



Reporting Aggregated Data Using the 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 by using the `GROUP BY` clause
- Include or exclude grouped rows by using the `HAVING` clause



Course RoadMap

Lesson 1: Introduction

Unit 1: Retrieving, Restricting,
and Sorting Data

**Unit 2: Joins, Subqueries, and
Set operators**

Unit 3: DML and DDL



**Lesson 6: Reporting Aggregated Data Using
Group Functions**



Lesson 7: Displaying Data from Multiple
Tables Using Joins



Lesson 8: Using Subqueries to Solve
Queries



Lesson 9: Using Set Operators



Lesson Agenda

- Group functions:
 - Types and syntax
 - Use AVG, SUM, MIN, MAX, COUNT
 - Use the DISTINCT keyword within group functions
 - NULL values in a group function
- Grouping rows:
 - GROUP BY clause
 - HAVING clause
- Nesting group functions



Group Functions

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

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	110	12000
5	110	8300
6	90	24000
7	90	17000
8	90	17000
9	60	9000
10	60	6000
...		
18	80	11000
19	80	8600
20	(null)	7000

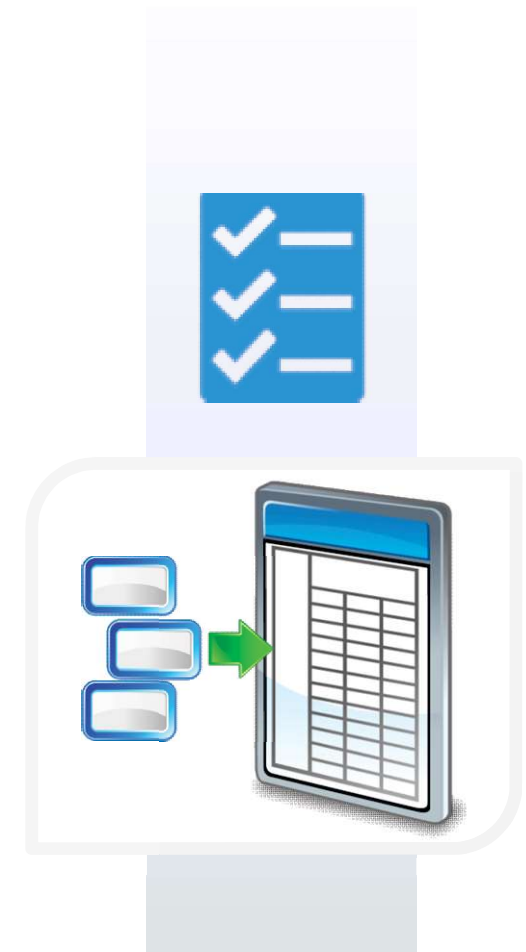
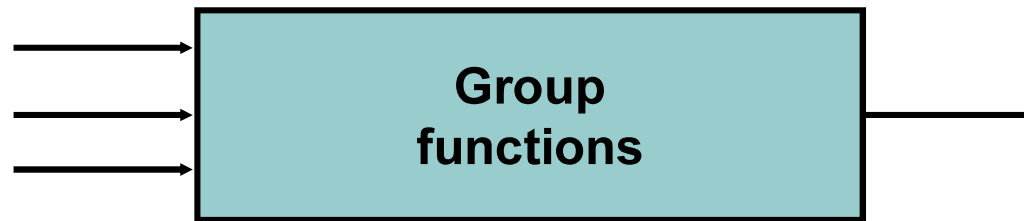
**Maximum salary in
EMPLOYEES table**

MAX(SALARY)
24000



Types of Group Functions

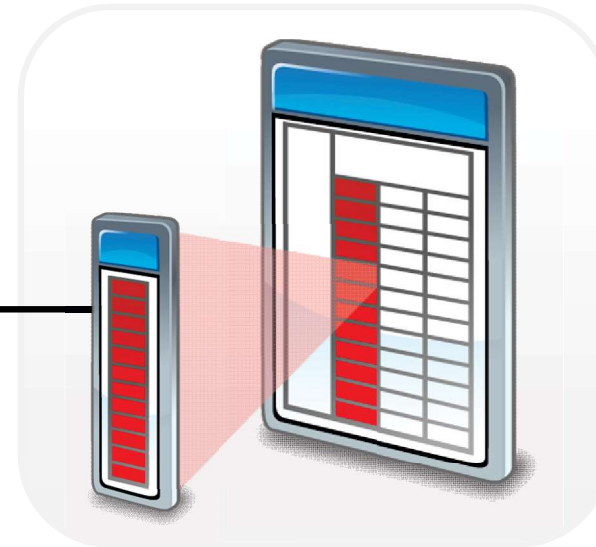
- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE



Group Functions: Syntax

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





Group all rows in a 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%';
```

	 AVG(SALARY)	 MAX(SALARY)	 MIN(SALARY)	 SUM(SALARY)
1	8150	11000	6000	32600

Using the MIN and MAX Functions

- You can use MIN and MAX for numeric, character, and date data types.

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

MIN(HIRE_DATE)	MAX(HIRE_DATE)
17-JUN-87	29-JAN-00

Using the COUNT Function

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

1

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

	COUNT(*)
1	5

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

2

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

	COUNT(COMMISSION_PCT)
1	3

Using the DISTINCT Keyword

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

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

COUNT(DISTINCTDEPARTMENT_ID)	
1	7

Group Functions and Null Values

Group functions ignore null values in the column:

1

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

	Avg	AVG(COMMISSION_PCT)
1		0.2125

The NVL function forces group functions to include null values:

2

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

	Avg	AVG(NVL(COMMISSION_PCT,0))
1		0.0425

Lesson Agenda

- Group functions:
 - Types and syntax
 - Use AVG, SUM, MIN, MAX, COUNT
 - Use the DISTINCT keyword within group functions
 - NULL values in a group function
- Grouping rows:
 - GROUP BY clause
 - HAVING clause
- Nesting group functions



Creating Groups of Data

EMPLOYEES

	DEPARTMENT_ID	SALARY	
1	10	4400	4400
2	20	13000	9500
3	20	6000	
4	50	2500	
5	50	2600	
6	50	3100	3500
7	50	3500	
8	50	5800	
9	60	9000	
10	60	6000	6400
11	60	4200	
12	80	11000	
13	80	8600	10033
14	80	10500	
15	90	17000	
16	90	24000	19333
17	90	17000	
18	110	8300	10150
19	110	12000	
20	(null)	7000	7000

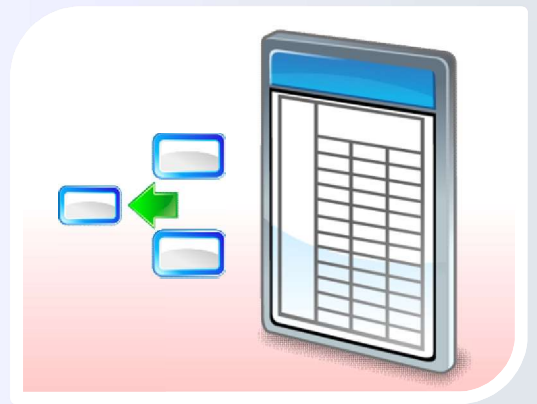
**Average
salary in the
EMPLOYEES
table for each
department**

	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

Creating Groups of Data:

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

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 ;
```

[illegible]

Grouping by More Than One Column

EMPLOYEES

	DEPARTMENT_ID	JOB_ID	SALARY
1	10	AD_ASST	4400
2	20	MK_MAN	13000
3	20	MK_REP	6000
4	50	ST_CLERK	2500
5	50	ST_CLERK	2600
6	50	ST_CLERK	3100
7	50	ST_CLERK	3500
8	50	ST_MAN	5800
9	60	IT_PROG	9000
10	60	IT_PROG	6000
11	60	IT_PROG	4200
12	80	SA_REP	11000
13	80	SA_REP	8600
14	80	SA_MAN	10500
15	90	AD_VP	17000
16	90	AD_PRES	24000
17	90	AD_VP	17000
18	110	AC_ACCOUNT	8300
19	110	AC_MGR	12000
20	(null)	SA_REP	7000

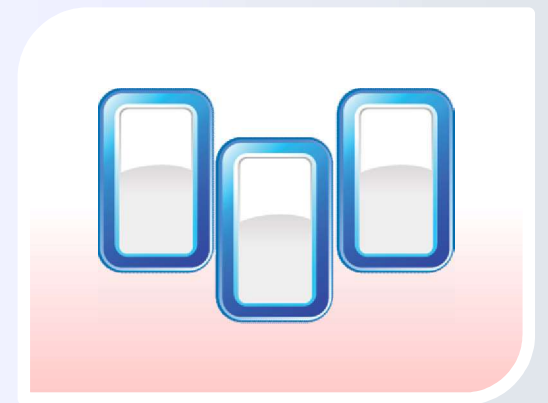
Add the salaries in the **EMPLOYEES** table for each job, grouped by department

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)
1	10	AD_ASST	4400
2	20	MK_MAN	13000
3	20	MK_REP	6000
4	50	ST_CLERK	11700
5	50	ST_MAN	5800
6	60	IT_PROG	19200
7	80	SA_MAN	10500
8	80	SA_REP	19600
9	90	AD_PRES	24000
10	90	AD_VP	34000
11	110	AC_ACCOUNT	8300
12	110	AC_MGR	12000
13	(null)	SA_REP	7000

Using the GROUP BY Clause

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

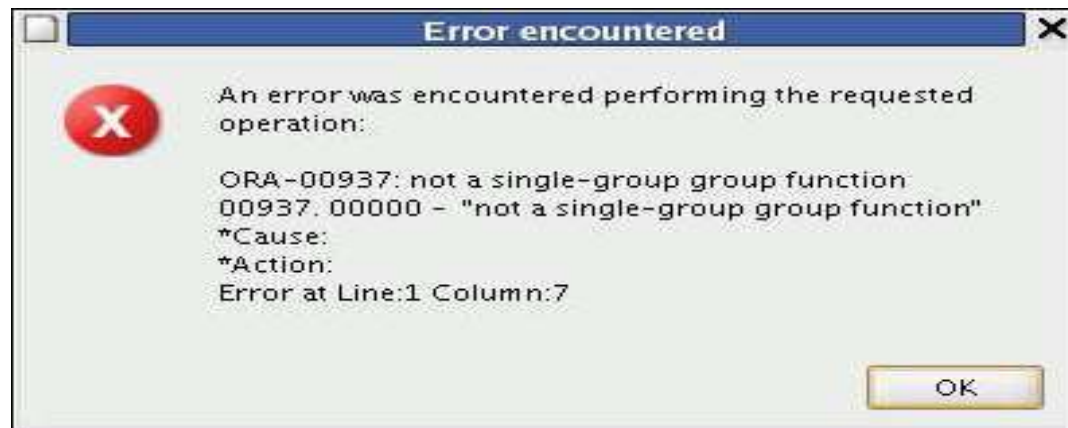
	DEPT_ID	JOB_ID	SUM(SALARY)
1	110	AC_ACCOUNT	8300
2	90	AD_VP	34000
3	50	ST_CLERK	11700
4	80	SA_REP	19600
5	110	AC_MGR	12000
6	50	ST_MAN	5800
7	80	SA_MAN	10500
8	20	MK_MAN	13000
9	90	AD_PRES	24000
10	60	IT_PROG	19200
11	(null)	SA_REP	7000
12	10	AD_ASST	4400
13	20	MK_REP	6000



Illegal Queries

- 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;
```

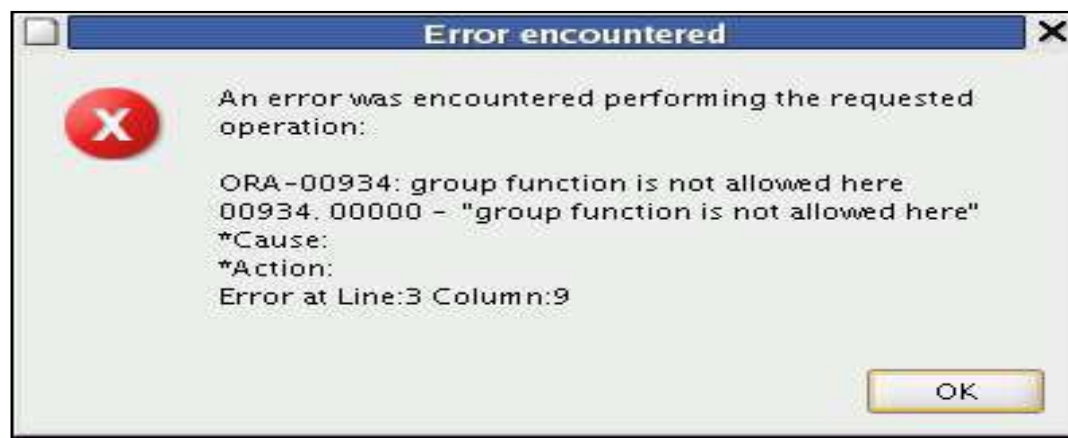


Column missing in the `GROUP BY` clause

Illegal Queries

- 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;
```



**Cannot use the `WHERE` clause
to restrict groups**

Restricting Group Results

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	110	12000
5	110	8300
6	90	24000
7	90	17000
8	90	17000
9	60	9000
10	60	6000
11	60	4200
12	50	5800
13	50	3500
14	50	3100
15	50	2600

...

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

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

Restricting Group Results

When you use the `HAVING` clause, the Oracle server restricts groups as follows:

- Rows are grouped.
- The group function is applied.
- 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)
1	20	13000
2	90	24000
3	110	12000
4	80	11000

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
1	IT_PROG	19200
2	AD_PRES	24000
3	AD_VP	34000

Lesson Agenda

- Group functions:
 - Types and syntax
 - Use AVG, SUM, MIN, MAX, COUNT
 - Use the DISTINCT keyword within group functions
 - NULL values in a group function
- Grouping rows:
 - GROUP BY clause
 - HAVING clause
- Nesting group functions



Nesting Group Functions

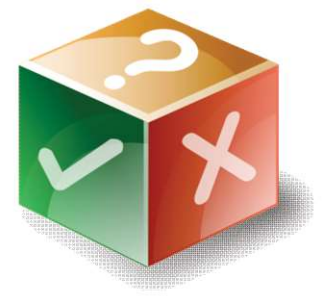
Display the maximum average salary:

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

	MAX(AVG(SALARY))
1	19333.3333333333333333333333333333333333

Identify the two guidelines for group functions and the `GROUP BY` clause.

- a. You cannot use a column alias in the `GROUP BY` clause.
- b. The `GROUP BY` column must be in the `SELECT` clause.
- c. By using a `WHERE` clause, you can exclude rows before dividing them into groups.
- d. The `GROUP BY` clause groups rows and ensures the order of the result set.
- e. If you include a group function in a `SELECT` clause, you must include a `GROUP BY` clause.



Summary

In this lesson, you should have learned how to:

- Use the group functions COUNT, MAX, MIN, SUM, AVG, LISTAGG, STDDEV, and VARIANCE
- Write queries that use the GROUP BY clause
- Write queries that use the HAVING clause

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



Practice 4: Overview

This practice covers the following topics:

- Writing queries that use group functions
- Grouping by rows to achieve more than one result
- Restricting groups by using the `HAVING` clause

