ORACLE

6

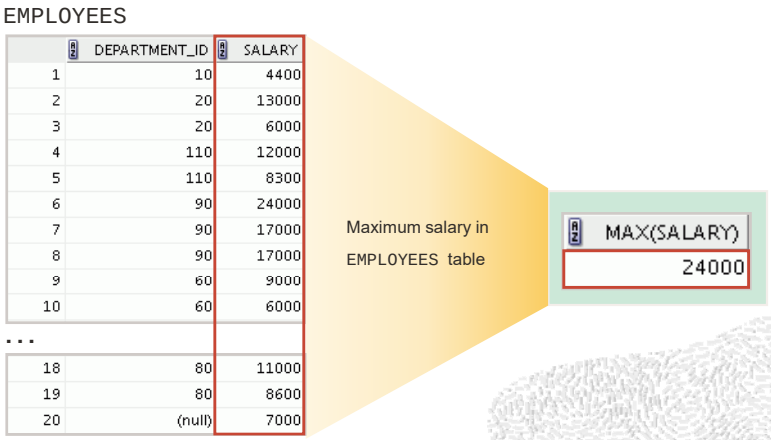
Reporting Aggregated Data Using the Group Functions

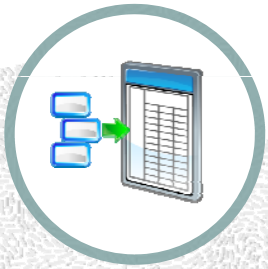
Lesson Agenda

- Group functions
- Grouping rows
- Nesting group functions

Group Functions

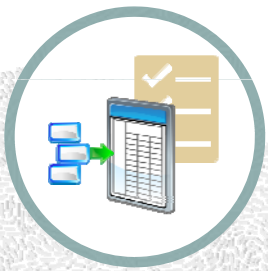
Group functions operate on sets of rows to give one result per group.





Types of Group Functions

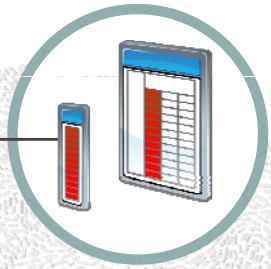
- **AVG**
- **COUNT***
- **MAX**
- **MIN**
- **SUM**
- **LISTAGG**
- **STDDEV**
- **VARIANCE**



Group Functions: Syntax

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

Group all rows in a column



Using the AVG and SUM Functions

You can use the AVG and SUM functions 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

#	AVG(salary)	MAX(salary)	MIN(salary)	SUM(salary)
1	8150.000000	11000.00	6000.00	32600.00

Using the MIN and MAX Functions

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

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

	MIN(LAST_NAME)	MAX(LAST_NAME)	MIN(HIRE_DATE)	MAX(HIRE_DATE)
1	Abel	Zlotkey	13-JAN-09	29-JAN-16

#	MIN(last_name)	MAX(last_name)	MIN(hire_date)	MAX(hire_date)
1	Abel	Zlotkey	2009-01-13	2016-01-29

Using the COUNT Function

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

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

1

	COUNT(*)
1	5

#	count(*)
1	5

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

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

2

	COUNT(COMMISSION_PCT)
1	0

#	COUNT(commission_pc
1	0

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

#	COUNT(DISTINCT department_id
1	7

Group Functions and Null Values in Oracle

- Group functions ignore null values in the column:

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

1

	AVG(COMMISSION_PCT)
1	0.2125

- The NVL function forces group functions to include null values:

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

2

	AVG(NVL(COMMISSION_PCT,0))
1	0.0425

Group Functions and Null Values in MySQL

- Group functions ignore null values in the column:

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

1

#	avg(commission_pc
1	0.212500

- The `IFNULL` function forces group functions to include null values:

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

2

#	AVG(IFNULL(commission_pct, 0
1	0.042500

Lesson Agenda

- Group functions
- Grouping rows
- Nesting group functions

Creating Groups of Data

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	2500
5	50	2600
6	50	3100
7	50	3500
8	50	5800
9	60	9000
10	60	6000
11	60	4200
12	80	11000
13	80	8600
...		
18	110	8300
19	110	12000
20	(null)	7000

4400

9500

3500

6400

10033

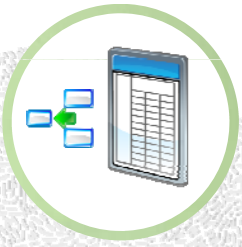
Average salary in the EMPLOYEES table for each department

	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

Creating Groups of Data: GROUP BY Clause Syntax

You can divide the 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 the 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	90	19333.333333333333333333333333333333333
3	20	9500
4	110	10154
5	50	3500
6	80	10033.333333333333333333333333333333333
7	60	6400
8	10	4400

#	department_id	AVG(salary)
1	NULL	7000.000000
2	10	4400.000000
3	20	9500.000000
4	50	3500.000000
5	60	6400.000000
6	80	10033.333333
7	90	19333.333333
8	110	10154.000000

15

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)
1	7000
2	19333.333333333333333333333333333333333
3	9500
4	10154
5	3500
6	10033.333333333333333333333333333333333
7	6400
8	4400

#	AVG(salary)
1	7000.000000
2	4400.000000
3	9500.000000
4	3500.000000
5	6400.000000
6	10033.333333
7	19333.333333
8	10154.000000

16

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
...			
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	110	AC_ACCOUNT	8300
2	110	AC_MGR	12008
3	10	AD_ASST	4400
4	90	AD_PRES	24000
5	90	AD_VP	34000
6	60	IT_PROG	19200
7	20	MK_MAN	13000
8	20	MK_REP	6000
9	80	SA_MAN	10500
10	80	SA_REP	19600
11	(null)	SA_REP	7000
12	50	ST_CLERK	11700
13	50	ST_MAN	5800

Using the GROUP BY Clause on Multiple Columns

```
SELECT   department_id, job_id, SUM(salary)
FROM     employees
WHERE   department_id > 40
GROUP BY department_id, job_id
ORDER BY department_id;
```

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)
1	50	ST_CLERK	11700
2	50	ST_MAN	5800
3	60	IT_PROG	19200
4	80	SA_MAN	10500
5	80	SA_REP	19600
6	90	AD_PRES	24000
7	90	AD_VP	34000
8	110	AC_ACCOUNT	8300
9	110	AC_MGR	12008

#	department_id	job_id	SUM(salary)
1	50	ST_CLERK	11700.00
2	50	ST_MAN	5800.00
3	60	IT_PROG	19200.00
4	80	SA_MAN	10500.00
5	80	SA_REP	19600.00
6	90	AD_PRES	24000.00
7	90	AD_VP	34000.00
8	110	AC_ACC...	8300.00
9	110	AC_MGR	12008.00

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;
```

1

A GROUP BY clause must be added to count the last names for each department_id.

```
ORA-00937: not a single-group group function
00937. 00000 - "not a single-group group function"
```

```
Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains
nonaggregated column 'hr.employees.department_id'; this is incompatible with sql_mode=only_full_group_by
```

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

2

Either add job_id in the GROUP BY clause or remove the job_id column from the SELECT list.

```
ORA-00979: not a GROUP BY expression
00979. 00000 - "not a GROUP BY expression"
```

```
Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated
column 'hr.employees.job_id' which is not functionally dependent on columns in GROUP BY clause; this is
incompatible with sql_mode=only_full_group_by
```

19

Illegal Queries Using Group Functions in a WHERE Clause

- You use the HAVING clause to restrict groups using a group function.
- You cannot use group functions in the WHERE clause.

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

```
ORA-00934: group function is not allowed here
00934. 00000 - "group function is not allowed here"
*Cause:
*Action:
Error at Line: 3 Column: 9
```

Cannot use a group function in a WHERE clause

Error Code: 1111. Invalid use of group function

20

Restricting Group Results

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	2500
5	50	2600
6	50	3100
7	50	3500
8	50	5800
9	60	9000
10	60	6000
11	60	4200
12	80	11000
13	80	8600
...		
18	110	8300
19	110	12000
20	(null)	7000

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

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

Restricting Group Results with the HAVING Clause

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

- 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	90	24000
2	20	13000
3	110	12008
4	80	11000

#	department_id	MAX(salary)
1	20	13000.00
2	80	11000.00
3	90	24000.00
4	110	12008.00

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

#	job_id	PAYROLL
1	IT_PROG	19200.00
2	AD_PRES	24000.00
3	AD_VP	34000.00

Lesson Agenda

- Group functions
- Grouping rows
- Nesting group functions

Nesting Group Functions in Oracle

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

	MAX(AVG(SALARY))
1	19333.333333333333333333333333333333333333

Summary

In this lesson, you should have learned how to:

- Use the group functions `COUNT, MAX, MIN, SUM,` and `AVG`
- 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];
```