**WINDOW FUNCTIONS**

**https://www.postgresql.org/docs/9.1/tutorial-window.html**

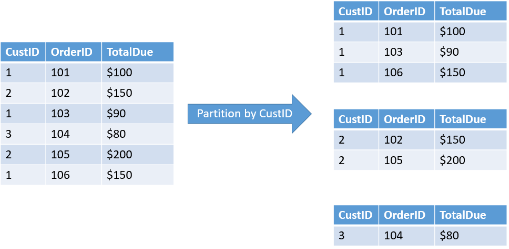
**https://blog.sqlauthority.com/2015/11/04/sql-server-what-is-the-over-clause-notes-from-the-field-101/**

**https://stackoverflow.com/questions/561836/oracle-partition-by-keyword**

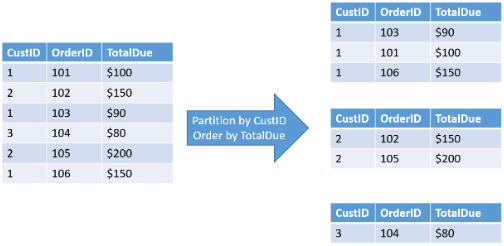
Cuando añadimos las ventanas T-SQL (window functions) debemos usar la clausula OVER siempre.

OVER tiene tres componentes: particiones, ordenar y enmarcar.

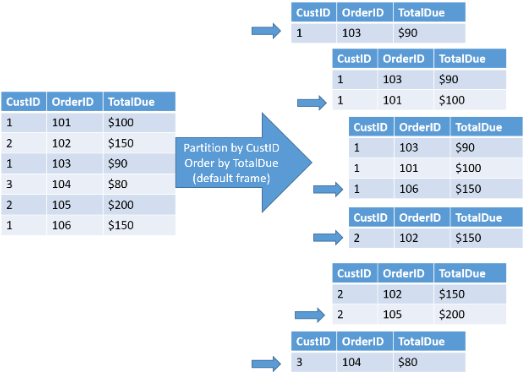
PARTITION BY es opcional y la soportan todas las window functions. La diferencia con respecto a GROUP BY es que PARTITION BY no agrupa todos los términos de una columna devolviendo una fila, si no que se obtienen varias filas:



Ordenar con la clausula OVER es permitido cuando los datos tienen que ser ordenados para ejecutar los cálculos. Por ejemplo ROW\_NUMBER requiere de la expresión ORDER BY en la clausula OVER porque las filas deben alinearse. De hecho, ORDER BY es soportado por la mayoría de las window functions. ORDER BY en la cláusula OVER no es admitido para calcular subtotales, por ejemplo. Tampoco necesitas los datos ordenados para calcular una suma. ORDER BY en la cláusula OVER no tiene nada que ver con la cláusula ORDER BY en sí misma.



Por último tenemos el “framing” (enmarque), que solo se usa en un par de situaciones: acumulación de agregados de las window functions (por ejemplo los “**running totals**”, “**moving average**”, etc) y las funciones **FIRST\_VALUE** y **LAST\_VALUE**.



Esta función es un poco especial, es como si la fila 1 solo “viera” la fila 1, la fila 2 a la 1 y 2, la fila 3 a la 1,2 y 3, y así sucesivamente.

El código de esos ejemplos:

SELECT CustomerID, SalesOrderID, TotalDue,

ROW\_NUMBER() OVER(PARTITION BY CustomerID ORDER BY SalesOrderID) AS RowNum,

SUM(TotalDue) OVER() AS GrandTotal,

SUM(TotalDue) OVER(PARTITION BY CustomerID) AS SubTotal,

SUM(TotalDue) OVER(PARTITION BY CustomerID ORDER BY SalesOrderID) AS RunningTotal,

LAG(SalesOrderID) OVER(PARTITION BY CustomerID ORDER BY SalesOrderID) AS PrevOrder,

FIRST\_VALUE(TotalDue) OVER(PARTITION BY CustomerID ORDER BY SalesOrderID) AS FirstAmt

FROM Sales.SalesOrderHeader;

Ver ejemplos prácticos con anotaciones del código en SQL\_6\_WindowFunctions.sql

**ALIAS para múltiples WINDOW FUNCTIONS**

Ver ejemplo en SQL\_6\_WindowFunctions.sql

**LAG & LEAD**

Sirven para coger datos de las filas previas y posteriores. LAG para las previas. LEAD para las posteriores.

Vienen bien si queremos comparar valores con las filas previas o siguientes o sacar valores relativos de una fila a otra. Ver ejemplos en SQL\_6\_WindowFunctions.sql

**NTILES (percentil)**

NTILE(x) OVER (ORDER BY value) AS xxx

Lo que hace es dividir en x zonas “value” y te dice en qué zona de esa división está el valor.