UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Faculty of Engineering, Built Environment and Information Technology

School of Information Technology
Department of Computer Science

COS326 Database Systems:
Practical 8 2022

Release Date: 11 October 2022
Submission Date: 16 October 2022
Lecturer: Mr S.M Makura

Total: 50 Marks

Objectives

1. Get exposure to the MongoDB document DBMS.
2. Learn how to create and use documents, collections and JavaScript functions for a MongoDB database.
3. Learn how to develop Java applications that access data in a MongoDB database.
4. Appreciate the differences between SQL and NoSQL databases.

For this practical exercise you will use the MongoDB database. When you are done: You must submit the following files:

1. You must submit files, named:
    a. ***.mongorc.js*** with all the JavaScript functions for Task 1 and Task 2.
    b. Compress the above documents into an archive and upload it to ClickUP **before** the due date/time. The file name for the archive must have your student number as part of the file name,
    e.g. **uxxxxxxxx-prac8.zip** or **uxxxxxxxx-prac8.tar.gz** where **xxxxxxxx** is your student number.

2. Late submissions will not be accepted.

Task 1: INSERT, UPDATE AND FIND QUERIES **[20 marks]**

Create the JavaScript functions given in the table below and store them in the file. *mongorc.js* in your home folder.

Mongo shell function name	**Description**
insertPoints(dbName, colName, xcount, ycount)	inserts documents of the form: { *x: xval, y: yval*} where *xval* = 2,4, …, *xcount* and *yval* = 2,4,…,*ycount*, into the collection *colName* in the database *dbName..*
findNearest((dbName, colName, xval, yval)	Find and print the document whose (x, y) Cartesian coordinates represent a point which is nearest to the point (*xval, yval*). Hint: Use Euclidean distance
updateYVals(dbName, colName, threshold, incr)	updates the *yval* values that are greater than *threshold* by adding *incr* to each of these values.
removeIfYless(dbName, colName, threshold)	removes all objects with *yval* values that are less than *threshold*.

Test the above functions in the mongo shell as follows:

1. Create a database called *prac8db,* a collection called *Cartesian* and a composite index on the *xval and yval* values.

[3 marks]

2. Use the function *insertPoints(dbName, colName, xcount, ycount)* to insert the objects {x*: xval, y: yval*} into the collection *Cartesian* in database *prac8db,* for *xcount* = 5, *ycount* = 5.

[4 marks]

3. Use the function *findNearest((dbName, colName, xval, yval)* to find and print the point which is nearest to coordinates (5, 7).

[6 marks]

4. Use the function *updateYVals(dbName, colName, threshold,incr*) to update the objects { *x: xval, y: yval*} in the collection *Cartesian* in database *prac8db.* Use a *incr* value of 10 and a *threshold* value of 2.

[4 marks]

5. Use the function *removeIfYless(dbName, colName, threshold)* to delete (remove) objects { *x: xval*, *y: yval*} from the collection *Cartesian* in database *prac8db*. Every document with *yval* < 4 should be removed.

[3 marks]

Task 2: FIND AND AGGREGATE QUERIES [30 marks]

Use the MongoDB manual data for states, cities and zip codes. A description of this data is given in the MongoDB manual. Here is a direct link:
https://www.mongodb.com/docs/manual/tutorial/aggregation-zip-code-data-set/

a. Use the *mongoimport* tool to import the data file *zipcodes.json* (available on ClickUP) into the collection called *zipcodes* in your *prac8db* database and then

b. Create the JavaScript functions given in the table below and store them in the same file (*.mongorc.js*) as for Task 1. For each function, you need to assign the query result to a cursor variable and then use a while loop to print (printjson) the objects in the query result.

Mongo shell function name	**Description**
1. allStatesPopulation(dbName, colName)	Shows the state name and population for all states as a list of {*state*, *population*} objects sorted in alphabetical (ascending) order of state name. You should use an aggregation pipeline.
2. oneStatePopulation(dbName, colName, stateName)	Shows the state name and population for the given *stateName* as a {*state*, *population*} object. You should use an aggregation pipeline.
3. allStatesPopulationMR(dbName, colName)	Shows the state name and population for all states as a list of {*state*, *population*} objects. You should use a **map-reduce** computation for the aggregation. The function should store the results in a collection called *states_population.*
4. placesInGrid(dbName, colName, lat1,lat2,lon1,lon2)	Shows the zip code, city name, location (loc) as [*latitude*, *longitude*] and state for the places located between latitudes *lat1* and *lat2* and between longitudes *lon1* and *lon2*.

Test the functions in the mongo shell as follows:

1. Use the *allStatesPopulation(dbName, colName)* function to show the state name and population for all states as a list of {*state, population*} pairs sorted in alphabetical (ascending) order of state name.

[8 marks]

2. Use the *oneStatePopulation(dbName, colName, stateName)* function to show the state name and population for one state of your choice as a{*state, population*} pair.

[8 marks]

3. Use the *allStatesPopulationMR(dbName, colName)* function to compute and show the total population of each state.

[7 marks]

4. Use the *placesInGrid(dbName, colName, lat1,lat2,lon1,lon2)* function to show the zip code, city name, location (loc) as [*latitude, longitude*] and state for the places located between latitudes *lat1,lat2* and longitudes *lon1* and *lon2* of your choice.

[7 marks]