**MySQL Tutorial**

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**.

MySQL database that provides for **how to manage database** and **to manipulate data** with the help of various **SQL queries**.

**These queries are**: insert records, update records, delete records, select records, create tables, drop tables, etc.

**What is Database?**

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.

**What is MySQL?**

MySQL is a multithreaded, multi-user SQL database management system which has more than 11 million installations. It is the world's second most popular and widely-used open source database. It is interesting how MySQL name was given to this query language. The term My is coined by the name of the daughter of co-founder Michael Widenius's daughter, and SQL is the short form of Structured Query Language. Using MySQL is free of cost for the developer, but enterprises have to pay a license fee to Oracle.

**Database :**

A database contains a **set of tables**.

**Table:**

A *table* is made up of ***rows***and ***columns****.*

Each table has a name, which is unique within the database.

**Columns:**

Each column has a **name** and a **data type**.

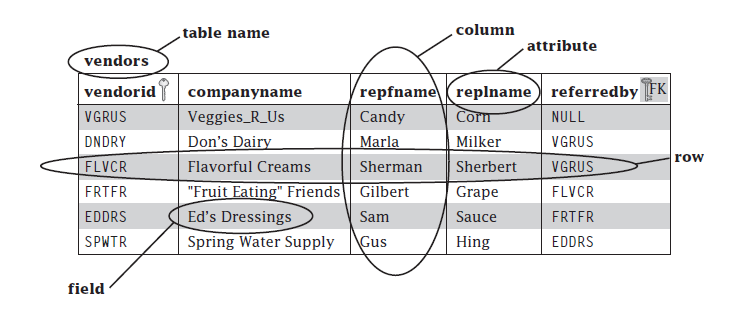
The name of a column need only be unique within a table so other tables in the same database can have columns of the same name.

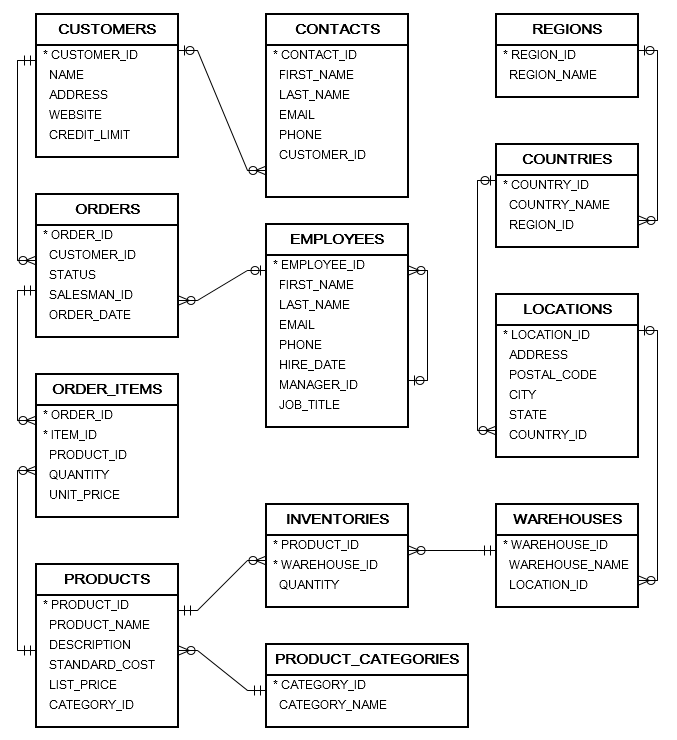
**Row:**

Each row constitutes one ***record***in the table.

A table may contain zero or more rows.

A row is subdivided into ***fields****,* one per column.





| **Table Names** | **Description** | **Records** |
| --- | --- | --- |
| CONTACTS | store contact person information of customers | 319 records |
| COUNTRIES | store country information | 25 records |
| CUSTOMERS | store customer master | 319 records |
| EMPLOYEES | store employee master | 107 records |
| INVENTORIES | store inventory information of products | 1112 records |
| LOCATIONS | store locations of warehouses | 23 records |
| ORDERS | store order header information | 105 records |
| ORDER\_ITEMS | store order line items | 665 records |
| PRODUCT\_CATEGORIES | store product categories | 5 records |
| PRODUCTS | store product information | 288 records |
| REGIONS | store regions where the company operates | 4 records |
| WAREHOUSES | store warehouse information | 9 records |

**MySQL Data Types**

MySQL supports a lot number of [SQL](https://www.javatpoint.com/sql-tutorial) 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,**
* [**JSON**](https://www.javatpoint.com/json-tutorial) **data types.**

**Numeric Data Type**

|  |  |
| --- | --- |
| **Data Type Syntax** | **Description** |
| TINYINT | It is a very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. We can specify a width of up to 4 digits. It takes 1 byte for storage. |
| SMALLINT | It is a small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. We can specify a width of up to 5 digits. It requires 2 bytes for storage. |
| MEDIUMINT | It is a medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. We can specify a width of up to 9 digits. It requires 3 bytes for storage. |
| 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. |
| 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. |

**Date and Time Data Type:**

|  |  |  |
| --- | --- | --- |
| **Data Type Syntax** | **Maximum Size** | **Explanation** |
| YEAR[(2|4)] | Year value as 2 digits or 4 digits. | The default is 4 digits. It takes 1 byte for storage. |
| DATE | Values range from '1000-01-01' to '9999-12-31'. | Displayed as 'yyyy-mm-dd'. It takes 3 bytes for storage. |
| TIME | Values range from '-838:59:59' to '838:59:59'. | Displayed as 'HH:MM:SS'. It takes 3 bytes plus fractional seconds for storage. |
| DATETIME | Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. | Displayed as 'yyyy-mm-dd hh:mm:ss'. It takes 5 bytes plus fractional seconds for storage. |
| TIMESTAMP(m) | Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' TC. | Displayed as 'YYYY-MM-DD HH:MM:SS'. It takes 4 bytes plus fractional seconds for storage. |

**String Data Types:**

|  |  |  |
| --- | --- | --- |
| **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. |
| BINARY(size) | It can have a maximum size of 255 characters. | Here size is the number of binary characters to store. Fixed-length strings. Space padded on the right to equal size characters. (introduced in MySQL 4.1.2) |
| VARBINARY(size) | It can have a maximum size of 255 characters. | Here size is the number of characters to store. Variable-length string. (introduced in MySQL 4.1.2) |
| ENUM | It takes 1 or 2 bytes that depend on the number of enumeration values. An ENUM can have a maximum of 65,535 values. | It is short for enumeration, which means that each column may have one of the specified possible values. It uses numeric indexes (1, 2, 3…) to represent string values. |
| SET | It takes 1, 2, 3, 4, or 8 bytes that depends on the number of set members. It can store a maximum of 64 members. | It can hold zero or more, or any number of string values. They must be chosen from a predefined list of values specified during table creation. |

**MySQL Variables**

Variables are used for storing data or information during the execution of a 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**

[MySQL](https://www.javatpoint.com/mysql-tutorial) 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**

By using the **SET** statement

1. SET @var\_name = value;

By using the **SELECT** statement

1. SELECT @var\_name := value;

**Local Variable**

It is a type of variable that is not prefixed by **@ symbol**. The local variable is a **strongly typed variable.** The scope of the **local variable is in a stored program block** in which it is declared. MySQL uses the **DECLARE** keyword to specify the local variable. The DECLARE statement also combines a **DEFAULT** clause to provide a default value to a variable. If you do not provide the DEFAULT clause, it will give the initial value **NULL**. It is mainly used in the stored procedure program.

**Syntax**

We can use the DECLARE statement with the following syntax:

1. DECLARE variable\_name datatype(size) [DEFAULT default\_value];

DECLARE total\_price Oct(8,2) DEFAULT 0.0;

**System Variable**

System variables are a special class to all program units, which contains **predefined** variables. MySQL contains various system variables that configure its operation, and each system variable contains a default value.

**MySQL 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.

**Syntax:**

1. CREATE DATABASE [IF NOT EXISTS] database\_name
2. [CHARACTER SET charset\_name]
3. [COLLATE collation\_name];

mysql>

CREATE DATABASE employeesdb;

SHOW CREATE DATABASE employeedb;

SHOW DATABASES;

USE emplyeedb;

**NOTE: All the database names, table names, and table field names are case sensitive. We must have to use proper names while giving any SQL command.**

**MySQL 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:**

1. USE database\_name;

**MySQL DROP Database**

DROP DATABASE [IF EXISTS] database\_name;

**MySQL 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

**Syntax:**

1. CREATE TABLE [IF NOT EXISTS] table\_name(
2. column\_definition1,
3. column\_definition2,
4. ........,
5. table\_constraints
6. );

**Parameter Explanation**

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. |

**NULL**

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.**

**Constraints**

The constraint in MySQL is used to **specify the rule that allows or restricts what values/data will be stored in the table**. They provide a suitable method to ensure data accuracy and integrity inside the table. It also helps to limit the type of data that will be inserted inside the table. If any interruption occurs between the constraint and data action, the action is failed.

**Constraints used in MySQL**

The following are the most common constraints used in the MySQL:

* **NOT NULL**

This constraint specifies that the *column cannot have NULL or empty values*.

* **CHECK**

It *controls the value* in a particular column. It ensures that the inserted value in a column *must be satisfied with the given condition.*

* **DEFAULT**

This constraint is used to *set the default value* for the particular column where we have not specified any value. It means the column must contain a value, including NULL.

* **PRIMARY KEY**

This constraint is used to *identify each record in a table uniquely*. If the column contains primary key constraints, then *it cannot be null or empty*. A table may have duplicate columns, but it can contain only one primary key. It always contains unique value into a column.

* **AUTO\_INCREMENT**

This constraint *automatically generates a unique number* whenever we insert a new record into the table. Generally, we use this constraint for the primary key field in a table.

* **UNIQUE**

This constraint ensures that *all values inserted into the column will be unique*. It means a column *cannot stores duplicate values.*

* **INDEX**

This constraint allows us to *create and retrieve values from the table very quickly and easily.* An index can be created using one or more than one column. It *assigns a ROWID* for each row in that way they were inserted into the table.

* **ENUM**

The ENUM data type in MySQL is a string object. It *allows us to limit the value chosen from a list of permitted values* in the column specification at the time of table creation.

* **FOREIGN KEY**

This constraint is *used to link two tables together.* It is also known as the referencing key. A foreign key column matches the primary key field of another table. It means a *foreign key field in one table refers to the primary key field of another table*.

 CREATE TABLE employee\_table(

id  int  NOT NULL AUTO\_INCREMENT,

name  varchar(45)  NOT NULL,

occupation  varchar(35)  NOT NULL,

PRIMARY KEY (id)

);

DESCRIBE  employee\_table;