**Lab 6**

**Objectives:**

The objective of this lab is to demonstrate the use of PHP for implementing core CRUD (Create, Read, Update, Delete) operations with a MySQL database. It aims to enhance practical knowledge of database interaction through forms and queries. The lab also focuses on using SQL for performing aggregate functions, ordering data, and applying joins for relational data access. Another key objective is to introduce Object-Oriented Programming (OOP) concepts in PHP such as classes, inheritance, constructors, destructors, access modifiers, and static members. Students will learn how to design reusable and modular code using OOP principles. The lab encourages building dynamic web pages that interact with backend databases effectively. Overall, it helps in understanding full-stack development components involving PHP and MySQL integration.

**1. Illustrate CRUD operation using php.**

1. **Create a new table named students in your database with columns: std\_id, Student\_name, RollNumber, marks\_obtained, result (Pass/Fail).**

**Source Code:**

**Query:**

USE lab6;

CREATE TABLE students (

std\_id INT AUTO\_INCREMENT PRIMARY KEY,

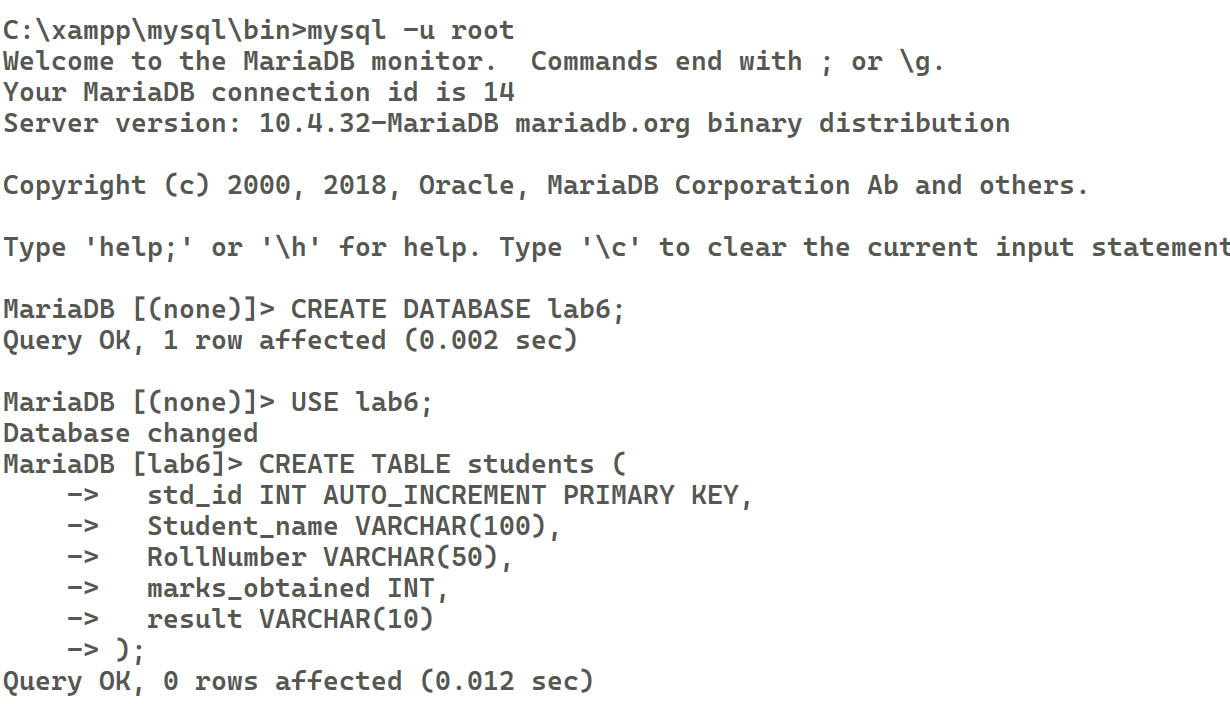
Student\_name VARCHAR(100),

RollNumber VARCHAR(50),

marks\_obtained INT,

result VARCHAR(10)

);



1. **Insert details of 5 students.**

**Query:**

INSERT INTO students (Student\_name, RollNumber, marks\_obtained, result)

VALUES

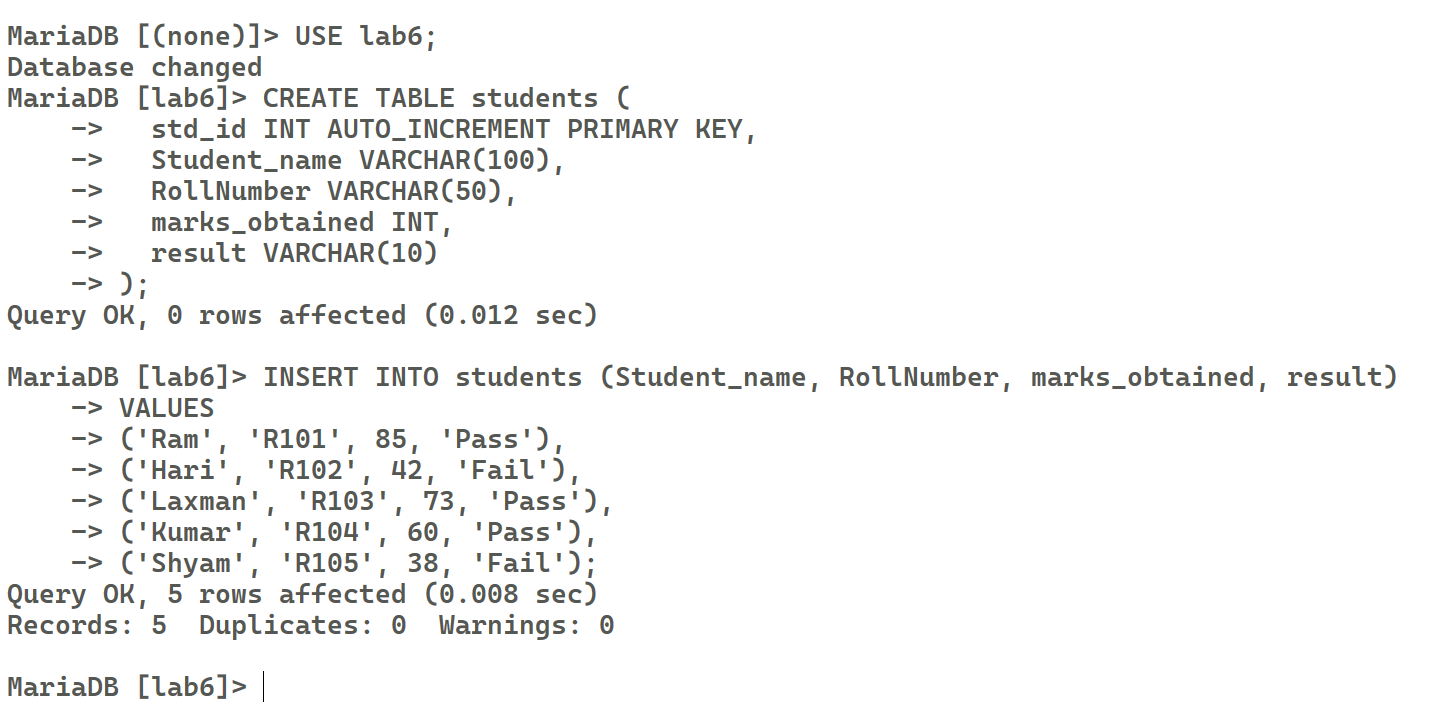
('Ram', 'R101', 85, 'Pass'),

('Hari', 'R102', 42, 'Fail'),

('Laxman', 'R103', 73, 'Pass'),

('Kumar', 'R104', 60, 'Pass'),

('Shyam', 'R105', 38, 'Fail');



1. **Display the result of 5 students in a table with buttons to edit and delete record**

**Source Code:**

**db.php**

<?php

$conn = new mysqli("localhost", "root", "", "lab6");

if ($conn->connect\_error) {

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

}

?>

**display.php**

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

<!DOCTYPE *html*>

<html>

<head>

    <title>Students Result</title>

</head>

<body>

    <h2>Students List</h2>

    <table border="1" *cellpadding*="8">

        <tr>

            <th>ID</th>

            <th>Name</th>

            <th>Roll Number</th>

            <th>Marks</th>

            <th>Result</th>

            <th>Actions</th>

        </tr>

        <?php

        $result = $conn->query("SELECT \* FROM students LIMIT 5");

        while ($row = $result->fetch\_assoc()) {

            echo "<tr>

                    <td>{$row['std\_id']}</td>

                    <td>{$row['Student\_name']}</td>

                    <td>{$row['RollNumber']}</td>

                    <td>{$row['marks\_obtained']}</td>

                    <td>{$row['result']}</td>

                    <td>

                        <a href='edit.php?id={$row['std\_id']}'>Edit</a> |

                        <a href='delete.php?id={$row['std\_id']}'

                        onclick=\"return confirm('Are you sure?');\">Delete</a>

                    </td>

                </tr>";

        }

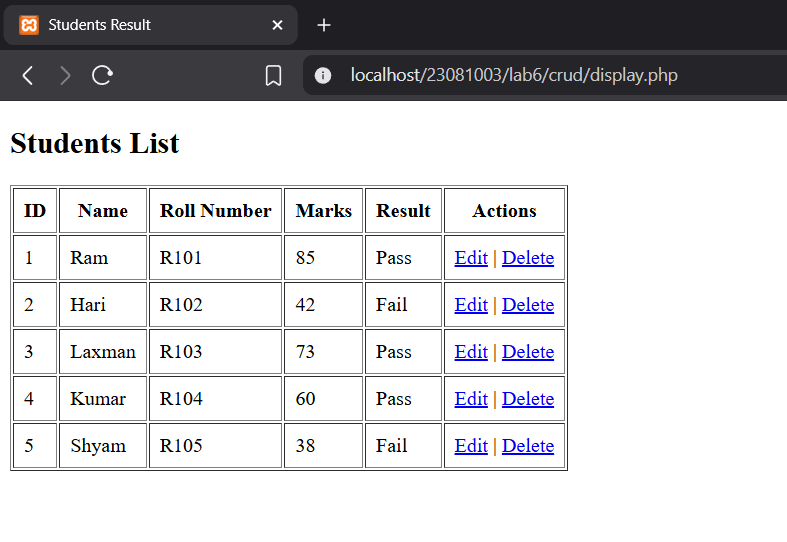
        ?>

    </table>

</body>

</html>

Output:



1. **Edit the marks of one of the students**

**Source Code:**

**edit.php**

<?php

include('db.php');

$result = $conn->query("SELECT \* FROM students");

?>

<table border="1">

    <tr>

        <th>ID</th>

        <th>Name</th>

        <th>Roll Number</th>

        <th>Marks</th>

        <th>Result</th>

        <th>Actions</th>

    </tr>

    <?php while($row = $result->fetch\_assoc()) { ?>

        <tr>

            <td><?= $row['std\_id'] ?></td>

            <td><?= $row['Student\_name'] ?></td>

            <td><?= $row['RollNumber'] ?></td>

            <td><?= $row['marks\_obtained'] ?></td>

            <td><?= $row['result'] ?></td>

            <td>

                <a *href*="edit.php?id=<?= $row['std\_id'] ?>">Edit</a> |

                <a *href*="delete.php?id=<?= $row['std\_id'] ?>" *onclick*="*return* confirm('Are you sure?')">Delete</a>

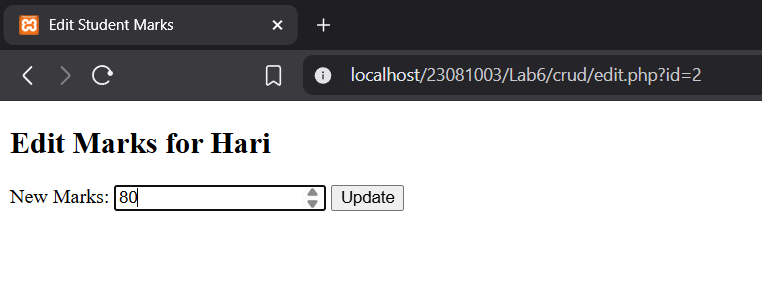
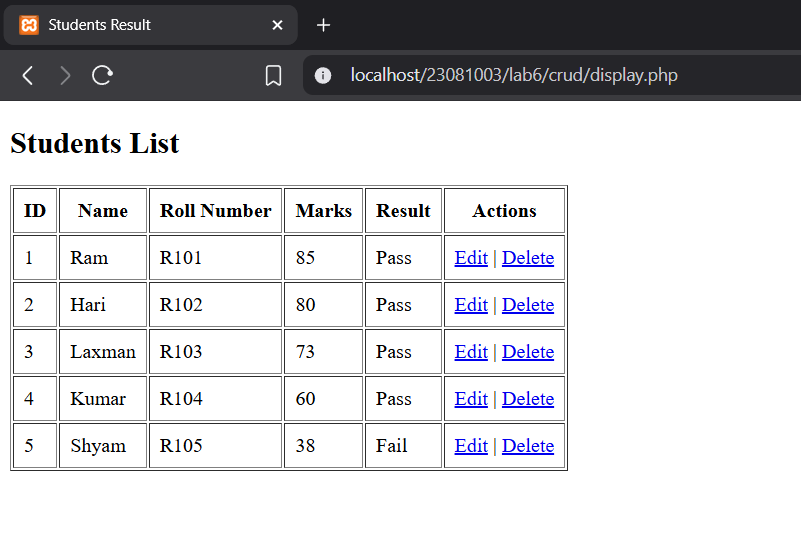
            </td>

        </tr>

    <?php } ?>

</table>

Output:



1. **Delete the record of one student**

**Source Code:**

**delete.php**

<?php

include('db.php');

// *Check if the ID is provided in the URL*

if (isset($\_GET['id'])) {

    $id = $\_GET['id'];

    // *Delete the record*

    $conn->query("DELETE FROM students WHERE std\_id = $id");

    // *Redirect to the main page after deletion*

    header("Location: edit.php");

    exit();

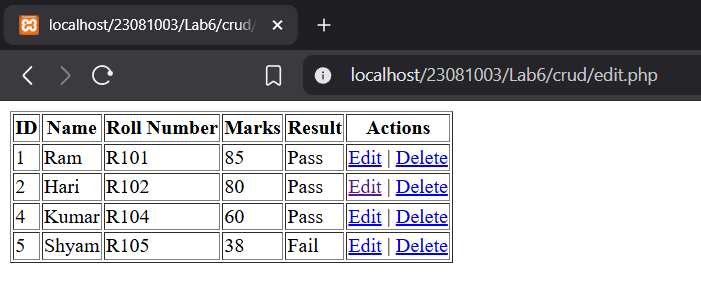
} else {

    die("Student ID not provided.");

}

Output:

http://localhost/23081003/Lab6/crud/delete.php?id=3

****

**2. In the above table:**

**a. Find the name of student with the highest marks**

**b. Find the name of student with lowest marks**

**c. Display the number of total students**

**d. Also show average marks obtained by the class**

**Source Code:**

**q2.php**

<?php

include('db.php');

$a = $conn->query("SELECT Student\_name, marks\_obtained FROM students ORDER BY marks\_obtained DESC LIMIT 1")->fetch\_assoc();

$b = $conn->query("SELECT Student\_name, marks\_obtained FROM students ORDER BY marks\_obtained ASC LIMIT 1")->fetch\_assoc();

$c = $conn->query("SELECT COUNT(\*) as total FROM students")->fetch\_assoc();

$d = $conn->query("SELECT AVG(marks\_obtained) as avg FROM students")->fetch\_assoc();

echo "a. Student with Highest Marks: {$a['Student\_name']} ({$a['marks\_obtained']})<br>";

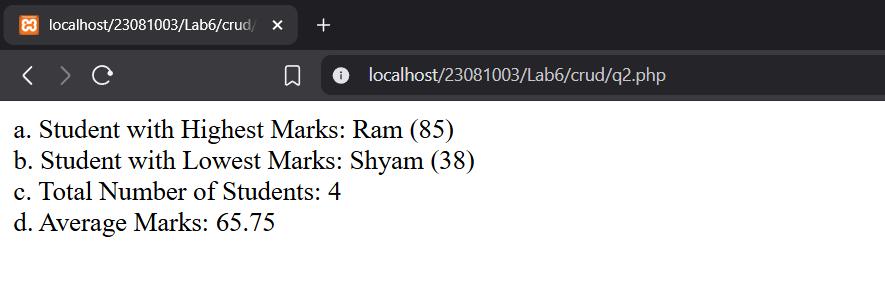
echo "b. Student with Lowest Marks: {$b['Student\_name']} ({$b['marks\_obtained']})<br>";

echo "c. Total Number of Students: {$c['total']}<br>";

echo "d. Average Marks: " . round($d['avg'], 2) . "<br>";

?>

Output:



**3. Display the result in ascending order and also find the number of students that have passed and failed using order by.**

**Source Code:**

**q3.php**

<?php

include('db.php');

// *Display all students in ascending order of marks*

echo "Students in Ascending Order of Marks:<br>";

$result = $conn->query("SELECT \* FROM students ORDER BY marks\_obtained ASC");

while ($row = $result->fetch\_assoc()) {

    echo "{$row['Student\_name']} - {$row['marks\_obtained']} - {$row['result']}<br>";

}

// *Count students who passed and failed*

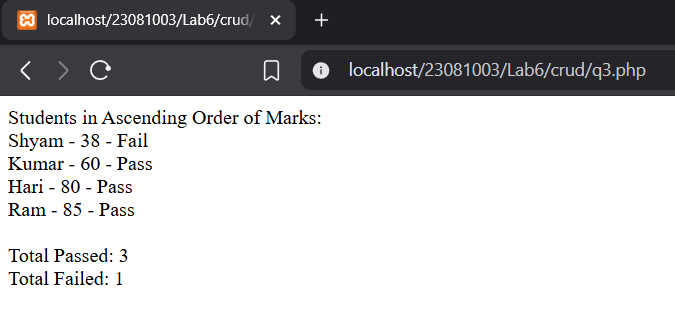
$pass = $conn->query("SELECT COUNT(\*) as count FROM students WHERE result='Pass'")->fetch\_assoc();

$fail = $conn->query("SELECT COUNT(\*) as count FROM students WHERE result='Fail'")->fetch\_assoc();

echo "<br>Total Passed: {$pass['count']}<br>";

echo "Total Failed: {$fail['count']}<br>";

?>

****Output:

**4. Demonstrate the role of joins in MYSQL. Create another table with student address and parents name, mobile phone, email. Display the parents name and email of the students who have failed the exam.**

**Source Code:**

**Query:**

USE lab6;

CREATE TABLE student\_info (

std\_id INT,

address VARCHAR(255),

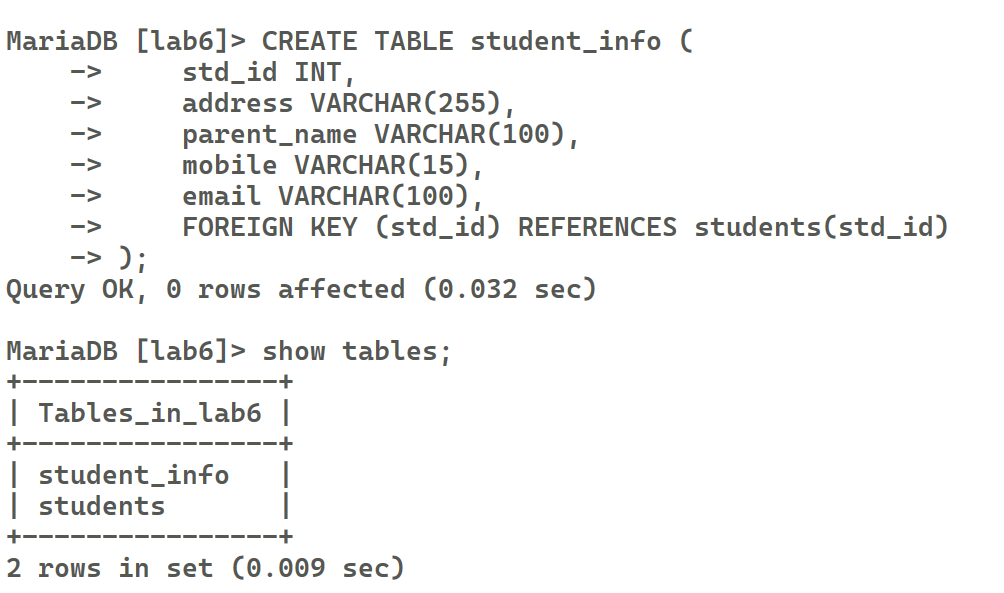
parent\_name VARCHAR(100),

mobile VARCHAR(15),

email VARCHAR(100),

FOREIGN KEY (std\_id) REFERENCES students(std\_id)

);



SHOW tables;

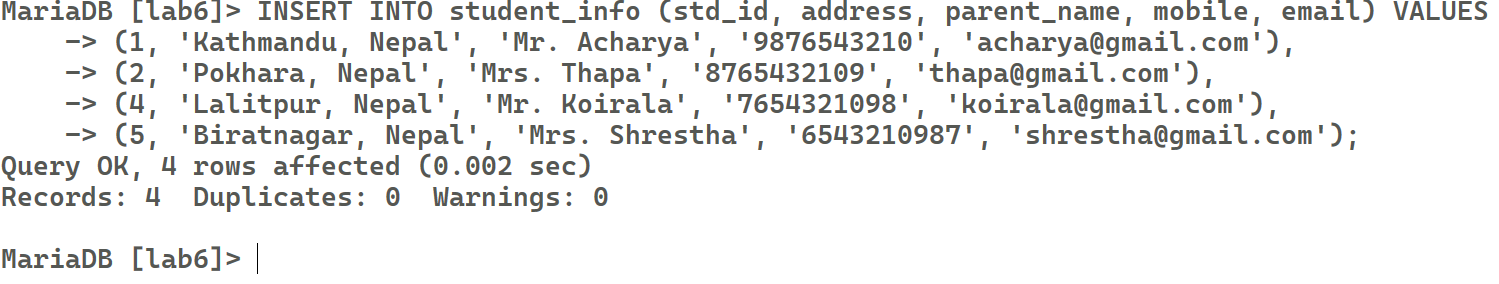
INSERT INTO student\_info (std\_id, address, parent\_name, mobile, email) VALUES

(1, 'Kathmandu, Nepal', 'Mr. Acharya', '9876543210', 'acharya@gmail.com'),

(2, 'Pokhara, Nepal', 'Mrs. Thapa', '8765432109', 'thapa@gmail.com'),

(3, 'Lalitpur, Nepal', 'Mr. Koirala', '7654321098', 'koirala@gmail.com'),

(4, 'Biratnagar, Nepal', 'Mrs. Shrestha', '6543210987', 'shrestha@gmail.com');



**q4.php:**

<?php

include('db.php');

$query = "SELECT s.student\_name, s.marks\_obtained, d.parent\_name, d.email

FROM students s

JOIN student\_info d ON s.std\_id = d.std\_id

WHERE s.result = 'Fail'";

$result = $conn->query($query);

echo "<table border='1' cellpadding='8' cellspacing='0'>";

echo "<tr><th>Student Name</th><th>Marks Obtained</th><th>Parent Name</th><th>Email</th></tr>";

while ($row = $result->fetch\_assoc()) {

    echo "<tr>

        <td>{$row['student\_name']}</td>

        <td>{$row['marks\_obtained']}</td>

        <td>{$row['parent\_name']}</td>

        <td>{$row['email']}</td>

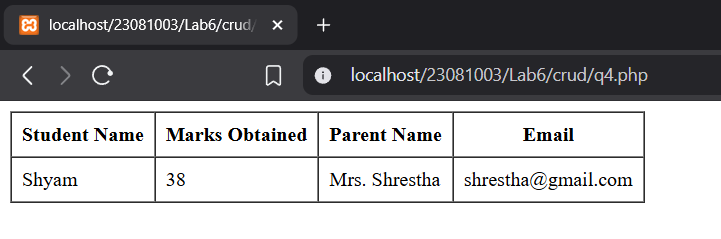
    </tr>";

}

echo "</table**>";**

?>

Output:



**5. Demonstrate how OOP is used in PHP. Show how we can use constructor and destructor in PHP**

**Source Code:**

**oop.php**

<?php

class Student {

*public* $name;

    // *Constructor*

    function \_\_construct($name) {

        $this->name = $name;

        echo "Created: $this->name<br>";

    }

    // *Destructor*

    function \_\_destruct() {

        echo "Destroyed: $this->name<br>";

    }

}

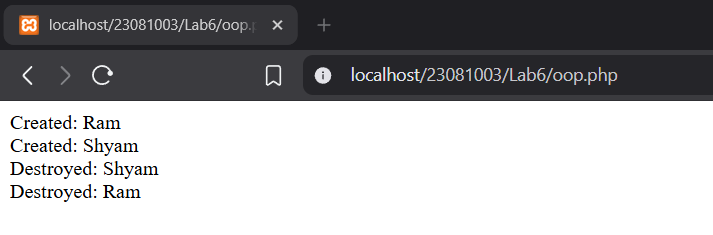
// *Create object*

$student1 = new Student("Ram");

$student2 = new Student("Shyam");

?>

Output:



**6. Create a Parent Class Lecturer. Derive two children class Parttime and Fulltime. The children class must share at least 2 properties each with parent class. Show case the working of inheritance and polymorphism in PHP**

**Source code:**

**q6.php**

<?php

// *Parent Class*

class Lecturer {

*public* $name;

*public* $salary;

    // *Constructor*

    function \_\_construct($name, $salary) {

        $this->name = $name;

        $this->salary = $salary;

    }

    // *Method to display details (Can be overridden)*

    function displayDetails() {

        echo "Name: $this->name <br>";

        echo "Salary: $this->salary <br>";

    }

}

// *Child Class Parttime*

class Parttime *extends* Lecturer {

*public* $hoursWorked;

    // *Constructor for Parttime (using parent constructor)*

    function \_\_construct($name, $salary, $hoursWorked) {

        parent::\_\_construct($name, $salary);

        $this->hoursWorked = $hoursWorked;

    }

    // *Overriding the displayDetails method (Polymorphism)*

    function displayDetails() {

        echo "Parttime Lecturer: $this->name <br>";

        echo "Salary: $this->salary <br>";

        echo "Hours Worked: $this->hoursWorked <br><br>";

    }

}

// *Child Class Fulltime*

class Fulltime *extends* Lecturer {

*public* $benefits;

    // *Constructor for Fulltime (using parent constructor)*

    function \_\_construct($name, $salary, $benefits) {

        parent::\_\_construct($name, $salary);

        $this->benefits = $benefits;

    }

    // *Overriding the displayDetails method (Polymorphism)*

    function displayDetails() {

        echo "Fulltime Lecturer: $this->name <br>";

        echo "Salary: $this->salary <br>";

        echo "Benefits: $this->benefits <br><br>";

    }

}

// *Create objects for both Parttime and Fulltime*

$parttimeLecturer = new Parttime("John", 3000, 20);  // *20 hours worked*

$fulltimeLecturer = new Fulltime("Alice", 5000, "Health Insurance");

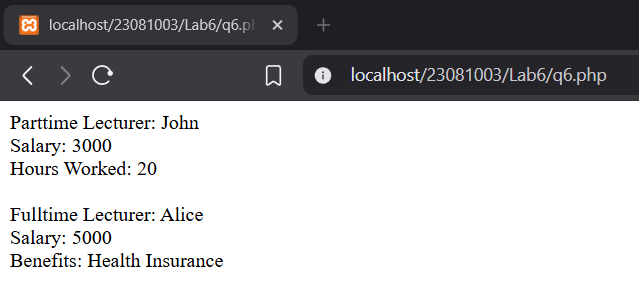
// *Display details using polymorphism*

$parttimeLecturer->displayDetails();

$fulltimeLecturer->displayDetails();

?>

Output:

**7. In the above question use different access modifiers to show their use case. Make use of setter and getter functions to access private and public properties**

**Source Code:**

**q7.php:**

<?php

// *Parent Class*

class Lecturer {

*public* $name;

*protected* $salary;

*private* $department;

    function \_\_construct($name, $salary, $department) {

        $this->name = $name;

        $this->salary = $salary;

        $this->department = $department;

    }

*public* function getSalary() { return $this->salary; }

}

// *Child Class Parttime*

class Parttime *extends* Lecturer {

*public* $hoursWorked;

    function \_\_construct($name, $salary, $department, $hoursWorked) {

        parent::\_\_construct($name, $salary, $department);

        $this->hoursWorked = $hoursWorked;

    }

    function display() {

        echo "Parttime: $this->name, Salary: {$this->getSalary()} <br>"; // *Corrected here*

    }

}

// *Child Class Fulltime*

class Fulltime *extends* Lecturer {

*public* $benefits;

    function \_\_construct($name, $salary, $department, $benefits) {

        parent::\_\_construct($name, $salary, $department);

        $this->benefits = $benefits;

    }

    function display() {

        echo "Fulltime: $this->name, Benefits: $this->benefits <br>";

    }

}

// *Objects*

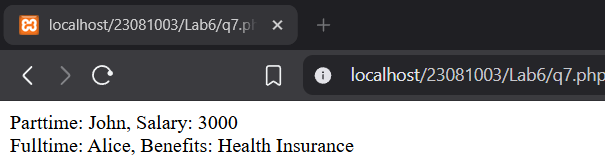
$parttime = new Parttime("John", 3000, "Math", 20);

$fulltime = new Fulltime("Alice", 5000, "CS", "Health Insurance");

$parttime->display();

$fulltime->display();

?>

Output:

**8. What are static members? WAP to show the use of static members in OOP**

**Source Code:**

**q8.php**

<?php

class Counter {

public static $count = 0;

public static function increment() {

self::$count++;

}

public static function display() {

echo "Count: " . self::$count . "<br>";

}

}

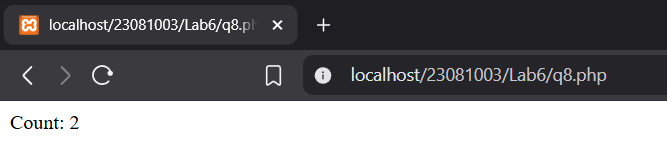
Counter::increment();

Counter::increment();

Counter::display(); // Output: Count: 2

?>

Output:

****

**Conclusion:**

In conclusion, this lab provided hands-on experience with PHP and MySQL to perform CRUD operations and manage dynamic data efficiently. It reinforced the understanding of how web applications interact with databases to store, retrieve, update, and delete records. Through the use of SQL queries, students practiced data manipulation and analysis, including sorting, aggregation, and joins. The implementation of Object-Oriented Programming concepts further strengthened coding skills by promoting better code structure and reusability. By exploring constructors, destructors, inheritance, access modifiers, and static members, students gained practical insights into OOP in PHP. The lab also highlighted the importance of integrating frontend and backend components. Overall, it laid a strong foundation for developing database-driven web applications using PHP.