How to Start MySql Server

Download latest version of XAMPP From <https://www.apachefriends.org/download.html>

Install XAMPP and open XAMPP Control Panel

From Control panel you need to start two services apache, mysql

Now open browser and enter URL <http://localhost/phpmyadmin/> or press admin button from XAMPP control panel.

Now you are connected with MySql server home page

**MySQL Tutorial**

MySQL is a widely used relational database management system (RDBMS).

MySQL is free and open-source.

MySQL is ideal for both small and large applications.

# Introduction to MySQL

MySQL is a very popular open-source relational database management system (RDBMS).

## What is MySQL?

* MySQL is a relational database management system
* MySQL is open-source
* MySQL is free
* MySQL is ideal for both small and large applications
* MySQL is very fast, reliable, scalable, and easy to use
* MySQL is cross-platform
* MySQL is compliant with the ANSI SQL standard
* MySQL was first released in 1995
* MySQL is developed, distributed, and supported by **Oracle Corporation**
* MySQL is named after co-founder Monty Widenius's daughter: My

## Who Uses MySQL?

* Huge websites like Facebook, Twitter, Airbnb, Booking.com, Uber, GitHub, YouTube, etc.
* Content Management Systems like WordPress, Drupal, Joomla!, Contao, etc.
* A very large number of web developers around the world

## Show Data On Your Web Site

To build a web site that shows data from a database, you will need:

* An RDBMS database program (like MySQL)
* A server-side scripting language, like PHP
* To use SQL to get the data you want
* To use HTML / CSS to style the page

## 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.

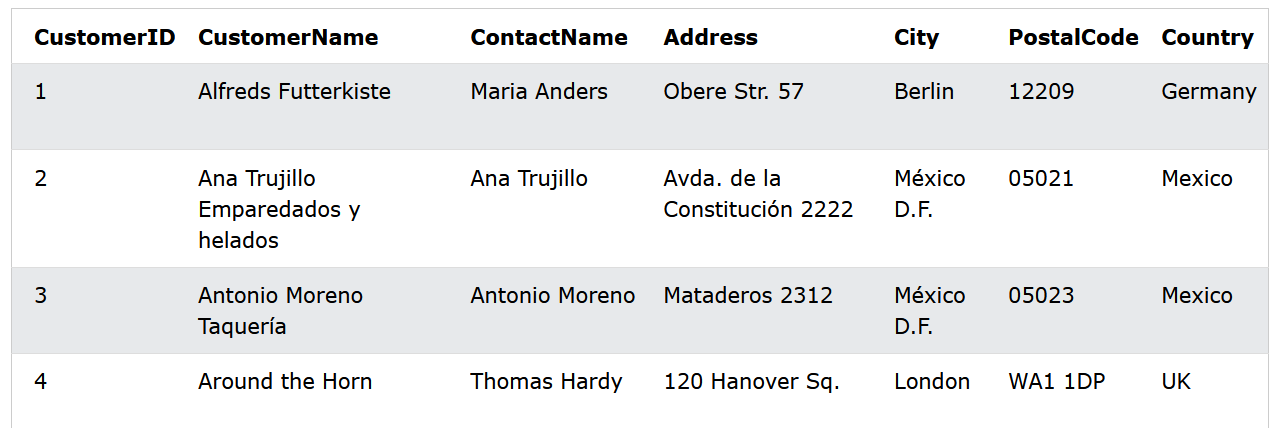
## What is a Database Table?

A table is a collection of related data entries, and it consists of columns and rows.

A column holds specific information about every record in the table.

A record (or row) is each individual entry that exists in a table.

Look at a selection from the Northwind "Customers" table:



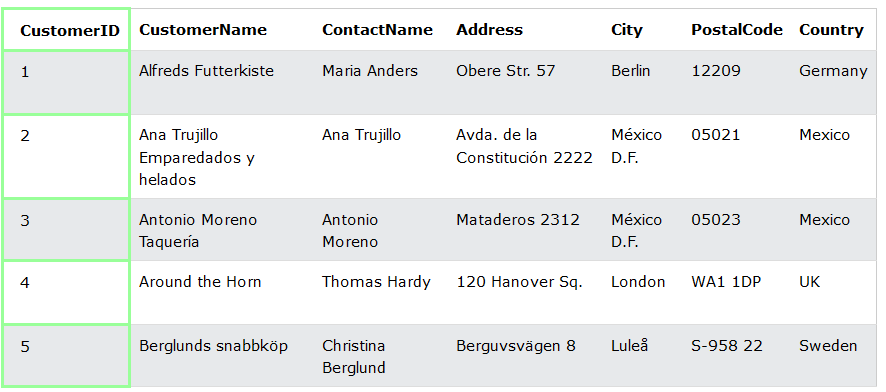
The columns in the "Customers" table above are: CustomerID, CustomerName, ContactName, Address, City, PostalCode and Country. The table has 4 records (rows).

## What is a Relational Database?

A relational database defines database relationships in the form of tables. The tables are related to each other - based on data common to each.

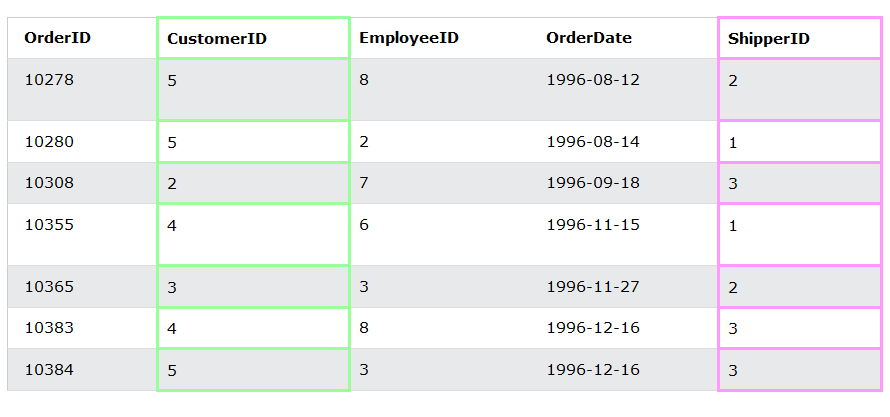
Look at the following three tables "Customers", "Orders", and "Shippers" from the Northwind database:

### Customers Table



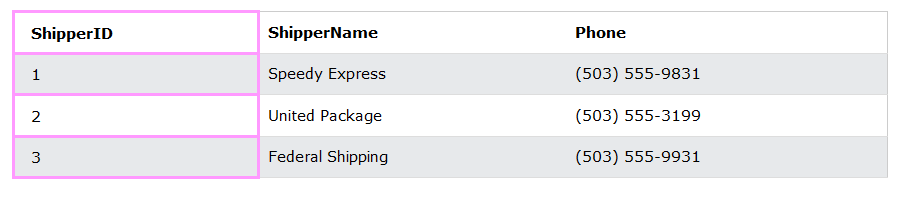
The relationship between the "Customers" table and the "Orders" table is the CustomerID column:

### Orders Table



The relationship between the "Orders" table and the "Shippers" table is the ShipperID column:

### Shippers Table



# MySQL SQL

## What is SQL (Structured Query Language)?

SQL is the standard language for dealing with Relational Databases.

SQL is used to insert, search, update, and delete database records.

## How to Use SQL

The following SQL statement selects all the records in the "Customers" table:

SELECT \* FROM Customers;

## Keep in Mind That...

* SQL keywords are NOT case sensitive: select is the same as SELECT

In this tutorial we will write all SQL keywords in upper-case.

## Semicolon after SQL Statements?

Some database systems require a semicolon at the end of each SQL statement.

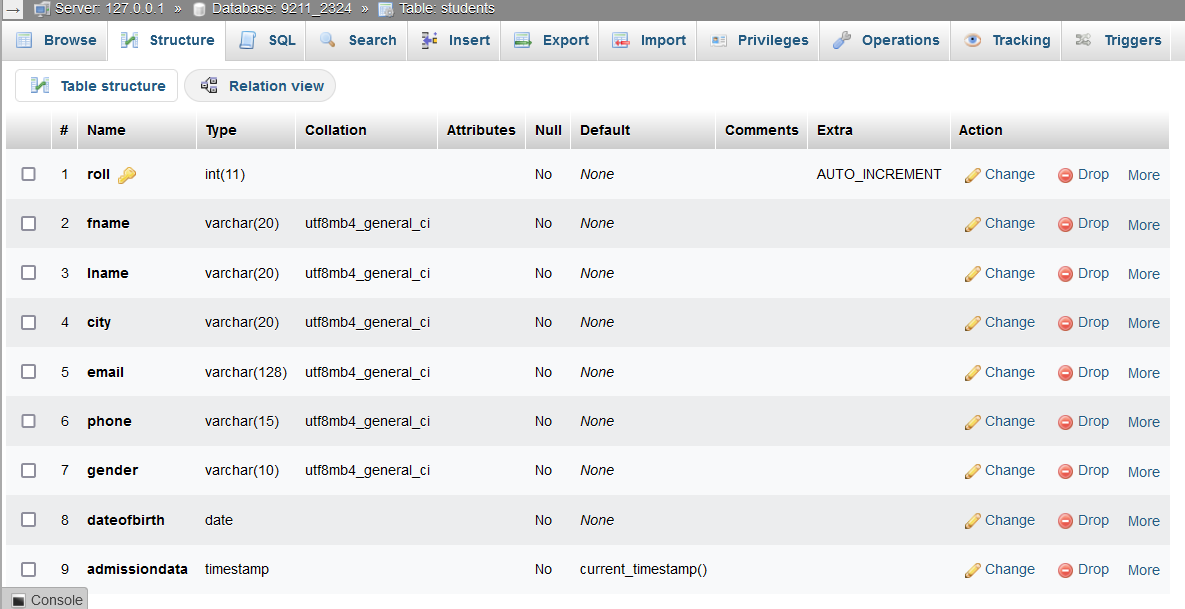
Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

In this tutorial, we will use semicolon at the end of each SQL statement.

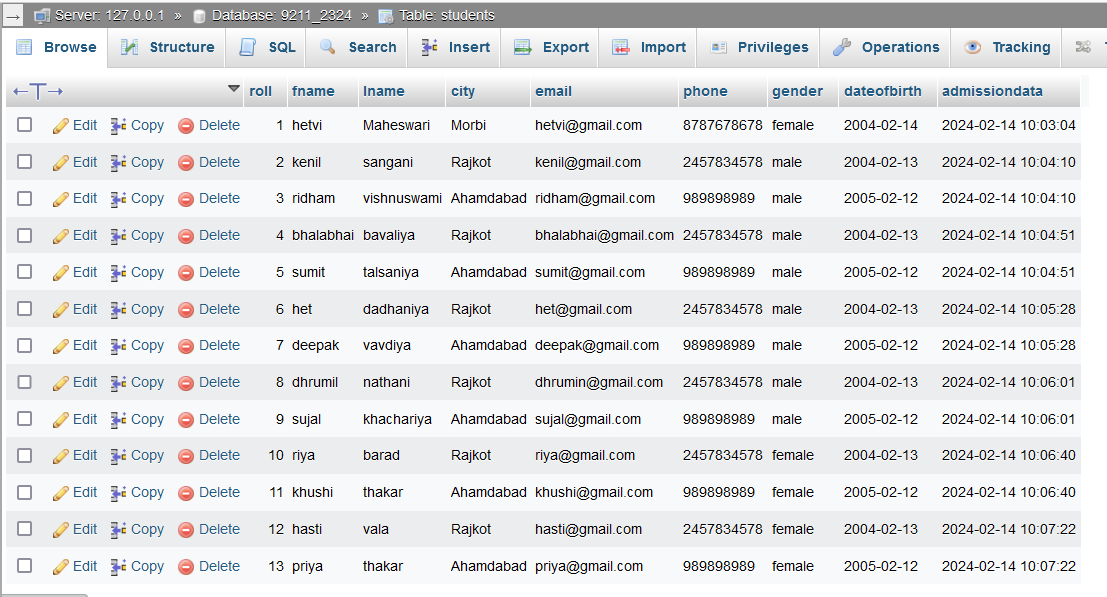
## Some of The Most Important SQL Commands

* 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
* 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

Create Database and Table as like following



And add some data as following



# MySQL SELECT Statement

## The MySQL 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;

SELECT \* from students;

SELECT roll, fname, lname, city from students

## The MySQL 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;

SELECT city from students;

SELECT DISTINCT city from students;

The following SQL statement counts and returns the number of different (distinct) city in the "Students" table:

SELECT count(DISTINCT city) from students;

# MySQL WHERE Clause

## The MySQL 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.!

SELECT \* FROM students WHERE roll = 1

SELECT \* FROM students WHERE roll > 5;

SELECT \* FROM students WHERE not roll > 5;

SELECT \* FROM students WHERE city = 'rajkot';

SELECT \* FROM students WHERE city <> 'rajkot';

SELECT \* FROM students WHERE not city = 'rajkot';

SELECT \* FROM students WHERE roll BETWEEN 1 and 5

# MySQL AND, OR and NOT Operators

## The MySQL 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 ...;

SELECT \* from students WHERE roll = 1

SELECT \* from students WHERE roll = 1 and city = 'Rajkot';

### OR Syntax

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

SELECT \* from students WHERE city = 'morbi' or city = 'Rajkot';

### NOT Syntax

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

SELECT \* from students WHERE not city = 'morbi' ;

## Combining AND, OR and NOT

You can also combine the AND, OR and NOT operators.

SELECT \* from students WHERE roll = 1 and city = 'rajkot' or city = 'morbi'

SELECT \* from students WHERE roll = 1 and (city = 'rajkot' or city = 'morbi')

# MySQL ORDER BY Keyword

## The MySQL 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;

SELECT \* from students;

SELECT \* from students ORDER by fname;

## ORDER BY DESC Example

The following SQL statement selects all customers from the "students" table, sorted DESCENDING by the "fname" column:

SELECT \* from students ORDER by fname DESC;

## ORDER BY Several Columns Example

SELECT \* from students ORDER by fname, city;

SELECT \* from students ORDER by fname asc, city DESC;

# MySQL INSERT INTO Statement

## The MySQL INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

### INSERT INTO Syntax

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*, ...);

INSERT into students (fname, lname, city, email, phone, gender, dateofbirth) VALUES ('Demo', 'text', 'example', 'demo@example.com', '9876543210', 'male', '2001-01-01')

**Did you notice that we did not insert any number into the CustomerID field?**  
The CustomerID column is an [auto-increment](https://www.w3schools.com/mysql/mysql_autoincrement.asp) field and will be generated automatically when a new record is inserted into the table.

## Insert Data Only in Specified Columns

It is also possible to only insert data in specific columns.

INSERT into students (fname, lname, city, email, phone) VALUES ('Demo', 'text', 'example', 'demo@example.com', '9876543210')

# MySQL NULL Values

## 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!

INSERT into students (fname, lname, city, email, phone) VALUES ('Demo', 'text', 'example', 'demo@example.com', '9876543210')

SELECT \* FROM `students` WHERE gender = '';

SELECT \* FROM `students` WHERE gender = 'NULL';

SELECT \* FROM `students` WHERE gender is null;

## How to Test for NULL Values?

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

We will have to use the IS NULL and IS NOT NULL operators instead.

### IS NULL Syntax

SELECT column\_namesFROM table\_name WHERE column\_name IS NULL;

SELECT \* FROM `students` WHERE gender is null;

### IS NOT NULL Syntax

SELECT column\_namesFROM table\_name WHERE column\_name IS NOT NULL;

SELECT \* FROM `students` WHERE gender is not null;

# MySQL UPDATE Statement

## The MySQL 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, ...

**Note:** Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!WHERE condition;

UPDATE students set gender = 'male' WHERE roll = 15 or roll = 16

UPDATE students set city = 'rajkot'

UPDATE students set city = 'surat' WHERE roll > 10

UPDATE students set city = 'ahamdabad' WHERE roll > 5 and roll < 10;

## UPDATE Multiple Records

It is the WHERE clause that determines how many records will be updated.

UPDATE students set gender = 'female', dateofbirth = '2001-01-15' WHERE roll >= 15

## Update Warning!

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

# MySQL LIMIT Clause

## The MySQL 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;

SELECT \* FROM `students`

SELECT \* FROM `students` limit 5;

What if we want to select records 6-10 (inclusive)?

MySQL provides a way to handle this: by using OFFSET.

The SQL query below says "return only 5 records, start on record 6 (OFFSET 5)":

SELECT \* FROM `students` limit 5 OFFSET 5;

SELECT \* FROM `students` limit 5 OFFSET 10;

SELECT \* FROM students WHERE city = 'rajkot' LIMIT 10 ;

SELECT \* from students LIMIT 5 OFFSET 10

SELECT \* from students LIMIT 10, 5; -- offset 10 limit 5

# MySQL MIN() and MAX() Functions

## MySQL MIN() and MAX() Functions

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

### MIN() Syntax

SELECT MIN(column\_name)  
FROM table\_name  
WHERE condition;

### MAX() Syntax

SELECT MAX(column\_name)  
FROM table\_name  
WHERE condition;

SELECT max(dateofbirth) FROM students

SELECT min(dateofbirth) FROM students;

SELECT min(roll) FROM students;

SELECT max(roll) FROM students;

# MySQL COUNT(), AVG() and SUM() Functions

## MySQL COUNT(), AVG() and SUM() Functions

The COUNT() function returns the number of rows that matches a specified criterion.

### COUNT() Syntax

SELECT COUNT(column\_name)  
FROM table\_name  
WHERE condition;

The AVG() function returns the average value of a numeric column.

### AVG() Syntax

SELECT AVG(column\_name)  
FROM table\_name  
WHERE condition;

The SUM() function returns the total sum of a numeric column.

### SUM() Syntax

SELECT SUM(column\_name)  
FROM table\_name  
WHERE condition;

SELECT COUNT(roll) from students

SELECT COUNT(roll) from students WHERE city = 'rajkot';

SELECT COUNT(roll) from students WHERE not city = 'rajkot';

SELECT sum(roll) from students

SELECT avg(roll) from students

SELECT sum(roll), avg(roll) from students;

# MySQL DELETE Statement

## The MySQL DELETE Statement

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

### DELETE Syntax

DELETE FROM table\_name WHERE condition;

**Note:** Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

Backup all the data of students table to studentsBackup table

create table studentsBackup as SELECT \* FROM students

## SQL DELETE Example

DELETE from students WHERE ROLL = 1

DELETE from students WHERE city = 'rajkot'

## 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;

DELETE from students

// restore

INSERT into students SELECT \* from studentsbackup

// when you delete all the data from table the auto increment is not reset to zero

INSERT INTO `students` (`roll`, `fname`, `lname`, `city`, `email`, `phone`, `gender`, `dateofbirth`, `admissiondata`) VALUES (NULL, 'brijesh', 'sinojiya', 'Morbi', 'demo@gmail.com', '998899889900', 'male', '2004-02-13', current\_timestamp());

A new record get a roll number as you leave before delete all the data

Use truncate table to delete all the data from table and also reset all auto increments

TRUNCATE TABLE students

INSERT INTO `students` (`roll`, `fname`, `lname`, `city`, `email`, `phone`, `gender`, `dateofbirth`, `admissiondata`) VALUES (NULL, 'brijesh', 'sinojiya', 'Morbi', 'demo@gmail.com', '998899889900', 'male', '2004-02-13', current\_timestamp());

Tuncate table students

Insert into students select \* from studentsbackup

DROP TABLE studentsbackup

# MySQL LIKE Operator

## The MySQL 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 Syntax

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

**Tip:** You can also combine any number of conditions using AND or OR operators.

SELECT \* from students WHERE fname like 'a%'

SELECT \* from students WHERE fname like 'a%'

SELECT \* from students WHERE fname like '%a%';

SELECT \* from students WHERE fname like '\_a%';

SELECT \* from students WHERE fname like '\_\_a%';

SELECT \* from students WHERE fname like '\_\_\_\_';

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 |
| 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" |

# MySQL Wildcards

## MySQL Wildcard Characters

A wildcard character is used to substitute one or more characters in a string.

Wildcard characters are used with the [LIKE](https://www.w3schools.com/mysql/mysql_like.asp) operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

### Wildcard Characters in MySQL

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Description** | **Example** |
| % | Represents zero or more characters | bl% finds bl, black, blue, and blob |
| \_ | Represents a single character | h\_t finds hot, hat, and hit |

The wildcards can also be used in combinations!

Here are some examples showing different LIKE operators with '%' and '\_' wildcards:

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

# MySQL IN Operator

## The MySQL IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

### IN Syntax

SELECT column\_name(s) FROM table\_name WHERE column\_name IN (value1, value2, ...);

or:

SELECT *column\_name(s)* FROM *table\_name* WHERE *column\_name* IN (*SELECT STATEMENT*);

SELECT \* from students WHERE city = 'rajkot' or city = 'surat' or city = 'baroda'

SELECT \* from students WHERE city in ('rajkot', 'baroda', 'surat')

SELECT \* from students WHERE city not in ('rajkot', 'baroda', 'surat');

# MySQL BETWEEN Operator

## The MySQL BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

### BETWEEN Syntax

SELECT column\_name(s) FROM table\_name WHERE column\_name BETWEEN value1 AND value2;

SELECT \* from students WHERE roll BETWEEN 1 and 5

SELECT \* from students WHERE roll not BETWEEN 1 and 5;

SELECT \* from students WHERE fname BETWEEN 'bhalabhai' and 'riya';

SELECT \* from students WHERE fname not BETWEEN 'bhalabhai' and 'riya';

SELECT \* FROM `students` WHERE dateofbirth BETWEEN '2001-01-01' and '2004-12-31'

SELECT \* FROM `students` WHERE dateofbirth not BETWEEN '2001-01-01' and '2004-12-31';

# MySQL Aliases

## MySQL Aliases

Aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of that query.

An alias is created with the AS keyword.

### Alias Column Syntax

SELECT column\_name AS alias\_name FROM table\_name;

### Alias Table Syntax

SELECT column\_name(s) FROM table\_name AS alias\_name;

SELECT roll, fname, lname, email, city, phone, dateofbirth from students

SELECT roll as "Roll Number", fname as "First Name", lname as "Last Name", email as "Email Address", city as HomeTown, phone as 'Phone Number', dateofbirth from students;

SELECT roll "Roll Number", fname "First Name", lname "Last Name", email "Email Address", city HomeTown, phone 'Phone Number', dateofbirth from students;

SELECT concat\_ws("\_", roll, fname, lname, email, city, phone, gender, dateofbirth) as "Student Information" from students

SELECT concat\_ws(" \* ", roll, fname, lname, email, city, phone, gender, dateofbirth) as "Student Information" from students;

**The following SQL statement is the same as above, but without aliases:**

SELECT students.roll, students.fname, students.lname, students.city, students.email, students.phone, students.gender, students.dateofbirth, students.admissiondata, marks.total, marks.result FROM students, marks WHERE students.roll = 1 and students.roll = marks.roll

**The following SQL statement is the same as above, but with aliases:**

SELECT s.roll, s.fname, s.lname, s.city, s.email, s.phone, s.gender, s.dateofbirth, s.admissiondata, m.total, m.result FROM students as s, marks as m WHERE s.roll = 1 and s.roll = m.roll;

SELECT s.roll, s.fname, s.lname, s.city, s.email, s.phone, s.gender, s.dateofbirth, s.admissiondata, m.total, m.result FROM students s, marks m WHERE s.roll = 1 and s.roll = m.roll;

Aliases can be useful when:

* There are more than one table involved in a query
* Functions are used in the query
* Column names are big or not very readable
* Two or more columns are combined together

SELECT s.roll, s.fname, s.lname, s.city, s.email, s.phone, s.gender, s.dateofbirth, s.admissiondata, a.absents, a.presents FROM students s, attendance a WHERE s.roll = 1 and s.roll = a.roll;

SELECT s.roll, s.fname, s.lname, s.city, s.email, s.phone, s.gender, s.dateofbirth, s.admissiondata, a.absents, a.presents, (a.absents+a.presents) FROM students s, attendance a WHERE s.roll = 1 and s.roll = a.roll;

SELECT s.roll, s.fname, s.lname, s.city, s.email, s.phone, s.gender, s.dateofbirth, s.admissiondata, a.absents, a.presents, (a.absents+a.presents) "Total Days" FROM students s, attendance a WHERE s.roll = 1 and s.roll = a.roll;

# MySQL Joins

## MySQL Joining Tables

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

Let's look at a selection from the "Orders" table:

|  |  |  |
| --- | --- | --- |
| **OrderID** | **CustomerID** | **OrderDate** |
| 10308 | 2 | 1996-09-18 |
| 10309 | 37 | 1996-09-19 |
| 10310 | 77 | 1996-09-20 |

Then, look at a selection from the "Customers" table:

|  |  |  |  |
| --- | --- | --- | --- |
| **CustomerID** | **CustomerName** | **ContactName** | **Country** |
| 1 | Alfreds Futterkiste | Maria Anders | Germany |
| 2 | Ana Trujillo Emparedados y helados | Ana Trujillo | Mexico |
| 3 | Antonio Moreno Taquería | Antonio Moreno | Mexico |

Notice that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship between the two tables above is the "CustomerID" column

## Supported Types of Joins in MySQL

* 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
*       

SELECT students.roll, students.fname, students.lname, students.city, students.email, students.phone, students.gender, students.dateofbirth, marks.total, marks.result from students INNER join marks on students.roll = marks.roll

# MySQL INNER JOIN Keyword

## MySQL INNER JOIN Keyword

The INNER JOIN keyword selects records that have matching values in both tables.



### INNER JOIN Syntax

SELECT column\_name(s) FROM table1 INNER JOIN table2ON table1.column\_name = table2.column\_name;

SELECT students.roll, students.fname, students.lname, students.dateofbirth, marks.total, marks.result from students INNER join marks on students.roll = marks.roll

Without alias

SELECT students.roll, students.fname, students.lname, students.city, students.email, students.phone, students.gender, students.admissiondata, marks.total, marks.result FROM students inner JOIN marks on students.roll = marks.roll

With alias

SELECT s.roll, s.fname, s.lname, s.city, s.email, s.phone, s.gender, s.admissiondata, m.total, m.result FROM students s inner JOIN marks m on s.roll = m.roll;

## JOIN Three Tables

SELECT students.roll, students.fname, students.lname, students.city, students.email, students.phone, students.gender, students.dateofbirth, marks.total, marks.result, attendance.absents, attendance.presents from students INNER join marks on students.roll = marks.roll INNER join attendance on students.roll = attendance.roll

SELECT s.roll, s.fname, s.lname, s.city, s.email, s.phone, s.gender, s.dateofbirth, m.total, m.result, a.absents, a.presents from students s INNER join marks m on s.roll = m.roll INNER join attendance a on s.roll = a.roll;

# MySQL LEFT JOIN Keyword

## MySQL LEFT JOIN Keyword

The LEFT JOIN keyword returns **all records from the left table** (table1), and the matching records (if any) from the right table (table2).



### LEFT JOIN Syntax

SELECT column\_name(s) FROM table1 LEFT JOIN table2ON table1.column\_name = table2.column\_name;

SELECT s.roll, s.fname, s.lname, s.city, s.email, s.phone, s.gender, s.dateofbirth, m.total, m.result from students s inner join marks m on s.roll = m.roll;

SELECT s.roll, s.fname, s.lname, s.city, s.email, s.phone, s.gender, s.dateofbirth, m.total, m.result from students s left join marks m on **s.roll** = m.roll

**Note:** The LEFT JOIN keyword returns all records from the left table (Customers), even if there are no matches in the right table (Orders) even no match found mysql replace required values with NULL keyword.

# MySQL RIGHT JOIN Keyword

## MySQL RIGHT JOIN Keyword

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records (if any) from the left table (table1).



### RIGHT JOIN Syntax

SELECT column\_name(s) FROM table1 RIGHT JOIN table2ON table1.column\_name = table2.column\_name;

SELECT s.roll, s.fname, s.lname, s.city, s.email, s.phone, s.gender, s.dateofbirth, m.total, m.result from students s inner join marks m on s.roll = m.roll;

SELECT s.roll, s.fname, s.lname, s.city, s.email, s.phone, s.gender, s.dateofbirth, m.total, m.result from students s right join marks m on s.roll = m.roll;

**Note:** The RIGHT JOIN keyword returns all records from the right table (Employees), even if there are no matches in the left table (Orders) even no match found mysql replace required values with NULL keyword.

.

# MySQL CROSS JOIN Keyword

## SQL CROSS JOIN Keyword

The CROSS JOIN keyword returns all records from both tables (table1 and table2).



### CROSS JOIN Syntax

SELECT column\_name(s) FROM table1 CROSS JOIN table2;

**Note:** CROSS JOIN can potentially **return very large result-sets**!

SELECT \* FROM students CROSS join marks

**Note:** The CROSS JOIN keyword returns all matching records from both tables whether the other table matches or not. So, if there are rows in "Customers" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Customers", those rows will be listed as well.

If you add a WHERE clause (if table1 and table2 has a relationship), the CROSS JOIN will produce the same result as the **INNER JOIN** clause:

SELECT \* FROM students CROSS join marks WHERE students.roll = marks.roll;

# MySQL Self Join

## MySQL Self Join

A self join is a regular join, but the table is joined with itself.

### Self Join Syntax

SELECT column\_name(s) FROM table1 T1, table1 T2 WHERE condition;

T1 and T2 are different table aliases for the same table.

SELECT s1.roll, s1.fname, s1.lname, s1.city, s1.email, s1.phone FROM students s1, students s2 WHERE s1.roll <> s2.roll and s1.city = 'Rajkot';

SELECT s1.roll, s1.fname, s1.lname, s1.city, s2.roll, s2.fname, s2.lname, s2.city FROM students s1, students s2 WHERE s1.roll <> s2.roll and s1.city = s2.city;

# MySQL UNION Operator

## The MySQL UNION Operator

The UNION operator is used to combine the result-set of two or more SELECT statements.

* Every SELECT statement within UNION must have the same number of columns
* The columns must also have similar data types
* The columns in every SELECT statement must also be in the same order

### UNION Syntax

SELECT column\_name(s) FROM table1  
UNION  
SELECT column\_name(s) FROM table2;

### UNION ALL Syntax

The UNION operator selects **only distinct values by default**. To allow duplicate values, use UNION ALL:

SELECT column\_name(s) FROM table1  
UNION ALL  
SELECT column\_name(s) FROM table2;

create table students1 as SELECT \* from students

SELECT \* FROM students

UNION

SELECT \* FROM students1

SELECT \* FROM students

UNION ALL

SELECT \* FROM students1

SELECT roll, fname, lname, city FROM students

UNION ALL

SELECT roll, fname, lname, city FROM students1;

SELECT roll, fname, lname, city FROM students

UNION ALL

SELECT roll, fname, lname FROM students1;

#1222 - The used SELECT statements have a different number of columns

SELECT roll, fname, lname, city FROM students WHERE city = 'Rajkot'

UNION ALL

SELECT roll, fname, lname, city FROM students1 WHERE city = 'Rajkot';

# MySQL GROUP BY Statement

## The MySQL GROUP BY Statement

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

SELECT DISTINCT city FROM students

### GROUP BY Syntax

SELECT column\_name(s) FROM table\_name WHERE condition GROUP BY column\_name(s)ORDER BY column\_name(s);

SELECT city, count(roll) from students GROUP by(city)

SELECT city, count(roll) from students GROUP by(city) ORDER by (COUNT(roll));

SELECT city, count(roll) from students GROUP by(city) ORDER by (COUNT(roll)) desc;

## GROUP BY With JOIN Example

SELECT students.roll, students.fname, students.lname, students.city, students.gender, students.phone, students.email, students.admissiondata, marks.total, marks.result from students inner join marks on students.roll = marks.roll

SELECT students.roll, students.fname, students.lname, students.city, students.gender, students.phone, students.email, students.admissiondata, marks.total, marks.result, count(marks.roll) as "Exam Attempt" from students inner join marks on students.roll = marks.roll GROUP by (marks.roll);

# MySQL HAVING Clause

## The MySQL HAVING Clause

The HAVING clause was added to SQL because the WHERE keyword **cannot** be used with aggregate functions.

SELECT students.roll, students.fname, students.lname, students.city, students.gender, students.phone, students.email, students.admissiondata, marks.total, marks.result, count(marks.roll) as "Exam Attempt" from students inner join marks on students.roll = marks.roll **GROUP by** (marks.roll) **where** count(marks.roll) > 2;

SELECT students.roll, students.fname, students.lname, students.city, students.gender, students.phone, students.email, students.admissiondata, marks.total, marks.result, count(marks.roll) as "Exam Attempt" from students inner join marks on students.roll = marks.roll GROUP by (marks.roll) having count(marks.roll) > 2;

SELECT city, COUNT(roll) from students GROUP by (city);

SELECT city, COUNT(roll) from students GROUP by (city) HAVING COUNT(roll) > 3;

SELECT city, COUNT(roll) from students GROUP by (city) HAVING COUNT(roll) > 3 ORDER by (COUNT(roll));

# MySQL EXISTS Operator

## The MySQL EXISTS Operator

The EXISTS operator is used to test for the existence of any record in a subquery.

The EXISTS operator returns TRUE if the subquery returns one or more records.

### EXISTS Syntax

SELECT column\_name(s) FROM table\_name WHERE EXISTS (SELECT column\_name FROM table\_name WHERE condition);

SELECT roll from marks WHERE result = 'pass';

SELECT \* from students WHERE EXISTS (SELECT roll from marks WHERE result = 'pass' and students.roll = marks.roll);

SELECT \* from students WHERE not EXISTS (SELECT roll from marks WHERE result = 'pass' and students.roll = marks.roll);

# MySQL ANY and ALL Operators

## The MySQL ANY and ALL Operators

The ANY and ALL operators allow you to perform a comparison between a single column value and a range of other values.

## The ANY Operator

The ANY operator:

* returns a boolean value as a result
* returns TRUE if ANY of the subquery values meet the condition

ANY means that the condition will be true if the operation is true for any of the values in the range.

### ANY Syntax

SELECT column\_name(s) FROM table\_name WHERE column\_name operator ANY (SELECT column\_name  FROM table\_name  WHERE condition);

**Note:** The operator must be a standard comparison operator (=, <>, !=, >, >=, <, or <=).

SELECT \* from students WHERE roll = any (SELECT roll FROM marks WHERE result = 'pass');

## The ALL Operator

The ALL operator:

* returns a boolean value as a result
* returns TRUE if ALL of the subquery values meet the condition
* is used with SELECT, WHERE and HAVING statements

ALL means that the condition will be true only if the operation is true for all values in the range.

### ALL Syntax With SELECT

SELECT ALL column\_name(s) FROM table\_name WHERE condition;

### ALL Syntax With WHERE or HAVING

SELECT column\_name(s) FROM table\_name WHERE column\_name operator ALL   (SELECT column\_name FROM table\_name  WHERE condition);

**Note:** The operator must be a standard comparison operator (=, <>, !=, >, >=, <, or <=).

This will of course return FALSE because the Quantity column has many different values (not only the value of pass):

SELECT \* from students WHERE roll = all (SELECT roll FROM marks WHERE result = 'pass');

# MySQL INSERT INTO SELECT Statement

## The MySQL INSERT INTO SELECT Statement

The INSERT INTO SELECT statement copies data from one table and inserts it into another table.

The INSERT INTO SELECT statement requires that the data types in source and target tables matches.

**Note:** The existing records in the target table are unaffected.

### INSERT INTO SELECT Syntax

Copy all columns from one table to another table:

INSERT INTO table2 SELECT \* FROM table1WHERE condition;

INSERT into students1 SELECT \* FROM students

INSERT into students1 SELECT \* FROM students WHERE City = 'rajkot'

Copy only some columns from one table into another table:

INSERT INTO *table2* (*column1*, *column2*, *column3*, ...) SELECT *column1*, *column2*, *column3*, ... FROM *table1* WHERE *condition*;

INSERT into students1 (Fname, lname, city) SELECT fname, lname, city from students

# MySQL CASE Statement

## The MySQL CASE Statement

The CASE statement goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.

If there is no ELSE part and no conditions are true, it returns NULL.

## CASE Syntax

CASE  
    WHEN condition1 THEN result1  
    WHEN condition2 THEN result2  
    WHEN conditionN THEN resultN  
    ELSE result  
END;

SELECT roll, fname, lname, city, email, phone, gender, dateofbirth from students

SELECT roll, fname, lname, city, email, phone, gender, case

WHEN gender = 'male' then 'Blue'

WHEN gender = 'female' then 'Pink'

else 'Black'

end as "Bag Color",

dateofbirth from students

SELECT roll, fname, lname, city, case

WHEN city = 'Rajkot' THEN 'Home Town'

WHEN city = 'Ahamdabad' THEN 'Far Town'

WHEN city = 'Gandhinagar' THEN 'Far Town'

WHEN city = 'Bhuj' THEN 'Much Far Town'

WHEN city = 'Baroda' THEN 'Too Much Far Town'

WHEN city = 'Surat' THEN 'Remote Town'

end as "Distance",email, phone, gender,dateofbirth from students;

# MySQL NULL Functions

## MySQL IFNULL() and COALESCE() Functions

SELECT roll, absents, presents, (absents +presents) as "Total Days" FROM attendance GROUP by (roll);

## MySQL IFNULL() Function

The MySQL [IFNULL()](https://www.w3schools.com/mysql/func_mysql_ifnull.asp) function lets you return an alternative value if an expression is NULL.

The example below returns 0 if the value is NULL:

SELECT roll, absents, presents, (ifnull(absents, 0) + ifnull(presents, 0)) as "Total Days" FROM attendance GROUP by (roll);

## MySQL COALESCE() Function

SELECT roll, absents, presents, (COALESCE(absents, 0) + COALESCE(presents, 0)) as "Total Days" FROM attendance GROUP by (roll);