# SQL window functions

From PostgreSQL documentation:

“ A *window function* performs a calculation across a set of table rows that are somehow related to the current row. This is comparable to the type of calculation that can be done with an aggregate function. But unlike regular aggregate functions, use of a window function does not cause rows to become grouped into a single output row — the rows retain their separate identities. Behind the scenes, the window function is able to access more than just the current row of the query result.”

Let’s run following SQL SELECT in Practical class database::

SELECT \*, SUM(Rating) OVER (ORDER BY EntryDate) FROM TravelMethod

This SELECT retrieves all data from table TravelMethod as well as field where all ratings are summed from beginning of table until current row (running sum).

We could run following select to get all travel data for each travel as well as average travel rating for each travel method type:

SELECT \*, AVG(Rating\*1.0) OVER (PARTITION BY TravelMethodType\_ID) AverageRating FROM TravelMethod

These examples show how OVER allows to select some specific rows together with some aggregated data from the same table.

Both possibilities (ORDER BY and PARTITION BY) can be used in the same select. For example next select gets travel ratings tgether with running sum ordered by date:

SELECT \*, SUM(Rating) OVER (PARTITION BY TravelMethodType\_ID ORDER BY EntryDate) FROM TravelMethod

In previous select data is ordered by TravelMethodType\_ID and for each TravelMethodType\_ID rows are ordered by dates. Running sum of ratings are calculated in each group of the same TravelMethodType\_ID.

ORDER and PARTITON defines window for which part and in which order running agregates are calculated.

Compare these three selects:

SELECT \*, SUM(Rating) OVER (PARTITION BY TravelMethodType\_ID) FROM TravelMethod

SELECT \*, SUM(Rating) OVER (PARTITION BY TravelMethodType\_ID ORDER BY EntryDate) FROM TravelMethod

SELECT \*, SUM(Rating) OVER (ORDER BY EntryDate) FROM TravelMethod

First select gets all ratings with rating sum for specific TravelMethodType\_ID. Running sum of ratings for specific TravelMethodType\_ID is selected in second select. Running sum of ratings for all rows is selected in third select.   
  
SUM, AVG and COUNT can be used as aggregate functions.

## Function ROW\_NUMBER()

Function ROW\_NUMBER() selects row’s number if rows are ordered by OVER … ORDER BY().

SELECT \*, ROW\_NUMBER() OVER (ORDER BY EntryDate) FROM TravelMethod

SELECT \*, ROW\_NUMBER() OVER (PARTITION BY TravelMethodType\_ID ORDER BY EntryDate) FROM TravelMethod

Global row numbering is selected for all table in first case. Row number starts with 1 for each TravelMethodType\_ID in second select

## RANK() un DENSE\_RANK()

Next example has similar select, only rows are ordered by Rating in descending order.

SELECT \*, ROW\_NUMBER() OVER (ORDER BY Rating DESC) FROM TravelMethod

In this case ROW\_NUMBER will indicate how good was travelling if ranked by Rating.

Some rows has the same rating and we would like to give them the same rank. We can use function RANK() in this case.

SELECT \*, RANK() OVER (ORDER BY Rating DESC) FROM TravelMethod

All travels with the same rating receive the same RANK() in this case.

All travels with Rating=5 has RANK() =1, there are 10 such travels. All travels with Rating=4 has RANK()=11, because travels with Rating=5 are in the first ten places. Therefore there are no travels with Rank from 2 to 10. We must use DENSE\_RANK() if we need all travels with Rating=4 get RANG()=2.

SELECT \*, DENSE\_RANK() OVER (ORDER BY Rating DESC) FROM TravelMethod

PARTITION BY must be added to calculate RANK by groups:

SELECT \*, RANK() OVER (PARTITION BY TravelMethodType\_ID ORDER BY Rating DESC)   
FROM TravelMethod

SELECT \*, DENSE\_RANK() OVER (PARTITION BY TravelMethodType\_ID ORDER BY Rating DESC)   
FROM TravelMethod

NTILE can be used to calculate in which part specific row is (for example in 2nd part of three).It can be used with function NTILE(n), where n is the number of parts.

Following example selects in which third each row is if ordered by rating:

SELECT \*, NTILE(3) OVER (ORDER BY Rating DESC) FROM TravelMethod

In which of ten parts is each row, if ordered by EntryDate:

SELECT \*, NTILE(10) OVER (ORDER BY EntryDate DESC) FROM TravelMethod

Select for different numbers of parts:

SELECT \*, NTILE(3) OVER (ORDER BY EntryDate DESC) as Trešdaļa  
, NTILE(10) OVER (ORDER BY EntryDate DESC) as Desmitdaļa  
, NTILE(100) OVER (ORDER BY EntryDate DESC) as Simtdaļa  
FROM TravelMethod

It is necessary to compare row data with data from some previous or next row. LAG gets data from some of previous row, Lead gets data from some of next row.

SELECT \*, EntryDate, LAG(EntryDate,1) OVER (PARTITION BY TravelMethodType\_ID ORDER BY EntryDate) FROM TravelMethod

SELECT \*, EntryDate, LAG(EntryDate,1) OVER (PARTITION BY TravelMethodType\_ID ORDER BY EntryDate) FROM TravelMethod

First/last row (or more then one row) of each part has no data from previous/next rows, because these rows has no previous/next row(s).

Functions LAG and LEAD are useful if we need to get some difference from two rows.

## Exercise

Select each row from Network Quality, add average quality of specific operator to each row. Order data by network operator and input date.

Select all rows of Network Quality. Add to each row number of rows for specific operator.

Select all rows of Network Quality, order by input date. Add row number for each row.

Select all rows of Network Quality, divide by operators, order rows for each operator by input date. Add row number for each row, starting numbering for each operator with 1.

Select all rows from Network Quality, divide by operators, order rows for each operator by Quality in descending order. Add for each row its number by quality rating. Rows with the same quality rating for the same operator must be with the same number.

Select all rows from Network Quality, divide by operators, order rows for each operator by Quality in descending order. Add for each row its number by quality rating. Rows with the same quality rating for the same operator must be with the same number. Row numbers must be without “holes”, e.g. 1,2,3 etc.

Select all rows from Network Quality, order by quality, add quintile each row is in.