

Objective: At the end of this lab session you should be able to write the SELECT command with HAVING clause for single table queries.

Section 1

The HAVING clause enables you to specify conditions on the logical groups created by the GROUP BY clause. So always HAVING clause should come after a GROUP BY clause. Since HAVING clause checks for a condition that applies for a group, always it includes an Aggregate Function. Conditions given in the HAVING clause filters the groups that qualify to appear in the final result.

The difference between WHERE and HAVING clauses.

The **WHERE** clause can be used to specify conditions that need to be checked **for each and every row in a table**, whereas the **HAVING** clause checks for conditions that need to be checked for **each and every logical group** created by the GROUP BY clause.

Syntax

```
SELECT Column1, Column2, aggregate_function (aggregate Column)
FROM table
[WHERE conditions]
GROUP BY Column1, Column2
HAVING condition;
```

- **Column1, column2,**

Columns that are not encapsulated within an aggregate function must be included in the GROUP BY Clause before the HAVING clause.

- **aggregate_function**

This is an aggregate function such as the SUM, COUNT, MIN, MAX, or functions.

- **aggregate_column**

This is the column or expression that the aggregate_function will be used on.

- **Table**

The tables that you wish to retrieve records from.

- **WHERE condition**

Optional. These are the conditions for the records to be selected.

- **HAVING condition**

This is a further condition applied on the logical groups. Only those groups whose condition evaluates to TRUE will be included in the result set.

When SQL statements have both a WHERE clause and HAVING clause, WHERE clause is applied first, then the selected rows are grouped using GROUP BY clause, and finally, the groups filtered according to the condition in HAVING clause.

Example:

What are the courses which offer more than 2 modules?

First of all, you need to identify the table that we need to find the answer. Offers table includes the necessary data as given in figure 1.

	CID	Mcode	Accademic_y...	Semest...
1	CS	IE3082	Y3	1
2	CS	IE3102	Y3	2
3	CS	IE4042	Y4	1
4	CS	IE4052	Y4	1
5	CSNE	IE3030	Y3	1
6	CSNE	IE3070	Y3	1
7	CSNE	IE3080	Y3	2
8	CSNE	IE4040	Y4	1
9	DS	IT3011	Y3	1
10	DS	IT3051	Y3	2
11	DS	IT3071	Y3	2
12	DS	IT4011	Y4	1
13	ISE	IE2051	Y2	2
14	ISE	IE3051	Y3	1
15	ISE	IE3081	Y3	2
16	ISE	IE4011	Y4	1
17	IT	IT1010	Y1	2
18	IT	IT1050	Y1	2
19	IT	IT1100	Y1	2
20	IT	IT2050	Y2	1
21	SE	IT3100	Y3	2
22	SE	SE3...	Y3	2
23	SE	SE3...	Y3	2
24	SE	SE4...	Y4	1

Figure 1: Data in the Offers table

Then to identify the number of modules offered by each course you need to group the records of Offers table using the GROUP BY clause based on the equality of the CID. Then a logical group is created including the modules offered by each course. Then you can check for groups which have more than two rows using the HAVING clause.

Following SQL query to get the answer for the above question.

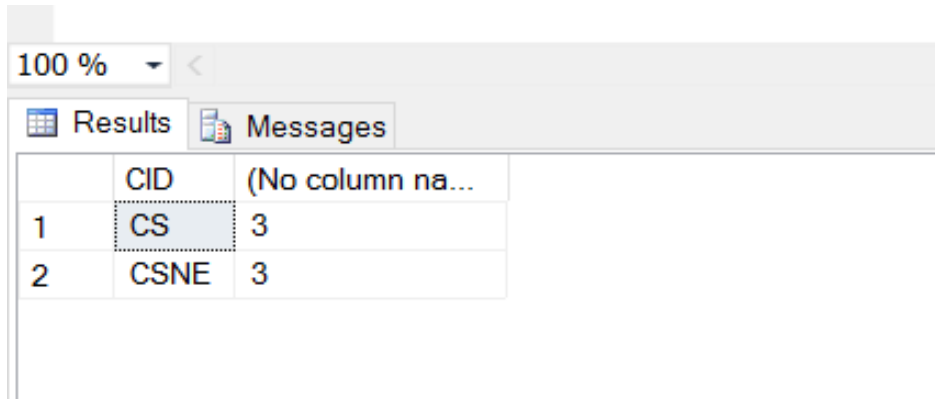
```

SELECT CID, COUNT(Mcode)
FROM Offers
WHERE Semester = 1
GROUP BY CID
HAVING COUNT(Mcode) > 2;

```

IT1090 – Information Systems and Data Modeling**Semester 2**

Then you can see following as the result for your above query, displaying the details of the courses that offer more than 2 modules.



100 % <

Results Messages

	CID	(No column na...
1	CS	3
2	CSNE	3

Section 2

- a) Display the number of students for each course? List the Course_ID of courses only if there are less than 10 students for the course.
- b) List the Course_ID and the number of modules offered for each course. Display only the course ids which have more than 3 modules offered in it. Sort the result according to the ascending order of the module count.
- c) Display the course id, academic year and the number of modules offered. The number of modules offered should be less than 10.
- d) List the courses that offer more than 2 modules for year 3 students?