**Experiment No. 5**

# Title: Database and PHP integration using PDO

**Batch: B-2** **Roll No.: 16010422234** **Experiment No:5**

---

**Aim:** Write a PHP program for database activities using PDO.

---

**Resources needed:** Windows OS, Web Browser, Editor, XAMPP Server

---

**Pre Lab/ Prior Concepts:**
Students should have prior knowledge of mysql database and PHP constructs.

---

**Theory:**
The PHP Data Objects (PDO) extension defines a lightweight, consistent interface for accessing databases in PHP. Each database driver that implements the PDO interface can expose database-specific features as regular extension functions. Note that you cannot perform any database functions using the PDO extension by itself; you must use a database-specific PDO driver to access a database server.
PDO provides a data-access abstraction layer, which means that, regardless of which database you're using, you use the same functions to issue queries and fetch data. PDO does not provide a database abstraction; it doesn't rewrite SQL or emulate missing features. You should use a full-blown abstraction layer if you need that facility.

Installation/Configuration Requirement
When installing PDO as a shared module, the php.ini file needs to be updated so that the PDO extension will be loaded automatically when PHP runs. You will also need to enable any database specific drivers there too; make sure that they are listed after the pdo.so line, as PDO must be initialized before the database-specific extensions can be loaded.
extension=php_pdo.dll extension=php_pdo_firebird.dll extension=php_pdo_informix.dll extension=php_pdo_mssql.dll extension=php_pdo_mysql.dll extension=php_pdo_oci.dll extension=php_pdo_oci8.dll extension=php_pdo_odbc.dll extension=php_pdo_pgsql.dll extension=php_pdo_sqlite.dll

Connecting to mysql
```
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
?>
```
If there are any connection errors, a PDOException object will be thrown. You may catch the exception if you want to handle the error condition, or you may opt to leave it for an application global exception handler that you set up via set_exception_handler().
```
<?php
```

```
try {
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
foreach($dbh->query('SELECT * from FOO') as $row) {
print_r($row);
}
$dbh = null;
} catch (PDOException $e) {
print "Error!: " . $e->getMessage() . "<br/>"; die();
}
?>
```

Closing the connection

Upon successful connection to the database, an instance of the PDO class is returned to your script. The connection remains active for the lifetime of that PDO object. To close the connection, you need to destroy the object by ensuring that all remaining references to it are deleted—you do this by assigning null to the variable that holds the object. If you don't do this explicitly, PHP will automatically close the connection when your script ends.

For eg.,

`$dbh = null;` should be added at the end of the file

Persistent connection

Many web applications will benefit from making persistent connections to database servers. Persistent connections are not closed at the end of the script, but are cached and re-used when another script requests a connection using the same credentials. The persistent connection cache allows you to avoid the overhead of establishing a new connection every time a script needs to talk to a database, resulting in a faster web application.

```
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass, array(
PDO::ATTR_PERSISTENT => true
));
?>
```

If you wish to use persistent connections, you must set PDO::ATTR_PERSISTENT in the array of driver options passed to the PDO constructor. If setting this attribute with PDO::setAttribute() after instantiation of the object, the driver will not use persistent connections.

Transactions are typically implemented by "saving-up" your batch of changes to be applied all at once; this has the nice side effect of drastically improving the efficiency of those updates. In other words, transactions can make your scripts faster and potentially more robust (you still need to use them correctly to reap that benefit).

TRANSACTIONS AND AUTOCOMMIT

Unfortunately, not every database supports transactions, so PDO needs to run in what is known as "auto-commit" mode when you first open the connection. Auto-commit mode means that every query that you run has its own implicit transaction, if the database supports it, or no transaction if the database doesn't support transactions. If you need a transaction, you must use the PDO::beginTransaction() method to initiate one. If the underlying driver does not support transactions, a PDOException will be thrown (regardless of your error handling settings: this is always a serious error condition). Once you are in a transaction, you may use PDO::commit() or PDO::rollBack() to finish it, depending on the success of the code you run during the transaction.

Example: In the following sample, let's assume that we are creating a set of entries for a new employee, who has been assigned an ID number of 23. In addition to entering the basic data for that person, we also need to record their salary. It's pretty simple to make two separate updates, but by enclosing them within the PDO::beginTransaction() and PDO::commit() calls, we are guaranteeing that no one else will be able to see those changes until they are complete. If something goes wrong, the catch block rolls back all changes made since the transaction was started, and then prints out an error message.

```
<?php try {
$dbh = new PDO('odbc:SAMPLE', 'db2inst1', 'ibmdb2', array(PDO::ATTR_PERSISTENT => true));
echo "Connected\n";
} catch (Exception $e) {
die("Unable to connect: " . $e->getMessage());
}
try {
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$dbh->beginTransaction();
$dbh->exec("insert into staff (id, first, last) values (23, 'Joe', 'Bloggs')");
$dbh->exec("insert into salarychange (id, amount, changedate) values (23, 50000, NOW())");
$dbh->commit();
} catch (Exception $e) {
$dbh->rollBack();
echo "Failed: " . $e->getMessage();
}
?>
```

## PREPARED STATEMENTS

Prepared Statement can be thought of as a kind of compiled template for the SQL that an application wants to run, that can be customized using variable parameters.

Prepared statements offer two major benefits:

The query only needs to be parsed (or prepared) once, but can be executed multiple times with the same or different parameters. When the query is prepared, the database will analyze, compile and optimize its plan for executing the query. For complex queries this process can take up enough time that it will noticeably slow down an application if there is a need to repeat the same query many times with different parameters. By using a prepared statement the application avoids repeating the analyze/compile/optimize cycle. This means that prepared statements use fewer resources and thus run faster.

The parameters to prepared statements don't need to be quoted; the driver automatically handles this. If an application exclusively uses prepared statements, the developer can be sure that no SQL

injection will occur (however, if other portions of the query are being built up with unescaped input, SQL injection is still possible).
Prepared statements are so useful that they are the only feature that PDO will emulate for drivers that don't support them. This ensures that an application will be able to use the same data access paradigm regardless of the capabilities of the database.

Insert using Prepared Statement

```
<?php
$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (?, ?)");
$stmt->bindParam(1, $name);
$stmt->bindParam(2, $value);
// insert one row
$name = 'one';
$value = 1;
$stmt->execute();
// insert another row with different values
$name = 'two';
$value = 2;
$stmt->execute();
?>
```

Retrieval using Prepared statement

```
<?php
$stmt = $dbh->prepare("SELECT * FROM REGISTRY where name = ?");
$stmt->execute([$_GET['name']]); foreach ($stmt as $row) { print_r($row);
}
?>
```

Stored Procedure prepared statement

```
<?php
$stmt = $dbh->prepare("CALL sp_takes_string_returns_string(?)");
$value = 'hello';
$stmt->bindParam(1, $value, PDO::PARAM_STR|PDO::PARAM_INPUT_OUTPUT, 4000);

// call the stored procedure
$stmt->execute();
print "procedure returned $value\n";
?>
```

PDO has following error handling strategies:
PDO::ERRMODE_SILENT
PDO will simply set the error code for you to inspect using the PDO::errorCode() and PDO::errorInfo() methods on both the statement and database objects;
PDO::ERRMODE_WARNING
In addition to setting the error code, PDO will emit a traditional E_WARNING message. This setting is useful during debugging/testing, if you just want to see what problems occurred without interrupting the flow of the application.
PDO::ERRMODE_EXCEPTION
In addition to setting the error code, PDO will throw a PDOException and set its properties to reflect the error code and error information. This setting is also useful during debugging, as it will effectively "blow up" the script at the point of the error, very quickly pointing a finger at potential problem areas in your code (remember: transactions are automatically rolled back if the exception causes the script to terminate).
Exception mode is also useful because you can structure your error handling more clearly than with traditional PHP-style warnings, and with less code/nesting than by running in silent mode and explicitly checking the return value of each database call.
Example #1 Create a PDO instance and set the error mode

```
<?php
$dsn = 'mysql:dbname=testdb;host=127.0.0.1';
$user = 'dbuser';
$password = 'dbpass';
$dbh = new PDO($dsn, $user, $password);
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
// This will cause PDO to throw a PDOException (when the table doesn't exist)
$dbh->query("SELECT wrongcolumn FROM wrongtable");
```

The above example will output:

```
Fatal error: Uncaught PDOException: SQLSTATE[42S02]: Base table or view not found: 1146 Table
'testdb.wrongtable' doesn't exist in /tmp/pdo_test.php:10
Stack trace:
#0 /tmp/pdo_test.php(10): PDO->query('SELECT wrongcol...') #1 {main}
thrown in /tmp/pdo_test.php on line 10
```

Example #2 Create a PDO instance and set the error mode from the constructor

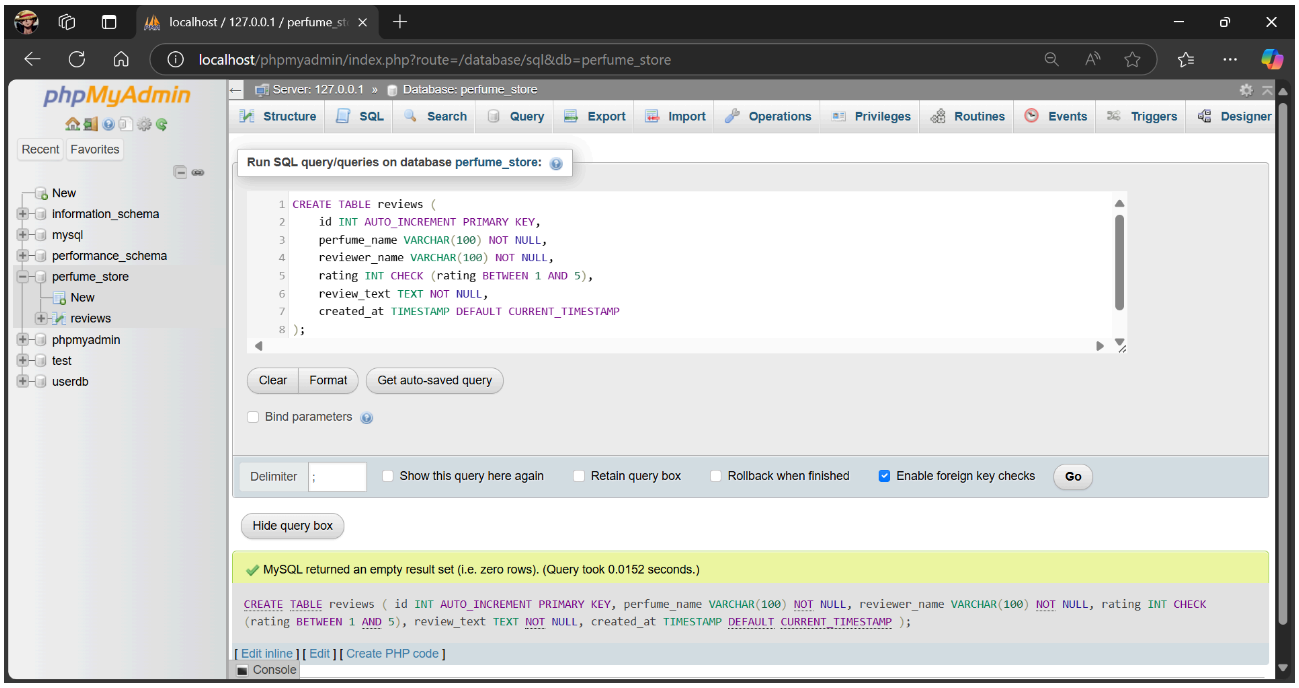
```
<?php
$dsn = 'mysql:dbname=test;host=127.0.0.1';
$user = 'googleguy';
$password = 'googleguy';
$dbh = new PDO($dsn, $user, $password, array(PDO::ATTR_ERRMODE =>
PDO::ERRMODE_WARNING));
// This will cause PDO to throw an error of level E_WARNING instead of an exception (when the
table doesn't exist)
$dbh->query("SELECT wrongcolumn FROM wrongtable");
```

**Activity:**
Write a PHP program to work with PDO databases.
Complete the following functionalities

- Creating a database.
- Creating a table.
- Inserting values into the table (values to be extracted from user input) using prepared statements.
- Retrieving values from the table using prepared statements and displaying to the user.
- Using Exception handling to show error wherever possible.
- Update some values of the table based on user input.

**Output: (Code with result snapshots)**

```
CREATE TABLE reviews (
    id INT AUTO_INCREMENT PRIMARY KEY,
    perfume_name VARCHAR(100) NOT NULL,
    reviewer_name VARCHAR(100) NOT NULL,
    rating INT CHECK (rating BETWEEN 1 AND 5),
    review_text TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```



Run SQL query/queries on database perfume_store:

```
CREATE TABLE reviews (
    id INT AUTO_INCREMENT PRIMARY KEY,
    perfume_name VARCHAR(100) NOT NULL,
    reviewer_name VARCHAR(100) NOT NULL,
    rating INT CHECK (rating BETWEEN 1 AND 5),
    review_text TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Delimiter ;  [✓ Enable foreign key checks]

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0152 seconds.)

```
CREATE TABLE reviews ( id INT AUTO_INCREMENT PRIMARY KEY, perfume_name VARCHAR(100) NOT NULL, reviewer_name VARCHAR(100) NOT NULL, rating INT CHECK (rating BETWEEN 1 AND 5), review_text TEXT NOT NULL, created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP );
```

**review_form.html**

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Perfume Review</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="review-container">
        <h2>Leave a Review</h2>
        <form action="submit_review.php" method="POST">
            <label for="perfume_name">Perfume Name:</label>
            <input type="text" id="perfume_name" name="perfume_name"
required>

            <label for="reviewer_name">Your Name:</label>
            <input type="text" id="reviewer_name" name="reviewer_name"
required>

            <label for="rating">Rating (1 to 5):</label>
            <input type="number" id="rating" name="rating" min="1" max="5"
required>

            <label for="review_text">Review:</label>
            <textarea id="review_text" name="review_text"
required></textarea>

            <input type="submit" value="Submit Review">
        </form>
    </div>
</body>
</html>
```

**style.css**

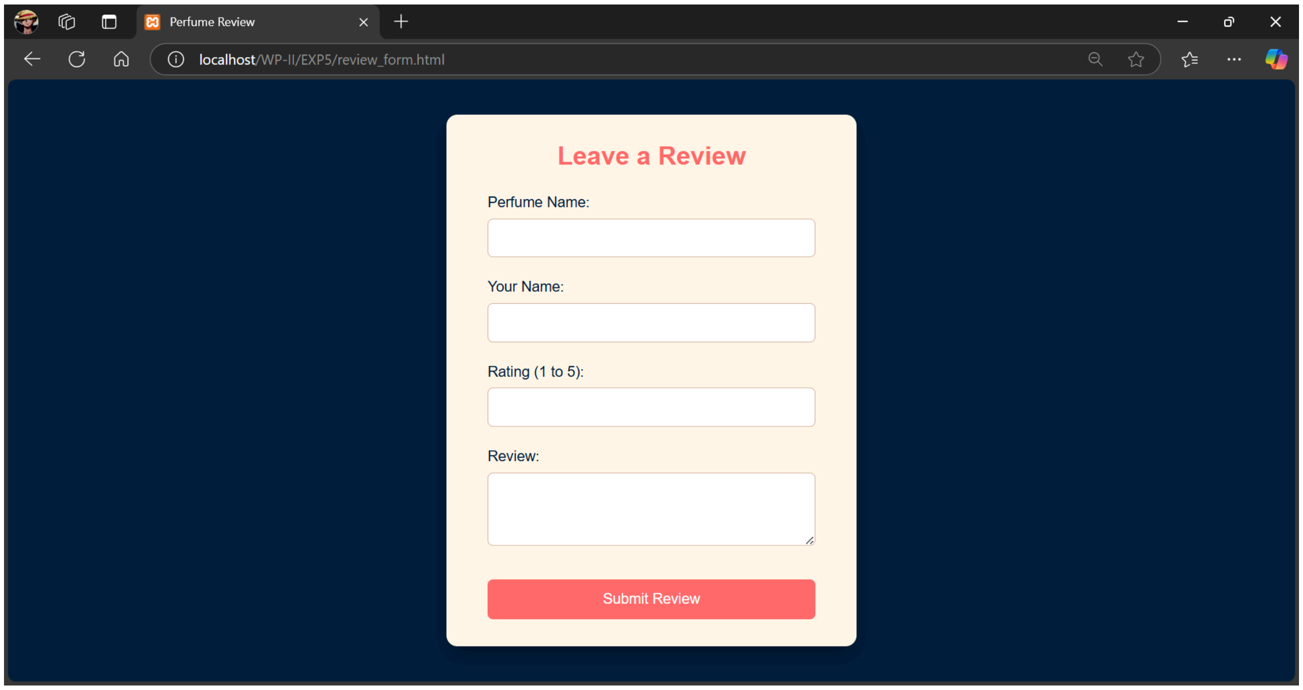
```css
/* Basic reset */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* Body styling */
body {
    font-family: Arial, sans-serif;
    background-color: #001f3f; /* Dark navy */
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    color: #ffffff;
}

/* Container styling */
.review-container {
    background-color: #fff5e6; /* Light peach */
    width: 450px;
    padding: 30px;
    border-radius: 12px;
    box-shadow: 0px 10px 20px rgba(0, 0, 0, 0.3);
    text-align: center;
}

/* Title styling */
.review-container h2 {
    font-size: 28px;
    color: #ff6b6b; /* Soft coral */
    margin-bottom: 25px;
}

/* Label styling */
.review-container label {
    display: block;
    font-size: 16px;
    color: #001f3f; /* Dark navy */
    margin-top: 15px;
    text-align: left;
    margin-left: 15px;
}

/* Input and textarea styling */
```

```
.review-container input[type="text"],
.review-container input[type="number"],
.review-container textarea {
    width: calc(100% - 30px); /* Ensures padding on both sides */
    padding: 12px;
    margin: 8px 15px 8px 15px; /* Equal margin on both sides */
    border: 1px solid #d9b5a5; /* Soft peach */
    border-radius: 6px;
    background-color: #ffffff; /* White for strong contrast */
    font-size: 15px;
    color: #001f3f;
}

/* Textarea styling adjustments */
.review-container textarea {
    resize: vertical;
    height: 80px;
}

/* Submit button styling */
.review-container input[type="submit"] {
    background-color: #ff6b6b; /* Coral */
    color: white;
    padding: 12px 20px;
    border: none;
    border-radius: 6px;
    font-size: 16px;
    margin-top: 25px;
    cursor: pointer;
    transition: background-color 0.3s ease;
    width: calc(100% - 30px); /* Matches input width */
}

.review-container input[type="submit"]:hover {
    background-color: #ff4c4c; /* Darker coral */
}
```

**submit_review.php**

```
<?php
// Database configuration
$host = 'localhost';
```

```php
$dbname = 'perfume_store';
$user = 'root';
$pass = '';

try {
    // Connect to the database
    $dbh = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);
    $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Insert form data into the 'reviews' table
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $perfume_name = $_POST['perfume_name'];
        $reviewer_name = $_POST['reviewer_name'];
        $rating = $_POST['rating'];
        $review_text = $_POST['review_text'];

        $sql = "INSERT INTO reviews (perfume_name, reviewer_name, rating,
review_text)
                VALUES (:perfume_name, :reviewer_name, :rating,
:review_text)";

        $stmt = $dbh->prepare($sql);
        $stmt->bindParam(':perfume_name', $perfume_name);
        $stmt->bindParam(':reviewer_name', $reviewer_name);
        $stmt->bindParam(':rating', $rating);
        $stmt->bindParam(':review_text', $review_text);
        $stmt->execute();

        echo "<p style='color:green; text-align:center;'>Thank you for your
review!</p>";
    }
} catch (PDOException $e) {
    echo "Error: " . $e->getMessage();
}
?>
```



Leave a Review

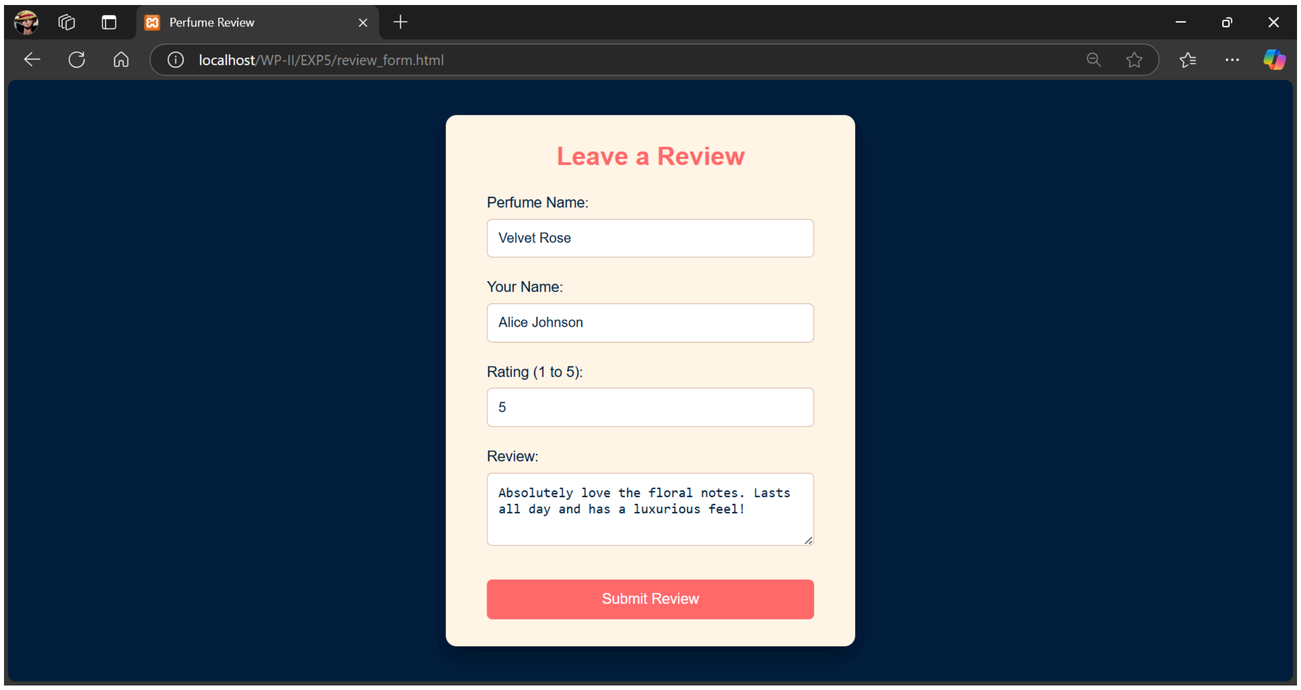
Perfume Name:

Your Name:

Rating (1 to 5):

Review:

Submit Review



Leave a Review

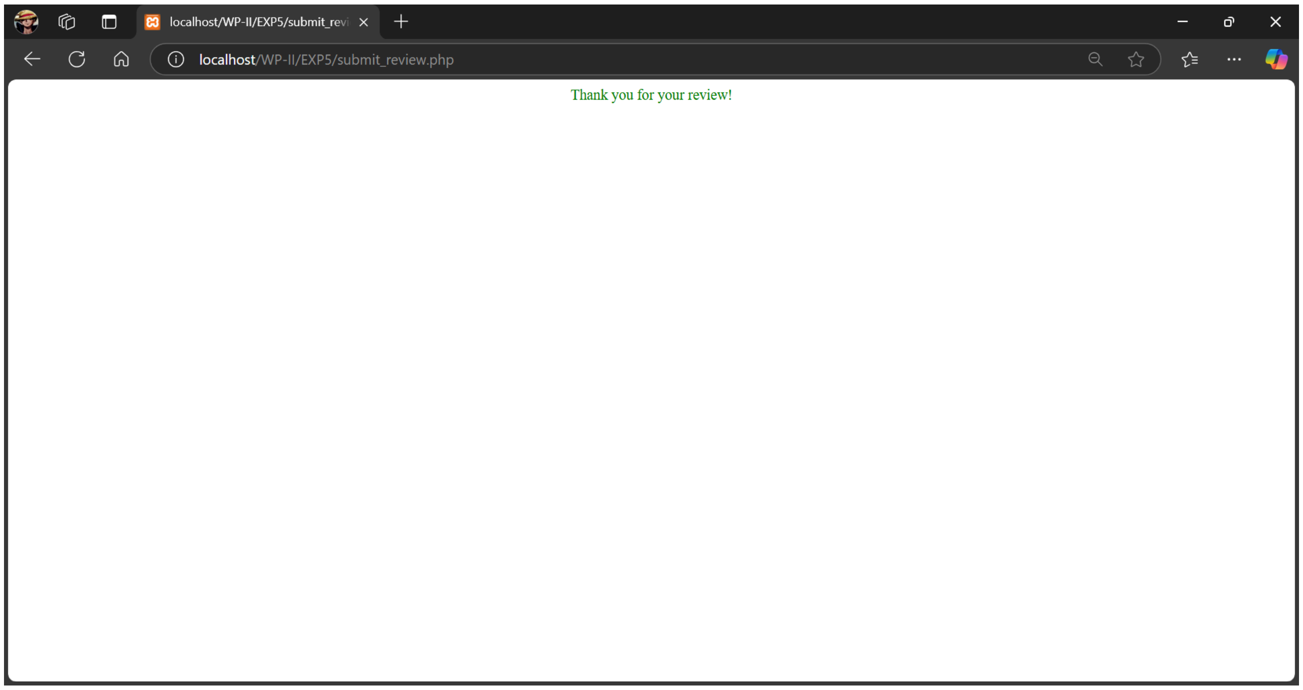
Perfume Name: Velvet Rose

Your Name: Alice Johnson

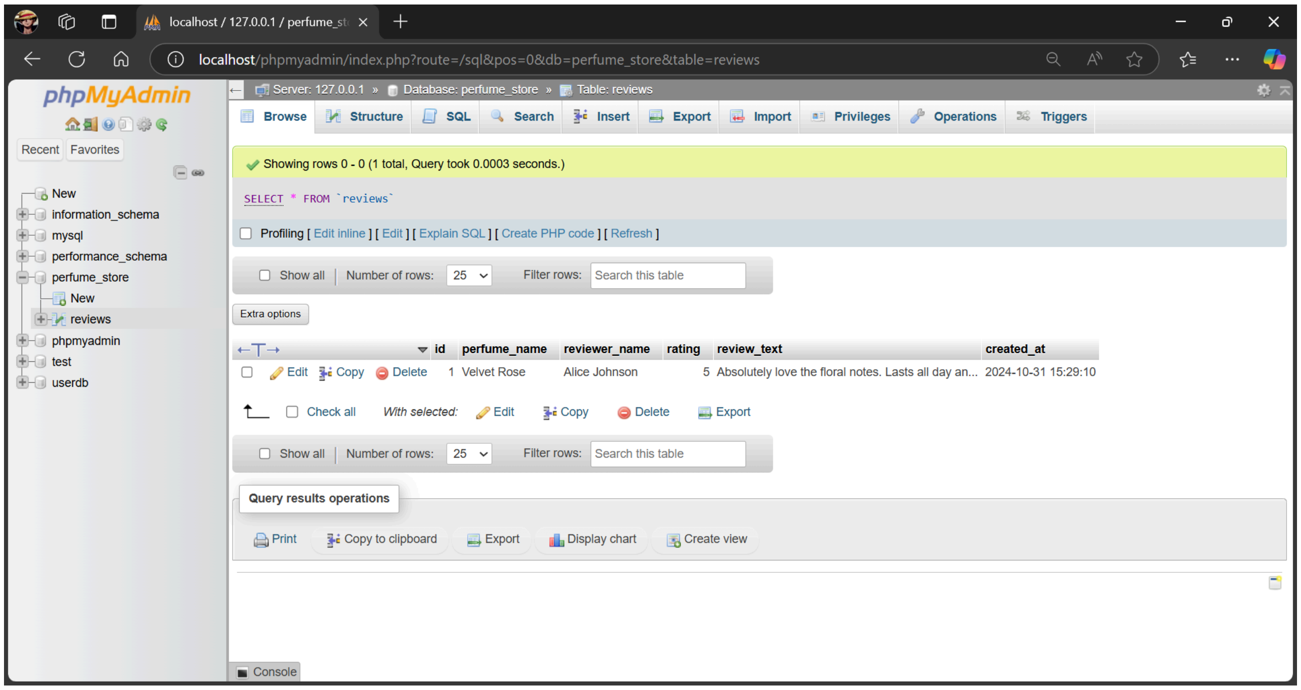
Rating (1 to 5): 5

Review: Absolutely love the floral notes. Lasts all day and has a luxurious feel!

Submit Review



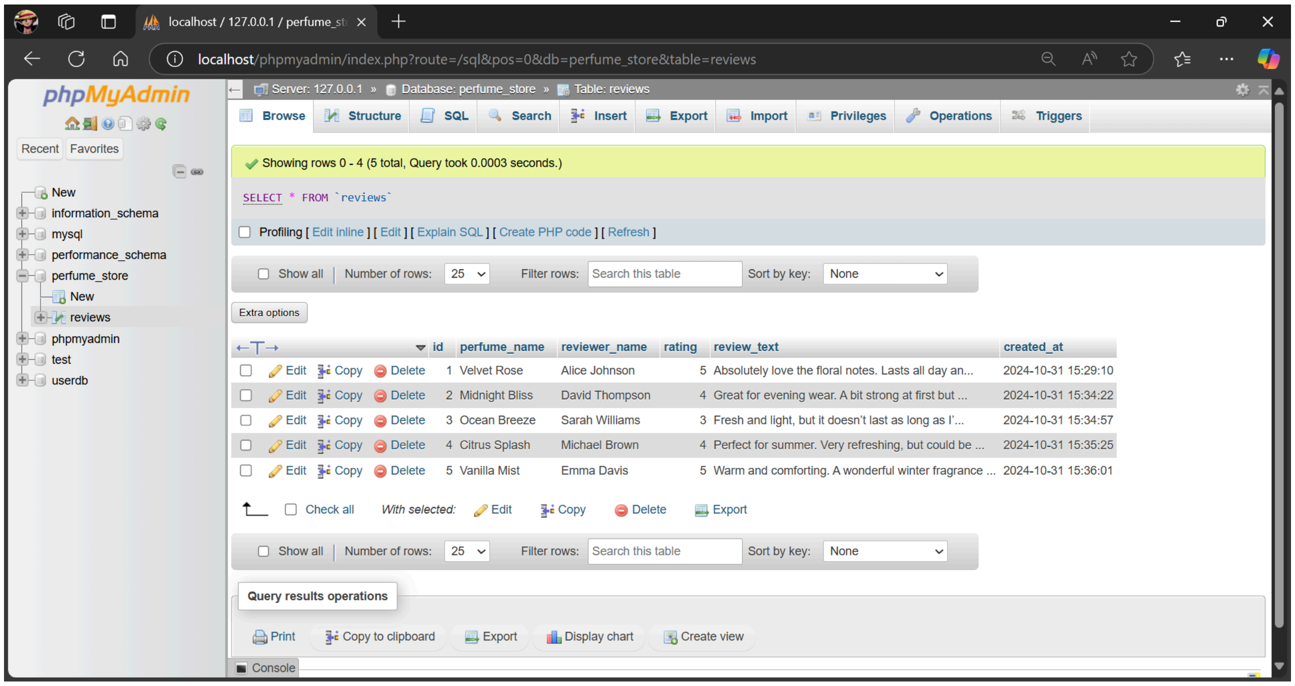
Thank you for your review!



Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)

```
SELECT * FROM `reviews`
```

| id | perfume_name | reviewer_name | rating | review_text | created_at |
|---|---|---|---|---|---|
| 1 | Velvet Rose | Alice Johnson | 5 | Absolutely love the floral notes. Lasts all day an... | 2024-10-31 15:29:10 |


localhost/phpmyadmin/index.php?route=/sql&pos=0&db=perfume_store&table=reviews

Server: 127.0.0.1 » Database: perfume_store » Table: reviews

Showing rows 0 - 4 (5 total, Query took 0.0003 seconds.)

```
SELECT * FROM `reviews`
```

| id | perfume_name | reviewer_name | rating | review_text | created_at |
|---|---|---|---|---|---|
| 1 | Velvet Rose | Alice Johnson | 5 | Absolutely love the floral notes. Lasts all day an... | 2024-10-31 15:29:10 |
| 2 | Midnight Bliss | David Thompson | 4 | Great for evening wear. A bit strong at first but ... | 2024-10-31 15:34:22 |
| 3 | Ocean Breeze | Sarah Williams | 3 | Fresh and light, but it doesn't last as long as I'... | 2024-10-31 15:34:57 |
| 4 | Citrus Splash | Michael Brown | 4 | Perfect for summer. Very refreshing, but could be ... | 2024-10-31 15:35:25 |
| 5 | Vanilla Mist | Emma Davis | 5 | Warm and comforting. A wonderful winter fragrance ... | 2024-10-31 15:36:01 |

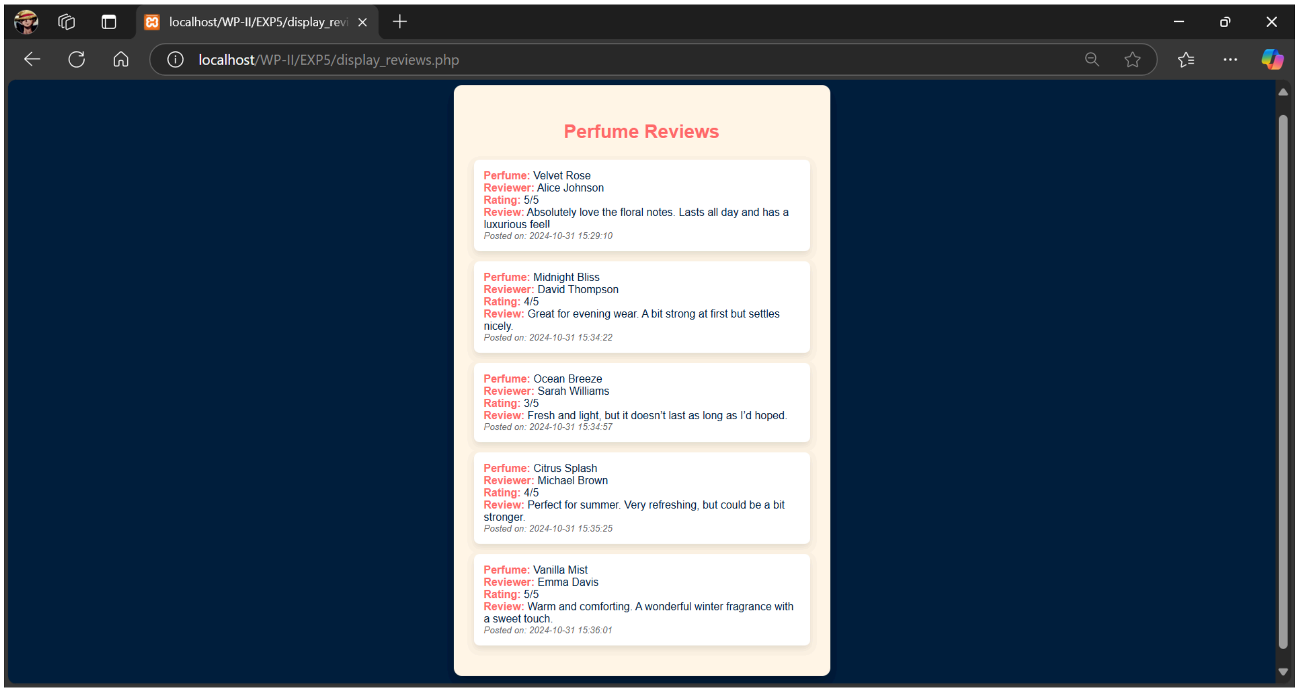
display_reviews.php

```
<link rel="stylesheet" href="styless.css">
<?php
// Database configuration
$host = 'localhost';
$dbname = 'perfume_store';
$user = 'root';
$pass = '';

try {
    // Connect to the database
    $dbh = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);
    $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Retrieve all reviews
    $sql = "SELECT * FROM reviews";
    $stmt = $dbh->query($sql);
    $reviews = $stmt->fetchAll(PDO::FETCH_ASSOC);

    echo "<div class='reviews-container'>";
    echo "<h2>Perfume Reviews</h2>";
    foreach ($reviews as $review) {
        echo "<div class='review'>";
        echo "<strong>Perfume:</strong> " .
htmlspecialchars($review['perfume_name']) . "<br>";
        echo "<strong>Reviewer:</strong> " .
htmlspecialchars($review['reviewer_name']) . "<br>";
        echo "<strong>Rating:</strong> " .
htmlspecialchars($review['rating']) . "/5<br>";
        echo "<strong>Review:</strong> " .
htmlspecialchars($review['review_text']) . "<br>";
        echo "<small><em>Posted on: " . $review['created_at'] .
"</em></small>";
        echo "</div>";
    }
    echo "</div>";

} catch (PDOException $e) {
    echo "Error: " . $e->getMessage();
}
?>
```

styless.css

```
/* Body styling */
body {
    font-family: Arial, sans-serif;
    background-color: #001f3f; /* Dark navy */
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 20px;
    color: #ffffff;
}

/* Container styling */
.reviews-container {
    background-color: #fff5e6; /* Light peach */
    width: 500px;
    padding: 30px;
    border-radius: 12px;
    box-shadow: 0px 10px 20px rgba(0, 0, 0, 0.3);
    text-align: center;
}
```

```css
/* Title styling */
.reviews-container h2 {
    font-size: 28px;
    color: #ff6b6b; /* Soft coral */
    margin-bottom: 25px;
}

/* Individual review styling */
.review {
    background-color: #ffffff;
    padding: 15px;
    margin-bottom: 15px;
    border-radius: 8px;
    text-align: left;
    color: #001f3f;
    box-shadow: 0px 5px 10px rgba(0, 0, 0, 0.1);
}

/* Review text styling */
.review strong {
    color: #ff6b6b; /* Coral color for labels */
}

.review small {
    color: #666666; /* Light gray for date */
}
```


localhost/WP-II/EXP5/display_revi
localhost/WP-II/EXP5/display_reviews.php

Perfume Reviews

**Perfume:** Velvet Rose
**Reviewer:** Alice Johnson
**Rating:** 5/5
**Review:** Absolutely love the floral notes. Lasts all day and has a luxurious feel!
*Posted on: 2024-10-31 15:29:10*

**Perfume:** Midnight Bliss
**Reviewer:** David Thompson
**Rating:** 4/5
**Review:** Great for evening wear. A bit strong at first but settles nicely.
*Posted on: 2024-10-31 15:34:22*

**Perfume:** Ocean Breeze
**Reviewer:** Sarah Williams
**Rating:** 3/5
**Review:** Fresh and light, but it doesn’t last as long as I’d hoped.
*Posted on: 2024-10-31 15:34:57*

**Perfume:** Citrus Splash
**Reviewer:** Michael Brown
**Rating:** 4/5
**Review:** Perfect for summer. Very refreshing, but could be a bit stronger.
*Posted on: 2024-10-31 15:35:25*

**Perfume:** Vanilla Mist
**Reviewer:** Emma Davis
**Rating:** 5/5
**Review:** Warm and comforting. A wonderful winter fragrance with a sweet touch.
*Posted on: 2024-10-31 15:36:01*

update_form.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Update Review</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="update-container">
        <h2>Update Review</h2>
        <form action="update_review.php" method="POST">
            <label for="id">Review ID:</label>
            <input type="number" id="id" name="id" required><br><br>

            <label for="rating">New Rating (1 to 5):</label>
            <input type="number" id="rating" name="rating" min="1" max="5"
required><br><br>

            <label for="review_text">New Review Text:</label>
            <textarea id="review_text" name="review_text"
required></textarea><br><br>

            <input type="submit" value="Update Review">
        </form>
    </div>
</body>
</html>
```

styles.css

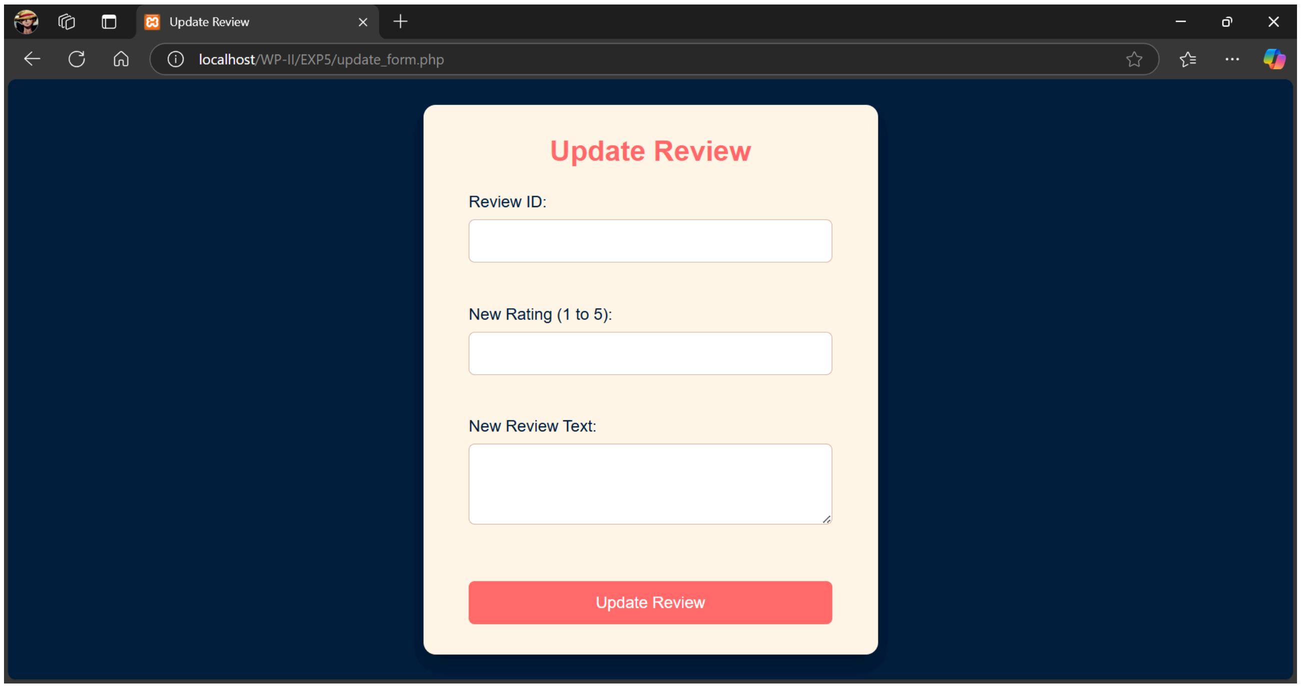
```
/* Basic reset */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* Body styling */
body {
    font-family: Arial, sans-serif;
    background-color: #001f3f; /* Dark navy */
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    color: #ffffff;
}

/* Container styling */
.update-container {
    background-color: #fff5e6; /* Light peach */
    width: 450px;
    padding: 30px;
    border-radius: 12px;
    box-shadow: 0px 10px 20px rgba(0, 0, 0, 0.3);
    text-align: center;
}

/* Title styling */
.update-container h2 {
    font-size: 28px;
    color: #ff6b6b; /* Soft coral */
    margin-bottom: 25px;
}
```

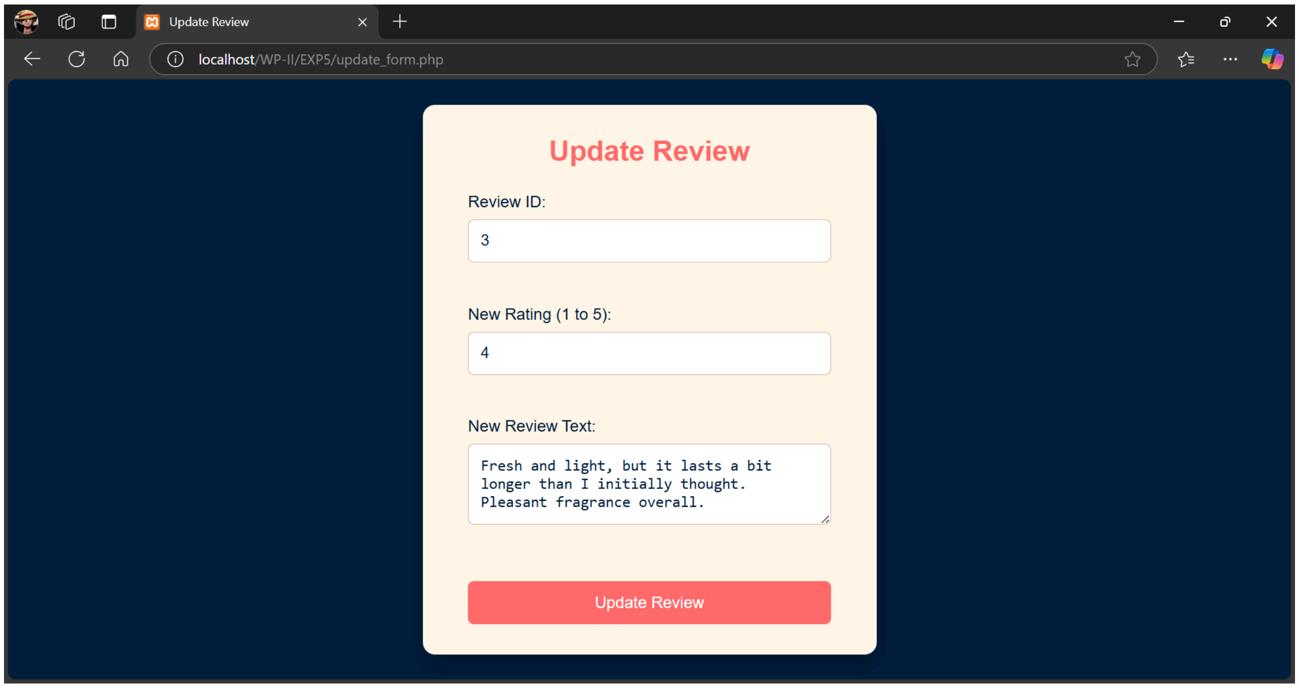
```css
/* Label styling */
.update-container label {
    display: block;
    font-size: 16px;
    color: #001f3f; /* Dark navy */
    margin-top: 15px;
    text-align: left;
    margin-left: 15px;
}

/* Input and textarea styling */
.update-container input[type="number"],
.update-container textarea {
    width: calc(100% - 30px); /* Ensures padding on both sides */
    padding: 12px;
    margin: 8px 15px; /* Equal margin on both sides */
    border: 1px solid #d9b5a5; /* Soft peach */
    border-radius: 6px;
    background-color: #ffffff; /* White for strong contrast */
    font-size: 15px;
    color: #001f3f;
}

/* Textarea styling adjustments */
.update-container textarea {
    resize: vertical;
    height: 80px;
}

/* Submit button styling */
.update-container input[type="submit"] {
    background-color: #ff6b6b; /* Coral */
    color: white;
    padding: 12px 20px;
    border: none;
    border-radius: 6px;
    font-size: 16px;
    margin-top: 25px;
    cursor: pointer;
    transition: background-color 0.3s ease;
    width: calc(100% - 30px); /* Matches input width */
}
```

```
.update-container input[type="submit"]:hover {
    background-color: #ff4c4c; /* Darker coral */
}
```

update_review.php

```
<?php
// Database configuration
$host = 'localhost';
$dbname = 'perfume_store';
$user = 'root';
$pass = '';

try {
    // Connect to the database
    $dbh = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);
    $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        // Get form data
        $id = $_POST['id'];
        $rating = $_POST['rating'];
        $review_text = $_POST['review_text'];

        // Update the review in the database
        $sql = "UPDATE reviews SET rating = :rating, review_text =
:review_text WHERE id = :id";
        $stmt = $dbh->prepare($sql);
        $stmt->bindParam(':id', $id, PDO::PARAM_INT);
        $stmt->bindParam(':rating', $rating, PDO::PARAM_INT);
        $stmt->bindParam(':review_text', $review_text, PDO::PARAM_STR);

        if ($stmt->execute()) {
            echo "<p style='color:green;'>Review updated successfully!</p>";
        } else {
            echo "<p style='color:red;'>Failed to update the review. Please
check the ID and try again.</p>";
        }
    }
} catch (PDOException $e) {
    echo "Error: " . $e->getMessage();
}
?>
```

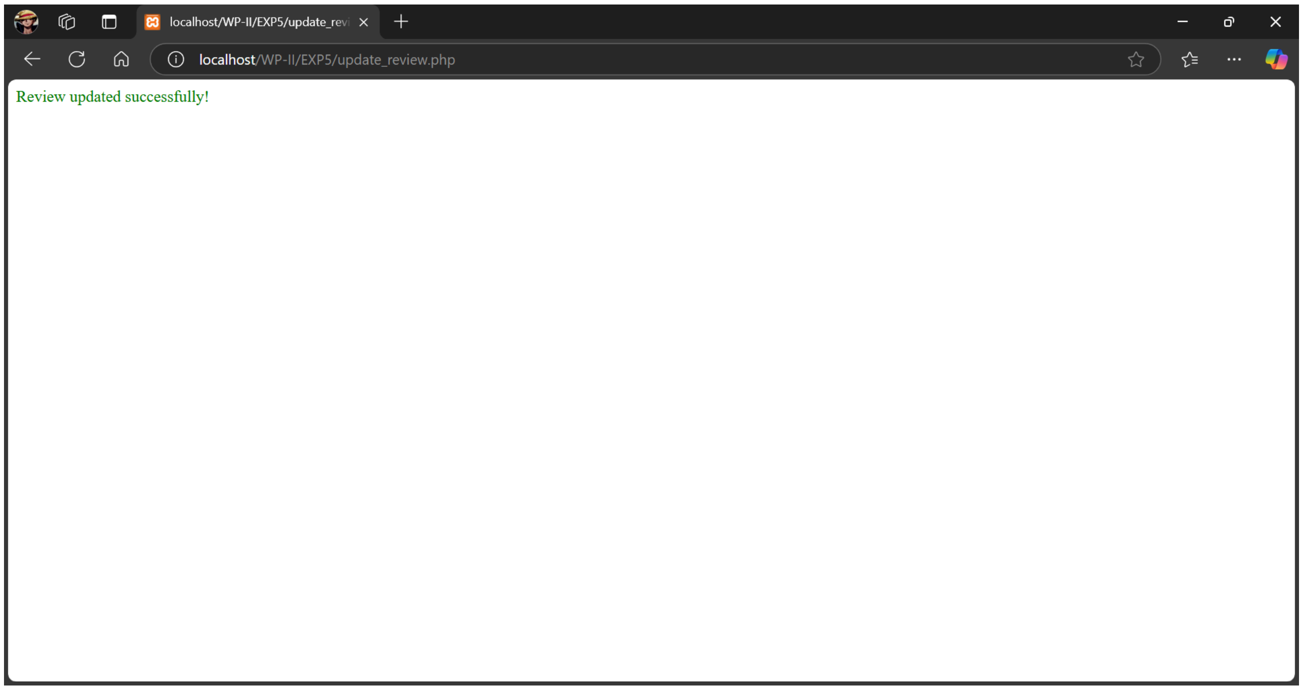




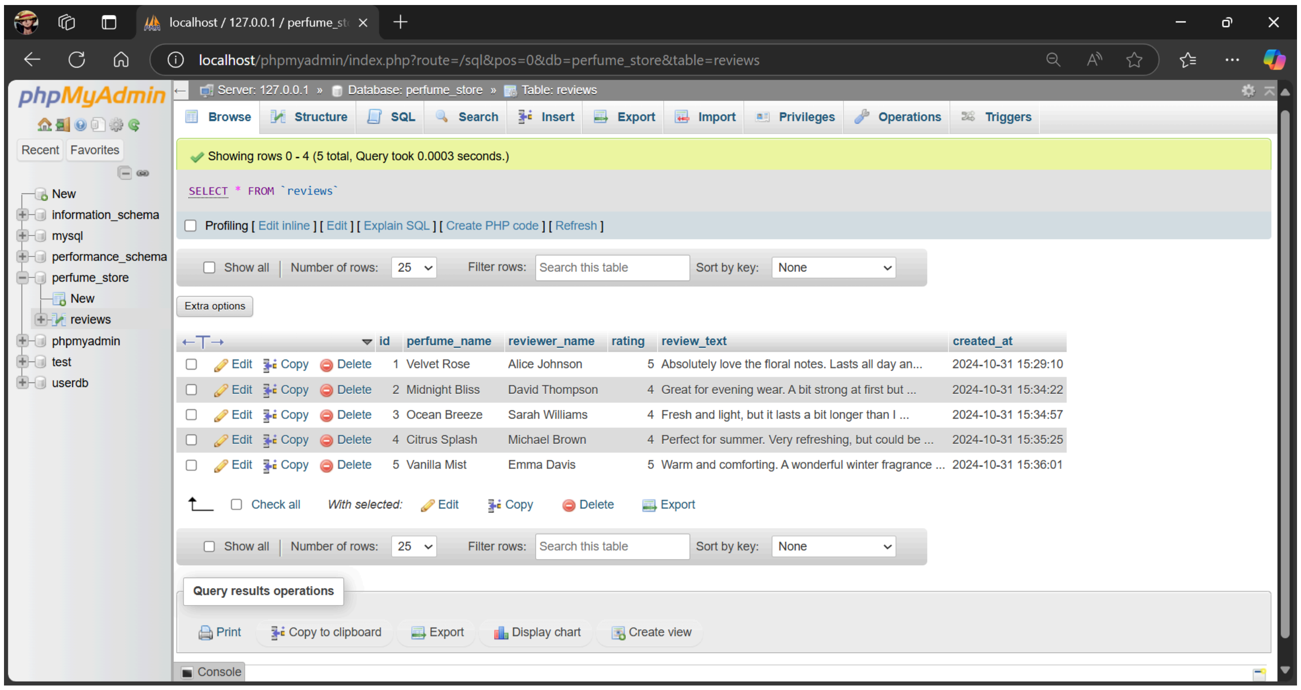
3

4

Fresh and light, but it lasts a bit longer than I initially thought. Pleasant fragrance overall.


localhost/WP-II/EXP5/update_revi
localhost/WP-II/EXP5/update_review.php

Review updated successfully!


localhost / 127.0.0.1 / perfume_st
localhost/phpmyadmin/index.php?route=/sql&pos=0&db=perfume_store&table=reviews
phpMyAdmin
Recent Favorites
New
information_schema
mysql
performance_schema
perfume_store
New
reviews
phpmyadmin
test
userdb
Server: 127.0.0.1 » Database: perfume_store » Table: reviews
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options
Check all With selected: Edit Copy Delete Export
Query results operations
Print Copy to clipboard Export Display chart Create view
Console

Showing rows 0 - 4 (5 total, Query took 0.0003 seconds.)

```
SELECT * FROM `reviews`
```

| | id | perfume_name | reviewer_name | rating | review_text | created_at |
|---|---|---|---|---|---|---|
| Edit Copy Delete | 1 | Velvet Rose | Alice Johnson | 5 | Absolutely love the floral notes. Lasts all day an... | 2024-10-31 15:29:10 |
| Edit Copy Delete | 2 | Midnight Bliss | David Thompson | 4 | Great for evening wear. A bit strong at first but ... | 2024-10-31 15:34:22 |
| Edit Copy Delete | 3 | Ocean Breeze | Sarah Williams | 4 | Fresh and light, but it lasts a bit longer than I ... | 2024-10-31 15:34:57 |
| Edit Copy Delete | 4 | Citrus Splash | Michael Brown | 4 | Perfect for summer. Very refreshing, but could be ... | 2024-10-31 15:35:25 |
| Edit Copy Delete | 5 | Vanilla Mist | Emma Davis | 5 | Warm and comforting. A wonderful winter fragrance ... | 2024-10-31 15:36:01 |

---

**Post Lab Questions:**

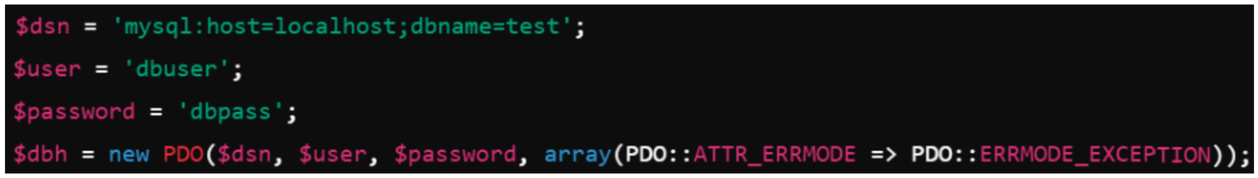
1) **Write in detail the importance of using Prepared Statements.**

Prepared statements are essential in database operations for two main reasons:

- Security: By separating SQL queries from user input, prepared statements effectively prevent SQL injection attacks, making the application more secure.
- Efficiency: Prepared statements allow the database to parse and compile the SQL

statement only once, even if it's executed multiple times with different parameters. This reduces the processing overhead for repeated executions and enhances performance.

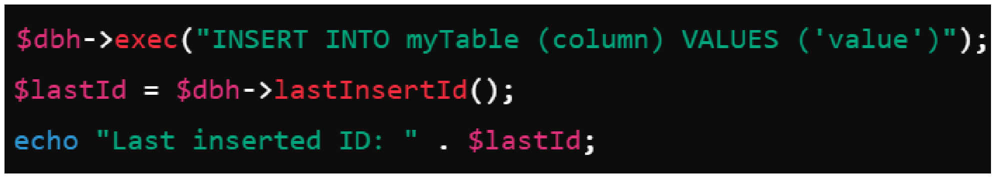
2) **Write in detail about PDO constructor function.**
The PDO constructor function initializes a connection to the database with parameters for the DSN (Data Source Name), username, and password, as well as an optional array for PDO attributes. Here's an example:



```php
$dsn = 'mysql:host=localhost;dbname=test';
$user = 'dbuser';
$password = 'dbpass';
$dbh = new PDO($dsn, $user, $password, array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));
```

The DSN specifies the database type, host, and database name, and attributes can control error handling, persistence, and other connection settings.

3) **Write a php program to return the id of the last inserted row.**
To get the last inserted row's ID, use the lastInsertId() method of the PDO object:



```php
$dbh->exec("INSERT INTO myTable (column) VALUES ('value')");
$lastId = $dbh->lastInsertId();
echo "Last inserted ID: " . $lastId;
```

4) **What are the essential components of a PDO database connection string?**
The PDO connection string (DSN) typically includes:

- Database Type (e.g., mysql, pgsql)
- Host (server location, e.g., localhost or IP)
- Database Name (name of the database to connect) Additional components can be specified, such as port numbers or charset.

5) **Explain the concept of database connection pooling and how it can be implemented with PDO.**
Connection pooling allows the reuse of database connections for multiple requests, reducing the overhead of creating new connections. While PDO doesn't natively support connection pooling, persistent connections (PDO::ATTR_PERSISTENT) mimic pooling by keeping a connection open between requests, allowing efficient reuse without opening and closing connections for each request.

---

**Outcomes: Carry out database operations using PHP**

---

**Conclusion: (Conclusion to be based on the objectives and outcomes achieved)**
The experiment successfully demonstrated how to interact with a MySQL database using PHP PDO. Key outcomes achieved include establishing secure and efficient connections, utilizing prepared statements to prevent SQL injection, and performing basic CRUD operations with exception handling for robust error management. The exercise highlights PDO’s versatility and the benefits of using an abstraction layer in PHP to access different databases with consistent functionality.

---

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of faculty in-charge with date**

---

**References:**

1) Thomson PHP and MySQL Web Development Addison-Wesley Professional , 5th Edition2016.
2) https://www.php.net/manual/en/book.pdo.php

---