4

Now 5 in 11g

Reporting Aggregate Data Using the Group Functions

Objectives

After completing this lesson AND
practicing, you will begin to understand the following:

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

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.

Lesson Agenda

Group Functions
Grouping Rows
Nesting Group Functions

What Are Group Functions

Group Functions ➔ operate on sets of rows
➔ to give one result per group of rows

EXAMPLE:

```
SELECT  AVG (salary)
FROM    employees;

AVG(SALARY)
     8775
```

One result from a set of rows

=========================

Single Row functions worked on single rows and returned 1 result per row

```
     SELECT  UPPER(last_name) …
```

Each row selected changed the format of whatever last_name was stored as to display in UPPER case

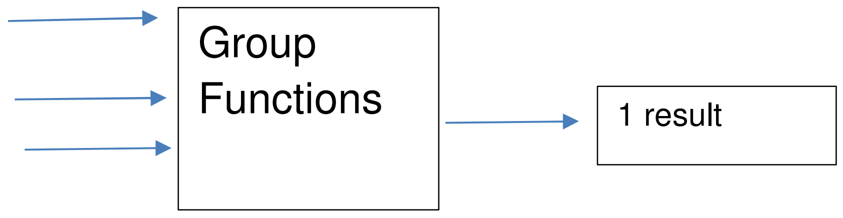
Group functions (multi-row functions)

→ Operate on sets of rows to give one result per group.

- These sets may comprise the entire table or the table split into groups

Types of Group Functions

AVG
COUNT
MAX
MIN
STDDEV
SUM
VARIANCE



OPTIONS you can use with the functions

(go to examples)

AVG ([distinct | ALL] EXPRE}) n)

- Average value of n, ignoring null values

COUNT ({ * [distinct | ALL] })

- Number of rows where expr evaluates to something other than null
- count all selected rows using * including duplicates and nulls unless use distinct

MAX([DISTINCT|ALL]expr)

- Maximum value of expr, ignoring null

MIN([DISTINCT|ALL]expr)

-Minimum value of expr, ignoring null values

SUM([DISTINCT|ALL]n)

- Sum values of n, ignoring null values

STDDEV([DISTINCT|ALL]x)

- Standard deviation of n, ignoring null

VARIANCE ([DISTINCT|ALL]x)

- Variance of n, ignoring null values

GROUP FUNCTION SYNTAX

PROBLEM:

President wants to know data about salaries, such as AVERAGE, what the highest and lowest paid person's salary is and the company's total salary payout.

➔ ➔ GO BACK to previous page and find the appropriate function

```
SELECT     AVG (SALARY),
           MAX (SALARY), -- highest paid
           MIN (SALARY), -- lowest opaid
           SUM (SALARY)  -- total of all salaries
FROM    EMPLOYEES;

AVG(SALARY) MAX(SALARY) MIN(SALARY) SUM(SALARY)
----------- ----------- ----------- -----------
       8775       24000        2500      175500
```

How would we change the SQL for the President to show the same results but for Sales Reps only

Show the same for all Sales Reps

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

```
AVG(SALARY) MAX(SALARY) MIN(SALARY) SUM(SALARY)
----------- ----------- ----------- -----------
       8150       11000        6000       32600
```

GUIDELINES

1
DISTINCT
- Makes the function consider only non-duplicate values

ALL
- Makes function consider every value

    DEFAULT value is ALL and does not need to be specified

2
The DATA TYPES with the syntax ***expr*** argument may be
    CHAR, VARCHAR2
    NUMBER, DATE

3
All group functions ignore null values. To substitute for null use NVL etc…

Using MIN, MAX function examples

You can use MIN and MAX for the following
- Numeric
- Character
- Date

Find the newest and oldest employee by hire_date

```
SELECT     MIN(HIRE_DATE),
           MAX (HIRE_DATE)
 FROM      EMPLOYEES;
```

This shows the most senior employees, the one working the longest and the most junior employee

```
MIN(HIRE_ MAX(HIRE_
--------- ---------
17-JUN-87 29-JAN-00
```

Poor titles due partly to column width too small to show title

Find the first person alphabetically by last name
Find the last employee by last name

➔ Applied to character columns

```
SELECT  min (last_name) as "First in line",
        max (last_name) as "Always last to be called"
FROM    employees;
```

```
First in line              Always last to be called
-------------------------- --------------------------
Abel                       Zlotkey
```

Using the Count Function

COUNT (*) – returns the number of rows in a table

```
SELECT  COUNT (*)
FROM    EMPLOYEES;
```

Using COUNT with an expression

```
SELECT  COUNT (commission_pct)
FROM    EMPLOYEES
WHERE DEPARTMENT_ID = 80;
```

NOTE:
Count supplies the number of row that satisfies the SELECT statement in a table

Count with an expression returns the number of rows that meet the condition

```
    select count(last_name)
    from employees
    where last_name between 'A' and 'G'    ➔ returns 5
```

Adding an expression returns non-null values

Adding DISTINCT returns the number of rows that are distinct from all the rows that are not null.

Find the percentage of employees that receive a commission

```
SELECT  COUNT (*),
        COUNT(COMMISSION_PCT),
        COUNT(COMMISSION_PCT)/COUNT(*)
FROM    EMPLOYEES
```

How many departments are there in the employees table?

DISTINCT Examples

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


COUNT (DISTINCTDEPARTMENT_ID)
----------------------------
                       7
```

What is the average commission percent paid?

GROUP FUNCTIONS and NULL

PROBLEM – average of just those who receive commission or of all employees

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

```
AVG (COMMISSION_PCT)
-------------------
              .2125
```

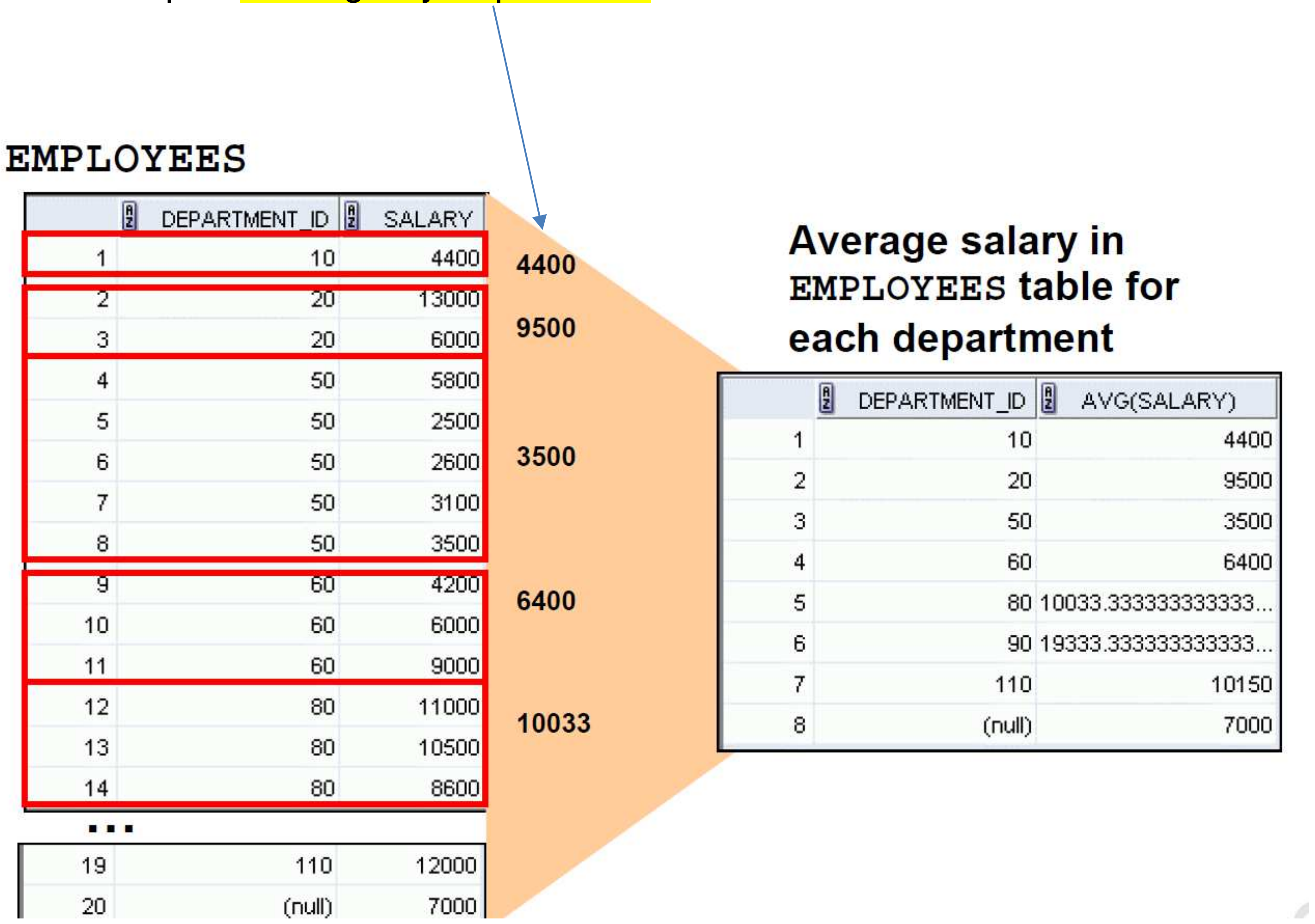
```
SELECT  AVG (NVL (commission_pct, 0))
FROM    employees;
```

```
AVG (NVL (COMMISSION_PCT,0))
--------------------------
                     .0425
```

Groups of Data

All group functions have treated the table so far as one large group

Sometimes the information needs to be divide into smaller groups
Example: Average by department



EMPLOYEES
4400
9500
3500
6400
10033
Average salary in EMPLOYEES table for each department

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	5800
5	50	2500
6	50	2600
7	50	3100
8	50	3500
9	60	4200
10	60	6000
11	60	9000
12	80	11000
13	80	10500
14	80	8600
...		
19	110	12000
20	(null)	7000

	DEPARTMENT_ID	AVG(SALARY)
1	10	4400
2	20	9500
3	50	3500
4	60	6400
5	80	10033.333333333333...
6	90	19333.333333333333...
7	110	10150
8	(null)	7000

GROUP BY

```
SELECT  DEPARTMENT_ID, AVG(SALARY)
FROM    EMPLOYEES;
```

```
ERROR at line 1:
ORA-00937: not a single-group group function
```

Why an error?

The use of department_id results in a row of output for each row in the employee table

The AVG wants a single result for the entire table.

➔ ➔ There is no sensible way to display that.

Introduces the GROUP BY to apply the group function by department_id

```
SELECT  DEPARTMENT_ID, AVG(SALARY)
FROM    EMPLOYEES
GROUP BY DEPARTMENT_ID;
```

```
DEPARTMENT_ID AVG(SALARY)
------------- -----------
                     7000
           90  19333.3333
           20        9500
          110       10150
           50        3500
           80  10033.3333
           60        6400
           10        4400

8 rows selected
```

Rewrite the code to clean up the output

To clean up output

```
SELECT  DEPARTMENT_ID, round(AVG(SALARY),0)
FROM    EMPLOYEES
GROUP BY DEPARTMENT_ID;
```

GROUP BY

The GROUP BY column does not need to be in the select

```
SELECT   AVG(SALARY)
FROM     EMPLOYEES
GROUP BY DEPARTMENT_ID;
```

```
AVG(SALARY)
-----------
       7000
 19333.3333
       9500
      10150
       3500
 10033.3333
       6400
       4400
```

Notice that the output is correct but is not very meaningful to the user

GROUP BY often needs an ORDER BY

```
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  10033.3333
           90  19333.3333
          110       10150
                     7000
```

Grouping by more than 1 column

Groups within groups

PROBLEM:

Display the total salary paid to each job title within each department

LOGIC
Group employee by department
Within department group job titles
Sum up that lower grouping

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

```
DEPARTMENT_ID JOB_ID     SUM(SALARY)
------------- ---------- -----------
          110 AC_ACCOUNT        8300
           90 AD_VP            34000
           50 ST_CLERK         11700
           80 SA_REP           19600
           50 ST_MAN            5800
           80 SA_MAN           10500
          110 AC_MGR           12000
           90 AD_PRES          24000
           60 IT_PROG          19200
           20 MK_MAN           13000
              SA_REP            7000
           10 AD_ASST           4400
           20 MK_REP            6000
```

Again, hard to see if it truly worked.

What would improve it?

IMPROVED

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

```
DEPARTMENT_ID JOB_ID     SUM(SALARY)
------------- ---------- -----------
           10 AD_ASST           4400
           20 MK_MAN           13000
           20 MK_REP            6000
           50 ST_CLERK         11700
           50 ST_MAN            5800
           60 IT_PROG          19200
           80 SA_MAN           10500
           80 SA_REP           19600
           90 AD_PRES          24000
           90 AD_VP            34000
          110 AC_ACCOUNT        8300
          110 AC_MGR           12000
              SA_REP            7000

 13 rows selected
```

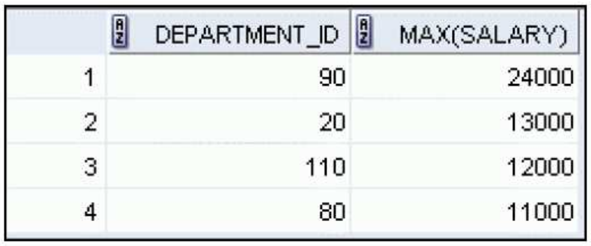
Restricting Which Groups to Show

→NOT by using the WHERE clause

→ using the HAVING clause

Find the maximum salary by department if maximum salary greater than 10,000

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



	DEPARTMENT_ID	MAX(SALARY)
1	90	24000
2	20	13000
3	110	12000
4	80	11000

Again nicer if put department in order
Add the ORDER BY clause

Nesting Group Functions

PROBLEM:
Display the department with the highest average salary

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

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

PRACTICE:

1 Write a query to determine how many job_ids there are.

2 Write a query to find out how many people have the same job

3 Determine the number of managers (without listing them)
HINT: use the manager_id

4 HR department want top know the range of salaries and what the difference is

1 Write a query to determine how many job_ids there are.

```
SELECT      count(distinct job_id)
FROM        employees
```

2 Write a query to find out how many people have the same job

```
SELECT      job_id, count(*)
FROM        employees
group by job_id
```

3 Determine the number of managers (without listing them)

HINT: use the manager_id

```
SELECT      count(distinct manager_id)
FROM        employees
```

4 HR department want top know the range of salaries and what the difference is

```
SELECT      max(salary),
       min(salary),
       max(salary)-min(salary) as "Difference"
FROM        employees
```