ORACLE®

ACADEMY

Database Programming with SQL

9-1
Using Group By and Having Clauses

ORACLE ACADEMY

Objectives

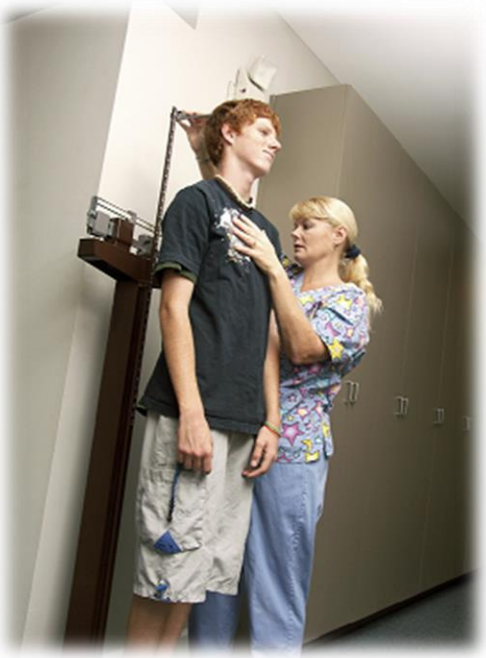
This lesson covers the following objectives:

- 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

- 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;
```



Purpose

- 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;
```

- 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
ORDER BY department_id;
```

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333333333333333
90	19333.3333333333333333
110	10150
-	7000

GROUP BY Use

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

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

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333333333333333
90	19333.3333333333333333
110	10150
-	7000

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)
7000
24000
13000
…

GROUP BY Example

- 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)
7000
24000
13000
…

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)
-	7000
90	24000
20	13000
…	…

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

- This example shows how many countries are in each region.
- Remember that group functions ignore null values, so if any country does not have a country name, it will not be included in the COUNT.

```
SELECT COUNT(country_name), region_id
FROM wf_countries
GROUP BY region_id
ORDER BY region_id;
```

COUNT(COUNTRY_NAME)	REGION_ID
15	5
28	9
21	11
8	13
7	14
8	15
5	17
17	18

COUNT

- Of course this is unlikely, but when constructing SQL statements, we have to think about all of the possibilities.
- It would be better to write the query using COUNT(*):

```
SELECT COUNT(*), region_id
FROM wf_countries
GROUP BY region_id
ORDER BY region_id;
```

- This would count all of the rows in each region group, without the need to check which columns contained NULL values.

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)
-	7000
90	17000
20	13000
…	…

More GROUP BY Examples

- Show the average population of all countries in each region.
- Round the average to a whole number.

```
SELECT region_id, ROUND(AVG(population)) AS population
FROM wf_countries
GROUP BY region_id
ORDER BY region_id;
```

- Count the number of spoken languages for all countries.

```
SELECT country_id, COUNT(language_id) AS "Number of languages"
FROM wf_spoken_languages
GROUP BY country_id;
```

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(*)
110	AC_ACCOUNT	1
50	ST_CLERK	4
80	SA_REP	2
90	AD_VP	2
50	ST_MAN	1
…	…	…

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

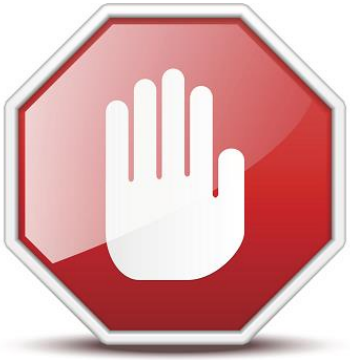
- Suppose we want to find the maximum salary in each department, but only for those departments which have more than one employee?
- What is wrong with this example?

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

ORA-00934: group function is not allowed here

HAVING

- 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

- 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
ORDER BY department_id;
```

DEPARTMENT_ID	MAX(SALARY)
20	13000
50	5800
60	9000
80	11000
90	24000
110	12000

HAVING

- This query finds the average population of the countries in each region.
- It then only returns the region groups with a lowest population greater than three hundred thousand.

```
SELECT region_id,
  ROUND(AVG(population))
FROM wf_countries
GROUP BY region_id
HAVING MIN(population)>300000
ORDER BY region_id;
```

REGION_ID	ROUND(AVG(POPULATION))
14	27037687
17	18729285
30	193332379
34	173268273
143	12023602
145	8522790
151	28343051

HAVING

- 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

ORACLE®

ACADEMY