Relational DB & SQL - C11

Window Functions

To do: Go through the activity to the end

Value Window Functions

In this part, we'll learn value window functions as the last category of window functions. They allow you to include values from other rows. Value Window Functions access a previous row without having to do a self-join. Some also call these functions 'offset functions'. The following table illustrates value window functions and their descriptions.

Function	Description
FIRST_VALUE	Get the value of the first row in a specified window frame.
LAG	Provide access to a row at a given physical offset that comes before the current row.
LAST_VALUE	Get the value of the last row in a specified window frame.
LEAD	Provide access to a row at a given physical offset that follows the current row.

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.

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

```

Let's start with `LAG()` and `LEAD()` functions. These functions are useful to compare rows to preceding or following rows. `LAG` returns data from previous rows and `LEAD` returns data from the following rows.

The following displays syntax of the `LAG` and `LEAD` function in particular.

```
LAG(column_name [,offset] [,default])
```

offset: Optional. It specifies the number of rows back from the current row from which to obtain a value. If not given, the default is 1. In that case, it returns the value of the previous value. If there is no previous row (the current row is the first), then returns `NULL`. Offset value must be a non-negative integer.

default: The value to return when the offset is beyond the scope of the partition. If a default value is not specified, `NULL` is returned.

Let's do an example.

query:

```
SELECT id, name,
        LAG(name) OVER(ORDER BY id) AS previous_name
FROM departments;
```

result:

```
id      name     previous_name
-----   -------  -------------
10238   Eric     NULL
13378   Karl     Eric
23493   Jason    Karl
30766   Jack     Jason
36299   Jane     Jack
40284   Mary     Jane
43087   Brian    Mary
53695   Richard  Brian
58248   Joseph   Richard
63172   David    Joseph
64378   Elvis    David
96945   John     Elvis
99231   Santosh  John

```

Let's do the same example by using `LEAD()` function.

```
LEAD(column_name [,offset] [,default])
```

query:

```
SELECT id, name,
        LEAD(name) OVER(ORDER BY id) AS next_name
FROM departments;
```

result:

```
id      name     next_name
-----   -------  ---------
10238   Eric     Karl
13378   Karl     Jason
23493   Jason    Jack
30766   Jack     Jane
36299   Jane     Mary
40284   Mary     Brian
43087   Brian    Richard
53695   Richard  Joseph
58248   Joseph   David
63172   David    Elvis
64378   Elvis    John
96945   John     Santosh
99231   Santosh  NULL

```

If you want to access two rows back from the current row, you need to specify the offset argument 2. The following query displays the values two rows back from the current row.

query:

```sql
SELECT id, name,
       LAG(name, 2) OVER(ORDER BY id) AS previous_name
FROM departments;
```

result:

```
id      name     previous_name
-----   -------  -------------
10238   Eric     NULL
13378   Karl     NULL
23493   Jason    Eric
30766   Jack     Karl
36299   Jane     Jason
40284   Mary     Jack
43087   Brian    Jane
53695   Richard  Mary
58248   Joseph   Brian
63172   David    Richard
64378   Elvis    Joseph
96945   John     David
99231   Santosh  Elvis

```

Let's do the examples with `FIRST_VALUE()` AND `LAST_VALUE()`

query:

```sql
SELECT id, name,
       FIRST_VALUE(name) OVER(ORDER BY id) AS first_name
FROM departments;
```

result:

```
id      name      the_first_name
-----   -------   --------------
10238   Eric      Eric
13378   Karl      Eric
23493   Jason     Eric
30766   Jack      Eric
36299   Jane      Eric
40284   Mary      Eric
43087   Brian     Eric
53695   Richard   Eric
58248   Joseph    Eric
63172   David     Eric
64378   Elvis     Eric
96945   John      Eric
99231   Santosh   Eric

```

As you see, for each row, `FIRST_VALUE()` function returns the first value from the whole name column sorted by id.

✍️ Because the default window frame covered all of the rows for each row.

query:

```sql
SELECT id, name,
       LAST_VALUE(name) OVER(ORDER BY id ROWS BETWEEN UNBOUNDED PRECEDING
              AND UNBOUNDED FOLLOWING) AS last_name
FROM departments;
```

result:

```
id      name      the_last_name
-----   --------  -------------
10238   Eric      Santosh
13378   Karl      Santosh
23493   Jason     Santosh
30766   Jack      Santosh
36299   Jane      Santosh
40284   Mary      Santosh
43087   Brian     Santosh
53695   Richard   Santosh
58248   Joseph    Santosh
63172   David     Santosh
64378   Elvis     Santosh
96945   John      Santosh
99231   Santosh   Santosh

```

In the example above, for each row, `LAST_VALUE()` function returns the last value from the whole name column sorted by id.

✍ We change the window frame. Because the default window frame didn't cover all of the rows for each row.

Previous Next

You have completed 38% of the lesson

38%

Jump to...