

SQL Server COALESCE

The COALESCE() function in SQL Server evaluates the arguments in **sequence and gives the first NON-NULL value** in a specified number of expressions. If it evaluates all the list values as null or as not found any non-null value, it returns NULL.

Syntax

The following are the syntax that illustrates the COALESCE() function:

1. **COALESCE**(value1, value2, value3....., valueN);

Parameter Explanation

This function accepts only one parameter, which is the list that has various values.

value1, value2,.....,valueN: It specifies the values of the list to return the **NON-NULL or NULL** value in the output. It can be of any type but should be the same for all expressions.

We can understand it more clearly through the following illustration:

CASE1: COALESCE(NULL, NULL);

CASE2: COALESCE(0, NULL);

Here we can see that the function can accept a number of arguments and return the first non-null value. In a case when all the values of the list are null, this function returns NULL. Therefore, CASE1 and CASE2 always return NULL because they cannot find any non-null value.

SQL Server Coalesce Example

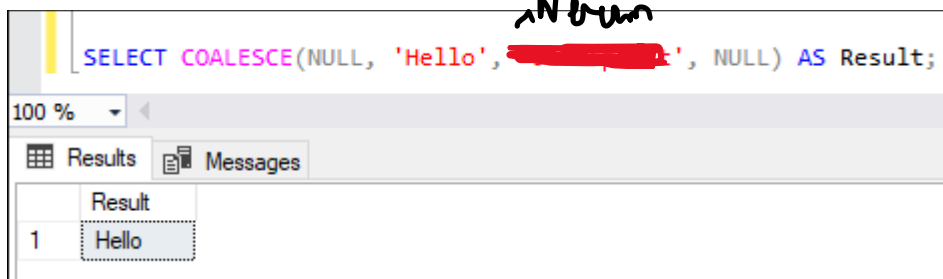
Let us understand the COALESCE() function with multiple examples. It is noted that we can use the COALESCE() function with the SELECT statement directly.

1. COALESCE function with expression as character string data

This example uses the COALESCE function with multiple values and returns the string **'Hello'** because it is the first non-null expression:

1. **SELECT COALESCE**(NULL, 'Hello', 'iNeuron', NULL) **AS** Result;

After execution, we will see the below output:



The screenshot shows a SQL query editor with the following text: `SELECT COALESCE(NULL, 'Hello', NULL, NULL) AS Result;`. The word "iNeuron" is handwritten above the query. The query is executed, and the results are displayed in a table with one row containing the value "Hello".

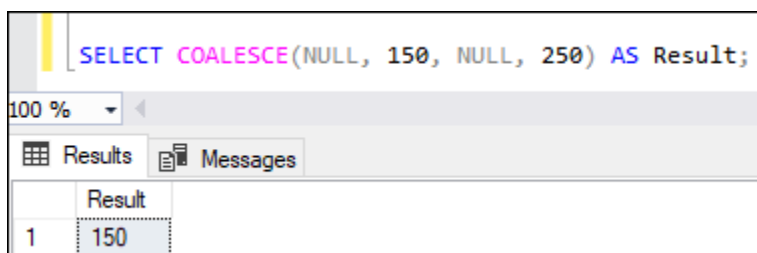
	Result
1	Hello

2. COALESCE function with expression as a numeric value

This example uses the COALESCE function with multiple values to evaluate a list of arguments and return the **first number**, which is non-null:

1. `SELECT COALESCE(NULL, 150, NULL, 250) AS Result;`

After execution, we will see the below output:



The screenshot shows a SQL query editor with the following text: `SELECT COALESCE(NULL, 150, NULL, 250) AS Result;`. The query is executed, and the results are displayed in a table with one row containing the value "150".

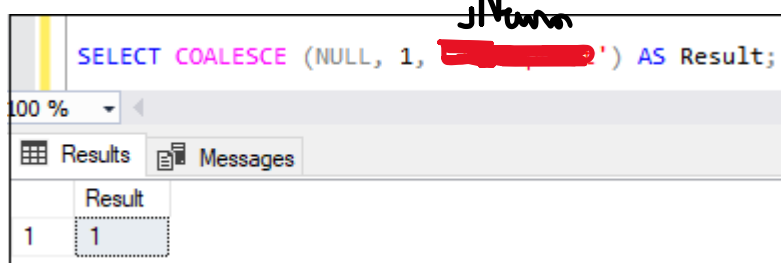
	Result
1	150

3. COALESCE function always evaluates an integer first

This example uses the COALESCE function with value as an integer followed by character expression and returns an integer as an output:

1. `SELECT COALESCE (NULL, 1, 'iNeuron') AS Result;`

After execution, we will see the below output:



The screenshot shows a SQL query editor with the following text: `SELECT COALESCE (NULL, 1, 'iNeuron') AS Result;`. The word "iNeuron" is handwritten above the query. The query is executed, and the results are displayed in a table with one row containing the value "1".

	Result
1	1

If we change the order of the integer and string (first string and then integer), **SQL Server** will **through an error**. See the below output:

```

SELECT COALESCE (NULL, 1, 1) AS Result;

```

100 %

Results Messages

Msg 245, Level 16, State 1, Line 9
Conversion failed when converting the varchar value '~~1~~' to data type int.

4. COALESCE function with tables

In this example, we will see how the coalesce() function works with the table. First, we will create a table named '**emp_contacts**' using the below statements:

```

CREATE TABLE emp_contact (
    id int,
    firstname VARCHAR(50) NOT NULL,
    lastname VARCHAR(50) NOT NULL,
    relationship VARCHAR(60),
    homenumber VARCHAR(25),
    worknumber VARCHAR(25),
    personalnumber VARCHAR(25)
);

```

Next, we will add data into this table as below:

```

INSERT INTO emp_contact (id, firstname, lastname, relationship, homenumber,
worknumber, personalnumber)
VALUES (1, 'Luisa', 'Huges', 'Wife', NULL, '920,176,1456', '928,132,2967'),
(2, 'Paul', 'Ward', 'spouse', NULL, NULL, '982,132,2867'),
(3, 'Rose', 'Huges', 'Daughter', NULL, NULL, NULL)

```

We can verify the table using the SELECT statement as below:

```

SELECT * FROM emp_contact;

```

100 %

Results Messages

	id	firstname	lastname	relationship	homenumber	worknumber	personalnumber
1	1	Luisa	Huges	Wife	NULL	920,176,1456	928,132,2967
2	2	Paul	Ward	spouse	NULL	NULL	982,132,2867
3	3	Rose	Huges	Daughter	NULL	NULL	NULL

Now, we will use the Coalesce function to select the columns **homenumber**, **worknumber**, and **personalnumber**. If the columns have NULL values, it will return the value 'NA' (not applicable). See the below code:

1. **SELECT** firstname+ ' ' +lastname **AS** fullname, relationship,
2. **COALESCE**(homenumber, worknumber, personalnumber, 'NA') **AS** phone
3. **FROM** emp_contact

After execution, we will see the below output:

	fullname	relationship	phone
1	Luisa Huges	Wife	920,176,1456
2	Paul Ward	spouse	982,132,2867
3	Rose Huges	Daughter	NA

COALESCE and CASE Expression

The COALESCE function is used as a syntactic shortcut for the CASE expression. The query optimizer will write COALESCE(expressions) in the CASE expression as below:

1. **CASE**
2. **WHEN** (exp1 **IS** NOT NULL) **THEN** exp1
3. **WHEN** (exp2 **IS** NOT NULL) **THEN** exp2
4. ?
5. ?
6. **ELSE** expN
7. **END**

Let us take a previous example of Coalesce function that selects the columns homenumber, worknumber, and personalnumber from the **emp_contact** table. If the columns have NULL values, it will return the value 'NA' (not applicable). We can write this coalesce function in CASE expression as follows:

1. **SELECT** firstname+ ' ' +lastname **AS** fullname, relationship,
2. **CASE**
3. **WHEN** homenumber **is** NOT NULL **THEN** homenumber
4. **WHEN** worknumber **is** NOT NULL **THEN** worknumber
5. **WHEN** personalnumber **is** NOT NULL **THEN** personalnumber
6. **ELSE** 'NA'
7. **END**

8. phone
9. **FROM** emp_contact

After execution, we will get the same output as the one that uses the coalesce function:

Results		Messages	
	fullname	relationship	phone
1	Luisa Huges	Wife	920,176,1456
2	Paul Ward	spouse	982,132,2867
3	Rose Huges	Daughter	NA

COALESCE vs. ISNULL

We will often be confused to distinguish between the ISNULL function and COALESCE expression because both have a similar purpose but can behave differently. Some of the key differences discussed below:

- The COALESCE() function evaluates the arguments in sequence and returns the first non-null value from a given list. In contrast, ISNULL() function tests whether the input expression is NULL or not. If the expression is NULL, the value passed to the argument list is returned; otherwise returns the expression.
- The COALESCE() function can evaluate the input values multiple times, whereas the ISNULL() function is evaluated only once.
- COALESCE() function works the same as the CASE expression rules and returns the value's data type with the highest precedence, while ISNULL returns the data type of the first parameter.
- The ISNULL and COALESCE function returns the **different nullability** of the result expression. Therefore, the expressions **ISNULL(NULL, 1)** and **COALESCE(NULL, 1)** are equal but give different nullability. The below example explains it more clearly:

```
CREATE TABLE #DemoTable1
(
  column1 INTEGER NULL,
  column2 AS COALESCE(col1, 0) PRIMARY KEY,
  column3 AS ISNULL(col1, 0)
);
```

```
CREATE TABLE #DemoTable2
(
    column1 INTEGER NULL,
    column2 AS COALESCE(col1, 0),
    column3 AS ISNULL(col1, 0) PRIMARY KEY
);
```

The **#DempTable1** cannot be created because the **PRIMARY KEY** cannot allow NULL values, and the nullability of the COALESCE expression for **column2** can return NULL.

The **#DempTable2** can be created because the ISNULL function's nullability evaluates to NOT NULL.

- The COALESCE() function can accept a variable number of parameters, whereas the ISNULL() function accepts only two parameters.