UNIT 5

INTRODUCTION TO MySQL

Database is a systematic collection of data. Databases support storage and manipulation of data. Databases make data management easy. In simple words data can be facts related to any object in consideration. For example your name, age, height, weight, etc. are some data related to you.

Database Management System (DBMS) is a collection of programs which enables its users to access database, manipulate data, reporting / representation of data. It also helps to control access to the database.

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is released under an open-source license. So you have nothing to pay to use it. It is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.

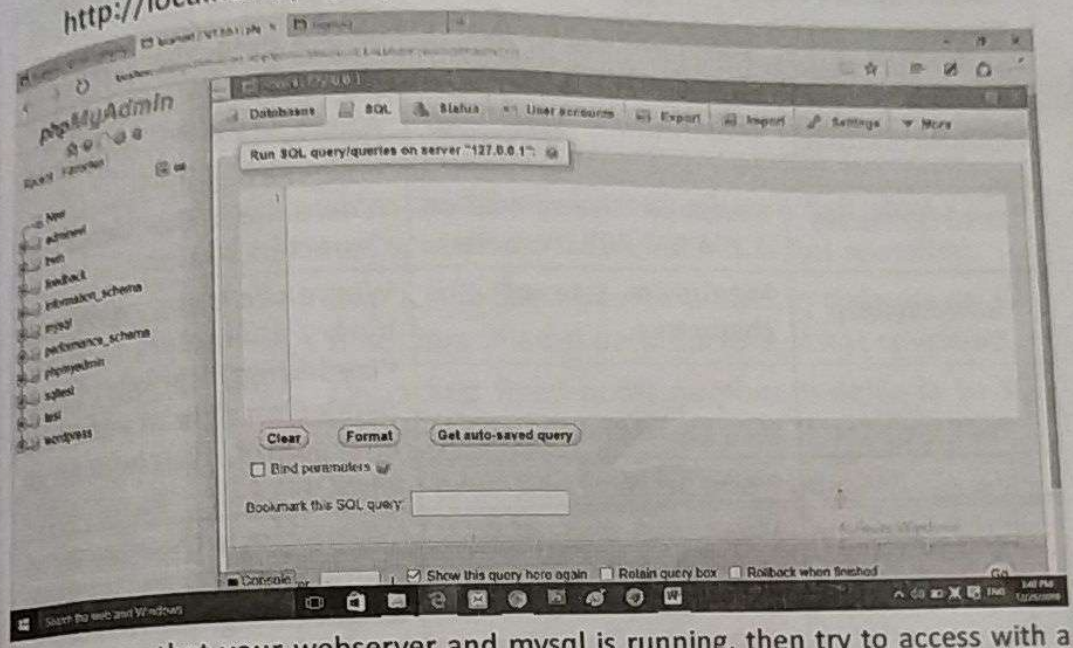
MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

MySQL works very well in combination of various programming languages like PERL, C, C++, JAVA and PHP. Out of these languages, PHP is the most popular one because of its web application development capabilities.

PHP provides various functions to access the MySQL database and to manipulate the data records inside the MySQL database. You would require to call the PHP functions in the same way you call any other PHP function.

The following figure shows the workspace of MySQL. For accessingMySQL use the following url in browser.

http://localhost/phpmyadmin/



* Make sure that your webserver and mysql is running, then try to access with a web browser.

MySQL datatypes

Every column in a table has a name and a data type. The data type tells MySQL how much physical storage to set aside for the column and the form in which the data is stored.

MySQL supports a number of SQL standard data types in various categories. There are three primary categories:

- Text
- Numbers
- Dates and times

Text Datatypes

The following are the textdatatypes in MySQL:

Data Type Syntax	Maximum Size	Explanation
CHAR(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Fixed-length strings. Space padded on right to equal size characters.
VARCHAR(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Variable-length string.

TINYTEXT(size)	Maximum size of 255 characters.	Where size is the number of characters to store.
TEXT(size)	Maximum size of 65,535 characters.	Where size is the number of characters to store.
MEDIUMTEXT(size)	Maximum size of 16,777,215 characters.	Where size is the number of characters to store.
LONGTEXT(size)	Maximum size of 4GB or 4,294,967,295 characters.	Where size is the number of characters to store.
BINARY(size)	Maximum size of 255 characters.	Where size is the number of binary characters to store. Fixed-length strings. Space padded on right to equal size characters. (Introduced in MySQL 4.1.2)
VARBINARY(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Variable-length string. (Introduced in MySQL 4.1.2)

Numeric Datatypes

The following are the Numeric Datatypes in MySQL:

Data Type Syntax	Explanation
BIT	Very small integer value that is equivalent to TINYINT(1).Signed values range from -128 to 127. Unsigned values range from 0 to 255.
TINYINT(m)	Very small integer value.Signed values range from -128 to 127. Unsigned values range from 0 to 255.
SMALLINT(m)	Small integer value.Signed values range from -32768 to 32767. Unsigned values range from 0 to 65535.
MEDIUMINT(m)	Medium integer value.Signed values range from -8388608 to 8388607. Unsigned values range from 0 to 16777215.
INT(m)	Standard integer value.Signed values range from -2147483648 to 2147483647. Unsigned values range from 0 to 4294967295.
INTEGER(m)	Standard integer value.Signed values range from -2147483648 to 2147483647. Unsigned values range from 0 to 4294967295. This is same as INT data type

BIGINT(m)	Big integer value.Signed values range from -9223372036854775808 to 9223372036854775807. Unsigned values range from 0 to 18446744073709551615.
DECIMAL(m,d)	Unpacked fixed point number.m defaults to 10, if not specified.d defaults to 0, if not specified.
DEC(m,d)	Unpacked fixed point number.m defaults to 10, if not specified.d defaults to 0, if not specified.
NUMERIC(m,d)	Unpacked fixed-point number.m defaults to 10, if not specified.d defaults to 0, if not specified.
FIXED(m,d)	Unpacked fixed-point number.m defaults to 10, if not specified.d defaults to 0, if not specified.
FLOAT(m,d)	Single precision floating point number.
DOUBLE(m,d)	Double precision floating point number.
DOUBLE PRECISION (m,d)	Double precision floating point number.
REAL(m,d)	Double precision floating point number.
FLOAT(p)	Floating point number.
BOOL	Synonym for TINYINT(1)
BOOLEAN	Synonym for TINYINT(1)

Date/Time Datatypes

The following are the Date/Time Datatypes in MySQL:

Data Type Syntax	Maximum Size	Explanation
DATE	Values range from '1000-01-01' to '9999-12-31'.	Displayed as 'YYYY-MM-DD'.
DATETIME	Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIMESTAMP(m)	Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC.	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIME	Values range from '-838:59:59' to '838:59:59'.	Displayed as 'HH:MM:SS'.
YEAR[(2|4)]	Year value as 2 digits or 4 digits.	Default is 4 digits.

Binary Large Object (BLOB) Datatypes

Blob (Binary large object) collection of binary data stored as data is stored in binary format. Mainly we use blob for store large amount of files/data. These are used to store data files like images, videos, an executables, etc. BLOBs are binary strings with no character set sorting, so they are treated as numeric values while TEXT objects are treated as character strings.

The following are the BLOB Datatypes in MySQL:

Data Type Syntax	Maximum Size	Explanation
TINYBLOB	Maximum size of 255 bytes.	
BLOB(size)	Maximum size of 65,535 bytes.	Where size is the number of characters to store (size is optional and was introduced in MySQL 4.1)
MEDIUMBLOB	Maximum size of 16,777,215 bytes.	
LONGTEXT	Maximum size of 4GB or 4,294,967,295 characters.	

Getting Information of a Database

- Listing the databases on the MySQL server host

  mysql>show databases;
- Access/change database

  mysql>Use [database_name]
- Showing the current selected database

  mysql> select database();
- Showing tables in the current database

  mysql>show tables;
- Showing the structure of a table

  mysql> describe [table_name];

SQL commands-CREATE, UPDATE, INSERT, DELETE, SELECT

Create Database

CREATE DATABASE is the SQL command for creating a database.

Example : CREATE DATABASE student;

IF NOT EXISTS

A single MySQL server could have multiple databases. If you are not the only one accessing the same MySQL server or if you have to deal with multiple databases there is a probability of attempting to create a new database with name of an existing database. IF NOT EXISTS let you to instruct MySQL server to check the existence of a database with a similar name prior to creating database.

When IF NOT EXISTS is used database is created only if given name does not conflict with an existing database's name. Without the use of IF NOT EXISTS MySQL throws an error. For example.

```
CREATE DATABASEIF NOT EXISTS student;
```

Selecting database

Creating a database does not select it for use; you must do that explicitly. To make menagerie the current database, use this statement:

```
USE student;
```

Your database needs to be created only once, but you must select it for use each time you begin a mysql session.

You can check that this is the active database by typing in the SELECT DATABASE( ); command:

```
mysql> SELECT DATABASE();
+----------+
| DATABASE() |
+----------+
| student |
+----------+
1 row in set (0.00 sec)
```

Creating Tables MySQL

Tables can be created using CREATE TABLE statement and it actually has the following syntax.

```
CREATE TABLE [IF NOT EXISTS] 'TableName' ('fieldname' dataType [optional parameters])
```

- "[IF NOT EXISTS]" is optional and only create the table if no matching table name is found.
- "[optional parameters]" additional information about a field such as "AUTO_INCREMENT", NOT NULL etc

```
CREATE TABLE IF NOT EXISTS student ( 'roll_number' INT AUTO_INCREMENT , 'full_name' VARCHAR(150) NOT NULL ,'gender' VARCHAR(6) , 'date_of_birth' DATE ,'address' VARCHAR(255) , 'contact_number' VARCHAR(75) , 'email' VARCHAR(255) , PRIMARY KEY ('roll_number') );
```

Or you can use the following general format without using 'if not exists'.

```
CREATE TABLE student ( 'roll_number' INT AUTO_INCREMENT , 'full_name' VARCHAR(150) NOT NULL , 'gender' VARCHAR(6) , 'date_of_birth' DATE , 'address' VARCHAR(255) , 'contact_number' VARCHAR(75) , 'email' VARCHAR(255) , PRIMARY KEY ('roll_number') );
```

Loading Data into a Table

After creating your table, you need to populate it. INSERT statements is used for this.You could add a new record using an INSERT statement like this:

```
INSERT INTO student VALUES ('101','Smith R','male','1999-03-30','house no:56, royal lane',7568412310,NULL);
```

String and date values are specified as quoted strings here. Also, with INSERT, you can insert NULLdirectly to represent a missing value. Following example show how to insert multiple rows in a single insert statement.

```
insert into 'student'(roll_number)VALUES(11),(12),(13)
```

Note that MySQL expects dates in 'YYYY-MM-DD' format.

SELECT Statement and Basic Querying Techniques

SELECT is used to retrieve rows selected from one or more tables. The most basic form of SELECT reads the data in all rows and columns from a table.

```
SELECT * FROM student;
```

A simple SELECT statement has four components:

1. The keyword SELECT.
2. The columns to be displayed. In our first example, we asked for all columns by using the asterisk (*) symbol as a wildcard character.
3. The keyword FROM.
4. The table name; in this example, the table name is student.

Choosing Columns

You've so far used the * wildcard character to retrieve all columns in a table. If you don't want to display all the columns, it's easy to be more specific by listing the columns you want, in the order you want them, separated by commas. For example, if you want only the full_name column from the student table, you'd type:

```
mysql> SELECT full_name FROM student;
```

If you want both the artist_name and the artist_id, in that order, you'd use:

```
mysql> SELECT full_name, roll_number FROM artist;
```

Deleting a Table

To delete an entire table, use the DROP TABLE command:

```
Drop table student;
```

The DELETE statement is used to remove one or more rows from a database.Once a row has been deleted, it cannot be recovered. The simplest use of DELETE is to remove all rows in a table.

```
DELETE from student
```

If the WHERE clause is not used in the DELETE query, then all the rows in a given table will be deleted.To remove one or more rows, but not

all rows in a table, you use a WHERE clause. This works in the same way as it does for SELECT.

```
DELETE FROM 'student' WHERE roll_number=2
```

Note that *you cannot delete a single column for a table. You can delete an entire row.*

If you want to remove all rows in a table, there's a faster method than removing them with DELETE. By using the TRUNCATE TABLE statement, MySQL takes the shortcut of dropping the table—that is, removing the table structures and then re-creating them. When there are many rows in a table, this is much faster.If you want to remove the data in the played table, you can write this:

```
TRUNCATE TABLE student;
```

UPDATE Statement

The Update command is used to modify rows in a table. The update command can be used to update a single field or multiple fields at the same time. It can also be used to update a table with values from another table. The basic syntax of the SQL Update command is as shown below.

```
UPDATE 'table_name' SET 'column_name' = 'new_value' [WHERE condition];
```

For example:

```
UPDATE 'student' SET 'contact_number'=123456978 WHERE roll_number=1
```

The "WHERE clause" is used to limit the number of rows affected by the UPDATE query.

PHP functions for MySQL connectivity and operation

It is mainly used to developpowerful applications including online shopping, news, sports, and blogs. Indeed, theApache web server, MySQL, and PHP together form three of the four components ofthe most popular of all web development platforms, LAMP. The "L" stands for Linux.The LAMP acronym is increasingly interpreted rather loosely as representing any opensource development platform for web database applications.

These are the four steps to access the database using PHP:

1. Connect to MySQL, using the mysql_connect( ) MySQL library function.
2. Select the database, using the mysql_select_db( ) MySQL library function.
3. Run the SQL query, using the mysql_query( ) MySQL library function.
4. Retrieve and display the data, using a while loop, the mysql_fetch_array( ), mysql_fetch_row(), etc MySQL library function, the print statement, and a loop.

Following are the different functions that are used to interact with MySQL using php.

✓ **mysql_connect()**

Connection to Mysql database can be established by using mysql_connect() function. We can check the success of the function by checking the result. We will get a true result in case connection is established. The syntax of mysql_connect() is

```
mysql_connect(server,user,passwd);
```

This function takes three parameters, first one is hostname then user-id and them password. We can give the port number along with the hostname also.

- **Server:** Optional " The host name running the database server. If not specified, then the default value will be **localhost:3306**.
- **User:** Optional " The username accessing the database. If not specified, then the default will be the name of the user that owns the server process.
- **Passwd:** Optional " The password of the user accessing the database. If not specified, then the default will be an empty password.

The above function will return true or false depending on the success of the connection. So we will add message to the above function like this.

```
if (!($connection = mysql_connect("localhost", "root", "the_mysql_
root_password")))
die("Cannot connect");
```

The function mysql_connect( ) opens a connection, and the three parameters to the function—"localhost", "root", and "*the_mysql_root_password*"—are the hostname of the server, the MySQL user, and the user's password, respectively.

The function does the same thing as running the MySQL monitor: it authenticates you, giving you access to the MySQL server so that you can run SQL statements. The other important feature is that a connection *resource handle* is returned and saved in a PHP variable $connection; this is used in the following two steps. The at symbol (@) tells PHP not to display its own error messages. If we discover a critical error, we call the die( ) function to display an error message and stop processing the script.

Another way of setting database connection with PHP is :

```
$server=localhost;
$user=root;
$password=root;
$con=mysql_connect ("$server","$user","$password");

if(!$con){die("Could not connect to MySQL");}
```

Mysql_connect once establish the connection the link will be present till the script execution is over. It will close it self once the script execution is over or the function mysql_close() is called.

✓ **mysql_select_db() function**

mysql_select_db It is used for Selecting a MySQL database

Syntax:

```
bool mysql_select_db ( string $database_name [, resource $link_identifier = NULL ] )
```

Sets the current active database on the server that's associated with the specified link identifier. Every subsequent call to mysql_query() will be made on the active database.

Parameters	Description
database_name	The name of the database that is to be selected.
link_identifier	If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() had been called with no arguments. If no connection is found or established, an E_WARNING level error is generated.
Return Values	Returns TRUE on success or FALSE on failure.

We can create a connection string and database connections at one go. This is required where all over the script we are using one database so better to select the database just after the connection is established. Here is the code.

```
$servername='localhost';
$dbuser='userid';
$dbpassword='password';
$dbname='db_name';
        $con=mysql_connect("$servername","$dbuser","$dbpassword");
        if(!$con){
        die("Could not connect to MySQL");
        }
        mysql_select_db("$dbname",$con) or die ("could not open
db".mysql_error());
```

✓ **mysql_close() function**

To close a mysql connection we can use mysql_close() function. It can takes a optional parameter as link and closes it. If no link identifier is specified then last opened connection is closed. It is not necessary to use mysql_close() function as all connections are closed at the end of the script execution. Here is the function

```
mysql_close()
```

✓ **mysql_query()**

The mysql_query() function executes a query on a MySQL database. This function returns the query handle for SELECT queries, TRUE/FALSE for other queries, or FALSE on failure. Note that this function fetches and buffers the recordset automatically.

Syntax :

mysql_query (string $query [, resource $link_identifier = NULL])

mysql_query() sends a unique query (multiple queries are not supported) to the currently active database on the server that's associated with the specified link_identifier.

Parameters

- query

  An SQL query

  The query string should not end with a semicolon. Data inside the query should be properly escaped(if the data in text use quotation marks.).
- link_identifier

If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() had been called with no arguments. If no connection is found or established, an E_WARNING level error is generated.

Return Values

For SELECT, SHOW, DESCRIBE, EXPLAIN and other statements returning resultset, **mysql_query()** returns a resourceon success(ie. TRUE), or FALSE on error.

The returned result resource should be passed to mysql_fetch_array(), and other functions for dealing with result tables, to access the returned data.

Use mysql_num_rows() to find out how many rows were returned for a SELECT statement ormysql_affected_rows() to find out how many rows were affected by a DELETE, INSERT, REPLACE, or UPDATE statement.

mysql_query() will also fail and return FALSE if the user does not have permission to access the table(s) referenced by the query.

Examples

Example 1 Invalid Query

The following query is syntactically invalid, so **mysql_query()** fails and returns FALSE.

```
<?php
$result = mysql_query('SELECT * WHERE 1=1');
if (!$result) {
   die('Invalid query: ' . mysql_error());
}
?>
```

Following example shows a valid query with mysql_query().

```
<?php
$query = "SELECT  * FROM student";
$result = mysql_query($query);
if (!$result) {
   $message  = 'Invalid query: ' . mysql_error() . "\n";   die($message);
}
while ($row = mysql_fetch_array ($result)) {
   echo $row['firstname'];   // or echo $row[0];
   echo $row['lastname'];   // echo $row[1];
   echo $row['phno'];// echo $row[2];
   echo $row['age'];              // echo $row[3];
}
?>
```

✓ **mysql_fetch_row()**

The mysql_fetch_row() function returns a row from a recordset as a numeric array. This function gets a row from the mysql_query() function

and returns an array on success, or FALSE on failure or when there are no more rows.

Syntax:

array mysql_fetch_row (resource $result)

Parameters

- result

The result resource that is being evaluated. This result comes from a call to mysql_query().

Note that after the data is retrieved, this function moves to the next row in the recordset.

Example 1 Fetching one row with mysql_fetch_row()

```
<?php
$result = mysql_query("SELECT id,email FROM customer WHERE id = 5");
if (!$result) {
   echo 'Could not run query: ' . mysql_error();
   exit;
}
$row = mysql_fetch_row($result);

echo $row[0]; // 5
echo $row[1]; // the email value
?>
```

✓ **mysql_fetch_array**

The mysql_fetch_array() function returns a row from a recordset as an associative array and/or a numeric array.

This function gets a row from the mysql_query() function and returns an array on success, or FALSE on failure or when there are no more rows.

Syntax

mysql_fetch_array(data,array_type)

Parameter	Description
data	Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function
array_type	Optional. Specifies what kind of array to return. Possible values: • MYSQL_ASSOC - Associative array • MYSQL_NUM - Numeric array • MYSQL_BOTH - Default. Both associative and numeric array

Note that after the data is retrieved, this function moves to the next row in the recordset. Each subsequent call to mysql_fetch_array() returns the next row in the recordset.

Example1 - mysql_fetch_array() with MYSQL_NUM

```
<?php
mysql_connect("localhost", "root", "abc123") or
  die("Could not connect: " . mysql_error());

mysql_select_db("mydb");

$result = mysql_query("SELECT id, name FROM student");

while ($row = mysql_fetch_array($result, MYSQL_NUM)) {
  printf("ID: %s  Name: %s", $row[0], $row[1]);
}

mysql_free_result($result);
?>
```

Example 2 – mysql_fetch_array() with MYSQL_ ASSOC

```
<?php
mysql_connect("localhost", "root", "abc123") or
  die("Could not connect: " . mysql_error());
mysql_select_db("mydb");

$result = mysql_query("SELECT id, name FROM student");
```

```
while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
  printf("ID: %s  Name: %s", $row["id"], $row["name"]);
}
mysql_free_result($result);
?>
```

Example 2 - mysql_fetch_array with MYSQL_ASSOC

```
<?php
mysql_connect("localhost", "root", "abc123") or
  die("Could not connect: " . mysql_error());
mysql_select_db("mydb");
$result = mysql_query("SELECT id, name FROM student");
while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
  printf("ID: %s  Name: %s", $row["id"], $row["name"]);
}
mysql_free_result($result);
?>
```

Example 3 - mysql_fetch_array with MYSQL_BOTH

```
<?php
mysql_connect("localhost", "root", "abc123") or
  die("Could not connect: " . mysql_error());
mysql_select_db("mydb");
$result = mysql_query("SELECT id, name FROM student");
while ($row = mysql_fetch_array($result, MYSQL_BOTH)) {
  printf ("ID: %s  Name: %s", $row[0], $row["name"]);
}
mysql_free_result($result);?>
```

✓ **mysql_result**

The mysql_result() function returns the value of a field in a recordset. This function returns the field value on success, or FALSE on failure.

Syntax

mysql_result(data,row,field)

Parameter	Description
Data	Required. Specifies which result handle to use. The data pointer is the return from the mysql_query() function
Row	Required. Specifies which row number to get. Row numbers start at 0
Field	Optional. Specifies which field to get. Can be field offset, field name or table.fieldname. If this parameter is not defined mysql_result() gets the first field from the specified row

Note that this function is slower than mysql_fetch_row(), mysql_fetch_array(), mysql_fetch_assoc() and mysql_fetch_object().

Example

```
<?php
$con = mysql_connect("localhost", "root", "abc123");
if (!$con)
 {
 die('Could not connect: ' . mysql_error());
 }
$db_selected = mysql_select_db("test_db", $con);
$sql = "SELECT * from student";
$result = mysql_query($sql,$con);

echo mysql_result($result,0);
mysql_close($con);
?>
```

✓ **mysql_list_fields**

mysql_list_fields() is used to list MySQL table fields.

Description

resource **mysql_list_fields** (string $database_name , string $table_name [, resource$link_identifier = NULL])

This function is deprecated. It is preferable to use mysql_query() to issue an SQL *SHOW COLUMNS FROM table [LIKE 'name']* statement instead.

Parameters

- database_name

  The name of the database that's being queried.

- table_name

  The name of the table that's being queried.

- link_identifier

  The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() had been called with no arguments. If no connection is found or established, an E_WARNING level error is generated.

Return Values

A result pointer resource on success, or FALSE on failure.The returned result can be used with mysql_field_flags(), mysql_field_len(), mysql_field_name(), andmysql_field_type().

Example – Alternate to deprecated mysql_list_fields()

```
<?php
$result = mysql_query("SHOW COLUMNS FROM student");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}
if (mysql_num_rows($result) > 0) {
    while ($row = mysql_fetch_assoc($result)) {
        print_r($row);
    }
}
?>
```

✓ **mysql_num_fields**

The mysql_num_fields() function returns the number of fields in a recordset. Or Retrieves the number of fields from a query.

Syntax:

```
int mysql_num_fields ( resource $result )
```

Parameters

- result

The result resource that is being evaluated. This result comes from a call to mysql_query().

Return Values

Returns the number of fields in the result set resource on success or FALSE on failure.

Example:

```
<?php
$con = mysql_connect("localhost", "root", "abc123");
if (!$con)
 {
 die('Could not connect: ' . mysql_error());
 }
$db_selected = mysql_select_db("test_db",$con);
$sql = "SELECT * FROM student";
$result = mysql_query($sql,$con);
echo mysql_num_fields($result);
mysql_close($con);
?>
```

insertion, updation and deletion of data using PHP

MySQL is a database server processing system. With the help of php scripts, we can perform a variety of data processing in the website server with easy, such as creating a database table, delete the database table,

storing a database table, to display the database table in the pages of a website.

For example, it is used for registration application, verification applications, applications on a cms and others. If you create own a website, you will be in desperate need of a MySQL database processing system.

Please ensure that to create a table staff in the MySQL database to practice the following database operations.

```
CREATE TABLE 'staff' ( 'id' int(11) NOT NULL auto_increment, 'name'
varchar(100) NOT NULL, 'email' varchar(100) NOT NULL, 'country'
varchar(25) NOT NULL, PRIMARY KEY ('id')
);
```

Create PHP Connect to the MySQL Database Server

Each summoning the MySQL database, then we have to open a connection to the database server first. Therefore, processing the MySQL database we must first master is on the MySQL database connection. The following php script to call the database server :

```
<?php
//these variables are used for database connection
$host = "localhost";
$user = "root";
$password = "";
$database_name = "staff_mgmt";
//connect to the database.
mysql_connect($host, $user, $password);
mysql_select_db($database_name);
?>
```

Each data processing, call php connection and put it on the database MySQL script processing. Another way is to put the php connection to another file and do the calling php file with the following script :

include("php-connect.php");
or
require("php-connect.php");

- **Create PHP INSERT Statement To Save Data in MySQL Database Table**

To save the data into a MySQL database table, we can use php insert statement. Here's the script:

```
<?php
//load database connection
include("php-connect.php");
$name = "rithika";
$email = "rithika@mail.com";
$country = "india";
//Command to insert into table
$query = "INSERT INTO staff (name,email,country) VALUES
('$name','$email','$country')";
//run the query to insert the person.
$result = mysql_query($query) OR die(mysql_error());
//let them know the person has been added.
echo "Data successfully inserted into the database table ... ";
?>
```

- **Create PHP UPDATE Statement For Editing Data in MySQL Database Table**

To edit the data that has been stored in a MySQL database table, we can use the php update statement. Here's the script :

```
<?php
//load database connection
include("php-connect.php");
$name = "rithika";
$email = " rithika@mail.com";
$country = "England";
```

```
//Make the query to update the table.
$query = "UPDATE staff SET country='$country' WHERE email='$email'";
//run the query to update the person.
$result = mysql_query($query) OR die(mysql_error());
//let them know the person has been added.
echo "Data successfully updated into the database table ... ";
?>
```

- **Create PHP SELECT Statement For Displaying Data in MySQL Database Table**

To display data from a MySQL database table, we can use php select statement. Here's the php script :

```
<?php
//load database connection
include("php-connect.php");
//make the query to run.
//Sort the last name in an ascending order (A-Z)
$query = "SELECT * FROM people ORDER BY name ASC";
$result = mysql_query($query) OR die(mysql_error());
//now we turn the results into an array and loop through them.
while($row = mysql_fetch_array($result))
{
$name = $row['name'];
$email = $row['email'];
$country = $row['country'];
echo "Data successfully selected in the database table ... ";
echo "<br/>name :".$name;
echo "<br/>email :".$email;
echo "<br/>country :".$country;
}
?>
```

- **Create PHP DELETE Statement To Delete Data in MySQL Database Table**

To delete data from a MySQL database table, we can use php delete statement. Here's the php script :

```
<?php
//load database connection
include("php-connect.php");

//get the userid of the person we're deleting.
$email = "elisa@mail.com";

//write the query to delete the person.
$query = "DELETE FROM people WHERE email='$email'";

//run the query to delete the person.
$result = mysql_query($query) OR die(mysql_error());

echo "Data successfully deleted in the database table ... ";
?>
```

Displaying data from MySQL in webpage

Here we'll see how to display the contents of mysql table in your php webpage.You can use the phpmyadmin or sql console from the web server to perform database operations with your webpage.

Do the following steps:

- Establish connection with database.
- Create an object (or reference) that holds the contents of database.
- Perform action on database (like reading or writing).
- Close the connection or Display the results.

So now we've a goal to display the contents of the database with the help of php and mysql. Let's first start with the database creation.

You can use phpmyadmin to create a database and populate the table with entries. I have created a database called *contacts* and inside it there is table called *bcard*. There are three fields – No, Name, Surname. We're going to make "No" a primary key.

Now let's write PHP code to fetch the content of our database "contacts".

```
< ?php
// Connects to your Database
mysql_connect("localhost", "root", "root") or die(mysql_error());
mysql_select_db("contacts") or die(mysql_error());

$data = mysql_query("SELECT * FROM bcard") or die(mysql_error());
 while($row = mysql_fetch_assoc($data)){
  echo "No: ".$row['No']."., Name:".$row['Name']
  ."., Surname:".$row['Surname']."<br/>";
  }
? >
```

First we're connecting to the mysql server and for that we're adding the credentials like – "name of the server" : localhost and username and password. My credentials and your web servers credentials will vary, so you have to check with xampp or uniform server on your desktop. In second part we're connecting to the table if the previous connection to the database is successful. If there is any error in these two steps, mysql error will be thrown on webpage.

In third step, we're creating an object or reference that fetches the content of our table 'bcard'. We're now placing the cursor on each row and printing out the content of database. Based on the content inside the table, you'll see the result in output. Make sure you don't make mistakes here or else there will be mysql error thrown for your invalid syntax.

REVIEW QUESTIONS

Part A

1. What are the features of MySQL?
2. How will you delete a table in MySQL?
3. Write down the syntax for creating tables in mysql.
4. Explain the syntax of mysql_fetch_array().

Part B

1. Explain different data types in MySQL.
2. Explain the different functions used for MySQL connectivity.
3. How will you display data from mysql database in a webpage?
4. Explain mysql_connect() function.

Part C

1. Explain the different steps to access MySQL database.
2. Create a login page using PHP as front end and MySQL as backend.
3. Explain the following sql commands:

   *create *upade * insert * delete * select

B.C.A DEGREE (C.B.C.S.S) EXAMINATION, MODEL QUESTION PAPER

IV Semester

Core Course – **WEB PROGRAMING USING PHP**

Time: Three hours Maximum: 80 marks

Part A

*Answer any **10**, Each question carries **2** marks.*

1. Define HTML.
2. What are links? How to insert hyperlinks in HTML?
3. What is style Sheets?
4. What are events in JavaScript?
5. How to create user defined objects in JavaScript?
6. What is the difference between GET and POST methods of form tag?
7. What is webpage?
8. What is a database?
9. How to insert comments in PHP?
10. Define echo.
11. What are the features of PHP?
12. Explain the function in PHP used to connect MySQL database. (10x2=20)

Part B

*Answer any **6**, Each question carries **5** marks.*

13. What is marquee? Explain its attributes with example.
14. Explain include( ) and require( ).
15. Write an HTML program for email registration form.
16. Explain frames in HTML with the help of example.
17. Explain different methods to insert CSS in webpage.
18. Explain popup boxes in JavaScript.
19. Write a JavaScript program to validate username and password.
20. Explain string functions in PHP.
21. Explain the different data types in PHP. (6x5=30)

Part C

*Answer any **2**, Each question carries **15** marks.*

22. Explain lists and form elements in HTML.
23. Explain decision control statements and loops in JavaScript.
24. Explain a. Session and Cookie management
    b. Error handling in PHP
25. Explain any five functions for MySQL database operations. (2x15=30)