**Contents**

[**Creating The Database 2**](#_Toc195999410)

[**Normalization 4**](#_Toc195999411)

[**Useful Queries For Our Database 6**](#_Toc195999412)

[**ER Diagram & Relational Schema 8**](#_Toc195999413)

[**Source Code 9**](#_Toc195999414)

[**What is solved? 13**](#_Toc195999415)

[**Challenges & Lessons Learned 13**](#_Toc195999416)

[**Application Sample 14**](#_Toc195999417)

**THIS PROJECT IS CREATED BY MUHMEDSADIQ JASIM 2ND YEAR NETWORK ENGINEERING DEPT.**

**Portfolio:** [**https://muhmedsadiqjasim.github.io/Portfolio/**](https://muhmedsadiqjasim.github.io/Portfolio/)

**تم إنشاء هذا المشروع بواسطة محمد صادق احمد جاسم – مرحلة ثانية – قسم شبكات**

**AL-NAHRAIN UNIVERSITY**

**Database Project**

**2025-04-20**

# **Creating The Database**

**CREATE DATABASE uscsrs;**

**USE uscsrs;**

**CREATE TABLE Student (**

**student\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**name VARCHAR(100) NOT NULL,**

**email VARCHAR(100) UNIQUE NOT NULL,**

**department VARCHAR(50) NOT NULL);**

**CREATE TABLE Club (**

**club\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**club\_name VARCHAR(100) NOT NULL,**

**description TEXT);**

**CREATE TABLE Session (**

**session\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**club\_id INT,**

**session\_title VARCHAR(100) NOT NULL,**

**session\_date DATE NOT NULL,**

**location VARCHAR(100),**

**FOREIGN KEY (club\_id) REFERENCES Club(club\_id));**

**CREATE TABLE Registration (**

**registration\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**student\_id INT,**

**session\_id INT,**

**registration\_date DATE NOT NULL,**

**FOREIGN KEY (student\_id) REFERENCES Student(student\_id),**

**FOREIGN KEY (session\_id) REFERENCES Session(session\_id));**

# **Normalization**

What the database will looks like if we don’t use normalization?

The tables of our database will look like that:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| student\_id | name | email | department | club\_name | session\_title | session\_date | … |
| 1 | Muhmedsadiq | example@gmail.com | Network Dept. | IT Students | SOC Introduction | 2025-05-01 | … |

SOOO…

This is called Unnormalized Form (UNF).  
We need to organize our database to read, insert and modify the data as we like.

To do that we will use (**1NF**, **2NF** and **3NF**).

**1st Normal Form (1NF)**

1. Each column should contain atomic (indivisible) values (Atomicity).
2. Each row should have a unique identifier (Primary Key).

We will make 4 tables each table will includes the data in a specific place.  
We will have (Student, Club, Session and Registration table).

For example, Student table will look like that:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| student\_id (PK) | name | session\_title | department | registration\_date |
| 1 | Ahmed Duraid | SOC Introduction | Network Engineering Dept. | 2025-04-28 |

This table looks better than the previous one for sure. We can read, insert and modify the data easily now but also we have something to do to make the database much better.

**2nd Normal Form (2NF) – “Before that apply the 1NF”**

* Remove Partial Dependencies.

Student Table

|  |  |
| --- | --- |
| student\_id (PK) | Name |
| 1 | Ahmed Duraid |

Session Table

|  |  |  |
| --- | --- | --- |
| session\_id (PK) | club\_name | session\_date |
| 1 | IT Students | 2025-05-01 |

Now we have to do some enhance. let’s go to the **3NF**.

**3rd Normal Form (3NF) – “Before that apply 1NF & 2NF”**

* Remove transitive dependencies.

|  |  |  |  |
| --- | --- | --- | --- |
| session\_id (PK) | session\_title | club\_id (FK) | session\_date |
| 1 | SOC Introduction | 1 | 2025-05-01 |

Note: I just showed the effect of normalization in general, I didn’t show the full effect on our database.

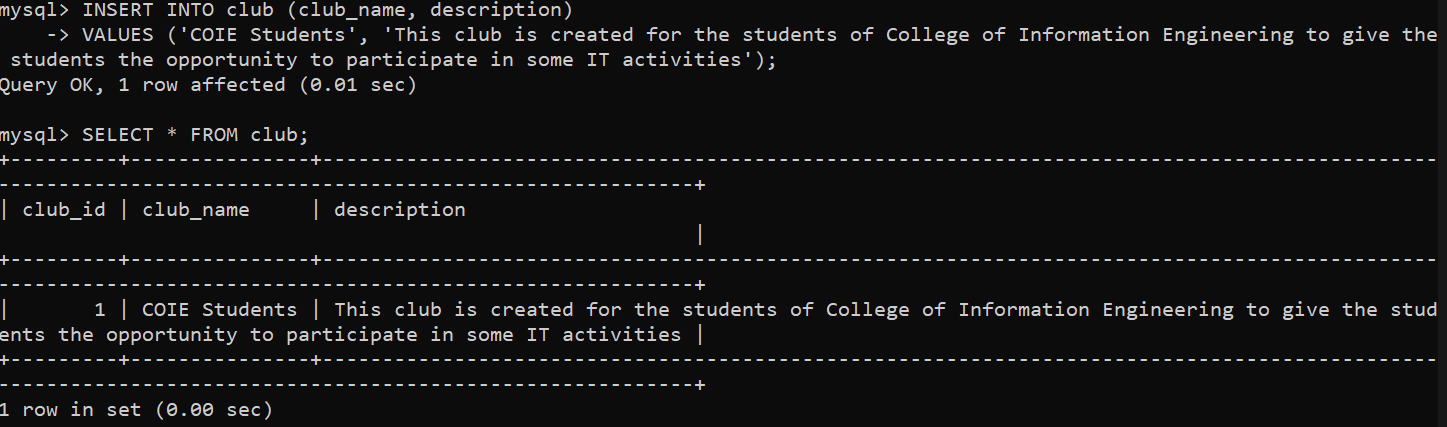
# **Useful Queries For Our Database**

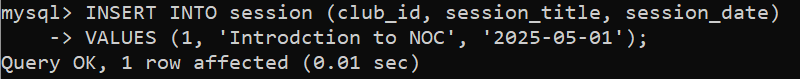
**INSERT – Add a New Student**

INSERT INTO Student (name, email, department)

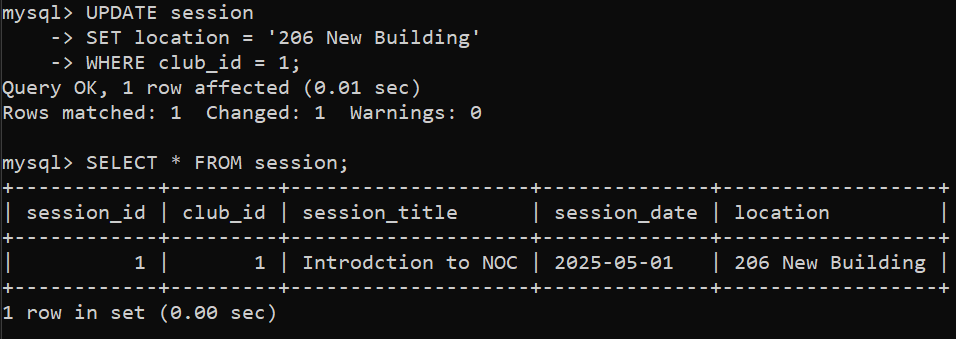
VALUES ('Ali Kareem', 'ali.kareem@email.com', 'Computer Engineering');







**Update – Add a New Location to the Session**

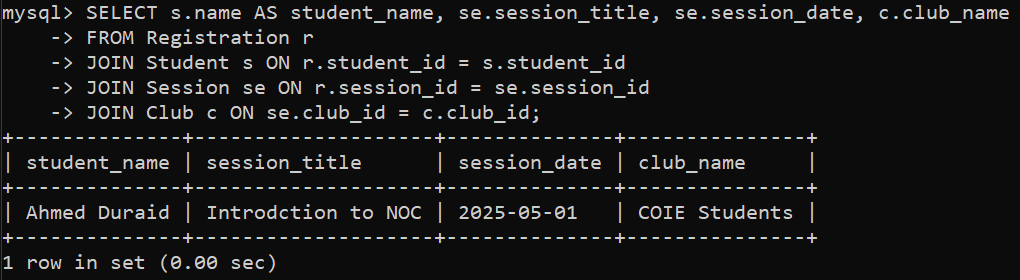
****

**DELETE – Remove a Student’s Registration**

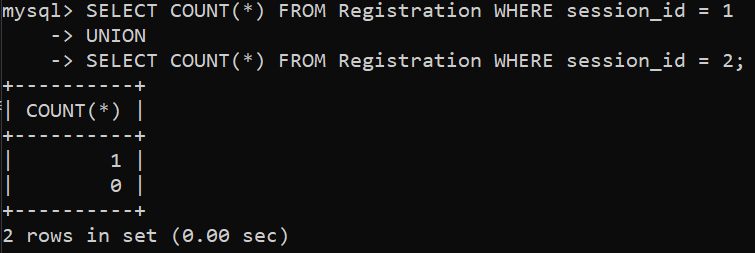
DELETE FROM Registration

WHERE registration\_id = 5;

**SELECT & JOIN – List Students and Their Sessions**



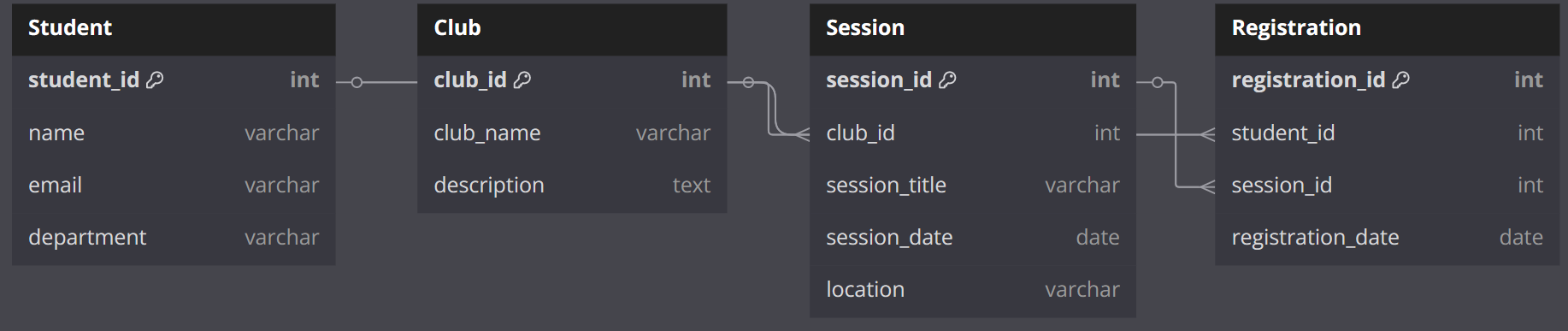
**SELECT, COUNT & UNION – Show the Number of registers in the Sessions**

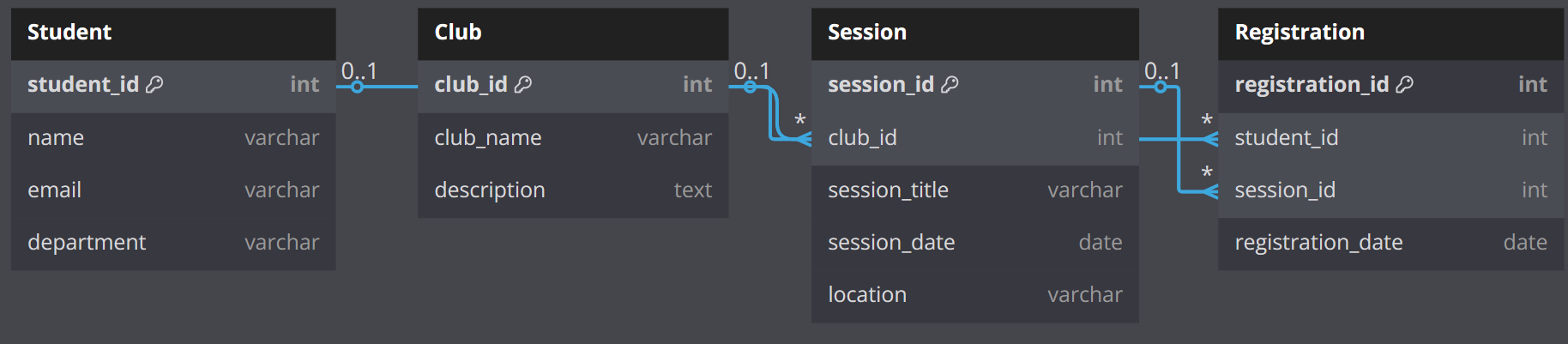
****

There are a lot more of queries that will help use in this database, but this is enough to demonstrate the benefits of use SQL for our data.

# **ER Diagram & Relational Schema**

**I use this site to help me doing the ER Diagram with the Relational Schema**[**https://dbdiagram.io/d**](https://dbdiagram.io/d)





**Explination**

1. **Student to Registration  
   Type: One-to-Many  
   One student can register for many sessions.**
2. **Registration to Student  
   Type: Many-to-One  
   Each registration is linked to one student.**
3. **Session to Registration  
   Type: One-to-Many  
   One session can have many student registrations.**
4. **Registration to Session  
   Type: Many-to-One  
   Each registration is for one session.**
5. **Club to Session  
   Type: One-to-Many  
   One club can create and manage many sessions.**
6. **Session to Club  
   Type: Many-to-One  
   Each session is part of one club.**

# **Source Code**

**db.php**

<?php

$hostname = "localhost";

$username = "root";

$password = "HamHamHam9911";

$database = "uscsrs";

$conn = new mysqli($hostname, $username, $password, $database);

if ($conn->connect\_error) {

    die("Connection failed: " . $conn->connect\_error);

}

?>

**index.php (main)**

<?php include "db.php"; ?>

<!DOCTYPE html>

<html>

<head>

    <title>Events</title>

    <link rel="stylesheet" href="stylez.css">

</head>

<body>

    <h2>Available Events</h2>

    <?php

    $q = $conn->query("

    SELECT se.session\_id, se.session\_title, se.location, c.club\_name

    FROM Session se

    JOIN Club c ON se.club\_id = c.club\_id

    WHERE se.session\_date >= CURDATE()

");

    while ($row = $q->fetch\_assoc()):

    ?>

        <div class="event">

            <p><?php echo $row['session\_title']; ?></p>

            <p><?php echo $row['club\_name']; ?></p>

            <p><?php echo $row['location']; ?></p>

            <a href="register.php?session\_id=<?php echo $row['session\_id']; ?>">Join</a>

        </div>

    <?php endwhile; ?>

</body>

</html>

**register.php**

<?php include "db.php"; ?>

<!DOCTYPE html>

<html>

<head>

    <title>Join Event</title>

    <link rel="stylesheet" href="stylez.css">

</head>

<body>

<?php

if ($\_SERVER["REQUEST\_METHOD"] === "POST") {

    $name = $\_POST["name"];

    $email = $\_POST["email"];

    $department = $\_POST["department"];

    $session\_id = $\_POST["session\_id"];

    $stmt = $conn->prepare("INSERT INTO Student (name, email, department) VALUES (?, ?, ?)");

    $stmt->bind\_param("sss", $name, $email, $department);

    $stmt->execute();

    $student\_id = $conn->insert\_id;

    $stmt2 = $conn->prepare("INSERT INTO Registration (student\_id, session\_id, registration\_date) VALUES (?, ?, CURDATE())");

    $stmt2->bind\_param("ii", $student\_id, $session\_id);

    $stmt2->execute();

    echo "<h2>You joined the event successfully!</h2>";

} else {

    $session\_id = $\_GET['session\_id'] ?? 1;

?>

    <h2>Join Event</h2>

    <form method="post">

        <input type="hidden" name="session\_id" value="<?php echo $session\_id; ?>">

        <input type="text" name="name" placeholder="Your name" required><br><br>

        <input type="email" name="email" placeholder="Your email" required><br><br>

        <select name="department" required>

            <option value="">Select Department</option>

            <option value="Information Engineering">Information Engineering</option>

            <option value="Computer Science">Computer Science</option>

            <option value="Arts">Arts</option>

        </select><br><br>

        <button type="submit">Join Now</button>

    </form>

<?php } ?>

</body>

</html>

**stylez.css**

body {

    background-color: #f5f5dc;

    font-family: Arial, sans-serif;

    color: #333;

    text-align: center;

    padding: 20px;

}

h2 {

    color: #5a4e3c;

    margin-bottom: 20px;

}

.event {

    background-color: #fff8dc;

    border: 1px solid #ddd;

    margin: 20px auto;

    padding: 15px;

    width: 300px;

    border-radius: 6px;

}

.event a {

    display: inline-block;

    margin-top: 10px;

    background-color: #d2b48c;

    color: white;

    padding: 8px 12px;

    text-decoration: none;

    border-radius: 4px;

}

.event a:hover {

    background-color: #c2a77c;

}

form {

    background-color: #fffaf0;

    padding: 20px;

    margin: 30px auto;

    width: 280px;

    border: 1px solid #ddd;

    border-radius: 6px;

}

input,

select {

    width: 90%;

    padding: 6px;

    margin-bottom: 12px;

    border: 1px solid #ccc;

    border-radius: 4px;

}

button {

    background-color: #d2b48c;

    color: white;

    border: none;

    padding: 8px 16px;

    border-radius: 4px;

    cursor: pointer;

}

button:hover {

    background-color: #c2a77c;

}

# **What is solved?**

This project solves the problem of managing student participation in university clubs and events. In many universities, it can be difficult to organize events, track which students joined which sessions, and collect student information efficiently.

With this simple website:

1. Students can view all upcoming events without needing to register first.
2. When interested in an event, they can easily sign up with their basic information (name, email, and department).
3. Organizers can later use the database to see who registered for each session.
4. Events that are already expired are not shown, which keeps the interface clean and relevant.

# **Challenges & Lessons Learned**

Firstly I hate PHP very very very much especially when I used it at this project.

Secondly when I created this project I learned all the basics that I want to have a great foundation in Databases, and that is it.

Thanks :)

# **Application Sample**

**What users can do**

**A screenshot of a computer

AI-generated content may be incorrect.**

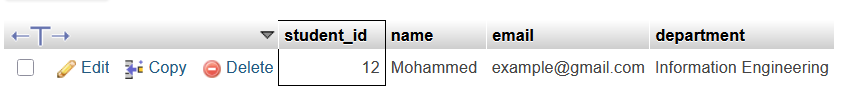
**A screenshot of a computer

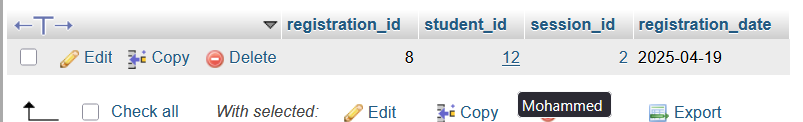
AI-generated content may be incorrect.**

**A close up of a text

AI-generated content may be incorrect.**

**What we will have**

****

****