# UNIT VI: PHP and MySQL

## 6.1 PHP: Introduction and basic syntax of PHP

PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side.

PHP was created by Rasmus Lerdorf in 1994 but appeared in the market in 1995. PHP 7.4.0 is the latest version of PHP, PHP is 8.2.8, released on July 4th, 2023.

PHP stands for Hypertext Preprocessor.

PHP is an interpreted language, i.e., there is no need for compilation.

PHP is a server-side scripting language, which is used to manage the dynamic content of the website.

PHP can be embedded into HTML.

PHP Features

**Open Source:**
PHP source code and software are freely available on the web. You can develop all the versions of PHP according to your requirement without paying any cost. All its components are free to download and use.

**Platform Independent:**

PHP is available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.

**Database Support:**

PHP supports all the leading databases such as MySQL

**Basic PHP Syntax**

A PHP script can be placed anywhere in the document.

A PHP script starts with **<?php** and ends with **?>**
**Ex.**

```
<?php
// PHP code goes here
?>
```

The default file extension for PHP files is ".php".

A PHP file normally contains HTML tags, and some PHP scripting code.

Statements are expressions terminated by semicolons.
Expressions are combinations of tokens.
Braces make blocks.
PHP is case sensitive.

## 6.2 decision and looping with examples

PHP provides us with four conditional or decision statements:

**1 if statement**
This statement allows us to set a condition. On being TRUE, the following block of code enclosed within the if clause will be executed
Syntax
```
if (condition){
   // if TRUE then execute this code
}
```
Ex.
```
<?php
$x = 12;

if ($x > 0) {
   echo "The number is positive";
}
?>
```

**2 if…else statement**
If a condition is TRUE then if block gets executed, otherwise else block gets executed.
Syntax
```
if (condition) {
   // if TRUE then execute this code
}
else{
   // if FALSE then execute this code
}
```

Ex.
```
<?php
```

```
$x = -12;

if ($x > 0) {
   echo "The number is positive";
}

else{
   echo "The number is negative";
}
?>
```

**3 if…elseif…else statement**

This allows us to use multiple if…else statements. We use this when there are multiple conditions of TRUE cases.

```
if (condition) {
   // if TRUE then execute this code
}
elseif {
   // if TRUE then execute this code
}
elseif {
   // if TRUE then execute this code
}
else {
   // if FALSE then execute this code
}
```

Ex.

```
<?php
$x = "August";

if ($x == "January") {
   echo "Happy Republic Day";
}

elseif ($x == "August") {
   echo "Happy Independence Day!!!";
}
else{
   echo "Nothing to show";
}
?>
```

**4 switch statement**

The “switch” performs in various cases i.e., it has various cases to which it matches the condition and appropriately executes a particular case block.

The **break** statement is used to stop the automatic control flow into the next cases and exit from the switch case.

The **default** statement contains the code that would execute if none of the cases match.

Syntax:

```
switch(n) {
   case statement1:
      code to be executed if n==statement1;
      break;
   case statement2:
      code to be executed if n==statement2;
      break;
   default:
      code to be executed if n != any case;
}
```

Ex.

```
<?php
$n = "February";

switch($n) {
   case "January":
      echo "Its January";
      break;
   case "February":
      echo "Its February";
      break;
   case "March":
      echo "Its March";
      break;
   default:
      echo "Doesn't exist";
}
?>
```

## B] Looping Statement

PHP supports four types of looping techniques.

### 1 for loop

This type of loops is used when the user knows in advance, how many times the block needs to execute.

These type of loops are also known as entry-controlled loops.
There are three main parameters to the code
Initialization, condition and increment/decrement.
Syntax:

```
for (initialization expression; test condition; update expression) {
   // code to be executed
}
```

Ex.

```
<?php
for ($num = 1; $num <= 10; $num += 2) {
   echo "$num \n";
}
?>
```

Output: 4 6 8 10 12.

### 2 while loop

The while loop is also an entry control loop like for loops i.e., it first checks the condition at the start of the loop and if its true then it enters the loop and executes the block of statements.

Syntax:

```
while (condition) {
   // code is executed
}
```

Ex.

```
<?php
$num = 2;
while ($num < 12) {
   $num += 2;
   echo $num, "\n";
} ?>
```

Output: 4 6 8 10 12

**3 do-while loop**
This is an exit control loop which means that it first enters the loop, executes the statements, and then checks the condition.

a statement is executed at least once on using the do…while loop.

Syntax:

```
do {
   //code is executed
} while (if condition is true);
```

Ex.

```
<?php
  $num = 2;
do {
   $num += 2;
   echo $num, "\n";
} while ($num < 12);

?>
```

Output: 4 6 8 10 12.

**4 foreach loop**
This loop is used to iterate over arrays. For every counter of loop, an array element is assigned and the next counter is shifted to the next element.

**Syntax :**

```
foreach (array_element as value) {
  //code to be executed
}
```

**Ex.**

```
<?php
   $arr = array (10, 20, 30, 40, 50, 60);
   foreach ($arr as $val) {
      echo "$val \n";
   }
   $arr = array ("Ram", "Laxman", "Sita");
   foreach ($arr as $val) {
      echo "$val \n";
   } ?>
```

Output: 10 20 30 40 50 60 Ram Laxman Sita.

## 6.3 PHP and HTML

PHP stands for Hypertext Preprocessor, which is an open source scripting language. It is a server-side scripting language and a powerful tool for creating a dynamic and interactive website.

HTML stands for Hypertext Markup Language, which is used to create web pages. It is basically used to create static web pages, but it can integrate with CSS, JavaScript, and PHP.

PHP is used in backend development, which interacts with databases to retrieve, store, and modify the information.

HTML is used in frontend development, which organizes the content of the website.

PHP is used to create a dynamic website. The output will depend on the browser.

HTML is used to create a static website. The output of static website remains the same on each time.

PHP code executes on web servers like Apache web server, IIS web server.

HTML code executes on web browsers like Chrome, Internet Explorer, etc.

PHP files save with .php extension.

HTML files save with .html extension.

## 6.4 Arrays

Array allows us to store multiple elements of similar or different data types under a single variable

An array is created using an array() function in PHP

Types of Array in PHP

**1 Indexed or Numeric Arrays:**
These type of arrays can be used to store any type of element, but an index is always a number. By default, the index starts at zero.

Syntax: $array_name=array("element 1","element 2",……,"element n");

Ex.

```
<?php
 $name_one = array("Zack", "Anthony", "Ram", "Salim", "Raghav");
foreach ($name_one as $val){
   echo $val. "\n";
}

//or
$round = count($name_one);
for($n = 0; $n < $round; $n++){
   echo $name_one[$n], "\n";
}

?>
```

**2 Associative Arrays:**
These types of arrays are similar to the indexed arrays but instead of linear storage, every value can be assigned with a user-defined key of string type.

```
<?php
$name_one = array("A"=>"1000", "B"=>"2000", "C"=>"3000");
echo $name_one["A"];
?>
Output:1000
```

**3 Multidimensional Arrays:**
Multi-dimensional arrays are such arrays that store another array at each index instead of a single element.

```
<?php

// Defining a multidimensional array
$favorites = array(
   array("name" => "GPS","mob" => "9996363636"),
   array(      "name" => "ABC","mob" => "9255566555"),
   array("name" => "XYZ","mob" => "9875147536"));

echo "ABC mobile number is: " . $favorites[1]["mob"];
?>
```

## 6.5 Functions

A function is a block of code written in a program to perform some specific task.

PHP provides us with two major types of functions:

**Built-in functions :**

PHP provides us with huge collection of built-in library functions. Such as echo() and print().

**User Defined Functions :**

Apart from the built-in functions, PHP allows us to create our own customised functions called the user-defined functions.

Creating a Function

Any name ending with an open and closed parenthesis is a function.
A function name always begins with the keyword function.
To call a function we just need to write its name followed by the parenthesis
A function name cannot start with a number. It can start with an alphabet or underscore.
A function name is not case-sensitive.

**Syntax:**

```
function function_name(){
   executable code;
}
```

**Example:**

```
<?php

// function along with three parameters
function display($num1, $num2, $num3)
{
   $product = $num1 * $num2 * $num3;

   return $product;  //returning the product
}

$retValue = display(2, 3, 5); // function call
echo "The product is $retValue";
?>
```

## 6.6 String

String is a collection of character. There are three ways of creating strings in PHP:

### 1. Single-quote strings:

This type of string does not process special characters inside quotes.

**Ex.**
```
$name='Gopal Shinde'
echo ' Your Name is  $name';// is not print variable value
```

**Output:**
```
Your Name is  $name
```

### 2.Double-quote strings :

double-quote strings in PHP are capable of processing special characters.

**Ex.**
```
$name='Gopal Shinde'
echo "Your Name is  $name";// is print variable value
```

**Output:**
```
Your Name is  Gopal Shinde
```

### 3. Heredoc:

Use the heredoc (<<< ) operator, after identifier name and which any text can be written as a new line is started.  To close the syntax, the same identifier is given without any tab or space.

**Ex.**
```
<?php
 $input  = <<<testHeredoc
Welcome to Php.
Started content writing in Php!.
I am enjoying this.
testHeredoc;
echo $input;
?>
```

**Output**:
```
Welcome to Php.Started content writing in Php!.I am enjoying this.
```

## 6.7 Form processing

HTML forms are used to send the user information to the server and returns the result back to the browser.

Then, the information can be validated either at the client-side or on the server-side.

To create a HTML form, form tag should be used. Attributes of Form Tag

**name or id** - It specifies the name of the form and is used to identify individual forms.

**action** - It specifies the location to which the form data has to be sent when the form is submitted.

**method** - It specifies the HTTP method that is to be used when the form is submitted. The possible values are get and post. If get method is used, the form data are visible to the users in the url. Default HTTP method is get.

**Form Validation**: Form validation is done to ensure that the user has provided the relevant information. Basic validation can be done using HTML elements.

PHP methods and arrays used in form processing are:

**isset():** This function is used to determine whether the variable or a form control is having a value or not.

**$_GET[]:** It is used the retrieve the information from the form control through the parameters sent in the URL. It takes the attribute given in the url as the parameter.

**Syntax**
$variable_name=$_GET [“form field name”];
Ex: $name=$_GET [“t1”];

**$_POST[]:** It is used the retrieve the information from the form control through the HTTP POST method. IT takes name attribute of corresponding form control as the parameter.

**Syntax**
$variable_name=$_POST [“form field name”];

**$_REQUEST[]:** It is used to retrieve an information while using a database.

**Example:**

```
if (isset($_POST['submit']))
{
   if ((!isset($_POST['firstname'])) || (!isset($_POST['lastname'])))
   {
      $error = "*" . "Please fill all the required fields";
   }
   else
   {
      $firstname = $_POST['firstname'];
      $lastname = $_POST['lastname'];
   }
}
?>
```

## 6.8 Date and Time Functions

The date/time functions allow you to get the date and time from the server where your PHP script runs.

Date and time are stored in computer in UNIX Timestamp format. It calculates time in number of seconds on GMT (greenwich mean time) i.e started from January 1, 1970, 00:00:00 GMT.

**Syntax**

```
string date ( string $format [, int $timestamp = time() ] )
```

**Parameters**

**Format**- Specifies the format of returned date and time

**Timestamp**- Current date can be used in place of timestamp

**Example**

```
<?php
$today =date('d/m/y');
echo $today;
?>
```

**Output** : 14/03/24

**Formatting options available in date() function:**

**d**: Represents day of the month; two digits with leading zeros (01 or 31).
**D**: Represents day of the week in the text as an abbreviation (Mon to Sun).
**m**: Represents month in numbers with leading zeros (01 or 12).
**M**: Represents month in text, abbreviated (Jan to Dec).
**y**: Represents year in two digits (08 or 14).
**Y**: Represents year in four digits (2008 or 2014).

The parts of the date can be separated by inserting other characters, like hyphens (-), dots (.), slashes (/), or spaces to add additional visual formatting.

The following characters can be used along with the **date() function to format the time string:**

**h**: Represents hour in 12-hour format with leading zeros (01 to 12).
**H**: Represents hour in 24-hour format with leading zeros (00 to 23).
**i**: Represents minutes with leading zeros (00 to 59).
**s**: Represents seconds with leading zeros (00 to 59).
**a**: Represents lowercase antemeridian and post meridian (am or pm).
**A**: Represents uppercase antemeridian and post meridian (AM or PM).

**Example :**

```
<?php
  echo date("h:i:s") . "\n";
  echo date("M,d,Y h:i:s A") . "\n";
  echo date("h:i a");
?>
```

**Output:**

```
07:59:28 Mar,14,2024 07:59:28 AM 07:59 am
```

## 6.9 Sending Email

One of useful benefits of PHP is sending Email from PHP is easy. On a properly configured server, the process is simple. PHP provides mail() function for sending email the syntax of this function is:
mail (to, subject, body, [headers]);

The to value or parameter to this function should be an email address or a series of addresses, separated by commas.

Any of these are allowed:
darsh@rediff.com,
darsh@rediff.com,
harsh@gmail.com.

```
$to = “darsh@rediff.com”;
$subject = “This is the subject”;
$body = 'This is the body’.
“It goes over multiple lines”;

mail ($to, $subject, $body);
```

You can create an email message that goes over multiple lines by having the text do exactly that within the quotation marks. You can also use the newline character (\n) within double quotation marks to accomplish this:

```
$body = "This is the body.\n It goes over multiple lines.”;
```

The mail() function takes a fourth, optional parameter for additional headers. This is where you could set the From, Reply-To, Cc, Bcc, and similar settings. For example,

```
mail ($to, $subject, $body, 'From:reader@example.com');
```

To use multiple headers of different types in your email, separate each with \r\n:

```
$headers="From:amol@yahoo.com\r\n";
$headers= $headers ."Cc: darsh@rediff.com, harsh@gmail.com\r\n"; mail ($to,
$subject, $body, $headers);
```

Although this fourth argument is optional, it is advised that you always include a from value (although that can also be established in PHP’s configuration file).

## 6.10 Files

The global predefined variable $_FILES is an associative array containing items uploaded via HTTP POST method. Uploading a file requires HTTP POST method form with enctype attribute set to multipart/form-data.

The _FILES array contains following properties –

**$_FILES['file']['name'] -** The original name of the file to be uploaded.

**$_FILES['file']['type'] -** The mime type of the file.

**$_FILES['file']['size'] -** The size, in bytes, of the uploaded file.

**$_FILES['file']['tmp_name'] -** The temporary filename of the file in which the uploaded file was stored on the server.

```
<form action="testscript.php" method="POST" enctype="multipart/form-
data">
  <input type="file" name="file">
  <input type ="submit" value="submit">
</form>
```

The PHP script is as follows:

**Example**

```
<?php
echo "Filename: " . $_FILES['file']['name']."<br>";
echo "Type : " . $_FILES['file']['type'] ."<br>";
echo "Size : " . $_FILES['file']['size'] ."<br>";
echo "Temp name: " . $_FILES['file']['tmp_name'] ."<br>";
echo "Error : " . $_FILES['file']['error'] . "<br>";
?>
```

**Output:**

```
Filename: hello.html
Type : text/html
Size : 56
Temp name: C:\xampp\tmp\php32CE.tmp
Error : 0
```

## 6.11 Cookies and Sessions

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer.

### Create Cookies With PHP

A cookie is created with the setcookie() function.

### Syntax

```
setcookie(name, value, expire, path, domain, secure, httponly);
```

Only the name parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

```
$cookie_name = "user";
$cookie_value = "Gopal Shinde";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
```

The "/" means that the cookie is available in entire website.

### Sessions

Session is the time span between login & logout on a particular website by user.

A PHP session solves this problem by allowing you to store user information on the server for later use

### Starting a PHP Session

Before you can store user information in your PHP session, you must first start up the session.

The session_start() function must appear BEFORE the <html> tag:

```
<?php session_start(); ?>
<html><body></body></html>
```

Storing a Session Variable

The correct way to store and retrieve session variables is to use the PHP $_SESSION variable:

## 6.12 Connecting to MySQL and Selecting the Database

The first step for Connecting to MySQL is to call mysqli_connect() function
The Syntax for this function is as follows:

```
$conn = mysqli_connect (hostname or servername, username, password,
db_name);
```

The first three arguments sent to the function (hostname, username, and password) are based upon the users of MySQL database.

The hostname value will be localhost.
The fourth argument is the name of the database to use.
This is the equivalent of saying USE databasename within the mysql client.
If the connection was made, the $con variable, short for database connection, will become a reference point for all of your subsequent database interactions. Most of the PHP functions for working with MySQL will take this variable as its first argument.

**Create a Connection to a MySQL Database**
Before you can access data in a database, you must create a connection to the database. In PHP, this is done with the mysqli_connect() function.

**Servername** – Specifies the server to connect to. Default value is "localhost".

**Username** - Specifies the username to log in with. Default value is the name of the user that owns the server process.

**Password** - Specifies the password to log in with. Default is ""

**DatabaseName**- to provide database name created in PhpMyAdmin.

Example
```
<?php
$con = mysqli_connect("localhost","root","",”my_db”);
if (!$con)
{
Print 'Could not connect: ' . mysqli_error();
}
Else
{
Print 'Connect ‘;
}
?>
```

## Closing a Connection

The connection will be closed automatically when the script ends. To close the connection before, use the mysqli_close() function:

**Syntax :** mysqli_close(connection variable);

## Selecting Database:

PHP uses mysqli_select_db function to select the database on which queries are to be performed. This function takes two parameters and returns TRUE on success or FALSE on failure.

**Syntax :** mysqli_select_db ( connection variable , string $dbname ) : bool

**Ex.**

```
$db = mysqli_select_db( $conn, 'stud' );
if(! $db) {
        die('Could not select database: ' . mysqli_error($conn));
}
```

## 6.13 Executing Simple Queries

Once you have successfully connected to and selected a database, you can start performing queries. These queries can be as basic as inserts, updates, and deletions or as involved as complex joins returning numerous rows.

In any case, the PHP function for executing a query is mysqli_query():

**Syntax :** mysqli_query(connection variable , string $query);

Ex. $result=mysqli_query($conn,$sql);

The mysqli_query() function takes the database connection as its argument and the query itself.

For simple queries like INSERT, UPDATE, DELETE, etc. (which do not return records)

**Example**

```
$sql=”select * from table_name where column_name=”value” ”;
$result=mysqli_query($conn,$sql);
```

## 1 Inserting Record into Database

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

The SQL query must be quoted in PHP
String values inside the SQL query must be quoted
Numeric values must not be quoted

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

**Syntax:**
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)

**Example**

```
<?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 MyMylist (firstname, lastname, email)
VALUES ('Shinde', 'Gopal', 'gopal@gmail.com')";
$sql = "DELETE FROM MyGuests WHERE id=3";
if (mysqli_query($conn, $sql)) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

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

## 6.14 Retrieving Query Results

First, we set up an SQL query that selects the column name from table.
The SELECT statement is used to select data from one or more tables:

**Syntax**

SELECT column_name(s) FROM table_name

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

SELECT * FROM table_name
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 no of column name.

**Example**

```
<?php
// Create connection
$conn = mysqli_connect("localhost","root","" ,"myDB");
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyList";
$result = mysqli_query($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);?>
```

## 6.15 Ensuring Secure SQL

Database security with respect to PHP comes down to three broad issues:

1. Protecting the MySQL access information

2. Not display too much about the database.

3. Being careful when running queries, particularly those involving user submitted data.

You can accomplish the first objective by securing the MySQL connection script outside of the Web directory so that it is never viewable through a Web browser.

The second objective is achieved by not letting the user see PHP's error messages or your queries (in these scripts, that information is printed out for your debugging purposes; you'd never want to do that on a live site).

For the third objective, there are numerous steps you can and should take, all based upon the premise of never trusting user supplied data.

**First**, validate that some value has been submitted, or that it is of the proper type (number, string, etc.).

**Second**, use regular expressions to make sure that submitted data matches what you would expect.

**Third**, you can typecast some values to guarantee that they're numbers.

**Fourth** recommendation is to run user submitted data through the mysql_real_ escape_string() function.

This function cleans data by escaping what could be problematic characters.

$clean = mysql_real_escape_string($dbc, data);

For security purposes, **mysql_real_escape_string()** should be used on every text input in a form.

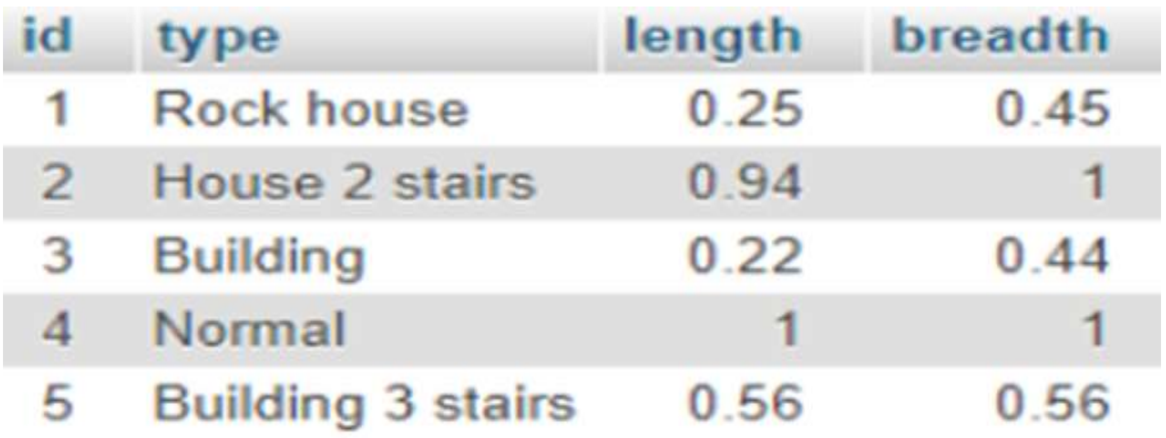
## 6.16 Counting Returned Records

We can get the total number of rows in a table by using the MySQL **mysqli_num_rows()** function.

**Syntax:**

```
mysqli_num_rows( result );
```

The result is to specify the result set identifier returned by mysqli_query() function.



| id | type | length | breadth |
|---|---|---|---|
| 1 | Rock house | 0.25 | 0.45 |
| 2 | House 2 stairs | 0.94 | 1 |
| 3 | Building | 0.22 | 0.44 |
| 4 | Normal | 1 | 1 |
| 5 | Building 3 stairs | 0.56 | 0.56 |

**Example:**

```
$sql = "SELECT * from building";

if ($result = mysqli_query($con, $sql)) {

   // Return the number of rows in result set
   $rowcount = mysqli_num_rows( $result );

   // Display result
   printf("Total rows in this table :  %d\n", $rowcount);
 }
```

**Output:**

```
Total rows in this table : 5
```

We count the table rows using MySQL count () function. It's an aggregate function used to count rows.

**Syntax:** select count (*) from table;

## 6.17 Updating Records with PHP

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
```

**Example:**

```
<?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 Mylist SET lastname='Ghodke' WHERE id=1";

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

mysqli_close($conn);
?>
```

**The End**