**MySql**

***Order by*** : **sắp xếp dữ liệu**

=> chiều sắp xếp : tăng dần (ASC) và giảm dần (DESC)

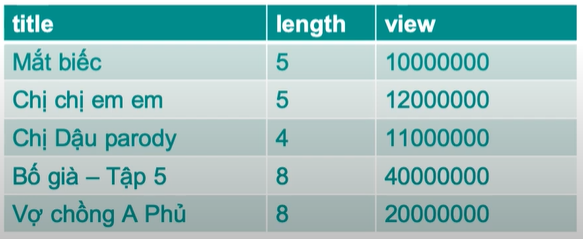
=> số lượng cột : 1 hoặc nhiều cột

**VD: sắp xếp giảm dần theo cột view của bảng FILM (1 cột)**

SELECT \*

FROM FILM

ORDER BY View DESC



**Order By theo 2 cột : Length và view**

SELECT \*

FROM FILM

ORDER BY length ASC ,view DESC



***AND/OR***

***AND:*** Thoả mãn **Tất cả** điều kiện

***OR:*** Thoả mãn **ít nhất 1** điều kiện

VD: lấy ra tất cả các bộ phim có rating = ‘PG-13’ và có length < 10



SELECT \*

FROM FILM

WHERE rating = ‘PG-13’ AND length < 10

* Thoả mãn cả 2 điều kiện => Chị dậu + vợ nhặt

SELECT \*

FROM FILM

WHERE year = 2019 OR length > 120

* Thoả mãn cả 2 điều kiện => Chị dậu + star wars

***Group by*** : **Gộp các bản ghi cùng giá trị thành 1 bản ghi duy nhất**

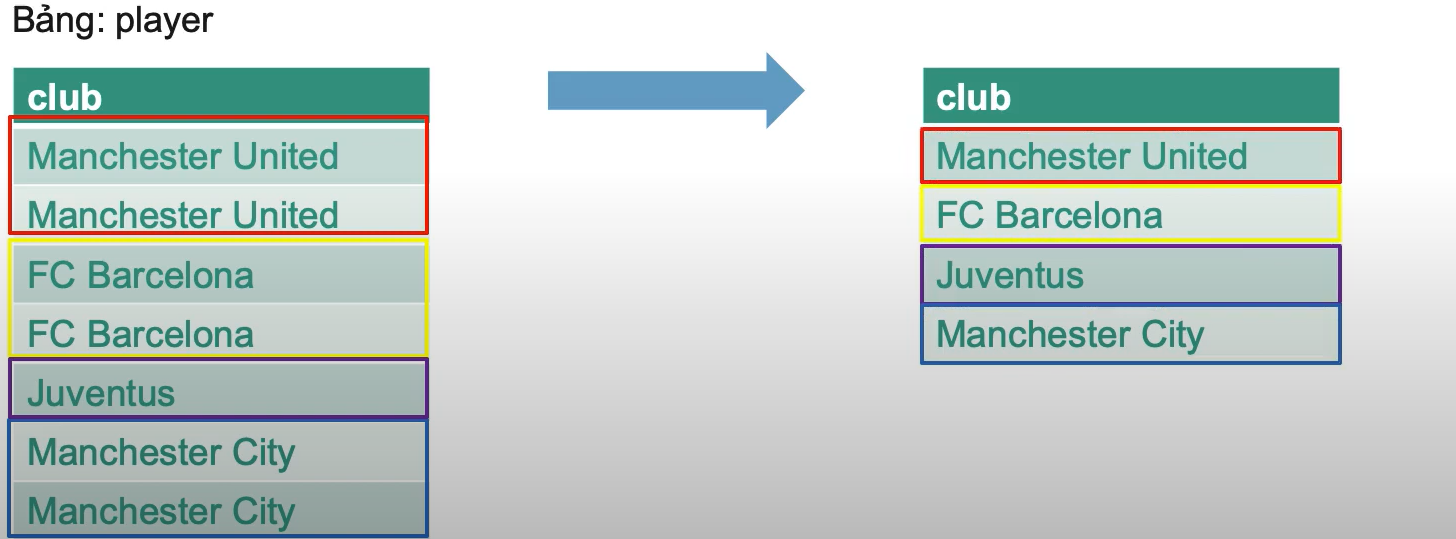
**Lấy dữ liệu từ cột club từ bảng player**

**Select** club

**From** player

**Groupby club**

* **Thường được sử dụng :** SUM-COUNT-AVG-MIN-MAX



Mỗi quốc gia có bao nhiêu cầu thủ : COUNT



SELECT nationality, count(ID)

FROM player

GROUP BY nationality

Tổng giá trị đội hình : SUM



SELECT club, SUM(value)

FROM player

GROUP BY club

Mức lương trung bình của các cầu thủ ở mỗi câu lạc bộ : AVG



SELECT club, AVG(wage)

FROM player

GROUP BY club

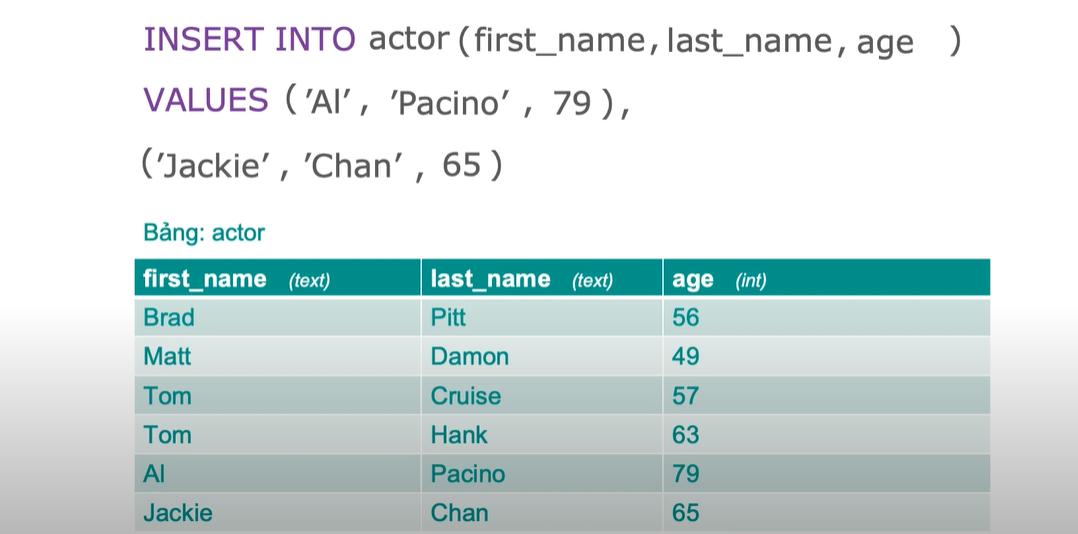
Mức lương cao nhất ở mỗi câu lạc bộ (MIN,MAX)

SELECT club,count(id) b MAX(wage)

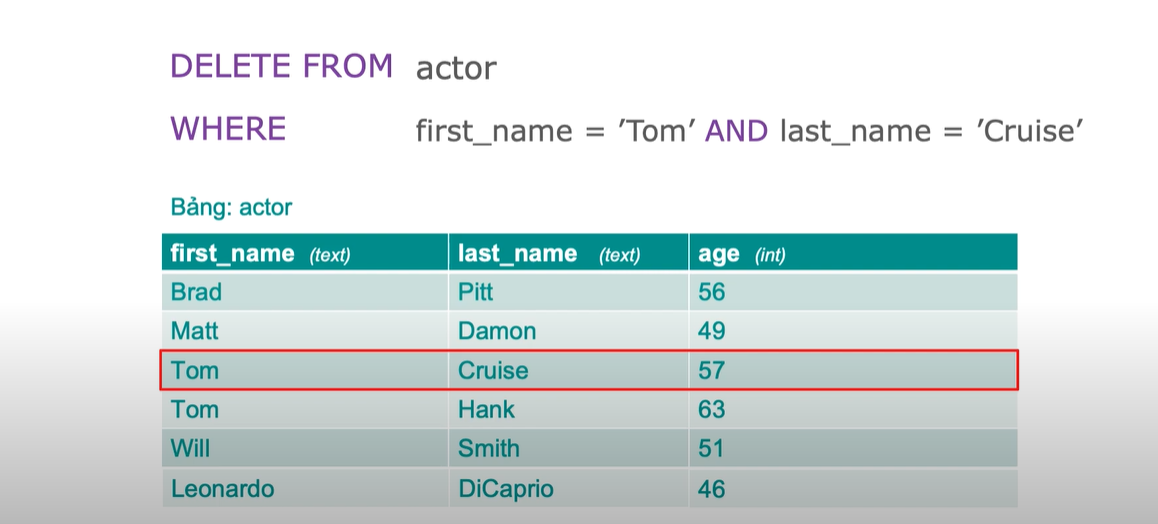
FROM player

GROUP BY club

**Insert**

****

**DELETE**

**Lưu ý :**

Viết điều kiện **Where** trước khi xoá

Delete không có **where** = **Xoá dữ liệu cả bảng**

**IN: =>** Được dùng trong mệnh đề Where

* Giá trị tồn tại trong 1 danh sách



**BETWEEN:**  Dùng trong mệnh đề where

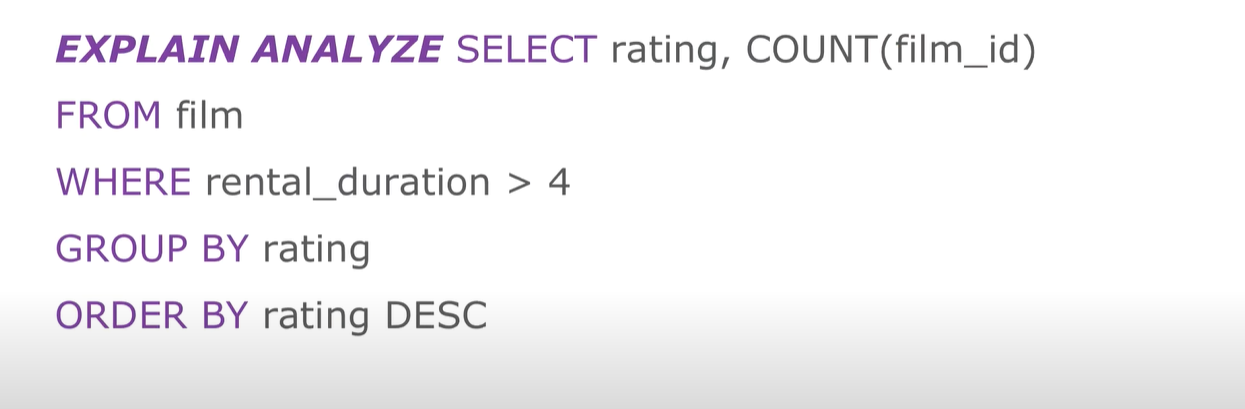
Giá trị nằm trong 1 khoảng.



**EXPLAIN ANALYZE với EXPLAIN**

1. **EXPLAIN** chỉ trả về thông tin về cách truy vấn sẽ được thực thi, bao gồm thứ tự các bước, tổng số hàng được trả về, v.v. Nó không thực sự thực hiện câu truy vấn.
2. **EXPLAIN ANALYZE** sẽ thực hiện câu truy vấn và trả về thông tin tương tự như **EXPLAIN**, nhưng nó còn bao gồm thời gian và số lượng bộ nhớ đã sử dụng cho mỗi bước. Điều này cho phép bạn đánh giá hiệu suất của câu truy vấn và tìm ra các cách để cải thiện nó.

Trong số các trường hợp, sử dụng **EXPLAIN ANALYZE** có thể là một cách tốt để tìm ra nguyên nhân gặp rủi ro hoặc chậm trễ trong truy vấn của bạn.

****

**Date truncate**

Hàm “**DATE\_TRUNC”**  dùng để làm tròn 1 giá trị ngày giờ về một đơn vị nhất định.

* Cú pháp : **DATE\_TRUNC(unit, date)**
* **Trong** đó:
  + **Unit:** là đơn vị muốn làm tròn : “second”,”minute”,”hour”,”day”,”week”,”day”,”quarter”,”year”.
  + **Date:** là giá trị ngày tháng truyền vào.

VD: date : 2022-12-31 23:59:59

Làm tròn về giờ sử dụng cú pháp:

**SELECT DATE\_TRUNC('hour', '2022-12-31 23:59:59');**

* **Kết quả: 2022-12-31 23:00:00**

**INNER JOIN – LEFT JOIN – RIGHT JOIN – FULL JOIN**

**INNER JOIN:** Giá trị trả về chỉ bao gồm những hàng có giá trị trùng hợp của cả 2 bảng.

**LEFT JOIN:** Kết quả trả về bao gồm tất cả các hàng trong bảng trái(**LEFT)** nếu có giá trị phù hợp hoặc giá trị null nếu không có giá trị phù hợp trong bảng phải.

**RIGHT JOIN:** Kết quả trả về bao gồm tất cả các hàng trong bảng phải(**RIGHT)** nếu có giá trị phù hợp hoặc giá trị null nếu không có giá trị phù hợp trong bảng trái.

**FULL JOIN :** Kếtt quả trả về bao gồm tất cả các hàng trong cả 2 bảng.

**SELECT customers.name, orders.order\_date**

**FROM customers**

**JOIN orders**

**ON customers.customer\_id = orders.customer\_id;**

**UNION**

**Union trong Mysql**  dùng để gộp 2 hay nhiều câu lệnh truy vấn select lại với nhau.

Điều kiện :

* Cùng số lượng cột
* Các cột phải cùng các kiểu dữ liệu với nhau.

VD: Union

SELECT order\_id, order\_date

FROM orders

UNION

SELECT customer\_id, date\_of\_birth

FROM customers;

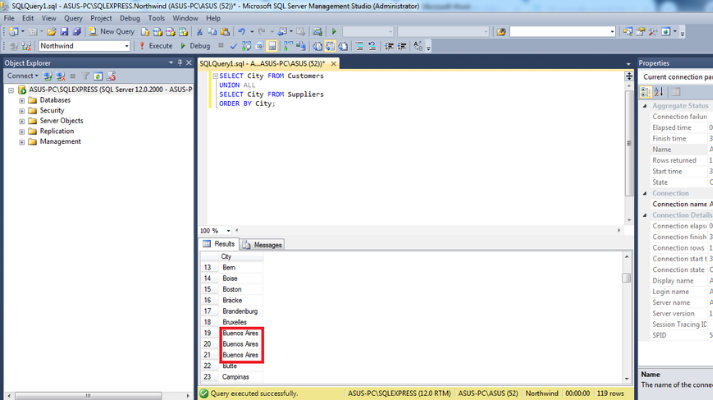
VD: Union All = > trả về giá trị bao gồm cả giá trị trùng lặp

SELECT City FROM Customers

UNION ALL

SELECT City FROM Suppliers

ORDER BY City;



**INDEX MYSQL**

Giả sử bạn có một bảng tên "customers" với các cột sau: id, first\_name, last\_name, email, và address. Bạn muốn đánh index cho cột "email". Để thực hiện việc này, bạn có thể sử dụng câu lệnh SQL sau:

**ALTER TABLE customers ADD INDEX email\_index (email);**

**Chú ý :**

Việc tìm kiếm và sắp xếp dữ liệu thường xuyên dùng câu lệnh **SELECT + ORDER BY** => nên đánh index

Làm việc với bảng lớn với dữ liệu lớn => nên dánh index

Sử dụng các lệnh **JOIN**  => nên đánh index

Sử dụng các câu lệnh where để tìm kiếm dữ liệu trong 1 cột cụ thể => nên đánh index

* Việc đánh index quá nhiều sẽ làm giảm tốc độ truy vấn của các truy vấn : **INSERT, UPDATE, DELETE.**