MySQL Tutorial for Beginners

Published On: September 30, 2024

MySQL Tutorial for Beginners

An open-source RDBMS called MySQL creates and manages databases using SQL. It is an in-demand skill in today's industries. Learn the fundamentals in our MySQL tutorial and become an expert in managing databases.

Download MySQL Tutorial PDF

Introduction to MySQL

MySQL stores data in tables with rows and columns arranged according to schemas since it is a relational database. A schema outlines the relationships between different tables and specifies how data is stored and arranged. Let's learn the following in this MySQL tutorial:

- Overview of MySQL
- MySQL Basics
- MySQL Stored Procedures
- MySQL Triggers
- MySQL Views
- MySQL Index
- Advantages of MySQL

Overview of MySQL

Tables that map to objects are used to store data in MySQL. Every table has a schema that specifies how many columns each row will have. Text, numbers, dates, timings, and even JSON can all be reliably stored and retrieved by developers. Our **Oracle DBA training in Chennai** provides a great career for beginners.

MySQL Interview Questions and Answers

Key Features of MySQL

SQL is used to communicate with a MySQL database (Structured Query Language). A fully functional programming language is not SQL. However, it provides a simple syntax for managing your database as a querying language:

- Creating, modifying, and removing tables.
- Table indexing.
- Accessing, inserting, deleting, and updating data from tables.
- Combining data from several tables.
- Executing arithmetic operations to retrieve data.
- Partitioning data.

With its extremely scalable performance, MySQL can handle even the largest applications. Hardware upgrades, horizontal sharding, indexing tables, and other optimization techniques can be used to accomplish this.

MySQL Basics

You will learn how to manage data in MySQL by using SQL statements in this section on MySQL basics.

Querying Data

Querying data with MySQL can be done with Select From and Select queries.

MERN Stack Tutorial for Web Development Aspirants

Published On: October 14, 2024

MERN Stack Tutorial for Web Development Aspirants There is a growing need for competent MERN...

Tableau Developer Salary in Chennai

Published On: October 12, 2024

Introduction A Tableau Developer designs, develops, and maintains dashboards and visualizations using Tableau software. Key...

VMware Tutorial for Cloud Computing Aspirants

Published On: October 12, 2024

VMware Tutorial for Cloud Computing Aspirants VMware software allows you to run a virtual machine...

VBA Macros Tutorial for Beginners

Published On: October 10, 2024

Select From

It is possible to choose data from one or more tables using the SELECT query. In MySQL, the syntax for writing a SELECT statement is as follows:

SELECT select_list

FROM table_name;

- After the SELECT keyword, first define which column or columns you want to select data from. If there is more than one column in the select_list, you must comma (,) separate them.
- After the FROM keyword, enter the name of the table from which you wish to select data.

> **Example**
>
> *SELECT employeeNumber,*
>
> *lastName,*
>
> *firstName,*
>
> *extension,*
>
> *email,*
>
> *officeCode,*
>
> *reportsTo,*
>
> *jobTitle*
>
> *FROM employees;*
>
> *Or*
>
> *SELECT **

VBA Macros Tutorial for Beginners VBA macros are programs that automate repetitive operations in Microsoft...

> *FROM employees;*

Data from every column in the employee's table is returned by the query. Because it selects information from every column in the table, the SELECT * is frequently referred to as "select star" or "select all." In actuality, SELECT * should only be used for ad hoc queries.

My Sql Course Syllabus PDF

Select Query

Discover how to use the MySQL "Select" statement without making use of a table reference.

Generally, to select data from a database table, you use a SELECT statement:

SELECT select_list

FROM table_name;

The FROM clause is not necessary when using MySQL for the SELECT statement. This implies that you can have a SELECT statement like this one without a FROM clause:

SELECT select_list;

Example

SELECT CONCAT('SLA',' ','Jobs');

Reshape your career with our **Oracle PL/SQL course program** at SLA.

Sorting Data with Order By

The order of rows in the return set is not determined when you query data from a table using the SELECT command. You add the ORDER BY clause to the SELECT statement to sort the rows in the result set.

Syntax

```
SELECT
  select_list
FROM
  table_name
ORDER BY
  column1 [ASC|DESC],
  column2 [ASC|DESC],
  ...;
```

The one or more columns you wish to sort after the ORDER BY clause are specified in this form.

The letters ASC and DESC represent ascending and descending, respectively. To sort the result set in ascending order, use ASC, and to sort it in descending order, use DESC.

Example

> The customers are sorted by last name in ascending order using the ORDER BY clause in the following query.

```
SELECT
  contactLastname,
  contactFirstname
FROM
  customers
ORDER BY
```

> *contactLastname;*

Filtering Data

There are numerous queries available for data filtering. We will see them one by one.

WHERE

To filter results according to predetermined criteria by using the WHERE clause.

For the rows that a query returns, you can set search criteria using the WHERE clause. The WHERE clause's syntax is displayed as follows:

SELECT

select_list

FROM

table_name

WHERE

search_condition;

> **Example**
>
> *SELECT*
>
> *lastname,*
>
> *firstname,*
>
> *jobtitle*
>
> *FROM*
>
> *employees*
>
> *WHERE*
>
> *jobtitle = 'Sales Rep';*

SELECT DISTINCT

To demonstrate how to remove duplicate rows from a result set using the DISTINCT operator in the SELECT statement.

You can receive duplicate rows when you query a table for data. Use the DISTINCT clause in the SELECT statement to get rid of these redundant rows.

> **Syntax**
>
> *SELECT DISTINCT*
>
> *select_list*
>
> *FROM*
>
> *table_name*
>
> *WHERE*
>
> *search_condition*
>
> *ORDER BY*
>
> *sort_expression;*

> **Example**
>
> *SELECT*
>
> *DISTINCT lastname*
>
> *FROM*
>
> *employees*
>
> *ORDER BY*
>
> *lastname;*

AND

To combine Boolean statements to create complicated conditions for data filtering. There is no built-in Boolean type in MySQL. Rather, it treats values that are not zero as true and values that are zero as false.

A logical operator known as AND combines two or more Boolean expressions and returns either 1, 0, or NULL.

Syntax

A AND B

Operands are A and B in this statement. They could be expressed or have literal values.

The outcomes of combining true, false, and null with the AND operator are shown in the following table.

	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

> **Example**
>
> *SELECT*
>
> *customername,*
>
> *country,*
>
> *state*
>
> *FROM*
>
> *customers*
>
> *WHERE*

> *country = 'USA' AND*
>
> *state = 'CA';*

OR

To demonstrate how to filter data by combining the OR and AND operators. A logical operator that combines two Boolean expressions is the MySQL OR operator.

A OR B

The OR operator gives 1 (true) if either A or B is non-zero if both A and B are not NULL. As an illustration:

SELECT 1 OR 1, 1 OR 0, 0 OR 1;

> **Example**
>
> *SELECT*
>
> *customername,*
>
> *country*
>
> *FROM*
>
> *customers*
>
> *WHERE country = 'USA' OR*
>
> *country = 'France';*

IN

To check if a value matches any value in a set using the IN operator in the WHERE clause. You may find out if a value matches any value in a list of values by using the IN operator.

Syntax

value IN (value1, value2, value3,…)

If the value matches any value in the list (value1, value2, value3,…), the IN operator returns 1 (true). If not, it yields 0.

With this syntax:

- The value to test should first be specified on the left side of the IN operator. A column or an expression can represent the value.
- Secondly, provide a list of values to match in the parenthesis, separated by commas.

Functionally speaking, the IN operator is the same as a combination of several OR operators:

value = value1 OR value = value2 OR value = value3 OR …

> **Example**
>
> *SELECT*
>
> *officeCode,*
>
> *city,*
>
> *phone,*
>
> *country*
>
> *FROM*
>
> *offices*
>
> *WHERE*
>
> *country IN (‘USA’ , ‘France’);*

NOT IN

To determine whether a value doesn’t match any value in a set, negate the IN operator using the NOT

operator. The IN operator is subverted by the NOT operator:

value NOT IN (value1, value2, value2)

If the value does not equal any value in the list, the NOT IN operator returns one. If not, it yields 0. The NOT IN operator is used in the following example to determine whether or not the number 1 is NOT IN the list (1,2,3):

SELECT 1 NOT IN (1,2,3);

Example

```
SELECT
    officeCode,
    city,
    phone
FROM
    offices
WHERE
    country NOT IN ('USA' , 'France')
ORDER BY
    city;
```

BETWEEN

To use the BETWEEN operator to query data based on a range. A logical operator that indicates whether or not a value is in a range is the BETWEEN operator.

Syntax

value BETWEEN low AND high;

If: The operator BETWEEN returns 1.

value >= low AND value <= high

If not, it yields 0. The BETWEEN operator returns NULL if either the value, low, or high is NULL.

SELECT 15 BETWEEN 10 AND 20;

> **Example**
>
> *SELECT*
>
> *productCode,*
>
> *productName,*
>
> *buyPrice*
>
> *FROM*
>
> *products*
>
> *WHERE*
>
> *buyPrice BETWEEN 90 AND 100;*

LIKE

To send a pattern-matching database query with wildcards like % and _. A logical operator called LIKE determines whether or not a string contains a given pattern.

Syntax

expression LIKE pattern ESCAPE escape_character

In this syntax, the LIKE operator returns 1 if the expression fits the pattern. If not, it yields 0. Percentage % and underscore _ are the two wildcard characters that MySQL offers for pattern

construction.

- Any string consisting of zero or more characters can be matched using the percentage (%) wildcard.
- Any single character can be matched using the underscore (_) wildcard.

> **Example**
>
> *SELECT*
>
> *employeeNumber,*
>
> *lastName,*
>
> *firstName*
>
> *FROM*
>
> *employees*
>
> *WHERE*
>
> *firstName LIKE 'a%';*

LIMIT

To utilize LIMIT to restrict the number of rows that the SELECT statement returns. The SELECT statement uses the LIMIT clause to limit how many results are returned.

One or two arguments are accepted by the LIMIT clause. Both parameters' values have to be either zero or positive integers.

Syntax

SELECT

select_list

FROM

table_name

LIMIT [offset,] row_count;

With this syntax:

- The offset indicates where the first row to be returned is offset. The first row's offset is zero, not one.
- The maximum number of rows to return is specified by the row_count.

> **Example**
>
> *SELECT*
>
> *select_list*
>
> *FROM*
>
> *table_name*
>
> *ORDER BY*
>
> *sort_expression*
>
> *LIMIT offset, row_count;*

SQL Server Database Administrator Salary in Chennai

IS NULL

Use the IS NULL operator to determine if a value is NULL or not. The IS NULL operator is used to determine if a value is NULL or not.

Syntax

value IS NULL

The expression yields a true result if the value is NULL. It returns false otherwise.

You can use the IS NULL comparison operator anywhere an operator can be used, such as in the WHERE or SELECT clause.

> **Example**
>
> *SELECT*
>
> *customerName,*
>
> *country,*
>
> *salesrepemployeenumber*
>
> *FROM*
>
> *customers*
>
> *WHERE*
>
> *salesrepemployeenumber IS NULL*
>
> *ORDER BY*
>
> *customerName;*

Joining Tables

Here are the queries used to join the tables in MySQL:

MySQL Aliases

Table and column aliases are the two types of aliases that MySQL provides.

Column Aliases: Column aliases are used in MySQL to give a temporary name to a column in the result set of the query.

Syntax

SELECT

[column_1 | expression] AS descriptive_name

FROM table_name;

> **Example**
>
> *SELECT*
>
> *CONCAT_WS(‘, ‘, lastName, firstname)*
>
> *FROM*
>
> *employees;*

Table Aliases: Table aliases let you give a table in a query a temporary name, much like column aliases do.

Syntax

table_name AS table_alias

> **Example**
>
> *SELECT*
>
> *e.firstName,*
>
> *e.lastName*
>
> *FROM*
>
> *employees e*
>
> *ORDER BY e.firstName;*

Joins

It provides you with a summary of the join types that MySQL supports, such as left, right, and inner joins.

Several related tables connected by common

columns—also referred to as foreign key columns—make up a relational database. As a result, each table's data is lacking from a business standpoint.

Inner Join: A table's rows that match rows in another table can be queried using an inner join.

The basic syntax of the inner join clause used to join tables table_1 and table_2 is displayed as follows:

SELECT column_list

FROM table_1

INNER JOIN table_2 ON join_condition;

Every record in the first table is compared to every row in the second table using the inner join clause. You can use the USING clause in place of the equality operator (=) in the join condition if the column names in the two tables used for matching are the same:

SELECT column_list

FROM table_1

INNER JOIN table_2 USING (column_name);

> **Example**
>
> *SELECT*
>
> *m.member_id,*
>
> *m.name AS member,*
>
> *c.committee_id,*
>
> *c.name AS committee*
>
> *FROM*
>
> *members m*

> *INNER JOIN committees c ON c.name = m.name;*

LEFT JOIN: return null if no matching rows are discovered in the right table, and all rows from the left table and matching rows from the right table.

- A left join needs a join predicate just like an inner join does. The ideas of left and right tables are introduced while merging two tables with a left join.
- Data is selected using the left join, beginning with the left table. The left join compares each row in the left table with each row in the right table.
- Stated differently, a left join selects all of the data from the left table regardless of whether matching rows exist in the right table.
- The left join utilizes NULLs for the columns of the row from the right table in the result set if no matching rows from the right table are found.

Syntax

SELECT column_list

FROM table_1

LEFT JOIN table_2 ON join_condition;

> **Example**
>
> *SELECT*
>
> *m.member_id,*
>
> *m.name AS member,*
>
> *c.committee_id,*
>
> *c.name AS committee*
>
> *FROM*

> *members m*
>
> *LEFT JOIN committees c USING(name);*

RIGHT JOIN: Returns null if no matching rows are discovered in the left table, and all rows from the right table plus matching rows from the left table.

Except for treating the left and right tables differently, the right join clause is comparable to the left join clause. Instead of using the left table to select data, the right join begins with the right table.

Syntax

SELECT column_list

FROM table_1

RIGHT JOIN table_2 ON join_condition;

> ### Example
>
> *SELECT*
>
> *m.member_id,*
>
> *m.name AS member,*
>
> *c.committee_id,*
>
> *c.name AS committee*
>
> *FROM*
>
> *members m*
>
> *RIGHT JOIN committees c on c.name = m.name;*

CROSS JOIN clause: The cross join clause lacks a join condition, in contrast to the inner join, left join, and right join. Rows from the connected tables are

made into a Cartesian product via the cross-join. To create the result set, the cross join joins every row from the first table with every row from the right table.

> **Syntax**
>
> *SELECT select_list*
>
> *FROM table_1*
>
> *CROSS JOIN table_2;*

> **Example**
>
> *SELECT*
>
> *m.member_id,*
>
> *m.name AS member,*
>
> *c.committee_id,*
>
> *c.name AS committee*
>
> *FROM*
>
> *members m*
>
> *CROSS JOIN committees c;*

Self-Join: You can link a table to itself using a self-join. You must execute a self-join using a standard join, such as a left join or inner join since MySQL lacks a specific self-join syntax.

The steps to do a self-join are as follows:

- **Make a table alias:** To distinguish between different instances of the table, give each one a distinct alias.
- **Give the join condition details:** Specify the comparison method to be used for each row in

the table. Usually, when doing a self-join, you compare values in columns that are part of the same table.

- **Choose the desired columns:** Decide the columns you wish to see in the final result set by selecting the desired columns.

> **Example**
>
> *SELECT*
>
> *CONCAT(m.lastName, ', ', m.firstName) AS Manager,*
>
> *CONCAT(e.lastName, ', ', e.firstName) AS 'Direct report'*
>
> *FROM*
>
> *employees e*
>
> *INNER JOIN employees m ON*
>
> *m.employeeNumber = e.reportsTo*
>
> *ORDER BY*
>
> *Manager;*

MySQL Procedures

> You will discover more about MySQL stored procedures in this part, along with their benefits and drawbacks. From the sample database, the SELECT statement that follows retrieves every row in the customer's table:
>
> *SELECT*

customerName,

city,

state,

postalCode,

country

FROM

customers

ORDER BY customerName;

The query above is encapsulated in a new stored procedure that is created using the CREATE PROCEDURE command that follows:

CREATE PROCEDURE GetCustomers()

BEGIN

SELECT

customerName,

city,

state,

postalCode,

country

FROM

customers

> *ORDER BY customerName;*
>
> *END$$*
>
> *DELIMITER ;*

When a stored procedure is called for the first time, MySQL performs the following actions:

- First, search the database catalog for the stored process by name.
- Second, put together the stored procedure's code.
- Third, put the stored procedure that has been compiled into a cache memory space.
- Execute the saved procedure last.

Begin your DB career with our **Oracle course in Chennai**.

Advantages of Stored Procedures

The benefits of stored procedures are as follows:

- **Reduce network traffic:** Stored procedures aid in cutting down on network traffic that flows between MySQL servers and applications. Applications just submit the name and parameters of the stored procedures, saving many long SQL statements.
- **Centralize business logic in the database:** Stored procedures can be used to create reusable business logic that can be used in various applications.

They simplify the procedure, making it unnecessary to repeat the same logic across several applications and resulting in a database that is more consistent.

- **Make the database more secure:** Enhance the security of the database by giving apps specified rights, such as access to particular stored procedures, without granting any rights

to the underlying tables.

MySQL Triggers

A stored program that is automatically launched in response to an insert, update, or delete that takes place in the related table is known as a trigger in MySQL. For instance, you can design a trigger that is automatically triggered before the insertion of a new row into a table.

Triggers that are triggered by the INSERT, UPDATE, or DELETE events can be used with MySQL. Row-level and statement-level triggers are the two categories of triggers that are defined by the SQL standard.

- Every time a row is added, modified, or removed, a row-level trigger is set off. The trigger is automatically triggered 100 times for each of the 100 rows that are affected, for instance, if a table has 100 rows that have been added, changed, or removed.
- For every transaction, a statement-level trigger is run just once, regardless of the number of rows that are added, changed, or removed.

MySQL only allows triggers at the row level. Triggers at the statement level are not supported.

Advantages of Triggers

- Triggers offer an additional means of verifying the accuracy of data.
- Errors from the database layer are handled by triggers.
- Triggers provide a different method for carrying out planned tasks. Because triggers are triggered automatically before or after changes are made to the data in a table, you can utilize them without having to wait for the scheduled events to complete.
- To audit data changes in tables, triggers can be helpful.

MySQL Views

> This section will teach you the basics of MySQL views and how to efficiently alter views. Using an inner join, this query retrieves information from the customers and payments tables:
>
> *SELECT*
>
> *customerName,*
>
> *checkNumber,*
>
> *paymentDate,*
>
> *amount*
>
> *FROM*
>
> *customers*
>
> *INNER JOIN*
>
> *payments USING (customerNumber);*

Putting the query in the database server and giving it a name is a better approach to accomplish this. We refer to this named query as a database view or just view for short.

> A view is, by definition, a named query that is kept in the database catalog. The build VIEW statement is used to build a new view. This statement builds a view called customer payments using the query mentioned above:
>
> *CREATE VIEW customerPayments*
>
> *AS*
>
> *SELECT*

```
    customerName,
    checkNumber,
    paymentDate,
    amount
FROM
    customers
INNER JOIN
    payments USING (customerNumber);
```

Accelerate your career with our **SQL Server DBA training in Chennai**.

MySQL Index

MySQL can quickly find rows with particular column values by using indexes. In the absence of an index, MySQL has to search through the whole table for the pertinent rows. The search becomes slower with larger tables.

- An index is a type of data structure, like a B-Tree, that increases table data retrieval performance at the expense of more writes and storage to keep it maintained.
- Indexes allow the query optimizer to find data fast in tables without having to search through each row for a specific query.
- MySQL produces a special index called PRIMARY automatically when you create a table with a primary key or unique key. We refer to this index as the clustered index.
- Because the index and the data are stored in the same table, the PRIMARY index is unique. The table's row order is enforced by the clustered index.
- Secondary indexes, also known as non-

clustered indexes, are indexes that are not the primary index.

Example

```
CREATE TABLE t(
  c1 INT PRIMARY KEY,
  c2 INT NOT NULL,
  c3 INT NOT NULL,
  c4 VARCHAR(10),
  INDEX (c2,c3)
);
```

Advantages of MySQL

One of the numerous benefits of MySQL, a relational database management system (RDBMS), is its versatility.

- **Easy to Use:** With a download and installation time of less than 30 minutes, MySQL is simple to set up and maintain.
- **Reliability:** MySQL is one of the most advanced and popular databases, having been used for about 30 years and tested under many conditions.
- **Scalability:** The most popular applications' expectations may be met by MySQL's scalability.
- Performance: There are several editions of the high-performance database system MySQL.
- **Security:** MySQL offers data security as well as adherence to legal and industrial requirements.
- **Flexibility:** The Document Store in MySQL allows users to create flexible database applications.

- **Connectivity:** MySQL is an excellent choice for online database access because of its speed, security, and connectedness.
- **Reduced total cost of ownership:** MySQL can reduce total cost of ownership and the cost of new initiatives.
- **Thorough workflow management:** database administration, data design, space expansion, and setup can all be automated with MySQL's self-management tools.

Applications and situations where MySQL is utilized include embedded systems, telephone, analytics, and reporting.

Conclusion

We hope this MySQL tutorial will be useful for beginners to get started with the database journey. Hone your skills with our **MySQL training in Chennai** for a promising career.

Share on your Social Media

Softlogic Academy

Softlogic Systems

KK Nagar [Corporate Office]

No.10, PT Rajan Salai, K.K. Nagar, Chennai – 600 078.
Landmark: Karnataka Bank Building
Phone: +91 86818 84318

Email: enquiry@softlogicsys.in
Map: Google Maps Link

OMR

No. E1-A10, RTS Food Street
92, Rajiv Gandhi Salai (OMR),
Navalur, Chennai - 600 130.
Landmark: Adj. to AGS Cinemas
Phone: +91 89256 88858
Email: info@softlogicsys.in
Map: Google Maps Link

Important Links

- Disclaimer
- Privacy Policy
- Terms and Conditions

Courses

- Python
- Software Testing
- Full Stack Developer
- Java
- Power BI
- Clinical SAS
- Data Science
- Embedded
- Cloud Computing
- Hardware and Networking
- VBA Macros
- Mobile App Development
- DevOps

Social Media Links

Review Sources

- Google
- Trustpilot
- Glassdoor
- Mouthshut
- Sulekha
- Justdial
- Ambitionbox
- Indeed
- Software Suggest
- Sitejabber