Lesson 06 Demo 03

Database Connectivity Commands

Objective: To connect the with a MySQL database using Node.js and perform various operations related to the management of a database

Tools required: Node Package Manager, Visual Studio Code, and MySQL

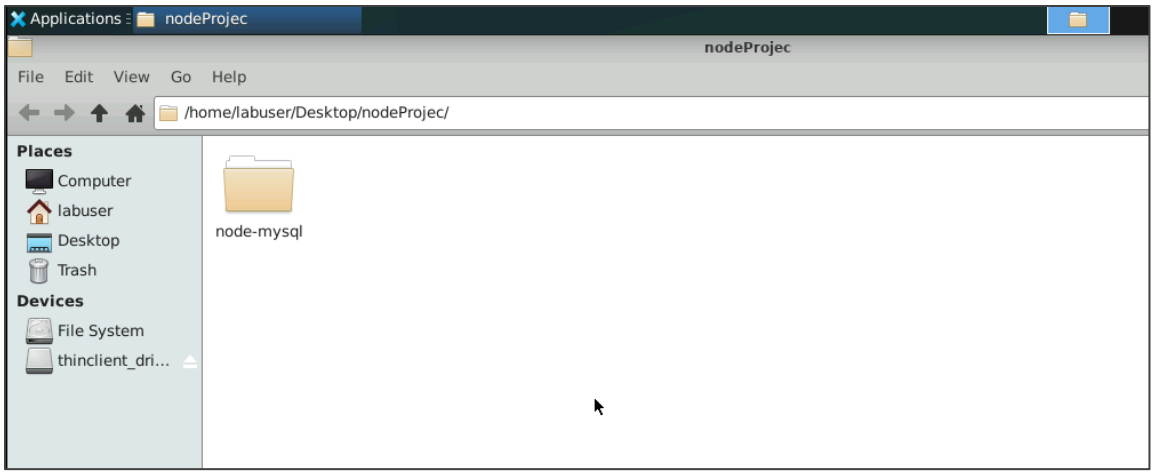
Prerequisites: Basic Linux Commands, NPM commands, SQL commands, and JavaScript

Steps to be followed:

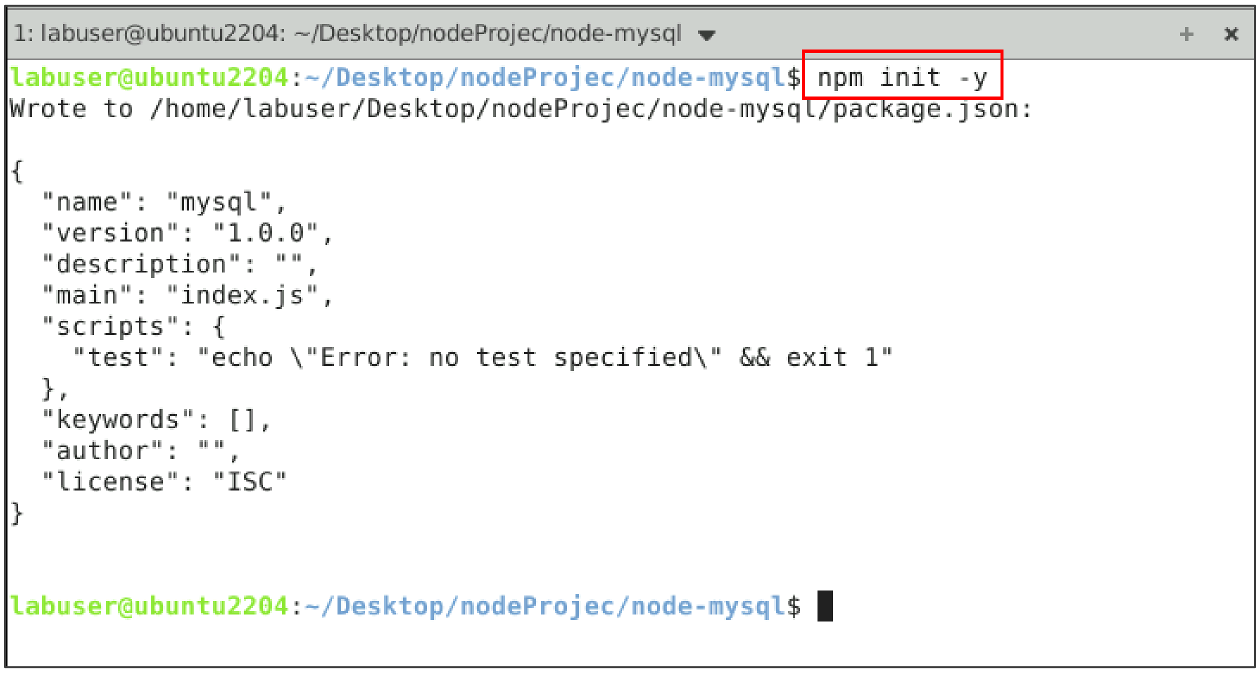
1. Install MySQL Node.js driver
2. Configure MySQL database server connection
3. Close the database connection
4. Create MySQL tables using Node.js
5. Insert data in MySQL tables using Node.js
6. Query the MySQL database using Node.js
7. Modify the MySQL tables with Node.js
8. Delete data from MySQL tables using Node.js

Step 1: Install MySQL Node.js driver

1.1 Create a folder named **node-mysql** for storing the NodeJS app

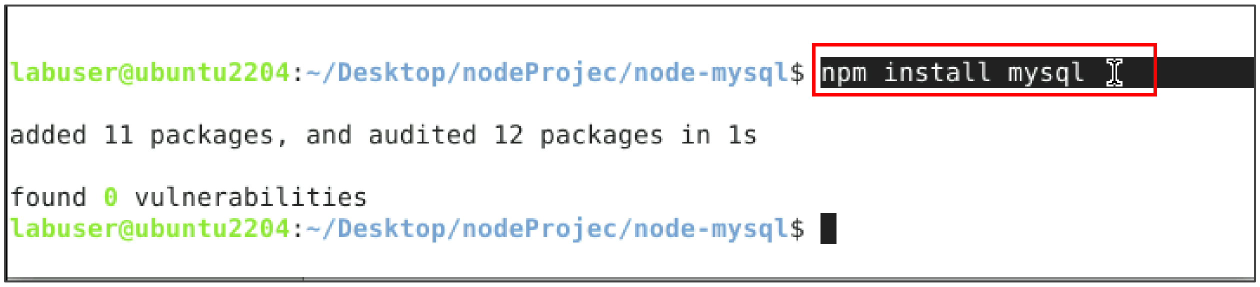


1.2 Open the project folder with the terminal and run the following command to initialize Node.js:
npm init -y



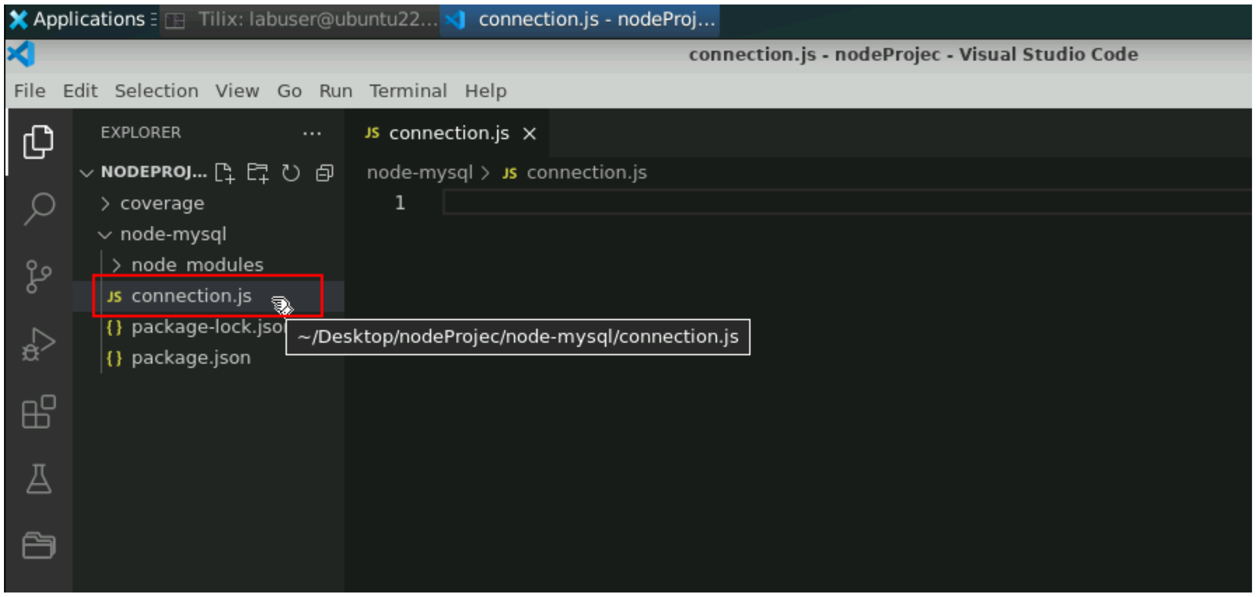
```
1: labuser@ubuntu2204: ~/Desktop/nodeProjec/node-mysql
labuser@ubuntu2204:~/Desktop/nodeProjec/node-mysql$ npm init -y
Wrote to /home/labuser/Desktop/nodeProjec/node-mysql/package.json:

{
  "name": "mysql",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}


labuser@ubuntu2204:~/Desktop/nodeProjec/node-mysql$
```

1.3 Install the MySQL package for Node.js by executing the following command:
npm install mysql

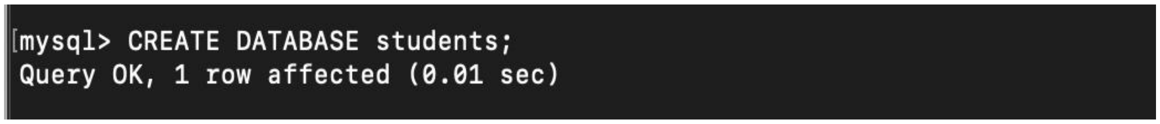


```
labuser@ubuntu2204:~/Desktop/nodeProjec/node-mysql$ npm install mysql

added 11 packages, and audited 12 packages in 1s

found 0 vulnerabilities
labuser@ubuntu2204:~/Desktop/nodeProjec/node-mysql$
```

1.4 Create the connection.js script within the **node-mysql** folder to store the code that links to the MySQL database server


Applications
Tilix: labuser@ubuntu22...
connection.js - nodeProj...
connection.js - nodeProjec - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
NODEPROJ...
coverage
node-mysql
node_modules
connection.js
package-lock.json
package.json
node-mysql > connection.js
~/Desktop/nodeProjec/node-mysql/connection.js

1.5 Leverage the **todoapp** database for demonstration purposes, and create the database in the MySQL database server by running the following CREATE DATABASE command:
CREATE DATABASE students



```
mysql> CREATE DATABASE students;
Query OK, 1 row affected (0.01 sec)
```

Step 2: Configure MySQL database server connection

2.1 Import the module **mysql** within the file **connection.js** using the command given below:

const mysql = require("mysql");

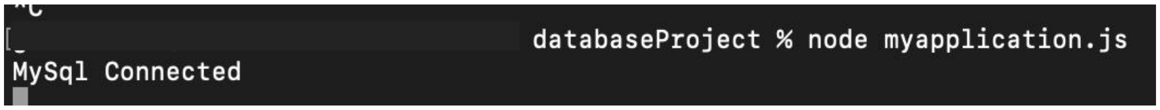
2.2 Use the following code to create a connection to the database:

```
// Create Connection
const db = mysql.createConnection({

host: "localhost",
  user: "root",
  password: "",
  database: "students",
});
```

2.3 Use the following code to connect to the database:

```
// Connect to Database
db.connect((err) => {
  if (err) {
    throw err;
  }
  console.log("MySql Connected");
});
```

2.4 Test the **connection.js** program with the help of the above code snippet:



```
^C
[_[redacted] databaseProject % node myapplication.js
MySql Connected
```

Step 3: Close the database connection

3.1 Use the following code to close a connection to the server:

```
connection.end((err) => {
 if (err) {
   return console.log('error:' + err.message);
 }
 console.log('Close the database connection.');
});
```

3.2 Use the following code statement to force the close of the connection:

```
connection.destroy();
```

Note: The **destroy()** method does not have a callback argument, unlike the **end()** method.

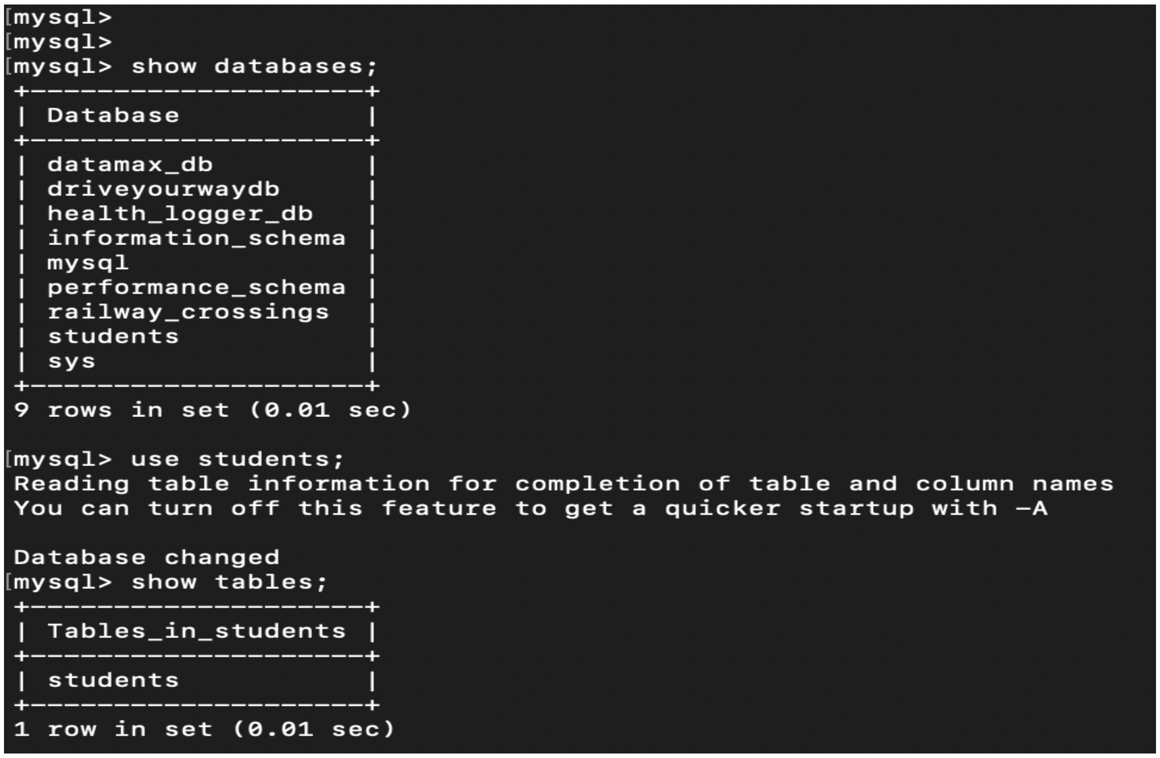
Step 4: Create MySQL Tables using Node.js

4.1 After connecting to the database, use the code within a new file **query.js** to create a new table inside the **todoapp** database:

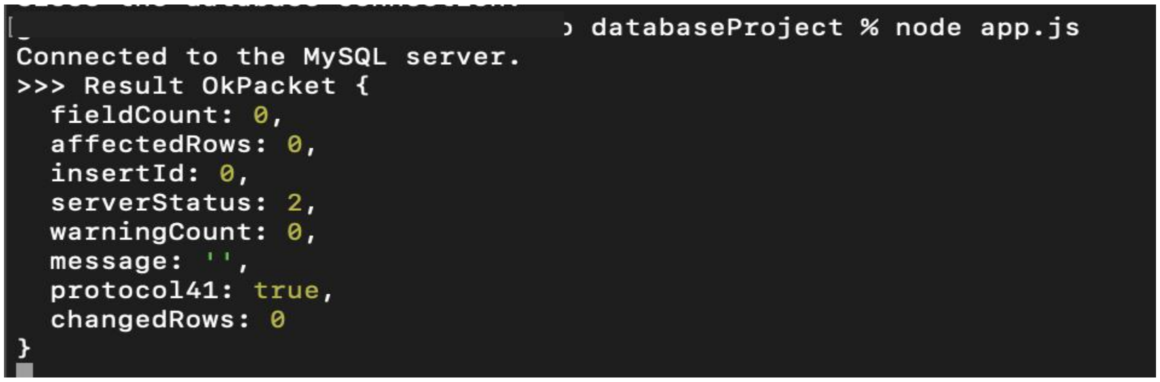
```
const mysql = require("mysql");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "students",
});

connection.connect((err) => {
  if (err) {
    return console.error('error: ' + err.message);
  }
  console.log('Connected to the MySQL server.');
  let createStdTable = `create table if not exists students(
    id int primary key auto_increment,
    name varchar(255) not null,
    email varchar(255) not null,
    phone varchar(255),
    address varchar(255)
  )`;

  connection.query(createStdTable, (error, result) => {
    if (error) throw error;
    console.log(">>> Result", result);
  })
})
```

4.2 Run the app by executing the following command in the terminal:

node query.js



```
[mysql>
[mysql>
[mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| datamax_db         |
| driveyourwaydb     |
| health_logger_db   |
| information_schema |
| mysql              |
| performance_schema |
| railway_crossings  |
| students           |
| sys                |
+--------------------+
9 rows in set (0.01 sec)

[mysql> use students;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
[mysql> show tables;
+--------------------+
| Tables_in_students |
+--------------------+
| students           |
+--------------------+
1 row in set (0.01 sec)
```



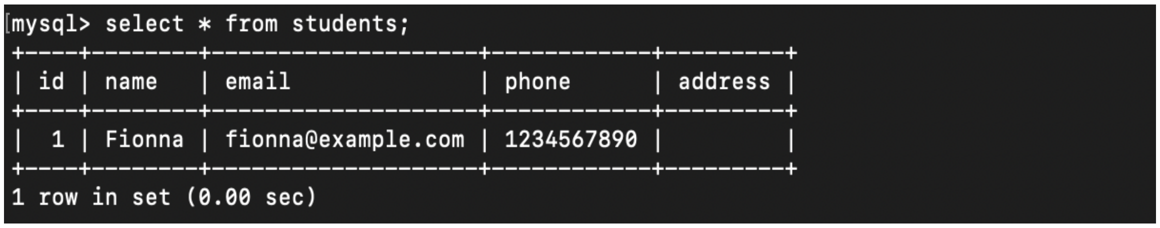
```
[[redacted] databaseProject % node app.js
Connected to the MySQL server.
>>> Result OkPacket {
  fieldCount: 0,
  affectedRows: 0,
  insertId: 0,
  serverStatus: 2,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0
}
```

Step 5: Insert data in MySQL tables using Node.js

5.1 Use the below code to insert data in MySQL tables with Node.js:

```
const mysql = require("mysql");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "students",
});

connection.connect((err) => {
  if (err) {
    return console.error('error: ' + err.message);
  }
  console.log('Connected to the MySQL server.');
  let createStdTable = `insert into students(name, email, phone, address) values
('Fionna', 'fionna@example.com', '1234567890', '')`;

  connection.query(createStdTable, (error, result, _) => {
    if (error) throw error;
    console.log(">>> Result", result);
  })
})
```

5.2 Run the app and verify in MySQL:



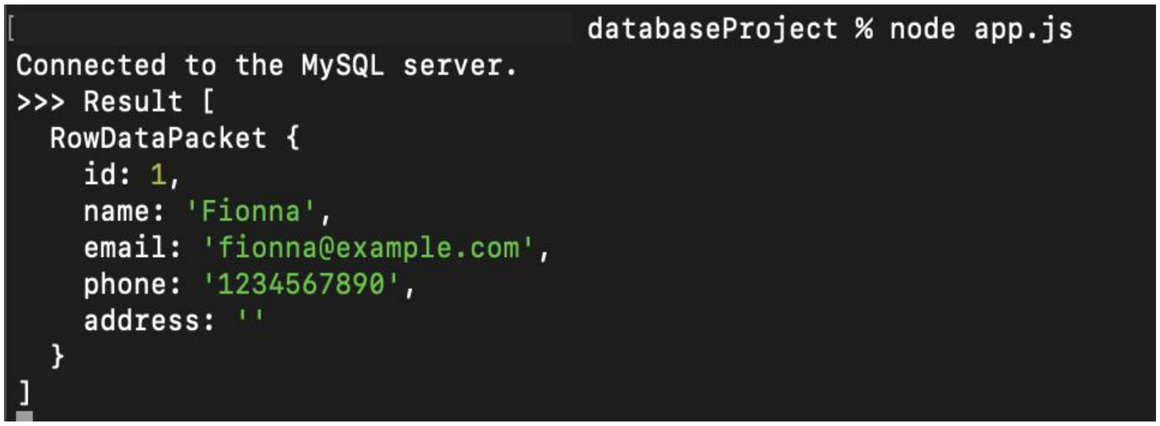
```
mysql> select * from students;
+----+--------+--------------------+------------+---------+
| id | name   | email              | phone      | address |
+----+--------+--------------------+------------+---------+
|  1 | Fionna | fionna@example.com | 1234567890 |         |
+----+--------+--------------------+------------+---------+
1 row in set (0.00 sec)
```

Step 6: Query the MySQL database using Node.js

6.1 Use the below code to query the MySQL database using Node.js, in the code snippet, select all the data from the **todo** table within the **todoapp** database:

```
const mysql = require("mysql");

// Create Connection
const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "students",
});

connection.connect((err) => {
  if (err) {
    return console.error('error: ' + err.message);
  }
  console.log('Connected to the MySQL server.');
  let queryStatement = `select * from students`;

  connection.query(createStdTable, (error, result, _) => {
    if (error) throw error;
    console.log(">>> Result", result);
  })
})
```

6.2 Run the app and verify in MySQL:



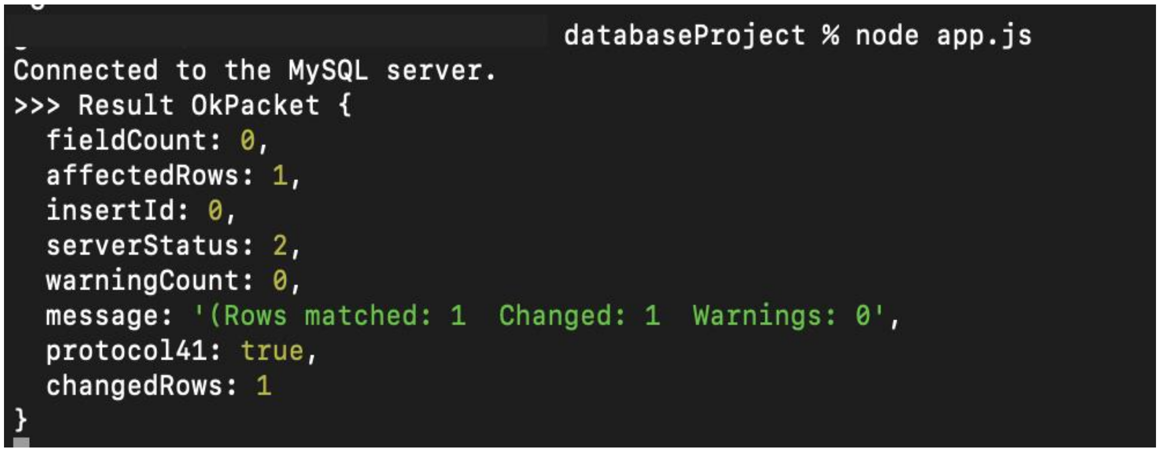
```
[                                    databaseProject % node app.js
Connected to the MySQL server.
>>> Result [
  RowDataPacket {
    id: 1,
    name: 'Fionna',
    email: 'fionna@example.com',
    phone: '1234567890',
    address: ''
  }
]
```

Step 7: Modify the MySQL tables with Node.js

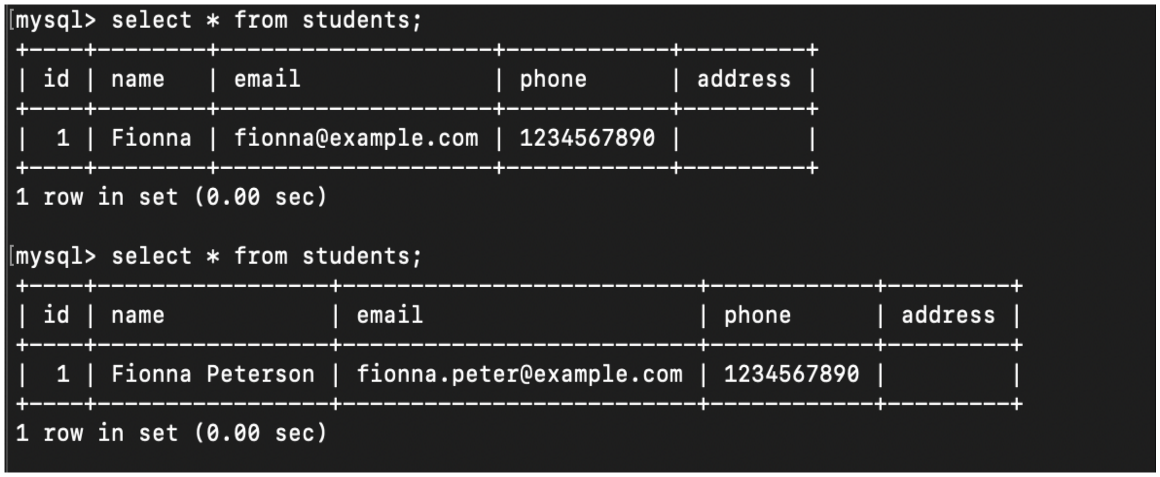
7.1 Use the following code snippet to update the status of a **todo** based on a specified **id**:

```
const mysql = require("mysql");

// Create Connection
const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "students",
});

connection.connect((err) => {
  if (err) {
    return console.error('error: ' + err.message);
  }
  console.log('Connected to the MySQL server.');
  let updateQuery = `update students set name = 'Fionna Peterson', email =
'fionna.peter@example.com' where id = 1 `;

  connection.query(updateQuery, (error, result, _) => {
    if (error) throw error;
    console.log(">>> Result", result);
  })
})
```

7.2 Run the app and verify in MySQL:



```
                                   databaseProject % node app.js
Connected to the MySQL server.
>>> Result OkPacket {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 2,
  warningCount: 0,
  message: '(Rows matched: 1  Changed: 1  Warnings: 0',
  protocol41: true,
  changedRows: 1
}
```

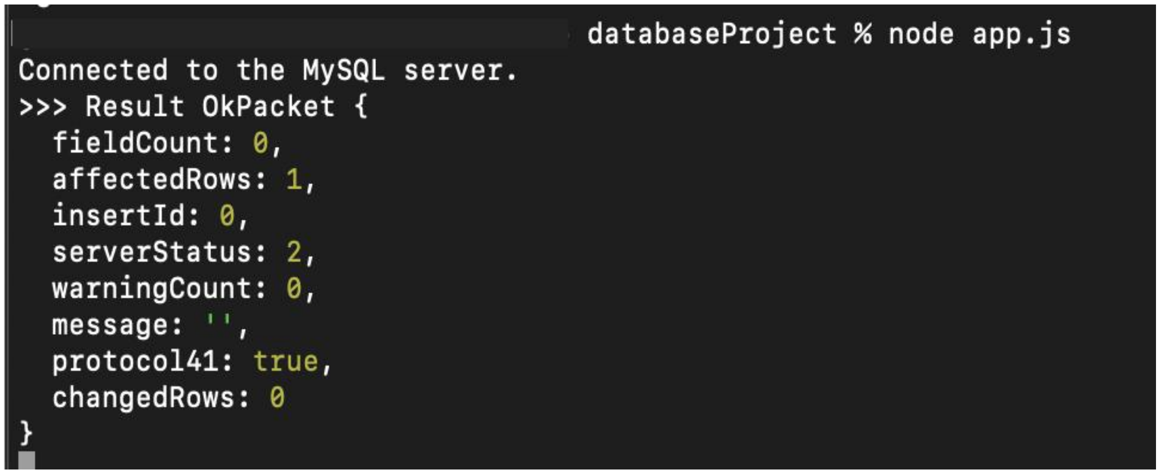

```
[mysql> select * from students;
+----+--------+--------------------+------------+---------+
| id | name   | email              | phone      | address |
+----+--------+--------------------+------------+---------+
|  1 | Fionna | fionna@example.com | 1234567890 |         |
+----+--------+--------------------+------------+---------+
1 row in set (0.00 sec)

[mysql> select * from students;
+----+-----------------+--------------------------+------------+---------+
| id | name            | email                    | phone      | address |
+----+-----------------+--------------------------+------------+---------+
|  1 | Fionna Peterson | fionna.peter@example.com | 1234567890 |         |
+----+-----------------+--------------------------+------------+---------+
1 row in set (0.00 sec)
```

Step 8: Delete Data from MySQL Tables using Node.js

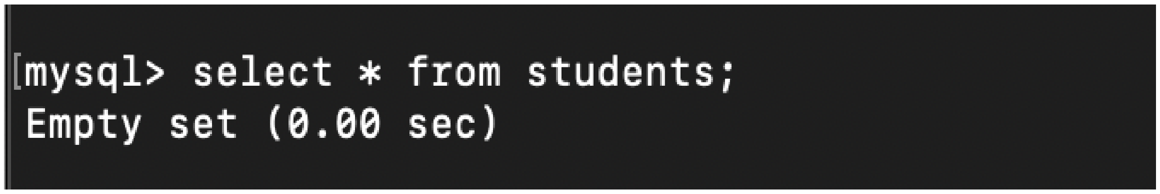
8.1 Use the following **delete.js** programs to delete a row in the **todo** table:

```
const mysql = require("mysql");

// Create Connection
const connection = mysql.createConnection({
  host: "localhost",

user: "root",
  password: "",
  database: "students",
});

connection.connect((err) => {
  if (err) {
    return console.error('error: ' + err.message);
  }
  console.log('Connected to the MySQL server.');

  let deleteQuery = `delete from students where id = 1`;
  connection.query(deleteQuery, (error, result, _) => {
    if (error) throw error;
    console.log(">>> Result", result);
  })
})
```

8.2 **Run the app and verify in MySQL:**



```
[redacted] databaseProject % node app.js
Connected to the MySQL server.
>>> Result OkPacket {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 2,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0
}
```



```
[mysql> select * from students;
 Empty set (0.00 sec)
```

By following these steps, you have successfully connected the with a MySQL database using Node.js and performed various operations related to the management of a database.