Relational DB & SQL - C11

Window Functions

To do: Go through the activity to the end

Ranking Window Functions

In this part, we'll learn ranking window functions. Ranking window functions return a ranking value for each row in a partition. Here are the window functions and their description used for ranking purposes.

Ranking Window Functions

CUME_DIST	Compute the cumulative distribution of a value in an ordered set of values.
DENSE_RANK	Compute the rank for a row in an ordered set of rows with no gaps in rank values.
NTILE	Divide a result set into a number of buckets as evenly as possible and assign a bucket number to each row.
PERCENT_RANK	Calculate the percent rank of each row in an ordered set of rows.
RANK	Assign a rank to each row within the partition of the result set.
ROW_NUMBER	Assign a sequential integer starting from one to each row within the current partition.

We'll not cover all of them in our course. However, you can easily try them on your own.
Let me remind you of the general window function syntax.

```
1  window function (column_name)
2  OVER ( [ PARTITION BY expr_list ] [ ORDER BY orders_list frame-clause ] )
3
```

Let's rank the employees based on their hire date.

"departments" table:

```
id     name     dept_name          seniority     graduation  salary
    hire_date
------ -------- ----------------   --------      ----------  ------
    ----------
10238  Eric     Economics          Experienced   MSc         72000    01-12
    -2019
13378  Karl     Music              Candidate     BSc         42000    01-01
    -2022
23493  Jason    Philosophy         Candidate     MSc         45000    01-01
    -2022
36299  Jane     Computer Science   Senior        PhD         91000    15-05
    -2018
30766  Jack     Economics          Experienced   BSc         68000    06-04
    -2020
40284  Mary     Psychology         Experienced   MSc         78000    22-10
    -2019
43087  Brian    Physics            Senior        PhD         93000    18-08
    -2017
53695  Richard  Philosophy         Candidate     PhD         54000    17-12
    -2021
58248  Joseph   Political Science  Experienced   BSc         58000    25-09
    -2021
63172  David    Art History        Experienced   BSc         65000    11-03
    -2021
64378  Elvis    Physics            Senior        MSc         87000    23-11
    -2018
96945  John     Computer Science   Experienced   MSc         80000    20-04
    -2019
99231  Santosh  Computer Science   Experienced   BSc         74000    07-05
    -2020
```

query:

```
SELECT name,
       RANK() OVER(ORDER BY hire_date DESC) AS rank_duration
FROM departments;
```

result:

```
name      rank_duration
--------  -------------
Karl      1
Jason     1
Richard   3
Joseph    4
David     5
Jack      6
Santos    7
Eric      8
Mary      9
John      10
Elvis     11
Jane      12
Brian     13
```

`RANK()` function assigns the same rank number if the hire_date value is same.

☝ Note: `RANK()` function assigns the row numbers of the values in the list created by the ordering rule. For the same values assigns their smallest row number.

Now, let's apply the same scenario by using the `DENSE_RANK` function.

query:

```
SELECT name,
       DENSE_RANK() OVER(ORDER BY hire_date DESC) AS rank_duration
FROM departments;
```

result:

```
name  rank_duration
------------  -------------
Karl          1
Jason         1
Richard       2
Joseph        3
David         4
Jack          5
Santos        6
Eric          7
Mary          8
John          9
Elvis         10
Jane          11
Brian         12
```

☝ Note: `DENSE_RANK()` returns the sequence numbers of the values in the list created by the ordering rule. For the same values assigns their smallest sequential integer.

Let's continue with another ranking function, `ROW_NUMBER()`.

`ROW_NUMBER()` assigns a sequential integer to each row. The row number starts with 1 for the first row.

If used with `PARTITION BY`, `ROW_NUMBER()` assigns a sequential integer to each row within the partition. The row number starts with 1 for the first row in each partition.

Let's give a sequence number to the employees in each seniority category according to their hire dates.

query:

```
SELECT name, seniority, hire_date,
       ROW_NUMBER() OVER(PARTITION BY seniority ORDER BY hire_date DESC) AS
           row_number
FROM departments
```

result:

```
name     seniority    hire_date    row_number
-------- -----------  ----------   ----------
Karl     Candidate    2022-01-01   1
Jason    Candidate    2022-01-01   2
Richard  Candidate    2021-12-17   3
Joseph   Experienced  2021-09-25   1
David    Experienced  2021-03-11   2
Jack     Experienced  2020-06-04   3
Santosh  Experienced  2020-05-07   4
Eric     Experienced  2019-12-01   5
Mary     Experienced  2019-10-22   6
John     Experienced  2019-04-20   7
Elvis    Senior       2018-11-23   1
Jane     Senior       2018-05-15   2
Brian    Senior       2017-08-18   3

```

☝ Note: We must use `ORDER BY` with ranking window functions.

Alright, you've got the logic of how a ranking window function works. You can easily apply other ranking window functions on your own. Let's continue with the last category of window function: Value Window Functions.

Previous

Next

You have completed 38% of the lesson

38%

Jump to...

Reset user tour on this page