DBS311\_

Lab 8b –

**MongoDB – Create/Delete Database/Collection/Documents)**

**Objective**

***Upload screenshots of successful run with outputs in a single document in blackboard within the deadline. Absolutely no email submission will be accepted. Capture screenshots of your code as well. Make sure the timestamp is visible in your screenshots.***

In this Lab, you learn to create and remove MongoDB

Databases

Collections

Documents

**This is a tutorial-based lab. You are required to run and capture output, specially the commands in the rectangular boxes of this file.**

**Getting Started with MongoDB**

Open your Windows command prompt and go the following directory where MongoDB is installed:

cd C:\Program Files\MongoDB\Server\4.2\**bin**

To run MongoDB, execute ***mongod***

mongod

That will have several lines of output

When MongoDB starts successfully, open another Windows command prompt and go the same *bin* directory:

cd C:\Program Files\MongoDB\Server\4.2\**bin**

and execute ***mongo***

mongo

This opens the command line shell where you can work with MongoDB.

**Or you can create a batch file:**

Create a new text file named *start\_mongodb.bat* using a text editor and save the following command in it:

start call "C:\Program Files\MongoDB\Server\4.2\bin\mongod.exe"

start call "C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe"

Every time you want to run MongoDB and open the client shell, execute this file.

To test your solutions before you submit, test them individually and make sure they work correctly.

You submit this file with answers (in the provided spaces).

Name the file “L07\_ID#\_LASTNAME.docx”.

**Tasks**

In this question you create a new database named *seneca* and a collection *student*. We store student data in this collection.

To create the database

use Seneca

you will see the following 🡺 cswitched to db seneca

This command makes “seneca” your current database. However, the database will not be created until you insert a document into this database.

**INSERT a document**

db.collection\_name.insertOne({})

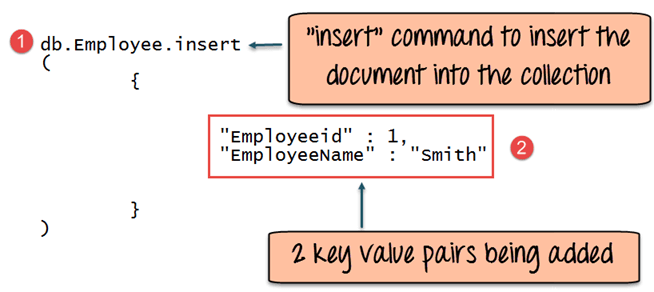
Supply the collection name as student

Insert a new document into your collection *student* with the following data:

first\_name: Sarah  
last\_name: Stone  
email: [s\_stone@email.com](mailto:s_stone@email.com)  
city: Toronto  
status: full-time  
gpa: 3.4  
program: CPA

Collection names

Here is another example

[](https://www.guru99.com/images/MongoDB/112115_0607_Introductio5.png)

watch the quotes and the commas

|  |
| --- |
| > db.student.insertOne ({  ... ... ... "first\_name" : "Sarah",  ... ... ... "last\_name" : "Stone",  Watch the name of what was entered this semester is different  ... ... ... "email" : "stone@email" ,  ... ... ... "status" : "full-time",  ... "gpa": "3.4" ,  ... "program" : "CPA"  ... }  ... )  **The output is as follows if successful:**  {  "acknowledged" : true,  "insertedId" : ObjectId("5f14533e055a850fa57b8d97")  } |

**FIND COMMAND**

Write a command to check if the document has been created successfully.

You use *find()* method to search and fetch documents.

See the following example:

db.student.find()

It will not show well with lots of documents

|  |
| --- |
| db.student.find()  { "\_id" : ObjectId("5f14533e055a850fa57b8d97"), "first\_name" : "Sarah", "last\_name" : "Stone", "email" : "stone@email", "status" : "full-time", "gpa" : "3.4", "program" : "CPA" }  > |

To see the result in *JSON* format, you can run the following statement:

db.student.find().forEach(printjson)

|  |
| --- |
| db.student.find().forEach(printjson)  {  "\_id" : ObjectId("5f14533e055a850fa57b8d97"),  "first\_name" : "Sarah",  "last\_name" : "Stone",  "email" : "stone@email",  "status" : "full-time",  "gpa" : "3.4",  "program" : "CPA"  } |

How many fields are in your document? \_\_\_6\_\_

Is there any new field added to your document? \_\_\_1\_\_\_

If yes, what is the name of the field? \_\_\_\_\_ \_ID

Write a command to remove the document that you created in Question 1.

(We have only one document at this time, but when removing documents make sure you will not remove some other documents if not needed. So, make sure your command will remove “Sarah Stone” from your collection. For now, we assume that we do not have duplicate names in our database.)

**Note:** To avoid making mistakes, you can first write a find command with the proper criteria to see if the required document is fetched. Then, you can use the same criteria in your delete statement.

(Write the statement to remove “Sarah Stone” from the database in the box below.)

ASIDE: Helpful hint:

Consider the mycol collection has the following data.

{\_id : ObjectId("507f191e810c19729de860e1"), title: "MongoDB Overview"},

{\_id : ObjectId("507f191e810c19729de860e2"), title: "NoSQL Overview"},

{\_id : ObjectId("507f191e810c19729de860e3"), title: "Tutorials Point Overview"}

Following example will remove all the documents whose title is 'MongoDB Overview'.

>db.mycol.remove({'title':'MongoDB Overview'})

WriteResult({"nRemoved" : 1})

> db.mycol.find()

{"\_id" : ObjectId("507f191e810c19729de860e2"), "title" : "NoSQL Overview" }

{"\_id" : ObjectId("507f191e810c19729de860e3"), "title" : "Tutorials Point Overview" }

Now remove Sarah

|  |
| --- |
| db.student.remove({"first\_name": "Sarah"})  WriteResult({ "nRemoved" : 1 }) 🡸 and the result is |

What is the message as a result of your delete statement? Copy the message in the following box:

|  |
| --- |
| WriteResult({ "nRemoved" : 1 }) |

To see if the document is removed successfully, write a search statement to see if the document exists.

(We look for one document not all).

|  |
| --- |
| I have an additional record so I can look for one only this way  db.student.find({"firsttest": "sarah"})  { "\_id" : ObjectId("5f14509691d897335b630bc9"), "firsttest" : "sarah" }  Doing the following  db.student.find({"first\_name": "Sarah"}) 🡸 nothing happens |

We want to add some students to our collection, but this time, we define the value for the *\_id* field.

(If the \_id is not defined in your document, it will be added automatically.)

To add these students, we want to store these documents into a variable first. The variable will be an array

EXAMPLE: of variable array followed by inserting the array. The example does not manually assign \_id

var myEmployee=

[{

"Employeeid" : 1,

"EmployeeName" : "Smith"

},

{

"Employeeid" : 2,

"EmployeeName" : "Mohan"

},

{

"Employeeid" : 3,

"EmployeeName" : "Joe"

},

];

db.Employee.insert(myEmployee);

\_id: 1001

first\_name: Sarah  
last\_name: Stone  
email: [s\_stone@email.com](mailto:s_stone@email.com)  
city: Toronto  
status: full-time  
gpa: 3.4  
program: CPA

\_id: 1002

first\_name: Jack  
last\_name: Adam  
email: [j\_adam@email.com](mailto:j_adam@email.com)  
city: North York  
status: part-time  
gpa: 3.6  
program: CPA

Define a variable named *starray* and add these two document to the variable.

(You are storing more than one document, so you need to define an array.

Using a sample:

|  |
| --- |
| var myEmployee=  ... [  ... {  ... "Employeeid" : 1,  ... "EmployeeName" : "Smith"  ... },  ... {  ... "Employeeid" : 2,  ... "EmployeeName" : "Mohan"  ... },  ... {  ... "Employeeid" : 3,  ... "EmployeeName" : "Joe"  ... },  ... ];  > |

Now, use the *myEmployees* array to insert the documents to your collection *student*. Write your insert statement in the box bellow.

|  |
| --- |
| > db.Employee.insert(myEmployee); |

What message is displayed after you execute the insert statement. Copy the message in the following box:

|  |
| --- |
| BulkWriteResult({  "writeErrors" : [ ],  "writeConcernErrors" : [ ],  "nInserted" : 3, ….. inserted 3  "nUpserted" : 0,  "nMatched" : 0,  "nModified" : 0,  "nRemoved" : 0,  "upserted" : [ ]  }) |

Redo with starray

var starray =

[{

"\_id": 1001,

"first\_name": "Sarah" ,

"last\_name" : "Stone",

"email" : "s\_stone@email.com" ,

"city" : "Toronto" ,

Watch the comma

"status" : "full-time" ,

"gpa" : "3.4",

"program" : "CPA"

},

{

"\_id": 1002,

"first\_name" : "Jack" ,

"last\_name" : "Adam" ,

"email" : "j\_adam@email.com",

"city" : "North York",

"status" : "part-time",

"gpa" : "3.6",

"program" : "CPA"

}];

db.student.insert(starray);

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 2,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

Write a statement that shows all documents inserted in your collection *student*:

|  |
| --- |
| db.student.find().forEach(printjson);  NOTE: It is forEach --- watch the case |

Remove all documents

|  |
| --- |
| db.student.remove ({});  WriteResult({ "nRemoved" : 3 }) |

Write a statement to drop the database *seneca*.

CAUTION: Before dropping a database, make sure your current database is the one you want to delete.

For this question, we want to delete (drop) the *seneca* database.

OR you can see what databases you have

show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

seneca 0.000GB

test 0.000GB

You can write the *use* statement before removing the database to make sure your current database is *seneca*.

use seneca

Or you can write the following statement to see the name of your current database:

db

db.getName()

I

If your current database is not *seneca*, write the use statement to switch to *seneca* and then delete the database.

What message is displayed after you execute the drop statement? Copy the message in the box below:

|  |
| --- |
| db.dropDatabase();  { "dropped" : "seneca", "ok" : 1 } |