**MySQL Commands and Introduction**

MySQL can be divided into two parts:

1. The Data Manipulation Language (DML)
2. Data Definition Language (DDL).

The query and update commands form the DML part of MYSQL:

* SELECT - extracts data from a database
* UPDATE - updates data in a database
* DELETE - deletes data from a database
* INSERT INTO - inserts new data into a database

The DDL part of MYSQL permits database tables to be created or deleted. It also defines indexes (keys), specify links between tables, and impose constraints between tables.

The most important DDL statements in MYSQL are:

* CREATE DATABASE - creates a new database
* ALTER DATABASE - modifies a database
* CREATE TABLE - creates a new table
* ALTER TABLE - modifies a table
* DROP TABLE - deletes a table
* CREATE INDEX - creates an index (search key)
* DROP INDEX - deletes an index

**MySQL DDL Commands (Revision)**

# **Create Database:**

The CREATE DATABASE statement is used to create a new SQL database.

Command: CREATE DATABASE databasename;

**Create Table:**

The CREATE TABLE statement is used to create a new table in a database.

Command: CREATE TABLE table\_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
   ....  
);

Example: CREATE TABLE Persons (

PersonID int,

LastName varchar(255),

FirstName varchar(255),

Address varchar(255),

City varchar(255)

);

**Drop Table:**

The DROP TABLE statement is used to drop an existing table in a database.

Command: DROP TABLE tablename;

**Alter Table:**

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table. The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

* ADD Column Command:
  + ALTER TABLE table\_name  
    ADD column\_name datatype;
* Drop Column Command:
  + ALTER TABLE table\_name  
    DROP COLUMN column\_name;
* Modify Column (data type) Command:
  + ALTER TABLE table\_name  
    ALTER COLUMN column\_name datatype;
* Add constraints Command:
  + ALTER TABLE table\_name  
    add primary key (column name);
  + ALTER TABLE table\_name  
    ADD FOREIGN KEY (column name) REFERENCES Persons(column name);

**SQL Constraints**

The following constraints are commonly used in SQL:

[NOT NULL](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value

[UNIQUE](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different

[PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

[FOREIGN KEY](https://www.w3schools.com/sql/sql_foreignkey.asp) - Prevents actions that would destroy links between tables

[CHECK](https://www.w3schools.com/sql/sql_check.asp) - Ensures that the values in a column satisfies a specific condition

[DEFAULT](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column if no value is specified

[CREATE INDEX](https://www.w3schools.com/sql/sql_create_index.asp) - Used to create and retrieve data from the database very quickly

**MySQL DML Commands**

### **INSERT INTO Statement:**

It is possible to write the INSERT INTO statement in two ways:

1. Specify both the column names and the values to be inserted:

INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:

INSERT INTO table\_name  
VALUES (value1, value2, value3, ...);

## **The SQL SELECT Statement**

The SELECT statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

### **SELECT Syntax**

SELECT column1, column2, ...  
FROM table\_name;

Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

SELECT \* FROM table\_name;

## **The SQL SELECT DISTINCT Statement**

The SELECT DISTINCT statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

### **SELECT DISTINCT Syntax**

SELECT DISTINCT column1, column2, ...  
FROM table\_name;

## **The SQL WHERE Clause**

The WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

### **WHERE Syntax**

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

**Note:** The WHERE clause is not only used in SELECT statements, it is also used in UPDATE, DELETE, etc.!

## **Operators in The WHERE Clause**

The following operators can be used in the WHERE clause:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Equal | SELECT \* FROM Products  WHERE Price = 18; |
| > | Greater than | SELECT \* FROM Products  WHERE Price > 30; |
| < | Less than | SELECT \* FROM Products  WHERE Price < 30; |
| >= | Greater than or equal | SELECT \* FROM Products  WHERE Price >= 30; |
| <= | Less than or equal | SELECT \* FROM Products  WHERE Price <= 30; |
| <> | Not equal. **Note:** In some versions of SQL this operator may be written as != | SELECT \* FROM Products  WHERE Price <> 18; |
| BETWEEN | Between a certain range | SELECT \* FROM Products  WHERE Price BETWEEN 50 AND 60; |
| LIKE | Search for a pattern | SELECT \* FROM Customers  WHERE City LIKE 's%'; |
| IN | To specify multiple possible values for a column | SELECT \* FROM Customers  WHERE City IN ('Paris','London'); |

## **The SQL AND, OR and NOT Operators**

The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition:

* The AND operator displays a record if all the conditions separated by AND are TRUE.
* The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

### **AND Syntax**

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 AND condition2 AND condition3 ...;

### **OR Syntax**

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 OR condition2 OR condition3 ...;

### **NOT Syntax**

SELECT column1, column2, ...  
FROM table\_name  
WHERE NOT condition;

## **The SQL ORDER BY Keyword**

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

### **ORDER BY Syntax**

SELECT column1, column2, ...  
FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC;

## **The SQL LIKE Operator**

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

* The percent sign (%) represents zero, one, or multiple characters
* The underscore sign (\_) represents one, single character
* Here are some examples showing different LIKE operators with '%' and '\_' wildcards:

|  |  |
| --- | --- |
| Here are some examples showing different LIKE operators with '%' and '\_' wildcards: |  |
|  |  |
| LIKE Operator | Description |
| WHERE CustomerName LIKE 'a%' | Finds any values that start with "a" |
| WHERE CustomerName LIKE '%a' | Finds any values that end with "a" |
| WHERE CustomerName LIKE '%or%' | Finds any values that have "or" in any position |
| WHERE CustomerName LIKE '\_r%' | Finds any values that have "r" in the second position |
| WHERE CustomerName LIKE 'a\_%' | Finds any values that start with "a" and are at least 2 characters in length |

## **The SQL UPDATE Statement**

The UPDATE statement is used to modify the existing records in a table.

### **UPDATE Syntax**

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

The following SQL statement updates the first customer (CustomerID = 1) with a new contact person and a new city.

### **Example**

UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1;

## **Update Warning!**

Be careful when updating records. If you omit the WHERE clause, ALL records will be updated!

## **The SQL DELETE Statement**

The DELETE statement is used to delete existing records in a table.

### **DELETE Syntax**

DELETE FROM table\_name WHERE condition;

## **Delete All Records**

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

DELETE FROM table\_name;