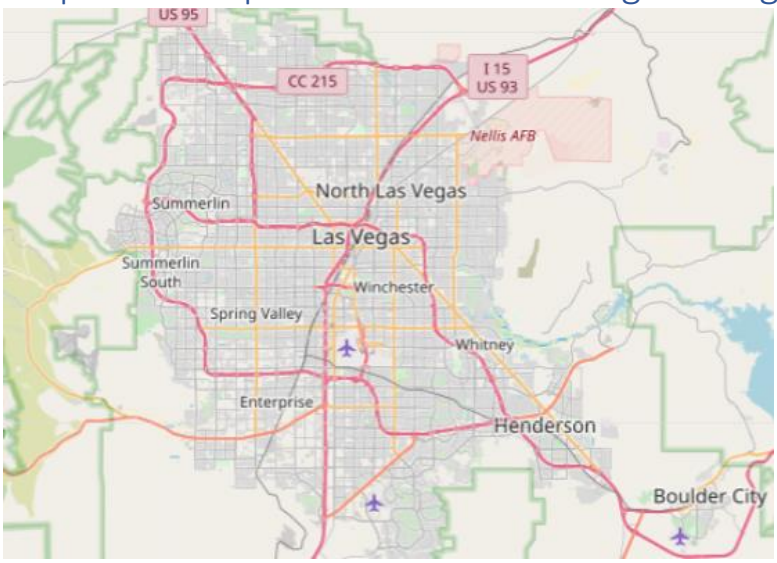
Data Wrangling Project – Aimee Rose

Map Area – Square Area Surrounding Las Vegas, Nevada



Pictured Area (note – area covered by the data set is slightly smaller and more skewed to the right as compared to the picture)
https://www.openstreetmap.org/export#map=10/36.1516/-115.1424

Las Vegas
https://www.openstreetmap.org/relation/170117

I thought it would be interesting to explore this area as it is where I live.

Files

Code Files:

- data.py – ~23KB
- schema.py – ~2KB
- schema.pyc – ~3KB
- createsamplefile – ~2KB

Source Data:

- Lasvegas.osm – Source data for square area around Las Vegas, NV. ~470MB
- Sample.osm – Sample of source data for square area mentioned above. ~48MB

Auditing Source Data:

- USPS Street Abrev.xlsx – ~22KB – File from downloaded data from the US Postal Service listing common abbreviations, USPS standard abbreviations, and their full equivalents.
    - USPS Street Abbrev.csv -- ~9KB – version of above file used in program – begins with contents of original file, then additional mappings derived from the mapping progress are appended to the file and are used in the cleaning process.

- Cities_List – ~1K – file created by cities auditing subroutine, cross reference to be used in update name function for cities.

Created CSVs

Nodes.csv – ~184MB

Nodes_tags.csv –~4MB

Ways.csv – ~14MB

Ways_nodes.csv – ~675KB

Ways_tags.csv – ~22.5MB

Database

OSM_Data_LV.db = ~249MB

Process Followed

I started with the code provided in the case study. Since the auditing of street types was included in that demonstration, I decided to begin with that for my cleaning task. However, when I work with data, I prefer to have a "source of truth" to start with, rather than just deciding what values should be. Considering that the data provided in the area I chose included only US values, it seemed like the most appropriate source of such validation data would be the United States Postal Service. So I went to their web site and found their list of the most commonly used street abbreviations, downloaded it, and saved it into an Excel file (included in submission).

I decided to convert this to CSV and read it in to serve in place of an in-script source. Next, I needed to handle any strange street types that were not found in this cross-reference. I decided to use the script to cycle through the remaining "weirdo" street types and use the code to help me map them. I wrote them back to the original file so that this cycle would not have to be redone at each run of the code.

After looking a bit at the data, I decided the next best field to be targeted for cleaning was City. There were a number of inconsistencies in the entries of this this data – including the state begin included with the city, misspellings such as "Las Vagas" and "Las Veggas." I followed a similar approach here, with the exception of starting with a file of outside data. Since the cities were very specific to this area, and were a small amount of values, I elected to cycle through these and map all by hand. In the case of both street and city, I chose to deal with the many variabilities in capitalization by forcing all values to upper case.

In the cases of both audit functions, I also wrote a preview function for each that allowed me to view all values generated from the auditing process (and later the update process) independent of the full "process map" function. This allowed me to actually run the auditing and update subroutines and view the results without having to run the full process. Because this saved time, I was actually able to run these on the full data set to test the results of the functions and use that input to refine the function design before testing it as part of the entire process. This also assisted in the writing and testing of the "Shape Element" function as well as the "Handle Tags" function I wrote to support it.

After testing these functions and processes incrementally, I was then able to use the sample.osm to test the full process. This identified a few additional bugs/adjustments which needed to be made to the code – such as the CSVs being written with empty lines in between the various records. After a successful test with the sample.OSM file with validation on, I ran the process on the full data set with validation off to speed the processing time, given the total size of the data file.

Analysis

Top Level Numbers

How Many Nodes?

Query
```
SELECT COUNT(DISTINCT id) from nodes;
```

Result
2204710

How Many Ways?

Query
```
SELECT COUNT(DISTINCT id) from ways;
```

Result
233422

How Many Users?

Query
```
SELECT Count(DISTINCT user) AS User_Count FROM
(
SELECT user
FROM ways
UNION ALL
SELECT user
FROM nodes
)
```

Result
1873

How Many Values Entered by Top 10 Users?

I looked at the amount of times each user appeared in the data set – I was surprised to find that only about 37% of the mentions were of the top 10. I was expecting a higher percentage. I was also surprised that unlike the sample project, we had only one in the top 10 that looked like it was a bot.

Query
```
SELECT user, Count(user) AS User_Count
FROM
(
SELECT user
FROM ways
UNION ALL
SELECT user
FROM nodes
)
GROUP BY user
ORDER BY User_Count DESC
LIMIT 10;
```

Results

User	User_Count
ALIMAMO	244964
STEPHEN SHOW	146574
THEDUTCHMAN13	113087
FINB2000	94121
ALECDHUSE	60777
WOODPECK_FIXBOT	55026
BMUSKAAN	52178
ABELLAO	49545
VENNREDD	47569
TOM_HOLLAND	45666

Reviewing City Results

I was particularly interested in the city results, since they were a target of my auditing program. I was pleased to see that it seemed that the cities values seemed well normalized in the final data set. What I found surprising was that, when broken down by how many times each city appeared in the data set, the most frequent value was NOT Las Vegas, even though the area selected focused on the core of this city, and population of Boulder City (https://en.wikipedia.org/wiki/Boulder_City,_Nevada) is only a small fraction of that of Las Vegas (https://en.wikipedia.org/wiki/Las_Vegas)

Query

```
SELECT key, type, value, count(value) AS Value_Count
FROM
(
SELECT  key, type, value
FROM nodes_tags
WHERE type like "city%" AND key like "addr%"
UNION ALL
SELECT key, type, value
FROM ways_tags
WHERE type LIKE "city%" AND key like "addr%"
)
GROUP BY key, type, value
ORDER BY Value_Count DESC;
```

Results

key	type	Value	Value_Count
addr	City	BOULDER CITY	2014
addr	City	LAS VEGAS	1028
addr	City	HENDERSON	185
addr	city	NORTH LAS VEGAS	143
addr	city	SUNRISE MANOR	18
addr	city	PARADISE	15
addr	city	SPRING VALLEY	4
addr	city	NELLIS AIR FORCE BASE	2
addr	city	BLUE DIAMOND	1
addr	city	ENTERPRISE	1
addr	city	WHITNEY	1

Why would Boulder City have so many entries? Given the large disparity in the population between the two cities, the idea that Boulder City had more items of interest than Las Vegas did not seem likely. Would it be possible that Boulder City just has a very dedicated user or two very thoroughly documenting its points of interest? Running a query breaking down the cities by the users who made the entries seems to support this hypothesis. The vast majority of entries related to Boulder City can be tracked back to a single user: “LGRV.” Further queries showed that this user only updated records relating to Boulder City, and all entries were related to addresses and buildings in the city.

Query

```
SELECT value, user, key, type, count(*) as
Record_Count
FROM
(
SELECT user, key, value, type
FROM ways, ways_tags
WHERE ways.id=ways_tags.id AND
ways_tags.type LIKE "city" AND
ways_tags.value="BOULDER CITY"
UNION ALL
SELECT user, key, value,type
FROM nodes, nodes_tags
```

Results

value	user	Key	type	Record_Count
BOULDER CITY	LGRV	Addr	city	1989
BOULDER CITY	SSR_317	Addr	city	5
BOULDER CITY	THEDUTCHMAN13	Addr	city	3
BOULDER CITY	B-JAZZ-BOT	Addr	city	2
BOULDER CITY	TOM_HOLLAND	Addr	city	2
BOULDER CITY	AM909	Addr	city	1

```
WHERE nodes.id=nodes_tags.id AND
nodes_tags.type LIKE "city" AND
nodes_tags.value="BOULDER CITY"
) comb
GROUP BY value, User, type, key
ORDER BY Record_Count DESC;
```

BOULDER CITY	BOLDERMAPPER	Addr	city	1
BOULDER CITY	CBEDDOW	Addr	city	1
BOULDER CITY	EDWARD	Addr	city	1
BOULDER CITY	FOSSOSM	Addr	city	1
BOULDER CITY	GOWESTTRAVEL	Addr	city	1
BOULDER CITY	JWHEELS9876	Addr	city	1
BOULDER CITY	KISAA	Addr	city	1
BOULDER CITY	MAXERICKSON	Addr	city	1
BOULDER CITY	STARGAZINGVIOLET	Addr	city	1
BOULDER CITY	STEPHEN_W	Addr	city	1
BOULDER CITY	TIMOTHY SMITH	Addr	city	1
BOULDER CITY	USER_5359	Addr	city	1

Possible Improvements

Overall, I was pleased with the results of the street type auditing, but I do see some areas for improvement in this area. While majority of the entries are well cleaned, I did discover a couple of flaws in the methods of cleaning that left some inconsistencies in the cleaning. In my code, I decided to use the regex provided in the case study to identify the street types, and then a simple string replacement to change the portion of the string that matched regex with its corresponding cleaned value.

Unfortunately, that had some unexpected consequences. For example, the most well-known street in Las Vegas, Las Vegas Boulevard, frequently appears in addresses with the cardinal direction following the street name. Since the regex was only looking for the final bit of field contents, for these entries, it only returned the cardinal direction (such as “South” or “S”). So, in cases where Boulevard was shortened to “BLVD” and a cardinal direction followed, the abbreviation for Boulevard was not cleaned.

In addition, in early tests of the code, I discovered that when the value returned by the regex was a single-letter abbreviation for a cardinal direction which had no punctuation, the code would replace any instance of that letter, even within a word. This returned results such as “LaSouth Vegas Boulevard South.” In response to this, I elected to ignore these situations and only do a replacement in instances where there was punctuation (Las Vegas Boulevard S.), leaving the “S” uncleaned. Were I do to pursue this project further, I would probably look to address both these situations to ensure consistency.

How would I address this? I would probably do an iteration through all words in the field value and map on a word-by-word level. This unfortunately, would also require potential rewrites of some or all of the supporting functions. Comparing whole words to the abbreviations lists would prevent the incorrect replacements that occurred using a straight substitution method, however, many of the entries also have single letters in the street names which are ***not*** intended to be abbreviations for example, “D Street” so I would need to examine the data an perhaps add additional logic to ensure no entries where the letters N, S, E and W are included in a street name but are not

intended to be cardinal directions. There may still also be unintended consequences that would have to be corrected for as well.

Resources

Project Guidance

C750 Getting Started with the project webinar recording (panopto.com)
https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=4f8920ae-7b84-4a81-9eeb-abd500008bcb

SQLITE

SQLITE Tutorial – Import CSV
https://www.sqlitetutorial.net/sqlite-import-csv/

Element Tree

Parsing XML Documents
https://pymotw.com/2/xml/etree/ElementTree/parse.html

Data Frames

Different Ways of Creating Dataframes
https://www.geeksforgeeks.org/different-ways-to-create-pandas-dataframe/

Using "Not In" operator for a dataframe
https://www.geeksforgeeks.org/check-if-a-value-exists-in-a-dataframe-using-in-not-in-operator-in-python-pandas/

Merging DataFrames
https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html

Using the From_Dict method in PANDAS
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.from_dict.html

Apply uppercase to a column in PANDAS
https://www.geeksforgeeks.org/apply-uppercase-to-a-column-in-pandas-dataframe/

How to get rid of unnamed 0 column in PANDAS DataFrame
https://stackoverflow.com/questions/36519086/how-to-get-rid-of-unnamed-0-column-in-a-pandas-dataframe

Splitting and Replacing Values in Strings

Count Occurrences of a Character in a String
https://www.geeksforgeeks.org/python-count-occurrences-of-a-character-in-string/

Does Python have a string 'contains' substring method?
https://stackoverflow.com/questions/3437059/does-python-have-a-string-contains-substring-method

How to split a string on the first occurrence in Python
https://www.kite.com/python/answers/how-to-split-a-string-on-the-first-occurrence-in-python

How to replace substrings of a string that match a pattern using regex in Python
https://www.kite.com/python/answers/how-to-replace-substrings-of-a-string-that-match-a-pattern-using-regex-in-python

Regex
https://docs.python.org/3/library/re.html

Dictionaries and Lists

How to append key value pairs in dictionary using dict update
https://thispointer.com/python-how-to-add-append-key-value-pairs-in-dictionary-using-dict-update/

How to append a dictionary to a list in Python
https://www.kite.com/python/answers/how-to-append-a-dictionary-to-a-list-in-python

USPS: Street Suffix Abbreviations Table
https://pe.usps.com/text/pub28/28apc_002.htm