UTN
Facultad Regional Córdoba

TUP

Tecnicatura Universitaria en Programación

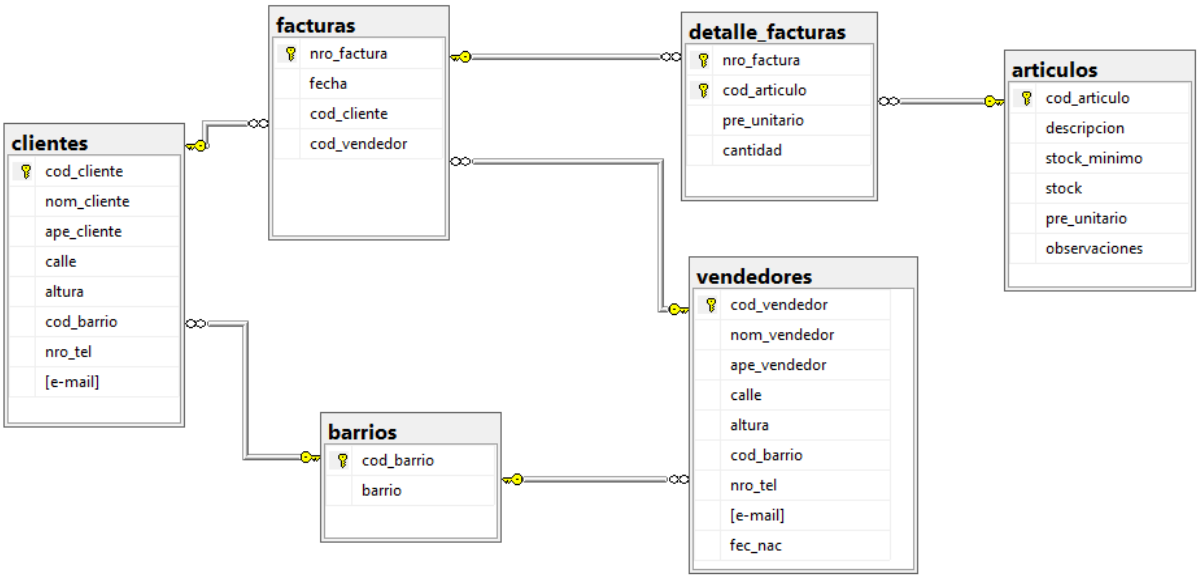
LABORATORIO DE COMPUTACIÓN II

Unidad Temática N°1:
Resumen de Datos

Guía
1° Año – 2° Cuatrimestre

Índice

Para la resolución de esta guía se utilizará la base de datos LIBRERÍA correspondiente a la facturación mayorista de un negocio de venta mayorista de artículos de librería cuyo diagrama en SQL Server es el siguiente:



Para obtener esta base de datos, descargue el script correspondiente que se encuentra en la uvs de esta asignatura y ejecútelo en Microsoft SQL Server Management Studio de su PC. Si está en una computadora de la Facultad deberá ejecutar Microsoft SQL Server Management Studio ingresar el nombre del servidor en el que se encuentra la base de datos Librería, el usuario y la contraseña

Problema 1.1: Combinación de resultados de consultas. UNION

Los responsables de la librería solicitan la emisión de una serie de listados en los que, en cada uno, se halla presente dos o más tablas de resultados. Para solucionarlo se va a utilizar el predicado *UNION*.

1. **Confeccionar un listado de los clientes y los vendedores indicando a qué grupo pertenece cada uno.**

Para el listado de clientes utilizaremos la siguiente consulta:

```
SELECT cod_cliente Código,
       ape_cliente +' '+ nom_cliente Nombre
  FROM clientes
```

Para la de vendedores, esta otra consulta:

```
SELECT cod_vendedor Código,
       ape_vendedor +' '+ nom_vendedor
  FROM vendedores
```

Esto nos estaría dando dos tablas de resultados. Para que ambas consultas aparezcan en una sola tabla de resultado escribiremos lo siguiente:

```
select cod_cliente Código,ape_cliente+' '+nom_cliente Nombre,'Cliente' Tipo
from clientes
union
select cod_vendedor Código,ape_vendedor+' '+nom_vendedor,'Vendedor'
  from vendedores
```

00 %

Results | Messages

	Código	Nombre	Tipo
1	1	Carrizo Martín	Vendedor
2	1	Perez Rodolfo	Cliente
3	2	Castillo Marta Analía	Cliente
4	2	Ledesma Mariela	Vendedor
5	3	Abarca Héctor	Cliente
6	3	Lopez Alejandro	Vendedor
7	4	Miranda Marcelo	Vendedor
8	4	Morales Santiago	Cliente
9	5	Monti Gabriel	Vendedor
10	5	Perez Carlos Antonio	Cliente
11	6	Juarez Susana	Vendedor
12	6	Morales Pilar	Cliente
13	7	Ortega Ana	Vendedor
14	7	Paez Roque	Cliente
15	8	Luque Elvira Josefa	Cliente
16	8	Monti Juan	Vendedor
17	9	Ortega Ana	Vendedor
18	9	Ruiz Marcos	Cliente

Teniendo en cuenta que solo se agrega un alias a la primera consulta y se crea una columna extra con una constante que indica el origen del registro es decir si es un cliente o un vendedor; a esta nueva columna también con un alias.

Cada una de las consultas tiene 3 columnas, donde la primera es un *INTEGER* en ambas consultas, la segunda es *VARCAHAR* y la tercera *VARCHAR*. Y por último si se quiere ver el listado ordenado de alguna manera, por ejemplo,

primero los clientes y luego los vendedores la cláusula *ORDER BY* será la última línea de todas las consultas.

```
SELECT cod_cliente Código,
       ape_cliente +' '+ nom_cliente Nombre,
       'Cliente' Tipo
  FROM clientes
UNION
 SELECT cod_vendedor Código,
       ape_vendedor +' '+ nom_vendedor,
       'Vendedor'
  FROM vendedores
  ORDER BY 3
```

La operación UNION podría producir resultados que contuvieran filas duplicadas, pero por omisión se eliminan. Si se desea mostrar las filas duplicadas en una operación UNION, se puede especificar la palabra clave ***ALL*** luego de la palabra *UNION*.

Ahora sí se puede proceder a la resolución del **problema 1.1** y los requerimientos son:

2. **Se quiere saber qué vendedores y clientes hay en la empresa; para los casos en que su teléfono y dirección de e-mail sean conocidos. Se deberá visualizar el código, nombre y si se trata de un cliente o de un vendedor. Ordene por la columna tercera y segunda.**
3. **Emitir un listado donde se muestren qué artículos, clientes y vendedores hay en la empresa. Determine los campos a mostrar y su ordenamiento.**
4. **Se quiere saber las direcciones (incluido el barrio) tanto de clientes como de vendedores. Para el caso de los vendedores, códigos entre 3 y 12. En ambos casos las direcciones deberán ser conocidas. Rotule como NOMBRE, DIRECCION, BARRIO, INTEGRANTE (en donde indicará si es cliente o vendedor). Ordenado por la primera y la última columna.**
5. **Ídem al ejercicio anterior, sólo que además del código, identifique de donde obtiene la información (de qué tabla se obtienen los datos).**
6. **Listar todos los artículos que están a la venta cuyo precio unitario oscile entre 10 y 50; también se quieren listar los artículos que fueron comprados por los clientes cuyos apellidos comiencen con "M" o con "P".**
7. **El encargado del negocio quiere saber cuánto fue la facturación del año pasado. Por otro lado, cuánto es la facturación del mes pasado, la de este mes y la de hoy (Cada pedido en una consulta distinta, y puede unirla en una sola tabla de resultado)**

Problema 1.2: Consultas Sumarias

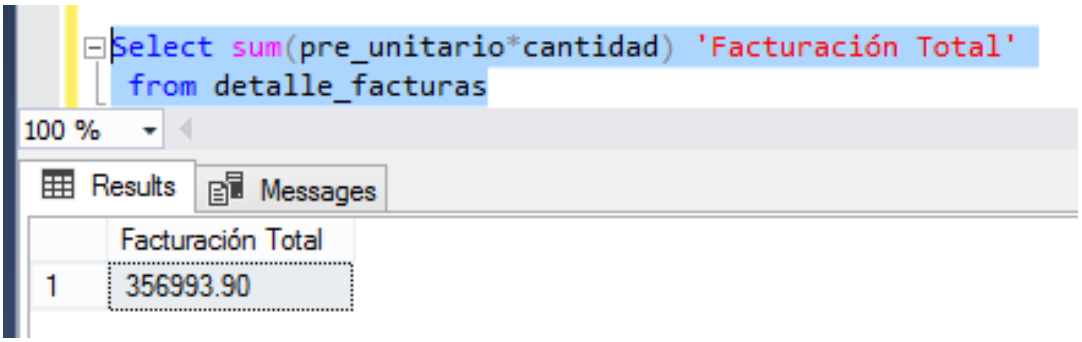
Para llevar un mejor control del funcionamiento del negocio, se pretende saber datos totalizadores, como, por ejemplo:

1. **Facturación total del negocio**

Para poder dar este tipo de resultados se van a utilizar consultas sumarias que emplea un conjunto de funciones de columnas o **funciones de agregado**.

Para este caso la función ***SUM*** para realizar la sumatoria de la cantidad por el precio unitario de todos los registros de la tabla detalle de facturas

La solución sería la siguiente:



2. **También se quiere saber el total de la factura Nro. 236, la cantidad de artículos vendidos, cantidad de ventas, el precio máximo y mínimo vendido.**

Aquí además de realizar la sumatoria de la cantidad por el precio unitario y la sumatoria de las cantidades con la función *SUM se* utilizarán las funciones *COUNT(*)* para contar registros, *MAX* y *MIN* para hallar el valor mayor y el valor menor del campo precio unitario. Filtrando desde la cláusula Where el número de factura solicitado

```
Select sum(pre_unitario*cantidad) Total, sum(cantidad) 'Cant.de Artículos',
count(*) 'Cant.de Items',max(pre_unitario)'Mayor precio',min(pre_unitario)'Menor precio'
 from detalle_facturas
 where nro_factura = 236
```

Results | Messages

Total	Cant.de Artículos	Cant.de Items	Mayor precio	Menor precio
1351.00	52	2	31.00	22.30

3. Se nos solicita además lo siguiente: ¿Cuánto se facturó el año pasado?

En este caso es necesaria la tabla factura ya que en ella se encuentra la fecha.

```
SELECT sum(pre_unitario*cantidad)'Facturación Total del año anterior'
  FROM detalle_facturas d, facturas f
 WHERE d.nro_factura = f.nro_factura
  and  YEAR(fecha) = YEAR(GETDATE())- 1
```

Hay que tener presente que la lista de selección **solo puede** contener campos o expresiones dentro de una función de agregado (o función de columna).

Cómo obtendría el siguiente resultado:

4. ¿Cantidad de clientes con dirección de e-mail sea conocido (no nulo)

Se utilizará la función ***COUNT*** pero hay que observar la siguiente consulta y su resultado, ¿qué diferencia hay entre ambas columnas? En una se utiliza ***COUNT(*)*** y en la otra ***COUNT(COLUMNA)***

```
Select count(*) 'cantidad de clientes',
    count([e-mail]) 'cantidad de clientes con e-mail conocido'
 from clientes
```

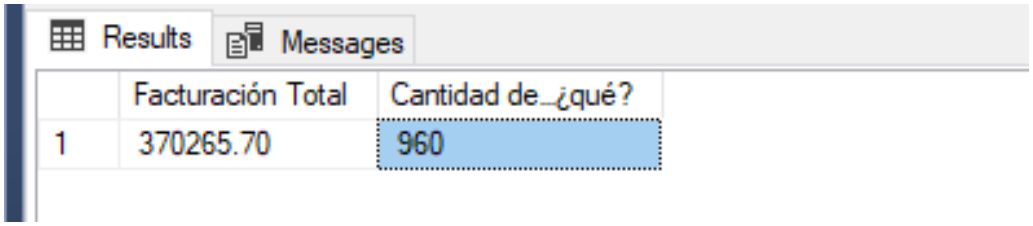
100 % | Results | Messages

	cantidad de clientes	cantidad de clientes con e-mail conocido
1	12	3

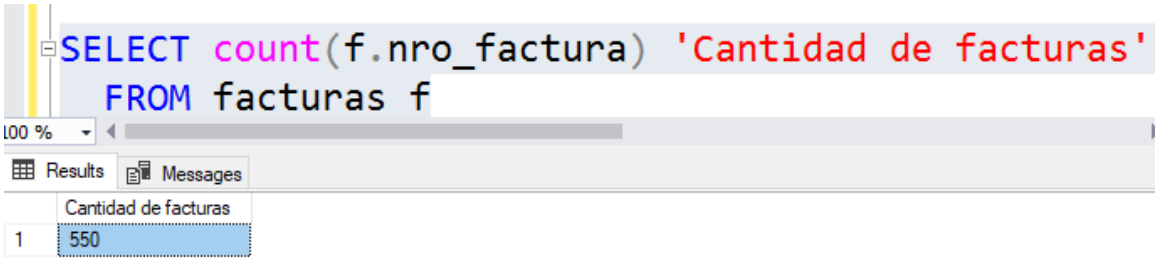
5. ¿Cuánto fue el monto total de la facturación de este negocio? ¿Cuántas facturas se emitieron?

```
SELECT sum(pre_unitario*cantidad) 'Facturación Total',
       count(f.nro_factura) 'Cantidad de…¿qué?'
  FROM detalle_facturas d, facturas f
 WHERE d.nro_factura = f.nro_factura
```

Ejecutar esta consulta y responder qué muestra la segunda columna. ¿Es la cantidad de facturas?



¿Cuántas facturas hay en la tabla facturas? Se pedirá a SQL que dé ese dato:

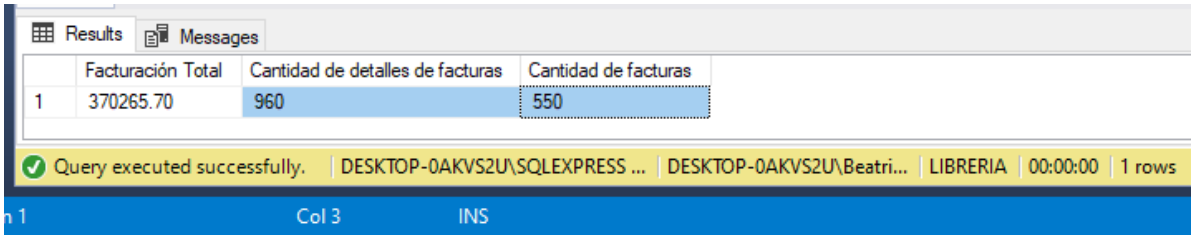


Las funciones `count(f.nro_factura)` de ambas consultas no dan lo mismo.

Tener en cuenta que el nro_factura nunca va a ser Null porque es clave primaria de la tabla facturas entonces `count(f.nro_factura)` va a contar todos los registros que surjan de la composición de la tabla facturas con detalle_facturas y como existe entre ellas una relación de uno a varios, la cantidad de registros es la que tiene la tabla detalle_facturas ya que puede haber varios detalles por cada facturas.

Si a la función de ***COUNT*** la combinamos con ***DISTINCT***, ésta función contará los números de facturas sin tener en cuenta las repeticiones, como se muestra en el ejemplo siguiente:

```
SELECT sum(pre_unitario*cantidad) 'Facturación Total',
       count(f.nro_factura)    'Cantidad    de    detalles    de
facturas',
       count(distinct f.nro_factura) 'Cantidad de facturas'
  FROM detalle_facturas d, facturas f
 WHERE d.nro_factura = f.nro_factura
```



6. **Se necesita conocer el promedio de monto facturado por factura el año pasado.**

Si utilizamos esto:

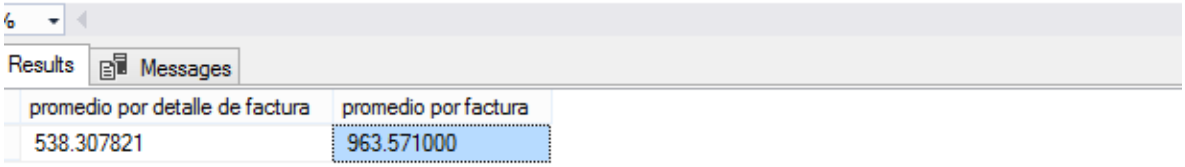
```
SELECT AVG(pre_unitario*cantidad) FROM detalle_facturas
```

La función de agregado ***AVG(COLUMNA)*** suma el valor de la columna y la divide por la cantidad de registros que contenga la consulta, en otras palabras lo que hace es la aplicación de esta operación: ***SUM(COLUMNA)/COUNT(*)*** con lo que no estaría dando el promedio por factura sino por detalle de factura para dar la solución a lo que pide el punto 4 habría que utilizar en la consulta anterior:

```
SUM(pre_unitario*cantidad) / COUNT(DISTINCT nro_factura)
```

Aquí se resuelve dando ambos resultados para que se entienda mejor:

```
select avg(pre_unitario*cantidad) 'promedio por detalle de factura',
       sum(pre_unitario*cantidad)/count(distinct d.nro_factura) 'promedio por factura'
from  detalle_facturas d, facturas f
where d.nro_factura=f.nro_factura
  and year(fecha)=year(getdate())-1
```



Resuelva el resto de los requerimientos del **problema 1.2** de los usuarios del sistema:

7. **Se quiere saber la cantidad de ventas que hizo el vendedor de código 3.**

8. **¿Cuál fue la fecha de la primera y última venta que se realizó en este negocio?**

9. **Mostrar la siguiente información respecto a la factura nro.: 450: cantidad total de unidades vendidas, la cantidad de artículos diferentes vendidos y el importe total.**

10. **¿Cuál fue la cantidad total de unidades vendidas, importe total y el importe promedio para vendedores cuyos nombres comienzan con letras que van de la "d" a la "l"?**

11. **Se quiere saber el importe total vendido, el promedio del importe vendido y la cantidad total de artículos vendidos para el cliente Roque Paez.**

12. **Mostrar la fecha de la primera venta, la cantidad total vendida y el importe total vendido para los artículos que empiecen con "C".**

13. **Se quiere saber la cantidad total de artículos vendidos y el importe total vendido para el periodo del 15/06/2011 al 15/06/2017.**

14. **Se quiere saber la cantidad de veces y la última vez que vino el cliente de apellido Abarca y cuánto gastó en total.**

15. **Mostrar el importe total y el promedio del importe para los clientes cuya dirección de mail es conocida.**

16. **Obtener la siguiente información: el importe total vendido y el importe promedio vendido para números de factura que no sean los siguientes: 13, 5, 17, 33, 24.**

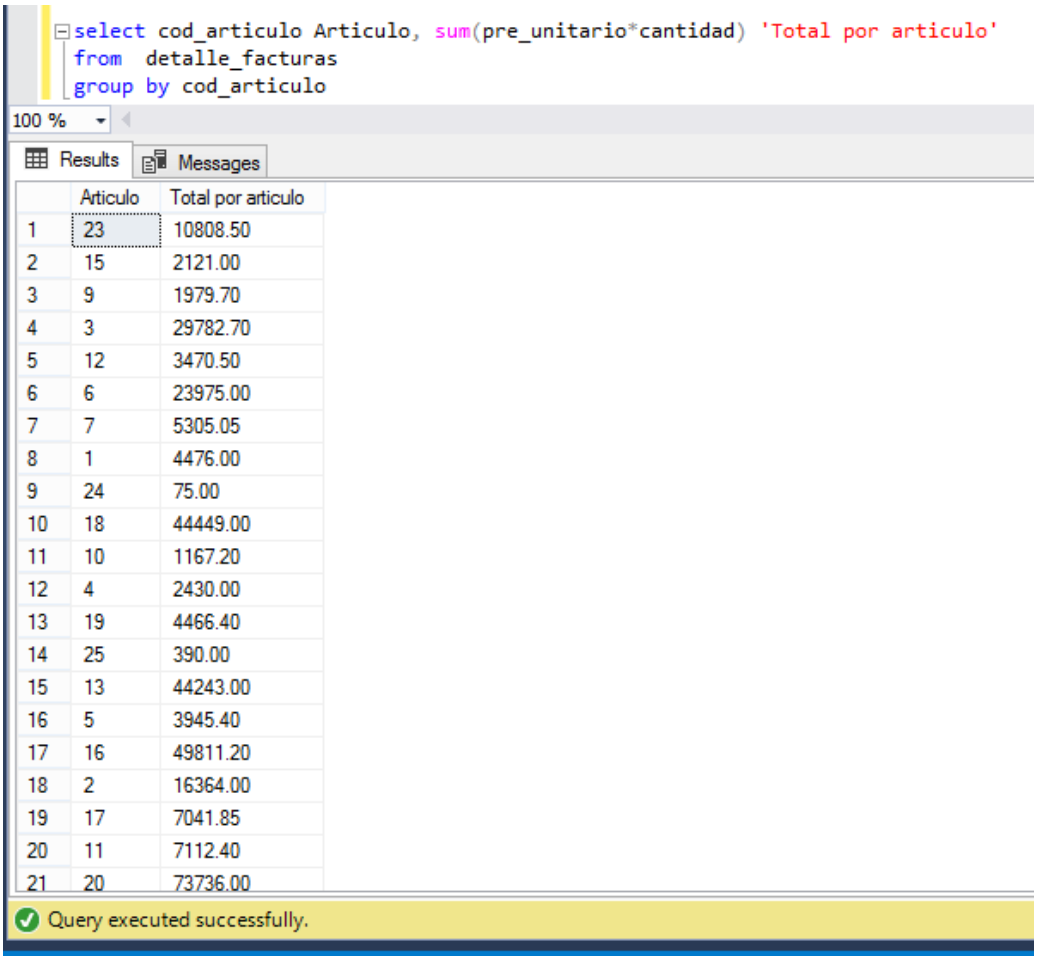
Problema 1.3: Consultas agrupadas: Cláusula GROUP BY

El gerente del negocio necesita otro tipo de información sumaria pero no totales generales sino totales agrupados por algún criterio en particular, por ejemplo:

1. **Los importes totales de ventas por cada artículo que se tiene en el negocio**

Para esto vamos a implementar el siguiente tema:

Las consultas sumarias son como totales finales de un informe; si lo que se necesita es sumarizar los resultados de la consulta a un nivel de subtotal se utiliza la cláusula ***GROUP BY*** de la sentencia ***SELECT***. Es decir, agrupar registros según los valores de una o más columnas y obtener totales de cada grupo.



	Articulo	Total por articulo
1	23	10808.50
2	15	2121.00
3	9	1979.70
4	3	29782.70
5	12	3470.50
6	6	23975.00
7	7	5305.05
8	1	4476.00
9	24	75.00
10	18	44449.00
11	10	1167.20
12	4	2430.00
13	19	4466.40
14	25	390.00
15	13	44243.00
16	5	3945.40
17	16	49811.20
18	2	16364.00
19	17	7041.85
20	11	7112.40
21	20	73736.00

En el punto 1 del problema habría que hacer la sumatoria del precio unitario por la cantidad y que por cada artículo diferente nos dé el resultado es decir que el *SELECT*, desde la tabla *detalle_facturas* agrupe por artículo por ello vamos a agregar una cláusula *GROUP BY cod_articulo* y por cada grupo de estos calcule *SUM*(*cantidad*pre_unitario*), de esta forma:

Observe la lista de selección (cláusula select):

```
SELECT cod_articulo Articulo,
       SUM(pre_unitario*cantidad)
       'Total por articulo'
```

ya no solo hemos incorporado una expresión dentro de una función de agregado, sino que además hay una columna extra: esta columna no es ni más ni menos que la misma por la que agrupamos:

```
GROUP BY cod_articulo
```

Esto estaría dando la facturación total por artículo de absolutamente toda la venta. Si se quisiera que fuera solo lo del año pasado ordenado por código de artículo, deberíamos agregar la tabla facturas y realizar la composición con la tabla detalle_facturas en el *WHERE* además de agregar la condición de búsqueda por el año; la consulta sería la siguiente:

```
SELECT cod_articulo Articulo,
       SUM(pre_unitario*cantidad)'Total por articulo'
  FROM detalle_facturas d, facturas f
 WHERE d.nro_factura = f.nro_factura
   AND YEAR(fecha) = YEAR(GETDATE())- 1
 GROUP BY cod_articulo
 ORDER BY cod_articulo
```

Preste atención en el orden de las cláusulas de toda la sentencia 1° **SELECT**, 2° **FROM**, 3° **WHERE**, 4° **GROUP BY** y, por último, **ORDER BY**.

Se puede observar que la lista de selección contiene únicamente campos o expresiones dentro de una función de agregado (función sumaria) y la columna (o columnas) de agrupación es decir el campo (o campos) incluido en el *GROUP BY*.

Múltiples columnas de agrupación.

SQL puede agrupar resultados de consultas en base a contenidos de dos o más columnas. Por ejemplo, calcular el total facturado por cada vendedor y a cada cliente el año pasado ordenado por vendedor primero y luego por cliente:

```
select v.cod_vendedor,ape_vendedor+' '+nom_vendedor'Vendedor', ape_cliente+' '+nom_cliente'Cliente',sum(pre_unitario*cantidad)'Total'
from  detalle_facturas d, facturas f,vendedores v, clientes c
where d.nro_factura=F.nro_factura and f.cod_cliente=c.cod_cliente
 and  f.cod_vendedor=v.cod_vendedor and year(fecha)=year(getdate())-1
group by v.cod_vendedor,ape_vendedor+' '+nom_vendedor,c.cod_cliente,ape_cliente+' '+nom_cliente
order by 2,3
```

100 %

Results | Messages

	cod_vendedor	Vendedor	Cliente	Total
1	1	Carrizo Martín	Abarca Héctor	817.00
2	1	Carrizo Martín	Castillo Marta Analía	558.00
3	1	Carrizo Martín	Luque Elvira Josefa	150.00
4	1	Carrizo Martín	Morales Santiago	3390.00
5	1	Carrizo Martín	Paez Roque	200.00
6	1	Carrizo Martín	Perez Carlos Antonio	3406.00
7	1	Carrizo Martín	Perez Rodolfo	551.50
8	1	Carrizo Martín	Ruiz Marcos	800.00
9	2	Ledesma Mariela	Abarca Héctor	996.00
10	2	Ledesma Mariela	Castillo Marta Analía	120.00
11	2	Ledesma Mariela	Luque Elvira Josefa	4759.00
12	2	Ledesma Mariela	Morales Pilar	4665.00
13	2	Ledesma Mariela	Morales Santiago	2692.50
14	2	Ledesma Mariela	Paez Roque	2690.00
15	2	Ledesma Mariela	Perez Carlos Antonio	299.00
16	2	Ledesma Mariela	Perez Rodolfo	1672.00
17	2	Ledesma Mariela	Ruiz Marcos	2574.50
18	3	Lopez Alejandro	Abarca Héctor	1174.00

Esta consulta respondería a la pregunta: ¿Cuánto le vendió cada vendedor a cada cliente el año pasado?

Observe detenidamente, qué es lo que se incluye como columnas de agrupación (en el *GROUP BY*) y qué columnas son las que se utilizaron en la lista de selección.

Ahora puede seguir resolviendo los requerimientos del problema 1.3:

2. **Por cada factura emitida mostrar la cantidad total de artículos vendidos (suma de las cantidades vendidas), la cantidad ítems que tiene cada factura en el detalle (cantidad de registros de detalles) y el Importe total de la facturación de este año.**

3. **Se quiere saber en este negocio, cuánto se factura:**
   a. **Diariamente**
   b. **Mensualmente**
   c. **Anualmente**

4. **Emitir un listado de la cantidad de facturas confeccionadas diariamente, correspondiente a los meses que no sean enero, julio ni diciembre. Ordene por la cantidad de facturas en forma descendente y fecha.**

5. **Se quiere saber la cantidad y el importe promedio vendido por fecha y cliente, para códigos de vendedor superiores a 2. Ordene por fecha y cliente.**

6. **Se quiere saber el importe promedio vendido y la cantidad total vendida por fecha y artículo, para códigos de cliente inferior a 3. Ordene por fecha y artículo.**

7. **Listar la cantidad total vendida, el importe total vendido y el importe promedio total vendido por número de factura, siempre que la fecha no oscile entre el 13/2/2007 y el 13/7/2010.**

8. **Emitir un reporte que muestre la fecha de la primer y última venta y el importe comprado por cliente. Rotule como CLIENTE, PRIMER VENTA, ÚLTIMA VENTA, IMPORTE.**

9. **Se quiere saber el importe total vendido, la cantidad total vendida y el precio unitario promedio por cliente y artículo, siempre que el nombre del cliente comience con letras que van de la "a" a la "m". Ordene por cliente, precio unitario promedio en forma descendente y artículo. Rotule como IMPORTE TOTAL, CANTIDAD TOTAL, PRECIO PROMEDIO.**

10. **Se quiere saber la cantidad de facturas y la fecha la primer y última factura por vendedor y cliente, para números de factura que oscilan entre 5 y 30. Ordene por vendedor, cantidad de ventas en forma descendente y cliente.**

Problema 1.4: Consultas agrupadas: Cláusula HAVING

1. **Se necesita saber el importe total de cada factura, pero solo aquellas donde ese importe total sea superior a 2500.**

En este caso se pide una restricción a las filas de resultado luego del agrupamiento y sabemos que el *WHERE* se aplica antes de la agrupación.

Para este caso vamos a utilizar cláusula *HAVIG* para agregar condiciones de búsqueda para grupos es decir para las filas que resultan de la agrupación y cálculo de resultados de las funciones de agregado.

```
Select nro_factura, sum(pre_unitario*cantidad) Total
  from detalle_facturas
  group by nro_factura
  having sum(pre_unitario*cantidad)>2500
```

100 %

Results | Messages

	nro_factura	Total
1	45	2565.00
2	167	2630.00
3	223	3012.50
4	287	2630.00
5	311	2630.00
6	403	3470.90
7	406	3058.00
8	414	2947.70
9	420	4170.00
10	422	3040.00
11	461	2510.00
12	465	3154.50
13	466	2662.50
14	470	2966.00
15	475	2878.50
16	513	2857.30
17	519	2533.50
18	522	3040.00
19	532	3040.00
20	537	2734.50

Se pueden utilizar las mismas condiciones o test utilizados para la cláusula WHERE.

La cláusula HAVING se utiliza para incluir o excluir grupos de filas de los resultados de la consulta, por lo que la condición de búsqueda que especifica debe ser aplicable al grupo en su totalidad en lugar de a filas individuales. Esto significa que un elemento que aparezca dentro de la condición de búsqueda en el HAVING pueden ser las mismas que las enumeradas en el GROUP BY.

Continuando con el **problema 1.4:**

2. **Se desea un listado de vendedores y sus importes de ventas del año 2017 pero solo aquellos que vendieron menos de $ 17.000.- en dicho año.**

Lo que se ve en este punto, son dos condiciones de búsqueda uno para las fechas de la factura y la otra sobre un importe total. ¿En qué cláusula deberíamos escribir cada condición de búsqueda? ¿En el *WHERE* o en el *HAVING*?

Se sabe que: la cláusula *WHERE* se aplica a filas individuales, por lo que las expresiones que contiene deben ser calculables para filas individuales y la cláusula *HAVING* se aplica a grupos de filas, por lo que las expresiones que contengan deben ser calculables para un grupo de filas.

Por lo que la condición de búsqueda sobre la fecha de la factura como no está incluida en la agrupación del *GROUP BY* ni tampoco en una función de agregado deberá ir en el *WHERE* como también las condiciones para realizar la composición de tablas. La condición referida al importe total de facturación como es una función de agregado no podrá ir en el *WHERE* sino en el *HAVING*.

Entonces la consulta quedaría:

```
Select f.cod_vendedor Código, ape_vendedor Apellido, sum(pre_unitario*cantidad) Total
 from detalle_facturas d, facturas f,vendedores v
 where d.nro_factura = f.nro_factura and f.cod_vendedor=v.cod_vendedor
   and year(fecha) = 2017
 group by f.cod_vendedor, ape_vendedor
 having sum(pre_unitario*cantidad)<17000
```

Results | Messages

Código	Apellido	Total
1	Carrizo	16380.30
5	Monti	14732.00

Ya podemos entonces terminar de resolver el problema nro. 1.4

3. **Se quiere saber la fecha de la primera venta, la cantidad total vendida y el importe total vendido por vendedor para los casos en que el promedio de la cantidad vendida sea inferior o igual a 56.**

4. **Se necesita un listado que informe sobre el monto máximo, mínimo y total que gastó en esta librería cada cliente el año pasado, pero solo donde el importe total gastado por esos clientes esté entre 300 y 800.**

5. **Muestre la cantidad facturas diarias por vendedor; para los casos en que esa cantidad sea 2 o más.**

6. **Desde la administración se solicita un reporte que muestre el precio promedio, el importe total y el promedio del importe vendido por artículo que no comiencen con "c", que su cantidad total vendida sea 100 o más o que ese importe total vendido sea superior a 700.**

7. **Muestre en un listado la cantidad total de artículos vendidos, el importe total y la fecha de la primer y última venta por cada cliente, para lo números de factura que no sean los siguientes: 2, 12, 20, 17, 30 y que el promedio de la cantidad vendida oscile entre 2 y 6.**

8. **Emitir un listado que muestre la cantidad total de artículos vendidos, el importe total vendido y el promedio del importe vendido por vendedor y**

por cliente; para los casos en que el importe total vendido esté entre 200 y 600 y para códigos de cliente que oscilen entre 1 y 5.

9. **¿Cuáles son los vendedores cuyo promedio de facturación el mes pasado supera los $ 800?**

10. **¿Cuánto le vendió cada vendedor a cada cliente el año pasado siempre que la cantidad de facturas emitidas (por cada vendedor a cada cliente) sea menor a 5?**

BIBLIOGRAFÍA

Gorman K., Hirt A., Noderer D., Rowland-Jones J., Sirpal A., Ryan D. & Woody B (2019) Introducing Microsoft SQL Server 2019. Reliability, scalability, and security both on premises and in the cloud. Packt Publishing Ltd. Birmingham UK

Microsoft (2021) SQL Server technical documentation. Disponible en: https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15

Opel, A. & Sheldon, R. (2010). Fundamentos de SQL. Madrid. Editorial Mc Graw Hill

Varga S., Cherry D., D'Antoni J. (2016). Introducing Microsoft SQL Server 2016 Mission-Critical Applications, Deeper Insights, Hyperscale Cloud. Washington. Microsoft Press