

## Tutorial/Laboratory 08 (Part 2)

# Database Operations with PHP and MySQL

### 1. LEARNING OUTCOMES

Upon completion of these laboratory exercises, you should be able to:

- Use MySQL Workbench to create databases and tables.
- Implement database functionality in a website using PHP and MySQL.

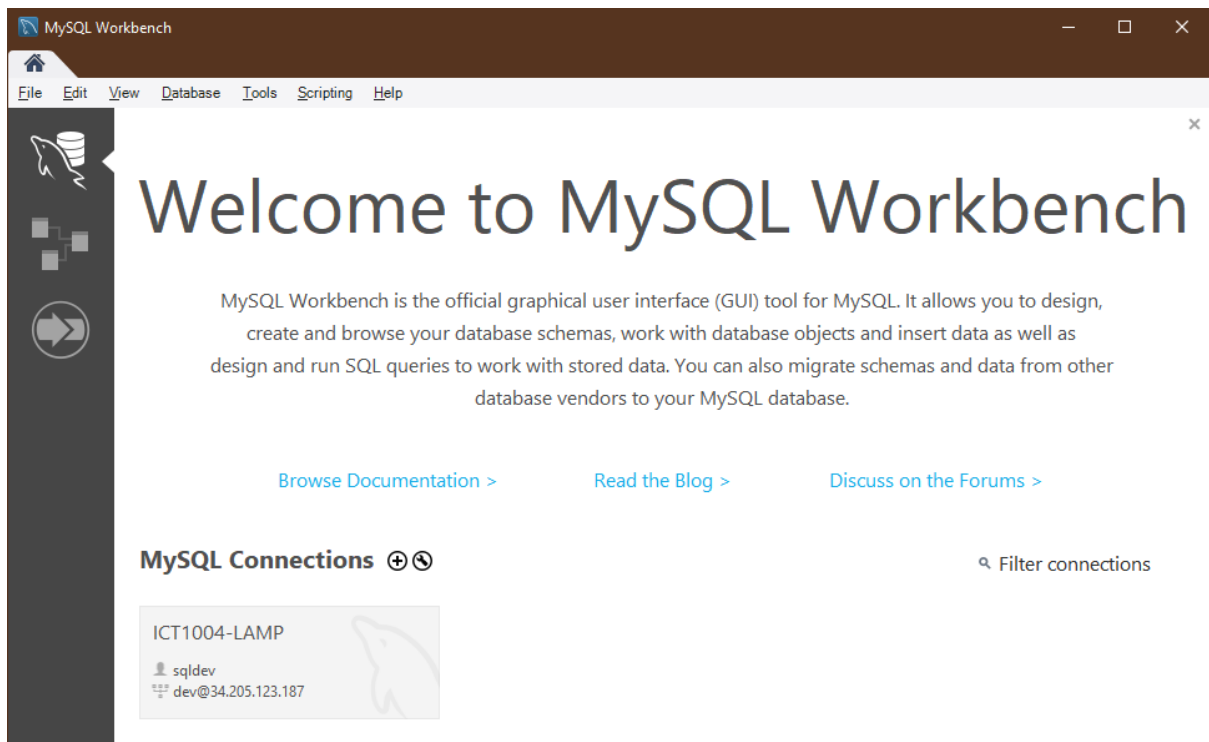
### 2. REQUIRED SOFTWARE

- Apache NetBeans 11.2 (or later):  
<https://netbeans.apache.org/download/index.html>
- FireFox (<https://www.mozilla.org/en-US/firefox/new/>) or Chrome (<https://www.google.com/chrome/>) web browser.
- AWS Educate account with EC2 instance running Ubuntu Server & LAMP stack (<https://www.awseducate.com/signin/SiteLogin>).
- MySQL Workbench (<https://dev.mysql.com/downloads/workbench/>)

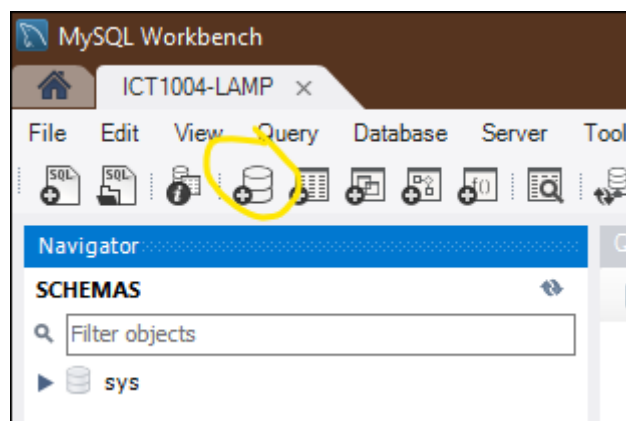
### 3. EXERCISE 1: CREATING A NEW DATABASE AND TABLE

In Part 1 of this Lab, we configured our database server with a static IP address and non-root development account, and we established a connection to the server using MySQL Workbench. In this exercise, we'll continue where we left off and add a database (schema) and table.

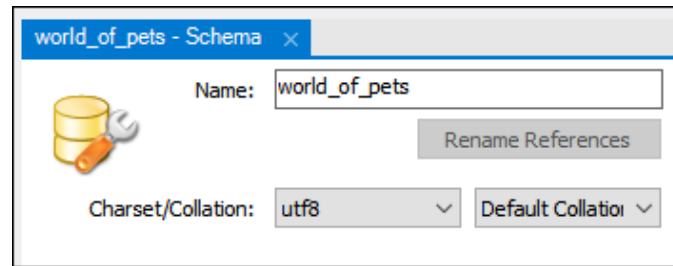
- 3.1 Open MySQL Workbench and click on the server connection that you created in Part 1 (be sure the AWS instance is running).



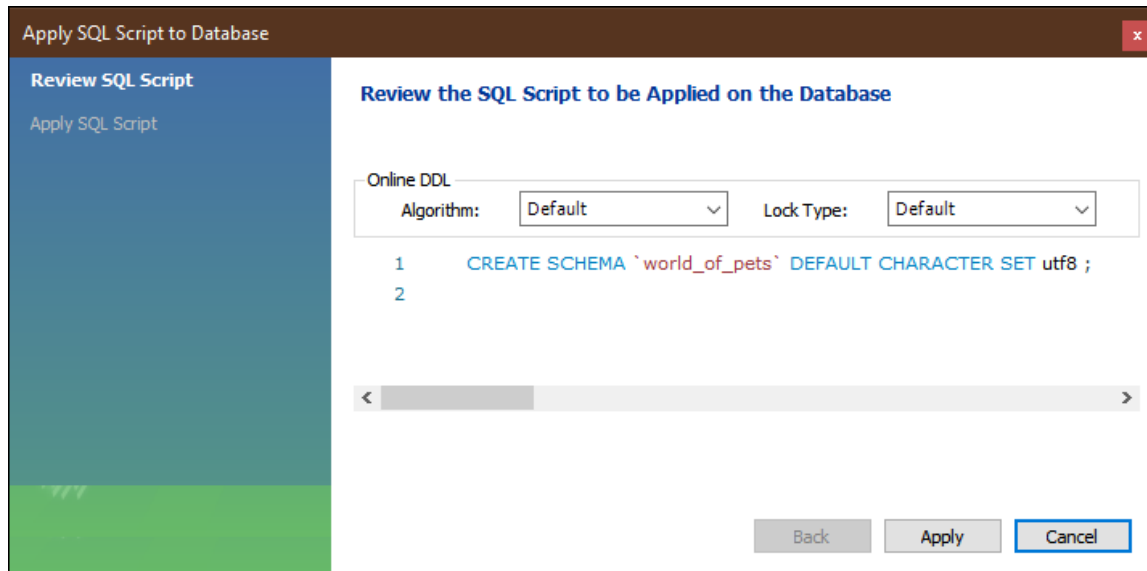
- 3.2 After clicking on the connection, you should see the “SCHEMAS” tab on the left. right-click here, or click on the “Create a new schema...” icon to create a new database.



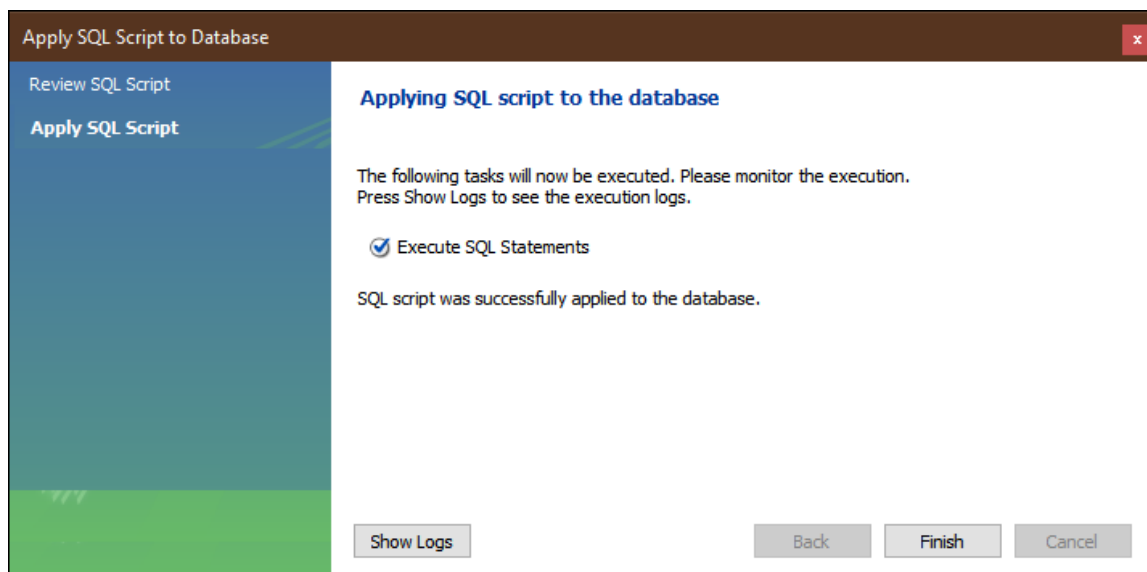
- 3.3 In the schema configuration panel, give the database a name, say “world\_of\_pets” since this will be used with the World of Pet website we’ve been building throughout this course. Choose utf8 for the default character set, and leave collation set to default.



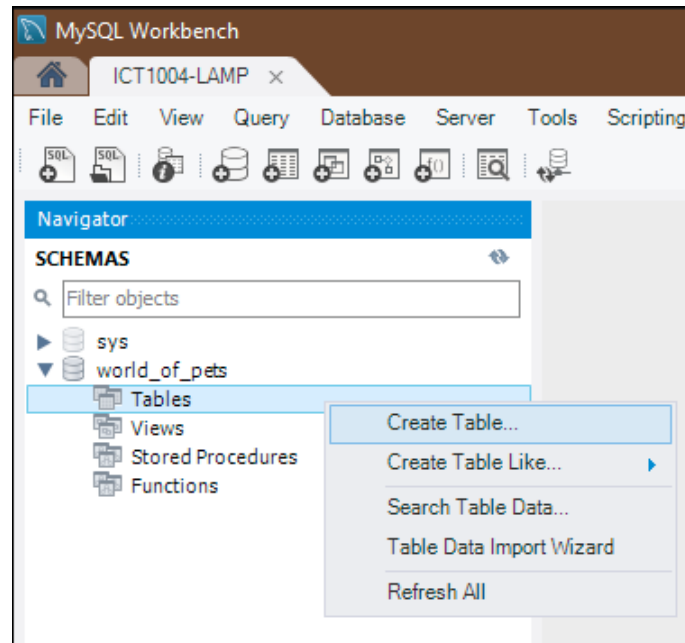
3.4 Click “Apply” to bring up the script review dialog.



3.5 Click “Apply” once again to create the schema. On success, you should see the following confirmation. Click “Finish” to complete the schema creation process.



3.6 Next, we’ll create a table. In the “SCHEMAS” tab, expand the newly created “world\_of\_pets” database, right-click on “Tables” and select “Create Table” from the pop-up menu.








- 3.7 Name the table “world\_of\_pets\_members” or similar, as this will be where we store our members’ registration information. Complete the following steps to configure the table.
- Set the charset to utf8 again.
  - Click under “Column Name” and add a **primary key** field called “member\_id” with the following attributes: **primary, not null, unique, unsigned** and **auto increment**.
  - Add remaining columns for first name (**fname**), last name (**lname**), email address (**email**), and **password** as indicated below. Datatype for all can be the default VARCHAR(45), and all should be “Not Null” except for **fname**, since it’s possible for a person to have only one name.

world\_of\_pets\_members - Table

Table Name:  Schema: **world\_of\_pets**

Charset/Collation:   Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 member_id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 fname	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 lname	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 email	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 password	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name:  Data Type:

Charset/Collation:

Comments:

Storage: ☐ Virtual ☐ Stored

☒ Primary Key ☒ Not Null ☒ Unique

☐ Binary ☒ Unsigned ☐ Zero Fill

☒ Auto Increment ☐ Generated

Columns | Indexes | Foreign Keys | Triggers | Partitioning | Options

Apply Revert

- 3.8 Click “Apply” to create the new table. On the next screen, you can review the script to be applied, and make changes if desired.

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm:  Lock Type:

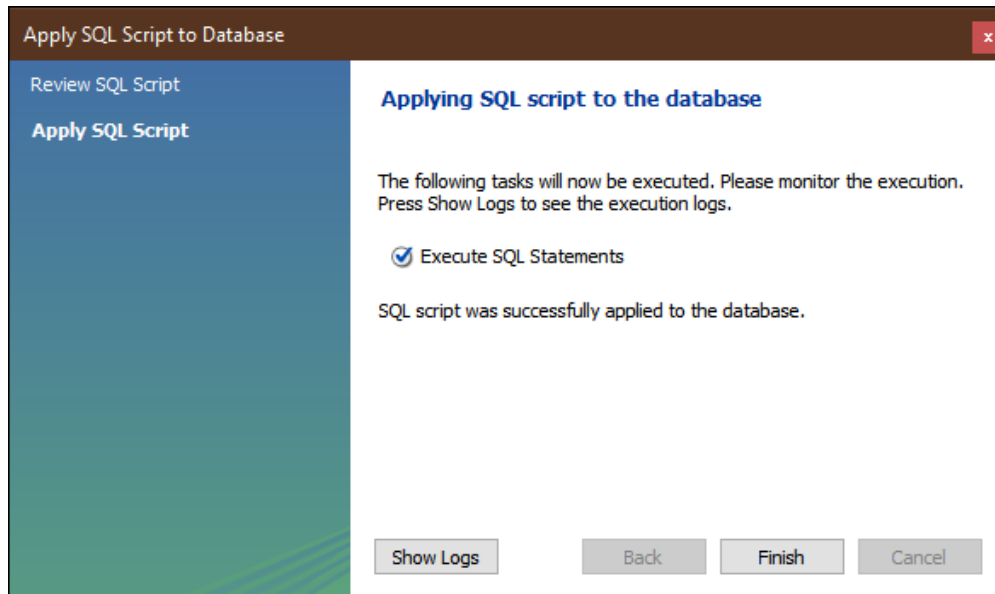
```

1  CREATE TABLE `world_of_pets`.`world_of_pets_members` (
2    `member_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
3    `fname` VARCHAR(45) NULL,
4    `lname` VARCHAR(45) NOT NULL,
5    `email` VARCHAR(45) NOT NULL,
6    `password` VARCHAR(45) NOT NULL,
7    PRIMARY KEY (`member_id`),
8    UNIQUE INDEX `member_id_UNIQUE` (`member_id` ASC))
9  ENGINE = InnoDB
10 DEFAULT CHARACTER SET = utf8;
11

```

Back Apply Cancel

- 3.9 Finally, hit “Apply” to create the table. You can view the logs or click “Finish” when done.



#### 4. EXERCISE 2: DATABASE OPERATIONS WITH PHP

- 4.1 Now that our database and table are set up, we can proceed to implement database functionality in our website. Start by making a copy of your latest World of Pets web application in NetBeans.

**Be sure to edit the project properties and update the Run Configuration->Remote Connection with the static IP address you created in Part 1.**

- 4.2 Firstly, we will store our database username and password in a secure location on the web server. You should avoid hard coding passwords in PHP source files, as a bug or runtime error could cause the PHP code to be displayed unparsed in the browser. Also, anyone with access to the source files would be able to see the password.

We'll use the common technique of storing the database credentials in a .ini file *outside* the root web directory with limited permissions. While this is not as stringent as hashing and more elaborate methods, it does provide some degree of security.

- a. Open a SSH session to your LAMP server and enter these commands to create the private database configuration file:

```
$ sudo mkdir /var/www/private
$ sudo nano /var/www/private/db-config.ini
```

- b. In the nano editor, add the text below. Replace “yourpassword” with the password you used for the database user (e.g. sqldev) you added in Part 1. The **dbname** should be set to whatever name you used when creating the database in 3.7 above. Note that **servername** is set to “localhost”, *not* the IP address of the Ubuntu Server instance. This is because the connection to the MySQL server is

relative to the Apache web server running on the same host. If the MySQL server were running on a different machine, then you could use that server's name to connect remotely.

```
[database]
servername = localhost
username = sqldev
password = yourpassword
dbname = world_of_pets
```

c. Save the file (CTRL-X, Y, ENTER).

- 4.3 Now we'll modify `process_register.php` so that we save the user's information in the database once it has been validated. The code fragment below shows how to retrieve the database login credentials from the config file, and how to write to a database using PHP's object-oriented MySQL extension (MySQLi). Your solution may vary but you should be able to adapt the concepts here to your own application. You may refer to the PHP website or [W3Schools](http://www.w3schools.com) for more details on MySQLi.

**Note:** the function `saveMemberToDB()` should only be called after all form data has been successfully validated.

```
<?php

/*
 * Helper function to write the member data to the DB
 */
function saveMemberToDB()
{
    global $fname, $lname, $email, $pwd, $errorMsg, $success;

    // Create database connection.
    $config = parse_ini_file('../private/db-config.ini');
    $conn = new mysqli($config['servername'], $config['username'],
        $config['password'], $config['dbname']);

    // Check connection
    if ($conn->connect_error)
    {
        $errorMsg = "Connection failed: " . $conn->connect_error;
        $success = false;
    }
    else
    {
        $sql = "INSERT INTO world_of_pets_members (fname, lname, email, password)";
        $sql .= " VALUES ('$fname', '$lname', '$email', '$pwd')";

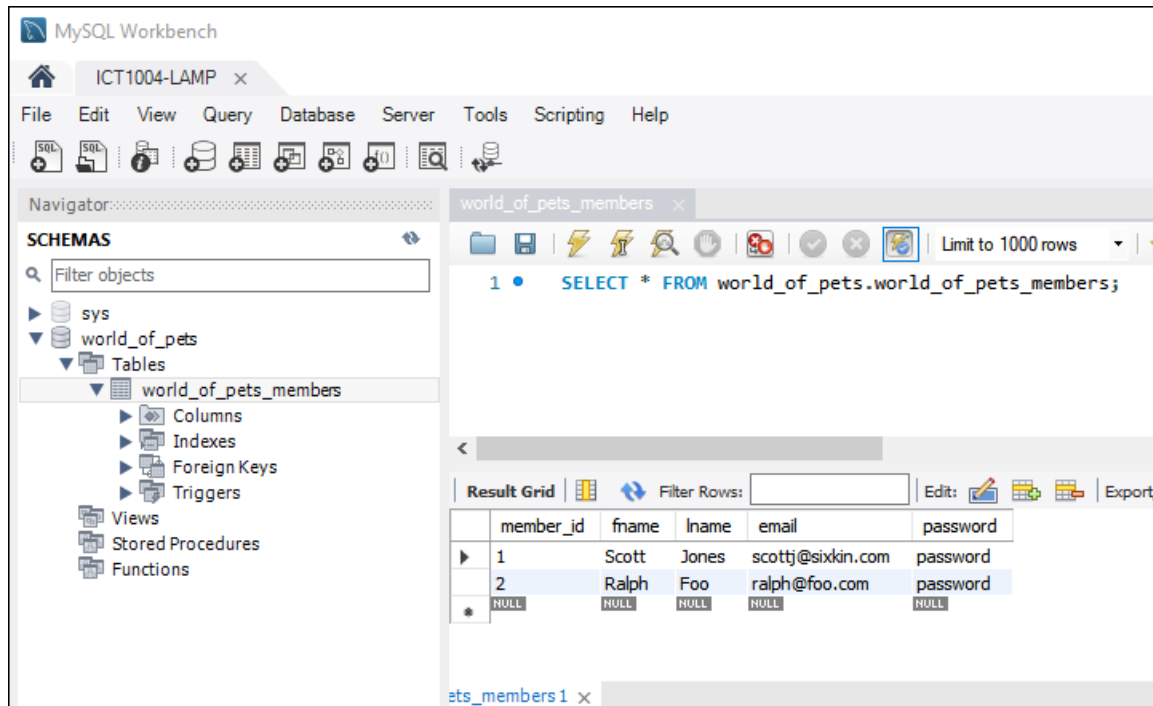
        // Execute the query
        if (!$conn->query($sql))
        {
            $errorMsg = "Database error: " . $conn->error;
            $success = false;
        }
    }

    $conn->close();
}

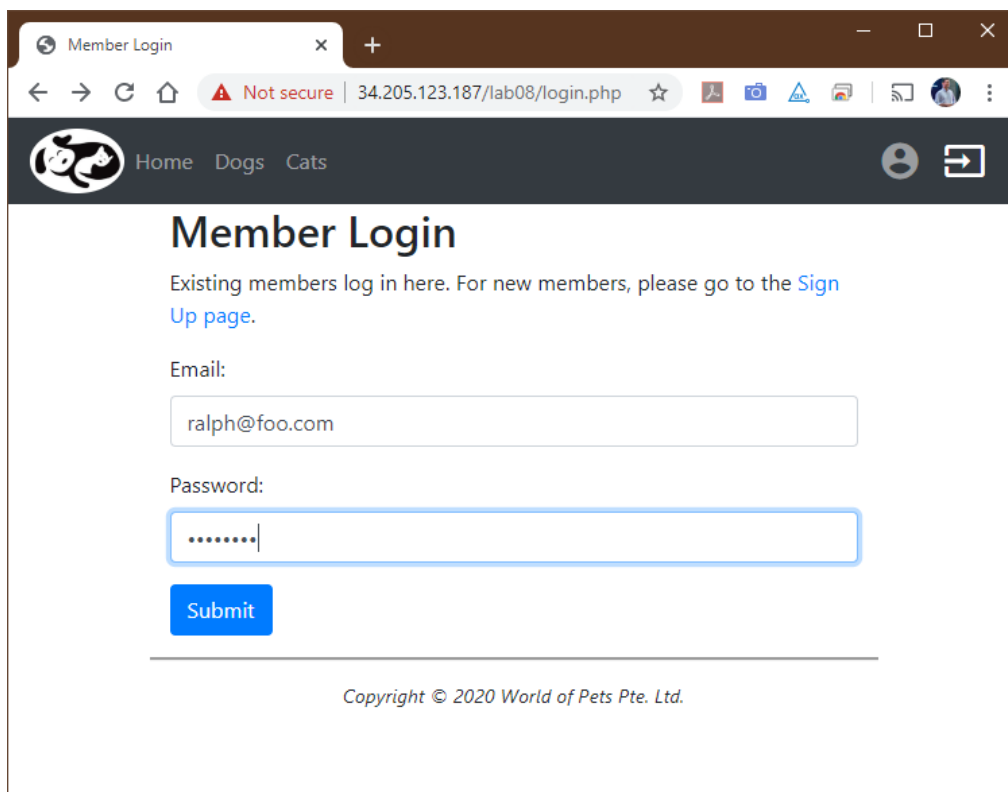
?>
```

- 4.4 Test your website by registering several new members, then view the table in MySQL Workbench to verify the entries.

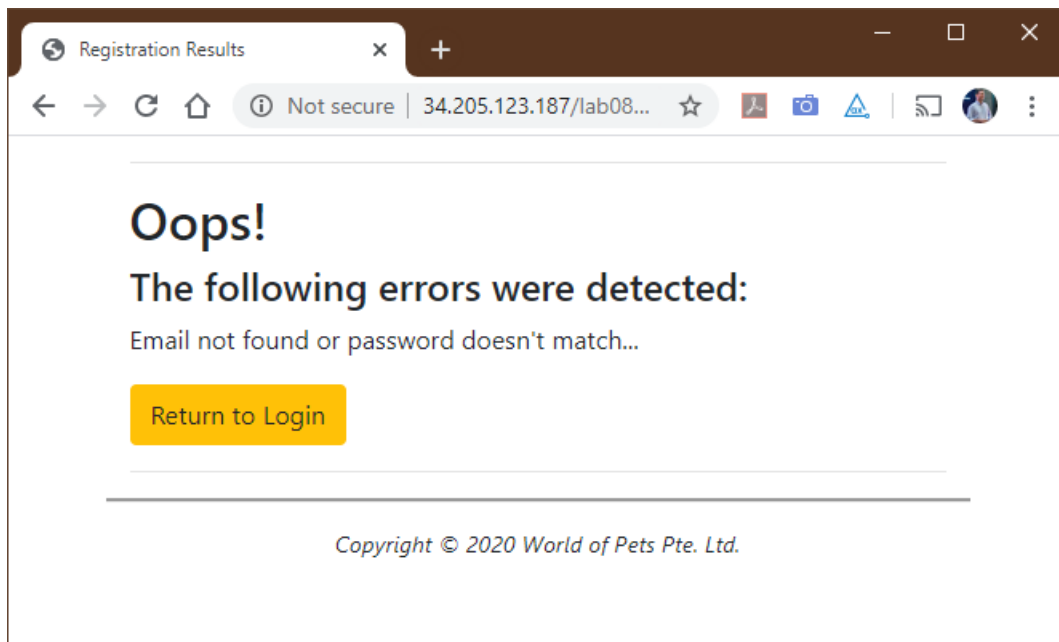




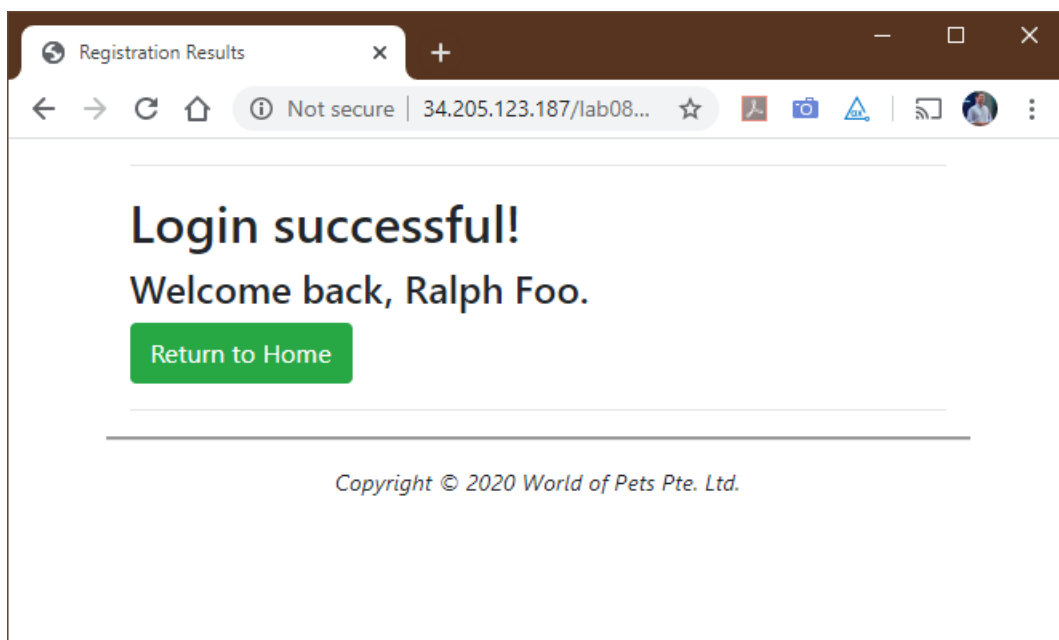
- 4.5 Now we need to add a login page (login.php). This should contain a form for the user to enter their email address and password. Set the action attribute to point to process\_login.php, which we'll create in the next step (e.g. `action="process_login.php"`). Remember to update your navigation menu to link to the new login page.



- 4.6 Next, we need to add the PHP page that processes the login (process\_login.php). In this page, you'll need to write PHP code that queries the database for the given user (don't forget to validate & sanitize the input). If they don't exist or if the password doesn't match, notify the user accordingly, e.g.:



...and if successful, you can display the user's info to confirm their login:



The following code fragment shows how you can retrieve records from the database matching a specific email address and password. Your solution may vary but you should be able to adapt the concepts here to your own website.

```
<?php

/*
 * Helper function to authenticate the login.
 */
function authenticateUser()
{
    global $fname, $lname, $email, $pwd, $errorMsg, $success;

    // Create database connection.
    $config = parse_ini_file('../private/db-config.ini');
    $conn = new mysqli($config['servername'], $config['username'],
        $config['password'], $config['dbname']);

    // Check connection
    if ($conn->connect_error)
    {
        $errorMsg = "Connection failed: " . $conn->connect_error;
        $success = false;
    }
    else
    {
        $sql = "SELECT * FROM world_of_pets_members WHERE ";
        $sql .= "email='$email' AND password='$pwd'";

        // Execute the query
        $result = $conn->query($sql);
        if ($result->num_rows > 0)
        {
            // Note that email field is unique, so should only have
            // one row in the result set.
            $row = $result->fetch_assoc();
            $fname = $row["fname"];
            $lname = $row["lname"];
        }
        else
        {
            $errorMsg = "Email not found or password doesn't match...";
            $success = false;
        }
        $result->free_result();
    }

    $conn->close();
}

?>
```

## 5. NEXT STEPS

This exercise has given you the basic foundation for setting up MySQL databases and implementing data access in your website. But this is only the bare essentials - here are a few suggestions for continuation that you may consider for your project and other websites:

- a. Add persistent login. In our example, the login page verifies that the username and password are correct in the database, but nothing actually changes after

logging in. In a proper web application, you'd use session variables or another method for keeping track of the logged in user, and a permissions system to determine which resources are only available to an authenticated user.

- b. Add a logout function. Obviously, we would want to also add a logout function, which would close any sessions and clear persistent information.
- c. Encrypt passwords and sensitive information. We didn't cover it in this module and it's left as a bonus for the project, but in a real-world application, you would always encrypt user passwords before storing them in the database. You would also use SSL to protect the data between the browser and the server.

## 6. SUBMISSION OF LAB ASSIGNMENT

- 6.1 In order to receive credit for this Lab assignment, you must submit your completed work to xSiTe LMS. To submit your work:
  - a. Save all files and close NetBeans.
  - b. In File Explorer, navigate to the location where you saved your project and right-click on the folder name, then select 'Send to -> Compressed (zipped) folder' and ZIP up your entire NetBeans project(s). **Note: only .zip format is acceptable, do not use .rar, .7z, or any other format.**
  - c. In the ICT1004 module on xSiTe, go to **Assessments->DropBox** and locate the Dropbox folder corresponding to this Lab. Click the link to open the Dropbox then hit the **Add a File** button to submit your .zip file. You may also add comments if desired. Be sure to hit **Submit** to complete your submission.