9.21. Window Functions

Window functions provide the ability to perform calculations across sets of rows that are related to the current query row. See Section 3.5 for an introduction to this feature, and Section 4.2.8 for syntax details.

The built-in window functions are listed in Table 9-53. Note that these functions must be invoked using window function syntax; that is an OVER clause is required.

In addition to these functions, any built-in or user-defined normal aggregate function (but not ordered-set or hypothetical-set aggregates) can be used as a window function; see Section 9.20 for a list of the built-in aggregates. Aggregate functions act as window functions only when an OVER clause follows the call; otherwise they act as regular aggregates.

Table 9-53. General-Purpose Window Functions

Function	Return Type	Description
`row_number()`	`bigint`	number of the current row within its partition, counting from 1
`rank()`	`bigint`	rank of the current row with gaps; same as `row_number` of its first peer
`dense_rank()`	`bigint`	rank of the current row without gaps; this function counts peer groups
`percent_rank()`	`double precision`	relative rank of the current row: (`rank` - 1) / (total rows - 1)
`cume_dist()`	`double precision`	relative rank of the current row: (number of rows preceding or peer with current row) / (total rows)
`ntile(num_buckets integer)`	`integer`	integer ranging from 1 to the argument value, dividing the partition as equally as possible
`lag(value anyelement [, offset integer [, default anyelement ]])`	`same type as value`	returns `value` evaluated at the row that is `offset` rows before the current row within the partition; if there is no such row, instead return `default` (which must be of the same type as `value`). Both `offset` and `default` are evaluated with respect to the current row. If omitted, `offset` defaults to 1 and `default` to null
`lead(value anyelement [, offset integer [,`	`same type`	returns `value` evaluated at the row that is `offset` rows after the current row within the partition; if there is no such row, instead return `default` (which must be of the same type as `value`). Both

`default anyelement ]])`	`as value`	`offset` and `default` are evaluated with respect to the current row. If omitted, `offset` defaults to 1 and `default` to null
`first_value(value any)`	`same type as value`	returns `value` evaluated at the row that is the first row of the window frame
`last_value(value any)`	`same type as value`	returns `value` evaluated at the row that is the last row of the window frame
`nth_value(value any, nth integer)`	`same type as value`	returns `value` evaluated at the row that is the `nth` row of the window frame (counting from 1); null if no such row

All of the functions listed in Table 9-53 depend on the sort ordering specified by the `ORDER BY` clause of the associated window definition. Rows that are not distinct in the `ORDER BY` ordering are said to be *peers*; the four ranking functions are defined so that they give the same answer for any two peer rows.

Note that `first_value`, `last_value`, and `nth_value` consider only the rows within the "window frame", which by default contains the rows from the start of the partition through the last peer of the current row. This is likely to give unhelpful results for `last_value` and sometimes also `nth_value`. You can redefine the frame by adding a suitable frame specification (`RANGE` or `ROWS`) to the `OVER` clause. See Section 4.2.8 for more information about frame specifications.

When an aggregate function is used as a window function, it aggregates over the rows within the current row's window frame. An aggregate used with `ORDER BY` and the default window frame definition produces a "running sum" type of behavior, which may or may not be what's wanted. To obtain aggregation over the whole partition, omit `ORDER BY` or use `ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING`. Other frame specifications can be used to obtain other effects.

> **Note:** The SQL standard defines a `RESPECT NULLS` or `IGNORE NULLS` option for `lead`, `lag`, `first_value`, `last_value`, and `nth_value`. This is not implemented in PostgreSQL: the behavior is always the same as the standard's default, namely `RESPECT NULLS`. Likewise, the standard's `FROM FIRST` or `FROM LAST` option for `nth_value` is not implemented: only the default `FROM FIRST` behavior is supported. (You can achieve the result of `FROM LAST` by reversing the `ORDER BY` ordering.)

Prev Aggregate Functions | Home | Up | Next Subquery Expressions