Code Functionality

CRITERIA	MEETS SPECIFICATIONS
Does the code function as expected?	The code meets all functionality requirements

Code Readability

CRITERIA	MEETS SPECIFICATIONS
Does project utilize good coding practices?	Files were divided by functionality as well as meet all pep standards according to PyCharm IDE
Is the code commented in a way that is useful and not superfluous?	Comments included throughout in harder to understand areas

Problems encountered in your map

CRITERIA	MEETS SPECIFICATIONS
Does the project document the challenges encountered during the wrangling?	Readme.md on github lists challenges encountered at the end of the file

Is data cleaned programmatically?	Problems that are programmatically cleanable are cleaned. Others are listed in readme.md The most logical fields that are easily checked by an automated system were the postcode (zip), street, state, city names, and county names. All of the postcodes in the area I audited should start with 85 and for standardization I would only be looking at the 5 digit postcodes, not the 5-4 style. There were many other formats but almost all of them contained a format where a postcode would be found somewhere like this "junk85###junk" the junk being optional on either end of the string and the 85### being the data I wanted. Some examples of this are ('84009;85009'), ('085028'), ('885203') For street names I created a list of the most common street name suffixes that are abbreviated and cleaned them using a dictionary of those abbreviations to their full names. This was addapted from the lessons at Udacity.com for this project. The street names caused some issues because this area has alot of streets that do not end with the typical endings, many street names do not have any suffix at all. The follwoing are teh top 5 isntances and thier total counts as examples: ('Highway', 481), ('Sol', 308), ('Grande', 284), ('Loop', 236), ('Pass', 235), Everything I audited should be in the state of Arizona in this project so it should all be able to be changed to a specific format, I chose AZ I changed all state Tags to AZ no matter what they were for this data set. City names are listed and most of the errors are spelling mistakes or are are Phoenix:city and would be easier to correct manually than through an automated system. In my audit function I checked county information but there was only 1 County (Maricopa) listed a total of 7 times.

Overview of the data

CRITERIA	MEETS SPECIFICATIONS
Is the OSM XML large enough?	The full file is >2.5 GB, The sample included is >2MB
Are overview statistics of the dataset computed?	Database queries are used to provide a statistical overview of the dataset, like: • size of the file (os query not db) (line 33-62 of output of full run file) • number of unique users (line 69 of output of full run file) • number of nodes and ways (line 63 and 64 in output of full run file)

```
directory = os.getcwd()
for root, dirs, files in os.walk(directory, topdown=False):
    for name in files:
        f = os.path.join(root, name)
        print (name, naturalsize(os.path.getsize(f)))
```

```
'SELECT count(distinct(user)) FROM (' \
        'SELECT user FROM nodes ' \
        'UNION ALL SELECT user FROM ways);'
```

```
query = 'SELECT count(id) FROM nodes; '
total_rows = 0
result = cur.execute(query)
for row in result:
    print 'Nodes in database:', row[0]
    total_rows += row[0]

query = 'SELECT count(id) FROM ways;'

result = cur.execute(query)
for row in result:
    print 'Ways in database:', row[0]
    total_rows += row[0]

query = 'SELECT count(id) FROM nodes_tags;'

result = cur.execute(query)
for row in result:
    print 'Nodes Tags in database:', row[0]
    total_rows += row[0]
```

```
query = 'SELECT count(id) FROM ways_nodes;'

result = cur.execute(query)
for row in result:
    print 'Ways Nodes in database:', row[0]
    total_rows += row[0]

query = 'SELECT count(id) FROM ways_tags;'

result = cur.execute(query)
for row in result:
    print 'Ways Tags in database:', row[0]
    total_rows += row[0]

print '\nTotal Rows in database:\033[1m', total_rows,
"\033[0m"
```

- number of chosen type of nodes, like cafes, shops etc. (line 73-119 of output of full run file)

```
print '\nTop 10 distinct keys:'
for row in result:
    print (row)

query = "SELECT DISTINCT value, Count(*) AS [Count] " \
        "FROM nodes_tags GROUP BY value, key HAVING
(((key)='brand')) " \
        "ORDER BY Count(*) DESC limit 10;"

result = cur.execute(query)
print '\nTop 10 brands keys:'
for row in result:
    print (row)

query = "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;"

result = cur.execute(query)
print '\nTop 10 cuisines:'
for row in result:
    print (row)
conn.close()
```

	Additional statistics not in the list above are computed. For SQL submissions some queries make use of more than one table. (line 85-95 of output of full run file)
	`query = "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;"`

Are the database queries documented?	Queries are within the main.py file beginning at line 100

Other ideas about the dataset

CRITERIA	MEETS SPECIFICATIONS
Are ideas for additional improvements included?	Idea for improvement listed in readme.md file regarding Asian restaurant categories

Are benefits and problems with additional improvements discussed?	See above answer

Thoroughness and Succinctness of Submission

CRITERIA	MEETS SPECIFICATIONS
Is the submission long enough to answer the questions?	Readme.md is approximately 1 page while output of full run is approximately 3 more pages