**Connecting Node.js to phpMyAdmin (MySQL Database)**

**Lesson Objective**

By the end of this lesson, students will be able to:

✅ Connect Node.js to a MySQL database (phpMyAdmin)

✅ Perform CRUD operations (Create, Read, Update, Delete)

✅ Use parameterized queries to prevent SQL injection

✅ Handle errors and display query results

**1. Setting Up the MySQL Connection**

Install the **`mysql`** package

First, initialize a Node.js project and install the **`mysql`** package:

*npm init -y*

*npm install mysql*

**Create a Database Connection Pool**

A connection pool improves performance by reusing database connections.

*const { createPool } = require('mysql');*

*const pool = createPool({*

*host: 'localhost', // MySQL server (usually 'localhost')*

*user: 'root', // Default username (change if needed)*

*password: '', // Default password (empty for XAMPP/WAMP)*

*database: 'courses', // Your database name*

*connectionLimit: 10 // Max connections*

*});*

**2. Performing CRUD Operations**

A. Reading Data (SELECT Query)

*pool.query('SELECT \* FROM course', (error, results, fields) => {*

*if (error) {*

*return console.error(error);*

*}*

*console.log(results); // Array of rows*

*});*

**B. Inserting Data (INSERT Query)**

Use **parameterized queries (`?`)** to prevent SQL injection.

*const studentData = ['Fredu', 'Durant', '2000-10-12', '2001-08-12'];*

*pool.query(*

*'INSERT INTO course (first\_name, last\_name, date\_of\_birth, enrollment\_date) VALUES (?, ?, ?, ?)',*

*studentData,*

*(error, results) => {*

*if (error) {*

*return console.error(error);*

*}*

*console.log('Inserted ID:', results.insertId); // Newly inserted ID*

*}*

*);*

**C. Updating Data (UPDATE Query)**

*pool.query(*

*'UPDATE course SET first\_name = ? WHERE course\_id = ?',*

*['Fredu Updated', 3], // New name, ID to update*

*(error, results) => {*

*if (error) {*

*return console.error(error);*

*}*

*console.log('Affected rows:', results.affectedRows);*

*}*

*);*

**D. Deleting Data (DELETE Query)**

*pool.query(*

*'DELETE FROM course WHERE course\_id = ?',*

*[1], // ID to delete*

*(error, results) => {*

*if (error) {*

*return console.error(error);*

*}*

*console.log('Deleted rows:', results.affectedRows);*

*}*

*);*

**3. Best Practices**

**Always Use Parameterized Queries**

❌ Bad: *`'SELECT \* FROM users WHERE username = "' + userInput + '"'`* (SQL Injection Risk!)

✅ Good: *`'SELECT \* FROM users WHERE username = ?', [userInput]`*

**Handle Errors Properly**

Always check `error` in callbacks.

**✅ Close Connections (Optional)**

If not using a pool, manually close connections:

*connection.end(); // For single connections*

// OR

*pool.end(); // For pools (when shutting down the app)*

**4. Exercises for Students**

**Exercise 1**

Create a new table `students` with:

- `student\_id` (INT, AUTO\_INCREMENT, PRIMARY KEY)

- `name` (VARCHAR)

- `email` (VARCHAR, UNIQUE)

- `age` (INT)

**Exercise 2**

Write a Node.js script to:

1. Insert 3 students into the `students` table.

2. Update the email of the second student.

3. Delete the first student.

4. Display all remaining students.

**Final Code Example (All CRUD in One File)**

*const { createPool } = require('mysql');*

*const pool = createPool({*

*host: 'localhost',*

*user: 'root',*

*password: '',*

*database: 'courses',*

*connectionLimit: 10*

*});*

***// SELECT***

*pool.query('SELECT \* FROM course', (err, results) => {*

*if (err) throw err;*

*console.log(results);*

*});*

***// INSERT***

*pool.query(*

*'INSERT INTO course(first\_name, last\_name) VALUES(?, ?)',*

*['John', 'Doe'],*

*(err, results) => {*

*if (err) throw err;*

*console.log('Inserted:', results.insertId);*

*}*

*);*

***// UPDATE***

*pool.query(*

*'UPDATE course SET first\_name=? WHERE course\_id=?',*

*['Jane', 1],*

*(err, results) => {*

*if (err) throw err;*

*console.log('Updated:', results.affectedRows);*

*}*

*);*

***// DELETE***

*pool.query(*

*'DELETE FROM course WHERE course\_id=?',*

*[2],*

*(err, results) => {*

*if (err) throw err;*

*console.log('Deleted:', results.affectedRows);*

*}*

*);*