

Database Programming

Using Group By and Having Clauses



ACADEMY

Objectives

This lesson covers the following objectives:

- Construct and execute a SQL query using GROUP BY
- Construct and execute a SQL query using GROUP BY ... HAVING
- Construct and execute a GROUP BY on more than one column
- Nest group functions

Purpose

What if you wanted to know the average height of all students? You could write a query that looks like this:

```
SELECT AVG(height) FROM students;
```

But what if you wanted to know the average height of the students based on their year in school? With what you know right now, you would have to write a number of different SQL statements to accomplish this:

```
SELECT AVG(height) FROM students WHERE year_in_school = 10;  
SELECT AVG(height) FROM students WHERE year_in_school = 11;  
SELECT AVG(height) FROM students WHERE year_in_school = 12;
```

Purpose (cont.)

And so on! To simplify problems like this with just one statement, you use the GROUP BY and HAVING clauses.

GROUP BY Use

You use the GROUP BY clause to divide the rows in a table into smaller groups. You can then use the group functions to return summary information for each group.

```
SELECT department_id, AVG(salary)
FROM employees
GROUP BY department_id;
```

DEPARTMENT_ID	AVG (SALARY)
10	4400
20	9500
50	3500
60	6400
...	...

GROUP BY Use (cont.)

In the SELECT statement shown, the rows are being grouped by department_id. The AVG function is then automatically applied to each group.

```
SELECT department_id, AVG(salary)
FROM employees
GROUP BY department_id;
```

DEPARTMENT_ID	AVG (SALARY)
10	4400
20	9500
50	3500
60	6400
...	...

GROUP BY Example

What if we wanted to find the maximum salary of employees in each department? We use a **GROUP BY** clause stating which column to use to group the rows.

```
SELECT MAX(salary)
FROM employees
GROUP BY department_id;
```

DEPT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
...	...

MAX (SALARY)
24000
9000
5800
...

GROUP BY Example (cont.)

But how can we tell which maximum salary belongs to which department?

DEPT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
...	...

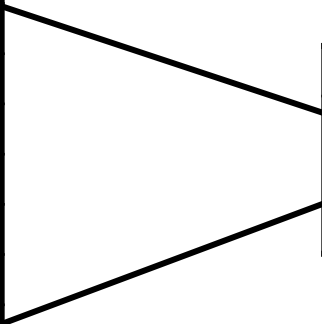
MAX (SALARY)
24000
9000
5800
...

GROUP BY in SELECT

Usually we want to include the GROUP BY column in the SELECT list.

```
SELECT department_id, MAX(salary)
FROM employees
GROUP BY department_id;
```

DEPT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
...	...



DEPT_ID	MAX (SALARY)
90	24000
60	9000
...	...

GROUP BY Clause

Group functions require that any column listed in the SELECT clause that is not part of a group function must be listed in a GROUP BY clause.

What is wrong with this example?

```
SELECT job_id, last_name, AVG(salary)
FROM employees
GROUP BY job_id;
```



ORA-00979: not a GROUP BY expression

COUNT



```
SELECT count(first_name), shirt_color  
FROM students  
GROUP BY shirt_color
```

This example shows how many students wear shirts of each color.

Remember that group functions ignore null values, so if any student does not have a first name, he will not be included in the COUNT. Of course this is unlikely, but when constructing SQL statements, we have to think about all of the possibilities.

COUNT (cont.)



```
SELECT count(first_name), shirt_color  
FROM students  
GROUP BY shirt_color
```

It would be better to start with:

```
SELECT COUNT(*), shirt_color
```

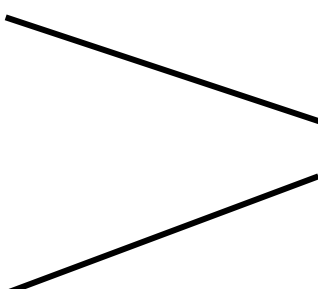
This would count all of the rows in each shirt color group.

WHERE Clause

We can also use a WHERE clause to exclude rows before the remaining rows are formed into groups.

```
SELECT department_id, MAX(salary)
FROM employees
WHERE last_name <> 'King'
GROUP BY department_id;
```

LAST_NAME	DEPT_ID	SALARY
King	90	24000
Kochhar	90	17000
De Haan	90	17000
Hunold	60	9000
Ernst	60	6000
Lorentz	60	4200
...



DEPT_ID	MAX (SALARY)
90	17000
60	9000
...	...

More GROUP BY Examples

Key terms used in this lesson included:

1. Show the average graduation rate of the schools in several cities; include only those students who have graduated in the last few years.

```
SELECT AVG(graduation_rate), city
FROM students
WHERE graduation_date >= '01-JUN-2007'
GROUP BY city;
```

2. Count the number of students in the school, grouped by grade; include all students.

```
SELECT COUNT(first_name), grade
FROM students
GROUP BY grade;
```

GROUP BY Guidelines

Important guidelines to remember when using a GROUP BY clause are:

- If you include a group function (AVG, SUM, COUNT, MAX, MIN, STDDEV, VARIANCE) in a SELECT clause along with any other individual columns, each individual column must also appear in the GROUP BY clause.
- You cannot use a column alias in the GROUP BY clause.
- The WHERE clause excludes rows before they are divided into groups.

Groups Within GROUPS

Sometimes you need to divide groups into smaller groups. For example, you may want to group all employees by department; then, within each department, group them by job. This example shows how many employees are doing each job within each department.

```
SELECT department_id, job_id, count(*)  
FROM employees  
WHERE department_id > 40  
GROUP BY department_id, job_id;
```

DEPT_ID	JOB_ID	COUNT(*)
50	ST_MAN	1
50	ST_CLERK	4
60	IT_PROG	3
80	SA_MAN	1
80	SA_REP	2
...

Nesting Group Functions

Group functions can be nested to a depth of two when GROUP BY is used.

```
SELECT max(avg(salary))  
FROM employees  
GROUP by department_id;
```

How many values will be returned by this query? The answer is one – the query will find the average salary for each department, and then from that list, select the single largest value.

HAVING

Suppose we want to find the maximum salary in each department, but only for those departments which have more than one employee? What is wrong with this example?

```
SELECT department_id, MAX(salary)
FROM employees
WHERE COUNT(*) > 1
GROUP BY department_id;
```



ORA-00934: group function is not allowed here

HAVING (cont.)

In the same way you used the WHERE clause to restrict the rows that you selected, you can use the HAVING clause to restrict groups.

In a query using a GROUP BY and HAVING clause, the rows are first grouped, group functions are applied, and then only those groups matching the HAVING clause are displayed.

HAVING (cont.)

The WHERE clause is used to restrict rows; the HAVING clause is used to restrict groups returned from a GROUP BY clause.

```
SELECT department_id,  
MAX(salary)  
FROM employees  
GROUP BY department_id  
HAVING COUNT(*) > 1;
```



DEPARTMENT_ID	MAX(SALARY)
20	13000
50	5800
60	9000
...	...

HAVING (cont.)

Although the HAVING clause can precede the GROUP BY clause in a SELECT statement, it is recommended that you place each clause in the order shown. The ORDER BY clause (if used) is always last!

```
SELECT column, group_function  
FROM table  
WHERE  
GROUP BY  
HAVING  
ORDER BY
```

Terminology

Key terms used in this lesson included:

- GROUP BY
- HAVING

Summary

In this lesson, you should have learned how to:

- Construct and execute a SQL query using GROUP BY
- Construct and execute a SQL query using GROUP BY ... HAVING
- Construct and execute a GROUP BY on more than one column
- Nest group functions