The Group Functions

- Jayendra Khatod

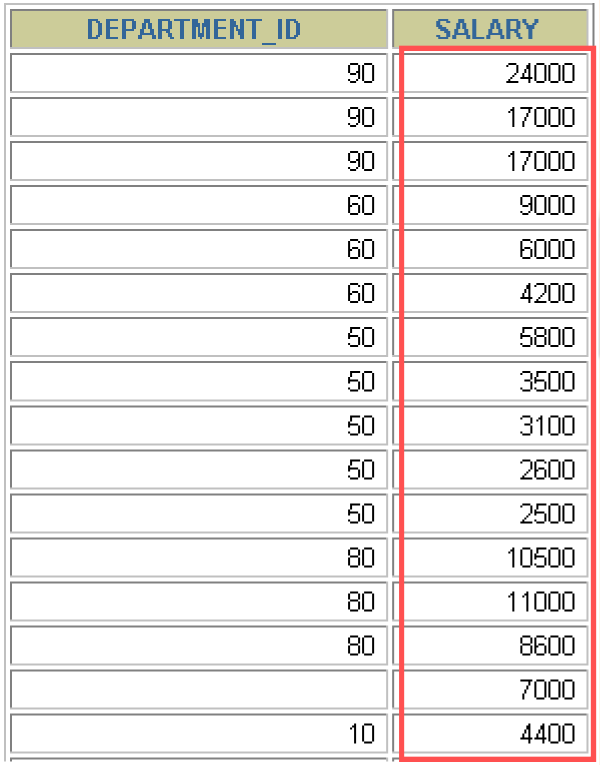
Objectives

- **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**

What Are Group Functions?

Group functions operate on sets of rows to give one result per group.

EMPLOYEES

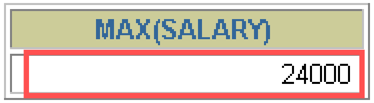


DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
	7000
10	4400

…

20 rows selected.

The maximum salary in the `EMPLOYEES` table.



MAX(SALARY)
24000

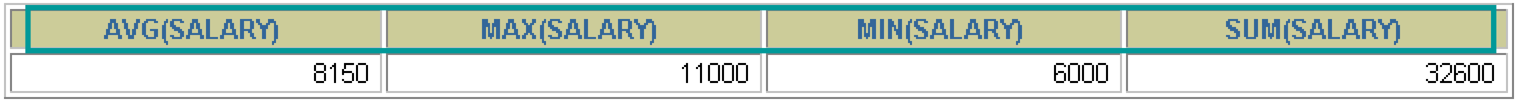
Examples of Group Functions

- AVG
- COUNT
- MAX
- MIN
- SUM

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%';
```

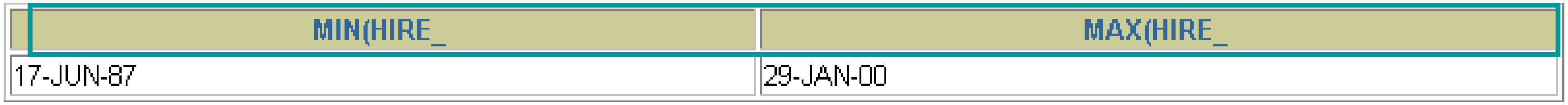


AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
8150	11000	6000	32600

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;
```

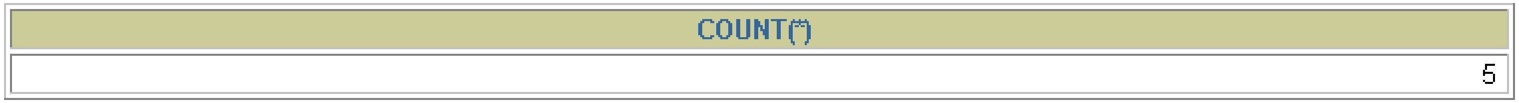


MIN(HIRE_	MAX(HIRE_
17-JUN-87	29-JAN-00

Using the COUNT Function

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

```
SELECT COUNT(*)
FROM   employees
```

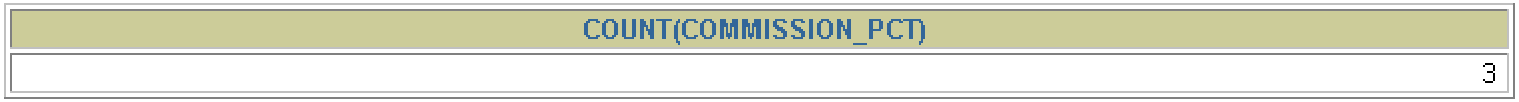


COUNT(*)
5

Using the COUNT Function

- **COUNT(*expr*) returns the number of rows with non-null values for the *expr*.**

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

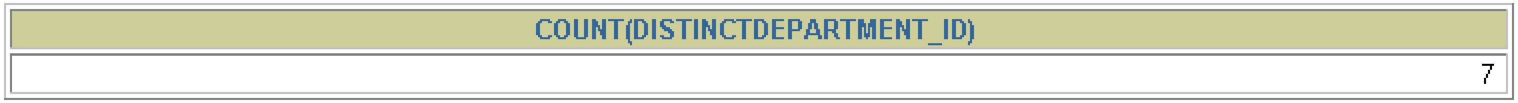


```
COUNT(COMMISSION_PCT)
                    3
```

Using the DISTINCT Keyword

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

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

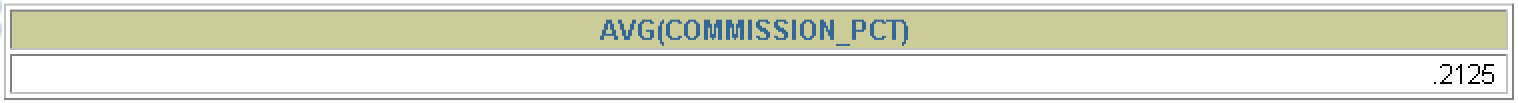


```
COUNT(DISTINCTDEPARTMENT_ID)
                           7
```

Group Functions and Null Values

Group functions ignore null values in the column.

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

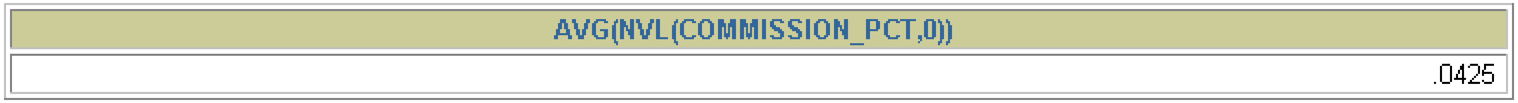


```
AVG(COMMISSION_PCT)
              .2125
```

Using the NVL Function with Group Functions

The NVL function forces group functions to include null values.

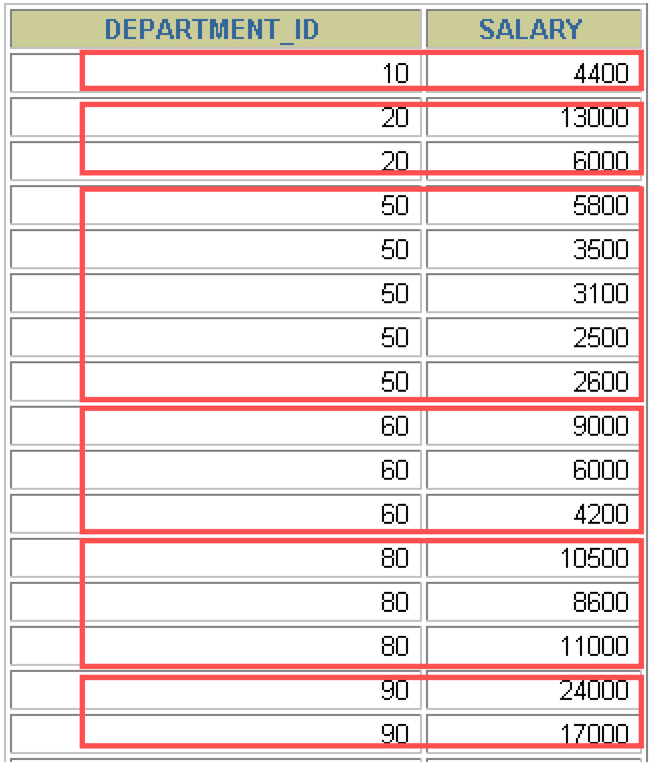
```
SELECT AVG(NVL(commission_pct, 0))
FROM   employees;
```



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

Creating Groups of Data

EMPLOYEES



DEPARTMENT_ID	SALARY
10	4400
20	13000
20	6000
50	5800
50	3500
50	3100
50	2500
50	2600
60	9000
60	6000
60	4200
80	10500
80	8600
80	11000
90	24000
90	17000

…

4400

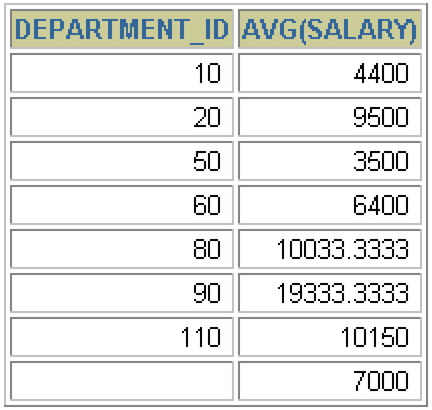
9500

3500

6400

10033

The average salary in `EMPLOYEES` table for each department.



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

Creating Groups of Data: The GROUP BY Clause

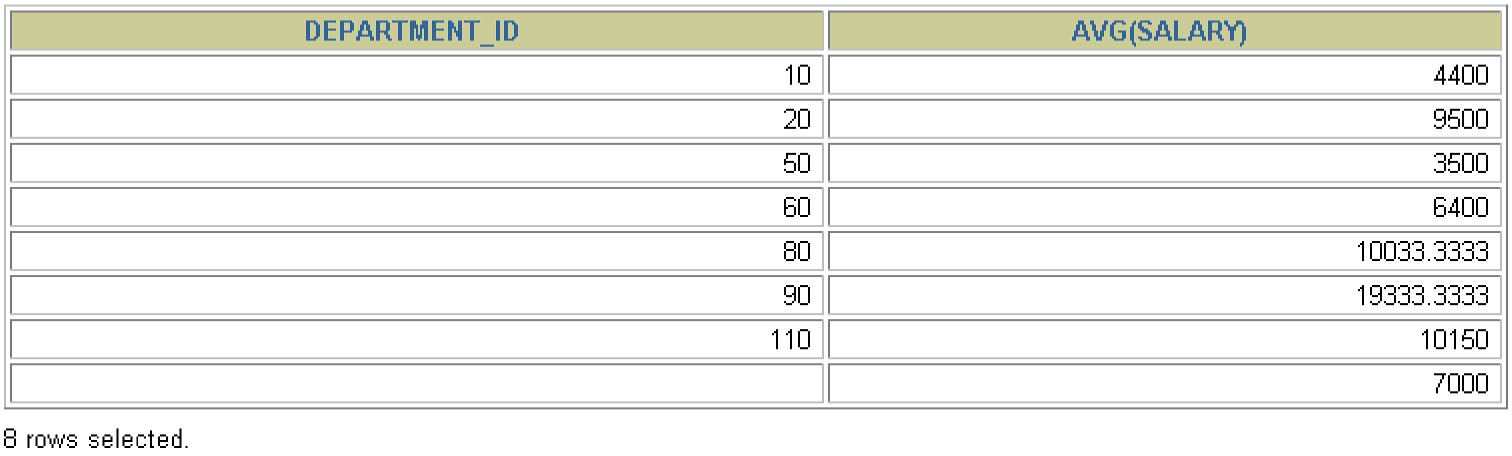
Divide rows in a table into smaller groups by using the GROUP BY clause.

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

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 ;
```



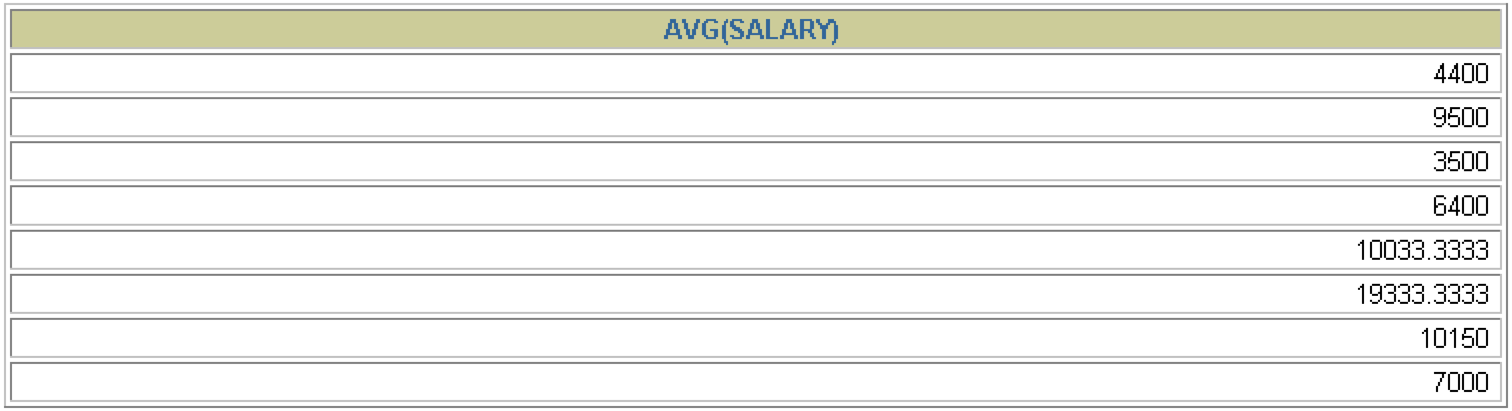
DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

8 rows selected.

Using the GROUP BY Clause

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

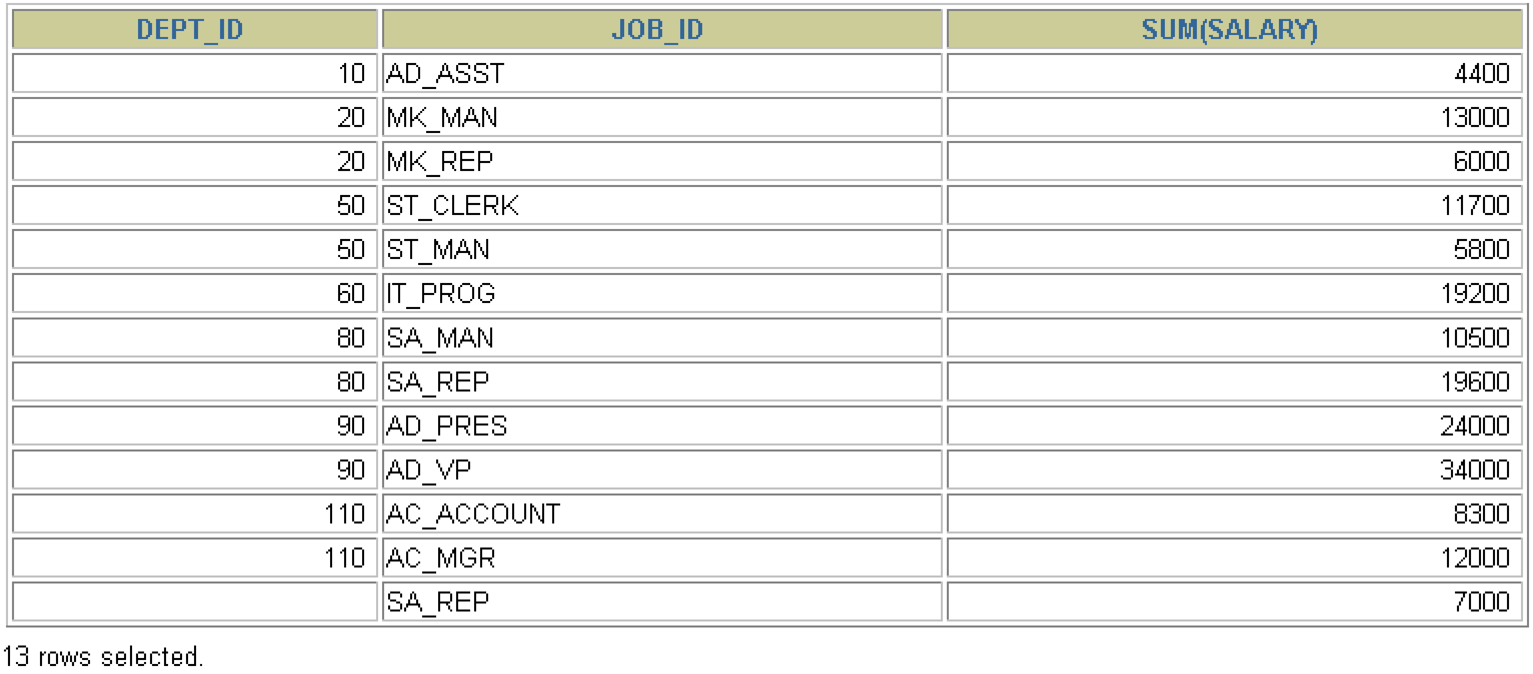
```
SELECT   AVG(salary)
FROM     employees
GROUP BY department_id ;
```



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

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 ;
```



DEPT_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.

Excluding Group Results: The HAVING Clause

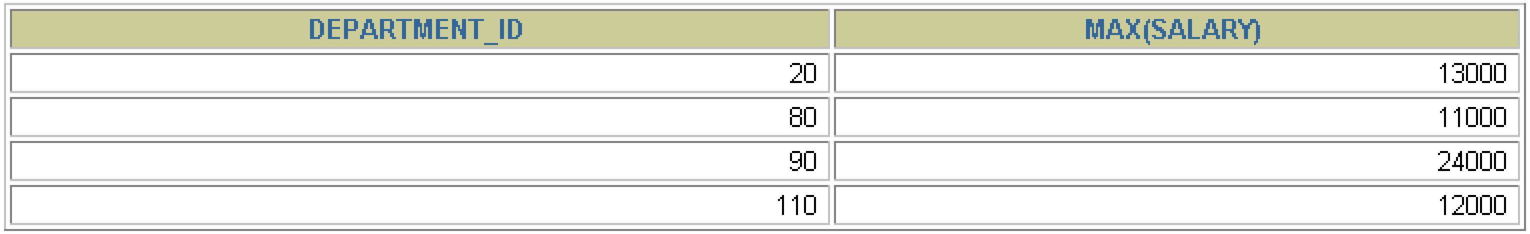
Use the HAVING clause to restrict groups:

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];
```

Using the HAVING Clause

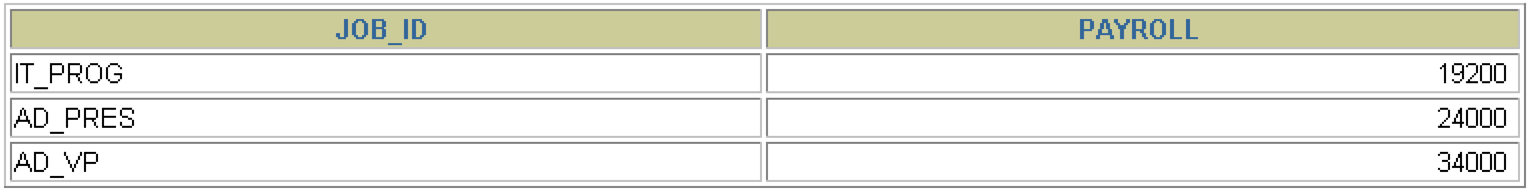
```
SELECT   department_id, MAX(salary)
FROM     employees
GROUP BY department_id
HAVING   MAX(salary)>10000 ;
```



DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

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);
```

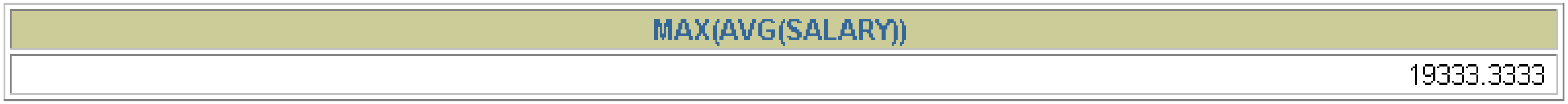


JOB_ID	PAYROLL
IT_PROG	19200
AD_PRES	24000
AD_VP	34000

Nesting Group Functions

Display the maximum average salary.

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



MAX(AVG(SALARY))
19333.3333

Summary

In this lesson, you should have learned how to:

- **Use the group functions COUNT, MAX, MIN, AVG**
- **Write queries that use the GROUP BY clause**
- **Write queries that use the HAVING clause**

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

Thank You !