OpenStreetMap Data Case Study

Map Area

Houston, United States

- https://www.openstreetmap.org/export#map=13/29.7552/-95.2944
- https://mapzen.com/data/metro-extracts/metro/houston_texas/

I lived in Houston for 6 years. I take my graduate school in Houston. So I want to get better understanding about Houston. That is the reason to study the data about Houston.

Problems Encountered in the Map

After I check the sample data, I noticed two problems with the data, which I will discuss in the following order:

- Inconsistent postal codes (“77096”, “77339-1510”, “TX 77005”)

- Abbreviated street name(‘Dr’ with ‘Drive’, ‘St’ with ‘Street’ , ‘Ave’ with ‘Avenue’, ‘Blvd’ with ‘Boulevard’, ‘Fwy’ with ‘Freeway’, ‘Road’ with ‘Rd’, ‘Lane’ with ‘Ln’, ‘Ste’ with ‘Suite’, ‘Pkwy’ with ‘Parkway’)

Audit data

Code to audit abbreviated street name

```
mapping = {"St": "Street", 'Rd':'Road', 'Ave':'Avenue', 'Dr':'Drive', 'Blvd':'Boulevard',
'Fwy':'Freeway', 'Pkwy':'Parkway'}

def auditstreet(name, mapping):
    words = name.split()
    for i in range(len(words)):
        if words[i] in mapping:
            if i != 0 and words[i-1].lower() not in ['suite', 'ste.', 'ste']:
                words[i] = mapping[words[i]]
    name = " ".join(words)
    return name
```

Code to audit inconsistent postal codes

Because I lived in the Houston area before. I know that the prefix of postcode in Houston area is ''77'. So I used the following code to fix the inconsistent postal codes.

```
def auditpostcode(value):
    value = str(value)
    p = value.find('77')
    if p == -1:
        return None
    else:
        return value[p:p+5]
```

Postal codes (after audit)

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags
         UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key='postcode'
GROUP BY tags.value
ORDER BY count DESC
LIMIT 10;
```

Here are the top ten results, beginning with the highest count:

```
77096,484
77449,271
77551,263
77401,212
77339,195
77494,177
77002,117
77586,101
77076,84
77006,69
```

After audit, all the inconsistence problem is solved.

Street name (after audit)

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags
         UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key='street'
GROUP BY tags.value
ORDER BY count DESC
LIMIT 10;
```

Here are the top ten results, beginning with the highest count:

"Beluche Drive",118
"Pine Street",108
"Dominique Drive",99
"Jason Street",94
"Kingwood Drive",92
"Bucktrout Lane",85
"Westheimer Road",83
"Holly Street",71
"Braesvalley Drive",56
"NASA Parkway",55

After audit, all the inconsistence problem is solved.

Sort cities by count

```
sqlite> SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key LIKE '%city'
GROUP BY tags.value
ORDER BY count DESC
LIMIT 10;
```

Houston,3957
Galveston,869
Katy,477
Tomball,311
Bellaire,262

Kingwood,227
"Sugar Land",125
Humble,100
Seabrook,99
"Missouri City",88

Data Overview and Additional Ideas

File sizes

```
Houston_texas.osm ......... 750 MB
houston.db .......... 437 MB
nodes.csv ............. 281 MB
nodes_tags.csv ........ 6.09 MB
ways.csv .............. 25.5 MB
ways_tags.csv ......... 69.7 MB
ways_nodes.cv ......... 99.0 MB
```

Number of nodes

```
sqlite> SELECT COUNT(*) FROM nodes;
```

3490896

Number of ways

```
sqlite> SELECT COUNT(*) FROM ways;
```

442305

Number of unique users

```
sqlite> SELECT COUNT(DISTINCT(uniontable.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways)
uniontable;
```

1984

Top 10 contributing users

```
sqlite> SELECT uniontable.user, COUNT(*) as num
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways)
uniontable
GROUP BY uniontable.user
ORDER BY num DESC
LIMIT 10;
```

woodpeck_fixbot,556865
TexasNHD,536682
afdreher,485689
scottyc,203234
cammace,193610
claysmalley,137302
brianboru,115728
skquinn,86063
RoadGeek_MD99,81058
25or6to4,58765

Number of users appearing only once (having 1 post)

```
sqlite> SELECT COUNT(*)
FROM
    (SELECT e.user, COUNT(*) as num
     FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
     GROUP BY e.user
     HAVING num=1)  u;
```

369

Additional Data Exploration

Top 10 appearing amenities

```
sqlite> SELECT value, COUNT(*) as num
FROM nodes_tags
WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 10;
```

place_of_worship,2205

school,803
fountain,722
restaurant,701
fast_food,640
fire_station,349
fuel,281
pharmacy,177
bank,173
police,160

Top gas station

```
sqlite> SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value= 'fuel') i
ON nodes_tags.id=i.id
WHERE nodes_tags.key='brand'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 10;
```

```
Shell|38
Chevron|14
Exxon|12
Valero|11
Citgo|4
Texaco|3
Gulf|2
Mobil|2
Raceway|2
Buc-ee's|1
```

Most popular banks

```
sqlite> SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='bank') i
ON nodes_tags.id=i.id
WHERE nodes_tags.key='name'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 10;
```

```
Chase|37
Wells Fargo|26
Bank of America|11
BBVA Compass|9
Capital One|8
Prosperity Bank|6
Moody National Bank|4
Amegy Bank|3
Capital One Bank|3
Frost Bank|3
```

Most popular cuisines

```
sqlite> SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='restaurant') i
ON nodes_tags.id=i.id
WHERE nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 10;
```

```
mexican,74
american,35
pizza,35
italian,32
chinese,28
seafood,21
burger,19
barbecue,16
sandwich,13
thai,10
```

Conclusion

After this review of the data, there has some wrong data, including the postcode and address. After audit, the data looks correct. Through SQL analysis, a simple analysis of restaurant, bank and gas station is obtained.