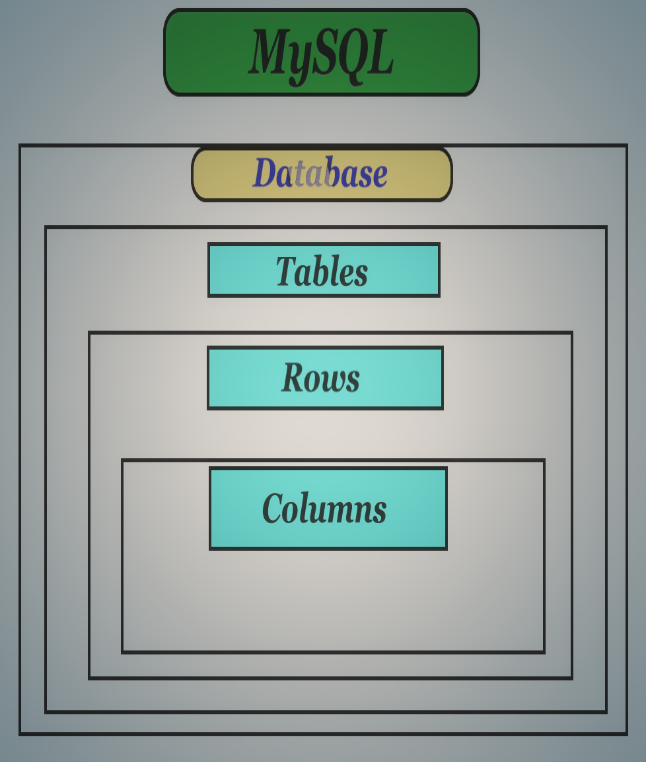
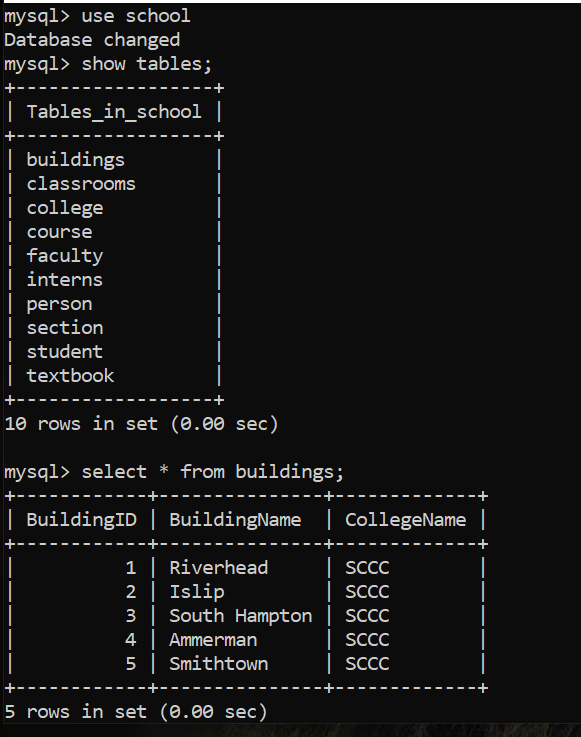
**MySQL**

## **What is MySQL?**

* MySQL is a relational database management system (RDBMS) developed by Oracle that is based on structured query language (SQL). A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or a place to hold the vast (spacious/large) amounts of information in a corporate network.
* So, records stored in the tables in form of rows and columns.
* It was created by MySQL AB which was Swedish company --->After that MySQL AB acquired by the sun microsystem ---> later on oracle acquired. (Right now, MySQL is owned by oracle).
* MySQL used by: Facebook, YouTube, Flicker, etc...
* Written in C and C++.
* Initial release: 23 may 1995.
* Current version: My SQL 8.0.29.

## **MySQL Example: -**

* How a table looks like? how everything works in MySQL.?

## **What is RDBMS?**

The software used to store, manage, query, and retrieve data stored in a relational database is called a **relational database management system (RDBMS).** The RDBMS provides an interface between users and applications and the database, as well as administrative functions for managing data storage, access, and performance.

## **Why it is** **called relational databases?**

* It is collection of data items with pre-defined relationships between them.
* (a relational database refers to database that stores data in structured format, using rows and columns.)
* This makes easy to locate and access specific values within the databases.
* The vales within each table are related to each other.
* It’s used for ability to create meaningful information by joining the tables.

### RDBMS: What is Table?

In Relational database model, a **table** is a collection of data elements organised in terms of rows and columns. A table is also considered as a convenient representation of **relations**. But a table can have duplicate row of data while a true **relation** cannot have duplicate data. Table is the most simplest form of data storage. Below is an example of an employee table.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | **Name** | **Age** | **Salary** |
| **1** | **Adam** | **34** | **13000** |
| **2** | **Alex** | **28** | **15000** |
| **3** | **Stuart** | **20** | **18000** |
| **4** | **Ross** | **42** | **19020** |

### RDBMS: What is a Tuple?

A single entry in a table is called a **Tuple** or **Record** or **Row**. A **tuple** in a table represents a set of related data. For example, the above **Employee** table has 4 tuples/records/rows.

Following is an example of single record or tuple.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Adam | 34 | 13000 |

### RDBMS: What is an Attribute?

A table consists of several records(row), each record can be broken down into several smaller parts of data known as **Attributes**. The above **Employee** table consist of four attributes, **ID**, **Name**, **Age** and **Salary**.

#### **Attribute Domain**

When an attribute is defined in a relation(table), it is defined to hold only a certain type of values, which is known as **Attribute Domain**.

Hence, the attribute **Name** will hold the name of employee for every tuple. If we save employee's address there, it will be violation of the Relational database model.

|  |
| --- |
| **Name** |
| **Adam** |
| **Alex** |
| **Stuart - 9/401, OC Street, Amsterdam** |
| **Ross** |

#### **What is a Relation Schema?**

A relation schema describes the structure of the relation, with the name of the relation (name of table), its attributes and their names and type.

#### **What is a Relation Key?**

A relation key is an attribute which can uniquely identify a particular tuple(row) in a relation(table).

#### **Relational Integrity Constraints**

Every relation in a relational database model should abide by or follow a few constraints to be a valid relation, these constraints are called as **Relational Integrity Constraints**.

The three main Integrity Constraints are:

1. Key Constraints
2. Domain Constraints
3. Referential integrity Constraints

#### **Key Constraints**

We store data in tables, to later access it whenever required. In every table one or more than one attributes together are used to fetch data from tables. The **Key Constraint** specifies that there should be such an attribute(column) in a relation(table), which can be used to fetch data for any tuple(row).

The Key attribute should never be **NULL** or same for two different row of data.

For example, in the **Employee** table we can use the attribute ID to fetch data for each of the employee. No value of ID is null and it is unique for every row, hence it can be our **Key attribute**.

#### **Domain Constraint**

Domain constraints refers to the rules defined for the values that can be stored for a certain attribute.

Like we explained above, we cannot store **Address** of employee in the column for **Name**.

Similarly, a mobile number cannot exceed 10 digits.

#### **Referential Integrity Constraint**

We will study about this in detail later. For now remember this example, if I say **Supriya** is my girlfriend, then a girl with name Supriya should also exist for that relationship to be present.

If a table reference to some data from another table, then that table and that data should be present for referential integrity constraint to hold true.

## **Database vs DBMS**

* **Database**
* A collection of organized, related tables (data stored in rows and columns)
* For example: - school is the database it contains tables like admission details, student details, staff details, examination details…
* **DBMS**
* DBMS -> Database Management System.
* It is Software which is used to create and maintain a database.
* For example: - Examples of DBMS are file systems Microsoft Access, XML file.

## **DBMS vs RDMS**

| **Sr. No.** | **Key** | **DBMS** | **RDBMS** |
| --- | --- | --- | --- |
| 1 | Definition | DBMS stands for Database Management System. | RDBMS stands for Relational Database Management System. |
| 2 | Data Storage | Data is stored as file. | Data is stored as tables. |
| 3 | Data Access | In DBMS, each data elements are to be accessed individually. | In RDBMS, multiple data elements can be accessed at same time. |
| 4 | Relationship | There is no relationship between data in DBMS. | Data is present in multiple tables which can be related to each other. |
| 5 | Normalization | Normalization cannot be achieved. | Normalization can be achieved. |
| 6 | Distributed database | DBMS has no support for distributed databases. | RDBMS supports distributed databases. |
| 7 | Data Quantity | DBMS deals with small quantity of data. | RDBMS deals with large quantity of data. |
| 8 | Data Redundancy | Data Redundancy is common in DBMS. | Data Redundancy can be reduced using key and indexes in RDBMS. |
| 9 | User | DBMS supports single user at a time. | RDBMS supports multiple users at a time. |
| 10 | Security | DBMS provides low security during data manipulation. | RDBMS has multilayer security during data manipulation. |
| 11 | Example | MS Access. | Oracle, SQL Server, MySQL. |

## **Types of DBMS**

There are mainly 4 types of DBMS, which are Hierarchical, Relational, Network, and Object-Oriented DBMS.

**1. Hierarchical Database**

These DBMS employ parent and child relationships to store the data. Hierarchical DBMS is wide.

They store data in a tree-like structure so that it is easy to find and use. Similarly, the configuration of the DBMS is present in the nodes of the tree.

**2. Network DBMS**

Network DBMS supports many-to-many relations which results in complex database structures.

RDM Server is a major example of the network DBMS.

**3. Relational DBMS**

Relational DBMS stores data using the database relationships in the form of tables, also known as relations or tuples.

They do not support many to many relationships and have pre-defined data types that they can support. They are the most popular DBMS type in the industry.

Example: Oracle, MySQL, MS SQL, etc.

**4. Object-Oriented Relational DBMS**

This DBMS supports the storage of miscellaneous data types. They store data in the form of objects.

The object has attributes (i.e., name, id, gender, etc.) and the logic for what needs to be done with the data. Example: PostgreSQL.

Now, that I have told you about DBMS, it’s time that we understand what is SQL?

## **Structured Query Language (SQL)**

SQL is the core of a relational database which is used for accessing and managing the database. By using SQL, you can add, update or delete rows of data, retrieve subsets of information, modify databases and perform many actions. **SQL commands are divided into four subgroups, DML, DDL, DCL, and TCL.**

DDL is short name of **Data Definition Language,** which deals with database schemas and descriptions, of how the data should reside in the database.

* [CREATE](https://www.w3schools.in/mysql/php-mysql-create/) - to create a database and its objects like (table, index, views, store procedure, function, and triggers)
* ALTER - alters the structure of the existing database
* DROP - delete objects from the database
* TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
* COMMENT - add comments to the data dictionary
* RENAME - rename an object

## **DML**

DML is short name of **Data Manipulation Language** which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

* [SELECT](https://www.w3schools.in/mysql/php-mysql-select/) - retrieve data from a database
* [INSERT](https://www.w3schools.in/mysql/php-mysql-insert/) - insert data into a table
* [UPDATE](https://www.w3schools.in/mysql/php-mysql-update/) - updates existing data within a table
* [DELETE](https://www.w3schools.in/mysql/php-mysql-delete/) - Delete all records from a database table
* MERGE - UPSERT operation (insert or update)
* CALL - call a PL/SQL or Java subprogram
* EXPLAIN PLAN - interpretation of the data access path
* LOCK TABLE - concurrency Control

## **DCL**

DCL is short name of **Data Control Language** which includes commands such as GRANT and mostly concerned with rights, permissions and other controls of the database system.

* GRANT - allow users access privileges to the database
* REVOKE - withdraw users access privileges given by using the GRANT command

## **TCL**

TCL is short name of Transaction Control Language which deals with a transaction within a database.

* COMMIT - commits a Transaction
* ROLLBACK - rollback a transaction in case of any error occurs
* SAVEPOINT - to roll back the transaction making points within groups
* SET TRANSACTION - specify characteristics of the transaction

## **What you can do with MySQL?**

There are three major ways to leverage the power of MySQL in central business functions developing online applications, data warehousing, and enterprise-level custom applications.

* Online Applications: Powering Web 2.0. ...
* Data Warehouse: Keeping Information Safe. ...
* Enterprise-level Custom Applications.

## **What are the benefits of MySQL?**

It is a stable, reliable and powerful solution with advanced features like the following:

* Data Security. ...
* On-Demand Scalability. ...
* High Performance. ...
* Round-the-clock Uptime. ...
* Comprehensive Transactional Support. ...
* Complete Workflow Control. ...
* Reduced Total Cost of Ownership. ...
* The Flexibility of Open Source.

## **Is MySQL a server or database?**

* MySQL is a relational database management system.
* To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server.

## **Why we use MySQL?**

* It is a free s/w, offers rich interface and very useful set of functions.
* MySQL server can handle very large database.
* Connectivity Speed and Security make MySQL very suited for accessing the database on a network.
* Its fast comparing to other databases like PostgreSQL, MSQL.
* It is supported well by the user community.

## **Difference Between SQL and MySQL**

| **SQL** | **MySQL** |
| --- | --- |
| SQL is Structured Query Language used to manage the relational databases. | MySQL is a relational database management system used to store, retrieve, modify and administer a database using SQL. We have a lot of database software available in the market. The popular ones include MySQL, SQL Server, Oracle, Informix, etc. |
| It’s a query language. | It’s database software. It uses SQL as a language to query the database. |
| Since this is a language, it does not get updates. SQL commands always remain the same. | Since it’s a software, it gets frequent updates. |

## **Who is using MySQL?**

Many companies reportedly use MySQL in their tech stacks, including Uber, Airbnb, and Shopify….

* Uber.
* Airbnb.
* Shopify.
* Pinterest.
* Netflix.
* Amazon.
* Udemy.
* Twitter.

## **Is MySQL a backend?**

* Back-End Languages: PHP, JavaScript, MySQL, and Python are **back-end languages** that manage data and input from website users.

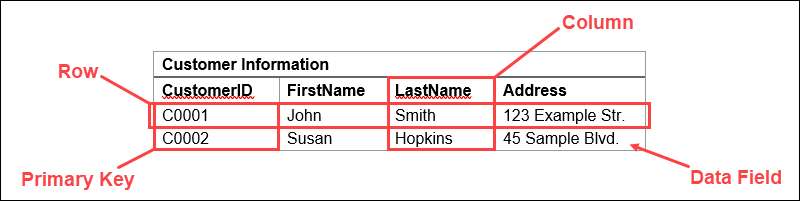
## **Is DB and schema the same?**

* The difference between **DATABASE** and **SCHEMA** terminology is the most common part of an interview question. The main difference between them is that **the database is a collection of interrelated data, whereas schema is the database's structural view**.

## **How Is Data in a Relational Database System Organized?**

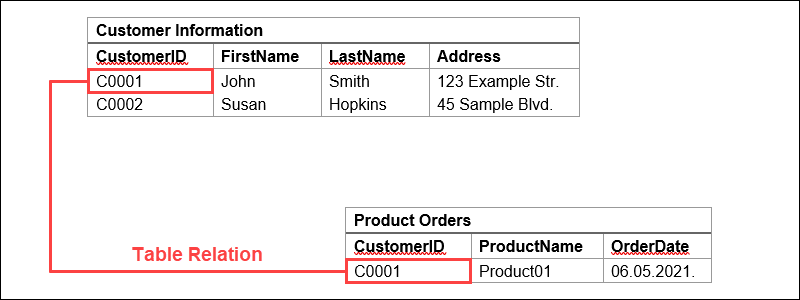
Relational database systems use a model that organizes data into **tables** of **rows** (also called***records*** or **tuples**) and **columns** (also called **attributes** or **fields**). Generally, columns represent categories of data, while rows represent individual instances.

Let's use a digital storefront as an example. Our database might have a table containing customer information, with columns representing customer names or addresses, while each row contains data for one individual customer.



These tables can be linked or related using **keys**. Each row in a table is identified using a unique key, called a **primary key.** This primary key can be added to another table, becoming a **foreign key.** The primary/foreign key relationship forms the basis of the way relational databases work.

Returning to our example, if we have a table representing product orders, one of the columns might contain customer information. Here, we can import a primary key that links to a row with the information for a specific customer.



This way, we can reference the data or duplicate data from the customer information table. It also means that these two tables are now related.

## **What is MySQL Workbench?**

* It is a user interface for MySQL i.e., a very good graphical tool for working with MySQL server and databases. (That easily creating database records and saving the records...)
* Supports MySQL server versions 5.6 and higher.

## **MySQL Features?**

* Based on SQL
* Open source (freely available)
* Friendly & compatible with PHP. Connect easily & manipulate databases. (Create a website and web database share it others on work on it.)
* Based on client-server architecture
* Very secure and highly flexible.
* Scalable: supports large databases
  + 50 million rows or more in table.
  + Default file size limit for a table is 4GB (but you can easily increase)
* It is compatible with major operating systems and languages like Windows, mac, and Linux.

## **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.



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.

A client can use any MySQL [GUI](https://www.javatpoint.com/gui-full-form). But, it is making sure that your GUI should be lighter and user-friendly to make your data management activities faster and easier. Some of the most widely used MySQL GUIs are MySQL Workbench, SequelPro, DBVisualizer, and the Navicat DB Admin Tool. Some GUIs are commercial, while some are free with limited functionality, and some are only compatible with MacOS. Thus, you can choose the GUI according to your needs.

## **RDBMS Terminology**

Before we proceed to explain the MySQL database system, let us revise a few definitions related to the database.

* **Database** − A database is a collection of tables, with related data.
* **Table** − A table is a matrix with data. A table in a database looks like a simple spreadsheet.
* **Column** − One column (data element) contains data of one and the same kind, for example the column postcode.
* **Row** − A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.
* **Redundancy** − Storing data twice, redundantly to make the system faster.
* **Primary Key** − A primary key is unique. A key value cannot occur twice in one table. With a key, you can only find one row.
* **Foreign Key** − A foreign key is the linking pin between two tables.
* **Compound Key** − A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
* **Index** − An index in a database resembles an index at the back of a book.
* **Referential Integrity** − Referential Integrity makes sure that a foreign key value always points to an existing row.

## **MySQL Data Types: -**

Each column in a database table is required to have a name and a data type.

An SQL developer must decide what type of data that will be stored inside each column when creating a table. The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.

In MySQL there are three main data types: string, numeric, and date and time.

### String Data Types

|  |  |
| --- | --- |
| **Data type** | **Description** |
| CHAR(size) | A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1 |
| VARCHAR(size) | A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535 |
| BINARY(size) | Equal to CHAR(), but stores binary byte strings. The size parameter specifies the column length in bytes. Default is 1 |
| VARBINARY(size) | Equal to VARCHAR(), but stores binary byte strings. The size parameter specifies the maximum column length in bytes. |
| TINYBLOB | For BLOBs (Binary Large OBjects). Max length: 255 bytes |
| TINYTEXT | Holds a string with a maximum length of 255 characters |
| TEXT(size) | Holds a string with a maximum length of 65,535 bytes |
| BLOB(size) | For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data |
| MEDIUMTEXT | Holds a string with a maximum length of 16,777,215 characters |
| MEDIUMBLOB | For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data |
| LONGTEXT | Holds a string with a maximum length of 4,294,967,295 characters |
| LONGBLOB | For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data |
| ENUM(val1, val2, val3, ...) | A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them |
| SET(val1, val2, val3, ...) | A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list |

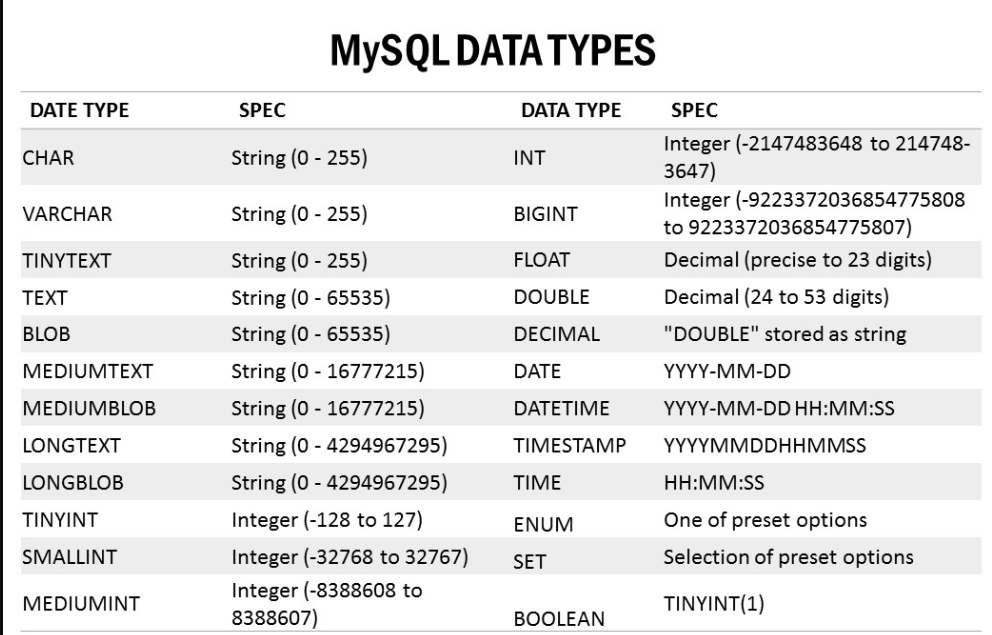
### Numeric Data Types

|  |  |
| --- | --- |
| **Data type** | **Description** |
| BIT(size) | A bit-value type. The number of bits per value is specified in size. The size parameter can hold a value from 1 to 64. The default value for size is 1. |
| TINYINT(size) | A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The size parameter specifies the maximum display width (which is 255) |
| BOOL | Zero is considered as false, nonzero values are considered as true. |
| BOOLEAN | Equal to BOOL |
| SMALLINT(size) | A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The size parameter specifies the maximum display width (which is 255) |
| MEDIUMINT(size) | A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The size parameter specifies the maximum display width (which is 255) |
| INT(size) | A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The size parameter specifies the maximum display width (which is 255) |
| INTEGER(size) | Equal to INT(size) |
| BIGINT(size) | A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The size parameter specifies the maximum display width (which is 255) |
| FLOAT(size, d) | A floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions |
| FLOAT(p) | A floating point number. MySQL uses the p value to determine whether to use FLOAT or DOUBLE for the resulting data type. If p is from 0 to 24, the data type becomes FLOAT(). If p is from 25 to 53, the data type becomes DOUBLE() |
| DOUBLE(size, d) | A normal-size floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter |
| DOUBLE PRECISION(size, d) |  |
| DECIMAL(size, d) | An exact fixed-point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. The maximum number for size is 65. The maximum number for d is 30. The default value for size is 10. The default value for d is 0. |
| DEC(size, d) | Equal to DECIMAL(size,d) |

**Note:** All the numeric data types may have an extra option: UNSIGNED or ZEROFILL. If you add the UNSIGNED option, MySQL disallows negative values for the column. If you add the ZEROFILL option, MySQL automatically also adds the UNSIGNED attribute to the column.

### Date and Time Data Types

|  |  |
| --- | --- |
| **Data type** | **Description** |
| DATE | A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31' |
| DATETIME(fsp) | A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time |
| TIMESTAMP(fsp) | A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT\_TIMESTAMP and ON UPDATE CURRENT\_TIMESTAMP in the column definition |
| TIME(fsp) | A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59' |
| YEAR | A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format. |



## **MySQL Functions**

MySQL has many built-in functions.

This reference contains string, numeric, date, and some advanced functions in MySQL.

***MySQL String Functions***

|  |  |
| --- | --- |
| **Function** | **Description** |
| [ASCII](https://www.w3schools.com/mysql/func_mysql_ascii.asp) | Returns the ASCII value for the specific character |
| [CHAR\_LENGTH](https://www.w3schools.com/mysql/func_mysql_char_length.asp) | Returns the length of a string (in characters) |
| [CHARACTER\_LENGTH](https://www.w3schools.com/mysql/func_mysql_character_length.asp) | Returns the length of a string (in characters) |
| [CONCAT](https://www.w3schools.com/mysql/func_mysql_concat.asp) | Adds two or more expressions together |
| [CONCAT\_WS](https://www.w3schools.com/mysql/func_mysql_concat_ws.asp) | Adds two or more expressions together with a separator |
| [FIELD](https://www.w3schools.com/mysql/func_mysql_field.asp) | Returns the index position of a value in a list of values |
| [FIND\_IN\_SET](https://www.w3schools.com/mysql/func_mysql_find_in_set.asp) | Returns the position of a string within a list of strings |
| [FORMAT](https://www.w3schools.com/mysql/func_mysql_format.asp) | Formats a number to a format like "#,###,###.##", rounded to a specified number of decimal places |
| [INSERT](https://www.w3schools.com/mysql/func_mysql_insert.asp) | Inserts a string within a string at the specified position and for a certain number of characters |
| [INSTR](https://www.w3schools.com/mysql/func_mysql_instr.asp) | Returns the position of the first occurrence of a string in another string |
| [LCASE](https://www.w3schools.com/mysql/func_mysql_lcase.asp) | Converts a string to lower-case |
| [LEFT](https://www.w3schools.com/mysql/func_mysql_left.asp) | Extracts a number of characters from a string (starting from left) |
| [LENGTH](https://www.w3schools.com/mysql/func_mysql_length.asp) | Returns the length of a string (in bytes) |
| [LOCATE](https://www.w3schools.com/mysql/func_mysql_locate.asp) | Returns the position of the first occurrence of a substring in a string |
| [LOWER](https://www.w3schools.com/mysql/func_mysql_lower.asp) | Converts a string to lower-case |
| [LPAD](https://www.w3schools.com/mysql/func_mysql_lpad.asp) | Left-pads a string with another string, to a certain length |
| [LTRIM](https://www.w3schools.com/mysql/func_mysql_ltrim.asp) | Removes leading spaces from a string |
| [MID](https://www.w3schools.com/mysql/func_mysql_mid.asp) | Extracts a substring from a string (starting at any position) |
| [POSITION](https://www.w3schools.com/mysql/func_mysql_position.asp) | Returns the position of the first occurrence of a substring in a string |
| [REPEAT](https://www.w3schools.com/mysql/func_mysql_repeat.asp) | Repeats a string as many times as specified |
| [REPLACE](https://www.w3schools.com/mysql/func_mysql_replace.asp) | Replaces all occurrences of a substring within a string, with a new substring |
| [REVERSE](https://www.w3schools.com/mysql/func_mysql_reverse.asp) | Reverses a string and returns the result |
| [RIGHT](https://www.w3schools.com/mysql/func_mysql_right.asp) | Extracts a number of characters from a string (starting from right) |
| [RPAD](https://www.w3schools.com/mysql/func_mysql_rpad.asp) | Right-pads a string with another string, to a certain length |
| [RTRIM](https://www.w3schools.com/mysql/func_mysql_rtrim.asp) | Removes trailing spaces from a string |
| [SPACE](https://www.w3schools.com/mysql/func_mysql_space.asp) | Returns a string of the specified number of space characters |
| [STRCMP](https://www.w3schools.com/mysql/func_mysql_strcmp.asp) | Compares two strings |
| [SUBSTR](https://www.w3schools.com/mysql/func_mysql_substr.asp) | Extracts a substring from a string (starting at any position) |
| [SUBSTRING](https://www.w3schools.com/mysql/func_mysql_substring.asp) | Extracts a substring from a string (starting at any position) |
| [SUBSTRING\_INDEX](https://www.w3schools.com/mysql/func_mysql_substring_index.asp) | Returns a substring of a string before a specified number of delimiter occurs |
| [TRIM](https://www.w3schools.com/mysql/func_mysql_trim.asp) | Removes leading and trailing spaces from a string |
| [UCASE](https://www.w3schools.com/mysql/func_mysql_ucase.asp) | Converts a string to upper-case |
| [UPPER](https://www.w3schools.com/mysql/func_mysql_upper.asp) | Converts a string to upper-case |

|  |  |
| --- | --- |
| **Handy MySQL Commands** | |
| **Description** | **Command** |
| To login (from unix shell) use -h only if needed. | [mysql dir]/bin/mysql -h hostname -u root -p |
| Create a database on the sql server. | create database [databasename]; |
| List all databases on the sql server. | show databases; |
| Switch to a database. | use [db name]; |
| To see all the tables in the db. | show tables; |
| To see database's field formats. | describe [table name]; |
| To delete a db. | drop database [database name]; |
| To delete a table. | drop table [table name]; |
| Show all data in a table. | SELECT \* FROM [table name]; |
| Returns the columns and column information pertaining to the designated table. | show columns from [table name]; |  | |
|  |  |  |
| Show certain selected rows with the value "whatever". | SELECT \* FROM [table name] WHERE [field name] = "whatever"; |  | |
|  |  |  |
| Show all records containing the name "Bob" AND the phone number '3444444'. | SELECT \* FROM [table name] WHERE name = "Bob" AND phone\_number = '3444444'; |  | |
|  |  |  |
| Show all records not containing the name "Bob" AND the phone number '3444444' order by the phone\_number field. | SELECT \* FROM [table name] WHERE name != "Bob" AND phone\_number = '3444444' order by phone\_number; |  | |
|  |  |  |
| Show all records starting with the letters 'bob' AND the phone number '3444444'. | SELECT \* FROM [table name] WHERE name like "Bob%" AND phone\_number = '3444444'; |  | |
|  |  |  |
| Use a regular expression to find records. Use "REGEXP BINARY" to force case-sensitivity. This finds any record beginning with a. | SELECT \* FROM [table name] WHERE rec RLIKE "^a$"; |  | |
|  |  |  |
| Show unique records. | SELECT DISTINCT [column name] FROM [table name]; |  |
| Show selected records sorted in an ascending (asc) or descending (desc). | SELECT [col1],[col2] FROM [table name] ORDER BY [col2] DESC; |  |
| Count rows. | SELECT COUNT(\*) FROM [table name]; |  | |
|  |  |  |
| Join tables on common columns. | select lookup.illustrationid, lookup.personid,person.birthday from lookup left join person on lookup.personid=person.personid=statement to join birthday in person table with primary illustration id; |  |
| Switch to the mysql db. Create a new user. | INSERT INTO [table name] (Host,User,Password) VALUES('%','user',PASSWORD('password')); |  |
| Change a users password.(from unix shell). | [mysql dir]/bin/mysqladmin -u root -h hostname.blah.org -p password 'new-password' |  |
| Change a users password.(from MySQL prompt). | SET PASSWORD FOR 'user'@'hostname' = PASSWORD('passwordhere'); |  |
| Switch to mysql db.Give user privilages for a db. | INSERT INTO [table name] (Host,Db,User,Select\_priv,Insert\_priv,Update\_priv,Delete\_priv,Create\_priv,Drop\_priv) VALUES ('%','db','user','Y','Y','Y','Y','Y','N'); |  |
| To update info already in a table. | UPDATE [table name] SET Select\_priv = 'Y',Insert\_priv = 'Y',Update\_priv = 'Y' where [field name] = 'user'; |  |
| Delete a row(s) from a table. | DELETE from [table name] where [field name] = 'whatever'; |  |
| Update database permissions/privilages. | FLUSH PRIVILEGES; |  |
| Delete a column. | alter table [table name] drop column [column name]; |  |
| Add a new column to db. | alter table [table name] add column [new column name] varchar (20); |  |
| Change column name. | alter table [table name] change [old column name] [new column name] varchar (50); |  |
| Make a unique column so you get no dupes. | alter table [table name] add unique ([column name]); |  |
| Make a column bigger. | alter table [table name] modify [column name] VARCHAR(3); |  |
| Delete unique from table. | alter table [table name] drop index [colmn name]; |  |
| Load a CSV file into a table. | LOAD DATA INFILE '/tmp/filename.csv' replace INTO TABLE [table name] FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' (field1,field2,field3); |  |
| Dump all databases for backup. Backup file is sql commands to recreate all db's. | [mysql dir]/bin/mysqldump -u root -ppassword --opt >/tmp/alldatabases.sql |  |
| Dump one database for backup. | [mysql dir]/bin/mysqldump -u username -ppassword --databases databasename >/tmp/databasename.sql |  |
| Dump a table from a database. | [mysql dir]/bin/mysqldump -c -u username -ppassword databasename tablename > /tmp/databasename.tablename.sql |  |
| Restore database (or database table) from backup. | [mysql dir]/bin/mysql -u username -ppassword databasename < /tmp/databasename.sql |  |
| Create Table Example 1. | CREATE TABLE [table name] (firstname VARCHAR(20), middleinitial VARCHAR(3), lastname VARCHAR(35),suffix VARCHAR(3), officeid VARCHAR(10),userid VARCHAR(15),username VARCHAR(8),email VARCHAR(35),phone VARCHAR(25), groups VARCHAR(15),datestamp DATE,timestamp time,pgpemail VARCHAR(255)); |  |
| Create Table Example 2. | create table [table name] (personid int(50) not null auto\_increment primary key,firstname varchar(35),middlename varchar(50),lastname varchar(50) default 'bato'); |  |