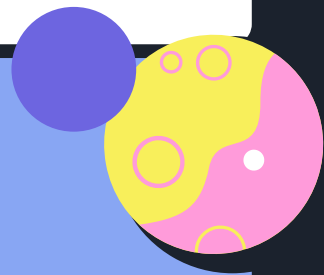




WHERE
VS
HAVING



Cho bảng **KetQuaHocTap** như sau:

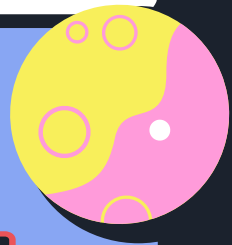
Tên	Môn học	Điểm
A	Toán	8
A	Vật lý	5
B	Toán	4
C	Vật lý	9

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
```

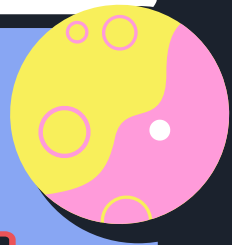


WHERE

Tên	Tổng điểm
A	8
C	9

HAVING

Tên	Tổng điểm
C	9





WHERE

Lọc từng dòng riêng lẻ

Không thể sử dụng được với các hàm tổng hợp(Sum, Count, Average,...).

Đứng trước GROUP BY

Dùng được với Select, Update, Delete

HAVING

Lọc theo từng nhóm

Được dùng với các hàm tổng hợp được define trên 1 tập hợp


Đứng sau GROUP BY

Chỉ dùng với Select






The image shows a stylized window frame with a white title bar at the top containing three colored circles (red, yellow, green). The main content area is dark blue with rounded corners. At the bottom of the window is a light blue bar containing a pink pill-shaped button, a white pill-shaped button, and a progress bar with green and white segments. A small red heart icon is located at the bottom right of the dark blue content area.

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.




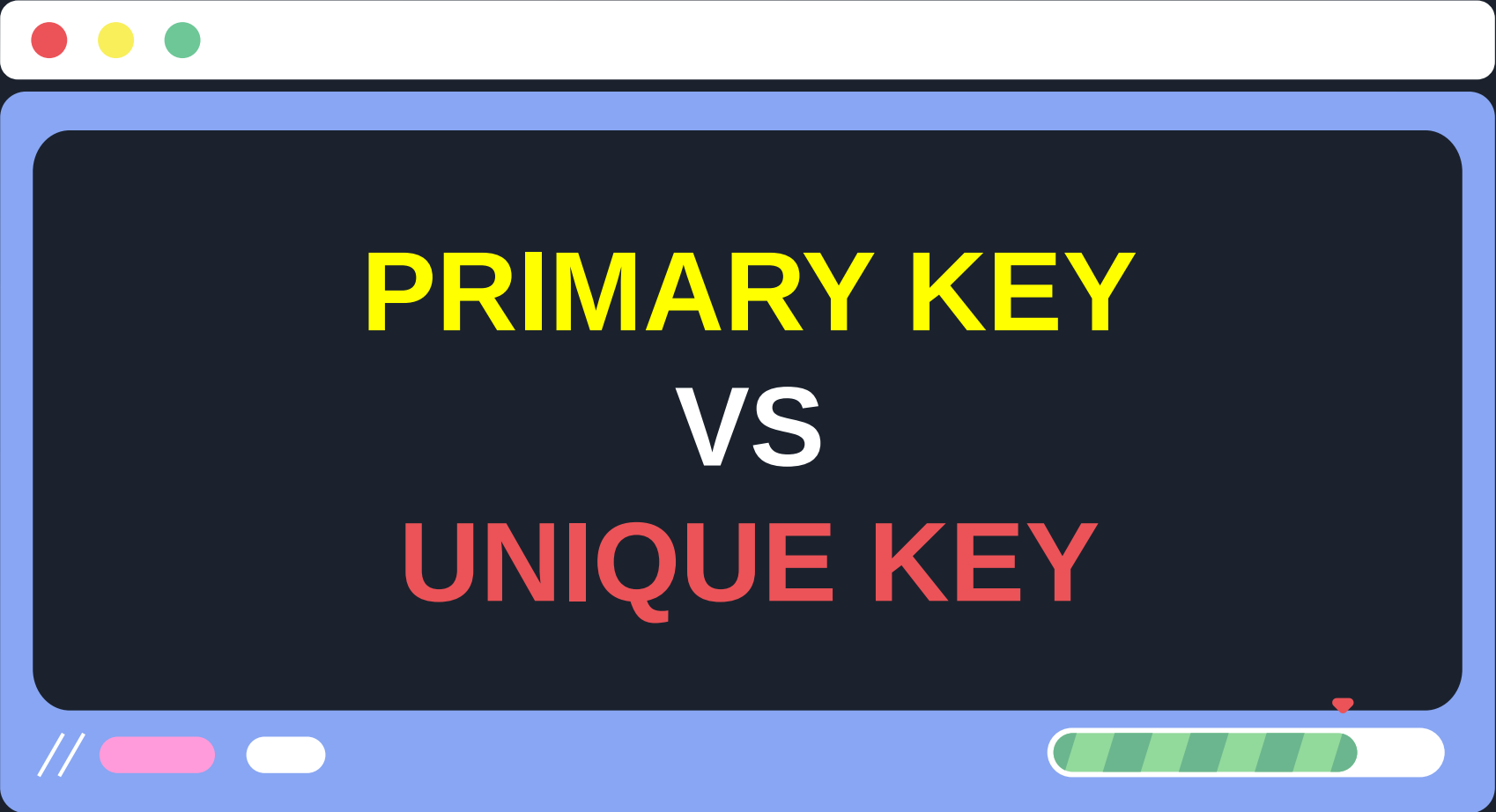


Nếu bạn muốn **nhóm các kết quả** của mình, hãy sử dụng **GROUP BY**, nếu bạn **chỉ** muốn **một danh sách duy nhất** của một cột cụ thể, hãy sử dụng **DISTINCT**.

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

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





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 POSTGRES SQL

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

CREATE DATABASE mydb

POSTGRESQL

\$ createdb mydb

XÓA DATABASE

SQL SEVER

DROP DATABASE mydb

POSTGRESQL

\$ dropdb mydb



ĐỔI TÊN TABLE

SQL SEVER

```
EXEC sp_rename 'SinhVien',  
'HocSinh';
```

POSTGRESQL

```
ALTER TABLE SinhVien  
RENAME TO HocSinh
```

ĐỔI TÊN COLUMN TRONG TABLE

SQL SEVER

```
EXEC sp_rename 'HocSinh.MaSV',  
'MaHS', 'COLUMN';
```

POSTGRESQL

```
ALTER TABLE HocSinh  
RENAME COLUMN MaSV  
TO MaHS;
```



THAY ĐỔI KIỂU DỮ LIỆU

SQL SEVER

```
ALTER TABLE SinhVien  
ALTER COLUMN MaHS char(10)
```

POSTGRESQL

```
ALTER TABLE SinhVien  
ALTER COLUMN MaHS TYPE  
char(10)
```

THÊM MỘT COLUMN VÀO TABLE

SQL SEVER

```
ALTER TABLE HocSinh  
ADD DiaChi nvarchar(50)
```

POSTGRESQL

```
ALTER TABLE HocSinh  
ADD COLUMN DiaChi nvarchar(50)
```



XÓA MỘT COLUMN KHỎI TABLE

SQL SEVER

```
ALTER TABLE HocSinh  
DROP DiaChi
```

POSTGRESQL

```
ALTER TABLE HocSinh  
DROP COLUMN DiaChi
```

XÓA DỮ LIỆU TABLE

SQL SEVER

```
DELETE HocSinh  
WHERE MaHS = 'CK101'
```

POSTGRESQL

```
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 NỐI



Phép kết nối (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 NỐI

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

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

```
+-----+-----+-----+-----+  
| 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
```

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

```
+-----+-----+-----+-----+  
| 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
```

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

```
+-----+-----+-----+-----+  
| 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

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

```
+-----+-----+-----+-----+
| 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



FUNCTION

SYNTAX

```

CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
    [ RETURNS rettype
      | RETURNS TABLE ( column_name column_type [, ...] ) ]
{ LANGUAGE lang_name
  | TRANSFORM { FOR TYPE type_name } [, ... ]
  | WINDOW
  | { IMMUTABLE | STABLE | VOLATILE }
  | [ NOT ] LEAKPROOF
  | { CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT }
  | { [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER }
  | PARALLEL { UNSAFE | RESTRICTED | SAFE }
  | COST execution_cost
  | ROWS result_rows
  | SUPPORT support_function
  | SET configuration_parameter { TO value | = value | FROM CURRENT }
  | AS 'definition'
  | AS 'obj_file', 'link_symbol'
  | sql_body
} ...

```



```
create [or replace] function function_name(param_list)
    returns return_type
    language plpgsql
as
$$
declare
-- variable declaration
begin
    -- logic
end;
$$
```

MIÊU TẢ CÁC THUỘC TÍNH



Phần tiêu đề của hàm

CREATE FUNCTION: dùng để tạo một function mới.

CREATE OR REPLACE FUNCTION: dùng để tạo mới 1 function hoặc thay thế 1 function hiện có.

MIÊU TẢ CÁC THUỘC TÍNH



Phần tiêu đề của hàm

fuction_name: Tên của hàm cần tạo

param_list: Danh sách các tham số của hàm được đặt trong dấu (). Một hàm có thể có 0 hoặc nhiều tham số.

return_type: Chỉ định kiểu dữ liệu trả về sau từ khóa returns

plpgsql: Tên của ngôn ngữ mà hàm triển khai. PostgreSQL hỗ trợ nhiều ngôn ngữ thủ tục.

MIÊU TẢ CÁC THUỘC TÍNH



Phần tiêu đề của hàm

Tên của hàm mới không được trùng với bất kỳ hàm hiện có nào có cùng kiểu đối số đầu vào trong cùng một lược đồ.

PostgreSQL cho phép nhiều hàm chia sẻ cùng một tên miễn là chúng có các đối số khác nhau (Gọi là overloading).

MIÊU TẢ CÁC THUỘC TÍNH



Phần thân của hàm

Sử dụng hằng số chuỗi được trích dẫn bằng dấu đô la, bắt đầu bằng \$\$ và kết thúc bằng \$\$.

Giữa \$\$ có thể đặt một khối chứa phần khai báo và logic của hàm.

MIÊU TẢ CÁC THUỘC TÍNH



Phần thân của hàm

Trong phần khai báo chúng ta sẽ khai báo một tên biến dùng để lưu trữ thông tin được chọn từ 1 bảng chỉ định.

Trong phần nội dung của khối chúng ta sẽ sử dụng các câu lệnh truy vấn để chọn ra những thông tin cần thiết sau đó gán kết quả cho biến đã chỉ định bên trên.

Ở cuối khối, câu lệnh return được sử dụng để trả về tệp lưu kết quả đã khai báo bên trên.

```
create or replace function function_name (  
    parameter_list  
)  
returns table ( column_list )  
language plpgsql  
as $$  
declare  
    -- variable declaration  
begin  
    -- body  
end; $$
```

MIÊU TẢ CÁC THUỘC TÍNH
























Thay vì trả về một giá trị đơn lẻ, cú pháp này cho phép trả về một bảng có danh sách cột được chỉ định.







Hàm trả về một truy vấn là kết quả của một câu lệnh lựa chọn. Lưu ý rằng các cột trong tập hợp kết quả phải giống với các cột trong bảng được xác định sau mệnh đề **returns table**.

VÍ DỤ VỀ FUNCTION

BẢNG HỌC SINH

		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
		<input type="text" value="mahs"/>	<input type="text" value="integer"/> 	<input type="text"/>	<input type="text"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="text"/>
		<input type="text" value="tenhs"/>	<input type="text" value="character varying"/> 	<input type="text" value="50"/>	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
		<input type="text" value="gioitinh"/>	<input type="text" value="character varying"/> 	<input type="text" value="10"/>	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
		<input type="text" value="diachi"/>	<input type="text" value="character varying"/> 	<input type="text" value="50"/>	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
		<input type="text" value="namsinh"/>	<input type="text" value="integer"/> 	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
		<input type="text" value="diemhk1"/>	<input type="text" value="integer"/> 	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
		<input type="text" value="diemhk2"/>	<input type="text" value="integer"/> 	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>

DỮ LIỆU BẢNG HỌC SINH


	mahs [PK] integer 	tenhs character varying (50) 	gioitinh character varying (10) 	diachi character varying (50) 	namsinh integer 	diemhk1 integer 	diemhk2 integer 
1	1	Nguyễn Văn A	Nam	Tp.HCM	2002	9	7
2	2	Lê Thị B	Nữ	Cà Mau	2001	8	9
3	3	Đỗ Văn C	Nam	An Giang	2004	7	8
4	4	Nguyễn Văn E	Nữ	Tiền Giang	2003	6	5
5	5	Lê Hồng H	Nam	Hậu Giang	2003	5	6
6	6	Nguyễn Văn G	Nữ	Vĩnh Long	2001	3	4
7	7	Phan Hoài H	Nam	Tp.HCM	2000	2	3
8	8	Trần Thị I	Nữ	Hà Nội	2002	5	6
9	9	Trương Văn K	Nam	Gia Lai	2000	10	9
10	10	Phạm Văn L	Nữ	Kon Tum	2002	7	10
11	11	Nguyễn Văn M	Nam	Tiền Giang	2004	2	5
12	12	Lê Thị N	Nữ	Long An	2000	3	7

FUNCTION ĐẾM TỔNG SỐ HỌC SINH

```
create OR REPLACE function total_hocsinh()  
returns int  
language plpgsql  
as  
$$  
declare  
    total integer;  
begin  
    select count(*)  
    into total  
    from hocsinh;  
    return total;  
end;  
$$;
```

KIỂM TRA FUNCTION VỪA TẠO

```
select total_hocsinh();
```


	total_hocsinh integer 
1	12

FUNCTION ĐẾM SỐ HỌC SINH CÓ DIEMHK1 NĂM TRONG KHOẢNG MONG MUỐN

```
create or REPLACE function get_hocsinh_count_diemhk1(diemhk1_from int, diemhk1_to int)
returns int
language plpgsql
as
$$
declare
    hocsinh_count integer;
begin
    select count(*)
    into hocsinh_count
    from hocsinh
    where diemhk1 between diemhk1_from and diemhk1_to;
    return hocsinh_count;
end;
$$;
```

KIỂM TRA FUNCTION VỪA TẠO

```
SELECT get_hocsinh_count_diemhk1(6,9);
```

	get_hocsinh_count_diemhk1 integer	
1		5

HÀM TÍNH ĐIỂM TRUNG BÌNH CỦA TỪNG HỌC SINH

```
CREATE OR REPLACE FUNCTION tinh_diem_trung_binh_tung_hoc_sinh()  
RETURNS TABLE  
(  
    mahs int,  
    tenhs varchar(50),  
    diemtb numeric  
)  
LANGUAGE plpgsql  
AS $$  
BEGIN  
    RETURN QUERY SELECT  
        hocsinh.mahs, hocsinh.tenhs, round((hocsinh.diemhk1 + hocsinh.diemhk2)/2.0, 2) AS diemtb  
    FROM  
        hocsinh;  
END;  
$$
```

KIỂM TRA FUNCTION VỪA TẠO

```
SELECT * FROM tinh_diem_trung_binh_tung_hoc_sinh();
```




	mahs integer 	tenhs character varying 	diemtb numeric 
1	1	Nguyễn Văn A	8.00
2	2	Lê Thị B	8.50
3	3	Đỗ Văn C	7.50
4	4	Nguyễn Văn E	5.50
5	5	Lê Hồng H	5.50
6	6	Nguyễn Văn G	3.50
7	7	Phan Hoài H	2.50
8	8	Trần Thị I	5.50
9	9	Trương Văn K	9.50
10	10	Phạm Văn L	8.50
11	11	Nguyễn Văn M	3.50
12	12	Lê Thị N	5.00

HÀM TRẢ VỀ NHỮNG SINH VIÊN TRONG TÊN CÓ CHỮ “L” VÀ CÓ ĐIỂM TRUNG BÌNH >6

```
CREATE OR REPLACE FUNCTION tinh_diem_trung_binh_lon_hon_6()
RETURNS TABLE
(
    mahs int,
    tenhs varchar(50),
    diemtb numeric
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY SELECT
        hocsinh.mahs, hocsinh.tenhs, round((hocsinh.diemhk1 + hocsinh.diemhk2)/2.0, 2) AS diemtb
    FROM
        hocsinh
    WHERE
        round((hocsinh.diemhk1 + hocsinh.diemhk2)/2.0, 2) > 6
        AND
        hocsinh.tenhs LIKE '%L%';
END;
$$
```

KIỂM TRA FUNCTION VỪA TẠO

```
SELECT * FROM tinh_diem_trung_binh_lon_hon_6();
```

	mahs integer 	tenhs character varying 	diemtb numeric 
1	2	Lê Thị B	8.50
2	10	Phạm Văn L	8.50

HÀM TRẢ VỀ NHỮNG SINH VIÊN TRONG TÊN CÓ CHỮ “N” CÓ ĐIỂM TRUNG BÌNH >5 VÀ NĂM SINH >2000

```
CREATE OR REPLACE FUNCTION danh_sach_hs_2000_co_ten_N_va_diem_tb_lon_hon_5()  
RETURNS TABLE (  
    tenhs varchar(50),  
    namsinh int,  
    diachi varchar(100),  
    diemtb numeric  
)  
LANGUAGE plpgsql  
AS $$  
BEGIN  
    RETURN QUERY SELECT  
        hocsinh.tenhs, hocsinh.namsinh, hocsinh.diachi, round((hocsinh.diemhk1 + hocsinh.diemhk2)/2.0,2) AS diemtb  
    FROM  
        hocsinh  
    WHERE  
        hocsinh.namsinh > 2000 AND  
        hocsinh.tenhs LIKE '%N%' AND  
        (hocsinh.diemhk1 + hocsinh.diemhk2)/2.0 > 5;  
END;  
$$
```

KIỂM TRA FUNCTION VỪA TẠO

```
SELECT * FROM danh_sach_hs_2000_co_ten_N_va_diem_tb_lon_hon_5();
```

	tenhs character varying 🔒	namsinh integer 🔒	diachi character varying 🔒	diemtb numeric 🔒
1	Nguyễn Văn A	2002	Tp.HCM	8.00
2	Nguyễn Văn E	2003	Tiền Giang	5.50

YÊU CẦU

- Hiển thị các case bị lỗi nếu Auto Delete không xóa được (hiển thị đầy đủ đường dẫn đến folder/file bị lỗi)
- Cho phép filter theo dự án
- Hiển thị thời gian chạy và gặp sự cố
- Hiển thị phân loại Auto Delete (delete data trên database/ export/ import)
- Hiển thị status xử lý theo từng trường hợp (Not Deleted/ Deleted on DD/MM/YY)
- Cho phép export danh sách, dữ liệu
- Hiển thị root cause nếu có

```
CREATE TABLE IF NOT EXISTS production.report_auto_delete_log
(
    error_case_id integer NOT NULL DEFAULT nextval('production.report_auto_delete_log_error_case_id_seq'::regclass),
    project_name character varying COLLATE pg_catalog."default",
    filepath character varying COLLATE pg_catalog."default",
    filename character varying COLLATE pg_catalog."default",
    auto_delete_type character varying COLLATE pg_catalog."default",
    processing_status character varying COLLATE pg_catalog."default",
    created_time timestamp without time zone,
    error_time timestamp without time zone,
    error_cause character varying COLLATE pg_catalog."default",
    last_update_time timestamp without time zone,
    history character varying COLLATE pg_catalog."default",
    status_delete integer DEFAULT 0,
    CONSTRAINT report_auto_delete_log_pkey PRIMARY KEY (error_case_id)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS production.report_auto_delete_log
    OWNER to ps_autodelete;
```


YÊU CẦU

Từ file **given** cho sẵn **compare** với **tbl_given_single_de** trong database của dự án lọc ra **những bản ghi của file given không nằm trong tbl_given_single_de**

CÁC CỘT VÀ DỮ LIỆU TRONG FILE COMPARE_GIVEN

data_value varchar(255) ▲	amount varchar(255) ▲	is_exist int4 ▲
aanna	(NULL)	(NULL)
abigail	(NULL)	(NULL)
abraham	(NULL)	(NULL)
ada	(NULL)	(NULL)
adalbero	(NULL)	(NULL)
adalbert	(NULL)	(NULL)

CÁC CỘT VÀ DỮ LIỆU TRONG BẢNG TBL_GIVEN_SINGLE_DE

 id	int4	▲	data_value	varchar	▲	data_value_lower	varchar	▲
14990			Johann			johann		
14991			Johannes			johannes		
14992			Joseph			joseph		
14993			Anna			anna		
14994			Carl			carl		
14995			Maria			maria		

```
---Compare given với table tbl_given_single_de  
select LOWER(data_value) from compare_given  
except  
select data_value_lower from tbl_given_single_de
```

lower ▲

johannn

ludwiga

thaeresia

stranen

meysenbourg

chatrine

```

CREATE
OR
REPLACE
FUNCTION db_23_0853_ancestry_2023.sp_check_2_consecutive_vowel(character varying, character varying) RETURNS refcursor LANGUAGE plpgsql
SET
search_path TO 'db_23_0853_ancestry_2023' AS $function $DECLARE var_sql varchar;

cursor_result refcursor := 'cursor_result';

BEGIN
/*
$1: batch name, empty mean all
$2: fieldname
*/
var_sql := '
    SELECT data_value as ' || quote_ident($2) || ', amount
    FROM sp_split_single_word_by_field($1,$2)
    WHERE data_value ~* ''[aeiou]{2}''gg
    ORDER BY 2 DESC, 1
';

OPEN cursor_result FOR
EXECUTE
(var_sql) USING $1,
    $2;

RETURN
    cursor_result;

END;

$function $

```

```

CREATE OR REPLACE FUNCTION db_23_0853_ancestry_2023.sp_check_n_consecutive_vowel(
batch_name character varying,
field_name character varying,
num_vowels integer
)
RETURNS refcursor
LANGUAGE plpgsql
SET search_path TO 'db_23_0853_ancestry_2023'
AS $function$
DECLARE
    var_sql varchar;
    cursor_result refcursor := 'cursor_result';
BEGIN
    /*
    $1: batch name, empty mean all
    $2: fieldname
    $3: number of consecutive vowels to check
    */
    var_sql := '
        SELECT data_value as ' || quote_literal(field_name) || ', amount
        FROM sp_split_single_word_by_field($1,$2)
        WHERE data_value ~* ' '([aeiou]{' || num_vowels || '})' '
        ORDER BY 2 DESC, 1
    ';
    OPEN cursor_result FOR
    EXECUTE (var_sql) USING batch_name, field_name;
    RETURN cursor_result;
END;
$function$

```

Tham số "num_vowels" được thêm vào để truyền số lượng nguyên âm/phụ âm cần lọc vào function. Biểu thức chính quy '[aeiou]{2}' đã được thay thế bằng "([aeiou]{' || num_vowels || '})" để lọc các từ có số lượng nguyên âm/phụ âm liên tiếp bằng số lượng được truyền vào tham số "num_vowels".

Thanks~

