MySQL

MySQL is a open source Relational Database Management System. MySQL is very fast reliable and flexible Database Management System. It provides a very high performance and it is multi threaded and multi user Relational Database management system.

MySQL is one of the most popular relational database Management System on the web. The MySQL Database has become the world's most popular open source Database, because it is free and available on almost all the platforms. The MySQL can run on Unix , window, and Mac OS. MySQL is used for the internet applications as it provides good speed and is very secure. MySQL was developed to manage large volumes of data at very high speed to overcome the problems of existing solutions. MySQL can be used for verity of applications but it is mostly used for the web applications on the internet.

MySQL Features

1. MySQL are very fast and much reliable for any type of application.
2. MySQL is very Lightweight application.
3. MySQL command line tool is very powerful and can be used to run SQL queries against database.
4. MySQL Supports indexing and binary objects.
5. It is allow changes to structure of table while server is running.
6. MySQL Written in C and C++ language.
7. MySQL code is tested with different compilers.
8. MySQL is available as a separate program for use in a client/server network environment.
9. The MySQL available for the most Unix operating platform.
10. MySQL are the available for window operating system window NT, window 95 ,and window 98.
11. Programming libraries for C, Python, PHP, Java , Delphi etc. are available to connect to MySQL database.

MySQL Advantages:

Reliability and Performance

MySQL is very reliable and high performance relational database management system. It can used to store many GB's of data into database.

Availability of Source

MySQL source code is available that's why now you can recompile the source code.

Cross-Platform support

MySQL supports more then twenty different platform including the major Linux distribution .Mac OS X ,Unix and Microsoft windows.

Powerful Uncomplicated software

The MySQL has most capabilities to handle most corporate database application and used to very easy and fast.

MySQL Data Types

Once you have identified the tables and the fields for your database, the next step is to determine each fields data type. With any relational database management system, you need to define what kind of information each field will contain. In most relational database management systems, there are three primary categories of field types:

- Text
- Numbers
- Dates and Times

MySQL Data Types

Note: The square brackets [] indicate an optional parameter to be put in parentheses, while parentheses () indicate required arguments.

Type {storage}	Name	Range	Attributes	Default
Numeric {1 byte}	TINYINT[(M)]	-128 TO 127 [0 to 255 if UNSIGNED]	AUTO_INCREMENT UNSIGNED, ZEROFILL, SERIAL DEFAULT VALUE	NULL [0 if NOT NULL]
Numeric {2 bytes}	SMALLINT[(M)]	-32,768 to 32,767 [0 to 65,535]	AUTO_INCREMENT, UNSIGNED, ZEROFILL, SERIAL DEFAULT VALUE	NULL [0 if NOT NULL]
Numeric {3 bytes}	MEDIUMINT[(M)]	-8,388,608 to 8,388,607 [0 to 16,777,215]	AUTO_INCREMENT, UNSIGNED, ZEROFILL, SERIAL DEFAULT VALUE	NULL [0 if NOT NULL]
Numeric {4 bytes}	INT[(M)]	-/+2.147E+9 [0 to 4.294E+9]	AUTO_INCREMENT, UNSIGNED, ZEROFILL, SERIAL DEFAULT VALUE	NULL [0 if NOT NULL]
Numeric {8 bytes}	BIGINT[(M)]	-/+9.223E+18 [0 to 18.45E+18]	AUTO_INCREMENT, UNSIGNED, ZEROFILL, SERIAL DEFAULT VALUE	NULL [0 if NOT NULL]
Numeric {4 or 8}	FLOAT(p)	p=0-24 --> "FLOAT" p=25-53 --> "DOUBLE"	UNSIGNED, ZEROFILL	NULL [0 if NOT NULL]
Numeric {4 bytes}	FLOAT[(M,D)]	Min=+/-1.175E-38 Max=+/-3.403E+38	UNSIGNED, ZEROFILL	NULL [0 if NOT NULL]
Numeric {8 bytes}	DOUBLE[(M,D)]	Min=+/-2.225E-308 Max=+/-1.798E+308	UNSIGNED, ZEROFILL	NULL [0 if NOT NULL]
Numeric {M+2}	DECIMAL[(M,[D])] Stored as string	Max Range = DOUBLE range Fixed point vs. DOUBLE float	UNSIGNED, ZEROFILL	NULL [0 if NOT NULL]
Bit {8 bytes}	BIT[(M)]	Binary. Display by [add zero or converting with BIN()]. M=1-64	Prior to 5.03 TINYINT(1) Synonym	NULL [0 if NOT NULL]
String	CHAR[(M)]	M=0-255 Characters, FIXED.	BINARY,	NULL

{M char's}		Right padded with spaces.	CHARACTER SET	["" if NOT NULL]
String {M char's[1]}	VARCHAR(M)	M=0-65,535 Characters M=0-255 <v5.0.3	BINARY, CHARACTER SET	NULL ["" if NOT NULL]
String {#char's[1]}	TINYTEXT[2]	0-255 Characters	BINARY, CHARACTER SET	NULL ["" if NOT NULL]
String {#char's[1]}	TEXT[2]	0-65,535 Char's	BINARY, CHARACTER SET	NULL ["" if NOT NULL]
String {#char's[1]}	MEDIUMTEXT[2]	0-16,777,215 Char's	BINARY, CHARACTER SET	NULL ["" if NOT NULL]
String {#char's[1]}	LONGTEXT[2]	0-4,294,967,295 Char's	BINARY, CHARACTER SET	NULL ["" if NOT NULL]
String {M bytes}	BINARY[(M)]	M=0-255 bytes, FIXED.	Global Only (case sensitive)	NULL ["" if NOT NULL]
String {M bytes}	VARBINARY(M)	0-65,535 bytes M=0-255 <v5.0.3	Global Only (case sensitive)	NULL ["" if NOT NULL]
String {#bytes[1]}	TINYBLOB	0-255 bytes	Global Only (case sensitive)	NULL ["" if NOT NULL]
String {#bytes[1]}	BLOB	0-65,535 bytes	Global Only (case sensitive)	NULL ["" if NOT NULL]
String {#bytes[1]}	MEDIUMBLOB	0-16,777,215 bytes	Global Only (case sensitive)	NULL ["" if NOT NULL]
String {#bytes[1]}	LONGBLOB	0-4,294,967,295 bytes	Global Only (case sensitive)	NULL ["" if NOT NULL]
String {1-2 bytes}	ENUM[2] ("A1","A2",...)	Column is exactly 1 of 1-65,535 values	CHARACTER SET	NULL [1st value if NOT NULL]
String {1-8 bytes}	SET[2] ("A1","A2",...)	Column is 0 or more values in list of 1-64 members	CHARACTER SET	NULL ["" if NOT NULL]
Date & Time {3 bytes}	DATE	"1000-01-01" - "9999-12-31"	Global Only (YYYY-MM-DD)	NULL ["0000-00-00" if NOT NULL]
Date & Time {8 bytes}	DATETIME	"1000-01-01 00:00:00" - "9999-12-31 23:59:59"	Global Only (YYYY-MM-DD hh:mm:ss)	NULL ["0000-00-00 00:00:00" if NOT NULL]
Date & Time {3 bytes}	TIME	"-838:59:59" - "838:59:59"	Global Only (hh:mm:ss)	NULL ["00:00:00" if NOT NULL]
Date & Time {4 bytes}	TIMESTAMP	19700101000000 - 2037+	Global Only (YYYYMMDDhhmmss)	Current Date & Time

Date & Time {1 bytes}	YEAR	1900 - 2155	Global Only (YYYY)	NULL ["0000" if NOT NULL]

MySQL Functions

Connecting To MySQL

- mysql_connect()

The mysql_connect() function opens a MySQL connection. This function returns the connection on success, or FALSE and an error on failure. You can hide the error output by adding an '@' in front of the function name.

mysql_connect(server,user,pwd,newlink,clientflag)

Parameter	Description
server	Optional. Specifies the server to connect to (can also include a port number. e.g. "hostname:port" or a path to a local socket for the localhost). Default value is "localhost:3306"
User	Optional. Specifies the username to log in with. Default value is the name of the user that owns the server process
Pwd	Optional. Specifies the password to log in with. Default is ""
newlink	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
clientflag	Optional. Can be 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

Tip: The connection will be closed as soon as the script ends. To close the connection before, use mysql_close().

Example

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

// some code

mysql_close($con);
?>
```

- mysql_select_db()

The mysql_select_db() function sets the active MySQL database. This function returns TRUE on success, or FALSE on failure.

mysql_select_db(database,connection)

Parameter	Description
database	Required. Specifies the database to select.
connection	Optional. Specifies the MySQL connection. If not specified, the last connection opened by mysql_connect() or mysql_pconnect() is used.

Example

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

$db_selected = mysql_select_db("test_db", $con);

if (!$db_selected)
  {
  die ("Can\'t use test_db : " . mysql_error());
  }

mysql_close($con);
?>
```

Making MySQL Queries

Create a Database

The CREATE DATABASE statement is used to create a database in MySQL.

```
CREATE DATABASE database_name
```

Create a Table

The CREATE TABLE statement is used to create a table in MySQL.

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
....
)
```

Example : The following example creates a database called "test_db" and a table named "Persons", with three columns. The column names will be "FirstName", "LastName" and "Age":

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

// Create database
if (mysql_query("CREATE DATABASE test_db",$con))
```

```
  {
  echo "Database created";
  }
else
  {
  echo "Error creating database: " . mysql_error();
  }

// Create table
mysql_select_db("test_db", $con);
$sql = "CREATE TABLE Persons ( FirstName varchar(15), LastName varchar(15), Age int )";
// Execute query
mysql_query($sql,$con);
mysql_close($con);
?>
```

Primary Keys and Auto Increment Fields

```
$sql = "CREATE TABLE Persons ( personID int NOT NULL AUTO_INCREMENT,
PRIMARY KEY(personID), FirstName varchar(15), LastName varchar(15), Age int )";
```

Note : Use this query in above example to create a primary key and auto increment field.

Insert Data Into a Database Table

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

mysql_select_db("test_db", $con);

mysql_query("INSERT INTO Persons (FirstName, LastName, Age)
VALUES ('Ashish', 'Modi', 29)");

mysql_close($con);
?>
```

Insert Data From a Form Into a Database

Now we will create an HTML form that can be used to add new records to the "Persons" table.

Here is the HTML form:

```
<html>
<body>
<form action="insert.php" method="post">
Firstname: <input type="text" name="firstname" />
Lastname: <input type="text" name="lastname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
</body>
</html>
```

When a user clicks the submit button in the HTML form in the example above, the form data is sent to "insert.php". The "insert.php" file connects to a database, and retrieves the values from the form with the PHP $_POST variables. Then, the mysql_query() function executes the INSERT INTO statement, and a new record will be added to the "Persons" table.

Here is the "insert.php" page:

```
<?php
$con = mysql_connect("localhost","","");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db("test_db", $con);
$sql="INSERT INTO Persons (FirstName, LastName, Age) VALUES
('$_POST[firstname]','$_POST[lastname]','$_POST[age]')";

if (!mysql_query($sql,$con))
  {
  die('Error: ' . mysql_error());
  }
echo "1 record added";
mysql_close($con)
?>
```

Select Data From a Database Table

The SELECT statement is used to select data from a database.

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

To get PHP to execute the statement above we must use the mysql_query() function.

Example

The following example selects all the data stored in the "Persons" table (The * character selects all the data in the table):

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

mysql_select_db("test_db", $con);

$result = mysql_query("SELECT * FROM Persons");

while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'] . " " . $row['LastName'];
  echo "<br />";
  }

mysql_close($con);
?>
```

The example above stores the data returned by the mysql_query() function in the $result variable. We use the mysql_fetch_array() function to return the first row from the recordset as an array. Each call to mysql_fetch_array() returns the next row in the recordset. The while loop loops through all the records in the recordset. To print the value of each row, we use the PHP $row variable ($row['FirstName'] and $row['LastName']).

The output of the code above will be:

Ashish Modi

Display the Result in an HTML Table

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

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

mysql_select_db("test_db", $con);
$result = mysql_query("SELECT * FROM Persons");
echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>";
while($row = mysql_fetch_array($result))
  {
  echo "<tr>";
  echo "<td>" . $row['FirstName'] . "</td>";
  echo "<td>" . $row['LastName'] . "</td>";
  echo "</tr>";
  }
echo "</table>";
mysql_close($con);
?>
```

The output of the code above will be:

Firstname	Lastname
Ashish	Modi

The WHERE clause

The WHERE clause is used to extract only those records that fulfill a specified criterion.

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

Example

The following example selects all rows from the "Persons" table where "FirstName='Ashish':

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

mysql_select_db("test_db", $con);

$result = mysql_query("SELECT * FROM Persons WHERE FirstName='Ashish'");

while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'] . " " . $row['LastName'];
  echo "<br />";
  }
?>
```

The output of the code above will be:

Ashish Modi

The ORDER BY Keyword

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

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

Example

The following example selects all the data stored in the "Persons" table, and sorts the result by the "Age" column:

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

mysql_select_db("test_db", $con);

$result = mysql_query("SELECT * FROM Persons ORDER BY age");

while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'];
  echo " " . $row['LastName'];
  echo " " . $row['Age'];
  echo "<br />";
  }

mysql_close($con);
?>
```

The output of the code above will be:

Ashish Modi 29
AAA BBB 35

Order by Two Columns

It is also possible to order by more than one column. When ordering by more than one column, the second column is only used if the values in the first column are equal:

SELECT column_name(s) FROM table_name ORDER BY column1, column2 ;

Update Data In a Database

The UPDATE statement is used to update existing records in a table.

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

Note: Notice the WHERE clause in the UPDATE syntax. The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

Example

Earlier in the tutorial we created a table named "Persons". Here is how it looks:

FirstName	LastName	Age
Ashish	Modi	29
AAA	BBB	33

The following example updates some data in the "Persons" table:

```
<?php
$con = mysql_connect("localhost","","");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db("test_db", $con);
mysql_query("UPDATE Persons SET Age=36 WHERE FirstName='AAA' AND
LastName='BBB'");

mysql_close($con);
?>
```

After the update, the "Persons" table will look like this:

FirstName	LastName	Age
Ashish	Modi	29
AAA	BBB	36

Delete Data In a Database

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

DELETE FROM table_name WHERE some_column = some_value ;

Example

FirstName	LastName	Age
Ashish	Modi	29
AAA	BBB	36

The following example deletes all the records in the "Persons" table where LastName='BBB':

```
<?php
$con = mysql_connect("localhost","","");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db("test_db", $con);
mysql_query("DELETE FROM Persons WHERE LastName='BBB'");

mysql_close($con);
?>
```

After the deletion, the table will look like this:

FirstName	LastName	Age
Ashish	Modi	29

Fetching Data From Database

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.

mysql_fetch_row(data)

Parameter	Description
data	Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function

Note: After the data is retrieved, this function moves to the next row in the recordset.

Example

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

$db_selected = mysql_select_db("test_db",$con);
$sql = "SELECT * from Persons WHERE Lastname='Modi'";
$result = mysql_query($sql,$con);
print_r(mysql_fetch_row($result));
mysql_close($con);
?>
```

The output of the code above could be:

```
Array
(
[0] => Ashish
)
```

mysql_fetch_object()

The mysql_fetch_object() function returns a row from a recordset as an object.

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

mysql_fetch_object(data)

Parameter	Description
data	Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function

Note: Each subsequent call to mysql_fetch_object() returns the next row in the recordset.

Example

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

$db_selected = mysql_select_db("test_db",$con);
$sql = "SELECT * from Persons";
$result = mysql_query($sql,$con);

while ($row = mysql_fetch_object($result))
  {
  echo $row->FirstName . "<br />";
  }

mysql_close($con);
?>
```

The output of the code above could be:

Ashish Modi
AAA BBB

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.

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: 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. Field names returned by this function are case-sensitive.

Example

```
<?php
$con = mysql_connect("localhost", "", "");
```

```
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

$db_selected = mysql_select_db("test_db",$con);
$sql = "SELECT * from Persons WHERE Lastname='Modi'";
$result = mysql_query($sql,$con);
print_r(mysql_fetch_array($result));
mysql_close($con);
?>
```

The output of the code above could be:

```
Array
(
[0] => Ashish
[FirstName] => Ashish
[1] => Modi
[LastName] => Modi
[2] => 22
[Age] => 22
)
```

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.

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: This function is slower than mysql_fetch_row(), mysql_fetch_array(), mysql_fetch_assoc() and mysql_fetch_object(). This function should not be used together with mysql_fetch_row(), mysql_fetch_array(), mysql_fetch_assoc() or mysql_fetch_object().

Example

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

$db_selected = mysql_select_db("test_db", $con);

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

```
$result = mysql_query($sql,$con);

echo mysql_result($result,0);

mysql_close($con);
?>
```

The output of the code above could be:

Ashish

mysql_affected_rows()

The mysql_affected_rows() function returns the number of affected rows in the previous MySQL operation. This function returns the number of affected rows on success, or -1 if the last operation failed.

Syntax : mysql_affected_rows(connection)

Parameter	Description
connection	Optional. Specifies the MySQL connection. If not specified, the last connection opened by mysql_connect() or mysql_pconnect() is used.

Example :

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

mysql_select_db("mydb");
mysql_query("DELETE FROM mytable WHERE id < 5");
$rc = mysql_affected_rows();
echo "Records deleted: " . $rc;

mysql_close($con);
?>
```

Output : Records deleted: 4

mysql_fetch_lengths()

The mysql_fetch_lengths() function returns the length of the contents of each field in a row. The row is retrieved by mysql_fetch_array(), mysql_fetch_assoc(), mysql_fetch_object(), or mysql_fetch_row(). This function returns a numeric array on success, or FALSE on failure or when there are no more rows.

Syntax : mysql_fetch_lengths(data)

Parameter	Description
data	Required. Specifies which data pointer to use. The data pointer is the

	result from the mysql_query() function

Example

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

$db_selected = mysql_select_db("test_db",$con);
$sql = "SELECT * from users WHERE name='xyz'";
$result = mysql_query($sql,$con);
print_r(mysql_fetch_row($result));
print_r(mysql_fetch_lengths($result));

mysql_close($con);
?>
```

The output of the code above could be:

```
Array
(
[0] => Refsnes
[1] => Kai Jim
[2] => Taugata 2
[3] => 22
)

Array
(
[0] => 7
[1] => 7
[2] => 9
[3] => 2
)
```

mysql_field_name()

The mysql_field_name() function returns the name of a field in a recordset. Returns the field name on success, or FALSE on failure.

Syntax : mysql_field_name(data,field_offset)

Parameter	Description
Data	Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function
field_offset	Required. Specifies which field to start returning. 0 indicates the first field

Example

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

$db_selected = mysql_select_db("test_db",$con);

$sql = "SELECT * from users";
$result = mysql_query($sql,$con);

$name = mysql_field_name($result, 2);

echo $name;

mysql_close($con);
?>
```

The output of the code above could be:

Email (field name)

mysql_num_rows()

The mysql_num_rows() function returns the number of rows in a recordset. This function returns FALSE on failure.

Syntax : mysql_num_rows(data)

Parameter	Description
data	Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function

Example

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

$db_selected = mysql_select_db("test_db",$con);

$sql = "SELECT * FROM users";
$result = mysql_query($sql,$con);
echo mysql_num_rows($result);

mysql_close($con);
?>
```

The output of the code above could be:

Fetch Table Name from Database (mysql_list_tables)

```
<?php
$dbname = 'test';
mysql_connect('localhost', 'root', '') or die('connection not create');
$sql = "SHOW TABLES FROM $dbname";
$result = mysql_query($sql);
if (!$result) {
   echo "DB Error, could not list tables\n";
   echo 'MySQL Error: ' . mysql_error();
   exit;
}
while ($row = mysql_fetch_row($result)) {
   echo "Table: {$row[0]}\n";
}
mysql_free_result($result);
?>
```