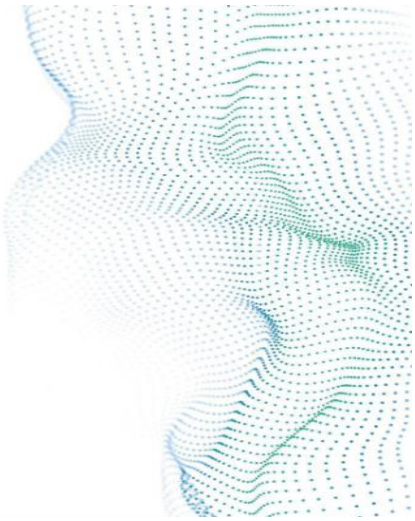
Module 1

MySql



MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company.

What is Database?

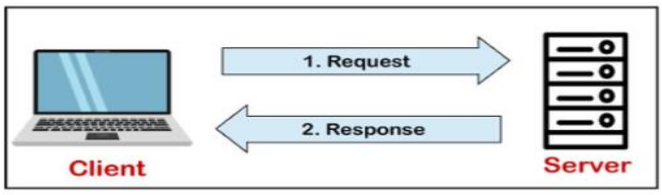
It is very important to understand the database before learning MySQL. A database is an application that stores the organized collection of records. It can be accessed and manage by the user very easily. It allows us to organize data into tables, rows, columns, and indexes to find the relevant information very quickly. Each database contains distinct API for performing database operations such as creating, managing, accessing, and searching the data it stores. Today, many databases available like MySQL, Sybase, Oracle, MongoDB, PostgreSQL, SQL Server, etc. In this section, we are going to focus on MySQL mainly.

The core of the MySQL database is the MySQL Server. This server is available as a separate program and responsible for handling all the database instructions, statements, or commands. The working of MySQL database with MySQL Server are as follows:

1. MySQL creates a database that allows you to build many tables to store and manipulate data and defining the relationship between each table.
2. Clients make requests through the GUI screen or command prompt by using specific SQL expressions on MySQL.
3. Finally, the server application will respond with the requested expressions and produce the desired result on the client-side.

How MySQL Works?

MySQL follows the working of Client-Server Architecture. This model is designed for the end-users called clients to access the resources from a central computer known as a server using network services. Here, the clients make requests through a graphical user interface (GUI), and the server will give the desired output as soon as the instructions are matched. The process of MySQL environment is the same as the client-server model.



MySQL is becoming so popular because of these following reasons:

- MySQL is an open-source database, so you don't have to pay a single penny to use it.
- MySQL is a very powerful program that can handle a large set of functionality of the most expensive and powerful database packages.
- MySQL is customizable because it is an open-source database, and the open-source GPL license facilitates programmers to modify the SQL software according to their own specific environment.
- MySQL is quicker than other databases, so it can work well even with the large data set.
- MySQL supports many operating systems with many languages like PHP, PERL, C, C++, JAVA, etc.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL is very friendly with PHP, the most popular language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

Data type

A Data Type specifies a particular type of data, like integer, floating points, Boolean, etc. It also identifies the possible values for that type, the operations that can be performed on that type, and the way the values of that type are stored. In MySQL, each database table has many columns and contains specific data types for each column.

We can determine the data type in MySQL with the following characteristics:

- The type of values (fixed or variable) it represents.
- The storage space it takes is based on whether the values are a fixed-length or variable length.
- Its values can be indexed or not.
- How MySQL performs a comparison of values of a particular data type.

MySQL supports a lot number of SQL standard data types in various categories. It uses many different data types that can be broken into the following categories: numeric, date and time, string types, spatial types, and JSON data types.

Numeric Data Type

MySQL has all essential SQL numeric data types. These data types can include the exact numeric data types (For example, integer, decimal, numeric, etc.), as well as the approximate numeric data types (For example, float, real, and double precision). It also supports BIT datatype to store bit values. In MySQL, numeric data types are categories into two types, either signed or unsigned except for bit data type.

Numeric data types

DOUBLE(m,d)	It is a double-precision floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a double. Real is a synonym for double. It requires 8 bytes for storage.
DECIMAL(m,d)	An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (m) and the number of decimals (d) is required. Numeric is a synonym for decimal.
BIT(m)	It is used for storing bit values into the table column. Here, M determines the number of bit per value that has a range of 1 to 64.
BOOL	It is used only for the true and false condition. It considered numeric value 1 as true and 0 as false.
BOOLEAN	It is Similar to the BOOL.

INT	It is a normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. We can specify a width of up to 11 digits. It requires 4 bytes for storage.
BIGINT	It is a large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. We can specify a width of up to 20 digits. It requires 8 bytes for storage.
FLOAT(m,d)	It is a floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This is not required and will default to 10,2, where 2 is the number of decimals, and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a float type. It requires 2 bytes for storage.

String Data Types:

The string data type is used to hold plain text and binary data, for example, files, images, etc. MySQL can perform searching and comparison of string value based on the pattern matching such as LIKE operator, Regular Expressions, etc.

The following table illustrates all string data types that support in MySQL:

Data Type Syntax	Maximum Size	Explanation
CHAR(size)	It can have a maximum size of 255 characters.	Here size is the number of characters to store. Fixed-length strings. Space padded on the right to equal size characters.
VARCHAR(size)	It can have a maximum size of 255 characters.	Here size is the number of characters to store. Variable-length string.
TINYTEXT(size)	It can have a maximum size of 255 characters.	Here size is the number of characters to store.
TEXT(size)	Maximum size of 65,535 characters.	Here size is the number of characters to store.
MEDIUMTEXT(size)	It can have a maximum size of 16,777,215 characters.	Here size is the number of characters to store.
LONGTEXT(size)	It can have a maximum size of 4GB or 4,294,967,295 characters.	Here size is the number of characters to store.

Variables

Variables are used for storing data or information during the execution of a program. It is a way of labeling data with an appropriate name that helps to understand the program more clearly by the reader. The main purpose of the variable is to store data in memory and can be used throughout the program.

MySQL can use variables in **three** different ways, which are given below:

1. User-Defined Variable
2. Local Variable
3. System Variable

User-Defined Variable

Sometimes, we want to pass values from one statement to another statement. The user-defined variable enables us to store a value in one statement and later can refer it to another statement. MySQL provides a **SET** and **SELECT** statement to declare and initialize a variable. The user-defined variable name starts with **@ symbol**.

The user-defined variables are not case-sensitive such as @name and @NAME; both are the same. A user-defined variable declares by one person cannot visible to another person. We can assign the user-defined variable into limited data types like integer, float, decimal, string, or NULL. The user-defined variable can be a maximum of **64 characters** in length.

Syntax

The following syntax is used to declare a user-defined variable.

1. By using the **SET** statement

```
SET @var_name = value;
```

NOTE: We can use either '=' or ':=' assignment operator with the SET statement.

2. By using the **SELECT** statement

```
SELECT @var_name := value;
```


Example1

Here, we are going to assign a value to a variable by using the SET statement.

```
mysql> SET @name='peter';
```

Then, we can display the above value by using the SELECT statement.

```
mysql> SELECT @name;
```

```
mysql> SET @name = 'peter';
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT @name;
+--------+
| @name  |
+--------+
| peter  |
+--------+
1 row in set (0.00 sec)
```

Example 2

Let us create table **students** in the MySQL database, as shown below:

studentid	firstname	lastname	class	age
1	Rinky	Ponting	12	20
2	Mark	Boucher	11	22
3	Sachin	Tendulkar	10	18
4	Peter	Fleming	10	22
5	Virat	Kohli	12	23
NULL	NULL	NULL	NULL	NULL

Run the following statement to get the maximum age of the student in the 'students' table and assign the age to the user-defined variable **@maxage**.

```
mysql> SELECT @maxage:= MAX(age) FROM students;
```

It will give the following output.

	@maxage: = MAX(age)
▶	23


```
mysql> SELECT firstname, lastname, age FROM students WHERE age = @maxage;
```

After successful execution of the above statement, we will get the following result:

	firstname	lastname	age
▶	Virat	Kohli	23

Example3

If we access the **undeclared** variable, it will give the **NULL** output.

```
Mysql> SELECT @var1;
```

Output

```
mysql> SELECT @var1;
+--------------+
| @var1        |
+--------------+
| 0x           |
+--------------+
1 row in set (0.00 sec)
```


Connection

A connection is a computer science facility that allows the user to connect with the database server software. **A user can connect with the database server, whether on the same machine or remote locations**. Therefore, if we want to work with the database server to send commands and receive answers in the form of a result set, we need connections. In this article, we are going to learn how we can connect to MySQL Server in various ways.

MySQL Connection Types

MySQL provides various ways to connect with the database server. **Once we have installed the MySQL server, we can connect it using any of the client programs that are listed below**:

1. Command-line client
2. MySQL Workbench
3. PHP Script.

MySQL Server Connection Using command-line client

MySQL command-line client program provides interaction with the database server in an interactive and non-interactive mode. We can see this program in the **bin directory of the MySQL's installation folder**. We can open the MySQL command prompt by navigating to the bin directory of the MySQL's installation folder and type:

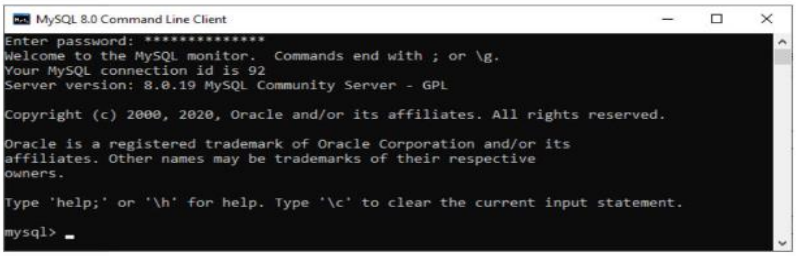
```
MySQL
```

If we find the MySQL program in the **PATH**, we can use the below command to connect to the MySQL Server:

```
mysql -u root -p
```

In the syntax, the **-u root indicates** that we will connect to the MySQL server using the root user account and **-p** instructs MySQL to ask for a password.

Next, we need to type the password for the root user account and press **Enter**. If everything is correct, it should give the screen as follows:

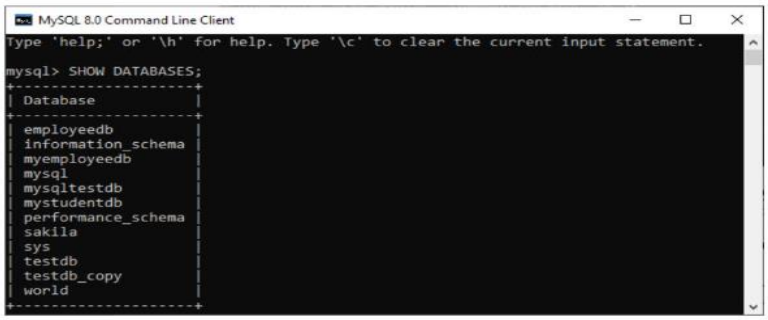


This screen indicates that we have successfully connected with the MySQL database server, where we can send commands and receive answers in the form of a result set.

Suppose we want to display all databases available in the current server; we can use the command as follows:

```
mysql> SHOW DATABASES;
```

It will give the below output:

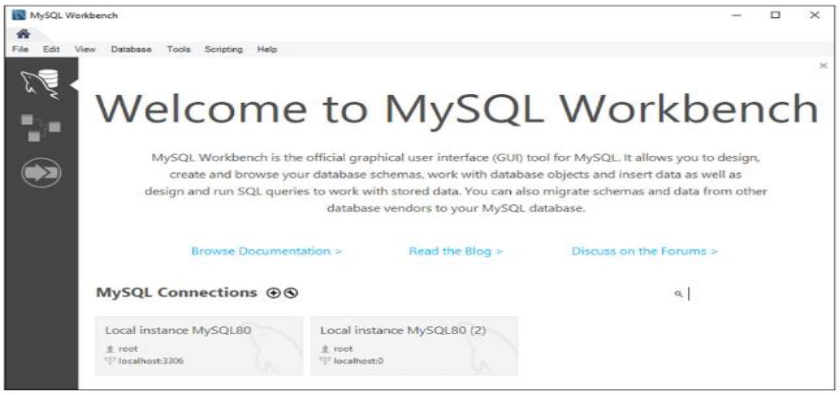


If you want **to disconnect the opened MySQL database server**, you need to use the exit command.

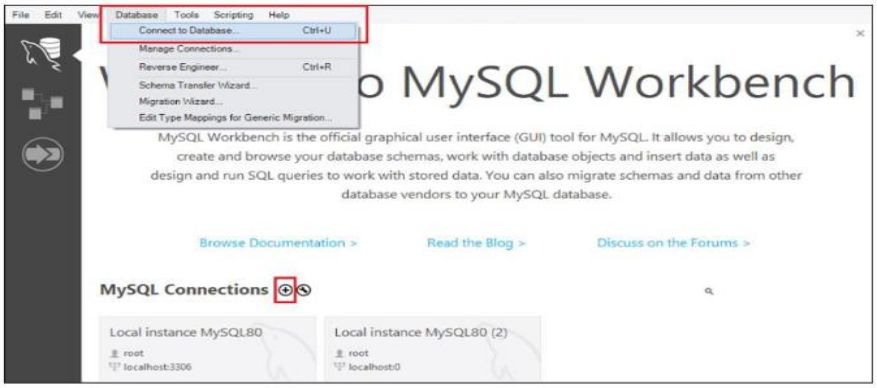
```
mysql> EXIT;
```


We can connect to the MySQL database server in workbench by using the following steps:

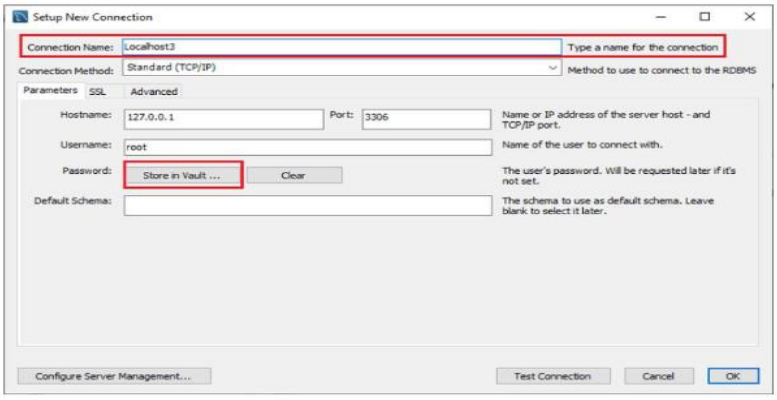
Step 1: Launch the MySQL Workbench. We should get the following screen:



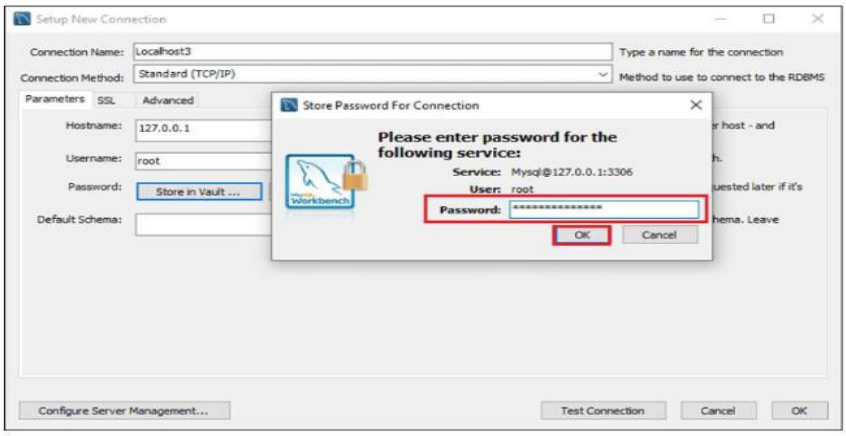
Step 2: Navigate to the menu bar, click on the **'Database'** and choose **Connect to Database** option or press the **CTRL+U** command. We can also connect with the database server by just clicking the **plus (+) button** located next to the MySQL Connections. See the below image:



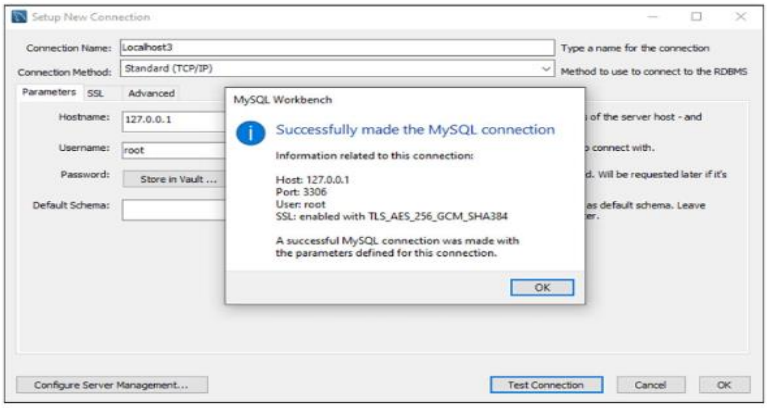
Step 4: Fill the box to create a connection, such as **connection name** and **username**, whatever you want. By default, the username is the **root**, but we can also change it with a different username in the Username textbox. After filling all boxes, click the **Store in Vault ... button** to write the password for the given user account.



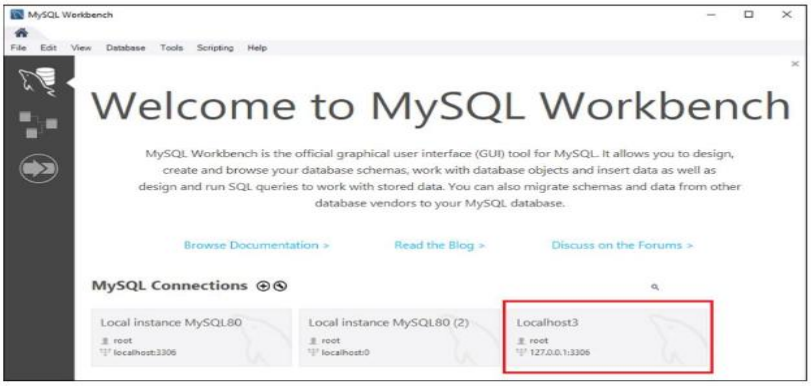
Step 5: We will get a new window to write the password and click the **OK** button.



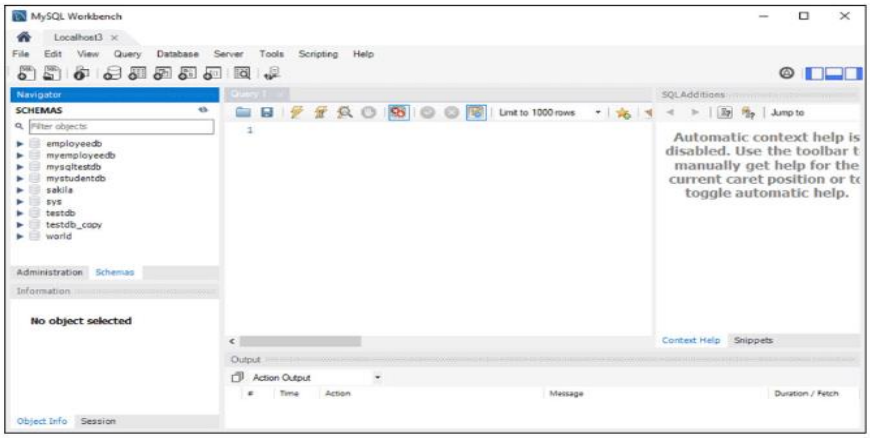
Step 6: After entering all the details, click on the **Test Connection** to test the database connectivity is successful or not. If the connection is successful, click on the **OK** button.



Step 7: Again, click on the **OK** button for saving connection setup. After finishing all the setup, we can see this connection under **MySQL Connections** for connecting to the MySQL database server. See the below output where we have **Localhost3** connection name:



Step 8: Now, we can click this newly created connection that displays the current schemas and a pane for entering queries:



Create database

A database is used to store the collection of records in an organized form. It allows us to hold the data into tables, rows, columns, and indexes to find the relevant information frequently. We can access and manage the records through the database very easily.

MySQL implements a database as a directory that stores all files in the form of a table. It allows us to create a database mainly in **two ways**:

1. MySQL Command Line Client
2. MySQL Workbench

MySQL Command Line Client

We can create a new database in MySQL by using the **CREATE DATABASE** statement with the below syntax:

```
CREATE DATABASE [IF NOT EXISTS] database_name
[CHARACTER SET charset_name]
[COLLATE collation_name];
```

Parameter Explanation

The parameter descriptions of the above syntax are as follows:

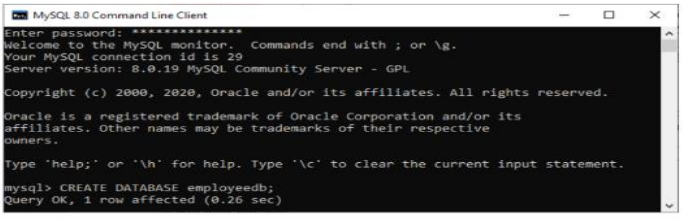
Parameter	Description
database_name	It is the name of a new database that should be unique in the MySQL server instance. The **IF NOT EXIST** clause avoids an error when we create a database that already exists.
charset_name	It is optional. It is the name of the character set to store every character in a string. MySQL database server supports many character sets. If we do not provide this in the statement, MySQL takes the default character set.
collation_name	It is optional that compares characters in a particular character set.

Example

Let us understand how to create a database in MySQL with the help of an example. Open the MySQL console and write down the password, if we have set during installation. Now we are ready to create a database. Here, we are going to create a database name **"employeedb"** using the following statement:

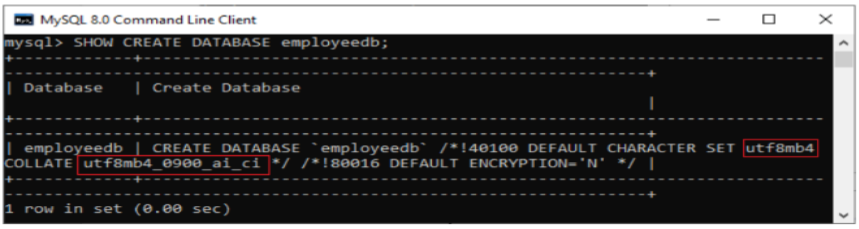
```
mysql> CREATE DATABASE employeesdb;
```

It will look like the below output:



We can review the newly created database using the below query that returns the database name, character set, and collation of the database:

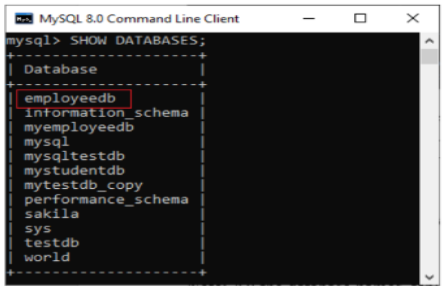
```
mysql> SHOW CREATE DATABASE employeedb;
```



We can check the created database using the following query:

```
mysql> SHOW DATABASES;
```

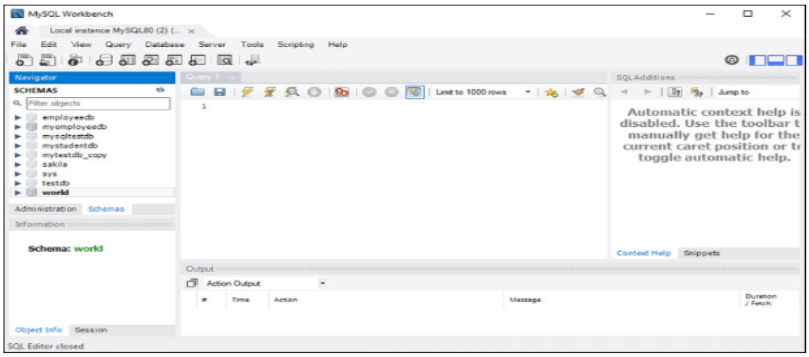
After executing the above query, we can see all the created databases in the server.



MySQL Workbench

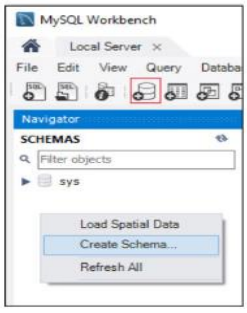
It is a visual database designing or GUI tool used to work with database architects, developers, and Database Administrators. This visual tool supports SQL development, data modeling, data migration, and comprehensive administration tools for server configuration, user administration, backup, and many more. It allows us to create new physical data models, E-R diagrams, and SQL development (run queries, etc.).

To create a new database using this tool, we first need to launch the MySQL Workbench and log in using the username and password that you want. It will show the following screen:

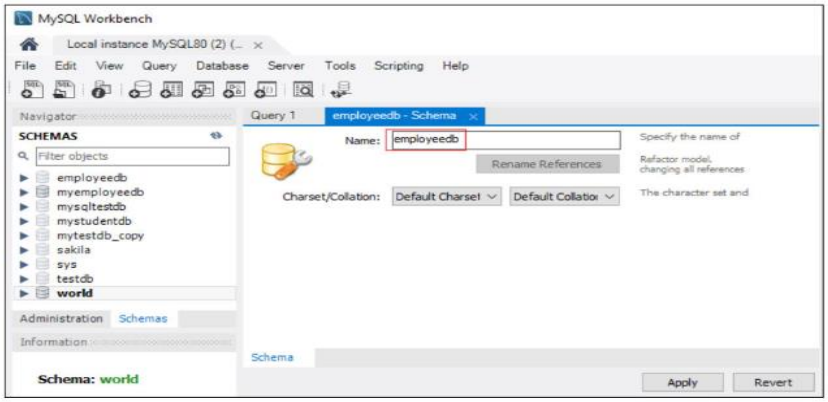


Now do the following steps for database creation:

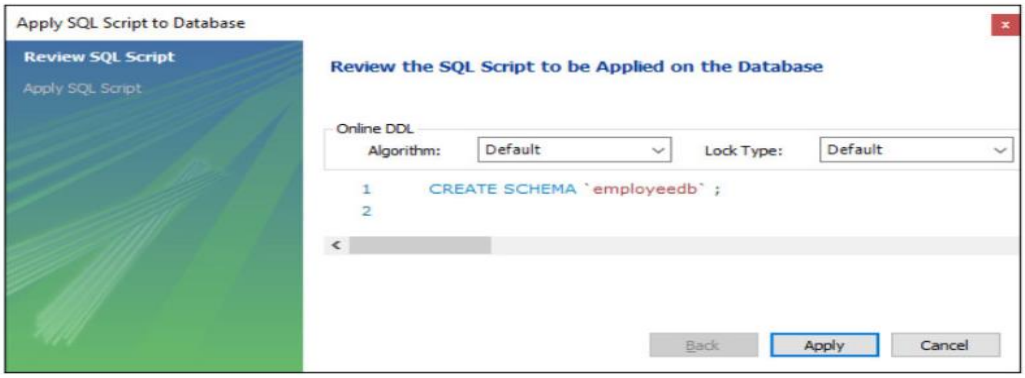
1. Go to the Navigation tab and click on the **Schema menu**. Here, we can see all the previously created databases. If we want to create a new database, right-click under the Schema menu and select Create Schema or click the database **icon (red rectangle)**, as shown in the following screen.



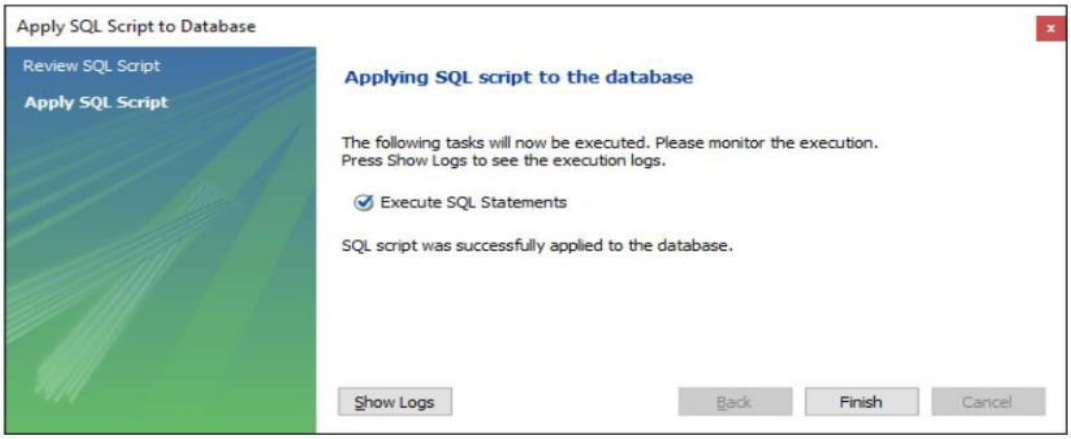
2. The new Schema window screen open. Enter the new database name (for example, **employeedb**) and use default character set and collation. Now, click on the Apply button as shown in the screen below:



3. A new popup window appears. Click on the **Apply** button.

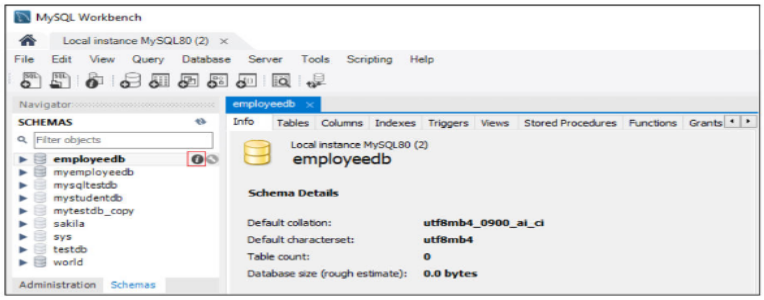


4. A new popup screen appears. Click on the **Finish** button to complete the database creation.



5. After successful database creation, we can see new databases in the Schema menu. If we do not see this, click on the **refresh icon** into the Schema menu.

6. We can see more information about the database by selecting the database and click on the **'i'** icon. The information window displays several options, like Table, Triggers, Indexes, Users, and many more.



7. MySQL Workbench does not provide an option to rename the database name, but we can create, update, and delete the table and data rows from the database.

Select Database

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

You can use SQL command **USE** to select a particular database.

Syntax:

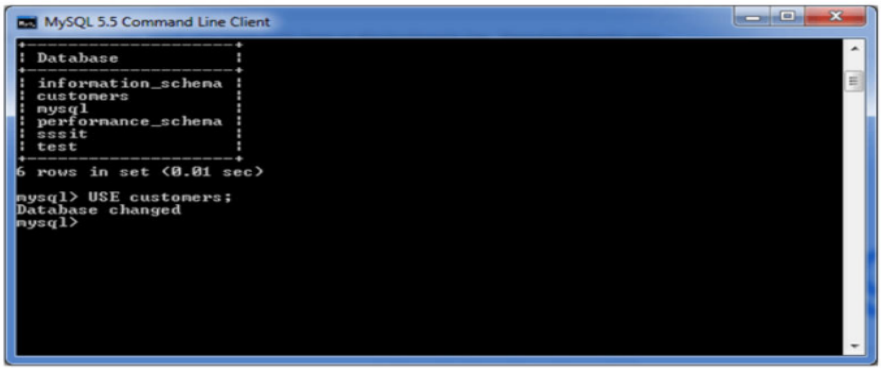
```
USE database_name;
```

Example:

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

```
USE customers;
```


It will look like this:



Show database

When we work with the MySQL server, it is a common task to show or list the databases, displaying the table from a particular database, and information of user accounts and their privileges that reside on the server. In this article, we are going to focus on how to list databases in the MySQL server.

We can list all the databases available on the MySQL server host using the following command, as shown below:

```
mysql> SHOW DATABASES;
```


Open the MySQL Command Line Client that appeared with a **mysql> prompt**. Next, **log in** to the MySQL database server using the **password** that you have created during the installation of MySQL. Now, you are connected to the MySQL server host, where you can execute all the SQL statements. Finally, run the SHOW Databases command to list/show databases.

We can see the following output that explains it more clearly:

MySQL also allows us another command to list the databases, which is a **SHOW SCHEMAS** statement. This command is the synonyms of the SHOW DATABASES and gives the same result. We can understand it with the following output:

List Databases Using Pattern Matching

Show Databases command in MySQL also provides an option that allows us to **filter** the returned database using different pattern matching with **LIKE** and **WHERE** clause. The LIKE clause list the database name that matches the specified pattern. The WHERE clause provides more flexibility to list the database that matches the given condition in the SQL statement.

Syntax

The following are the syntax to use pattern matching with Show Databases command:

```
mysql> SHOW DATABASES LIKE pattern;
OR,
mysql> SHOW DATABASES WHERE expression;
```

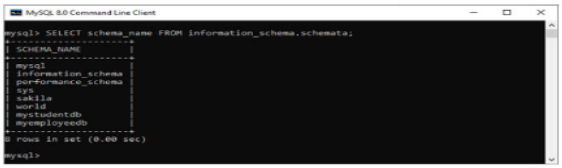
We can understand it with the example given below where **percent (%) sign** assumes zero, one, or multiple characters:

```
mysql> SHOW DATABASES LIKE "%schema";
```


Sometimes the LIKE clause is not sufficient; then, we can make a more complex search to query the database information from the schemata table in the information schema. The information schema in MySQL is an information database so that we can use it to get the output using the SHOW DATABASES command.

```
mysql> SELECT schema_name FROM information_schema.
```


This statement will give the same result as the SHOW DATABASES command:



Now, we are going to see how we can use the WHERE clause with the SHOW DATABASES command. This statement returns the database whose schema name starts with **"s":**

```
mysql>SELECT schema_name FROM information_schema.s
```

It will give the following output:

Drop database

We can drop/delete/remove a MySQL database quickly with the MySQL DROP DATABASE command. It will delete the database along with all the tables, indexes, and constraints permanently. Therefore, we should have to be very careful while removing the database in MySQL because we will lose all the data available in the database. If the database is not available in the MySQL server, the DROP DATABASE statement throws an error.

MySQL allows us to drop/delete/remove a database mainly in **two ways:**

- MySQL Command Line Client
- MySQL Workbench

MySQL Command Line Client

We can drop an existing database in MySQL by using the DROP DATABASE statement with the below syntax:

```
DROP DATABASE [IF EXISTS] database_name;
```

In MySQL, we can also use the below syntax for deleting the database. It is because the **schema** is the synonym for the database, so we can use them interchangeably.


```
DROP SCHEMA [IF EXISTS] database_name;
```

Parameter Explanation

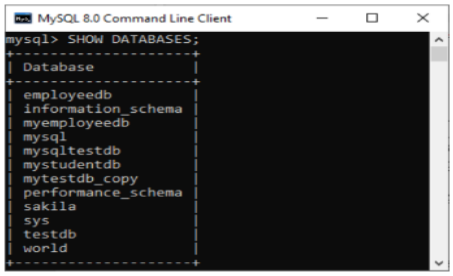
The parameter descriptions of the above syntax are as follows:

Parameter	Description
database_name	It is the name of an existing database that we want to delete from the server. It should be unique in the MySQL server instance.
IF EXISTS	It is optional. It is used to prevent from getting an error while removing a database that does not exist.

Example

Let us understand how to drop a database in MySQL with the help of an example. Open the MySQL console and write down the password, if we have set during installation. Now we are ready to delete a database.

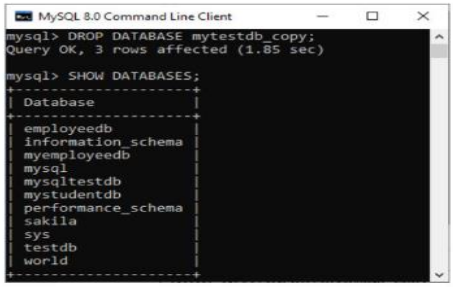
Next, use the **SHOW DATABASES** statement to see all available database in the server:



Suppose we want to remove a database named **"mytestdb_copy"**. Execute the below statement:


```
DROP DATABASE mytestdb_copy;
```

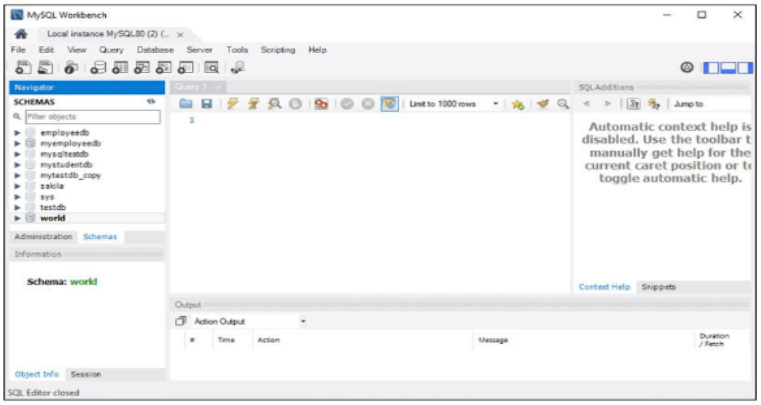
Now we can verify that either our database is removed or not by executing the following query. It will look like this:



From the above, we can see that the database "mytestdb_copy" is removed successfully.

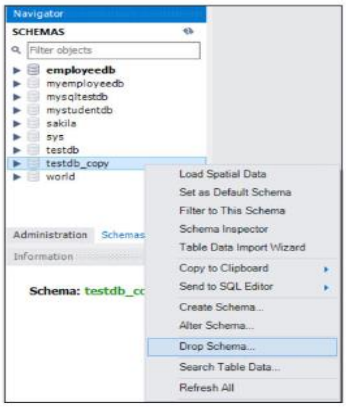
DROP Database using MySQL Workbench

To drop a database using this tool, we first need to launch the MySQL Workbench and log in with the **username** and **password** to the MySQL server. It will show the following screen:

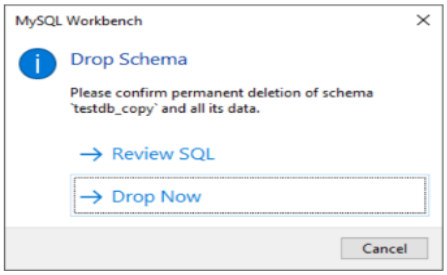


Now do the following steps for database deletion:

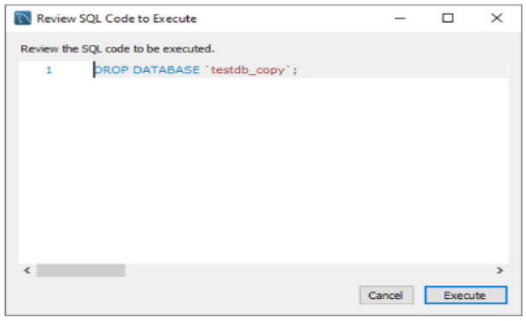
1. Go to the Navigation tab and click on the **Schema menu**. Here, we can see all the previously created databases. If we want to delete a database, right-click the database that you want to remove, for example, **testdb_copy** under the Schema menu and select **Drop Schema** option, as shown in the following screen.



When we click the Drop Schema option, MySQL Workbench displays a dialog box to confirm the deletion process. If we select **Review SQL**, it will produce the SQL statement that will be executed. And if we choose **Drop Now** option, the database will be deleted permanently.



If we want the safe deletion of the database, it is required to choose the Review SQL option. Once we sure, click the Execute button to execute the statement. The below screen explains it more clearly:



Once we click the execute button, MySQL will return the below message indicating that the database is dropped successfully. Since the database testdb_copy is an empty database, the number of affected rows is zero.

Copy Database

A database is an application used for storing the organized collection of records that can be accessed and manage by the user. It holds the data into tables, rows, columns, and indexes to quickly find the relevant information.

MySQL copy or clone database is a feature that allows us to create a **duplicate copy of an existing database**, including the table structure, indexes, constraints, default values, etc. Making a duplicate copy of an original database into a new database is very useful when accidentally our database is **lost or failure**. The most common use of making a duplicate copy of the database is for data backups. It is also useful when planning the major changes to the structure of the original database.

In MySQL, making the clone of an original database is a **three-step process:** First, the original database records are dumped (copied) to a temporary file that holds the SQL commands for reinserting the data into the new database. Second, it is required to create a new database. Finally, the SQL file is processed, and the data will be copied into the new database.

1. First, use the **CREATE DATABASE** statement to create a new database.
2. Second, store the data to an **SQL file**. We can give any name to this file, but it must end with a **.sql** extension.
3. Third, export all the database objects along with its data to copy using the **mysqldump** tool and then import this file into the new database.

For the demonstration, we will copy the **testdb** database to **testdb_copy** database using the following steps:

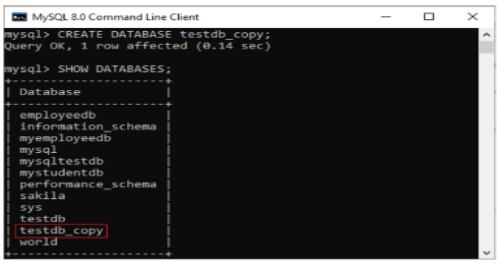
Open the MySQL console and write down the password, if we have set during installation. Now we are ready to create a duplicate database of testdb using the command below:

```
mysql> CREATE DATABASE testdb_copy;
```

Next, use the SHOW DATABASES statement for verification:

```
mysql> SHOW DATABASES;
```

This command will return all available database in the server where we can see the newly created database in red rectangle box:



Now, open a DOS or terminal window to access the MySQL server on the command line. For example, if we have installed the MySQL in the **C folder**, copy the following folder and paste it in our DOS command. Then, press the **Enter** key.

```
C:\Users\javatpoint> CD C:\Program Files\MySQL\MySQL Server 8.0\bin
```


In the next step, we need to use the mysqldump tool to copy the database objects and data into the SQL file. Suppose we want to dump (copy) the database objects and data of the testdb into an SQL file located at **D:\Database_backup folder.** To do this, execute the below statement:

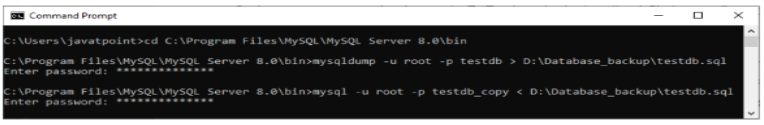

```
mysqldump -u root -p testdb > D:\Database_backup\testdb.sql
Enter password: **********
```

The above statement instructs mysqldump tool to log in to the MySQL database server using the username and password and then exports the database objects and data of the testdb database to **D:\Database_backup\testdb.sql.** It is to note that the operator (>) used for exporting the database from one location to another.

In the next step, we need to import the D:\Database_backup\testdb.sql file into testdb_copy database. To do this, execute the below statement:

```
mysql -u root -p testdb_copy < D:\Database_backup\testdb.sql
Enter password: **********
```

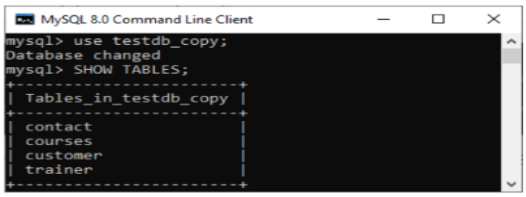
It is to note that the operator (<) used for importing the database from one location to another.



Finally, we can verify whether the above operation is successful or not by using the **SHOW TABLES** command in the MySQL command-line tool:

Finally, we can verify whether the above operation is successful or not by using the **SHOW TABLES** command in the MySQL command-line tool:

```
mysql> SHOW TABLES;
```



In this output, we can see that all the objects and data from the testdb database to testdb_copy database have successfully copied.

Create Table

A table is used to organize data in the form of rows and columns and used for both storing and displaying records in the structure format. It is similar to worksheets in the spreadsheet application. A table creation command requires **three things**:

- Name of the table
- Names of fields
- Definitions for each field

MySQL allows us to create a table into the database mainly in **two ways**:

1. MySQL Command Line Client
2. MySQL Workbench

MySQL Command Line Client

MySQL allows us to create a table into the database by using the **CREATE TABLE** command. Following is a generic **syntax** for creating a MySQL table in the database.

```
CREATE TABLE [IF NOT EXISTS] table_name(
   column_definition1,
   column_definition2,
   .....,
   table_constraints
);
```

The parameter descriptions of the above syntax are as follows:

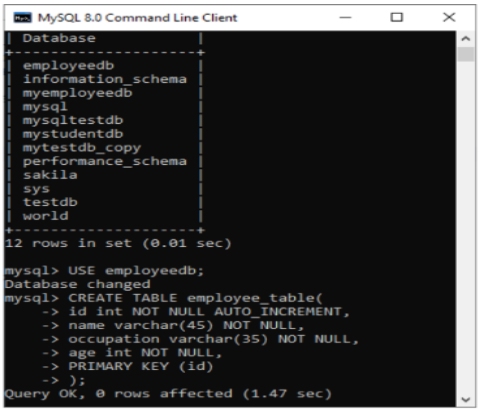
Parameter	Description
database_name	It is the name of a new table. It should be unique in the MySQL database that we have selected. The **IF NOT EXIST** clause avoids an error when we create a table into the selected database that already exists.
column_definition	It specifies the name of the column along with data types for each column. The columns in table definition are separated by the comma operator. The syntax of column definition is as follows: **column_name1 data_type(size) [NULL | NOT NULL]**
table_constraints	It specifies the table constraints such as PRIMARY KEY, UNIQUE KEY, FOREIGN KEY, CHECK, etc.

Example

Let us understand how to create a table into the database with the help of an example. Open the MySQL console and write down the password, if we have set during installation. Now open the database in which you want to create a table. Here, we are going to create a table name **"employee_table"** in the database **"employeedb"** using the following statement:


```
mysql> CREATE TABLE employee_table(
   id int NOT NULL AUTO_INCREMENT,
   name varchar(45) NOT NULL,
   occupation varchar(35) NOT NULL,
   age int NOT NULL,
   PRIMARY KEY (id)
);
```

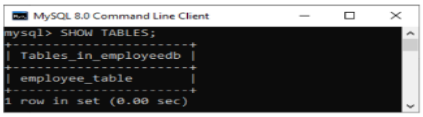

Visual representation of creating a MySQL table:



We need to use the following command to see the newly created table:

```
mysql> SHOW TABLES;
```

It will look like the below output:

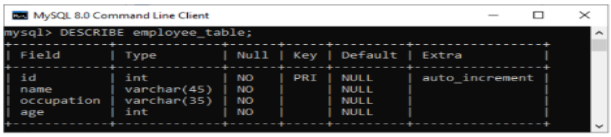


See the table structure:

We can use the following command to see the information or structure of the newly created table:

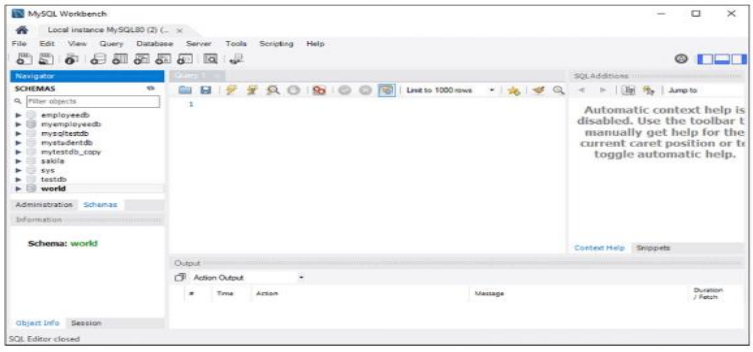
```
mysql> DESCRIBE employee_table;
```

It will look like this:



Create Table Using MySQL Workbench

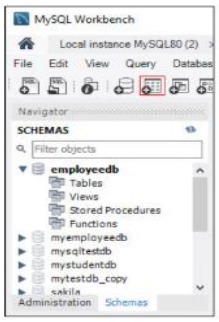
It is a visual GUI tool used to create databases, tables, indexes, views, and stored procedures quickly and efficiently. To create a new database using this tool, we first need to launch the MySQL Workbench and log in using the username and password that you want. It will show the following screen:



Now do the following steps for table creation:

1. Go to the Navigation tab and click on the **Schema menu**. Here, we can see all the previously created databases. Now we are ready to select the database in which a table is created.

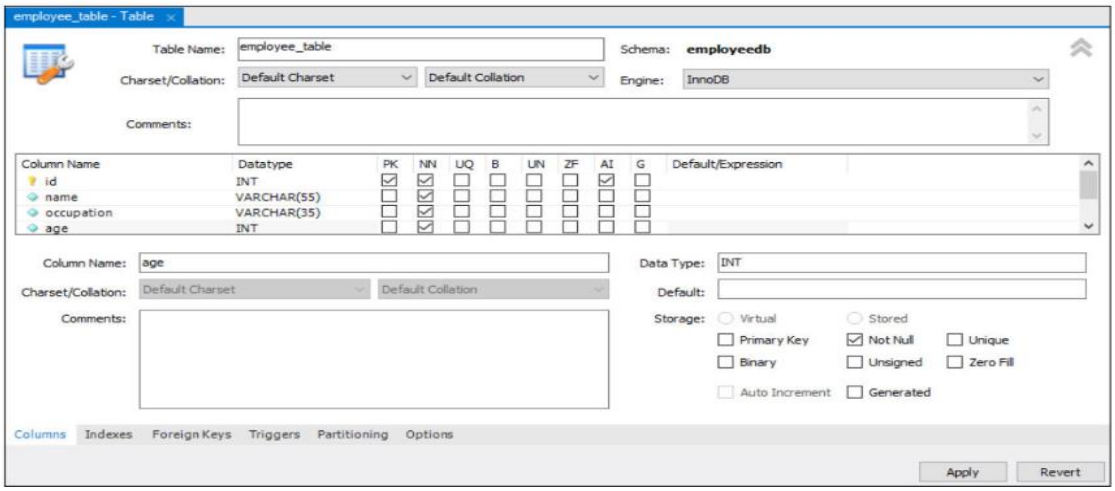
2. Select the database, double click on it, and we will get the sub-menu under the database. These **sub-menus** are Tables, Views, Functions, and Stored Procedures, as shown in the below screen.



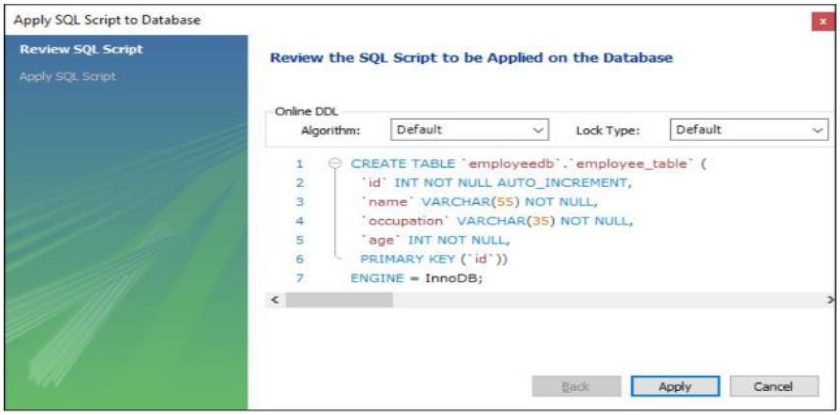
3. Select Tables sub-menu, right-click on it, and select **Create Table** option. We can also click on create a new table icon (shown in red rectangle) to create a table.

4. On the new table screen, we need to fill all the details to create a table. Here, we will enter the table name (**for example**, employee_table) and use default collation and engine.

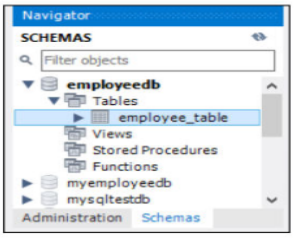
5. Click inside the middle window and fill the column details. Here, the column name contains many attributes such as Primary Key(PK), Not Null (NN), Unique Index (UI), Binary(B), Unsigned Data type(UN), Auto Incremental (AI), etc. The following screen explains it more clearly. After filling all the details, click on the **Apply** button.



6. As soon as you click on the Apply button, it will open the SQL statement window. Again, click on the Apply button to execute the statement and **Finish** button to save the changes.



7. Now, go to the Schema menu and select the database which contains the newly created table, as shown in the screen below.



Update Query

MySQL UPDATE query is a DML statement used to modify the data of the MySQL table within the database. In a real-life scenario, records are changed over a period of time. So, we need to make changes in the values of the tables also. To do so, it is required to use the UPDATE query.

The UPDATE statement is used with the **SET** and **WHERE** clauses. The SET clause is used to change the values of the specified column. We can update single or multiple columns at a time.

Syntax

Following is a generic syntax of UPDATE command to modify data into the MySQL table:

```
UPDATE table_name
SET column_name1 = new-value1,
      column_name2=new-value2, ...
[WHERE Clause]
```


Parameter	Descriptions
table_name	It is the name of a table in which we want to perform updation.
column_name	It is the name of a column in which we want to perform updation with the new value using the SET clause. If there is a need to update multiple columns, separate the columns with a comma operator by specifying the value in each column.
WHERE Clause	It is optional. It is used to specify the row name in which we are going to perform updation. If we omit this clause, MySQL updates all rows.

Note:

- This statement can update values in a single table at a time.
- We can update single or multiple columns altogether with this statement.
- Any condition can be specified by using the WHERE clause.
- WHERE clause is very important because sometimes we want to update only a single row, and if we omit this clause, it accidentally updates all rows of the table.

The UPDATE command supports these modifiers in MySQL:

LOW_PRIORITY: This modifier instructs the statement to delay the UPDATE command's execution until no other clients reading from the table. It takes effects only for the storage engines that use only table-level locking.

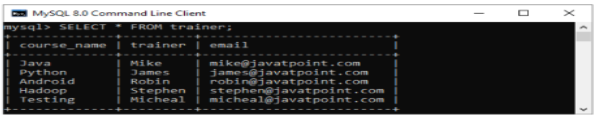
IGNORE: This modifier allows the statement to do not abort the execution even if errors occurred. If it finds **duplicate-key** conflicts, the rows are not updated.

Therefore, the full syntax of UPDATE statement is given below:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_name
  SET column_assignment_list
  [WHERE condition]
```


Example:

Let us understand the UPDATE statement with the help of various examples. Suppose we have a table **"trainer"** within the **"testdb"** database. We are going to update the data within the "trainer" table.



Update Single Column

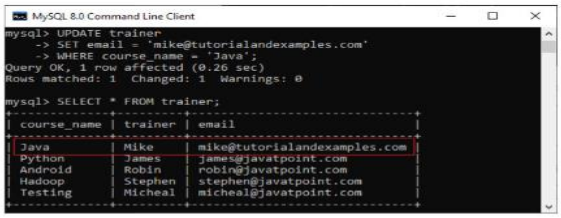
This query will update the **email id of Java** course with the new id as follows:

```
UPDATE trainer
SET email = 'mike@tutorialandexamples.com'
WHERE course_name = 'Java';
```

After successful execution, we will verify the table using the below statement:

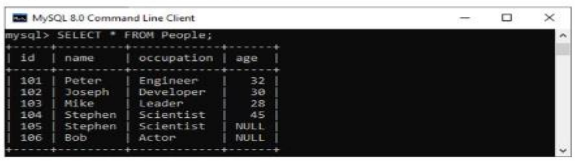
```
SELECT * FROM trainer;
```


In the output, we can see that our table is updated as per our conditions.



Update Multiple Columns

The UPDATE statement can also be used to update multiple columns by specifying a comma-separated list of columns. Suppose we have a table as below:



This statement explains will update the **name** and **occupation** whose **id = 105** in the **People** table as follows:

```
UPDATE People
SET name = 'Mary', occupation = 'Content Writer'
WHERE id = 105;
```

We can verify the output below:

Delete

A column is a series of cells in a table that may contain text, numbers, and images. Every column stores one value for each row in a table. In this section, we are going to discuss how to add or delete columns in an existing table.

How can we add a column in MySQL table?

MySQL allows the **ALTER TABLE ADD COLUMN** command to add a new column to an existing table. The following are the syntax to do this:

```
ALTER TABLE table_name
   ADD COLUMN column_name column_definition [FIRST|AFTER existing_column];
```


- First, we need to specify the table name.
- Next, after the ADD COLUMN clause, we have to specify the name of a new column along with its definition.
- Finally, we need to specify the FIRST or AFTER keyword. The FIRST Keyword is used to add the column as the first column of the table. The AFTER keyword is used to add a new column after the existing column. If we have not provided these keywords, MySQL adds the new column as the last column in the table by default.

Sometimes it is required to add **multiple columns** into the existing table. Then, we can use the syntax as follows:

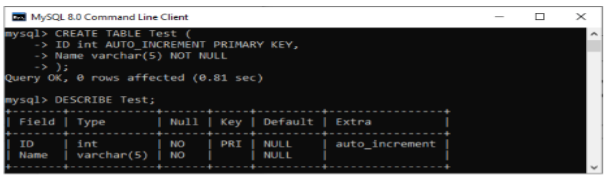
```
ALTER TABLE table_name
   ADD COLUMN column_name1 column_definition [FIRST|AFTER existing_column],
   ADD COLUMN column_name2 column_definition [FIRST|AFTER existing_column];
```

MySQL ADD COLUMN Example

Let us understand it with the help of various examples. Here, we will create a table named **"Test"** using the following statements:

```
CREATE TABLE Test (
   Stude_id int AUTO_INCREMENT PRIMARY KEY,
   Name varchar(55) NOT NULL
);
```

The table structure looks like the below image:



After creating a table, we want to add a new column named City to the Test table. Since we have not specified the new column position explicitly after the column name, MySQL will add it as the last column.

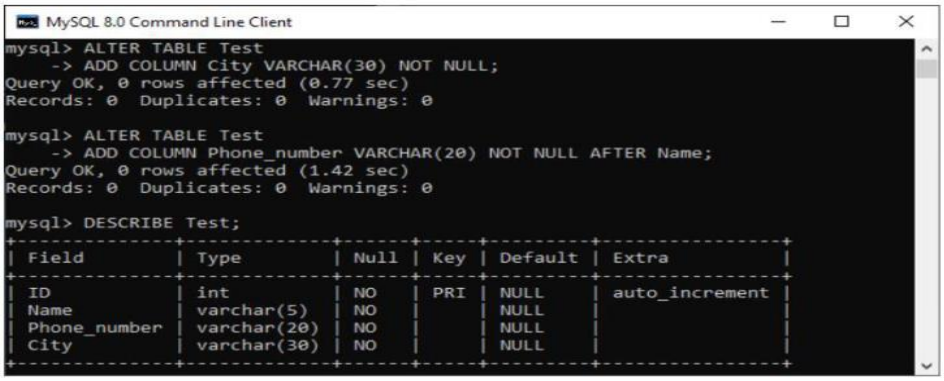
```
ALTER TABLE Test
ADD COLUMN City VARCHAR(30) NOT NULL;
```

Next, we want to add a new column named **Phone_number** to the **Test** table. This time, we will explicitly specify the new column position so that MySQL adds the column to the specified place.

```
ALTER TABLE Test
ADD COLUMN Phone_number VARCHAR(20) NOT NULL AFTER Name;
```

In the below output, we can see that the two columns are added successfully at the specified position.

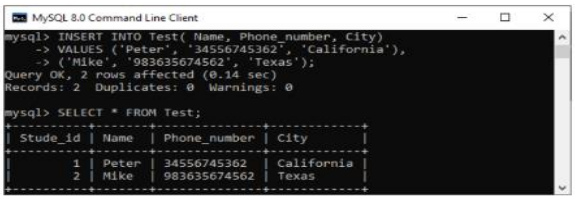
In the below output, we can see that the two columns are added successfully at the specified position.



Let us add some data into the Test table using the INSERT statement as follows:


```
INSERT INTO Test( Name, Phone_number, City)
VALUES ('Peter', '34556745362', 'California'),
('Mike', '983635674562', 'Texas');
```

It will look like this.

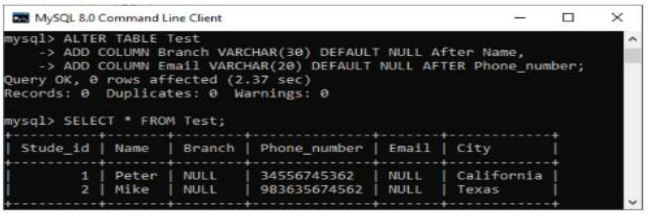


Suppose we want to add more than one column ,**(Branch, Email)** in the Test table. In that case, execute the statement as follows:

```
ALTER TABLE Test
ADD COLUMN Branch VARCHAR(30) DEFAULT NULL After Name,
ADD COLUMN Email VARCHAR(20) DEFAULT NULL AFTER Phone_number;
```

It is to note that columns Branch and Email are assigned to default value **NULL**. However, the Test table already has data so that MySQL will use null values for those new columns.

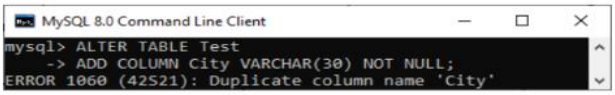
We can verify the record in the Test table as below:



If we accidentally add a new column with the existing column name, MySQL will **throw an error**. For example, execute the below statement that issues an error:

```
ALTER TABLE Test
ADD COLUMN City VARCHAR(30) NOT NULL;
```


We will get the following error message.



How can we rename a column in MySQL table?

MySQL allows the **ALTER TABLE CHANGE COLUMN** statement to change the old column with a new name. The following are the syntax to do this:

```
ALTER TABLE table_name
  CHANGE COLUMN old_column_name new_column_name column_definition [FIRST|AFTER existing_column];
```


In the above,

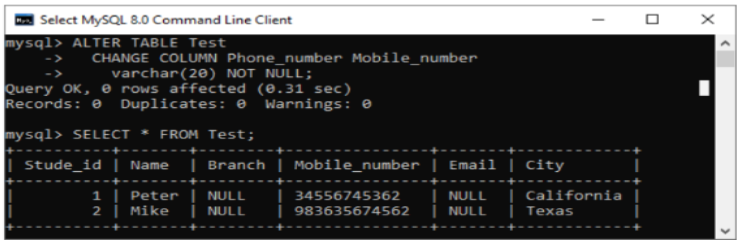
- First, we need to specify the table name.
- Next, after the CHANGE COLUMN clause, we have to specify the old column name and new column name along with its definition. We must have to specify the column definition even it will not change.
- Finally, we need to specify the FIRST or AFTER keyword. It is optional that specified when we need to change the column name at the specific position.

MySQL RENAME COLUMN Example

This example shows how we can change the column name in the MySQL table:

```
ALTER TABLE Test
  CHANGE COLUMN Phone_number Mobile_number
    varchar(20) NOT NULL;
```

This statement will change the column name **Phone_number** with the new name **Mobile_number** in the Test table. The below output explains it more clearly.



How can we drop a column from MySQL table?

Sometimes, we want to remove single or multiple columns from the table. MySQL allows the **ALTER TABLE DROP COLUMN** statement to delete the column from the table. The following are the syntax to do this:

```
ALTER TABLE table_name  DROP COLUMN column_name;
```

In the above,

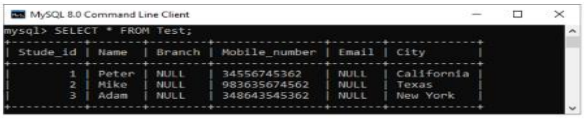
- First, we need to specify the **table name** from which we want to remove the column.
- Next, after the **DROP COLUMN** clause, we have to specify the column name that we want to delete from the table. It is to note that the COLUMN keyword is optional in the DROP COLUMN clause.

If we want to remove **multiple columns** from the table, execute the following statements:

```
ALTER TABLE table_name
DROP COLUMN column_1,
DROP COLUMN column_2,
.....;
```

MySQL DROP COLUMN Example

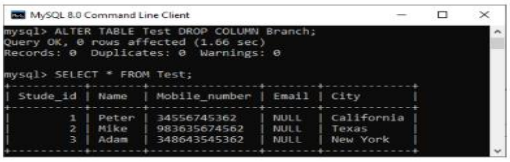
This example explains how we can delete a column from the MySQL table. Here, we will take a table **"Test"** that we have created earlier and look like the below image:



Suppose we want to delete a column name **"Branch"** from the Test table. To do this, execute the below statement:

```
ALTER TABLE Test DROP COLUMN Branch;
```

After successful execution, we can verify the result below where a column Branch is deleted from the table:



In some cases, it is required to remove multiple columns from the table. To do this, we need to execute the below statement:

```
ALTER TABLE Test
   DROP COLUMN Mobile_number,
   DROP COLUMN Email;
```

The command will delete both columns. We can verify it using the queries given in the below image.