

Group Functions

Group Functions

- Group functions
 - Operate on a set of rows
 - Return a single result
 - Aggregate functions
 - Ignore NULL values, except COUNT(*)

Function	Returns
AVG	The average value of a given column
COUNT(column_name)	The total number of values in a given column, excluding NULL values
COUNT(*)	The total number of rows in a table
MAX	The largest value in a given column
MIN	The smallest value in a given column
SUM	The sum of the numeric values in a given column

Figure 11-1: SQL aggregate functions

MIN and MAX Functions

- Returns the minimum and maximum value from a set of rows
- Can be used with **any data type**

Example 11-1 MIN Function

Use MIN function to return the lowest (minimum) salary

Data Set ds11_1



EMPLOYEE_ID	DEPARTMENT_ID	SALARY
-----	-----	-----
10	200	108521
11	NULL	NULL
12	200	127644
13	100	94732
14	200	126296
15	300	54870
16	200	64631
17	200	45049
18	100	65650
19	200	81271

SQL Statement

```
SELECT MIN( salary ) AS "Lowest salary"  
FROM ds11_1;
```

Result Set

Lowest salary

45049

Example 11-2 MAX Function

Use MAX function to return the highest (maximum) salary

Data Set ds11_2

EMPLOYEE_ID	DEPARTMENT_ID	SALARY
10	200	108521
11	NULL	NULL
12	200	127644
13	100	94732
14	200	126296
15	300	54870
16	200	64631
17	200	45049
18	100	65650
19	200	81271

SQL Statement

```
SELECT MAX( salary ) AS "Highest salary"  
FROM ds11_2;
```

Result Set

Highest salary
127644

MIN & MAX Functions with Dates

Example 11-13 Use the MIN and MAX functions to return the earliest (minimum) and latest (maximum) dates

Data Set ds11_13

EMPLOYEE_ID	HIRE_DATE
10	2018-05-20
11	2019-10-23 (Latest hire date)
12	2017-12-18 (Earliest hire date)
13	2018-10-23

SQL Statement

```
SELECT
    MIN(hire_date) AS earliest_hire_date,
    TO_CHAR( MAX(hire_date), 'mm/dd/yyyy' ) AS latest_hire_date
FROM ds11_13;
```

Result Set

EARLIEST_HIRE_DATE	LATEST_HIRE_DATE
2017-12-18	10/23/2019

AVG Function

Example 11-3 AVG Function

Use AVG function to return the average salary

Data Set ds11_3

EMPLOYEE_ID	DEPARTMENT_ID	SALARY
10	200	108521
11	NULL	NULL
12	200	127644
13	100	94732
14	200	126296
15	300	54870
16	200	64631
17	200	45049
18	100	65650
19	200	81271

SQL Statement

```
SELECT AVG( salary ) AS "Average salary"  
FROM ds11_3;
```

Result Set

Average salary
85407.11111111111111111111111111

SUM Function

Example 11-6
SUM Function

Return the total salary for all employees

Data Set ds11_6

EMPLOYEE_ID	DEPARTMENT_ID	SALARY
-----	-----	-----
10	200	108521
11	NULL	NULL
12	200	127644
13	100	94732
14	200	126296
15	300	54870
16	200	64631
17	200	45049
18	100	65650
19	200	81271

SQL Statement

```
SELECT SUM( salary ) AS "Total Salary"  
FROM ds11_6;
```

Result Set

Total Salary

768664

Types of COUNT Functions

- There are two variations of the COUNT function:
 - COUNT(*)
 - Count total rows
 - Does not consider any data, including NULLs
 - COUNT(column_name)
 - Count rows that contain non-NULL values
 - Ignores NULL values

Example 11-4
COUNT(*)
Function

Use COUNT(*) function to return the number of employees

Data Set ds11_4

EMPLOYEE_ID	DEPARTMENT_ID	SALARY
-----	-----	-----
10	200	108521
11	NULL	NULL
12	200	127644
13	100	94732
14	200	126296
15	300	54870
16	200	64631
17	200	45049
18	100	65650
19	200	81271

SQL Statement

```
SELECT COUNT(*) AS "Count"  
FROM ds11_4;
```

Result Set

Count

10

Example 11-5
Count(COLUMN_NAME)
Function

Use COUNT(column_name) function to return the number of employees that have been assigned to a department. That is, the department id does not contain NULL.

Data Set ds11_5

EMPLOYEE_ID	DEPARTMENT_ID	SALARY
-----	-----	-----
10	200	108521
11	NULL	NULL
12	200	127644
13	100	94732
14	200	126296
15	300	54870
16	200	64631
17	200	45049
18	100	65650
19	200	81271

SQL Statement

```
SELECT COUNT(department_id) AS "Emp Count"  
FROM ds11_5;
```

Result Set

Emp Count

9

COUNT & SUM Functions

Example 11-7
SUM & COUNT
Functions

Use COUNT and SUM functions to return a string containing a count of the number of employees and total salary

Data Set ds11_7

EMPLOYEE_ID	DEPARTMENT_ID	SALARY
10	200	108521
11	NULL	NULL
12	200	127644
13	100	94732
14	200	126296
15	300	54870
16	200	64631
17	200	45049
18	100	65650
19	200	81271

SQL Statement

```
SELECT 'Total salaries for ' || COUNT(*) || ' employees is ' ||  
       SUM( salary ) AS "Total Salary"  
FROM ds11_7;
```

Result Set

```
Total Salary  
-----  
Total salaries for 10 employees is 768664
```


Group Functions & NULLs

Example 11-8

Using COUNT (*) and SUM functions together with NULL values. Return employee count, average salary, and total salary.

Note: This query return incorrect results

Data Set ds11_8

EMPLOYEE_ID	DEPARTMENT_ID	SALARY
-----	-----	-----
10	200	108521
11	NULL	NULL
12	200	127644
13	100	94732
14	200	126296
15	300	54870
16	200	64631
17	200	45049
18	100	65650
19	200	81271

SQL Statement

```
SELECT COUNT(*) AS "Employee Count",  
       DECIMAL( AVG( salary ), 7,2) AS "Average Salary",  
       SUM( salary ) AS "Total Salary"  
FROM ds11_8;
```

Result Set

Employee Count	Average Salary	Total Salary
-----	-----	-----
10	85407.11	768664

Example 11-9

Using COUNT (column_name) and SUM functions together with NULL values. Return employee count (salary column), average salary, and total salary.

Data Set ds11_9

EMPLOYEE_ID	DEPARTMENT_ID	SALARY
-----	-----	-----
10	200	108521
11	NULL	NULL
12	200	127644
13	100	94732
14	200	126296
15	300	54870
16	200	64631
17	200	45049
18	100	65650
19	200	81271

SQL Statement

```
SELECT COUNT(salary) AS "Employee Count",  
       DECIMAL( AVG( salary ), 7,2) AS "Average Salary",  
       SUM( salary ) AS "Total Salary"  
FROM ds11_9;
```

Result Set

Employee Count	Average Salary	Total Salary
-----	-----	-----
9	85407.11	768664

Example 11-10

Using IS NOT NULL in a search condition Return employee count, average salary, and total salary where salary is not null.

Data Set ds11_10

EMPLOYEE_ID	DEPARTMENT_ID	SALARY
-----	-----	-----
10	200	108521
11	NULL	NULL
12	200	127644
13	100	94732
14	200	126296
15	300	54870
16	200	64631
17	200	45049
18	100	65650
19	200	81271

SQL Statement

```
SELECT COUNT(*)                AS "Employee Count",  
       DECIMAL( AVG( salary ), 7,2) AS "Average Salary",  
       SUM( salary )           AS "Total Salary"  
FROM ds11_10  
WHERE salary IS NOT NULL;
```

Result Set

Employee Count	Average Salary	Total Salary
-----	-----	-----
9	85407.11	768664

COALESCE with Group Functions

- With group functions, the COALESCE function is nested inside the group function
- When it is necessary to include rows with NULL values, use the COALESCE function to add a value to the NULL rows

Using COALESCE with group functions

```
SELECT commission_pct  
FROM employees;
```

```
COMMISSION_PCT  
-----  
NULL  
NULL  
NULL  
NULL  
NULL  
NULL  
0.20  
0.30  
0.20  
0.15  
NULL  
NULL  
NULL  
NULL  
NULL  
NULL  
NULL  
NULL  
NULL  
NULL
```

COALESCE

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

AVG(COMMISSION_PCT)	AVG(COALESCE(COMMISSION_PCT,0))
.2125	.0425

Example 11-11

Use the MIN and MAX functions to return the lowest (MIN) and highest (MAX) salary values. Use COALESCE to set new values when salary contains NULL.

Data Set ds11_11

EMPLOYEE_ID	DEPARTMENT_ID	SALARY
10	200	108521
11	NULL	NULL
12	200	127644
13	100	94732
14	200	126296
15	300	54870
16	200	64631
17	200	45049
18	100	65650
19	200	81271

SQL Statement

```
SELECT
  MIN( COALESCE(salary, 30000) ) AS min_salary,
  MAX( COALESCE(salary, 200000) ) AS max_salary
FROM ds11_11;
```

Result Set

MIN_SALARY	MAX_SALARY
30000	200000

Example 11-12

Use the COALESCE function to set a new value for null values

Data Set ds11_12

EMPLOYEE_ID	DEPARTMENT_ID	SALARY
10	200	108521
11	NULL	NULL
12	200	127644
13	100	94732
14	200	126296
15	300	54870
16	200	64631
17	200	45049
18	100	65650
19	200	81271

SQL Statement

```
SELECT
  SUM(salary) AS total_salary,
  COUNT(salary) AS emp_count,
  DECIMAL( AVG(salary), 7,2 ) AS avg_no_null,
  DECIMAL( AVG(COALESCE(salary, 0)), 7,2 ) AS avg_with_null
FROM ds11_12;
```

Result Set

TOTAL_SALARY	EMP_COUNT	AVG_NO_NULL	AVG_WITH_NULL
768664	9	85407.11	76866.40

DISTINCT with Group Functions

- DISTINCT can be used with all group functions
- DISTINCT makes the function consider only non-duplicate values

Example 11-14

Use the DISTINCT keyword with the COUNT(department_id) function to return the number of different departments

Data Set ds11_14

EMPLOYEE_ID	DEPARTMENT_ID	SALARY
10	200	108521
11	NULL	NULL
12	200	127644
13	100	94732
14	200	126296
15	300	54870
16	200	64631
17	200	45049
18	100	65650
19	200	81271

SQL Statement

```
SELECT COUNT(DISTINCT department_id) AS "Dept Count"  
FROM ds11_14;
```

Result Set

Dept Count
3

More Than One GROUP Function

- Can have more than one group function in the SELECT clause, on the same or different columns

```
SELECT DECIMAL( AVG(salary), 7,2) AS avg_salary,  
       MIN(salary) AS low_salary,  
       MAX(salary) AS high_salary,  
       TO_CHAR( MIN(hire_date), 'mm/dd/yyyy' ) AS low_hire_date  
FROM employees  
WHERE department_id = 50;
```

AVG_SALARY	LOW_SALARY	HIGH_SALARY	LOW_HIRE_DATE
3500.00	2500.00	5800.00	10/17/1995

Group Function Caution

- Important things you should know about group functions:
 - Group functions cannot be used in the WHERE clause:

```
SELECT last_name, first_name  
FROM employees  
WHERE salary = MIN(salary);
```



ORA-00934: group function is not allowed here

Rules for GROUP Functions

- Used in the SELECT clause
- **Cannot** be used in the WHERE clause
- Returns one result
- Group functions ignore NULL values except COUNT(*)

Rules for GROUP Functions

- MIN and MAX can be used with any data type
- SUM and AVG can be used only with numeric data types
- Use DISTINCT to suppress duplicate values

