

CASE statement in SQL

- SQL also has a CASE construct
- Used when a value can be different based on certain conditions.
- Can be used in any part of an S Q L query where a value is expected
- Applicable when querying, inserting, or updating tuples

Suppose you have an EmployeeCase table.

```
CREATE TABLE EmployeeCase (  
EmployeeID INT,  
EmployeeName VARCHAR(100) NOT NULL,  
Gender VARCHAR(1) NOT NULL,  
StateCode VARCHAR(20) NOT NULL,  
Salary INT NOT NULL,  
PRIMARY KEY (EmployeeID)  
);  
/
```

```
INSERT INTO EmployeeCASE VALUES (201, 'Jerome', 'M', 'FL', 83000);  
INSERT INTO EmployeeCASE VALUES (202, 'Ray', 'M', 'AL', 88000);  
INSERT INTO EmployeeCASE VALUES (203, 'Stella', 'F', 'AL', 76000);  
INSERT INTO EmployeeCASE VALUES (204, 'Gilbert', 'M', 'Ar', 42000);  
INSERT INTO EmployeeCASE VALUES (205, 'Edward', 'M', 'FL', 93000);  
INSERT INTO EmployeeCASE VALUES (206, 'Ernest', 'F', 'Al', 64000);  
INSERT INTO EmployeeCASE VALUES (207, 'Jorge', 'F', 'IN', 75000);  
INSERT INTO EmployeeCASE VALUES (208, 'Nicholas', 'F', 'Ge', 71000);  
INSERT INTO EmployeeCASE VALUES (209, 'Lawrence', 'M', 'IN', 95000);  
INSERT INTO EmployeeCASE VALUES (210, 'Salvador', 'M', 'Co', 75000);
```

```
select * from EmployeeCASE;/
```

201	Jerome	M	FL	83000
202	Ray	M	AL	88000
203	Stella	F	AL	76000
204	Gilbert	M	Ar	42000
205	Edward	M	FL	93000
206	Ernest	F	Al	64000
207	Jorge	F	IN	75000
208	Nicholas	F	Ge	71000
209	Lawrence	M	IN	95000
210	Salvador	M	Co	75000

A simple CASE statement expression

In this format, we evaluate one expression against multiple values. In a simple case statement, it evaluates conditions one by one. Once the condition and expression are matched, it returns the expression mentioned in THEN clause.

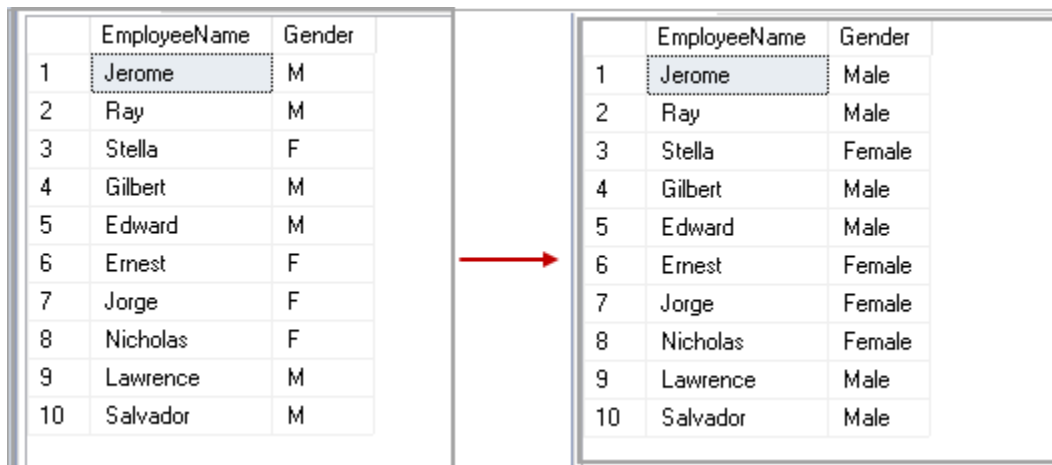
We have following syntax for a case statement in SQL with a simple expression

```
SELECT CASE Expression
When expression1 Then Result1
When expression2 Then Result2
...
ELSE Result
END
```

Usually, we store abbreviations in a table instead of its full form. For example, in my Employee table, I have used abbreviations in Gender and StateCode. I want to use a Case statement to return values as **Male** and **Female** in the output instead of **M** and **F**.

Execute the following code and notice that we want to evaluate **CASE Gender** in this query.

In the following image, you can notice a difference in output using a Case statement in SQL.



	EmployeeName	Gender
1	Jerome	M
2	Ray	M
3	Stella	F
4	Gilbert	M
5	Edward	M
6	Ernest	F
7	Jorge	F
8	Nicholas	F
9	Lawrence	M
10	Salvador	M

	EmployeeName	Gender
1	Jerome	Male
2	Ray	Male
3	Stella	Female
4	Gilbert	Male
5	Edward	Male
6	Ernest	Female
7	Jorge	Female
8	Nicholas	Female
9	Lawrence	Male
10	Salvador	Male

The CASE statement and comparison operator

In this format of a CASE statement in SQL, we can evaluate a condition using comparison operators. Once this condition is satisfied, we get an expression from the corresponding THEN in the output.

We can see the following syntax for Case statement with a comparison operator.

```

CASE
  WHEN ComparisionCondition THEN result
  WHEN ComparisionCondition THEN result
  ELSE other
END

```

Suppose we have a salary band for each *designation*. If employee salary is in between a particular range, we want to get designation using a Case statement.

In the following query, we are using a comparison operator and evaluate an expression.

```

Select EmployeeName,
CASE
  WHEN Salary >=80000 AND Salary <=100000 THEN 'Director'
  WHEN Salary >=50000 AND Salary <80000 THEN 'Senior Consultant'
  Else 'Director'
END AS Designation
from Employee

```

In the following image you can see, we get designation as per the condition specified in CASE statement.

	EmployeeName	Salary	Designation
1	Jerome	83000.00	Director
2	Ray	88000.00	Director
3	Stella	76000.00	Senior Consultant
4	Gilbert	42000.00	Director
5	Edward	93000.00	Director
6	Ernest	64000.00	Senior Consultant
7	Jorge	75000.00	Senior Consultant
8	Nicholas	71000.00	Senior Consultant
9	Lawrence	95000.00	Director
10	Salvador	75000.00	Senior Consultant

Case Statement with Order by clause

We can use Case statement with order by clause as well. In SQL, we use Order By clause to sort results in ascending or descending order.

Suppose in a further example; we want to sort result in the following method.

1. For Female employee, employee salaries should come in descending order
2. For Male employee, we should get employee salaries in ascending order

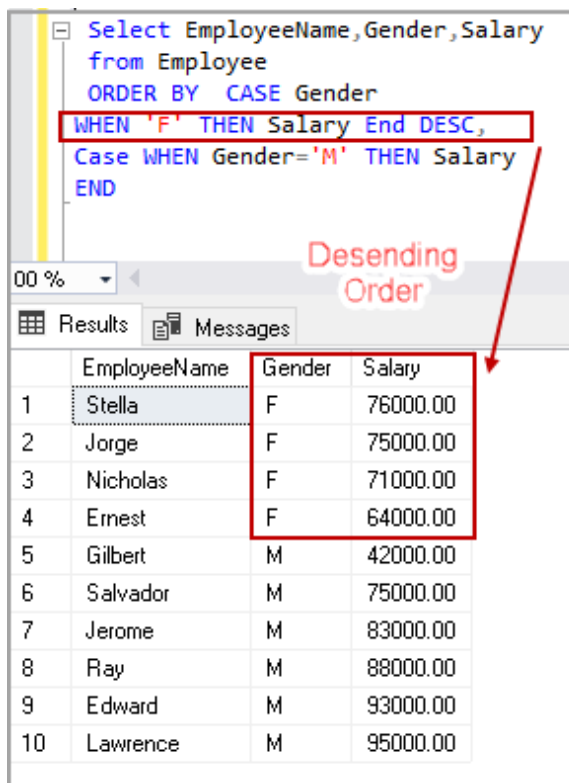
We can define this condition with a combination of Order by and Case statement. In the following query, you can see we specified Order By and Case together. We defined sort conditions in case expression.

```
Select EmployeeName, Gender, Salary
from Employee
ORDER BY CASE Gender
WHEN 'F' THEN Salary End DESC,
Case WHEN Gender='M' THEN Salary
END
```

In the output, we have satisfied our sort requirement in ascending or descending order

For Female employee, salary is appearing in descending order

For Male employee, salary is appearing in ascending order

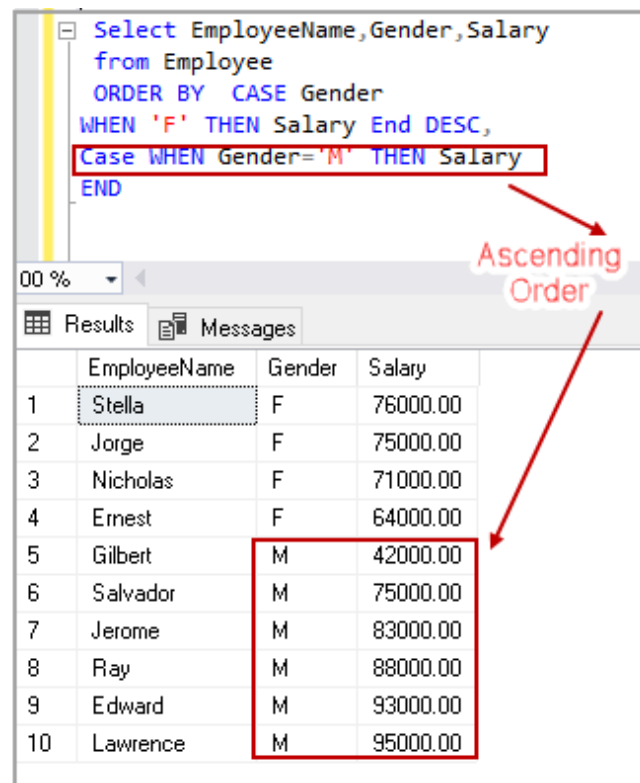


The screenshot shows a SQL query window with the following query:

```
Select EmployeeName, Gender, Salary
from Employee
ORDER BY CASE Gender
WHEN 'F' THEN Salary End DESC,
Case WHEN Gender='M' THEN Salary
END
```

The query results are displayed in a table with columns: EmployeeName, Gender, and Salary. The results are sorted according to the query. A red box highlights the first four rows (Stella, Jorge, Nicholas, Ernest) which are all female (F) and sorted by salary in descending order (76000.00, 75000.00, 71000.00, 64000.00). A red arrow points from the text "Descending Order" to this box. Another red box highlights the remaining six rows (Gilbert, Salvador, Jerome, Ray, Edward, Lawrence) which are all male (M) and sorted by salary in ascending order (42000.00, 75000.00, 83000.00, 88000.00, 93000.00, 95000.00).

	EmployeeName	Gender	Salary
1	Stella	F	76000.00
2	Jorge	F	75000.00
3	Nicholas	F	71000.00
4	Ernest	F	64000.00
5	Gilbert	M	42000.00
6	Salvador	M	75000.00
7	Jerome	M	83000.00
8	Ray	M	88000.00
9	Edward	M	93000.00
10	Lawrence	M	95000.00



The screenshot shows a SQL query window with the following query:

```
Select EmployeeName, Gender, Salary
from Employee
ORDER BY CASE Gender
WHEN 'F' THEN Salary End DESC,
Case WHEN Gender='M' THEN Salary
END
```

The query results are displayed in a table with columns: EmployeeName, Gender, and Salary. The results are sorted according to the query. A red box highlights the first four rows (Stella, Jorge, Nicholas, Ernest) which are all female (F) and sorted by salary in descending order (76000.00, 75000.00, 71000.00, 64000.00). A red arrow points from the text "Ascending Order" to this box. Another red box highlights the remaining six rows (Gilbert, Salvador, Jerome, Ray, Edward, Lawrence) which are all male (M) and sorted by salary in ascending order (42000.00, 75000.00, 83000.00, 88000.00, 93000.00, 95000.00).

	EmployeeName	Gender	Salary
1	Stella	F	76000.00
2	Jorge	F	75000.00
3	Nicholas	F	71000.00
4	Ernest	F	64000.00
5	Gilbert	M	42000.00
6	Salvador	M	75000.00
7	Jerome	M	83000.00
8	Ray	M	88000.00
9	Edward	M	93000.00
10	Lawrence	M	95000.00

Case Statement in SQL with Group by clause

We can use a Case statement with Group By clause as well. Suppose we want to group employees based on their salary. We further want to calculate the minimum and maximum salary for a particular range of employees. In the following query, you can see that we have Group By clause and it contains i with the condition to get the required output.

```

Select
CASE
  WHEN Salary >=80000 AND Salary <=100000 THEN 'Director'
  WHEN Salary >=50000 AND Salary <80000 THEN 'Senior Consultant'
  Else 'Director'
END AS Designation,
Min(salary) as MinimumSalary,
Max(Salary) as MaximumSalary
from Employee
Group By
CASE
  WHEN Salary >=80000 AND Salary <=100000 THEN 'Director'
  WHEN Salary >=50000 AND Salary <80000 THEN 'Senior Consultant'
  Else 'Director'
END

```

We have following output of this query. In this output, we get minimum and maximum salary for a particular designation.

	Designation	MinimumSalary	MaximumSalary
1	Director	42000.00	95000.00
2	Senior Consultant	64000.00	76000.00

Update statement with a CASE statement

We can use a Case statement in SQL with update DML as well. Suppose we want to update **Statecode** of employees based on Case statement conditions.

In the following code, we are updating statecode with the following condition.

- If employee statecode is **AR**, then update to **FL**
- If employee statecode is **GE**, then update to **AL**
- For all other statecodes update value to **IN**

Execute the following update command to fulfil our requirement using a Case statement.


```

UPDATE employee
SET StateCode =
CASE StateCode
  WHEN 'Ar' THEN 'FL'
  WHEN 'GE' THEN 'AL'
  ELSE 'IN'
END

```

In the following output, you can see old Statcode (left-hand side) and updated Statecode for the employees based on our conditions in the Case statement.

	EmployeeID	EmployeeName	Gender	StateCode	Salary
1	201	Jerome	M	FL	83000.00
2	202	Ray	M	AL	88000.00
3	203	Stella	F	AL	76000.00
4	204	Gilbert	M	Ar	42000.00
5	205	Edward	M	FL	93000.00
6	206	Ernest	F	Al	64000.00
7	207	Jorge	F	IN	75000.00
8	208	Nicholas	F	Ge	71000.00
9	209	Lawrence	M	IN	95000.00
10	210	Salvador	M	Co	75000.00



	EmployeeID	EmployeeName	Gender	StateCode	Salary
1	201	Jerome	M	IN	83000.00
2	202	Ray	M	IN	88000.00
3	203	Stella	F	IN	76000.00
4	204	Gilbert	M	FL	42000.00
5	205	Edward	M	IN	93000.00
6	206	Ernest	F	IN	64000.00
7	207	Jorge	F	IN	75000.00
8	208	Nicholas	F	AL	71000.00
9	209	Lawrence	M	IN	95000.00
10	210	Salvador	M	IN	75000.00

The following example shows that employees are receiving different raises in different departments on the COMPANY schema.

```

UPDATE EMPLOYEE
SET    Salary =
CASE WHEN Dno = 5 THEN Salary + 2000
      WHEN Dno = 4 THEN Salary + 1500
      WHEN Dno = 1 THEN Salary + 3000
      ELSE Salary + 0
END;

```

Case Statement limitations

- We can have multiple conditions in a Case statement; however, it works in a sequential model. If one condition is satisfied, it stops checking further conditions
- We cannot use a Case statement for checking NULL values in a table

Conclusion

The Case statement in SQL provides flexibility in writing t-SQL for DDL and DML queries. It also adds versatility to SQL Server queries. You should practice the Case statement in your queries.

<https://www.sqlshack.com/case-statement-in-sql/>

Tutorial Question of CASE

Compute the pension benefit for each employee based on his/her salary in the Employee table and these conditions:

PensionLevel	Salary	Pension
PL1	=<60,000	50% of salary
PL2	>60,000 and =<86,000	50% of salary up to 60,000 plus 25% of salary portion above 60,000
PL3	>86,000	75% of salary up to 60,000 plus 50% of salary portion > 60,000 and <= 86,000 plus 25% of salary portion above 86,000

- So, for someone making 32,000, their pension would be $.50 * 32,000 = 16,000$
- For someone making 68,000, their benefit would be
 $.50 * 60,000 + .25 * (68,000 - 60,000) = 30,000 + 2,000 = 32,000$
- For someone making 90,000 their benefit would be:
 $.75 * 60,000 + .50 * (60,000 - 86,000) + .25 * (90,000 - 86,000) =$
 $45,000 + 13,000 + 4,000 = 62,000$

Write a SQL statement to display

SSN, Lname, Salary,

Pension Level 1 amount(display as PL1),

Pension Level 2 amount (display as PL2),

Pension Level 3 amount (display as PL3), and Total Pension.

Order by salary followed by Lname.