MySQL Tutorial for Beginners

Introduction

Do you have fear about understanding tough syntax, tough JOINs, or the possibility of deleting the data unintentionally? Just getting into databases can be intimidating. Our MySQL tutorial for beginners will walk you through making sense of these tough spots. From pure basics and on, we will build your expertise with confidence and understanding as we explore knowledge about data retrieval and data manipulation.

Ready to progress from confusion to competence? See the entire roadmap and subjects that will be covered by taking a look at our **MySQL Course Syllabus**.

Why Students or Freshers Learn MySQL?

Learning MySQL or learning SQL skills, in general, is one of the most essential things once someone is getting into the tech industry. It's the common language understood by data.

- **High Job Demand:** Almost all tech jobs, be it a developer or a data scientist, need someone with knowledge of SQL. It is often listed as one among the most sought-after skills.
- **Foundation for All Data:** All applications, sites, and business processes are based on databases. MySQL, as the most widely used Open Source RDBMS solution, will instruct you on how to organize, store, and manage this very essential information.
- **Easy to Learn:** SQL languages have a syntax similar to everyday English words like SELECT, INSERT, UPDATE, which are easier to learn compared with hard programming languages.
- **Flexibility and Interoperability:** MySQL skills can be adapted and reapplied with ease to various other databases, such as PostgreSQL and MS SQL. It works with coding languages like Python, Java, and PHP.
- **Career Growth:** It opens doors to career opportunities in Database Administration (DBA), Software Development – Backend, Data Analytics, and Business Intelligence.

Ready to land that first job? Just practice with these most common ones. Check out our Top **MySQL Interview Questions and Answers** and ace your next tech interview!

Step-by-Step MySQL Tutorial for Beginners

In this MySQL tutorial for beginners, we will lead you step-by-step from configuring your working environment to running sophisticated queries. By the end of it, you will be confident with the basics necessary for storing and manipulating data like a pro.

Phase 1: Installation and Setup

Before we proceed with coding, we should have all necessary tools. We recommend using MySQL Community Edition and MySQL Workbench. MySQL Workbench is an advanced GUI tool.

Step 1: MySQL Server and MySQL Workbench Installation

Download MySQL Installer: Visit the MySQL website and download MySQL Installer for Windows, macOS, and Linux.

Run MySQL Installer:

- Choose “Developer Default”. It will include MySQL Server, MySQL Shell Community Edition, and MySQL Workbench Community.
- Follow the prompts, setting a secure root password this is important!
- Keep the default port of 3306.

Completion: By the end of this you should have both MySQL Server running in the background and MySQL Workbench installed.

Step 2: Connect MySQL Server with MySQL Workbench

- **Open MySQL Workbench:** To connect MySQL with MySQL Workbench, start MySQL Workbench on your Computer.
- **Create a MySQL Connection:** A local instance should be running on MySQL workbench with the label “Local Instance 3306”.
- **Connect to MySQL:** To connect MySQL, click on the connection box. A new window will appear.
- **New Query Tab:** Once connected, a new SQL query tab will open. You will be writing and executing all of your MySQL commands within this tab.

Phase 2: Understanding Databases and Basic Commands

SQL (Structured Query Language) is the standard language for such relational databases as MySQL. It enables us to communicate with the database server.

Step 3. The Database Hierarchy

In MySQL, data are arranged in a strict hierarchy:

1. **Server:** The program on your computer that provides major functions (MySQL Server).
2. **Schemas/Databases:** The containers that hold related tables.
    - **Example:** *ecommerce_db, university_db*
3. **Tables:** This is the main form of data storage, through which data is kept in rows and columns.
    - **Example:** *customers table or products table.*

Step 4. The Basics

Creating and Managing DatabasesFirst of all, let us learn to manage the highest level of organization: the database (schema).

Command	**Purpose**	**Example**
CREATE DATABASE	Creates a new database.	CREATE DATABASE bookstore_db;
DROP DATABASE	Permanently deletes a database. Use with EXTREME caution.	DROP DATABASE bookstore_db;

SHOW DATABASES	Lists all databases on the server.	SHOW DATABASES;
USE	Selects a database to work with (must be done before working with tables).	USE bookstore_db;

SQL Queries to Practice:

— 1. Check existing databases

SHOW DATABASES;

— 2. Create our new database

CREATE DATABASE library_db;

— 3. Select the new database to make it active

USE library_db;

— You will see a confirmation message and the schema navigator will update.

Step 5. Creating Your First Table

Tables are created based on columns. Columns have a data type. Data types identify the type of information stored within columns.

Data Type	Description	Example Use

INT	Whole numbers (integers).	Book ID, Quantity
VARCHAR(size)	Variable length string/text. Must define a max size.	Book Title, Author Name
TEXT	Larger strings of text.	Book Description
DATE	Date values (YYYY-MM-DD).	Publication Date
DECIMAL(p,s)	Decimal numbers (p=precision, s=scale).	Book Price (DECIMAL(5,2))

Syntax for *Create Table*:

```
CREATE TABLE table_name (
    column1_name data_type PRIMARY KEY,
    column2_name data_type,
    column3_name data_type NOT NULL,
    …
);
```

Key Concepts in Table Definition:

- **PRIMARY KEY:** A column that uniquely identifies every row (record). It should be there in every table.
- **NOT NULL:** Guarantees that a value needs to be supplied for the given column.

Example: Creating a books table:

```
USE library_db;

CREATE TABLE books (

    book_id INT PRIMARY KEY,

    title VARCHAR(255) NOT NULL,

    author VARCHAR(100) NOT NULL,

    publication_year INT,

    price DECIMAL(5, 2)

);
```

— To verify the table structure:

```
DESCRIBE books;
```

Phase 3: CRUD Operations (DML)

The four basic operations relating to database management are Create, Read, Update, and Delete, commonly abbreviated as CRUD operations.

Step 6. CREATE: Inserting Data (INSERT

The INSERT statement inserts new records/rows into a table.

Syntax:

INSERT INTO table_name (column1, column2, …)

VALUES (value1, value2, …);

Let’s insert some books:

USE library_db;

INSERT INTO books (book_id, title, author, publication_year, price)

VALUES (101, ‘The Great Gatsby’, ‘F. Scott Fitzgerald’, 1925, 12.99);

INSERT INTO books (book_id, title, author, publication_year, price)

VALUES (102, ‘To Kill a Mockingbird’, ‘Harper Lee’, 1960, 15.50);

— Inserting multiple rows at once:

INSERT INTO books (book_id, title, author, publication_year, price)

VALUES

(103, ‘1984’, ‘George Orwell’, 1949, 9.99),

(104, ‘Moby Dick’, ‘Herman Melville’, 1851, 18.75);

Step 7. READ: Retrieving Data (SELECT)

The SELECT statement is the most common query. It is used for data extraction from one or more tables.

The simplest form of a SELECT query will include:

SELECT column1, column2, …

FROM table_name

WHERE condition

ORDER BY column_name;

Selecting All Data:

The asterisk (*) represents “all columns.”

— Select all columns and all rows from the books table:

*SELECT * FROM books;*

Selecting Specific Columns:

— Select only the title and author of all books:

SELECT title, author FROM books;

Filtering Data with WHERE:

The WHERE clause filters data based on specific conditions.

Operator	**Description**
=	Equal to
!= or <>	Not equal to

> or <	Greater than/Less than
BETWEEN	Values within a range (inclusive)
LIKE	Pattern matching
AND, OR	Combining multiple conditions

Examples:

— Find the book published in 1960:

*SELECT * FROM books*

WHERE publication_year = 1960;

— Find books that cost less than $15.00:

SELECT title, price

FROM books

WHERE price < 15.00;

— Find books by George Orwell OR published before 1900:

*SELECT **

FROM books

WHERE author = ‘George Orwell’ OR publication_year < 1900;

Sorting Data with ORDER BY:

The ORDER BY clause is used for sorting.

- **ASC:** Ascending (A-Z, 1-10_ – Default
- **DESC:** Descending (Z-A, 10-1)

— Get all books, sorted by price from cheapest to most expensive:

```
SELECT title, price

FROM books

ORDER BY price ASC;
```

— Get all books, sorted by publication year from newest to oldest:

```
SELECT title, author, publication_year

FROM books

ORDER BY publication_year DESC;
```

Step 8. UPDATE: Modifying Existing Data (UPDATE)

The UPDATE statement alters values inside existing records. The WHERE clause is CRITICAL here. You will end up updating all records in that table without it.

Syntax:

```
UPDATE table_name

SET column1 = new_value1, column2 = new_value2, …

WHERE condition;
```

Example:

Suppose there was an increase in the price for ‘The Great Gatsby’:

— Update the price of the book with book_id 101 to 14.99:

```
UPDATE books

SET price = 14.99

WHERE book_id = 101;
```

— Verify the change:

```
SELECT * FROM books WHERE book_id = 101;
```

Step 9. DELETE: Deleting Data (DELETE)

The DELETE statement will eliminate one or more records from a table. Again, the WHERE clause plays a very important role here. Without it, you will delete all records from that table.

Syntax:

```
DELETE FROM table_name

WHERE condition;
```

Example:

Let’s remove ‘Moby Dick’ from the library:

— Delete the book with book_id 104:

DELETE FROM books

WHERE book_id = 104;

— Verify the deletion:

*SELECT * FROM books; — Only three books should remain*

Phase 4: Important Intermediate Concepts

These concepts are very useful for developing efficient databases.10. Aggregate Functions

These functions carry out an operation on a collection of records and generate a single value.

Step 10: Aggregate Function

Aggregate function is used to calculate a set of rows to return a single summary value.

Function	Description
COUNT()	Returns the number of rows.
SUM()	Returns the sum of values in a column.
AVG()	Returns the average value of a column.
MAX()	Returns the largest value in a column.

MIN()	Returns the smallest value in a column.

Examples:

— How many books are in the library?

SELECT COUNT(*) AS total_books FROM books;

— What is the average price of a book?

SELECT AVG(price) AS average_price FROM books;

— What is the price of the most expensive book?

SELECT MAX(price) AS highest_price FROM books;

11. Grouping Data (GROUP BY)

The GROUP BY clause is frequently used in conjunction with aggregate functions. It groups the rows with common values for specified columns into summary rows.

Example:

— We need to know how many books each author has:

SELECT author, COUNT(*) AS num_of_books

FROM books

GROUP BY author;

— We need the average price of books, grouped by the publication year:

SELECT publication_year, AVG(price) AS avg_price

FROM books

GROUP BY publication_year;

Step 12. Creating Relationships (FOREIGN KEY)

A relational database uses a key to connect tables.

- **Primary Key (PK):** Identifies uniquely a row within its own table.
- **Foreign Key (FK):** It is a column or group of columns in a table that contains the primary key value from another table. It defines a relationship.

Example: Linking books and authors (a better structure):

Now, let's make a new authors' table and relate it with our books table.

— Create the new authors table (Primary Key is author_id)

CREATE TABLE authors (

author_id INT PRIMARY KEY,

author_name VARCHAR(100) NOT NULL

);

— Insert authors

INSERT INTO authors (author_id, author_name) VALUES

(1, 'F. Scott Fitzgerald'),

(2, 'Harper Lee'),

```
    (3, 'George Orwell');

— Drop the old books table and recreate it with the Foreign Key

DROP TABLE books;

CREATE TABLE books (

    book_id INT PRIMARY KEY,

    title VARCHAR(255) NOT NULL,

    — This author_id links back to the authors table

    author_id INT,

    publication_year INT,

    price DECIMAL(5, 2),

    — Define the Foreign Key constraint

    FOREIGN KEY (author_id) REFERENCES authors(author_id)

);

— Re-insert books using the new author_id:

INSERT INTO books (book_id, title, author_id, publication_year, price)

VALUES

    (101, 'The Great Gatsby', 1, 1925, 14.99),
```

(102, ‘To Kill a Mockingbird’, 2, 1960, 15.50),

(103, ‘1984’, 3, 1949, 9.99);

Step 13. Retrieving Data from Multiple Tables (JOIN)

The greatest concept that can be implemented with SQL would be JOIN. It enables you to merge data from two or more tables based on a common column within the tables (Foreign Key).

The most common type of join operation performed on tables is INNER JOIN. It retrieves records with equal entries in both tables.

Syntax:

SELECT column1, column2, …

FROM table_A

INNER JOIN table_B ON table_A.fk_column = table_B.pk_column;

Example: Find the book title AND then get the full list of authors:

SELECT

b.title,

a.author_name

FROM

books AS b — AS b is a common alias for books table

INNER JOIN

authors AS a — AS a is an alias for authors table

ON

b.author_id = a.author_id; — Join condition: FK in books = PK in authors

You have successfully accomplished the necessary steps for MySQL configuration, creating a database, completing basic CRUD operations, and understanding concepts about connecting and retrieving information from multiple tables based on a JOIN. Almost all data-driven projects on earth rely on these concepts.

Practicing continuously and engaging with problems on a real-world scenario are the best ways to learn MySQL. Access our **MySQL Online Course** and develop your skills so you can progress from a beginner to an intermediate level.

Real Time Examples for MySQL Tutorial for Beginners

To properly learn MySQL, it is necessary to venture beyond textbook theory and apply skills to 'real-world' problems. Some projects below represent common industry needs and will help affirm knowledge of CRUD and relationship concepts:

Creating a Simple To-Do List Application Database

- **Focus:** Learning CRUD Operations and NOT NULL constraints.
- **Task:** Create a 'tasks' table with columns including task_id (Primary Key, INT), task_description (VARCHAR), is_complete (BOOLEAN or TINYINT), and created_date (DATE).
- **Obtained Skill:** Performing an INSERT operation to enter new tasks, a SELECT operation with a WHERE clause on is_complete = 0 to display pending tasks, and an UPDATE operation to complete a task. These three operations form the basis for most backend interactions on the average web app.

Creating an Order Tracking System for Customers

- **Focus:** Exploring One-to-Many Associations and JOINs
- **Task:** Create two tables: customers and orders. Connect these two tables with a Foreign Key customer_id in orders, which will reference the Primary Key in customers.
- **Obtained Skill:** First, we will learn about writing an INNER JOIN query to obtain a specific customer's name as well as all information about recent orders.

Performing Sales Total Calculations with Aggregate Functions

- **Focus:** Use of GROUP BY and Aggregate Functions (SUM, COUNT, AVG).
- **Task:** Use the orders table (from previous example) which has an order_amount column.
- **Obtained Skill:** Calculate total revenues per customer (SUM(order_amount) grouped by customer_id), as well as average order value for all orders made (AVG(order_amount)). Both are fundamental data analysis needs.

Ready to put your knowledge into practice and start your portfolio? Check out our **MySQL Project Ideas** and get cracking on building your own projects and getting ready for job interviews.

FAQs About MySQL Tutorial for Beginners

1. What is MySQL used for?

MySQL is a Relational Database Management System and an open-source database. It uses SQL as its query language. MySQL stores and manages data and retrieves data in tables with rows and columns. MySQL is the main database for large scale applications like online stores, online communities like Facebook, and X.

2. Can I use MySQL in mobile?

You don’t usually use MySQL on your mobile app directly. You use your mobile app to connect with your secured backend server programming languages like PHP, Python, and so on, which then accesses your central MySQL database.

3. What is the full form of MySQL?

MySQL full form – My Structured Query Language

4. What’s the difference between SQL and MySQL?

SQL stands for Structured Query Language. It is a language for handling and manipulating data within a database. MySQL, on the hand, is a Database Management System. It usesSQL. You can equate SQL with the syntax and MySQL with an application that follows these rules.

5. Can I learn MySQL in 2 days?

You can be familiar with basic CRUD operations and MySQL syntax within two days. But acquiring expertise, including concepts like advanced queries and indexing, will take several weeks or months.

6. Is MySQL written in C or C++?

MySQL uses C and C++. MySQL uses C and C++ because C/C++offers better performance benefits. A database server requires handling very high-speed data operations and concurrency. So, MySQL uses C/C++to make it very fast and very efficient.

7. Does MySQL require coding?

Yes. MySQL requires coding because you have to be proficient in Structured Query Language (SQL). Although there are graphical interfaces for simple operations, efficient data management and analysis with MySQL are dependent on coding expertise.

8. Is MySQL 100% free?

MySQL Community Edition is completely free, as it falls under the GPL license. But aside from that, there are paid Enterprise Editions offered by Oracle, which have full support services, extra sophisticated monitoring tools, and additional security functionalities.

9. Is MySQL difficult to learn?

MySQL is not very hard to learn. The simplest form of SQL commands are fairly easy and descriptive, using words like SELECT and INSERT. Ease of learning would relate to understanding relational database concepts, learning about JOINs, and perfecting efficient queries.

10. Is MySQL still relevant in 2026?

It absolutely is. MySQL still ranks among the most widely used databases. Knowledge of SQL, which MySQL supports, remains a fundamental skill set for highly sought-after skills such as Data Analyst, Data Scientist, and Backend Developer.

Conclusion

You have effectively completed the primer levels of MySQL and have proficient knowledge of basic commands (CRUD operations) and have understood perhaps the most paramount concept – table relationships and JOINs. You have at your fingertips a universal language for almost all types of technical jobs, ranging from web development to data analysis. You will be confident once you start practicing.

Ready to start your first portfolio project and learn more about querying and database structure? Enroll today in our complete **MySQL Course in Chennai** and learn all about optimization and advanced database skills.