**Download Xampp**

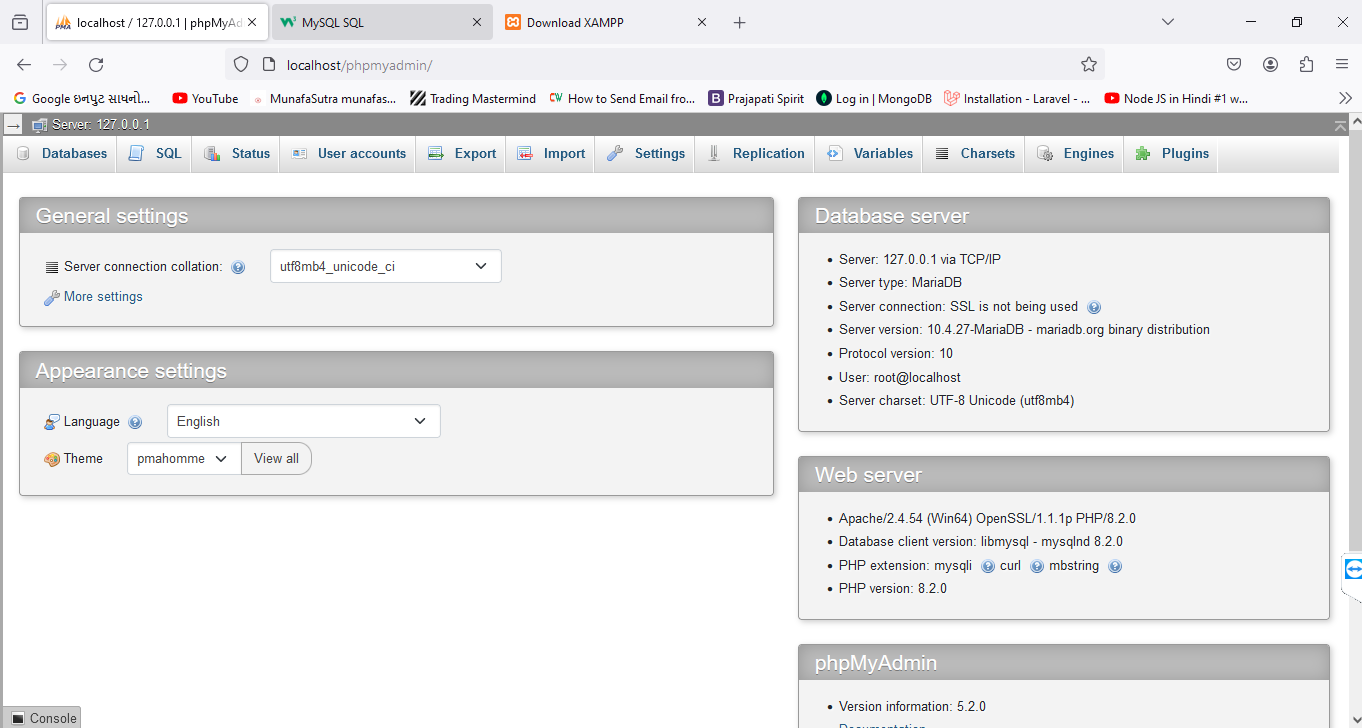
Open [h**ttps://www.apachefriends.org/download.html**](https://www.apachefriends.org/download.html) and download latest version for your operating system.

Install Downloaded software

Open Xampp Admin panel and start apache and mysql service

Close the Xampp Admin panel and open browser

In browser addressbar enter url : localhost/phpMyAdmin

****

**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

# MySQL RDBMS

## 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 5 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.

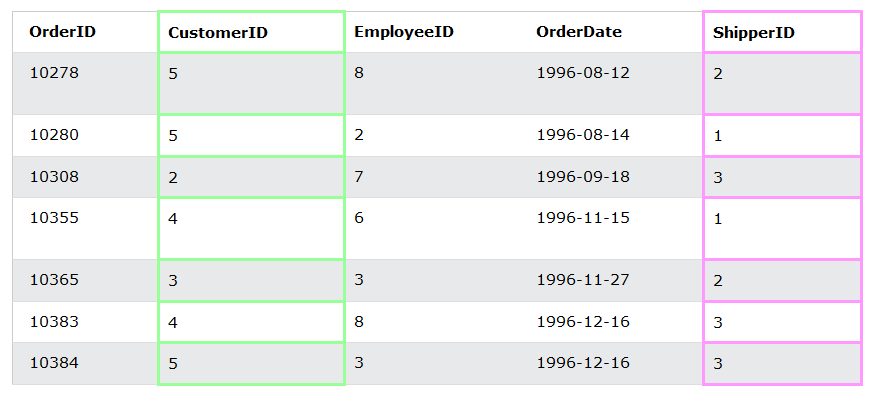
Look at the following three tables "Customers", "Orders", and "Shippers" from the Northwind database:

### Customers Table



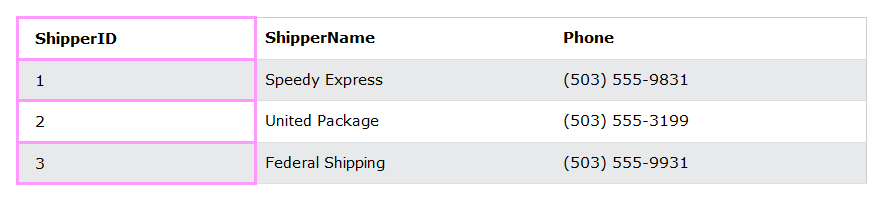
The relationship between the "Customers" table and the "Orders" table is the CustomerID column:

### Orders Table



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

### Shippers Table



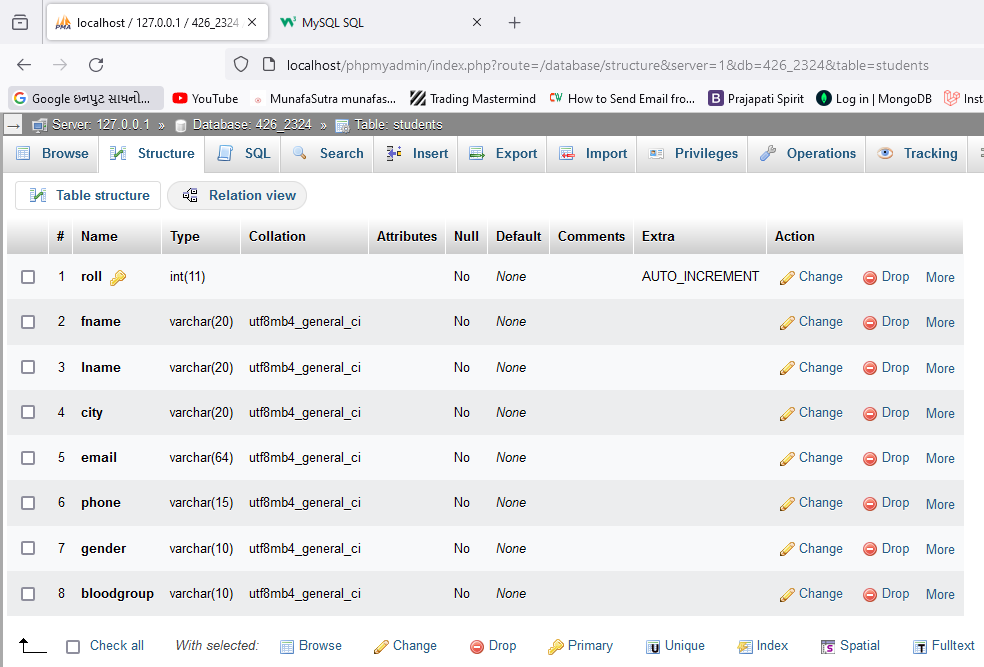
Before begin with MYsql Tutorial you need to create new Database and table with some of the data.

Open database section from localhost/phpMyAdmin -> then select create database section and enter your required database name then press create.

After successfully creation of new database, you will be redirected to newly created database page.

Create new table as following

Enter table name -> students -> create 8 columns



Create above table and add some of the data as below



# 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:

### Example

SELECT \* FROM Customers;

SELECT \* from students;

## 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.

select \* from students;

SELECT \* from students;

## 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

# 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.

SELECT roll, fname, lname from students

SELECT roll, fname, lname, city from students;

If you want to select all the fields available in the table, use the following syntax:

SELECT \* FROM *table\_name*;

SELECT \* FROM students;

## SELECT \* Example

The following SQL statement selects ALL the columns from the "Customers" table:

### Example

SELECT \* FROM Customers;

## 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 DISTINCT city from students

## SELECT Example Without DISTINCT

The following SQL statement selects all (including the duplicates) values from the "Country" column in the "Customers" table:

SELECT city from students;

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;

SELECT \* FROM students WHERE city = 'rajkot'

SELECT \* FROM students WHERE roll > 5

SELECT \* FROM students WHERE roll = 5;

SELECT \* FROM students WHERE roll <> 5;

SELECT \* FROM students WHERE not roll = 5;

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

## Text Fields vs. Numeric Fields

SQL requires single quotes around text values (most database systems will also allow double quotes).

However, numeric fields should not be enclosed in quotes:

SELECT \* from students WHERE gender = 'male'

SELECT \* from students WHERE gender = male

SELECT \* from students WHERE roll = 8

SELECT \* from students WHERE roll = '8';

# 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 ...;

### 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;

SELECT roll, fname, lname, city from students WHERE roll = 1 and city = 'rajkot'

SELECT roll, fname, lname, city from students WHERE roll = 1 and city = 'surat';

SELECT roll, fname, lname, city from students WHERE city = 'rajkot' or city = 'surat';

SELECT roll, fname, lname, city from students WHERE not (city = 'rajkot' or city = 'surat');

SELECT roll, fname, lname, city from students WHERE not city = 'rajkot' ;

## 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 = 'surat')

# 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;

SELECT \* from students ORDER by fname desc;

## ORDER BY Several Columns Example

SELECT \* from students ORDER by fname desc, lname 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 Example

The following SQL statement inserts a new record in the "Students" table:

INSERT into students (fname, lname, city, email, phone, gender, bloodgroup) VALUES ('patel', 'riya', 'ahamdabad', 'riya@gmail.com', '9988999900', 'female', 'o positive');

1 row inserted.   
Inserted row id: **11** (Query took 0.0032 seconds.)

**Did you notice that we did not insert any number into the Roll field?**  
The **roll** 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 into students (fname, lname, city, email, phone, gender, bloodgroup) VALUES ('patel', 'siya', 'ahamdabad', 'siya@gmail.com', '9988999900', 'female', 'o positive');

## Insert Data Only in Specified Columns

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

INSERT into students (fname, lname, city, gender, bloodgroup) VALUES ('patel', 'siya', 'ahamdabad', 'female', 'o positive');

INSERT into students VALUES ('patel', 'jiya', 'jiya@gmail.com', '9988998899','ahamdabad', 'female', 'o positive');

#1136 - Column count doesn't match value count at row 1

INSERT into students VALUES (null, 'patel', 'jiya', 'jiya@gmail.com', '9988998899','ahamdabad', 'female', 'o positive');

# 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!

SELECT \* from students WHERE email is null

SELECT \* from students WHERE email = '';

## 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;

### IS NOT NULL Syntax

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

INSERT into students (fname, lname, city, gender, bloodgroup) VALUES ('patel', 'siya', 'ahamdabad', 'female', 'o positive');

INSERT into students (fname, lname, city, gender, bloodgroup) VALUES ('pandya', 'priya', 'ahamdabad', 'female', 'o positive');

SELECT \* from students WHERE email is null

SELECT \* from students WHERE email is not null;

**Tip:** Always use IS NULL to look for NULL values.

## The IS NOT NULL Operator

The IS NOT NULL operator is used to test for non-empty values (NOT NULL values).

# 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, ...  
WHERE condition;

**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!

UPDATE students set city = 'Gandhinagar' WHERE roll = 13

UPDATE students set city = 'Bhuj' WHERE roll > 13

## UPDATE Multiple Records

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

UPDATE students set email = 'demo@gmail.com', phone = '0000000000' where email is null

## Update Warning!

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

UPDATE students set bloodgroup = 'O Negative'

UPDATE students SET bloodgroup = 'B +ve' WHERE roll >= 12

# 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 LIMIT 5

SELECT \* FROM students WHERE city = 'rajkot' LIMIT 5;

What if we want to select records 4 - 6 (inclusive)?

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

The SQL query below says "return only 3 records, start on record 4 (OFFSET 3)":

SELECT \* FROM students LIMIT 5 OFFSET 5;

SELECT \* FROM students LIMIT 5 OFFSET 10;

SELECT \* FROM students LIMIT 10, 5;

SELECT \* FROM students LIMIT 0, 5;

SELECT \* FROM students LIMIT 20, 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(roll) FROM students

SELECT min(roll) FROM students;

SELECT max(fees) FROM students

SELECT min(fees) 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 where fees > 10000;

SELECT COUNT(roll), sum(fees) from students where fees > 10000;

SELECT COUNT(roll), sum(fees) from students where city = 'rajkot';

SELECT COUNT(roll), sum(fees), avg(fees) from students where city = 'rajkot';

# 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

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 not like '\_a%';

### 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.

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" |

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

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

# 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!

SELECT \* from students WHERE city like 'raj%'

# 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 = 'baroda' or city = 'jetpur'

## IN Operator Examples

SELECT \* from students WHERE city in ('Rajkot', 'baroda', 'jetpur');

SELECT \* from students WHERE city not in ('Rajkot', 'baroda', 'jetpur');

# 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 >= 5 and roll <= 10

SELECT \* from students WHERE roll between 1 and 5

SELECT \* from students WHERE roll between 5 and 10;

SELECT \* from students WHERE fname between 'aryan' and 'nishit';

SELECT \* from students WHERE fees between 10000 and 15000;

SELECT \* from students WHERE fees not between 10000 and 15000;

## BETWEEN with IN Example

SELECT \* from students WHERE fees not between 10000 and 15000 and roll in (1,2,3,4,5);

SELECT \* from students WHERE fees not between 10000 and 15000 and roll <= 5;

## BETWEEN Text Values Example

SELECT \* from students WHERE fname between 'aryan' and 'nishit';

## BETWEEN Dates Example

SELECT \* from students WHERE admissiondate BETWEEN '2020-01-01' and '2022-12-31'

SELECT \* from students WHERE admissiondate not BETWEEN '2020-01-01' and '2022-12-31';

# 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!

## SQL DELETE Example

DELETE from students WHERE city = 'Bhuj'

DELETE from students WHERE bloodgroup = 'B +ve'

## 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 students

Truncate table students

Delete all the data from table and reset all the auto increment from table

TRUNCATE TABLE students

# 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;

SELECT roll, fname, lname FROM students;

SELECT roll as "Roll Number", fname as "First Name", lname as "Last Name" FROM students;

### Alias Table Syntax

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

**Note:** Single or double quotation marks are required if the alias name contains spaces:

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

SELECT concat\_ws(" - ",roll, fname, lname, city, email, gender, phone ) from students;

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

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

# 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.

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

SELECT students.roll, students.fname, students.lname, attendance.absents, attendance.presents from **students inner join attendance** on students.roll = attendance.roll

SELECT s.roll, s.fname, s.lname, a.absents, a.presents from students as s inner join attendance as a on s.roll = a.roll;

SELECT s.roll, s.fname, s.lname, a.absents, a.presents from students s inner join attendance a on s.roll = a.roll;

## 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

Top of Form

## 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.city, students.email, students.phone, students.gender, attendance.absents, attendance.presents from students INNER join attendance on students.roll = attendance.roll

SELECT s.roll, s.fname, s.lname, s.city, s.email, s.phone, s.gender, s.admissiondate, a.absents, a.presents from students s **INNER join** attendance a on s.roll = a.roll

Bottom of Form

## JOIN Three Tables

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

**Note:** The LEFT JOIN keyword returns all records from the left table (Customers), even if there are no matches in the right table (Orders). If not data match in right table empty fields filled with NULL value.

SELECT students.roll, students.fname, students.lname, students.city, students.phone, students.email, attendance.absents, attendance.presents from students **LEFT JOIN** attendance on **students.roll** = attendance.roll

# 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;

**Note:** The RIGHT JOIN keyword returns all records from the right table (Employees), even if there are no matches in the left table (Orders). If not data match in left table empty fields filled with NULL value.

SELECT students.roll, students.fname, students.lname, students.city, students.phone, students.email, attendance.absents, attendance.presents from students **RIGHT JOIN** attendance on students.roll = **attendance.roll**;

# 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!

**Note:** The CROSS JOIN keyword returns all matching records from both tables whether the other table matches or not.

SELECT \* from students CROSS JOIN attendance;

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 attendance WHERE students.roll = attendance.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, concat\_ws("-", s1.fname, s1.lname) as "Student From Table 1", s2.roll, concat\_ws("-", s2.fname, s2.lname) as "Student From Table 2" from students s1, students s2 WHERE s1.roll <> s2.roll and s1.city = s2.city ORDER by s1.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

CREATE table students1 as SELECT \* from students WHERE city = 'Rajkot'

### 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;

SELECT \* from students

UNION

SELECT \* from students1

SELECT \* from students

UNION ALL

SELECT \* from students1;

## SQL UNION With WHERE

SELECT \* from students WHERE city = 'Rajkot'

UNION ALL

SELECT \* 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 gender, COUNT(gender) as "Total Students" from students GROUP by gender

SELECT city, COUNT(city) as "Total Students From " from students GROUP by(city)

SELECT city, COUNT(city) as "Total Students From " from students GROUP by(city) ORDER by (COUNT(city));

SELECT city, COUNT(city) as "Total Students From " from students GROUP by(city) ORDER by (COUNT(city)) desc;

### GROUP BY Syntax

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

## GROUP BY With JOIN Example

SELECT students.roll, students.fname, students.lname, COUNT(fees.roll) as "Times", sum(fees.amount) as "Total Payment" from students INNER join fees on students.roll = fees.roll GROUP by (fees.roll);

SELECT students.roll, students.fname, students.lname, COUNT(fees.roll) as "Times", sum(fees.amount) as "Total Payment" from students INNER join fees on students.roll = fees.roll GROUP by (fees.roll) ORDER by sum(fees.amount) desc;

# 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 city, count(roll) from students GROUP BY(city) HAVING COUNT(roll) >= 5;

SELECT city, count(roll) from students GROUP BY(city) HAVING COUNT(roll) >= 2;

SELECT students.roll, students.fname, students.lname, COUNT(fees.roll) as "Total Times", sum(fees.amount) as "Total Amount" FROM students INNER JOIN fees on students.roll = fees.roll GROUP by (fees.roll) HAVING sum(fees.amount) >= 20000 ORDER by (sum(fees.amount)) desc;

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

SELECT students.roll, students.fname, students.lname, COUNT(fees.roll) as "Total Times", sum(fees.amount) as "Total Amount" FROM students INNER JOIN fees on students.roll = fees.roll WHERE students.roll = 1 or students.roll = 2 GROUP by (fees.roll) HAVING sum(fees.amount) >= 20000 ORDER by (sum(fees.amount)) desc;

## 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 \* from students WHERE 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);

SELECT \* from students

SELECT \* from marks

SELECT \* from marks WHERE result <> 'pass';

SELECT roll from marks WHERE result <> 'pass';

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

SELECT \* from students WHERE **roll** = any (SELECT **roll** from attendance WHERE presents > 150);

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

## 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 <=).

The following SQL statement lists the ProductName if ALL the records in the OrderDetails table has Quantity equal to 10. This will of course return FALSE because the Quantity column has many different values (not only the value of 10):

SELECT roll, fname, lname from students where roll = all (SELECT roll from attendance WHERE presents > 100 AND students.roll = attendance.roll);

SELECT roll,fname,lname from students where roll = all (SELECT roll from attendance WHERE presents > 150 AND students.roll = attendance.roll);