# SQL Là Gì?

SQL là viết tắt của *Structured Query Language* - ngôn ngữ khai báo (declarative language) để lưu trữ, thao tác và truy xuất dữ liệu được lưu trữ trong cơ sở dữ liệu quan hệ (relational database). Đây là ngôn ngữ truy vấn phổ biến nhất dành cho cơ sở dữ liệu.

Một số từ viết tắt thông dụng:

* **SQL**: Structured Query Language
* **DB**:Database
* **RDBMS**: Relational Database Management System

# Kiểu Dữ Liệu (Data Types)

Mỗi cột (column) trong bảng cơ sở dữ liệu (database table) bắt buộc phải có **tên** và **kiểu dữ liệu**.

Nhờ kiểu dữ liệu, SQL hiểu được loại dữ liệu nào được mong đợi trong mỗi cột và từ đó, giúp SQL biết cách để tương tác với dữ liệu được lưu trữ một cách đúng đắn.

**Lưu ý**: Các hệ cơ sở dữ liệu khác nhau (Database Management System) có cách đặt tên khác nhau cho các kiểu dữ liệu. Và ngay cả khi có tên giống nhau, kích thước và các chi tiết khác cũng có thể khác nhau.

Dưới đây là danh sách các kiểu dữ liệu được MySQL, SQL Server và MS Access SQL quy định:

<https://www.w3schools.com/sql/sql_datatypes.asp>

# Lệnh Thao Tác Với Database

## CREATE DATABASE

Câu lệnh CREATE DATABASE được sử dụng để tạo cơ sở dữ liệu SQL mới.

**Cú pháp:**

CREATE DATABASE Database-Name;

Lưu ý:

* Tên cơ sở dữ liệu phải là duy nhất trong RDBMS.
* Đảm bảo rằng bạn có quyền quản trị viên trước khi tạo bất kỳ cơ sở dữ liệu nào. Vì nếu bạn không có quyền quản trị, bạn không được phép tạo cơ sở dữ liệu mới được.

**Ví dụ:**

Nếu bạn muốn tạo một cơ sở dữ liệu mới có tên TestDB, chạy lệnh sau:

SQL> CREATE DATABASE TestDB;

Sau khi TestDB được tạo, bạn có thể kiểm tra nó trong danh sách CSDL như sau:

SQL> SHOW DATABASES;

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

| Database      |

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

| Armood        |

| TutorialPoint |

| Orig          |

| TestDB        | 🡪 Điều này khẳng định TestDB đã được tạo thành công

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

## DROP DATABASE

Câu lệnh DROP DATABASE được sử dụng để xóa cơ sở dữ liệu hiện có trong hệ CSDL.

**Cú pháp:**

DROP DATABASE Database-Name;

Lưu ý: Đảm bảo rằng bạn có quyền quản trị viên trước khi xóa bất kỳ cơ sở dữ liệu nào. Vì nếu bạn không có quyền quản trị, bạn không được phép xóa cơ sở dữ liệu được.

**Ví dụ:**

Nếu bạn muốn xóa cơ sở dữ liệu có tên TestDB, chạy lệnh sau:

DROP DATABASE TestDB;

Sau khi TestDB được xóa, bạn có thể kiểm tra nó trong danh sách CSDL như sau:

SQL> SHOW DATABASES;

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

| Database      |

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

| Armood        |

| TutorialPoint |

| Orig          |

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

## USE

Khi bạn có nhiều cơ sở dữ liệu trong hệ thống CSDL của mình, SQL sẽ không biết được bạn muốn thao tác với CSDL nào. Vì vậy, bận cần chọn CSDL mà bạn muốn thao tác.

Câu lệnh USE được sử dụng để chọn bất kỳ CSDL hiện có nào trong hệ CSDL.

**Cú pháp:**

USE Database-Name;

**Ví dụ:**

Đầu tiên kiểm tra các CSDL có sẵn với lệnh SHOW DATABASES như sau:

SQL> SHOW DATABASES;

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

| Database      |

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

| Armood        |

| TutorialPoint |

| Orig          |

| TestDB        |

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

Bây giờ, nếu bạn muốn làm việc với cơ sở dữ liệu TestDB, chạy lệnh:

SQL> USE TestDB;

# Lệnh Thao Tác Với Table

## CREATE TABLE

Để tạo bảng trong CSDL, chúng ta cần đặt tên cho bảng và xác định các cột của nó, cũng như các kiểu dữ liệu cho mỗi cột.

Câu lệnh CREATE TABLE được sử dụng để tạo một bảng mới.

**Cú pháp:**

CREATE TABLE Table-Name(

column1 datatype,

column2 datatype,

column3 datatype,

.....

columnN datatype,

PRIMARY KEY(one or more columns)

);

**Ví dụ:**

Ví dụ dưới đây tạo bảng Customers với primary key là ID. Thuộc tính ràng buộc (constraint) là NOT NULL cho ID, Name và Age. Và NULL cho Address và Salary.

SQL> CREATE TABLE Customers(

ID INT NOT NULL,

Name VARCHAR (20) NOT NULL,

Age INT NOT NULL,

Address CHAR (25),

Salary DECIMAL (18, 2),

PRIMARY KEY (ID)

);

Bạn có thể xác minh xem bảng của bạn đã được tạo thành công hay chưa bằng cách xem thông báo được hiển thị bởi DBMS sau mỗi lần thực thi một câu lệnh nào đó. Hoặc có thể sử dụng câu lệnh DESC như sau:

SQL> DESC Customers;

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

| Field | Type | Null | Key | Default | Extra |

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

| ID | int(11) | NO | PRI | | |

| Name | varchar(20) | NO | | | |

| Age | int(11) | NO | | | |

| Address | char(25) | YES | | NULL | |

| Salary | decimal(18,2) | YES | | NULL | |

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

## DROP TABLE

Câu lệnh DROP TABLE được sử dụng để xóa bảng và tất cả dữ liệu của nó (values, indexes, triggers, constraints và permission specifications).

**Cú pháp:**

DROP TABLE table-name;

**Ví dụ:**

Giả sử CSDL của bạn đã có bảng Customers và sau đó chúng tôi xóa nó khỏi CSDL như sau:

SQL> DESC Customers;

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

| Field | Type | Null | Key | Default | Extra |

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

| ID | int(11) | NO | PRI | | |

| Name | varchar(20) | NO | | | |

| Age | int(11) | NO | | | |

| Address | char(25) | YES | | NULL | |

| Salary | decimal(18,2) | YES | | NULL | |

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

Điều này có nghĩa là bảng Customers có sẵn trong CSDL.

Bây giờ, hãy xóa nó:

SQL> DROP TABLE Customers;

Sau khi xóa, nếu bạn thử lệnh DESC, bạn sẽ nhận được lỗi sau:

SQL> DESC Customers;

ERROR 1146 (42S02): Table 'TEST.Customers' doesn't exist

Điều này có nghĩa là bảng Customers trong CSDL TEST không còn tồn tại (do nó đã bị xóa rồi mà).

## SELECT

### Không điều kiện

Câu lệnh SELECT được sử dụng để tìm kiếm/truy xuất dữ liệu từ bảng CSDL. Kết quả trả được thể hiện dưới dạng một bảng kết quả.

**Cú pháp:**

SELECT column1, column2, ... columnN FROM table-name;

Mẹo: Nếu bạn muốn tìm/truy xuất tất cả các cột (column) và trường (field) trong bảng, chỉ việc thay thế tên cột bằng dấu \*:

SELECT \* FROM table-name;

**Ví dụ:**

Giả sự bạn có bảng Customers như sau:

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

| ID | Name | Age | Address | Salary |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | Kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

Để truy xuất tất cả trường trong cột ID, Name và Salary, chạy lệnh:

SQL> SELECT ID, Name, Salary FROM Customers;

Kết quả:

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

| ID | Name | Salary |

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

| 1 | Ramesh | 2000.00 |

| 2 | Khilan | 1500.00 |

| 3 | Kaushik | 2000.00 |

| 4 | Chaitali | 6500.00 |

| 5 | Hardik | 8500.00 |

| 6 | Komal | 4500.00 |

| 7 | Muffy | 10000.00 |

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

Nhưng nếu bạn chỉ muốn truy xuất tất cả trường trong tất cả cột, chạy lệnh:

SQL> SELECT \* FROM Customers;

This would produce the result as shown below:

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

| ID | Name | Age | Address | Salary |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | Kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

### Có điều kiện

Để chỉ định điều kiện khi tìm/truy xuất dữ liệu từ bảng, bạn thêm từ khóa WHERE sau mỗi câu lệnh SELECT. Nó rất hữu ích khi bạn chỉ muốn tìm/truy xuất một số trường cụ thể từ bảng, chứ không phải là tất cả các trường:

**Cú pháp:**

SELECT column1, column2, columnN

FROM table-name

WHERE [condition]

Để chỉ định điều kiện, sử dụng các toán tử so sánh như >, <, =, LIKE, NOT, ...

**Ví dụ:**

Giả sử bạn có bảng Customers như sau:

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

| ID | Name | Age | Address | Salary |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | Kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

Để tìm/truy xuất các trường từ các cột ID, Name và Salary với điều kiện là Salary phải lớn hơn hoặc bằng 2000, chạy lệnh sau:

SQL> SELECT ID, Name, Salary

FROM Customers

WHERE Salary > 2000;

Kết quả:

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

| ID | Name | Salary |

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

| 4 | Chaitali | 6500.00 |

| 5 | Hardik | 8500.00 |

| 6 | Komal | 4500.00 |

| 7 | Muffy | 10000.00 |

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

### SELECT INTO

Câu lệnh SELECT INTO được sử dụng để **sao chép dữ liệu từ bảng này vào một bảng khác**. Tên và kiểu dữ liệu của cột trong bảng mới sẽ giống với bảng cũ.

**Cú pháp:**

SELECT column1, column2, column3, ...

INTO new-table [IN external-db]

FROM old-table

WHERE condition;

**Ví dụ:**

Câu lệnh sau tạo một bản sao lưu của Customers:

SQL> SELECT \* INTO CustomersBk

FROM Customers

Câu lệnh sau sử dụng mệnh đề IN để sao chép bảng vào một bảng mới được lưu trong CSDL khác:

SQL> SELECT \* INTO CustomersBk IN 'Backup.mdb'

FROM Customers;

Câu lệnh sau chỉ sao chép cột CustomerName và ContactName vào bảng mới:

SQL> SELECT CustomerName, ContactName INTO CustomersBk

FROM Customers;

Câu lệnh sau chỉ sao chép khách hàng là German vào bảng mới:

SQL> SELECT \* INTO CustomersGermany

FROM Customers

WHERE Country = 'Germany';

### SELECT DISTINCT

Từ khóa DISTINCT được sử dụng cùng với lệnh SELECT để chỉ **tìm/truy xuất các trường là duy nhất**. Tức là tự động bỏ qua các trường có giá trị trùng lặp.

**Cú pháp:**

SELECT DISTINCT column1, column2, ... columnN

FROM table-name

WHERE [condition]

**Ví dụ:**

Giả sử bạn có bảng Customers như sau:

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

| ID | Name | Age | Address | Salary |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | Kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

Đầu tiên, chạy lệnh SELECT để tìm/truy xuất tất cả trường từ cột Salary:

SQL> SELECT Salary FROM Customers;

Kết quả. Để ý giá trị Salary là 2000 xuất hiện đến 2 lần:

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

| Salary |

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

| 1500.00 |

| 2000.00 |

| 2000.00 |

| 4500.00 |

| 6500.00 |

| 8500.00 |

| 10000.00 |

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

Bây giờ, chạy lệnh SELECT DISTINCT và bạn sẽ thấy sự khác biệt:

SQL> SELECT DISTINCT Salary FROM Customers;

Lúc này, bạn sẽ thấy những giá trị Salary 2000 chỉ còn xuất hiện đúng 1 lần:

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

| Salary |

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

| 1500.00 |

| 2000.00 |

| 4500.00 |

| 6500.00 |

| 8500.00 |

| 10000.00 |

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

## INSERT INTO

Câu lệnh INSERT INTO được sử dụng để thêm hàng mới vào bảng trong CSDL.

**Cú pháp:**

INSERT INTO Table-Name (column1, column2, column3, ... columnN)

VALUES (value1, value2, value3, ... valueN);

Lưu ý: Đảm bảo rằng tên cột (column1 …) và giá trị của cột (value1 …) có cùng thứ tự.

**Ví dụ:**

Câu lệnh sau tạo 3 hàng mới cho bảng Customers:

SQL>

INSERT INTO Customers (ID, Name, Age, Address, Salary)

VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );

INSERT INTO Customers (ID, Name, Age, Address, Salary)

VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );

INSERT INTO Customers (ID, Name, Age, Address, Salary)

VALUES (3, 'Kaushik', 23, 'Kota', 2000.00 );

Chạy lệnh SELECT và bạn sẽ thấy kết quả là 3 hàng mới đã được add thêm vào bảng:

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

| ID | Name | Age | Address | Salary |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | Kaushik | 23 | Kota | 2000.00 |

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

## UPDATE

Câu lệnh UPDATE được sử dụng để sửa đổi các trường đã có trong bảng.

**Cú pháp:**

UPDATE table-name

SET column1 = value1, column2 = value2...., columnN = valueN

WHERE [condition];

Bạn có thể sử dụng mệnh đề WHERE chúng với lệnh UPDATE để chỉ cập nhật các hàng mà bạn đã chọn; nếu không tất cả các hàng sẽ bị ảnh hưởng.

**Ví dụ:**

Giả sự bạn có bảng Customers như sau:

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

| ID | Name | Age | Address | Salary |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | Kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

Lệnh sau sẽ cập nhật Address cho khách hàng có ID là 6:

SQL> UPDATE Customers

SET Address = 'Pune'

WHERE ID = 6;

Kiểm tra lại bảng, bạn sẽ thấy Address của ID 6 đã chuyển sáng giá trị mới là Pune:

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

| ID | Name | Age | Address | Salary |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | Kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | Pune | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

Nếu bạn không dùng UPDATE với mệnh đề WHERE, tất cả hàng trong bảng sẽ bị ảnh hưởng:

SQL> UPDATE Customers

SET Address = 'Pune', Salary = 1000.00;

Now, Customers table would have the following records:

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

| ID | Name | Age | Address | Salary |

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

| 1 | Ramesh | 32 | Pune | 1000.00 |

| 2 | Khilan | 25 | Pune | 1000.00 |

| 3 | Kaushik | 23 | Pune | 1000.00 |

| 4 | Chaitali | 25 | Pune | 1000.00 |

| 5 | Hardik | 27 | Pune | 1000.00 |

| 6 | Komal | 22 | Pune | 1000.00 |

| 7 | Muffy | 24 | Pune | 1000.00 |

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

## DELETE

Câu lệnh DELETE được sử dụng để xóa hàng đã có khỏi bảng.

**Cú pháp:**

DELETE FROM table-name

WHERE [condition];

Bạn có thể sử dụng mệnh đề WHERE chúng với lệnh DELETE để chỉ cập nhật các hàng mà bạn đã chọn; nếu không tất cả các hàng sẽ bị ảnh hưởng.

**Ví dụ:**

Giả sự bạn có bảng Customers như sau:

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

| ID | Name | Age | Address | Salary |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | Kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

Lệnh sau sẽ xóa khách hàng có ID là 6 khỏi bảng:

SQL> DELETE FROM Customers

WHERE ID = 6;

Kiểm tra kết quả:

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

| ID | Name | Age | Address | Salary |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | Kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

Nếu bạn không dùng DELETE với mệnh đề WHERE, tất cả hàng trong bảng sẽ bị xóa. Điều đó có nghĩa là bảng của bạn chẳng còn lại gì:

SQL> DELETE FROM Customers;

Kiểm tra kết quả:

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

| ID | Name | Age | Address | Salary |

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

## JOIN

Câu lệnh JOIN được sử dụng để kết hợp các hàng từ hai hoặc nhiều bảng trong CSDL thông qua một giá trị chung.

### Ví dụ:

Consider the following two tables:

<https://www.w3schools.com/sql/sql_join.asp>

Here, it is noticeable that the join is performed in the WHERE clause. [Operators](#_30j0zll) can be used to join tables, but the most common operator is the =.

### Different Types of JOIN

* INNER JOIN − returns rows when there is a match in both tables.
* LEFT JOIN − returns all rows from the left table, even if there are no matches in the right table.
* RIGHT JOIN − returns all rows from the right table, even if there are no matches in the left table.
* FULL JOIN − returns rows when there is a match in one of the tables.
* SELF JOIN − is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in Câu lệnh statement.
* CARTESIAN JOIN − returns the Cartesian product of the sets of records from the two or more joined tables.

<https://www.w3schools.com/sql/sql_join_inner.asp>

<https://www.w3schools.com/sql/sql_join_left.asp>

<https://www.w3schools.com/sql/sql_join_right.asp>

<https://www.w3schools.com/sql/sql_join_full.asp>

<https://www.w3schools.com/sql/sql_join_self.asp>

# Operators

## Arithmetic Operators

+, -, \*, / and %

Same as C.

## Comparison Operators

=, !=, >, >=, <, <=

Same as C.

## Logical Operators

### AND

The AND operator is used to combine multiple conditions to narrow data in an SQL statement.

Consider the Customers table having the following records:

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

| ID | Name | Age | Address | Salary |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | Kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

To fetch the ID, Name and Salary fields from the Customers table for a customer with name Hardik and salary greater than 2000, chạy lệnh sau:

SQL> SELECT ID, Name, Salary

FROM Customers

WHERE Name = 'Hardik' AND Salary > 2000;

Kết quả:

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

| ID | Name | Salary |

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

| 5 | Hardik | 8500.00 |

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

### OR

The OR operator is used to combine multiple conditions to expand data in an SQL statement.

Consider the Customers table having the following records:

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

| ID | Name | Age | Address | Salary |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | Kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

To fetch the ID, Name and Salary fields from the Customers table where the salary is greater than 2000 or the age is less than 25 years, chạy lệnh sau:

SQL> SELECT ID, Name, Salary

FROM Customers

WHERE Salary > 2000 OR age < 25;

Kết quả:

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

| ID | Name | Salary |

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

| 3 | Kaushik | 2000.00 |

| 4 | Chaitali | 6500.00 |

| 5 | Hardik | 8500.00 |

| 6 | Komal | 4500.00 |

| 7 | Muffy | 10000.00 |

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

### NOT

### IN

<https://www.w3schools.com/sql/sql_in.asp>

### BETWEEN

<https://www.w3schools.com/sql/sql_between.asp>

### UNION

<https://www.w3schools.com/sql/sql_union.asp>

### EXISTS

<https://www.w3schools.com/sql/sql_exists.asp>

### ANY and ALL

<https://www.w3schools.com/sql/sql_any_all.asp>

### LIKE

<https://www.tutorialspoint.com/sql/sql-like-clause.htm>

# Constraints

Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

Constraints could be either on a column level or a table level. The column level constraints are applied only to one column, whereas the table level constraints are applied to the whole table.

Following are some of the most commonly used constraints available in SQL:

* NOT NULL − Ensures that a column cannot have NULL value.
* DEFAULT − Provides a default value for a column when none is specified.
* UNIQUE − Ensures that all values in a column are different.
* PRIMARY − Uniquely identifies each row/record in a database table.
* FOREIGN − Uniquely identifies a row/record in any of the given database table.
* CHECK − Ensures that all the values in a column satisfies certain conditions.
* INDEX − Creates and retrieves data from the database very quickly.

## Create Constraints

Constraints can be specified when a table is created with the CREATE TABLE statement (as shown [here](#_gjdgxs)) or you can use the ALTER TABLE statement to create constraints even after the table is created.

## Dropping Constraints

Any constraint that you have defined can be dropped using the ALTER TABLE command with the DROP CONSTRAINT option.

For Ví dụ:, to drop the primary key constraint in the Employees table, chạy lệnh sau:

ALTER TABLE Employees DROP CONSTRAINT Employees\_PK;

## Integrity Constraints

Integrity constraints are used to ensure accuracy and consistency of the data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity.

There are many types of integrity constraints that play a role in Referential Integrity (RI). These constraints include *Primary Key*, *Foreign Key*, *Unique Constraints* and other constraints which are mentioned above.

# Sorting and Grouping

## ORDER BY

<https://www.tutorialspoint.com/sql/sql-order-by.htm>

<https://www.tutorialspoint.com/sql/sql-sorting-results.htm>

## GROUP BY

<https://www.tutorialspoint.com/sql/sql-group-by.htm>

# Aliases

<https://www.w3schools.com/sql/sql_alias.asp>

# Wildcards

<https://www.w3schools.com/sql/sql_wildcards.asp>

# TOP, LIMIT or ROWNUM

<https://www.tutorialspoint.com/sql/sql-top-clause.htm>

# Functions

## MIN() and MAX()

<https://www.w3schools.com/sql/sql_min_max.asp>

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

<https://www.w3schools.com/sql/sql_count_avg_sum.asp>