# Unit 5 Databases

## 1. Introduction

Databases are the storage places where users can store their own or organizational or commercial data in the form of records. Records are stored in tables which are part of the database.

Using programming languages, programmers can fetch data from the databases without directly opening the database.

PHP has support for most commercial and open source databases.

## 2. Using PHP to Access Databases

The database used is MYSQL database. MYSQL is the popular open source database system.

1. **Database, tables:** The data in MySQL is stored in tables. A table is a collection of columns and rows which contain related data. Such multiple tables form one database. In MYSQL multiple databases can be created.
2. **Query:** A query is a request made to the database to provide with data or store data.

*There are two main types of queries:*

a. ***DDL:*** Data Definition Language (DDL) statements are used to define the database structure or schema. DDL simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in database, therefore language statements like CREATE TABLE or ALTER TABLE belongs to the DDL. DDL is about "metadata".

b. ***DML:*** Data Manipulation Language (DML) statements are used for managing data within schema objects, DML deals with data manipulation, and therefore includes most common SQL statements such as SELECT, INSERT, etc. DML allows to add / modify / delete data itself. DML is used to manipulate with the existing data in the database objects (insert, select, update, delete).

*Following are the frequently used query statements:*

| DDL statements | | |
|---|---|---|
| 1. | create | Creates a new table. |
| 2. | alter | Alters the existing table. |
| 3. | drop | Removes the existing table. |

| DML statements | | |
|---|---|---|
| 1. | select | Retrieve/ pick up data from the user. |
| 2. | insert | Insert data in the table. |
| 3. | delete | Delete the record from the table. |
| 4. | update | Update few of the data entries in the existing record. |

All these query statements have their own syntax and clauses in MySQL.PHP uses the same syntax to execute queries.

There are two ways to access databases from PHP.

1. Use a database-specific extension.
2. Use the database-independent PEAR DB library.

**Both these approaches have their own advantages and disadvantages:**

1. If we use database-specific extension, then the code is very much tied to that database only. So, changing one database to other involves many changes in the code.

2. PEAR DB approach is quite different than this and it is not bound to any specific database. It uses object oriented approach. But it is bit slower and do not give advantages of specific database.

## Database-Specific Extension

Following are the different steps taken by PHP to open the database and execute different queries on it.

**Step 1:** Create a Connection to MySQL Database.

**Step 2:** Create a Database (can be done in MySQL).

**Step 3:** Create a table (can be done in MySQL).

**Step 4:** Execute the query (select or any other).

**Step 5:** Display the Result in an HTML Table.

**Step 6:** Close the connection.

*Following are the various functions used in PHP to perform these tasks.*

### 1. Open the Connection

*mysql_connect()*

Before we can access data in a database, we must create a connection to the database. This is done with the mysql_connect() function.

*Syntax*

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

*Parameters*

| Parameter | Description |
|---|---|
| Server name | The server to connect to. Default value is "localhost". But if we need to connect to the remote host, then specify its ip-address. |
| Username | Specifies the username to log in with. |
| Password | Specifies the password for the user name. Default is " ". |

*Example,*

```
$con = mysql_connect("localhost","user","user123");
```

On success, it returns the connection.

On failure, it returns false.

*Following code snippet is frequently used to establish the connection.*

```
<?php
   $con = mysql_connect("localhost","user","user123");
   if(!$con)
     {
        die('Could not connect to the server:'. sql_error());
     }
  ?>
```

PHP provides two sets of file related functions, and the way in which they handle files:

i File handle or the file pointer: It is an integer value used to identify the file we want to work with.

ii. Other way is directly using filename strings.

**2. Close Connection**

*mysql_close()*

The connection created to the database, will be closed automatically when the script ends. If we want to close the connection before that, use the mysql_close() function.

*Syntax*

```
mysql_close(connection_name);
```

*Parameters*

| Parameter | Description |
|---|---|
| Connection_name | The connection created with mysql_connect(). |

*Example,*

```
mysql_close($con);
```

**3. Execute the Query**

*mysql_query()*

Using this function a query or command is sent to a MySQL connection.

*Syntax*

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

*Parameters*

| Parameter | Description |
|---|---|
| sql | The query statement. Actual query can be written instead of storing it in variable. |
| con | Name of the connection. Used to create database/ table. |

Note All the DDL (Create, Alter, Drop) and DML (Select, Insert, Update, Delete) statements are executed using mysql_query() command.

## 4. Create a Database

A database stores one or more tables.

In MySQL, database is created using the CREATE DATABASE statement.

*Syntax*

```
CREATE DATABASE database_name ;
```

In PHP, the mysql_query() function is used to execute the statement.

*Example,*

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

On success, it returns true otherwise it returns false.

*Following code snippet is frequently used.*

```
if(mysql_query("CREATE DATABASE  sampleDatabase ",$con))
{
  echo "Database created successfully."
}
else
{
  echo "Error creating database:" Mysql_error();
}
```

## 5. Create a Table

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

*Example.*

The following *example* creates a table named "Students", with three columns. The column names will be "Roll", "Name" and "Class".

```
create table Students
(
    Roll int PRIMARY KEY,
    Name varchar(20),
    Class varchar(10)
 );
```

The mysql_query() function is used to execute CREATE TABLE statement.

The following code snippet demonstrates use of create table.

```
// Create table
mysql_select_db("sampleDatabase", $con);
 $sql = "create table Students
    (
        Roll int PRIMARY KEY,
        Name varchar(20),
        Class varchar(10)
    )";
  // Execute query
       mysql_query($sql,$con);
```

**Important:** Before executing Create Table statement, always select Database.

## 6. Accessing Rows in Resultset

*i. mysql_fetch_array():* This function is used to fetch a result row as an associative array, a numeric array, or both.

*Syntax:*

```
array mysql_fetch_array (resource $result[,int $result_type
        = MYSQL_BOTH])
```

*Parameters*

| Parameter | Description |
|---|---|
| result | The result set returned by mysql_query() function. |
| result_type | The type of array that is to be fetched. It's a constant and can take any of these values: MYSQL_ASSOC, MYSQL_NUM, and MYSQL_BOTH. |

On success, it returns the first row from the record set as an array. To access the next row, this function is used repeatedly in some loop. It has internal data pointer

to move ahead to the next record. It uses the recordset returned by mysql_query() function.

On failure, it returns FALSE when there are no more rows.

To print the value of each row and the fields, PHP variable $row is used.

The following code demonstrates use of mysql_fetch_array() and $row variable.

```
<?php
$con = mysql_connect("localhost","user","user123");
if(!$con)
{
    die('Could not connect: ' . mysql_error());
}
mysql_select_db("sampleDatabase", $con);
$result = mysql_query("SELECT * FROM Students");
while($row = mysql_fetch_array($result))
 {
    echo $row['Roll'] . " " . $row['Name'];
    echo "<br />";
 }
mysql_close($con);
?>
```

*Important:* This function is well suitable for large result sets as it returns the entire row in an array. It is much quicker for such data retrieval.

ii. *mysql_result()*

This function is used to get result data. It retrieves the values stored in one cell from result set.

*Syntax*

```
string mysql_result(resource $result , int $row[,mixed $field =0 ])
```

*Parameters*

| Parameter | Description |
|---|---|
| result | The result set returned by mysql_query() function. |
| row | The row number from the result to select the values. Row numbers start at 0. |
| field | The name or offset of the field being retrieved. It can be the field's offset, the field's name. |

On success, it returns the value of the field.

On failure, it returns false.

*Example,*

```
$query= "select * from student;";
$result = mysql_query($query);
echo mysql_result($result,0,'roll');
echo mysql_result($result,0,'name');
```

It shows the values (roll, name) of the first row from result set.

**7. To Get the Number of Rows in Result**

*mysql_num_rows()*

This function is used to get the number of rows, i.e., the count in result set.

*Syntax*

```
int mysql_num_rows(resource $result)
```

*Parameters*

| Parameter | Description |
|---|---|
| result | The result set returned by mysql_query() function. |

On success, it returns the number of rows in a result set.

On failure, it returns **FALSE**.

*Example,*

```
$numrows = mysql_numrows($result);
```

*Important:* This function is well suitable for the query like Select.

**8. To Get the Number of Affected Rows**

*mysql_affected_rows()*

This function is used to get number of affected rows in the last MySQL operation. The operations are delete, update, insert etc.

*Syntax*

```
int mysql_affected_rows([ resource $link_identifier ])
```

*Parameter*

| Parameter | Description |
|---|---|
| link_identifier | (Optional) It is MySQL connection. If it is not specified, the last link opened by mysql_connect() is assumed. |

On success, it returns the count of rows. It returns –1 if the last query is failed.

*Example,*

```
$query="update student set name = "Lucky" where roll=5;";
$result = mysql_query($query);
$numrows=mysql_affected_rows();
```

## 3. Relational Databases and SQL

Relational databases store and manipulates data for the user. The data is organized in databases which is a collection of tables. Each table is made up of number of columns/fields. Each column has its own data type to store the values.

Following are the detailed programs to handle different DDL and DML statements.

### Execute DDL Statements

*Program 5.1: Create Table Statement*

```
<?php
//Establish connection to MySQL Db Engine
$user = "root";
$pwd ="";
$host ="localhost";
$dbcon = @mysql_connect($host,$user,$pwd);//or die("Connection
// Failure");
$database="php_data";
// ******** CHECKS THE CONNECTION STATUS ********
if(!$dbcon)
{
  echo "Connection Failure.";exit();
}
else
  echo "Connection successful";
echo "<br>";
// ************** OPEN THE DATABASE *****************
$status = mysql_select_db($database,$dbcon);
if(!$status)
{ echo "Database not selected";
}
```

```
else
   echo "Database available";
   echo "<br>";
//************** EXECUTE THE Create table QUERY *****************
$query="create table Department(
                                    Dno int PRIMARY KEY,
                                    Dname char(20)
                                  ) ;  ";
$result = mysql_query($query);
echo "Table created successfully.";
mysql_close($dbcon);
?>
```

**Explanation**

1. Here, we establish the connection with MySQL using mysql_connect() function.

   When we use @ symbol before mysql_connect() function, it does not show any system messages.

2. We check the status of the connection explicitly by if-else. Otherwise, same can be achieved by specifying die().die() executes on the failure of function.

3. Connect to the database by using mysql_select_db() function. It accepts database name and connection resource. Database name must be same as we use in MySQL. Here also we check the success of the function with if-else. Here we use 'php_data' database.

4. Execute the create statement by mysql_query() function. Its syntax is same as in MySQL.

5. Close the connection.

Note

Often, tables are created in databases and rows are accessed from PHP script.

*Program 5.2: Drop Table Statement*

```
<?php
   //Establish connection to MySQL Db Engine
      $user = "root";
      $pwd = "";
      $host = "localhost";
      $dbcon = @mysql_connect($host,$user,$pwd);
//or die("Connection Failure");
$database="php_data";
// ************ CHECKS THE CONNECTION STATUS ********
```

```
if(!$dbcon)
{
   echo "Connection Failure.";exit();
}
else
   echo "Connection successful";
   echo "<br>";
// ************** OPEN THE DATABASE *****************
$status = mysql_select_db($database, $dbcon);
   if(!$status)
      echo "Database not selected";
   else
      echo "Database available";
echo "<br>";
//************** EXECUTE THE Create table QUERY *****************
$query = "drop table department";
$result = mysql_query($query);
//echo "Table dropped successfully.";
mysql_close($dbcon);
?>
```

*Program 5.3: Alter Table Statement*

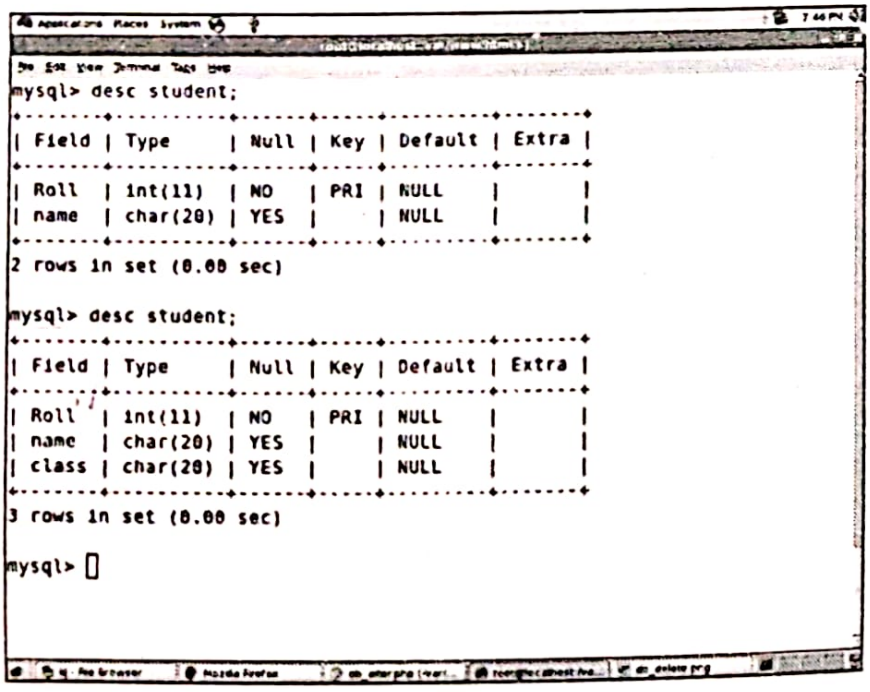
```
<?php
//Establish connection to MySQL Db Engine
   $user = "root";
   $pwd ="";
   $host ="localhost";
   $dbcon = @mysql_connect($host,$user,$pwd);
//or die("Connection //Failure");
   $database ="php_data";
// ************ CHECKS THE CONNECTION STATUS ********
if(!$dbcon)
{
  echo "Connection Failure.";exit();
}
else
   echo "Connection successful";
// ************** OPEN THE DATABASE *****************
$status = mysql_select_db($database,$dbcon);
if(!$status)
```

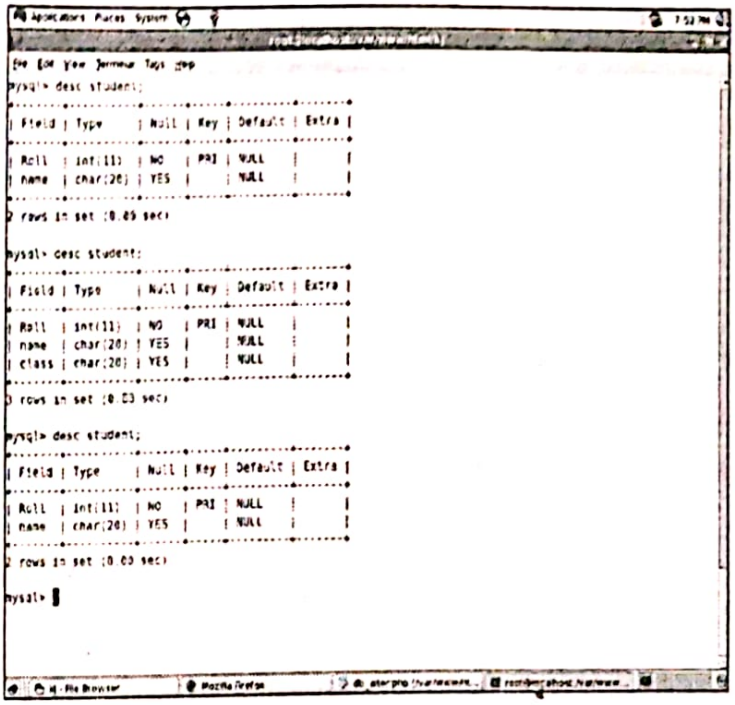
```
    echo "Database not selected";
  else
    echo "Database available";
   echo "<br>";
//************** EXECUTE THE ALTER QUERY *****************
$query="alter table student add column(class char(20));";// ADDS //THE
COLUMN class in table student.
$query = "alter table student drop column class ;";// DROPS THE
// COLUMN class FROM table student.
  $result = mysql_query($query);
  echo "Table altered successfully.";
  mysql_close($dbcon);
?>
```

***Output***

Following images demonstrate the alter operation as can be seen in MySQL.

1. Add column
2. Drop column






*Explanation*

1. Establish the connection with MySQL.
2. Open the database.
3. Execute the Alter statement. Using Alter, we can add or drop column from the table.
4. We specify both the queries. Execute them alternatively to get the effect of Alter statement.
5. Close the connection.

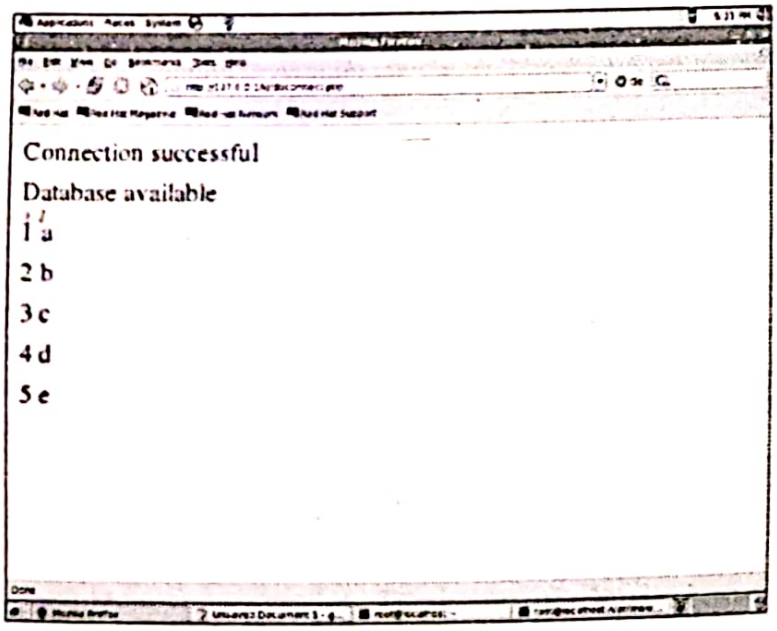
## Executing DML Statements

*Program 5.4: Execute Select Statement*

```
<?php
//Establish connection to MySQL Db Engine
   $user = "root"; // User name
   $pwd = "";      // Password
   $host = "localhost"; // server identification.
   $dbcon= @mysql_connect($host,$user,$pwd);//or die("Connection
// Failure");
$database = "php_data";
// ************ CHECKS THE CONNECTION STATUS ********
if(!$dbcon)
{
   echo "Connection Failure.";exit();
```

```
)
else
   echo "Connection successful";
// ************** OPEN THE DATABASE *****************
$status = mysql_select_db($database,$dbcon);
   if(!$status)
      echo "Database not selected";
   else
      echo "Database available";
echo "<br>";
//************** EXECUTE THE QUERY *****************
$query = "select * from student;";
$result = mysql_query($query);
$numrows = mysql_numrows($result);
$i=0;
// ************* SHOW THE RESULT ***************
echo "Roll Name";
while($i<$numrows)
{
   echo mysql_result($result,$i,'roll');
   echo mysql_result($result,$i,'name');
   $i++;
   echo "<br>";
}
// ***************** CLOSE THE CONNECTION ***************
mysql_close($dbcon);
?>
```

*Output*



*Explanation*

1. We establish the connection with MySQL using mysql_connect() function.
2. Connect to the database by using mysql_select_db() function.
3. We execute the select query by mysql_query() function.
4. Find out the number of rows returned by mysql_numrows() function. We use this number as the limiting number in while loop.
5. Using mysql_result() function, we display the result.
6. Close the connection with mysql_close() function.

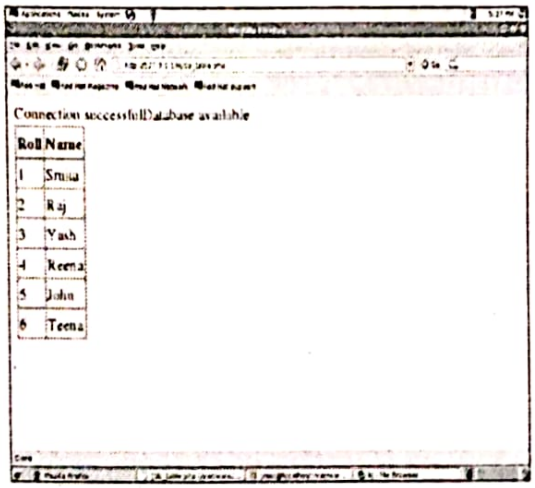
*Program 5.5: Show the Result in Table*

```
echo "<table border ='1' cellspacing ='0' align=left>
<tr>
<th>Roll</th>
<th>Name</th>
</tr>";
while($i<$numrows)
{
   echo "<tr>";
   echo "<td>".mysql_result($result,$i,'roll')."</td>";
   echo "<td>". mysql_result($result,$i,'name')."</td>";
   echo "</tr>";
   $i++;
}
echo "</table>";
```

*Explanation*

1. The code till we execute the query remains the same.
2. The difference to observe here is the insertion of html tags to insert the table. Understand the way html tags are used in while loop with echo statements.

*Output*



*Program 5.6: Execute Insert Statement*

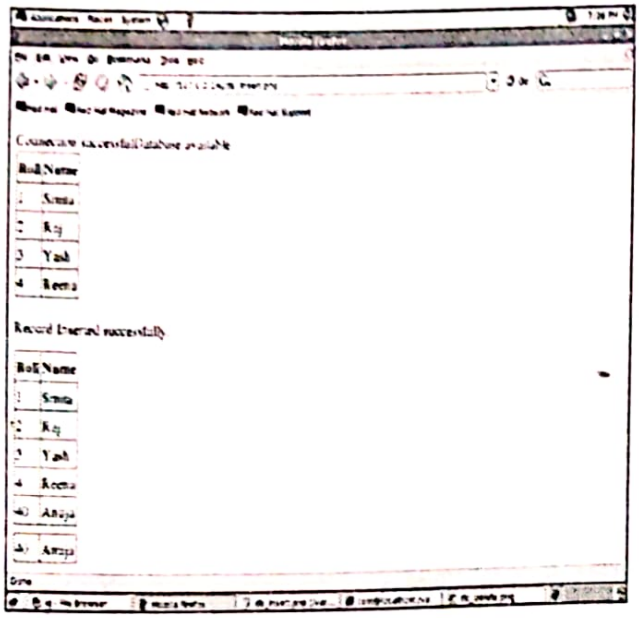
```
<?php
  //Establish connection to MySQL Db Engine
    $user = "root";
    $pwd = "";
    $host = "localhost";
    $dbcon = @mysql_connect($host,$user,$pwd);//or die("Connection
// Failure");
$database = "php_data";
// ************ CHECKS THE CONNECTION STATUS ********
if(!$dbcon)
{
  echo "Connection Failure.";exit();
}
else
  echo "Connection successful";
// ************** OPEN THE DATABASE *****************
$status = mysql_select_db($database,$dbcon);
  if(!$status)
    echo "Database not selected";
  else
    echo "Database available";
echo "<br>";
//************** EXECUTE THE QUERY *****************
$query = "select * from student;";
$result = mysql_query($query);
$numrows = mysql_numrows($result);
$i=0;
// ************ SHOW THE RESULT IN TABLE BEFORE INSERT***************//
echo "<table border ='1' cellspacing ='0' align = left>
<tr>
<th>Roll</th>
<th>Name</th>
</tr>";
while($i<$numrows)
{
  echo "<tr>";
  echo "<td>".mysql_result($result,$i,'roll')."</td>";
  echo "<td>".mysql_result($result,$i,'name')."</td>";
```

```
    echo "</tr>";
    $i++;
  }
  echo "</table>";
  //********************** EXECUTE INSERT QUERY **********
  $query_insert = "insert into student values (40,'Anuja');";
  @mysql_query($query_insert) or die("Query execution failed.");
  echo "<br><br><br><br><br><br><br><br><br> <br> Record Inserted
  successfully.";
  // ************ SHOW THE RESULT IN TABLE AFTER INSERT***************//
  $query = "select * from student";
  $result = mysql_query($query);
  $numrows = mysql_numrows($result);
  $i=0;
  echo "<br><br><br>";
  echo "<table border ='1' cellspacing ='0' align=left>
  <tr>
  <th>Roll</th>
  <th>Name</th>
  </tr>";
  while($i<$numrows)
  {
    echo "<tr>";
    echo "<td>".mysql_result($result,$i,'roll')."</td>";
    echo "<td>".mysql_result($result,$i,'name')."</td>";
    echo "</tr>";
    $i++;
  }
  echo "</table>";
  // ***************** CLOSE THE CONNECTION **************
  mysql_close($dbcon);
  ?>
```

*Explanation*

1. We establish the connection with MySQL using mysql_connect() function.
2. Connect to the database by using mysql_select_db() function.
3. We execute the insert statement using mysql_query() function. The syntax for the insert statement is same as it is in MySQL. If we try to insert same data again and again and if violates database constraints as Primary key, then mysql_query() fails.
4. Here, we show the data from table two times before and after the insert statement.
5. Close the connection.

*Output*

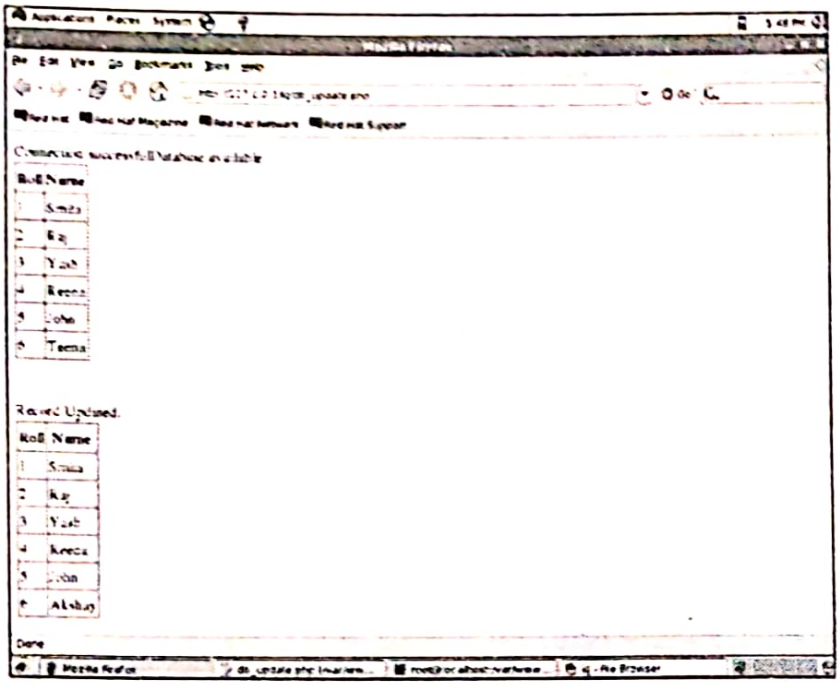


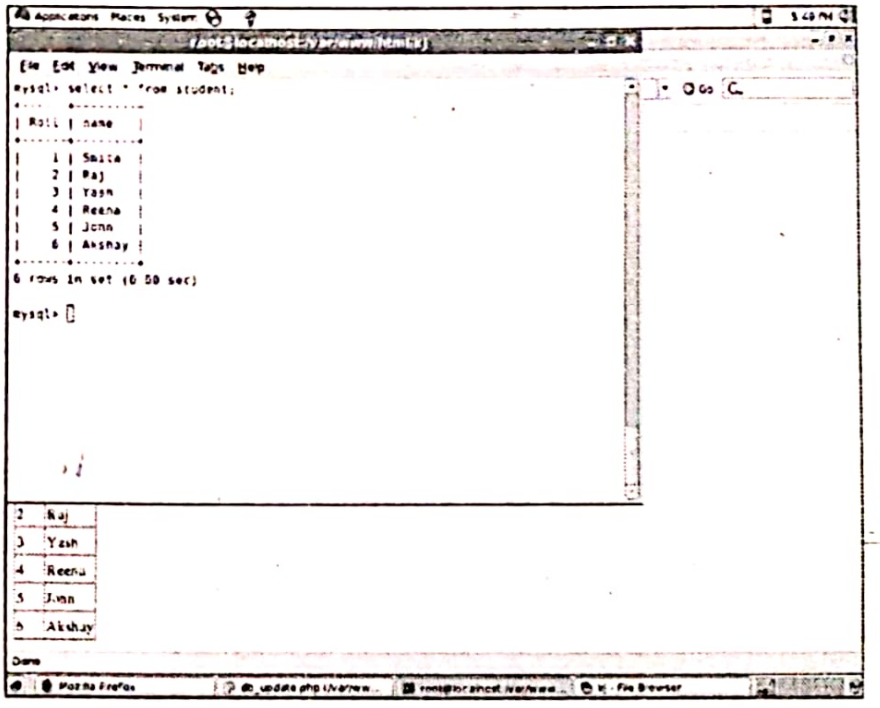
*Program 5.7: Execute Update Statement*

```
<?php
//Establish connection to MySQL Db Engine
   $user = "root";
   $pwd = "";
   $host = "localhost";
   $dbcon = @mysql_connect($host,$user,$pwd);
//or die("Connection Failure");
$database = "php_data";
// ************ CHECKS THE CONNECTION STATUS ********
if(!$dbcon)
{
   echo "Connection Failure.";exit();
}
else
   echo "Connection successful";
// ************** OPEN THE DATABASE ******************
$status = mysql_select_db($database,$dbcon);
   if(!$status)
      echo "Database not selected";
   else
      echo "Database available";
   echo "<br>";
   $query = "Select * from student;";
   $result = mysql_query($query);
   $numrows = mysql_numrows($result);
   $i=0;
```

```
// ************ SHOW THE RESULT IN TABLE before UPDATE
***************//
   echo "<table border ='1' cellspacing ='0' align=left>
<tr>
<th>Roll</th>
<th>Name</th>
</tr>";
while($i<$numrows)
{
   echo "<tr>";
   echo "<td>".mysql_result($result,$i,'roll')."</td>";
   echo "<td>".mysql_result($result,$i,'name')."</td>";
   echo "</tr>";
   $i++;
}
echo"</table>";echo"<br><br><br><br><br><br>";
//************** EXECUTE THE Update QUERY *****************
   $query_update ="update student set name='Akshay' where roll= 6;";
   @mysql_query($query_update) or die("Query execution failed.");
   echo "<br><br><br><br>Record Updated.";
   $query = "Select * from student;";
   $result = mysql_query($query);
   $numrows = mysql_numrows($result);
   $i = 0;
// ************ SHOW THE RESULT IN TABLE AFTER UPDATE ***************//
echo "<table border ='1' cellspacing ='0' align=left>
<tr>
<th>Roll</th>
<th>Name</th>
</tr>";
while($i<$numrows)
{
   echo "<tr>";
   echo "<td>".mysql_result($result,$i,'roll')."</td>";
   echo "<td>".mysql_result($result,$i,'name')."</td>";
   echo "</tr>";
   $i++;
}
echo "</table>";
// ***************** CLOSE THE CONNECTION **************
mysql_close($dbcon);
?>
```

*Output*




Observe the effect of update operation in MySQL also.

*Explanation*

1. We establish the connection with MySQL using mysql_connect() function.
2. Connect to the database by using mysql_select_db() function.
3. We execute the update statement using mysql_query() function. The syntax for the update statement is same as it is in MySQL.
4. Here, we show the data from table two times before and after the update statement to understand the effect.
5. Close the connection.

*Program 5.8: Execute Delete Statement*

```
<?php
//Establish connection to MySQL Db Engine
$user = "root";
$pwd = "";
$host = "localhost";
$dbcon = @mysql_connect($host,$user,$pwd);
//or die("Connection Failure");
$database = "php_data";
// ************ CHECKS THE CONNECTION STATUS ********
if(!$dbcon)
{
  echo "Connection Failure.";exit();
}
else
  echo "Connection successful";
// ************** OPEN THE DATABASE *****************
$status = mysql_select_db($database,$dbcon);
  if(!$status)
    echo "Database not selected";
  else
     echo "Database available";
         echo "<br>";
//************** EXECUTE THE QUERY *****************
         $query ="select * from student;";
$result = mysql_query($query);
$numrows = mysql_numrows($result);
$i=0;
```

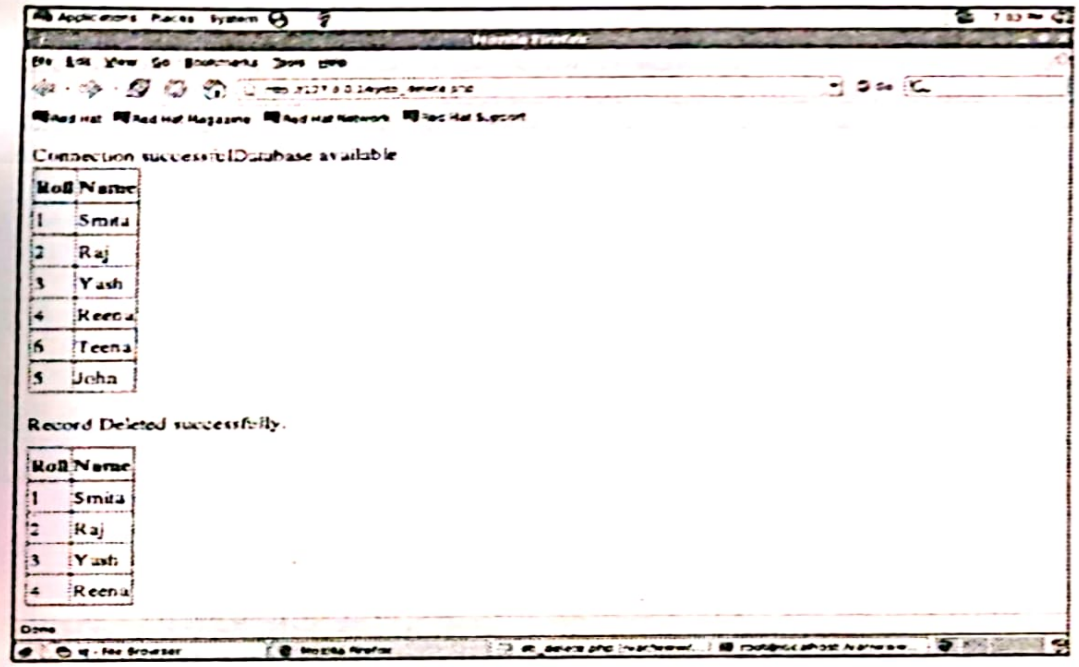
```
// ********** SHOW THE RESULT IN TABLE BEFORE DELETE*************//
echo "<table border ='1' cellspacing ='0' align=left>
<tr>
<th>Roll</th>
<th>Name</th>
</tr>";
while($i<$numrows)
{ echo "<tr>";
  echo "<td>".mysql_result($result,$i,'roll')."</td>";
  echo "<td>". mysql_result($result,$i,'name')."</td>";
  echo "</tr>";
  $i++;
}
echo "</table>";
//******************** EXECUTE DELETE QUERY **********
$query_delete = "delete from student where roll>4;";
@mysql_query($query_delete) or die("Query execution failed.");
echo"<br><br><br><br><br><br><br><br><br>    <br>    Record    Deleted
successfully.";
// ************ SHOW THE RESULT IN TABLE AFTER DELETE***************//
$query = "select * from student;";
$result = mysql_query($query);
$numrows = mysql_numrows($result);
$i=0;
echo "<br><br><br>";
echo "<table border ='1' cellspacing = '0' align=left>
<tr>
<th>Roll</th>
<th>Name</th>
</tr>";
while($i<$numrows)
{
  echo "<tr>";
  echo "<td>".mysql_result($result,$i,'roll')."</td>";
  echo "<td>".mysql_result($result,$i,'name')."</td>";
  echo "</tr>";
  $i++;
}
echo "</table>";
// *************** CLOSE THE CONNECTION **************
mysql_close($dbcon);
?>
```

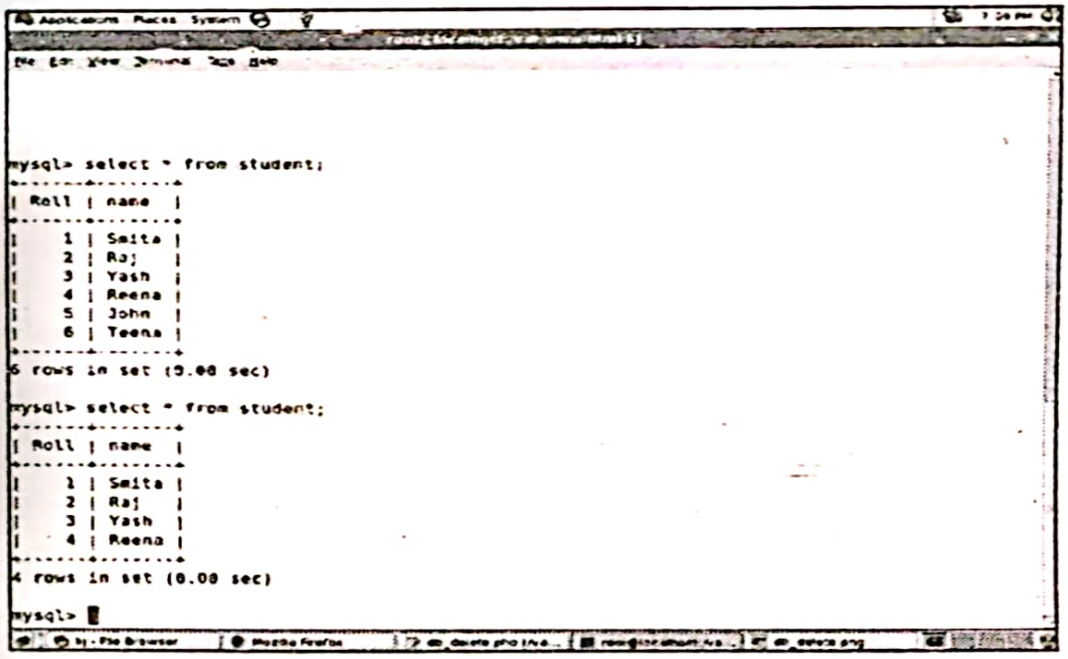
*Output*

Observe the effect

*Explanation*

1. We establish
2. Connect to th




:t of delete operation in MySQL also.

sh the connection with MySQL using mysql_connect() function.

the database by using mysql_select_db() function.

3. We execute the delete statement using mysql_query() function. The syntax for the delete statement is same as it is in MySQL.
4. Here, we show the data from table two times before and after the delete statement to understand the effect.

## 4. PEAR DB Basics

1

October 2018 – 5M
Write a note on PEAR DB basics.

PEAR is short for "PHP Extension and Application Repository" and is pronounced just like the fruit. The purpose of PEAR is to provide:

1. A structured library of open-sourced code for PHP users.
2. A system for code distribution and package maintenance.
3. A standard style for code written in PHP.
4. The PHP Foundation Classes (PFC).
5. The PHP Extension Community Library (PECL).
6. A web site, mailing lists and download mirrors to support the PHP/PEAR community.

PEAR is a community-driven project with the PEAR Group as the governing body.

PEAR::DB is a Database Abstraction Layer for multiple database platforms.

Abstraction is a technique which simplifies something complex. It does this by removing non-essential parts of the object, allowing us to concentrate on the important parts. In the case of database abstraction, the complexities of connecting to a database is hidden behind a standard API, thereby allowing the programmer to connect to many different types of databases without relearning the methods and syntax peculiar to each different type.

This is another approach to access database. PHP has PEAR DB library to

1. Connect to a database,
2. Issue queries,
3. Check for errors, and
4. Transform the results of queries into HTML.

The library is object-oriented. It has a number of methods:

few are **class** methods: `(DB::connect(), DB::iserror())`

while other are **object** methods: `($db->query(), $q->fetchInto()).`

## Data Source Names

A Data Source Name (DSN) is a data structure that contains the information about a specific database that an Open Database Connectivity (ODBC) driver needs in order to connect to it. Included in the DSN, which resides either in the registry or as a separate text file, is information such as the name, directory and driver of the database, and, depending on the type of DSN, the ID and password of the user.

A *Data Source Name* (DSN) is a string that specifies:

1. Where the database is located,
2. What kind of database it is,
3. The username and
4. Password

To use when connecting to the database. DSN is formed as URL-like string.

```
type(dbsyntax)://username:password@protocol+hostspec/database
```

The field `type is` mandatory.

*Example,*

| | Syntax | Description |
|---|---|---|
| 1. | mysql:///php_data | : only database name |
| 2. | mysql://localhost/php_data | : server name + database name |
| 3. | mysql://user@localhost/php_data | : user+ server name + database name |
| 4. | mysql://user@tcp+localhost/php_data | : user+ protocol+ server name + database name |
| 5. | mysql://user:user123@localhost/php_data | : user+password+ server name + database name |

Note DSN name can be stored in a PHP file and include it wherever required.

## Connecting to the Database

**Apr. 2019 – 5M**
What are the different PEAR DB methods? Explain with an example.

### DB::connect()

After a DSN is created, establish a connection to the database using the connect() method.

It is a class method.

This returns a database object required to use for executing queries.

*Syntax*

```
$db = DB::connect(DSN [, options ]);
```

*Parameters*

| Parameter | Description |
|---|---|
| DSN | Data Source Name created by specifying above discussed options. |
| `options` | `(Optional) its` value can be Boolean, or an array of options settings. It can take any of the following values: |

| Option | Controls |
|---|---|
| persistent | Connection persists between accesses |
| optimize | What to optimize for |
| debug | To display debugging information |

How to use:

```
$db = DB::connect($dsn, array('debug' => 1, 'optimize' =>'portability'));
```

## Checking Errors

### DB_ERROR

This method is used to return if an error occurs.

It is a class method.

It returns true if an error occurs while working with any database object. On error, the program stops executing and error message is displayed.

**How to use**

```
$database = DB::connect($datasource);
if(DB::isError($database))
{
  die($database->getMessage());
}
```

## Issuing a Query

**query()**

Using this method, SQL query can be sent to the database.

This is an object method.

On success it returns an object that can be used to access the results.

If SQL statements as INSERT, UPDATE, DELETE are executed successfully then the function returns DB_OK constant.

**How to use**

```
$result = $db->query(sql);
```

## Fetching Results from a Query

PEAR DB library has two methods for fetching data from a query result object.

1. One returns an array corresponding to the next row.
2. Store the row array into a variable passed as a parameter.

**1. Returning the Row**

***fetchRow()***

This method on a query result returns an array of the next row of results.

*Syntax*

```
$row = $result->fetchRow([ mode ]);
```

*Parameters*

| Parameter | Description |
|---|---|
| mode | (optional) It controls the format of the result array returned. |

It's values are:

i. DB_FETCHMODE_ORDERED: Default value

ii. DB_FETCHMODE_ASSOC: It creates an array whose keys are the column names and whose values are the values from those columns.

ON success, it returns an array of data.

It returns NULL if there is no more data,

On error, it returns DB_ERROR.

*How to use*

```
while($row = $result->fetchRow())
{
  if(DB::isError($row))
  {
    die($row->getMessage());
  }
}
```

**2. Storing the Row**

***fetchInto()***

This method also returns the next row, but stores it into the array variable passed as a parameter.

*Syntax*

```
$var = $result->fetchInto(array, [mode]);
```

*Parameters*

| Parameter | Description |
|---|---|
| array | array variable into which result will be stored. |
| mode | (optional) It controls the format of the result array returned. It is same as in fetchRow(). |

It returns NULL if there is no more data.

On error it returns DB_ERROR.

*How to use*

```
while($var = $result->fetchInto($row))
{
  if(DB::isError($var))
  { die($var->getMessage());
  }
}
```

The row returned is an indexed array, in which the positions in the array correspond to the order of the columns in the returned result.

*Example,*

```
$row = $result->fetchRow();
if(DB::isError($row))
{
  die($row->getMessage());
}
var_dump($row);
array(3)
{
  [0]=>
  string(5) "John"
  [1]=>
  string(4) "TY18"
  [2]=>
string(12) "Bsc Comp Sci"
}
```

If the fetch mode is passed as DB_FETCHMODE_ASSOC, it creates an array whose keys are the column names and whose values are the values from those columns:

```
$row = $result->fetchRow(DB_FETCHMODE_ASSOC);
if(DB::isError($row))
{
  die($row->getMessage());
}
var_dump($row);
array(3)
{
  ["Name"]=>
  string(5) "John"
  ["Roll"]=>
  string(4) "TY18"
  ["Degree"]=>
  string(12) "Bsc Comp Sci"
}
```

## Free up Memory

*free()*

This function is used to free up the memory consumed by the query. The memory is returned to the operating system. This is required since a query result object usually holds all the rows returned by the query.

**How to use**

```
$result->free();
```

If the programmer does not take this step to free the memory then even free() is automatically called on all queries when the PHP script ends.

### Disconnecting the Database

***disconnect()***

This method is used to disconnect from the database.

It is an object method.

**How to use**

```
$database->disconnect();
```

If the programmer does not explicitly disconnect the database then even all database connections are disconnected when the PHP script ends.

## 5. Advanced Database Techniques

1

**Apr. 2019 – *5M***
Write any three advanced database techniques.

*PEAR DB library has functions which help to*

1. Fetching result rows,
2. A unique row ID system and
3. Prepare/execute steps to improve the performance of repeated queries.

**1. Placeholders**

If the programmer wants to accept certain values from the user and use them in any query as the input values (insert query), the PEAR DB library supports different placeholders.

*Syntax*

```
$result = $db->query(SQL, values);
```

*Parameters*

| Parameter | Description |
|---|---|
| SQL | Pass the query() function SQL with in place of specific values. |
| values | It is an array of values to insert into the SQL. |

*Example,*

This code snippet inserts three rows into the Student table.

```
$students = (array(array(TY10, 'Amit'),
             array(TY20,'Pooja'),
             array(TY105,'Aamir')
             );
foreach($students as $student)
{
 $database->query('INSERT INTO students (Roll,Name) VALUES
(?,?)', $student);
}
```

There are three characters that you can use as placeholder values in an SQL query:

?: A string or number, which will be quoted if required (recommended).

|: A string or number, which will never be quoted.

&: A filename, the contents of which will be included in the statement.

**2. Prepare/Execute**

prepare() , execute(), and executeMultiple()

Sometimes the same query can be executed repeatedly. In this case, it can be compiled once and execute it multiple times.

***i. prepare()***

This method is used to compile the query.

*Syntax*

```
$result= $database->prepare(SQL);
```

*Parameter*

| Parameter | Description |
|---|---|
| SQL | The query statement to be compiled. |

On success, it returns a compiled query object.

### ii. *execute()*

This method is used to fill in any place holders in the query and send them to the database.

*Syntax*

```
$result = $database->execute(compiled, values);
```

*Parameters*

| Parameter | Description |
|---|---|
| compiled | It is the object returned by prepare() method. |
| values | It is an array containing the values for the placeholders in the query. |

On success, it returns a query response object.

On error, it returns DB_ERROR.

*Example,*

```
$students = (array (array(TY10, 'Amit'),
             array (TY20,'Pooja'),
             array (TY105,'Aamir')
             );
$compile_result = $query-> prepare('INSERT INTO
students (Roll, Name) VALUES (?,?)');
$database->query('INSERT INTO students (Roll,Name) VALUES
(?,?)', $student);
foreach($students as $student)
{
    $database ->execute ($compile_result, $student);
}
```

Here, we compile insert statement and then in a loop we execute it number of times.

### iii. *executeMultiple()*

This method takes a two-dimensional array of values to insert values in the query.

*Syntax*

```
$responses = $db->executeMultiple(compiled, values);
```

*Parameters*

| Parameter | Description |
|---|---|
| compiled | It is the object returned by prepare() method. |
| values | It is an array containing the values for the placeholders in the query. |

The values array should be numerically indexed from 0 and have values that are arrays of values to insert.

The compiled query is executed once for every entry in values, and the query responses are collected in `$responses`.

*Example,*

So the code in previous *example* can be written as follows:

```
$students = (array(array(TY10, 'Amit'),
            array(TY20,'Pooja'),
            array(TY105,'Aamir')
            );
$compile_result = $query-> prepare('INSERT INTO students
(Roll, Name) VALUES(?,?)');
$database->query('INSERT INTO students(Roll,Name) VALUES
(?,?)', $student);
$database->insertMultiple($compiled, $students);
```

### 3. Shortcuts

PEAR DB provides a number of methods that perform a query and fetch the results in one step: getOne(), getRow(), getCol(), getAssoc(), and getAll(). All of these methods permit placeholders.

The getOne() method fetches the first column of the first row of data returned by an SQL query:

| | Function | Syntax | Description |
|---|---|---|---|
| 1. | `getOne()` | $value = $db->getOne(SQL [, values ]); | It returns the first column of the first row of data returned by an SQL query. |
| 2. | `getRow()` | $row = $db->getRow(SQL [, values ]]); | It returns the first row of data returned by an SQL query. |
| 3. | `getCol()` | $col = $db->getCol(SQL [,column [, *values* ]); | It returns a single column from the data returned by an SQL query. |
| 4. | `getAssoc()` | $result = $db->getAssoc (SQL,$force_array,params,fetchmode) | Runs the query provided and puts the entire result set into an associative array then frees the result set. |
| 5. | `getAll()` | $all = $db->getAll(SQL[, *values* [, *fetchmode* ]]); | It returns an array of all the rows returned by the query. |

*Examples,*

i. getOne()

```
$ans = $db->getOne("SELECT Roll FROM students");
if(DB::isError($ans))
{
```

```
  die($ans->getMessage());
}
echo "The Roll numbers start with: $ans";
```

Here, we fetch the first roll number of the student table and store in $ans.

ii. getRow()

```
list($roll, $name) = $db->getRow("SELECT roll, name FROM
students where marks=(select max(marks) from students)");
echo "($roll, $name got the highest marks)";
```

Here, the SQL statement passed to the getRow() function will return the single row.(We assume only one student got the highest marks.)

iii. getCol()

```
$Names = $db->getAll("SELECT Name FROM students");
foreach($Names as $Name)
{
  echo "$Name\n";
}
```

This will result in all the names from student table.

iv. getAll()

```
$results = $db->getAll("SELECT Roll, Name FROM students ORDER
BY Roll");
foreach($results as $result)
{
  echo "$result[0]. $result[1]";
}
```

On error, all these methods return DB_ERROR.

## 4. Details about a Query Response

PEAR DB library has different functions to provide with information on a query result object. The following table gives details about these functions:

| | Function | Syntax | Description |
|---|---|---|---|
| 1. | numRows () | $count = $response->numRows() | Outputs the number of rows returned from a SELECT query. |
| 2. | numCols() | $count = $response->numCols(); | Outputs the number of columns returned from a SELECT query. |
| 3. | affectedRows() | $count = $response->affectedRows(); | Outputs the number of rows affected by an INSERT, DELETE, or UPDATE operation. |
| 4. | tableInfo() | $info = $response->tableInfo(); | Outputs detailed information on the type and flags of fields returned from a SELECT operation. |

## 5. Sequences

Some of the RDBMS supports the feature to assign unique row Ids. (MySQL's AUTO_INCREMENT).

PEAR DB library has sequences which can be an alternative to database-specific ID assignment.

*nextID()*

This method returns the next ID for the given sequence.

*Syntax*

```
$id = $db->nextID(sequence);
```

In general, there is one sequence per table for which you want unique IDs. In student table, roll number can be given a unique id.

```
$students = (array(array('Lucky', 'TY'),
             array('Smita',TY),
             array('Satish', 'SY')
             );
foreach($students as $student)
{
  $id = $db->nextID('students');
  splice($student, 0, 0, $id);
  $db->query('INSERT INTO students (name, class, roll) VALUES
(?,?,?)', $student);
}
```

A sequence is actually a table in the database which keeps track of the last-assigned ID. You can explicitly create and destroy sequences with the createSequence() and dropSequence() methods.

```
$res = $db->createSequence(sequence);
$res = $db->dropSequence(sequence);
```

On success, they return the result object from the create or drop query.

On error, DB_ERROR is returned.

## 6. Metadata

Metadata means data about the data. Databases, tables store the user information. But information about databases, tables, users, etc., is metadata.

*getListOf()*

This method is used to know the information on available databases, users, views, and functions.

*Syntax*

```
$data = $db->getListOf(what);
```

*Parameter*

| Parameter | Description |
|---|---|
| what | This is the value identifying the database feature to list. |

Return value is database dependent, i.e., tables, databases, users, view, functions.

*Example,*

```
<?php
$data = $db->getListOf('tables');
if(PEAR::isError($data))
{
   die($data->getMessage());
}
echo($data);
?>
$dbs = $db->getListOf("databases");
```

## 7. Transactions

*commit(), rollback()*

Some RDBMSs support *transactions*, to commit a series of database changes all applied at once or rolled back, i.e., discarded, with the changes not applied to the database.

PEAR DB library has the commit() and rollback() methods to do the same with transactions.

*How to use*

```
$res = $db->commit();
$res = $db->rollback();
```

If the backend database do not support these features then the methods return DB_ERROR.

*Example,*

*commit()*

The COMMIT() is the transactional function used to save changes invoked by a transaction to the database.

The COMMIT function saves all transactions to the database since the last COMMIT or ROLLBACK command.

```
<?php
$db->autoCommit(false);
$db->query('CREATE TABLE sample(col integer)');
$db->commit();
$db->query('INSERT INTO sample(col)VALUES(11)');
$response = & $db->query('SELECT col FROM sample');
if(DB::isError($response))
{
  echo $response->getMessage();
}
whil($response->fetchInto($row, DB_FETCHMODE_ORDERED))
{
   echo $row[0]. "\n";
}
$response->free();
$db->query('DROP TABLE sample');
$db->commit();
?>
```

*rollback()*

The ROLLBACK() is the transactional function used to undo transactions that have not already been saved to the database.

The ROLLBACK command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

```
<?php
$db->autoCommit(false);
$db->query('INSERT INTO sample(col)VALUES(11)');
$res = & $db->query('SELECT b FROM blue');
if(DB::isError($res))
{
```

```
    echo $res->getMessage(), "\n";
}
while($res->fetchInto($row, DB_FETCHMODE_ORDERED))
{
    if($row[0] == 'problem')
    {
        $db->rollback();
    }
}
$res->free();
$db->query('DROP TABLE sample');
$db->commit();
?>
```

## 6. Simple Applications

We consider a simple application of Employee-Department

Employee: Department shares one: many relationship.

The database schema is as follows:

**Employee:**

| Field | Data Type |
|---|---|
| Eno | Int (primary key) |
| Ename | Varchar |
| Salary | Int |
| Dno | Int (Foreign Key) |

**Department:**

| Field | Data Type |
|---|---|
| Dno | Int |
| Dname | varchar |

Overall, we develop 4 different web pages:

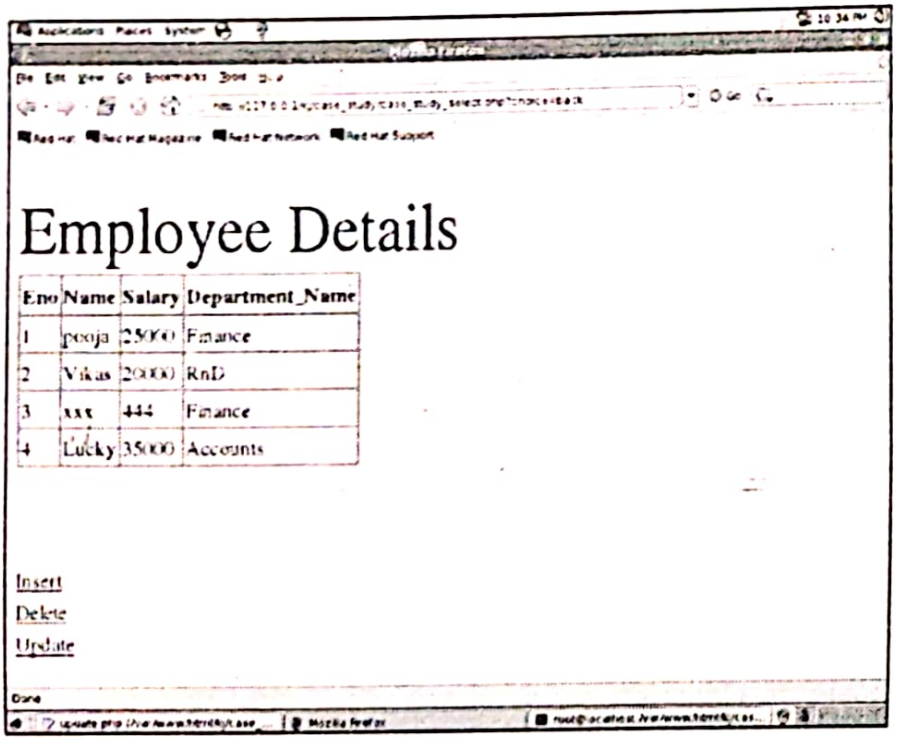
1. Main web page: Employee details
2. Record Insertion form
3. Record Deletion form
4. Record Updating form

We develop a main web page on which, we display the contents of the Employee table. We display the department name from the Department table along with Employee details.

*The PHP code for this page is as follows:*

```
<?php
     //Establish connection to MySQL Db Engine
     $user = "root";
     $pwd ="";
     $host = "localhost";
     $dbcon = @mysql_connect($host,$user,$pwd);
   //or die("Connection Failure");
   $database = "php_data";
// ************ CHECKS THE CONNECTION STATUS ********
if(!$dbcon)
{
   echo "Connection Failure.";exit();
}
// ************** OPEN THE DATABASE *****************
$status = mysql_select_db($database,$dbcon);
   if(!$status)
      echo "Database not selected";
echo "<br>";
echo "<font bold size = 16>Employee Details</font>";
//************** EXECUTE THE QUERY *****************
$query ="select Eno, Ename, Salary, Department.Dname from Employee,
Department where Employee.Dno = Department.Dno;";
$result = mysql_query($query);
$numrows = mysql_numrows($result);
$i = 0;
// ************ SHOW THE RESULT IN TABLE***************
echo "<table border ='1' cellspacing ='0' align=left>
<tr>
<th>Eno</th>
<th>Name</th>
<th>Salary</th>
<th>Department_Name</th>
</tr>";
while($i<$numrows)
{
```

```
    echo "<tr>";
    echo "<td>".mysql_result($result,$i,'Eno')."</td>";
    echo "<td>".mysql_result($result,$i,'Ename')."</td>";
    echo "<td>".mysql_result($result,$i,'Salary')."</td>";
    echo "<td>".mysql_result($result,$i,'Dname')."</td>";
    echo "</tr>";
    $i++;
}
echo "</table>";
echo "<input type ='hidden' name ='choice' value
      ='view'><br><br><br><br><br><br><br><br><br><br>";
echo "<A HREF ='insert.php?choice = insert'>Insert</A><br>";
echo "<A HREF ='delete.php?choice = delete'>Delete</A><br>";
echo "<A HREF ='update.php?choice = update'>Update</A><br>";
echo "<br>";
// ********** CLOSE THE CONNECTION **********
mysql_close($dbcon);
?>
```




Clicking on the hyperlinks, appropriate pages will be opened to perform insert, update or delete action on the database.

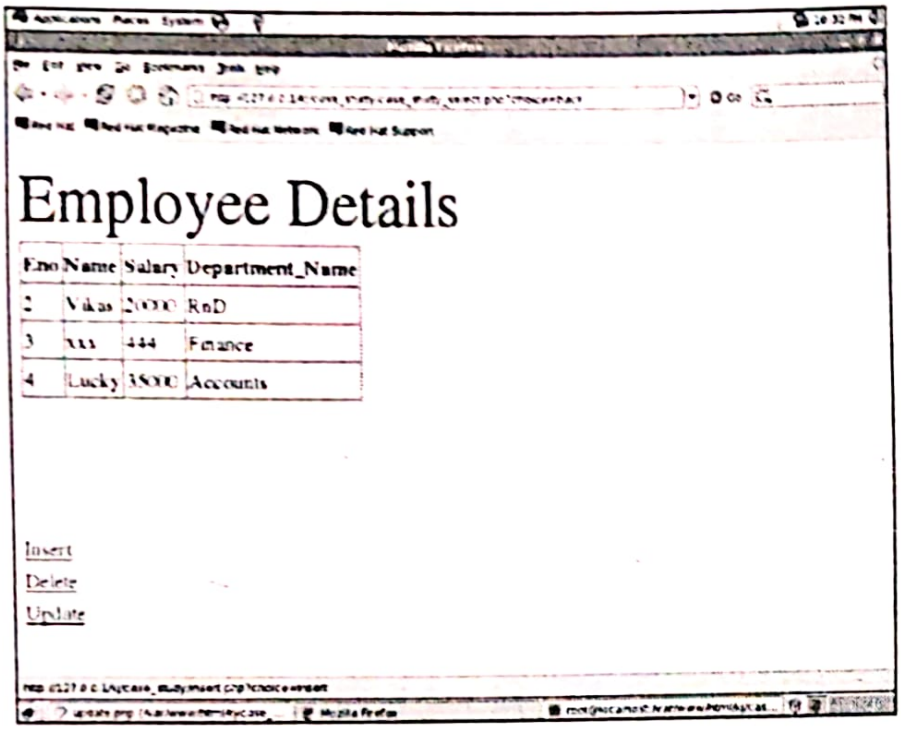
*The code and output windows are shown below:*

## 1. Record Insertion

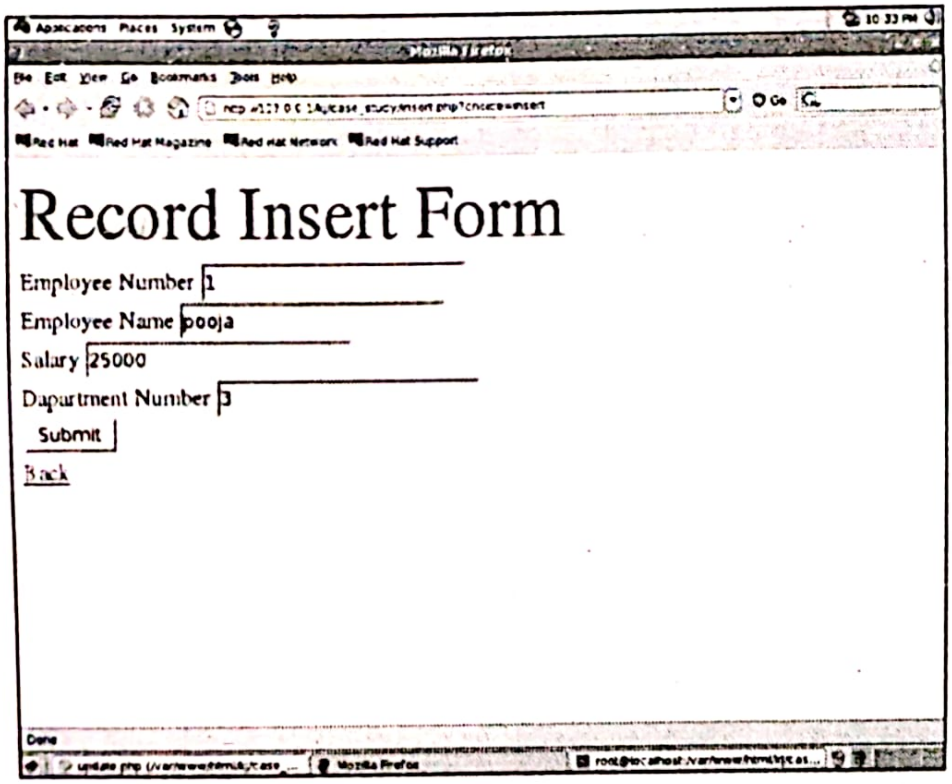
We collect the Employee details from user and insert the record in database.

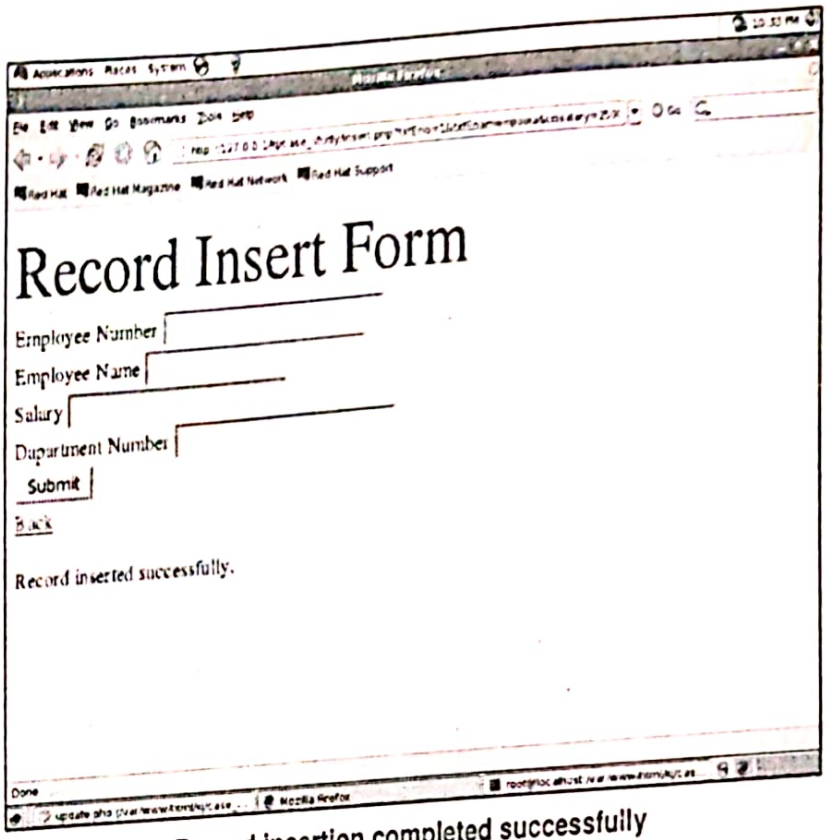
```
<html>
<form>
<form action = "<?php $_SERVER['PHP_SELF']?>" method="GET">
<font size = 8>Record Insert Form</font><br>
Employee Number <Input type ="text" name ="txtEno"><br>
Employee Name <input type ="text" name ="txtEname"><br>
Salary <input type ="text" name ="txtsalary"><br>
Department Number <input type ="text" name ="txtDno"><br>
<input type ="Submit" name ="submit" value ="Submit"><br>
<A HREF ='case_study_select.php?choice = back'>Back</A>
</form>
</html>
<?php
   $user = "root";
   $pwd = "";
   $host = "localhost";
   $dbcon = @mysql_connect($host,$user,$pwd);
//or die("Connection    Failure");
$database = "php_data";
// ************ CHECKS THE CONNECTION STATUS ********
if(!$dbcon)
{  echo "Connection Failure.";
     exit();
}
// ************** OPEN THE DATABASE *****************
$status = mysql_select_db($database,$dbcon);
if(!$status)
  echo "Database not selected";
//************** EXECUTE THE QUERY *****************
$eno = $_GET['txtEno'];
$dno = $_GET['txtDno'];
$ename = $_GET['txtEname'];
$salary = $_GET['txtsalary'];
$query_insert = "insert into Employee
values($eno,'$ename',$salary,$dno);";
$result = mysql_query($query_insert);
if($result)
```

```
echo "Record inserted successfully.";
echo "$cnt";
?>
```

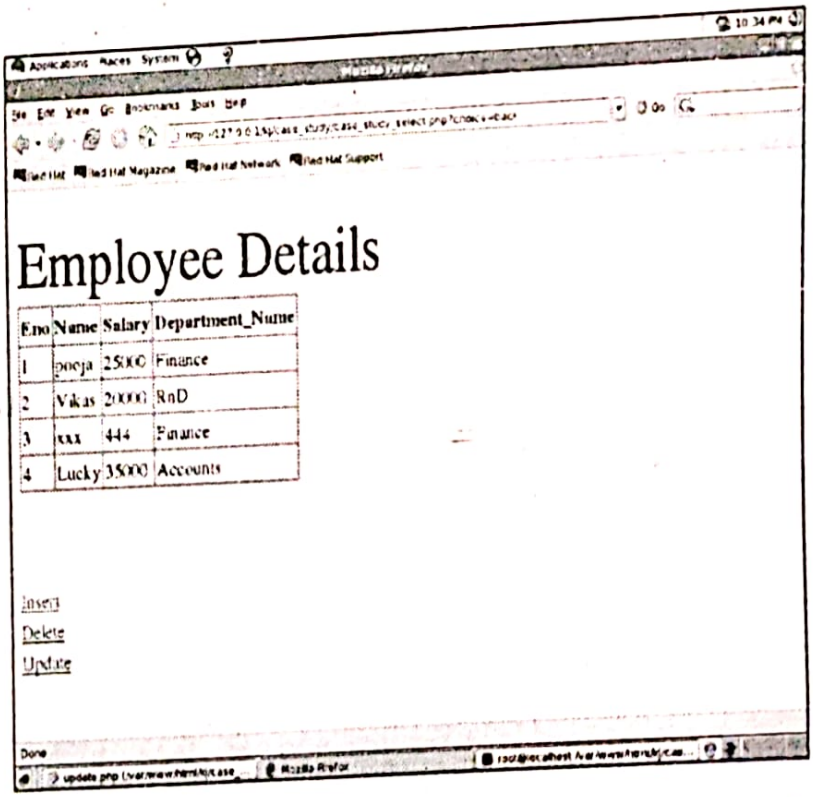



**Employee details before record insertion**






Record insertion completed successfully

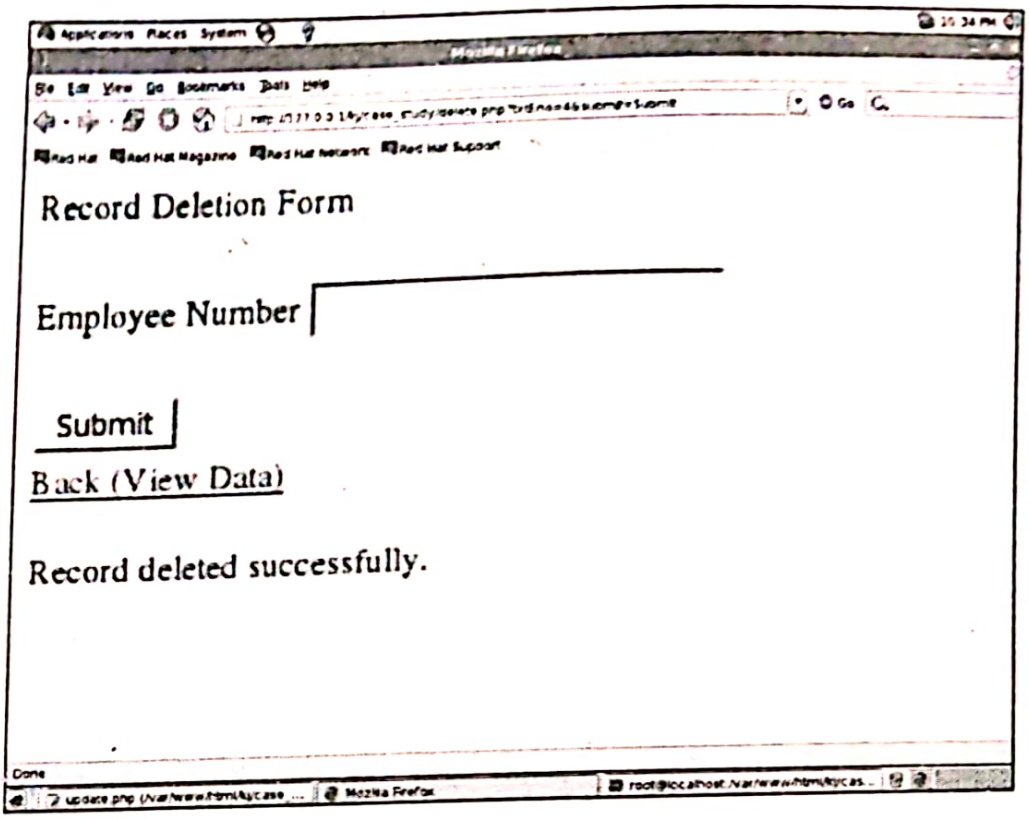



Observe the newly inserted record in the database shown on page1

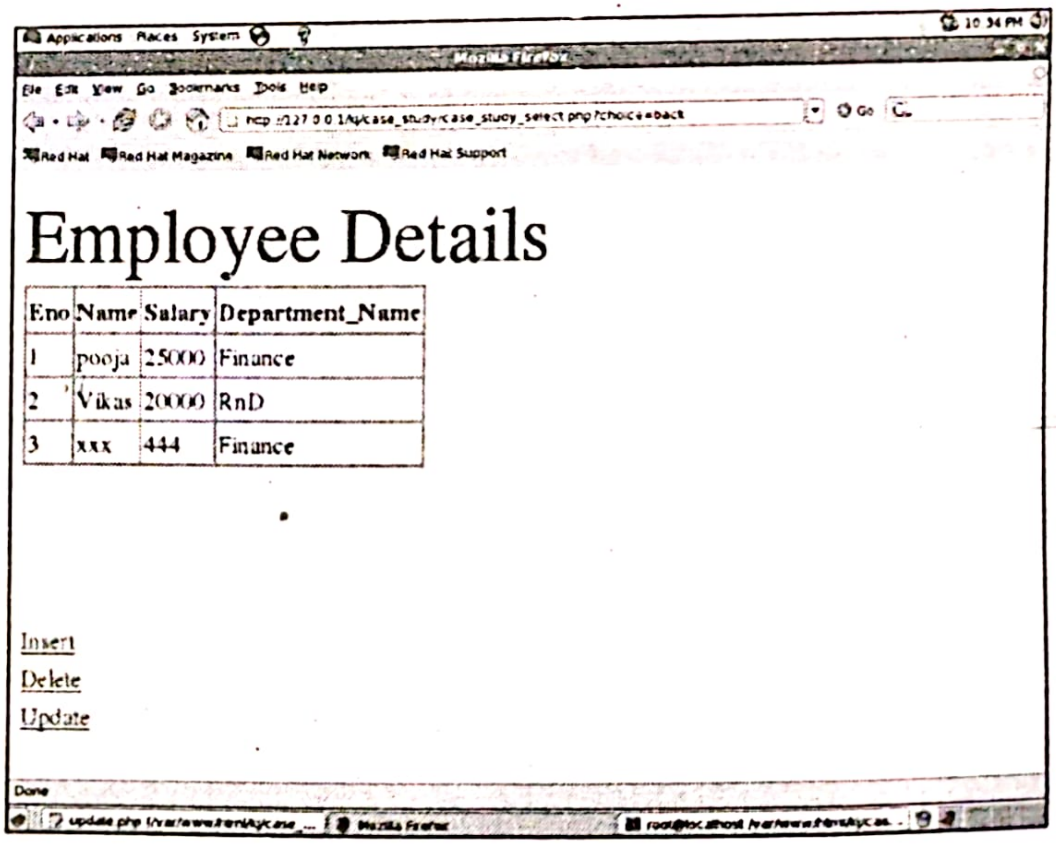
## 2. Record Deletion

We accept the Employee Number from the user and delete that record from the Employee table.

```
<html>
<body>
<form>
<form action = "<?php $_SERVER['PHP_SELF']?>" method = "GET">
  Record Deletion Form<br>
<br>
Employee Number
<Input type = "text" name = "txtEno"><br><br>
<Input type = "Submit" name = "submit" value = "Submit"><br>
<A HREF ='case_study_select.php? choice = back'>
  Back(View   Data)</A>
</form>
</body>
</html>
<?php
$user = "root";
$pwd ="";
$host="localhost";
$dbcon = @mysql_connect($host,$user,$pwd);
//or die("Connection Failure");
$database ="php_data";
// ************ CHECKS THE CONNECTION STATUS ********
if(!$dbcon)
{
echo "Connection Failure.";exit();
}
// ************** OPEN THE DATABASE *****************
$status = mysql_select_db($database,$dbcon);
if(!$status)
  echo "Database not selected";
//************** EXECUTE THE QUERY *****************
$eno = $_GET['txtEno'];
$query_delete ="delete from Employee where Eno = $eno;";
$result = mysql_query($query_delete);
if($result)
echo "Record deleted successfully.";
?>
```




Observe the effect of record deletion on the first page.

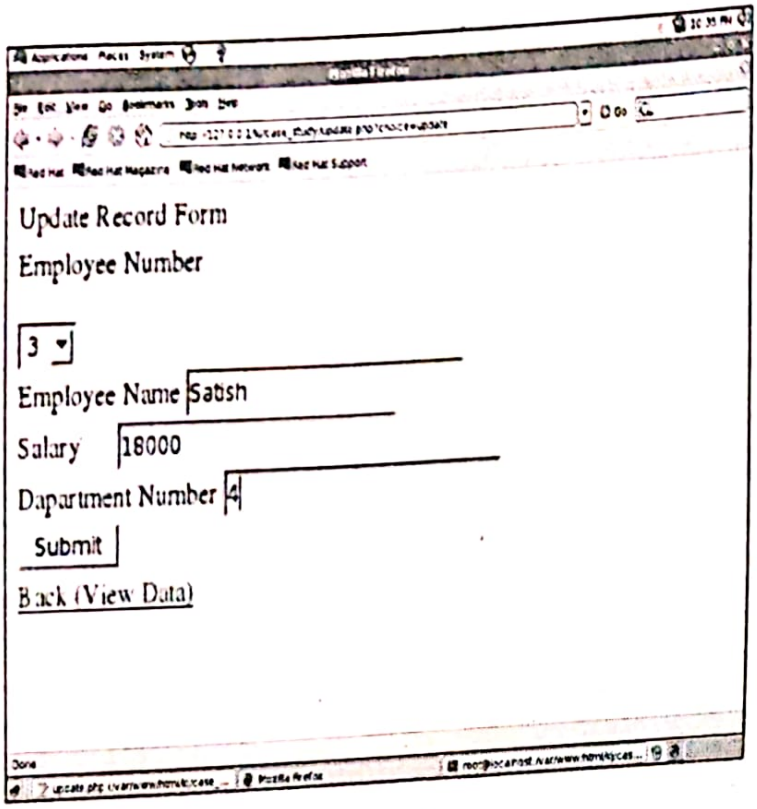


### 3. Record Updation

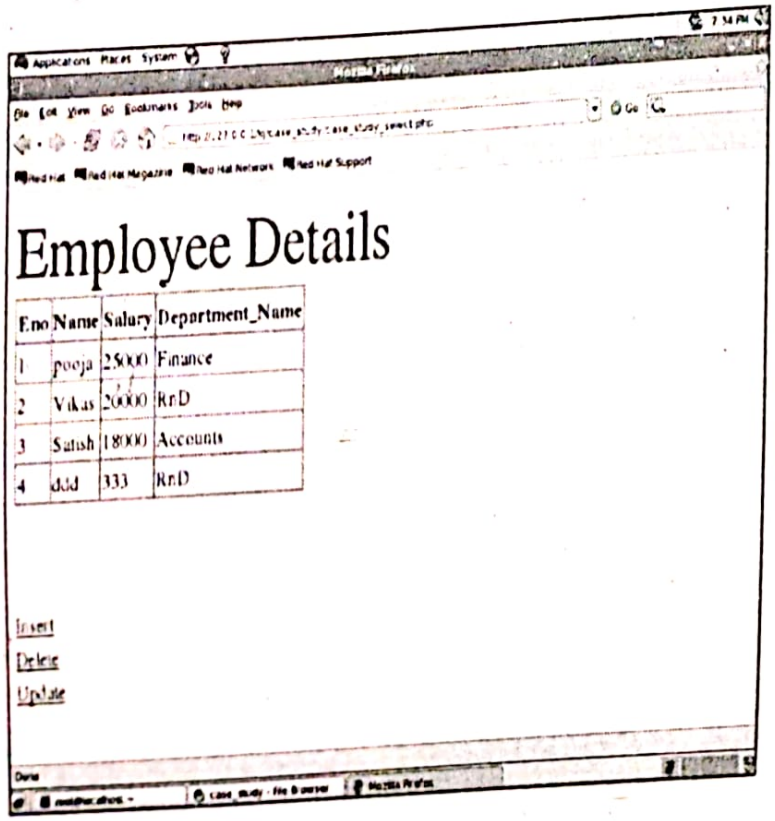
Here, we display the existing employee numbers in a list box. The user will select the employee number and then will provide new values for record updation.

```
<html>
<body>
<form>
  <form action ="<?php $_SERVER['PHP_SELF']?>" method="GET">
    Update Record Form<br>
    Employee Number
</form>
</body>
</html>
<?php
    $user = "root";
    $pwd = "";
    $host = "localhost";
    $dbcon = @mysql_connect($host,$user,$pwd);
//or die("Connection Failure");
    $database = "php_data";
// ************ CHECKS THE CONNECTION STATUS ********
if(!$dbcon)
{
  echo "Connection Failure.";exit();
}
// ************** OPEN THE DATABASE *****************
$status = mysql_select_db($database,$dbcon);
if(!$status)
  echo "Database not selected";
//************** EXECUTE THE QUERY *****************
$query = "select Eno from Employee;";
$result = mysql_query($query);
$numrows = mysql_numrows($result);
$i = 0;
```

```
// ************ SHOW THE RESULT IN TABLE**************
echo "<select name='EmpNo'>";
while($i<$numrows)
{
   echo "<option>".mysql_result($result,$i,'Eno')."</option>";
   $i++;
}
echo "<Input type='hidden' name='t1'> ";
echo "<br> Employee Name  <input type ='text' name
       ='txtEname'><br>";
echo "Salary      <input type ='text' name
     ='txtsalary'><br>";
echo "Dapartment Number <input type ='text' name
     ='txtDno'><br>";
echo "<input type = 'Submit' name ='submit' value
     ='Submit'><br>";
echo "<A HREF ='case_study_select.php?choice=back'>Back(View
Data)</A>";
$eno = $_GET['EmpNo'];
$dno = $_GET['txtDno'];
$ename = $_GET['txtEname'];
$salary = $_GET['txtsalary'];
echo $eno;
$query_update = "update employee set Ename
='$ename',salary=$salary,Dno = $dno where Eno = $eno;";
$result = mysql_query($query_update);
if($result)
echo "Record updated successfully.";
mysql_close($dbcon);
?>
```




Observe the effect of updation on main page, in Employee Details form.



# SUMMARY

- **PHP** have good working support for different databases.
- There are two ways to access databases from PHP.
  - i. Use a database-specific extension.
  - ii. Use the database-independent PEAR DB library.
- **Database-specific extension:** Different functions available to work with database:
  - i. **mysql_connect():** Create a connection to the database.
  - ii. **mysql_fetch_array():** This function is used to fetch a result row as an associative array, a numeric array, or both.
  - iii. **mysql_result():** This function is used to get result data. It retrieves the values stored in one cell from result set.
  - iv. **mysql_query():** A query or command is sent to a MySQL connection.
  - v. **mysql_close():** To close the connection before system does.
  - vi. **mysql_num_rows ():** This function is used to get the number of rows i.e. the count in result set.
  - vii. **mysql_affected_rows():** This function is used to get number of affected rows in the last MySQL operation. The operations are delete, update, insert etc.
- **PEAR DB basics :** PEAR DB library is object-oriented.
- **Advanced Database Techniques**
  - i. PEAR DB library have functions which help to
    - a. Fetching result rows,
    - b. A unique row ID system and
    - c. Prepare/execute steps to improve the performance of repeated queries is method.
  - ii. The PEAR DB library supports different placeholders. ?,! and &
  - iii. Prepare(): This method is used to compile the query.
  - iv. Execute(): This method is used to fill in any placeholders in the query and send to the database.
  - v. Transactions are handled with commit(), rollback() functions.

# Exercises

**A. Answer the following questions:** **[1 Mark]**

1. Explain the purpose and syntax of the following functions: (1 per function)
   a. mysql_connect()
   b. mysql_close()
   c. mysql_query()
   d. mysql_fetch_array()
   e. mysql_result()
   f. mysql_num_rows ()
   g. mysql_affected_rows()
2. How to find out number of rows returned by the resultset? Explain with example.
3. What happens if we try to insert the record in a table which is already present?
4. What is PEAR DB Library?
5. Explain advantages and disadvantages of using PEAR DB to access database.
6. What is DSN? Explain with example.
7. How to connect to the database using PEAR DB?
8. How the SQL statement is executed in PEAR DB?
9. With example explain fetchRow().
10. What is the difference between free() and disconnect() method?
11. Which are the different placeholders used in SQL query?
12. With example explain the use of prepare( ) and execute( ) method.

13. With example explain the use of (1 each)

a. getOne() b. getRow()

c. getCol() d. getAssoc()

e. getAll()

14. Explain the use of following methods (1 each)

a. nextID() b. getListOf()

c. commit() d. rollback()

**B. Answer the following questions: [5 Marks]**

1. What are the different steps required to connect to the databases through PHP? Explain.
2. What are the different ways to access database from PHP? Explain.
3. What are the advantages and disadvantages of using database extension to get connected to the database?
4. Which are the different class methods and object methods available in PEAR DB library? Explain.
5. How to insert the values accepted from user in the query? Explain with example.
6. Which are the different PEAR DB methods to provide information on a query result object? Explain in detail with example.
7. What is a sequence? How PHP supports sequence? Explain with example.

# Questions Asked in Previous Year Exams

## 1 Mark

1. Which of the following functions is used to compile the query? [April 2019]

   i. **prepare()** ii. execute()

   iii. executemultiple() iv. none of these

## 5 Marks

1. What are the different PEAR DB methods? Explain with an example. [April 2019]

2. Consider the following relational database: [April 2019]

   Project(P_group_no, project_title)

   Student(seat_no, name, class, P_group_no)

   Write a PHP script to accept project title and display list of students those who are working in particular project.

3. Write a PHP program to accept student rno, name, on page1.php and marks of three subjects on page2.php and display students all information page3.php . [April 2019]

4. Write any three advanced database techniques. [April 2019]

5. Write a note on PEAR DB basics. [October 2018]

6. Consider the following entities and their relationship: [October 2018]

   Student (studid, name, class)

   Competition (cno, cname, type)

   Relation between student and competition is many to many with extra attribute rank and year. Write a PHP script to accept competition name from user and print the student name who secures first rank in that competition.