Inconsistent Capital Bike Share Values:
   - SQL Query:

     ```
     select id, key, value
     from nodes_tags, (select id as the_id from nodes_tags where value = 'bicycle_rental') as subq
     where id = the_id and key = 'name';
     ```

   - Example Inconsistent Values:

Id	**Key**	**Value**
2276095306	name	Capital BikeShare
2276095321	name	Metro Center / 12th and G St NW
2276095353	name	Thomas Circle
2276095430	name	11th and F St NW
2276095399	name	Connecticut Ave and Newark St NW / Cleveland Park

- These values are inconsistent since some values do not list the name of the bike share location but rather only “Capital BikeShare”. Other values list the name of the bike share but not the location of the bike share. Still other values list the location of the bike share but not the name of the bike share. Finally, some list the location of the bike share and then the name of the bike share and others list the name of the bike share then the location of the bike share. To correct this inconsistency I would add a tag with k=’intersection’ and v= the street crossing. I would then make the tag which has a k = ‘name’ have a value equal to the name of the bike share. Finally, I would remove all tags which have k = ‘name’ and a v=’Capital BikeShare’. Instead, these nodes should have a tag with a k=’network’ and a v = ‘’Capital BikeShare’.
- I did not clean this specific inconsistency.

Overview Of The Data:

- Size of File:
  - http://www.sqlite.org/pragma.html#pragma_page_countsqlite
  - SQL Code:PRAGMA page_count;
  - Output: 255505
  - SQL Code: PRAGMA page_size;
  - Output: 1024
  - Math: 255505 pages x 1024 bytes/page x 1/1024 kilobyte/byte x 1/1024 Megabytes/kilobyte
  - = 249.52 MB
- Number of Unique Users:
  - http://stackoverflow.com/questions/6712127/select-countdistinct-name-from-several-tables
  - SQL Code:

```
select count(distinct x.user)
from (select user
        from nodes
       union all
       select user
         from ways) x;
```

  - Output:
    - 1855 users
- Top 3 Users with most posts:
  - SQL Code:

```
select x.user, count(x.user) as num
from (select user
        from nodes
       union all
       select user
         from ways) x
group by x.user
order by num
desc
limit 3;
```

  - Output:

User	# Posts

aude	693618
woodpeck_fixbot	157689
ingalls	150859

- Number of nodes and ways:
  - SQL Code:
    - select count(distinct id)
    - from nodes;
  - Output
    - 2058806 nodes
  - SQL Code:
    - select count(distinct id)
    - from ways;
  - Output:
    - 235462 ways
- Top 20 most prevalent amenities:
  - SQL Code:
    - select value, count(*) as num
    - from nodes_tags
    - where key = 'amenity'
    - group by value
    - order by num
    - DESC
    - limit 20;
  - Output:

Amenity	**Prevalence**
restaurant	1358
school	662
place_of_worship	589
fast_food	484
cafe	444
bank	341
bench	269
parking	258
post_box	170
bicycle_parking	168
fuel	167
waste_basket	159
drinking_water	145
bar	132
bicycle_rental	129
pub	116
pharmacy	102
post_office	91
kindergarten	89
toilets	78

- Most Popular Fast Food:
  - SQL Code:
    - select value, count(*) as num
    - from nodes_tags, (select id as the_id from nodes_tags where value = 'fast_food')

- where id = the_id and key = 'name'
- group by value
- order by num
- desc
- limit 20;

- Output:

Fast Food Restaurant	# of Restaurants
Subway	47
"McDonald's"	31
Chipotle	19
"Five Guys"	13
"Dunkin' Donuts"	10
"Burger King"	8
"Domino's"	5
"Papa John's"	5
"Pizza Hut"	5
Quiznos	5
"Taylor Gourmet"	5
"Wendy's"	5
"Baja Fresh"	4
Chick-fil-A	4
"Chop't"	4
"Domino's Pizza"	4
Baskin-Robbins	3
"California Tortilla"	3
"Jimmy John's"	3
"Julia's Empanadas"	3

Additional Ideas:

- Ideas for Improving Data
  - I found many issues throughout the data with data points that were completely incorrect. Implementing a way to not allow users to input data, which is obviously incorrect, would greatly increase the data's validity.
    - For example, some postal codes only had 4 digits not 5. No zip code in the USA is less than 5 digits. I feel creating a way to not allow users to import a data point when a postal code with less 5 digits in the USA would decrease postal code errors.
    - All of the incorrect phone numbers in my specific extract were phone numbers missing digits. For this reason, I believe allowing phone numbers to be entered as only 10 digits numbers would decrease the amount of incorrect phone numbers that would be entered into the data points.
    - I understand that making these improvements could be difficult for a few reasons. First, implementing any constraint on data inputs decreases the amount information that can be disclosed in such data points. For example, if someone wanted to enter an extension of a specific party for phone number they could not. This could increase the frustration for those who create data points in the system since it

        is an open source system and decrease the participation of such people in the future.

- Additional Data Exploration:
    - I feel to get a better grasp of the density of certain objects on the map, finding the prevalence of those objects per area would be useful. For example, how many Starbucks per square foot are there in certain areas of DC and NYC.

Conclusion:

- Although there is a large amount of free map data about my area of DC, I found the data to be very inconsistent with many inaccuracies. If feel with a better-enforced system for entering certain data point values this inconsistent and inaccurate information would be greatly decreased. Sadly, it seems implementing these changes could increase the amount of capital it costs to create these maps and since these maps are free, it is hard to argue with such data inaccuracies.