Lab 2: Starting with MongoDB

Due date: April 17, 10:00am.

Note: Lab 3 will come out on Friday, April 17. Lab 2 submissions will be in grace period for most of April 17, but I highly recommend on-time submission, so that you could start dealing with Lab 3, which will be much bigger.

Lab Assignment

Assignment Preparation

This is an individual lab. I expect every person to complete it without consulting others.

This is a short lab to give you some familiarity with MongoDB's `db.<collection>.find` command. To complete this lab, you will need to create a simple Javascript program, which, largely will consist of your queries and the loops over the cursor printing the results.

Data

You will continue working with the `daily.json` file representing the COVID-19 information in the US. For your work, you can use the `daily.json` file provided to you for Lab 1, but your code shall work properly with any version of this file – the tests may involve me using a fresh version that covers additional days.

Queries

The main objective of this lab is for each of you to get comfortable using MongoDB's `find()` command. To that extent, you will write a few MongoDB queries.

Query preparation and submission. You will submit your deliverables in two separate ways.

1. **Text file.** Create text file `lab02.mongo` and include all your queries and commands there. Each query must be on its own lines, prefaced with a Javascript comment line specifying the query number and with at least one empty line between queries. The header of the file must contain one or more Javascript comment lines with your name and other information about the file. The expected format is something like this:

   ```
   // CSC 369. Lab 2
   // Alex Dekhtyar
   // COVID-19 dataset

   // Query 1
   db.covid.find(...)

   // Query 2
   db.covid.find(...)

   ...

   //end of queries
   ```

2. **Log File.** Using Linux's `script` command record a complete `mongoDBV` session in which you (see below for specific instructions for each step):

   - start the mongoDB session, and authenticate against the `csc369users` database.
   - switch over to your database (database carrying the name of your user account), and load the contents of the `daily.json` file into a collection called `covid` (showing first that prior to loading the collection is empty).
   - execute all `db.covid.find()` queries from this lab.
   - execute all data alteration commands from this lab, following the instructions below.
   - exit the `mongoDB` session.

Queries

Write MongoDB `db.<collection>.find()` queries that produce answers to each of the questions below. Each question must be answered with exactly one `find()` command. Each query below specifies both what you should find, **and** what information shall be displayed to the user. Only explicitly asked for information shall be displayed (the exceptions to this rule are any queries where you are explicitly asked to return entire documents).

Note: This assignment does not involve work with JSON arrays and embedded objects. You will have ample opportunity to practice those in Lab 3.

1. Report the total number of positive cases and the number of new cases in New York state on April 1.
2. Report the total number of documents in the collection in which the number of of new cases exceeds 1,000.
3. Report one full document from the database with the largest number of COVID-19-related deaths in a state **on that day**[1].
4. Report for how many days the state of California is missing data about the number of hospitalized COVID-19 patients.
5. Report the date, how many positive cases and how many deaths were in the state of New Jersey on the earliest day when the total cumulative number of deaths exceeded 500.
6. Report information about the state of COVID-19: number of positive cases, number of negative tests, number of deaths, number of new positive cases for all New England states (CT, MA, RI, NH, VT, ME) on April 1 ordered in descending order by the total cumulative number of positive cases.

Data Manipulation

For this part of the assignment, you will create four additional collections. You will upload the `daily.json` data to each collection, and then will write commands performing some data manipulation. You will add each task with `db.collection.find()` or `db.collection.find().pretty()` command to display all data stored in the collection.

1. **Collection** `california`. After uploading the data into this collection, write a sequence of MongoDB commands the leaves in the collection only the data for the state of California for the months of April.
2. **Collection** `oregon`. After uploading the data into this collection, write a sequence of MongoDB commands that keeps in the collection only the information about the days when Oregon had more than 80 new cases. Output all objects that remain in the collection. Update the object for April 2 to include only the date, name of state, number of positive cases, number of deaths, and the number of new positive cases for the day, as well as a new attribute `situation` whose value is set to `"under control"`. Report the contents of the collection again.
3. **Collection** `keydays`. After uploading the data into this collection write a sequence of MongoDB commands that keeps in the collection only the documents about the days in which a state had between 100 and 120 (inclusively) new positive cases, and between 5 and 8 new deaths.

[1]The method you will be using for this does not guarantee retrieval of ALL such documents - only one. Hence the formulation ”report one full document. . . ”.

4. **hospitalization**. After uploading the data into this collection write a sequence of MongoDB commands that keeps in the collection only the documents that contain information about all of the following: currently hospitalized patients, total number of patients hospitalized in the state to day with COVID-19, total number of patients on ventilator, current number of patients on ventilator. Output the total number of records left in the collection. Find the record with the largest number of current patients on ventilator. Remove all other records from the collection. Replace the remaining record in the collection with a record that contains only the name of the state, the date, and the number of people currently on the ventilator (you can look up those values, of course).

Submission

Submit the following files:

- Your `lab02.mongo` file containing all commands for each query and each manipulation exercise.
- A file called `session.out` containing the results of you running all exercises above in a MongoDB session. Use `script` command to record this session.
- Any files you use as part of your recorded interaction with MongoDB.
- `README` file.

All submitted files must contain your name on them.

Submit all your code in a single archive (zip or tar.gz).

NOTE: I will attempt to install `handin` on `ambari-head`. If successful, you will submit on `ambari-head`, otherwise - you will submit on `unixN` servers. Either way, you will submit using the following command:

Use `handin` to submit as follows:

```
$ handin dekhtyar lab02 <FILES>
```

Good Luck!