MODULE-05

Accessing MySQL –Using MySQL Client and Using php MyAdmin, MySQL Commands, Using **PHP with MySQL:** PHP MySQL Functions, Connecting to MySQL and Selecting the Database, Executing Simple Queries, Retrieving Query Results, Counting Returned Records, Updating Records with PHP.

ACCESSING MYSQL

USING MYSQL CLIENT AND USING PHPMYADMIN

What is a MySQL client?

- MySQL client is a common name for tools that are designed to connect to MySQL Server.
- The MySQL client, also known as the MySQL command-line interface (CLI), is a powerful tool for interacting with MySQL databases.
- Client programs are used to send commands or queries to the server and allow managing data in the databases stored on the server.
- It is the tool that allows sending commands to MySQL Server from the command line.
- Accessing MySQL typically involves using a client application or a programming language that supports MySQL connections.

USES OF MYSQL CLIENT SERVER : Here are some common uses of the MySQL client:

1. **Executing SQL Queries:** The primary use of the MySQL client is to execute SQL queries directly against a MySQL database. This allows users to retrieve, manipulate, and manage data stored in the database.
2. **Database Administration:** Administrators use the MySQL client to perform various database administration tasks, such as creating and dropping databases, creating and altering tables, and managing users and permissions.
3. **Importing and Exporting Data:** The MySQL client allows users to import data into MySQL databases from external sources, such as CSV files or other database systems. It also provides functionality for exporting data from MySQL databases to various formats.

4. **Database Backup and Restoration:** Administrators use the MySQL client to perform database backups and restorations. This involves dumping the contents of a database to a file for backup purposes and restoring a database from a backup file when necessary.
5. **Database Monitoring and Troubleshooting:** The MySQL client provides tools for monitoring database performance and diagnosing issues. Users can view server status, check query execution times, and identify slow queries that may be impacting performance.

ACCESSING MYSQL DATABASE CLIENT SERVER

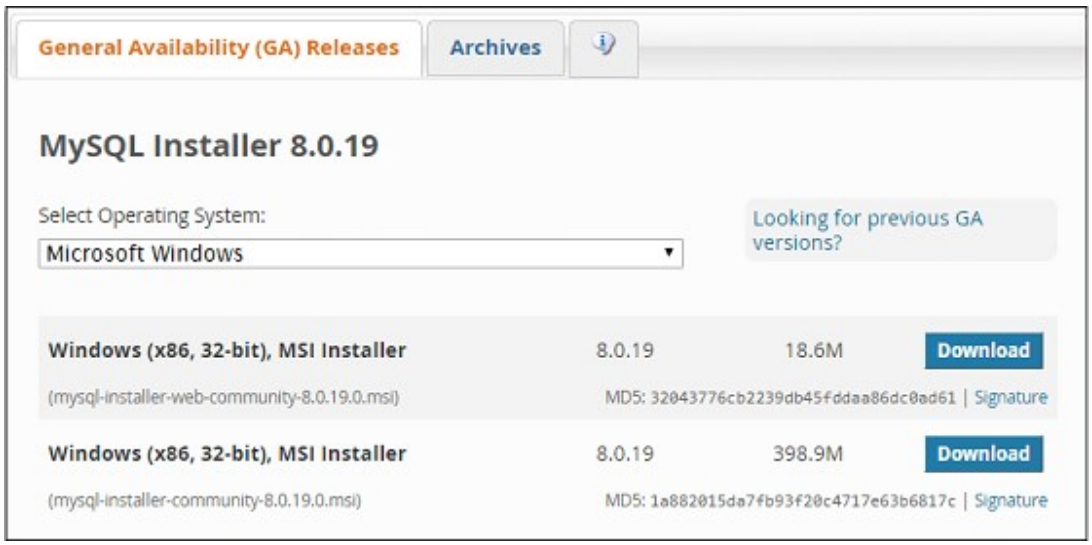
To use MySQL Client to interact with a MySQL server, follow these steps:

Step1:Install MySQL Client:

- If you haven't already installed MySQL Client, you can do so by downloading and installing MySQL Community Server from the official MySQL website: https://dev.mysql.com/downloads/mysql/
- Follow the installation instructions specific to your operating system.

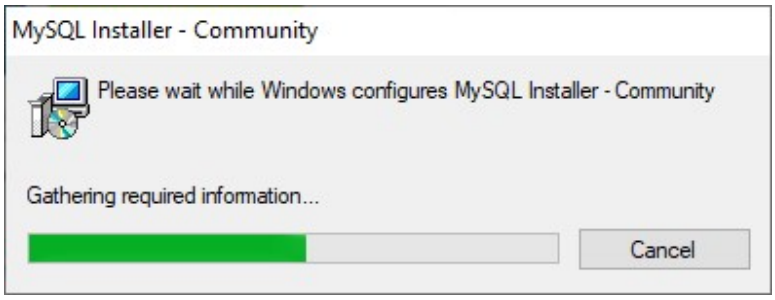
Step2: Go to the official website of MySQL and download the community server edition software. Here, you will see the option to choose the Operating System, such as Windows.

Step3: Next, there are two options available to download the setup. Choose the version number for the MySQL community server, which you want. If you have good internet connectivity, then choose the mysql-installer-web-community. Otherwise, choose the other one.

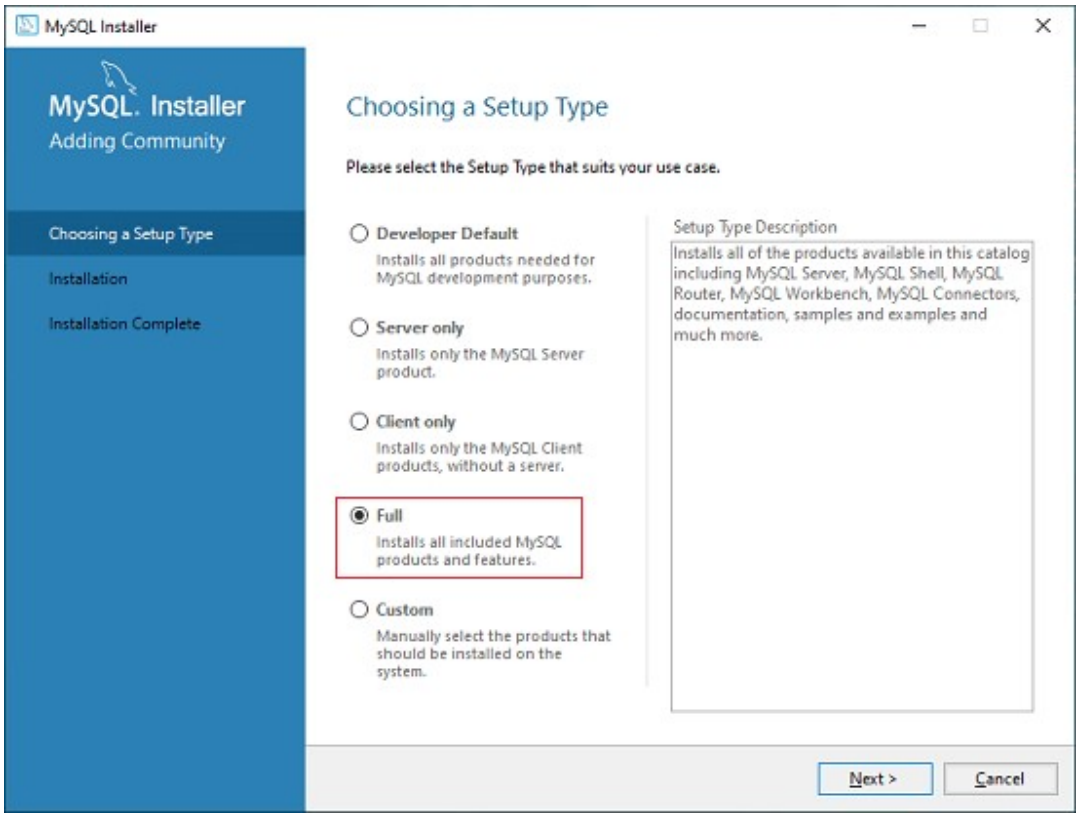


Installing MySQL on Windows

Step 1: After downloading the setup, unzip it anywhere and double click the MSI **installer .exe file.** It will give the following screen:



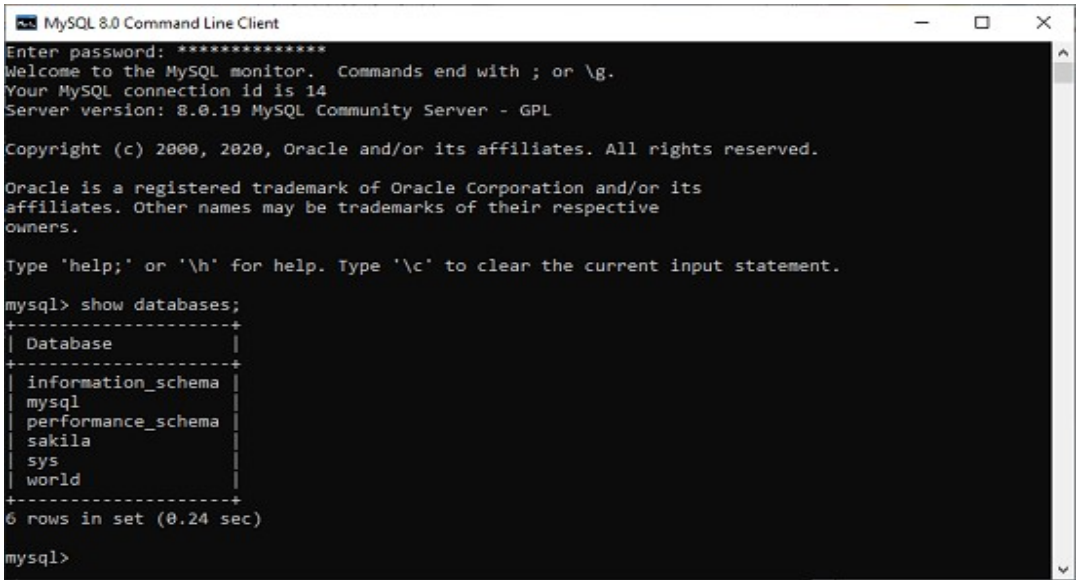
Step 2: In the next wizard, choose the **Setup Type**. There are several types available, and you need to choose the appropriate option to install MySQL product and features. Here, we are going to select the **Full** option and click on the Next button.



Step 3: Now, the MySQL installation is complete. Click on the Finish button.

Open your MySQL **Command Line Client**; it should have appeared with a **mysql> prompt**. If you have set any password, write your password here. Now, you are connected to the MySQL server, and you can execute all the SQL command at mysql> prompt as follows:

For example: Check the already created databases with show databases command:



MySQL command-line client commands

This command allows us to connect with MySQL Server with a username and passwords using below syntax.

1.mysql -u [username] -p;

If you want to connect with a particular database, use this syntax:

2.mysql -u [username] -p [database];

If you want to set a new password, use this syntax:

3.mysqladmin -u root password your_password;

We can use the **"exit"** command to quit the MySQL command-line client.

We can clear the console window in Linux using the below command:

4.mysql> system clear;

It is to be noted that there is no command available for Windows to clear the console window of MySQL.

2.Connect to MySQL Server:

- Once MySQL Client is installed, open your terminal or command prompt.
- Type the following command to connect to your MySQL server

>>>mysql -u username -p -h hostname

- Replace username with your MySQL username, hostname with the hostname or IP address of the MySQL server, and -p will prompt you for your MySQL password.
- Press Enter, and you'll be prompted to enter your MySQL password.

3.Execute SQL Queries:

- After successfully connecting to the MySQL server, you'll see a MySQL prompt (mysql>).
- You can now execute SQL queries directly within the MySQL Client.

mysql> SHOW DATABASES;

mysql> USE your_database_name;

mysql> SHOW TABLES;

mysql> SELECT * FROM your_table_name;

4.Exit MySQL Client:

- To exit the MySQL Client, type exit or quit and press Enter.

Using php MyAdmin

phpMyAdmin provides a user interface through which we can execute query within SQL. We can also paste the query into SQL to test our output, whereas on MySQL Console we cannot copy and paste queries. We have to write query every time to execute on MySQL console.

Database - phpMyAdmin supports databases.

- MySQL 5.5 or latest versions
- MariaDB 5.5 or latest versions

Download the MySQL database from here https://dev.mysql.com/downloads/file/?id=486088 or MariaDB database from here https://mariadb.org/download/.

Web Browser - Web browser is required to access phpMyAdmin with enabled cookie and JavaScript. It can be **Chrome, Internet Explorer**, etc.

SOME KEY FEATURES OF PHPMYADMIN

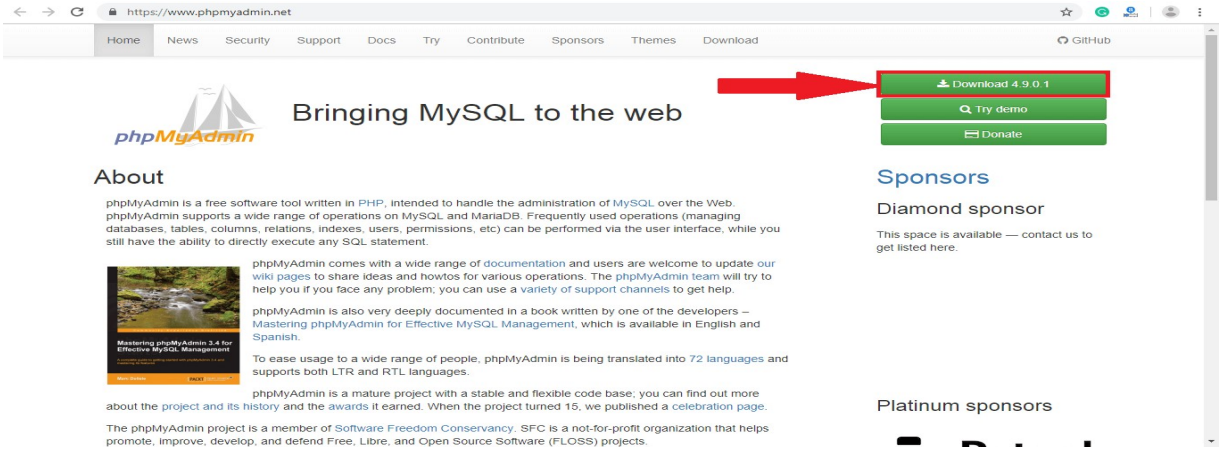
- **Intuitive Web Interface**: phpMyAdmin provides an easy-to-use interface for tasks like browsing databases, creating tables, managing users, and executing SQL statements[1].

- **MySQL Administration**: You can perform various administration tasks, including creating databases, running queries, and adding user accounts[2].
- **Multi-Language Support**: phpMyAdmin is available in 72 languages and supports both left-to-right (LTR) and right-to-left (RTL) languages[1].
- **Export and Import Data**: You can export data to formats like CSV, SQL, XML, PDF, and more. Importing data from CSV and SQL files is also supported[1].
- **Server Maintenance**: phpMyAdmin allows you to manage server configuration, databases, and tables

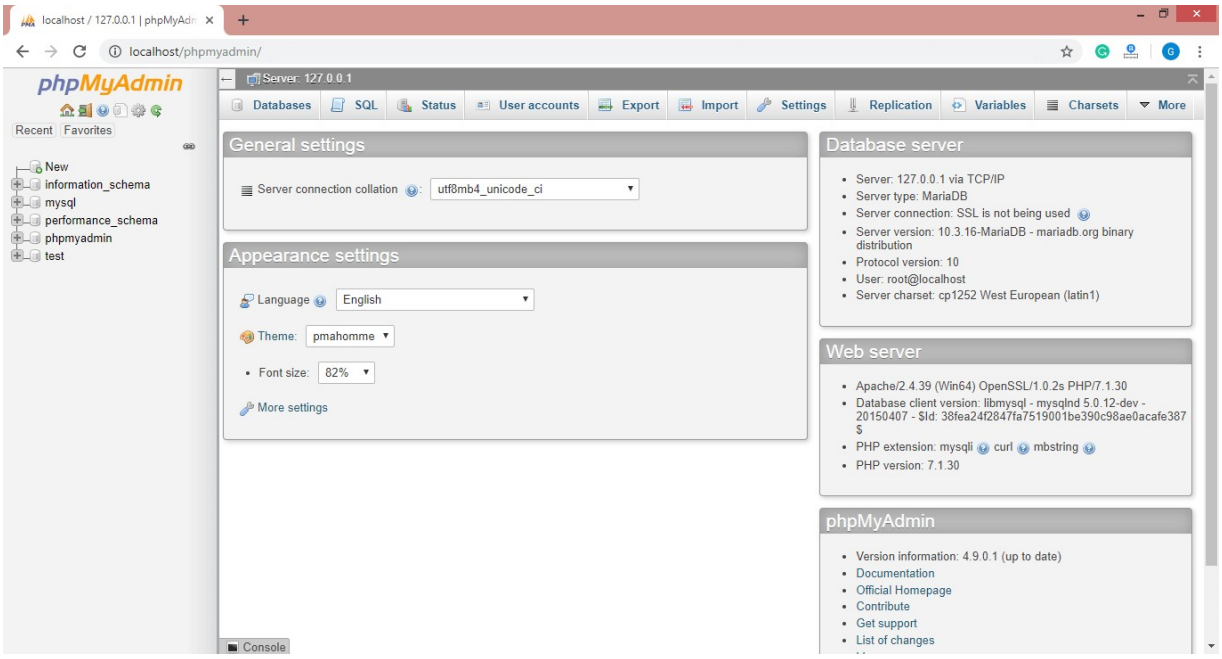
How to install phpMyAdmin

Here we are going to discuss that how to download phpMyAdmin on Windows operating system. Below are the steps-

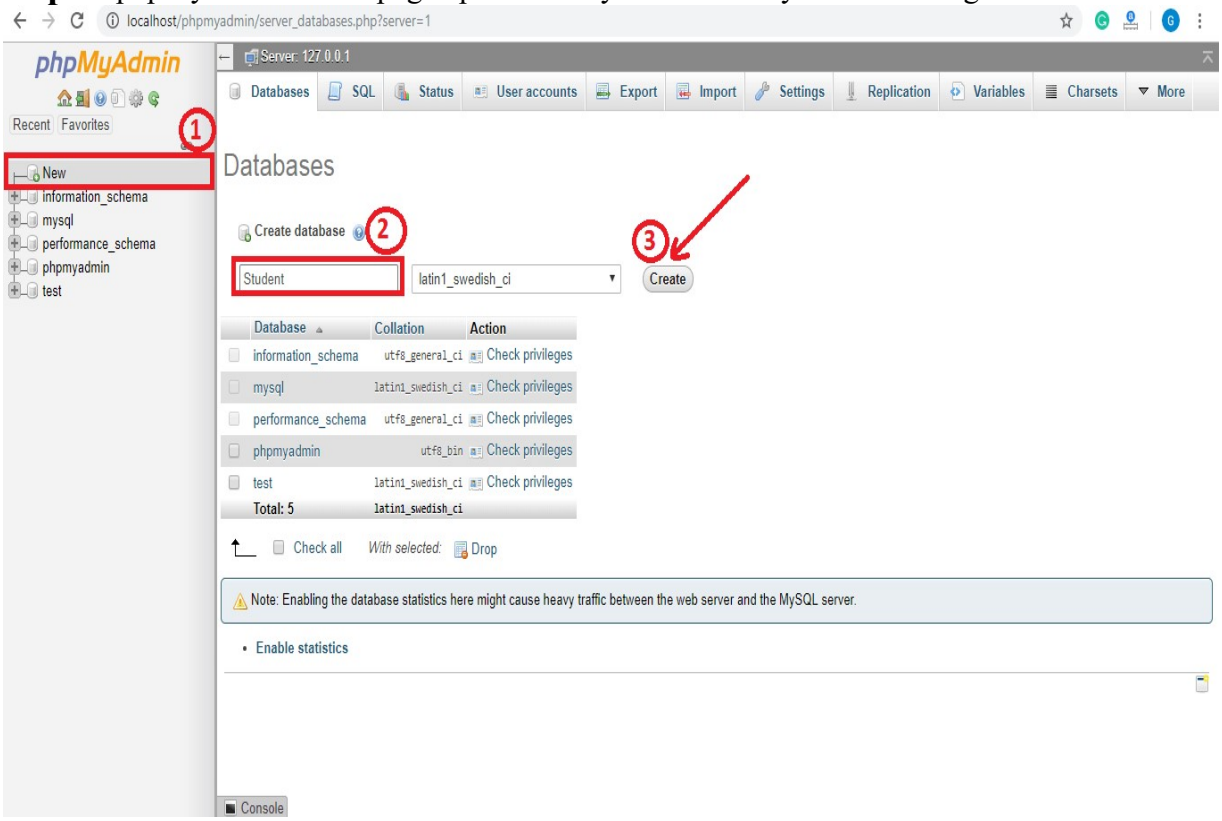
Step 1: Download the latest version of phpMyAdmin software tool from here https://www.phpmyadmin.net/ as per the following instruction. Click on the download button to start downloading.

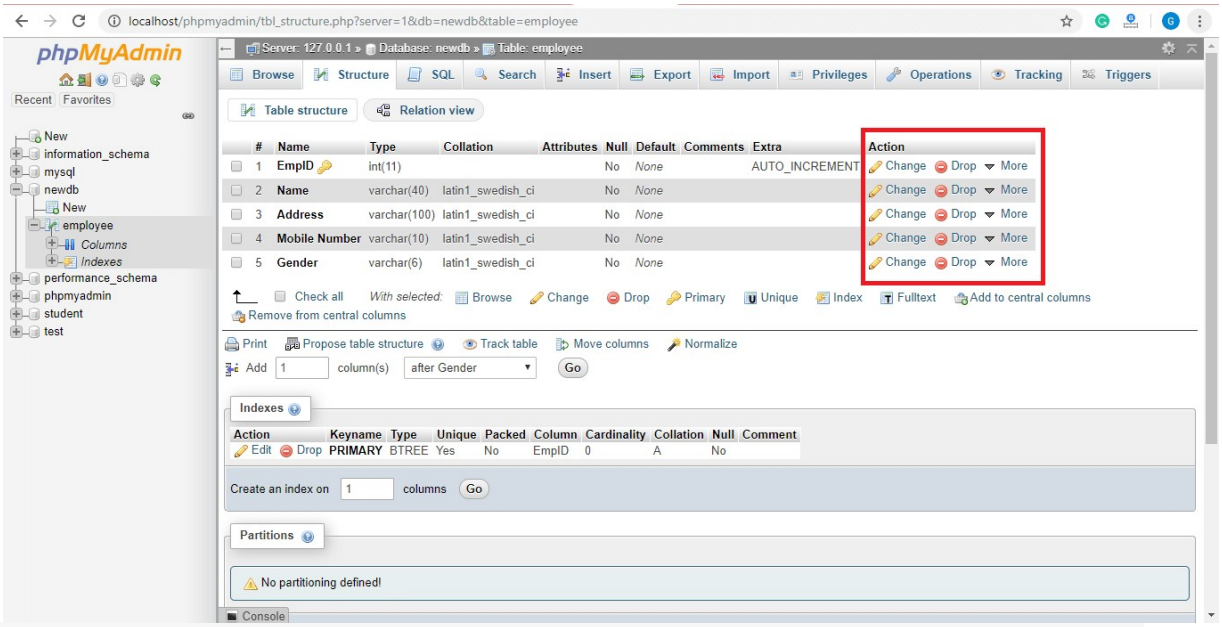


Step 2: Click on any of the below link of **phpMyAdmin** for the databaseyou want to access.



Step 3: phpMyAdmin home page opens. Now you can modify databaseusing the same.





DIFFERENCE BETWEEN USING MYSQL CLIENT AND USING PHP MYADMIN

USING MYSQL CLIENT SERVER	USING PHP MYADMIN SERVER
MySQL Client: It's a command-line tool that needs to be installed and configured separately on a server. You interact with it by typing commands directly	**phpMyAdmin**: A web-based application that can be installed on a server with a web server (like Apache or Nginx) and PHP. It provides a graphical user interface (GUI) for managing MySQL databases.
MySQL Client: Lacks a built-in GUI. Typically managed using command-line tools or scripts.	**phpMyAdmin**: Provides a user-friendly web-based interface with features like database management, query execution, and import/export functionalities.
MySQL Client: Has built-in security features (user authentication, access control).	**phpMyAdmin**: Depends on the security measures implemented by the web server or hosting environment. Additional security may be needed.
MySQL Client: Works with different programming languages and frameworks.	**phpMyAdmin**: Specifically designed for managing MySQL databases.

MySQL Client: Supports advanced concepts (transactions, stored procedures, triggers, views).	**phpMyAdmin**: Offers many features but may not match the full functionality of direct MySQL interaction.
MySQL Client: Accessed remotely using client applications or libraries.	**phpMyAdmin**: Accessed through a web browser, making it accessible from anywhere with an internet connection.

MYSQL COMMANDS

MySQL is a powerful relational database management system, and knowing some essential commands can be very useful.

Here are some basic MySQL commands.

1.Connecting to MySQL Server.

>>>mysql -u [username] –p

This command connects to the MySQL server using the specified username and prompts for the password.

2.**Connecting to a Specific Database**

>>>mysql -u [username] -p [database]

This command connects to the MySQL server and selects the specified database.

3.**Setting a New Password**.

>>>mysqladmin -u root password [your_password]

4. Exiting MySQL Command-Line Client.

>>>exit

This command exits the MySQL command-line client and returns you to the system prompt.

5.Clearing the Console Window (Windows).

>>>cls

1. Database Operations:

a. Create Database:

- ✓ **Syntax**: CREATE DATABASE database_name;
- ✓ **Explanation**: This command is used to create a new database in MySQL.
- ✓ **Example**: CREATE DATABASE my_database;

b. Drop Database:

- ✓ **Syntax**: DROP DATABASE database_name;
- ✓ **Explanation**: Drops (deletes) an existing database and all its tables.

- ✓ **Example**: DROP DATABASE my_database;

c. Use Database:

- ✓ **Syntax**: USE database_name;
- ✓ **Explanation**: Switches to a specified database, making it the active database for subsequent queries.
- ✓ **Example**: USE my_database;

d. Show Databases:

- ✓ **Syntax**: SHOW DATABASES;
- ✓ **Explanation**: Lists all databases on the MySQL server.
- ✓ **Example**: SHOW DATABASES;

2. Table Operations:

a. Create Table:

Syntax: CREATE TABLE table_name (

column1 datatype,

column2 datatype,

...

);

Explanation: Creates a new table with specified columns and their data types.
Example: CREATE TABLE employees (

id INT AUTO_INCREMENT PRIMARY KEY,

name VARCHAR(50),

age INT,

department VARCHAR(50)

);

b. Drop Table:

- ✓ **Syntax**: DROP TABLE table_name;
- ✓ **Explanation**: Deletes an existing table from the database.
- ✓ **Example**: DROP TABLE employees;

c. Alter Table (Add Column):

Syntax: ALTER TABLE table_name ADD COLUMN column_name datatype;

- ✓ **Explanation**: Adds a new column to an existing table.
- ✓ **Example**: ALTER TABLE employees ADD COLUMN salary DECIMAL(10, 2);

d. Show Tables:

Syntax: **SHOW TABLES;**

- ✓ **Explanation**: Lists all tables in the current database.
- ✓ **Example**: SHOW TABLES;

3. Data Manipulation:

a. Insert Data:

Syntax :INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);

Explanation: Inserts new rows into a table with specified values.

INSERT INTO employees (name, age, department, salary) VALUES ('Janavi', 30, 'HR', 50000.00);

b. Select Data:

Syntax: SELECT columns FROM table_name WHERE condition;

- ✓ **Explanation**: Retrieves data from a table based on specified conditions.
- ✓ **Example**: SELECT * FROM employees WHERE department = 'HR';

c. Update Data:

Syntax: UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;

- ✓ **Explanation**: Modifies existing data in a table based on specified conditions.
- ✓ **Example**: UPDATE employees SET salary = 55000.00 WHERE name = 'John';

d. Delete Data:

Syntax: DELETE FROM table_name WHERE condition;

- ✓ **Explanation**: Deletes rows from a table based on specified conditions.
- ✓ **Example**: DELETE FROM employees WHERE age > 60;

4. User and Permission Management:

a. Create User:

Syntax: CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';

- ✓ **Explanation**: Creates a new user with a specified username and password.
- ✓ **Example**: CREATE USER 'myuser'@'localhost' IDENTIFIED BY 'mypassword';

b. Grant Privileges:

Syntax: GRANT privileges ON database_name.table_name TO 'username'@'localhost';

- ✓ **Explanation**: Grants specific privileges to a user on a database or table.
- ✓ **Example**: GRANT SELECT, INSERT, UPDATE ON my_database.* TO 'myuser'@'localhost';

c. Revoke Privileges:

Syntax: REVOKE privileges ON database_name.table_name FROM 'username'@'localhost';

- ✓ **Explanation**: Revokes previously granted privileges from a user on a database or table.
- ✓ **Example**: REVOKE INSERT ON my_database.* FROM 'myuser'@'localhost';

d. Drop User:

Syntax: DROP USER 'username'@'localhost';

- ✓ **Explanation**: Deletes an existing user from the MySQL server.
- ✓ **Example**: DROP USER 'myuser'@'localhost';

5. Miscellaneous:

a. Show MySQL Version:

Syntax: SELECT VERSION();

- ✓ **Explanation**: Displays the version of the MySQL server.
- ✓ **Example**: SELECT VERSION();

b. Exit MySQL:

Syntax: EXIT;

- ✓ **Explanation**: Exits the MySQL command-line client.
- ✓ **Example**: EXIT;

PHP with MySQL: PHP MySQL Functions:PHP MySQL functions, explaining each with syntax and examples, using both MySQLi and PDO extensions.

MySQLi Functions: MySQLi (MySQL Improved) provides procedural and object oriented interface to data and its management. The i extension MySQL functions allows the user to access its database servers.

1. Establishing a Connection:

Syntax: `mysqli_connect(servername, username, password, dbname);`

Example:

```
<?php
```

```
$conn = mysqli_connect("localhost", "username", "password", "database");
if (!$conn) {
   die("Connection failed: " . mysqli_connect_error());
}
?>
```

This function establishes a connection to the MySQL database server. You provide the hostname, username, password, and database name as parameters. Optionally, you can specify the port and socket for the connection.

2. Executing Queries:

Syntax: mysqli_query(connection, query);

Example

```
<?php
$result = mysqli_query($conn, "SELECT * FROM users");
?>
```

This function executes a MySQL query on the specified connection. It takes the connection object and the SQL query as parameters.

3. Fetching data:

Syntax: mysqli_fetch_assoc(result);

Example:

```
<?php
while ($row = mysqli_fetch_assoc($result)) {
   echo "ID: " . $row["id"] . ", Name: " . $row["name"] . "<br>";
}
?>
```

This function fetches a result row as an associative array from the result set returned by mysqli_query(). It returns NULL if there are no more rows in the result set.

4. Handling Errors:

Syntax: mysqli_error(connection);

Example:

```
<?php
if (mysqli_error($conn)) {
   die("Query failed: " . mysqli_error($conn));
}
```

```
?>
```

This function returns a string description of the last error that occurred during the most recent MySQL function call.

5. Closing Connections:

Syntax: mysqli_close(connection);

Example:

```
<?php
mysqli_close($conn);
?>
```

This function closes the connection to the MySQL server that's associated with the specified connection object.

PDO Functions:

1. Establishing a Connection:

Syntax: new PDO(dsn, username, password, options);

Example:

```
<?php
try {
    $pdo = new PDO("mysql:host=localhost;dbname=database", "username", "password");
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Connection failed: " . $e->getMessage());
}
?>
```

This code creates a new PDO object representing a connection to the MySQL server. You provide the Data Source Name (DSN), username, and password as parameters.

2. Executing Queries:

Syntax: $pdo->query(query);

Example:

```
<?php
$stmt = $pdo->query("SELECT * FROM users");
?>
```

This function prepares and executes a SQL statement. It returns a PDOStatement object representing the result set of the query.

3. Fetching Data:

Syntax: $stmt->fetch(fetch_style, cursor_orientation, cursor_offset);

Example:

```
<?php
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
    echo "ID: " . $row["id"] . ", Name: " . $row["name"] . "<br>";
}
?>
```

This method fetches the next row from a result set as an associative array, numeric array, or both, depending on the fetch style specified.

4. Handling Errors:

PDO automatically throws a PDOException when an error occurs. You can use try-catch blocks to handle exceptions.

5.Closing Connections:

There's no explicit closing function in PDO. You can set the variable holding the connection to null to close it.

CONNECTING TO MYSQL AND SELECTING THE DATABASE:

To create and establish a connection to MySQL using PHP, you can use either MySQLi (MySQL Improved) or PDO (PHP Data Objects) extension. Below, I'll explain how to create and establish a connection using both methods:

PHP 5 and later can work with a MySQL database using:

- ✓ MySQLi extension (the 'i' is abbreviation for improved)
- ✓ PDO (PHP Data Objects)

PDO will work with 12 different database systems, whereas MySQLi will only work with MySQL databases.

- ✓ So, if you have to shift your project to use alternative database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the complete code — queries included.

Using MySQLi:

1. **Include Connection Parameters:** Before establishing a connection, define the necessary connection parameters such as hostname, username, password, and database name.
2. **Create a Connection:** Use the mysqli_connect() function to establish a connection to the MySQL server. Pass the connection parameters as arguments to this function.
3. **Check Connection:** Verify if the connection was successful using the mysqli_connect_errno() function. If the connection failed, display an error message.
4. **Close Connection:** Once you're done with database operations, close the connection using the mysqli_close() function.

Def: MySQL is a popular relational database management system (RDBMS) that is commonlyused in conjunction with PHP to create dynamic web applications.

- In the PHP code provided, the NEW keyword is used to create a new instance of the mysqli class, which represents a connection to a MySQL database

To create A Folder name **LIBRARY in htdocs or xampp/htdocs.**

Source Code:QUERY-CREATE DATABASE library;

After creation of Database library.

Create BOOKS Table in the created database

USE library;

```
CREATE TABLE books (
id INT AUTO_INCREMENT PRIMARY KEY,
title VARCHAR(255) NOT NULL,
author VARCHAR(255) NOT NULL,
price  INT(100),
  quantity INT (100)
);
```

After creation of database LIBRARY and BOOKSTo eastablish the connection using php script

SOURCE CODE:

1.File NAME: db_connection.php

```
<?php
$servername = "localhost";
```

```
$username = "root";
$password = "";
$dbname = "library";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
      die("Connection failed: " . $conn->connect_error);
}
echo " LIBRARY DATABASE CONNECTED SUCCESSFULLY "
?>
```

OUTPUT: LIBRARY DATABASE CONNECTED SUCCESSFULLY

Using PDO:

1. **Include Connection Parameters:** Define the necessary connection parameters such as hostname, username, password, and database name.
2. **Create a Connection:** Create a new PDO object by passing the Data Source Name (DSN), username, and password to the PDO constructor.
3. **Set Error Mode:** Set the error mode to exception using $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION). This will throw exceptions when errors occur during database operations.
4. **Check Connection:** Wrap the connection code inside a try-catch block to catch any exceptions that may occur during connection establishment.
5. **Close Connection:** Once you're done with database operations, set the PDO object to null to close the connection.

Example:

```
<?php
// Connection parameters
  $servername = "localhost";
  $username = "root";
  $password = "";
  $dbname = "library";
try {
   // Create connection
```

```
    $conn = new PDO("mysql:host=$servername;dbname=$database", $username, $password);
    // Set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
} catch(PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
}
// Close connection
$conn = null;
?>
```

Selecting the Database: To select a database in MySQL using PHP, you can use the mysqli_select_db() function with MySQLi extension or specify the database name in the DSN (Data Source Name) when creating a PDO connection. Below, I'll explain both methods:

Using MySQLi:

- ✓ **Establish Connection:** First, establish a connection to the MySQL server using MySQLi functions like mysqli_connect().
- ✓ **Select Database:** After connecting to the MySQL server, use the mysqli_select_db() function to select the desired database. Pass the connection object and the name of the database as arguments to this function.

Example:

```
<?php
// Connection parameters
  $servername = "localhost";
  $username = "root";
  $password = "";
  $dbname = "library";
// Create connection
$conn = mysqli_connect($servername, $username, $password,$database);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
```

```
}
// Select database
if (!mysqli_select_db($conn, $database)) {
   die("Database selection failed: " . mysqli_error($conn));
}

echo "Database selected successfully";
// Close connection
mysqli_close($conn);
?>
```

```
<?php
$sql = "SELECT * FROM books";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
while($row = $result->fetch_assoc()) {

echo"<tr><td>".$row["title"]."</td><td>".$row["author"]."</td><td>".$row["price"]
."</td><td>".$row["quantity"]."</td></tr>";
    }
} else {
    echo "0 results";
}
?>
```

MySQL SELECT Database:SELECT Database is used in MySQL to select a particular database to workwith. This query is used when multiple databases are available with MySQLServer

To use SQL command USE to select a particular database.

Syntax:

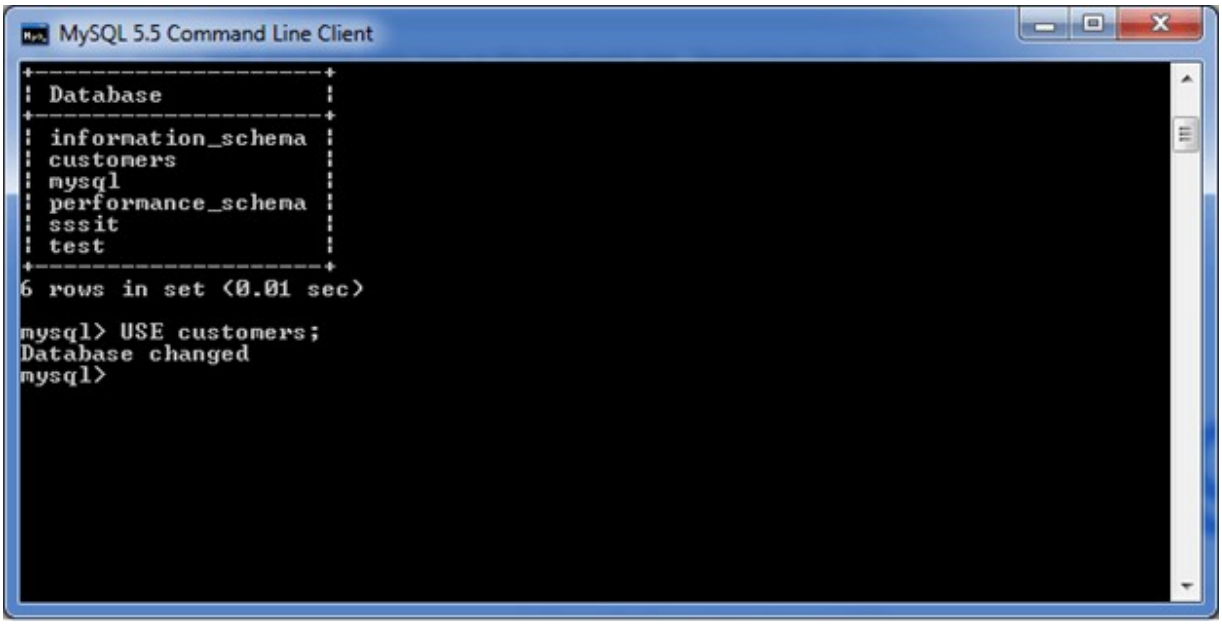
1. USE database_name;

Example:

Let's take an example to use a database name "customers".

1. USE customers;

It will look like this:



Select Data From a MySQL Database

The SELECT statement is used to select data from one or more tables:

SELECT column_name(s) FROM table_name

or we can use the * character to select ALL columns from a table:

SELECT * FROM table_name

Executing Simple Queries: Executing simple queries typically refers to performing basic database operations such as SELECT, INSERT, UPDATE, and DELETE in a database system like MySQL using a programming language like PHP. Here's a breakdown of each operation:

- ✓ A list of commonly used MySQL queries to create database, use database, create table, insert record, update record, delete record, select record, truncatetable and drop table are given below.

MySQL Create Database

The **create database** query in MySQL can be used to create a database in the MySQL server.

Syntax

Following is the syntax for the query −

```
CREATE  DATABASE  databasename;
```

Example

In the following query, we are creating a database named tutorials.

```
CREATE  DATABASE  College;
```

2.MySQL Use Database

The MySQL **use database** query is used to select a database to perform operations such as creating, inerting, updating tables or views, etc.

Syntax

Following is the syntax for the query −

```
USE database_name;
```

Example

The following query selects a database named tutorials −

```
USE college;
```

3.MySQL Create Query

The MySQL create query can be used to create databases, tables, indexes, views, etc.

Syntax

Following is the syntax for the query −

```
CREATE [table table_name |index index_name | view view_name];
```

Example

Here, we are creating a table named STUDENTS using the following CREATE query −

```
CREATE TABLE CUSTOMERS (
   ID int,
   NAME varchar(20),
   AGE int,
   PRIMARY KEY (ID)
);
```

4.MySQL Insert Query

The MySQL insert query can be used to insert records within a specified table.

Syntax

Following is the syntax for the query −

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

Example

In the following query, we are inserting some records into a table named CUSTOMERS −

```
INSERT INTO CUSTOMERS (ID, NAME, AGE) VALUES (1, "Nikhilesh", 28);
INSERT INTO STUDENTS (ID, NAME, AGE) VALUES (2, "Akhil", 23);
INSERT INTO STUDENTS (ID, NAME, AGE) VALUES (3, "Sushil", 35);
```

5.MySQL Update Query

The MySQL **update query** can be used to modify the existing records in a specified table.

Syntax

Following is the syntax for the query −

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Example

```
UPDATE CUSTOMERS SET NAME = "Nikhil" WHERE ID = 1;
```

6.MySQL Alter Query

The **ALTER query** in MySQL can be used to add, delete, or modify columns in an existing table.

Syntax

Following is the syntax for the query −

```
ALTER TABLE table_name
[ADD|DROP] column_name datatype;
```

Example

Here, we are trying to add a column named ADDRESS to the existing CUSTOMERS table.

```
ALTER TABLE CUSTOMERS
ADD COLUMN ADDRESS varchar(50);
```

7.MySQL Delete Query

The **Delete query** in MySQL can be used to delete existing records in a specified table.

Syntax

Following is the syntax for the query −

```
DELETE FROM table_name WHERE condition;
```

Example

In the following query, we are deleting a record from CUSTOMERS table where the ID is equal to 3.

```
DELETE FROM CUSTOMERS WHERE ID = 3;
```

8.MySQL Truncate Table Query

The MySQL **truncate table** query can be used to remove all the records but not the table itself.

Syntax

Following is the syntax for the query −

TRUNCATE [TABLE] table_name;

Example

In the following query, we are removing all the records from the CUSTOMERS table using the truncate table query −

```
TRUNCATE TABLE CUSTOMERS;
```

9.MySQL Drop Query

The MySQL **drop query** is used to delete an existing table in a database.

Syntax

Following is the syntax for the query −

DROP TABLE table_name;

Example

Here, we are trying to delete the table named CUSTOMERS using the drop table query.

```
DROP TABLE CUSTOMERS;
```

Example:

```
<?php
// Establishing connection
  $servername = "localhost";
  $username = "root";
  $password = "";
  $dbname = "library";
// Create connection
$conn = new mysqli($servername, $username, $password, $database);


// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "SELECT id, name, email FROM users";
$result = $conn->query($sql);
// Displaying results
 if ($result->num_rows > 0) {
while($row = $result->fetch_assoc()) {
 echo "ID: " . $row["id"]. " - Name: " . $row["name"]. " - Email: " . $row["email"]. "<br>";
}
$sql = "INSERT INTO users (name, email) VALUES ('John Doe', 'john@example.com')";
if ($conn->query($sql) === TRUE) {
echo "New record created successfully";
} else {
echo "Error: " . $sql . "<br>"
}
// Executing UPDATE query
$sql = "UPDATE users SET email = 'new_email@example.com' WHERE id = 1";
if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
```

```
}
$sql = "DELETE FROM users WHERE id = 1";
if ($conn->query($sql) === TRUE) {
 echo "Record deleted successfully";
} else {
echo "Error deleting record: " . $conn->error;
}
// Closing connection
$conn->close();
?>
```

Retrieving Query Results:Retrieving query results in PHP and MySQL involves fetching data from the database after executing a SELECT query. Here's a detailed description of the process:

1. **Establish Connection**:
   - ✓ Use PHP's MySQLi or PDO extension to connect to the MySQL database.
   - ✓ Provide the hostname, username, password, and database name.
2. **Execute SELECT Query**:
   - ✓ Use mysqli_query() function (MySQLi) or $pdo->query() method (PDO) to execute the SELECT query.
   - ✓ The SELECT query specifies the columns to retrieve, the table from which to retrieve the data, and optional conditions to filter the results.
3. **Process Results**:
   - ✓ Use functions like mysqli_fetch_assoc(), mysqli_fetch_array(), or $pdoStatement->fetch() to retrieve rows from the result set.
   - ✓ Loop through the result set to fetch each row one by one until all rows are fetched or processed.
   - ✓ The fetched rows typically return associative arrays (MySQLi) or objects (PDO) where the keys or properties correspond to column names.
4. **Display or Use Results**:
   - ✓ Inside the loop, access column values of each row using associative array notation ($row["column_name"] in MySQLi) or object properties ($row->column_name in PDO).

- ✓ Process and use the fetched data as needed, such as displaying it on a web page, storing it in variables, or further computations.

5. **Close Connection**:
   - ✓ Once the retrieval and processing are done, close the database connection using mysqli_close() function (MySQLi) or setting $pdo variable to null (PDO).

Built in function to retrieve the query result

- ✓ mysql_fetch_row() - Get a result row as an enumerated array.
- ✓ mysql_fetch_array() - Fetch a result row as an associative array, a numericarray, or both.
- ✓ mysql_fetch_assoc() - Fetch a result row as an associative array.
- ✓ mysql_fetch_object() - Fetch a result row as an object.

Retrieving query results in PHP and MySQL involves executing a SELECT query to fetch data from a database table.

syntax and an example:

SELECT column1, column2, ...

FROM table_name

WHERE condition;

- column1, column2, ...: The columns to retrieve from the table. Use * to select all columns.
- table_name: The name of the table from which to retrieve data.
- condition: Optional. Specifies the conditions that must be met for the records to be retrieved. If omitted, all rows in the table will be returned.

```
<?php
// Establishing connection
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "library";
<?php
// Establish connection
$conn = new mysqli("localhost", "username", "password", "database");

// Check connection
```

```
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Execute SELECT query
$sql = "SELECT id, name, email FROM users";
$result = $conn->query($sql);

// Process results
if ($result->num_rows > 0) {
    // Output data of each row
    while($row = $result->fetch_assoc()) {
        echo "ID: " . $row["id"]. " - Name: " . $row["name"]. " - Email: " . $row["email"]. "<br>";
    }
} else {
    echo "0 results";
}

// Close connection
$conn->close();
?>
```

- We establish a connection to the MySQL database.
- We execute a SELECT query to retrieve the `id`, `name`, and `email` columns from the `users` table.
- We process the results using a `while` loop to fetch each row from the result set (`$result`) and display the column values.
- If there are no results, we output "0 results".

Counting Returned Records : Counting returned records in PHP and MySQL involves using the `COUNT()` function in SQL to determine the number of rows that would be returned by a query. Here's how you can do it:

1. **Using COUNT() Function in SQL**:
    - In the SELECT query, use the `COUNT()` function to count the number of rows returned by the query.

- Optionally, you can specify conditions in the WHERE clause to count only specific records.

2. **Fetch and Process the Result in PHP**:
   - After executing the SELECT query, fetch the result using MySQLi or PDO.
   - Retrieve the count value from the result set using appropriate methods or functions.

Syntax:

```
SELECT COUNT(*) AS total_records FROM table_name WHERE condition;
```

- COUNT(*): Counts the total number of rows returned by the query.
- AS total_records: Assigns an alias to the count result for easy access.
- table_name: The name of the table from which to count records.
- condition: Optional. Specifies the conditions that must be met for the records to be counted. If omitted, all rows in the table will be counted.

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "library";

// Establish connection
$conn = new mysqli("localhost", "username", "password", "database");

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Execute SELECT query to count records
$sql = "SELECT COUNT(*) AS total_records FROM users WHERE status = 'active'";
$result = $conn->query($sql);

// Process result
if ($result->num_rows > 0) {
    // Fetch count value
    $row = $result->fetch_assoc();
    $total_records = $row['total_records'];
    echo "Total Active Users: " . $total_records;
} else {
    echo "0 records";
}

// Close connection
$conn->close();
?>
```

In this example:

- We establish a connection to the MySQL database.
- We execute a SELECT query using `COUNT(*)` to count the total number of active users in the `users` table.
- We assign an alias `total_records` to the count result for easy access.
- We fetch and process the count value from the result set.
- Finally, we close the database connection.

Updating Records with PHP: Updating records with PHP and MySQL involves executing an UPDATE query to modify existing data in a database table. Here's a step-by-step guide on how to update records:

1. **Establish Connection**: Connect to the MySQL database using MySQLi or PDO extension in PHP.
2. **Prepare and Execute UPDATE Query**:
   - ✓ Construct an UPDATE query specifying the table name, columns to update, and the new values.
   - ✓ Optionally, include a WHERE clause to specify the condition for updating specific records.
   - ✓ Execute the UPDATE query using mysqli_query() (MySQLi) or $pdo->query() (PDO).
3. **Check for Success**: Check whether the UPDATE query was successful or not.

Updating records with PHP involves executing an UPDATE query to modify existing data in a MySQL database table.

syntax :

UPDATE table_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

- ✓ table_name: The name of the table to update.
- ✓ column1, column2, ...: The columns to update.
- ✓ value1, value2, ...: The new values to set for the specified columns.
- ✓ condition: Optional. Specifies the condition that must be met for the records to be updated. If omitted, all rows in the table will be updated.

<?php

```
// Establishing connection
  $servername = "localhost";
  $username = "root";
  $password = "";
  $dbname = "library";
    // Establish connection
    $conn = new mysqli("localhost", "username", "password", "database");


    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }


    // Prepare and execute UPDATE query
    $sql = "UPDATE users SET email = 'new_email@example.com' WHERE id = 1";
    if ($conn->query($sql) === TRUE) {
        echo "Record updated successfully";
    } else {
        echo "Error updating record: " . $conn->error;
    }


    // Close connection
    $conn->close();
    ?>
```

- We establish a connection to the MySQL database.
- We execute an UPDATE query to set the email to `'new_email@example.com'` for the record with `id` equal to 1 in the `users` table.
- We check if the UPDATE query was successful and output the appropriate message.
- Finally, we close the database connection.