BASES DE DATOS

CONSULTAS COMPLEJAS

NOTA: Se utilizarán las tablas del usuario USERMB creadas en las prácticas anteriores de Consultas Simples.

CONSULTAS MULTITABLA

1) Obtener los códigos de los productos que tienen existencias iguales a cero o que aparezcan en pedidos del año 2000.

```
SELECT ID_FAB,ID_PRODUCTO
FROM PRODUCTOS
WHERE EXISTENCIAS = 0
UNION ALL
SELECT FABRICANTE, PRODUCTO
FROM PEDIDOS
WHERE EXTRACT(YEAR FROM FECHA_PEDIDO) = 2000
ORDER BY ID_PRODUCTO;
```

2) Combinar los empleados con las oficinas para saber la ciudad de la oficina donde trabaja cada empleado.

Con un producto cartesiano:

```
SELECT VENDEDORES.*,CIUDAD
FROM VENDEDORES, OFICINAS
WHERE VENDEDORES.OFICINA_VEND = OFICINAS.OFICINA;
```

Si utilizamos INNER JOIN:

```
SELECT VENDEDORES.*, CIUDAD
FROM VENDEDORES INNER JOIN OFICINAS
ON VENDEDORES.OFICINA_VEND = OFICINAS.OFICINA;
```

3) Obtener una lista de los empleados con los datos de su oficina. Si un empleado no tiene oficina aparecerá con sus datos normales y los datos de su oficina a nulos.

```
SELECT *
FROM VENDEDORES LEFT JOIN OFICINAS
ON VENDEDORES.OFICINA_VEND = OFICINAS.OFICINA;
```

4) Obtener una lista de los empleados con los datos de su oficina. Si alguna oficina no está asignada a ningún empleado aparecerán sus datos con los datos del empleado a nulos.

```
SELECT *
FROM VENDEDORES RIGHT JOIN OFICINAS
ON VENDEDORES.OFICINA_VEND = OFICINAS.OFICINA
```

5) Listar los pedidos mostrando su número, importe, nombre del cliente, y el límite de crédito del cliente correspondiente (todos los pedidos tienen cliente y representante).

```
SELECT NUM_PEDIDO, IMPORTE, CLIENTES.NOMBRE AS CLIENTE, LIMITECREDITO
FROM PEDIDOS INNER JOIN CLIENTES
ON PEDIDOS.CLIENTE = CLIENTES.NUM_CLIENTE;
```

En este ejercicio no pueden haber pedidos sin cliente, y lo que nos interesa son los pedidos, luego tampoco tienen que aparecer los clientes que no tienen pedidos, por lo tanto utilizamos un INNER JOIN.

6) Listar los datos de cada uno de los empleados, la ciudad y región en donde trabaja.

```
SELECT VENDEDORES.*, CIUDAD, REGION
FROM VENDEDORES LEFT JOIN OFICINAS
ON VENDEDORES.OFICINA = OFICINAS.OFICINA;
```

Aquí hemos utilizado LEFT JOIN para que también salgan los empleados que no tienen oficina asignada.

7) Listar cuántos empleados están asignados a cada oficina, indicar el número de oficina y cuántos hay asignados.

Con esta solución:

```
SELECT OFICINA_VEND, COUNT(*) AS CUANTOS_EMPLEADOS
FROM VENDEDORES
GROUP BY OFICINA_VEND;
```

obtenemos el listado pedido pero no aparecen las oficinas que no tienen empleados asignados ya que sacamos la información de la tabla empleados y aparece una fila con valor nulo en oficina que contiene el número de empleados que no tienen oficina. Si quisiéramos listar incluso las que no tengan empleados :

```
SELECT OFICINAS.OFICINA, COUNT(NUM_EMPLE) AS CUANTOS_EMPLEADOS
FROM VENDEDORES RIGHT JOIN OFICINAS
ON VENDEDORES.OFICINA_VEND = OFICINAS.OFICINA
GROUP BY OFICINAS.OFICINA;
```

Utilizamos un RIGHT JOIN para que el origen de datos incluya también una fila por cada oficina que no tenga empleados.

En el GROUP BY y en la lista de selección hay que indicar el campo oficina de la tabla oficinas, si ponemos el de la tabla empleados, agrupará todas las oficinas que no tienen empleados en una fila (la columna empleados.oficina contiene valor nulo para esas filas).

Aquí no podemos utilizar COUNT(*) por que las oficinas sin empleados aparecerían con 1 en la columna cuantos_empleados ya que para esa oficina hay una fila.

8) Para cada empleado, obtener su número, nombre, e importe vendido por ese empleado a cada cliente indicando el número de cliente.

```
SELECT NUM_EMPLE, NOMBRE, CLIENTE, SUM(IMPORTE) AS TOTAL_VENDIDO
FROM VENDEDORES INNER JOIN PEDIDOS
ON PEDIDOS.VENDEDOR = VENDEDORES.NUM_EMPLE
GROUP BY NUM_EMPLE, NOMBRE, CLIENTE;
```

Necesitamos la tabla de pedidos para el importe vendido a un cliente, necesitamos la tabla empleados para el nombre del representante, la de clientes no la necesitamos ya que nos piden el número de cliente y este está en pedidos.

La agrupación básica que debemos realizar es por num_emple y después por CLIENTE, pero como aparece el nombre del empleado en la lista de selección, hay que incluirlo también en el GROUP BY.

Después de determinar la agrupación básica que nos hace falta, siempre que se incluye una columna adicional en el GROUP BY hay que comprobar que esa nueva columna no cambia la agrupación básica.

Por ejemplo no podríamos añadir al GROUP BY la columna FECHA_PEDIDO ya que se formarían más grupos.

Otra opción:

```
SELECT NUM_EMPLE, NOMBRE, CLIENTE, SUM(IMPORTE) AS TOTAL_VENDIDO
FROM VENDEDORES LEFT JOIN PEDIDOS
ON PEDIDOS.VENDEDOR = VENDEDORES. NUM_EMPLE
GROUP BY NUM_EMPLE, NOMBRE, CLIENTE;
```

Si queremos que salgan todos los empleados incluso los que no aparezcan en los pedidos habría que sustituir el INNER por un LEFT.

9) Se quiere obtener los datos de los clientes que tienen un representante y los datos de la oficina de este representante.

Combinamos empleados con oficinas para obtener los datos de la oficina de cada empleado, y luego añadimos los clientes de cada representante, así obtenemos los clientes que tienen un representante asignado y los datos de la oficina del representante asignado.

```
SELECT *
FROM CLIENTES INNER JOIN
 (VENDEDORES LEFT JOIN OFICINAS
ON VENDEDORES.OFICINA_VEND = OFICINAS.OFICINA)
ON CLIENTES.VENDEDOR_CLIENTE = VENDEDORES.NUM_EMPLE;
```

10) Listar de cada producto, su descripción, precio y cantidad total pedida, incluyendo sólo los productos cuya cantidad total pedida sea superior al 75% del stock; y ordenado por cantidad total pedida.

```
SELECT DESCRIPCION, PRECIO, SUM(CANTIDAD) AS TOTAL_PEDIDO
FROM PRODUCTOS INNER JOIN PEDIDOS
ON PEDIDOS.FABRICANTE = PRODUCTOS.ID_FAB AND
PEDIDOS.PRODUCTO = PRODUCTOS.ID_PRODUCTO
GROUP BY ID_FAB, ID_PRODUCTO, DESCRIPCION, PRECIO, EXISTENCIAS
HAVING SUM(CANTIDAD) > EXISTENCIAS * 0.75
ORDER BY 3;
```

La agrupación básica es por idfab e idproducto ya que son los dos campos que conjuntamente identifican un producto.

Como descripción y precio aparecen en la lista de selección y no modifican la agrupación básica los incluimos en el GROUP BY.

Como existencias aparece en el HAVING y no modifica la agrupación básica lo incluimos también en el GROUP BY.

Para calcular el 75% de las existencias multiplicamos existencias por 0,75; observar que en la sentencia SQL hay que utilizar el punto para indicar los decimales.

Para indicar la columna de ordenación no podemos utilizar el alias campo, utilizamos el número de orden de la columna dentro de la lista de selección. En este caso la suma de importes es la tercera columna.

11) Obtener el importe total de ventas de todos los empleados y el mayor objetivo de las oficinas asignadas a los empleados:

```
SELECT SUM(VENDEDORES.VENTAS), MAX(OBJETIVO)
FROM VENDEDORES LEFT JOIN OFICINAS
ON VENDEDORES.OFICINA_VEND=OFICINAS.OFICINA;
```

Combinamos empleados con oficinas por un LEFT JOIN para que aparezcan en el origen de datos todos los empleados incluso los que no tengan una oficina asignada,

así el origen de datos estará formado por una tabla con tantas filas como empleados haya en la tabla empleados, con los datos de cada empleado y de la oficina a la que está asignado. De esta tabla sacamos la suma del campo ventas (importe total de ventas de todos los empleados) y el objetivo máximo. Observar que el origen de datos no incluye las oficinas que no tienen empleados asignados, por lo que esas oficinas no entran a la hora de calcular el valor máximo del objetivo.

12) Listar las oficinas del este indicando para cada una de ellas su número, ciudad, números y nombres de sus empleados. Hacer una versión en la que aparecen sólo las que tienen empleados, y hacer otra en las que aparezcan las oficinas del este que no tienen empleados.

```
SELECT OFICINAS.OFICINA, CIUDAD, NUM_EMPLE, NOMBRE
FROM OFICINAS INNER JOIN VENDEDORES
ON OFICINAS.OFICINA = VENDEDORES.OFICINA_VEND
WHERE REGION = 'Este';
```

Como la columna de emparejamiento oficinas.oficina es clave principal en la tabla oficinas, es mejor utilizar el JOIN que un producto cartesiano. Emparejamos las dos tablas por el campo oficina. Las oficinas que no tengan empleados no salen (es un INNER).

Como queremos sólo las oficinas del este añadimos la cláusula WHERE con la condición.

Observar que en la lista de selección la columna oficina está cualificada (su nombre está precedido del nombre de la tabla), es necesario cualificarla porque en las dos tablas existe una columna llamada oficina y el sistema no sabría cuál de las dos escoger

```
SELECT OFICINAS.OFICINA, CIUDAD, NUM_EMPLE, NOMBRE
FROM OFICINAS LEFT JOIN VENDEDORES
ON OFICINAS.OFICINA = VENDEDORES.OFICINA_VEND
WHERE REGION = 'Este';
```

Si queremos que también aparezcan las oficinas que no tienen empleados cambiamos INNER por LEFT (queremos todas las oficinas y la tabla oficinas está a la izquierda de la palabra JOIN).

```
SELECT OFICINAS.OFICINA, CIUDAD, NUMEMP, NOMBRE
FROM VENDEDORES RIGHT JOIN OFICINAS
ON OFICINAS.OFICINA = VENDEDORES.OFICINA_VEND
WHERE REGION = 'Este';
```

Esta SELECT es equivalente a la anterior.

13) ¿Cuál es el importe total de los pedidos realizados por el empleado Tom Snyder?

```
SELECT SUM(IMPORTE) AS TOTAL_PEDIDOS_TOM
FROM VENDEDORES INNER JOIN PEDIDOS
ON VENDEDORES.NUM_EMPLE = PEDIDOS.VENDEDOR
WHERE NOMBRE = 'Tom Snyder';
```

El importe total lo tenemos que sacar de la tabla de pedidos, y además sólo nos interesan los de Tom Snyder. Como nos dan el nombre del representante en vez de su número y en el pedido sólo tenemos el número de representante tenemos que añadir a las líneas de cada pedido, los datos del representante correspondiente, por lo que el origen de datos debe ser el que aparece en la FROM.

14) Listar las oficinas con objetivo superior a 600.000 indicando para cada una de ellas el nombre de su director.

```
SELECT OFICINAS.*, NOMBRE AS DIRECTOR
FROM VENDEDORES RIGHT JOIN OFICINAS
ON VENDEDORES.NUMEMP = OFICINAS.DIRECTOR
WHERE OBJETIVO > 600000;
```

Nos interesan las oficinas con objetivo superior a 600.000 luego nos tenemos que asegurar que salgan todas incluso si no tienen director asignado por eso utilizamos RIGHT JOIN.

15) Listar los pedidos superiores a 25.000, incluyendo el nombre del empleado que tomó el pedido y el nombre del cliente que lo solicitó.

```
SELECT NUM_PEDIDO, IMPORTE, VENDEDORES.NOMBRE AS REPRESENTANTE, CLIENTES.EMPRESA AS CLIENTE
FROM (PEDIDOS INNER JOIN CLIENTES
ON PEDIDOS.CLIENTE = CLIENTES.NUM_CLIENTE)
INNER JOIN VENDEDORES
ON PEDIDOS.VENDEDOR = VENDEDORES.NUM_EMPLE
WHERE IMPORTE > 25000;
```

En este ejercicio no pueden haber pedidos sin representante ni cliente, y lo que nos interesa son los pedidos, luego tampoco tienen que aparecer los representantes que no tienen pedidos ni los clientes que no tienen pedidos, por lo tanto utilizamos un INNER JOIN.

Primero añadimos a cada línea de pedido los datos del cliente correspondiente (con el primer INNER) y a cada fila resultante añadimos los datos del representante correspondiente.

El representante que nos interesa es el que ha realizado el pedido y ese dato lo tenemos en el campo VENDEDOR de pedidos por eso la condición de emparejamiento es pedidos.VENDEDOR = empleados.rep.

Si hubiésemos querido el nombre del representante asignado al cliente, la condición hubiera sido clientes.vendedorcliente = empleados.numemp.

16) Hallar los empleados que realizaron su primer pedido el mismo día en que fueron contratados.

```
SELECT VENDEDORES.*
FROM VENDEDORES INNER JOIN PEDIDOS
ON PEDIDOS.VENDEDOR = VENDEDORES.NUM_EMPLE
WHERE FECHA_PEDIDO = FECHA_CONTRATO;
```

Los representantes que buscamos tienen un pedido con la misma fecha que la de su contrato, tenemos que añadir a los pedidos los datos del representante correspondiente para poder comparar los dos campos

17) Listar los empleados con una cuota superior a la de su director. Para cada empleado sacar sus datos y el número, nombre y cuota de su director.

```
SELECT VENDEDORES.*, DIRECTORES.NUM_EMPLE AS NUM_DIRECTOR,
DIRECTORES.NOMBRE AS NOMBRE_DIRECTOR, DIRECTORES.CUOTA AS
CUOTA_DIRECTOR
FROM VENDEDORES INNER JOIN VENDEDORES DIRECTORES
ON VENDEDORES.DIRECTOR = DIRECTORES.NUM_EMPLE
WHERE VENDEDORES.CUOTA > DIRECTORES.CUOTA;
```

En una misma línea necesito los datos del empleado y los datos de su jefe, luego tengo que combinar empleados con empleados. No interesan los empleados que no tienen jefe luego utilizo INNER. El alias de tabla es obligatorio ya que combino empleados con la misma.

18) Listar los códigos de los empleados que tienen una línea de pedido superior a 10.000 o que tengan una cuota inferior a 10.000.

```
SELECT NUM_EMPLE
FROM VENDEDORES LEFT JOIN PEDIDOS
ON PEDIDOS.VENDEDOR = VENDEDORES.NUM_EMPLE
WHERE IMPORTE > 10000 OR CUOTA < 10000;
```

Una posible solución es combinar pedidos con empleados para poder seleccionar las líneas de importe > 10000 o cuota < 10000. Hay que utilizar LEFT para que puedan aparecer empleados con cuota < 10000 que no tengan pedidos

```
SELECT VENDEDOR
FROM PEDIDOS
WHERE IMPORTE > 10000
UNION
SELECT NUM_EMPLE
```

```
FROM VENDEDORES
WHERE CUOTA < 10000;
```

Esta es otra solución, obtener por una parte los códigos de los empleados con una línea de pedido > 10000, por otra parte los códigos de los empleados con cuota < 10000 y finalmente unir las dos listas con una UNION.

SUBCONSULTAS

1) Listar los números de clientes y nombres de empresa atendidos por "Bill Adams".

```
SELECT NUM_CLIENTE, EMPRESA
FROM CLIENTES
WHERE VENDEDOR_CLIENTE = (SELECT NUM_EMPLE
FROM VENDEDORES
WHERE NOMBRE ='Bill Adams');
```

2) Listar los nombres de los vendedores que trabajan en oficinas cuyas ventas superan su objetivo.

```
SELECT NUM_EMPLE, NOMBRE
FROM VENDEDORES
WHERE OFICINA_VEND IN (SELECT OFICINA
FROM OFICINAS
WHERE VENTAS > OBJETIVO);
```

3) Listar las oficinas (número y ciudad) en las que haya algún vendedor cuya cuota represente más del 55 por 100 del objetivo de la oficina.

```
SELECT OFICINA, CIUDAD
FROM OFICINAS
WHERE OFICINA IN (SELECT OFICINA_VEND
FROM VENDEDORES
WHERE CUOTA> (0.55 * OFICINAS.OBJETIVO));
```

4) Lista los nombres y edades de los vendedores que tienen cuotas por encima del promedio (de las cuotas).

```
SELECT NOMBRE, EDAD
FROM VENDEDORES
WHERE CUOTA > (SELECT AVG(CUOTA)FROM VENDEDORES);
```

5) Lista los clientes cuyos vendedores están asignados a oficinas de la región de ventas "Este".

```
SELECT NUM_CLIENTE, EMPRESA
FROM CLIENTES
WHERE VENDEDOR_CLIENTE IN (SELECT NUM_EMPLE
FROM VENDEDORES
WHERE OFICINA_VEND IN (SELECT OFICINA
FROM OFICINAS
WHERE REGION = 'Este'));
```

O también:

```
SELECT NUM_CLIENTE, EMPRESA
FROM CLIENTES
WHERE VENDEDOR_CLIENTE IN (SELECT NUM_EMPLE
FROM VENDEDORES, OFICINAS
WHERE OFICINA_VEND=OFICINA AND REGION='Este');
```

O también:

```
SELECT NUM_CLIENTE, EMPRESA
FROM CLIENTES, VENDEDORES, OFICINAS
WHERE VENDEDOR_CLIENTE=NUM_EMPLE AND OFICINA_VEND=OFICINA AND
REGION='Este';
```

6) Lista los vendedores que tienen más de 40 años y que dirigen a algún vendedor cuyas ventas están por encima de la cuota.

```
SELECT NUM_EMPLE, NOMBRE
FROM VENDEDORES
WHERE EDAD>40 AND
NUM_EMPLE IN (SELECT DIRECTOR
FROM VENDEDORES
WHERE VENTAS > CUOTA);
```

Otra opción sería con estrategia de la tabla duplicada imaginaria.

```
SELECT R.NUM_EMPLE, R.NOMBRE
FROM VENDEDORES R, VENDEDORES E
WHERE R.EDAD>40 AND R.NUM_EMPLE=E.DIRECTOR AND E.VENTAS > E.CUOTA;
```

7) Listar los vendedores que no trabajan en oficinas dirigidas por el empleado 108.

Solución 1

Obtenemos los empleados tales que no exista una oficina igual a la suya que además esté dirigida por el empleado 108, con esta solución sí aparecen los empleados que no tienen oficina.

```
SELECT NUM_EMPLE, NOMBRE, OFICINA_VEND
FROM VENDEDORES
WHERE NOT EXISTS (SELECT * FROM OFICINAS WHERE OFICINA_VEND= OFICINA AND DIRECTOR = 108);
```

Con la subconsulta siguiente obtenemos la lista de las oficinas dirigidas por el empleado 108. Al final se obtienen los empleados cuya oficina no esté en esa lista, es decir salen los empleados asignados a una oficina no dirigida por el 108. Pero no salen los empleados que no tienen oficina asignada ya que su campo oficina es nulo por lo que el resultado de la comparación es nulo, no es verdadero y no se seleccionan.

```
SELECT NUM_EMPLE, NOMBRE, OFICINA_VEND
FROM VENDEDORES
WHERE OFICINA_VEND NOT IN (SELECT OFICINA FROM OFICINAS WHERE DIRECTOR = 108);
```

Solución 2

Con esta subconsulta obtenemos la lista de las oficinas dirigidas por el empleado 108. Al final se obtienen los empleados cuya oficina no esté en esa lista. Pero no salen los empleados que no tienen oficina asignada ya que su campo oficina es nulo por lo que el resultado de la comparación es nulo, no es verdadero y no se seleccionan.

```
SELECT NUM_EMPLE, NOMBRE, OFICINA_VEND
FROM VENDEDORES
WHERE (OFICINA_VEND NOT IN (SELECT OFICINA FROM OFICINAS WHERE DIRECTOR = 108) ) OR ( OFICINA_VEND IS NULL);
```

Con esta solución tenemos el mismo problema que con NOT IN , cuando la oficina del empleado es nula todos los resultados de las comparaciones individuales son nulos por los que el test ALL da nulo y no se seleccionan los empleados con oficina nula.

SELECT NUM_EMPLE, NOMBRE, OFICINA_VEND
FROM VENDEDORES
WHERE OFICINA_VEND <> ALL (SELECT OFICINA FROM OFICINAS WHERE DIRECTOR = 108);

8) Listar los productos (idfab, idproducto y descripción) para los cuales no se ha recibido ningún pedido de 25000 o más.

En este caso es más cómodo utilizar NOT EXISTS ya que hay que preguntar por el idfab e idproducto a la vez.
SELECT ID_FAB, ID_PRODUCTO, DESCRIPCION
FROM PRODUCTOS
*WHERE NOT EXISTS (SELECT * FROM PEDIDOS WHERE FABRICANTE = ID_FAB AND PRODUCTO = ID_PRODUCTO AND IMPORTE >= 25000);*

9) Listar los clientes asignados a Bill Adams que no han remitido un pedido superior a 3000.

Como utilizamos NOT IN debemos asegurarnos de que la subconsulta no devuelva nulos. En este caso CLIENTE es un campo de pedidos que admite nulos por lo que tenemos que añadir en el WHERE de la subconsulta AND CLIENTE IS NOT NULL.
SELECT NUM_CLIENTE, EMPRESA
FROM CLIENTES
WHERE VENDEDOR_CLIENTE IN (SELECT NUM_EMPLE FROM VENDEDORES WHERE NOMBRE = 'Bill Adams')
AND NUM_CLIENTE NOT IN (SELECT CLIENTE FROM PEDIDOS WHERE IMPORTE > 3000 AND CLIENTE IS NOT NULL);

10) Listar las oficinas en donde todos los vendedores tienen ventas que superan al 50% del objetivo de la oficina.

*SELECT **
FROM OFICINAS
*WHERE (OBJETIVO * 0.5) <= ALL (SELECT VENTAS FROM VENDEDORES WHERE OFICINA_VEND = OFICINA AND VENTAS IS NOT NULL);*

Si un empleado no tiene ventas queremos que no sea tomado en cuenta, por eso añadimos AND ventas IS NOT NULL. Además esta solución no vale porque salen las oficinas que no tienen empleados.
Hay que añadir una condición para que se consideren sólo las oficinas con empleados como muestra la solución siguiente:

*SELECT **
FROM OFICINAS
*WHERE ((OBJETIVO * 0.5) <= ALL (SELECT VENTAS FROM VENDEDORES WHERE OFICINA_VEND = OFICINA AND VENTAS IS NOT NULL))*
*AND (EXISTS (SELECT * FROM VENDEDORES WHERE OFICINA_VEND = OFICINA));*

Esta es otra posible solución, calculamos la menor venta de los empleados de la oficina y si esta es mayor que el 50% del objetivo de la oficina quiere decir que todos los empleados de esa oficina tienen ventas iguales o superiores. Si la oficina no tiene empleados, la subconsulta no devuelve ninguna fila y como estamos utilizando una comparación simple el resultado es nulo, luego no salen las oficinas que no tienen empleados. Esta solución es mucho más corta y elegante.

```
SELECT *
FROM OFICINAS
WHERE (OBJETIVO * .5) <= (SELECT MIN(VENTAS) FROM VENDEDORES WHERE OFICINA_VEND = OFICINA);
```

11) Listar las oficinas que tengan un objetivo mayor que la suma de las cuotas de sus vendedores.

```
SELECT *
FROM OFICINAS
WHERE OBJETIVO > (SELECT SUM(CUOTA) FROM VENDEDORES WHERE OFICINA_VEND = OFICINA);
```

12) Listar el código y nombre de los empleados ordenándolos por fecha de contrato, mostrando únicamente los dos primeros (serán los dos más antiguos).

En general sería una instrucción sencilla como esta donde TOP 2 selecciona las 2 filas de arriba.

```
SELECT TOP 2 NUM_EMPLE, NOMBRE
FROM VENDEDORES
ORDER BY FECHA_CONTRATO;
```

Pero en Oracle es un poco más complicado. Primero generamos el listado ordenado:

```
SELECT NUM_EMPLE, NOMBRE, FECHA_CONTRATO
FROM VENDEDORES ORDER BY FECHA_CONTRATO
```

Y este listado se lo damos en el FROM a la sentencia que extrae las dos primeras filas. En esta sentencia utilizamos ROWNUM para ver el número de fila y lo ponemos como condición con WHERE:

```
WHERE ROWNUM<=2
```

La sentencia final completa es:

```
SELECT NUM_EMPLE, NOMBRE, FECHA_CONTRATO
FROM (SELECT NUM_EMPLE, NOMBRE, FECHA_CONTRATO FROM VENDEDORES ORDER BY FECHA_CONTRATO)
WHERE ROWNUM<=2;
```

13) Listar el código y nombre de los empleados mostrando únicamente los tres últimos contratados.

```
SELECT NUM_EMPLE, NOMBRE, FECHA_CONTRATO
FROM (SELECT NUM_EMPLE, NOMBRE, FECHA_CONTRATO FROM VENDEDORES ORDER BY FECHA_CONTRATO DESC)
WHERE ROWNUM<=3;
```

14) Mostrar al empleado que más ventas ha realizado

```
SELECT NUM_EMPLE, NOMBRE, VENTAS
FROM (SELECT NUM_EMPLE, NOMBRE, VENTAS FROM VENDEDORES ORDER BY VENTAS DESC)
WHERE ROWNUM=1;
```

15) Listar las cuatro líneas de pedido más caras (las de mayor importe).

```
SELECT *
FROM (SELECT * FROM PEDIDOS ORDER BY IMPORTE DESC)
WHERE ROWNUM<=4;
```

Para obtener las más caras tenemos que ordenar por importe y en orden descendente para que aparezcan las más caras primero. De esta selección mostramos las cuatro primeras filas.

16) Obtener las mismas columnas que en el ejercicio anterior pero sacando únicamente las 5 líneas de pedido de menor precio unitario (importe/cantidad).

```
SELECT *
FROM (SELECT NUM_PEDIDO, FECHA_PEDIDO, CLIENTE, VENDEDOR, FABRICANTE, PRODUCTO, CANTIDAD, IMPORTE, IMPORTE/CANTIDAD AS PRECIOUNITARIO FROM PEDIDOS ORDER BY PRECIOUNITARIO)
WHERE ROWNUM<=5;
```

CONSULTAS RESUMEN MULTITABLA

1) Saber cuántas oficinas tienen empleados con ventas superiores a su cuota, no queremos saber cuáles sino cuántas hay

Vamos a verlo con dos consultas separadas

CONSULTA: DISTINTAS_OFICINAS

```
SELECT DISTINCT OFICINA_VEND
FROM VENDEDORES
WHERE VENTAS > CUOTA
```

CONSULTA: CUENTA

```
SELECT COUNT(*) AS CUANTAS_OFICINAS
FROM DISTINTAS_OFICINAS
```

Si contamos las oficinas directamente de la tabla empleados nos salen 9 oficinas ya que la función COUNT(nb columna) cuenta los valores no nulos pero los valores repetidos los cuenta tantas veces como se repiten, como tenemos oficinas que se repiten en la columna oficina de la tabla oficinas, esas oficinas son contadas varias veces, hay que contar los valores distintos.

En otros SQL la función COUNT puede llevar delante del nombre de la columna la cláusula DISTINCT que indica que sólo se tienen que tener en cuenta valores distintos (no cuenta los repetidos), por ejemplo COUNT(DISTINCT oficina), es una opción muy útil que desgraciadamente no incluye el SQL de Microsoft JET. Para solucionar el problema se resuelve con dos consultas, una con la cual nos quedamos con los valores distintos (en la solución la consulta se llama distintas_oficinas), y la otra que nos cuenta esos valores.

La consulta queda:

```
SELECT COUNT(*) AS CUANTAS_OFICINAS
FROM (SELECT DISTINCT OFICINA_VEND
FROM VENDEDORES
WHERE VENTAS > CUOTA);
```

2) Listar los pedidos superiores a 25.000, mostrando el nombre del cliente que remitió el pedido, el nombre del vendedor asignado a ese cliente, el número de pedido y el importe.

```
SELECT EMPRESA, NOMBRE, NUM_PEDIDO, IMPORTE
FROM CLIENTES C, PEDIDOS P, VENDEDORES R
WHERE IMPORTE>25000 AND P.CLIENTE=C.NUM_CLE
AND C.VENDEDOR_CLIENTE = R.NUM_EMPLE;
```

3) Para cada oficina con dos o más personas, mostrar el nombre (la ciudad) de la oficina, la cuota total y las ventas totales de todos los vendedores que trabajan en esa oficina.

```
SELECT CIUDAD, SUM(CUOTA), SUM(VENDEDORES.VENTAS)
FROM OFICINAS, VENDEDORES
WHERE OFICINA = OFICINA_VEND
GROUP BY CIUDAD -- hay una oficina por ciudad
HAVING COUNT(*) >= 2;
```

SQL gestiona esta consulta del modo siguiente:

a) Combina (operación "join") las tablas OFICINAS y VENDEDORES para obtener la ciudad en donde trabaja cada vendedor.

b) Agrupa las filas resultantes por oficina (columna CIUDAD).

c) Elimina los grupos con dos o menos filas, puesto que representan oficinas que no satisfacen el criterio de la cláusula HAVING.

d) Calcula la cuota total y las ventas totales para cada grupo, es decir, por cada oficina.

4) Mostrar la descripción, el precio, las existencias y la cantidad total de los pedidos de cada producto para los cuales la cantidad total pedida es superior al 75 por 100 de las existencias.

```
SELECT DESCRIPCION, PRECIO, EXISTENCIAS, SUM(CANTIDAD)
FROM PRODUCTOS, PEDIDOS
WHERE FABRICANTE = ID_FAB AND PRODUCTO = ID_PRODUCTO
GROUP BY ID_FAB, ID_PRODUCTO, DESCRIPCION, PRECIO, EXISTENCIAS
HAVING SUM(CANTIDAD) > (0.75 * EXISTENCIAS);
```

Para procesar esta consulta, SQL efectúa conceptualmente los siguientes pasos:

a) Combina las Tablas PRODUCTOS y PEDIDOS para obtener la descripción, precio y existencias de cada producto pedido.

b) Agrupa las filas resultantes por fabricante y dentro de cada fabricante, por producto.

c) Elimina los grupos en donde la cantidad pedida es menor al 75% de las existencias.

d) Calcula la cantidad total pedida para cada grupo.

e) Genera una fila resumen de resultados por cada grupo.

f) Ordena los resultados para que los productos con el mayor valor de existencias aparezcan en primer lugar.