Name of Student: Samyak S Lamsoge		
Roll Number: 29		**LAB Assignment Number: 7**
Title of LAB Assignment: Create an application to establish a connection with the MySQL database, perform basic database operations on it(student db consisting rollno,name,address),insert 10 records, update a particular student's record,delete a record.		
DOP: 11-10-2023		**DOS: 17-10-2023**
CO Mapped: CO2	**PO Mapped: PO5, PSO1**	**Signature:**

PRACTICAL 7

Aim: Create an application to establish a connection with the MySQL database and perform basic database operations on it(student db consisting rollno,name,address),insert 10 records, update a particular student's record,delete a record.

Theory:

To work with MySQL in a Node.js application, you typically use a MySQL library. We'll use the 'mysql' library

1. createConnection(config):

This function is used to create a MySQL database connection. You provide a configuration object with connection details such as host, user, password, and database.

Syntax: `const connection = mysql.createConnection(config);`

Example:

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


const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'password',
    database: 'mydb',
});
```

2. CRUD Functions (Create, Read, Update, Delete):

Here are simple examples of CRUD operations using the query() function:

Create (INSERT):

Syntax:

```
const query = "INSERT INTO table_name (column1, column2) VALUES (?, ?)";
const values = [value1, value2];
connection.query(query, values, callback);
```

Example:

```
const newUser = { username: 'john', email: 'john@example.com' };
const query = 'INSERT INTO users (username, email) VALUES (?, ?)';
connection.query(query, [newUser.username, newUser.email], (err, result) => {
    if (err) throw err;
    console.log('New user inserted, ID:', result.insertId);
});
```

Read (SELECT):

Syntax:

```
const query = "SELECT * FROM table_name WHERE condition";
connection.query(query, callback);
```

Example:

```
connection.query('SELECT * FROM users', (err, results, fields) => {
    if (err) throw err;
    console.log('Query results:', results);
});
```

Update (UPDATE):

Syntax:

```
const sql = "UPDATE table_name SET column1 = value1 WHERE condition";
connection.query(sql, callback);
```

Example:

```
const userId = 1;
const newEmail = 'newemail@example.com';
const sql = 'UPDATE users SET email = ? WHERE id = ?';
connection.query(sql, [newEmail, userId], (err, result) => {
    if (err) throw err;
    console.log('User updated:', result.affectedRows);
});
```

Delete (DELETE):

Syntax:

```
const sql = "DELETE FROM table_name WHERE condition";
```

```
connection.query(sql, callback);
```

Example:

```
const userIdToDelete = 2;

const sql = 'DELETE FROM users WHERE id = ?';

connection.query(sql, [userIdToDelete], (err, result) => {

    if (err) throw err;

    console.log('User deleted:', result.affectedRows);

});
```

3. connection.end(callback):

This function is used to close the MySQL connection. It takes an optional callback function that is executed once the connection is closed.

Syntax:

```
connection.end(callback);
```

Example:

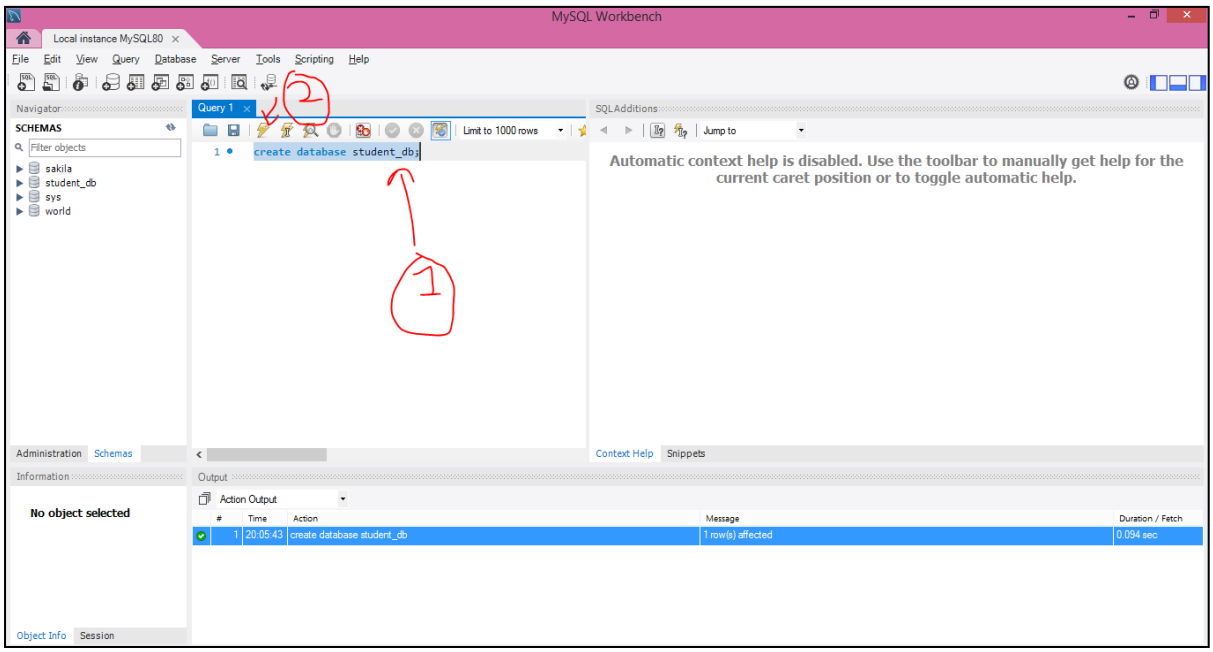
```
connection.end((err) => {

    if (err) {

        console.error('Error closing the MySQL connection:', err);

    }

    console.log('MySQL connection closed');

});
```

Steps to execute the program

Step 1: install mysql workbench:
https://youtu.be/GwHpIl0vqY4?si=dMQDg3sV9xD-P_7Y

Step 2: create a database student_db

```
create database student_db;
```



Step 3: execute the program

node "Practical 7.js"

---

1. **Aim: Create an application to establish a connection with the MySQL database and perform basic database operations on it(student db consisting rollno,name,address),insert 10 records, update a particular student's record,delete a record.**

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

```
const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'root',
    database: 'student_db',
});

// Function to establish a connection to the MySQL database
function connectToDatabase() {
    connection.connect((error) => {
        if (error) throw error;

        console.log('Connected to the database\n');
    });
}

// Function to create the students table if it doesn't exist
function createStudentTable() {
    const createTableQuery =
        `
    CREATE TABLE IF NOT EXISTS students
    (
      rollno INT AUTO_INCREMENT PRIMARY KEY,
      name VARCHAR(255) NOT NULL,
      address VARCHAR(255)
    )
```

```
    `;

    connection.query(createTableQuery, (error) => {
        if (error) throw error;
    });
}

function displayAllRecords(message) {
    const query = 'SELECT * FROM students';

    // Execute the query
    connection.query(query, (error, results) => {
        if (error) throw error

        console.log('\n', message);

        // Process the results here
        for (let row of results) {
            console.log(row.rollno, row.name, row.address);
        }
    });
}

// Function to insert a student record
function insertStudentRecord(student) {
    const insertQuery = 'INSERT INTO students SET ?';
```

```
    connection.query(insertQuery, student, (error, results) => {
        if (error) throw error

        console.log(`Inserted record with ID ${results.insertId}`);
    });
}

// Function to update a student record
function updateStudentRecord(rollno, updatedStudent) {
    const updateQuery = 'UPDATE students SET ? WHERE rollno = ?';

    connection.query(updateQuery, [updatedStudent, rollno], (error,
results) => {
        if (error) throw error

        console.log(`\nUpdated ${results.affectedRows} record(s)`);
    });
}

// Function to delete a student record
function deleteStudentRecord(rollno) {
    const deleteQuery = 'DELETE FROM students WHERE rollno = ?';

    connection.query(deleteQuery, rollno, (error, results) => {
        if (error) throw error

        console.log(`\nDeleted ${results.affectedRows} record(s)`);
```

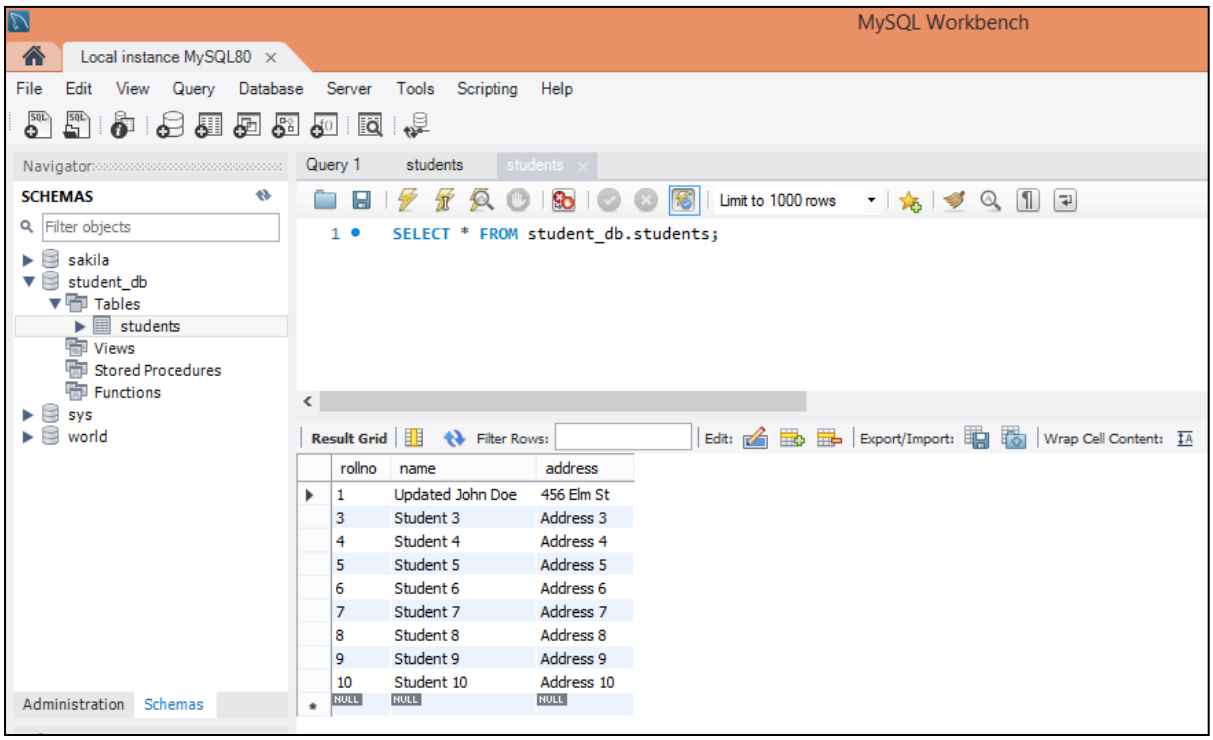
```
    });
}

// Main function
function main() {
    try {
        connectToDatabase();

        createStudentTable();

        // insert 10 records
        for (let i = 1; i <= 10; i++) {
            const newStudent = { name: `Student ${i}`, address:
`Address ${i}` };
            insertStudentRecord(newStudent);
        }

        displayAllRecords('select * from students (after inserting 10
rows)')

        const updatedStudentInfo = { name: 'Updated John Doe', address:
'456 Elm St' };
        updateStudentRecord(1, updatedStudentInfo);

        displayAllRecords('select * from students (after updating
roll_no 1)')
```

```
deleteStudentRecord(2);


        displayAllRecords('select * from students (after deleting
roll_no 2)')
    }
    catch (error) {
        console.log(error.message);
    }
    finally {
        connection.end();
    }
}


main();
```

Output:

```
Connected to the database

Inserted record with ID 1
Inserted record with ID 2
Inserted record with ID 3
Inserted record with ID 4
Inserted record with ID 5
Inserted record with ID 6
Inserted record with ID 7
Inserted record with ID 8
Inserted record with ID 9
Inserted record with ID 10

 select * from students (after inserting 10 rows)
1 Student 1 Address 1
2 Student 2 Address 2
3 Student 3 Address 3
4 Student 4 Address 4
5 Student 5 Address 5
6 Student 6 Address 6
7 Student 7 Address 7
8 Student 8 Address 8
9 Student 9 Address 9
10 Student 10 Address 10
```

```
Updated 1 record(s)

 select * from students (after updating roll_no 1)
1 Updated John Doe 456 Elm St
2 Student 2 Address 2
3 Student 3 Address 3
4 Student 4 Address 4
5 Student 5 Address 5
6 Student 6 Address 6
7 Student 7 Address 7
8 Student 8 Address 8
9 Student 9 Address 9
10 Student 10 Address 10

Deleted 1 record(s)

 select * from students (after deleting roll_no 2)
1 Updated John Doe 456 Elm St
3 Student 3 Address 3
4 Student 4 Address 4
5 Student 5 Address 5
6 Student 6 Address 6
7 Student 7 Address 7
8 Student 8 Address 8
9 Student 9 Address 9
10 Student 10 Address 10
```



Conclusion: We learnt about working with an SQL DB in Node Js.