# Department of Computing

**CS-213: Advanced Programming**

**Name:** M. Hasnain Naeem

**Reg #:** 212728

**Class:** BSCS 7B

# Lab 06: Node.js MongoDB

**Date: 3rd October, 2019**

**Time: 10:00-01:00pm & 02:00-05:00pm**

# Instructor: Dr. Sidra Sultana

**Lab Engineer: Ms. Ayesha Asif**

# 

# Lab 06: Node.js MongoDB

**Introduction**

Node.js can be used in database applications. One of the most popular NoSQL database is MongoDB.

**Objectives**

This lab will get you familiar with the node.js MongoDB environment.

**Tools/Software Requirement**

Node.js, Notepad

**Description**

To be able to experiment with the code examples, you will need access to a MongoDB database.

You can download a free MongoDB database at [https://www.mongodb.com](https://www.mongodb.com/)**Install MongoDB Driver**

Let us try to access a MongoDB database with Node.js.

To download and install the official MongoDB driver, open the Command Terminal and execute the following:

Download and install mongodb package:

C:\Users\*Your Name*>npm install mongodb

Now you have downloaded and installed a mongodb database driver.

Node.js can use this module to manipulate MongoDB databases:

var mongo = require('mongodb');

**Helping Material**

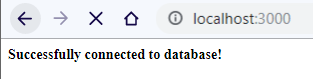
Slides of Lecture 7

<https://www.w3schools.com/nodejs/nodejs_mongodb.asp>

**Lab Tasks**

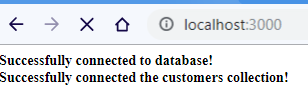
**Task 1:** Create a database named "mydb". Save the code in a file called "demo\_create\_mongo\_db.js" and run the file.

|  |
| --- |
| *let* http = require('http'); *let* MongoClient = require('mongodb').MongoClient;  http.createServer(*function*(request, response) {  response.writeHead(200, {"Content-Type": "text/html"});  *let* url = "mongodb://localhost:27017/lab5";  MongoClient.connect(url, *function*(err, db) {  *if* (err) *throw* err;  response.write("<b>Successfully connected to database!</b><br />");  db.close();  });  *// response.write("<h3>Results:</h3><pre>" + JSON.stringify(result, undefined, 4) + "</pre>");* }).listen(3000); console.log("server is running at: http://127.0.0.1:3000/"); |



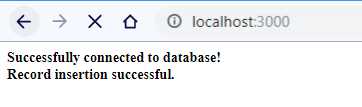
**Task 2:** Create a collection called "customers". Save the code in a file called "demo\_mongodb\_createcollection.js" and run the file.

|  |
| --- |
| *let* http = require('http'); *let* MongoClient = require('mongodb').MongoClient; *const* dbName = "lab6";  http.createServer(*function*(request, response) {  response.writeHead(200, {"Content-Type": "text/html"});  *let* url = "mongodb://localhost:27017";  MongoClient.connect(url, {  useNewUrlParser: *true*,  useUnifiedTopology: *true* }, (err, client) => {  *if* (err) {  console.error(err);  }  response.write("<b>Successfully connected to database!</b><br />");  *let* db = client.db(dbName);  db.createCollection("customers", *function*(err, results){  *if*(err) *throw* err;  response.write("<b>Successfully connected the customers collection!</b><br />");  *for*(*let* i =0; i < results.length; i++)  response.write("<h3>Results:</h3><pre>" + JSON.stringify(results[i], *undefined*, 4) + "</pre>");  });  }) }).listen(3000); console.log("server is running at: http://127.0.0.1:3000/"); |



**Task 3:** Insert a document in the "customers" collection. Save the code in a file called "demo\_mongodb\_insert.js" and run the file.

|  |
| --- |
| db.collection("customers").insertOne(record, *function*(err, results){  *if*(err) *throw* err;  response.write("<b>Record insertion successful.</b><br />");  *for*(*let* i =0; i < results.length; i++)  response.write("<h3>Results:</h3><pre>" + JSON.stringify(results[i], *undefined*, 4) + "</pre>"); }) |



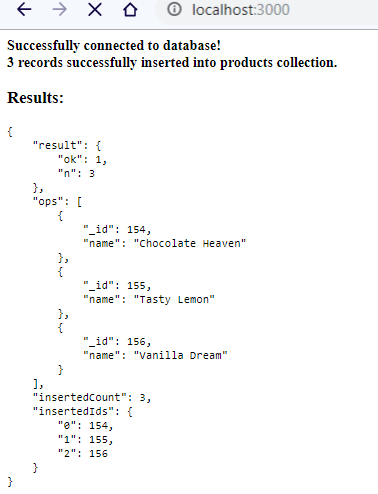
**Task 4:** Insert multiple documents in the "customers" collection. Save the code in a file called "demo\_mongodb\_insert\_multiple.js" and run the file.

|  |
| --- |
| *let* documents = [  { name: 'John', address: 'Highway 71'},  { name: 'Peter', address: 'Lowstreet 4'},  { name: 'Amy', address: 'Apple st 652'},  { name: 'Chuck', address: 'Main Road 989'},  { name: 'Viola', address: 'Sideway 1633'} ]; db.collection("customers").insertMany(documents, *function*(err, results){  *if*(err) *throw* err;  response.write("<b>Record insertion successful.</b><br />");  response.write("<h3>Results:</h3><pre>" + JSON.stringify(results, *undefined*, 4) + "</pre>"); }) |



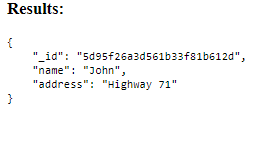
**Task 5:** Insert three records in a "products" table, with specified \_id fields. Save the code in a file called "demo\_mongodb\_insert\_id.js" and run the file.

|  |
| --- |
| *let* documents = [  { \_id: 154, name: 'Chocolate Heaven'},  { \_id: 155, name: 'Tasty Lemon'},  { \_id: 156, name: 'Vanilla Dream'}  ];  db.collection("products").insertMany(documents, *function*(err, results){  *if*(err) *throw* err;  response.write("<b>" + results.insertedCount + " records successfully inserted into products collection.</b><br />");  response.write("<h3>Results:</h3><pre>" + JSON.stringify(results, *undefined*, 4) + "</pre>"); }) |



**Task 6:** Find the first document in the customer’s collection. Save the code in a file called "demo\_mongodb\_findone.js" and run the file.

|  |
| --- |
| *let* db = client.db(dbName);  db.collection("customers").findOne({}, *function*(err, results){  *if*(err) *throw* err;  response.write("<h3>Results:</h3><pre>" + JSON.stringify(results, *undefined*, 4) + "</pre>");  }) }) |



**Task 7:** Return the fields "name" and "address" of all documents in the customers collection.

|  |
| --- |
| db.collection("customers").find({}).toArray(*function*(err, results){  *if*(err) *throw* err;  response.write("<h3>Results</h3>");  *for*(*let* i =0; i < results.length; i++){  response.write("<b>Record # " + i + ":</b><br />");  response.write("Name: " + results[i].name + "<br />");  response.write("Address: " + results[i].address + "<br />");  response.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_<br />");  } }); |

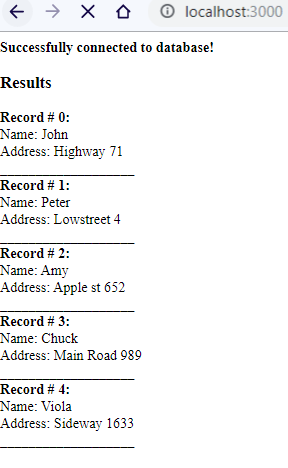
**Alternatively, we can use projection in 2 ways:**

**1st:**

|  |
| --- |
| db.collection("customers").find({},{projection: {\_id: 0, name:1, address:1}}).toArray(*function*(err, results){  *if*(err) *throw* err;  response.write("<h3>Results</h3>");  *for*(*let* i =0; i < results.length; i++){  response.write("<b>Record # " + i + ":</b><br />");  response.write("Name: " + results[i].name + "<br />");  response.write("Address: " + results[i].address + "<br />");  response.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_<br />");  } }); |

**2nd:**

|  |
| --- |
| db.collection("customers").find({},{projection: {name:1, address:1}}).toArray(*function*(err, results){  *if*(err) *throw* err;  response.write("<h3>Results</h3>");  *for*(*let* i =0; i < results.length; i++){  response.write("<b>Record # " + i + ":</b><br />");  response.write("Name: " + results[i].name + "<br />");  response.write("Address: " + results[i].address + "<br />");  response.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_<br />");  } }); |



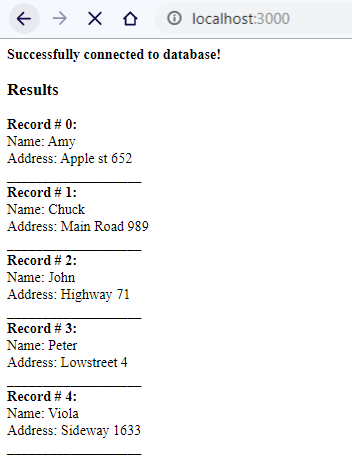
**Task 8:** Find documents with the address "Park Lane 38". Save the code in a file called "demo\_mongodb\_query.js" and run the file.

|  |
| --- |
| *let* query = {address: "Park Lane 38"}; db.collection("customers").find(query).toArray(*function*(err, results){  *if*(err) *throw* err;  response.write("<h3>Results</h3>");  *for*(*let* i =0; i < results.length; i++){  response.write("<b>Record # " + i + ":</b><br />");  response.write("Name: " + results[i].name + "<br />");  response.write("Address: " + results[i].address + "<br />");  response.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_<br />");  } }); |

**No results.**

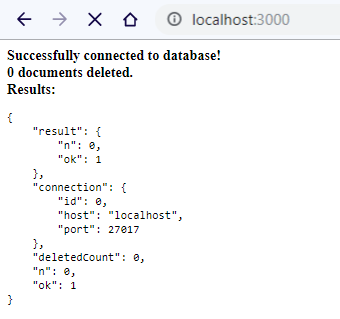
**Task 9:** Sort the result alphabetically by name. Save the code in a file called "demo\_sort.js" and run the file.

|  |
| --- |
| *let* sortStr = {name: 1}; db.collection("customers").find().sort(sortStr).toArray(*function*(err, results){  *if*(err) *throw* err;  response.write("<h3>Results</h3>");  *for*(*let* i =0; i < results.length; i++){  response.write("<b>Record # " + i + ":</b><br />");  response.write("Name: " + results[i].name + "<br />");  response.write("Address: " + results[i].address + "<br />");  response.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_<br />");  } }); |



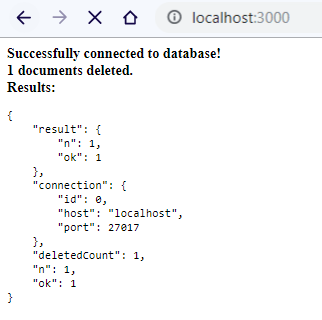
**Task 10:** Delete the document with the address "Mountain 21". Save the code in a file called "demo\_delete.js" and run the file.

|  |
| --- |
| *let* query = {address: "Mountain 21"}; db.collection("customers").deleteMany(query, *function*(err, results){  *if*(err) *throw* err;  response.write("<b>" + results.result.n + " documents deleted.</b><br />");  response.write("<b>Results: </b><pre>" + JSON.stringify(results, *undefined*, 4) + "</pre>");  }); |



**Task 11:** Delete all documents were the address starts with the letter "O". Save the code in a file called "demo\_delete\_many.js" and run the file

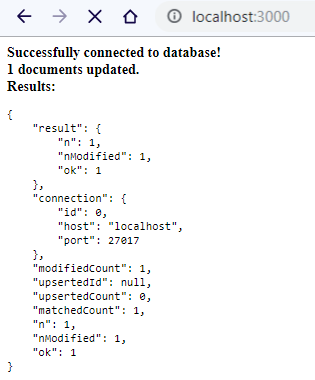
|  |
| --- |
| *let* query = {address: /^o/}; db.collection("customers").deleteMany(query, *function*(err, results){  *if*(err) *throw* err;  response.write("<b>" + results.result.n + " documents deleted.</b><br />");  response.write("<b>Results: </b><pre>" + JSON.stringify(results, *undefined*, 4) + "</pre>");  }); |



**Note:** I added a document starting with “o” to demonstrate the different query.

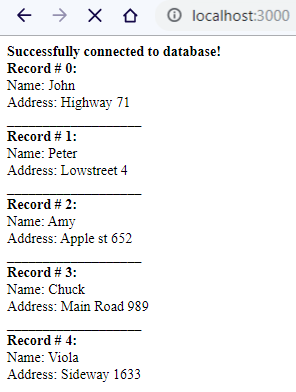
**Task 13:** Update the document with the address "Valley 345" to name="Mickey" and address="Canyon 123". Save the code in a file called "demo\_update\_one.js" and run the file

|  |
| --- |
| *let* query = {address: "Valley 345"}; *let* newValues = { $set: {name: "Mickey", address: "Canyon 123" } }; db.collection("customers").updateOne(query, newValues,*function*(err, results){  *if*(err) *throw* err;  response.write("<b>" + results.result.n + " documents updated.</b><br />");  response.write("<b>Results: </b><pre>" + JSON.stringify(results, *undefined*, 4) + "</pre>");  }); |



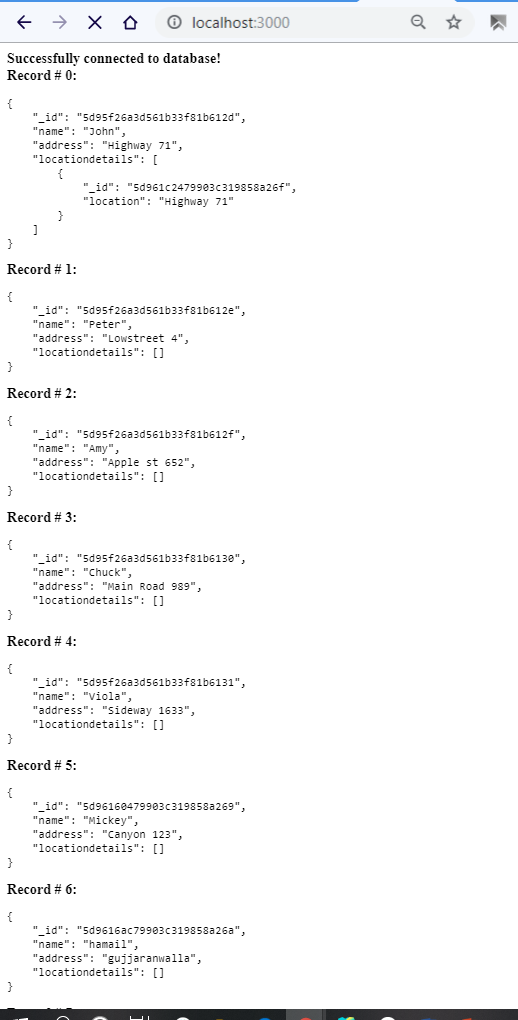
**Task 14:** Consider you have a "customers" collection. Limit the result to only return 5 documents. Save the code above in a file called "demo\_mongodb\_limit.js" and run the file.

|  |
| --- |
| db.collection("customers").find().limit(5).toArray(*function*(err, results){  *if*(err) *throw* err;  *for*(*let* i =0; i < results.length; i++){  response.write("<b>Record # " + i + ":</b><br />");  response.write("Name: " + results[i].name + "<br />");  response.write("Address: " + results[i].address + "<br />");  response.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_<br />");  } }); |



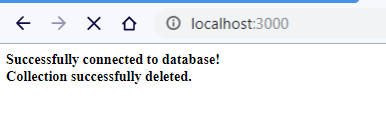
**Task 15:** Practice the Join operations on different tables.

|  |
| --- |
| db.collection("customers").aggregate([  {  $lookup:  {  from: 'locations',  localField: 'address',  foreignField: 'location',  as: 'locationdetails'  }  } ]).toArray(*function*(err, results){  *if*(err) *throw* err;  *for*(*let* i =0; i < results.length; i++){  response.write("<b>Record # " + i + ":</b><pre>" + JSON.stringify(results[i], *undefined*, 4) + "</pre>");  } }); |



**Task 12:** Delete the "customers" table. Save the code in a file called "demo\_drop.js" and run the file.

|  |
| --- |
| *let* db = client.db(dbName); db.dropCollection("customers", *function*(err, isDeleted){  *if*(err) *throw* err;  *if*(isDeleted)  response.write("<b>Collection successfully deleted.</b><br />");  *else* response.write("<b>Collection deletion failed.</b><br />"); }); |



### Deliverables

Compile a single word document by filling in the solution part and submit this Word file on LMS. This lab grading policy is as follows: The lab is graded between 0 to 10 marks. The submitted solution can get a maximum of 5 marks. At the end of each lab or in the next lab, there will be a viva related to the tasks. The viva has a weightage of 5 marks. Insert the solution/answer in this document. You must show the implementation of the tasks in the designing tool, along with your complete Word document to get your work graded. You must also submit this Word document on the LMS. In case of any problems with submissions on LMS, submit your Lab assignments by emailing it to Ms. Ayesha Asif: [ayesha.asif@seecs.edu.pk](mailto:ayesha.asif@seecs.edu.pk).