

School of Information and Computer  
Technology Sirindhorn International Institute  
of Technology Thammasat University  
CSS326 Database Programming Laboratory

*Laboratory #9: Database Programming with PHP*

**Objectives:** To develop the ability to interact with database using PHP.  
To understand how to create dynamic website with data driven model.  
To understand how to use file to act as simple database in PHP.

## 1 Introduction

In PHP, most of the time, you will work with some kind of storage. In the early days of computing, before the advent of sophisticated database management systems (DBMS), the primary method of data storage and retrieval was through file systems. This approach was relatively simple and intuitive. However, they have limitations, especially as applications become more complex and data intensive. As a result, modern applications often rely on more advanced database management systems (DBMS) like MySQL, PostgreSQL, MongoDB, and others. DBMS provides features like data integrity, structured querying, concurrent access control, and scalability, making them suitable for a wide range of applications, from small-scale websites to large-scale enterprise systems.

## 2 Using File with PHP to store data

Using files to store data in PHP is a straightforward process. You can perform operations like writing, reading, and appending data to files. The step of writing data to file is as following:

### 2.1 Opening a File:

To open a file, you'll use the **fopen()** function. It takes two parameters: the file path and the mode in which you want to open the file (e.g., read, write, etc.).

Mode is parameter specifies the type of access you want to have to the file. It's a string that contains one or more of the following characters:

- **r**: Read only. Opens the file for reading. The file pointer is at the beginning of the file. If the file does not exist or cannot be found, it will generate an error.
- **w**: Write only. Opens the file for writing. If the file does not exist, PHP will attempt to create it. If it does exist, it will be truncated (i.e., its contents will be deleted).
- **a**: Append. Opens the file for writing but places the file pointer at the end of the file. If the file does not exist, PHP will attempt to create it.
- **x**: Exclusive. Creates and opens a new file for writing. If the file already exists, **fopen()** will fail.
- **b**: Binary mode. This is used for operations that deal with binary data.
- **t**: Text mode. This is used for operations that deal with text data.
- **+**: Open for reading and writing.

```
$file = fopen("example.txt", "w") or die("Unable to open file!");
```

In this example, we're opening a file named `example.txt` in write mode ("**w**"). This line of code is a PHP statement that involves opening a file using the **fopen()** function. The code actually comprise two major component:

- **\$file = fopen("example.txt", "w"):**
  - **fopen()**: This is a built-in PHP function used to open a file. It returns a file pointer resource if successful, or **FALSE** if it encounters an error.
  - **"example.txt"**: This is the name of the file you want to open. It's specified as a string, and it should include the file's path if it's not in the same directory as the PHP script.
  - **"w"**: This is the mode in which you want to open the file. In this case, it stands for "write mode". This mode is used to open the file for writing. If the file doesn't exist, PHP will try to create it. If it already exists, the content will be truncated (i.e., erased).
  - **\$file**: This is a variable that will hold the file pointer resource returned by **fopen()**. It allows you to perform operations on the file, such as writing data to it.
- **or die("Unable to open file!"):**
  - **or**: This is a logical operator that allows you to execute a different statement or expression if the previous one (**fopen()**) fails.
  - **die()**: This is a PHP function that terminates the script and displays an error message. It's often used for handling critical errors.
  - **"Unable to open file!"**: This is the error message that will be displayed if **fopen()** fails to open the file. It indicates that there was an issue with opening the specified file.

## 2.2 Writing to a File:

To write data to a file, use the **fwrite()** function.

```
$file = fopen("example.txt", "w") or die("Unable to open file!");
$text = "Hello, World!";
fwrite($file, $text);
fclose($file);
```

This code will write the string "Hello, World!" to the file `example.txt`.

## 2.3 Reading from a File:

To read data from a file, use the **fread()** function.

```
$file = fopen("example.txt", "r") or die("Unable to open file!");
$data = fread($file, filesize("example.txt"));
fclose($file);
```

In this example, we're reading the entire content of `example.txt` into the variable `$data`.

## 2.4 Appending to a File:

To append data to an existing file, open it in append mode ("**a**").

```
$file = fopen("example.txt", "a") or die("Unable to open file!");
$text = "\nThis is a new line.";
fwrite($file, $text);
fclose($file);
```

This code will append a new line to the end of the file.

## 2.5 Checking if a File Exists:

You can use the **file\_exists()** function to check if a file exists.

```
if (file_exists("example.txt")) {
    echo "The file exists!";
} else {
    echo "The file does not exist.";
}
```

## 2.6 Deleting a File:

To delete a file, use the **unlink()** function.

```
if (unlink("example.txt")) {
    echo "File deleted successfully.";
} else {
    echo "Unable to delete the file.";
}
```

## 2.7 Handling Errors:

It's important to handle potential errors when working with files. Functions like **fopen()** and **fwrite()** can fail for various reasons, so it's a good practice to use **or die()** or implement more robust error handling.

```
$file = fopen("nonexistent_file.txt", "r") or die("Unable to open file!");
$data = fread($file, filesize("nonexistent_file.txt"));

if ($data === false) {
    die("Error reading file.");
}

fclose($file);
```

In this example, we attempt to open a file named **nonexistent\_file.txt** in read mode ("r"). Since this file doesn't exist, the **fopen()** function will fail, and the **or die()** statement will be triggered. The script will display the message "Unable to open file!" and terminate.

If the file were to be opened successfully, but there was an issue reading the content (e.g., due to permissions or other unforeseen reasons), the subsequent **if (\$data === false)** block would catch the error and display the message "Error reading file."

**EXERCISE 1: SIMPLE FILE AS DATABASE**

- Implement the code below and try to access it via the web server.
- Make sure it looks like figure 1.
- Try to change the text to the gray panel at the top by adding new note and save.
- **BONUS:** Can you load data back into the **textarea**, so that we can modify the old text instead of replacing them.

```
<?php
if (isset($_POST["submit"])) {
    $file = fopen("mydb.txt", "w");
    fwrite($file, $_POST['note']);
    fclose($file);
}
?>

<!DOCTYPE html>
<html>
<head>
<title>Simple File DB</title>
</head>
<body>
<p style="border:solid 1px gray;background-color:#EEEEEE">
    <?php
        readfile("mydb.txt");
    ?>
</p>
<hr/>
<form action="simple_filedb.php" method="POST">
    <b>Note</b>; <textarea name="note" cols="30" rows="5"></textarea> <br>
    <input type="submit" value="Save to DB" name="submit">
</form>
</body>
</html>
?>
```

Hello world! This is my first file as DB. Very easy!

Note:

Save to DB

Figure 1 Expected output of simple filedb

**EXERCISE 2: WHY USING FILE AS DATABASE IS A BAD IDEA?**

- Implement the following code as file **count\_me.php**.
- Add the following data text file: **counter.txt**.
- Open a browser window, navigate to **count\_me.php** on the **localhost**
- Try changing the number in **counter.txt** to 1000(You must change and save in 5 secs)
- See if the number changes back. Observe it behavior.

```
<?php
$file = fopen("counter.txt", "r+") or die("Unable to open file!");

// Read the current counter value
$currentValue = intval(fread($file, filesize("counter.txt")));
```

```
// Simulate a delay
sleep(5); // Sleep for 5 seconds (simulated delay)

// Simulate user update (increment counter)
$newValue = $currentValue + 1;

// Move to the beginning of the file to update the counter
fseek($file, 0);
fwrite($file, $newValue);

fclose($file);

echo "Counter updated successfully! New value: $newValue";
```

update\_profile.php

0

counter.txt

**Explanation:**

- The script attempts to open **counter.txt** and read the existing number.
- The **sleep(5)** function simulates a delay to exaggerate the issue. In real-world scenarios, this delay might not exist, but concurrent updates can still occur.
- Finally, it updates the value in the file and closes it.

Observe that the counter value is changed inconsistency.

**Note:** This exercise demonstrates a simplified scenario. In real-world applications, you would implement more robust methods to handle concurrency issues, such as using a database management system with transaction control.

**Note:** You can increase the delay time of sleep function if you can't alter counter.txt within 5 seconds after you open **count\_me.php**

## 3 Database Connection

### 3.1 Establish the connection from PHP server to DB server.

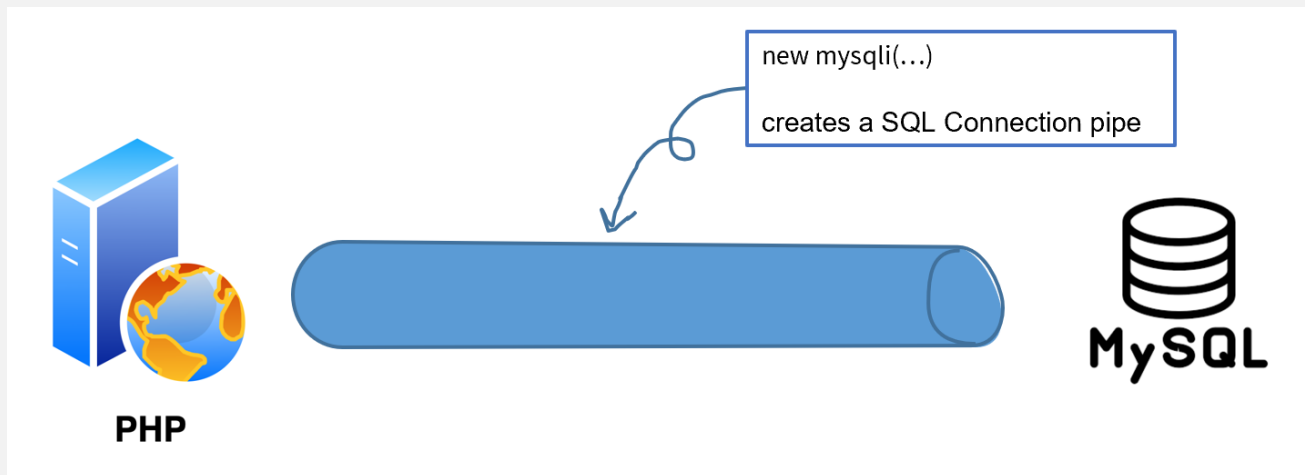


Figure 2 Illustrate the pipe analogy to a connection object

Before you can do anything with database, you will first need to form up the connection object. The connection object acts like a pipe or a bridge between your PHP script (as depicted in Figure 2) and the MySQL database. It allows you to send queries to the database and receive results back.

This line creates a new MySQLi connection object called **\$conn** to establish the connection to the database. The MySQLi class provides methods to interact with the database.

```
new mysqli($servername, $username, $password, $dbname);
```

For example, you can use methods like **\$conn->query()** to execute SQL queries and retrieve data from the database. This object also provides various other methods for tasks like error handling, transaction management, etc.

**\$servername, \$username, \$password, and \$dbname:** These are variables that hold the connection details.

- **\$servername:** Contains the hostname of the database server. In this case, it's set to "localhost", indicating that the database server is on the same machine as the web server.
- **\$username:** The username used to authenticate with the MySQL database. Here, it's set to "root", which is commonly used for local development environments.
- **\$password:** The password associated with the username. In this example, it's also set to "root".
- **\$dbname:** Specifies the name of the database you want to connect to.

#### Example:

```
<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "db_name"; // Replace "db_name" with the actual name of your database.

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
```

```
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

- **if (\$conn->connect\_error) {...}**: This checks if there was an error while establishing the connection. If an error occurs, it executes the code within the curly braces.
- **die("Connection failed: " . \$conn->connect\_error);**: If there is an error, this line terminates the script and displays an error message that includes the specific connection error.

## 4 Sending SQL to the Database

After successfully establishing a connection to the database, the next step is to send SQL queries for various operations like inserting, updating, and deleting data. Sending SQL queries to the database is crucial in database programming because it enables developers to interact with the database, retrieve, modify, and manage data.

The form of SQL execution can be divided into

### 4.1 Inserting data

To insert data, you will need the connection object that you have established earlier (the **\$conn**). Then, you need to construct an SQL statement for inserting data. The following code will insert the data into a table with the given SQL statement.

```
<?php
    $mysqli = new mysqli('localhost','user','password','dbname');
    if($mysqli->connect_errno){
        echo $mysqli->connect_errno." : ".$mysqli->connect_error;
    }

    $q='CREATE table product(p_id int unsigned not null auto_increment
    primary key, p_name varchar(30), p_price int)';
    if($mysqli->query($q)){
        echo 'CREATE was successful.';
    }else{
        Echo 'CREATE failed. Error: '.$mysqli->error ;
    }

?>
```

Observe that SQL statement here is merely a string value. We will prepare the whole statement before sending it to execute at the database server. The code below shows how to send SQL command to the server via the connection we have opened earlier.

**Example:**

```

<?php
$mysqli = new mysqli('localhost','root','','new');
if($mysqli->connect_errno){
    echo $mysqli->connect_errno.": ".$mysqli->connect_error;
}
$recs=array(
    array('Pencil',10),array('Eraser',5),
    array('Mouse',600),array('Printer',4000)
);
foreach($recs as $r){
    $q="INSERT INTO product(p_name, p_price) VALUES('$r[0]', $r[1])";
    if(!$mysqli->query($q)){
        echo "INSERT failed. Error: ".$mysqli->error ;
        break;
    }
}
?>

```

**4.2 Problem with SQL string**

A certain symbol cannot be put directly into SQL string(e.g. single quote or double quote).

```

<?php
$mysqli = new mysqli('localhost','root','','new');
if($mysqli->connect_errno){
    echo $mysqli->connect_errno.": ".$mysqli->connect_error;
}
$r=array("Idiot's Guide Book",1200);
$q="INSERT INTO product(p_name, p_price) VALUES('$r[0]', $r[1])";
if(!$mysqli->query($q)){
    echo "INSERT failed. Error: ".$mysqli->error ;
}
?>

```

The code above will show the error as:

```
INSERT failed. Error: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 's Guide Book', 1200)' at line 1
```

To circumvent this, we can use a function to **escape** string properly.



```

<?php
$mysqli = new mysqli('localhost','root','','new');
if($mysqli->connect_errno){
echo $mysqli->connect_errno.": ".$mysqli->connect_error;
}
$r=array("Idiot's Guide Book",1200);
// $q="INSERT INTO product(p_name, p_price) VALUES('$r[0]', $r[1])";
$q="INSERT INTO product(p_name, p_price)
VALUES('".$mysqli->real_escape_string($r[0])."', $r[1])";

if(!$mysqli->query($q)){
echo "INSERT failed. Error: ".$mysqli->error ;
}
?>

```

### 4.3 Getting back row's ID from inserting

Sometimes, you will be inserting data into a table with AUTO INCREMENT. You may need to get back the latest ID that you have just inserted. The code below show how:

```

<?php
$mysqli = new mysqli('localhost','root','','new');
if($mysqli->connect_errno){
echo $mysqli->connect_errno.": ".$mysqli->connect_error;
}
$r=array("Idiot's Guide Book",1200);
$q="INSERT INTO product(p_name, p_price)
VALUES('".$mysqli->real_escape_string($r[0])."', $r[1])";
if(!$mysqli->query($q)){
echo "INSERT failed. Error: ".$mysqli->error ;
}
$id = $mysqli->insert_id;
echo "Record $id is inserted";
?>

```

Record 6 is inserted

## 5 Retrieve back data and render HTML

Retrieving data and displaying it in a particular format is one of the most, if not the most, important processes in database operations.

### 5.1 Fetching data from DB

To get data you need to send SQL string to be executed at the server side. Then, using fetch operation such as **fetch\_array()** to iterate over each row of return data. Code below shows how:

```

<?php
    $mysqli = new mysqli('localhost','root','','new');
    if($mysqli->connect_errno){
        echo $mysqli->connect_errno.": ".$mysqli->connect_error;
    }
    if($result=$mysqli->query('show tables')){
    while($row=$result->fetch_array()){
        echo$row[0].'<br>';
    }
    $result->free();
    }else{
        echo "Retrieval failed";
    }
?>

```

## 5.2 Display data in formatted HTML

You can combine the **fetch\_array()** loop to generate HTML to display nicely formatted data.

```

<?php
    $mysqli = new mysqli('localhost','root','','new');
    if($mysqli->connect_errno){
        echo $mysqli->connect_errno.": ".$mysqli->connect_error;
    }
    $q="select p_name, p_price from product where p_price> 100; ";
    if($result=$mysqli->query($q)){
        echo '<table border="1">';
        echo '<tr><th>Name</th><th>Price</th></tr>';
        while($row=$result->fetch_array()){
            echo "<tr>";
            echo "<td>".$row['p_name']. "</td>";
            echo "<td>".$row['p_price']. "</td>";
            echo "</tr>";
        }
        Echo '</table>';
        $result->free();
    }else{
        Echo "Retrieval failed: ".$mysqli->error ;
    }
?>

```

## 5.3 Getting total rows retrieved

You can count the total number of rows retrieved (not the total number of rows in the table) using the code as following:

```

<?php
$mysqli = new mysqli('localhost','root','','new');
if($mysqli->connect_errno){
    echo $mysqli->connect_errno.": ".$mysqli->connect_error;
}
$q="select p_id from product where p_name like 'P%'; ";
if($result=$mysqli->query($q)){
    $count=$result->num_rows;
    Echo "There are $count products starting with P.";
    $result->free();
}else{
    Echo "Query failed: ".$mysqli->error ;
}
?>

```

### 5.4 Jump to a particular row by index

You can jump to a particular row by row index(index of rows returned, not the rows in table)

```

<?php
$mysqli = new mysqli('localhost','root','','new');
if($mysqli->connect_errno){
    echo $mysqli->connect_errno.": ".$mysqli->connect_error;
}
$q='select p_name, p_price from product order by p_price limit 3;';
if($result=$mysqli->query($q)){
    // Seek to the third row (row index starts from 0)
    $result->data_seek(2);
    $row=$result->fetch_array();
    Echo $row['p_name']. " has the third lowest price which is 
    ".$row['p_price'];
    $result->free();
}else{
    Echo "Query failed: ".$mysqli->error;
}
?>

```

## 6 Deleting data

Deleting data is mostly the work of SQL command rather than PHP script. The code below shows how:

```

<?php
    $mysqli = new mysqli('localhost','root','','new');
    if($mysqli->connect_errno){
        echo $mysqli->connect_errno.": ".$mysqli->connect_error;
    }
    $q="DELETE FROM product where p_id=5";
    if(!$mysqli->query($q)){
        echo "DELETE failed. Error: ".$mysqli->error ;
    }
    ?>

```

### 6.1 Combine the power of \$\_GET and delete to create delete link

You can combine the link to create PHP script that take \$\_GET request command to delete data from the table.

```

<?php
    $mysqli = new mysqli('localhost','root','','new');
    if($mysqli->connect_errno){
        echo $mysqli->connect_errno.": ".$mysqli->connect_error;
    }
    $q="select * from product";
    if($result=$mysqli->query($q)){
        echo '<table border="1">';
        echo '<tr><th>Name</th><th>Price</th><th>Delete</th></tr>';
        while($row=$result->fetch_array()){
            echo "<tr>";
            echo "<td>".$row['p_name']. "</td>";
            echo "<td>".$row['p_price']. "</td>";
            echo "<td><a href='delinfo.php?id=" .
                $row['p_id'] . "'> Delete</a></td>";
            echo "</tr>";
        }
        echo '</table>';
        $result->free();
    }else{
        echo "Retrieval failed: ".$mysqli->error ;
    }
    ?>

```

**viewinfo.php**

```
<?php
$mysqli = new mysqli('localhost','root','','new');
if($mysqli->connect_errno){
echo $mysqli->connect_errno." : ".$mysqli->connect_error;
}
$p_id = $_GET['id'];
$mysqli = new mysqli('localhost','root','root','staff');
if($mysqli->connect_errno){
echo $mysqli->connect_errno." : ".$mysqli->connect_error;
}
$q="DELETE FROM product where p_id=$p_id";
if(!$mysqli->query($q)){
echo "DELETE failed. Error: ".$mysqli->error ;
}
$mysqli->close();
//redirect
header("Location: viewinfo.php");
?>
```

**delinfo.php**



## 7 Updating data

In principle, updating data is also simple SQL that is sending to be executed at the database as well. However, in practice, it mostly comes along with data display. Users need to see the data they want to edit before making any change.

### 7.1 Retrieve data and form a link to the HTML forms for editing.

```
<?php
require_once('connect_1.php');
$q="select * from product";
if($result=$mysqli->query($q)){
echo '<table border="1">';
echo '<tr><th>ID</th><th>Name</th><th>Price</th><th>Edit</th></tr>';
while($row=$result->fetch_array()){
echo "<tr>";
echo "<td>".$row['p_id']. "</td>";
echo "<td>".$row['p_name']. "</td>";
echo "<td>".$row['p_price']. "</td>";
echo "<td><a href='practice_2edit.php?id="
.$row['p_id']."'> Edit</a></td>";
echo "</tr>";
}
echo '</table>';
$result->free();
}else{
echo "Retrieval failed: ".$mysqli->error ;
}
?>
```

### 7.2 Edit form

```
<?php
$p_id = $_GET['id'];
require_once('connect_1.php');
$q="SELECT * FROM product where p_id=$p_id";
$result = $mysqli->query($q);
echo "<form action='practice_2update.php' method='post'>";
while($row=$result->fetch_array()){
echo "Product ID: <input type='text' name='id'
value=".$row['p_id']. " Disabled><br>";
echo "<input type='hidden' name='p_id' value='".$row['p_id']."'>";
echo "Product Name: <input type='text' name='p_name'
value=".$row['p_name']. "><br>";
echo "Product Price: <input type='text' name='p_price'
value=".$row['p_price']. "><br>";
echo "<input type='submit' value='submit'>";
}
$mysqli->close();
?>
```








### 7.3 Send update SQL command to the server













```
<?php
require_once('connect_1.php');
$p_id = $_POST['p_id'];
$p_name = $_POST['p_name'];
$p_price = $_POST['p_price'];

$q="UPDATE product SET p_name='$p_name', p_price='$p_price'
where p_id=$p_id";
if(!$mysqli->query($q)){
echo "UPDATE failed. Error: ".$mysqli->error ;
}
$mysqli->close();
//redirect
header("Location: practice_2.php");
?>
```

#### EXERCISE 3: LET'S CREATE COMBO BOX EDITOR

##### 1. Modify the table structure to be as follows:

					p_type_ID	p_type
<input type="checkbox"/>	 Edit	 Copy	 Delete		1	accessories
<input type="checkbox"/>	 Edit	 Copy	 Delete		2	stationary

<div><div><div></div><div></div><div></div></div></div>				p_id	p_name	p_price	p_type_ID	
<input type="checkbox"/>		Edit	 Copy	 Delete	1	Pencil case	10	2
<input type="checkbox"/>		Edit	 Copy	 Delete	2	Eraser	5	2
<input type="checkbox"/>		Edit	 Copy	 Delete	3	Mouse	600	1
<input type="checkbox"/>		Edit	 Copy	 Delete	4	Printer	4000	1

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	p_type_ID	int(10)		UNSIGNED	No	None		AUTO_INCREMENT	Change  Drop  More
<input type="checkbox"/> 2	p_type	varchar(25)	utf8mb4_general_ci		No	None			Change  Drop  More



## 2. Create product list using the following code:

```
<?php
require_once('connect_1.php');
$q="select * from product";
if($result=mysqli->query($q)){
    echo '<table border="1">';
    echo '<tr><th>ID</th><th>Name</th><th>Price</th><th>Product type</th><th>Edit</th></tr>';
    while($row=$result->fetch_array()){
        echo "<tr>";
        echo "<td>".$row['p_id']. "</td>";
        echo "<td>".$row['p_name']. "</td>";
        echo "<td>".$row['p_price']. "</td>";
        $q1="select * from product_type";
        $result1=mysqli->query($q1);
        while($row1=$result1->fetch_array()){
            if ($row1[0]==$row['p_type_ID']){
                echo "<td>".$row1[1]. "</td>";
            }
        }
        echo "<td><a href='practice_2edit.php?id="
        . $row['p_id']. "'> Edit</a></td>";
        echo "</tr>";
    }
    echo '</table>';
    $result->free();
}else{
    echo "Retrieval failed: ".$mysqli->error ;
}
?>
```

## 3. Add the combo box

```
<?php
$p_id = $_GET['p_id'];
require_once('connect_2.php');
$q3="SELECT * FROM product where p_id=$p_id ";
$result3 = mysqli->query($q3);
echo "<form action='practice_3update.php' method='post'>";
while($row3=$result3->fetch_array()){
    echo "Product ID: <input type='text' name=id
    value=".$row3['p_id']. " Disabled<br>";
    echo "<input type='hidden' name=p_id value='".$row3['p_id']."'>";
    echo "Product Name: <input type='text' name=p_name1
    value=".$row3['p_name']. "><br>";
    echo "Product Price: <input type='text' name=p_price1
    value=".$row3['p_price']. "><br>";
    echo "Product Type: <select name=p_type1>";
    $q4="select p_type_ID, p_type from product_type";
    if($result4=mysqli->query($q4)){
        while($row4=$result4->fetch_array()){
            echo "<option value='".$row4[0]. "' ";
            if ($row4[0] == $row3['p_type_ID'])
                echo " SELECTED ";
            echo ">".$row4[1]. "</option>";
        }
    }else{
        echo 'Query error: '.$mysqli->error;
    }
    echo "</select><br>";
    echo "<input type='submit' name=sub value=submit>";
}
$result3->free();
$result4->free();
?>
```



#### 4. YOU TASK: Modify the update code so that it update product type(p\_type1)

```
<?php
require_once('connect_2.php');
if (isset($_POST['sub'])) {
    [REDACTED]
}
if (!$mysqli->query($q)) {
    echo "UPDATE failed. Error: " . $mysqli->error ;
}
}
//redirect
header("Location: practice_3.php");
?>
```

## 8 Summary

Using PHP with a MySQL database involves several key steps. Firstly, establish a connection by creating a **mysqli** object, providing server details, username, password, and database name; this establishes a channel between your PHP code and the MySQL database. Next, execute queries with **\$conn->query(\$sql)** to perform actions like retrieval, insertion, updating, or deletion of data. After executing a SELECT query, handle the results using methods like **fetch\_assoc()** or **fetch\_array()**. It's important to handle potential errors, especially during connection setup, to ensure graceful handling of failures. Additionally, consider using prepared statements with placeholders and parameter binding for security against SQL injection attacks. Lastly, close the connection using **\$conn->close()** when your interactions with the database are complete, to free up system resources and prevent leaving connections open. Throughout your application, implement error handling with try-catch blocks to manage situations where queries or connections may fail, creating a robust and secure interaction between your PHP code and the MySQL database.