Tutorial/Laboratory 09

# Database Operations with PHP and MySQL

## 1. LEARNING OUTCOMES

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

- Add and grant permissions to non-root MySQL users.
- Create a database server connection in MySQL Workbench.
- Use MySQL Workbench to create databases and tables.
- Implement database functionality in a website using PHP and MySQL.

## 2. REQUIRED SOFTWARE

- Google Cloud account with VM instance running Ubuntu 22.04 and LAMP stack (Lab08).
- MySQL Workbench (https://dev.mysql.com/downloads/workbench/)

## 3. ADD A NON-ROOT MYSQL USER AND GRANT PERMISSIONS

In the previous Lab, we installed MySQL with the default **root** user and locked it down for security reasons. We need to add a new non-root user for development purposes that has only the necessary permissions for managing databases.

3.1 First, be sure you are on the ICT Wi-fi network or your own hotspot, since port 22 (SSH) is blocked on the SIT Wi-fi. Then log into your Ubuntu instance using SSH and follow the steps below to create the new user.

a. Run the MySQL monitor using the command below to log in as the MySQL **root** user. When prompted for the password, *enter the MySQL root password that you set previously in Lab08*.

```
$ sudo mysql -u root -p
```

b. With the MySQL monitor running, execute the command below to create the new user (`inf1005-sqldev` in this example). Replace 'password' with the password of your choice, and be sure to remember it! Also don't forget to include the semi-colon (;) at the end of statements in the MySQL monitor.

```
mysql> CREATE USER 'inf1005-sqldev'@'localhost' IDENTIFIED BY 'password';
```

c. Next, we need to grant privileges to the user. It's good practice to only grant the minimum privileges required. The `inf1005-sqldev` account will be used for carrying

out database operations for our web applications, so we need to give it all the CRUD related privileges.

```
mysql> GRANT CREATE, SELECT, INSERT, UPDATE, DELETE, REFERENCES, DROP, ALTER,
SHOW DATABASES ON * . * TO 'inf1005-sqldev'@'localhost';
```

d. Lastly, always flush privileges to force new ones to take effect.

```
mysql> FLUSH PRIVILEGES;
```

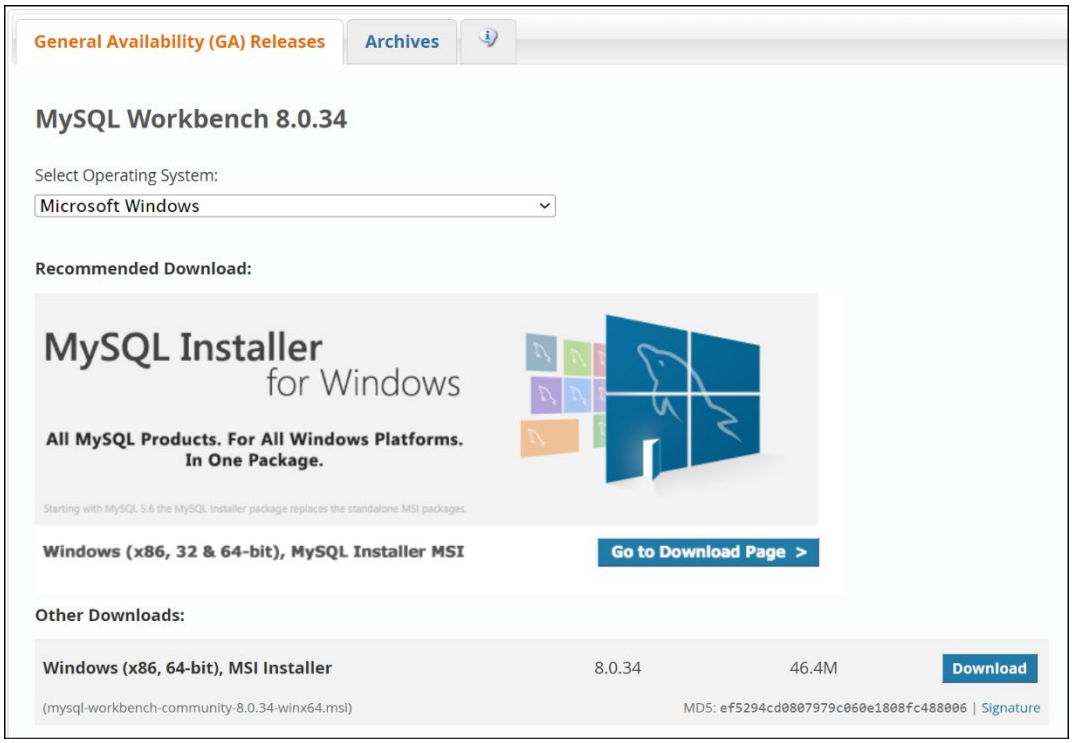
e. To check that privileges for `inf1005-sqldev` were set up properly, log out of the MySQL shell (type `exit` at the shell prompt) and log back in as `inf1005-sqldev`, then execute the `SHOW GRANTS` command (don't forget the semi-colon at the end).

```
inf1005-dev@inf1005-lamp:~$ mysql -u inf1005-sqldev -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.34-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW GRANTS;
+-----------------------------------------------------------------------------
-----------------------------------+
| Grants for inf1005-sqldev@localhost
|
+-----------------------------------------------------------------------------
-----------------------------------+
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, ALTER, SHOW
DATABASES ON *.* TO `inf1005-sqldev`@`localhost` |
+-----------------------------------------------------------------------------
-----------------------------------+
1 row in set (0.05 sec)

mysql> exit
Bye
inf1005-dev@inf1005-lamp:~$
```

## 4. EXERCISE 4: CONNECTING TO THE DATABASE SERVER USING MYSQL WORKBENCH

Now that the MySQL server is configured, we can try connecting to it. MySQL Workbench is a popular tool for managing databases securely using SSH tunneling.

4.1 Download and install MySQL Workbench.




You can skip the Oracle account login/signup and click on the '**No thanks, just start my download.**' link near the bottom of the page. Once the download is complete, run the installer (choose the "Complete" option when prompted). After the installation is complete, launch MySQL Workbench.

4.2 In MySQL Workbench, create a database connection. Click on the '+' sign next to "MySQL Connections" to add a connection.

a. In the "Setup New Connection" dialog box, enter a connection name of your choice (use a name that helps you remember what that database is for).

b. Change "Connection Method" to "Standard TCP/IP over SSH".

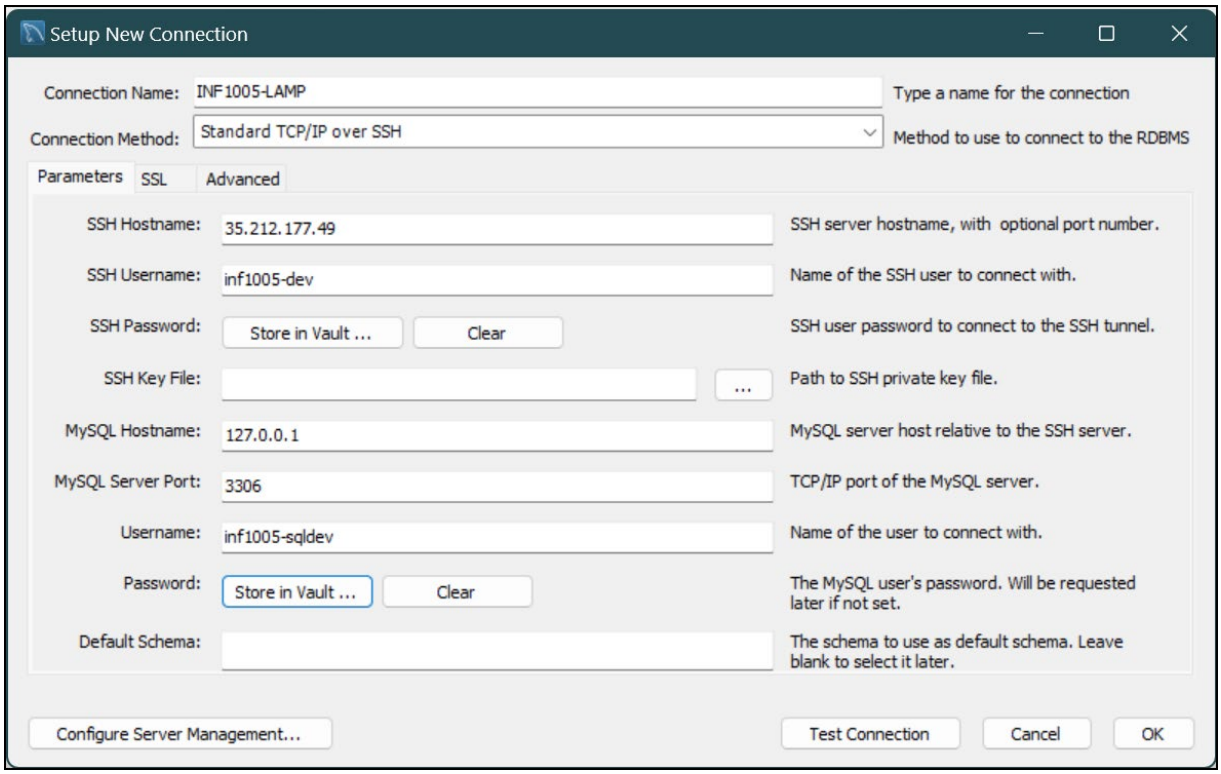
c. For "SSH Hostname" enter the static IP address of your LAMP server.

d. For "SSH Username" enter `inf1005-dev` (or whichever user account you SSH into the server with). This is *not* the MySQL user (that's further down).

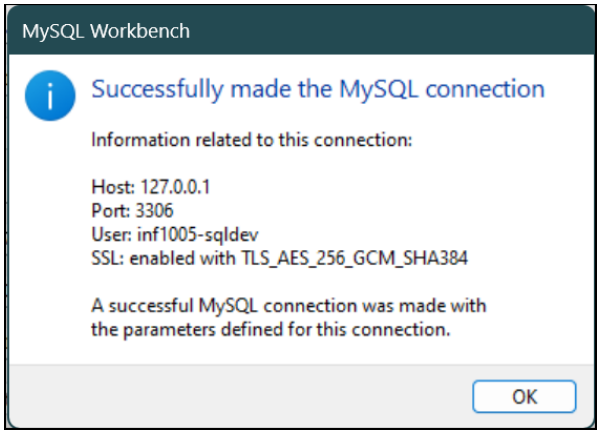
e. For “SSH Password” click the **Store in Vault…** button and enter the password you have been using to log in via SSH.

f. For “MySQL Hostname” and “MySQL Server Port” you can leave the default values. We’ll be logging into the database server as a local user (after tunneling through SSH).

g. For “Username” and “Password” enter the MySQL user (e.g., `inf1005-sqldev`) that you created earlier, and store that password in the vault.




4.3 Click the “Test Connection” button to see if everything works. You should see the message below if the connection was successful.

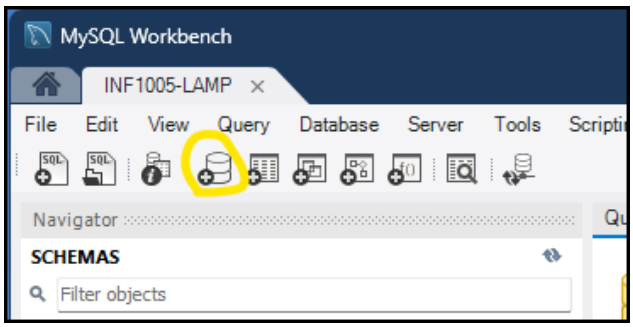



## 5. EXERCISE 5: CREATING A NEW DATABASE AND TABLE

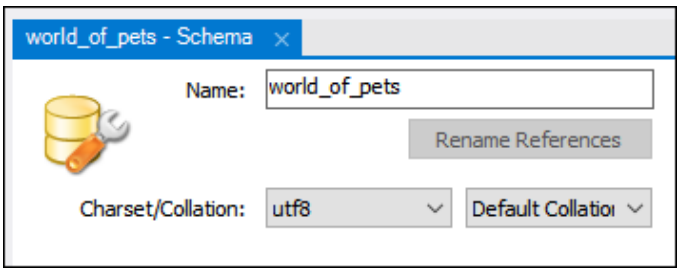
Now that we've established a connection to the server using MySQL Workbench, we can add a database (schema) and table.

5.1 Open MySQL Workbench and click on the server connection that you created earlier (be sure your LAMP instance is running).

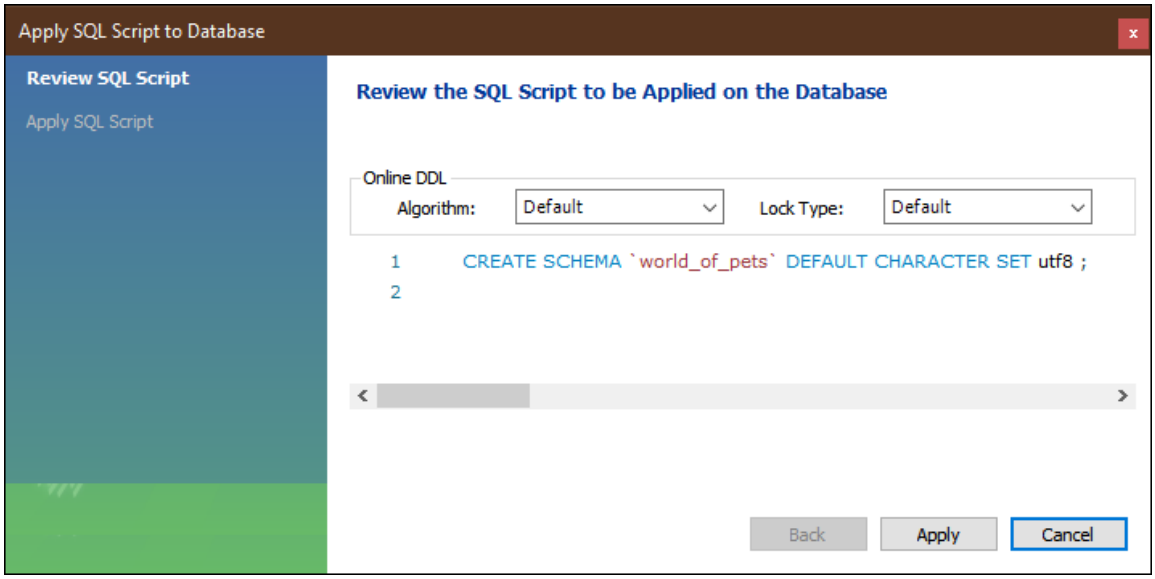
5.2 After clicking on the connection, you should see the **SCHEMAS** tab on the left. Right-Click here, or click on the "Create a new schema…" icon to create a new database.



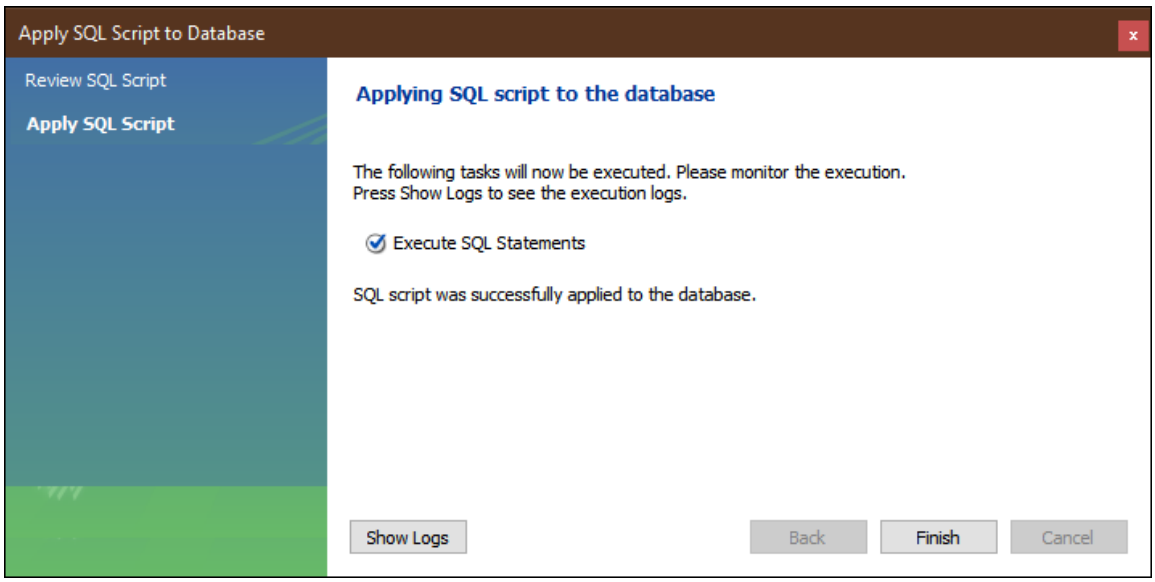

5.3 In the schema configuration panel, give the database a name, say "world_of_pets" since this will be used with the World of Pet website we've been building throughout this course. Choose utf8 for the character set, and leave collation set to default.



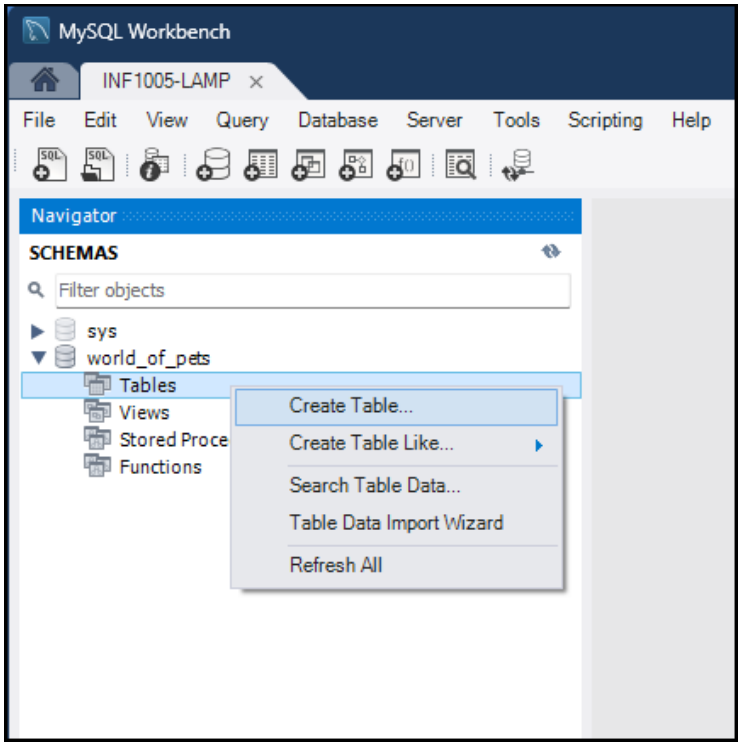
5.4 Click “Apply” to bring up the script review dialog.




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




5.6 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.

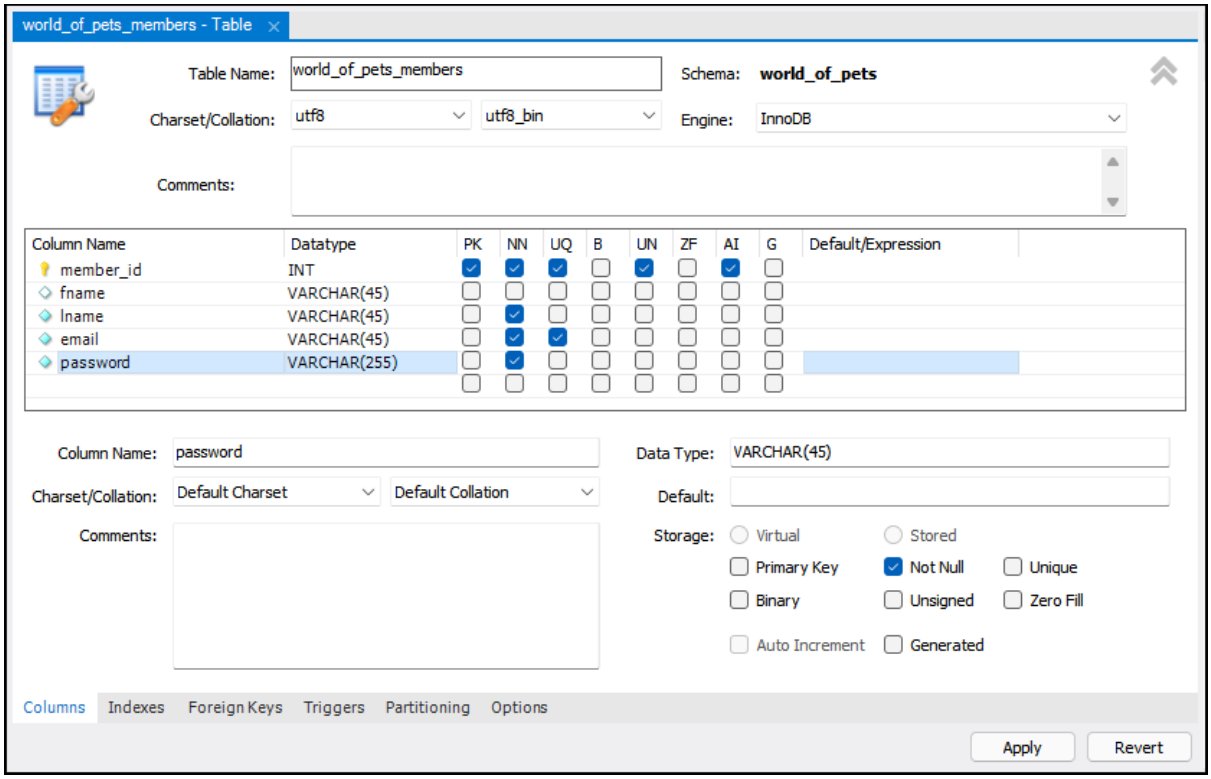



5.7 Name the table “world_of_pets_members” or similar, as this will be where we store our members’ registration information. Complete the following steps to configure the table.

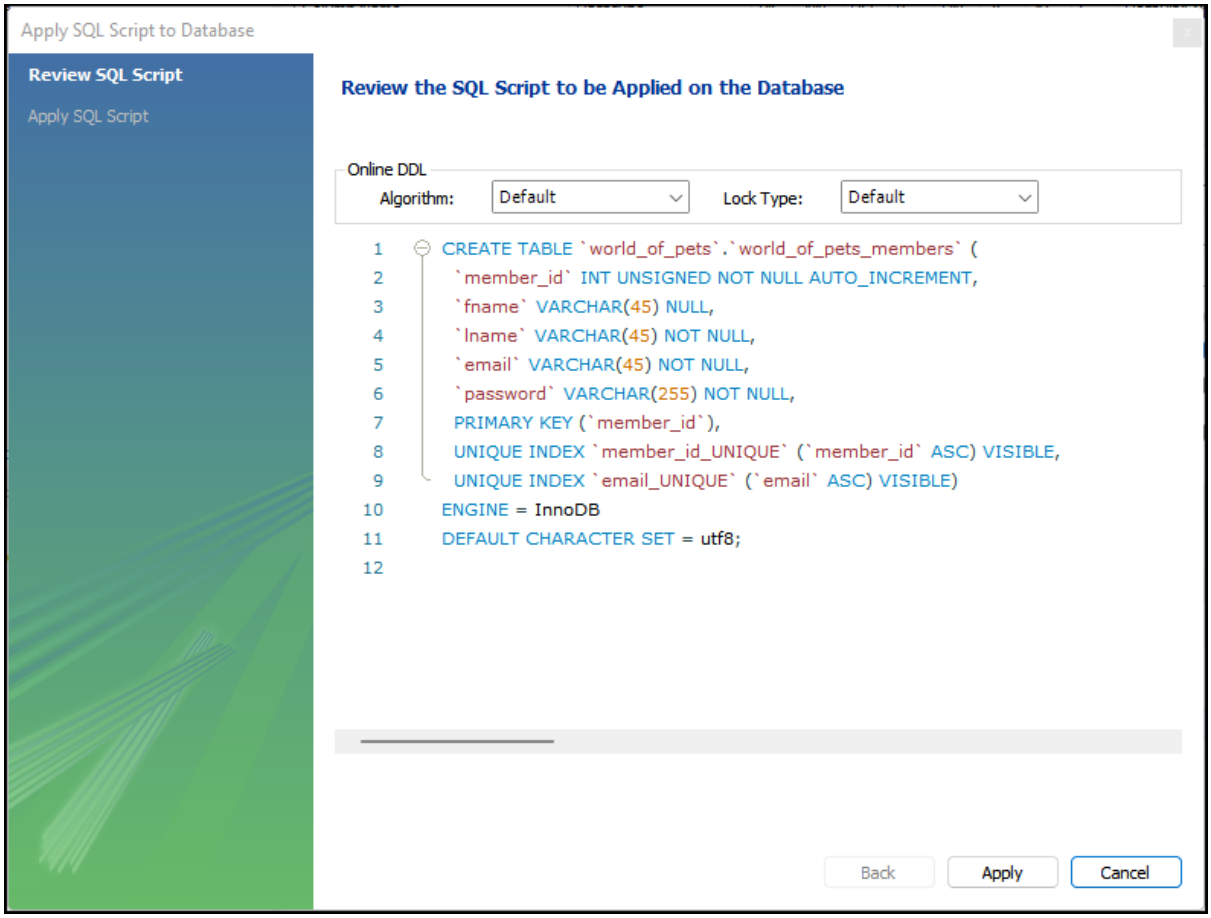
a. Set the charset to utf8 again.

b. Click under “Column Name” and add a **primary key** field called “member_id” of type **INT** with the following attributes: **primary**, **not null**, **unique**, **unsigned** and **auto increment**. Primary keys are always unique within a table and are used to differentiate between records with the same values (e.g., two people with the same name).

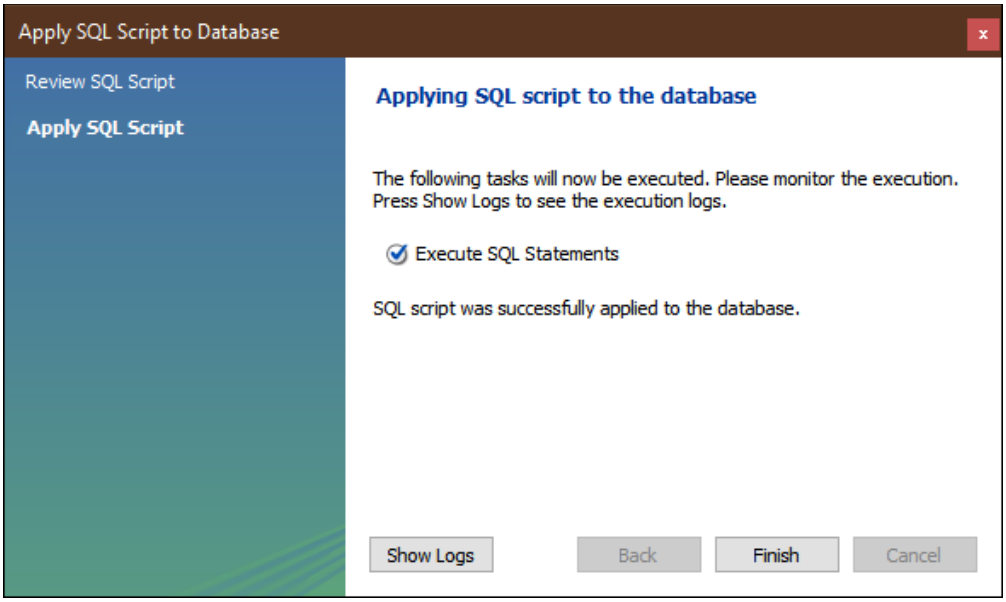
c. Add remaining columns for first name (**fname**), last name (**lname**), email address (**email**), and **password** as indicated below. Datatype can be the default VARCHAR(45) for all except for **password** – for that you should use VARCHAR(255) to allow for large hash strings. All fields should be “Not Null” except for **fname**, since it’s possible for a person to have only one name. In our example, we’ll also make the **email** field **unique**, since we’ll be using the email address for the login and must ensure that no two members have the same email address.




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




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


Apply SQL Script to Database
Review SQL Script
Apply SQL Script
Applying SQL script to the database
The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.
Execute SQL Statements
SQL script was successfully applied to the database.
Show Logs
Back
Finish
Cancel

## 6. EXERCISE 6: DATABASE OPERATIONS WITH PHP

6.1 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 and open it in Visual Studio Code.

6.2 Firstly, we will store our database credentials (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 "password" with the password you used for the database user (e.g., `inf1005-sqldev`) you added in the earlier exercise. 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 VM 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 = "inf1005-sqldev"
password = "password"
dbname = "world_of_pets"
```

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

6.3 Now we'll modify `process_register.php` so that we save the user's information in the database once it has been validated and sanitized. 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 for more details on MySQLi.

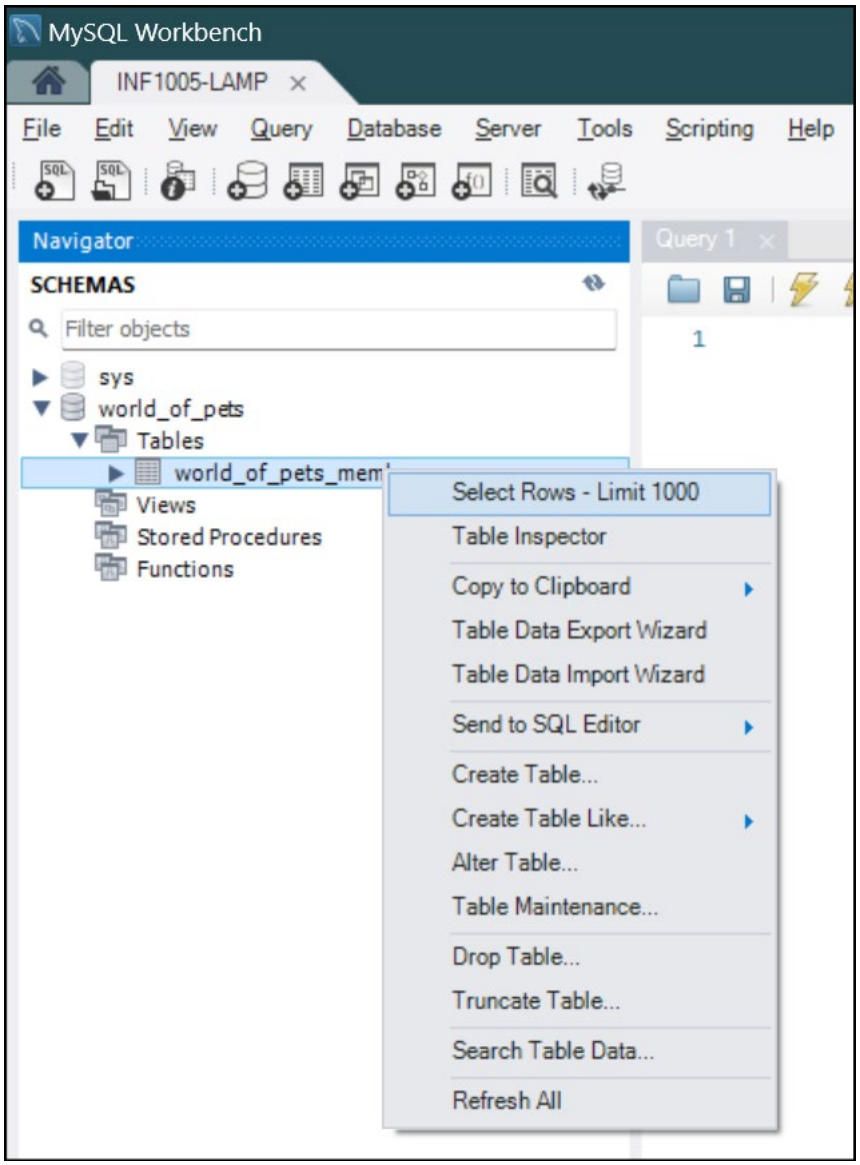
**Note:** the function `saveMemberToDB()` should only be called *after* all form data has been successfully validated/sanitized. However, as an additional protection against SQL Injection attacks, we will use prepared statements.

```php
/*
 * Helper function to write the member data to the database.
 */
function saveMemberToDB()
{
    global $fname, $lname, $email, $pwd_hashed, $errorMsg, $success;

    // Create database connection.
    $config = parse_ini_file('/var/www/private/db-config.ini');
    if (!$config)
    {
        $errorMsg = "Failed to read database config file.";
        $success = false;
    }
    else
    {
        $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
        {
            // Prepare the statement:
            $stmt = $conn->prepare("INSERT INTO world_of_pets_members
                (fname, lname, email, password) VALUES (?, ?, ?, ?)");

            // Bind & execute the query statement:
            $stmt->bind_param("ssss", $fname, $lname, $email, $pwd_hashed);
            if (!$stmt->execute())
            {
                $errorMsg = "Execute failed: (" . $stmt->errno . ") " .
                    $stmt->error;
                $success = false;
            }
            $stmt->close();
        }

        $conn->close();
    }
}
```

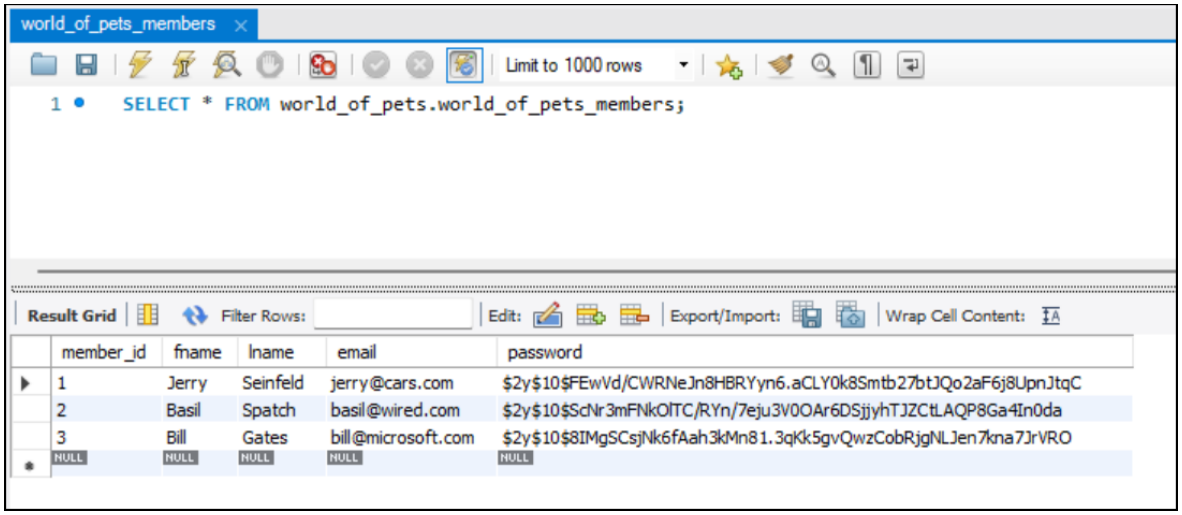
6.4 Publish your changed files to the LAMP server using SFTP (follow the procedure from the previous lab).

6.5 Test your website by going to the registration page and registering several members.

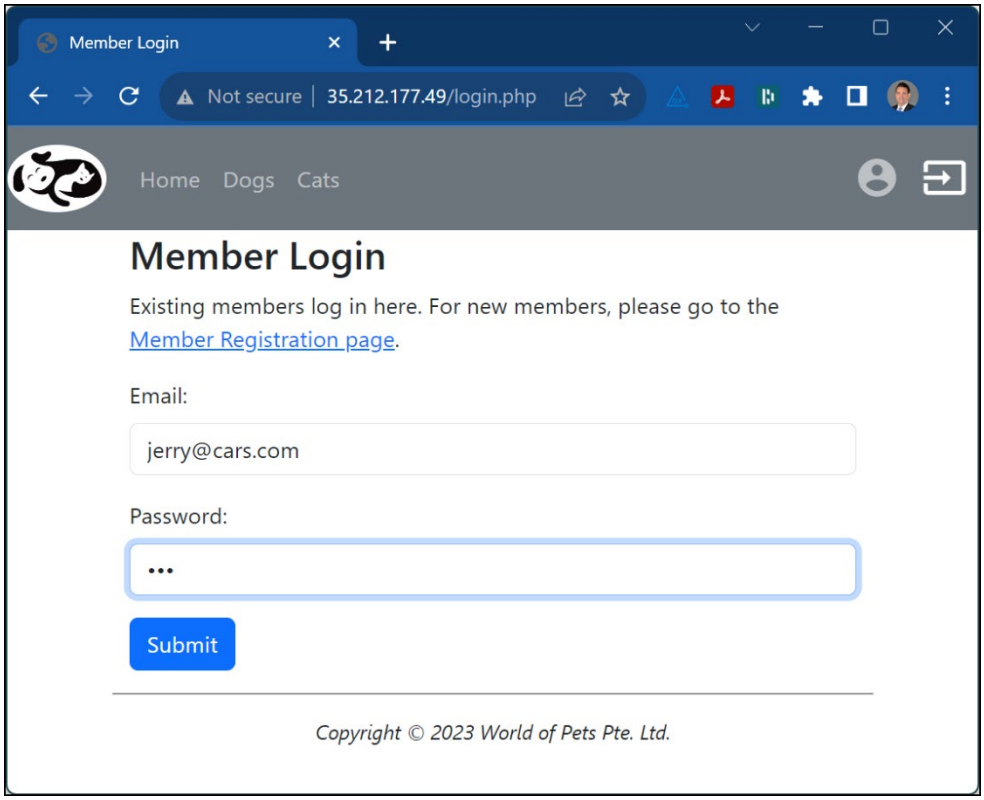
6.6 Check that the records were successfully saved in the database. To do this, go to MySQL Workbench and Right-Click on the table, then choose 'Select Rows' from the pop-up menu (you can also click the grid icon after the table name):



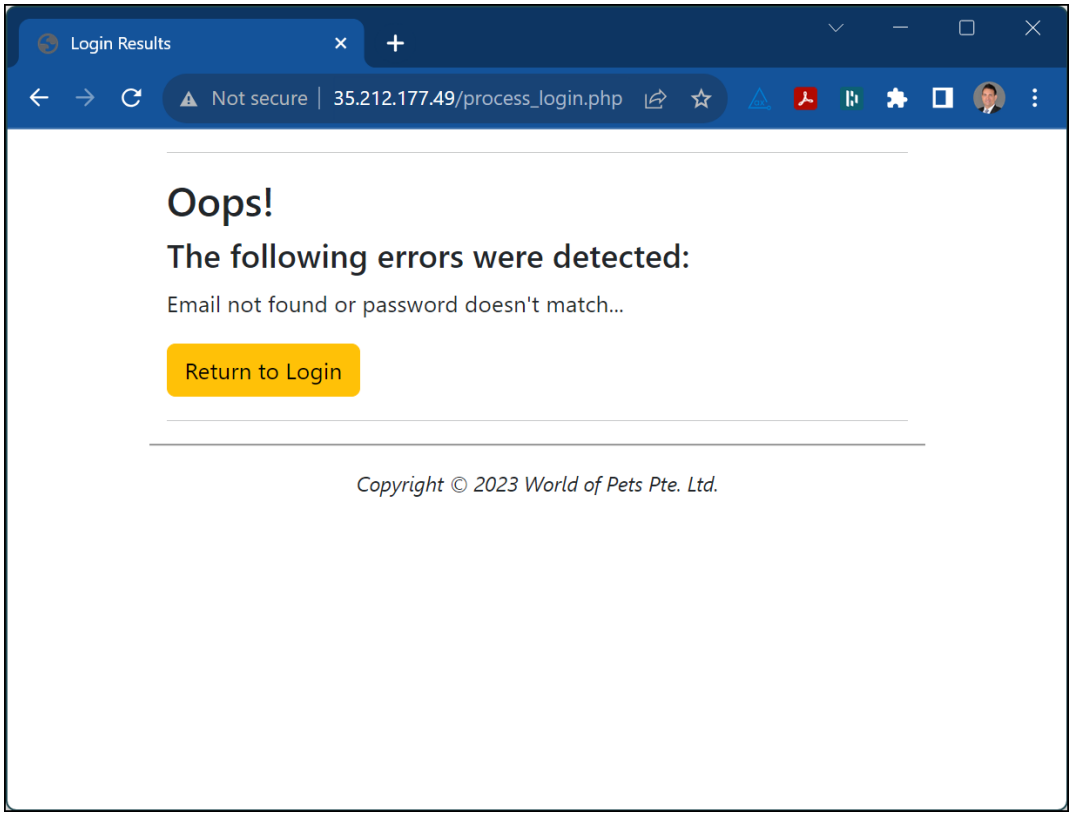

You should see the newly added records. Note that the passwords were hashed using the PHP function `password_hash()` before saving to the database:



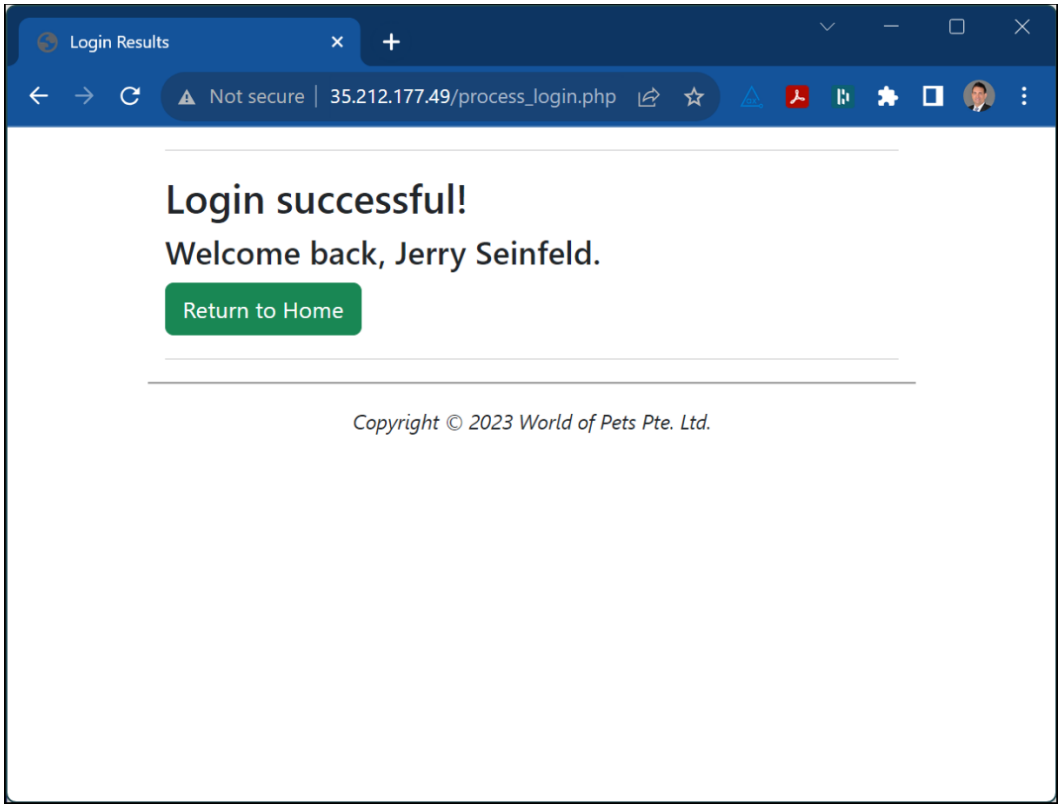

6.7 Now we need to add a login page (login.php). This should contain a form for the user to enter their email address and password. Set the action attribute to point to `process_login.php`, which we'll create in the next step (e.g., `action="process_login.php"`). Remember to update your navigation menu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



6.8 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 the input). If they don't exist or if the password doesn't match, notify the user accordingly. Example:




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




The code fragment on the next page 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.

**Don't forget to SFTP your files to the LAMP server after any changes!**

```
function authenticateUser()
{
    global $fname, $lname, $email, $pwd_hashed, $errorMsg, $success;

    // Create database connection.
    $config = parse_ini_file('/var/www/private/db-config.ini');
    if (!$config)
    {
        $errorMsg = "Failed to read database config file.";
        $success = false;
    }
    else
    {
        $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
        {
            // Prepare the statement:
            $stmt = $conn->prepare("SELECT * FROM world_of_pets_members WHERE email=?");

            // Bind & execute the query statement:
            $stmt->bind_param("s", $email);
            $stmt->execute();
            $result = $stmt->get_result();
            if ($result->num_rows > 0)
            {
                // Note that email field is unique, so should only have one row.
                $row = $result->fetch_assoc();
                $fname = $row["fname"];
                $lname = $row["lname"];
                $pwd_hashed = $row["password"];

                // Check if the password matches:
                if (!password_verify($_POST["pwd"], $pwd_hashed))
                {
                    // Don’t tell hackers which one was wrong, keep them guessing…
                    $errorMsg = "Email not found or password doesn't match...";
                    $success = false;
                }
            }
            else
            {
                $errorMsg = "Email not found or password doesn't match...";
                $success = false;
            }
            $stmt->close();
        }
        $conn->close();
    }
}
```

## 7. NEXT STEPS

This Lab 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. Use SSL. Ideally, we would obtain a certificate for our web server and use SSL to protect the data between the browser and the server. This is not a requirement for the Project or the lab exercises, but is something you should definitely do in a real-world, production website.

## 8. SUBMISSION OF LAB ASSIGNMENT

8.1 In order to receive credit for this Lab assignment, you must submit your completed work to xSiTe LMS before the end of the Lab session. To submit your work:

a. Save all files and close Visual Studio Code.

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 project. **Note: only .zip format is acceptable, do not use .rar, .7z, or any other format.**

c. In the INF1005 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.