Introduction to database

You deal with data every day…

When you want to listen to your favorite songs, you open your playlist from your smartphone. In this case, the playlist is a database.

When you take a photo and upload it to your account on a social network like Facebook, your photo gallery is a database.

When you browse an e-commerce website to buy shoes, clothes, etc., you use the shopping cart database.

Databases are everywhere. So what is a database?  By definition, a database is merely a structured collection of data.

The data relating to each other by nature, e.g., a product belonged to a product category and associated with multiple tags. Therefore, we use the term **relational database**.

In the relational database, we model data like products, categories, tags, etc., using tables. A table contains columns and rows. It is like a spreadsheet.

A table may relate to another table using a relationship, e.g., one-to-one and one-to-many relationships.

Because we deal with a significant amount of data, we need a way to define the databases, tables, etc., and process data more efficiently. Besides, we want to turn the data into information.

And this is where SQL comes to play.

SQL – the language of the relational database

SQL stands for the structured query language.

SQL is the standardized language used to access the database.

ANSI/SQL defines the SQL standard. The current version of SQL is SQL:2016. Whenever we refer to the SQL standard, we mean the current SQL version.

SQL contains three parts:

1. Data definition language includes statements that help you define the database and its objects, e.g., tables, [views](https://www.mysqltutorial.org/mysql-views-tutorial.aspx),[triggers](https://www.mysqltutorial.org/mysql-triggers.aspx), [stored procedures](https://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx), etc.
2. Data manipulation language contains statements that allow you to [update](https://www.mysqltutorial.org/mysql-update-data.aspx) and [query data](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx).
3. Data control language allows you to [grant the permissions](https://www.mysqltutorial.org/mysql-grant.aspx) to a user to access specific data in the database.

Now, you understand database and SQL, and it’s time to answer the next question…

What is MySQL

MySQL? What?

My is the daughter’s name of the [MySQL’s co-founder, Monty Widenius](https://en.wikipedia.org/wiki/Michael_Widenius).

The name of MySQL is the combination of My and SQL, MySQL.

MySQL is a database management system that allows you to manage relational databases. It is open source software backed by Oracle. It means you can use MySQL without paying a dime. Also, if you want, you can change its source code to suit your needs.

Even though MySQL is open source software, you can buy a commercial license version from Oracle to get premium support services.

MySQL is pretty easy to master in comparison with other database software like Oracle Database, or Microsoft SQL Server.

MySQL can run on various platforms UNIX, Linux, Windows, etc. You can install it on a server or even in a desktop. Besides, MySQL is reliable, scalable, and fast.

The official way to pronounce MySQL is *My Ess Que Ell, not My Sequel.*However, you can pronounce it whatever you like, who cares?

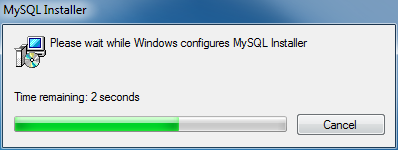
If you develop websites or web applications, MySQL is a good choice. MySQL is an essential component of the LAMP stack, which includes Linux, Apache, MySQL, and PHP.

# **Install MySQL**

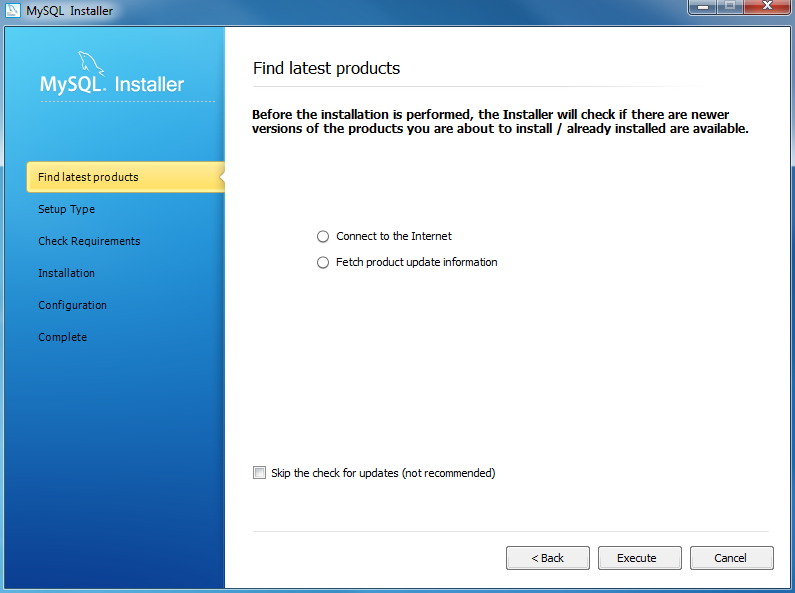
To download MySQL installer, go to the following link <http://dev.mysql.com/downloads/installer/>.

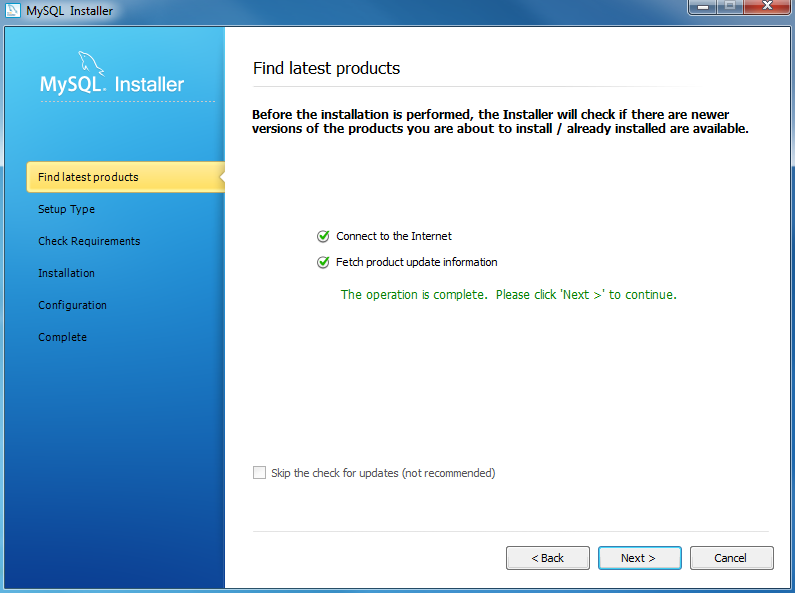
## Install MySQL via MySQL Installer

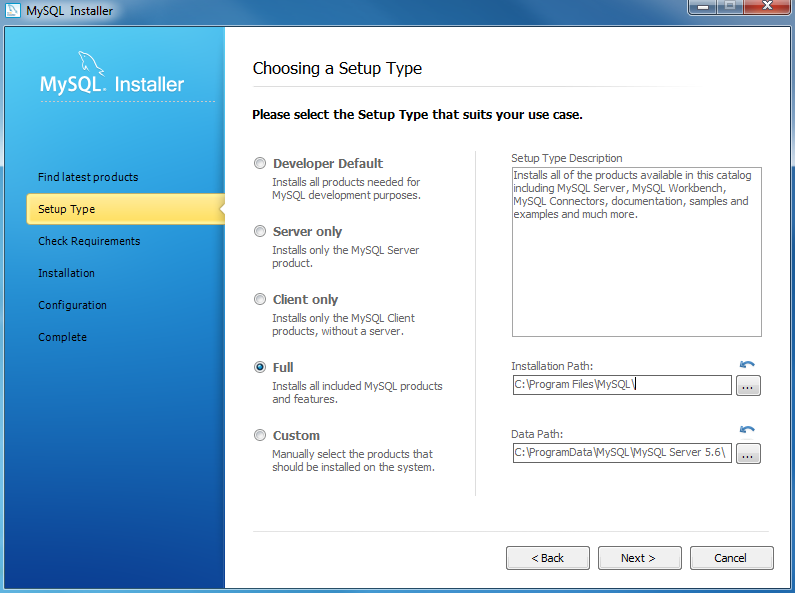
To install MySQL using the MySQL installer, double-click on the MySQL installer file and follow the steps below:

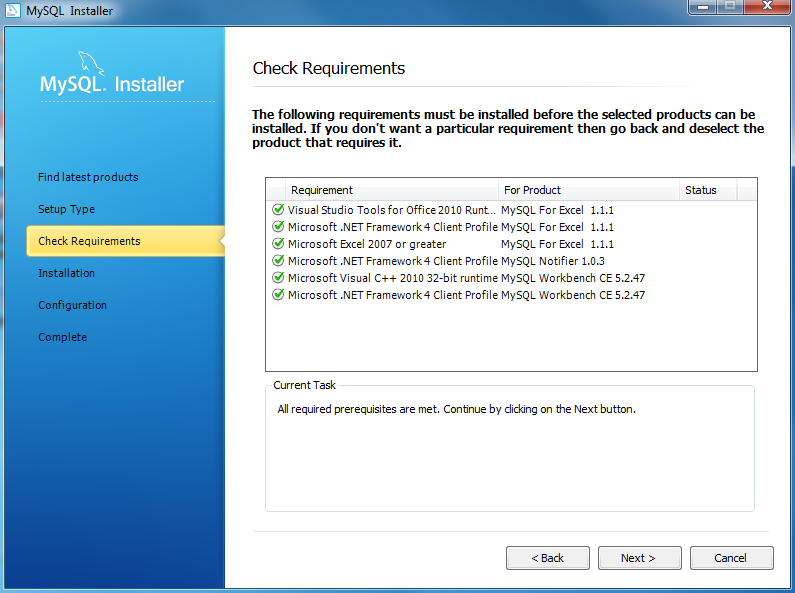
  
Install MySQL Step 1: Windows configures MySQL Installer

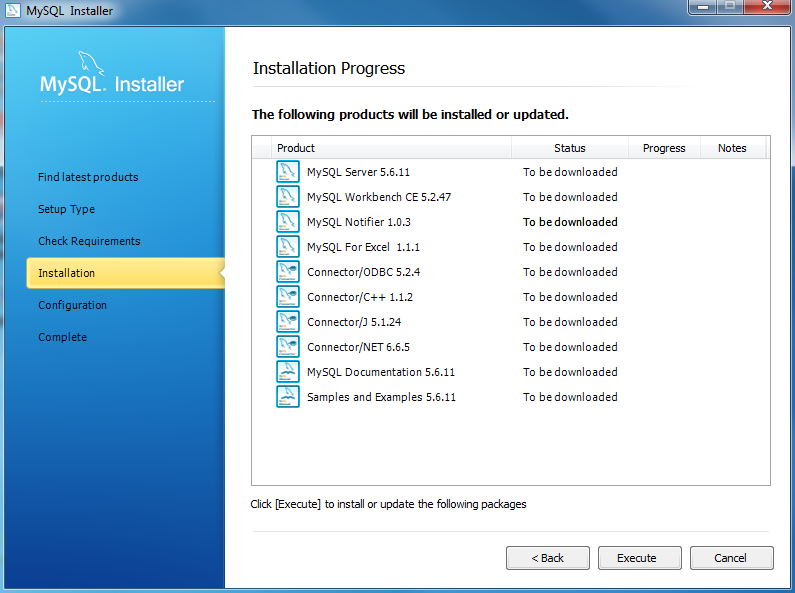
  
Install MySQL Step 2 – Welcome Screen: A welcome screen provides several options. Choose the first option: Install MySQL Products

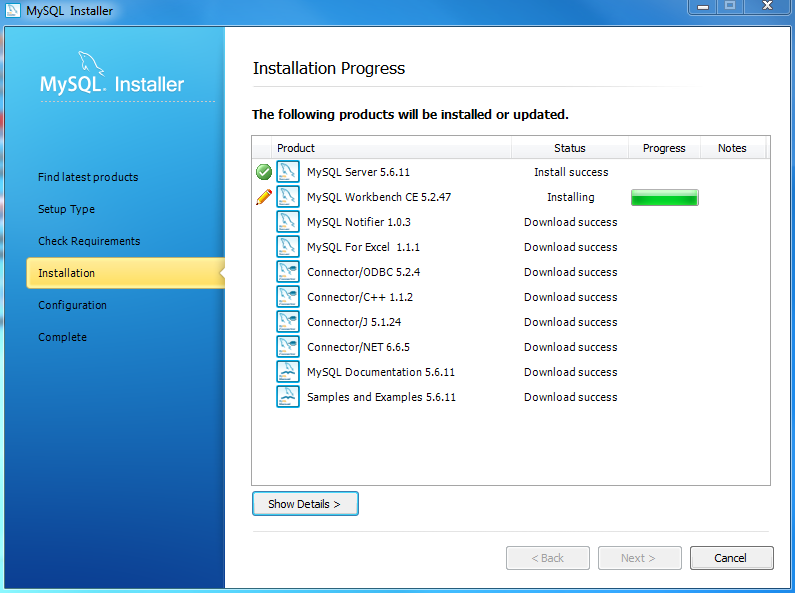
  
Install MySQL Step 3 – Download the latest MySQL products: MySQL installer checks and downloads the latest MySQL products including MySQL server, MySQL Workbench, etc.

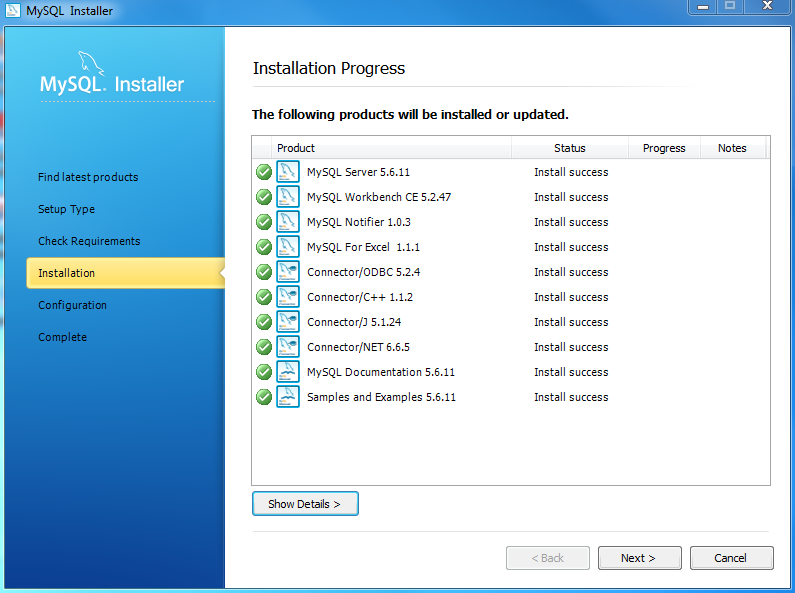
  
Install MySQL Step 4: Click the Next button to continue

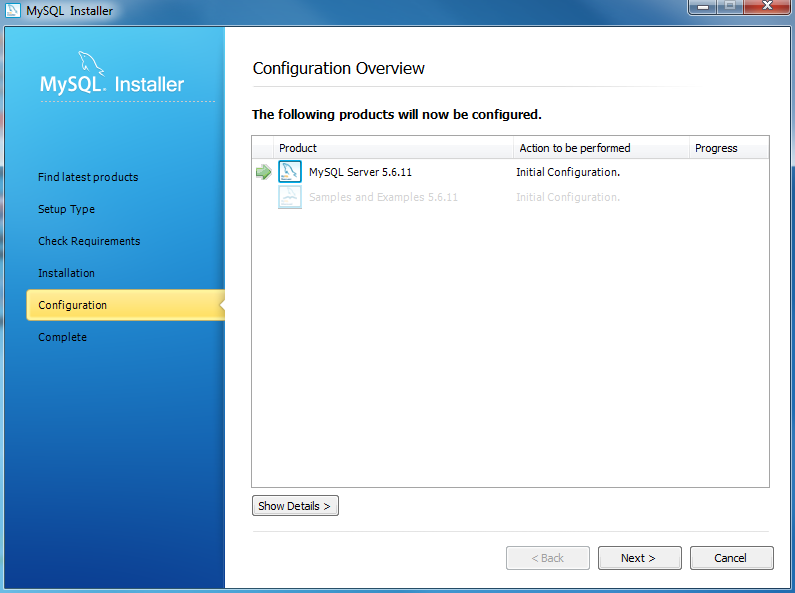
  
Install MySQL Step 5 – Choosing a Setup Type: there are several setup types available. Choose the Full option to install all MySQL products and features.

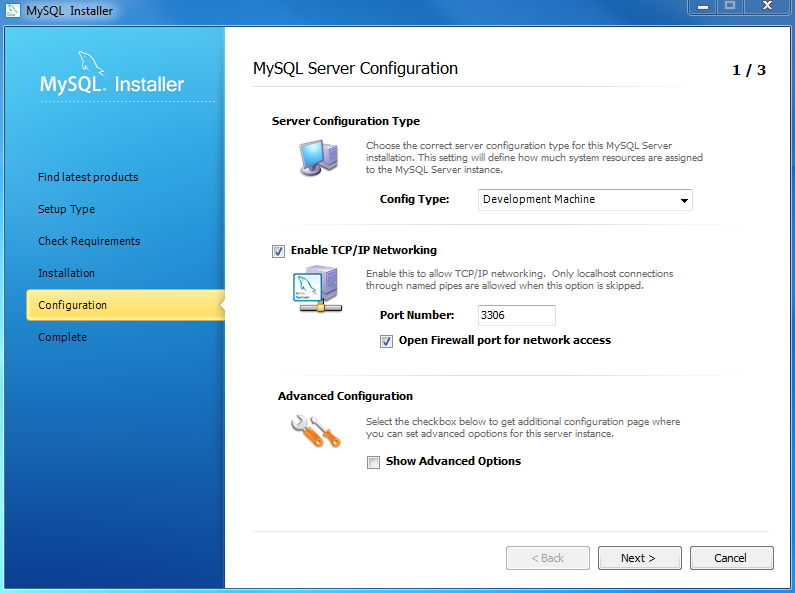
  
Install MySQL Step 6 – Checking Requirements

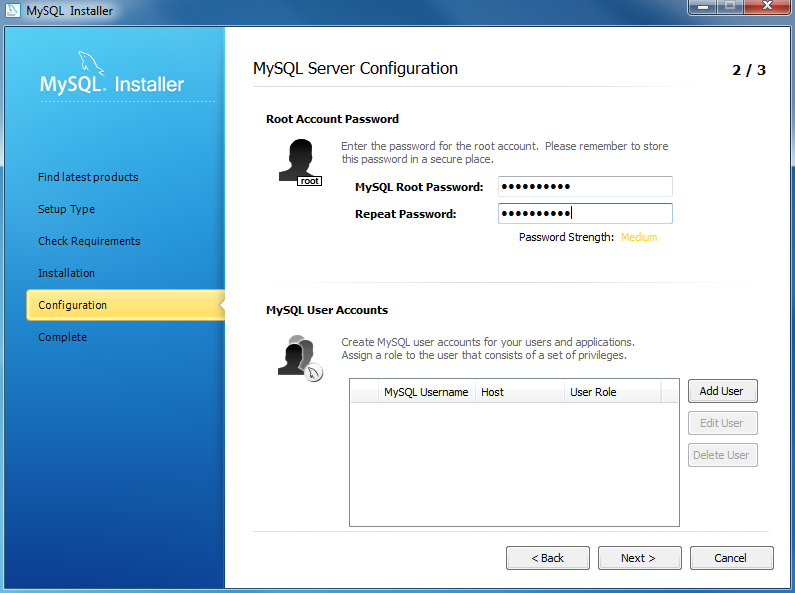
  
Install MySQL Step 7 – Installation Progress: MySQL Installer downloads all selected products. It will take a while, depending on which products you selected and the speed of your internet connection.

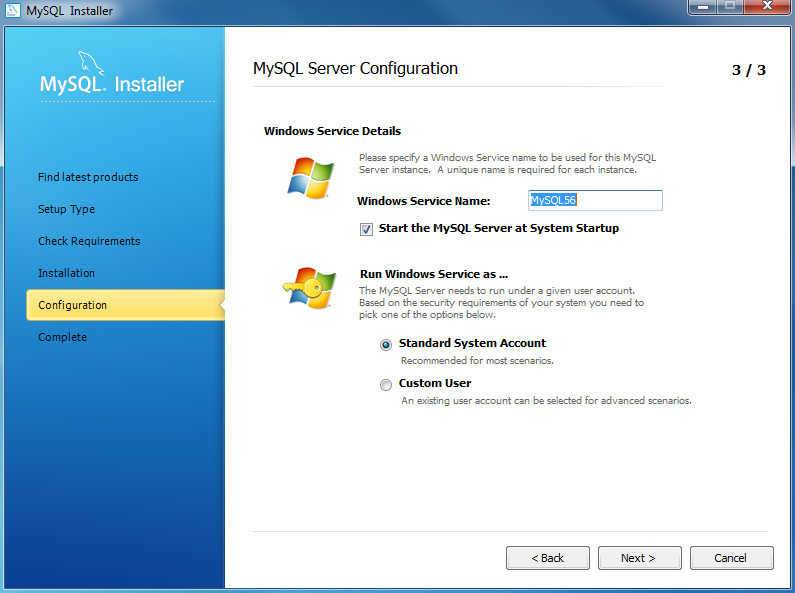
  
Install MySQL Step 7 – Installation Progress: downloading Products in progress.

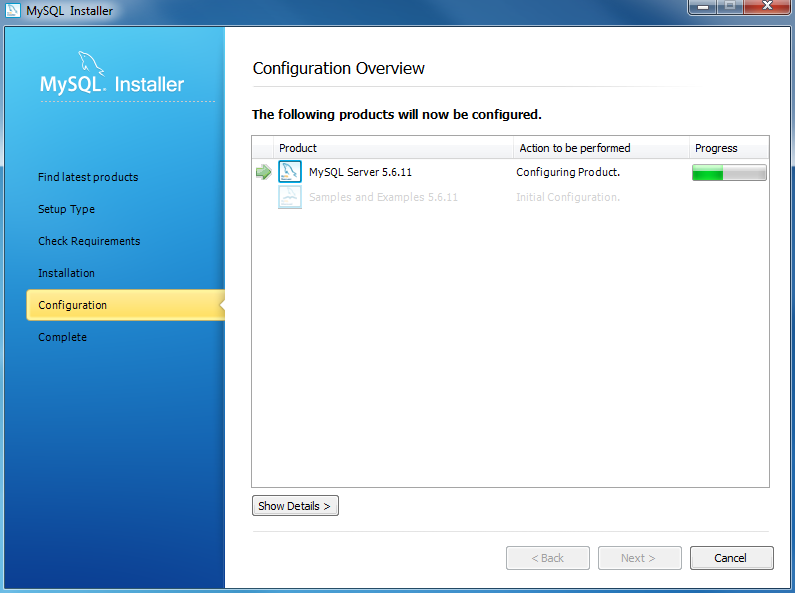
  
Install MySQL Step 7 – Installation Progress: Complete Downloading. Click the **Next** button to continue…

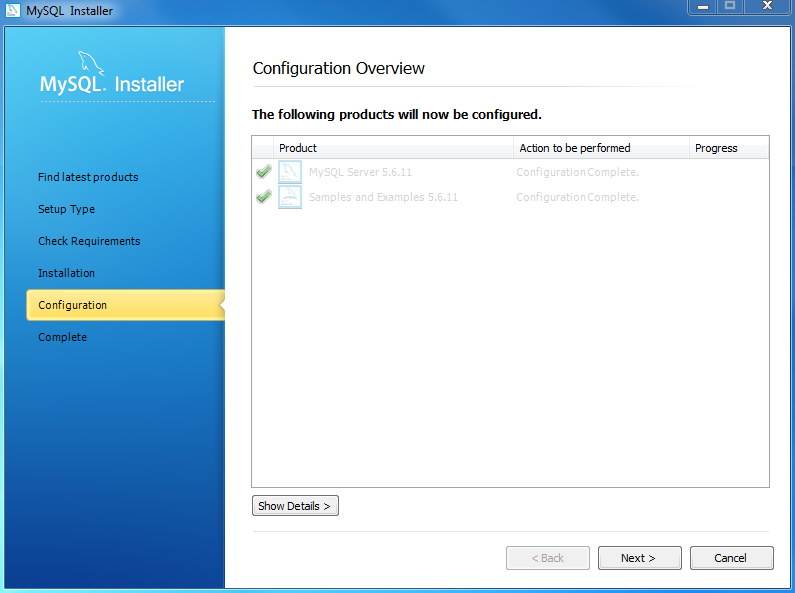
  
Install MySQL Step 8 – Configuration Overview. Click the Next button to configure MySQL Database Server

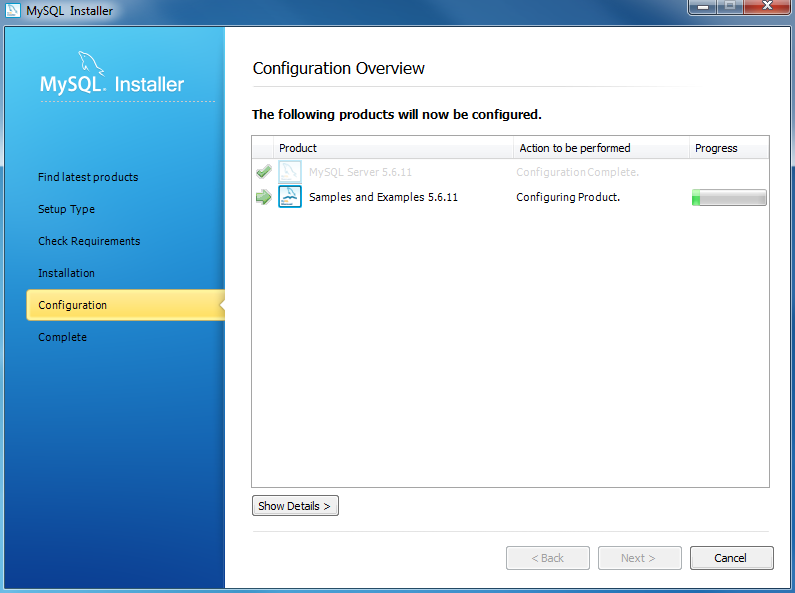
  
Install MySQL Step 8.1 – MySQL Server Configuration: choose Config Type and MySQL port (3006 by default) and click Next button to continue.

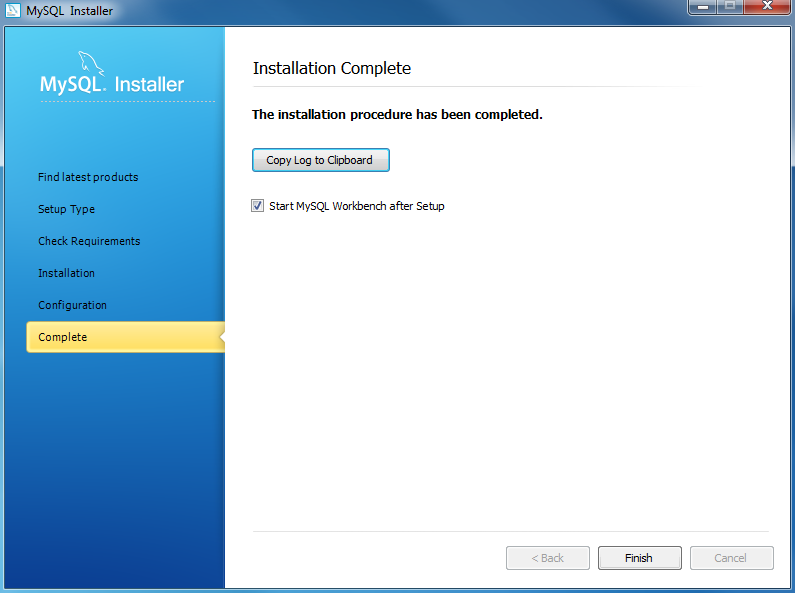
  
Install MySQL Step 8.1 – MySQL Server Configuration: choose a password for the root account. Please note the password download and keep it securely if you are installing MySQL database server on a production server. If you want to add a more MySQL user, you can do it in this step.

  
Install MySQL Step 8.1 – MySQL Server Configuration: choose Windows service details including Windows Service Name and account type, then click Next button to continue.

  
Install MySQL Step 8.1 – MySQL Server Configuration – In Progress: MySQL Installer is configuring MySQL database server. Wait until it is done and click the Next button to continue.

  
Install MySQL Step 8.1 – MySQL Server Configuration – Done. Click the Next button to continue.

  
Install MySQL Step 8.2 – Configuration Overview: MySQL Installer installs sample databases and sample models



Install MySQL Step 9 – Installation Completes: the installation completes. Click the **Finish** button to close the installation wizard and launch the MySQL Workbench.

# **Connect to MySQL Server**

Once you have the [MySQL Server installed](https://www.mysqltutorial.org/install-mysql/), you can connect to it using any client program such as mysql command-line client and MySQL workbench.

## Connect to MySQL Using mysql command-line client

mysql is a command-line client program that allows you to interact with MySQL in the interactive and non-interactive mode.

The mysql command-line client is typically located in the bin directory of the MySQL’s installation folder.

To invoke the mysql program, you just simply navigate to the bin directory of the MySQL’s installation folder and type:

|  |  |
| --- | --- |
| 1 | mysql |

If the mysql program is already in the PATH, you can simply invoke it using mysql command.

To connect to the MySQL Server, you use this command:

|  |  |
| --- | --- |
| 1 | shell>mysql -u root -p |

-u root means that you connect to the MySQL Server using the user account root.

-p instructs mysql to prompt for a password.

You type the password for the user account root and press Enter:

|  |  |
| --- | --- |
| 1 | Enter password: \*\*\*\*\*\*\*\* |

If everything is OK, you will connect to the MySQL Server with the following command:

|  |  |
| --- | --- |
| 1 | mysql> |

To display the databases in the current server, you use the [SHOW DATABASES](https://www.mysqltutorial.org/mysql-show-databases/) statement:

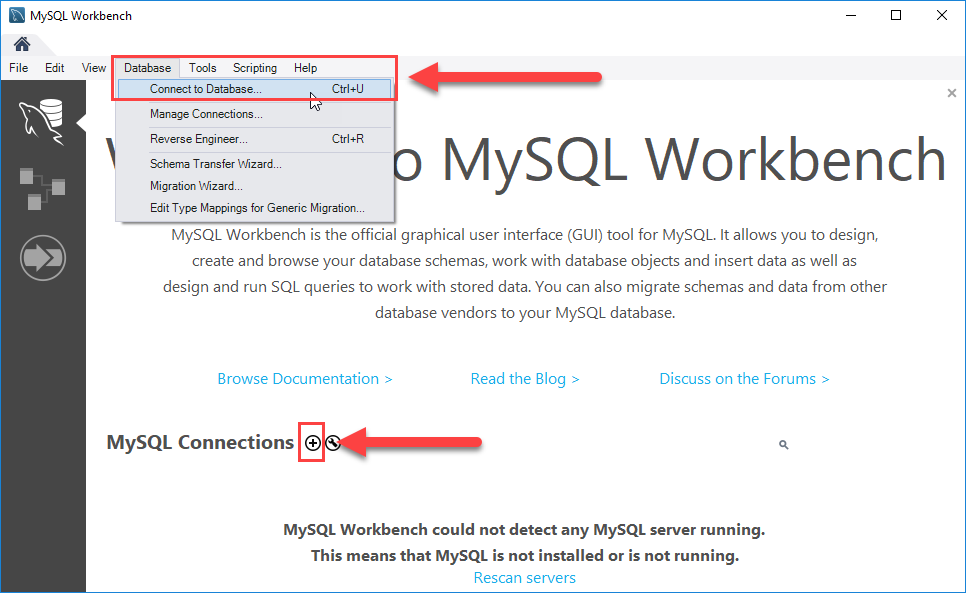
|  |  |
| --- | --- |
| 1 | mysql> show databases; |

Here is the output:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | +--------------------+  | Database           |  +--------------------+  | information\_schema |  | mysql              |  | performance\_schema |  | sys                |  +--------------------+  4 rows in set (0.01 sec) |

## Connect to MySQL Using MySQL Workbench

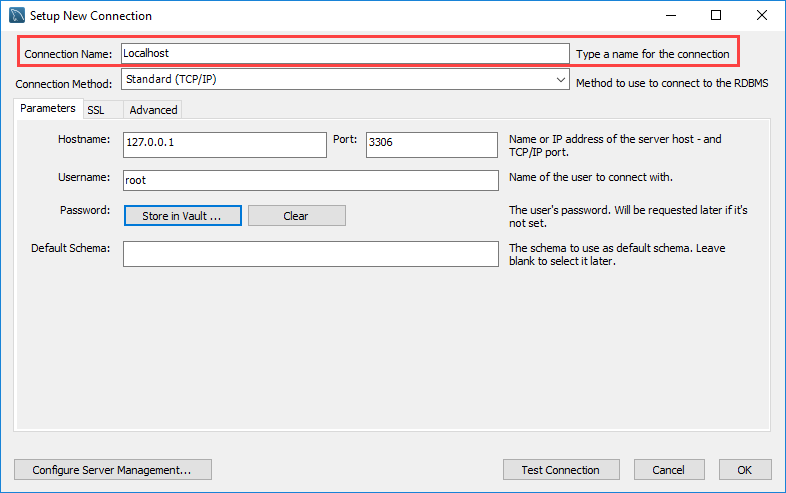
Step 1. Launch the MySQL Workbench.



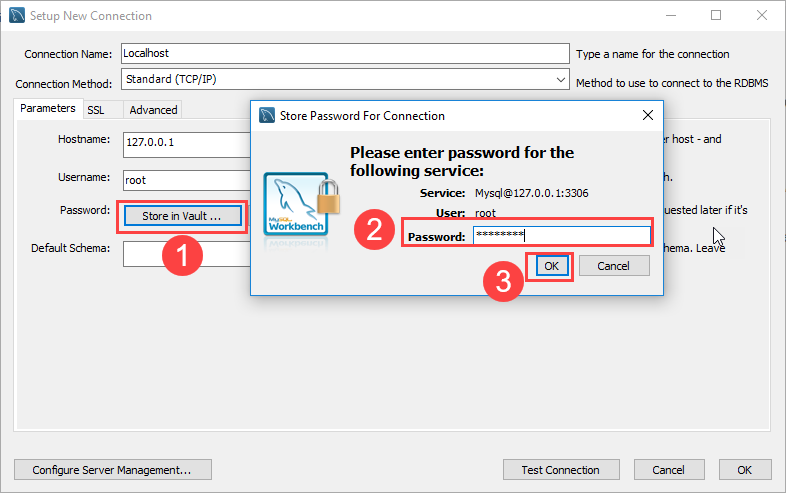
You can connect to a MySQL Server using the Database > Connect to Database… menu or click the + button that locates next to the MySQL Connections.

Just click the + button in next to the MySQL Connections to continue.

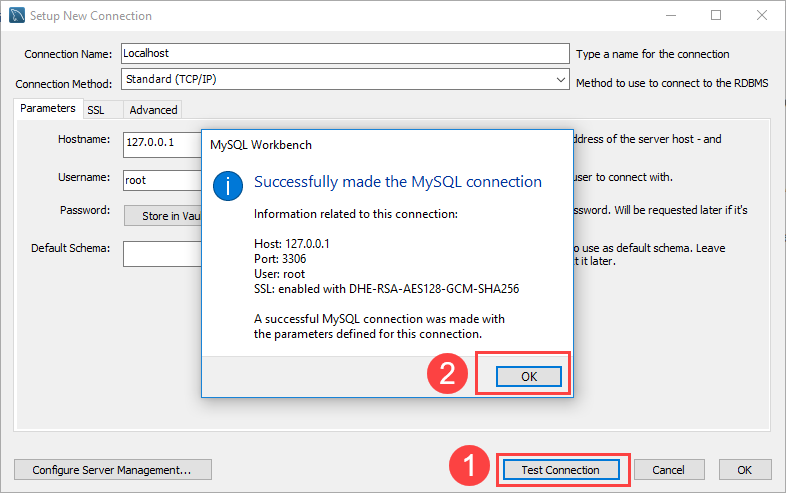
Step 2. Enter the connection name e.g., Localhost. You can name it whatever makes sense to you. By default, the username is root. If you use a different user account, you can change it in the Username textbox.



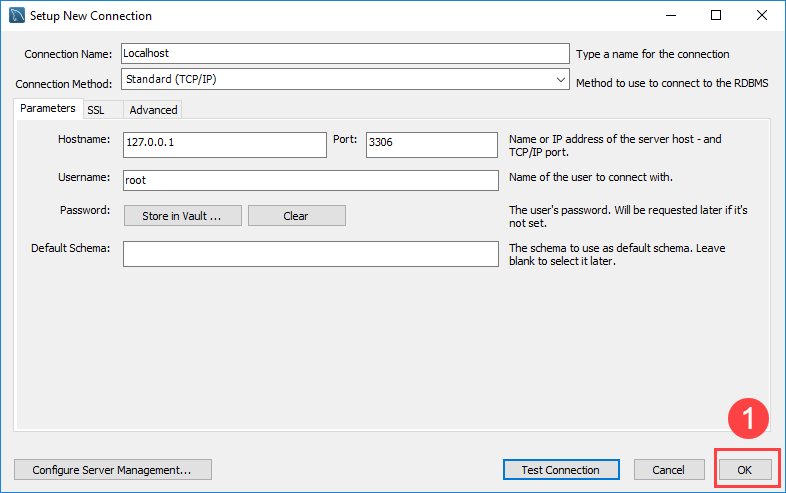
Step 3. Click the Store in Vault ... button to enter the password for the provided user account. A window will display. You enter the password and click the OK button.



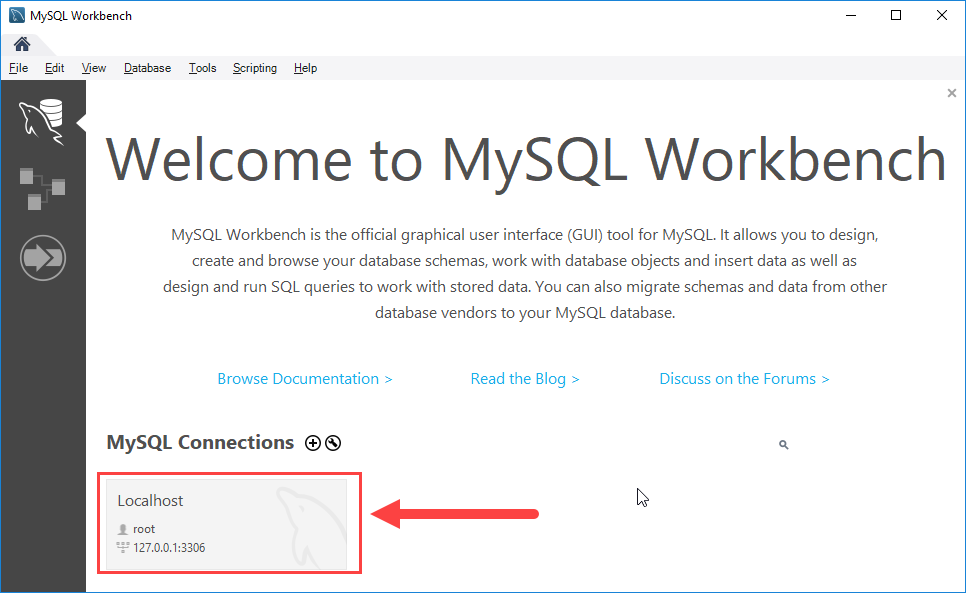
Step 4. Click the Test Connection button to test if the connection to the MySQL Server is successful or not. Then click the OK button if the connection is established successfully.



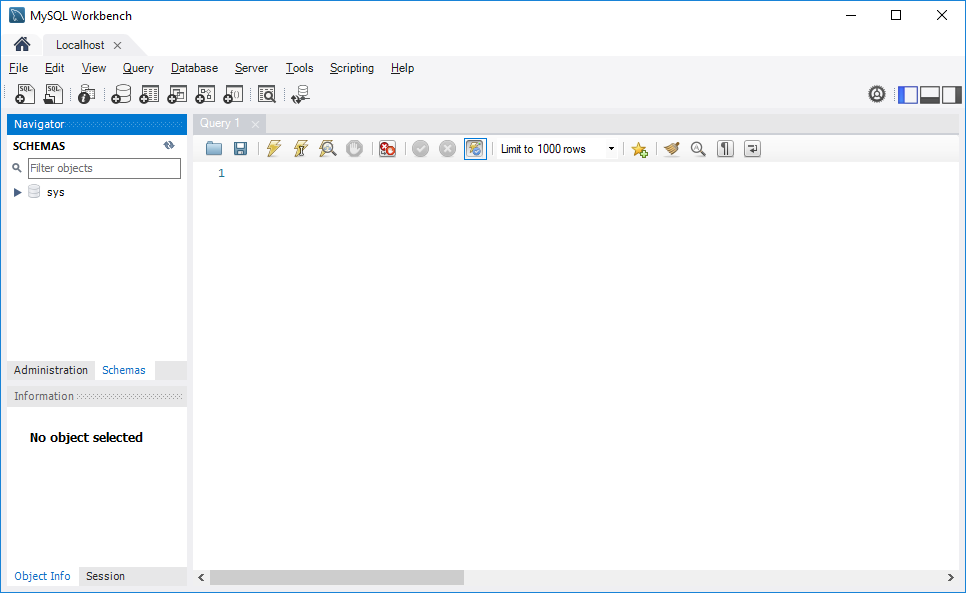
Step 5. Click the OK button to save the connection.



Step 6. Click the newly created connection under MySQL Connections to connect to the MySQL Server:



Step 7. MySQL Workbench display with the current schemas and a pane for entering queries:



# **MySQL Sample Database**

## Download MySQL Sample Database

<https://sp.mysqltutorial.org/wp-content/uploads/2018/03/mysqlsampledatabase.zip>

After uncompressing the  sampledatabase.zip file, you can load the sample database into MySQL database server  as:

### **Step 1**

Download the classicmodels database from the [MySQL sample database](https://www.mysqltutorial.org/mysql-sample-database.aspx) section.

### **Step 2**

Unzip the downloaded file into a temporary folder. You can use any folder you want. To make it simple, we will unzip the file to the C:\temp  folder.

If you use another operating system such as macOS, Linux, or Unix, please feel free to unzip it to any directory you like.

### **Step 3**

Connect to the MySQL server using the mysql client program. The mysql program is located in the bindirectory of the MySQL installation folder.

|  |  |
| --- | --- |
| 1  2 | > mysql -u root -p  Enter password: \*\*\*\*\*\*\*\* |

You will need to enter the password for the root user account to log in.

### **Step 4**

Use the source command to load data into the MySQL Server:

|  |  |
| --- | --- |
| 1 | mysql> source c:\temp\mysqlsampledatabase.sql |

### **Step 5**

Use the [SHOW DATABASES](https://www.mysqltutorial.org/mysql-show-databases/) command to list all databases in the current server:

|  |  |
| --- | --- |
| 1 | mysql> show databases; |

The output will look like the following that includes the newly created classicmodels database:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | +--------------------+  | Database           |  +--------------------+  | classicmodels      |  | information\_schema |  | mysql              |  | performance\_schema |  | sys                |  +--------------------+ |

test it by using the following SQL statements:

|  |  |
| --- | --- |
| 1  2 | USE classicmodels;  SELECT \* FROM customers; |

Basically, those statements switch the current database to classicmodels and query data from the customers table. If you see the customer data returned, you have successfully imported the sample database into the MySQL database server.

MySQL Sample Database Schema

The MySQL sample database schema consists of the following tables:

* **Customers**: stores customer’s data.
* **Products**: stores a list of scale model cars.
* **ProductLines**: stores a list of product line categories.
* **Orders**: stores sales orders placed by customers.
* **OrderDetails**: stores sales order line items for each sales order.
* **Payments**: stores payments made by customers based on their accounts.
* **Employees**: stores all employee information as well as the organization structure such as who reports to whom.
* **Offices**: stores sales office data.



# **MySQL CREATE DATABASE**

CREATE DATABASE [IF NOT EXISTS] database\_name

## Creating a new database using mysql program

To create a new database via the mysql program, you use the following steps:

First, log in to the MySQL Server using the root user

|  |  |
| --- | --- |
| 1  2 | >mysql -u root -p  Enter password: \*\*\*\*\*\*\*\* |

Type the password for the root user and press Enter.

Next, to display the existing database in the server to make sure that you are not creating a new database that already exists, you use the SHOW DATABASES command as follows:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | mysql> SHOW DATABASES;    +--------------------+  | Database           |  +--------------------+  | classicmodels      |  | information\_schema |  | mysql              |  | performance\_schema |  | sys                |  +--------------------+  5 rows in set (0.00 sec) |

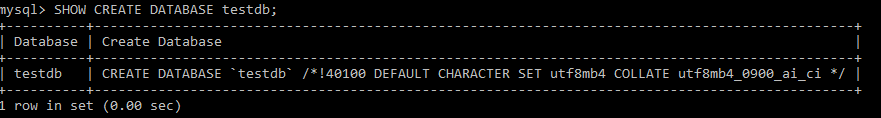
MySQL returns five existing databases in the current server.

Then, issue the CREATE DATABASE command with the database e.g., testdb and press Enter:

|  |  |
| --- | --- |
| 1  2 | mysql> CREATE DATABASE testdb;  Query OK, 1 row affected (0.12 sec) |

After that, if you want to review the created database, you can use the SHOW CREATE DATABASE command:

|  |  |
| --- | --- |
| 1 | mysql> SHOW CREATE DATABASE testdb; |



MySQL returns the database name and the character set and collation of the database.

Finally, to access the newly created database, you use the USE database command as follows:

|  |  |
| --- | --- |
| 1  2 | mysql> USE testdb;  Database changed |

Now, you can start [creating tables](https://www.mysqltutorial.org/mysql-create-table/) and other databases objects within the  testdb database.

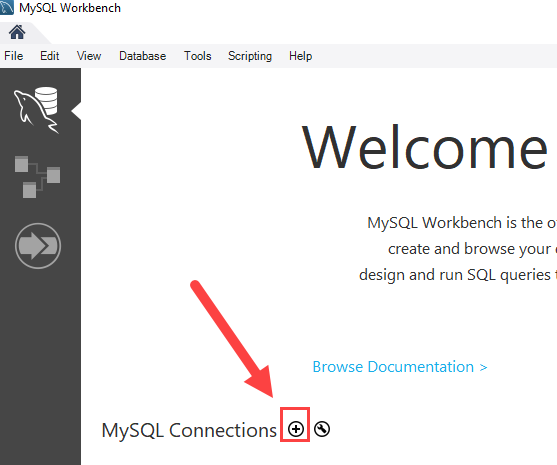
To quit the **mysql** program, type exit command:

|  |  |
| --- | --- |
| 1  2 | mysql> exit  Bye |

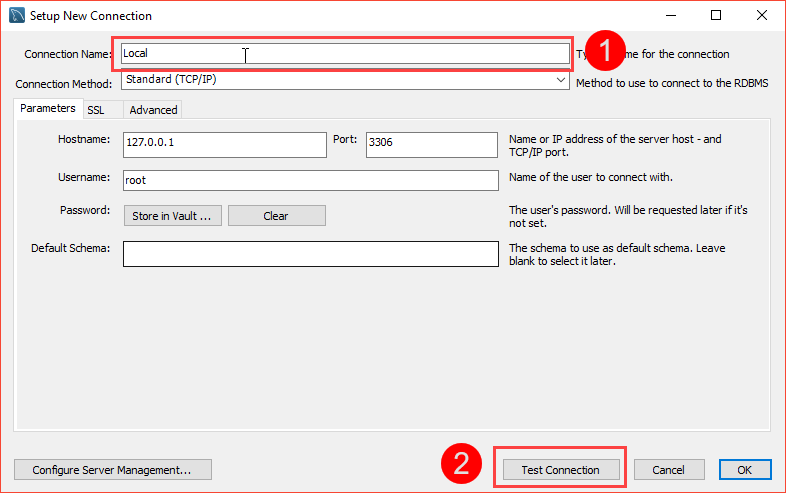
## Creating a new database using MySQL Workbench

To create a new database using the MySQL Workbench, you follow these steps:

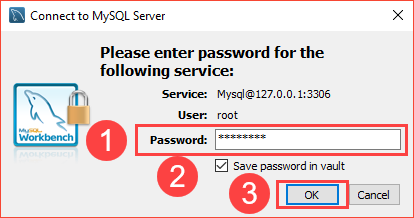
First, launch the MySQL Workbench and click the **setup new connection** button as shown in the following screenshot:



Second, type the name for the connection and click the **Test Connection** button.

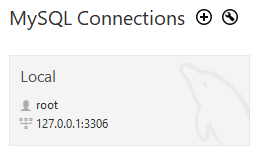


MySQL Workbench displays a dialog asking for the password of the root user:

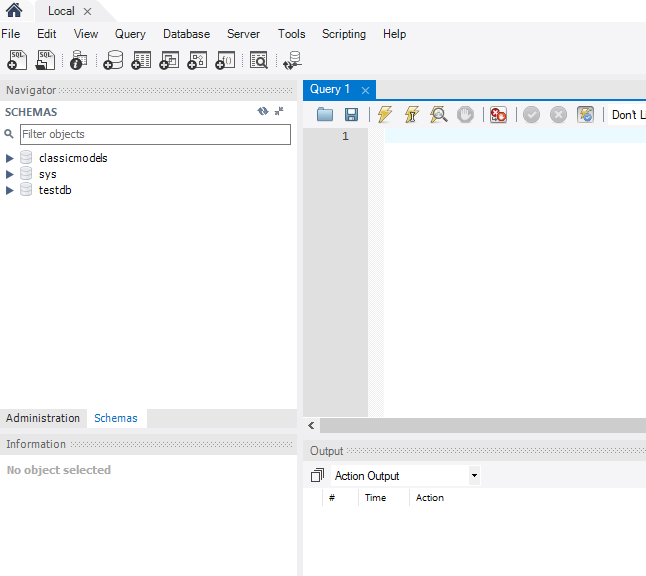


You need to (1) type the password for the root user, (2) check the **Save password in vault**, and (3) click **OK** button.

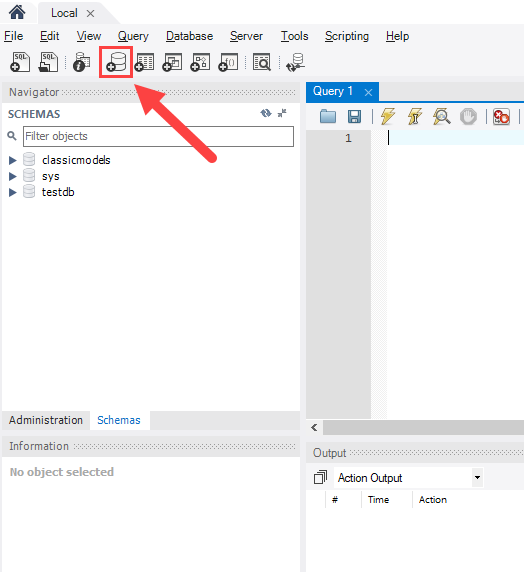
Third, double-click the connection name **Local** to connect to the MySQL Server.



MySQL Workbench opens the following window which consists of four parts: Navigator, Query, Information, and Output.

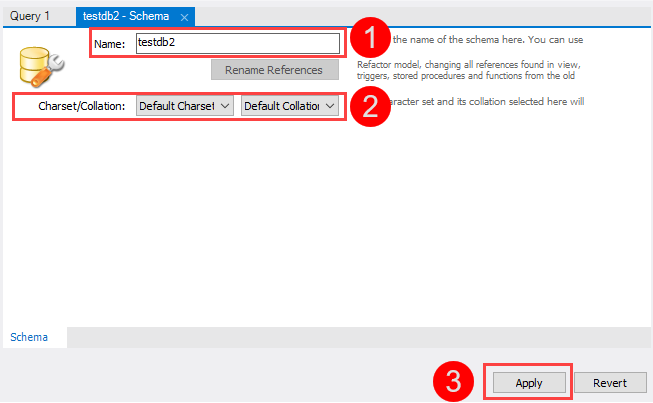


Fourth, click the **create a new schema in the connected server** button from the toolbar:

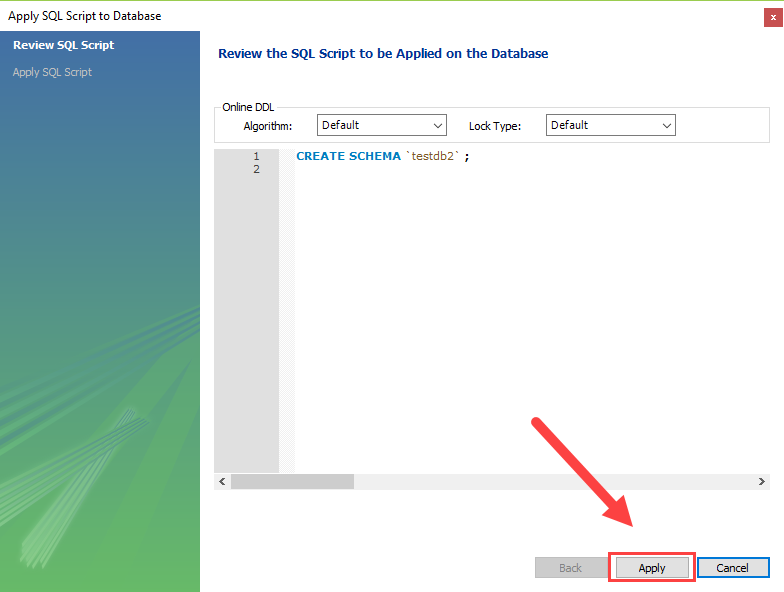


In MySQL, the schema is the synonym for the database. Creating a new schema also means creating a new database.

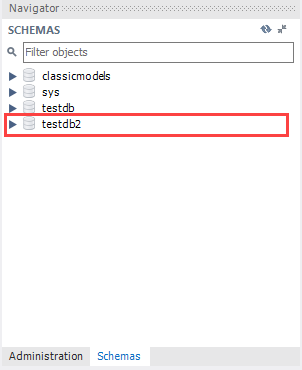
Fifth, the following window is open. You need to (1) enter the schema name, (2) change the character set and collation if necessary, and click the **Apply** button:



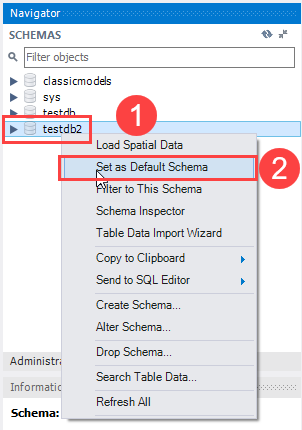
Sixth, MySQL Workbench opens the following window that displays the SQL script which will be executed. Note that the CREATE SCHEMA statement command has the same effect as the CREATE DATABASE statement.



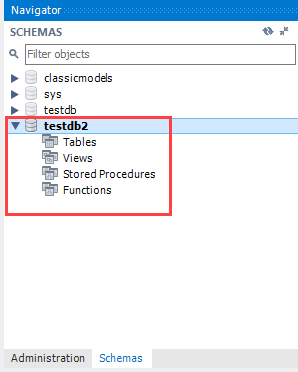
If everything is fine, you will see the new database created and showed in the **schemas** tab of the **Navigator** section.



Seventh, to select the testdb2 database, (1) right click the database name and (2) choose **Set as Default Schema** menu item:



The testdb2 node is open as shown in the following screenshot.



Now, you can work with testdb2 from the MySQL Workbench.

# **MySQL DROP DATABASE**

The DROP DATABASE statement drops all tables in the database and deletes the database permanently. Therefore, you should be very careful when using this statement.

The following shows the syntax of the DROP DATABASE statement:

|  |  |
| --- | --- |
| 1 | DROP DATABASE [IF EXISTS] database\_name; |

In MySQL, the schema is the synonym for the database, therefore, you can use them interchangeably:

|  |  |
| --- | --- |
| 1 | DROP SCHEMA [IF EXISTS] database\_name; |

# **MySQL CREATE TABLE**

## MySQL CREATE TABLE syntax

The CREATE TABLE statement allows you to create a new table in a database.

The following illustrates the basic syntax of the CREATE TABLE  statement:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1  2  3  4  5  6 | CREATE TABLE [IF NOT EXISTS] table\_name(     column\_1\_definition,     column\_2\_definition,     ...,     table\_constraints  ) **MySQL CREATE TABLE simple example** The following statement creates a new table named tasks:   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10 | CREATE TABLE IF NOT EXISTS tasks (      task\_id INT AUTO\_INCREMENT PRIMARY KEY,      title VARCHAR(255) NOT NULL,      start\_date DATE,      due\_date DATE,      status TINYINT NOT NULL,      priority TINYINT NOT NULL,      description TEXT,      created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  ) |   Once you execute the CREATE TABLE statement to create the tasks table, you can view its structure by using the DESCRIBE statement:   |  |  | | --- | --- | | 1 | DESCRIBE tasks; |   MySQL CREATE TABLE - DESCRIBE table |

To change one or more tables, we use the RENAME TABLE statement as follows:

|  |  |
| --- | --- |
| 1 | RENAME TABLE old\_table\_name TO new\_table\_name; |

## Renaming multiple tables

We can also use the RENAME TABLE statement to rename multiple tables at a time. See the following statement:

|  |  |
| --- | --- |
| 1  2 | RENAME TABLE old\_table\_name\_1 TO new\_table\_name\_2,               old\_table\_name\_2 TO new\_table\_name\_2,... |

## Renaming tables using ALTER TABLE statement

We can rename a table using the ALTER TABLE statement as follows:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE old\_table\_name  RENAME TO new\_table\_name; |

MySQL constraints

* [NOT NULL constraint](https://www.mysqltutorial.org/mysql-not-null-constraint/) – introduce you to the NOT NULL constraint and show you how to declare a  NOT NULL column or add a NOT NULL constraint to an existing column.
* [Primary key constraint](https://www.mysqltutorial.org/mysql-primary-key/) – guide you on how to use the primary key constraint to create the primary key for a table.
* [Foreign key constraint](https://www.mysqltutorial.org/mysql-foreign-key/)  – introduce you to the foreign key and show you step by step how to create and drop foreign keys.
* [Disable foreign key checks](https://www.mysqltutorial.org/mysql-disable-foreign-key-checks/) – learn how to disable foreign key checks.
* [UNIQUE constraint](https://www.mysqltutorial.org/mysql-unique-constraint/) – show you how to use UNIQUE constraint to enforce the uniqueness of values in a column or a group of columns in a table.
* [CHECK constraint](https://www.mysqltutorial.org/mysql-check-constraint/) – learn how to create CHECK constraints to ensure data integrity.
* [CHECK constraint emulation](https://www.mysqltutorial.org/mysql-check-constraint-emulation/)  – if you use MySQL 8.0.15 or earlier version, you can emulate CHECKconstraints using views or triggers.

The NOT NULL constraint is a column constraint that ensures values stored in a column are not [NULL](https://www.mysqltutorial.org/mysql-null/).

The syntax of defining a NOT NULL constraint is as follows:

|  |  |
| --- | --- |
| 1 | column\_name data\_type NOT NULL; |

## MySQL NULL values

In MySQL, a NULL value means unknown. A NULL value is different from zero (0) or an empty string ''.

A NULL value is not equal to anything, even itself. If you compare a NULL value with another NULL value or any other value, the result is NULL because the value of each NULL value is unknown.

Generally, you use the NULL value to indicate that the data is missing, unknown, or not applicable. For example, the phone number of a potential customer may be NULL and can be added later.

MySQL primary key

A primary key is a column or a set of columns that uniquely identifies each row in the table.  The primary key follows these rules:

* A primary key must contain unique values. If the primary key consists of multiple columns, the combination of values in these columns must be unique.
* A primary key column cannot have [NULL](https://www.mysqltutorial.org/mysql-null/) values. Any attempt to [insert](https://www.mysqltutorial.org/mysql-insert-statement.aspx) or [update](https://www.mysqltutorial.org/mysql-update-data.aspx) NULL to primary key columns will result in an error. Note that MySQL implicitly adds a NOT NULL constraint to primary key columns.
* A table can have one an only one primary key.
* If the primary key has one column, you can use the PRIMARY KEY constraint as a column constraint:

|  |  |
| --- | --- |
| 1  2  3  4 | CREATE TABLE table\_name(      primary\_key\_column datatype PRIMARY KEY,      ...  ); |

When the primary key has more than one column, you must use the PRIMARY KEY constraint as a table constraint.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CREATE TABLE table\_name(      primary\_key\_column1 datatype,      primary\_key\_column2 datatype,      ...,      PRIMARY KEY(column\_list)  ); |

The following example [creates a table](https://www.mysqltutorial.org/mysql-create-table/) named users whose primary key is the user\_id column:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CREATE TABLE users(     user\_id INT AUTO\_INCREMENT PRIMARY KEY,     username VARCHAR(40),     password VARCHAR(255),     email VARCHAR(255)  ); |

### **Define PRIMARY KEY constraints using ALTER TABLE**

If a table, for some reasons, does not have a primary key, you can use the [ALTER TABLE](https://www.mysqltutorial.org/mysql-alter-table.aspx)statement to add a primary key to the table as follows:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE table\_name  ADD PRIMARY KEY(column\_list); |

[Home](https://www.mysqltutorial.org/) / [Basic MySQL Tutorial](https://www.mysqltutorial.org/basic-mysql-tutorial.aspx) / MySQL UNIQUE Constraint

# **MySQL UNIQUE Constraint**

**Summary**: in this tutorial, you will learn about MySQL UNIQUE constraint and how to use UNIQUEconstraint to enforce the uniqueness of values in a column or a group of columns in a table.

## Introduction to MySQL UNIQUE constraint

Sometimes, you want to ensure values in a column or a group of columns are unique. For example, email addresses of users in the users table, or phone numbers of customers in the customers table should be unique. To enforce this rule, you use a UNIQUE constraint.

A UNIQUE constraint is an integrity constraint that ensures values in a column or group of columns to be unique.  A UNIQUE constraint can be either a column constraint or a table constraint.

To define a UNIQUE constraint for a column when you [create a table](https://www.mysqltutorial.org/mysql-create-table/), you use this syntax:

|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE table\_name(      ...,      column\_name data\_type UNIQUE,      ...  ); |

## Add new unique constraint

The following ALTER TABLE ADD CONSTRAINT adds a unique constraint to a column of an existing table:

|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE table\_name  ADD CONSTRAINT constraint\_name  UNIQUE (column\_list); |

### **MySQL CHECK constraint – column constraint example**

This statement creates a new parts table:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CREATE TABLE parts (      part\_no VARCHAR(18) PRIMARY KEY,      description VARCHAR(40),      cost DECIMAL(10,2 ) NOT NULL CHECK (cost >= 0),      price DECIMAL(10,2) NOT NULL CHECK (price >= 0)  ); |

Here is the basic syntax of the DROP TABLE statement:

|  |  |
| --- | --- |
| 1  2 | DROP TABLE [IF EXISTS] table\_name [, table\_name] ... |

Logically, the TRUNCATE TABLE statement is like a [DELETE](https://www.mysqltutorial.org/mysql-delete-statement.aspx) statement without a [WHERE](https://www.mysqltutorial.org/mysql-where/) clause that deletes all rows from a table, or a sequence of [DROP TABLE](https://www.mysqltutorial.org/mysql-drop-table) and [CREATE TABLE](https://www.mysqltutorial.org/mysql-create-table/) statements.

However, the TRUNCATE TABLE statement is more efficient than the DELETE statement because it drops and recreates the table instead of deleting rows one by one.

Here is the basic syntax of theTRUNCATE TABLE statement:

|  |  |
| --- | --- |
| 1 | TRUNCATE [TABLE] table\_name; |

# **MySQL ALTER TABLE**

### **Add a column to a table**

To [add a column](https://www.mysqltutorial.org/mysql-add-column/) to a table, you use the ALTER TABLE ADD syntax:

|  |  |
| --- | --- |
| 1  2  3  4 | ALTER TABLE table\_name  ADD      new\_column\_name column\_definition      [FIRST | AFTER column\_name] |

The following example uses the ALTER TABLE ADD statement to add a column at the end of the vehiclestable:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE vehicles  ADD model VARCHAR(100) NOT NULL; |

### **Add multiple columns to a table**

To [add multiple columns](https://www.mysqltutorial.org/mysql-add-column/) to a table, you use the following form of the ALTER TALE ADD statement:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | ALTER TABLE table\_name      ADD new\_column\_name column\_definition      [FIRST | AFTER column\_name],      ADD new\_column\_name column\_definition      [FIRST | AFTER column\_name],      ...; |

For example, this statement adds two columns color and note to the vehicles table:

|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE vehicles  ADD color VARCHAR(50),  ADD note VARCHAR(255); |

### **Modify a column**

Here is the basic syntax for modifying a column in a table:

|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE table\_name  MODIFY column\_name column\_definition  [ FIRST | AFTER column\_name]; |

Then, modify the note column:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE vehicles  MODIFY note VARCHAR(100) NOT NULL; |

Second, use the ALTER TABLE MODIFY statement to modify multiple columns:

|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE vehicles  MODIFY year SMALLINT NOT NULL,  MODIFY color VARCHAR(20) NULL AFTER make; |

To [drop a column](https://www.mysqltutorial.org/mysql-drop-column/) in a table, you use the ALTER TABLE DROP COLUMN statement:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE table\_name  DROP COLUMN column\_name; |

## MySQL ALTER TABLE – Rename table

To [rename a table](https://www.mysqltutorial.org/mysql-rename-table/), you use the ALTER TABLE RENAME TO statement:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE table\_name  RENAME TO new\_table\_name; |

This example renames the vehicles table to cars:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE vehicles  RENAME TO cars; |

ALTER TABLE vendors

ADD COLUMN phone VARCHAR(15) AFTER name;

Fourth, add two more columns email and hourly\_rate to the vendors table at the same time.

|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE vendors  ADD COLUMN email VARCHAR(100) NOT NULL,  ADD COLUMN hourly\_rate decimal(10,2) NOT NULL; |

# **MySQL Insert**

The INSERT statement allows you to insert one or more rows into a table. The following illustrates the syntax of the INSERT statement:

|  |  |
| --- | --- |
| 1  2 | INSERT INTO table(c1,c2,...)  VALUES (v1,v2,...); |

To [insert multiple rows](https://www.mysqltutorial.org/mysql-insert-multiple-rows/) into a table using a single INSERT statement, you use the following syntax:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | INSERT INTO table(c1,c2,...)  VALUES     (v11,v12,...),     (v21,v22,...),      ...     (vnn,vn2,...); |

# **MySQL SELECT**

SELECT lastName

FROM employees;

SELECT

    lastname,

    firstname,

    jobtitle

FROM

    employees;

SELECT \*

FROM employees;

## Introduction to MySQL ORDER BY clause

When you use the [SELECT](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement to query data from a table, the result set is not sorted. It means that the rows in the result set can be in any order.

To sort the result set, you add the ORDER BY clause to the SELECT statement. The following illustrates the syntax of the ORDER BY  clause:

SELECT

   select\_list

FROM

   table\_name

ORDER BY

   column1 [ASC|DESC],

   column2 [ASC|DESC],

   ...;

By default, the ORDER BY clause uses ASC if you don’t explicitly specify any option.

Therefore, the following clauses are equivalent:

|  |  |
| --- | --- |
| 1 | ORDER BY column1 ASC; |

and

|  |  |
| --- | --- |
| 1 | ORDER BY column1; |

If you want to sort the result set by multiple columns, you specify a comma-separated list of columns in the ORDER BY clause:

|  |  |
| --- | --- |
| 1  2  3 | ORDER BY     column1,     column2; |

It is possible to sort the result by a column in ascending order, and then by another column in descending order:

|  |  |
| --- | --- |
| 1  2  3 | ORDER BY      column1 ASC,      column2 DESC; |

In this case, the ORDER BY clause:

* First, sort the result set by the values in the column1 in ascending order.
* Then, sort the *sorted* result set by the values in the column2  in descending order. Note that the order of values in the column1 will not change in this step, only the order of values in the column2changes.

## Introduction to MySQL WHERE clause

The WHERE clause allows you to specify a search condition for the rows returned by a query. The following shows the syntax of the WHERE clause:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT      select\_list  FROM      table\_name  WHERE      search\_condition; |

The search\_condition is a combination of one or more predicates using the logical operator [AND](https://www.mysqltutorial.org/mysql-and/), [OR](https://www.mysqltutorial.org/mysql-or/)and NOT.

In MySQL, a predicate is a Boolean expression that evaluates to TRUE, FALSE, or UNKNOWN.

Any row from the table\_name that causes the search\_condition to evaluate to TRUE will be included in the final result set.

Besides the SELECT statement, you can use the WHERE clause in the [UPDATE](https://www.mysqltutorial.org/mysql-update-data.aspx) or [DELETE](https://www.mysqltutorial.org/mysql-delete-statement.aspx) statement to specify which rows to update or delete.

### **1) Using MySQL WHERE clause with equal operator example**

The following query uses the WHERE clause to find all employees whose job titles are Sales Rep:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      lastname,      firstname,      jobtitle  FROM      employees  WHERE      jobtitle = 'Sales Rep'; |

In this example, the SELECT statement examines all rows of the employees table and selects only row whose value in the jobTitle column is Sales Rep.

### **2) Using MySQL WHERE clause with AND operator**

The following example uses the WHERE clause to find employees whose job titles are Sales Rep and office codes are 1:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | SELECT      lastname,      firstname,      jobtitle,      officeCode  FROM      employees  WHERE      jobtitle = 'Sales Rep' AND      officeCode = 1; |

In this example, the expression in the WHERE clause uses the [AND](https://www.mysqltutorial.org/mysql-and/) operator to combine two conditions:

|  |  |
| --- | --- |
| 1 | jobtitle = 'Sales Rep' AND officeCode = 1; |

The AND operator evaluates to TRUE only if both expressions evaluate to TRUE. Therefore, the query returns rows whose values in the jobTitle column is Sales Rep and officeCode is 1.

### **3) Using MySQL WHERE clause with OR operator**

This query finds employees whose job title is Sales Rep or employees who locate the office with office code 1:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | SELECT      lastName,      firstName,      jobTitle,      officeCode  FROM      employees  WHERE      jobtitle = 'Sales Rep' OR      officeCode = 1  ORDER BY      officeCode ,      jobTitle; |

he [OR](https://www.mysqltutorial.org/mysql-or/) operator evaluates to TRUE only if one of the expression evaluates to TRUE:

|  |  |
| --- | --- |
| 1 | jobtitle = 'Sales Rep' OR officeCode = 1 |

Therefore, the query returns any employee who has the job title Sales Rep or office code 1.

### **4) Using MySQL WHERE with BETWEEN operator example**

The [BETWEEN](https://www.mysqltutorial.org/mysql-between) operator returns TRUE if a value is in a range of values:

|  |  |
| --- | --- |
| 1 | expression BETWEEN low AND high |

The following query finds employees who locate in offices whose office code is from 1 to 3:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      firstName,      lastName,      officeCode  FROM      employees  WHERE      officeCode BETWEEN 1 AND 3  ORDER BY officeCode; |

### **5) Using MySQL WHERE with the LIKE operator example**

The [LIKE](https://www.mysqltutorial.org/mysql-like/) operator evaluates to TRUE if a value matches a specified pattern. To form a pattern, you use % and \_ wildcards. The % wildcard matches any string of zero or more characters while the \_ wildcard matches any single character.

This query finds employees whose last names end with the string 'son':

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      firstName,      lastName  FROM      employees  WHERE      lastName LIKE '%son'  ORDER BY firstName; **A) Using MySQL LIKE with the percentage (%) wildcard examples** This example uses the LIKE operator to find employees whose first names start with a:   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8 | SELECT      employeeNumber,      lastName,      firstName  FROM      employees  WHERE      firstName LIKE 'a%'; |   This example uses the LIKE operator to find employees whose last names end with on e.g., Patterson, Thompson:   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8 | SELECT      employeeNumber,      lastName,      firstName  FROM      employees  WHERE      lastName LIKE '%on'; |   For example, to find all employees whose last names contain on , you use the following query with the pattern %on%   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8 | SELECT      employeeNumber,      lastName,      firstName  FROM      employees  WHERE      lastname LIKE '%on%'; |   To find employees whose first names start with  T , end with m, and contain any single character between e.g., Tom , Tim, you use the underscore (\_) wildcard to construct the pattern as follows:   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8 | SELECT      employeeNumber,      lastName,      firstName  FROM      employees  WHERE      firstname LIKE 'T\_m'; |   uppose you want to search for employees whose last names don’t start with the character B, you can use the NOT LIKE with a pattern as shown in the following query:   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8 | SELECT      employeeNumber,      lastName,      firstName  FROM      employees  WHERE      lastName NOT LIKE 'B%'; |  MySQL LIKE operator with ESCAPE clause Sometimes the pattern, which you want to match, contains wildcard character e.g., 10%, \_20, etc. In this case, you can use the ESCAPE clause to specify the escape character so that MySQL will interpret the wildcard character as a literal character. If you don’t specify the escape character explicitly, the backslash character \ is the default escape character.  For example, if you want to find products whose product codes contain the string \_20 , you can use the pattern %\\_20% as shown in the following query:   |  |  | | --- | --- | | 1  2  3  4  5  6  7 | SELECT      productCode,      productName  FROM      products  WHERE      productCode LIKE '%\\_20%'; |   Or you can specify a different escape character e.g., $ by using the ESCAPE clause:   |  |  | | --- | --- | | 1  2  3  4  5  6  7 | SELECT      productCode,      productName  FROM      products  WHERE      productCode LIKE '%$\_20%' ESCAPE '$'; | |

### **6) Using MySQL WHERE clause with the IN operator example**

The [IN](https://www.mysqltutorial.org/sql-in.aspx) operator returns TRUE if a value matches any value in a list.

|  |  |
| --- | --- |
| 1 | value IN (value1, value2,...) |

The following example uses the WHERE clause with the IN operator to find employees who locate in the office with office code 1.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | SELECT      firstName,      lastName,      officeCode  FROM      employees  WHERE      officeCode IN (1 , 2, 3)  ORDER BY      officeCode; |

### **7) Using MySQL WHERE  clause with the IS NULL operator**

To check if a value is [NULL](https://www.mysqltutorial.org/mysql-null/) or not, you use the [IS NULL](https://www.mysqltutorial.org/mysql-is-null/) operator, not the equal operator (=). The IS NULL operator returns TRUE if a value is NULL.

|  |  |
| --- | --- |
| 1 | value IS NULL |

In the database world, NULL is a marker that indicates a piece of information is missing or unknown. It is not equivalent to the number 0 or an empty string.

This statement uses the WHERE clause with the IS NULL operator to get the row whose value in the reportsTo column is NULL:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      lastName,      firstName,      reportsTo  FROM      employees  WHERE      reportsTo IS NULL; |

### **8) Using MySQL WHERE clause with comparison operators**

The following table shows the comparison operators that you can use to form the expression in the WHERE clause.

| **Operator** | **Description** |
| --- | --- |
| = | Equal to. You can use it with almost any data types. |
| <> or != | Not equal to |
| < | Less than. You typically use it with numeric and date/time data types. |
| > | Greater than. |
| <= | Less than or equal to |
| >= | Greater than or equal to |

The following query uses the not equal to (<>) operator to find all employees who are not the Sales Rep:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      lastname,      firstname,      jobtitle  FROM      employees  WHERE      jobtitle <> 'Sales Rep'; |

## ntroduction to MySQL DISTINCT clause

When querying data from a table, you may get duplicate rows. In order to remove these duplicate rows, you use the DISTINCT clause in the [SELECT](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement.

Here is the syntax of the DISTINCT clause:

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT DISTINCT      select\_list  FROM      table\_name; |

## MySQL DISTINCT and NULL values

If a column has [NULL](https://www.mysqltutorial.org/mysql-null/) values and you use the DISTINCT clause for that column, MySQL keeps only one NULL value because DISTINCT treats all NULL values as the same value.

## MySQL DISTINCT with multiple columns

You can use the DISTINCT clause with more than one column. In this case, MySQL uses the combination of values in these columns to determine the uniqueness of the row in the result set.

For example, to get a unique combination of city and state from the customers table, you use the following query:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT DISTINCT      state, city  FROM      customers  WHERE      state IS NOT NULL  ORDER BY      state,      city; |

## DISTINCT clause vs. GROUP BY clause

If you use the [GROUP BY](https://www.mysqltutorial.org/mysql-group-by.aspx) clause in the SELECT statement without using [aggregate functions](https://www.mysqltutorial.org/mysql-aggregate-functions.aspx), the GROUP BYclause behaves like the DISTINCT clause.

The following statement uses the GROUP BY clause to select the unique states of customers from the customers table.

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT      state  FROM      customers  GROUP BY state; |

## MySQL DISTINCT with LIMIT clause

In case you use the DISTINCT clause with the [LIMIT](https://www.mysqltutorial.org/mysql-limit.aspx) clause, MySQL immediately stops searching when it finds the number of unique rows specified in the LIMIT clause.

The following query selects the first five non-null unique states in the customers table.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT DISTINCT      state  FROM      customers  WHERE      state IS NOT NULL  LIMIT 5; |

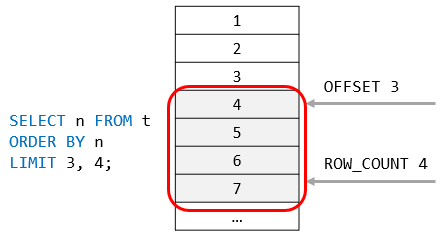
## Introduction to MySQL LIMIT clause

The LIMIT clause is used in the [SELECT](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement to constrain the number of rows to return. The LIMITclause accepts one or two arguments. The values of both arguments must be zero or positive [integers](https://www.mysqltutorial.org/mysql-int/).

The following illustrates the LIMIT clause syntax with two arguments:

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT      select\_list  FROM      table\_name  LIMIT [offset,] row\_count; |

The following picture illustrates the LIMIT clause:



[Home](https://www.mysqltutorial.org/) / [Basic MySQL Tutorial](https://www.mysqltutorial.org/basic-mysql-tutorial.aspx) / MySQL LIMIT

# **MySQL LIMIT**

**Summary**: in this tutorial, you will learn how to use MySQL LIMIT clause to constrain the number of rows returned by a query.

## Introduction to MySQL LIMIT clause

The LIMIT clause is used in the [SELECT](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement to constrain the number of rows to return. The LIMITclause accepts one or two arguments. The values of both arguments must be zero or positive [integers](https://www.mysqltutorial.org/mysql-int/).

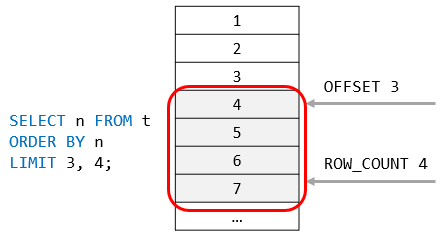
The following illustrates the LIMIT clause syntax with two arguments:

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT      select\_list  FROM      table\_name  LIMIT [offset,] row\_count; |

In this syntax:

* The offset specifies the offset of the first row to return. The offset of the first row is 0, not 1.
* The row\_count specifies the maximum number of rows to return.

The following picture illustrates the LIMIT clause:



When you use the LIMIT clause with one argument, MySQL will use this argument to determine the maximum number of rows to return from the first row of the result set.

Therefore, these two clauses are equivalent:

|  |  |
| --- | --- |
| 1 | LIMIT row\_count; |

And

|  |  |
| --- | --- |
| 1 | LIMIT 0 , row\_count; |

In addition to the above syntax, MySQL provides the following alternative LIMIT clause for compatibility with [PostgreSQL](http://www.postgresqltutorial.com/postgresql-limit/).

|  |  |
| --- | --- |
| 1 | LIMIT row\_count OFFSET offset |

### **LIMIT and ORDER BY clauses**

The SELECT statement without an [ORDER BY](https://www.mysqltutorial.org/mysql-order-by/) clause returns rows in an unspecified order. It means that rows can be in any order. When you apply the LIMIT clause to this unordered result set,  you will not know which rows the query will return.

For example, you may want to get fifth through tenth rows, but fifth through tenth in what order? The order of rows is unknown unless you specify the ORDER BY clause.

Therefore, it is a good practice to always use the LIMIT clause with the ORDER BY clause to constraint the result rows in unique order.

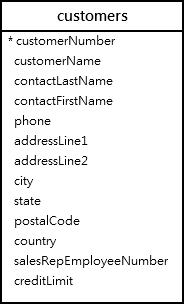
|  |  |
| --- | --- |
| 1  2  3  4 | SELECT select\_list  FROM table\_name  ORDER BY order\_expression  LIMIT offset, row\_count; |

The following picture illustrates the evaluation order of the LIMIT clause in the SELECT statement:



## MySQL LIMIT examples

We’ll use the customers table from the [sample database](https://www.mysqltutorial.org/mysql-sample-database.aspx) for demonstration.



### **1) Using MySQL LIMIT to get the highest or lowest rows**

This statement uses the LIMIT clause to get the top five customers who have the highest credit:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      customerNumber,      customerName,      creditLimit  FROM      customers  ORDER BY creditLimit DESC  LIMIT 5; |

Similarly, this example uses the LIMIT clause to find 5 customers who have the lowest credits:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      customerNumber,      customerName,      creditLimit  FROM      customers  ORDER BY creditLimit  LIMIT 5; |