

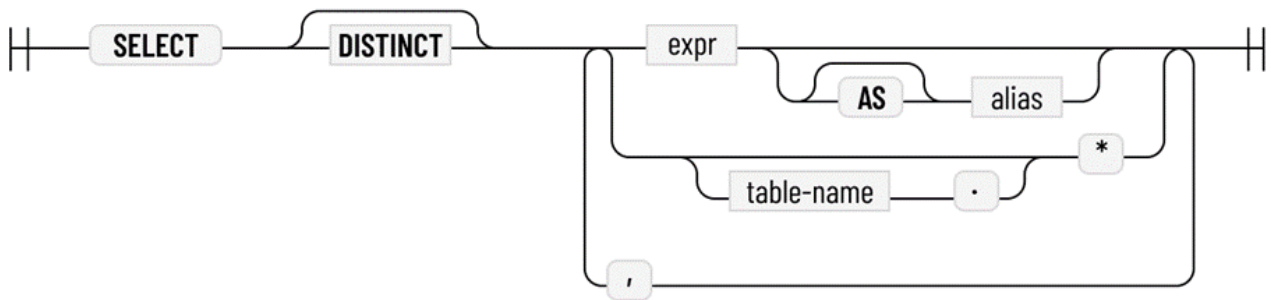
# Chương 2. CÚ PHÁP TRUY VẤN

- [2.1. SELECT](#)
- [2.2. FROM](#)
- [2.3. JOINS](#)
- [2.4. WHERE](#)
- [2.5. GROUP BY](#)
- [2.6. ORDER BY](#)
- [2.7. LIMIT](#)

## 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
```

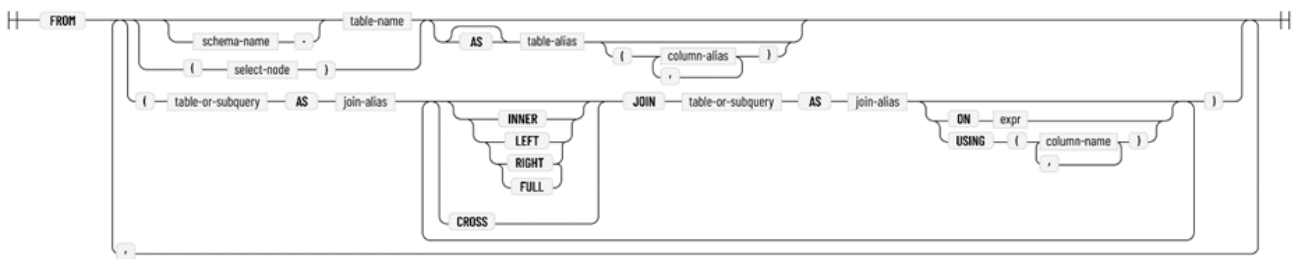
Trả về tổng số hàng trong bảng mat\_hang nhóm theo cột "**mat\_hang**":

```
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

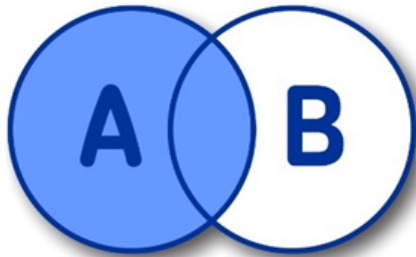
**Outer Joins & Inner Join** sử dụng các điều kiện để khớp các hàng từ hai bảng thông qua mệnh đề ON, thông thường là sử dụng các khóa, Tuy nhiên, cũng có nhiều trường hợp các hàng khớp với nhau thông qua các cột không phải khóa. Ititan cung cấp 4 loại **Outer Join** là: Left join, Right join và Full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

### b) Cross Join

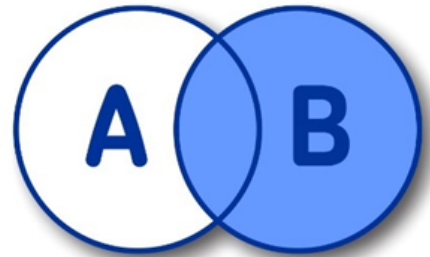
**Cross Join** là phép Joins không cần điều kiện, mỗi hàng từ bảng đầu tiên sẽ được kết hợp với tất cả các hàng từ bảng thứ hai. **Cross Join** thực hiện phép nhân Descartes (Cartesian product) giữa các bảng. Điều này có nghĩa là nó sẽ kết hợp từng hàng của bảng đầu tiên với từng hàng của bảng thứ hai.

### c) Minh họa:

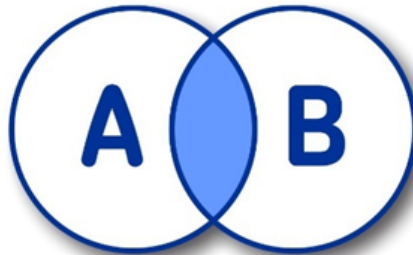
## SQL JOINS



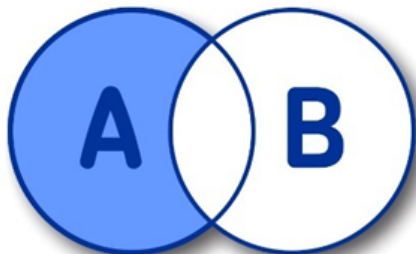
SELECT \* FROM  
A **LEFT** JOIN B  
ON A.KEY = B.KEY



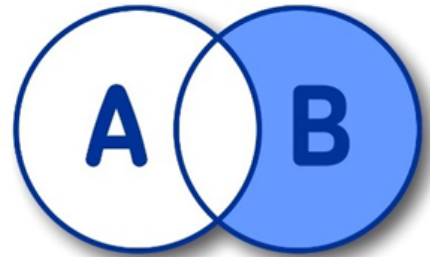
SELECT \* FROM  
A **RIGHT** JOIN B  
ON A.KEY = B.KEY



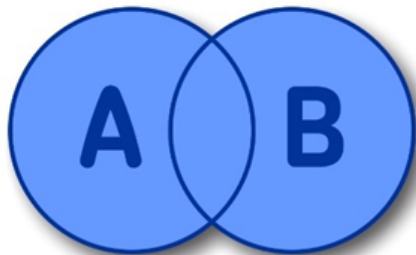
SELECT \* FROM  
A **INNER** JOIN B  
ON A.KEY = B.KEY



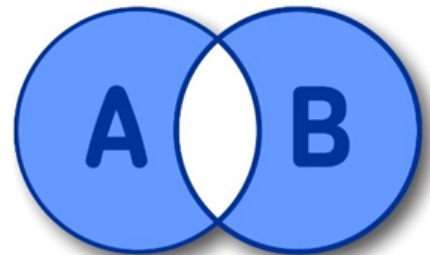
SELECT \* FROM A  
**LEFT** JOIN B  
ON A.KEY = B.KEY  
WHERE B.KEY IS NULL



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

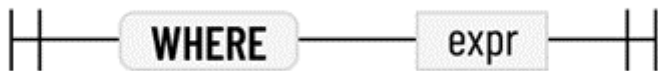


SELECT \* FROM A **FULL**  
JOIN B ON A.KEY =  
B.KEY WHERE A.KEY IS  
NULL OR B.KEY IS NULL

## 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;
```

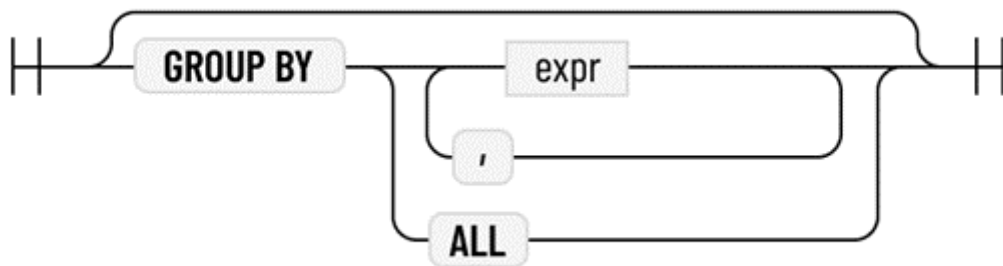


## 2.5. GROUP BY

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;
```

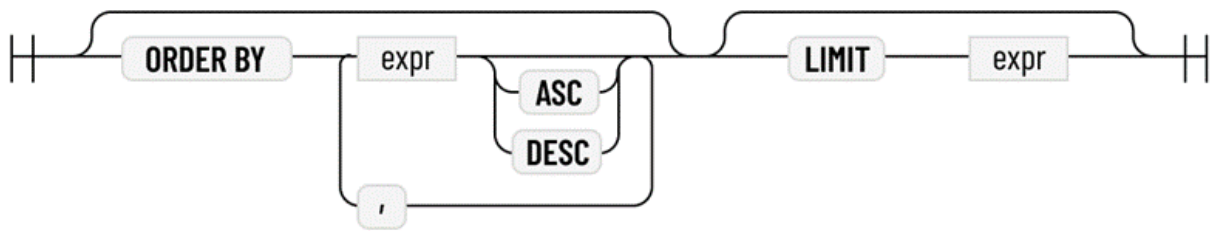
Tính trung bình thu nhập theo từng khu vực đường trong các thành phố khác nhau:

```
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;
```

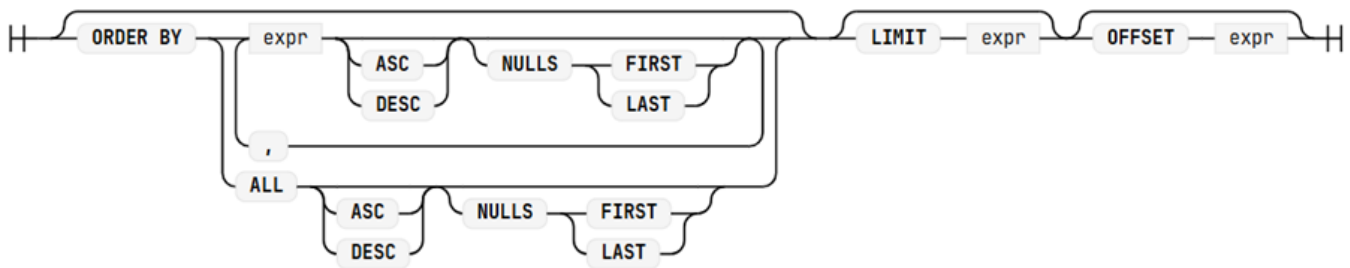
Sắp xếp thứ tự các hàng dữ liệu theo cột "**city**" theo thứ tự giảm dần:

```
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;
```