Practicar con el GROUP BY

Es frecuente que los alumnos confundan el GROUP BY con el ORDER BY, es algo bastante normal porque a veces, en el lenguaje hablado decimos "agrupando por ciudad" cuando lo que queremos decir es que los registros de la misma ciudad aparezcan juntos y esto no es un GROUP BY, es un ORDER BY.

El GROUP BY se emplea normalmente cuando tenemos en la lista de selección una función de agregado y queremos obtener un resultado "por cada …algo" y ese "algo" es lo que tenemos que incluir en la cláusula GROUP BY.

Prueba estos ejemplos con la base de datos Gestion Simples.

1.- Obtener cuántos pedidos ha realizado cada representante.
Queremos contar los pedidos pero **de cada** empleado→

```
SELECT COUNT(*) AS [Cantidad de pedidos]
FROM Pedidos
GROUP BY rep
```

Prueba la sentencia y observa el resultado, este ejemplo es el mismo que el primero de los ejemplos de funciones de agregado al que hemos añadido el GROUP BY. Antes salía una única fila de resultados, ahora salen tantas filas como valores distintos hay en la columna rep de la tabla Pedidos.

Para que el resultado sea mejor vamos a añadir a la lista de selección el código de representante:

```
SELECT rep, COUNT(*) AS [Cantidad de pedidos]
FROM Pedidos
GROUP BY rep
```

Así sabemos a quién corresponde cada total.

2.- Cambiaremos la instrucción anterior para que ahora aparezcan todos los empleados, incluso los que no tienen pedidos.

Esto sólo se puede hacer utilizando también la tabla de Empleados (donde tenemos todos los empleados). Así que añadiremos a cada pedido el empleado correspondiente y para que salgan también los empleados que no tienen pedidos utilizaremos un OUTER JOIN:

```
SELECT numemp, COUNT(numpedido) AS [Cantidad de pedidos]
FROM Pedidos RIGHT JOIN Empleados ON rep=numemp
GROUP BY numemp
```

Ojo que ahora rep puede ser nulo (en los empleados que no tienen pedidos) así que hay que agrupar por numemp, de lo contario se juntarían todos los empleados sin pedidos en un solo total.

Lo puedes probar cambiando en la lista de selección y en el GROUP BY numemp por rep.

Además nos aparece otro problema, al poner RIGHT pueden haber filas de empleados sin pedido, si ponemos COUNT (*) nos contará 1 en esas filas cuando el empleado no tiene pedidos. Así que hay que utilizar COUNT(columna) y poner una columna que solo toma el valor nulo en estos casos. Tiene que ser cualquier columna que no sea nula si la fila corresponde a un pedido, y sea nula cuando la fila corresponda a un empleado sin pedido. Por ejemplo numpedido o codigo.

3.- Ahora queremos que aparezca también el nombre del empleado:

Podrías pensar, pues como tengo el nombre en la tabla Empleados y esa tabla está en la FROM, añado el nombre en la lista de selección y a correr:

```
SELECT numemp, nombre, COUNT(numpedido) AS [Cantidad de pedidos]
FROM Pedidos RIGHT JOIN Empleados ON rep=numemp
GROUP BY numemp
```

Pruébalo. ¿Qué ocurre? Fíjate bien en el mensaje de error, seguro que te aparece más de una vez en todo el curso.

Lo que pasa es que hemos puesto en la lista de selección un campo suelto (que no está contenido en una función de agregado) que no está en el GROUP BY, y eso NUNCA lo podré hacer, en el examen yo también sacaré tarjeta roja si lo haces (bueno amarilla). Ojo con este error, es grave pero fácil de detectar y corregir.

Es muy importante entender lo que hace el GROUP BY y qué pasa si añado una nueva columna a la cláusula.

Por ejemplo tenemos esta SELECT:

```
SELECT …
FROM oficinas
GROUP BY region
```

El resultado tendrá tantas filas como regiones distintas hayan en la tabla oficinas.

Si al GROUP BY le añadimos una columna:

```
SELECT …
FROM oficinas
GROUP BY region, oficina
```

Ahora tendré una fila por cada región y valor de ventas es decir que si en la misma región todas las oficinas tienen ventas distintas aparecerán tantas filas como oficinas, pero si en la misma región hay varias oficinas con las mismas ventas, saldrá una sola fila agrupándolas. Los grupos han cambiado.

Antes de añadir el nombre en el GROUP BY, siempre hay que preguntarse si al añadir el nuevo campo no van a cambiar los grupos de filas, si no cambian, se puede añadir el campo al GROUP BY sin problemas, pero si cambiasen los grupos entonces habría que pensar si realmente queremos el campo en la lista de selección.

En este caso, vamos a tener un grupo de filas por cada código de empleado, si añadimos el nombre al GROUP BY tendremos un grupo de filas por cada código y nombre, pero como por cada código hay un solo nombre, los grupos serán los mismos → puedo añadir el campo al GROUP BY:

```
SELECT numemp, nombre, COUNT(numpedido) AS [Cantidad de pedidos]
FROM Pedidos RIGHT JOIN Empleados ON rep=numemp
GROUP BY numemp, nombre
```

Imagina que en vez del nombre quisieras añadir la fecha del pedido a la lista de selección. Si añades fechapedido al GROUP BY los grupos van a cambiar porque un código de empleado puede tener varias fechas de pedido por lo que los nuevos grupos serán más pequeños, pruébalo:

```
SELECT numemp, fechapedido, COUNT(numpedido) AS [Cantidad de pedidos]
FROM Pedidos RIGHT JOIN Empleados ON rep=numemp
GROUP BY numemp, fechapedido
```

Y entonces el resumen va a cambiar, ahora no contaré el número de pedidos por empleado sino el número de pedidos por empleado y fecha, ¿Es este el total que quiero obtener? Si la respuesta es que sí, entonces puedo añadir fechapedido a la lista de selección y al GROUP BY, si la respuesta es que no, entonces no puedo incluir la fecha. Esto es bastante lógico, si se quiere un total por representante, es decir una sóla línea por representante ¿Qué fecha se visualizaría, si el representante puede tener varias fechas?

En resumen, para saber si necesitas un GROUP BY te tienes que preguntar si quieres una sóla fila de resultados o si quieres los totales de cada X, esa X será lo que pondrás inicialmente en el GROUP BY. Después podrás añadir al GROUP BY las columnas que quieras poner en la lista de selección siempre que no cambien los grupos al añadir la columna.

4.- Ahora queremos saber en cada oficina cuántos empleados tenemos asignados a esta oficina y cuánto (en euros) vendieron el año pasado. Sólo aparecerán las oficinas que tengan empleados.

Necesito los pedidos y la tabla empleados para saber a qué oficina pertenece el empleado que ha realizado el pedido. OUTER JOIN porque interesan los empleados que no han vendido nada (interesan las oficinas que tengan empleados).

```
SELECT
FROM empleados LEFT JOIN Pedidos ON rep=numemp
```

No nos interesan todos los pedidos, sólo los del año pasado:

```
SELECT
FROM empleados LEFT JOIN Pedidos ON rep=numemp
WHERE YEAR(fechapedido) = YEAR(GETDATE())-1
```

En **cada** oficina tenemos que **contar** los empleados →

```
SELECT oficina, COUNT(DISTINCT numemp) AS [Cuantos empleados]
FROM empleados LEFT JOIN Pedidos ON rep=numemp
WHERE YEAR(fechapedido) = YEAR(GETDATE())-1
GROUP BY oficina
```

Como queremos un resultado por cada oficina empezamos por agrupar por oficina. Para contar los empleados de la oficina tenemos que utilizar COUNT() pero como el origen (FROM) incluye la tabla pedidos vamos a tener el mismo empleado repetido tantas veces como pedidos tenga, luego tenemos que contar los numemp sin contar las repeticiones → COUNT(DISTINCT numemp).

También queremos saber cuánto vendieron, para eso tenemos que sumar los importes de los pedidos:

```
SELECT oficina, COUNT(DISTINCT numemp) AS [Cuantos empleados],
SUM(importe) AS [Importe vendido]
FROM empleados LEFT JOIN Pedidos ON rep=numemp
WHERE YEAR(fechapedido) = YEAR(GETDATE())-1
GROUP BY oficina
```

Si has probado las instrucciones puede que te hayas dado cuenta de un problema, el LEFT que hemos puesto no sirve de nada porque el WHERE eliminará del origen las filas añadidas por el LEFT y además eliminará los empleados que tengan pedidos pero no en esas fechas.

Podríamos en este caso pensar en añadir ALL al GROUP BY:

```
SELECT oficina, COUNT(DISTINCT numemp) AS [Cuantos empleados],
SUM(importe) AS [Importe vendido]
FROM empleados LEFT JOIN Pedidos ON rep=numemp
WHERE YEAR(fechapedido) = YEAR(GETDATE())-1
GROUP BY ALL oficina
```

Ahora sí aparecen todas las oficinas de los empleados (porque el ALL añade al resultado los valores de la columna de agrupación que no aparecen por estar filtrados por el WHERE) pero no cuenta bien los empleados que hay en cada oficina, porque el ALL añade los valores de las columnas de agrupación pero los resultados de las funciones de agregado (en este caso el COUNT y SUM) se calculan teniendo en cuenta sólo las filas que cumplen el WHERE.

Si no hubiesemos tenido que sacar el nº de empleados nos hubiera valido el ALL, pero en este caso tenemos que hacer que las filas de los empleados que no tienen pedidos o pedidos fuera de las fechas sean tenidas en cuenta, se resuelve mediante tabla derivada:

```
SELECT oficina, COUNT(DISTINCT numemp) AS [Cuantos empleados],
SUM(importe) AS [Importe vendido]
FROM empleados LEFT JOIN (SELECT * FROM Pedidos WHERE YEAR(fechapedido) =
YEAR(GETDATE())-1) AS Ped ON rep=numemp
GROUP BY oficina
```

Por una parte sacamos los pedidos del año anterior y los juntamos con todos los empleados, la consulta de resumen ya no tiene WHERE luego todas las filas van a contar en el resultado y los empleados que no tienen pedidos o tienen pedidos fuera del WHERE se añaden con el LEFT y no se quitan con el WHERE porque ya no hay WHERE.

Prúebalo y compara el resultado con la SELECT anterior (la del GROUP BY ALL).

Cuando trabajamos con OUTER JOINs es muy importante fijarse bien en los campos que utilizamos (los campos que pueden estar en las dos tablas, como por ejemplo aquí los campos rep y numemp). Copia la instrucción anterior y cambia el COUNT(DISTINCT numemp) por COUNT(DISTINCT rep). Ejecuta las dos instrucciones y compara los resultados. ¿Qué ocurre, y por qué?

5.- Ahora vamos a agrupar por varios campos. Queremos saber cuántos pedidos se han realizado en cada oficina, indicando la ciudad y la región de la oficina. Se considera como oficina del pedido la oficina del representante que ha realizado el pedido.

Tenemos que juntar pedidos con empleados y con oficinas:

```
SELECT
FROM Oficinas LEFT JOIN (empleados LEFT JOIN pedidos ON rep = numemp) ON
oficinas.oficina=empleados.oficina
```

LEFT porque interesan los empleados que no tienen pedidos y las oficinas que no tienen empleados.

Si quieres añade un * a la lista de selección y ejecuta para ver el origen de datos que tienes.

Ahora de todas estas filas queremos saber cuántos pedidos tenemos por oficina→ contar numpedido agrupando por oficina, como el campo oficina está en las dos tablas hay que cualificarlo:

```
SELECT oficinas.oficina, COUNT(numpedido) AS [Cuantos pedidos]
FROM Oficinas LEFT JOIN (empleados LEFT JOIN pedidos ON rep = numemp) ON
oficinas.oficina=empleados.oficina
GROUP BY oficinas.oficina
```

En el COUNT podemos poner cualquier campo de Pedidos que no pueda tener valores nulos.

Las filas del origen que representan realmente pedidos son las que tienen valores en los campos de pedidos, las que tienen NULL en todos estos campos son las filas añadidas por los LEFT, no son pedidos reales. Por eso si queremos contar los pedidos tenemos que contar los valores que se encuentran en una columna de Pedidos que sabemos que no admite nulos.

Recuerda que el COUNT cuenta valores no nulos.

Prueba a poner COUNT(*). ¿Qué ocurre?

El COUNT(*) cuenta filas dentro de cada grupo, dentro de cada oficina hay al menos una fila incluso cuando esa oficina no tiene pedidos, la añadida por primer el LEFT. Si la oficina tiene varios empleados sin pedidos (filas añadidas por el segundo LEFT), va a contar uno por cada empleado cuando no debería contar nada.

Además nos piden la ciudad y la región de la oficina, para poder añadir estos campos en la lista de selección, los tenemos que colocar en el GROUP BY.

¿Si añadimos la ciudad cambiarán los grupos? No, porque por cada oficina hay una sola ciudad → Añadimos ciudad al GROUP BY

```
SELECT ciudad, oficinas.oficina, COUNT(numpedido) AS [Cuantos pedidos]
FROM Oficinas LEFT JOIN (empleados LEFT JOIN pedidos ON rep = numemp) ON
oficinas.oficina=empleados.oficina
GROUP BY oficinas.oficina, ciudad
```

¿Si añadimos la región cambiarán los grupos? No, porque por cada oficina hay una sola región → Añadimos región al GROUP BY

```
SELECT region,ciudad, oficinas.oficina, COUNT(numpedido) AS [Cuantos pedidos]
FROM Oficinas LEFT JOIN (empleados LEFT JOIN pedidos ON rep = numemp) ON
oficinas.oficina=empleados.oficina
GROUP BY oficinas.oficina, ciudad, region
```

Para ver mejor el resultado ordenamos:

```
SELECT region,ciudad, oficinas.oficina, COUNT(numpedido) AS [Cuantos pedidos]
FROM Oficinas LEFT JOIN (empleados LEFT JOIN pedidos ON rep = numemp) ON
oficinas.oficina=empleados.oficina
GROUP BY oficinas.oficina, ciudad, region
ORDER BY region, ciudad, oficinas.oficina
```

Hemos escrito una sentencia compleja, pero haciéndolo paso por paso, siguiendo un orden y centrándonos en una sóla cosa en cada paso, es más facil.

6.- Ahora, además quiero que aparezca cuántos pedidos se han vendido en cada ciudad, en cada región y en total.

Aquí lo que nos piden es obtener resúmenes parciales por cada campo que aparece en el GROUP BY→ Hay que utilizar ROLLUP.

```
SELECT region,ciudad, oficinas.oficina, COUNT(numpedido) AS [Cuantos pedidos]
FROM Oficinas LEFT JOIN (empleados LEFT JOIN pedidos ON rep = numemp) ON
oficinas.oficina=empleados.oficina
GROUP BY oficinas.oficina, ciudad, region WITH ROLLUP
ORDER BY region, ciudad, oficinas.oficina
```

Ejecuta y observa el resultado. No es exactamente lo que queríamos, queremos que cuando acaben todas las oficinas de una ciudad aparezca el total de esta ciudad y así sucesivamente.

Con el ROLLUP sí importa el orden que pongamos las columnas de agrupación en el GROUP BY y quitar el ORDER BY→

```
SELECT region,ciudad, oficinas.oficina, COUNT(numpedido) AS [Cuantos pedidos]
FROM Oficinas LEFT JOIN (empleados LEFT JOIN pedidos ON rep = numemp) ON
oficinas.oficina=empleados.oficina
GROUP BY region, ciudad, oficinas.oficina WITH ROLLUP
```

Pero así el resultado es un poco difícil de interpretar porque no se ven claramente las filas que son totales.

Vamos a utilizar la función GROUPING() para saber cuáles son esa filas.

```
SELECT region,ciudad, oficinas.oficina, COUNT(numpedido) AS [Cuantos
pedidos], GROUPING(oficinas.oficina) AS [Total ciudad], GROUPING(ciudad) AS
[Total región], GROUPING(region) AS [Total general]
FROM Oficinas LEFT JOIN (empleados LEFT JOIN pedidos ON rep = numemp) ON
oficinas.oficina=empleados.oficina
GROUP BY region, ciudad, oficinas.oficina WITH ROLLUP
```

Fíjate en el resultado. La fila que lleva un 1 en los tres GROUPING() es el total general ya que totaliza todas las regiones, las filas que llevan un 1 en GROUPING(ciudad) y GROUPING(oficinas.oficina) totalizan todas las ciudades de una misma región (del grupo superior), las que llevan un 1 sólo en el GROUPING(oficinas.oficina) totalizan oficinas de cada ciudad (del grupo superior) y las que llevan ceros en todos los GROUPING() son las filas del detalle (las que saldrían sin ROLLUP).

Podemos utilizar estas funciones para dejar el resultado más claro. Por ejemplo poniendo puntos en la columna oficina cuando es un total:

En la tercera columna ahora queremos que aparezcan tres puntos si el GROUPING(oficinas.oficina) es uno y sino que aparezca el código de la oficina (oficinas.oficina). Es decir que ahora en la columna aparecerá un valor u otro dependiendo del valor de un campo, para eso utilizamos el CASE campo WHEN valor de campo THEN valor devuelto ELSE valor devuelto.

```
SELECT region,ciudad, CASE GROUPING(oficinas.oficina) WHEN 1 THEN '…' ELSE
STR(oficinas.oficina,3) END AS Ofi, COUNT(numpedido) AS [Cuantos pedidos],
GROUPING(oficinas.oficina) AS [Total ciudad], GROUPING(ciudad) AS [Total
región], GROUPING(region) AS [Total general]
FROM Oficinas LEFT JOIN (empleados LEFT JOIN pedidos ON rep = numemp) ON
oficinas.oficina=empleados.oficina
GROUP BY region, ciudad, oficinas.oficina WITH ROLLUP
```

Los posibles valores devueltos por CASE deben ser del mismo tipo, como en un caso devuelve una cadena (`'…'`) y la oficina es un entero, tenemos que convertir esta a cadena, lo hacemos por ejemplo con la función STR().

Podemos hacer lo mismo en la columna Ciudad:

```
SELECT region,CASE GROUPING(ciudad) WHEN 1 THEN '…' ELSE ciudad END AS Ciu,
CASE GROUPING(oficinas.oficina) WHEN 1 THEN '…' ELSE STR(oficinas.oficina,3)
END AS Ofi, COUNT(numpedido) AS [Cuantos pedidos], GROUPING(oficinas.oficina)
AS [Total ciudad], GROUPING(ciudad) AS [Total región], GROUPING(region) AS
[Total general]
FROM Oficinas LEFT JOIN (empleados LEFT JOIN pedidos ON rep = numemp) ON
oficinas.oficina=empleados.oficina
GROUP BY region, ciudad, oficinas.oficina WITH ROLLUP
```

Y para el total general:

Podemos hacer lo mismo en la columna Ciudad:

```
SELECT CASE GROUPING(region) WHEN 1 THEN 'Total General' ELSE region END AS
region,CASE GROUPING(ciudad) WHEN 1 THEN '…' ELSE ciudad END AS Ciu, CASE
GROUPING(oficinas.oficina) WHEN 1 THEN '…' ELSE STR(oficinas.oficina,3) END
AS Ofi, COUNT(numpedido) AS [Cuantos pedidos], GROUPING(oficinas.oficina) AS
[Total ciudad], GROUPING(ciudad) AS [Total región], GROUPING(region) AS
[Total general]
FROM Oficinas LEFT JOIN (empleados LEFT JOIN pedidos ON rep = numemp) ON
oficinas.oficina=empleados.oficina
GROUP BY region, ciudad, oficinas.oficina WITH ROLLUP
```

Como ves la función GROUPING() puede ser muy útil.