

DEPARTMENT OF COMPUTER STUDIES

(Applications Development and Emerging Technologies)

PRE-SUMMATIVE ASSESSMENT

6

MYSQL DATABASE AND RESET PASSWORD FEATURE

Student Name / Group Name:	Jan Andrei Reyes
Section:	TN22
Professor:	Ms. Kim Giselle Bautista

I. PROGRAM OUTCOME/S (PO) ADDRESSED BY THE LABORATORY EXERCISE

- Design, implement and evaluate computer-based systems or applications to meet desired needs and requirements.

II. COURSE LEARNING OUTCOME/S (CLO) ADDRESSED BY THE LABORATORY EXERCISE

- Understand and apply best practices and standards in the development of website.

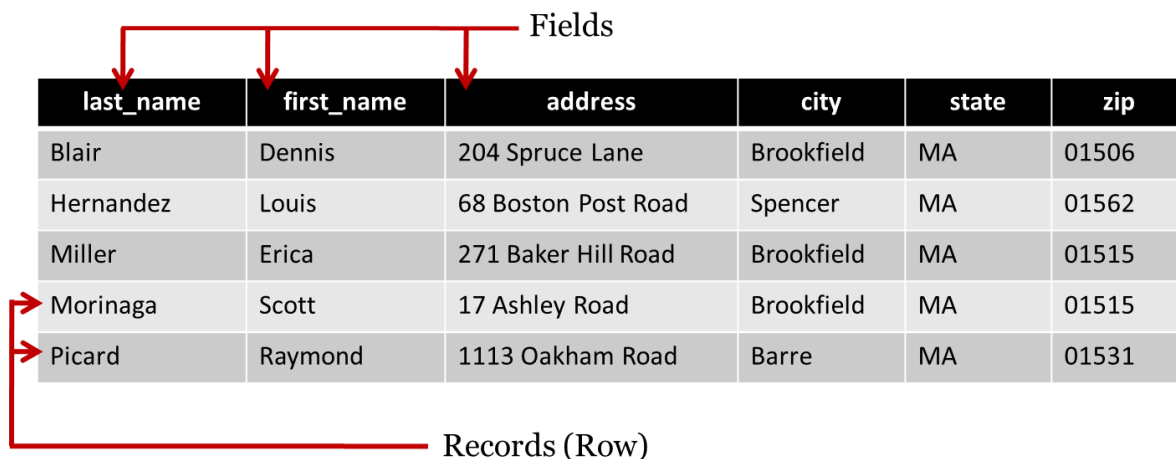
III. INTENDED LEARNING OUTCOME/S (ILO) OF THE LABORATORY EXERCISE

At the end of this exercise, students must be able to:

- To provide a good background of Relational Database using MySQL.
- To know the importance of Database in Web Application using MySQL.
- To Identify the importance of Database Structure in constructing tables.
- To be familiar with the syntax in managing users and database.
- To define a good structure of tables in a given database for data storage.
- To be familiar in the common syntax of creating database and tables and the correct data type to be used for each field.

IV. BACKGROUND INFORMATION

Example: Employee Directory DB



The diagram shows a table with 6 columns and 6 rows. Red arrows point from the label 'Fields' to the column headers: last_name, first_name, address, city, state, and zip. Another red arrow points from the label 'Records (Row)' to the first row of data: Blair, Dennis, 204 Spruce Lane, Brookfield, MA, 01506.

last_name	first_name	address	city	state	zip
Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506
Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562
Miller	Erica	271 Baker Hill Road	Brookfield	MA	01515
Morinaga	Scott	17 Ashley Road	Brookfield	MA	01515
Picard	Raymond	1113 Oakham Road	Barre	MA	01531

Also called as **flat-file database** that stores information in a single table.

PHP MySQL Database

With PHP, you can connect to and manipulate databases.

MySQL is the most popular database system used with PHP.

What is MySQL?

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Monty Widenius's daughter: My

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

Databases are useful for storing information categorically. A company may have a database with the following tables:

- Employees
- Products
- Customers
- Orders

PHP + MySQL Database System

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Database Queries

A query is a question or a request.

We can query a database for specific information and have a recordset returned.

Look at the following query (using standard SQL):

```
SELECT LastName FROM Employees
```

The query above selects all the data in the "LastName" column from the "Employees" table.

PHP Connect to MySQL

PHP 5 and later can work with a MySQL database using:

- **MySQLi extension** (the "i" stands for improved)
- **PDO (PHP Data Objects)**

Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.

Should I Use MySQLi or PDO?

If you need a short answer, it would be "Whatever you like".

Both MySQLi and PDO have their advantages:

PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.

So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.

Both are object-oriented, but MySQLi also offers a procedural API.

Both support Prepared Statements. Prepared Statements protect from SQL injection, and are very important for web application security.

MySQL Examples in Both MySQLi and PDO Syntax

In this, and in the following chapters we demonstrate three ways of working with PHP and MySQL:

- MySQLi (object-oriented)
- MySQLi (procedural)
- PDO

MySQLi Installation

For Linux and Windows: The MySQLi extension is automatically installed in most cases, when php5 mysql package is installed.

For installation details, go to: <http://php.net/manual/en/mysqli.installation.php>

PDO Installation

For installation details, go to: <http://php.net/manual/en/pdo.installation.php>

Open a Connection to MySQL

Before we can access data in the MySQL database, we need to be able to connect to the server:

Example (MySQLi Object-Oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
```

```
// Create connection
```

```

$conn = new mysqli($servername, $username, $password);

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

```

Note on the object-oriented example above:

\$connect_error was broken until PHP 5.2.9 and 5.3.0. If you need to ensure compatibility with PHP versions prior to 5.2.9 and 5.3.0, use the following code instead:

```

// Check connection
if (mysqli_connect_error()) {
    die("Database connection failed: " . mysqli_connect_error());
}

```

Example (MySQLi Procedural)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>

```

Example (PDO)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";

```

```
try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB", $username,
$password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
} catch(PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
}
?>
```

Note: In the PDO example above we have also **specified a database (myDB)**. PDO require a valid database to connect to. If no database is specified, an exception is thrown.

Tip: A great benefit of PDO is that it has an exception class to handle any problems that may occur in our database queries. If an exception is thrown within the try{ } block, the script stops executing and flows directly to the first catch(){ } block.

Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

MySQLi Object-Oriented:

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

MySQLi Procedural:

```
mysqli_close($conn);
```

PDO:

```
$conn = null;
```

PHP Create a MySQL Database

A database consists of one or more tables.

You will need special CREATE privileges to create or to delete a MySQL database.

Create a MySQL Database Using MySQLi and PDO

The CREATE DATABASE statement is used to create a database in MySQL.

The following examples create a database named "myDB":

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}

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

Note: When you create a new database, you must only specify the first three arguments to the mysqli object (servername, username and password).

Tip: If you have to use a specific port, add an empty string for the database-

name argument, like this: new mysqli("localhost", "username", "password", "", port)

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Create database
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Note: The following PDO example create a database named "myDBPDO":

Example (PDO)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
    $conn = new PDO("mysql:host=$servername", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sql = "CREATE DATABASE myDBPDO";
```

```

    // use exec() because no results are returned
    $conn->exec($sql);
    echo "Database created successfully<br>";
} catch(PDOException $e) {
    echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>

```

Tip: A great benefit of PDO is that it has exception class to handle any problems that may occur in our database queries. If an exception is thrown within the try{ } block, the script stops executing and flows directly to the first catch(){ } block. In the catch block above we echo the SQL statement and the generated error message.

PHP MySQL Create Table

A database table has its own unique name and consists of columns and rows.

Create a MySQL Table Using MySQLi and PDO

The CREATE TABLE statement is used to create a table in MySQL.

We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":

```

CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)

```

Notes on the table above:

The data type specifies what type of data the column can hold. For a complete reference of all the available data types, go to our [Data Types reference](#).

After the data type, you can specify other optional attributes for each column:

- NOT NULL - Each row must contain a value for that column, null values are not allowed
- DEFAULT value - Set a default value that is added when no other value is passed
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

Each table should have a primary key column (in this case: the "id" column). Its value must be unique for each record in the table.

The following examples shows how to create the table in PHP:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
```

```

    );

    if ($conn->query($sql) === TRUE) {
        echo "Table MyGuests created successfully";
    } else {
        echo "Error creating table: " . $conn->error;
    }

    $conn->close();
    ?>

```

Example (MySQLi Procedural)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)";

if (mysqli_query($conn, $sql)) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . mysqli_error($conn);
}

mysqli_close($conn);
?>

```

Example (PDO)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // sql to create table
    $sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)";

    // use exec() because no results are returned
    $conn->exec($sql);
    echo "Table MyGuests created successfully";
} catch(PDOException $e) {
    echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

Insert Data Into MySQL Using MySQLi and PDO

After a database and a table have been created, we can start adding data in them.

Here are some syntax rules to follow:

- The SQL query must be quoted in PHP

- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

To learn more about SQL, please visit our [SQL tutorial](#).

In the previous chapter we created an empty table named "MyGuests" with five columns: "id", "firstname", "lastname", "email" and "reg_date". Now, let us fill the table with data.

Note: If a column is AUTO_INCREMENT (like the "id" column) or TIMESTAMP with default update of current_timestamp (like the "reg_date" column), it is no need to be specified in the SQL query; MySQL will automatically add the value.

The following examples add a new record to the "MyGuests" table:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
```

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

Example (MySQLi Procedural)

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = mysqli_connect($servername, $username, $password, $dbname);  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli_connect_error());  
}  
  
$sql = "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('John', 'Doe', 'john@example.com')";  
  
if (mysqli_query($conn, $sql)) {  
    echo "New record created successfully";  
} else {  
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);  
}  
  
mysqli_close($conn);  
?>
```

Example (PDO)

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDBPDO";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,  
$password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
}
```

```

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
// use exec() because no results are returned
$conn->exec($sql);
echo "New record created successfully";
} catch(PDOException $e) {
    echo $sql . "<br>" . $e->getMessage();
}

```

```

$conn = null;

```

```

?>

```

Get ID of The Last Inserted Record

If we perform an INSERT or UPDATE on a table with an AUTO_INCREMENT field, we can get the ID of the last inserted/updated record immediately.

In the table "MyGuests", the "id" column is an AUTO_INCREMENT field:

```

CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)

```

The following examples are equal to the examples from the previous page ([PHP Insert Data Into MySQL](#)), except that we have added one single line of code to retrieve the ID of the last inserted record. We also echo the last inserted ID:

Example (MySQLi Object-oriented)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {

```



```

    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    $last_id = $conn->insert_id;
    echo "New record created successfully. Last inserted ID is: " .
    $last_id;
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>

```

Example (MySQLi Procedural)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
    $last_id = mysqli_insert_id($conn);
    echo "New record created successfully. Last inserted ID is: " .
    $last_id;
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

```

```
mysqli_close($conn);  
?>
```

Example (PDO)

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDBPDO";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,  
$password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
    $sql = "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('John', 'Doe', 'john@example.com')";  
    // use exec() because no results are returned  
    $conn->exec($sql);  
    $last_id = $conn->lastInsertId();  
    echo "New record created successfully. Last inserted ID is: " .  
$last_id;  
} catch(PDOException $e) {  
    echo $sql . "<br>" . $e->getMessage();  
}  
  
$conn = null;  
?>
```

Insert Multiple Records Into MySQL Using MySQLi and PDO

Multiple SQL statements must be executed with the `mysqli_multi_query()` function.

The following examples add three new records to the "MyGuests" table:

Example (MySQLi Object-oriented)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com')";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";

if ($conn->multi_query($sql) === TRUE) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>

```

Note that each SQL statement must be separated by a semicolon.

Example (MySQLi Procedural)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

```

```

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com')";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";

if (mysqli_multi_query($conn, $sql)) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>

```

The PDO way is a little bit different:

Example (PDO)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // begin the transaction
    $conn->beginTransaction();
    // our SQL statements
    $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')");
    $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com')");
    $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')");

    // commit the transaction

```

```

$conn->commit();
echo "New records created successfully";
} catch(PDOException $e) {
    // roll back the transaction if something failed
    $conn->rollback();
    echo "Error: " . $e->getMessage();
}

$conn = null;
?>

```

Prepared Statements and Bound Parameters

A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency.

Prepared statements basically work like this:

1. Prepare: An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?"). Example: INSERT INTO MyGuests VALUES(?, ?, ?)
2. The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it
3. Execute: At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values

Compared to executing SQL statements directly, prepared statements have three main advantages:

- Prepared statements reduce parsing time as the preparation on the query is done only once (although the statement is executed multiple times)
- Bound parameters minimize bandwidth to the server as you need send only the parameters each time, and not the whole query
- Prepared statements are very useful against SQL injections, because parameter values, which are transmitted later using a different protocol, need not be correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur.

Example (MySQLi with Prepared Statements)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

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

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

// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email)
VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
$stmt->execute();

echo "New records created successfully";

$stmt->close();
```

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

Code lines to explain from the example above:

```
"INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)"
```

In our SQL, we insert a question mark (?) where we want to substitute in an integer, string, double or blob value.

Then, have a look at the `bind_param()` function:

```
$stmt->bind_param("sss", $firstname, $lastname, $email);
```

This function binds the parameters to the SQL query and tells the database what the parameters are. The "sss" argument lists the types of data that the parameters are. The s character tells mysql that the parameter is a string.

The argument may be one of four types:

- i - integer
- d - double
- s - string
- b - BLOB

We must have one of these for each parameter.

By telling mysql what type of data to expect, we minimize the risk of SQL injections.

Note: If we want to insert any data from external sources (like user input), it is very important that the data is sanitized and validated.

Example (PDO with Prepared Statements)

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDBPDO";
```

```

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // prepare sql and bind parameters
    $stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname,
email)
VALUES (:firstname, :lastname, :email)");
    $stmt->bindParam(':firstname', $firstname);
    $stmt->bindParam(':lastname', $lastname);
    $stmt->bindParam(':email', $email);

    // insert a row
    $firstname = "John";
    $lastname = "Doe";
    $email = "john@example.com";
    $stmt->execute();

    // insert another row
    $firstname = "Mary";
    $lastname = "Moe";
    $email = "mary@example.com";
    $stmt->execute();

    // insert another row
    $firstname = "Julie";
    $lastname = "Dooley";
    $email = "julie@example.com";
    $stmt->execute();

    echo "New records created successfully";
} catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
$conn = null;
?>

```

Select Data From a MySQL Database

The SELECT statement is used to select data from one or more tables:

```
SELECT column_name(s) FROM table_name
```

or we can use the * character to select ALL columns from a table:

```
SELECT * FROM table_name
```

Select Data With MySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table and displays it on the page:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
        $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

Code lines to explain from the example above:

First, we set up an SQL query that selects the id, firstname and lastname columns from the MyGuests table. The next line of code runs the query and puts the resulting data into a variable called \$result.

Then, the `function num_rows()` checks if there are more than zero rows returned.

If there are more than zero rows returned, the function `fetch_assoc()` puts all the results into an associative array that we can loop through. The `while()` loop loops through the result set and outputs the data from the id, firstname and lastname columns.

The following example shows the same as the example above, in the MySQLi procedural way:

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
        $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
```

```
mysqli_close($conn);  
?>
```

You can also put the result in an HTML table:

Example (MySQLi Object-oriented)

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect_error) {  
    die("Connection failed: " . $conn->connect_error);  
}  
  
$sql = "SELECT id, firstname, lastname FROM MyGuests";  
$result = $conn->query($sql);  
  
if ($result->num_rows > 0) {  
    echo "<table><tr><th>ID</th><th>Name</th></tr>";  
    // output data of each row  
    while($row = $result->fetch_assoc()) {  
        echo "<tr><td>".$row["id"]."</td><td>".$row["firstname"]."  
".$row["lastname"]."</td></tr>";  
    }  
    echo "</table>";  
} else {  
    echo "0 results";  
}  
$conn->close();  
?>
```

Select Data With PDO (+ Prepared Statements)

The following example uses prepared statements.

It selects the id, firstname and lastname columns from the MyGuests table and displays it in an HTML table:

Example (PDO)

```
<?php
echo "<table style='border: solid 1px black;'>";
echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";

class TableRows extends RecursiveIteratorIterator {
    function __construct($it) {
        parent::__construct($it, self::LEAVES_ONLY);
    }

    function current() {
        return "<td style='width:150px;border:1px solid black;'>" .
parent::current(). "</td>";
    }

    function beginChildren() {
        echo "<tr>";
    }

    function endChildren() {
        echo "</tr>" . "\n";
    }
}

$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests");
    $stmt->execute();

    // set the resulting array to associative
    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
    foreach(new TableRows(new RecursiveArrayIterator($stmt-
>fetchAll())) as $k=>$v) {
        echo $v;
```

```

    }
} catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
$conn = null;
echo "</table>";
?>

```

Select and Filter Data From a MySQL Database

The WHERE clause is used to filter records.

The WHERE clause is used to extract only those records that fulfill a specified condition.

```
SELECT column_name(s) FROM table_name WHERE column_name operator value
```

Select and Filter Data With MySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table where the lastname is "Doe", and displays it on the page:

Example (MySQLi Object-oriented)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE

```

```

lastname='Doe';
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
$row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>

```

Code lines to explain from the example above:

First, we set up the SQL query that selects the id, firstname and lastname columns from the MyGuests table where the lastname is "Doe". The next line of code runs the query and puts the resulting data into a variable called \$result.

Then, the `function num_rows()` checks if there are more than zero rows returned.

If there are more than zero rows returned, the function `fetch_assoc()` puts all the results into an associative array that we can loop through. The `while()` loop loops through the result set and outputs the data from the id, firstname and lastname columns.

The following example shows the same as the example above, in the MySQLi procedural way:

Example (MySQLi Procedural)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {

```

```

    die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE
lastname='Doe'";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
$row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}

mysqli_close($conn);
?>

```

You can also put the result in an HTML table:

Example (MySQLi Object-oriented)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE
lastname='Doe'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "<table><tr><th>ID</th><th>Name</th></tr>";
    // output data of each row
    while($row = $result->fetch_assoc()) {

```

```

        echo "<tr><td>".$row["id"]."</td><td>".$row["firstname"]."
".$row["lastname"]."</td></tr>";
    }
    echo "</table>";
} else {
    echo "0 results";
}
$conn->close();
?>

```

Select Data With PDO (+ Prepared Statements)

The following example uses prepared statements.

It selects the id, firstname and lastname columns from the MyGuests table where the lastname is "Doe", and displays it in an HTML table:

Example (PDO)

```

<?php
echo "<table style='border: solid 1px black;'>";
echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";

class TableRows extends RecursiveIteratorIterator {
    function __construct($it) {
        parent::__construct($it, self::LEAVES_ONLY);
    }

    function current() {
        return "<td style='width:150px;border:1px solid black;'>" .
parent::current(). "</td>";
    }

    function beginChildren() {
        echo "<tr>";
    }

    function endChildren() {
        echo "</tr>" . "\n";
    }
}

```



```

$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests
WHERE lastname='Doe'");
    $stmt->execute();

    // set the resulting array to associative
    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
    foreach(new TableRows(new RecursiveArrayIterator($stmt-
>fetchAll())) as $k=>$v) {
        echo $v;
    }
}
catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
$conn = null;
echo "</table>";
?>

```

Select and Order Data From a MySQL Database

The ORDER BY clause is used to sort the result-set in ascending or descending order.

The ORDER BY clause sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

```
SELECT column_name(s) FROM table_name ORDER BY column_name(s) ASC|DESC
```

Select and Order Data With MySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table. The records will be ordered by the lastname column:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
        $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

Code lines to explain from the example above:

First, we set up the SQL query that selects the id, firstname and lastname columns from the MyGuests table. The records will be ordered by the lastname column. The next line of code runs the query and puts the resulting data into a variable called \$result.

Then, the `function num_rows()` checks if there are more than zero rows returned.

If there are more than zero rows returned, the function `fetch_assoc()` puts all the results into an associative array that we can loop through. The `while()` loop

loops through the result set and outputs the data from the id, firstname and lastname columns.

The following example shows the same as the example above, in the MySQLi procedural way:

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
        $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}

mysqli_close($conn);
?>
```

You can also put the result in an HTML table:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
```

```

$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "<table><tr><th>ID</th><th>Name</th></tr>";
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<tr><td>".$row["id"]."</td><td>".$row["firstname"]."
".$row["lastname"]."</td></tr>";
    }
    echo "</table>";
} else {
    echo "0 results";
}
$conn->close();
?>

```

Select Data With PDO (+ Prepared Statements)

The following example uses prepared statements.

Here we select the id, firstname and lastname columns from the MyGuests table. The records will be ordered by the lastname column, and it will be displayed in an HTML table:

Example (PDO)

```

<?php
echo "<table style='border: solid 1px black;'>";
echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";

```

```

class TableRows extends RecursiveIteratorIterator {
    function __construct($it) {
        parent::__construct($it, self::LEAVES_ONLY);
    }

    function current() {
        return "<td style='width:150px;border:1px solid black;'" .
parent::current(). "</td>";
    }

    function beginChildren() {
        echo "<tr>";
    }

    function endChildren() {
        echo "</tr>" . "\n";
    }
}

$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests
ORDER BY lastname");
    $stmt->execute();

    // set the resulting array to associative
    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
    foreach(new TableRows(new RecursiveArrayIterator($stmt-
>fetchAll())) as $k=>$v) {
        echo $v;
    }
} catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
$conn = null;
echo "</table>";
?>

```

Delete Data From a MySQL Table Using MySQLi and PDO

The DELETE statement is used to delete records from a table:

```
DELETE FROM table_name
WHERE some_column = some_value
```

Notice the WHERE clause in the DELETE syntax: The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

Let's look at the "MyGuests" table:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30
3	Julie	Dooley	julie@example.com	2014-10-26 10:48:23

The following examples delete the record with id=3 in the "MyGuests" table:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";
```

```

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>

```

Example (MySQLi Procedural)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>

```

Example (PDO)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";

```

```

$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // sql to delete a record
    $sql = "DELETE FROM MyGuests WHERE id=3";

    // use exec() because no results are returned
    $conn->exec($sql);
    echo "Record deleted successfully";
} catch(PDOException $e) {
    echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>

```

After the record is deleted, the table will look like this:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

Update Data In a MySQL Table Using MySQLi and PDO

The UPDATE statement is used to update existing records in a table:

```

UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value

```

Notice the WHERE clause in the UPDATE syntax: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

Let's look at the "MyGuests" table:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

The following examples update the record with id=2 in the "MyGuests" table:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}

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

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
```

```

$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if (mysqli_query($conn, $sql)) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>

```

Example (PDO)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

    // Prepare statement
    $stmt = $conn->prepare($sql);

    // execute the query
    $stmt->execute();

    // echo a message to say the UPDATE succeeded
    echo $stmt->rowCount() . " records UPDATED successfully";
}

```

```

} catch(PDOException $e) {
    echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>

```

After the record is updated, the table will look like this:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Doe	mary@example.com	2014-10-23 10:22:30

V. GRADING SYSTEM / RUBRIC (please see separate sheet)

VI. LABORATORY ACTIVITY

1. Using MySQL create 10 records for the DOG Information use the following fields Name, Breed, Age, Address, Color, Height and Weight. Integrate HTML, CSS and PHP.

Dog Information Form

Name:

Breed:

Age:

Address:

Color:

Height (cm):

Weight (kg):

Add Dog

Dog added successfully!

After click the save button, information will save to the database

2. Create separate webpage to show all record of the dog.

All Dog Records

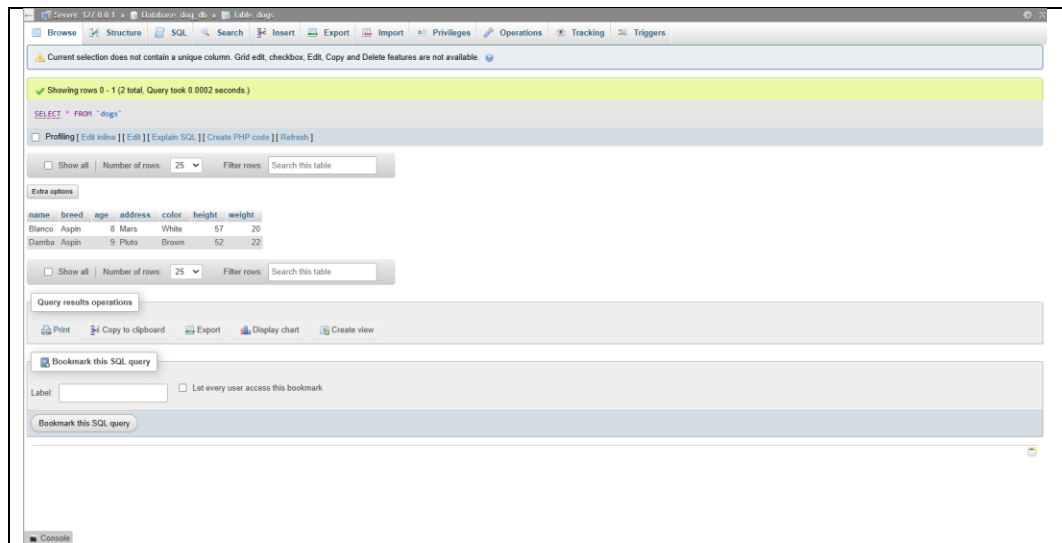
Dog 1

Name: Blanco
Breed: Aspin
Age: 1yrs old
Address: Mars
Color: White
Height: 57 ft
Weight: 20 kilos

Dog 2

Name: Damba
Breed: Aspin
Age: 5yrs old
Address: Pluto
Color: Brown
Height: 52 ft
Weight: 22 kilos

3. Screenshot the result of your database from the XAMPP. See the example below



Snip and paste your source codes here. Snip it directly from the IDE so that colors of the codes are preserved for readability. Include additional pages if necessary.

VII. QUESTION AND ANSWER

1. What is database?

A database is a place where data is stored and organized so it can be easily accessed, managed, and updated.

2. Give at least 5 where you can use a database

1. Online shopping websites (like Amazon)
2. Banking systems
3. School management systems
4. Hospital records
5. Social media platforms

3. What is MySQL?

MySQL is a popular software used to create and manage databases. It helps store and organize data in a structured way.

4. What is the importance of MySQL in PHP?

PHP is a programming language used to build websites. PHP works with MySQL to store user data, show content from databases, for forms, and to save and manage website contents.

5. Different command function used in MySQL?

SELECT, INSERT, UPDATE, DELETE, CREATE, etc.

6. Give 10 Examples of websites that use database.

Facebook, YouTube, Amazon, Shopee, Lazada, Netflix, Twitter, Instagram, LinkedIn, Wikipedia.

VIII. REFERENCES

1. <https://www.w3schools.com/css/>
2. <https://www.w3schools.com/html/>
3. https://www.w3schools.com/php/php_variables.asp
4. https://www.w3schools.com/php/php_mysql_intro.asp
5. https://www.w3schools.com/php/php_mysql_connect.asp
6. https://www.w3schools.com/php/php_mysql_create.asp
7. https://www.w3schools.com/php/php_mysql_create_table.asp
8. https://www.w3schools.com/php/php_mysql_insert.asp
9. https://www.w3schools.com/php/php_mysql_insert_multiple.asp
10. https://www.w3schools.com/php/php_mysql_select.asp
11. https://www.w3schools.com/php/php_mysql_select_where.asp
12. https://www.w3schools.com/php/php_mysql_select_orderby.asp
13. https://www.w3schools.com/php/php_mysql_delete.asp
14. https://www.w3schools.com/php/php_mysql_update.asp
15. <https://skillforge.com/how-to-create-a-database-using-phpmyadmin-xampp/>

Note: The following rubrics/metrics will be used to grade students' output.

Program (100 pts.)	(Excellent)	(Good)	(Fair)	(Poor)
Program execution (20pts)	Program executes correctly with no syntax or runtime errors (18-20pts)	Program executes with less than 3 errors (15-17pts)	Program executes with more than 3 errors (12-14pts)	Program does not execute (10-11pts)
Correct output (20pts)	Program displays correct output with no errors (18-20pts)	Output has minor errors (15-17pts)	Output has multiple errors (12-14pts)	Output is incorrect (10-11pts)
Design of output (10pts)	Program displays more than expected (10pts)	Program displays minimally expected output (8-9pts)	Program does not display the required output (6-7pts)	Output is poorly designed (5pts)
Design of logic (20pts)	Program is logically well designed (18-20pts)	Program has slight logic errors that do no significantly affect the results (15-17pts)	Program has significant logic errors (3-5pts)	Program is incorrect (10-11pts)
Standards (20pts)	Program code is stylistically well designed (18-20pts)	Few inappropriate design choices (i.e. poor variable names, improper indentation) (15-17pts)	Several inappropriate design choices (i.e. poor variable names, improper indentation) (12-14pts)	Program is poorly written (10-11pts)
Delivery (10pts)	The program was delivered on time. (10pts)	The program was delivered a day after the deadline. (8-9pts)	The program was delivered two days after the deadline. (6-7pts)	The program was delivered more than two days after the deadline. (5pts)