Ejercicios. Realización de consultas SQL Curso 2023/2024
1.3 Gestión de ventas

Antonio Atienza Cano

CONSULTAS BASICAS
---

1. Devuelve un listado con todos los pedidos que se han realizado. Los pedidos deben estar ordenados por
la fecha de realización, mostrando en primer lugar los pedidos más recientes.

```
SELECT *
FROM pedido
ORDER BY Fecha DESC;
```

2. Devuelve todos los datos de los dos pedidos de mayor valor

```
SELECT *
FROM pedido
ORDER BY total DESC
LIMIT 2;
```

3. Devuelve un listado con los identificadores de los clientes que han realizado algún pedido. Tenga en cuen-
ta que no debe mostrar identificadores que estén repetidos

```
SELECT DISTINCT id_cliente
FROM pedido;
```

4. Devuelve un listado de todos los pedidos que se realizaron durante el año 2017, cuya cantidad total sea
superior a 500€

```
SELECT *
FROM pedido
WHERE YEAR(fecha) = 2017 AND total > 500;
```

5. Devuelve un listado con el nombre y los apellidos de los comerciales que tienen una comisión entre 0.05
y 0.11.

```
SELECT nombre, apellido1, apellido2
FROM comercial
WHERE comisión BETWEEN 0.05 AND 0.11;
```

6. Devuelve el valor de la comisión de mayor valor que existe en la tabla comercial.

```
SELECT MAX(comisión) AS comisión_mayor
FROM comercial;
```

7. Devuelve el identificador, nombre y primer apellido de aquellos clientes cuyo segundo apellido no es
NULL. El listado deberá estar ordenado alfabéticamente por apellidos y nombre.

```
SELECT id, nombre, apellido1
FROM cliente
WHERE apellido2 IS NOT NULL
ORDER BY apellido1, nombre;
```

8. Devuelve un listado de los nombres de los clientes que empiezan por A y terminan por n y también los
nombres que empiezan por P. El listado deberá estar ordenado alfabéticamente.

```
SELECT nombre
FROM cliente
WHERE (nombre LIKE 'A%n' OR nombre LIKE 'P%')
ORDER BY nombre;
```

9. Devuelve un listado de los nombres de los clientes que no empiezan por A. El listado deberá estar orde-
nado alfabéticamente.

```
SELECT nombre
FROM cliente
WHERE nombre NOT LIKE 'A%'
ORDER BY nombre;
```

10. Devuelve un listado con los nombres de los comerciales que terminan por el o o. Tenga en cuenta que
se deberán eliminar los nombres repetidos.

```
SELECT DISTINCT nombre
FROM comercial
WHERE nombre LIKE '%o' OR nombre LIKE '%O';
```

CONSULTAS MULTITABLA

---

1. Devuelve un listado con el identificador, nombre y los apellidos de todos los clientes que han realizado
algún pedido. El listado debe estar ordenado alfabéticamente y se deben eliminar los elementos repeti-
dos

```
SELECT DISTINCT c.id, c.nombre, c.apellido1, c.apellido2
FROM cliente c
JOIN pedido p ON c.id = p.id_cliente
ORDER BY c.apellido1, c.nombre;
```

2. Devuelve un listado que muestre todos los pedidos que ha realizado cada cliente. El resultado debe mos-
trar todos los datos de los pedidos y del cliente. El listado debe mostrar los datos de los clientes ordena-
dos alfabéticamente

```
SELECT c.id AS id_cliente, c.nombre AS nombre_cliente, c.apellido1 AS apellido1_cliente,
c.apellido2 AS apellido2_cliente,
    p.id AS id_pedido, p.total, p.fecha, p.id_comercial
FROM cliente c
JOIN pedido p ON c.id = p.id_cliente
ORDER BY c.apellido1, c.nombre, p.fecha;
```

3. Devuelve un listado que muestre todos los pedidos en los que ha participado un comercial. El resultado
debe mostrar todos los datos de los pedidos y de los comerciales. El listado debe mostrar los datos de los
comerciales ordenados alfabéticamente.

```
SELECT c.*, p.*
FROM comercial c
JOIN pedido p ON c.id = p.id_comercial
ORDER BY c.apellido1, c.nombre, p.fecha;
```

4. Devuelve un listado que muestre todos los clientes, con todos los pedidos que han realizado y con los
datos de los comerciales asociados a cada pedido.

```
SELECT c.id AS id_cliente, c.nombre AS nombre_cliente, c.apellido1 AS apellido1_cliente, c.apellido2 AS apellido2_cliente,
    p.id AS id_pedido, p.total, p.fecha, p.id_comercial,
    co.nombre AS nombre_comercial, co.apellido1 AS apellido1_comercial, co.apellido2 AS apellido2_comercial
FROM cliente c
JOIN pedido p ON c.id = p.id_cliente
JOIN comercial co ON p.id_comercial = co.id
ORDER BY c.apellido1, c.nombre, p.fecha;
```

5. Devuelve un listado de todos los clientes que realizaron un pedido durante el año 2017, cuya cantidad
esté entre 300 € y 1000 €

```
SELECT DISTINCT c.id, c.nombre, c.apellido1, c.apellido2
FROM cliente c
JOIN pedido p ON c.id = p.id_cliente
WHERE YEAR(p.fecha) = 2017 AND p.total BETWEEN 300 AND 1000;
```

6. Devuelve el nombre y los apellidos de todos los comerciales que ha participado en algún pedido realizado
por María Santana Moreno.

```
SELECT DISTINCT co.nombre, co.apellido1, co.apellido2
FROM cliente cl
JOIN pedido pe ON cl.id = pe.id_cliente
JOIN comercial co ON pe.id_comercial = co.id
WHERE cl.nombre = 'María' AND cl.apellido1 = 'Santana' AND cl.apellido2 = 'Moreno';
```

7. Devuelve el nombre de todos los clientes que han realizado algún pedido con el comercial Daniel Sá
ez Vega.

```
SELECT DISTINCT cl.nombre
FROM cliente cl
JOIN pedido pe ON cl.id = pe.id_cliente
JOIN comercial co ON pe.id_comercial = co.id
WHERE co.nombre = 'Daniel' AND co.apellido1 = 'Sáez' AND co.apellido2 = 'Vega';
```

CONSULTAS MULTITABLA (COMPOSICIÓN EXTERNA)
---

1. Devuelve un listado con todos los clientes junto con los datos de los pedidos que han realizado. Este lista-
do también debe incluir los clientes que no han realizado ningún pedido. El listado debe estar ordenado
alfabéticamente por el primer apellido, segundo apellido y nombre de los clientes

```
SELECT c.id, c.nombre, c.apellido1, c.apellido2, p.id AS id_pedido, p.total, p.fecha,
p.id_comercial
FROM cliente c
LEFT JOIN pedido p ON c.id = p.id_cliente
ORDER BY c.apellido1, c.apellido2, c.nombre;
```

2. Devuelve un listado con todos los comerciales junto con los datos de los pedidos que han realizado. Este
listado también debe incluir los comerciales que no han realizado ningún pedido. El listado debe estar
ordenado alfabéticamente por el primer apellido, segundo apellido y nombre de los comerciales.

```
SELECT co.id, co.nombre, co.apellido1, co.apellido2, p.id AS id_pedido, p.total, p.fecha,
p.id_cliente
FROM comercial co
LEFT JOIN pedido p ON co.id = p.id_comercial
ORDER BY co.apellido1, co.apellido2, co.nombre;
```

3. Devuelve un listado que solamente muestre los clientes que no han realizado ningún pedido.

```
SELECT c.id, c.nombre, c.apellido1, c.apellido2
FROM cliente c
LEFT JOIN pedido p ON c.id = p.id_cliente
WHERE p.id IS NULL;
```

4. Devuelve un listado que solamente muestre los comerciales que no han realizado ningún pedido.

```
SELECT co.id, co.nombre, co.apellido1, co.apellido2
FROM comercial co
LEFT JOIN pedido p ON co.id = p.id_comercial
WHERE p.id IS NULL;
```

5. Devuelve un listado con los clientes que no han realizado ningún pedido y de los comerciales que no han
participado en ningún pedido. Ordene el listado alfabéticamente por los apellidos y el nombre. En en
listado deberá diferenciar de algún modo los clientes y los comerciales

```
SELECT 'Cliente' AS tipo, c.id, c.nombre, c.apellido1, c.apellido2
FROM cliente c
LEFT JOIN pedido p ON c.id = p.id_cliente
WHERE p.id IS NULL
UNION ALL
SELECT 'Comercial' AS tipo, co.id, co.nombre, co.apellido1, co.apellido2
FROM comercial co
LEFT JOIN pedido p ON co.id = p.id_comercial
WHERE p.id IS NULL
ORDER BY apellido1, apellido2, nombre;
```

6. ¿Se podrían realizar las consultas anteriores con NATURAL LEFT JOIN o NATURAL RIGHT JOIN?
Justifique su respuesta.

No, las consultas anteriores no se pueden realizar utilizando NATURAL LEFT JOIN o NATURAL RIGHT JOIN debido a la forma en que estas cláusulas funcionan y a la estructura de las consultas requeridas.

El NATURAL JOIN combina columnas de dos tablas basándose en que tienen nombres de columna idénticos y elimina duplicados. En el caso de las consultas anteriores,
la combinación se realiza no solo en base a columnas con nombres idénticos,
sino también a través de condiciones específicas como la comparación de identificadores entre tablas. Además, la necesidad de diferenciar entre clientes y comerciales en el resultado no se puede lograr directamente con NATURAL JOIN.

Por lo tanto, la utilización de NATURAL LEFT JOIN o NATURAL RIGHT JOIN no sería adecuada para estas consultas específicas.
En su lugar, es necesario utilizar LEFT JOIN y UNION ALL para obtener el resultado deseado, como se mostró en las respuestas anteriores.

CONSULTAS RESUMEN
---

1. Calcula la cantidad total que suman todos los pedidos que aparecen en la tabla pedido.

```
SELECT SUM(total) AS total_pedidos
FROM pedido;
```

2. Calcula la cantidad media de todos los pedidos que aparecen en la tabla pedido.

```
SELECT AVG(total) AS media_pedidos
FROM pedido;
```

3. Calcula el número total de comerciales distintos que aparecen en la tabla pedido

```
SELECT COUNT(DISTINCT id_comercial) AS total_comerciales
FROM pedido;
```

4. Calcula el número total de clientes que aparecen en la tabla cliente.

```
SELECT COUNT(*) AS total_clientes
FROM cliente;
```

5. Calcula cuál es la mayor cantidad que aparece en la tabla pedido.

```
SELECT MAX(total) AS mayor_cantidad
FROM pedido;
```

6. Calcula cuál es la menor cantidad que aparece en la tabla pedido.

```
SELECT MIN(total) AS menor_cantidad
FROM pedido;
```

7. Calcula cuál es el valor máximo de categoría para cada una de las ciudades que aparece en la tabla
cliente.

```
SELECT ciudad, MAX(categoría) AS max_categoria
FROM cliente
GROUP BY ciudad;
```

8. Calcula cuál es el máximo valor de los pedidos realizados durante el mismo día para cada uno de los clien-
tes. Es decir, el mismo cliente puede haber realizado varios pedidos de diferentes cantidades el mismo
día. Se pide que se calcule cuál es el pedido de máximo valor para cada uno de los días en los que un
cliente ha realizado un pedido. Muestra el identificador del cliente, nombre, apellidos, la fecha y el valor
de la cantidad

```
SELECT p.id_cliente, c.nombre, c.apellido1, c.apellido2, p.fecha, MAX(p.total) AS
max_valor_pedido
FROM pedido p
JOIN cliente c ON p.id_cliente = c.id
GROUP BY p.id_cliente, c.nombre, c.apellido1, c.apellido2, p.fecha;
```

9. Calcula cuál es el máximo valor de los pedidos realizados durante el mismo día para cada uno de los
clientes, teniendo en cuenta que sólo queremos mostrar aquellos pedidos que superen la cantidad de
2000 €

```
SELECT p.id_cliente, c.nombre, c.apellido1, c.apellido2, p.fecha, MAX(p.total) AS
max_valor_pedido
FROM pedido p
JOIN cliente c ON p.id_cliente = c.id
GROUP BY p.id_cliente, c.nombre, c.apellido1, c.apellido2, p.fecha
HAVING MAX(p.total) > 2000;
```

10. Calcula el máximo valor de los pedidos realizados para cada uno de los comerciales durante la fecha
2016-08-17. Muestra el identificador del comercial, nombre, apellidos y total.

```
SELECT p.id_comercial, co.nombre, co.apellido1, co.apellido2, MAX(p.total) AS
max_valor_pedido
FROM pedido p
JOIN comercial co ON p.id_comercial = co.id
WHERE p.fecha = '2016-08-17'
GROUP BY p.id_comercial, co.nombre, co.apellido1, co.apellido2;
```

11. Devuelve un listado con el identificador de cliente, nombre y apellidos y el número total de pedidos que
ha realizado cada uno de clientes. Tenga en cuenta que pueden existir clientes que no han realizado nin-
gún pedido. Estos clientes también deben aparecer en el listado indicando que el número de pedidos
realizados es 0

```
SELECT c.id, c.nombre, c.apellido1, c.apellido2, COUNT(p.id) AS total_pedidos
FROM cliente c
LEFT JOIN pedido p ON c.id = p.id_cliente
GROUP BY c.id, c.nombre, c.apellido1, c.apellido2;
```

12. Devuelve un listado con el identificador de cliente, nombre y apellidos y el número total de pedidos que
ha realizado cada uno de clientes durante el año 2017.

```
SELECT c.id, c.nombre, c.apellido1, c.apellido2, COUNT(p.id) AS total_pedidos
FROM cliente c
LEFT JOIN pedido p ON c.id = p.id_cliente
WHERE YEAR(p.fecha) = 2017
GROUP BY c.id, c.nombre, c.apellido1, c.apellido2;
```

13. Devuelve un listado que muestre el identificador de cliente, nombre, primer apellido y el valor de la máxi-
ma cantidad del pedido realizado por cada uno de los clientes. El resultado debe mostrar aquellos clien-
tes que no han realizado ningún pedido indicando que la máxima cantidad de sus pedidos realizados es 0.
Puede hacer uso de la función IFNULL

```
SELECT c.id, c.nombre, c.apellido1, IFNULL(MAX(p.total), 0) AS maxima_cantidad_pedido
FROM cliente c
LEFT JOIN pedido p ON c.id = p.id_cliente
GROUP BY c.id, c.nombre, c.apellido1;
```

14. Devuelve cuál ha sido el pedido de máximo valor que se ha realizado cada año.

```
SELECT p.*
FROM pedido p
JOIN (
    SELECT YEAR(fecha) AS año, MAX(total) AS max_valor_pedido
    FROM pedido
    GROUP BY año
) AS max_pedidos
ON YEAR(p.fecha) = max_pedidos.año AND p.total = max_pedidos.max_valor_pedido;
```

15. Devuelve el número total de pedidos que se han realizado cada año

```
SELECT YEAR(fecha) AS año, COUNT(*) AS total_pedidos
FROM pedido
GROUP BY año;
```

CONSULTAS SUBCONSULTAS
---

1. Devuelve un listado con todos los pedidos que ha realizado Adela Salas Díaz. (Sin utilizar INNER
JOIN).

```
SELECT *
FROM pedido
WHERE id_cliente = (
   SELECT id
   FROM cliente
   WHERE nombre = 'Adela' AND apellido1 = 'Salas' AND apellido2 = 'Díaz'
);
```

2. Devuelve el número de pedidos en los que ha participado el comercial Daniel Sáez Vega. (Sin utilizar INNER JOIN)

```
SELECT COUNT(*)
FROM pedido
WHERE id_comercial = (
   SELECT id
   FROM comercial
   WHERE nombre = 'Daniel' AND apellido1 = 'Sáez' AND apellido2 = 'Vega'
);
```

3. Devuelve los datos del cliente que realizó el pedido más caro en el año 2019. (Sin utilizar INNER JOIN)

```
SELECT *
FROM cliente
WHERE id = (
   SELECT id_cliente
   FROM pedido
   WHERE YEAR(fecha) = 2019
   ORDER BY total DESC
   LIMIT 1
);
```

4. Devuelve la fecha y la cantidad del pedido de menor valor realizado por el cliente Pepe Ruiz Santana

```
SELECT fecha, total
FROM pedido
WHERE id_cliente = (
   SELECT id
   FROM cliente
```

```
   WHERE nombre = 'Pepe' AND apellido1 = 'Ruiz' AND apellido2 = 'Santana'
)
ORDER BY total ASC
LIMIT 1;
```

5. Devuelve un listado con los datos de los clientes y los pedidos, de todos los clientes que han realizado un
pedido durante el año 2017 con un valor mayor o igual al valor medio de los pedidos realizados durante
ese mismo año.

```
SELECT c.id, c.nombre, c.apellido1, c.apellido2, p.id AS id_pedido, p.total
FROM cliente c
JOIN pedido p
ON c.id = p.id_cliente
WHERE YEAR(p.fecha) = 2017
AND p.total >= (
   SELECT AVG(total)
   FROM pedido
   WHERE YEAR(fecha) = 2017
);
```

//

1.3.7.2 Subconsultas con ALL y ANY
6. Devuelve el pedido más caro que existe en la tabla pedido sin hacer uso de MAX, ORDER BY ni LIMIT.

```
SELECT *
FROM pedido
WHERE total >= ALL (SELECT total FROM pedido WHERE total IS NOT NULL);
```

7. Devuelve un listado de los clientes que no han realizado ningún pedido. (Utilizando ANY o ALL).

```
SELECT *
FROM cliente
WHERE id <> ALL  SELECT (id_cliente FROM pedido);
```

8. Devuelve un listado de los comerciales que no han realizado ningún pedido. (Utilizando ANY o ALL).

```
SELECT *
FROM comercial
WHERE id NOT IN (SELECT id_comercial FROM pedido WHERE id_comercial IS NOT
NULL);
```

1.3.7.3 Subconsultas con IN y NOT IN
9. Devuelve un listado de los clientes que no han realizado ningún pedido. (Utilizando IN o NOT IN).

```
SELECT *
FROM cliente
WHERE id NOT IN (SELECT id_cliente FROM pedido);
```

10. Devuelve un listado de los comerciales que no han realizado ningún pedido. (Utilizando IN o NOT IN).

```
SELECT *
FROM comercial
WHERE id NOT IN (SELECT id_comercial FROM pedido);
```

1.3.7.4 Subconsultas con EXISTS y NOT EXISTS
11. Devuelve un listado de los clientes que no han realizado ningún pedido. (Utilizando EXISTS o NOT EXISTS).

```
SELECT *
FROM cliente c
WHERE NOT EXISTS (SELECT id_cliente FROM pedido p WHERE p.id_cliente = c.id);
```

12. Devuelve un listado de los comerciales que no han realizado ningún pedido. (Utilizando EXISTS o NOT EXISTS).

```
SELECT *
FROM comercial c
WHERE NOT EXISTS (SELECT id_comercial
                  FROM pedido p
                 WHERE p.id_comercial = c.id);
```