5-Aggregating Data Using Group Functions

Objectives

After completing this lesson, you should be able to do the following:
•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

What Are Group Functions?

Group functions operate on sets of rows to give one result per group.
EMPLOYEES
The maximum
salary in the
EMPLOYEES table.

Types of Group Functions

• AVG
•COUNT
•MAX
•MIN
•SUM

Group Functions Syntax

```
SELECT [ column ,] group_function(column), ...
FROM table
[WHERE condition ]
[GROUP BY column ]
[ORDER BY column ];
```

Using the AVG and SUM Functions

You can use AVG and SUM for numeric data.

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

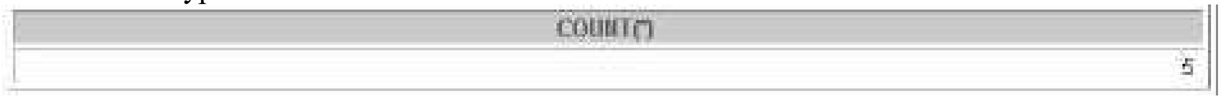
Using the MIN and MAX Functions

You can use MIN and MAX for any data type.

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

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

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



COUNT(*)
5

Using the COUNT Function COUNT(*)

returns the number of rows in a table.

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

Using the COUNT Function

- **COUNT(expr) returns the number of rows with non-null values for the expr.**
- **Display the number of department values in the EMPLOYEES table, excluding the null values.**

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

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

Using the DISTINCT Keyword

- COUNT(DISTINCT expr) returns the number of distinct nonnull values of the expr .
- Display the number of distinct department values in the EMPLOYEES table.

```
SELECT COUNT(DISTINCT department_id)
FROM   employees;
```

Group Functions and Null Values

Group functions ignore null values in the column.

```
SELECT AVG(commission_pct)
FROM   employees;
```

Using the NVL Function with Group Functions

The NVL function forces group functions to include null values.

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

Creating Groups of Data

EMPLOYEES
4400
9500
The average salary 3500 in EMPLOYEES
Table 6400 for each department.

Creating Groups of Data:

GROUP BY Clause Syntax

```
SELECT column , group_function(column)
FROM table
[WHERE condition ]
[GROUP BY group_by_expression ]
[ORDER BY column ];
```

Divide rows in a table into smaller groups

Using the GROUP BY Clause

All columns in the SELECT list that are not in group functions must be in the GROUP BY clause.

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

Using the GROUP BY Clause

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

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

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

Grouping by More Than One Column

EMPLOYEES
Add up the salaries in the
EMPLOYEES table
for each job,
grouped by
department.

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;
```

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;
SELECT department_id, COUNT(last_name)
*
ERROR at line 1:
ORA-00937: not a single-group group function
```

```
SELECT department_id, count(last_name)
FROM   employees
GROUP BY department_id;
```

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

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;
WHERE  AVG(salary) > 8000
*
ERROR at line 3:
ORA-00934: group function is not allowed here
```

Illegal Queries Using Group Functions (continued)

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

Excluding Group Results

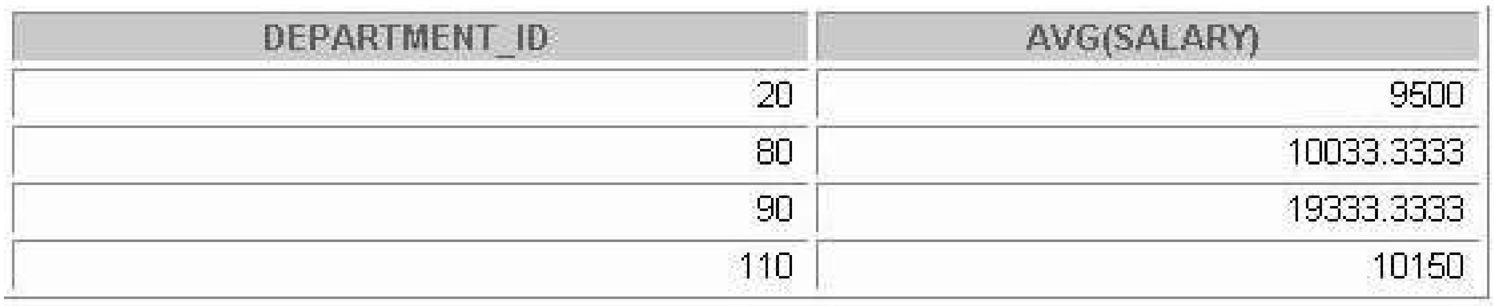
EMPLOYEES

The maximum
salary
per department
when it is
greater than
$10,000.

Excluding Group Results: The HAVING Clause

Use the HAVING clause to restrict groups:
1. Rows are grouped.
2. The group function is applied.
3. Groups matching the
HAVING clause are displayed.

```
SELECT column , group_function
FROM table
[WHERE condition]
[GROUP BY group_by_expression ]
[HAVING group_condition]
[ORDER BY column ];
```



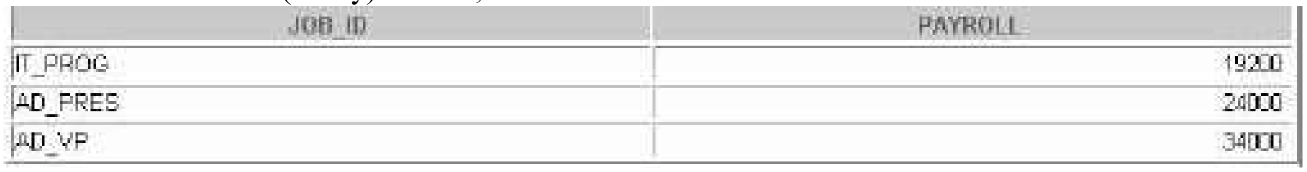
DEPARTMENT_ID	AVG(SALARY)
20	9500
80	10033.3333
90	19333.3333
110	10150

Using the HAVING Clause

```
SELECT   department_id, MAX(salary)
FROM     employees
GROUP BY department_id
HAVING   MAX(salary)>10000;
```

Using the HAVING Clause

```
SELECT   department_id, AVG(salary)
FROM     employees
GROUP BY department_id
HAVING   max(salary)>10000;
```

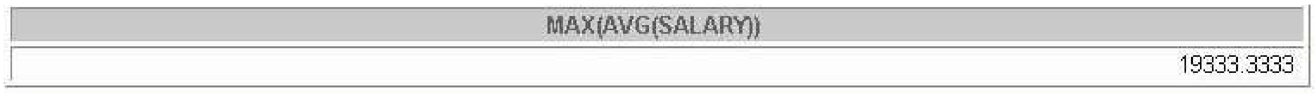


JOB_ID	PAYROLL
IT_PROG	19200
AD_PRES	24000
AD_VP	34000

Using the HAVING Clause

```
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);
```

.



```
MAX(AVG(SALARY))
      19333.3333
```

Nesting Group Functions

Display the maximum average salary.

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

Paper-Based Questions

For questions 1 through 3, circle either True or False.

Note:

Column aliases are used for the queries.

Practice 4

Determine the validity of the following three statements. Circle either True or False.

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. 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.
5. 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.

6. Write a query to display the number of people with the same job.
7. Determine the number of managers without listing them. Label the column
Number of Managers .
Hint: Use the MANAGER_ID column to determine the number of managers.
8. Write a query that displays the difference between the highest a nd lowest salaries. Label the column DIFFERENCE.

If you have time, complete the following exercises:

9. Display the manager number and the salary of the lowest paid employee for that manager.
Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is less than $6,000. Sort the output in descending order of salary.

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.

If you want an extra challenge, complete the following exercises:

11. Create a query that will display the total number of employees a nd, of that total, the number of employees hired in 1995, 1996, 1997, and 1998. Create appropriate column headings.
12. Create a matrix query to display the job, the salary for that job based on department number, and the total salary for that job, for departments 20, 50, 80, and 90, giving each column an appropriate heading.