MySQL

MySQL – PHP Syntax

The PHP functions for use with MySQL have the following general format:

```
mysql_function(value,value,...);
```

The second part of the function name is specific to the function, usually a word that describes what the function does. The following are two of the functions, which we will use in our tutorial:

```
mysqli_connect($connect);
mysqli_query($connect,"SQL statement");
```

The following example shows a generic syntax of PHP to call any MySQL function.

```
<html>
<head>
<title>PHP with MySQL</title>
</head>
<body>
<?php
   $retval = mysql_function(value, [value,...]);
   if( !$retval )
   {
        die ( "Error: a related error message" );
   }
   // Otherwise MySQL  or PHP Statements
?>
</body>
</html>
```

MySQL – PHP Syntax

MySQL Connection Using MySQL Binary

You can establish the MySQL database using the **mysql** binary at the command prompt.

Example

Here is a simple example to connect to the MySQL server from the command prompt –

```
[root@host]# mysql -u root -p
Enter password:******
```

This will give you the mysql> command prompt, where you will be able to execute any SQL command.

The following code block shows the result of above command:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2854760 to server version: 5.0.9

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

In the above example, we have used **root** as a user, but you can use any other user as well. Any user will be able to perform all the SQL operations, which are allowed to that user.

You can disconnect from the MySQL database anytime using the **exit** command at mysql> prompt.

```
mysql> exit
Bye
```

MySQL – PHP Syntax

MySQL Connection Using PHP Script

PHP provides **mysql_connect()** function to open a database connection. This function takes five parameters and returns a MySQL link identifier on success or FALSE on failure.

Syntax:

```
connection mysql_connect(server,user,passwd,new_link,client_flag);
```

Parameter	Description
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.
new_link	Optional – If a second call is made to mysql_connect() with the same arguments, no new connection will be established; instead, the identifier of the already opened connection will be returned.
client_flags	Optional – A combination of the following constants: • MYSQL_CLIENT_SSL – Use SSL encryption. • MYSQL_CLIENT_COMPRESS – Use compression protocol. • MYSQL_CLIENT_IGNORE_SPACE – Allow space after function names. • MYSQL_CLIENT_INTERACTIVE – Allow interactive timeout seconds of inactivity before closing the connection.

MySQL – PHP Syntax

You can disconnect from the MySQL database anytime using another PHP function **mysql_close()**. This function takes a single parameter, which is a connection returned by the **mysql_connect()** function.

Syntax:

```
bool mysql_close ( resource $link_identifier );
```

If a resource is not specified, then the last opened database is closed. This function returns true if it closes the connection successfully otherwise it returns false.

MySQL – PHP Syntax

Example

Try the following example to connect to a MySQL server:

```
<html>
<head>
<title>Connecting MySQL Server</title>
</head>
<body>
<?php
   $dbhost = 'localhost:3306';
   $dbuser = 'guest';
   $dbpass = 'guest123';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);
   if(! $conn )
   {
     die('Could not connect: ' . mysql_error());
   }
   echo 'Connected successfully';
   mysql_close($conn);
?>
</body>
</html>
```

MySQL – Create Database

Create Database Using mysqladmin

You would need special privileges to create or to delete a MySQL database. So assuming you have access to the root user, you can create any database using the mysql **mysqladmin** binary.

Example

Here is a simple example to create a database called **TUTORIALS** –

```
[root@host]# mysqladmin -u root -p create TUTORIALS

Enter password:******
```

This will create a MySQL database called TUTORIALS.

MySQL – Create Database

Create a Database Using PHP Script

PHP uses **mysql_query** function to create or delete a MySQL database. This function takes two parameters and returns TRUE on success or FALSE on failure.

Syntax:

```
bool mysql_query( sql, connection );
```

Parameter	Description
sql	Required – SQL query to create or delete a MySQL database.
connection	Optional – if not specified, then the last opened connection by mysql_connect will be used.

MySQL – Create Database

Try out the following example to create a database:

```
<html>
<head>
<title>Creating MySQL Database</title>
</head>
<body>
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully<br />';
$sql = 'CREATE DATABASE TUTORIALS';
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not create database: ' . mysql_error());
}
echo "Database TUTORIALS created successfully\n";
mysql_close($conn);
?>
</body>
</html>
```

MySQL – Drop Database

Drop a Database using mysqladmin

You would need special privileges to create or to delete a MySQL database. So, assuming you have access to the root user, you can create any database using the mysql **mysqladmin** binary.

Be careful while deleting any database because you will lose your all the data available in your database.

Here is an example to delete a database (TUTORIALS) created in the previous chapter:

```
[root@host]# mysqladmin -u root -p drop TUTORIALS
Enter password:******
```

This will give you a warning and it will confirm if you really want to delete this database or not.

```
Dropping the database is potentially a very bad thing to do.
Any data stored in the database will be destroyed.


Do you really want to drop the 'TUTORIALS' database [y/N] y
Database "TUTORIALS" dropped
```

MySQL – Drop Database

Drop Database using PHP Script

PHP uses **mysql_query** function to create or delete a MySQL database. This function takes two parameters and returns TRUE on success or FALSE on failure.

Syntax:

```
bool mysql_query( sql, connection );
```

Parameter	Description
sql	Required – SQL query to create or delete a MySQL database.
connection	Optional – if not specified, then the last opened connection by mysql_connect will be used.

MySQL – Drop Database

Example

Try the following example to delete a database:

```
<html>
<head>
<title>Deleting MySQL Database</title>
</head>
<body>
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully<br />';
$sql = 'DROP DATABASE TUTORIALS';
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not delete database: ' . mysql_error());
}
echo "Database TUTORIALS deleted successfully\n";
mysql_close($conn);
?>
</body>
</html>
```

MySQL – Select Database

Selecting MySQL Database from the Command Prompt

It is very simple to select a database from the mysql> prompt. You can use the SQL command **use** to select a database.

Example

Here is an example to select a database called **TUTORIALS**:

```
[root@host]# mysql -u root -p

Enter password:******

mysql> use TUTORIALS;

Database changed

mysql>
```

Now, you have selected the TUTORIALS database and all the subsequent operations will be performed on the TUTORIALS database.

NOTE: All the database names, table names, table fields name are case sensitive. So you would have to use the proper names while giving any SQL command.

MySQL – Select Database

Selecting a MySQL Database Using PHP Script

PHP provides function **mysql_select_db** to select a database. It returns TRUE on success or FALSE on failure.

Syntax:

```
bool mysql_select_db( db_name, connection );
```

Parameter	Description
db_name	Required – MySQL Database name to be selected.
connection	Optional – if not specified, then the last opened connection by mysql_connect will be used.

MySQL – Select Database

Example

Here is an example showing you how to select a database.

```
<html>
<head>
<title>Selecting MySQL Database</title>
</head>
<body>
<?php
$dbhost = 'localhost:3036';
$dbuser = 'guest';
$dbpass = 'guest123';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_select_db( 'TUTORIALS' );
mysql_close($conn);
?>
</body>
</html>
```

MySQL – Datatypes

MySQL uses all the standard ANSI SQL numeric data types, so if you're coming to MySQL from a different database system, these definitions will look familiar to you.

The following list shows the common numeric data types and their descriptions:

- **INT** – A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
- **TINYINT** – A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
- **SMALLINT** – A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
- **MEDIUMINT** – A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
- **BIGINT** – A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.
- **FLOAT(M,D)** – A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.
- **DOUBLE(M,D)** – A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.
- **DECIMAL(M,D)** – An unpacked floating-point number that cannot be unsigned. In the unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

MySQL – Datatypes

Date and Time Types

The MySQL date and time datatypes are as follows:

- **DATE** – A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th 1973 would be stored as 1973-12-30.
- **DATETIME** – A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th 1973 would be stored as 1973-12-30 15:30:00.
- **TIMESTAMP** – A timestamp between midnight, January 1st 1970 and sometime in 2037. This looks like the previous DATETIME format only, but without the hyphens between numbers; 3:30 in the afternoon on December 30th 1973 would be stored as 19731230153000 (YYYYMMDDHHMMSS).
- **TIME** – Stores the time in a HH:MM:SS format.
- **YEAR(M)** – Stores a year in a 2-digit or a 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be between 1970 to 2069 (70 to 69). If the length is specified as 4, then YEAR can be 1901 to 2155. The default length is 4.

MySQL – Datatypes

String Types

Although the numeric and date types are fun, most data you'll store will be in a string format. This list describes the common string datatypes in MySQL.

- **CHAR(M)** – A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
- **VARCHAR(M)** – A variable-length string between 1 and 255 characters in length. For example, VARCHAR(25). You must define a length when creating a VARCHAR field.
- **BLOB or TEXT** – A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data. The difference between the two is that the sorts and comparisons on the stored data are **case sensitive** on BLOBs and are **not case sensitive** in TEXT fields. You do not specify a length with BLOB or TEXT.
- **TINYBLOB or TINYTEXT** – A BLOB or TEXT column with a maximum length of 255 characters. You do not specify a length with TINYBLOB or TINYTEXT.
- **MEDIUMBLOB or MEDIUMTEXT** – A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.
- **LONGBLOB or LONGTEXT** – A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.
- **ENUM** – An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL).

  For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

MySQL – Create Tables

Syntax: Here is a generic SQL syntax to create a MySQL table:

```
CREATE TABLE table_name (column_name column_type);
```

Now, we will create the following table in the **TUTORIALS** database.

```
tutorials_tbl(
   tutorial_id INT NOT NULL AUTO_INCREMENT,
   tutorial_title VARCHAR(100) NOT NULL,
   tutorial_author VARCHAR(40) NOT NULL,
   submission_date DATE,
   PRIMARY KEY ( tutorial_id )
);
```

Here, a few items need explanation:

- Field Attribute **NOT NULL** is being used because we do not want this field to be NULL. So, if a user will try to create a record with a NULL value, then MySQL will raise an error.
- Field Attribute **AUTO_INCREMENT** tells MySQL to go ahead and add the next available number to the id field.
- Keyword **PRIMARY KEY** is used to define a column as a primary key. You can use multiple columns separated by a comma to define a primary key.

MySQL – Create Tables

Creating Tables from Command Prompt

It is easy to create a MySQL table from the mysql> prompt. You will use the SQL command **CREATE TABLE** to create a table.

Example

Here is an example, which will create **tutorials_tbl**:

```
root@host# mysql -u root -p
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> CREATE TABLE tutorials_tbl(
   -> tutorial_id INT NOT NULL AUTO_INCREMENT,
   -> tutorial_title VARCHAR(100) NOT NULL,
   -> tutorial_author VARCHAR(40) NOT NULL,
   -> submission_date DATE,
   -> PRIMARY KEY ( tutorial_id )
   -> );
Query OK, 0 rows affected (0.16 sec)
mysql>
```

MySQL – Create Tables

Creating Tables Using PHP Script

To create new table in any existing database you would need to use PHP function **mysql_query()**. You will pass its second argument with a proper SQL command to create a table.

Example

The following program is an example to create a table using PHP script:

```
<html>
<head>
<title>Creating MySQL Tables</title>
</head>
<body>
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully<br />';
$sql = "CREATE TABLE tutorials_tbl( ".
       "tutorial_id INT NOT NULL AUTO_INCREMENT, ".
       "tutorial_title VARCHAR(100) NOT NULL, ".
       "tutorial_author VARCHAR(40) NOT NULL, ".
       "submission_date DATE, ".
       "PRIMARY KEY ( tutorial_id )); ";
mysql_select_db( 'TUTORIALS' );
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not create table: ' . mysql_error());
}
echo "Table created successfully\n";
mysql_close($conn);
?>
</body>
</html>
```

MySQL – Drop Tables

Syntax: Here is a generic SQL syntax to drop a MySQL Table:

```
DROP TABLE table_name ;
```

Dropping Tables from the Command Prompt

To drop tables from the command prompt, we need to execute the **DROP TABLE** SQL command at the mysql> prompt.

Example

The following program is an example which deletes the **tutorials_tbl**:

```
root@host# mysql -u root -p
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> DROP TABLE tutorials_tbl
Query OK, 0 rows affected (0.8 sec)
mysql>
```

MySQL – Drop Tables

Dropping Tables Using PHP Script

To drop an existing table in any database, you would need to use the PHP function **mysql_query()**. You will pass its second argument with a proper SQL command to drop a table.

Example

```
<html>
<head>
<title>Creating MySQL Tables</title>
</head>
<body>
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully<br />';
$sql = "DROP TABLE tutorials_tbl";
mysql_select_db( 'TUTORIALS' );
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not delete table: ' . mysql_error());
}
echo "Table deleted successfully\n";
mysql_close($conn);
?>
</body>
</html>
```

MySQL – Insert Query

To insert data into a MySQL table, you would need to use the SQL **INSERT INTO** command. You can insert data into the MySQL table by using the mysql> prompt or by using any script like PHP.

Syntax: Here is a generic SQL syntax of INSERT INTO command to insert data into the MySQL table.

```
INSERT INTO table_name ( field1, field2,...fieldN )
                       VALUES
                       ( value1, value2,...valueN );
```

To insert string data types, it is required to keep all the values into double or single quotes. For example – **"value"**.

Inserting Data from the Command Prompt

To insert data from the command prompt, we will use SQL INSERT INTO command to insert data into MySQL table tutorials_tbl.

MySQL – Insert Query

Example

The following example will create 3 records into **tutorials_tbl** table:

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> INSERT INTO tutorials_tbl
     ->(tutorial_title, tutorial_author, submission_date)
     ->VALUES
     ->("Learn PHP", "John Poul", NOW());
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO tutorials_tbl
     ->(tutorial_title, tutorial_author, submission_date)
     ->VALUES
     ->("Learn MySQL", "Abdul S", NOW());
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO tutorials_tbl
     ->(tutorial_title, tutorial_author, submission_date)
     ->VALUES
     ->("JAVA Tutorial", "Sanjay", '2007-05-06');
Query OK, 1 row affected (0.01 sec)
mysql>
```

Inserting Data Using a PHP Script

You can use the same SQL INSERT INTO command into the PHP function **mysql_query()** to insert data into a MySQL table.

Example

This example will take three parameters from the user and will insert them into the MySQL table:

```
<html>
<head>
<title>Add New Record in MySQL Database</title>
</head>
<body>
<?php
if(isset($_POST['add']))
{
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
if(! get_magic_quotes_gpc() )
{
   $tutorial_title = addslashes ($_POST['tutorial_title']);
   $tutorial_author = addslashes ($_POST['tutorial_author']);
}
else
{
   $tutorial_title = $_POST['tutorial_title'];
   $tutorial_author = $_POST['tutorial_author'];
}
$submission_date = $_POST['submission_date'];

$sql = "INSERT INTO tutorials_tbl ".
       "(tutorial_title,tutorial_author, submission_date) ".
       "VALUES ".
       "('$tutorial_title','$tutorial_author','$submission_date')";
mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not enter data: ' . mysql_error());
}
echo "Entered data successfully\n";
mysql_close($conn);
}
else
{
?>
<form method="post" action="<?php $_PHP_SELF ?>">
<table width="600" border="0" cellspacing="1" cellpadding="2">
<tr>
<td width="250">Tutorial Title</td>
<td>
<input name="tutorial_title" type="text" id="tutorial_title">
</td>
</tr>
<tr>
<td width="250">Tutorial Author</td>
<td>
<input name="tutorial_author" type="text" id="tutorial_author">
</td>
</tr>
<tr>
<td width="250">Submission Date [ yyyy-mm-dd ]</td>
<td>
<input name="submission_date" type="text" id="submission_date">
</td>
</tr>
<tr>
<td width="250"> </td>
<td> </td>
</tr>
<tr>
<td width="250"> </td>
<td>
<input name="add" type="submit" id="add" value="Add Tutorial">
</td>
</tr>
</table>
</form>
<?php
}
?>
</body>
</html>
```

MySQL – Select Query

Syntax: Here is a generic SQL syntax of the SELECT command to fetch data from the MySQL table:

```
SELECT field1, field2,...fieldN table_name1, table_name2...
[WHERE Clause]
[OFFSET M ][LIMIT N]
```

- You can use one or more tables separated by comma to include various conditions using a WHERE clause, but the WHERE clause is an optional part of the SELECT command.
- You can fetch one or more fields in a single SELECT command.
- You can specify star (*) in place of fields. In this case, SELECT will return all the fields.
- You can specify any condition using the WHERE clause.
- You can specify an offset using **OFFSET** from where SELECT will start returning records. By default, the offset starts at zero.
- You can limit the number of returns using the **LIMIT** attribute.

MySQL – Select Query

Fetching Data from a Command Prompt

This will use SQL SELECT command to fetch data from the MySQL table – **tutorials_tbl**.

Example

The following example will return all the records from the **tutorials_tbl** table:

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl
+-------------+----------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+----------------+-----------------+-----------------+
|           1 | Learn PHP      | John Poul       | 2007-05-21      |
|           2 | Learn MySQL    | Abdul S         | 2007-05-21      |
|           3 | JAVA Tutorial  | Sanjay          | 2007-05-21      |
+-------------+----------------+-----------------+-----------------+
3 rows in set (0.01 sec)

mysql>
```

MySQL – Select Query

Fetching Data Using a PHP Script

You can use the same SQL SELECT command into a PHP function **mysql_query()**. This function is used to execute the SQL command and then later another PHP function – **mysql_fetch_array()** can be used to fetch all the selected data. This function returns the row as an associative array, a numeric array or both. This function returns FALSE if there are no more rows.

The following program is a simple example which will show how to fetch / display records from the **tutorials_tbl** table.

Example

The following code block will display all the records from the tutorials_tbl table.

The content of the rows is assigned to the variable $row and the values in that row are then printed.

NOTE: Always remember to put curly brackets when you want to insert an array value directly into a string.

In the above example, the constant **MYSQL_ASSOC** is used as the second argument to the PHP function **mysql_fetch_array()**, so that it returns the row as an associative array. With an associative array you can access the field by using their name instead of using the index.

MySQL – Select Query

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
$sql = 'SELECT tutorial_id, tutorial_title,
              tutorial_author, submission_date
        FROM tutorials_tbl';
mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_ASSOC))
{
    echo "Tutorial ID :{$row['tutorial_id']}  <br> ".
         "Title: {$row['tutorial_title']} <br> ".
         "Author: {$row['tutorial_author']} <br> ".
         "Submission Date : {$row['submission_date']} <br> ".
         "--------------------------------<br>";
}
echo "Fetched data successfully\n";
mysql_close($conn);
?>
```

MySQL – Select Query

PHP provides another function called **mysql_fetch_assoc()**, which also returns the row as an associative array.

Example

Try out the following example to display all the records from the tutorial_tbl table using mysql_fetch_assoc() function.

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
$sql = 'SELECT tutorial_id, tutorial_title,
              tutorial_author, submission_date
        FROM tutorials_tbl';
mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_assoc($retval))
{
    echo "Tutorial ID :{$row['tutorial_id']}  <br> ".
         "Title: {$row['tutorial_title']} <br> ".
         "Author: {$row['tutorial_author']} <br> ".
         "Submission Date : {$row['submission_date']} <br> ".
         "--------------------------------<br>";
}
echo "Fetched data successfully\n";
mysql_close($conn);
?>
```

MySQL – Select Query

You can also use the constant **MYSQL_NUM** as the second argument to the PHP function mysql_fetch_array(). This will cause the function to return an array with the numeric index.

Example

Try out the following example to display all the records from tutorials_tbl table using the MYSQL_NUM argument.

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
$sql = 'SELECT tutorial_id, tutorial_title,
                tutorial_author, submission_date
        FROM tutorials_tbl';
mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_NUM))
{
    echo "Tutorial ID :{$row[0]}  <br> ".
         "Title: {$row[1]} <br> ".
         "Author: {$row[2]} <br> ".
         "Submission Date : {$row[3]} <br> ".
         "--------------------------------<br>";
}
echo "Fetched data successfully\n";
mysql_close($conn);
?>
```

MySQL – Select Query

Releasing Memory

It is a good practice to release cursor memory at the end of each SELECT statement. This can be done by using the PHP function **mysql_free_result()**. The following program is the example to show how it should be used.

Example

Try out the following example:

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
$sql = 'SELECT tutorial_id, tutorial_title,
                tutorial author, submission date
        FROM tutorials_tbl';
mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_NUM))
{
    echo "Tutorial ID :{$row[0]}  <br> ".
         "Title: {$row[1]} <br> ".
         "Author: {$row[2]} <br> ".
         "Submission Date : {$row[3]} <br> ".
         "--------------------------------<br>";
}
mysql_free_result($retval);
echo "Fetched data successfully\n";
mysql_close($conn);
?>
```

MySQL – Where Clause

We have seen the SQL **SELECT** command to fetch data from a MySQL table. We can use a conditional clause called the **WHERE Clause** to filter out the results. Using this WHERE clause, we can specify a selection criteria to select the required records from a table.

Syntax: The following code block has a generic SQL syntax of the SELECT command with the WHERE clause to fetch data from the MySQL table:

```
SELECT field1, field2,...fieldN table_name1, table_name2...
[WHERE condition1 [AND [OR]] condition2.....
```

- You can use one or more tables separated by a comma to include various conditions using a WHERE clause, but the WHERE clause is an optional part of the SELECT command.
- You can specify any condition using the WHERE clause.
- You can specify more than one condition using the **AND** or the **OR** operators.
- A WHERE clause can be used along with DELETE or UPDATE SQL command also to specify a condition.

The **WHERE** clause works like an **if condition** in any programming language. This clause is used to compare the given value with the field value available in a MySQL table. If the given value from outside is equal to the available field value in the MySQL table, then it returns that row.

MySQL – Where Clause

Here is the list of operators, which can be used with the **WHERE** clause.

Assume field A holds 10 and field B holds 20, then:

Operator	Description	Example
=	Checks if the values of the two operands are equal or not, if yes, then the condition becomes true.	(A = B) is not true.
!=	Checks if the values of the two operands are equal or not, if the values are not equal then the condition becomes true.	(A != B) is true.
>	Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true.	(A > B) is not true.
<	Checks if the value of the left operand is less than the value of the right operand, if yes then the condition becomes true.	(A < B) is true.
>=	Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true.	(A >= B) is not true.
<=	Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true.	(A <= B) is true.

MySQL – Where Clause

Fetching Data from the Command Prompt

This will use the SQL SELECT command with the WHERE clause to fetch the selected data from the MySQL table – **tutorials_tbl**.

Example

The following example will return all the records from the **tutorials_tbl** table for which the author name is **Sanjay**.

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl WHERE tutorial_author='Sanjay';
+-------------+----------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+----------------+-----------------+-----------------+
|           3 | JAVA Tutorial  | Sanjay          | 2007-05-21      |
+-------------+----------------+-----------------+-----------------+
1 rows in set (0.01 sec)

mysql>
```

MySQL – Where Clause

Unless performing a **LIKE** comparison on a string, the comparison is not case sensitive. You can make your search case sensitive by using the **BINARY** keyword as follows:

```
root@host# mysql -u root -p password;

Enter password:*******

mysql> use TUTORIALS;

Database changed

mysql> SELECT * from tutorials_tbl \

          WHERE BINARY tutorial_author='sanjay';

Empty set (0.02 sec)


mysql>
```

MySQL – Where Clause

Fetching Data Using a PHP Script

You can use the same SQL SELECT command with the WHERE CLAUSE into the PHP function **mysql_query()**. This function is used to execute the SQL command and later another PHP function **mysql_fetch_array()** can be used to fetch all the selected data. This function returns a row as an associative array, a numeric array, or both. This function returns FALSE if there are no more rows.

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
$sql = 'SELECT tutorial_id, tutorial_title,
                tutorial_author, submission_date
        FROM tutorials_tbl
        WHERE tutorial_author="Sanjay"';
mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_ASSOC))
{
    echo "Tutorial ID :{$row['tutorial_id']}  <br> ".
         "Title: {$row['tutorial_title']} <br> ".
         "Author: {$row['tutorial_author']} <br> ".
         "Submission Date : {$row['submission_date']} <br> ".
         "--------------------------------<br>";
}
echo "Fetched data successfully\n";
mysql_close($conn);
?>
```

MySQL – UPDATE Query

There may be a requirement where the existing data in a MySQL table needs to be modified. You can do so by using the SQL **UPDATE** command. This will modify any field value of any MySQL table.

Syntax: The following code block has a generic SQL syntax of the UPDATE command to modify the data in the MySQL table:

```
UPDATE table_name SET field1=new-value1, field2=new-value2
[WHERE Clause]
```

- You can update one or more field altogether.
- You can specify any condition using the WHERE clause.
- You can update the values in a single table at a time.

The WHERE clause is very useful when you want to update the selected rows in a table.

MySQL – UPDATE Query

Updating Data from the Command Prompt

This will use the SQL UPDATE command with the WHERE clause to update the selected data in the MySQL table – **tutorials_tbl**.

Example

The following example will update the **tutorial_title** field for a record having the **tutorial_id** as 3.

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> UPDATE tutorials_tbl
    -> SET tutorial_title='Learning JAVA'
    -> WHERE tutorial_id=3;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0


mysql>
```

MySQL – UPDATE Query

Updating Data Using a PHP Script

You can use the SQL UPDATE command with or without the WHERE CLAUSE into the PHP function – **mysql_query()**. This function will execute the SQL command in a similar way it is executed at the mysql> prompt.

Example

Try out the following example to update the **tutorial_title** field for a record having tutorial_id as 3.

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
$sql = 'UPDATE tutorials_tbl
        SET tutorial_title="Learning JAVA"
        WHERE tutorial_id=3';
mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not update data: ' . mysql_error());
}
echo "Updated data successfully\n";
mysql_close($conn);
?>
```

MySQL – Delete Query

If you want to delete a record from any MySQL table, then you can use the SQL command **DELETE FROM**. You can use this command at the mysql> prompt as well as in any script like PHP.

Syntax: The following code block has a generic SQL syntax of the DELETE command to delete data from a MySQL table.

```
DELETE FROM table_name [WHERE Clause]
```

- If the WHERE clause is not specified, then all the records will be deleted from the given MySQL table.
- You can specify any condition using the WHERE clause.
- You can delete records in a single table at a time.

The WHERE clause is very useful when you want to delete selected rows in a table.

MySQL – Delete Query

Deleting Data from the Command Prompt

This will use the SQL DELETE command with the WHERE clause to delete selected data into the MySQL table – **tutorials_tbl**.

Example

The following example will delete a record from the tutorial_tbl whose tutorial_id is 3.

```
root@host# mysql -u root -p password;

Enter password:*******

mysql> use TUTORIALS;

Database changed

mysql> DELETE FROM tutorials_tbl WHERE tutorial_id=3;

Query OK, 1 row affected (0.23 sec)


mysql>
```

MySQL – Delete Query

Deleting Data Using a PHP Script

You can use the SQL DELETE command with or without the WHERE CLAUSE into the PHP function – **mysql_query()**. This function will execute the SQL command in the same way as it is executed at the mysql> prompt.

Example

Try the following example to delete a record from the tutorial_tbl whose tutorial_id is 3.

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
$sql = 'DELETE FROM tutorials_tbl
        WHERE tutorial_id=3';

mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not delete data: ' . mysql_error());
}
echo "Deleted data successfully\n";
mysql_close($conn);
?>
```

MySQL – LIKE Clause

We have seen the SQL **SELECT** command to fetch data from the MySQL table. We can also use a conditional clause called as the **WHERE** clause to select the required records.

A WHERE clause with the 'equal to' sign (=) works fine where we want to do an exact match. Like if "tutorial_author = 'Sanjay'". But there may be a requirement where we want to filter out all the results where tutorial_author name should contain "jay". This can be handled using **SQL LIKE Clause** along with the WHERE clause.

If the SQL LIKE clause is used along with the **%** character, then it will work like a meta character (*) as in UNIX, while listing out all the files or directories at the command prompt. Without a **%** character, the LIKE clause is very same as the **equal to** sign along with the WHERE clause.

Syntax: The following code block has a generic SQL syntax of the SELECT command along with the LIKE clause to fetch data from a MySQL table.

```
SELECT field1, field2,...fieldN table_name1, table_name2...
WHERE field1 LIKE condition1 [AND [OR]] filed2 = 'somevalue'
```

- You can specify any condition using the WHERE clause.
- You can use the LIKE clause along with the WHERE clause.
- You can use the LIKE clause in place of the **equal to** sign.
- When LIKE is used along with **%** sign then it will work like a meta character search.
- You can specify more than one condition using **AND** or **OR** operators.
- A WHERE...LIKE clause can be used along with DELETE or UPDATE SQL command also to specify a condition.

MySQL – LIKE Clause

Using the LIKE clause at the Command Prompt

This will use the SQL SELECT command with the WHERE...LIKE clause to fetch the selected data from the MySQL table – **tutorials_tbl**.

Example

The following example will return all the records from the **tutorials_tbl** table for which the author name ends with **jay**:

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl
    -> WHERE tutorial_author LIKE '%jay';
+-------------+----------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+----------------+-----------------+-----------------+
|           3 | JAVA Tutorial  | Sanjay          | 2007-05-21      |
+-------------+----------------+-----------------+-----------------+
1 rows in set (0.01 sec)

mysql>
```

Using LIKE clause inside PHP Script

You can use similar syntax of the WHERE...LIKE clause into the PHP function – **mysql_query()**. This function is used to execute the SQL command and later another PHP function – **mysql_fetch_array()** can be used to fetch all the selected data, if the WHERE...LIKE clause is used along with the SELECT command.

But if the WHERE...LIKE clause is being used with the DELETE or UPDATE command, then no further PHP function call is required.

Example

Try out the following example to return all the records from the **tutorials_tbl** table for which the author name contains **jay**:

MySQL – LIKE Clause

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
$sql = 'SELECT tutorial_id, tutorial_title,
                tutorial_author, submission_date
        FROM tutorials_tbl
        WHERE tutorial_author LIKE "%jay%"';

mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_ASSOC))
{
    echo "Tutorial ID :{$row['tutorial_id']}  <br> ".
         "Title: {$row['tutorial_title']} <br> ".
         "Author: {$row['tutorial_author']} <br> ".
         "Submission Date : {$row['submission_date']} <br> ".
         "--------------------------------<br>";
}
echo "Fetched data successfully\n";
mysql_close($conn);
?>
```

MySQL – Sorting Results

We have seen the SQL **SELECT** command to fetch data from a MySQL table. When you select rows, the MySQL server is free to return them in any order, unless you instruct it otherwise by saying how to sort the result. But, you sort a result set by adding an **ORDER BY** clause that names the column or columns which you want to sort.

Syntax:

The following code block is a generic SQL syntax of the SELECT command along with the ORDER BY clause to sort the data from a MySQL table.

```
SELECT field1, field2,...fieldN table_name1, table_name2...
ORDER BY field1, [field2...] [ASC [DESC]]
```

- You can sort the returned result on any field, if that field is being listed out.
- You can sort the result on more than one field.
- You can use the keyword ASC or DESC to get result in ascending or descending order. By default, it's the ascending order.
- You can use the WHERE...LIKE clause in the usual way to put a condition.

MySQL – Sorting Results

Using ORDER BY clause at the Command Prompt

This will use the SQL SELECT command with the **ORDER BY** clause to fetch data from the MySQL table – **tutorials_tbl**.

Example

Try out the following example, which returns the result in an ascending order.

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl ORDER BY tutorial_author ASC
+-------------+----------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+----------------+-----------------+-----------------+
|           2 | Learn MySQL    | Abdul S         | 2007-05-24      |
|           1 | Learn PHP      | John Poul       | 2007-05-24      |
|           3 | JAVA Tutorial  | Sanjay          | 2007-05-06      |
+-------------+----------------+-----------------+-----------------+
3 rows in set (0.42 sec)

mysql>
```

Verify all the author names that are listed out in the ascending order.

MySQL – Sorting Results

Using ORDER BY clause inside a PHP Script

You can use a similar syntax of the ORDER BY clause into the PHP function – **mysql_query()**. This function is used to execute the SQL command and later another PHP function **mysql_fetch_array()** can be used to fetch all the selected data.

Example

Try out the following example, which returns the result in a descending order of the tutorial authors.

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
$sql = 'SELECT tutorial_id, tutorial_title,
                tutorial_author, submission_date
        FROM tutorials_tbl
        ORDER BY  tutorial_author DESC';

mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_ASSOC))
{
     echo "Tutorial ID :{$row['tutorial_id']}  <br> ".
          "Title: {$row['tutorial_title']} <br> ".
          "Author: {$row['tutorial_author']} <br> ".
          "Submission Date : {$row['submission_date']} <br> ".
          "--------------------------------<br>";
}
echo "Fetched data successfully\n";
mysql_close($conn);
?>
```

MySQL – Using Join

You can use multiple tables in your single SQL query. The act of joining in MySQL refers to smashing two or more tables into a single table.

You can use JOINS in the SELECT, UPDATE and DELETE statements to join the MySQL tables. We will see an example of the LEFT JOIN also which is different from the simple MySQL JOIN.

Using Joins at the Command Prompt

Assume we have two tables **tcount_tbl** and **tutorials_tbl**, in TUTORIALS. Now take a look at the examples given below:

Example

Try out the following examples:

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * FROM tcount_tbl;
+-----------------+----------------+
| tutorial_author | tutorial_count |
+-----------------+----------------+
| mahran          |             20 |
| mahnaz          |           NULL |
| Jen             |           NULL |
| Gill            |             20 |
| John Poul       |              1 |
| Sanjay          |              1 |
+-----------------+----------------+
6 rows in set (0.01 sec)
```

```
mysql> SELECT * from tutorials_tbl;

+-------------+---------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+---------------+-----------------+-----------------+
|           1 | Learn PHP      | John Poul       | 2007-05-24      |
|           2 | Learn MySQL    | Abdul S         | 2007-05-24      |
|           3 | JAVA Tutorial  | Sanjay          | 2007-05-06      |
+-------------+---------------+-----------------+-----------------+
3 rows in set (0.00 sec)
mysql>

mysql> SELECT a.tutorial_id, a.tutorial_author, b.tutorial_count
    -> FROM tutorials_tbl a, tcount_tbl b
    -> WHERE a.tutorial_author = b.tutorial_author;
+-------------+-----------------+----------------+
| tutorial_id | tutorial_author | tutorial_count |
+-------------+-----------------+----------------+
|           1 | John Poul       |              1 |
|           3 | Sanjay          |              1 |
+-------------+-----------------+----------------+
2 rows in set (0.01 sec)
mysql>
```

MySQL – Using Join

Using Joins in a PHP Script

You can use any of the above-mentioned SQL query in the PHP script. You only need to pass the SQL query into the PHP function **mysql_query()** and then you will fetch results in the usual way.

Example

Try the following example:

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
$sql = 'SELECT a.tutorial_id, a.tutorial_author, b.tutorial_count
        FROM tutorials_tbl a, tcount_tbl b
        WHERE a.tutorial_author = b.tutorial_author';

mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_ASSOC))
{
    echo "Author:{$row['tutorial_author']}  <br> ".
         "Count: {$row['tutorial_count']} <br> ".
         "Tutorial ID: {$row['tutorial_id']} <br> ".
         "--------------------------------<br>";
}
echo "Fetched data successfully\n";
mysql_close($conn);
?>
```

MySQL – Using Join

MySQL LEFT JOIN

A MySQL left join is different from a simple join. A MySQL LEFT JOIN gives some extra consideration to the table that is on the left.

If I do a **LEFT JOIN**, I get all the records that match in the same way and in addition I get an extra record for each unmatched record in the left table of the join; thus ensuring (in my example) that every AUTHOR gets a mention.

Example

Try the following example to understand the LEFT JOIN.

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT a.tutorial_id, a.tutorial_author, b.tutorial_count
    -> FROM tutorials_tbl a LEFT JOIN tcount_tbl b
    -> ON a.tutorial_author = b.tutorial_author;
+-------------+-----------------+----------------+
| tutorial_id | tutorial_author | tutorial_count |
+-------------+-----------------+----------------+
|           1 | John Poul       |              1 |
|           2 | Abdul S         |           NULL |
|           3 | Sanjay          |              1 |
+-------------+-----------------+----------------+
3 rows in set (0.02 sec)
```

MySQL – Null Values

We have seen the SQL **SELECT** command along with the **WHERE** clause to fetch data from a MySQL table, but when we try to give a condition, which compares the field or the column value to **NULL**, it does not work properly.

To handle such a situation, MySQL provides three operators:

- **IS NULL:** This operator returns true, if the column value is NULL.
- **IS NOT NULL:** This operator returns true, if the column value is not NULL.
- **<=>:** This operator compares values, which (unlike the = operator) is true even for two NULL values.

The conditions involving NULL are special. You cannot use = **NULL** or != **NULL** to look for NULL values in columns. Such comparisons always fail because it is impossible to tell whether they are true or not. Sometimes, even NULL = NULL fails.

To look for columns that are or are not NULL, use **IS NULL** or **IS NOT NULL**.

MySQL – Null Values

Using NULL values at the Command Prompt

Assume that there is a table called **tcount_tbl** in the TUTORIALS database and it contains two columns namely **tutorial_author** and **tutorial_count**, where a NULL tutorial_count indicates that the value is unknown.

Example

Try the following examples:

```
root@host# mysql -u root -p password;
Enter password:*******

mysql> use TUTORIALS;
Database changed

mysql> create table tcount_tbl
    -> (
    -> tutorial_author varchar(40) NOT NULL,
    -> tutorial_count  INT
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO tcount_tbl
    -> (tutorial_author, tutorial_count) values ('mahran', 20);
mysql> INSERT INTO tcount_tbl
    -> (tutorial_author, tutorial_count) values ('mahnaz', NULL);
mysql> INSERT INTO tcount_tbl
    -> (tutorial_author, tutorial_count) values ('Jen', NULL);
mysql> INSERT INTO tcount_tbl
    -> (tutorial_author, tutorial_count) values ('Gill', 20);

mysql> SELECT * from tcount_tbl;
+-----------------+----------------+
| tutorial_author | tutorial_count |
+-----------------+----------------+
| mahran          |             20 |
| mahnaz          |           NULL |
| Jen             |           NULL |
| Gill            |             20 |
+-----------------+----------------+
4 rows in set (0.00 sec)

mysql>
```

You can see that = and != do not work with NULL values as follows:

```
mysql> SELECT * FROM tcount_tbl WHERE tutorial_count = NULL;
Empty set (0.00 sec)
mysql> SELECT * FROM tcount_tbl WHERE tutorial_count != NULL;
Empty set (0.01 sec)
```

To find the records where the tutorial_count column is or is not NULL, the queries should be written as shown in the following program.

```
mysql> SELECT * FROM tcount_tbl
    -> WHERE tutorial_count IS NULL;
+-----------------+----------------+
| tutorial_author | tutorial_count |
+-----------------+----------------+
| mahnaz          |           NULL |
| Jen             |           NULL |
+-----------------+----------------+
2 rows in set (0.00 sec)

mysql> SELECT * from tcount_tbl
    -> WHERE tutorial_count IS NOT NULL;
+-----------------+----------------+
| tutorial_author | tutorial_count |
+-----------------+----------------+
| mahran          |             20 |
| Gill            |             20 |
+-----------------+----------------+
2 rows in set (0.00 sec)
```

MySQL – Null Values

Handling NULL Values in a PHP Script

You can use the **if...else** condition to prepare a query based on the NULL value.

Example

The following example takes the **tutorial_count** from outside and then compares it with the value available in the table.

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
if( isset($tutorial_count ))
{
   $sql = 'SELECT tutorial_author, tutorial_count
            FROM  tcount_tbl
            WHERE tutorial_count = $tutorial_count';
}
else
{
   $sql = 'SELECT tutorial_author, tutorial_count
            FROM  tcount_tbl
            WHERE tutorial_count IS $tutorial_count';
}
mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_ASSOC))
{
    echo "Author:{$row['tutorial_author']}  <br> ".
         "Count: {$row['tutorial_count']} <br> ".
         "--------------------------------<br>";
}
echo "Fetched data successfully\n";
mysql_close($conn);
?>
```

Obtaining and Using MySQL Metadata

There are three types of information, which you would like to have from MySQL.

- **Information about the result of queries:** This includes the number of records affected by any SELECT, UPDATE or DELETE statement.
- **Information about the tables and databases:** This includes information pertaining to the structure of the tables and the databases.
- **Information about the MySQL server:** This includes the status of the database server, version number, etc.

It is very easy to get all this information at the MySQL prompt, but while using PERL or PHP APIs, we need to call various APIs explicitly to obtain all this information.

Obtaining the Number of Rows Affected by a Query

PHP Example

In PHP, invoke the **mysql_affected_rows( )** function to find out how many rows a query changed.

```
$result_id = mysql_query ($query, $conn_id);
# report 0 rows if the query failed
$count = ($result_id ? mysql_affected_rows ($conn_id) : 0);
print ("$count rows were affected\n");
```

MySQL – Database Info

Listing Tables and Databases

It is very easy to list down all the databases and the tables available with a database server. Your result may be **null** if you don't have the sufficient privileges.

Apart from the method which is shown in the following code block, you can use **SHOW TABLES** or **SHOW DATABASES** queries to get the list of tables or databases either in PHP or in PERL.

PHP Example

```
<?php
$con = mysql_connect("localhost", "userid", "password");
if (!$con)
{
  die('Could not connect: ' . mysql_error());
}

$db_list = mysql_list_dbs($con);

while ($db = mysql_fetch_object($db_list))
{
  echo $db->Database . "<br />";
}
mysql_close($con);
?>
```