

CSE 7345 Fall 2018
Quest 7 MongoDB 2

Due Sunday Oct 14

Use Mongo on the Lyle server or on your own machine.

Download from Canvas:

zipcodes.states.gps.csv includes zip codes, GPS Long, Lat, city, state, county

Note: leading zeros in zip codes are not given

	A	B	C	D	E	F
1	zip_code	latitude	longitude	city	state	county
2	501	40.92233	-72.6371	Holtsville	NY	Suffolk
3	544	40.92233	-72.6371	Holtsville	NY	Suffolk
4	601	18.16527	-66.7226	Adjuntas	PR	Adjuntas
5	602	18.3931	-67.181	Aguada	PR	Aguada
6	603	18.45591	-67.1458	Aguadilla	PR	Aguadilla
7	604	18.49352	-67.1359	Aguadilla	PR	Aguadilla

32992	75204	32.80003	-96.7812	Dallas	TX	Dallas
32993	75205	32.82603	-96.7843	Dallas	TX	Dallas
32994	75206	32.83388	-96.7715	Dallas	TX	Dallas
32995	75207	32.76313	-96.7984	Dallas	TX	Dallas
32996	75208	32.73634	-96.823	Dallas	TX	Dallas
32997	75209	32.76727	-96.7776	Dallas	TX	Dallas
32998	75210	32.76948	-96.7505	Dallas	TX	Dallas
32999	75211	32.73648	-96.835	Dallas	TX	Dallas
33000	75212	32.76727	-96.7776	Dallas	TX	Dallas
33001	75214	32.84125	-96.6984	Dallas	TX	Dallas

zipcodes.txt

10463, 06520, 00603, 75225, 90210, 99999

mysteryLatLong.txt

40.8276, -73.92614, 41.94887, -87.65778, 32.7582813, -97.1105791

Part A. Do Mongo Indexes Matter

Comparing performance with and without index

No Index Query

- use **mongoimport** to create and populate a collection called zipstates based on **zipcodes.states.gps.csv**
- Use Python and pymongo to query the zipstates collection to find the document with zipcode = 10463 and display the city and state for zipcode = 10463
- Determine the time and number of documents searched in your query by adding **.explain("executionStats")** to the find command and programmatically extract the number of records(*totalDocsExamined*) and time it took (*executionTimeMillis*) from the returned json.
- Note: there are some annoying differences when using the mongo shell vs. pymonogo. For example:
 - Mongo Shell: `db.mycollection.find(query).explain("executionStats")`
 - pymongo: `db.mycollection.find(query).explain().['executionStats']`
- Write code to extract the relevant data from the json/bson document returned from **explain()**

Index Query

Use either the mongo shell or pymongo to create an index for the zipcode field

- repeat the find command used above, and extract the relevant data from the json/bson document returned from **explain()**
- **Create a graph** comparing the time and number of records for no index and with index
- Be sure to annotate your results graph or graphs for readability and understandability.

Part B. Find Location of Zip Codes

The file **zipcodes.txt** contains a list of zip codes. Write a python program that connects to mongo and finds the city and states for each of the zip codes in the list.

Display a table of the form:

Zip	City	State
10463	Bronx	NY
75225	Dallas	TX
06520	New Haven	CT
99999	Not Valid	Zip

If the zip code is not a valid zipcode, display 'Not Valid Zip'

Challenge: The zip codes in **zipcodes.txt** (that you are searching for) contain leading zeros in the zip code (e.g. 06205). The CSV file **zipcodes.states.gps.csv** used to build mongo does not contain leading zeros for the zip codes. Neither Mongo or Python allow leading zeros in integers.

You are not allowed to modify the csv file or zipcodes.txt. However, you can do whatever else you like to resolve this problem. Resolve the problem as elegantly as you can. Your output should include leading zeros in zip codes (as shown above)

Part C. Find nearest zip code to mystery GPS coordinates

The file **mysteryLatLong.txt** contains 3 lat, long GPS coordinates

Rollo wants to find the closest zip code for each of the zipcodes in **mysterylocs.csv**.

- Use Mongo to find the closest zipcode, city and state for each of the mystery gps locations.
- Display the results as a table:

GPS-Coordinates City State zipcode

Here is a Python function that computes the distance in kilometers between two lat long points.

```
from math import radians, cos, sin, asin, sqrt
def haversine(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    # Radius of earth in kilometers is 6371
    km = 6371 * c
    return km
```

What do these mystery zip codes have in common?

Submit:

PDF with

- Name
- ID
- Quest Number and Name
- Part A.
 - Understandable graph of results
 - Description of results
- Part B.
 - Zip City State table
 - Explanation of how you solved the challenge with related code 'snippets' – not entire code
- Part C.
 - Table: GPS-Coordinates City State zipcode
 - What the locations have in common

ZIP File with related code