# DML. Las consultas de resumen
# DML. Summary Queries

Índice

# 1. Introducción

Una de las funcionalidades de la sentencia SELECT es el permitir obtener resúmenes *SUMMARIES* de los datos contenidos en las columnas de las tablas.

Para poder llevarlo a cabo la sentencia SELECT consta de una serie de cláusulas específicas (GROUP BY, HAVING), y Transact-SQL tiene definidas unas funciones para poder realizar estos cálculos, las funciones de agregado *AGGREGATE FUNCTIONS* (también llamadas funciones de columna).

La diferencia entre una consulta de resumen y una consulta de las que hemos visto hasta ahora es que en las consultas normales las filas del resultado se obtienen directamente de las filas del origen de datos y cada dato que aparece en el resultado tiene su dato correspondiente en el origen de la consulta mientras que las filas generadas por las consultas de resumen no representan datos del origen sino un total calculado sobre estos datos. Esta diferencia hará que las consultas de resumen tengan algunas limitaciones que veremos a lo largo del tema.

Un ejemplo sería:

```
SELECT Oficina, Region, Ventas
FROM Oficinas
ORDER BY Region
```

| Oficina | Region | Ventas |
|---|---|---|
| 24 | centro | 150.000 Pts. |
| 23 | centro | |
| 28 | este | 0 Pts. |
| 13 | este | 368.000 Pts. |
| 12 | este | 735.000 Pts. |
| 11 | este | 693.000 Pts. |
| 26 | norte | |
| 22 | oeste | 185.000 Pts. |
| 21 | oeste | 836.000 Pts. |

```
SELECT Region, SUM (Ventas)
FROM Oficinas
GROUP BY Region
```

| Region | Suma de Ventas |
|---|---|
| centro | 150.000 |
| este | 1.796.000 |
| norte | |
| oeste | 1.021.000 |

A la izquierda tenemos una consulta simple que nos saca las oficinas con sus ventas ordenadas por región, y a la derecha una consulta de resumen que obtiene la suma de las ventas de las oficinas de cada región

# 2. Las funciones de agregado

Una función de agregado SQL acepta un grupo de datos (normalmente una columna de datos) como argumento, y produce un único dato que resume *SUMMARIZES* el grupo. Por ejemplo la función AVG() acepta una columna de datos y devuelve la media aritmética (AVeraGe) de los valores contenidos en la columna.

El mero hecho de utilizar una función de agregado en una consulta, convierte ésta en una consulta de resumen. Esto es importante tenerlo en cuenta porque las consultas de resumen tienen algunas limitaciones que hay que tener siempre presentes.

Todas las funciones de agregado tienen una estructura muy parecida:

```
Función ([ALL|DISTINCT] expression)
```

El grupo de valores sobre el que actúa la función lo determina el resultado de la expresión que será un nombre de columna o una expresión basada en una columna o varias del origen de datos. En la expresión nunca puede aparecer una función de agregado ni una subconsulta.

La palabra ALL indica que se tiene que tomar en cuenta todos los valores de la columna (excepto los valores nulos ya que no son valores). ALL es el valor por defecto.

La palabra DISTINCT hace que se consideren todas las repeticiones del mismo valor como uno sólo (sólo considera valores distintos).

Todas las funciones de agregado se aplican a las filas del origen de datos una vez ejecutada la cláusula WHERE (si la hubiera).

Si exceptuamos la función COUNT, todas las funciones de agregado ignoran los valores NULL.

Una función de agregado puede aparecer en la lista de selección en cualquier lugar en el que puede aparecer un nombre de columna. Puede, por ejemplo, formar parte de una expresión pero no se pueden anidar funciones de agregado.

Tampoco se pueden mezclar funciones de columna con nombres de columna ordinarios. Hay excepciones a esta regla pero únicamente cuando definimos agrupaciones y subconsultas que veremos más adelante.

## 2.1. La función COUNT

```
COUNT ({[ALL|DISTINCT] expresion | * } )
```

*Expresion* puede ser de cualquier tipo excepto **text**, **image** o **ntext**. No se permite utilizar funciones de agregado ni subconsultas. El tipo de dato devuelto es **int**.

Si el número de valores devueltos por *expresion* es superior a $2^{31}$-1, COUNT genera un error, en ese caso hay que utilizar la función COUNT_BIG.

La función cuenta los valores distintos de NULL que hay en la columna. La palabra ALL indica que se tienen que tomar todos los valores de la columna, mientras que DISTINCT hace que se consideren todas las repeticiones del mismo valor como uno solo. Estos parámetros son opcionales, por defecto se considera ALL.

Por ejemplo:

```
SELECT COUNT(region) FROM oficinas;
```

Devuelve 9 porque tenemos nueve valores no nulos en la columna *region*. A la hora de interpretar un COUNT es conveniente no olvidar que cuenta valores no nulos, por ejemplo si interpretáramos la sentencia tal cual se lee, "cuántas regiones tenemos en oficinas" sería erróneo, realmente estamos obteniendo cuántas oficinas tienen una región asignada (cuántos valores no nulos hay en la columna región).

```
SELECT COUNT(DISTINCT region) FROM oficinas;
```

Devuelve 4 porque tenemos cuatro valores distintos, no nulos, en la columna región, los valores repetidos los considera sólo una vez. Ahora sí nos devuelve cuántas regiones tenemos en oficinas.

Si utilizamos * en vez de expresión, devuelve el número de filas del origen que nos queda después de ejecutar la cláusula WHERE.

COUNT(*****) no acepta parámetros y no se puede utilizar con DISTINCT. COUNT(*****) no requiere un parámetro *expression* porque, por definición, no utiliza información sobre ninguna columna específica. En el recuento se incluyen las filas que contienen valores NULL.

```
SELECT COUNT(*) FROM oficinas;
```

Nos dice cuántas oficinas tenemos.

```
SELECT COUNT(*) FROM empleados WHERE oficina=12;
```

Obtiene el número de empleados asignados a la oficina 12.

Si tenemos una columna que no puede contener valores nulos (porque así está definida, por ejemplo), con un COUNT(esacolumna) se obtiene el mismo resultado que COUNT(*), en este caso hay que utilizarlo en vez de COUNT(columna) porque el COUNT(*) es más rápido, normalmente es un valor que el sistema ya tiene calculado, si te fijas cuando ejecutas una consulta te dice x filas recuperadas (justo ese número).

Por ejemplo:

```
SELECT COUNT(*) FROM empleados WHERE ventas>10000;
```

Es mejor que:

```
SELECT COUNT(numemp) FROM empleados WHERE ventas>10000;
```

Porque numemp es clave primaria, por lo que no admite nulos.

Otro ejemplo:

```
SELECT COUNT(*) FROM empleados WHERE oficina IS NOT NULL;
```

Es mejor que:

```
SELECT COUNT(oficina) FROM empleados WHERE oficina IS NOT NULL;
```

Aunque el campo oficina admite nulos en la tabla. Las filas del origen después de aplicar el WHERE no tienen valores nulos en oficina, por lo que la primera es mejor porque se calcula más rápidamente.

## 2.2. La función COUNT_BIG

Funciona igual que la función COUNT. La única diferencia entre ambas funciones está en los valores devueltos, COUNT_BIG siempre devuelve un valor de tipo **bigint** y por lo tanto admite más valores de entrada, no está limitado a $2^{31}$-1 valores de entrada como COUNT.

## 2.3. La función MAX

```
MAX ([ALL|DISTINCT] expression)
```

Devuelve el valor máximo de la expresión sin considerar los nulos.

MAX se puede usar con columnas numéricas, de caracteres y de **datetime**, pero no con columnas de **bit**. No se permiten funciones de agregado ni subconsultas.

Utilizar DISTINCT no tiene ningún sentido con MAX (el valor máximo será el mismo si consideramos las repeticiones o no) y sólo se incluye para la compatibilidad con SQL-92.

Por ejemplo:

```
SELECT COUNT(ventas) AS VentasTotales, MAX(objetivo) AS MayorObjetivo
FROM oficinas;
```

Devuelve cuántas oficinas tienen ventas y de todas las oficinas el máximo objetivo.

## 2.4. La función MIN

```
MIN ([ALL|DISTINCT] expression)
```

Devuelve el valor mínimo de la expresión sin considerar los nulos.

MIN se puede usar con columnas numéricas, de caracteres y de **datetime**, pero no con columnas de **bit**. No se permiten funciones de agregado ni subconsultas.

Utilizar DISTINCT no tiene ningún sentido con MIN (el valor mínimo será el mismo si consideramos las repeticiones o no) y sólo se incluye para la compatibilidad con SQL-92.

## 2.5. La función SUM

```
SUM ([ALL|DISTINCT] expresion )
```

Devuelve la suma de los valores devueltos por la expresión.
Sólo puede utilizarse con columnas numéricas.
El resultado será del mismo tipo aunque puede tener una precisión mayor.

```
SELECT SUM(importe) FROM pedidos;
```

Obtiene el importe total vendido en todos los pedidos.

```
SELECT SUM(ventas) AS VentasTotales, MAX(objetivo) AS MayorObjetivo
FROM oficinas;
```

Devuelve la suma de las ventas de todas las oficinas y de los objetivos de todas las oficinas, el de mayor importe.

### 2.6. La función AVG

```
AVG ([ALL|DISTINCT] expresion )
```

Devuelve el promedio de los valores de un grupo, para calcular el promedio se omiten los valores nulos.

El grupo de valores lo determina el resultado de la expresión que será un nombre de columna o una expresión basada en una columna o varias del origen de datos.
La función se aplica también a campos numéricos, y en este caso el tipo de dato del resultado puede cambiar según las necesidades del sistema para representar el valor del resultado.

### 2.7. La función VAR

```
VAR ([ALL|DISTINCT] expresion )
```

Devuelve la varianza estadística de todos los valores de la expresión especificada.
VAR sólo se puede utilizar con columnas numéricas. Los valores NULL se pasan por alto.

### 2.8. La función VARP

```
VARP ([ALL|DISTINCT] expresion )
```

Devuelve la varianza estadística de la población para todos los valores de la expresión especificada.
Sólo se puede utilizar con columnas numéricas. Los valores NULL se pasan por alto.

### 2.9. La función STDEV

```
STDEV ([ALL|DISTINCT] expresion )
```

Devuelve la desviación típica estadística de todos los valores de la expresión especificada.
Sólo se puede utilizar con columnas numéricas. Los valores NULL se pasan por alto.

### 2.10. La función STDEVP

```
STDEVP ([ALL|DISTINCT] expresion )
```

Devuelve la desviación estadística estándar para la población de todos los valores de la expresión especificada.
Sólo se puede utilizar con columnas numéricas. Los valores NULL se pasan por alto.

### 2.11. La función GROUPING

```
GROUPING (nb_columna)
```

Es una función de agregado que genera como salida una columna adicional con el valor 1 si la fila se agrega mediante el operador CUBE o ROLLUP, o el valor 0 cuando la fila no es el resultado de CUBE o ROLLUP.

Nb_columna tiene que ser una de las columnas de agrupación y la cláusula GROUP BY debe contener el operador CUBE o ROLLUP.

En el siguiente punto, cuando veamos las cláusulas CUBE y ROLLUP quedará más claro.

## 3. Agrupamiento de filas (cláusula GROUP BY).

Hasta ahora las consultas sumarias que hemos visto obtienen totales de todas las filas del origen y producen una única fila de resultado.

Muchas veces cuando calculamos resúmenes nos interesan totales parciales, por ejemplo saber de cada empleado cuánto ha vendido, y cuál ha sido su pedido máximo, de cada cliente cuándo fue la última vez que nos compró, etc.

En todos estos casos en vez de obtener una fila única de resultados necesitamos una fila por cada empleado, cliente, etc.

Podemos obtener estos subtotales con la cláusula GROUP BY.

```
GROUP BY [ ALL ] expresion_agrupacion [ ,...n ]
    [ WITH { CUBE | ROLLUP } ]
```

Una consulta con una cláusula GROUP BY agrupa los datos de la tabla origen y produce una única fila resultado por cada grupo formado. Las columnas indicadas en el GROUP BY se llaman ***columnas de agrupación o agrupamiento*** *GROUPING COLUMS*.

Cuando queremos realizar una agrupación múltiple, por varias columnas, éstas se indican en la cláusula GROUP BY en el orden de mayor a menor agrupación igual que con la cláusula ORDER BY.

e*xpresion_agrupacion* puede ser una columna o una expresión no agregada que haga referencia a una columna devuelta por la cláusula FROM. Un alias de columna que esté definido en la lista de selección no puede utilizarse para especificar una columna de agrupamiento.

No se pueden utilizar columnas de tipo **text**, **ntext** e **image** en e*xpresion_agrupacion*.

En las cláusulas GROUP BY que no contengan CUBE o ROLLUP, el número de columnas de agrupación está limitado por los tamaños de columna de GROUP BY, las columnas de agregado y los valores de agregado que participan en la consulta. Este límite procede del límite de 8.060 bytes de la tabla de trabajo intermedia que se necesita para contener los resultados intermedios de la consulta. Se permite un máximo de 10 expresiones de agrupamiento cuando se especifica CUBE o ROLLUP.

Si en la columna de agrupación existen valores nulos, se generará una fila de resumen para este "valor", en este caso se considera el valor nulo como otro valor cualquiera.

Ejemplo:

```
SELECT oficina, count(numemp) AS [Número de empleados]
FROM empleados
GROUP BY oficina;
```

Resultado:

| Oficina | Número de empleados |
|---|---|
| NULL | 2 |
| 11 | 2 |
| 12 | 3 |
| 13 | 1 |
| 21 | 2 |
| 22 | 1 |

Hay empleados sin oficinas (con oficina a nulo), estos forman un grupo con el valor NULL en oficina, en este caso hay dos empleados así.

Podemos indicar varias columnas de agrupación.

Ejemplo:

```
SELECT rep, clie, count(numpedido) AS [Número de pedidos], MAX(importe) AS
[Importe máximo]
FROM pedidos
WHERE YEAR(fechapedido) = 1997
GROUP BY  rep, clie
ORDER BY rep, clie;
```

Resultado:

| Rep | clie | Número de pedidos | Importe máximo |
|---|---|---|---|
| 101 | 2113 | 1 | 225,00 |
| 102 | 2106 | 2 | 21,30 |
| 102 | 2120 | 1 | 37,50 |
| 103 | 2111 | 2 | 21,00 |
| 105 | 2103 | 4 | 275,00 |
| 105 | 2111 | 1 | 37,45 |
| 106 | 2101 | 1 | 14,58 |
| 107 | 2109 | 1 | 313,50 |
| 107 | 2124 | 2 | 24,30 |
| 108 | 2112 | 1 | 29,25 |
| 108 | 2114 | 1 | 71,00 |
| 108 | 2118 | 3 | 14,20 |

De cada representante obtenemos el número de pedidos y el importe máximo vendido a cada cliente, de las ventas de 1997. La cláusula ORDER BY se ha incluido para que las filas aparezcan ordenadas y quede más claro.

Hemos dicho que los resúmenes se calculan sobre todas las filas del origen después de haber ejecutado el WHERE, pues ALL permite obtener un resumen de las filas que no cumplen el WHERE.

ALL Incluye todos los grupos y conjuntos de resultados, incluso aquellos en los que no hay filas que cumplan la condición de búsqueda especificada en la cláusula WHERE. Cuando se especifica ALL, se devuelven valores NULL para las columnas de resumen de los grupos que no cumplen la condición de búsqueda. No se puede especificar ALL con los operadores CUBE y ROLLUP.

GROUP BY ALL no se admite en consultas que tienen acceso a tablas remotas si también hay una cláusula WHERE en la consulta.

Por ejemplo, vamos a modificar la consulta anterior:

```
SELECT rep, clie, count(numpedido) AS [Número de pedidos], MAX(importe) AS
[Importe máximo]
FROM pedidos
WHERE YEAR(fechapedido) = 1997
GROUP BY  ALL rep, clie
ORDER BY rep, clie;
```

Resultado:

| Rep | clie | Número de pedidos | Importe máximo |
|---|---|---|---|
| 101 | 2102 | 0 | NULL |
| 101 | 2108 | 0 | NULL |
| 101 | 2113 | 1 | 225,00 |
| 102 | 2106 | 2 | 21,30 |
| 102 | 2120 | 1 | 37,50 |
| 103 | 2111 | 2 | 21,00 |
| 105 | 2103 | 4 | 275,00 |
| 105 | 2111 | 1 | 37,45 |
| 106 | 2101 | 1 | 14,58 |
| 106 | 2117 | 0 | NULL |
| 107 | 2109 | 1 | 313,50 |
| 107 | 2124 | 2 | 24,30 |
| 108 | 2112 | 1 | 29,25 |
| 108 | 2114 | 1 | 71,00 |
| 108 | 2118 | 3 | 14,20 |

¿Cuál ha sido el efecto de añadir ALL? Se han añadido filas para las filas del origen que no cumplen la condición del WHERE pero sin que intervengan en el cálculo de las funciones de agregado.

Por ejemplo el representante 101 tiene pedidos con el cliente 2102 pero estos pedidos no son del año 1997, por eso aparece la primera fila (no estaba en el resultado de la otra consulta) pero con 0 y NULL como resultados de las funciones de agregado.

ROLLUP especifica que, además de las filas que normalmente proporciona GROUP BY, se incluyen filas de resumen en el conjunto de resultados. Los grupos se resumen en un orden jerárquico, desde el nivel inferior del grupo hasta el superior. La jerarquía del grupo se determina por el orden en que se especifican las columnas de agrupamiento. Cambiar el orden de las columnas de agrupamiento puede afectar al número de filas generadas en el conjunto de resultados.

Por ejemplo:

```
SELECT rep, clie, count(numpedido) AS [Número de pedidos], MAX(importe) AS
[Importe máximo]
FROM pedidos
WHERE YEAR(fechapedido) = 1997
GROUP BY  rep, clie WITH ROLLUP;
```

Resultado:

| Rep | clie | Número de pedidos | Importe máximo |
|---|---|---|---|
| 101 | 2113 | 1 | 225,00 |
| 101 | NULL | 1 | 225,00 |
| 102 | 2106 | 2 | 21,30 |
| 102 | 2120 | 1 | 37,50 |
| 102 | NULL | 3 | 37,50 |
| 103 | 2111 | 2 | 21,00 |
| 103 | NULL | 2 | 21,00 |
| 105 | 2103 | 4 | 275,00 |
| 105 | 2111 | 1 | 37,45 |
| 105 | NULL | 5 | 275,00 |
| 106 | 2101 | 1 | 14,58 |
| 106 | NULL | 1 | 14,58 |
| 107 | 2109 | 1 | 313,50 |
| 107 | 2124 | 2 | 24,30 |
| 107 | NULL | 3 | 313,50 |
| 108 | 2112 | 1 | 29,25 |
| 108 | 2114 | 1 | 71,00 |
| 108 | 2118 | 3 | 14,20 |
| 108 | NULL | 5 | 71,00 |
| - | - | - | - |
| NULL | NULL | 23 | 313,50 |

Efecto: Se han añadido automáticamente subtotales por cada nivel de agrupamiento y una línea de totales generales al final. En este caso no hemos incluido ORDER BY porque las filas salen ya ordenadas.

CUBE especifica que, además de las filas que normalmente proporciona GROUP BY, deben incluirse filas de resumen en el conjunto de resultados. Se devuelve una fila de resumen GROUP BY por cada posible combinación de grupo y subgrupo del conjunto de resultados. En el resultado se muestra una fila de resumen GROUP BY como NULL, pero se utiliza para indicar todos los valores.

Por ejemplo:

```
SELECT rep, clie, count(numpedido) AS [Número de pedidos], MAX(importe) AS
[Importe máximo]
FROM pedidos
WHERE YEAR(fechapedido) = 1997
GROUP BY  rep, clie WITH CUBE;
```

Resultado:

| Rep | clie | Número de pedidos | Importe máximo |
|---|---|---|---|
| 101 | 2113 | 1 | 225,00 |
| 101 | NULL | 1 | 225,00 |
| 102 | 2106 | 2 | 21,30 |
| 102 | 2120 | 1 | 37,50 |
| 102 | NULL | 3 | 48,80 |
| 103 | 2111 | 2 | 21,00 |
| 103 | NULL | 2 | 21,00 |
| 105 | 2103 | 4 | 275,00 |
| 105 | 2111 | 1 | 37,45 |
| 105 | NULL | 5 | 312,45 |
| 106 | 2101 | 1 | 14,58 |
| 106 | NULL | 1 | 14,58 |
| 107 | 2109 | 1 | 313,50 |
| 107 | 2124 | 2 | 24,30 |
| 107 | NULL | 3 | 337,80 |
| 108 | 2112 | 1 | 29,25 |
| 108 | 2114 | 1 | 71,00 |
| 108 | 2118 | 3 | 14,20 |
| 108 | NULL | 5 | 114,45 |
| - | - | - | - |
| NULL | NULL | 23 | 450,00 |
| NULL | 2101 | 1 | 14,58 |
| NULL | 2103 | 4 | 275,00 |
| NULL | 2106 | 2 | 21,30 |
| NULL | 2107 | 1 | 6,32 |
| NULL | 2108 | 1 | 56,25 |
| NULL | 2109 | 1 | 313,50 |
| NULL | 2111 | 3 | 37,45 |
| NULL | 2112 | 2 | 450,00 |
| NULL | 2113 | 1 | 225,00 |
| NULL | 2114 | 1 | 71,00 |
| NULL | 2118 | 3 | 14,20 |
| NULL | 2120 | 1 | 37,50 |
| NULL | 2124 | 2 | 24,30 |

Efecto: Obtenemos además de los resultados obtenidos con ROLLUP (los totales por cada representante), los totales por el otro criterio (los totales por cada cliente).

El número de filas de resumen del conjunto de resultados se determina mediante el número de columnas que contiene la cláusula GROUP BY. Cada operando (columna) de la cláusula GROUP BY se enlaza según el agrupamiento NULL y se aplica el agrupamiento al resto de los operandos (columnas). CUBE devuelve todas las combinaciones posibles de grupo y subgrupo.

Tanto si utilizamos CUBE como ROLLUP, nos será útil la función de agregado GROUPING.

Si cogemos por ejemplo la primera fila remarcada (101 NULL …) el valor NULL, no sabemos si se refiere a una fila de subtotal o a que el representante 101 ha realizado un pedido sin número de cliente. Para poder salvar este problema se utiliza la función de agregado GROUPING.

```
SELECT rep, clie, count(numpedido) AS [Número de pedidos], MAX(importe) AS
[Importe máximo], GROUPING(clie) AS [Fila resumen]
FROM pedidos
WHERE YEAR(fechapedido) = 1997
GROUP BY  rep, clie WITH ROLLUP;
```

| Rep | clie | Número de pedidos | Importe máximo | Fila Resumen |
|---|---|---|---|---|
| 101 | 2113 | 1 | 225,00 | 0 |
| 101 | NULL | 1 | 225,00 | 1 |
| 102 | 2106 | 2 | 21,30 | 0 |
| 102 | 2120 | 1 | 37,50 | 0 |
| 102 | NULL | 3 | 48,80 | 1 |
| 103 | 2111 | 2 | 21,00 | 0 |

Las filas que corresponden a subtotales aparecen con un 1 y las normales con un cero.

Ahora que estamos más familiarizados con las columnas de agrupamiento debemos comentar una regla a no olvidar:

**EN LA LISTA DE SELECCIÓN DE UNA CONSULTA DE RESUMEN UN NOMBRE DE COLUMNA NO PUEDE APARECER FUERA DE UNA FUNCIÓN DE AGREGADO SI NO ES UNA COLUMNA DE AGRUPACIÓN.**

## 4. Selección sobre grupos de filas, la cláusula HAVING

Cuando queremos incluir una cláusula de selección sobre las filas del origen, utilizamos la cláusula WHERE, pero cuando estamos definiendo una consulta de resumen, no podemos utilizar esta cláusula para seleccionar filas del resultado ya que cada una de éstas representa un grupo de filas de la tabla original. Para seleccionar filas del resumen tenemos la cláusula HAVING.

```
 HAVING condición de búsqueda
```

HAVING funciona igual que la cláusula WHERE pero en vez de actuar sobre las filas del origen de datos, actúa sobre las filas del resultado, selecciona grupos de filas por lo que la condición de búsqueda sufrirá alguna limitación, la misma que para la lista de selección:
Ejemplo:

```
SELECT oficina, count(numemp) AS [Número de empleados]
FROM empleados
GROUP BY oficina
HAVING COUNT(numemp)<2;
```

Resultado:

| Oficina | Número de empleados |
|---|---|
| 13 | 1 |
| 22 | 1 |

Esta SELECT es la misma que la del primer ejemplo del apartado sobre la cláusula GROUP BY, la diferencia es que le hemos añadido la cláusula HAVING, que hace que del resultado sólo se visualicen los grupos que cumplan la condición. Es decir sólo aparecen las oficinas que tienen menos de 2 empleados.

Siempre que en una condición de selección haya una función de columna, la condición deberá incluirse en la cláusula HAVING, además, como HAVING filtra filas del resultado, sólo puede contener expresiones (nombres de columnas, expresiones, funciones…) que también pueden aparecer en la lista de selección, por lo que también se aplica la misma regla a no olvidar:

**EN LA CLÁUSULA HAVING UN NOMBRE DE COLUMNA NO PUEDE APARECER FUERA DE UNA FUNCIÓN DE AGREGADO SI NO ES UNA COLUMNA DE AGRUPACIÓN.**

Las expresiones que pongamos en HAVING no tienen porqué aparecer en la lista de selección, por ejemplo esta sentencia es correcta:

```
SELECT oficina, count(numemp) AS [Número de empleados]
FROM empleados
GROUP BY oficina
HAVING SUM(ventas)> 100000;
```

Recupera el nº de oficina y cuántos empleados tiene la oficina, pero únicamente de las oficinas cuyos empleados hayan vendido entre todos más de 100000 €.

# 5. Uso de tablas derivadas

El uso de tablas derivadas puede solventar alguno de los problemas que se nos pueden plantear con consultas de resumen. Como por ejemplo calcular una función de columna sobre otra función de columna. Hemos visto que no podemos poner dentro de una función de columna otra función de columna. Por ejemplo si queremos saber de media cuántos empleados tienen nuestras oficinas, tenemos que calcular la media del nº de empleados de cada oficina es decir AVG(COUNT(numemp)) pero esto no lo podemos hacer porque sería una función de columna -COUNT()- dentro de otra función de columna -AVG()-, para resolverlo podemos utilizar una tabla derivada, en la tabla derivada obtenemos el nº de empleados que tiene cada oficina y luego sobre esa tabla calculamos la media:

```
SELECT AVG(cuantosempleados) AS media
FROM (SELECT COUNT(numemp) AS cuantosempleados FROM Empleados GROUP BY
oficina) AS derivada;
```

En Transact-SQL es obligatorio utilizar un alias de tabla para la tabla derivada, la consulta que la genera se escribe entre paréntesis y no lleva punto y coma final como vimos en el tema de consultas multitabla.