

SQL General Functions: NVL, NVL2, DECODE, COALESCE, NULLIF, LNNVL and NANVL

Introduction

Today, we are going to learn about SQL general Functions. The General Functions we are going to learn about are:

- NVL()
- NVL2()
- DECODE()
- COALESCE()
- LNNVL()

1.) NVL()

This is one of the functions of SQL extensively used in Structured Query Language (SQL).

This function can hold two input values only. If input values are more than 2 then an error is returned. This function returns the first NOT NULL value when searched in the function.

If both the inputs are NULL, then there is no output for this function.

The input data type can be Integer, Floating Point Number, String, Character input, etc.

Syntax

1. NVL (input 1, input 2)

Example Queries

```
SQL> select NVL(1, 2) from dual;
```

```
NVL(1,2)
```

```
SQL> select NVL(NULL, 1) from dual;
```

```
NVL(NULL,1)
```

```
-----
```

```
1
```

```
SQL> select NVL(1.029384, 1.029384) from dual;
```

```
NVL(1.029384,1.029384)
```

```
-----
```

```
1.029384
```

```
SQL> select NVL(NULL, 1.029384) from dual;
```

```
NVL(NULL,1.029384)
```

```
-----
```

```
1.029384
```

```
SQL> select NVL('JOE', 'ROOT') from dual;
```

```
NVL
```

```
---
```

```
JOE
```

```
SQL> select NVL(NULL, 'ROOT') from dual;
```

```
NVL(
```

```
----
```

```
ROOT
```

```
SQL> select COMMISSION_PERCENTAGE from sal;
```

```
COMMISSION_PERCENTAGE
```

```
-----
```

```
.15
```

```
.1
```

```
.01
```

.33

SQL> **select** NVL(COMMISION_PERCENTAGE, 0) **from** sal;

NVL(COMMISION_PERCENTAGE,0)

 .15
 .1
 .01
 0
 0
 .33
 0
 0

8 **rows** selected.

SQL> **SELECT** id, **name**, sal, NVL (COMMISION_PERCENTAGE, 0),
2 (sal) + (sal * NVL (COMMISION_PERCENTAGE, 0))
3 monthly_salary **FROM** sal;

	ID	NAME	SAL	NVL(COMMISION_PERCENTAGE,0)	MONTHLY_SALARY
1.	---	-----	-----	-----	-----
	1	Joe Root	75000	0.15	86250
	2	Ros Taylor	90000	0.1	99000
	3	Paul Adams	50000	0.01	50500
	4	Victor Lee	43000	0	43000
	5	Matt Potts	20000	0	20000
	6	James Anderson	200000	0.33	266000
	7	Craig Overton	11000	0	11000
	8	Rory Burns	9000	0	9000

8 **rows** selected.

SQL> **SELECT** id, **name**, sal, NVL (COMMISION_PERCENTAGE, 0),
2 (sal * 12) + (sal * NVL (COMMISION_PERCENTAGE, 0) *12)

```

3      yearly_salary FROM sal;
ID  NAME                SAL    NVL(COMMISION_PERCENTAGE,0)  YEARLY_
SALARY

```

1	Joe Root	75000	0.15	1035000
2	Ros Taylor	90000	0.1	1188000
3	Paul Adams	50000	0.01	606000
4	Victor Lee	43000	0	516000
5	Matt Potts	20000	0	240000
6	James Anderson	200000	0.33	3192000
7	Craig Overton	11000	0	132000
8	Rory Burns	9000	0	108000

8 rows selected.

2.) NVL2()

This is one of the functions of SQL extensively used in Structured Query Language (SQL).

This function can hold three input values only. If input values are more than three then an error is returned. This function returns the first value after NOT NULL value is found, when searched in the function.

If the first value is NOT NULL, second value is returned.

If the first value is NULL and the second value is NOT NULL, third value is returned.

The returned value can also be a NULL value too.

The working of this functioning is as same as NVL () in SQL.

If both the inputs are NULL, then there is no output for this function.

The input data type can be Integer, Floating Point Number, String, Character input, etc.

Syntax

1. NVL2(input 1, input 2, input 3)

Example Queries

1. SQL> **select** NVL2(1, 2, 3) **from** dual;

2.

3. NVL2(1,2,3)

4. -----

5. 2

6.

7. SQL> **select** NVL2(2, 2, 3) **from** dual;

8.

9. NVL2(2,2,3)

10. -----

11. 2

12.

13. SQL> **select** NVL2(2, 4, 3) **from** dual;

14.

15. NVL2(2,4,3)

16. -----

17. 4

18.

19. SQL> **select** NVL2(2, NULL, 3) **from** dual;

20.

21. NVL2(2,NULL,3)

22. -----

23.

24. SQL> **select** NVL2('Kevin', 'Pitersen', ' SA / ENG ') **from** dual;

25.

26. NVL2('KE

27. -----

28. Pitersen

29.

30. SQL> **select** NVL2('NULL', 'Pitersen', ' SA / ENG ') **from** dual;

31.

32. NVL2('NU

33. _ _ _ _ _

34. Pitersen

35.

36. SQL> **select** NVL2(NULL, 'Pitersen', ' SA / ENG ') **from** dual;

37.

38. NVL2(NULL,

39. _ _ _ _ _

40. SA / ENG

41.

42. SQL> **select** NVL2('Kevin', NULL, ' SA / ENG ') **from** dual;

43.

44. N

45. _

46.

47. SQL> **select** NVL2(56.2, 35.6, 23.4) **from** dual;

48.

49. NVL2(56.2,35.6,23.4)

50. _ _ _ _ _

51. 35.6

52.

53. SQL> **select** NVL2(NULL, 35.6, 23.4) **from** dual;

54.

55. NVL2(NULL,35.6,23.4)

56. _ _ _ _ _

57. 23.4

58.

59. SQL> **select** NVL2(NULL, NULL, 23.4) **from** dual;

60.

61. NVL2(NULL,NULL,23.4)

62. _ _ _ _ _

63. 23.4

64.

65. SQL> **select** NVL2(NULL, NULL, NULL) **from** dual;

66.

67. N

68. _

69.

70.

71. SQL> **select** NVL2(NULL, 23.4, NULL) **from** dual;

72.

73. NVL2(NULL,23.4,NULL)

74. _____

75.

76.

77. SQL> **select** NVL2(56.2, 23.4, NULL) **from** dual;

78.

79. NVL2(56.2,23.4,NULL)

80. _____

81. 23.4

82.

83. SQL > **select** * **from** sal;

84. ID **NAME** SAL COMMISSION_PERCENTAGE

85. _ _ _____ _ _ _____

86. 1 Joe Root 75000 0.15

87. 2 Ros Taylor 90000 0.1

88. 3 Paul Adams 50000 0.01

89. 4 Victor Lee 43000

90. 5 Matt Potts 20000

91. 6 James Anderson 200000 0.33

92. 7 Craig Overton 11000

93. 8 Rory Burns 9000

94. 8 **rows** selected.

95. SQL> **SELECT** id, **name**, sal, NVL2(COMMISSION_PERCENTAGE, sal, 0),

96. 2 (sal) + (sal * NVL2(COMMISSION_PERCENTAGE, NULL, 0.1))

97. 3 Wierd_salary **FROM** sal;

98.

99. ID **NAME** SAL NVL2(COMMISSION_PERCENTAGE,sal,0) WEI
 RD_SALARY

100. _ _ _____ _ _ _____

101. 1 Joe Root 75000 75000

102. 2 Ros Taylor 90000 90000

103. 3 Paul Adams 50000 50000

104. 4 Victor Lee 43000 0 47300

105.	5	Matt Potts	20000	0	22000
106.	6	James Anderson	200000	200000	
107.	7	Craig Overton	11000	0	12100
108.	8	Rory Burns	9000	0	9900

3.) DECODE()

This is also one of the expressions used in SQL. This Decode expression is used as IF, ELSE IF, ELSE IF Ladder style. This decode works on the basis of the condition specified.

Any kind of operation specified is going to work here.

The input types must be chosen based on the data types specified.

Syntax

1. DECODE (**column name**, number 1 **to** be searched, result 1 **to** be updated
2. , number 2 **to** be searched, result 2 **to** be updated
3. , number 3 **to** be searched, result 3 **to** be updated
4.
5. number n **to** be searched, result n **to** be updated , **default**)

Example Queries

1. SQL > **select * from** ipla;
- 2.
- 3.
4. SID SNAME SAL AGE
5. ---
6. 1 mahi 12 40
7. 2 kohli 14 33
8. 3 DK 6.25 33
9. 4 warner 6.75 33
10. 5 rahul 16 29
11. 6 pandya 14 27
- 12.
13. SQL > **SELECT** Sname, sid, sal,
14. 2 DECODE (sid, 1, 1.5*sal,
15. 3 2, 4*sal,


```

16. 4          3, 9*sal,
17. 5          4, 10.25*sal,
18. 6          sal)
19. 7          "REVISED SALARY"
20. 8 from ipla ;
21.
22. SNAME      SID      SAL      REVISED SALARY
23. ---      -
24. mahi        1        12        18
25. kohli       2        14        56
26. DK          3        6.25      56.25
27. warner      4        6.75      69.1875
28. rahul       5        16        16
29. pandya      6        14        14
30.
31. 6 rows selected.
32. SQL > select * from ipla;
33.
34.
35. SID SNAME      SAL      AGE
36. --- -
37. 1 mahi        12      40
38. 2 kohli       14      33
39. 3 DK          6.25    33
40. 4 warner      6.75    33
41. 5 rahul       16      29
42. 6 pandya      14      27
43. 7 Tim David   8.25    26
44. SQL > SELECT Sid, Sname, Sal, Age,
45. 2          DECODE (sid, 1, 2 * sid * sal,
46. 3          2, 3 * sid * sal,
47. 4          3, 5 * sid * sal,
48. 5          4, 10 * sid * sal,
49. 6          5, 12 * sid * sal,
50. 7          6, 15 * sid * sal,
51. 8          sal/2)
52. 9          "UPGRADED SALARY"

```

53. 10 **from** ipla ;

54.

55. SID SNAME SAL AGE UPGRADED SALARY

56. _ _ _ _ _

57. 1 mahi 12 40 24

58. 2 kohli 14 33 84

59. 3 DK 6.25 33 93.75

60. 4 warner 6.75 33 270

61. 5 rahul 16 29 960

62. 6 pandya 14 27 1260

63. 7 Tim David 8.25 26 4.125

64.

65. 7 **rows** selected.

4.) COALESCE()

This also one of the expression used in SQL. This expression works similar to NVL () expression. The only difference it can accept inputs greater than two. It returns the first NOT NULL input element.

The input data type can be anything. The inputs can be int, float, string, character, number, etc.

Syntax

1. **COALESCE** (input 1, input 2, input 3, , input n)

Example Queries

1. SQL> **select COALESCE**(NULL, 1) **from** dual;

2.

3. **COALESCE**(NULL,1)

4. _ _ _ _ _

5. 1

6. SQL> **select COALESCE**(1, 2, 2) **from** dual;

7.

8. **COALESCE**(1,2,2)

9. _ _ _ _ _

10. 1

11.

```

12. SQL> select COALESCE(NULL, 2, 2) from dual;
13.
14. COALESCE(NULL,2,2)
15. _-----
16.          2
17.
18. SQL> select COALESCE(NULL, NULL, 2) from dual;
19.
20. COALESCE(NULL,NULL,2)
21. _-----
22.          2
23.
24. SQL> select COALESCE(NULL, NULL, NULL) from dual;
25.
26. C
27. _
28.
29.
30. SQL> select COALESCE(NULL, NULL, NULL, 1, 2, 3, 4, 5, 6) from dual;
31.
32. COALESCE(NULL,NULL,NULL,1,2,3,4,5,6)
33. _-----
34.          1
35. SQL> select COALESCE(NULL, 'NULL', 'Stuart Broad', 'Adam Gilchrist') from dual;
36.
37.
38. COAL
39. _ _ _ _
40. NULL

```

5.) LNNVL()

This is one of the function of SQL which is used in SQL. This is used to convert True to False and False to True.

The LNNVL () function has the capacity to hold a condition. This makes the condition go reverse.

If the condition is $SID = 2$. Then $LNNVL (SID = 2)$ is equivalent to $SID \neq 2$.

Syntax

1. $LNNVL$ (Condition)

Example Queries

1. SQL> **select** * **from** ipla;

2.

3. SID SNAME SAL AGE

4. ---

5. 1 mahi 12 40

6. 2 kohli 14 33

7. 3 DK 6.25 33

8. 4 warner 6.75 33

9. 5 rahul 16 29

10. 6 pandya 14 27

11. 7 Tim David 8.25 26

12.

13. 7 **rows** selected.

14.

15. SQL> **select** * **from** ipla **where** sid=2;

16. SID SNAME SAL AGE

17. ---

18. 2 kohli 14 33

19.

20. SQL> **select** * **from** ipla **where** $LNNVL (sid = 2)$;

21.

22. SID SNAME SAL AGE

23. ---

24. 1 mahi 12 40

25. 3 DK 6.25 33

26. 4 warner 6.75 33

27. 5 rahul 16 29

28. 6 pandya 14 27

29. 7 Tim David 8.25 26

30. 6 **rows** selected.

31. SQL> **select** * **from** sal;

	ID	NAME	SAL	COMMISION_PERCENTAGE
34.	1	Joe Root	75000	0.15
35.	2	Ros Taylor	90000	0.1
36.	3	Paul Adams	50000	0.01
37.	4	Victor Lee	43000	
38.	5	Matt Potts	20000	
39.	6	James Anderson	200000	0.33
40.	7	Craig Overton	11000	
41.	8	Rory Burns	9000	

42. 8 **rows** selected

43.

44.

45. SQL> **select** * **from** sal **where** NVL (COMMISION_PERCENTAGE, 0) =0;

	ID	NAME	SAL	COMMISION_PERCENTAGE
48.	4	Victor Lee	43000	
49.	5	Matt Potts	20000	
50.	7	Craig Overton	11000	
51.	8	Rory Burns	9000	

52. 4 **rows** selected.

53. SQL> **select** * **from** sal **where** LNNVL (NVL (COMMISION_PERCENTAGE, 0) = 0) ;

54.

	ID	NAME	SAL	COMMISION_PERCENTAGE
57.	1	Joe Root	75000	0.15
58.	2	Ros Taylor	90000	0.1
59.	3	Paul Adams	50000	0.01
60.	6	James Anderson	200000	0.33

61. 4 **rows** selected

6.) NANVL ()

If the input value n2 is NaN (not a number), this method returns an alternative value n1, and if n2 is not NaN, it returns n2. Only floating-point numbers of the types BINARY FLOAT or BINARY DOUBLE can be used with this function.

The function accepts any numeric or nonnumeric data type as an input, with the ability to implicitly convert to a numeric data type.

The method returns BINARY DOUBLE if the parameter is BINARY FLOAT. If not, the function returns a numeric data type that matches the parameter.

Syntax

1. NANVL (input 1, input 2)

Example Queries

1. SQL> **SELECT** * **FROM** FPT;
- 2.
3. DEC_NUM BIN_DOUBLE BIN_FLOAT
4. -----
5. 3563.971 3.564E+003 3.564E+003
6. SQL> **INSERT INTO** FPT **VALUES** (0, 'NaN', 'NaN');
- 7.
8. 1 row created.
- 9.
10. SQL> **SELECT** * **FROM** FPT ;
- 11.
12. DEC_NUM BIN_DOUBLE BIN_FLOAT
13. -----
14. 3563.971 3.564E+003 3.564E+003
15. 0 Nan Nan
16. SQL> **SELECT** bin_float, NANVL (bin_float,0)
17. 2 **FROM** float_point_test;
- 18.
19. BIN_FLOAT NANVL(BIN_FLOAT,0)
20. -----
21. 1.514E+003 1.514E+003
22. Nan 0
- 23.
- 24.
25. MULTI QUERY APPROACH
26. SQL> **select** * **from** ipla;
- 27.

28.

29. SID SNAME SAL AGE

30. ---

31. 1 mahi 12 40

32. 2 kohli 14 33

33. 3 DK 6.25 33

34. 4 warner 6.75 33

35. 5 rahul 16 29

36. 6 pandya 14 27

37. 7 Tim David 8.25 26

38. SQL> select Sname, sid, sal,

39. 2 DECODE (SID, 1, (sal + sid * 5),

40. 3 2, (sal + sid*4),

41. 4 3, NVL(sal*4, NULL),

42. 5 4, NVL2(NULL, sal*3, sal*4),

43. 6 5, COALESCE(NULL, NULL, sal+4),

44. 7 6, 3 * sal,

45. 8 sal*2)

46. 9 "REVISED SALARY "

47. 10 from ipla;

48.

49.

50. SID SNAME SAL REVISED SALARY

51. ---

52. 1 mahi 12 17

53. 2 kohli 14 22

54. 3 DK 6.25 25

55. 4 warner 6.75 27

56. 5 rahul 16 20

57. 6 pandya 14 42

58. 7 Tim David 8.25 16.5

This is all about SQL general functions: NVL, NVL2, DECODE, COALESCE, NULLIF, LNNVL and NANVL