Fall 2018 Comp 430/533
Assignment 6
Due: Date of course final.

A6: NoSQL

The goal of this assignment is to explore using a NoSQL database.

Grading

This assignment is worth 100 points.

Be sure to include both your code and your output in your submission.

Academic Honesty

The following level of collaboration is allowed on this assignment:

You may discuss the assignment with your classmates at a high level. Any issues getting MongoDB running is totally fine. What is not allowed is direct examination of anyone else's NoSQL code (on a computer, email, whiteboard, etc.) or allowing anyone else to see your NoSQL code. You may also use (and in fact are encouraged to use) the MongoDB reference manual

`https://docs.mongodb.com/manual/`

You may use the search engine of your choice to look up the syntax for MongoDB commands, but may not use it to find answers.

You MAY post and discuss results with your classmates.

Loading the data

1. Start your mongodb server

```
mongod -auth -port 27017 -dbpath /data/db
```

2. Load the data files into MongoDB using the mongodbImport program

   (a) `mongoimport --db ricedb --collection foodInfo --jsonArray --file foodInfo.json --username ricedb --password myPwd`
   2012 documents.

   (b) `mongoimport --db ricedb --collection truckEvent --jsonArray --file truckEvent.json --username ricedb --password myPwd`
   63 documents.

   (c) `mongoimport --db ricedb --collection sale --jsonArray --file sale.json --username ricedb --password myPwd`
   6807 documents.

   (d) `mongoimport --db ricedb --collection ingredient --jsonArray --file ingredient.json --username ricedb --password myPwd`
   78 documents.

   (e) `mongoimport --db ricedb --collection unit --jsonArray --file unit.json --username ricedb --password noSqlIsBest`
   86 documents.

   If you need to reload the data, empty the collections by running the following commands in the mongodb client

   ```
   db.foodInfo.drop()
   db.truckEvent.drop()
   db.sale.drop()
   db.ingredient.drop()
   db.unit.drop()
   ```

1 Queries

Be sure that you are using the ricedb database!

```
use ricedb
```

1. (2 points) Write a query that returns a single document from the foodInfo collection.

2. (4 points) Write a query that pretty prints (using the "pretty()" function) all documents with portion_display_name = "regular Oreo".

3. (4 points) Write a query that pretty prints all the documents whose display_name contains “ice cream” (case insensitive) and that have less than 200 calories.

4. (4 points) Write a query that pretty prints the 5 truckEvent documents that describe events that started closest in time to, but strictly before, September 7, 2017. Order the events from closest time to furthest away.

   In other words, if we have:
   Day1 Day2 Day3 Day4 Day5 Day6 Day7 Day8 Day9 Day10
   and we ask for the 3 days before Day7, you would return:
   Day6
   Day5
   Day4

5. (4 points) Write a query that returns the total number of products sold during the truckEvent with eventId 16.

6. (8 points, 2 points each sub part) There are 4 parts to this question.

   (a) Change the name of a truckEvent document where the eventName is “Pi Day” to “Pie Day”. Include your query here.

   (b) Write a query that pretty prints all truckEvent documents where the eventName is “Pi Day” OR “Pie Day”.

   (c) Change the eventName of all truckEvent documents where the eventName is “GSA Coffee Break” to “GSA Study Break”.
   How many documents were updated? (Paste your MongdoDB command and the output of your MongoDB command)

   (d) Write a query that returns the number of truckEvent documents with eventName “GSA Study Break” OR “GSA Coffee Break”. Include the number of documents found.

7. We want to learn about the calories in each of the different toppings. To help us figure this out, we have two new collections: foodInfo and unit. The foodInfo collection contains nutritional information about commonly consumed items. The unit collection maps different portion names to common unit measures (e.g. cup to ounces).

   We want to match up toppings in our ingredient collection with toppings in the foodInfo collection. However, there are some challenges. Since there are multiple foodInfo documents with the same display_name, we need to disambiguate. One approach is to create a new collection that only has documents where category = "topping" from the foodInfo collection.

   (a) (6 points) Using an aggregate function, create a new collection, called "foodInfoToppings" that contains only the display_name, portion_display_name, and calories fields from foodInfo. Include every document where category = "topping". Include your aggregation code in your submission as well as a query to obtain the results.

   (b) (15 points) Now, we want to know which toppings have the highest calories per ounce. Using the aggregation function that we went over in class, filter the ingredient collection to only contain documents where category = "topping" and then "join" these results to the foodInfoToppings collection. Note that the ingName field in ingredient will match the display_name field in foodInfoToppings.

       Next, "join" the resulting dataset to the unit collection to get the number of ounces for each topping. Compute the calories per ounce (use the $divide aggregation function to do the division) and extract the following fields:

       - ingId
       - ingName
       - portion_display_name
       - calories
       - ounces
       - calPerOunce (computed)

       Sort by calPerOunce, descending, and pretty print the three toppings with the highest calorie per ounce.

2 Streaming Twitter Data

As we discussed in class, Document stores are designed to handle streaming data in formats that can vary. For the second part of this assignment, you will collect some twitter data about ice cream and / or food trucks, store it in Mongodb and then query the data.

We will use the tweepy Python library to collect tweets (`http://www.tweepy.org/`).

1. Create a twitter account, if you don't already have one. Go to `https://developer.twitter.com/content/developer-twitter/en.html` and create an account.
2. Go to `https://apps.twitter.com/` and log in
3. Click on "Create New App"
4. Give your app a name, description, and provide a website (you can use the course piazza site)
5. Agree to the terms and conditions
6. Go to the "Keys and Access Tokens" tab
7. Copy your Consumer Key (API) and Consumer Secret (API Secret) (DO NOT SHARE THESE!!)
8. Click on "Create my access token"
9. Copy your Access Token and Access Token Secret (DO NOT SHARE THESE!!)
10. The file getTweets.py contains skeleton code to filter tweets based on keywords. Download this file.
11. Update the file to include your Consumer Keys and Access Tokens
12. Fill in the missing code to add a field with the retrieval date and time and to store the tweet in the collection.

    SUBMIT ONLY THE CODE YOU ENTERED where it says "YOUR CODE HERE"

13. Run the file and look at the content of the tweets

    You can do this with the command

    `python getIceCreamTweets.py`

    or

    `python3 getIceCreamTweets.py`

14. The program will continue to run until you kill it using Ctrl-C

Once you have your code running and have collected a number of tweets, write code to query the mongodb database.

Your code must generate an exact answer to the question provided. No human interpretation or intervention. You may write a script that produces the result, but you cannot, for example, write code that produces a document and then manually take a value from the document and plug it into more code.

1. (2 points) Add a field called "netId" with the value set to your netId to all of the records.
2. (2 points) How many tweets were longer than 140 characters?
3. (4 points) How many of the tweets contain the hashtag '#foodtruck'? Note that the tweet text can be in the text field or in the full_text field of the extended_tweet document.
4. (4 points) What are the top 5, non-whitespace only, user locations (exclude locations which contain only whitespace or tabs)? Display the location value and the counts. Subsort by location, if there are ties. You might find this webpage helpful: `https://en.wikipedia.org/wiki/Regular_expression` for determining a regular expressrion for Space and tab.
5. (2 points) How many of the tweets were retweeted?
6. (4 points) What how many followers does the tweeter with the most followers have?

7. (4 points) Use mongoexport to export your tweets to a json file. Name the file icecream<yourNetIdHere>.json. List your export statement here.

   A skeleton is:

   ```
   mongoexport --db ricedb --collection icecream
   --out icecream<yourNetIdHere>.json --username ricedb
   --password "myPwd"
   ```

3 Short Answers

1. (2 points) List 1 advantage and 1 disadvantage of embedding documents within other documents
2. (2 points) The tweets contain a number of fields with 0 or null values. What is an advantage of including these fields?
3. (2 points) What is a disadvantage?

4 What to submit

Submit a zip file containing:

1. A .js file that contains your mongodb javascript executable code. Basically, we want to be able to cut & paste your code into the Mongodb client and have it run, or run it at the command line. This means that any comments or text answers in this file should be in Mongodb comments. A line is considered a comment if it starts with "//".
2. Your modified getIceCreamTweets.py file (you can take out the access codes)
3. (5 points) A json file containing the tweets you collected
4. A text .txt file with your answers