# Department of Computing

**CS-213: Advanced Programming**

**Name:** M. Hasnain Naeem

**Reg #:** 212728

**Class:** BSCS 7B

# Lab 5: Node.js MySQL

**Date: 3rd October, 2019**

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

# Instructor: Dr. Sidra Sultana

**Lab Engineer: Ms. Ayesha Asif**

# 

# Lab 5: Node.js MySQL

**Introduction**

Node.js can be used in database applications. One of the most popular databases is MySQL.client.

**Objectives**

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

**Tools/Software Requirement**

Node.js, Notepad

**Description**

## **MySQL Database**

To be able to experiment with the code examples, you should have MySQL installed on your computer.

You can download a free MySQL database at <https://www.mysql.com/downloads/>.

**Install MySQL Driver**

Once you have MySQL up and running on your computer, you can access it by using Node.js.

To access a MySQL database with Node.js, you need a MySQL driver. This tutorial will use the "mysql" module, downloaded from NPM.

To download and install the "mysql" module, open the Command Terminal and execute the following:

C:\Users\Your Name>npm install mysql

Now you have downloaded and installed a mysql database driver.

Node.js can use this module to manipulate the MySQL database:

var mysql = require('mysql');

**Helping Material**

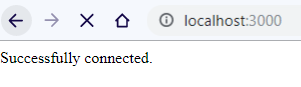
Slides of Lecture 6

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

**Lab Tasks**

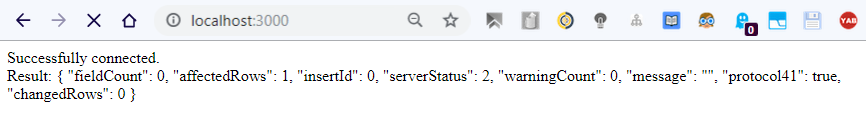
Task 1: Start by creating a connection to the database. Use the username and password from your MySQL database.

|  |
| --- |
| *let* http = require('http'); *let* mysql = require('mysql');  http.createServer(*function*(request, response) {  response.writeHead(200, {"Content-Type": "text/html"});  *let* con = mysql.createConnection({  host: "localhost",  port: "3306",  user: "root",  password: ""  });   con.connect(*function*(err){  *if*(err) *throw* err;  response.write("Successfully connected.<br //>");  }); }).listen(3000); console.log("server is running at: http://127.0.0.1:3000/"); |



**Task 2: Create a database named "aplab5". Save the code in a file called "demo\_create\_db.js" and run the file.**

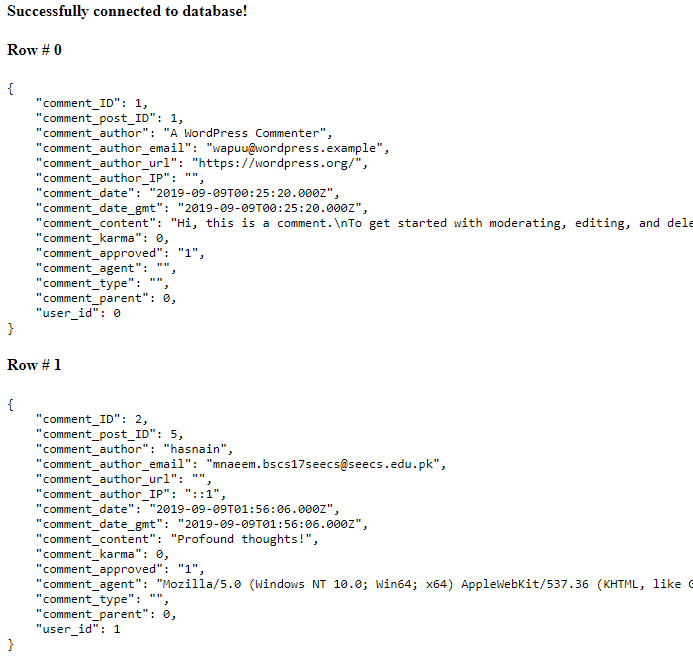
|  |
| --- |
| con.query("CREATE DATABASE IF NOT EXISTS aplab5;", *function* (err, result) {  response.write("Result: " + JSON.stringify(result, *undefined*, 4)); }); |



**Task 3: Use SQL statements to read from (or write to) a MySQL database. The query method takes an sql statements as a parameter and returns the result.**

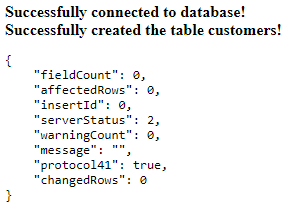
Reading from a Wordpress database table:

|  |
| --- |
| *let* con = mysql.createConnection({  host: "localhost",  port: "3306",  user: "root",  password: "",  database: "wp-1" });  con.connect(*function*(err){  *if*(err) *throw* err;  response.write("<b>Successfully connected to database!</b><br />");  con.query("SELECT \* FROM wp\_comments;", *function* (err, result) {  *if*(err) *throw* err;  *for*(*let* i =0; i < result.length; i++){  response.write("<h4>Row # " + i + "</h4>");  response.write("<pre>" + JSON.stringify(result[i], *undefined*, 4) + "</pre>");  }  }); |



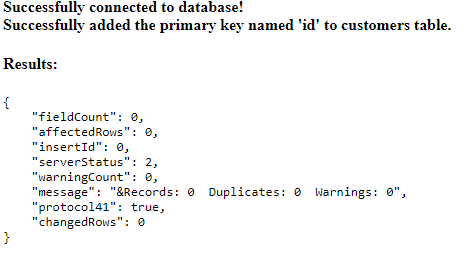
**Task 4: Create a table named "customers". Save the code above in a file called "demo\_create\_table.js" and run the file**

|  |
| --- |
| con.query("CREATE TABLE customers" +  "(id INT," +  "fname VARCHAR(50)," +  "lname VARCHAR(50)," +  "location VARCHAR(200)" +  ");",  *function* (err, result) {  *if*(err) *throw* err;  response.write("<b>Successfully created the table customers!</b><br />");  response.write("<pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |



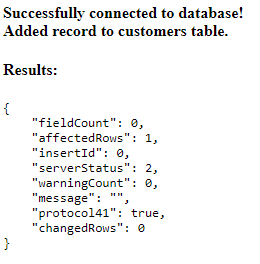
**Task 5: Create primary key when creating the table. If the table already exists, use the ALTER TABLE keyword.**

|  |
| --- |
| *let* queryStr = "ALTER TABLE customers ADD PRIMARY KEY(id);"; con.query(queryStr,  *function* (err, result) {  *if*(err) *throw* err;  response.write("<b>Successfully added the primary key named 'id' to customers table.</b><br />");  response.write("<h4>Results:</h4><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |



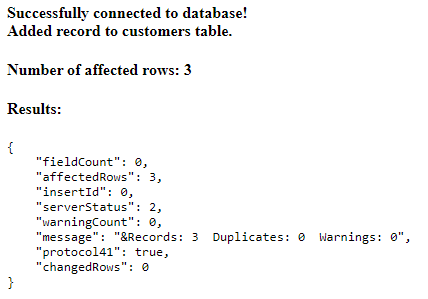
**Task 6: Insert a record in the "customers" table. Save the code above in a file called "demo\_db\_insert.js", and run the file.**

|  |
| --- |
| *let* queryStr = "INSERT INTO customers VALUES (0, 'Hasnain', 'Naeem', 'Sahiwal');"; con.query(queryStr,  *function* (err, result) {  *if*(err) *throw* err;  response.write("<b>Added record to customers table.</b><br />");  response.write("<h4>Results:</h4><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |



**Task 7: Fill the "customers" table with multiple data. Save the code above in a file called "demo\_db\_insert\_multple.js", and run the file. Return the number of affected rows**

|  |
| --- |
| *let* queryStr = "INSERT INTO customers VALUES ?;"; *let* values = [  [1, "ahmad", "khan", "sahiwal"],  [2, "daim", "ejaz", "lahore"],  [3, "saqib", "ashraf", "sahiwal"], ]; con.query(queryStr, [values], *function* (err, result) {  *if*(err) *throw* err;  response.write("<b>Added record to customers table.</b><br />");  response.write("<h4>Number of affected rows: " + result.affectedRows + "</h4>");  response.write("<h4>Results:</h4><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |

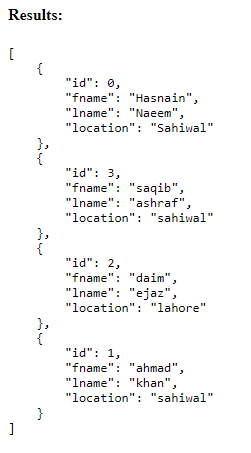


**Adding 2 more records:**

|  |
| --- |
| *let* queryStr = "INSERT INTO customers VALUES ?;"; *let* values = [  [4, "ali", "nazeer", "Valley 345"],  [5, "shah", "nawaz", "Park Lane 38"],  [6, "syed", "akbar", "karachi"],  [7, "haris", "owais", "karachi"],  [ 8, "muazzam", "ali", "layyah"],  ]; con.query(queryStr, [values], *function* (err, result) {  *if*(err) *throw* err;  response.write("<b>Added record to customers table.</b><br />");  response.write("<h4>Number of affected rows: " + result.affectedRows + "</h4>");  response.write("<h4>Results:</h4><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |

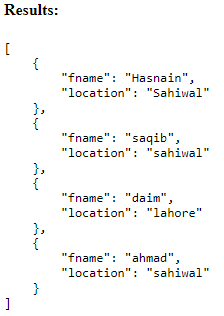
**Task 8: Select all records from the "customers" table, and display the result object. Save the code above in a file called "demo\_db\_select.js" and run the file.**

|  |
| --- |
| *let* queryStr = "SELECT \* FROM customers;"; con.query(queryStr, *function* (err, result) {  *if*(err) *throw* err;  response.write("<h4>Results:</h4><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |



**Task 9: Select name and address from the "customers" table, and display the return object. Save the code above in a file called "demo\_db\_select2.js" and run the file.**

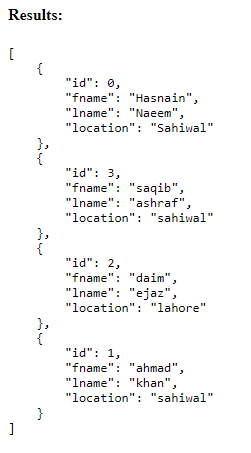
|  |
| --- |
| *let* queryStr = "SELECT fname, location FROM customers;"; con.query(queryStr, *function* (err, result) {  *if*(err) *throw* err;  response.write("<h4>Results:</h4><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |



**Task 10: Select all records from the "customers" table, and display the fields object. Save the code above in a file called "demo\_db\_select\_fields.js" and run the file**

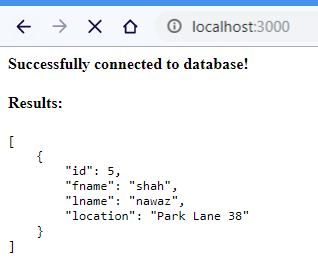
**(Repeated Task)**

|  |
| --- |
| *let* queryStr = "SELECT \* FROM customers;"; con.query(queryStr, *function* (err, result) {  *if*(err) *throw* err;  response.write("<h4>Results:</h4><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |



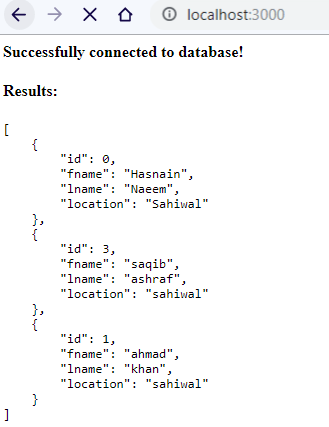
**Task11: Select record(s) with the address "Park Lane 38". Save the code above in a file called "demo\_db\_where.js" and run the file**

|  |
| --- |
| *let* queryStr = "SELECT \* FROM customers WHERE location='Park Lane 38';"; con.query(queryStr, *function* (err, result) {  *if*(err) *throw* err;  response.write("<h4>Results:</h4><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |



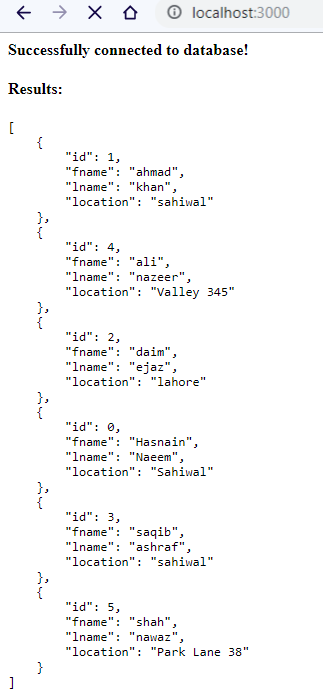
**Task 12: Select records where the address starts with the letter 'S'. Save the code above in a file called "demo\_db\_where\_s.js" and run the file.**

|  |
| --- |
| *let* queryStr = "SELECT \* FROM customers WHERE location LIKE 's%'"; con.query(queryStr, *function* (err, result) {  *if*(err) *throw* err;  response.write("<h4>Results:</h4><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |



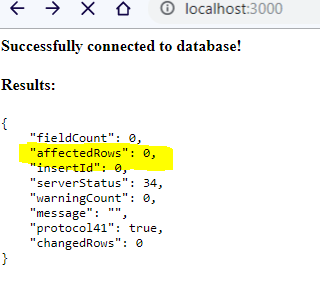
**Task 13: Sort the result alphabetically by name. Save the code above in a file called "demo\_db\_orderby.js" and run the file**

|  |
| --- |
| *let* queryStr = "SELECT \* FROM customers ORDER BY fname;"; con.query(queryStr, *function* (err, result) {  *if*(err) *throw* err;  response.write("<h4>Results:</h4><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |



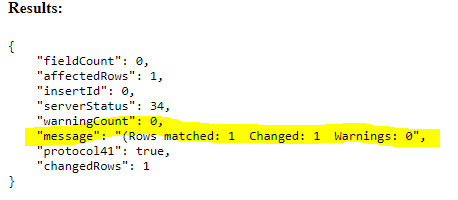
**Task 14: Delete any record with the address "Mountain 21". Save the code above in a file called "demo\_db\_delete.js" and run the file.**

|  |
| --- |
| *let* queryStr = "DELETE FROM customers WHERE location='Mountain 21';"; con.query(queryStr, *function* (err, result) {  *if*(err) *throw* err;  response.write("<h4>Results:</h4><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |



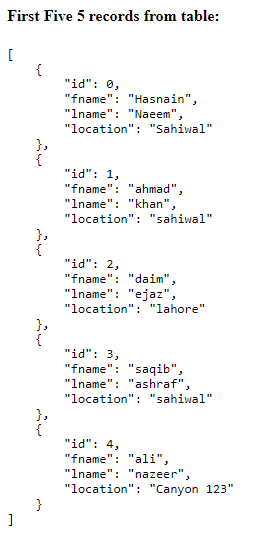
**Task 16:** Overwrite the address column from "Valley 345" to "Canyon 123". Save the code above in a file called "demo\_db\_update.js" and run the file

|  |
| --- |
| *let* queryStr = "UPDATE customers SET location='Canyon 123' WHERE location='Valley 345'"; con.query(queryStr, *function* (err, result) {  *if*(err) *throw* err;  response.write("<h4>Results:</h4><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |



**Task 17: Select the 5 first records in the "customers" table. Save the code above in a file called "demo\_db\_limit.js" and run the file. Now Start from position 3, and return the next 5 records.**

|  |
| --- |
| *let* queryStr = "SELECT \* FROM customers ORDER BY id LIMIT 5;"; con.query(queryStr, *function* (err, result) {  *if*(err) *throw* err;  response.write("<h4>First Five 5 records from table:</h4><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |



**Second Part:**

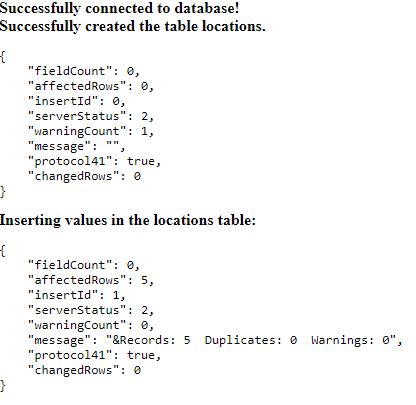
|  |
| --- |
| *let* queryStr = "SELECT \* FROM customers ORDER BY id LIMIT 5 OFFSET 2;"; con.query(queryStr, *function* (err, result) {  *if*(err) *throw* err;  response.write("<h4>Starting from 3rd, 5 records are (ID begins from 0):</h4><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }); |



**Task 18: Practice the Join operations on different tables.**

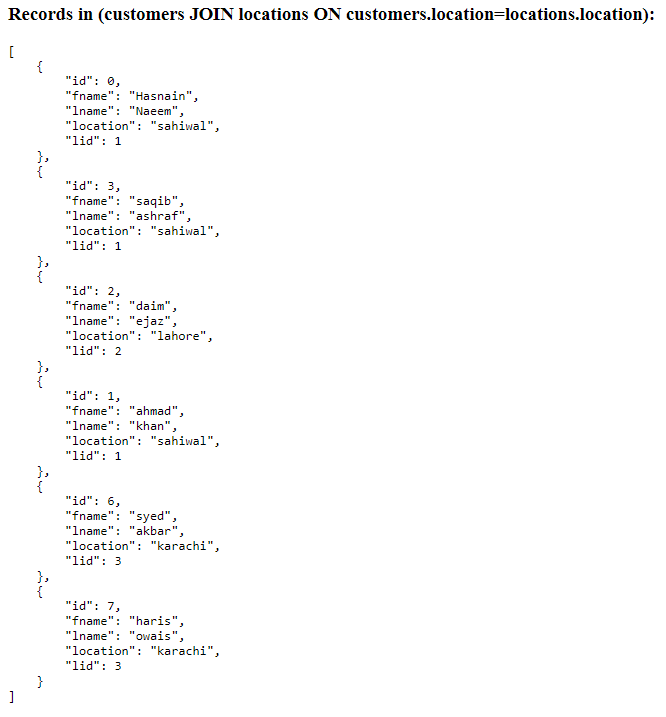
1. Creating new table named “locations” and inserting rows

|  |
| --- |
| *let* queryStr = "CREATE TABLE IF NOT EXISTS Locations(" +  "lid INT PRIMARY KEY AUTO\_INCREMENT," +  "location VARCHAR(200)," +  "FOREIGN KEY (lid) REFERENCES customers(id));"; con.query(queryStr,  *function* (err, result) {  *if*(err) *throw* err;  response.write("<b>Successfully created the table locations.</b><br />");  response.write("<pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>");  }); queryStr = "INSERT INTO locations(location) VALUES?;"; values = [  ["sahiwal"],  ["lahore"],  ["karachi"],  ["peshawar"],  ["quetta"] ]; con.query(queryStr,[values], *function*(err, result){  *if*(err) *throw* err;  response.write("<b>Inserting values in the locations table:</b><br />");  response.write("<pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>"); }) |



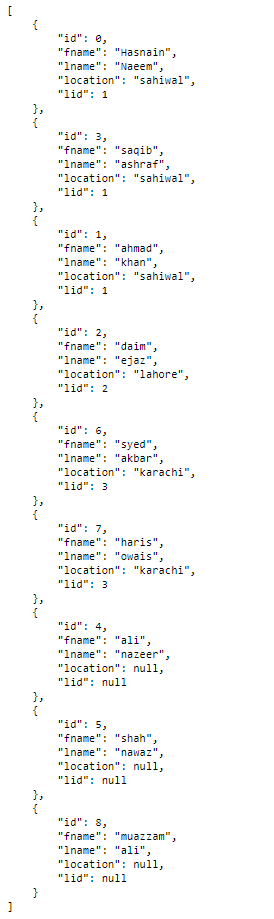
1. **Joining customers and locations ON location**

|  |
| --- |
| *let* queryStr = "SELECT \* FROM customers JOIN locations ON customers.location=locations.location;"; con.query(queryStr,  *function* (err, result) {  *if*(err) *throw* err;  response.write("<h3>Records in (customers JOIN locations ON customers.location=locations.location):</h3><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>");  }); |



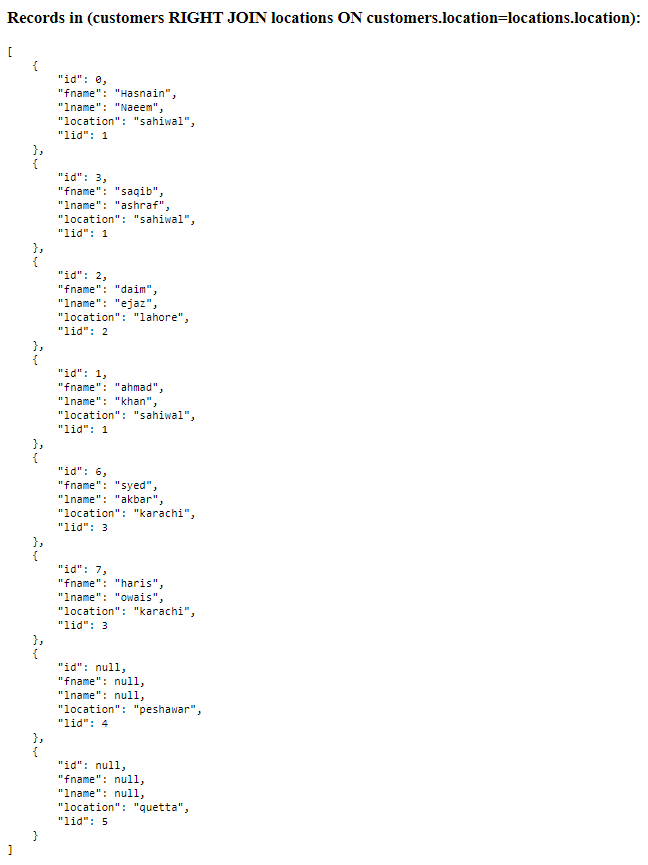
1. **Left Join: ON customers.location=locations.location**

|  |
| --- |
| *let* queryStr = "SELECT \* FROM customers LEFT JOIN locations ON customers.location=locations.location;"; con.query(queryStr,  *function* (err, result) {  *if*(err) *throw* err;  response.write("<h3>Records in (customers LEFT JOIN locations ON customers.location=locations.location):</h3><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>");  }); |



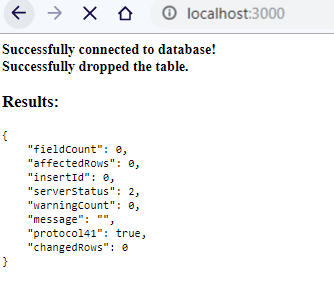
1. **Right Join: ON customers.location=locations.location**

|  |
| --- |
| *let* queryStr = "SELECT \* FROM customers RIGHT JOIN locations ON customers.location=locations.location;"; con.query(queryStr,  *function* (err, result) {  *if*(err) *throw* err;  response.write("<h3>Records in (customers RIGHT JOIN locations ON customers.location=locations.location):</h3><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>");  }); |



**Task 15: Delete the table "customers". Save the code above in a file called "demo\_db\_drop\_table\_if.js" and run the file.**

|  |
| --- |
| *let* queryStr = "DROP TABLE IF EXISTS customers;"; con.query(queryStr,  *function* (err, result) {  *if*(err) *throw* err;  response.write("<b>Successfully dropped the table.</b><br />");  response.write("<h3>Results:</h3><pre>" + JSON.stringify(result, *undefined*, 4) + "</pre>");  }); |



### 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).