Lab 10: Working with BLOB (Binary Large Object)

MySQL provides a BLOB type that can hold a large amount of data. The maximum value of a BLOB object is specified by the available memory and the communication package size, which can be changed by using the max_allowed_packet variable in MySQL and post_max_size in the PHP settings.

In this lab exercise, you will learn how to upload binary objects to a server and save them into database fields. These binary objects include pictures, documents, executables, etc. In addition, you will learn how to prepare them for downloading, both inline and as individual objects.

You need to have completed previous lab exercises Handling File Uploads and Working with Database before continuing.

10.1 Database Field for BLOB

Database field needs to be set as a BLOB data type in order to store a binary object.

For this exercise, add a BLOB field *ProductPhoto* to table *Product*. This additional field will store a product picture.

10.2 Upload a File into Database

The technique to upload a file into a database has three parts: HTML upload, reading uploaded file into string, and inserting the string into a database field.

You have learnt how to upload a file using an HTML and storing it into a folder in previous lab exercise. We will revisit and expand on that exercise. You also have learnt how to insert a string into a database field in previous lab. The missing link is how to read an uploaded file into string, and this is achieved using keyword *file_get_contents*.

Format : <variable> = file_get_contents ($_FILES ['<uploaded file>'] ['tmp_name']);

Example : $uploaded_photo = file_get_contents ($_FILES ['photo'] ['tmp_name']);

Please refer to Figure 1 to 3 below for an example.

```
<form action="db_uploadphoto.php" method="post" enctype="multipart/form-data">
    Select image to upload:
    <input type="file" name="FileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>
```

Figure 1: HTML form for uploading a file.

```php
<?php

$uploaded_photo = file_get_contents ($_FILES["FileToUpload"]["tmp_name"]);

?>
```

Figure 2: Reading the uploaded file, converting its content into a string, and placing the string into a variable.

```php
<?php

$prodCat = "Cable";

$dbConn = new PDO("mysql:host=localhost;dbname=classlabs", "root", "MySQL", $opt);

$sql = "UPDATE product SET ProductPhoto = ? WHERE ProductCategory = ?";
$dbrs = $dbConn->prepare($sql);
$dbrs->execute(array($uploaded_photo, $prodCat));

$numrow = $dbrs->rowCount();
echo "There are $numrow row(s) affected" . "\n<br>";

?>
```

Figure 3: Upload photo into Product table, updating all records where ProductCategory is Cable.

10.3 Display Stored Images from Database

Figure 4 below shows an example code to retrieve a stored image from Product table where Product ID is 1.

Note that it uses keyword *fetch*. This keyword is used to fetch a single line of record from a selected *recordset*. For a *recordset* with multiple lines of records, follow the example in previous lab exercise that uses *foreach* loop.

```php
<?php

$dbConn = new PDO("mysql:host=localhost;dbname=classlabs", "root", "MySQL", $opt);

$sql = "SELECT * FROM product WHERE ProductID = 1";
$dbrs = $dbConn->prepare($sql);
$dbrs->execute();
$dbrow = $dbrs->fetch();

echo "Product image for : <p>";
echo "Product ID: " . $dbrow["ProductID"] . "<br>";
echo "Product Name: " . $dbrow["ProductName"] . "<br>";
echo "Product Category: " . $dbrow["ProductCategory"] . "<p>";

echo "<img src='" . base64_encode( $dbrow["ProductPhoto"] ) . "'/>";

?>
```

Figure 4: Retrieve a stored image from a database and display it on the web browser.

Your tasks:

1. Write a PHP code to update all records in Product table with individual images.
2. Load all records from Product table into an HTML table, along with their respective images.
3. Allow the user to update individual product image.

-- END --