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

What Are Group Functions?

- Unlike single-row functions, group 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.

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

Maximum salary in EMPLOYEES table

	MAX(SALARY)
1	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];
```

- **Guidelines for Using Group Functions**
  - `DISTINCT` makes the function consider only non duplicate values; `ALL` makes it consider every value, including duplicates. The default is `ALL` and therefore does not need to be specified.
  - The data types for the functions with an `expr` argument may be `CHAR`, `VARCHAR2`, `NUMBER`, or `DATE`.
  - All group functions ignore null values. To substitute a value for null values, use the `NVL`, `NVL2`, or `COALESCE` functions.

Using the AVG and SUM Functions

- You can use `AVG` and `SUM` for numeric data.
- Displays the average, highest, lowest, and sum of monthly salaries for all sales representatives. A1

```
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

- Note: You can use `AVG`, `SUM`, `MIN`, and `MAX` functions against columns that can store numeric data.

Using the MIN and MAX Functions

- You can use MIN and MAX for numeric, character, and date data types.
- Displays the most junior and most senior employees. A2

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

MIN(HIRE_DATE)	MAX(HIRE_DATE)
1 17-JUN-87	29-JAN-00

- Displays the employee last name that is first and the employee last name that is last in an alphabetized list of all employees: A3

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

- **Note:** The AVG, SUM, VARIANCE, and STDDEV functions can be used only with numeric data types. MAX and MIN cannot be used with LOB or LONG data types.

Using the COUNT Function

- The COUNT function has three formats:
  - COUNT(*)
  - COUNT(*expr*)
  - COUNT(DISTINCT *expr*)
- COUNT(*) returns the number of rows in a table that satisfy the criteria of the SELECT statement, including duplicate rows and rows containing null values in any of the columns.
- If a WHERE clause is included in the SELECT statement, COUNT(*) returns the number of rows that satisfy the condition in the WHERE clause.
- In contrast, COUNT(*expr*) returns the number of non-null values that are in the column identified by *expr*.
- COUNT(DISTINCT *expr*) returns the number of unique, non-null values that are in the column identified by *expr*.

- COUNT(*) returns the number of rows in a table:
- Displays the number of employees in department 50. A4

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

COUNT(*)
1 5

- COUNT(expr) returns the number of rows with non-null values for expr:
- Displays the number of employees in department 80 who can earn a commission.A5

```
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: A6

```
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:A7

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

AVG(COMMISSION_PCT)
1 0.2125

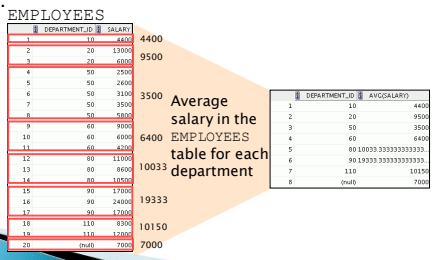
- The NVL function forces group functions to include null values:A8

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

AVG(NVL(COMMISSION_PCT,0))
1 0.0425

Creating Groups of Data

- All group functions have treated the table as one large group of information.
- However, you need to divide the table of information into smaller groups. You can do this by using the GROUP BY clause.



Creating Groups of Data: GROUP BY Clause Syntax

- You can use the GROUP BY clause to divide the rows in a table into groups. You can then use the group functions to return summary information for each group.

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

- *group_by_expression:* specifies columns whose values determine the basis for grouping rows

- **Guidelines**
  - If you include a group function in a SELECT clause, you cannot select individual results as well, *unless* the individual column appears in the GROUP BY clause. You receive an error message if you fail to include the column list in the GROUP BY clause.
  - Using a WHERE clause, you can exclude rows before dividing them into groups.
  - You must include the *columns* in the GROUP BY clause.
  - You cannot use a column alias in the GROUP BY clause.

Using the GROUP BY Clause

- When using the GROUP BY clause, make sure that all columns in the SELECT list that are not group functions are included in the GROUP BY clause.
- Displays the department number and the average salary for each department. G1

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

	DEPARTMENT_ID	AVG(SALARY)
1	(null)	7000
2	20	9500
3	50	3500
4	60	6400
5	80	10033.33333333333333333333333333333333333333
6	90	19333.33333333333333333333333333333333333333
7	110	10150
8	(null)	7000

```
SELECT    department_id, salary
FROM      employees
GROUP BY department_id ;
```


Using the GROUP BY Clause

- The GROUP BY column does not have to be in the SELECT list.
- Displays the average salaries for each department without displaying the respective department numbers.

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

	AVG(SALARY)
1	7000
2	9500
3	19333.33333333333333333333333333333333333333
4	10150
5	3500
6	10033.33333333333333333333333333333333333333
7	4400
8	6400

Results do not look meaningful

- You can use the group function in the ORDER BY clause: G2

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

DEPARTMENT_ID	AVG(SALARY)
50	3475.55555555555555555555555555555555555556
30	4150
10	4400
60	5760
40	6500
-	7000
100	8600
80	8955.88235294117647058823529411764705882
20	9500
70	10000

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

ORA-00979: not a GROUP BY expression

Grouping by More Than One Column: Groups Within Groups

- Sometimes you need to see results for groups within groups.
- Displays the total salary that is paid to each job title in each department.

EMPLOYEES

DEPARTMENT_ID	JOB_ID	SALARY
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	2500
50	ST_CLERK	2600
50	ST_CLERK	3100
50	ST_CLERK	3500
50	ST_MAN	5800
60	IT_PROG	9000
60	IT_PROG	6000
60	IT_PROG	4200
80	SA_REP	11000
80	SA_REP	8600
80	SA_MAN	10500
90	AD_VP	17000
90	AD_PRES	24000
90	AD_VP	17000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
(null)	SA_REP	7000

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

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
(null)	SA_REP	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 ;
```

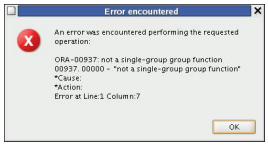
G3

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

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: G4

```
SELECT department_id, COUNT(last_name)
FROM   employees;
```



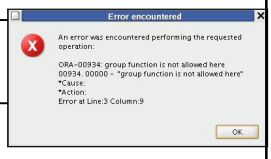
- Whenever you use a mixture of individual items (DEPARTMENT_ID) and group functions (COUNT) in the same SELECT statement, you must include a GROUP BY clause that specifies the individual items (in this case, DEPARTMENT_ID).
- If the GROUP BY clause is missing, then the error message "not a single-group group function" appears. You can correct the error in the slide by adding the GROUP BY clause: G5

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


Illegal Queries Using Group Functions

- Display of average salaries of those departments that have an average salary greater than $8,000.

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



- 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.

- You can correct the error in the example by using the HAVING clause to restrict groups G6

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

Restricting Group Results

- In the same way that you use the WHERE clause to restrict the rows that you select, you use the HAVING clause to restrict groups.
- To find the maximum salary in each of the departments that have a maximum salary greater than $10,000, you need to do the following:
  1. Find the average salary for each department by grouping by department number.
  2. Restrict the groups to those departments with a maximum salary greater than $10,000.

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 with the HAVING Clause

- 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

- Displays department numbers and maximum salaries for those departments with a maximum salary that is greater than $10,000. G7

```
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

- Displays the job ID and total monthly salary for each job that has a total payroll exceeding $13,000. G8

```
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