National University of Computer & Emerging Sciences, Peshawar
Computer Science Department
Spring 2025, Lab Manual - 06

Course Code: CL-2005	Course : Database Systems Lab
Instructor :	Yasir Arfat

Contents:

Connectivity:

1. PHP with MYSQL.
2. JAVA with Oracle SQL.
3. C# with SQL Server.

PHP Connectivity with MYSQL Using XAMPP:

Xampp:

XAMPP is a widely used cross-platform web server that allows developers to create and test their applications on a local server. It was developed by Apache Friends and is an open-source package that simplifies web development.

The name **XAMPP** is an abbreviation:

X – Cross-Platform

A – Apache (Web Server)

M – MySQL (Database)

P – PHP (Scripting Language)

P – Perl (Programming Language)

XAMPP provides an easy-to-use distribution that includes essential components like the Apache server, MariaDB, PHP, and Perl. It also features a command-line executable, making it a powerful tool for web development and testing.

Download Xampp: https://www.apachefriends.org/download.html

MYSQL:

MySQL is a relational database management system (RDBMS) based on SQL (Structured Query Language). It is widely used for various applications, including data warehousing, e-commerce, and logging systems.

In this Lab, we are using MySQL, but it's important to note that there isn't much difference between MySQL and standard SQL. The fundamental concepts and queries remain the same, making it easy to adapt to other SQL-based databases in the future.

PHP with MySQL:

PHP 5 and later can work with a MySQL database using MYSQLi and PDO, both are Object Oriented:

MySQLi extension (the "i" stands for improved) will work only for MySQL databases.

PDO (PHP Data Objects) PDO will work on 12 different database systems.

MYSQLi is further bifurcated to facilitate users:

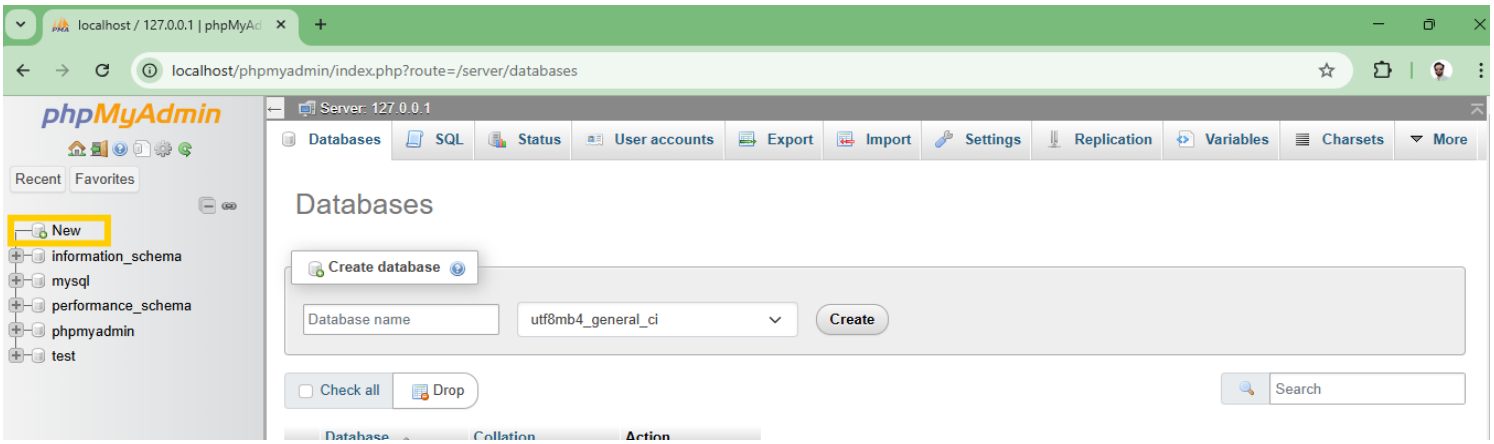
i) **MYSQL Object Oriented**
ii) **ii) MysQli (Procedural)**

In this lab activity, we will use the MYSQLi in a procedural manner.

Step-01) For this, First open Xampp control Panel and start the Apache and MYSQL servers.

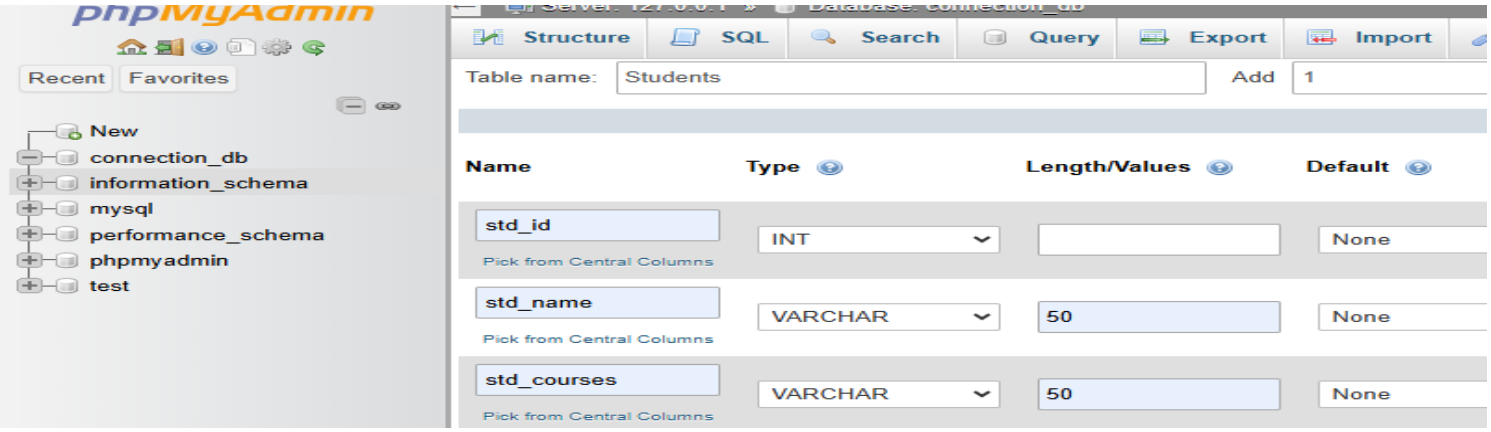
Step-02) Create a database on the phpmyadmin:

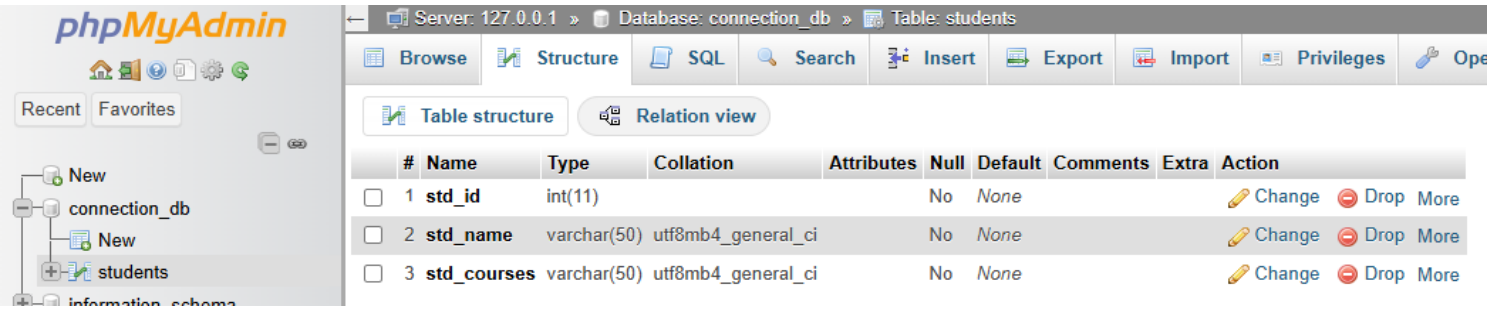
Open the browser and enter **: http://localhost/phpmyadmin/** and then create a database by clicking the NEW button on the left pane. Enter the name of db to be "**Connection_db".**



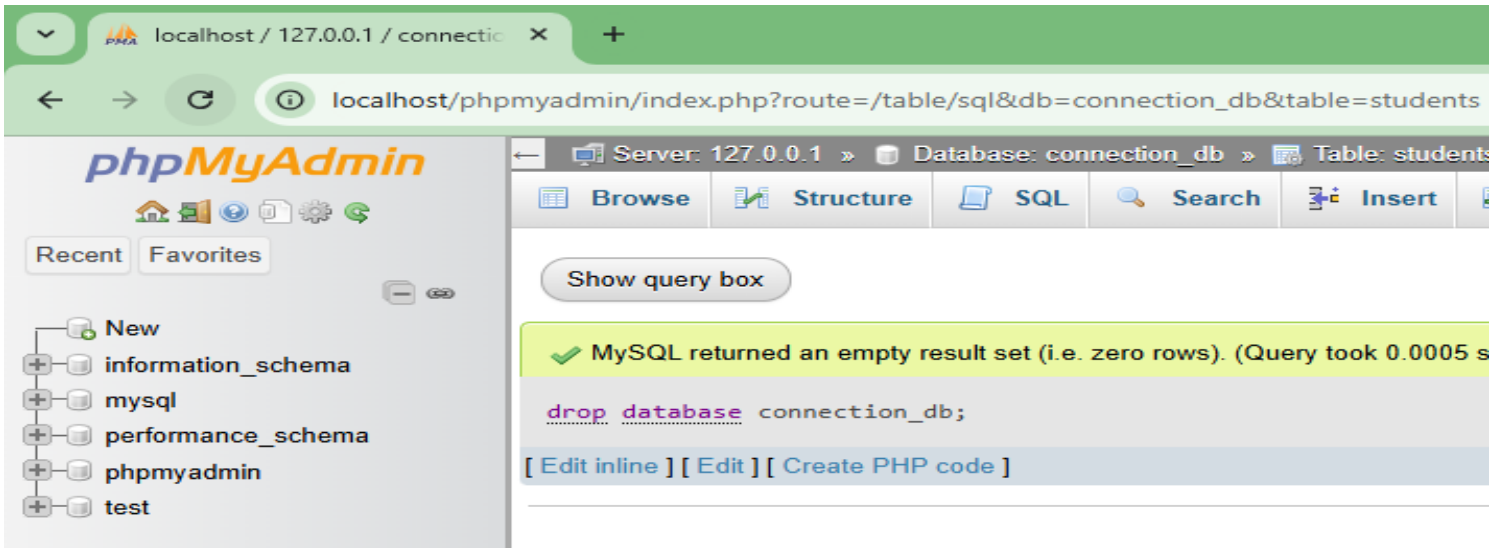
Step-03) Create a table inside the Connection_db named Students, with std_id,

std_name, std_courses as fields.





We have checked how we want to use phpMyAdmin but now let's do the same with php scripts and for that we will first remove the work we have done



Step-01: Creating a Connection to the Database:

bef

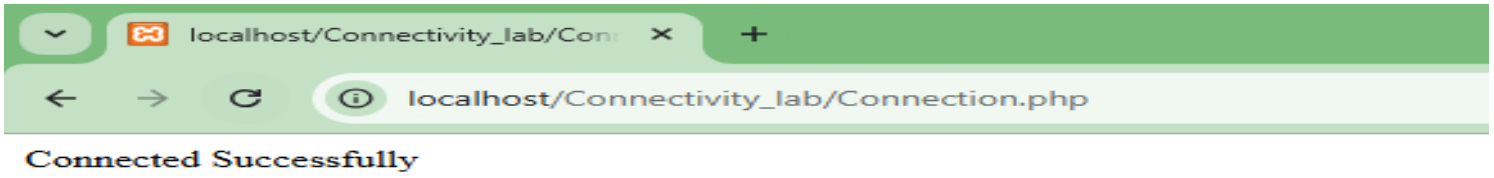
Create a folder Connectivity_lab in Xampp/htdocs folder

Create a new file Connection.php file inside Connectivity_lab folder in VS Code, containing Following code:

```
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";
    // Create connection
    $conn = mysqli_connect($servername, $username, $password);
    // Check connection
    if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
    }
    echo "Connected Successfully";
?>
```

Now run the above code in browser using

http://localhost/Connectivity_lab/Connection.php



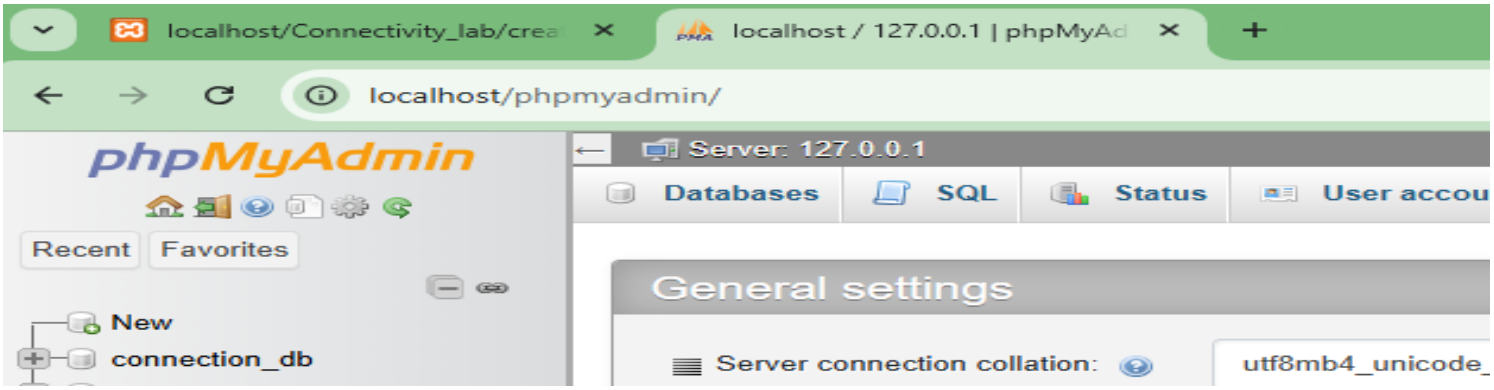
Creating a Database using PHP:

Lets create a new file Create_db.php to create database from php scripts

```
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";
    // Create connection
    $conn = mysqli_connect($servername, $username, $password);
    // Check connection
    if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
    }
    echo "Connected Successfully";
?>
```

Now run the above code in browser using
http://localhost/Connectivity_lab/create_db.php

After running the above code you can check phpMyAdmin and you will see



Creating a table:

Creating Tables: Two ways

1. Directly on phpMyAdmin (we already saw that in creating database in start)
2. Front end (Text Editor)

1. Direct on phpMyAdmin:

Follow these steps:

1. Select the database in which you want to create a table.
2. Fill out the table name and quantity of fields then click Go.
3. Give every field a proper name ,data type, size and Constraint (if any).
4. Click Go.

2. Front-End:

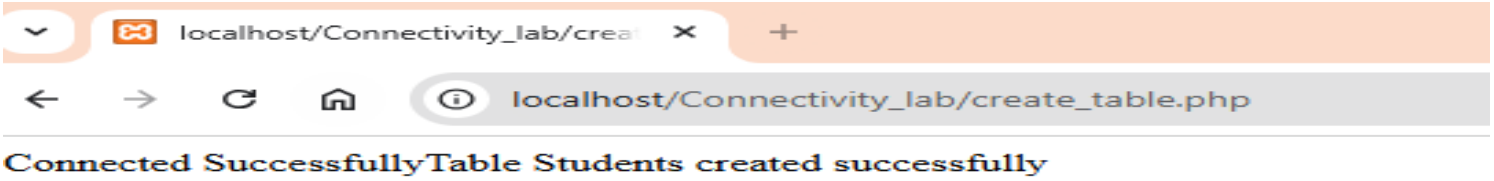
Lets create a new file Create_table.php to create database from php scripts

```
<?php
    include 'Connection.php';

    $db_name ="Connection_db";
    $conn = mysqli_connect($servername, $username, $password, $db_name);

    $sql = "CREATE TABLE Students
    ( std_id INT(11) PRIMARY KEY,
    std_name VARCHAR(30) NOT NULL,
    std_courses TEXT(30) NOT NULL)";
    if (mysqli_query($conn, $sql)) {
    echo "Table Students created successfully";
    } else {
```

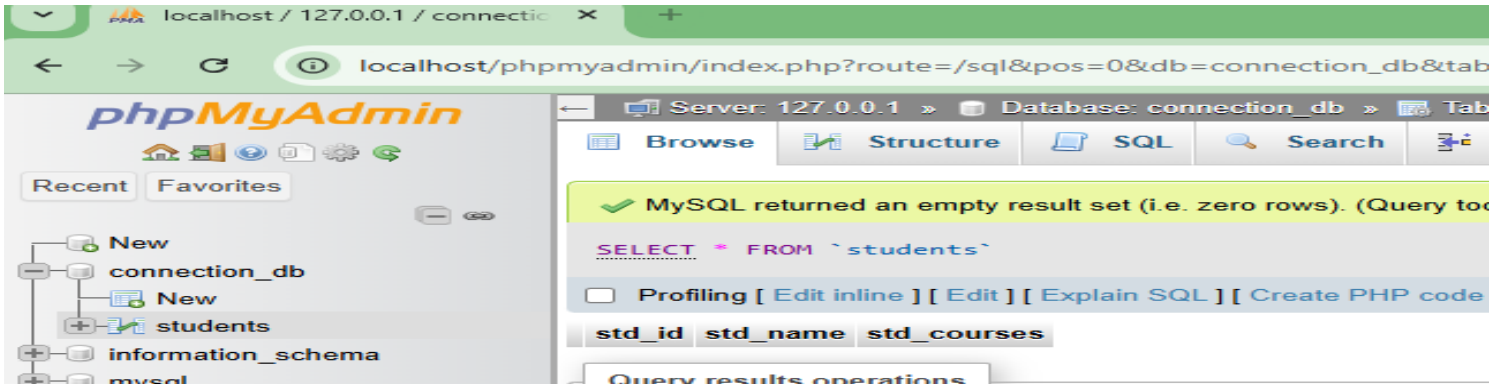
```
    echo "Error creating table: " . mysqli_error($conn);

}
?>
```

Run the above code using

http://localhost/Connectivity_lab/create_table.php



You can verify it by checking phpMyAdmin



After the database table is created , we may now add data into it:

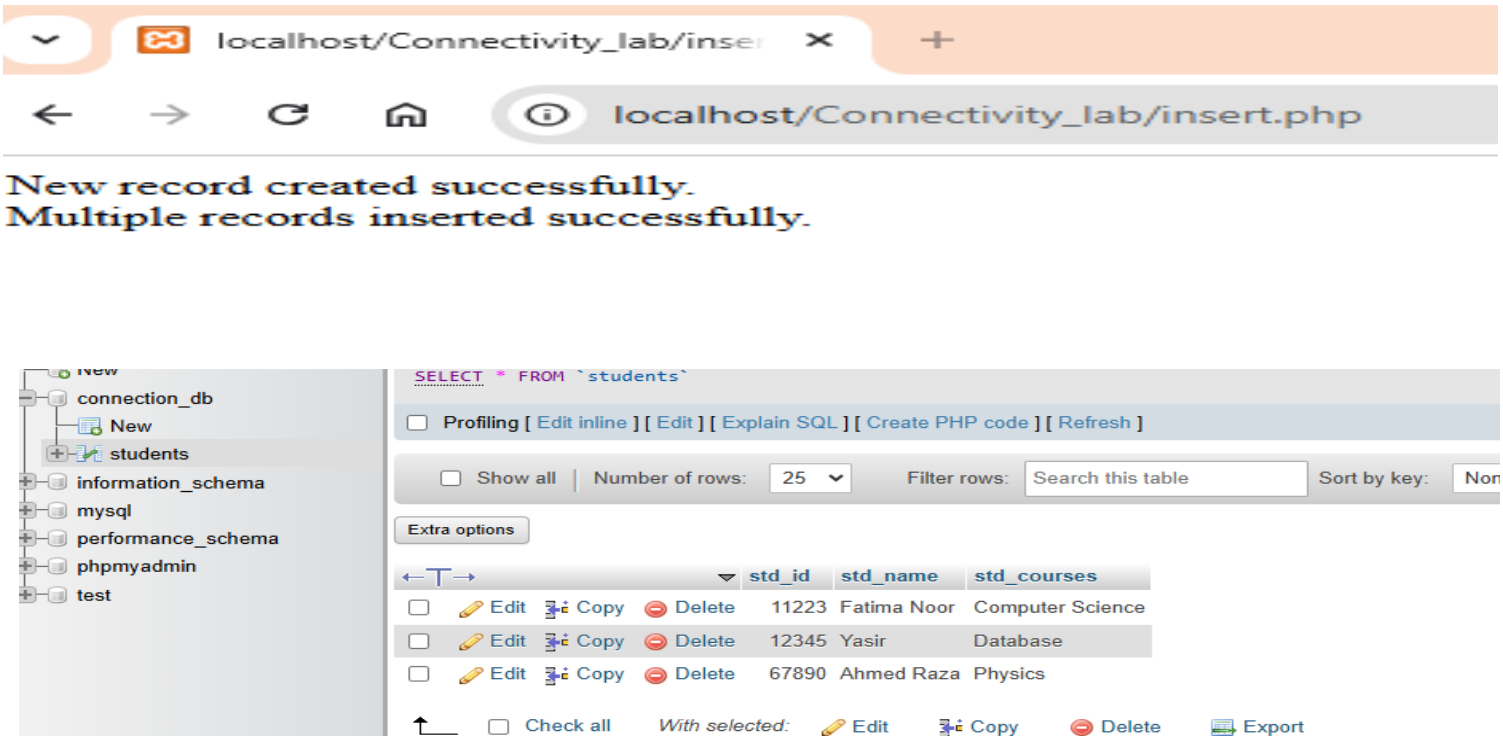
Inserting Data into Table:

Lets create another file insert.php and the following code

```
<?php
// Database connection
$conn = mysqli_connect("localhost", "root", "", "Connection_db");

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
```

```php
        // Insert single record
        $sql1 = "INSERT INTO Students (std_id, std_name, std_courses) VALUES
        (12345, 'Yasir', 'Database')";

        if (mysqli_query($conn, $sql1)) {
            echo "New record created successfully.<br>";
        } else {
            echo "Error: " . mysqli_error($conn) . "<br>";
        }

        // Insert multiple records
        $sql2 = "INSERT INTO Students (std_id, std_name, std_courses) VALUES
        (67890, 'Ahmed Raza', 'Physics'),
        (11223, 'Fatima Noor', 'Computer Science')";

        if (mysqli_query($conn, $sql2)) {
            echo "Multiple records inserted successfully.";
        } else {
            echo "Error: " . mysqli_error($conn);
        }

        // Close connection
        mysqli_close($conn);
?>
```

Run the above code using http://localhost/Connectivity_lab/insert.php



Prepared Statements and Bound Parameters:

A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency.

Prepared statements basically work like this:

Prepare: An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?"). Example: INSERT INTO Student VALUES(?, ?, ?)

The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it

Execute: At a later time, the application binds the values to the parameters, andthe database executes the statement. The application may execute the statement as many times as it wants with different values

Advantages:

Compared to executing SQL statements directly, prepared statements have three main

advantages:

Prepared statements reduces parsing time as the preparation on the query is done only once (although the statement is executed multiple times)

Bound parameters minimize bandwidth to the server as you need send only the parameters each time, and not the whole query

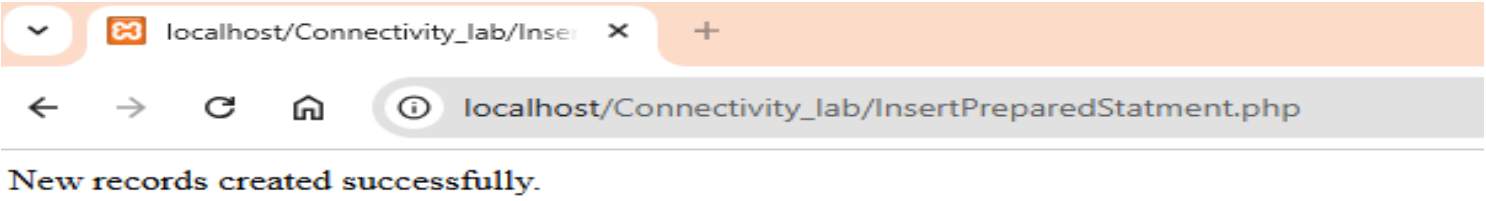
Prepared statements are very useful against SQL injections, because parameter values, which are transmitted later using a different protocol, neednot be correctly escaped. If the original statement template is not derivedfrom external input, SQL injection cannot occur.

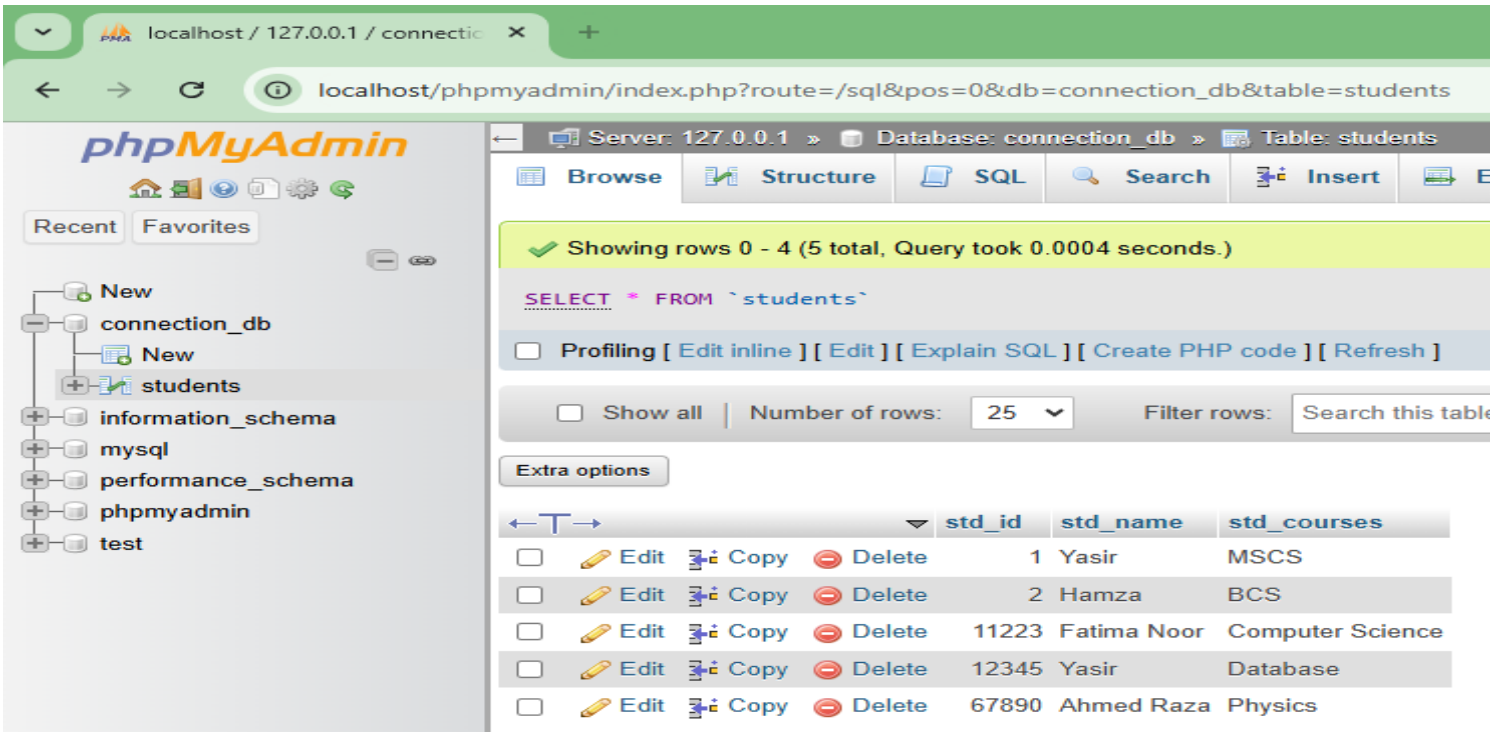
Now Lets create a new file InsertPreparedStatment.php

```php
<?php

$conn = mysqli_connect("localhost", "root", "", "Connection_db");

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Prepare statement
$stmt = mysqli_prepare($conn, "INSERT INTO Students (std_id, std_name, std_courses)
VALUES (?, ?, ?)");
```

```php
// Bind parameters
mysqli_stmt_bind_param($stmt, "iss", $id, $name, $course);

// Insert first record
$id = 1;
$name = "Yasir";
$course = "MSCS";
mysqli_stmt_execute($stmt);

// Insert second record
$id = 2;
$name = "Hamza";
$course = "BCS";
mysqli_stmt_execute($stmt);

echo "New records created successfully.";

// Close statement and connection
mysqli_stmt_close($stmt);
mysqli_close($conn);
?>
```

Now run the above code using http://localhost/Connectivity_lab/InsertPreparedStatment.php



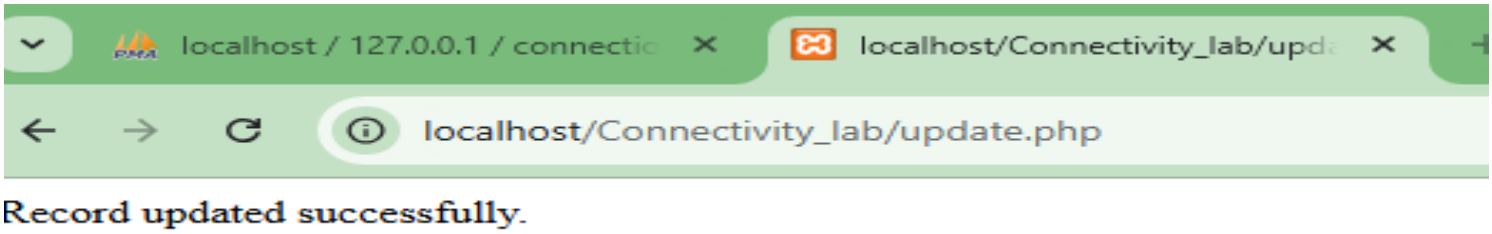


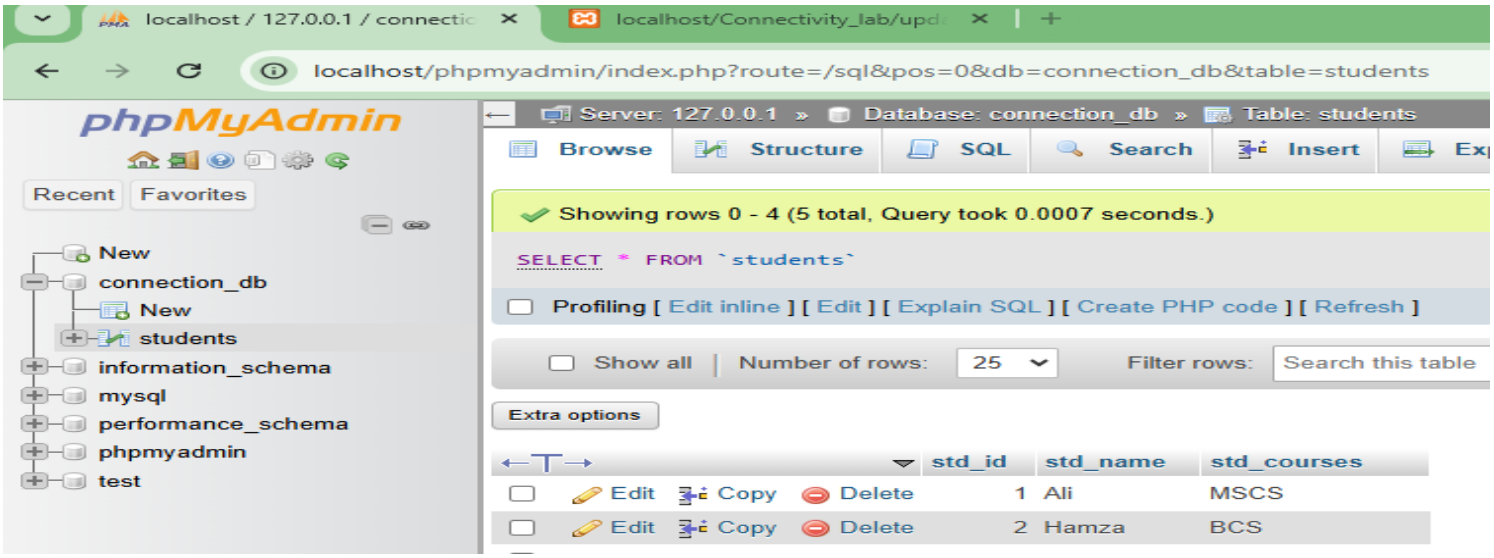
Updating Data into Table

lets create new file name update.php and add following code

```php
<?php

$conn = mysqli_connect("localhost", "root", "", "Connection_db");

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Update query
$sql = "UPDATE Students SET std_name='Ali' WHERE std_id=1";

if (mysqli_query($conn, $sql)) {
    echo "Record updated successfully.";
} else {
    echo "Error updating record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Run the above code using http://localhost/Connectivity_lab/update.php



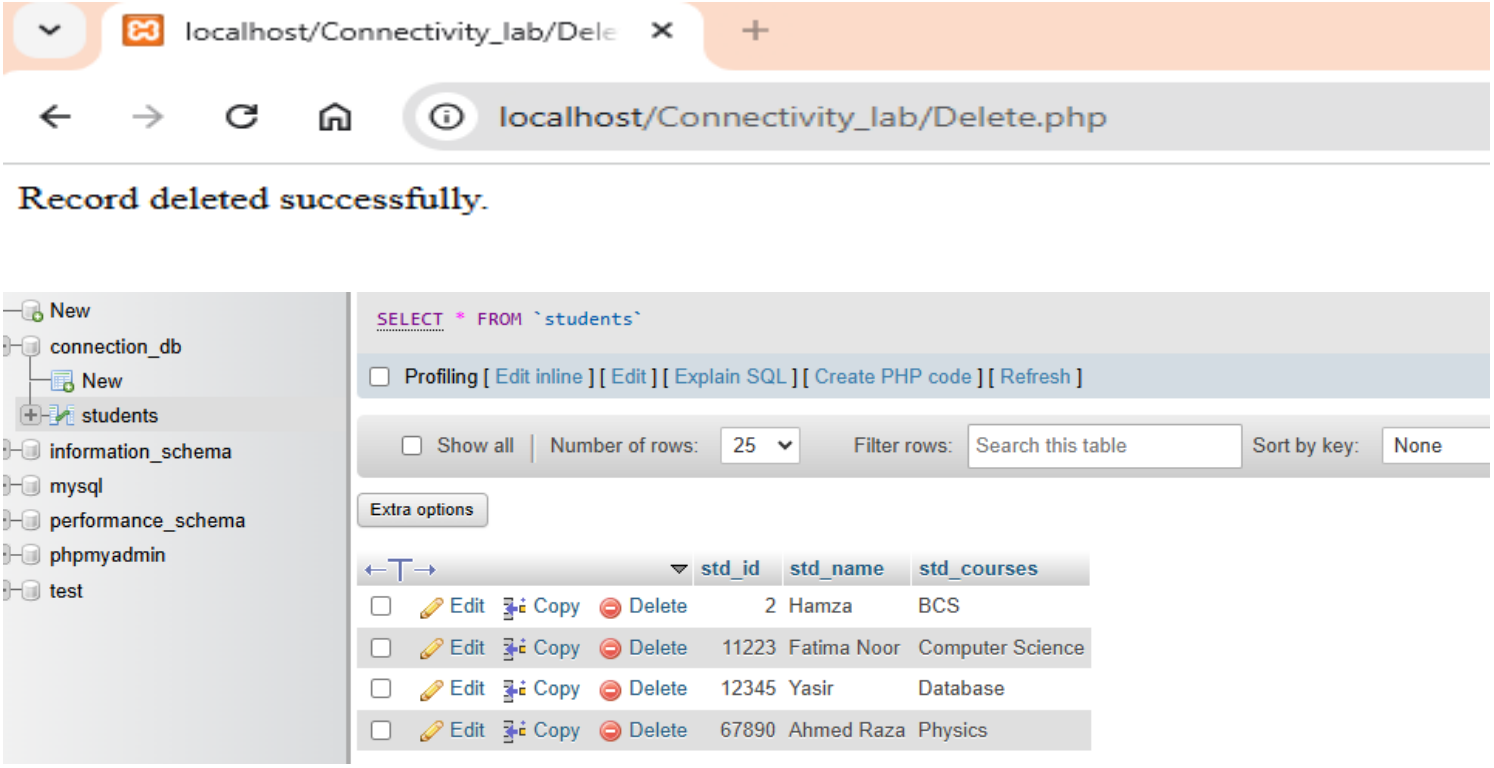


Deleting Data from table:

lets create new file name delete.php and add following code

```
<?php

$sql = "DELETE FROM Students WHERE id=1";
if (mysqli_query($conn, $sql)) {
echo "Record deleted successfully";
} else {
echo "Error deleting record: " . mysqli_error($conn);
}

?>
```

Now run the above code using http://localhost/Connectivity_lab/Delete.php



Complete Crud Operation with Php

Now we have enough understanding of how PHP and MySQL work together. So, now let's create a CRUD operation for the students table.

CRUD stands for:

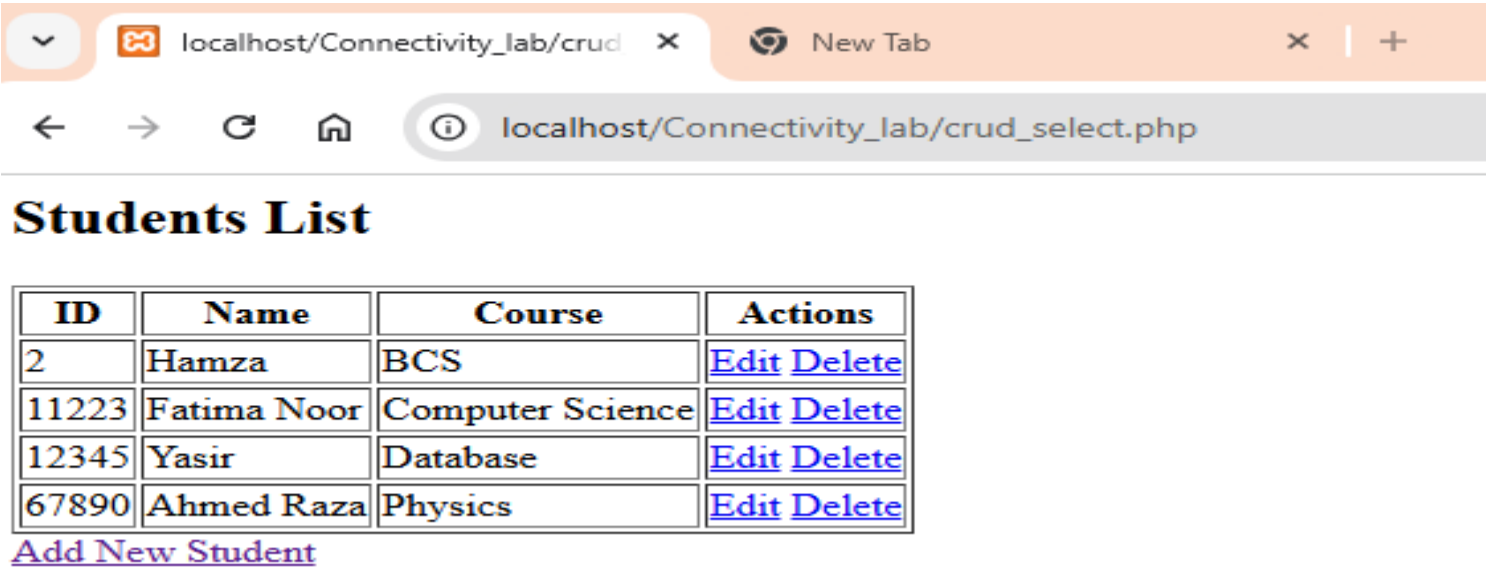
- Create – Insert new student records.
- Read – Display student records.
- Update – Edit student records.
- Delete – Remove student records.

Now lets start by creating our first file crud_select.php and add the following code

```
<?php
$conn = mysqli_connect("localhost", "root", "", "Connection_db");

$result = mysqli_query($conn, "SELECT * FROM Students");
?>

<h2>Students List</h2>
```

```
<table border="1">
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Course</th>
        <th>Actions</th>
    </tr>
    <?php while ($row = mysqli_fetch_assoc($result)) { ?>
        <tr>
            <td><?= $row['std_id'] ?></td>
            <td><?= $row['std_name'] ?></td>
            <td><?= $row['std_courses'] ?></td>
            <td>
                <a href="crud_update.php?id=<?= $row['std_id'] ?>">Edit</a>
                <a href="crud_delete.php?id=<?= $row['std_id'] ?>" onclick="return confirm('Are
you sure?')">Delete</a>
            </td>
        </tr>
    <?php } ?>
</table>
<a href="crud_insert.php">Add New Student</a>
```

Now run the following code using http://localhost/Connectivity_lab/crud_select.php

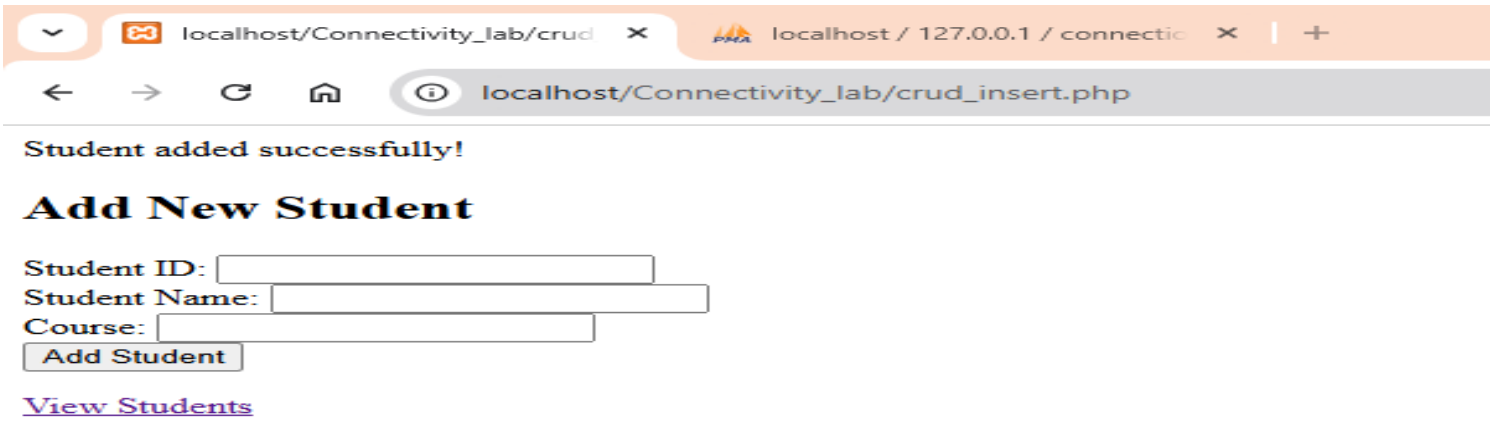


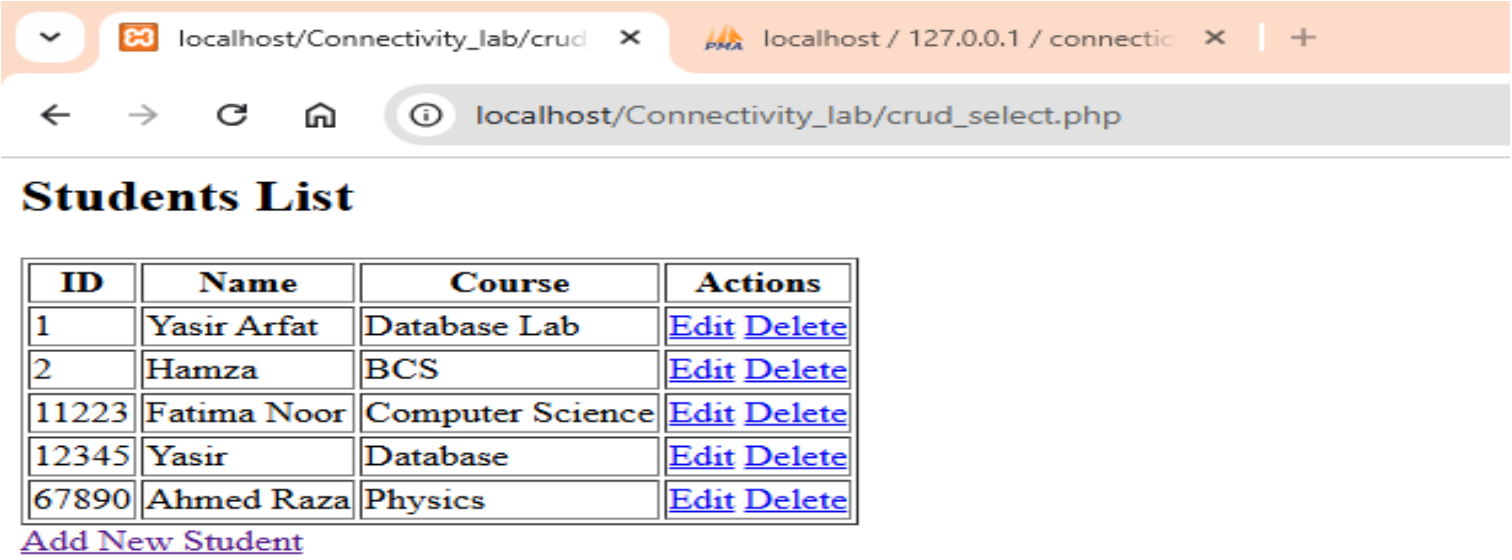
Now lets create crud_insert and add the following code

```
<?php
$conn = mysqli_connect("localhost", "root", "", "Connection_db");


if (isset($_POST['submit'])) {
    $id = $_POST['std_id'];
    $name = $_POST['std_name'];
    $course = $_POST['std_courses'];
```

```
    $sql = "INSERT INTO Students (std_id, std_name, std_courses) VALUES (?, ?, ?)";
    $stmt = mysqli_prepare($conn, $sql);
    mysqli_stmt_bind_param($stmt, "iss", $id, $name, $course);

    if (mysqli_stmt_execute($stmt)) {
        echo "Student added successfully!";
    } else {
        echo "Error: " . mysqli_error($conn);
    }

    mysqli_stmt_close($stmt);
}
?>

<h2>Add New Student</h2>
<form method="POST" >
    Student ID: <input type="number" name="std_id" required><br>
    Student Name: <input type="text" name="std_name" required><br>
    Course: <input type="text" name="std_courses" required><br>
    <button type="submit" name="submit">Add Student</button>
</form>
<a href="crud_select.php">View Students</a>
```

Now run the above code using http://localhost/Connectivity_lab/crud_insert.php

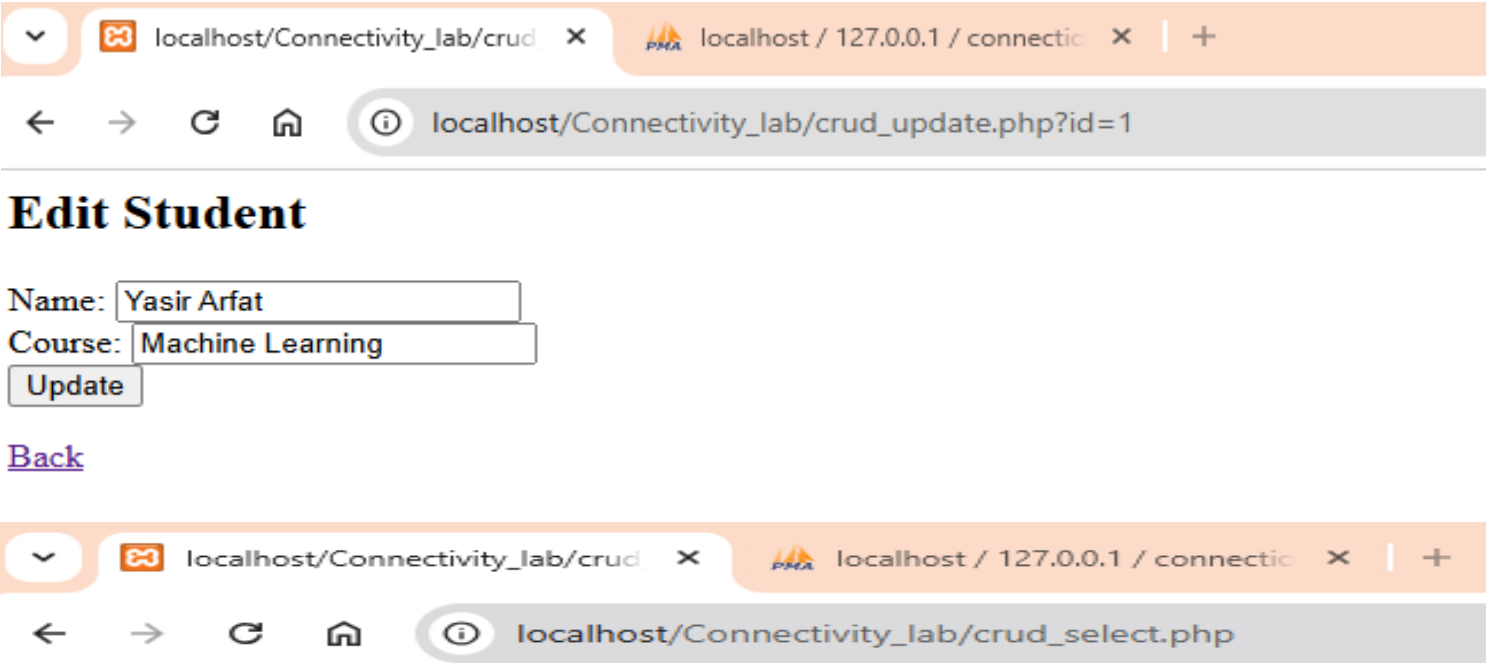


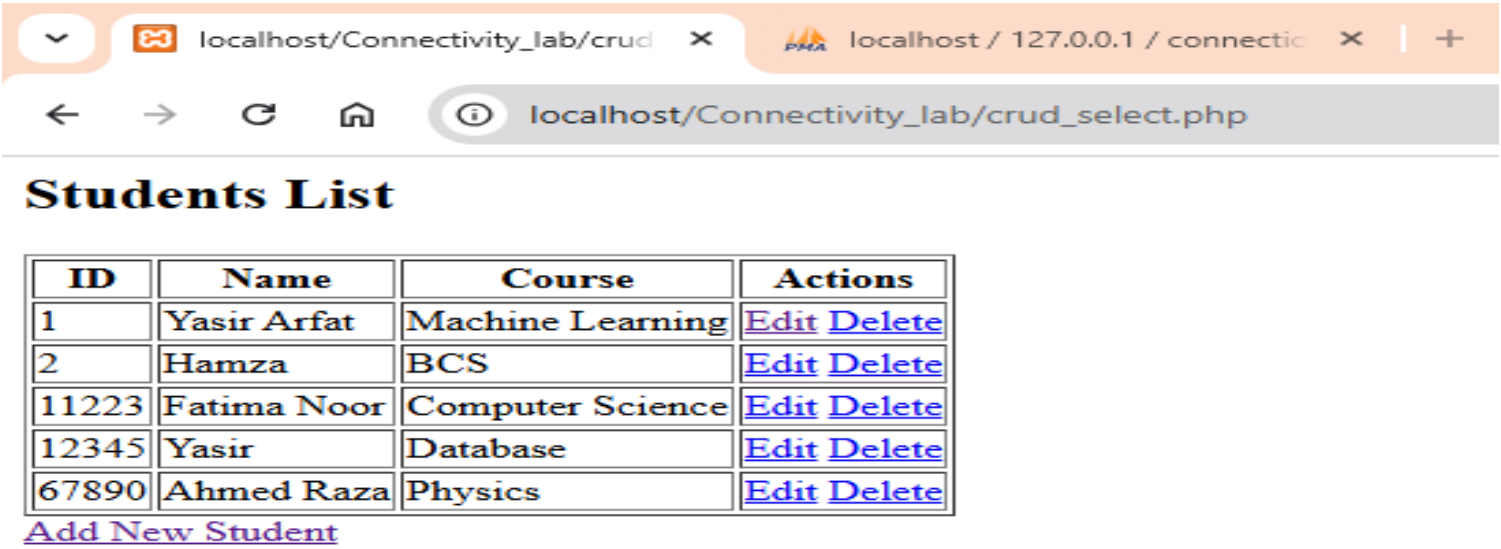


Now lets create new file crud_update.php file and add the following code

```php
<?php
$conn = mysqli_connect("localhost", "root", "", "Connection_db");

if (isset($_GET['id'])) {
    $id = $_GET['id'];
    $result = mysqli_query($conn, "SELECT * FROM Students WHERE std_id=$id");
    $row = mysqli_fetch_assoc($result);
}

if (isset($_POST['update'])) {
    $id = $_POST['std_id'];
    $name = $_POST['std_name'];
    $course = $_POST['std_courses'];

    $sql = "UPDATE Students SET std_name=?, std_courses=? WHERE std_id=?";
    $stmt = mysqli_prepare($conn, $sql);
    mysqli_stmt_bind_param($stmt, "ssi", $name, $course, $id);

    if (mysqli_stmt_execute($stmt)) {
        echo "Student updated successfully!";
    } else {
        echo "Error: " . mysqli_error($conn);
    }

    mysqli_stmt_close($stmt);
    header("Location: crud_select.php");
}
?>
```

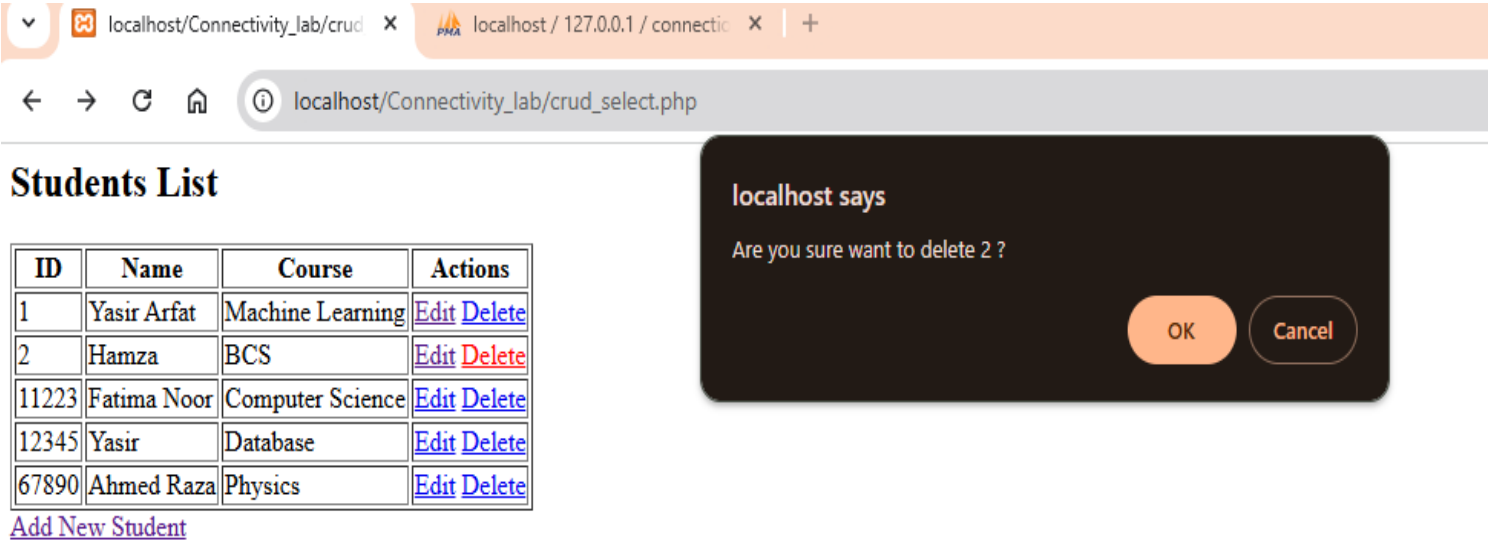
```php
<h2>Edit Student</h2>
<form method="POST">
    <input type="hidden" name="std_id" value="<?= $row['std_id'] ?>">
    Name: <input type="text" name="std_name" value="<?= $row['std_name'] ?>" required><br>
    Course: <input type="text" name="std_courses" value="<?= $row['std_courses'] ?>"
required><br>
    <button type="submit" name="update">Update</button>
</form>
<a href="crud_select.php">Back</a>
```

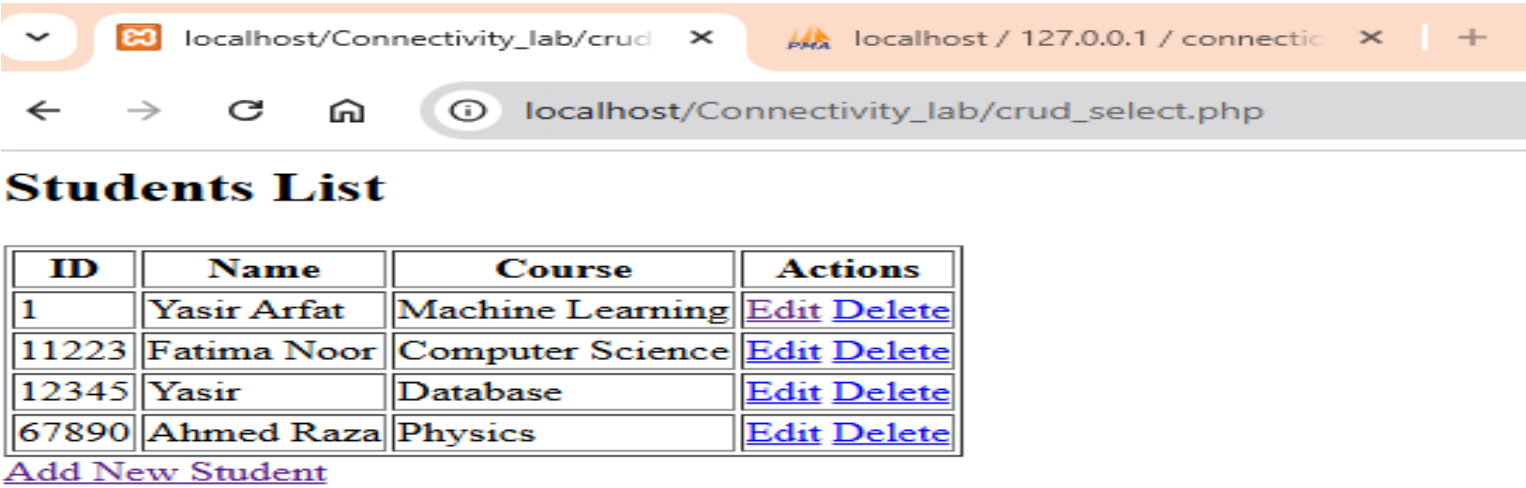




Now lets crate a crud_delete.php

```php
<?php
$conn = mysqli_connect("localhost", "root", "", "Connection_db");

if (isset($_GET['id'])) {
    $id = $_GET['id'];
    $sql = "DELETE FROM Students WHERE std_id=?";
    $stmt = mysqli_prepare($conn, $sql);
    mysqli_stmt_bind_param($stmt, "i", $id);

    if (mysqli_stmt_execute($stmt)) {
        echo "Student deleted successfully!";
    } else {
        echo "Error: " . mysqli_error($conn);
    }

    mysqli_stmt_close($stmt);
    header("Location: crud_select.php");
}
?>
```





Crud in Java

For Crud in Java Kindly See the second Attached File

C# Connectivity with SQL Server:

C# and .Net can work with a majority of databases, the most common being Oracle and Microsoft SQL Server. But with every database, the logic behind working with all of them is mostly the same.

In this lab ,we will look at working with Microsoft SQL Server as our database. For learning purposes, you can download and use the Microsoft SQL Server Express

Edition, which is a free database software provided by Microsoft from the following link:

https://www.microsoft.com/en-pk/download/details.aspx?id=42299 For

Visual Studio Download:

https://visualstudio.microsoft.com/vs/community/

In working with databases, the following are the concepts which are common to all databases.

Connection – To work with the data in a database, the first obvious step is the connection. The connection to a database normally consists of the below- mentioned parameters.

SQL Command in C#

SqlCommand in C# allow the user to query and send the commands to the database. SQL command is specified by the SQL connection object. Two methods are used, ExecuteReader method for results of query and ExecuteNonQuery for insert, Update, and delete commands. It is the method that is best for the different commands.

How to connect C# to Database

Let"s now look at the code, which needs to be kept in place to create a connection to a database. In our example, we will connect to a database which has the name of Connection_db.

We will see a simple Windows forms application to work with databases. We will have a simple button called "Test" which will be used to connect/(Insert) to the database.

So let"s follow the below steps to achieve this

Step 1) The first step involves the creation of a new project in Visual Studio. After launching Visual Studio, you need to choose the menu option New->Project.

C# SQL SERVER Database

Step 2) The next step is to choose the project type as a Windows Forms application. Here, we also need to mention the name and location of our project.

In the project dialog box, we can see various options for creating different types of projects in Visual Studio. Select Windows Forms Application.

We then give a name for the application which in our case is “WindowsFormApp2”.

Finally, we click the „OK“ button to let Visual Studio to create our project.

Step 3) Now add some buttons and other widgets from the toolbox to the Windows form. Put the text property of the Button as "Test". This is how it will look like:

Before this, Add the following package to your import list, to import all the functionalities of SqlClient.

```
using System.Data.SqlClient;
```

Step 4) Now save the form and then in the Event handler for test button , paste the following Code:

Add teh following code in the Form Event Handler:

```
public string constring = "Data Source=HP-PC;Initial Catalog=Connection_db;Integrated Security=True";

SqlConnection conn = new SqlConnection(constring);

conn.Open();

if(conn.State==System.Data.ConnectionState.Open)

        {

            string q = "Insert into Test(ID, NAME) values ('"+txtID.Text.ToString()+"','"+txtName.Text.ToString()+"')";

            SqlCommand cmd = new SqlCommand(q, conn);

          cmd.ExecuteNonQuery();

          MessageBox.Show("Inserted Successfully!");

        }
```

```
conn.Close();
```

Code Explanation:-

The first step is to create variables, which will be used to create the connection string and the connection to the SQL Server database.

The next step is to create the connection string. The connecting string needs to be specified correctly for C# to understand the connection string. The connection string consists of the following parts

Data Source – This is the name of the server on which the database resides. In our case, it resides on a machine called "HP-PC"

The Initial Catalog is used to specify the name of the database : Initial Catalog=Connection_db

Next, we assign the connecting string to the variable conn. The variable conn, which is of type SqlConnection is used to establish the connection to the database.

Next, we use the Open method of the conn variable to open a connection to the database.

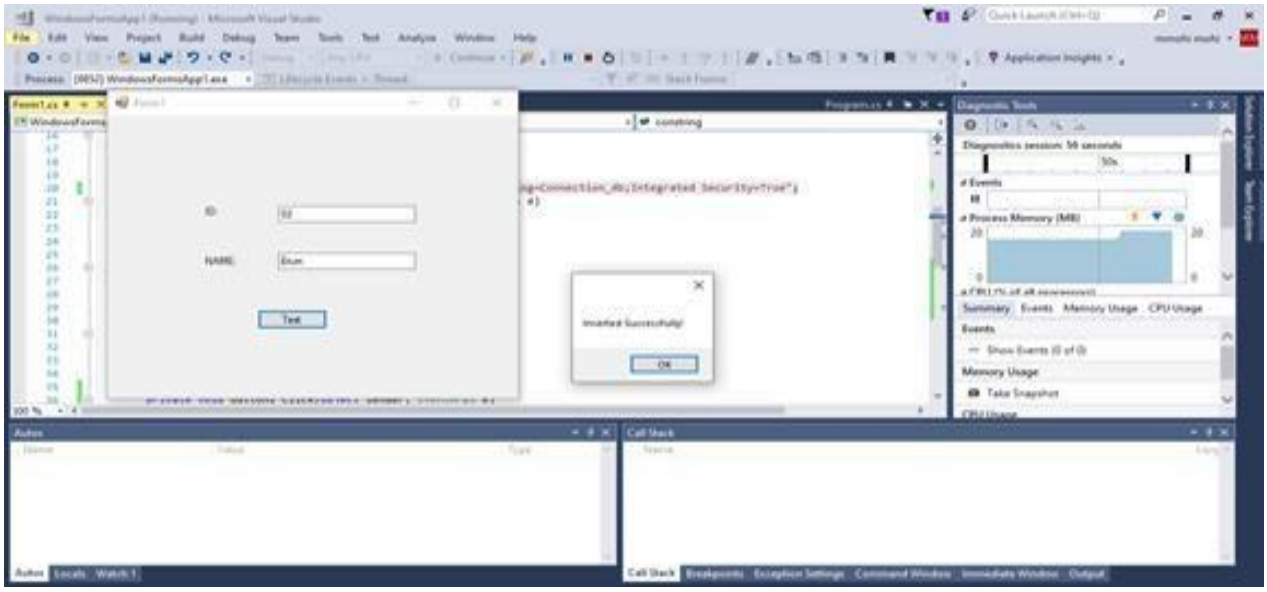
Then we insert the data using the windows form, and click insert.

If the process of insertion is successful, the message of successful insertion will be displayed.

Once the operation is completed successfully, we then close the connection to the database. It is always a good practice to close the connection to the database if nothing else is required to be done on the database.

When the above code is set, and the project is executed using Visual Studio, you will get the below output. Once the form is displayed, click the Test button.

Once we click Test, The output Should be:


NAME
Test
OK
Diagnostic Tools
Summary
Events
Memory Usage
CPU Usage
Autos
Call Stack