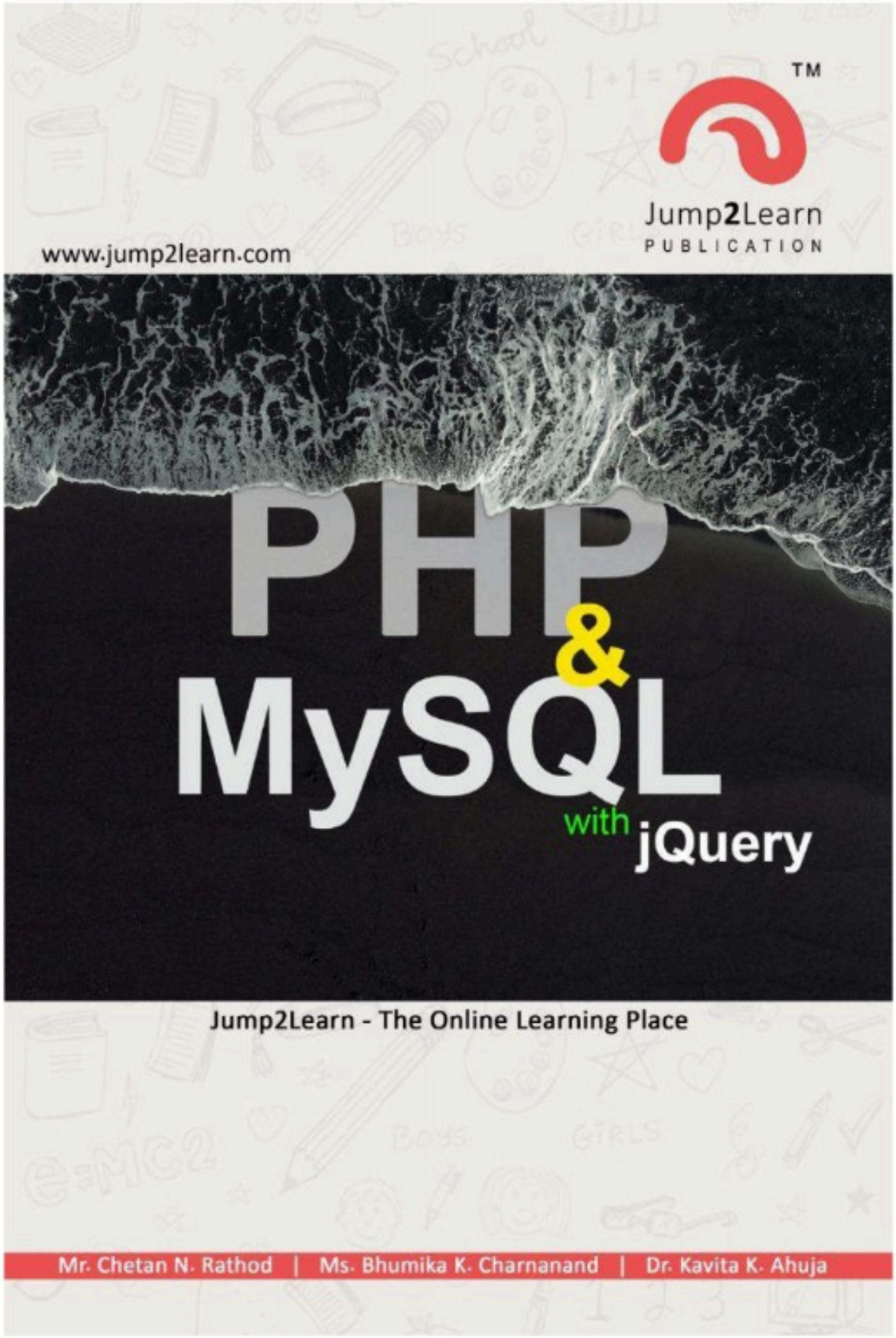
Jump2Learn
PUBLICATION
www.jump2learn.com
PHP
&
MySQL
with jQuery
Jump2Learn - The Online Learning Place
Mr. Chetan N. Rathod | Ms. Bhumika K. Charnanand | Dr. Kavita K. Ahuja

Unit - 4

MY SQL

Datatypes of MySQL
Attributes of Table in MySQL
Types of tables in MySQL
Functions of MySQL
Query in MySQL: Select, Insert, Update, Delete
Order By
Database connectivity of PHP with MySQL

MY SQL:

- MySQL is the most popular open-source database system.
- MySQL is a database server
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- The data in MySQL is stored in database objects called tables.
- A table is a collection of related data entries and it consists of columns and rows.
- Databases are useful for storing information.

DATABASE TABLES

A database contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Employees"). Tables contain records (rows) with data. Below is an example of a table called "Persons":

LastName	FirstName	Address	City
Rathod	Chetan	Adajan	Surat
Charnanand	Bhumika	Piplod	Surat
Ahuja	Kavita	Maninagar	Navsari

The table above contains three records (one for each person) and four columns (LastName, FirstName, Address, and City).

DIFFERENT DATATYPES OF MYSQL

char(length):

Character - based data can be stored in this field, You have to specify the fixed field length .So length of the field's data is fixed.

varchar(length):

Character - based data can be in this field, and the data can vary in length from 0 up to 255 characters.

int(length):

Integers that can range from _ 2,147,483,648 to +2,147,483,647 can be stored in this field.

int(length) unsigned:

Positive integers (and zero) up to 4,294,967,295 can be in this field.

text:

Any character - based data can be in this field, with a maximum size of 65,536 characters.

Decimal(length,dec):

Numeric field that can store decimals.

enum(" option1 " , " option2 " , ...):

Allows only certain values to be stored in this field, such as " true " and " false , " or a list of states. 65,535 different options are allowed.

Date:

Stores a date in YYYY - MM – DD format.

Time:

Stores time in hh:mm:ss format.

Datetime:

Multipurpose field that stores both the date and time together as

YYYY - MM - DD hh:mm:ss .

year(length):

Stores a year. By default, the year is four digits it is also possible to specify a two - digit format by using the length parameter.

tinyint(length):

Numeric field that stores integers from - 128 to 127. (Adding the unsigned parameter allows storage of 0 to 255.)

smallint(length):

Numeric field that stores integers from - 32,768 to 32,767. (Adding the unsigned parameter allows storage of 0 to 65,535.)

mediumint(length):

Numeric field that stores integers from - 8,388,608 to 8,388,607. (Adding the unsigned parameter allows storage of 0 to 16,777,215.)

bigint(length):

Numeric field that stores integers from - 9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. (Adding the unsigned parameter allows storage of 0 to 18,446,744,073,709,551,615.)

tinytext:

Allows storage of up to 255 characters.

Mediumtext:

Allows storage of up to 1,677,215 characters.

longtext:

Allows storage of up to 4,294,967,295 characters.

blob:

Equal to a text field, except that it is case - sensitive when sorting and comparing. Stores up to 65,535 characters. blob are generally used to store binary data.

tinyblob:

Equal to the tinytext field, except that it is case - sensitive when sorting and comparing; tinyblob are generally used to store binary data.

mediumblob:

Equal to the mediumtext field, except that it is case - sensitive when sorting and comparing; mediumblob are generally used to store binary data.

longblob:

Equal to the longtext field, except that it is case - sensitive when sorting and comparing; longblob are generally used to store binary data.

ATTRIBUTES OF TABLES IN MYSQL :

AUTO_INCREMENT:

The AUTO_INCREMENT attribute is use to assign unique integer identifiers to newly inserted rows. It increment the field value by 1 whenever a new raw is inserted. The AUTO_INCREMENT mostly used with PRIMARY KEY.

Example:

```
id SMALLINT NOT NULL AUTO_INCREMENT PRIMARY KEY
```


BINARY:

The BINARY attribute is only used with CHAR and VARCHAR values. When columns are assigned this attribute, they will be sorted in case-sensitive fashion (in accordance with their ASCII machine values).

Example:

```
hostname CHAR(25) BINARY NOT NULL
```

DEFAULT:

The DEFAULT attribute ensures that some constant value will be assigned when no other value is available. This value must be a constant, because MySQL does not allow functional or expressional values to be inserted. If the NULL attribute has been assigned to this field, the default value will be null if no default is specified.

Example:

```
flag  ENUM('0','1') NOT NULL DEFAULT '0'
```

INDEX:

Indexing a column creates a sorted array of keys for that column, each of which points to its corresponding table row. Searching this ordered key array from the input criteria increases the performance over searching the entire unindexed table because MySQL will already have the sorted array.

Example:

```
CREATE TABLE em(
id VARCHAR(8) NOT NULL,
firstname VARCHAR(15) NOT NULL,
lastname VARCHAR(25) NOT NULL,
phone VARCHAR(10) NOT NULL,
INDEX lastname (lastname),
PRIMARY KEY(id));
```

An index could be added after a table has been created by making use of MySQL's CREATE INDEX command:

Example:

CREATE INDEX lastname ON emp (lastname(7));

NATIONAL:

The NATIONAL attribute is used only with the CHAR and VARCHAR data types.

When specified, it ensures that the column uses the default character set, which MySQL already does by default. In short, this attribute is offered as an aid in database compatibility.

NOT NULL:

Defining a column as NOT NULL will disallow any attempt to insert a NULL value into the column. NOT NULL attribute is always suggested the user that all necessary values have been passed to the query.

Example:

zipcode VARCHAR(10) NOT NULL

NULL:

The NULL attribute indicates that no value can exist for the given field. **Note**: NULL is a mathematical term specifying "nothingness" rather than an empty string or zero. The NULL attribute is assigned to a field by default.

PRIMARY KEY:

The PRIMARY KEY attribute is used to guarantee uniqueness for a given row.

No values of this column (as a primary key) are repeatable or nullable. There are two other ways to ensure a record's uniqueness:

1. **SINGLE-FIELD PRIMARY KEYS:**

Single-field primary keys are typically used when there is a nonmodifiable unique identifier for each row entered into the database, such as a part number or Social EmpID.

Note: that this key should never change once set.

2. MULTIPLE-FIELD PRIMARY KEYS:

Multiple-field primary keys can be useful when it is not possible to guarantee uniqueness from any single field within a record. Thus, multiple fields are joined to

ensure uniqueness. The following three examples demonstrate creation of the auto-increment, single-field, and multiple-field primary key fields, respectively.

I. CREATING AN AUTOMATICALLY INCREMENTING PRIMARY KEY:

```
CREATE TABLE emp
(
id SMALLINT NOT NULL AUTO_INCREMENT,
firstname VARCHAR(15) NOT NULL,
lastname VARCHAR(25) NOT NULL,
email VARCHAR(55) NOT NULL,
PRIMARY KEY(id)
 );
```


II. CREATING A SINGLE-FIELD PRIMARY KEY:

```
CREATE TABLE citizens (
id VARCHAR(9) NOT NULL,
firstname VARCHAR(15) NOT NULL,
lastname VARCHAR(25) NOT NULL,
zipcode VARCHAR(9) NOT NULL,
PRIMARY KEY(id));
```

III. CREATING A MULTIPLE-FIELD PRIMARY KEY:

```
CREATE TABLE friends (
firstname VARCHAR(15) NOT NULL,
lastname VARCHAR(25) NOT NULL,
nickname varchar(15) NOT NULL,
PRIMARY KEY(lastname, nickname));
```


UNIQUE:

A column assigned the UNIQUE attribute will ensure that all values possess different values, except that NULL values are repeatable. You typically designate a column as UNIQUE to ensure that all fields within that column are distinct—for example, to prevent the same e-mail address from being inserted into a table.

```
email VARCHAR(55) UNIQUE
```

ZEROFILL:

The ZEROFILL attribute is available to any of the numeric types and will result in the replacement of all remaining field space with zeroes. For example, the default width of an unsigned INT is 10; therefore, a zero-filled INT value of 4 would be represented as 0000000004.

Example:

odometer MEDIUMINT UNSIGNED ZEROFILL NOT NULL

TYPES OF TABLES IN MySQL

MySQL table types/storage engines are essential features that can be used effectively for maximizing the database's performance. It handles create, read, and update operations for storing and managing the information in a database. In this tutorial, we are going to understand various storage engines or table types used in MySQL.

The following are various table types/storage engines supports in MySQL:

- ISAM
- MyISAM
- MERGE
- InnoDB
- MEMORY (HEAP)
- ARCHIVE
- BDB
- CSV
- FEDERATED

ISAM Table

It is abbreviated as Indexed Sequential Access Method. This table type/storage engine has been deprecated and removed from MySQL version 5.x. MyISAM now replaces the functionalities of this table. The size of an ISAM table is 4 GB, which requires expensive hardware. It is not portable.

MyISAM Table

It is an extension of the ISAM storage engine. The MyISAM table types are optimized for **compression and speed and can be easily portable** between system to system. Before version 5.5, if we do not specify the table type during table creation, it was the default storage engine. From version 5.x, InnoDB is used as the default table type/storage engine.

The MyISAM table size is dependent on the OS and can be up to **256 TB**. It can be compressed into **read-only** tables that save spaces in memory. MyISAM table type can store 64 keys per table and contains 1024 bytes maximum key length. The MyISAM tables work very fast, but they are not transaction-safe.

InnoDB Table

The InnoDB tables in MySQL fully support transaction-safe storage engine with ACID-compliant. It is the first table type that supports foreign keys. The InnoDB tables also provide optimal performance. Its size can be up to 64TB. InnoDB tables are also portable between systems to systems similar to MyISAM. InnoDB tables can also be checked and repairs by MySQL whenever necessary.

MERGE Table

MERGE table is also known as MRG_MyISAM. This table combines multiple MyISAM tables with a similar structure (identical column and index information with the same order) into a single table. This table uses indexes of the component tables because it does not have its own indexes. When we join multiple tables, it can also be used to speed up the database's performance. We can perform only INSERT, SELECT, DELETE, and UPDATE operations on the MERGE tables. If we use the DROP TABLE query in this storage engine, MySQL only removed the MERGE specification, and the underlying tables cannot be affected.

Memory Table

The memory table type/storage engine creates tables, which will be stored in our memory. It is also known as HEAP before MySQL version 4.1. This table type is faster than MyISAM because it uses hash indexes that retrieve results faster. We already know that data stored in memory can be crashed due to power issues or hardware failure. Therefore, we can only use this table as temporary work areas or read-only caches for data pulled from other tables. Hence, the memory/heap tables will be lost whenever the MySQL server halts or restarts. The memory/heap table's data life depends on the uptime of the database server.

CSV Table

The CSV table type/storage engine stores data in comma-separated values in a file. It provides a **convenient way to migrate data into many different software packages**, such as spreadsheet software. This table type is not as good as a general database engine; however, it enables us to exchange our data most effectively and easily. In addition, it will scan the whole table during the read operation.

FEDERATED Table

The FEDERATED table type/storage engine supports MySQL from version 5.03 that allows access data from a remote MySQL server without using the cluster/replication

technology. The federated storage engine located in local storage does not store any data. If we will query data from a federated table stored in local memory, MySQL automatically pulled data from the remote federated tables. It is to note that it's a way for a server, not for a client, for accessing a remote database. It is an effective way to combine data from more than one host or copy data from remote databases into local tables without using the data import and export method.

ARCHIVE Table

This table type/storage engine allows us to store a large volume of data in a compressed format to save disk space and cannot be modified. Thus, it is the perfect storage engine to store log data that is no longer in active use, such as the old invoice or sales data. It compresses the data during the insertion and can decompress it using the Zlib library.
The archive tables only support INSERT and SELECT queries. It does not support most of the data types, such as index data type, without which we need to scan a full table for reading rows.
Since it stored the information in a compressed format and if we want to read the table, we first need to decompress the information. This process will take time to perform complex searches and retrievals. Therefore, if we have to perform a large number of queries in these tables, it is beneficial to use another table such as MyISAM.

BDB Table

BDB stands for the Berkeley DB engine, which is developed by SleepyCat software. It is similar to InnoDB in the transaction-safe. It is based on the hash storage mechanism that makes the recovery of information very quickly. It supports page-level locking, but the data file is not portable.

FUNCTIONS OF MySQL IN PHP:

mysql_connect():

Create a Connection to a MySQL Database Before you can access data in a database, you must create a connection to the database.

Syntax:

mysql_connect(servername,username,password);

Parameter Description

servername	Optional. Specifies the server to connect to. Default value is "localhost:3306"
username	Optional. Specifies the username to log in with. Default value is the name of the user that owns the server process
password	Optional. Specifies the password to log in with. Default is ""

mysql_close():

To close the connection with MySQL Database.

Syntax:

mysql_close(connection name)

Example:

```
<?php
$con = mysql_connect("localhost","j2L","abc123");
if (!$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.

Syntax:

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:

```
mysql_select_db("test_db", $con);
```

mysql_query():

To run MySQL query we have touse this function.

Syntax:

```
mysql_query("Query")
```

Example:

```
mysql_query("CREATE DATABASE my_db",$con)
```


mysql_fetch_array()

Data required which specifies data pointer to use and the data pointer is the result from the mysql_query() function

Syntax:

```
mysql_fetch_array(data,array_type)
```


Example:

```
$db_selected=mysql_select_db("test_db",$con);
$sql="select *from person where lastname= 'Rathod' ";
$result=mysql_query($sql,$con);
Print_r(mysql_fetch_array($result));
```

mysql_num_rows():

The mysql_num_rows() function returns the number of rows in a recordset.

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:

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

Query in MySQL: Select, Insert, Update, Delete:

Queries:A query is a command or a request. With MySQL, we can query a database for specific information.

Example:

```
SELECT FirstName FROM Persons
```

The query above selects all the data in the "FirstName" column from the "Persons" table and display:

FirstName
Chetan
Bhumika
Kavita

CREATE DATABASE AND TABLES:

A database holds one or multiple tables.

1. CREATE A DATABASE:

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

Syntax:

CREATE DATABASE database_name

In PHP to execute this statement we must use the mysql_query() function. This function is used to send a query or command to a MySQL connection.

Example:

The following example creates a database called "my_db":

```
<?php
    $con = mysql_connect("localhost","root","");
    if (!$con)
     {
        die('Could not connect: ' . mysql_error());
     }
    if (mysql_query("CREATE DATABASE my_db",$con))
     {
        echo "Database created";
     }
    else
     {
        echo "Error creating database: " . mysql_error();
     }
    mysql_close($con);
?>
```


2. CREATE A TABLE:

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

Syntax:

```
CREATE TABLE table_name
(
  column_name1 data_type,
  column_name2 data_type,
  column_name3 data_type,
  ....
```

)

We must add the CREATE TABLE statement to the mysql_query() function to execute the command.

Example:

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

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con)
 {
   die('Could not connect: ' . mysql_error());
 }
// Create database
if (mysql_query("CREATE DATABASE my_db",$con))
 {
  echo "Database created";
 }
else
 {
  echo "Error creating database: " . mysql_error();
 }
// Create table
mysql_select_db("my_db", $con);
$sql = "CREATE TABLE Persons
 (
  FirstName varchar(15),
  LastName varchar(15),
  Age int )";
// Execute query
mysql_query($sql,$con);
mysql_close($con);
?>
```

Note:

A database must be selected before a table can be created. The database is selected with the mysql_select_db() function.

Each table should have a primary key field. A primary key is used to uniquely identify the rows in a table. Each primary key value must be unique within the table. The primary key field cannot be null.(because the database engine requires a value to locate the record.)

Example:

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

The above example sets the personID field as the primary key field. The primary key field is often used with the AUTO_INCREMENT setting. AUTO_INCREMENT automatically increases the value of the field by 1 each time a new record is added. To ensure that the primary key field cannot be null, we must add the NOT NULL setting to the field.

2. INSERT DATA INTO A DATABASE TABLE:

The INSERT INTO statement is used to add new records to a database table.

Syntax:

It is possible to write the INSERT INTO statement in two forms.

The first form doesn't specify the column names where the data will be inserted, only their values:

```
INSERT INTO table_name
VALUES (value1, value2, value3,...)
```

The second form specifies both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

In PHP to execute the statements above we must use the mysql_query() function. This function is used to send a query or command to a MySQL connection.

Example:

We have already created a table named "Persons", with three columns; "Firstname", "Lastname" and "Age". We will use the same table in this example.

The following example adds two new records to the "Persons" table:

```
<?php
  $con = mysql_connect("localhost","root","");
        if (!$con)
        {
          die('Could not connect: ' . mysql_error());
        }
  mysql_select_db("my_db", $con);
  mysql_query("INSERT INTO Persons (FirstName, LastName, Age)
  VALUES ('Chetan', 'Rathod', '27')");
  mysql_query("INSERT INTO Persons (FirstName, LastName, Age)
  VALUES ('Bhumika', 'Charnanand', '25')");
  mysql_query("INSERT INTO Persons (FirstName, LastName, Age)
  VALUES ('Kavita', 'Ahuja', '26')");
 mysql_close($con);
?>
```

Insert Data From a Form Into a Database

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

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, 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.

B. HERE IS THE "INSERT.PHP" PAGE:

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


5. SELECT DATA FROM A DATABASE TABLE

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

Syntax:

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

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

This function is used to send a query or command to a MySQL connection.

Example:

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

```
<?php
    $con = mysql_connect("localhost","root","");
        if (!$con)
         {
           die('Could not connect: ' . mysql_error());
         }
    mysql_select_db("my_db", $con);
    $result = mysql_query("SELECT * FROM Persons");
    while($row = mysql_fetch_array($result))
     {
        echo $row['FirstName'] . " " . $row['LastName'] . " " . $row['Age'];
        echo "<br />";
     }
    mysql_close($con);
?>
```

The example above stores the data returned by the mysql_query() function in the $result variable.

Here 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.

Through while loop we can fetch all the records in the recordset($row).

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:

Chetan Rathod 27
Bhumika Charnanand 25
Kavita Ahuja 26

6. 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
```

```
$con = mysql_connect("localhost","root","");
        if (!$con)
         {
            die('Could not connect: ' . mysql_error());
         }
mysql_select_db("my_db", $con);
$result = mysql_query("SELECT * FROM Persons");
echo "<table border='1'>
<tr>
        <th>Firstname</th>
        <th>Lastname</th>
        <th>Age</th>
</tr>";
        while($row = mysql_fetch_array($result))
         {
                echo "<tr>";
                echo "<td>" . $row['FirstName'] . "</td>";
                echo "<td>" . $row['LastName'] . "</td>";
                echo "<td>" . $row['Age'] . "</td>";
                echo "</tr>";
         }
echo "</table>";
mysql_close($con);
?>
```

The output of the code above will be:

Firstname	Lastname	Age
Chetan	Rathod	27
Bhumika	Charnanand	25
Kavita	Ahuja	26

7. UPDATE DATA IN A DATABASE:

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

Syntax:

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

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

In PHP to execute the statement above we must use the mysql_query() function. This function is used to send a query or command to a MySQL connection.

Example:

Table named "Persons". Here is how it looks:

FIRSTNAME	LASTNAME	AGE
Chetan	Rathod	28
Bhumika	Charnanand	25
Kavia	Ahuja	26

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

```
<?php
        $con = mysql_connect("localhost","root","");
        if (!$con)
         {
           die('Could not connect: ' . mysql_error());
         }
        mysql_select_db("my_db", $con);
        mysql_query("UPDATE Persons SET Age = '28'
        WHERE FirstName = 'Chetan' AND LastName = 'Rathod'");
        mysql_close($con);
?>
```

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

FIRSTNAME	LASTNAME	AGE
Chetan	Rathod	28
Bhumika	Charnanand	25
Kavita	Ahuja	26

8. DELETE DATA IN A DATABASE:

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

Syntax

```
DELETE FROM table_name
```

```
WHERE column_name = value
```

Note: The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

In PHP to execute the statement above we must use the mysql_query() function. This function is used to send a query or command to a MySQL connection.

Example:

Look at the following "Persons" table:

FIRSTNAME	LASTNAME	AGE
Chetan	Rathod	28
Bhumika	Charnanand	25
Kavita	Ahuja	26

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

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

After the deletion, the table will look like this:

FIRSTNAME	LASTNAME	AGE
Chetan	Rathod	28
Bhumika	Charnanand	25

9. DROPPING TABLES AND COLUMNS:

To drop or delete tables Drop command is used. DROP command drop(delete) table as well as all the data of the table.

Syntax:

DROP tablename;

Here table name is the name of the table you want to delete.

For example, to delete the persons table from the "my_db" database, you would type the following command:

DROP persons;

This will delete the entire table and all the data inside the table.

Note: Remember there are no warnings from the monitor. After you drop something, the only way to get it back is through a backup log.

If you need to delete a column from a table, enter the following command:

ALTER TABLE tablename DROP columnname;

Here table name is the table that holds the column you want to delete, and columnname is the column you want to delete.If you want to delete the LastName column of the Persons table, you would enter the following

statement:

ALTER TABLE Customers DROP LastName;

This will delete the column LastName and all the information that the column stored.

ORDER BY:

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.

Syntax:

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

```
                }
            mysql_select_db("my_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:

Bhumika Charnanand 25
Chetan Rathod 28

ORDER BY MORE THAN ONE COLUMN

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
```


- **Alias:**

To alias a table is to give it another name.

It help to save time and space when identifying the tables from which you want to get information.

The keyword AS must be used when you alias with MySQL.

Aliasing looks like the following:

```
SELECT O.Order_ID from Orders AS O, Customers AS C
WHERE C.Customer_ID = O.Customer_ID
```

As you can see, aliasing allows you to use a kind of shorthand when referencing tables, instead of typing the following code.

```
SELECT Orders.Order_ID FROM Orders, Customers
WHERE Customers.Customer_ID = Orders.Customer_ID
```

- It can save your time .
- It is very easy then typing the full table name out.
- You have to state the table name when using more than one table in the FROM clause so that the database knows which field you are talking about.

DATABASE CONNECTIVITY OF PHP WITH MYSQL:

Before you can access data in a database, you must create a connection to the database.In PHP, this is done with the mysql_connect() function.

Syntax:

```
mysql_connect(servername,username,password);
```

Parameter	Description
Servername	It is Optional. Specifies the server to connect to. Default value is "localhost:3306"
Username	It is Optional. Specifies the username to log in with. Default value is the name of the user that owns the server process
Password	It is Optional. Specifies the password to log in with. Default is ""

Note: There are more available parameters, but the ones listed above are the most important.

Example:

In the following example we store the connection in a variable ($con) for later use in the script.

The "die" part will be executed if the connection fails:

```
<?php
    $con=mysql_connect("localhost","root","");
    if(!$con)
    {
      die('Couldnotconnect:'.mysql_error());
    }
//somecode
?>
```


Closing a Connection:

The connection will be closed automatically when the script ends.

To close the connection before, use the mysql_close() function:

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

Sample Program for Insert, Update & Delete (Using query string)

DatabaseName: student_db

Table Name: student_info

You can create table using query or GUI in phpMyadmin.

Query:

```
CREATE TABLE IF NOT EXISTS student_info
(
 sid int(11) NOT NULL AUTO_INCREMENT,
 name varchar(30) NOT NULL,
 rolno varchar(10) NOT NULL,
 mno varchar(10) NOT NULL,
 PRIMARY KEY (`id`)
);
```

Code:

```
<html>
    <head>
        <title>All~In~One</title>
        <style type="text/css">
            .instable
            {
                /*margin-left:250px;*/
                border:solid 10px #666666;
                box-shadow: 0px 10px 10px 10px #c3c3c3;
                border-radius:20px;
                width:430px;
            }
            .maintable
```

```
                {
                        /*margin-left:250px;*/
                        border:solid 10px #666666;
                        box-shadow: 0px 10px 10px 10px #c3c3c3;
                        border-radius:20px;
                        width:850px;
                        background:linear-gradient(to bottom,White,#c3c3c3);
                        /*height:400px;                         */
                }
                .tableheading
                {
                        color:Black;
                        text-align:center;
                        font-size:x-large;
                }
                .insbtn
                {
                        width:250px;
                        background-color:white;
                        border:2px solid black;
                        border-radius:15px;
                        font-size:large;
                }

                .insbtn:hover
                {
                        background-color:#c3c3c3;
                        border:2px solid white;
                        color:white;
                }
        </style>
    </head>
    <?php
//---------------
//Global Code
    mysql_connect("localhost","root","") or die(mysql_error("Fail to connect"));
    mysql_select_db("student_db");
```

```
//----------------
//On InsertClick Button of Insert Sub Form
        if(isset($_GET['btnIns']))
                {
                $name = $_GET['name'];
                $rolno = $_GET['rolno'];
                $phn = $_GET['mno'];
                mysql_query("insert into student_info (name,rolno,mno) values
('$name','$rolno','$phn')");
                header("location: all_in_one.php");
                }
//----------------
//On DeleteClick of Main Form
                if(isset($_GET['act']))
                {
                        if($_GET['act'] == 'del')
                        {
                        $id= $_GET['del_id'];
                        mysql_query("delete from student_info where sid='$id'");
                        header("location: all_in_one.php");
                        }
                }
//----------------
//On EditClick of Edit Sub Form
                if(isset($_GET['btnEdit']))
                {
                $sid = $_GET['id'];
                $name = $_GET['name'];
                $rolno = $_GET['rolno'];
                $phn = $_GET['mno'];
                $sql=mysql_query("update student_info set name='$name',
                rolno='$rolno', mno='$phn' where sid='$sid'");
                header("location: all_in_one.php");
                }
            ?>
<body bgcolor="lavender">
```

```
<!------------------------------------------------------------------------------------------------------>
<!--                    Main form show data
                       -->
<!------------------------------------------------------------------------------------------------------>
                <form method="get" action="#">
                <br><br><br>
<table border="10" cellspacing="5" cellpadding="5" class="maintable" align="center">
                                <tr class="tableheading">
                                        <td> Student ID:</td>
                                        <td> Name : </td>
                                        <td> Roll No : </td>
                                        <td> Phone No : </td>
                                        <td> Edit | Delete</td>
                                </tr>
                                <tr>
        <td colspan="5" align="center" style="font-size:x-large;">
        <input type="submit" name="ShowIns" value="Insert" class="insbtn">
                                </td>
                                </tr>
        <?php
                $result=mysql_query("select * from student_info");
                while($row = mysql_fetch_array($result))
                {
                ?>
                <tr>
                        <td style="text-align:right;"><?php echo $row["sid"]; ?></td>
                        <td><?php echo $row["name"]; ?></td>
                        <td><?php echo $row["rolno"]; ?></td>
                        <td><?php echo $row["mno"]; ?></td>
                <td><a href="all_in_one.php?act=edit&stud_id=<?php echo
$row['sid']; ?>" >Edit</a> |
<a href="all_in_one.php?act=del&del_id=<?php echo $row['sid']; ?>">Delete</a>
                </td>
                </tr>
<?php
        }
        ?>
```

```
</table>
</form>
<!------------------------------------------------------------------------------------------------------------>
<!--                        Sub form show Edit form
                                    -->
<!------------------------------------------------------------------------------------------------------------>
<?php
if(isset($_GET['act']))
{
        if($_GET['act'] == 'edit')
        {
                $id=$_GET['stud_id'];
                $sql=mysql_query("select * from student_info where sid='$id'");
                while($row=mysql_fetch_array($sql))
                {
                ?>
                <form method="get">
                <br><br><br>
<table border="10" cellspacing="5" cellpadding="5" class="instable" align="center">
                <tr>
        <td>ID :</td>
        <td>
        <input type="text" name="id" value=<?php echo $row['sid'] ?>
style="width:250px;"><br>
        </td>
        </tr>
        <tr>
        <td>Name :</td>
        <td>
        <input type="text" name="name" value=<?php echo $row['name'] ?>
style="width:250px;"><br>
        </td>
        </tr>
        <tr>
        <td>Roll no :</td>
        <td>
```

```
        <input type="text" name="rolno" value=<?php echo $row['rolno'] ?>
style="width:250px;"><br>
        </td>
        </tr>
        <tr>
        <td>
         Phone No :</td>
        <td>
        <input type="text" name="mno" value=<?php echo $row['mno'] ?>
style="width:250px;">
        </td>
        </tr>
        <tr>
        <tdcolspan="2"></td>
        <td colspan="2">
        <input type="submit" name="btnEdit" value="Edit" style="width:105px;
margin-left:100px;">
        </td>
        </tr>
        </table>
        </form>
        <?php
        }
        }
        }
        ?>
<!------------------------------------------------------------------------------>
<!--                Sub form show Insert Form                                 -->
<!------------------------------------------------------------------------------>
<?php
if(isset($_GET['ShowIns']))
{
?>
<form method="get">
<br><br><br>
<table border="10" cellspacing="5" cellpadding="5" class="instable" align="center">
<tr>
```

```
<td>Name :
</td>
<td>
<input type="text" name="name" style="width:250px;"><br>
</td>
</tr>
<tr>
<td>Roll no :
</td>
<td>
<input type="text" name="rolno" style="width:250px;"><br>
</td>
</tr>
<tr>
<td>
Phone No :
</td>
<td>
<input type="text" name="mno" style="width:250px;">
</td>
</tr>
<tr>
<tdcolspan="2"></td>
<td colspan="2">
<input type="submit" name="btnIns" value="Insert" style="width:105px; margin-
left:100px;">
</td>
</tr>
</table>
</form>
<?php
}
?>
<!---------------------------------------------------------------------------------------------------->
</body>
</html>
```

For more Practical Programs visit website: **www.jump2learn.com**