

Tài liệu sử dụng Cloud Query

Tuần viết tài liệu

- [Chương 1. KIỂU DỮ LIỆU](#)
 - [1.1. Kiểu dữ liệu dạng số](#)
 - [1.2. Kiểu dữ liệu ngày tháng](#)
 - [1.3. Kiểu dữ liệu nhị phân](#)
 - [1.4. Kiểu dữ liệu dạng chuỗi](#)
- [Chương 2. CÚ PHÁP TRUY VẤN](#)
 - [2.1. SELECT](#)
 - [2.2. FROM](#)
 - [2.3. JOINS](#)
 - [2.4. WHERE](#)
 - [2.5. GROUP BY](#)
 - [2.6. ORDER BY](#)
 - [2.7. LIMIT](#)
- [Chương 3. BIỂU THỨC](#)
 - [3.1. CASE](#)
 - [3.2. So sánh](#)
 - [3.3. Toán tử IN](#)
 - [3.4. Toán tử logic](#)
- [Chương 4. CÁC HÀM CHỨC NĂNG](#)
 - [4.1. Hàm thời gian](#)

- [4.2. Hàm toán học](#)
- [4.3. Hàm tổng hợp](#)
- [4.4. Hàm chuyển đổi](#)
- [4.5. Hàm chuỗi](#)

Chương 1. KIỂU DỮ LIỆU

1.1. Kiểu dữ liệu dạng số

1.1. Kiểu dữ liệu dạng số

1.1.1. DECIMAL

DECIMAL là kiểu dữ liệu dạng số thập phân với miền giá trị và độ dài không cố định.

Ví dụ: 12.24, 976.54922

1.1.2. INT

INT là kiểu dữ liệu số nguyên 4-bytes. Miền giá trị của **INT** mặc định từ -2,147,483,648 đến 2,147,483,647.

- **INT8:** Phạm vi từ -128 đến 127.
- **INT16:** Phạm vi từ -32,768 đến 32,767.
- **INT32:** Phạm vi từ -2,147,483,648 đến 2,147,483,647.
- **INT64:** Phạm vi từ -9,223,372,036,854,775,808 đến 9,223,372,036,854,775,807.

Ví dụ: 5678

1.1.3. SHORT

SHORT là kiểu dữ liệu số nguyên 2-bytes. Miền giá trị của **SHORT** mặc định từ -32,768 đến 32,767.

Ví dụ: 9382

1.1.4. LONG

LONG là kiểu dữ liệu số nguyên 4-bytes. Miền giá trị của **LONG** mặc định từ -9,223,372,036,854,775,808 đến 9,223,372,036,854,775,807.

Ví dụ: 84700374

1.1.5. UNIT

UINT là kiểu dữ liệu số nguyên không âm. Miền giá trị mặc định từ 0 đến 4,294,967,295. Ititan hỗ trợ **UNIT** dưới dạng ép kiểu dữ liệu.

- **UINT16:** Phạm vi từ 0 đến 65,535.
- **UINT32:** Phạm vi từ 0 đến 4,294,967,295.
- **UINT64:** Phạm vi từ 0 đến 18,446,744,073,709,551,615.

Ví dụ: 1239, 8776, 26

1.1.6. FLOAT

Được sử dụng để biểu diễn số thực (số có phần thập phân) với độ lớn 32 bit và độ chính xác tới 6

chữ số phần thập phân.

Ví dụ: 32.98

1.1.7. DOUBLE

Được sử dụng để biểu diễn số thực (số có phần thập phân) với độ lớn 64 bit và độ chính xác tới 15 số thập phân.

Ví dụ: 652.583827523

1.2. Kiểu dữ liệu ngày tháng

1.2. Kiểu dữ liệu ngày tháng

1.2.1. DATE

Kiểu dữ liệu **DATE** cho phép tương tác và lưu trữ các dạng dữ liệu thời gian/ngày tháng.

Ví dụ: 30/05/2024 14:22:30

1.3. Kiểu dữ liệu nhị phân

1.3. Kiểu dữ liệu nhị phân

1.3.1. BOOLEAN

BOOLEAN là kiểu dữ liệu chỉ nhận 3 giá trị: True(1), False(0), Null

Ví dụ: True, False, True

1.4. Kiểu dữ liệu dạng chuỗi

1.4. Kiểu dữ liệu dạng chuỗi

1.4.1. **STRING**

STRING là kiểu dữ liệu dạng văn bản. Sql Lab hỗ trợ string theo định dạng Unicode.

Ví dụ: iTitan xin chào, iNet Solutions

1.4.2. **CHAR**

CHAR là một kiểu dữ liệu cơ bản được sử dụng để lưu trữ các ký tự đơn lẻ, chẳng hạn như chữ cái, chữ số hoặc ký tự đặc biệt.

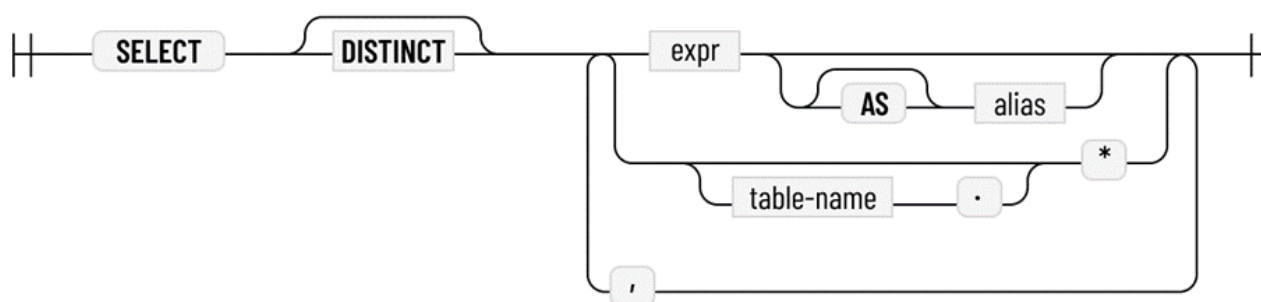
Ví dụ: a, b, @

Chương 2. CÚ PHÁP TRUY VẤN

2.1. SELECT

Mệnh đề **SELECT** chỉ định danh sách các cột sẽ được trả về bởi truy vấn. Mặc dù nó xuất hiện đầu tiên trong câu lệnh, nhưng về mặt logic, các biểu thức trong mệnh đề này chỉ được thực thi ở giai đoạn cuối cùng. Mệnh đề **SELECT** có thể chứa các biểu thức biến đổi đầu ra, cũng như các hàm tổng hợp.

a) Cú pháp:



b) Ví dụ:

Chọn tất cả các cột từ bảng "**table**":

```
SELECT *  
FROM table
```

Thực hiện phép tính toán trên cột và thêm bí danh:

```
SELECT add(col1,col2) as res, sqrt(col1) as root  
FROM table
```

Trả về tổng số hàng trong bảng:

```
SELECT count(*)  
FROM table
```

SELECT - *

Chọn tất cả các cột từ bảng "**mat_hang**":

```
SELECT *  
FROM mat_hang
```

Biểu thức ngôi sao (*) hoạt động như một ký tự đại diện (wildcard), chỉ định tất cả các cột từ một hoặc nhiều bảng trong truy vấn.

SELECT - DISTINCT

Chọn tất cả các mặt hàng duy nhất từ bảng "**mat_hang**".

```
SELECT DISTINCT *  
FROM table
```

Mệnh đề **DISTINCT** có thể được sử dụng để chỉ trả về các hàng duy nhất trong kết quả – do đó, bất kỳ hàng trùng lặp nào cũng sẽ bị lọc ra.

Lưu ý: Các truy vấn bắt đầu bằng **SELECT DISTINCT** thực hiện việc loại bỏ các bản ghi trùng lặp, đây là một thao tác tốn kém tài nguyên. Do đó, chỉ sử dụng **DISTINCT** khi thực sự cần thiết.

SELECT - Hàm tổng hợp

Trả về tổng số hàng trong bảng "**mat_hang**":

```
SELECT count(*)  
FROM table
```

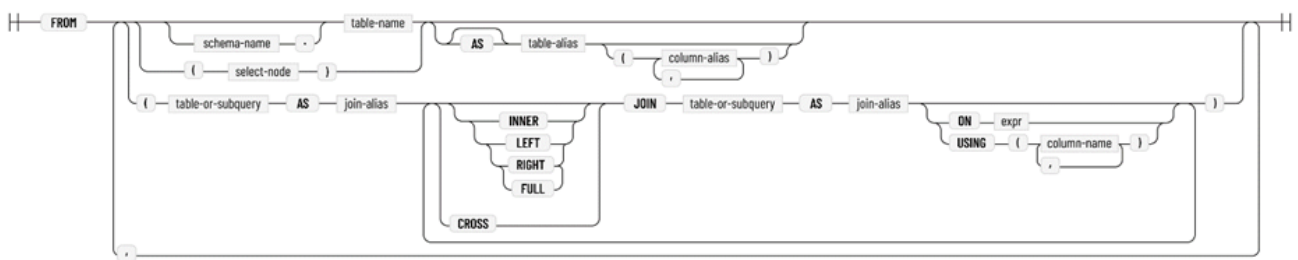
Trả về tổng số hàng trong bảng mat_hang nhóm theo cột "**mat_hang**":

```
SELECT city, count(*)  
FROM addresses  
GROUP BY city;
```

2.2. FROM

Mệnh đề **FROM** chỉ định nguồn dữ liệu mà phần còn lại của truy vấn sẽ hoạt động trên đó. Về mặt logic, mệnh đề **FROM** là nơi bắt đầu thực hiện truy vấn. Mệnh đề **FROM** có thể chứa một bảng duy nhất, sự kết hợp của nhiều bảng được nối với nhau bằng mệnh đề **JOIN** hoặc một truy vấn **SELECT** khác bên trong truy vấn con.

a) Cú pháp:



b) Ví dụ:

Chọn tất cả các cột từ bảng **"table_name"**:

```
SELECT *
FROM table_name;
```

Chọn tất cả các cột từ bảng **"table_name"** với bí danh tn:

```
SELECT tn.*
FROM table_name tn;
```

Chọn tất cả các cột từ truy vấn con:

```
SELECT *
FROM (SELECT * FROM table_name);
```

Chọn tất cả các cột từ hai bảng bằng phép **"Joins"**:

```
SELECT *  
FROM table_name  
JOIN other_table ON (table_name.key = other_table.key);
```

2.3. JOINS

JOINS được sử dụng để nối hai bảng theo chiều ngang với mỗi hàng kết quả đều chứa thông tin từ cả hai bảng bên trái và bên phải.

a) Outer Joins & Inner Join

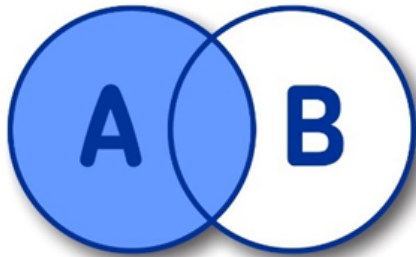
Outer Joins & Inner Join sử dụng các điều kiện để khớp các hàng từ hai bảng thông qua mệnh đề ON, thông thường là sử dụng các khóa, Tuy nhiên, cũng có nhiều trường hợp các hàng khớp với nhau thông qua các cột không phải khóa. Ititan cung cấp 4 loại **Outer Join** là: Left join, Right join và Full join

b) Cross Join

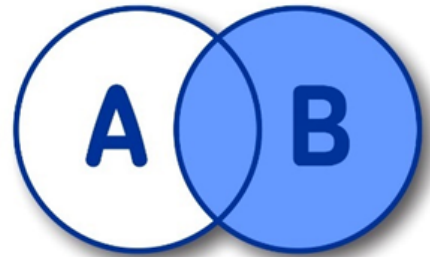
Cross Join là phép Joins không cần điều kiện, mỗi hàng từ bảng đầu tiên sẽ được kết hợp với tất cả các hàng từ bảng thứ hai. **Cross Join** thực hiện phép nhân Descartes (Cartesian product) giữa các bảng. Điều này có nghĩa là nó sẽ kết hợp từng hàng của bảng đầu tiên với từng hàng của bảng thứ hai.

c) Minh họa:

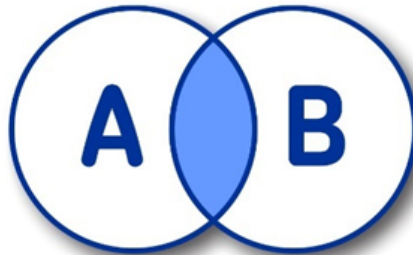
SQL JOINS



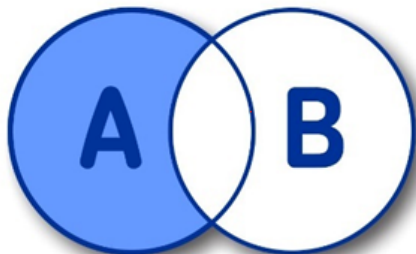
SELECT * FROM
A **LEFT** JOIN B
ON A.KEY = B.KEY



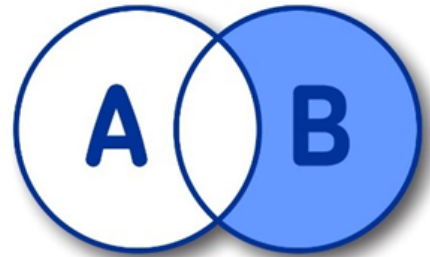
SELECT * FROM
A **RIGHT** JOIN B
ON A.KEY = B.KEY



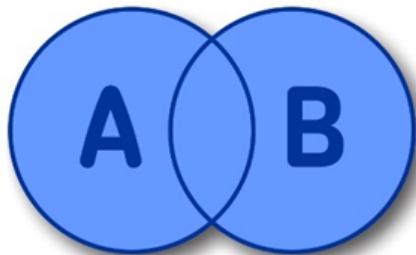
SELECT * FROM
A **INNER** JOIN B
ON A.KEY = B.KEY



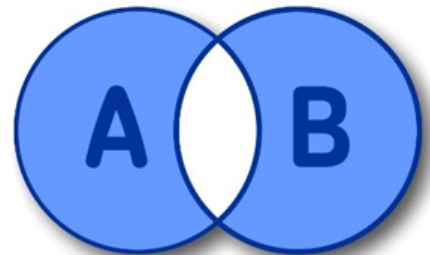
SELECT * FROM A
LEFT JOIN B
ON A.KEY = B.KEY
WHERE B.KEY IS NULL



SELECT * FROM A
RIGHT JOIN B
ON A.KEY = B.KEY
WHERE A.KEY IS NULL



SELECT * FROM A
FULL JOIN B
ON A.KEY = B.KEY

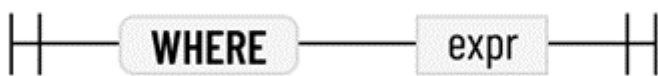


SELECT * FROM A **FULL**
JOIN B ON A.KEY =
B.KEY WHERE A.KEY IS
NULL OR B.KEY IS NULL

2.4. WHERE

Mệnh đề **WHERE** chỉ định bất kỳ bộ lọc nào để áp dụng cho dữ liệu. Điều này cho phép người sử dụng lọc những thông tin cần thiết. Về mặt logic, mệnh đề **WHERE** được áp dụng ngay sau mệnh đề **FROM**.

a) Cú pháp:



b) Ví dụ:

Chọn tất cả các hàng có id bằng 3:

```
SELECT *  
FROM table_name  
WHERE id = 3;
```

Chọn tất cả các hàng khớp với biểu thức "**Like**":

```
SELECT *  
FROM table_name  
WHERE name LIKE '%mark%';
```

Chọn tất cả các hàng có id bằng 3 hoặc bằng 7:

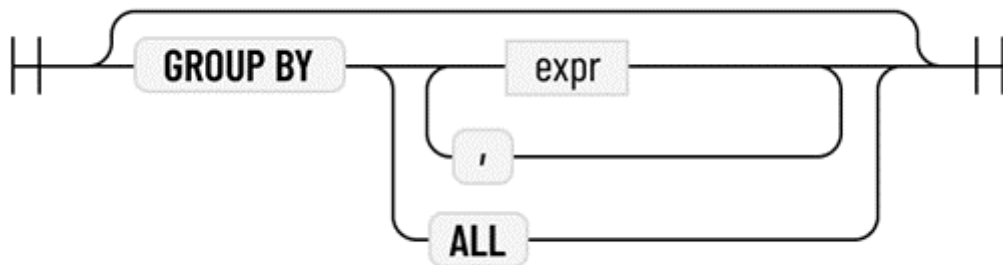
```
SELECT *  
FROM table_name  
WHERE id = 3 OR id = 7;
```


2.5. GROUP BY

Mệnh đề **GROUP BY** chỉ định những cột nhóm sẽ được sử dụng để thực hiện các phép tổng hợp nào trong mệnh đề **SELECT**. Nếu mệnh đề **GROUP BY** được chỉ định, truy vấn luôn là truy vấn tổng hợp, ngay cả khi không có phép tổng hợp nào có trong mệnh đề **SELECT**.

Khi mệnh đề **GROUP BY** được chỉ định, tất cả các bộ dữ liệu có dữ liệu khớp trong các cột nhóm (tức là tất cả các bộ dữ liệu thuộc cùng một nhóm) sẽ được kết hợp. Các giá trị của chính các cột nhóm không thay đổi và bất kỳ cột nào khác có thể được kết hợp bằng cách sử dụng hàm tổng hợp.

a) Cú pháp:



b) Ví dụ:

Đếm số lượng địa chỉ có trong các thành phố khác nhau:

```
SELECT city, count(*)
FROM addresses
GROUP BY city;
```

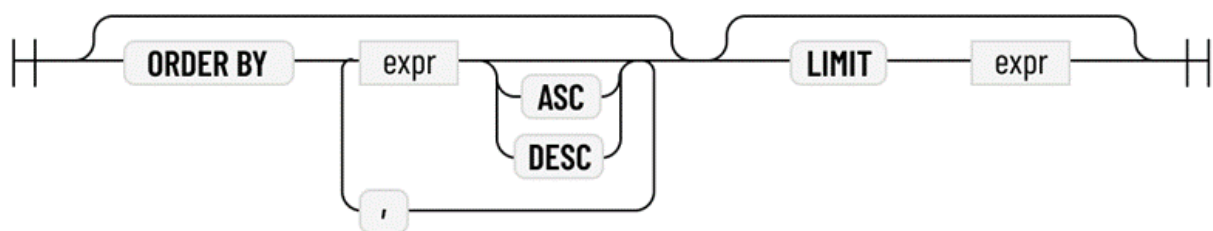
Tính trung bình thu nhập theo từng khu vực đường trong các thành phố khác nhau:

```
SELECT city, street_name, avg(income)
FROM addresses
GROUP BY city, street_name;
```

2.6. ORDER BY

ORDER BY sắp xếp kết quả đầu ra của truy vấn. Về mặt logic, nó được áp dụng ở cuối truy vấn. Mệnh đề **ORDER BY** sắp xếp các hàng theo tiêu chí sắp xếp theo thứ tự tăng dần hoặc giảm dần. Mệnh đề **ORDER BY** có thể chứa một hoặc nhiều biểu thức. Mỗi biểu thức có thể tùy ý được theo sau bởi một công cụ sửa đổi thứ tự (ASC hoặc DESC, mặc định là ASC)

a) Cú pháp:



b) Ví dụ:

Sắp xếp thứ tự các hàng dữ liệu theo cột "**city**":

```
SELECT *  
FROM table_name  
ORDER BY name;
```

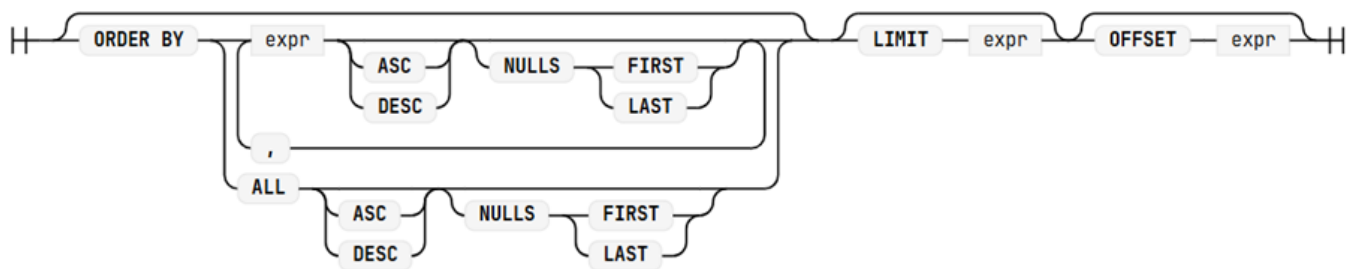
Sắp xếp thứ tự các hàng dữ liệu theo cột "**city**" theo thứ tự giảm dần:

```
SELECT *  
FROM table_name  
ORDER BY city DESC;
```

2.7. LIMIT

LIMIT là một mệnh đề dùng để giới hạn số lượng hàng dữ liệu được trả về trong một truy vấn . Về mặt logic, nó được áp dụng ở cuối truy vấn. Mệnh đề **LIMIT** hạn chế số lượng hàng được truy vấn.

a) Cú pháp:



b) Ví dụ:

Lấy 5 hàng dữ liệu trong truy vấn.

```
SELECT *
FROM table
LIMIT 5;
```

Lấy 5 giá trị lớn nhất trong cột “**name**”.

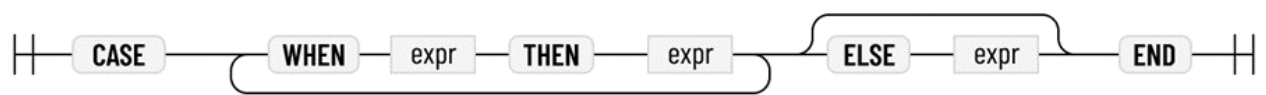
```
SELECT city, count(*) AS population
FROM table
GROUP BY name
ORDER BY value DESC
LIMIT 5;
```

Chương 3. BIỂU THỨC

3.1. CASE

CASE thực hiện việc thêm giá trị cho một cột mới dựa trên điều kiện.

a) Cú pháp:



b) Ví dụ:

Thêm giá trị theo điều kiện của cột i:

```
-- i[1,2,3]
SELECT i, CASE WHEN i > 2 THEN 1 ELSE 0 END AS test
FROM integers;
```

i	test
1	0
2	0
3	1

Phần **WHEN THEN** của biểu thức điều kiện **CASE** có thể được nối tiếp, bất cứ khi nào bất kỳ điều kiện nào trả về giá trị true cho một bộ đơn lẻ, biểu thức tương ứng sẽ được đánh giá và trả về.

```
SELECT i, CASE WHEN i = 1 THEN 10 WHEN i = 2 THEN 20 ELSE 0 END AS test
FROM integers;
```

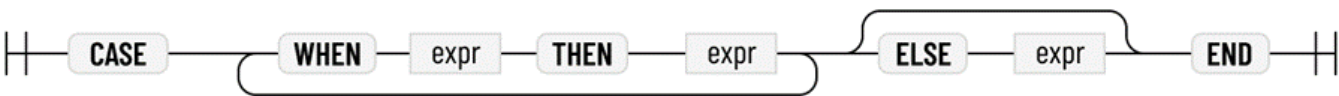
i	test
1	10
2	20
3	0

3.2. So sánh

Toán tử so sánh:

SQL_Lab cung cấp 6 toán tử so sánh tiêu chuẩn. Bất cứ khi nào một trong hai đối số đầu vào là **NULL** thì đầu ra của phép so sánh là **NULL**.

Cú pháp:



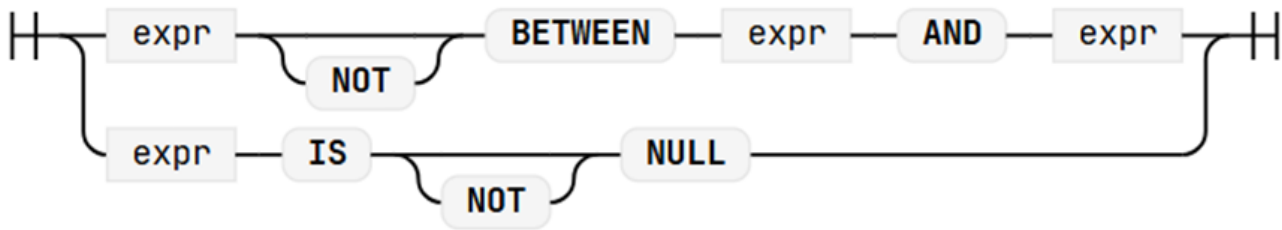
Toán tử so sánh:

Toán tử	Mô tả	Ví dụ	Kết quả
<	Bé hơn	2 < 3	True
>	Lớn hơn	2 > 3	False
<=	Bé hơn hoặc bằng	2 <= 3	True
>=	Lớn hơn hoặc bằng	4 >= Null	Null
=	Bằng	Null = Null	Null
< > hoặc	Không bằng	2 < > 2	False

BEETWEEN và IS [NOT] NULL

Bên cạnh các toán tử so sánh chuẩn, còn có các toán tử **BETWEEN** và **IS (NOT) NULL**. Các toán tử này hoạt động rất giống các toán tử, nhưng có cú pháp đặc biệt theo chuẩn SQL. Chúng được hiển thị trong bảng bên dưới.

Cú pháp:



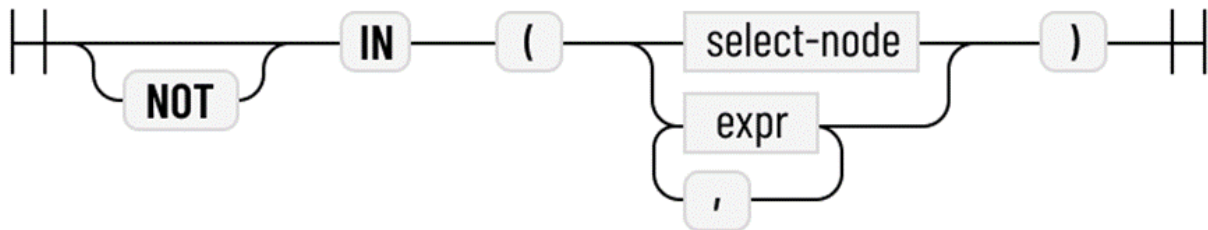
BETWEEN và IS [NOT] NULL

Thuộc tính	Mô tả
a BETWEEN x AND y	$x < a$ và $a < y$
a NOT BETWEEN x AND y	$x > a$ và $a > y$
IS NULL	true nếu biểu thức NULL, false nếu ngược lại
IS NOT NULL	false nếu biểu thức NULL, true nếu ngược lại

Lưu ý: **BETWEEN** và **NOT BETWEEN** chỉ tương đương với các ví dụ bên dưới trong trường hợp cả a, x và y đều cùng kiểu, vì **BETWEEN** sẽ ép kiểu tất cả các đầu vào của nó thành cùng kiểu.

3.3. Toán tử IN

a) Cú pháp



b) Toán tử

IN

Toán tử **IN** kiểm tra sự chứa đựng của biểu thức bên trái bên trong tập hợp các biểu thức ở phía bên phải (P). Toán tử **IN** trả về true nếu biểu thức có trong P, false nếu biểu thức không có trong P và P không có giá trị NULL hoặc NULL nếu biểu thức không có trong P và P có giá trị NULL.

```
SELECT 'Toán' IN ('Toán', 'Văn');  
-- true
```

```
SELECT 'Toán' IN ('Văn', 'Anh');  
-- false
```

```
SELECT 'Toán' IN ('Văn', Toán', NULL);  
-- true
```

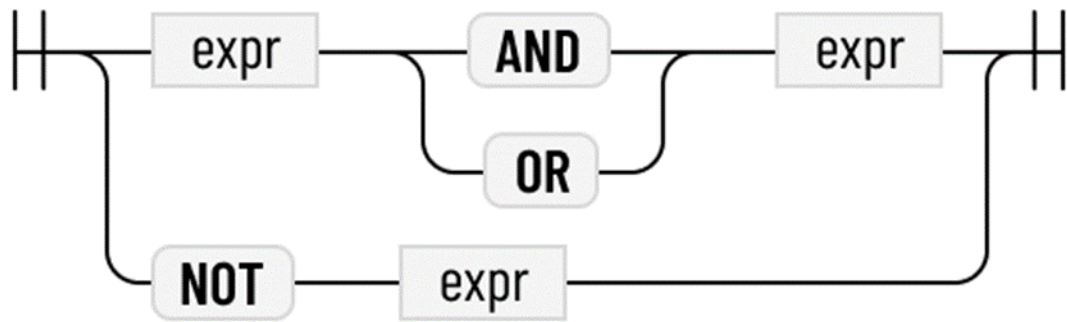
```
SELECT 'Toán' IN ('Văn', 'Anh', NULL);  
-- NULL
```


NOT IN

NOT IN có thể được sử dụng để kiểm tra xem một phần tử có tồn tại trong tập hợp hay không. x **NOT IN** y tương đương với **NOT**(x **IN** y)

3.4. Toán tử logic

a) Cú pháp:



b) Toán tử:

Toán tử nhị phân:

a	b	a AND b	a OR b
true	true	true	True
true	false	false	false
true	NULL	NULL	true
false	false	false	false
false	NULL	false	NULL
NULL	NULL	NULL	NULL

Lưu ý: Các toán tử logic liên quan đến NULL không phải lúc nào cũng có giá trị là NULL. Ví dụ: NULL AND false sẽ đánh giá là sai và NULL OR true sẽ đánh giá là đúng. Dưới đây là các bảng sự thật đầy đủ.

Toán tử một ngôi:

a	NOT a
true	false
false	true
NULL	NULL

Chương 4. CÁC HÀM CHỨC NĂNG

4.1. Hàm thời gian

4.1.1. ADD_SECOND

Thêm một khoảng thời gian (tính bằng giây) vào một trường dữ liệu ngày tháng.

Cú pháp:

ADD_SECOND (trường_du_lieu_thoi_gian **DATETIME**, giá trị *INT*) => *DATETIME*

- **trường_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.
- **giá trị:** Số giây cần thêm.

Ví dụ:

```
SELECT ADD_SECOND('15-08-2019 10:01:30',30)
-- 15-08-2019 10:02:00
```

4.1.2. ADD_MINUTE

Thêm một khoảng thời gian (tính bằng phút) vào một trường dữ liệu ngày tháng.

Cú pháp:

ADD_MINUTE (trường_du_lieu_thoi_gian *DATETIME*, giá trị *INT*) => *DATETIME*

- **trường_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.
- **giá trị:** Số phút cần thêm.

Ví dụ:

```
SELECT ADD_MINUTE('15-08-2019 10:01:30',30)
```

4.1.3. ADD_HOUR

Thêm một khoảng thời gian (tính bằng giờ) vào một trường dữ liệu ngày tháng.

Cú pháp:

ADD_HOUR (truong_du_lieu_thoi_gian DATETIME, gia_tri INT) => DATETIME

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.
- **gia_tri:** Số giờ cần thêm.

Ví dụ:

```
SELECT ADD_HOUR('15-08-2019 10:01:30',5)
-- 15-08-2019 15:01:30
```

4.1.4. ADD_DAY

Thêm một khoảng thời gian (tính bằng ngày) vào một trường dữ liệu ngày tháng.

Cú pháp:

ADD_DAY (truong_du_lieu_thoi_gian DATETIME, gia_tri INT) => DATETIME

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.
- **gia_tri:** Số ngày cần thêm.

Ví dụ:

```
SELECT ADD_DAY('15-08-2019 10:01:30',30)
-- 15-08-2019 10:01:30
```

4.1.5. ADD_WEEK

Thêm một khoảng thời gian (tính bằng tuần) vào một trường dữ liệu ngày tháng.

Cú pháp:

ADD_WEEK (truong_du_lieu_thoi_gian DATETIME, gia_tri INT) => DATETIME

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.
- **gia_tri:** Số tuần cần thêm.

Ví dụ:

```
SELECT ADD_WEEK('15-08-2019 10:01:30',30)
-- 15-08-2019 10:02:00
```

4.1.6. ADD_MONTH

Thêm một khoảng thời gian (tính bằng tháng) vào một trường dữ liệu ngày tháng.

Cú pháp:

ADD_MONTH (truong_du_lieu_thoi_gian DATETIME, gia_tri INT) => DATETIME

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.
- **gia_tri:** Số tháng cần thêm.

Ví dụ:

```
SELECT ADD_MONTH('15-08-2019 10:01:30',6)
-- 15-12-2019 10:01:30
```

4.1.7. ADD_QUARTER

Thêm một khoảng thời gian (tính bằng quý) vào một trường dữ liệu ngày tháng.

Cú pháp:

ADD_QUARTER (truong_du_lieu_thoi_gian DATETIME, gia_tri INT) => DATETIME

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

- **gia_tri:** Số quý cần thêm.

Ví dụ:

```
SELECT ADD_QUARTER('15-08-2019 10:01:30',1)
-- 15-11-2019 10:01:30
```

4.1.8. ADD_YEAR

Thêm một khoảng thời gian (tính bằng năm) vào một trường dữ liệu ngày tháng.

Cú pháp:

ADD_YEAR (truong_du_lieu_thoi_gian DATETIME, gia_tri INT) => DATETIME

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.
- **gia_tri:** Số năm cần thêm.

Ví dụ:

```
SELECT ADD_YEAR('15-08-2019 10:01:30',5)
-- 15-08-2024 10:01:30
```

4.1.9. SECOND

Trả về giá trị giây (từ 0 -59) trong trường dữ liệu thời gian.

Cú pháp:

SECOND (truong_du_lieu_thoi_gian DATETIME) => INT

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT SECOND('15-08-2019 10:01:30')
-- 30
```

4.1.10. MINUTE

Trả về giá trị phút (từ 0 -59) trong trường dữ liệu thời gian.

Cú pháp:

MINUTE (truong_du_lieu_thoi_gian DATETIME) => INT

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT MINUTE('15-08-2019 10:01:30')  
-- 1
```

4.1.11. HOUR

Trả về giá trị giờ (từ 0- 23) trong trường dữ liệu thời gian.

Cú pháp:

HOUR (truong_du_lieu_thoi_gian DATETIME) => INT

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT HOUR('15-08-2019 10:01:30')  
-- 10
```

4.1.12. DAY

Trả về giá trị ngày (từ 1- 31) trong trường dữ liệu thời gian.

Cú pháp:

DAY (truong_du_lieu_thoi_gian DATETIME) => INT

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT DAY('15-08-2019 10:01:30')  
-- 15
```

4.1.13. MONTH

Trả về giá trị tháng (từ 1- 12) trong trường dữ liệu thời gian.

Cú pháp:

MONTH (truong_du_lieu_thoi_gian DATETIME) => INT

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT MONTH('15-08-2019 10:01:30')  
-- 8
```

4.1.14. QUARTER

Trả về giá trị quý (từ 1- 4) trong trường dữ liệu thời gian.

Cú pháp:

QUARTER (truong_du_lieu_thoi_gian DATETIME) => INT

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT QUARTER('15-08-2019 10:01:30')  
-- 3
```

4.1.15. YEAR

Trả về giá trị năm trong trường dữ liệu thời gian.

Cú pháp:

YEAR (truong_du_lieu_thoi_gian DATETIME) => INT

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT YEAR('15-08-2019 10:01:30')  
-- 2019
```

4.1.16. LASTDAY

Trả về ngày cuối cùng trong tháng với trường dữ liệu thời gian.

Cú pháp:

LASTDAY (truong_du_lieu_thoi_gian DATETIME) => DATETIME

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT LASTDAY('15-08-2019 10:01:30')  
-- 31-08-2019 10:01:30
```

4.1.17. TRUNC_SECOND

Trả về giá trị thời gian với độ chính xác được tính theo giây.

Cú pháp:

TRUNC_SECOND (truong_du_lieu_thoi_gian DATETIME) => INT

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT TRUNC_SECOND('15-08-2019 10:01:30')  
-- 15-08-2019 10:01:30
```

4.1.18. TRUNC_MINUTE

Trả về giá trị thời gian với độ chính xác được tính theo phút.

Cú pháp:

TRUNC_MINUTE (truong_du_lieu_thoi_gian) => DATETIME

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT TRUNC_MINUTE('15-08-2019 10:01:30')  
-- 15-08-2019 10:01:00
```

4.1.19. TRUNC_HOUR

Trả về giá trị thời gian với độ chính xác được tính theo giờ.

Cú pháp:

TRUNC_HOUR (truong_du_lieu_thoi_gian DATETIME) => DATETIME

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT TRUNC_HOUR('15-08-2019 10:01:30')  
-- 15-08-2019 10:00:00
```

4.1.20. TRUNC_DAY

Trả về giá trị thời gian với độ chính xác được tính theo ngày.

Cú pháp:

TRUNC_DAY (truong_du_lieu_thoi_gian DATETIME) => DATETIME

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT TRUNC_DAY ('15-08-2019 10:01:30')  
-- 15-08-2019 00:00:00
```

4.1.21. TRUNC_WEEK

Trả về giá trị thời gian với độ chính xác được tính theo tuần.

Cú pháp:

TRUNC_WEEK (truong_du_lieu_thoi_gian DATETIME) => DATETIME

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT TRUNC_WEEK('15-08-2019 10:01:30')  
-- 04-08-2019 00:00:00
```

4.1.22. TRUNC_MONTH

Trả về giá trị thời gian với độ chính xác được tính theo tháng.

Cú pháp:

TRUNC_MONTH (truong_du_lieu_thoi_gian DATETIME) => DATETIME

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT TRUNC_MONTH('15-08-2019 10:01:30')  
-- 01-08-2019 00:00:00
```

4.1.23. TRUNC_QUARTER

Trả về giá trị thời gian với độ chính xác được tính theo quý.

Cú pháp:

TRUNC_QUARTER (truong_du_lieu_thoi_gian DATETIME) => DATETIME

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT TRUNC_QUARTER('15-08-2019 10:01:30')  
-- 01-07-2019 00:00:00
```

4.1.24. TRUNC_YEAR

Trả về giá trị thời gian với độ chính xác được tính theo năm.

Cú pháp:

TRUNC_YEAR (truong_du_lieu_thoi_gian DATETIME) => DATETIME

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT TRUNC_YEAR('15-08-2019 10:01:30')  
-- 01-01-2019 00:00:00
```

4.1.25. DOM

Trả về thứ tự ngày trong tháng từ một biểu thức thời gian.

Cú pháp:

DOM (truong_du_lieu_thoi_gian DATETIME) => INT

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT DOM('15-08-2019 10:01:30')  
-- 15
```

4.1.26. DOW

Trả về thứ tự ngày trong tuần từ một biểu thức thời gian.

Cú pháp:

DOW (truong_du_lieu_thoi_gian DATETIME) => INT

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

```
SELECT DOW('15-08-2019 10:01:30')  
-- 4(15-08-2019 là thứ 5, 4 là thứ tự của thứ 5 trong tuần)
```

4.1.27. DOY

Trả về thứ tự ngày trong năm từ một biểu thức thời gian.

Cú pháp:

DOY (truong_du_lieu_thoi_gian DATETIME) => INT

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

```
SELECT DOY('15-08-2019 10:01:30')  
-- 227(Năm 2019 có 365 ngày, 15-08-2019 là ngày thứ 227)
```

4.1.28. WOY

Trả về thứ tự tuần trong năm từ một biểu thức thời gian.

Cú pháp:

WOY (truong_du_lieu_thoi_gian DATETIME) => INT

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

```
SELECT WOY('15-08-2019 10:01:30')  
-- 33(Một năm có 54 tuần, ngày 15-08-2019 là tuần thứ 33 trong năm)
```

4.1.29. YWEEK

Trả về thứ tự tuần trong năm từ một biểu thức thời gian.

Cú pháp:

YWEEK (truong_du_lieu_thoi_gian DATETIME) => INT

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT YWEEK('15-08-2019 10:01:30')  
-- 33
```

4.1.30. EPOCH

Tính số giây đã trôi qua từ một mốc dữ liệu thời gian so với ngày 1 tháng 1 năm 1970 lúc 00:00:00 giờ UTC.

Cú pháp:

EPOCH (truong_du_lieu_thoi_gian DATETIME) => BIGINT

- **truong_du_lieu_thoi_gian:** Biểu thức hoặc cột trả về kiểu dữ liệu thời gian/ngày tháng.

Ví dụ:

```
SELECT EPOCH('24/06/2024 10:01:30')  
-- 1719223290
```

```
SELECT EPOCH('01/01/1970 00:00:00')  
-- 0
```


4.2. Hàm toán học

4.2.1. ABS

Tính giá trị tuyệt đối của một biểu thức số.

Cú pháp:

ABS (truong_du_lieu_so Số) => Số

- **truong_du_lieu_so:** Trường dữ liệu dạng số cần tính giá trị tuyệt đối.

Ví dụ về ABS:

```
SELECT ABS(-10)
-- 10
```

```
SELECT ABS(0)
-- 0
```

4.4.2. ACOS

Tính arccosine của một biểu thức số.

Cú pháp:

ACOS (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** Số có đơn vị là radians và kiểu dữ liệu dạng số.

Ví dụ về ACOS:

```
SELECT ACOS(0)
-- 1.5707963267948966
```

```
SELECT ACOS(1.0)
```

```
-- 0
```

```
SELECT ACOS(-1)
```

```
-- 3.141592653589793
```

4.2.3. ADD

Tính giá trị của phép cộng giữa hai trường dữ liệu dạng số hoặc một trường dữ liệu và một số.

Cú pháp:

ADD (truong_du_lieu_so_1 Số, truong_du_lieu_so_2 Số) => Số

- **truong_du_lieu_so_1:** DOUBLE, INTEGER, BIGINT, DECIMAL, hoặc FLOAT.
- **truong_du_lieu_so_2:** DOUBLE, INTEGER, BIGINT, DECIMAL, hoặc FLOAT.

Ví dụ về ADD với dữ liệu số :

```
Select ADD(2, 3)
```

```
-- 6
```

Ví dụ về ADD cột "col" có hai giá trị [2,3,4]:

```
Select ADD(col, 3)
```

```
-- 5
```

```
-- 6
```

```
-- 7
```

4.2.4. ASIN

Tính arcsine của một biểu thức số.

Cú pháp:

ASIN (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** Số có đơn vị là radians và kiểu dữ liệu dạng số.

Ví dụ về Asin:

```
SELECT ASIN(0)
-- 0.0
```

```
SELECT ASIN(1)
-- 1.5707963267948966
```

```
SELECT ASIN(-1)
-- -1.5707963267948966
```

4.2.5. ATAN

Tính arctang của một biểu thức số.

Cú pháp:

ATAN (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** Số có đơn vị là radians và kiểu dữ liệu dạng số.

Ví dụ về ATAN:

```
SELECT ATAN(-1)
-- -0.7853981633974483
```

```
SELECT ATAN(19564.7)
-- 1.5707452143321894
```

4.2.6. ATAN2

Tính arctang của tỷ số giữa hai trường dữ liệu.

Cú pháp:

ATAN2 (y Số, x Số) => DOUBLE

- **y:** Giá trị đầu vào kiểu số thực đại diện cho tọa độ y, nằm trong khoảng từ âm vô cùng đến dương vô cùng.
- **x:** Giá trị đầu vào kiểu số thực đại diện cho tọa độ x, nằm trong khoảng từ âm vô cùng đến dương vô cùng.

Ví dụ về ATAN2:

```
SELECT ATAN2(1,0)
-- 1.5707452143321894
```

```
SELECT ATAN2(0.0,1.0)
-- 0
```

```
SELECT ATAN2(0.0,-1.0)
-- 3.141592653589793
```

```
SELECT ATAN2(-0.000000000001,-1.0)
-- -3.141592653579793
```

4.2.7. CBRT

Tính căn bậc 3 của các hàng trong một trường dữ liệu.

Cú pháp:

CBRT (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** trường dữ liệu dạng số cần tính căn bậc 3.

Ví dụ về CBRT:

```
SELECT CBRT(5)
-- 1.709975946676697
```

```
SELECT CBRT(120)
-- 4.932424148653812
```

```
SELECT CBRT(99.5)
-- 4.638049208321277
```

4.2.8. CEIL

Trả về giá trị bằng hoặc lớn hơn gần nhất nếu là số thập phân của các hàng trong trường dữ liệu đầu vào.

Cú pháp:

CEIL (truong_du_lieu_so Số) => INT

Ví dụ về CEIL:

```
SELECT CEIL(37.775420706711)
-- 38
```

```
SELECT CEIL(3.1459)
-- 4
```

```
SELECT CEIL(-37.775420706711)
-- -37
```

```
SELECT CEIL(0)
-- 0
```

4.2.9. COS

Tính cosine của một biểu thức số.

Cú pháp:

COS (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** Số có đơn vị là radians và kiểu dữ liệu dạng số.

Ví dụ về COS:

```
SELECT COS(0)
```

```
-- 1.0
```

```
SELECT COS(1.0)
```

```
-- 0.5403023058681398
```

```
SELECT COS(-1)
```

```
-- 0.5403023058681398
```

4.2.10. COSH

Tính hyperbolic cosine của một biểu thức số.

Cú pháp:

COSH (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** Số có đơn vị là radians và kiểu dữ liệu dạng số.

Ví dụ về COSH:

```
SELECT COSH(0)
```

```
-- 1.0
```

```
SELECT COSH(1.0)
```

```
-- 1.543080634815244
```

```
SELECT COSH(-1)
```

```
-- 1.543080634815244
```

4.2.11. COT

Tính cotang của một biểu thức số.

Cú pháp:

COT (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** Số có đơn vị là radians và kiểu dữ liệu dạng số.

Ví dụ về COT:

```
SELECT COT(0)
```

```
-- 1.0
```

```
SELECT COT(1.0)
```

```
-- 0.6420926159343306
```

```
SELECT COT(-1)
```

```
-- -0.6420926159343306
```

4.2.12. DEGRESS

Chuyển đổi đơn vị từ Radians sang Degress (Độ)

Cú pháp:

DEGRESS (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** Số có đơn vị là radians và có kiểu dữ liệu dạng số.

Ví dụ về DEGRESS:

```
SELECT DEGRESS(0)
```

```
-- 0.0
```

```
SELECT DEGRESS(-1)
```

```
-- 57.29577951308232
```

4.2.13. DIV

Tính kết quả của phép chia giữa hai biểu thức đầu vào

Cú pháp:

DIV (truong_du_lieu_so_1 Số, truong_du_lieu_so_2 Số) => Số

- **truong_du_lieu_so_1:** Số bị chia
- **truong_du_lieu_so_2:** Số chia, có thể là một số cụ thể hoặc một trường dữ liệu dạng số.

Ví dụ về DIV:

```
SELECT DIV(6,2)
-- 3
```

```
SELECT DIV(revenue,order)
-- 3
-- 8
-- 9
```

4.2.14. DIVIDE

Tính kết quả của phép chia lấy số nguyên giữa hai biểu thức số

Cú pháp:

DIVIDE (truong_du_lieu_so_1 Số, truong_du_lieu_so_2 Số) => Giá trị dạng số

- **truong_du_lieu_so_1:** Số bị chia
- **truong_du_lieu_so_2:** Số chia

Ví dụ về DIVIDE:

```
SELECT DIVIDE(revenue,2)
-- 1
-- 9
-- 20
```


4.2.15. EXP

Tính lũy thừa của số e (cơ số của logarit tự nhiên, xấp xỉ 2.718281...) với số mũ lần lượt là các giá trị trong một trường dữ liệu.

Cú pháp:

EXP (trường_du_lieu_so Số) => DOUBLE

- **trường_du_lieu_so:** Giá trị số mũ để nâng e lên

Ví dụ về EXP:

```
SELECT EXP(1)
-- 2.718281828459045
```

```
SELECT EXP(10.0)
-- 22026.465794806718
```

4.2.16. FACTORIAL

Tính giai thừa của một giá trị số.

Cú pháp:

FACTORIAL (gia_tri INT) => BIGINT

- **gia_tri:** Giá trị số nguyên từ 0- 20 trong một trường dữ liệu.

Ví dụ về FACTORIAL:

```
SELECT FACTORIAL(5)
-- 120
```

```
SELECT FACTORIAL(20)
-- 2432902008176640000
```

4.2.17. FLOOR

Trả về giá trị bằng hoặc nhỏ hơn gần nhất nếu là số thập phân của trường dữ liệu đầu vào.

Cú pháp:

FLOOR (truong_du_lieu_so Số) => INT

- **truong_du_lieu_so:** Giá trị số lớn hơn 0

Ví dụ về FLOOR:

```
SELECT FLOOR(0)
-- 0
```

```
SELECT CEIL(45.76)
-- 45
```

```
SELECT FLOOR(-1.3)
-- -2
```

4.2.18. LOG

Tính logarit tự nhiên của các giá trị trong một trường dữ liệu.

Cú pháp:

LOG (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** Giá trị số lớn hơn 0.

Ví dụ về LOG:

```
SELECT LOG(0)
-- null
```

```
SELECT LOG(1)
-- 0
```

```
SELECT LOG(5)
-- 1.6094379124341003
```

4.2.19. LOG10

Tính logarit cơ số 10 của trường dữ liệu đầu vào dạng số.

Cú pháp:

LOG10 (truong_du_lieu_so Số) => Giá trị dạng số

- **truong_du_lieu_so:** Giá trị số lớn hơn 0.

Ví dụ về LOG10:

```
SELECT LOG10(20.5)
-- 1.3117538610557542
```

```
SELECT LOG10(100)
-- 2.0
```

4.2.20. LOG1P

Tính logarit tự nhiên của 1 cộng một giá trị trong trường dữ liệu dạng số.

Cú pháp:

LOG1P (truong_du_lieu_so Số) => Giá trị dạng số

- **truong_du_lieu_so:** Giá trị số lớn hơn -1.

Ví dụ về LOG1P

```
SELECT LOG1P(0)
-- 0
```

```
SELECT LOG1P(0.6931471805599453)
-- 0
```

```
SELECT LOG1P(5)
-- 1.791759469228055
```

4.2.21. MULTIPLY

Tính kết quả của phép nhân giữa các dòng trong hai trường dữ liệu dạng số.

Cú pháp:

MULTIPLY (truong_du_lieu_so Số) => Giá trị dạng số

- **truong_du_lieu_so:** Giá trị của biểu thức muốn tính logarit.

Ví dụ về MULTIPLY:

```
SELECT MULTIPLY(10,2)
-- 20
```

```
SELECT MULTIPLY(-2.0,2.0)
-- -4.0
```

```
SELECT MULTIPLY(0,2)
-- 0
```

4.2.22. RADIANS

Chuyển đổi đơn vị từ Degress (Độ) sang Radians

Cú pháp:

RADIANS (x Số) => DOUBLE

- **x:** Số có đơn vị là degress và kiểu dữ liệu là DOUBLE, INTEGER, BIGINT, DECIMAL, hoặc FLOAT.

Ví dụ về RADIANS:

```
SELECT RADIANS(45)
-- 0.7853981633974483
```

4.2.23. ROUND

Trả về giá trị sau khi làm tròn cho các giá trị đầu vào. Nếu không nhập số lượng chữ số thập phân số nguyên gần nhất sẽ được trả về.

Cú pháp:

ROUND (truong_du_lieu_so Số, Gia_tri INT) => Số

- **truong_du_lieu_so:** Số cần làm tròn.
- **gia_tri:** Số lượng chữ số ở phần thập phân.

Ví dụ về ROUND:

```
SELECT ROUND(24,0)
-- 24
```

```
SELECT ROUND(24,-2)
-- 0
```

```
SELECT ROUND(24.35,1)
-- 24.4
```

4.2.24. SIN

Tính sine của một biểu thức số.

Cú pháp:

SIN (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** Số có đơn vị là radians và kiểu dữ liệu là DOUBLE, INTEGER, DECIMAL, hoặc FLOAT.

Ví dụ về SIN:

```
SELECT SIN(360)
-- 0.9589157234143065
```

```
SELECT SIN(510.89)
-- 0.9282211721815067
```

```
SELECT SIN(-1)
-- -0.8414709848078965
```

4.2.25. SINH

Tính hyperbolic sine của một biểu thức số.

Cú pháp:

SINH (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** Số có đơn vị là radians và kiểu dữ liệu dạng số.

Ví dụ về SINH:

```
SELECT SINH(1)
-- 1.1752011936438014
```

```
SELECT SINH(1.5)
-- 2.1292794550948173
```

4.2.26. SQRT

Tính căn bậc hai của trường dữ liệu dạng số không âm.

Cú pháp:

SQRT (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** Trường dữ liệu cần tính căn bậc hai

Ví dụ về SQRT:

```
SELECT (25.25)
-- 5.024937810560445
```

```
SELECT (25)
-- 5.0
```

4.2.26. SUBTRACT

Tính kết quả của phép trừ giữa các dòng trong hai trường dữ liệu dạng số.

Cú pháp:

SUBTRACT (truong_du_lieu_so Số, so_tru Số) => Giá trị dạng số

- **truong_du_lieu_so:** Giá trị của trường dữ liệu được coi như số bị trừ.
- **so_tru:** Số trừ, có thể là một trường dữ liệu dạng số hoặc một số cụ thể

Ví dụ về SUBTRACT:

```
SELECT SUBTRACT (10,2)
-- 8
```

```
SELECT SUBTRACT (-2.0,2.0)
-- -4.0
```

```
SELECT SUBTRACT(0,2)
-- -2
```

4.2.28. TAN

Tính tang của một biểu thức số.

Cú pháp:

TAN (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** Số có đơn vị là radians và kiểu dữ liệu là DOUBLE, INTEGER, DECIMAL, hoặc FLOAT

Ví dụ về TAN:

```
SELECT TAN(180.8)
-- -6.259341891872157
```

```
SELECT TAN(1200)
-- -0.08862461268886584
```

4.2.29. TANH

Tính hyperbolic tang của một biểu thức số.

Cú pháp:

TANH (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** Số có đơn vị là radians và kiểu dữ liệu là: DOUBLE, INTEGER, BIGINT, DECIMAL, hoặc FLOAT.

Ví dụ về TANH:

```
SELECT TANH(1.5)
-- 0.9051482536448664
```

```
SELECT TANH(1)
-- 0.7615941559557649
```


4.3. Hàm tổng hợp

4.3.1. AVG

Tính giá trị trung bình của tất cả các giá trị một trường dữ liệu.

Cú pháp:

AVG (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so:** Trường dữ liệu cần tính giá trị trung bình với kiểu dữ liệu là DOUBLE, INTEGER, BIGINT, DECIMAL, hoặc FLOAT.

Ví dụ về AVG:

```
SELECT AVG(1.5)
-- 1.5
```

Ví dụ về AVG: cột val[0.6348, -1.301466]:

```
SELECT AVG("val")
-- -0.333333
```

Ví dụ về AVG: với hàm GROUP BY:

```
SELECT mat_hang, AVG(san_luong)
FROM inet.sample
GROUP BY mat_hang
-- Cà rốt, 3000
-- Cam sành, 25.25
-- Ổi, 70
```

4.3.2. COUNT

Đếm tổng số lượng của các giá trị trong một trường dữ liệu.

Cú pháp:

COUNT (Truong_du_lieu_so Số) => BIGINT

- **Truong_du_lieu_so**: Trường dữ liệu cần đếm với kiểu dữ liệu dạng số.

Ví dụ về COUNT:

```
SELECT COUNT(mat_hang)
-- 10
```

Ví dụ về COUNT: với hàm GROUP BY:

```
SELECT mat_hang, COUNT(san_luong)
FROM inet.sample
GROUP BY mat_hang
-- Cà rốt, 3
-- Cam sành, 5
-- Ổi, 2
```

4.3.3. COUNTDISTINCT

Đếm tổng số lượng của các giá trị trong một trường dữ liệu.

Cú pháp:

COUNT (truong_du_lieu_so Số) => BIGINT

- **truong_du_lieu_so**: Trường dữ liệu cần đếm với kiểu dữ liệu dạng số

Ví dụ về COUNT: với hàm GROUP BY:

```
SELECT mat_hang, COUNT(san_luong)
FROM inet.sample
GROUP BY mat_hang
-- Cà rốt, 3
-- Cam sành, 5
-- Ổi, 2
```

4.3.4. COUNTEXISTING

4.3.5. COUNTMISING

4.3.6. FIRST

4.3.7. FIRSTQUATILE

4.3.8. LAST

4.3.9. MAX

Tính giá trị lớn nhất của các giá trị trong một trường dữ liệu.

Cú pháp:

MAX (truong_du_lieu_so Số) => BIGINT

- **truong_du_lieu_so:** Trường dữ liệu cần đếm với kiểu dữ liệu dạng số

Ví dụ về MAX:

```
SELECT MAX(san_luong)
-- 300
```

Ví dụ về MAX: với hàm GROUP BY:

```
SELECT mat_hang, MAX(san_luong)
FROM inet.sample
GROUP BY mat_hang
-- Cà rốt, 300
-- Cam sành, 59
-- Ổi, 272
```

4.3.10.MEAN

4.3.11.MEDIAN

4.3.12.MIN

Tính giá trị nhỏ nhất của các giá trị trong một trường dữ liệu.

Cú pháp:

MIN (truong_du_lieu_so Số) => BIGINT

- **truong_du_lieu_so:** Trường dữ liệu cần đếm với kiểu dữ liệu dạng số

Ví dụ về MIN:

```
SELECT MIN(san_luong)
-- 10
```

Ví dụ về MIN: với hàm GROUP BY:

```
SELECT mat_hang, MIN(san_luong)
FROM inet.sample
GROUP BY mat_hang
-- Cà rốt, 46
-- Cam sành, 10
-- Ổi, 39
```

4.3.13.P25TH

4.3.14.P50TH

4.3.15.P75TH

4.3.16.P90TH

4.3.17.P99TH

4.3.18.SECONDQUARTILE

4.3.19.STD

4.3.20.SUM

Tính tổng của các giá trị trong một trường dữ liệu.

Cú pháp:

SUM (truong_du_lieu_so Số) => DOUBLE

- **truong_du_lieu_so**: Trường dữ liệu tính tổng đếm với kiểu dữ liệu dạng số.

Ví dụ về SUM:

```
SELECT SUM(san_luong)
-- 10
```

Ví dụ về SUM: với hàm GROUP BY:

```
SELECT mat_hang, SUM(san_luong)
FROM inet.sample
GROUP BY mat_hang
-- Cà rốt, 46
-- Cam sành, 10
-- Ổi, 39
```

1.1.21.THIRDQUATILE

1.1.22.VARIANCE

4.4. Hàm chuyển đổi

4.4.1. CASTBIGINT

Chuyển đổi kiểu dữ liệu của một cột thành số nguyên BIGINT.

Cú pháp:

CASTBIGINT (truong_du_lieu_so Số) => BIGINT

- truong_du_lieu_so: trường dữ liệu cần chuyển với kiểu dữ liệu là DOUBLE, INTEGER, BIGINT, DECIMAL, hoặc FLOAT.

Ví dụ về CASTBIGINT: cột col[1,2,10.5]:

```
SELECT CASTBIGINT(col)
-- 1, 2, 10
```

4.4.2. CASTDECIMAL

Chuyển đổi kiểu dữ liệu của một cột thành số thực DECIMAL.

Cú pháp:

CASTDECIMAL (truong_du_lieu_so Số) => DECIMAL

- truong_du_lieu_so: Trường dữ liệu cần đếm với kiểu dữ liệu là DOUBLE, INTEGER, BIGINT, DECIMAL, hoặc FLOAT.

Ví dụ về CASTDECIMAL: cột col[1,2,10.5]:

```
SELECT CASTDECIMAL(col)
-- 1.0 ,2.0 , 10.5
```

4.4.3. CASTDOUBLE

Chuyển đổi kiểu dữ liệu của một cột thành nguyên Double.

Cú pháp:

CASTDECIMAL (truong_du_lieu_so Số) => DOUBLE

- truong_du_lieu_so: Trường dữ liệu cần đếm với kiểu dữ liệu là DOUBLE, INTEGER, BIGINT, DECIMAL, hoặc FLOAT.

Ví dụ về CASTDECIMAL: cột col[1,2,10.5]:

```
SELECT CASTDECIMAL(col)
-- 1.0 ,2.0 , 10.5
```

4.4.4. CASTFLOAT

Chuyển đổi kiểu dữ liệu của một cột thành số thực Float.

Cú pháp:

CASTDECIMAL (truong_du_lieu_so Số) => FLOAT

- truong_du_lieu_so: Trường dữ liệu cần đếm với kiểu dữ liệu là DOUBLE, INTEGER, BIGINT, DECIMAL, hoặc FLOAT.

Ví dụ về CASTDECIMAL: cột col[1,2,10.5]:

```
SELECT CASTDECIMAL(col)
-- 1.0 ,2.0 , 10.5
```

4.4.5. CASTINT

Chuyển đổi kiểu dữ liệu của một cột thành số nguyên Int.

Cú pháp:

CASTDECIMAL (truong_du_lieu_so Số) => INT

- `truong_du_lieu_so`: Trường dữ liệu cần đếm với kiểu dữ liệu là DOUBLE, INTEGER, BIGINT, DECIMAL, hoặc FLOAT.

Ví dụ về CASTDECIMAL: cột col[1,2,10.5]:

```
SELECT CASTDECIMAL(col)
-- 1 ,2 , 10
```

4.4.6. CASTLONG

Chuyển đổi kiểu dữ liệu của một cột thành số nguyên Long.

Cú pháp:

CASTDECIMAL (`truong_du_lieu_so` Số) => LONG

- `truong_du_lieu_so`: Trường dữ liệu cần đếm với kiểu dữ liệu dạng số

Ví dụ về CASTDECIMAL: cột col[1,2,10.5]:

```
SELECT CASTDECIMAL(col)
-- 1, 2, 10
```

4.4.7. HASH

Trả về giá trị băm cho trường dữ liệu được truyền vào, hàm HASH có giá trị null dù cho giá trị đầu vào null

Cú pháp:

HASH (`truong_du_lieu`) => BIGINT

- `truong_du_lieu`: Giá trị được truyền vào để băm.

Ví dụ về HASH:

```
SELECT HASH('Ititan xin chào')
-- -1965350004
```

```
SELECT HASH('15/08/2019 10:01:30')
```

```
-- -44832748
```

4.5. Hàm chuỗi

4.5.1. ASCII

Chuyển kí tự dạng chuỗi thành mã ASCII tương ứng.

Cú pháp:

BTRIM (truong_du_lieu STRING) => STRING

- **truong_du_lieu:** Chuỗi được truyền vào để loại bỏ khoảng trắng.

Ví dụ về ASCII:

```
SELECT ASCII('iNet solutions')
```

```
-- 105(mã ASCII của ký tự 'i')
```

```
SELECT ASCII('W')
```

```
-- 87
```

4.5.2. BTRIM

Loại bỏ khoảng trắng ở đầu và cuối kí tự.

Cú pháp:

BTRIM (truong_du_lieu STRING) => STRING

- **truong_du_lieu:** Chuỗi được truyền vào để loại bỏ khoảng trắng.

Ví dụ về BTRIM:

```
SELECT BTRIM(' Ititan xin chào ')
```

```
-- Ititan xin chào
```

4.5.3. CONCAT

Nối hai hoặc nhiều chuỗi lại với nhau thành một chuỗi duy nhất.

Cú pháp:

CONCAT (trung_du_lieu_1 STRING, trung_du_lieu_2 STRING) => Chuỗi

- **trung_du_lieu_1:** Chuỗi ban đầu.
- **trung_du_lieu_2:** Trường dữ liệu dạng chuỗi hoặc chuỗi.

Ví dụ về CONCAT:

```
SELECT CONCAT('Ititan xin chào','iNet solutions')
```

```
-- Ititan xin chàoiNet solutions
```

4.5.4. INITCAP

Chuyển đổi các chuỗi kí tự trong biểu thức với chữ cái đầu tiên của mỗi chữ được viết hoa.

Cú pháp:

INITCAP (trung_du_lieu STRING) => STRING

- **trung_du_lieu:** Chuỗi được truyền vào để xử lý.

Ví dụ về INITCAP:

```
SELECT INITCAP('ititan xin chào')
```

```
-- Ititan Xin Chào
```

4.5.5. INSTR

Kiểm tra sự tồn tại của một chuỗi con bên trong một chuỗi lớn hơn.

Cú pháp:

INSTR (truong_du_lieu STRING, chuoi_con STRING) => BOOL

- **truong_du_lieu:** Chuỗi được truyền vào để xử lý.
- **chuoi_con:** Chuỗi được truyền vào để so sánh

Ví dụ về INSTR:

```
SELECT INSTR('ititan xin chào','iti')
```

```
-- 1
```

```
SELECT INSTR('ititan xin chào','abc')
```

```
-- 0
```

4.5.6. LENGTH

Trả về độ dài của chuỗi kí tự.

Cú pháp:

LENGTH (truong_du_lieu STRING) => INT

- **truong_du_lieu:** Chuỗi được truyền vào để lấy độ dài.

Ví dụ về LENGTH:

```
SELECT LENGTH('Ititan xin chào')
```

```
-- 15
```

4.5.7. LOWER

Trả về tất cả các kí tự trong một chuỗi thành chữ không in hoa.

Cú pháp:

LOWER (truong_du_lieu STRING) => STRING

- **truong_du_lieu:** Chuỗi được truyền vào để viết hoa.

Ví dụ về LOWER:

```
SELECT LOWER('ITITAN XIN CHÀO')
```

```
-- ititan xin chào
```

4.5.8. LPAD

Thêm kí tự được chỉ định hoặc khoảng trắng cho đến khi chuỗi đạt đủ số lượng.

Cú pháp:

LPAD (truong_du_lieu STRING, do_dai INT, ki_tu STRING) => Chuỗi

- **truong_du_lieu:** Chuỗi được truyền vào để thêm kí tự.
- **do_dai:** Số lượng kí tự.
- **ki_tu:** kí tự hoặc chuỗi để thêm.

Ví dụ về LPAD:

```
SELECT LPAD('Ititan xin chào', 20, ' ')
```

```
-- Ititan xin chào
```

```
SELECT LPAD('iNet Solution', 18, 'i')
```

```
-- iiiiiiNet Solution
```

Lưu ý: Nếu ki_tu không được nhập vào, giá trị mặc định là khoảng trắng sẽ được thêm. Nếu chuỗi lớn hơn do_dai sẽ trả về chuỗi gốc ban đầu.

4.5.9. LTRIM

Trả về giá trị lặp lại của các kí tự trong trường dữ liệu.

Cú pháp:

LTRIM (truong_du_lieu STRING) => STRING

- **truong_du_lieu:** Chuỗi được truyền vào để loại bỏ khoảng trắng.

Ví dụ về LTRIM:

```
SELECT LTRIM(' iTitan xin chào')
```

```
-- iTitan xin chào
```

4.5.10. MASK_FIRST

Giấu một số ký tự đầu tiên của trường dữ liệu dạng chuỗi.

Cú pháp:

MASK_FIRST (truong_du_lieu STRING, do_dai) => STRING

- **truong_du_lieu:** Chuỗi được truyền vào để loại bỏ khoảng trắng.
- **do_dai:** số kí tự cần che giấu.

Ví dụ về MASK_FIRST:

```
SELECT MASK_FIRST('Ititan xin chào',4)
```

```
-- Xxxxan xin chào
```

4.5.11. MASK_LAST

Giấu một số ký tự đầu tiên của trường dữ liệu dạng chuỗi

Cú pháp:

MASK_LAST (truong_du_lieu String, do_dai) => String

- **truong_du_lieu:** Chuỗi được truyền vào để loại bỏ khoảng trắng.
- **do_dai:** số ký tự cần che giấu.

Ví dụ về MASK_LAST:

```
SELECT MASK_LAST('Ititan xin chào',5)
```

```
-- Ititan xinxxxxx
```

4.5.12. REPEAT

Trả về giá trị lặp lại của các ký tự trong trường dữ liệu.

Cú pháp:

REVERSE (truong_du_lieu STRING, so_lan_lap INT) => STRING

- **truong_du_lieu:** Chuỗi được truyền vào để lặp.
- **so_lan_lap:** Số lần lặp lại các ký tự

Ví dụ về REPEAT:

```
SELECT REPEAT('Ititan xin chào')
```

```
-- Ititan xin chàoItitan xin chàoItitan xin chào
```

4.5.13. REPLACE

Tìm kiếm và thay thế giá trị của một chuỗi kí tự.

Cú pháp:

REPLACE (truong_du_lieu STRING, ki_tu_1 STRING, ki_tu_2 STRING) => STRING

- **truong_du_lieu:** Chuỗi được truyền vào để thêm kí tự.
- **ki_tu_1:** Kí tự tìm kiếm.
- **ki_tu_2:** Kí tự thay thế.

Ví dụ về REPLACE:

```
SELECT REPLACE('Ititan xin chào', 'i', 'y')
```

```
-- ltytan xyn chào
```

```
SELECT REPLACE('iNet Solution', 'Solution', 'xin chào')
```

```
-- iNet xin chào
```

4.5.14. REVERSE

Đảo ngược vị trí các kí tự của chuỗi trong biểu thức.

Cú pháp:

REVERSE (truong_du_lieu STRING) => STRING

- **truong_du_lieu:** Chuỗi được truyền vào để đảo ngược.

Ví dụ về REVERSE:

```
SELECT REVERSE('Ititan xin chào')
```

```
-- oàhC nix natitl
```

4.5.15. RTRIM

Loại bỏ các kí tự khoảng trắng ở cuối chuỗi kí tự.

Cú pháp:

RTRIM (truong_du_lieu STRING) => STRING

- **truong_du_lieu:** Chuỗi được truyền vào để loại bỏ khoảng trắng.

Ví dụ về RTRIM:

```
SELECT RTRIM('Ititan xin chào  ')
```

```
-- Ititan xin chào
```

4.5.16. STRPOS

Trả về vị trí lần đầu tiên xuất hiện của kí tự trong chuỗi.

Cú pháp:

STRPOS (truong_du_lieu STRING, ki_tu STRING) => INT

- **truong_du_lieu:** Chuỗi được truyền vào để thêm kí tự.
- **ki_tu:** Kí tự tìm kiếm.

Ví dụ về STRPOS:

```
SELECT STRPOS('Ititan xin chào', 'i')
```

```
-- 3
```

```
SELECT STRPOS('iNet Solution', 'iNet')
```

```
-- 1
```

```
SELECT STRPOS('iNet Solution', 'x')
```

4.5.17. STR_LEFT

Trích xuất một phần chuỗi kí tự bên trái của cột được truyền vào.

Cú pháp:

STR_LEFT (truong_du_lieu STRING, do_dai INT) => STRING

- **truong_du_lieu:** Chuỗi được truyền vào để trích xuất.
- **do_dai:** Số kí tự muốn trích xuất.

Ví dụ về STR_LEFT:

```
SELECT STR_LEFT('Ititan xin chào',3)
```

```
-- Iti
```

4.5.18. STR_RIGHT

Trích xuất một phần chuỗi kí tự bên phải của cột được truyền vào.

Cú pháp:

RTRIM (truong_du_lieu STRING, do_dai INT) => STRING

- **truong_du_lieu:** Chuỗi được truyền vào để trích xuất.
- **do_dai:** Số kí tự muốn trích xuất.

Ví dụ về RTRIM:

```
SELECT RTRIM('Ititan xin chào',3)
```

```
-- hào
```

4.5.19. SUBSTR

Trích xuất một phần chuỗi kí tự bên của trường dữ liệu được truyền vào.

Cú pháp:

STR_LEFT (truong_du_lieu STRING, vi_tri INT, do_dai INT) => STRING

- **truong_du_lieu:** Chuỗi được truyền vào để trích xuất.
- **do_dai:** Số kí tự muốn trích xuất.

Ví dụ về SUBSTR:

```
SELECT SUBSTR('Ititan xin chào',3,4)

-- itan
```

4.5.20. UPPER

Trả về tất cả các kí tự trong một chuỗi thành chữ in hoa.

Cú pháp:

UPPER (truong_du_lieu STRING) => STRING

- **truong_du_lieu:** Chuỗi được truyền vào để in hoa.

Ví dụ về UPPER:

```
SELECT UPPER('Ititan xin chào')

-- ITITAN XIN CHÀO
```