What is MySQL?

MySQL is a popular open-source **Relational Database Management System** (RDBMS) that uses **SQL** (Structured Query Language) for database operations. While MySQL is a specific database system accessible for free and supports various programming languages.

What is MySQL?

- **MySQL** is an **open-source** relational database management system (**RDBMS**) developed by Oracle Corporation.
- It uses Structured Query Language (**SQL**) for database management and is known for its reliability, speed and ease of use.
- MySQL is widely used for various applications, from small websites to large-scale enterprise systems.

Why Use MySQL

MySQL is a popular choice for managing **relational databases** for several reasons:

1. **Open Source:** MySQL is open-source software, which means it's **free to use** and has a large community of developers contributing to its improvement.
2. **Relational:** MySQL follows the relational database model, allowing users to organize data into **tables** with **rows** and **columns**, facilitating efficient **data storage** and retrieval.
3. **Reliability:** MySQL has been around for a long time and is known for its **stability** and **reliability**.
4. **Performance:** MySQL is optimized for performance, making it capable of handling **high-volume transactions** and large datasets efficiently.
5. **Scalability:** MySQL can scale both **vertically** and **horizontally** to accommodate growing data and user loads. You can add more resources to a single server or distribute the workload across multiple servers using techniques like sharding or replication.
6. **Compatibility:** MySQL is widely supported by many **programming languages**, frameworks, and tools. It offers connectors and APIs for popular languages like PHP, Python, Java, and more, making it easy to integrate with your existing software stack.
7. **Security:** MySQL provides robust **security features** to protect your data, including access controls, encryption, and auditing capabilities. With proper configuration, you can ensure that only authorized users have access to sensitive information.

Who Uses MySQL?

MySQL is a widely-used relational database management system (RDBMS) that caters to various user groups, from small businesses to large enterprises. **Small to Medium-Sized Businesses (SMBs)**: MySQL is popular among SMBs due to its cost-effectiveness, ease of use, and flexibility. These businesses leverage MySQL for managing their customer data, sales transactions, and other operational databases.

2. **Large Enterprises**: Many large organizations use MySQL for its scalability and reliability. Companies like Facebook, Google, and Adobe rely on MySQL to handle large-scale databases and high-traffic applications.
3. **Web Developers**: MySQL is a favourite among web developers because it integrates seamlessly with popular web development technologies such as PHP and JavaScript. It powers many websites and web applications, from blogs to e-commerce platforms.
4. **Educational Institutions**: MySQL is frequently used in academic settings for teaching database management and SQL skills. Its open-source nature makes it a cost-effective choice for educational purposes.

Applications of MySQL

MySQL has used in various applications across a wide range of industries and domains, because of to its versatility, reliability, and performance. Here are some common applications of MySQL:

1. **E-commerce:** MySQL is extensively used in e-commerce platforms for managing **product catalogs**, **customer data**, orders, and transactions.
2. **Content Management Systems (CMS):** Many popular CMS platforms rely on MySQL as their backend database to store **website content**, **user profiles**, comments, and configuration settings.
3. **Financial Services:** MySQL is employed in **financial applications**, including banking systems, payment processing platforms, and accounting software, to **manage transactional data**, customer accounts, and financial records.
4. **Healthcare:** MySQL is used in **healthcare applications** for storing and managing **patient records**, medical histories, treatment plans, and diagnostic information.
5. **Social Media:** MySQL powers the backend databases of many social media platforms, including **user profiles**, posts, comments, likes, and connections.

PHP Database connection

The collection of related data is called a database. XAMPP stands for cross-platform, Apache, MySQL, PHP, and Perl. It is among the simple light-weight local servers for website development.

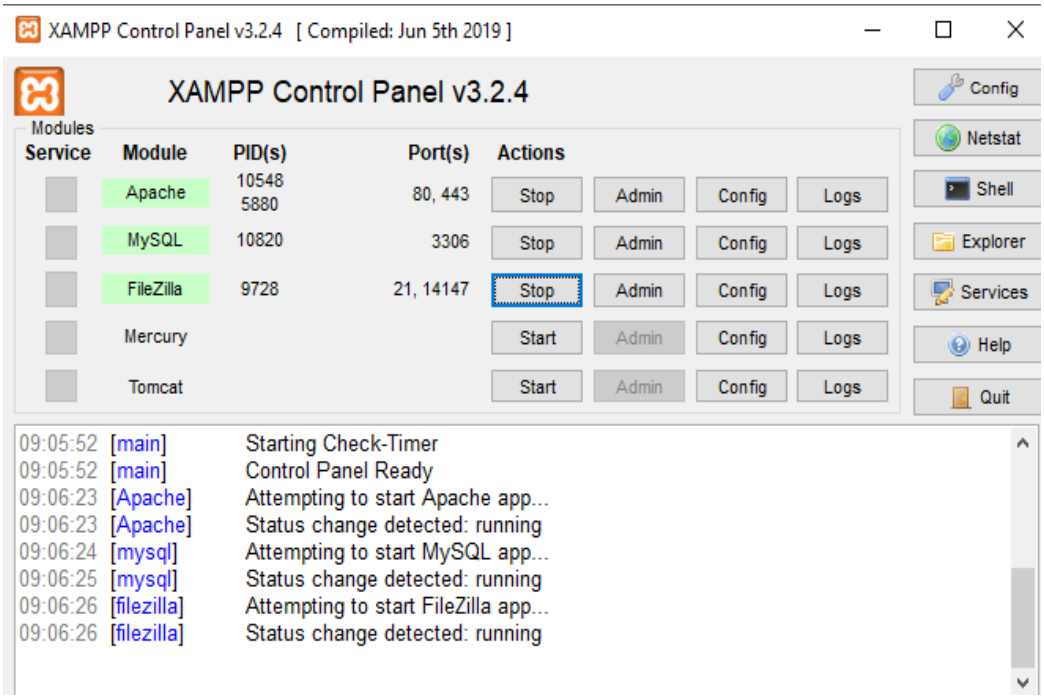
Requirements: XAMPP web server procedure:

- Start XAMPP server by starting Apache and MySQL.
- Write PHP script for connecting to XAMPP.
- Run it in the local browser.
- Database is successfully created which is based on the PHP code.

In PHP, we can connect to the database using XAMPP web server by using the following path. "localhost/phpmyadmin"

Steps in Detail:

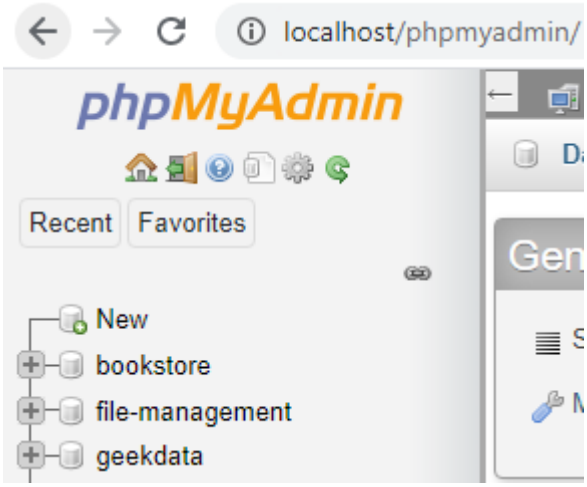
- Open XAMPP and start running Apache, MySQL and FileZilla



- Now open your PHP file and write your PHP code to create database and a table in your database.

```php
<?php
$servername = "localhost"; // Server name must be localhost
$username = "root"; // In my case, user name will be root
$password = ""; // Password is empty
// Creating a connection
$conn = new mysqli($servername,$username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failure: "
        . $conn->connect_error);
}
$sql = "CREATE DATABASE geekdata"; // Creating a database
if ($conn->query($sql) === TRUE) {
    echo "Database with name geekdata";
} else {
    echo "Error: " . $conn->error;
}
$conn->close();// Closing connection
?>
```

- Save the file as data.php in *htdocs* folder under XAMPP folder.
- Then open your web browser and type *localhost/data.php*
- Finally the database is created and connected to PHP.
- If you want to see your database, just type *localhost/phpmyadmin* in the web browser and the database can be found.



Open a Connection to MySQL

Before we can access data in the MySQL database, we need to be able to connect to the server:

PHP mysqli connect() Function

Open a new connection to the MySQL server.

Syntax

Object oriented style:

```
$mysqli -> new mysqli(host, username, password, dbname, port, socket)
```

Procedural style:

```
mysqli_connect(host, username, password, dbname, port, socket)
```

Parameter Values

Parameter	Description
host	Optional. Specifies a host name or an IP address
username	Optional. Specifies the MySQL username
password	Optional. Specifies the MySQL password
dbname	Optional. Specifies the default database to be used
port	Optional. Specifies the port number to attempt to connect to the MySQL server
socket	Optional. Specifies the socket or named pipe to be used

Technical Details

Return Value:	Returns an object representing the connection to the MySQL server
PHP Version:	5+

PHP mysqli connect_error() Function

Return the error description from the last connection error, if any

Syntax

Object oriented style:

```
$mysqli -> connect_error
```

Procedural style:

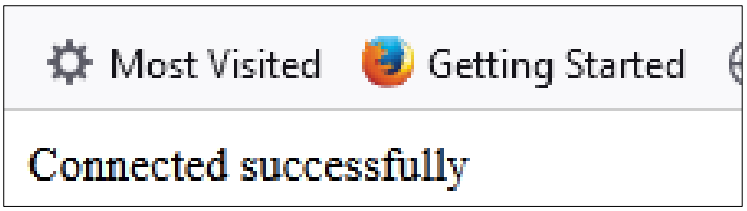
```
mysqli_connect_error();
```

Example (MySQLi Object-Oriented)

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

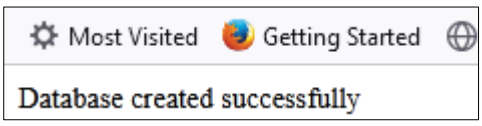
Example (MySQLi Procedural)

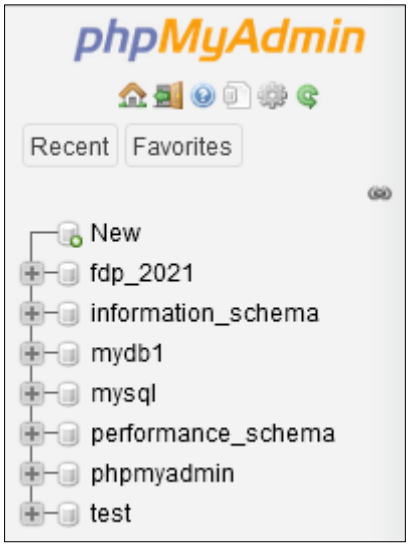
```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
// Create connection
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```



Create a MySQL Database (Object Oriented Style)

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error)
{
    die("Connection failed: " . $conn->connect_error);
}
// Create database
$sql = "CREATE DATABASE myDB1";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}
$conn->close();// Close database
?>
```

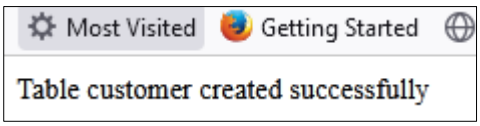


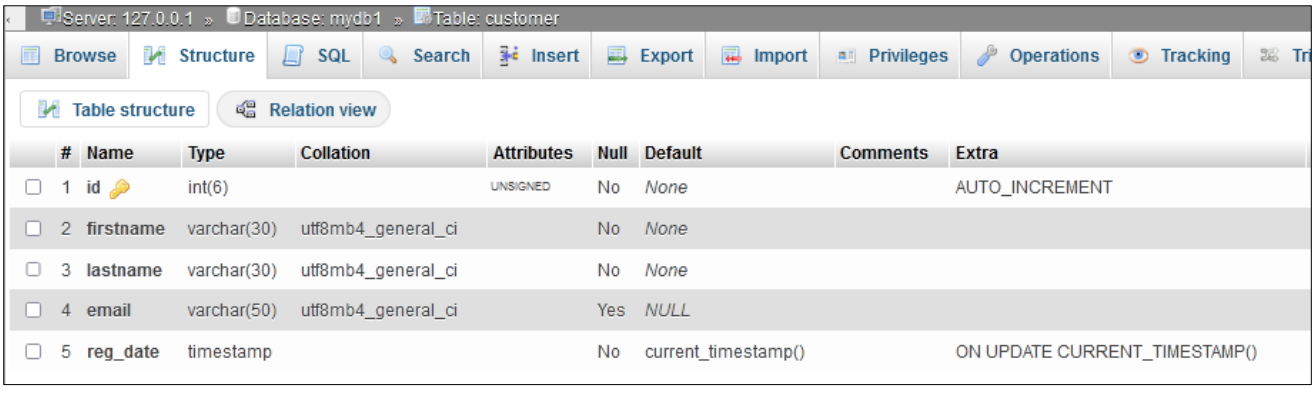


Create a MySQL Table

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

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB1";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
// sql to create table
$sql = "CREATE TABLE customer (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
    echo "Table customer created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```





#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(6)		UNSIGNED	No	None		AUTO_INCREMENT
2	firstname	varchar(30)	utf8mb4_general_ci		No	None		
3	lastname	varchar(30)	utf8mb4_general_ci		No	None		
4	email	varchar(50)	utf8mb4_general_ci		Yes	NULL		
5	reg_date	timestamp			No	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()

Insert Data into MySQL

After a database and a table have been created, we can start adding data in them.

Here are some syntax rules to follow:

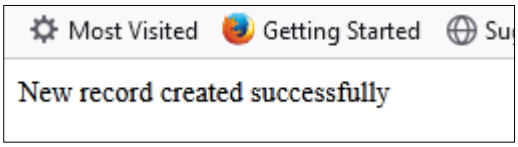
- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

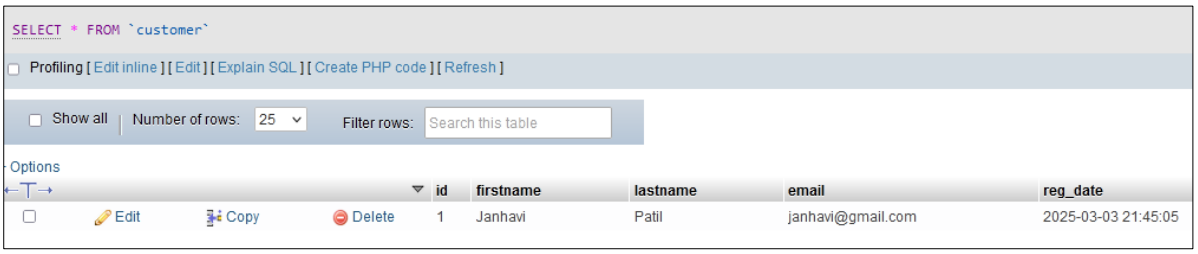
The INSERT INTO statement is used to add new records to a MySQL table:

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

Example

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB1";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO customer (firstname, lastname, email)
VALUES ('Janhavi', 'Patil', 'janhavi@gmail.com')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```



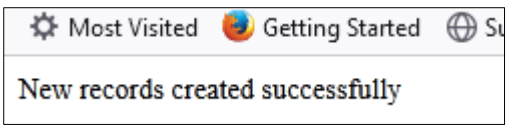


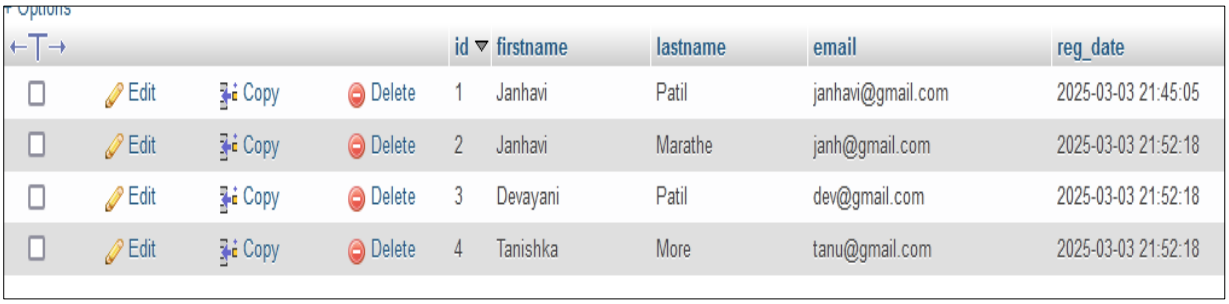
Insert Multiple Records Into MySQL

Multiple SQL statements must be executed with the `mysqli_multi_query()` function.

Example

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB1";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO customer (firstname, lastname, email)
VALUES ('Janhavi', 'Marathe', 'janh@gmail.com');";
$sql .= "INSERT INTO customer (firstname, lastname, email)
VALUES ('Devayani', 'Patil', 'dev@gmail.com');";
$sql .= "INSERT INTO customer (firstname, lastname, email)
VALUES ('Tanishka', 'More', 'tanu@gmail.com')";

if ($conn->multi_query($sql) === TRUE) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```





				id	firstname	lastname	email	reg_date
	Edit	Copy	Delete	1	Janhavi	Patil	janhavi@gmail.com	2025-03-03 21:45:05
	Edit	Copy	Delete	2	Janhavi	Marathe	janh@gmail.com	2025-03-03 21:52:18
	Edit	Copy	Delete	3	Devayani	Patil	dev@gmail.com	2025-03-03 21:52:18
	Edit	Copy	Delete	4	Tanishka	More	tanu@gmail.com	2025-03-03 21:52:18

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

Example

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB1";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM customer";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"].
" " . $row["lastname"]. "<br>";
    }
} else {
    echo "0 results"; }
$conn->close();
?>
```

- First, we set up an SQL query that selects the **id, firstname** and **lastname** columns from the **customer** table.
- The next line of code runs the query and puts the resulting data into a variable called **$result**.
- Then, the function **num_rows()** checks if there are more than zero rows returned.
- If there are more than zero rows returned, the function **fetch_assoc()** puts all the results into an associative array that we can loop through. The while() loop loops through the result set and outputs the data from the id, firstname and lastname columns.

Update Data in a MySQL Table

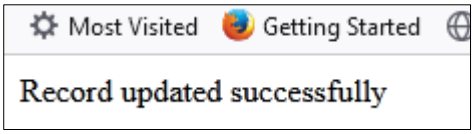
The UPDATE statement is used to update existing records in a table:

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

Example

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB1";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "UPDATE customer SET lastname='Bhamare' WHERE id=3";
```

```php
if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}

$conn->close();
?>
```



id	firstname	lastname	email	reg_date
1	Janhavi	Patil	janhavi@gmail.com	2025-03-03 21:45:05
2	Janhavi	Marathe	janh@gmail.com	2025-03-03 21:52:18
3	Devayani	Bhamare	dev@gmail.com	2025-03-03 22:09:10
4	Tanishka	More	tanu@gmail.com	2025-03-03 21:52:18

Delete Data from a MySQL Table

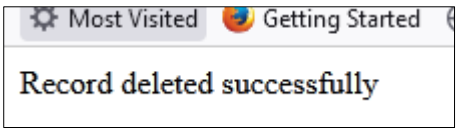
The DELETE statement is used to delete records from a table:

DELETE FROM table_name
WHERE some_column = some_value

Example

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB1";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM customer WHERE id=3";
```

```php
if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```



id	firstname	lastname	email	reg_date
1	Janhavi	Patil	janhavi@gmail.com	2025-03-03 21:45:05
2	Janhavi	Marathe	janh@gmail.com	2025-03-03 21:52:18
4	Tanishka	More	tanu@gmail.com	2025-03-03 21:52:18