WHERE
VS
HAVING

Cho bảng **KetQuaHocTap** như sau:

Tên	Môn học	Điểm
A	Toán	8
A	Vật lý	5
B	Toán	4
C	Vật lý	9

WHERE

```
SELECT Tên, sum(Điểm) as
Tổng điểm
FROM KetQuaHocTap
WHERE Điểm > 5
GROUP BY Tên
```

HAVING

```
SELECT Tên, sum(Điểm) as
Tổng điểm
 FROM KetQuaHocTap
WHERE Điểm > 5
GROUP BY Tên
HAVING sum(Điểm) > 8
```

WHERE

Tên	Tổng điểm
A	8
C	9

HAVING

Tên	Tổng điểm
C	9

WHERE	HAVING
Kiểm tra các điều kiện cho các thuộc tính của bảng, trả kết quả đối chiếu với từng dòng	Giới hạn nhóm các hàng trả về trong bảng, trả kết quả đối chiếu cho các nhóm (Sum, Count, Avg,…) được tạo bởi GROUP BY.
Không thể sử dụng được với các Aggregate Functions.	Dùng với các điều kiện Aggregate Functions được define trên 1 tập hợp
Đứng trước GROUP BY	Đứng sau GROUP BY
Dùng được với Select, Update, Delete	Chỉ dùng với Select

DISTINCT
VS
GROUP BY

Cả hai mệnh đề **DISTINCT** và **GROUP BY** đều **làm giảm số lượng bản ghi trả về** trong tập kết quả **bằng cách loại bỏ các bản ghi trùng lặp.**

Nếu bạn muốn **nhóm các kết quả** của mình, hãy sử dụng **GROUP BY**, nếu bạn **chỉ** muốn **một danh sách duy nhất** của **một cột cụ thể**, hãy sử dụng **DISTINCT.**

GROUP BY được sử dụng để **nhóm các hàng mà bạn muốn tính toán**. **DISTINCT** sẽ **KHÔNG thực hiện bất kỳ phép tính nào.**

PRIMARY KEY
VS
UNIQUE KEY

	Primary Key	Unique Key
Công dụng	Nó được sử dụng để làm định danh duy nhất cho mỗi hàng trong bảng.	Xác định tính duy nhất của một hàng, nhưng không là khóa chính.
NULL	Không chấp nhận giá trị Null	Chấp nhận giá trị NULL
Số lượng khóa có thể được xác định trong bảng	Chỉ có duy nhất 1 primary key trong 1 bảng	Có thể nhiều hơn 1
Index	Tạo chỉ mục theo nhóm	Tạo chỉ mục không phân cụm

SQL SEVER
VS
POSTGRESQL

VỀ LÝ THUYẾT

SQL SEVER	POSTGRESQL
Hệ thống quản trị cơ sở dữ liệu quan hệ	Hệ thống quản trị cơ sở dữ liệu quan hệ đối tượng
Sản phẩm thương mại của Microsoft	Mã nguồn mở (Hoàn toàn miễn phí)
Chỉ chạy trên Microsoft hoặc Linux	Chạy trên hầu hết các hệ điều hành

VỀ CÂU LỆNH

TẠO DATABASE	
SQL SEVER	POSTGRESQL
CREATE DATABASE mydb	**$ createdb** mydb

XÓA DATABASE	
SQL SEVER	POSTGRESQL
DROP DATABASE mydb	**$ dropdb** mydb

ĐỔI TÊN TABLE	
SQL SEVER	POSTGRESQL
EXEC sp_rename 'SinhVien', 'HocSinh';	**ALTER TABLE** SinhVien **RENAME TO** HocSinh

ĐỔI TÊN COLUMN TRONG TABLE	
SQL SEVER	POSTGRESQL
EXEC sp_rename 'HocSinh.MaSV', 'MaHS', 'COLUMN';	**ALTER TABLE** HocSinh **RENAME COLUMN** MaSV **TO** MaHS;

THAY ĐỔI KIỂU DỮ LIỆU	
SQL SEVER	POSTGRESQL
ALTER TABLE SinhVien **ALTER COLUMN** MaHS char(10)	**ALTER TABLE** SinhVien **ALTER COLUMN** MaHS **TYPE** char(10)

THÊM MỘT COLUMN VÀO TABLE	
SQL SEVER	POSTGRESQL
ALTER TABLE HocSinh **ADD** DiaChi nvarchar(50)	**ALTER TABLE** HocSinh **ADD COLUMN** DiaChi nvarchar(50)

XÓA MỘT COLUMN KHỎI TABLE	
SQL SEVER	POSTGRESQL
ALTER TABLE HocSinh **DROP** DiaChi	**ALTER TABLE** HocSinh **DROP COLUMN** DiaChi

XÓA DỮ LIỆU TABLE	
SQL SEVER	POSTGRESQL
DELETE HocSinh **WHERE** MaHS = ‘CK101’	**DELETE FROM** HocSinh **WHERE** MaHS = ‘CK101’;

PHÉP KẾT NỐI JOIN

ĐỊNH NGHĨA

Phép kết nối (JOIN) được dùng để **lấy dữ liệu từ nhiều bảng**, **xảy ra khi 2 hoặc nhiều bảng được kết nối với nhau trong một lệnh SQL**. **Có ba loại kết nối** sau: **nội (inner join), ngoại (outer join)** và **chéo(cross join). Kết ngoại được chia ra** thêm thành **kết ngoại bên trái** (left outer join), **kết ngoại bên phải** (right outer join), và **kết ngoại đủ** (full outer join).

PHÉP KẾT NỘI

Phép kết nội (inner join) thực chất là **tìm giao của hai bảng dữ liệu.** Đây là loại kết hợp thường được dùng nhất và được xem như là phép kết hợp mặc định.

Cần đặc biệt chú ý khi kết hợp những bảng trên những cột mà có thể là NULL vì giá trị NULL sẽ không bao giờ có tương ứng.

Bảng "employee"

LastName	DepartmentID
Smith	34
Jones	33
Robinson	34
Jasper	36
Steinberg	33
Rafferty	31

Bảng "department"

DepartmentName	DepartmentID
Sales	31
Engineering	33
Clerical	34
Marketing	35

VÍ DỤ VỀ PHÉP KẾT NỘI

```
SELECT *
FROM employee
INNER JOIN department
ON employee.DepartmentID = department.DepartmentID
```

```
+----------+--------------+----------------+--------------+
| LastName | DepartmentID | DepartmentName | DepartmentID |
+----------+--------------+----------------+--------------+
| Smith    |           34 | Clerical       |           34 |
| Jones    |           33 | Engineering    |           33 |
| Robinson |           34 | Clerical       |           34 |
| Steinberg|           33 | Engineering    |           33 |
| Rafferty |           31 | Sales          |           31 |
+----------+--------------+----------------+--------------+
```

PHÉP KẾT NGOẠI BÊN TRÁI

Phép kết ngoại bên trái (left outer join) trả về tất cả các bản ghi từ bảng bên trái và các bản ghi phù hợp từ bảng bên phải. Nếu mệnh đề ON không khớp với bản ghi nào trong bảng bên phải thì LEFT JOIN sẽ vẫn trả về một hàng trong kết quả, nhưng giá trị là NULL trong mỗi cột từ bảng bên phải.

Nó **trả về tất cả những giá trị từ bản bên trái** + **những giá trị tương ứng với bảng bên phải hoặc là null** (khi những giá trị ở bảng bên phải không tương ứng)

VÍ DỤ VỀ PHÉP KẾT NGOẠI BÊN TRÁI

```
SELECT *
FROM employee
LEFT OUTER JOIN department
ON employee.DepartmentID = department.DepartmentID
```

```
+----------+-------------+----------------+-------------+
| LastName | DepartmentID | DepartmentName | DepartmentID |
+----------+-------------+----------------+-------------+
| Smith |  34 | Clerical  |  34 |
| Jones |  33 | Engineering  |  33 |
| Robinson |  34 | Clerical  |  34 |
| Jasper  |  36 | NULL  |  NULL |
| Steinberg |  33 | Engineering  |  33 |
| Rafferty |  31 | Sales  |  31 |
+----------+-------------+----------------+-------------+
```

PHÉP KẾT NGOẠI BÊN PHẢI

Phép kết ngoại bên phải (right outer join) trả về tất cả các bản ghi từ bảng bên PHẢI và các bản ghi phù hợp từ bảng bên TRÁI. Nếu mệnh đề ON không khớp với bản ghi nào trong bảng bên trái thì RIGHT JOIN sẽ vẫn trả về một hàng trong kết quả, nhưng giá trị là NULL trong mỗi cột từ bảng bên trái.

Nó **trả về tất cả những giá trị** từ **bảng bên phải** + **giá trị tương ứng** từ **bảng bên trái (hoặc null)**

VÍ DỤ VỀ PHÉP KẾT NGOẠI BÊN PHẢI

```
SELECT *
FROM employee
RIGHT OUTER JOIN department
ON employee.DepartmentID = department.DepartmentID
```

LastName	DepartmentID	DepartmentName	DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Steinberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	Marketing	35

PHÉP KẾT NGOẠI ĐỦ

Phép kết ngoại đủ (full outer join) sự **kết hợp** của cả **kết quả của cả phép kết ngoại bên trái và phép kết ngoại bên phải.** Phép kết nối này **đưa ra bản ghi của cả hai bảng dữ liệu,** và **lấp đầy những dòng tương ứng bị thiếu của cả hai phía bằng** giá trị **NULL.**

VÍ DỤ VỀ PHÉP KẾT NGOẠI ĐỦ

```
SELECT *
FROM employee
FULL OUTER JOIN department
ON employee.DepartmentID = department.DepartmentID
```

```
+-----------+--------------+----------------+--------------+
| LastName | DepartmentID | DepartmentName | DepartmentID |
+-----------+--------------+----------------+--------------+
| Smith |  34 | Clerical  |  34 |
| Jones |  33 | Engineering  |  33 |
| Robinson |  34 | Clerical  |  34 |
| Jasper  |  36 | NULL  |  NULL |
| Steinberg |  33 | Engineering  |  33 |
| Rafferty |  31 | Sales  |  31 |
| NULL |  NULL | Marketing |  35 |
+-----------+--------------+----------------+--------------+
```

PHÉP KẾT CHÉO

Phép kết chéo (cross join) sẽ nối mỗi bản ghi từ bảng đầu tiên với mỗi bản ghi từ bản thứ hai. Nói cách khác cross join trả về tích Descartes các bản ghi từ cả 2 bảng.

Không giống như các phép nối đã đề cập ở trên, cross join không thiết lập mối quan hệ giữa các bảng được join.

```
SELECT *
FROM employee CROSS JOIN department;
```

LastName	DepartmentID	DepartmentName	DepartmentID
Smith	34	Sales	31
Smith	34	Engineering	33
Smith	34	Clerical	34
Smith	34	Marketing	35
Jones	33	Sales	31
Jones	33	Engineering	33
Jones	33	Clerical	34
Jones	33	Marketing	35
Robinson	34	Sales	31
Robinson	34	Engineering	33
Robinson	34	Clerical	34
Robinson	34	Marketing	35
Jasper	36	Sales	31
Jasper	36	Engineering	33
Jasper	36	Clerical	34
Jasper	36	Marketing	35
Steinberg	33	Sales	31
Steinberg	33	Engineering	33
Steinberg	33	Clerical	34
Steinberg	33	Marketing	35
Rafferty	31	Sales	31
Rafferty	31	Engineering	33
Rafferty	31	Clerical	34
Rafferty	31	Marketing	35


Thanks~