

THE CASE STATEMENT



```
SELECT
    column_name(s)
    CASE
        WHEN condition_1 THEN result_1
        WHEN condition_2 THEN result_2
        ...
        ELSE
    END AS
FROM
    table_name;
```

The Case Statement is used within a Select Statement when we want to return a specific value based on a condition. Its syntax can vary depending on what we want to show.

A screenshot of the SQL Developer interface. The code editor shows a SELECT statement with a CASE expression:

```
1 • SELECT
2     emp_no,
3     first_name,
4     last_name,
5     CASE
6         WHEN gender = 'M' THEN 'Male'
7         ELSE 'Female'
8     END AS gender
9
10    FROM
11        employees;
12
13 • SELECT
14     emp_no,
```

The CASE block is highlighted with a blue rectangle. To the right is a Result Grid showing employee data with the 'gender' column highlighted:

emp_no	first_name	last_name	gender
10003	Porto	Bamford	Male
10004	Christian	Koblick	Male
10005	Kvočka	Málik	Male
10006	Anneke	Preusko	Female
10007	Tsvetan	Zelinski	Female
10008	Sanive	Kelloufi	Male

CASE
WHEN... THEN...
ELSE...
END AS...

In this example, when the value of the column is M, it will return "Male", and if it is F, "Female". As simple as that. The syntax of the case constructs starts with the keyword "Case" followed by "When", and a conditional expression containing the word "Then". After that, we have "Else" as a final expression if all conditions mentioned turn out false. Furthermore, "End" is an obligatory part of the syntax.

A screenshot of the SQL Developer interface, similar to the previous one, but with the entire CASE block wrapped in a BEGIN...END block:

```
12
13 • SELECT
14     emp_no,
15     first_name,
16     last_name,
17     CASE gender
18         WHEN 'M' THEN 'Male'
19         ELSE 'Female'
20     END AS gender
21
22    FROM
23        employees;
```

The entire CASE block is highlighted with a large blue rectangle. Below is a Result Grid showing the same employee data, with the 'gender' column highlighted:

emp_no	first_name	last_name	gender
10001	Georgi	Pacello	Male
10002	Bezalel	Simmel	Female
10003	Porto	Bamford	Male
10004	Christian	Koblick	Male
10005	Kvočka	Málik	Male
10006	Anneke	Preusko	Female

Now, to explore another way in which we can rewrite the Case Statement. We can obtain the same result by putting the name of the column once right after the word "Case". Then, we should write the corresponding value after the "WHEN" keyword without using the equals operator. When using the IF NOT NULL syntax, we can't do what we did here and place 'gender' after 'CASE' – the screenshot below is an example of when we can't do this but it's not that deep – we're pretty advanced at reading SQL code now.

```

25 • SELECT
26     e.emp_no,
27     e.first_name,
28     e.last_name,
29     CASE
30         WHEN dm.emp_no IS NOT NULL THEN 'Manager'
31         ELSE 'Employee'
32     END AS is_manager
33 FROM
34     employees e
35     LEFT JOIN
36     dept_manager dm ON dm.emp_no = e.emp_no
37 WHERE
38     e.emp_no > 109990;

```

```

39
40
41 • SELECT
42     emp_no,
43     first_name,
44     last_name,
45     IF(gender = 'M', 'Male', 'Female') AS gender
46 FROM condition

```

IF()

The first expression within the parentheses is the condition which we want to be true. If it is true, then the return value will be the second expression of this construct. If it's false, the return value will be the one written in the third place, being 'Female', so this structure is an unspoken IF THEN ELSE setup.

Client Connections	35	LEFT JOIN
Users and Privileges	36	dept_manager dm ON dm.emp_no = e.emp_no
Status and System Variables	37	WHERE
Data Export	38	e.emp_no >
Data Import/Restore	39	
NICE	40	
Startup / Shutdown	41 • SELECT	
Server Logs	42	emp_no,
Op	43	first_name,
GM	44	last_name,
Da		Male
Per		
Per		
MAS		

IF VS CASE

you can have just one conditional expression

we can have multiple conditional expressions

```

50 • SELECT
51     dm.emp_no,
52     e.first_name,
53     e.last_name,
54     MAX(s.salary) - MIN(s.salary) AS salary_difference,
55     CASE
56         WHEN MAX(s.salary) - MIN(s.salary) > 30000 THEN 'Salary was raised by more than $30,000'
57         WHEN MAX(s.salary) - MIN(s.salary) BETWEEN 20000 AND 30000 THEN
58             'Salary was raised by more than $20,000 but less than $30,000'
59         ELSE 'Salary was raised by less than $20,000'
60     END AS salary_increase
61 FROM
62     dept_manager dm
63     JOIN
64     employees e ON e.emp_no = dm.emp_no
65     JOIN
66     salaries s ON s.emp_no = dm.emp_no
67 GROUP BY s.emp_no;

```

The SQL CASE Statement - Exercise #1

Retrieve the employee number (`emp_no`), first name (`first_name`), and last name (`last_name`) of all employees from the `employees` table whose employee number is greater than 10005. Join this information with the data from the department manager `dept_manager` table to add a fourth column named `is_manager`, containing the string '`Manager`' if the employee number of the given employee is not a null value, and '`Employee`' otherwise.

This topic is covered in

Lecture 258: The SQL CASE Statement

```
1 SELECT
2 e.emp_no,
3 e.first_name,
4 e.last_name,
5 CASE
6 WHEN dm.emp_no IS NOT NULL THEN 'Manager'
7 ELSE 'Employee'
8 END AS is_manager
9 FROM
10 employees e
11 LEFT JOIN
12 dept_manager dm
13 ON e.emp_no = dm.emp_no
14 WHERE e.emp_no > 10005;
15
```

Run query Reset

Result Success

The SQL CASE Statement - Exercise #2

Your analytics task is to decide if the salary raises of all managers whose employee numbers over 10005 have been significant. To do this, you need to retrieve the following table containing eight columns:

emp_no	first_name	last_name	hire_date	min_salary	max_salary	salary_difference	salary_raise
10006	Anatole	Rendfurd	1986-06-02	40000	60098	20098	significant
10007	Towain	Zwolenski	1986-07-10	50724	88070	37346	significant
10008	Savva	Kelsoff	1984-09-18	46671	52668	5997	insignificant

1. `emp_no` from the `dept_manager` table.
2. `first_name`, `last_name`, and `hire_date` from the `employees` table.
3. `min_salary`, `max_salary`, and `salary_difference` from the `salaries` table.

`salary_raise` to indicate whether the raise is

```
1 SELECT
2 dm.emp_no,
3 e.first_name,
4 e.last_name,
5 e.hire_date,
6 MIN(s.salary) AS min_salary,
7 MAX(s.salary) AS max_salary,
8 MAX(s.salary) - MIN(s.salary) AS salary_difference,
9 CASE
10 WHEN MAX(s.salary) - MIN(s.salary) <= 10000 AND MAX(s.salary) - MIN(s.salary) > 0
11 THEN 'insignificant'
12 WHEN MAX(s.salary) - MIN(s.salary) > 10000 THEN 'significant'
13 ELSE 'salary decrease'
14 END AS salary_raise
15 FROM
16 dept_manager dm
17 JOIN
18 employees e ON dm.emp_no = e.emp_no
19 JOIN
20 salaries s ON s.emp_no = dm.emp_no
```

Run query Reset

Result Success

The full syntax is on MySQL

The SQL CASE Statement - Exercise #3

Retrieve the employee number (`emp_no`), first name (`first_name`), and last name (`last_name`) of all employees from the `employees` table who also have records in the department employees table `dept_emp`. Add a fourth column named `current_status` displaying "Currently working" if their contract in the `dept_emp` table ends on or after January 1, 2025, or later. Otherwise, display "No longer with the company". Use GROUP BY on the employee number, first name, and last name to obtain the desired result.

This topic is covered in

Lecture 258: The SQL CASE Statement

```
1 SELECT
2 e.emp_no,
3 e.first_name,
4 e.last_name,
5 CASE
6 WHEN MAX(de.to_date) > '2024-12-31' THEN "Currently working"
7 ELSE "No longer with the company"
8 END AS current_status
9 FROM
10 employees e
11 JOIN
12 dept_emp de
13 ON e.emp_no = de.emp_no
14 GROUP BY e.emp_no, e.first_name, e.last_name;
```

Run query Reset

Result Success