

## Lab 7 – Week 10

### (MongoDB – Create/Delete Database/Collection/Documents)

#### Objective

In this Lab, you learn to create and remove MongoDB

- Databases
- Collections
- Documents

#### Getting Started

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`

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.pdf”.

## Tasks

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

➤ use *seneca*

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

```
db.collection_name.insertOne({})
```

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

Command used for the insertion:

```
db.student.insertOne({"first_name": "Sarah", "last_name": "Stone", "email":  
"s_stone@email.com", "city": "Toronto", "status": "full-time", "gpa": 3.4, "program":  
"CPA"})
```

2. 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()`

```
{ "_id" : 3, "name" : "Bao Ziglar", "scores" : [ { "score" :  
71.64343899778332, "type" : "exam" }, { "score" : 24.80221293650313,  
"type" : "quiz" }, { "score" : 42.26147058804812, "type" : "homework" } ]  
}  
{ "_id" : 6, "name" : "Jenette Flanders", "scores" : [ { "score" :  
37.32285459166097, "type" : "exam" }, { "score" : 28.32634976913737,  
"type" : "quiz" }, { "score" : 81.57115318686338, "type" : "homework" } ]  
}  
{ "_id" : 0, "name" : "aimee Zank", "scores" : [ { "score" :  
1.463179736705023, "type" : "exam" }, { "score" : 11.78273309957772,  
"type" : "quiz" }, { "score" : 35.8740349954354, "type" : "homework" } ]  
}  
{ "_id" : 8, "name" : "Daphne Zheng", "scores" : [ { "score" :  
22.13583712862635, "type" : "exam" }, { "score" : 14.63969941335069,  
"type" : "quiz" }, { "score" : 75.94123677556644, "type" : "homework" } ]  
}
```

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

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

```
{  
  "_id" : 3,  
  "name" : "Bao Ziglar",  
  "scores" : [  
    {  
      "score" : 71.64343899778332,  
      "type" : "exam"  
    },  
    {  
      "score" : 24.80221293650313,  
      "type" : "quiz"  
    },  
    {  
      "score" : 42.26147058804812,  
      "type" : "homework"  
    }  
  ]  
}  
{  
  "_id" : 6,  
  "name" : "Jenette Flanders",  
  "scores" : [  
    {  
      "score" : 37.32285459166097,  
      "type" : "exam"  
    },  
    {  
      "score" : 28.32634976913737,  
      "type" : "quiz"  
    },  
    {  
      "score" : 81.57115318686338,  
      "type" : "homework"  
    }  
  ]  
}
```

```

        "score" : 37.32285459166097,
        "type" : "exam"
      },
      {
        "score" : 28.32634976913737,
        "type" : "quiz"
      },
      {
        "score" : 81.57115318686338,
        "type" : "homework"
      }
    ]
  }

```

How many fields are in your document? There are total 8 fields in the document that I inserted

Is there any new field added to your document? Yes, there is a new field in the document that is added by itself

If yes, what is the name of the field? The name of the field that is added by itself is \_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.)

At first, I ran command of searching. That is,

```
- db.student.find({"first_name" : "Sarah", "last_name" : "Stone"})
```

Later, after this I ran the command to remove the document from the database. The command is,

```
- db.student.deleteOne({"first_name" : "Sarah", "last_name" : "Stone"})
```

Here, I have used deleteOne as there was just one document in the database to be removed.

Moreover, the removing of document can also be done in another way too. The other way of doing it is,

```
- db.student.remove({"first_name" : "Sarah", "last_name" : "Stone"})
```

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

The result that I got after removing of the document using the delete command is,  
- { "acknowledged" : true, "deletedCount" : 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).

For checking whether the document is removed or not, I had again used the find command and also as there was only one document added so I had used only find, that is:

- db.student.find({"first\_name" : "Sarah", "last\_name" : "Stone"})

But, if there were plenty of documents and out of all of those documents only if one document is removed and if for checking that single document that it is removed or not out of all those documents findOne can also be used. I had also used findOne, that is:

- db.student.findOne({"first\_name" : "Sarah", "last\_name" : "Stone"})

Therefore, I used two commands to check they are:

- db.student.find({"first\_name" : "Sarah", "last\_name" : "Stone"})

- db.student.findOne({"first\_name" : "Sarah", "last\_name" : "Stone"})

4. 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.)

*\_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

To add these students, we want to store these documents into a variable first. 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.

```
starray = [{"_id" : 1001, "first_name" : "Sarah", "last_name" : "Stone", "email" :  
"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", "city" : "North York", "status" : "part-time", "gpa" : 3.6, "program"  
: "CPA"}]
```

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

Insert Statement:  
Db.student.insertMany(starray)

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

```
{ "acknowledged" : true, "insertedIds" : [ 1001, 1002 ] }
```

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

```
db.student.find({})
```

5. Write a statement to remove all documents in the *student* collection.

For removing all the documents altogether from the student collection:  
`db.student.deleteMany({ })`

6. Write a statement to drop the database *seneca*.

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.

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 *db* or `db.getName()` statement to see the name of your current database:

➤ `db`

➤ `db.getName()`

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

`db.dropDatabase()`

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

The result that I got after dropping the database is as follows:  
{ "dropped" : "seneca", "ok" : 1 }

