LAB 5: Writing Group functions

Objectives

- ✓ Identify the available group functions
- ✓ Describe the use of group functions
- ✓ Group data using the GROUP BY clause
- ✓ Include or exclude grouped rows by using the HAVING clause

Lesson Aim

This lesson further addresses functions. It focuses on obtaining summary information, such as averages, for groups of rows. It discusses how to group rows in a table into smaller sets and how to specify search criteria for groups of rows.

Types of Group Functions

• AVG • COUNT • MAX • MIN • STDDEV • SUM • VARIANCE

Guidelines for Using Group Functions

• DISTINCT makes the function consider only nonduplicate values; ALL makes it consider every value including duplicates. The default is ALL and therefore does not need to be specified.
• The data types for the functions with an expr argument may be CHAR, VARCHAR2, NUMBER, or DATE.
• All group functions ignore null values. To substitute a value for null values, use the NVL functions.
• The Oracle Server implicitly sorts the result set in ascending order when using a GROUP BY clause. To override this default ordering, DESC can be used in an ORDER BY clause.

Group Functions

You can use AVG, SUM, MIN, and MAX functions against columns that can store numeric data. The example in the slide displays the average, highest, lowest, and sum of monthly salaries for all sales representatives.

```
SELECT AVG(salary), MAX(salary),
MIN(salary), SUM(salary)
FROM employees
WHERE job_id LIKE '%REP%';
```

Note: AVG, SUM, VARIANCE, and STDDEV functions can be used only with numeric data types.

Min and Max Function

You can use the MIN and MAX functions for any data type. The following example displays the employee last name that is first and the employee last name that is the last in an alphabetized list of all employees.

```
SELECT MIN(last_name), MAX(last_name)
FROM employees;
```

The slide example displays the most junior and most senior employee.

```
SELECT MIN(hire_date), MAX(hire_date)
FROM employees;
```

The COUNT Function

The COUNT function has three formats:

• COUNT(*) • COUNT(*expr*) • COUNT(DISTINCT *expr*)

COUNT(*) returns the number of rows in a table that satisfy the criteria of the SELECT statement, including duplicate rows and rows containing null values in any of the columns. If a WHERE clause is

included in the SELECT statement, COUNT(*) returns the number of rows that satisfies the condition in the WHERE clause. In contrast, COUNT(*expr*) returns the number of nonnull values in the column identified by *expr*. COUNT(DISTINCT *expr*) returns the number of unique, non-null values in the column identified by *expr*.

```
SELECT COUNT(*)
FROM employees
WHERE department_id = 50;

SELECT COUNT(commission_pct)
FROM employees
WHERE department_id = 80;
```

Using the NVL Function with Group Functions

The NVL function forces group functions to include null values. In the example in the slide, the average is calculated based on *all* rows in the table, regardless of whether null values are stored in the COMMISSION_PCT column. The average is calculated as the total commission paid to all employees divided by the total number of employees in the company.

```
SELECT AVG(NVL(commission_pct, 0))
FROM employees;
```

The GROUP BY Clause

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

In the syntax:

group_by_expression specifies columns whose values determine the basis for grouping rows

Guidelines

• If you include a group function in a SELECT clause, you cannot select individual results as well, *unless* the individual column appears in the GROUP BY clause. You receive an error message if you fail to include the column list in the GROUP BY clause.
• Using a WHERE clause, you can exclude rows before dividing them into groups.
• You must include the columns in the GROUP BY clause.
• You cannot use a column alias in the GROUP BY clause.
• By default, rows are sorted by ascending order of the columns included in the GROUP BY list. You can override this by using the ORDER BY clause.

Using the GROUP BY Clause

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

The GROUP BY column does not have to be in the SELECT list.

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

You can use the group function in the ORDER BY clause.

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

Using the GROUP BY Clause on Multiple Columns

```
SELECT department_id dept_id, job_id, SUM(salary)
FROM employees
GROUP BY department_id, job_id;
```

You can return summary results for groups and subgroups by listing more than one GROUP BY column. You can determine the default sort order of the results by the order of the columns in the GROUP BY clause. Here is how the SELECT statement on the slide, containing a GROUP BY clause, is evaluated:

• The SELECT clause specifies the column to be retrieved:
– Department number in the EMPLOYEES table
– Job ID in the EMPLOYEES table
– The sum of all the salaries in the group that you specified in the GROUP BY clause
• The FROM clause specifies the tables that the database must access: the EMPLOYEES table
• The GROUP BY clause specifies how you must group the rows:
– First, the rows are grouped by department number
– Second, within the department number groups, the rows are grouped by job ID

So the SUM function is being applied to the salary column for all job IDs within each department number group.

Illegal Queries Using Group Functions

Any column or expression in the SELECT list that is not an aggregate function must be in the GROUP BY clause**.**

```
SELECT department_id, COUNT(last_name)
FROM employees;
```

Illegal Queries Using Group Functions

- ✓ You cannot use the WHERE clause to restrict groups.
- ✓ You use the HAVING clause to restrict groups.
- ✓ You cannot use group functions in the WHERE clause.

```
SELECT department_id, AVG(salary)
FROM employees
WHERE AVG(salary) > 8000
GROUP BY department_id;
```

Illegal Queries Using Group Functions (continued)

The WHERE clause cannot be used to restrict groups. The SELECT statement in the slide results in an error because it uses the WHERE clause to restrict the display of average salaries of those departments that have an average salary greater than $8,000.

You can correct the slide error by using the HAVING clause to restrict groups.

```
SELECT department_id, AVG(salary)
FROM employees
HAVING AVG(salary) > 8000
GROUP BY department_id;
```

Using the HAVING Clause (continued)

```
SELECT job_id, SUM(salary) PAYROLL
FROM employees
WHERE job_id NOT LIKE '%REP%'
GROUP BY job_id
HAVING SUM(salary) > 13000
ORDER BY SUM(salary);
```

The example in the slide displays the job ID and total monthly salary for each job with a total payroll exceeding $13,000. The example excludes sales representatives and sorts the list by the total monthly salary.

Nesting Group Functions

Display the maximum average salary.

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

Group functions can be nested to a depth of two. The example in the slide displays the maximum average salary.

Practice

1. Group functions work across many rows to produce one result per group. True/False
2. Group functions include nulls in calculations. True/False
3. The WHERE clause restricts rows prior to inclusion in a group calculation. True/False
4. Display the highest, lowest, sum, and average salary of all employees.
5. Modify the query 4 to display the minimum, maximum, sum, and average salary for each job type
6. Display the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number. Place your SQL statement in a text file named lab5_6.sql.

Maximum	Minimum	Sum	Average
24000	2500	175500	8775

7. Modify the query in lab5_4.sql to display the minimum, maximum, sum, and average salary for each job type. Resave lab5_4.sql to lab5_5.sql. Run the statement in lab5_5.sql.

JOB_ID	Maximum	Minimum	Sum	Average
AC_ACCOUNT	8300	8300	8300	8300
AC_MGR	12000	12000	12000	12000
AD_ASST	4400	4400	4400	4400
AD_PRES	24000	24000	24000	24000
AD_VP	17000	17000	34000	17000
IT_PROG	9000	4200	19200	6400
MK_MAN	13000	13000	13000	13000
MK_REP	6000	6000	6000	6000
SA_MAN	10500	10500	10500	10500
SA_REP	11000	7000	26600	8867
ST_CLERK	3500	2500	11700	2925
ST_MAN	5800	5800	5800	5800

12 rows selected.

8. Modify the query 4 to display the minimum, maximum, sum, and average salary for each job type
9. Write a query to display the number of people with the same job.

JOB_ID	COUNT(*)
AC_ACCOUNT	1
AC_MGR	1
AD_ASST	1
AD_PRES	1
AD_VP	2
IT_PROG	3
MK_MAN	1
MK_REP	1
SA_MAN	1
SA_REP	3
ST_CLERK	4
ST_MAN	1

12 rows selected.

10. Write a query to display each department's name, location, number of employees, and the average salary for all employees in that department. Label the columns `Name`, `Location`, `Number of People`, and `Salary`, respectively. Round the average salary to two decimal places.

Name	Location	Number of People	Salary
Accounting	1700	2	10150
Administration	1700	1	4400
Executive	1700	3	19333.33
IT	1400	3	6400
Marketing	1800	2	9500
Sales	2500	3	10033.33
Shipping	1500	5	3500

7 rows selected.

11. Create a query that will display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998. Create appropriate column headings.

TOTAL	1995	1996	1997	1998
20	1	2	2	3