MySQLPHP

Database

- Almost all web application on the net access a database e.g. shopping sites, message boards, search engines
- Relational databases (uses tables) are the most common type used
- Small application may use just one table, larger applications may have hundreds of tables

Database

- Relational databases model data by storing rows and columns in tables. The power of the relational database lies in its ability to efficiently retrieve data from those tables and in particular where there are multiple tables and the relationships between those tables involved in the query.

Relational DBs

- Data is stored in tables. Rows and columns in tables can be related to rows and columns in other tables in the dB
- Each table usually has a primary key
- Structured Query Language (SQL) is used to query the database
- Common SQL Statements: CREATE, SELECT, INSERT, UPDATE,DELETE

customer ID	Name	Phone
John	1	879687
Liz	2	975645
Rory	3	321544

```
SELECT * from customers;

SELECT name, phone from customers
where customer_ID = 2;
```

Relational DBs

- INSERT

```
INSERT INTO CUSTOMERS (customer_ID, name, phone)
VALUES (5, "JOHN", "875895")
```

- UPDATE

```
UPDATE CUSTOMERS SET NAME = "Robert" WHERE
CUSTOMER_ID = 1
```

- DELETE

```
DELETE FROM CUSTOMERS WHERE CUSTOMER_ID = 2
```

Common Database Systems

- Three Major Database Management Systems in wide use
  - MySQL- Simple fast and scalable - commercial open source
  - Oracle- Large, commercial, enterprise-scale, very very tweakable
  - SqlServer- Very nice - from Microsoft (also Access)

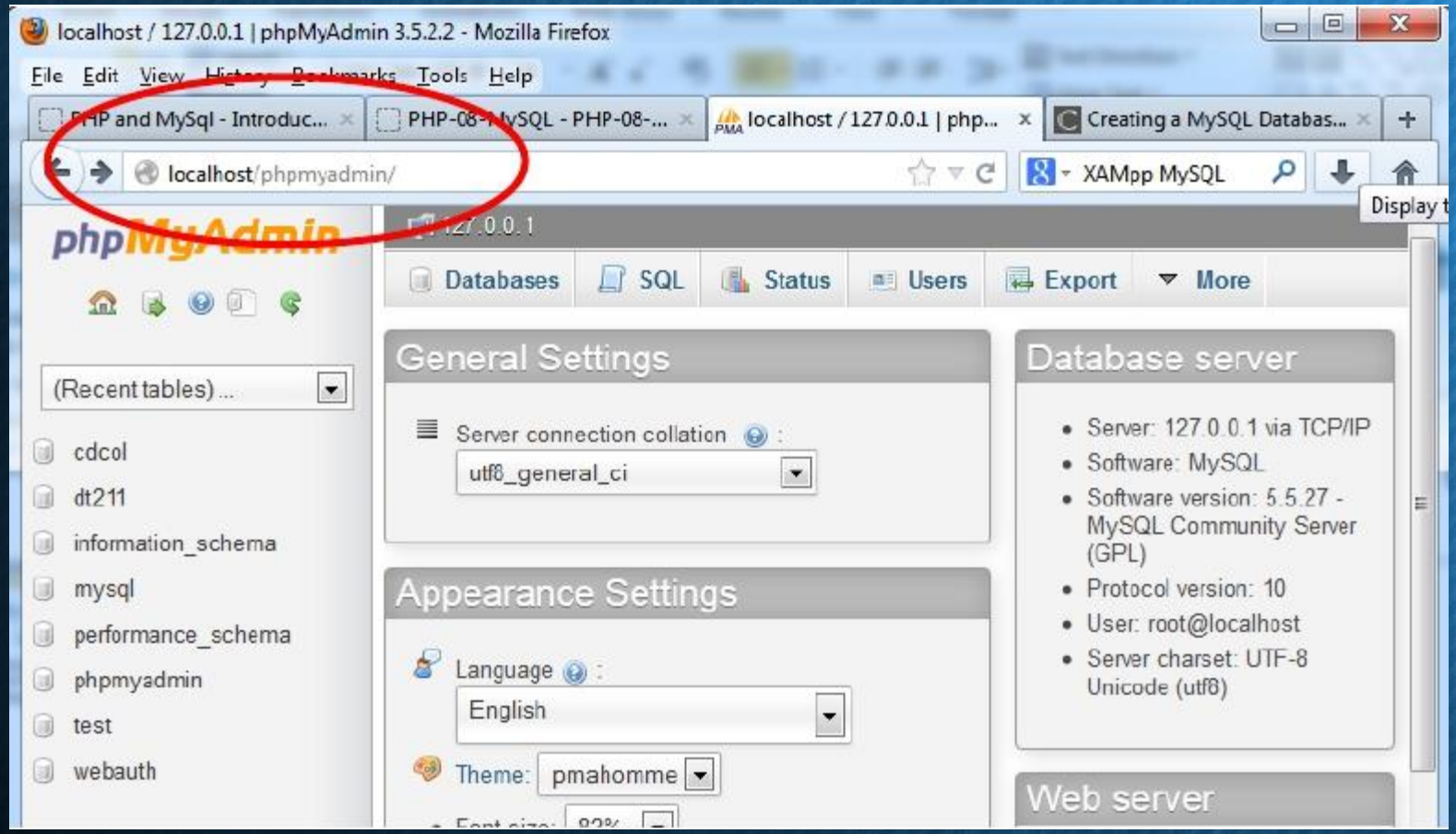
- Many other smaller projects, free and open source
  - HSQL,SQLite, MariaDB, Postgress, ...

MySQL Database

- 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 supports 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

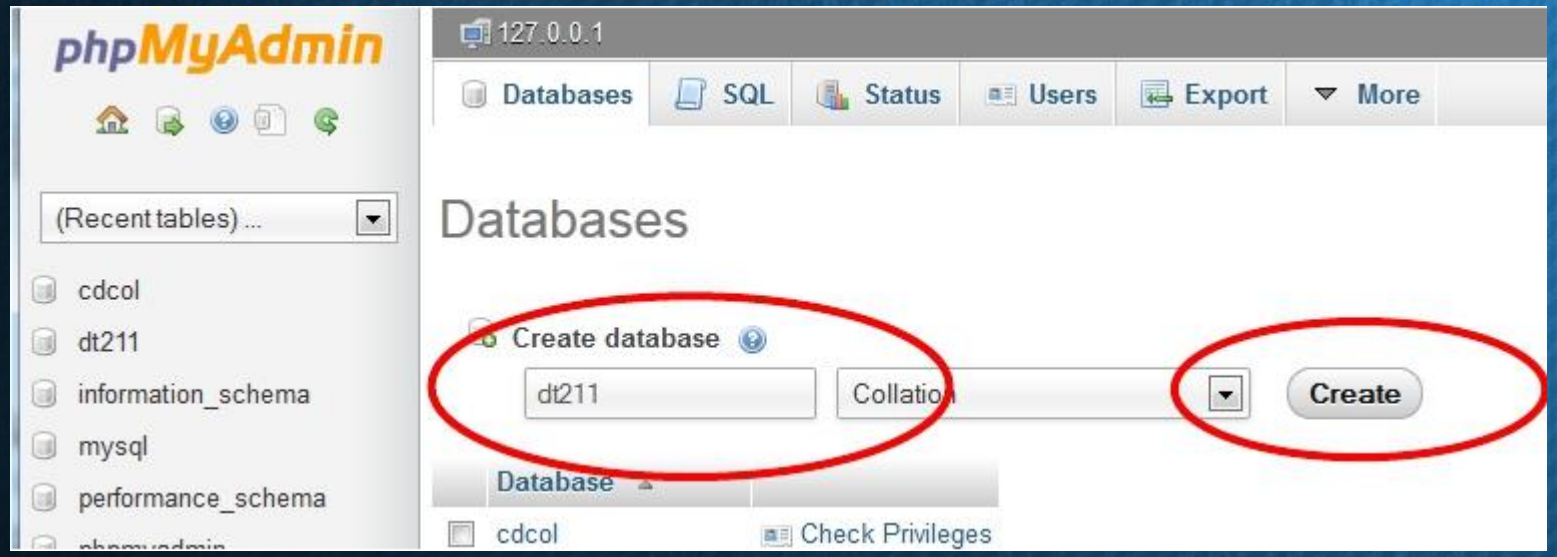
Creating a MySQL database using XAMPP

- Open your browser and enter http://localhost/phpmyadmin. This will bring you to the MySQL setup page:



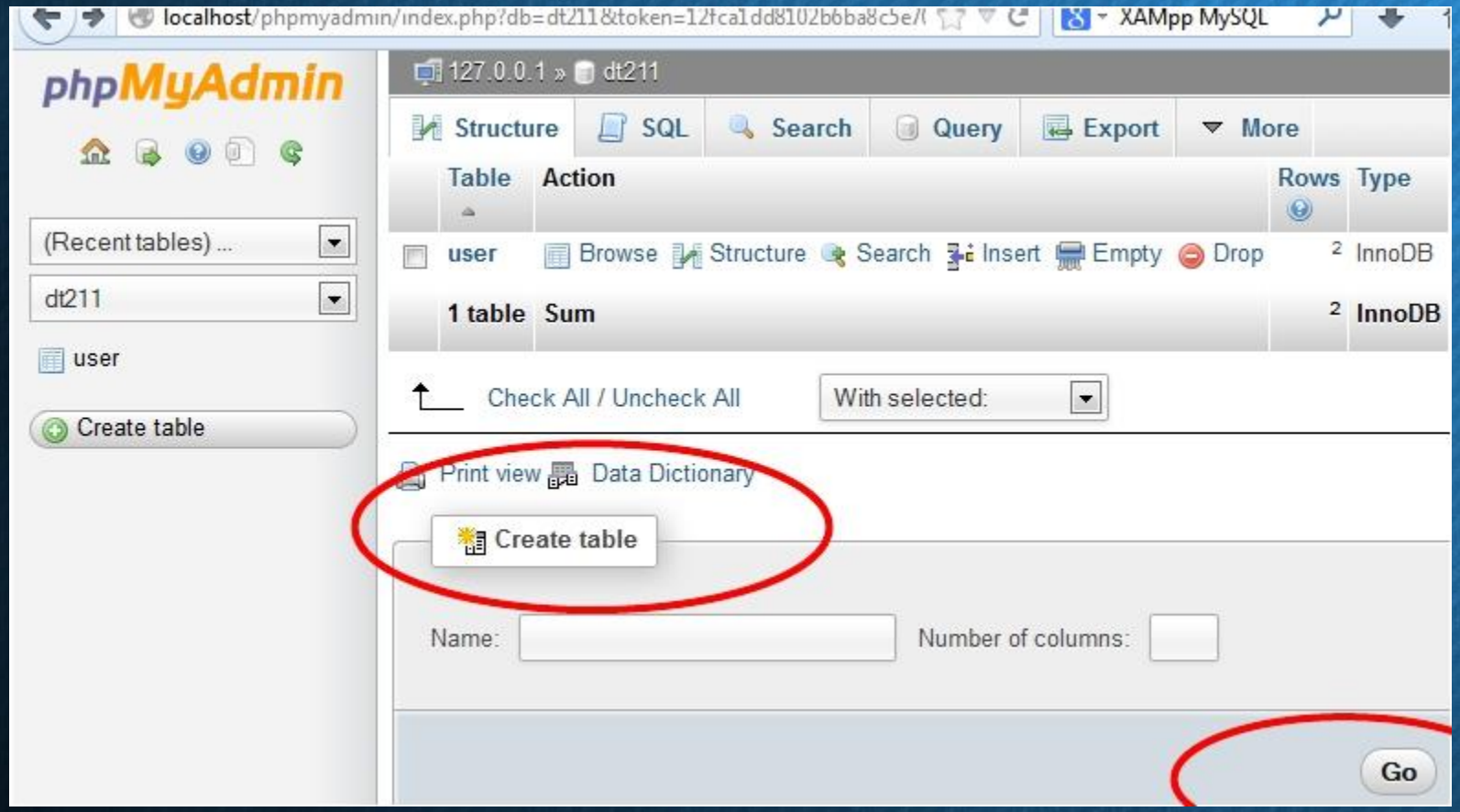
Creating a MySQL database using XAMPP

- Enter a name for the database, then click on the Create button. The name must be 64 characters or less and composed of letters, numbers and underscores. Avoid using numbers at the start of the name.



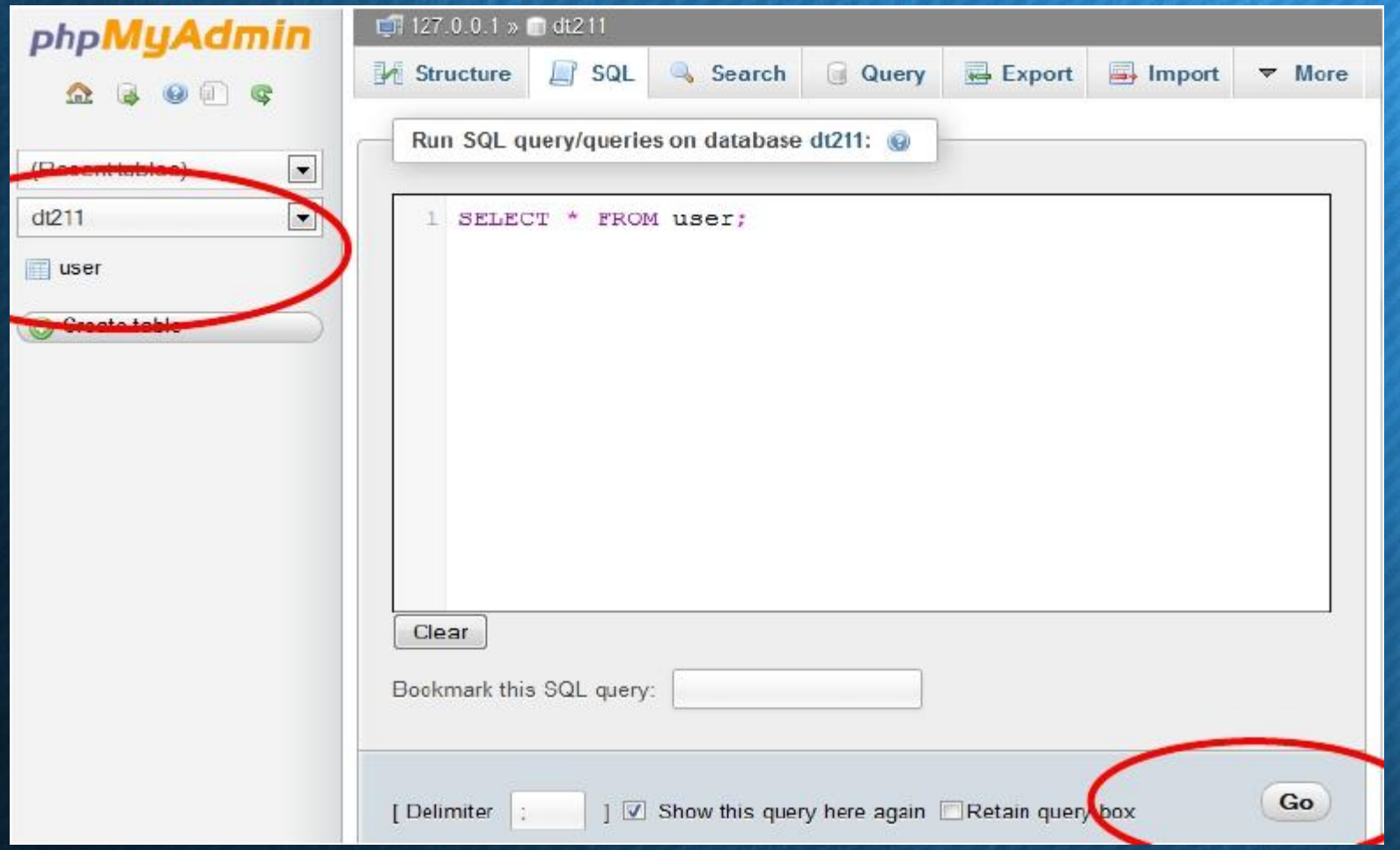
Creating a MySQL database using XAMPP

- Ensure the database was successfully created. Then you can start to create tables and relationships among tables.



Creating a MySQL database using XAMPP

- You can also run SQL query to create/alter tables and manipulate data in tables.



PHP Connect to MySQL

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

- MySQLi extension (the "i" stands for improved), **all our examples are using MySQLi**
- PDO (PHP Data Objects)

Access MySQL Using PHP

Parameter	Description
host	Optional. Either a host name or an IP address
username	Optional. The MySQL user name
password	Optional. The password to log in with
dbname	Optional. The default database to be used when performing queries

- Note: There are more available parameters, but the ones listed above are the most important.

PHP Connect to MySQL Example

```
<?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";
?>
```

Create a MySQL Database Using MySQLi

```php
<?php
$servername = "localhost";
$username = "root";
$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();
?>
```

Create a Table in MySQL Database

```php
<?php
$servername = "localhost";
$username = "root";
$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
$sql = "CREATE TABLE user (
UserID INT(6) UNSIGNED AUTO_INCREMENT
PRIMARY KEY,
UserName VARCHAR(30) NOT NULL,
Password VARCHAR(30) NOT NULL,
email VARCHAR(50) )";
if ($conn->query($sql) === TRUE) {
  echo "Table user created
successfully";
} else {
  echo "Error creating table: " .
$conn->error; }
$conn->close();
?>
```

Insert Data Into a Table

- The INSERT INTO statement is used to add new records to a database table.

Syntax

- It is possible to write the INSERT INTO statement in two forms.
- The first form doesn't specify the column names where the data will be inserted, only their values:

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

- The second form specifies both the column names and the values to be inserted:

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

Insert Data From a Form Into a Database

- Now we will create an HTML form that can be used to add new records to the “user" table.
- Here is the HTML form:

```
<p>Add A New User</p>
<form method="post">
<p>Name:
<input type="text" name="name"></p>
<p>Email:
<input type="email" name="email"></p>
<p>Password:
<input type="password" name="password"></p>
<p><input type="submit" value="Add New"/></p>
</form>
```

Insert Data From a Form Into a Database

- When a user clicks the submit button in the HTML form in the example last page, the form data is insert to database.
- Here is the php part:

```
<?php
require_once "database.php";
if ( isset($_POST['name']) && isset($_POST['email'])
&& isset($_POST['password'])) {
$n = $_POST['name'];
$e = $_POST['email'];
$p = $_POST['password'];
$sql = "INSERT INTO user (UserName, Password, email)
VALUES ('$n', '$p', '$e')";
if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error; }
$conn->close(); }
?>
```

Insert Data From a Form Into a Database

- Put together:

```
<?php
require_once "database.php";
if ( isset($_POST['name']) && isset($_POST['email'])
&& isset($_POST['password'])) {
$n = $_POST['name'];
$e = $_POST['email'];
$p = $_POST['password'];
$sql = "INSERT INTO user (UserName, Password, email)
VALUES ('$n', '$p', '$e')";
if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}
$conn->close();
}
?>
<p>Add A New User</p>
<form method="post">
<p>Name:
<input type="text" name="name"></p>
<p>Email:
<input type="email" name="email"></p>
<p>Password:
<input type="password" name="password"></p>
<p><input type="submit" value="Add New"/></p>
</form>
```

Select Data From a Database Table

- The SELECT statement is used to select data from a database
- Syntax:

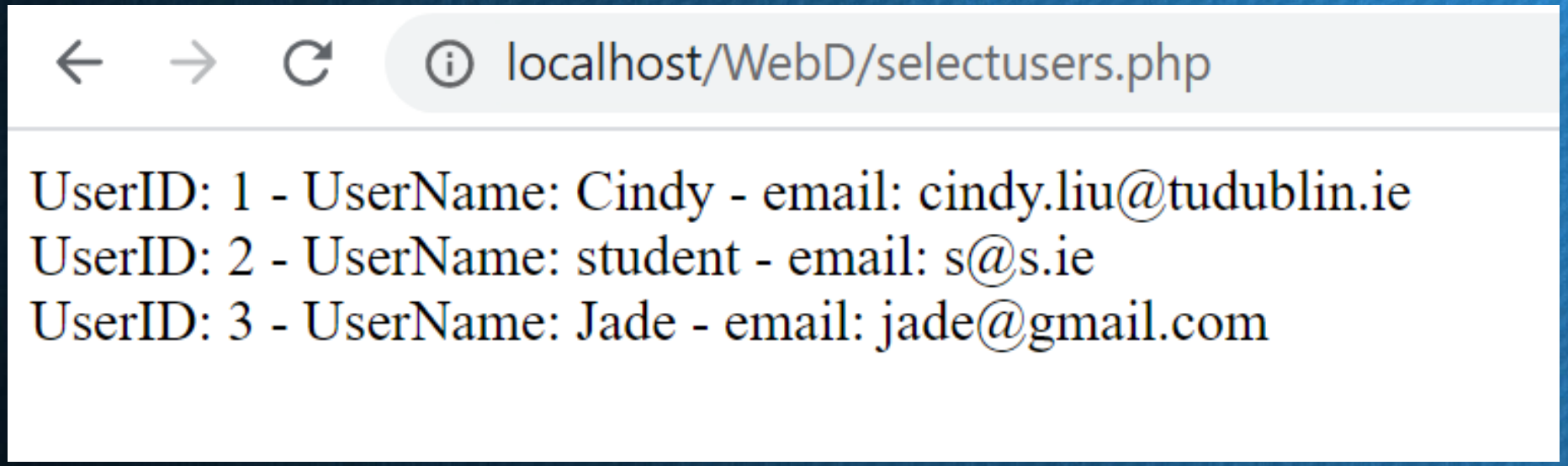
```
SELECT   column_name(s)
FROM   table_name;
```

- To get PHP to execute the statement above we must use the mysqli_query() function. This function is used to send a query or command to a MySQL connection.

SELECT Example

```
<?php
$servername = "localhost";
$username = "root";
$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 UserID, Username,
Password, email FROM user";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
  // output data of each row
while($row = $result-
>fetch_assoc()) {
 echo "UserID: " . $row["UserID"].
" - UserName: " . $row["Username"].
" - email: " . $row["email"].
"<br>";
} }else {
  echo "0 results";
}
$conn->close();
?>
```

Display the Result



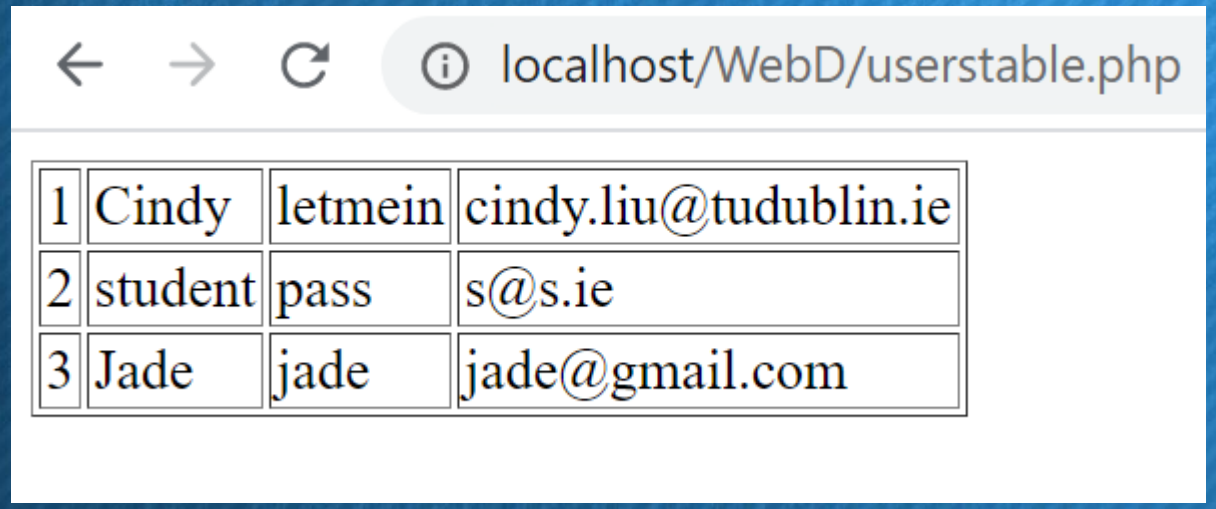
Display the Result in an HTML Table

The following example selects the same data as the example above, but will display the data in an HTML table:

```php
if ($result->num_rows > 0) {
  // output data of each row in a table
echo "<table border='1'>";
while($row = $result->fetch_assoc()) {

echo "<tr><td>";
echo($row["UserID"]);
echo("</td><td>");
echo($row["Username"]);
echo("</td><td>");
echo($row["Password"]);
echo("</td><td>");
echo($row["email"]);
echo("</tr>\n");
}
echo "</table>\n";
} else {
  echo "0 results";
}
$conn->close();
?>
```

Display the Result in an HTML Table

The output of the code above will be:



The WHERE clause

- The WHERE clause is used to extract only those records that fulfil a specified criterion.
- Syntax:

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

- To get PHP to execute the statement above we must use the mysqli_query() function again.

The ORDER BY Keyword

- The ORDER BY keyword is used to sort the data in a recordset.
- The ORDER BY keyword sort the records in ascending order by default.
- If you want to sort the records in a descending order, you can use the DESC keyword.
- Syntax:

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

Update Data In a Database

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

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

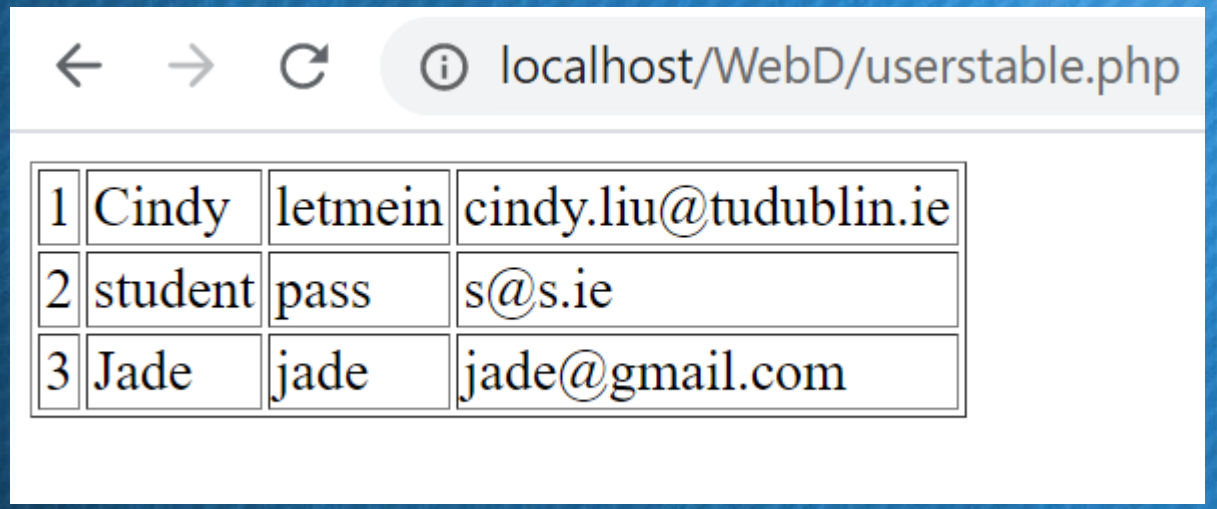
Update Example

The following example updates some data in the “user" table

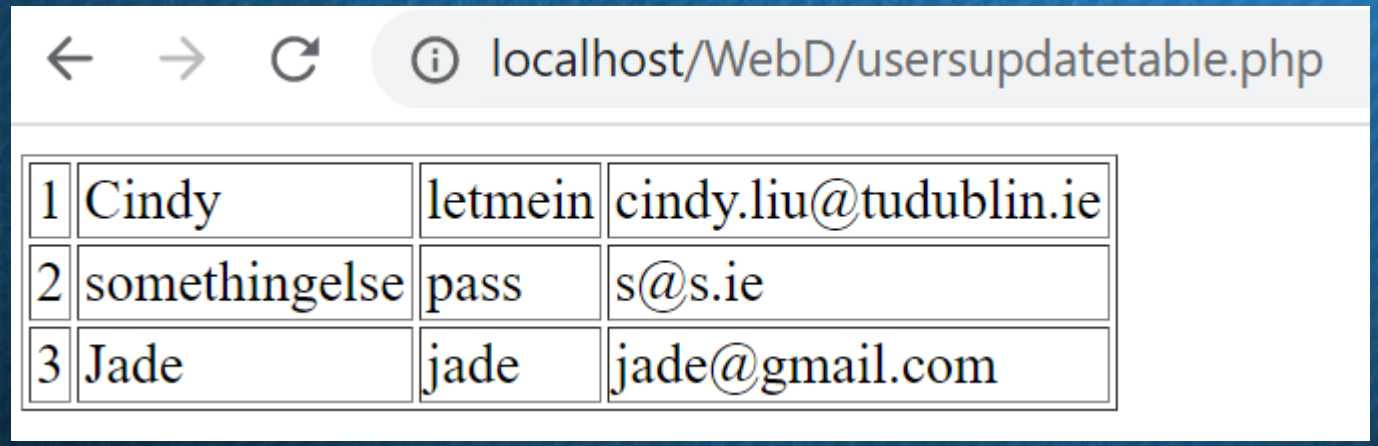
```php
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
//Update Username
$sql = "UPDATE user SET Username = 'somethingelse' WHERE UserID=2";
$conn->query($sql);
//Display user data
$sql = "SELECT UserID, Username, Password, email FROM user";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
  // output data of each row in a table
echo "<table border='1'>";
while($row = $result->fetch_assoc()) {

echo "<tr><td>";
echo($row["UserID"]);
echo("</td><td>");
echo($row["Username"]);
echo("</td><td>");
echo($row["Password"]);
echo("</td><td>");
echo($row["email"]);
echo("</tr>\n");
}
echo "</table>\n";
} else {
  echo "0 results";
}
```

Display the Result in an HTML Table

Before the update:



The output of the code above will be:



Delete Data In a Database

- The DELETE FROM statement is used to delete records from a database table.
- Syntax:

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

Delete Data In a Database

- 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!
- The following example deletes the record in the “user" table where UserID is gaven:

Delete Data In a Database

```php
<?php
require_once "database.php";
if ( isset($_POST['id']) ) {
$id = $_POST['id'];
$sql = "DELETE FROM user WHERE UserID = $id";
echo "<pre>\n$sql\n</pre>\n";
$conn->query($sql);
}
?>
<p>Delete A User</p>
<form method="post">
<p>UserID to Delete:
<input type="text" name="id"></p>
<p><input type="submit" value="Delete"/></p>
</form>
```

Summery

- Making database connections
- Sanitizing HTML
- Inserting and Deleting
- Security: Sanitizing SQL