**MySQL**

MySQL is a popular open-source relational database management system (RDBMS) that is widely used for storing and managing structured data.

**What is RDBMS?**

RDBMS stands for Relational Database Management System.

RDBMS is a program used to maintain a relational database.

RDBMS is the basis for all modern database systems such as MySQL, Microsoft SQL Server, Oracle, and Microsoft Access.

RDBMS uses [SQL queries](https://www.w3schools.com/sql/default.asp) to access the data in the database.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CustomerID** | **CustomerName** | **ContactName** | **Address** | **City** | **PostalCode** | **Country** |
| 1 | Alfreds Futterkiste | Maria Anders | Obere Str. 57 | Berlin | 12209 | Germany |
| 2 | Ana Trujillo Emparedados y helados | Ana Trujillo | Avda. de la Constitución 2222 | México D.F. | 05021 | Mexico |
| 3 | Antonio Moreno Taquería | Antonio Moreno | Mataderos 2312 | México D.F. | 05023 | Mexico |
| 4 | Around the Horn | Thomas Hardy | 120 Hanover Sq. | London | WA1 1DP | UK |
| 5 | Berglunds snabbköp | Christina Berglund | Berguvsvägen 8 | Luleå | S-958 22 | Sweden |

**SELECT Columns Example**

SELECT CustomerName, City, Country FROM Customers;

## SELECT \* Example

SELECT \* FROM Customers;

## WHERE Clause Example

SELECT \* FROM Customers  
WHERE Country = 'Mexico';

## 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 ...;  
  
EX-2:

SELECT \* FROM Customers  
WHERE Country = 'Germany' AND City = 'Berlin';

### OR Syntax

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

EX-2:

SELECT \* FROM Customers  
WHERE City = 'Berlin' OR City = 'Stuttgart';

### NOT Syntax

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

Ex-2:

SELECT \* FROM Customers  
WHERE NOT Country = 'Germany';

## 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;

Ex-1;

SELECT \* FROM Customers  
ORDER BY Country;

(Ascending order)

Ex-2;

SELECT \* FROM Customers  
ORDER BY Country DESC ;

(Descending Order)

## INSERT INTO Example

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

## What is a NULL Value?

A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

**Note:** A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!

Ex-1:

SELECT CustomerName, ContactName, Address  
FROM Customers  
WHERE Address IS NOT NULL;

## UPDATE Statement

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

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

## The MySQL DELETE Statement

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

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

## LIMIT Clause

The LIMIT clause is used to specify the number of records to return.

The LIMIT clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

### LIMIT Syntax

SELECT column\_name(s)  
FROM table\_nameWHERE condition  
LIMIT number;

ex-1;

SELECT \* FROM Customers  
LIMIT 3;

## MIN() Example

SELECT MIN(Price) AS SmallestPrice  
FROM Products;

## MAX() Example

SELECT MAX(Price) AS SmallestPrice  
FROM Products;

## COUNT() Example

SELECT COUNT(ProductID)  
FROM Products;

## AVG() Example

SELECT AVG(Price)  
FROM Products;

## SUM() Example

SELECT SUM(Quantity)  
FROM OrderDetails;

**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

The percent sign and the underscore can also be used in combinations!

|  |  |
| --- | --- |
| **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 |
| WHERE CustomerName LIKE 'a\_\_%' | Finds any values that start with "a" and are at least 3 characters in length |
| WHERE ContactName LIKE 'a%o' | Finds any values that start with "a" and ends with "o" |

### LIKE Syntax

SELECT column1, column2, ...  
FROM table\_name  
WHERE columnN LIKE pattern;

Ex-1:

SELECT \* FROM Customers  
WHERE CustomerName LIKE 'a%';

**Database:**

* **Definition**: A database is a structured collection of data organized into tables. It's used for efficient data storage, retrieval, and management.

CREATE DATABASE mydb;

***Table:***

* ***Definition: A table is a database object consisting of rows and columns to store related data.***

CREATE TABLE users (

id INT PRIMARY KEY,

name VARCHAR(50),

email VARCHAR(100)

);

## PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

***id INT PRIMARY KEY;***

## FOREIGN KEY Constraint

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the [PRIMARY KEY](https://www.w3schools.com/MySQL/mysql_primarykey.asp) in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

***user\_id INT,***

***FOREIGN KEY (user\_id) REFERENCES users(id);***

## ALTER TABLE Statement

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.

ALTER TABLE Customers  
DROP COLUMN Email;

# MySQL DROP TABLE Statement

DROP TABLE Shippers;

## TRUNCATE TABLE

The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.

TRUNCATE TABLE table\_name;

# Joins

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

**Supported Types of Joins**

* INNER JOIN: Returns records that have matching values in both tables
* LEFT JOIN: Returns all records from the left table, and the matched records from the right table
* RIGHT JOIN: Returns all records from the right table, and the matched records from the left table
* CROSS JOIN: Returns all records from both tables

Customers Table:

| customer\_id | customer\_name |

|------------|---------------|

| 1 | Alice |

| 2 | Bob |

| 3 | Carol |

| 4 | David |

Orders Table:

| order\_id | order\_date | customer\_id |

|----------|-------------|-------------|

| 101 | 2023-01-10 | 1 |

| 102 | 2023-01-15 | 2 |

| 103 | 2023-02-20 | 1 |

| 104 | 2023-03-05 | 3 |

Products Table:

| product\_id | product\_name | price |

|------------|--------------|-------|

| 201 | Laptop | 800 |

| 202 | Smartphone | 300 |

| 203 | Tablet | 400 |

| 204 | TV | 600 |

OrderDetails Table:

| order\_id | product\_id | quantity |

|----------|------------|----------|

| 101 | 201 | 1 |

| 102 | 202 | 2 |

| 102 | 203 | 1 |

| 103 | 201 | 3 |

| 104 | 204 | 1 |

## INNER JOIN Example

SELECT Orders.OrderID, Customers.CustomerName  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

+---------------+------------+

| customer\_name | order\_date |

+---------------+------------+

| Alice | 2023-01-10 |

| Bob | 2023-01-15 |

| Alice | 2023-02-20 |

+---------------+------------+

## LEFT JOIN Example

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID  
ORDER BY Customers.CustomerName;

+---------------+------------+

| customer\_name | order\_date |

+---------------+------------+

| Alice | 2023-01-10 |

| Bob | 2023-01-15 |

| Alice | 2023-02-20 |

| Carol | NULL |

+---------------+------------+

## RIGHT JOIN Example

SELECT Orders.OrderID, Employees.LastName, Employees.FirstName  
FROM Orders  
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID  
ORDER BY Orders.OrderID;

+---------------+------------+

| customer\_name | order\_date |

+---------------+------------+

| Alice | 2023-01-10 |

| Bob | 2023-01-15 |

| Alice | 2023-02-20 |

| NULL | 2023-03-05 |

+---------------+------------+

## CROSS JOIN Example

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
CROSS JOIN Orders  
WHERE Customers.CustomerID=Orders.CustomerID;

## Self Join Example

SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City  
FROM Customers A, Customers B  
WHERE A.CustomerID <> B.CustomerID  
AND A.City = B.City  
ORDER BY A.City;