Group Functions

Silvana Greca
University of Tirana

Objectives

By the end of this lesson, you will be able to:

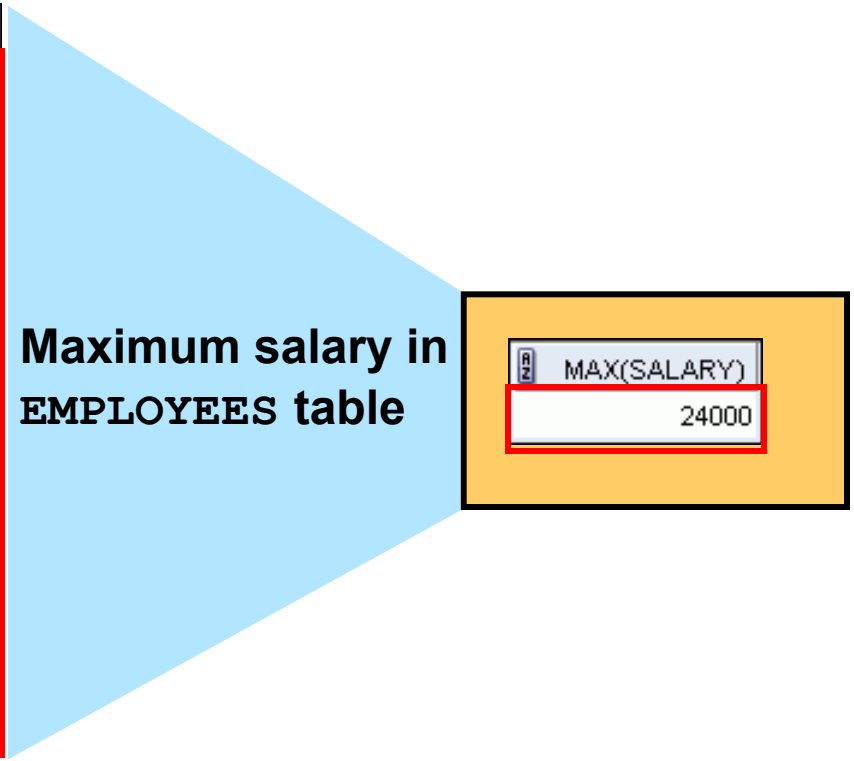
- Identify the different types of group functions
- Describe their purpose and usage
- Apply the GROUP BY clause to aggregate data
- Use the HAVING clause to include or exclude aggregated rows

What Are Group Functions?

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

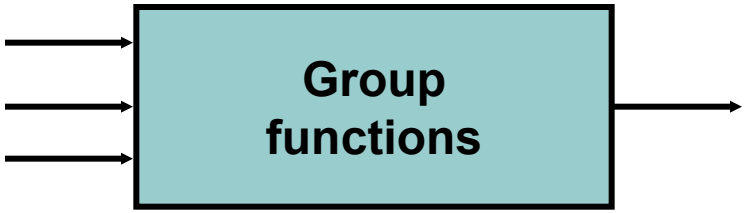
EMPLOYEES

	DEPARTMENT_ID	SALARY
1	90	24000
2	90	17000
3	90	17000
4	60	9000
5	60	6000
6	60	4200
7	50	5800
8	50	3500
9	50	3100
10	50	2600
...		
18	20	6000
19	110	12000
20	110	8300



Types of Group Functions

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



Group Functions: Syntax

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

Query Result

SQL | All Rows Fetched: 1 in 0.005 seconds

	AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
1	8272.727272727272727272727272727272727273	11500	6000	273000

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;
```

Query Result

SQL | All Rows Fetched: 1 in 0.005 seconds

	MIN(HIRE_DATE)	MAX(HIRE_DATE)
1	13-JAN-01	21-APR-08

Using the COUNT Function

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

```
FROM      employees;

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

Query Result | Query Result 1 | Query Result 2

SQL | All Rows Fetched: 1 in 0.003 seconds

	COUNT(*)
1	45

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

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

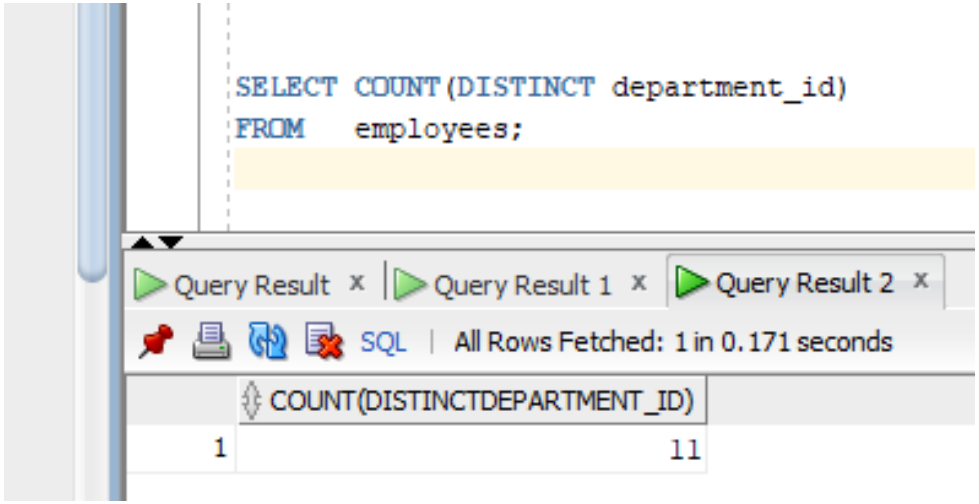
Query Result | Query Result 1 | Query Res

SQL | All Rows Fetched: 1 in 0.002 sec

	COUNT(COMMISSION_PCT)
1	34

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:



Group Functions and Null Values

Group functions ignore null values in the column:

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

Query Result

SQL | All Rows Fetched: 1 in 0.002 sec

	AVG(COMMISSION_PCT)
1	0.222857142857142...

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

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

Query Result

SQL | All Rows Fetched: 1 in 0.136 second

	AVG(NVL(COMMISSION_PCT,...
1	0.072897196261682242...

Creating Groups of Data

EMPLOYEES

	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

4400

9500

3500

6400

10033

Average salary in 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

GROUP BY Clause Syntax

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

You can divide rows in a table into smaller groups by using the GROUP BY clause.

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 ;
```

Query Result

SQL | All Rows Fetched: 12 in 0.111 seconds

	DEPARTMENT_ID	AVG(SALARY)
1	50	3475.555555555555555555555555555555555556
2	40	6500
3	110	10154
4	90	20000
5	30	4150
6	70	10000
7	(null)	7000
8	10	4400
9	20	9500

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 ;
```

Query Result

SQL | All Rows Fetched: 12 in

	AVG(SALARY)
1	3475.555555555555...
2	6500
3	10154
4	20000
5	4150
6	10000
7	7000
8	4400
9	9500

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_MAN	5800
5	50	ST_CLERK	2500
6	50	ST_CLERK	2600
7	50	ST_CLERK	3100
8	50	ST_CLERK	3500
9	60	IT_PROG	4200
10	60	IT_PROG	6000
11	60	IT_PROG	9000
12	80	SA_REP	11000
13	80	SA_MAN	10500
14	80	SA_REP	8600
...			
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

GROUP BY Clause on Multiple Columns

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

Query Result

SQL | All Rows Fetched: 20 in 0.018 seconds

	DEPT_ID	JOB_ID	SUM(SALARY)
1	10	AD_ASST	4400
2	20	MK_MAN	13000
3	20	MK_REP	6000
4	30	PU_CLERK	13900
5	30	PU_MAN	11000
6	40	HR_REP	6500
7	50	SH_CLERK	64300
8	50	ST_CLERK	55700
9	50	ST_MAN	36400

Dbms Output

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;
```

Query Result

SQL | Executing:SELECT department_id, COUNT(last_name)FROM employees in 0 seconds

```
ORA-00937: not a single-group group function
00937. 00000 - "not a single-group group function"
*Cause:
*Action:
Error at Line: 191 Column: 8
```

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

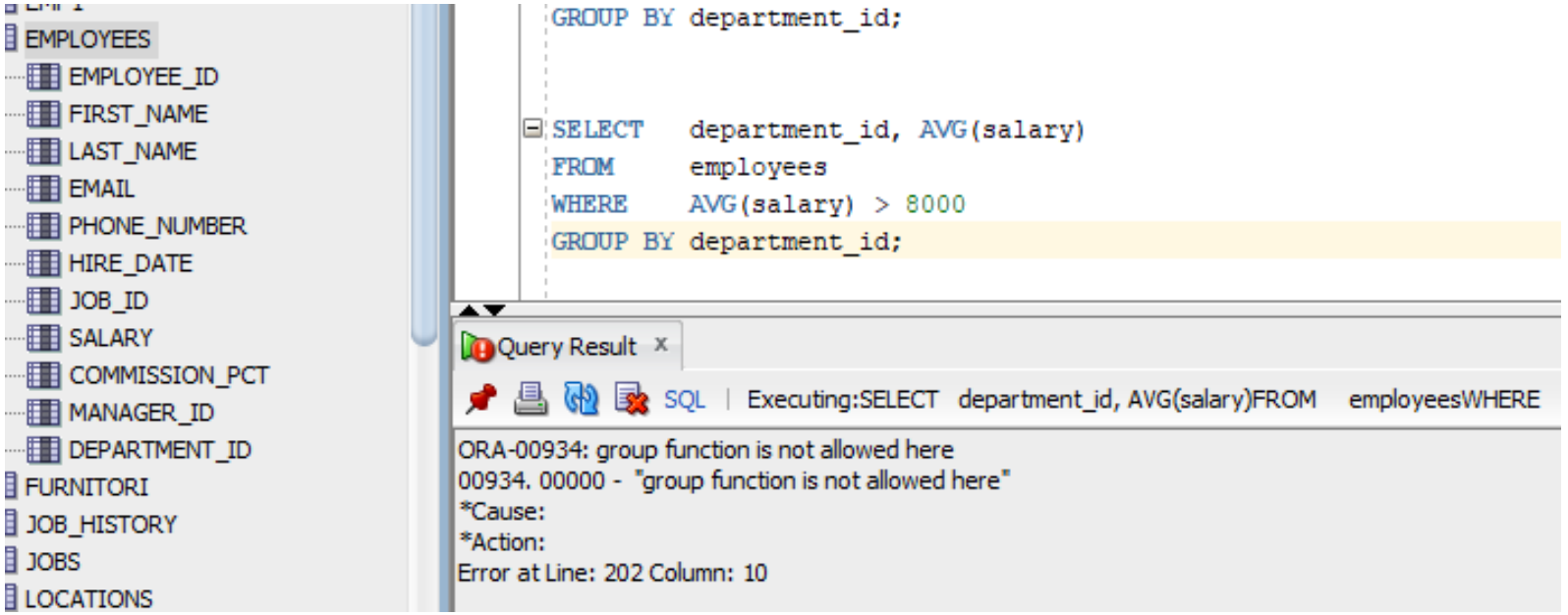
Query Result

SQL | Executing:SELECT department_id, job_id, COUNT(last_name)FROM employeesGROUP BY department_id in 0 seconds

```
ORA-00979: not a GROUP BY expression
00979. 00000 - "not a GROUP BY expression"
*Cause:
*Action:
Error at Line: 195 Column: 23
```

Illegal Queries Using Group Functions

- The WHERE clause cannot be applied to restrict groups.
- The HAVING clause is specifically used to restrict grouped data.
- Group functions are not allowed in the WHERE clause.



Restricting Group Results

EMPLOYEES

	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

...

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	80	11000
3	90	24000
4	110	12000

Restricting Group Results with the `HAVING` Clause

When using the HAVING clause, the Oracle server processes grouped data as follows:

- The rows are grouped.
- Aggregate functions are evaluated.
- Groups that meet the HAVING criteria are returned.

```
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 ;
```

Query Result

SQL | All Rows Fetched: 6 in 0.005 seconds

	DEPARTMENT_ID	MAX(SALARY)
1	110	12008
2	90	26000
3	30	11000
4	20	13000
5	100	12008
6	80	14000

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);
```

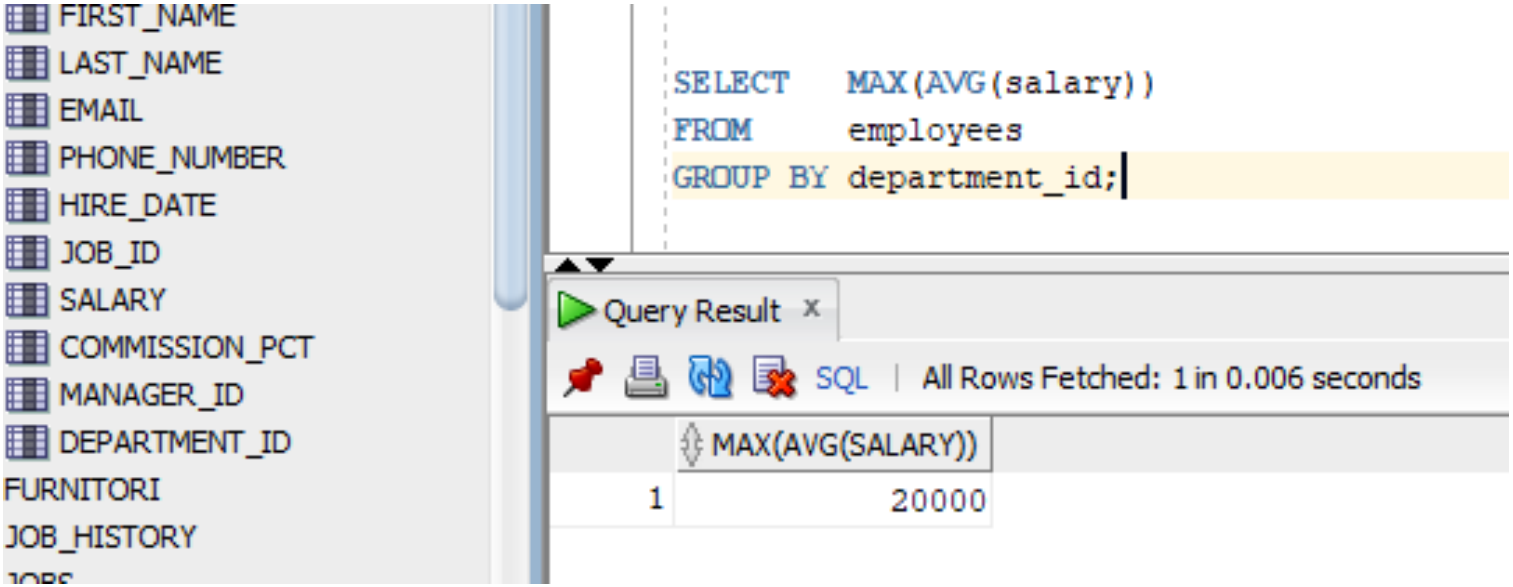
Query Result

SQL | All Rows Fetched: 9 in 0.029 seconds

	JOB_ID	PAYROLL
1	PU_CLERK	13900
2	AD_PRES	26000
3	IT_PROG	28800
4	AD_VP	34000
5	ST_MAN	36400
6	FI_ACCOUNT	39600
7	ST_CLERK	55700
8	SA_MAN	61000
9	SH_CLERK	64300

Nesting Group Functions

Display the maximum average salary:



Summary

In this lesson, you should now be able to:

- Apply the group functions COUNT, MAX, MIN, SUM, and AVG
- Construct queries using the GROUP BY clause
- Construct queries using the HAVING clause

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