

Jehnsy Jesyd Mendoza Colmenares-02200131005

TALLER 3

GESTION DE VENTAS:

- Consultas sobre una tabla

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;
```

id	total	fecha	id_cliente	id_comercial
16	2.389,23	2019-03-11	1	5
15	370,85	2019-03-11	1	5
13	545,75	2019-01-25	6	1
8	1.983,43	2017-10-10	4	6
1	150,5	2017-10-05	5	2
3	65,26	2017-10-05	2	1
5	948,5	2017-09-10	5	2
12	3.045,6	2017-04-25	2	1
14	145,82	2017-02-02	6	1
9	2.480,4	2016-10-10	8	3
2	270,65	2016-09-10	1	5
11	75,29	2016-08-17	3	7
4	110,5	2016-08-17	8	3
6	2.400,6	2016-07-27	7	1
7	5.760	2015-09-10	2	1
10	250,45	2015-06-27	8	2

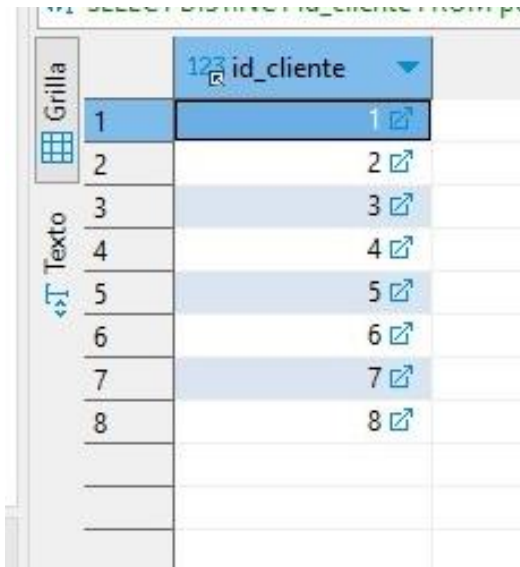
2. Devuelve todos los datos de los dos pedidos de mayor valor.

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

id	total	fecha	id_cliente	id_comercial
7	5.760	2015-09-10	2	1
12	3.045,6	2017-04-25	2	1

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

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



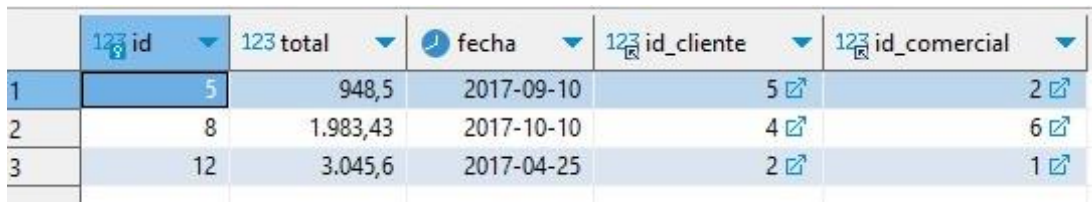
The screenshot shows a database interface with a table titled 'id_cliente'. The table has a single column with 8 rows of data, each containing a unique integer value from 1 to 8. The interface includes a sidebar with 'Grilla' and 'Texto' views, and a search bar at the top.

	id_cliente
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8

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;
```



The screenshot shows a database interface with a table containing 3 rows of data. The columns are 'id', 'total', 'fecha', 'id_cliente', and 'id_comercial'. The data shows three orders from 2017, all with total values greater than 500€.

	id	total	fecha	id_cliente	id_comercial
1	5	948,5	2017-09-10	5	2
2	8	1.983,43	2017-10-10	4	6
3	12	3.045,6	2017-04-25	2	1

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;
```

	ABC nombre ▼	ABC apellido1 ▼	ABC apellido2 ▼
1	Diego	Flores	Salas
2	Antonio	Vega	Hernández
3	Alfredo	Ruiz	Flores

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

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

	123 max_comisión ▼
1	0,150000006

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;
```

	123 id	ABC nombre	ABC apellido1
1	9	Guillermo	López
2	5	Marcos	Loyola
3	1	Aarón	Rivero
4	3	Adolfo	Rubio
5	8	Pepe	Ruiz
6	2	Adela	Salas
7	10	Daniel	Santana
8	6	María	Santana

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;
```

	ABC nombre
1	Aarón
2	Adrián
3	Pepe
4	Pilar

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

```
SELECT nombre
FROM cliente
WHERE NOT (nombre LIKE 'A%')

ORDER BY nombre;
```

	ABC nombre ▼
1	Daniel
2	Guillermo
3	Marcos
4	María
5	Pepe
6	Pilar

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 apellido1 LIKE '%o' OR apellido1 LIKE '%O'
OR apellido2 LIKE '%o' OR apellido2 LIKE '%O';
```

	ABC nombre ▼
1	Antonio

- Consultas multitable

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 repetidos.

```
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;
```

	123 id	ABC nombre	ABC apellido1	ABC apellido2
1	5	Marcos	Loyola	Méndez
2	1	Aarón	Rivero	Gómez
3	3	Adolfo	Rubio	Flores
4	8	Pepe	Ruiz	Santana
5	7	Pilar	Ruiz	[NULL]
6	2	Adela	Salas	Díaz
7	6	María	Santana	Moreno
8	4	Adrián	Suárez	[NULL]

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

```
SELECT c.id AS cliente_id, c.nombre AS cliente_nombre, c.apellido1 AS
cliente_apellido1, c.apellido2 AS cliente_apellido2,
p.id AS pedido_id, p.total AS pedido_total, p.fecha AS pedido_fecha
FROM cliente c
INNER JOIN pedido p ON c.id = p.id_cliente
ORDER BY c.nombre, p.fecha;
```

	123 cliente_id	ABC cliente_nombre	ABC cliente_apellido1	ABC cliente_apellido2	123 pedido_id	123 pedido_total	pedido_fecha
1	1	Aarón	Rivero	Gómez	2	270,65	2016-09-10
2	1	Aarón	Rivero	Gómez	16	2.389,23	2019-03-11
3	1	Aarón	Rivero	Gómez	15	370,85	2019-03-11
4	2	Adela	Salas	Díaz	7	5.760	2015-09-10
5	2	Adela	Salas	Díaz	12	3.045,6	2017-04-25
6	2	Adela	Salas	Díaz	3	65,26	2017-10-05
7	3	Adolfo	Rubio	Flores	11	75,29	2016-08-17
8	4	Adrián	Suárez	[NULL]	8	1.983,43	2017-10-10
9	5	Marcos	Loyola	Méndez	5	948,5	2017-09-10
10	5	Marcos	Loyola	Méndez	1	150,5	2017-10-05
11	6	María	Santana	Moreno	14	145,82	2017-02-02
12	6	María	Santana	Moreno	13	545,75	2019-01-25
13	8	Pepe	Ruiz	Santana	10	250,45	2015-06-27
14	8	Pepe	Ruiz	Santana	4	110,5	2016-08-17
15	8	Pepe	Ruiz	Santana	9	2.480,4	2016-10-10
16	7	Pilar	Ruiz	[NULL]	6	2.400,6	2016-07-27

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 pedido.id, pedido.total, pedido.fecha, cliente.nombre, cliente.apellido1,
cliente.apellido2, comercial.nombre, comercial.apellido1, comercial.apellido2,
comercial.comisión
FROM pedido
INNER JOIN cliente ON pedido.id_cliente = cliente.id
INNER JOIN comercial ON pedido.id_comercial = comercial.id

ORDER BY comercial.nombre, comercial.apellido1, comercial.apellido2
```

	cliente_id	ABC cliente_nombre	ABC cliente_apellido1	ABC cliente_apellido2	pedido_id	pedido_total	pedido_fecha
1	1	Aarón	Rivero	Gómez	2	270,65	2016-09-10
2	1	Aarón	Rivero	Gómez	16	2.389,23	2019-03-11
3	1	Aarón	Rivero	Gómez	15	370,85	2019-03-11
4	2	Adela	Salas	Díaz	7	5,760	2015-09-10
5	2	Adela	Salas	Díaz	12	3.045,6	2017-04-25
6	2	Adela	Salas	Díaz	3	65,26	2017-10-05
7	3	Adolfo	Rubio	Flores	11	75,29	2016-08-17
8	4	Adrián	Suárez	[NULL]	8	1.983,43	2017-10-10
9	5	Marcos	Loyola	Méndez	5	948,5	2017-09-10
10	5	Marcos	Loyola	Méndez	1	150,5	2017-10-05
11	6	María	Santana	Moreno	14	145,82	2017-02-02
12	6	María	Santana	Moreno	13	545,75	2019-01-25
13	8	Pepe	Ruiz	Santana	10	250,45	2015-06-27
14	8	Pepe	Ruiz	Santana	4	110,5	2016-08-17
15	8	Pepe	Ruiz	Santana	9	2.480,4	2016-10-10
16	7	Pilar	Ruiz	[NULL]	6	2.400,6	2016-07-27

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 cliente.id, cliente.nombre, cliente.apellido1, cliente.apellido2, pedido.id
as id_pedido, pedido.total, pedido.fecha, comercial.nombre as nombre_comercial,
comercial.apellido1 as apellido1_comercial, comercial.apellido2 as
apellido2_comercial, comercial.comisión
FROM cliente
LEFT JOIN pedido ON cliente.id = pedido.id_cliente
LEFT JOIN comercial ON pedido.id_comercial = comercial.id

ORDER BY cliente.nombre, cliente.apellido1, cliente.apellido2, pedido.fecha DESC
```

	ABC nombre	ABC apellido1	ABC apellido2	id_pedido	total	fecha	ABC nombre_comercial	ABC apellido1_comercial	ABC apellido2_comercial
1	Aarón	Rivero	Gómez	15	370,85	2019-03-11	Antonio	Carretero	Ortega
2	Aarón	Rivero	Gómez	16	2.389,23	2019-03-11	Antonio	Carretero	Ortega
3	Aarón	Rivero	Gómez	2	270,65	2016-09-10	Antonio	Carretero	Ortega
4	Adela	Salas	Díaz	3	65,26	2017-10-05	Daniel	Sáez	Vega
5	Adela	Salas	Díaz	12	3.045,6	2017-04-25	Daniel	Sáez	Vega
6	Adela	Salas	Díaz	7	5,760	2015-09-10	Daniel	Sáez	Vega
7	Adolfo	Rubio	Flores	11	75,29	2016-08-17	Antonio	Vega	Hernández
8	Adrián	Suárez	[NULL]	8	1.983,43	2017-10-10	Manuel	Domínguez	Hernández
9	Daniel	Santana	Loyola	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
10	Guillermo	López	Gómez	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
11	Marcos	Loyola	Méndez	1	150,5	2017-10-05	Juan	Gómez	López
12	Marcos	Loyola	Méndez	5	948,5	2017-09-10	Juan	Gómez	López
13	María	Santana	Moreno	13	545,75	2019-01-25	Daniel	Sáez	Vega
14	María	Santana	Moreno	14	145,82	2017-02-02	Daniel	Sáez	Vega
15	Pepe	Ruiz	Santana	9	2.480,4	2016-10-10	Diego	Flores	Salas
16	Pepe	Ruiz	Santana	4	110,5	2016-08-17	Diego	Flores	Salas
17	Pepe	Ruiz	Santana	10	250,45	2015-06-27	Juan	Gómez	López

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

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

ORDER BY c.apellido1, c.apellido2, c.nombre;
```

	id	nombre	apellido1	apellido2
1	5	Marcos	Loyola	Méndez

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

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

	nombre	apellido1	apellido2
1	Daniel	Sáez	Vega

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

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

	ABC nombre ▼
1	Adela
2	Pilar
3	María

8. Devuelve un listado con todos los clientes junto con los datos de los pedidos que han realizado. Este listado 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.nombre, c.apellido1, c.apellido2, p.id, p.total, p.fecha
FROM cliente c
LEFT JOIN pedido p ON c.id = p.id_cliente

ORDER BY c.apellido1, c.apellido2, c.nombre
```

	ABC nombre ▼	ABC apellido1 ▼	ABC apellido2 ▼	123 id ▼	123 total ▼	🕒 fecha ▼
1	Guillermo	López	Gómez	[NULL]	[NULL]	[NULL]
2	Marcos	Loyola	Méndez	1	150,5	2017-10-05
3	Marcos	Loyola	Méndez	5	948,5	2017-09-10
4	Aarón	Rivero	Gómez	2	270,65	2016-09-10
5	Aarón	Rivero	Gómez	15	370,85	2019-03-11
6	Aarón	Rivero	Gómez	16	2.389,23	2019-03-11
7	Adolfo	Rubio	Flores	11	75,29	2016-08-17
8	Pilar	Ruiz	[NULL]	6	2.400,6	2016-07-27
9	Pepe	Ruiz	Santana	4	110,5	2016-08-17
10	Pepe	Ruiz	Santana	9	2.480,4	2016-10-10
11	Pepe	Ruiz	Santana	10	250,45	2015-06-27
12	Adela	Salas	Díaz	3	65,26	2017-10-05
13	Adela	Salas	Díaz	7	5.760	2015-09-10
14	Adela	Salas	Díaz	12	3.045,6	2017-04-25
15	Daniel	Santana	Loyola	[NULL]	[NULL]	[NULL]
16	María	Santana	Moreno	13	545,75	2019-01-25
17	María	Santana	Moreno	14	145,82	2017-02-02
18	Adrián	Suárez	[NULL]	8	1.983,43	2017-10-10

9. 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 c.id, c.nombre, c.apellido1, c.apellido2, p.id as id_pedido, p.total,
p.fecha, cl.nombre as nombre_cliente, cl.apellido1 as apellido1_cliente,
cl.apellido2 as apellido2_cliente
FROM comercial c
}LEFT JOIN pedido p ON p.id_comercial = c.id
LEFT JOIN cliente cl ON cl.id = p.id_cliente
ORDER BY c.apellido1, c.apellido2, c.nombre
```

	id	ABC nombre	ABC apellido1	ABC apellido2	id_pedido	123 total	fecha	ABC nombre_cliente	ABC apellido1_cliente	ABC apellido2_cliente
1	5	Antonio	Carretero	Ortega	2	270,65	2016-09-10	Aarón	Rivero	Gómez
2	5	Antonio	Carretero	Ortega	15	370,85	2019-03-11	Aarón	Rivero	Gómez
3	5	Antonio	Carretero	Ortega	16	2.389,23	2019-03-11	Aarón	Rivero	Gómez
4	6	Manuel	Dominguez	Hernández	8	1.983,43	2017-10-10	Adrián	Suárez	[NULL]
5	3	Diego	Flores	Salas	4	110,5	2016-08-17	Pepe	Ruiz	Santana
6	3	Diego	Flores	Salas	9	2.480,4	2016-10-10	Pepe	Ruiz	Santana
7	2	Juan	Gómez	López	1	150,5	2017-10-05	Marcos	Loyola	Méndez
8	2	Juan	Gómez	López	5	948,5	2017-09-10	Marcos	Loyola	Méndez
9	2	Juan	Gómez	López	10	250,45	2015-06-27	Pepe	Ruiz	Santana
10	4	Marta	Herrera	Gil	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
11	8	Alfredo	Ruiz	Flores	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
12	1	Daniel	Sáez	Vega	3	65,26	2017-10-05	Adela	Salas	Díaz
13	1	Daniel	Sáez	Vega	6	2.400,6	2016-07-27	Pilar	Ruiz	[NULL]
14	1	Daniel	Sáez	Vega	7	5.760	2015-09-10	Adela	Salas	Díaz
15	1	Daniel	Sáez	Vega	12	3.045,6	2017-04-25	Adela	Salas	Díaz
16	1	Daniel	Sáez	Vega	13	545,75	2019-01-25	María	Santana	Moreno
17	1	Daniel	Sáez	Vega	14	145,82	2017-02-02	María	Santana	Moreno
18	7	Antonio	Vega	Hernández	11	75,29	2016-08-17	Adolfo	Rubio	Flores

10. 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
ORDER BY c.apellido1, c.apellido2, c.nombre;
```

	id	ABC nombre	ABC apellido1	ABC apellido2
1	9	Guillermo	López	Gómez
2	10	Daniel	Santana	Loyola

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

```
SELECT id, nombre, apellido1, apellido2, comisión
FROM comercial
WHERE id NOT IN (SELECT DISTINCT id_comercial FROM pedido)
ORDER BY apellido1, apellido2, nombre;
```

	123 id	ABC nombre	ABC apellido1	ABC apellido2	123 comisión
1	4	Marta	Herrera	Gil	0,14
2	8	Alfredo	Ruiz	Flores	0,05

12. 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 el listado deberá diferenciar de algún modo los clientes y los comerciales.

```
SELECT 'Cliente' AS tipo, id, nombre, apellido1, apellido2
FROM cliente
WHERE id NOT IN (SELECT DISTINCT id_cliente FROM pedido)

UNION

SELECT 'Comercial' AS tipo, id, nombre, apellido1, apellido2
FROM comercial
WHERE id NOT IN (SELECT DISTINCT id_comercial FROM pedido)

ORDER BY apellido1, apellido2, nombre;
```

	ABC tipo	123 id	ABC nombre	ABC apellido1	ABC apellido2
1	Comercial	4	Marta	Herrera	Gil
2	Cliente	9	Guillermo	López	Gómez
3	Comercial	8	Alfredo	Ruiz	Flores
4	Cliente	10	Daniel	Santana	Loyola

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;
```

123 total_pedidos	
1	20.992,83

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

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

123 media_pedidos	
1	1.312,051875

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

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

	123 total_comerciales	
1		6

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

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

	123 total_clientes	
1		10

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

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

	123 mayor_cantidad
1	5.760

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

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

	123 menor_cantidad
1	65,26

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;
```

	ABC ciudad	123 max_categoria
1	Almería	200
2	Cádiz	100
3	Granada	225
4	Huelva	200
5	Jaén	300
6	Sevilla	300

8. Calcula cuál es el máximo valor de los pedidos realizados durante el mismo día para cada uno de los clientes. 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 c.id AS id_cliente, c.nombre, c.apellido1, c.apellido2, p.fecha,
MAX(p.total) AS maximo_pedido
FROM cliente c
LEFT JOIN pedido p ON c.id = p.id_cliente
GROUP BY c.id, p.fecha;
```

	123 id_cliente	ABC nombre	ABC apellido1	ABC apellido2	📅 fecha	123 maximo_pedido
1	1	Aarón	Rivero	Gómez	2016-09-10	270,65
2	1	Aarón	Rivero	Gómez	2019-03-11	2.389,23
3	2	Adela	Salas	Díaz	2015-09-10	5.760
4	2	Adela	Salas	Díaz	2017-04-25	3.045,6
5	2	Adela	Salas	Díaz	2017-10-05	65,26
6	3	Adolfo	Rubio	Flores	2016-08-17	75,29
7	4	Adrián	Suárez	[NULL]	2017-10-10	1.983,43
8	5	Marcos	Loyola	Méndez	2017-09-10	948,5
9	5	Marcos	Loyola	Méndez	2017-10-05	150,5
10	6	María	Santana	Moreno	2017-02-02	145,82
11	6	María	Santana	Moreno	2019-01-25	545,75
12	7	Pilar	Ruiz	[NULL]	2016-07-27	2.400,6
13	8	Pepe	Ruiz	Santana	2015-06-27	250,45
14	8	Pepe	Ruiz	Santana	2016-08-17	110,5
15	8	Pepe	Ruiz	Santana	2016-10-10	2.480,4
16	9	Guillermo	López	Gómez	[NULL]	[NULL]
17	10	Daniel	Santana	Loyola	[NULL]	[NULL]

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
FROM pedido p
JOIN cliente c ON p.id_cliente = c.id
WHERE (p.fecha, p.id_cliente, p.total) IN (
    SELECT fecha, id_cliente, MAX(total)
    FROM pedido
    WHERE total > 2000
    GROUP BY fecha, id_cliente
)

GROUP BY p.fecha, p.id_cliente;

```

	id_cliente	nombre	apellido1	apellido2	fecha	max_valor
1	2	Adela	Salas	Díaz	2015-09-10	5.760
2	7	Pilar	Ruiz	[NULL]	2016-07-27	2.400,6
3	8	Pepe	Ruiz	Santana	2016-10-10	2.480,4
4	2	Adela	Salas	Díaz	2017-04-25	3.045,6
5	1	Aarón	Rivero	Gómez	2019-03-11	2.389,23

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

```

	id_comercial	nombre	apellido1	apellido2	max_valor
1	3	Diego	Flores	Salas	110,5
2	7	Antonio	Vega	Hernández	75,29

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 ningún 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 num_pedidos
FROM cliente c
LEFT JOIN pedido p ON c.id = p.id_cliente

GROUP BY c.id, c.nombre, c.apellido1, c.apellido2;
```

	id	nombre	apellido1	apellido2	num_pedidos
1	1	Aarón	Rivero	Gómez	3
2	2	Adela	Salas	Díaz	3
3	3	Adolfo	Rubio	Flores	1
4	4	Adrián	Suárez	[NULL]	1
5	5	Marcos	Loyola	Méndez	2
6	6	María	Santana	Moreno	2
7	7	Pilar	Ruiz	[NULL]	1
8	8	Pepe	Ruiz	Santana	3
9	9	Guillermo	López	Gómez	0
10	10	Daniel	Santana	Loyola	0

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 num_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;
```

	id	nombre	apellido1	apellido2	num_pedidos
1	2	Adela	Salas	Díaz	2
2	4	Adrián	Suárez	[NULL]	1
3	5	Marcos	Loyola	Méndez	2
4	6	María	Santana	Moreno	1

13. Devuelve un listado que muestre el identificador de cliente, nombre, primer apellido y el valor de la máxima cantidad del pedido realizado por cada uno de los clientes. El resultado debe mostrar aquellos clientes 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 max_cantidad
FROM cliente c
LEFT JOIN pedido p ON c.id = p.id_cliente

GROUP BY c.id, c.nombre, c.apellido1;
```

	id	nombre	apellido1	max_cantidad
1	1	Aarón	Rivero	2.389,23
2	2	Adela	Salas	5.760
3	3	Adolfo	Rubio	75,29
4	4	Adrián	Suárez	1.983,43
5	5	Marcos	Loyola	948,5
6	6	María	Santana	545,75
7	7	Pilar	Ruiz	2.400,6
8	8	Pepe	Ruiz	2.480,4
9	9	Guillermo	López	0
10	10	Daniel	Santana	0

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

```
SELECT YEAR(fecha) AS año, MAX(total) AS maximo
FROM pedido

GROUP BY YEAR(fecha);
```

	año	maximo
1	2.015	5.760
2	2.016	2.480,4
3	2.017	3.045,6
4	2.019	2.389,23

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 YEAR(fecha) ;
```

	123 año ▼	123 total_pedidos ▼
1	2.015	2
2	2.016	5
3	2.017	6
4	2.019	3

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'  
)  
;
```

	123 id	123 total	🕒 fecha	123 id_cliente	123 id_comercial
1	3	65,26	2017-10-05	2	1
2	7	5.760	2015-09-10	2	1
3	12	3.045,6	2017-04-25	2	1

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'  
)
```

	123 COUNT(*)
1	6

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

```
SELECT id, nombre, apellido1, apellido2
FROM cliente
WHERE id = (
    SELECT id_cliente
    FROM pedido
    WHERE total = (
        SELECT MAX(total)
        FROM pedido
        WHERE YEAR(fecha) = 2019
    )
)
```

	123 id	ABC nombre	ABC apellido1	ABC apellido2
1	1	Aarón	Rivero	Gómez

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;
```

	🕒 fecha	123 total
1	2016-08-17	110,5

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,
p.fecha
FROM cliente c, pedido p
WHERE c.id = p.id_cliente
AND YEAR(p.fecha) = 2017
AND p.total >= (SELECT AVG(total) FROM pedido WHERE YEAR(fecha) = 2017)

ORDER BY c.id, p.fecha;
```

	123 id	ABC nombre	ABC apellido1	ABC apellido2	123 id_pedido	123 total	📅 fecha
1	2	Adela	Salas	Díaz	12	3.045,6	2017-04-25
2	4	Adrián	Suárez	[NULL]	8	1.983,43	2017-10-10

6. Devuelve el pedido más caro que existe en la tabla pedido si hacer uso de MAX, ORDER BY ni LIMIT.

```
SELECT id, total
FROM pedido
WHERE total >= ALL (SELECT total FROM pedido);
```

	123 id	123 total
1	7	5.760

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

```
SELECT id, nombre, apellido1, apellido2
FROM cliente
WHERE id <> ALL (
    SELECT DISTINCT id_cliente
    FROM pedido
)
```

SELECT id, nombre, apellido1, apellido2 FROM cliente

	id	nombre	apellido1	apellido2
1	9	Guillermo	López	Gómez
2	10	Daniel	Santana	Loyola

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

```
SELECT *
FROM comercial
WHERE NOT EXISTS (
    SELECT 1
    FROM pedido
    WHERE pedido.id_comercial = comercial.id
);
```

	id	nombre	apellido1	apellido2	comisión
1	4	Marta	Herrera	Gil	0,14
2	8	Alfredo	Ruiz	Flores	0,05

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

```
SELECT id, nombre, apellido1, apellido2
FROM cliente
WHERE id NOT IN (SELECT id_cliente FROM pedido);
```

	id	nombre	apellido1	apellido2
1	9	Guillermo	López	Gómez
2	10	Daniel	Santana	Loyola

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

```
SELECT id, nombre, apellido1, apellido2
FROM comercial
WHERE id NOT IN (SELECT DISTINCT id_comercial FROM pedido);
```

	id	nombre	apellido1	apellido2
1	4	Marta	Herrera	Gil
2	8	Alfredo	Ruiz	Flores

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

```
SELECT id, nombre, apellido1, apellido2
FROM cliente
```

```

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

```

	123 id	ABC nombre	ABC apellido1	ABC apellido2
1	9	Guillermo	López	Gómez
2	10	Daniel	Santana	Loyola

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

```

SELECT id, nombre, apellido1, apellido2, comisión
FROM comercial c
WHERE NOT EXISTS (
    SELECT 1
    FROM pedido p
    WHERE p.id_comercial = c.id
) ;

```

	123 id	ABC nombre	ABC apellido1	ABC apellido2	123 comisión
1	4	Marta	Herrera	Gil	0,14
2	8	Alfredo	Ruiz	Flores	0,05

GESTION DE EMPLEADOS

Consultas sobre una tabla

1. Lista el primer apellido de todos los empleados.

```
SELECT apellido1 FROM empleado;
```

	ABC apellido1 ▼
1	Rivero
2	Salas
3	Rubio
4	Suárez
5	Loyola
6	Santana
7	Ruiz
8	Ruiz
9	Gómez
10	Flores
11	Herrera
12	Salas
13	Sáez

2. Lista el primer apellido de los empleados eliminando los apellidos que estén repetidos.

```
SELECT DISTINCT apellido1 FROM empleado;
```

	ABC apellido1 ▼
1	Rivero
2	Salas
3	Rubio
4	Suárez
5	Loyola
6	Santana
7	Ruiz
8	Gómez
9	Flores
10	Herrera
11	Sáez

3. Lista todas las columnas de la tabla empleado.

```
SELECT * FROM empleado;
```

id	nif	nombre	apellido1	apellido2	id_departamento
1	32481596F	Aarón	Rivero	Gómez	1
2	Y5575632D	Adela	Salas	Díaz	2
3	R6970642B	Adolfo	Rubio	Flores	3
4	77705545E	Adrián	Suárez	[NULL]	4
5	17087203C	Marcos	Loyola	Méndez	5
6	38382980M	María	Santana	Moreno	1
7	80576669X	Pilar	Ruiz	[NULL]	2
8	71651431Z	Pepe	Ruiz	Santana	3
9	56399183D	Juan	Gómez	López	2
10	46384486H	Diego	Flores	Salas	5
11	67389283A	Marta	Herrera	Gil	1
12	41234836R	Irene	Salas	Flores	[NULL]
13	82635162B	Juan Antonio	Sáez	Guerrero	[NULL]

4. Lista el nombre y los apellidos de todos los empleados.

```
SELECT nombre, apellido1, apellido2 FROM empleado;
```

	nombre	apellido1	apellido2
1	Aarón	Rivero	Gómez
2	Adela	Salas	Díaz
3	Adolfo	Rubio	Flores
4	Adrián	Suárez	[NULL]
5	Marcos	Loyola	Méndez
6	María	Santana	Moreno
7	Pilar	Ruiz	[NULL]
8	Pepe	Ruiz	Santana
9	Juan	Gómez	López
10	Diego	Flores	Salas
11	Marta	Herrera	Gil
12	Irene	Salas	Flores
13	Juan Antonio	Sáez	Guerrero

5. Lista el identificador de los departamentos de los empleados que aparecen en la tabla empleado.

```
SELECT id_departamento FROM empleado;
```

	id_departamento
1	[NULL]
2	[NULL]
3	1
4	1
5	1
6	2
7	2
8	2
9	3
10	3
11	4
12	5
13	5

6. Lista el identificador de los departamentos de los empleados que aparecen en la tabla empleado, eliminando los identificadores que aparecen repetidos.

```
SELECT DISTINCT id_departamento FROM empleado;
```

	id_departamento
1	[NULL]
2	1
3	2
4	3
5	4
6	5

7. Lista el nombre y apellidos de los empleados en una única columna.

```
SELECT CONCAT(nombre, ' ', apellido1, ' ', apellido2) AS  
nombre_completo  
  
FROM empleado;
```

	ABC nombre_completo ▼
1	Aarón Rivero Gómez
2	Adela Salas Díaz
3	Adolfo Rubio Flores
4	[NULL]
5	Marcos Loyola Méndez
6	María Santana Moreno
7	[NULL]
8	Pepe Ruiz Santana
9	Juan Gómez López
10	Diego Flores Salas
11	Marta Herrera Gil
12	Irene Salas Flores
13	Juan Antonio Sáez Guerrero

8. Lista el nombre y apellidos de los empleados en una única columna, convirtiendo todos los caracteres en mayúscula.

```
SELECT UPPER(CONCAT(nombre, ' ', apellido1, ' ', apellido2)) AS  
nombre_completo  
  
FROM empleado;
```

	ABC nombre_completo ▼
1	AARÓN RIVERO GÓMEZ
2	ADELA SALAS DÍAZ
3	ADOLFO RUBIO FLORES
4	[NULL]
5	MARCOS LOYOLA MÉNDEZ
6	MARÍA SANTANA MORENO
7	[NULL]
8	PEPE RUIZ SANTANA
9	JUAN GÓMEZ LÓPEZ
10	DIEGO FLORES SALAS
11	MARTA HERRERA GIL
12	IRENE SALAS FLORES
13	JUAN ANTONIO SÁEZ GUERRI

9. Lista el nombre y apellidos de los empleados en una única columna, convirtiendo todos los caracteres en minúscula.

```
SELECT LOWER(CONCAT(nombre, ' ', apellido1, ' ', apellido2)) AS  
nombre_completo  
FROM empleado;
```

	ABC nombre_completo ▼
1	aarón rivero gómez
2	adela salas díaz
3	adolfo rubio flores
4	[NULL]
5	marcos loyola méndez
6	maría santana moreno
7	[NULL]
8	pepe ruiz santana
9	juan gómez lópez
10	diego flores salas
11	marta herrera gil
12	irene salas flores
13	juan antonio sáez guerrero

10. Lista el identificador de los empleados junto al nif, pero el nif deberá aparecer en dos columnas, una mostrará únicamente los dígitos del nif y la otra la letra.

```
SELECT id, SUBSTR(nif, 1, 8) AS nif_digitos, SUBSTR(nif, 9, 1) AS  
nif_letra
```

```
FROM empleado;
```

	123 id ▼	ABC nif_digitos ▼	ABC nif_letra ▼
1	5	17087203	C
2	1	32481596	F
3	6	38382980	M
4	12	41234836	R
5	10	46384486	H
6	9	56399183	D
7	11	67389283	A
8	8	71651431	Z
9	4	77705545	E
10	7	80576669	X
11	13	82635162	B
12	3	R6970642	B
13	2	Y5575632	D

11. Lista el nombre de cada departamento y el valor del presupuesto actual del que dispone. Para calcular este dato tendrá que restar al valor del presupuesto inicial (columna presupuesto) los gastos que se han generado (columna gastos). Tenga en cuenta que en algunos casos pueden existir valores negativos. Utilice un alias apropiado para la nueva columna columna que está calculando.

```
SELECT nombre, (presupuesto - gastos) AS presupuesto_actual  
FROM departamento;
```

	ABC nombre ▼	123 presupuesto_actual ▼
1	Desarrollo	114.000
2	Sistemas	129.000
3	Recursos Humanos	255.000
4	Contabilidad	107.000
5	I+D	-5.000
6	Proyectos	0
7	Publicidad	-1.000

12. Lista el nombre de los departamentos y el valor del presupuesto actual ordenado de forma ascendente.

```
SELECT nombre, (presupuesto - gastos) AS presupuesto_actual  
FROM departamento
```

```
ORDER BY presupuesto_actual ASC;
```

	ABC nombre ▼	123 presupuesto_actual ▼
1	I+D	-5.000
2	Publicidad	-1.000
3	Proyectos	0
4	Contabilidad	107.000
5	Desarrollo	114.000
6	Sistemas	129.000
7	Recursos Humanos	255.000

13. Lista el nombre de todos los departamentos ordenados de forma ascendente.

```
SELECT nombre  
FROM departamento  
ORDER BY nombre ASC;
```

	ABC nombre ▼
1	Contabilidad
2	Desarrollo
3	I+D
4	Proyectos
5	Publicidad
6	Recursos Humanos
7	Sistemas

14. Lista el nombre de todos los departamentos ordenados de forma descendente.

```
SELECT nombre  
FROM departamento  
  
ORDER BY nombre DESC;
```

	ABC nombre ▼
1	Sistemas
2	Recursos Humanos
3	Publicidad
4	Proyectos
5	I+D
6	Desarrollo
7	Contabilidad

15. Lista los apellidos y el nombre de todos los empleados, ordenados de forma alfabética teniendo en cuenta en primer lugar sus apellidos y luego su nombre.

```
SELECT apellido1, apellido2, nombre  
FROM empleado
```

```
ORDER BY apellido1 ASC, apellido2 ASC, nombre ASC;
```

	ABC apellido1 ▼	ABC apellido2 ▼	ABC nombre ▼
1	Flores	Salas	Diego
2	Gómez	López	Juan
3	Herrera	Gil	Marta
4	Loyola	Méndez	Marcos
5	Rivero	Gómez	Aarón
6	Rubio	Flores	Adolfo
7	Ruiz	[NULL]	Pilar
8	Ruiz	Santana	Pepe
9	Sáez	Guerrero	Juan Antonio
10	Salas	Díaz	Adela
11	Salas	Flores	Irene
12	Santana	Moreno	María
13	Suárez	[NULL]	Adrián

16. Devuelve una lista con el nombre y el presupuesto, de los 3 departamentos que tienen mayor presupuesto.

```
SELECT nombre, presupuesto AS presupuesto_actual  
FROM departamento
```

```
ORDER BY presupuesto DESC  
LIMIT 3;
```

	ABC nombre ▼	123 presupuesto_actual ▼
1	I+D	375.000
2	Recursos Humanos	280.000
3	Sistemas	150.000

17. Devuelve una lista con el nombre y el presupuesto, de los 3 departamentos que tienen menor presupuesto.

```
SELECT nombre, presupuesto AS presupuesto_actual  
FROM departamento  
ORDER BY presupuesto ASC  
LIMIT 3;
```

	ABC nombre ▼	123 presupuesto_actual ▼
1	Publicidad	0
2	Proyectos	0
3	Contabilidad	110.000

18. Devuelve una lista con el nombre y el gasto, de los 2 departamentos que tienen mayor gasto.

```
SELECT nombre, gastos  
FROM departamento  
ORDER BY gastos DESC  
LIMIT 2;
```

	ABC nombre ▼	123 gastos ▼
1	I+D	380.000
2	Recursos Humanos	25.000

19. Devuelve una lista con el nombre y el gasto, de los 2 departamentos que tienen menor gasto.

```
SELECT nombre, gastos
FROM departamento
ORDER BY gastos ASC
LIMIT 2;
```

	ABC nombre ▼	123 gastos ▼
1	Proyectos	0
2	Publicidad	1.000

20. Devuelve una lista con 5 filas a partir de la tercera fila de la tabla empleado. La tercera fila se debe incluir en la respuesta. La respuesta debe incluir todas las columnas de la tabla empleado.

```
SELECT *
FROM empleado
```

```
LIMIT 2, 5;
```

	123 id ▼	ABC nif ▼	ABC nombre ▼	ABC apellido1 ▼	ABC apellido2 ▼	123 id_departamento ▼
1	3	R6970642B	Adolfo	Rubio	Flores	3 🔗
2	4	77705545E	Adrián	Suárez	[NULL]	4 🔗
3	5	17087203C	Marcos	Loyola	Méndez	5 🔗
4	6	38382980M	María	Santana	Moreno	1 🔗
5	7	80576669X	Pilar	Ruiz	[NULL]	2 🔗

21. Devuelve una lista con el nombre de los departamentos y el presupuesto, de aquellos que tienen un presupuesto mayor o igual a 150000 euros.

```
SELECT nombre, presupuesto  
FROM departamento  
WHERE presupuesto >= 150000;
```

SELECT nombre, presupuesto FROM departa

	ABC nombre	123 presupuesto
1	Sistemas	150.000
2	Recursos Humanos	280.000
3	I+D	375.000

22. Devuelve una lista con el nombre de los departamentos y el gasto, de aquellos que tienen menos de 5000 euros de gastos.

```
SELECT nombre, gastos  
FROM departamento  
WHERE gastos < 5000;
```

	ABC nombre	123 gastos
1	Contabilidad	3.000
2	Proyectos	0
3	Publicidad	1.000

23. Devuelve una lista con el nombre de los departamentos y el presupuesto, de aquellos que tienen un presupuesto entre 100000 y 200000 euros. Sin utilizar el operador BETWEEN.

```
SELECT nombre, presupuesto  
FROM departamento
```

```
WHERE presupuesto > 100000 AND presupuesto < 200000;
```

	ABC nombre ▼	123 presupuesto ▼
1	Desarrollo	120.000
2	Sistemas	150.000
3	Contabilidad	110.000

24. Devuelve una lista con el nombre de los departamentos que no tienen un presupuesto entre 100000 y 200000 euros. Sin utilizar el operador BETWEEN.

```
SELECT nombre, presupuesto  
FROM departamento
```

```
WHERE presupuesto < 100000 OR presupuesto > 200000;
```

	ABC nombre ▼	123 presupuesto ▼
1	Recursos Humanos	280.000
2	I+D	375.000
3	Proyectos	0
4	Publicidad	0

25. Devuelve una lista con el nombre de los departamentos que tienen un presupuesto entre 100000 y 200000 euros. Utilizando el operador BETWEEN.

```
SELECT nombre FROM departamento WHERE presupuesto BETWEEN 100000 AND 200000;
```

	ABC nombre ▼
1	Desarrollo
2	Sistemas
3	Contabilidad

26. Devuelve una lista con el nombre de los departamentos que no tienen un presupuesto entre 100000 y 200000 euros. Utilizando el operador BETWEEN.

```
SELECT nombre, presupuesto  
FROM departamento  
WHERE presupuesto NOT BETWEEN 100000 AND 200000;
```

	ABC nombre ▼	123 presupuesto ▼
1	Recursos Humanos	280.000
2	I+D	375.000
3	Proyectos	0
4	Publicidad	0

27. Devuelve una lista con el nombre de los departamentos, gastos y presupuesto, de aquellos departamentos donde los gastos sean mayores que el presupuesto del que disponen.

```
SELECT nombre, presupuesto, gastos  
FROM departamento
```

```
WHERE gastos > presupuesto;
```

	ABC nombre ▼	123 presupuesto ▼	123 gastos ▼
1	I+D	375.000	380.000
2	Publicidad	0	1.000

28. Devuelve una lista con el nombre de los departamentos, gastos y presupuesto, de aquellos departamentos donde los gastos sean menores que el presupuesto del que disponen.

```
SELECT nombre, presupuesto, gastos  
FROM departamento
```

```
WHERE gastos < presupuesto;
```

	ABC nombre ▼	123 presupuesto ▼	123 gastos ▼
1	Desarrollo	120.000	6.000
2	Sistemas	150.000	21.000
3	Recursos Humanos	280.000	25.000
4	Contabilidad	110.000	3.000

29. Devuelve una lista con el nombre de los departamentos, gastos y presupuesto, de aquellos departamentos donde los gastos sean iguales al presupuesto del que disponen.

```
SELECT nombre, presupuesto, gastos
FROM departamento
WHERE presupuesto = gastos;
```

	ABC nombre ▼	123 presupuesto ▼	123 gastos ▼
1	Proyectos	0	0

30. Lista todos los datos de los empleados cuyo segundo apellido sea NULL.

```
SELECT * FROM empleado WHERE apellido2 IS NULL;
```

	123 id ▼	ABC nif ▼	ABC nombre ▼	ABC apellido1 ▼	ABC apellido2 ▼	123 id_departamento ▼
1	4	77705545E	Adrián	Suárez	[NULL]	4 ↗
2	7	80576669X	Pilar	Ruiz	[NULL]	2 ↗

31. Lista todos los datos de los empleados cuyo segundo apellido no sea NULL.

```
SELECT *
FROM empleado

WHERE apellido2 IS NOT NULL;
```

	id	nif	nombre	apellido1	apellido2	id_departamento
1	1	32481596F	Aarón	Rivero	Gómez	1
2	2	Y5575632D	Adela	Salas	Díaz	2
3	3	R6970642B	Adolfo	Rubio	Flores	3
4	5	17087203C	Marcos	Loyola	Méndez	5
5	6	38382980M	María	Santana	Moreno	1
6	8	71651431Z	Pepe	Ruiz	Santana	3
7	9	56399183D	Juan	Gómez	López	2
8	10	46384486H	Diego	Flores	Salas	5
9	11	67389283A	Marta	Herrera	Gil	1
10	12	41234836R	Irene	Salas	Flores	[NULL]
11	13	82635162B	Juan Antonio	Sáez	Guerrero	[NULL]

32. Lista todos los datos de los empleados cuyo segundo apellido sea López.

```
SELECT * FROM empleado WHERE apellido2 = 'López';
```

	id	nif	nombre	apellido1	apellido2	id_departamento
1	9	56399183D	Juan	Gómez	López	2

33. Lista todos los datos de los empleados cuyo segundo apellido sea Díaz o Moreno. Sin utilizar el operador IN.

```
SELECT *  
FROM empleado  
WHERE apellido2 = 'Díaz' OR apellido2 = 'Moreno';
```

	id	nif	nombre	apellido1	apellido2	id_departamento
1	2	Y5575632D	Adela	Salas	Díaz	2
2	6	38382980M	María	Santana	Moreno	1

34. Lista todos los datos de los empleados cuyo segundo apellido sea Díaz o Moreno. Utilizando el operador IN.

```
SELECT * FROM empleado WHERE apellido2 IN ('Díaz', 'Moreno');
```

SELECT * FROM empleado WHERE apellido2 | Enter a SQL expression to filter results (use Ctrl+Space)

	id	nif	nombre	apellido1	apellido2	id_departamento
1	2	Y5575632D	Adela	Salas	Díaz	2
2	6	38382980M	María	Santana	Moreno	1

35. Lista los nombres, apellidos y nif de los empleados que trabajan en el departamento 3.

```
SELECT nombre, apellido1, apellido2, nif
FROM empleado
WHERE id_departamento = 3;
```

	ABC nombre ▼	ABC apellido1 ▼	ABC apellido2 ▼	ABC nif ▼
1	Adolfo	Rubio	Flores	R6970642B
2	Pepe	Ruiz	Santana	71651431Z

36. Lista los nombres, apellidos y nif de los empleados que trabajan en los departamentos 2, 4 o 5.

```
SELECT nombre, apellido1, apellido2, nif
FROM empleado
```

```
WHERE id_departamento IN (2, 4, 5);
```

	ABC nombre ▼	ABC apellido1 ▼	ABC apellido2 ▼	ABC nif ▼
1	Adela	Salas	Díaz	Y5575632D
2	Adrián	Suárez	[NULL]	77705545E
3	Marcos	Loyola	Méndez	17087203C
4	Pilar	Ruiz	[NULL]	80576669X
5	Juan	Gómez	López	56399183D
6	Diego	Flores	Salas	46384486H

• Consultas multitabla

1. Devuelve un listado con los empleados y los datos de los departamentos donde trabaja cada uno.

```
SELECT e.*, d.*
FROM empleado e
JOIN departamento d ON e.id_departamento = d.id;
```

	123 id	ABC nif	ABC nombre	ABC apellido1	ABC apellido2	123 id_departamento	123 id	ABC nombre	123 presupuesto	123 gastos
1	1	32481596F	Aarón	Rivero	Gómez	1	1	Desarrollo	120.000	6.000
2	2	Y5575632D	Adela	Salas	Díaz	2	2	Sistemas	150.000	21.000
3	3	R6970642B	Adolfo	Rubio	Flores	3	3	Recursos Humanos	280.000	25.000
4	4	77705545E	Adrián	Suárez	[NULL]	4	4	Contabilidad	110.000	3.000
5	5	17087203C	Marcos	Loyola	Méndez	5	5	I+D	375.000	380.000
6	6	38382980M	María	Santana	Moreno	1	1	Desarrollo	120.000	6.000
7	7	80576669X	Pilar	Ruiz	[NULL]	2	2	Sistemas	150.000	21.000
8	8	71651431Z	Pepe	Ruiz	Santana	3	3	Recursos Humanos	280.000	25.000
9	9	56399183D	Juan	Gómez	López	2	2	Sistemas	150.000	21.000
10	10	46384486H	Diego	Flores	Salas	5	5	I+D	375.000	380.000
11	11	67389283A	Marta	Herrera	Gil	1	1	Desarrollo	120.000	6.000

2. Devuelve un listado con los empleados y los datos de los departamentos donde trabaja cada uno. Ordena el resultado, en primer lugar por el nombre del departamento (en orden alfabético) y en segundo lugar por los apellidos y el nombre de los empleados.

```
SELECT e.nombre, e.apellido1, e.apellido2, d.nombre,
d.presupuesto, d.gastos

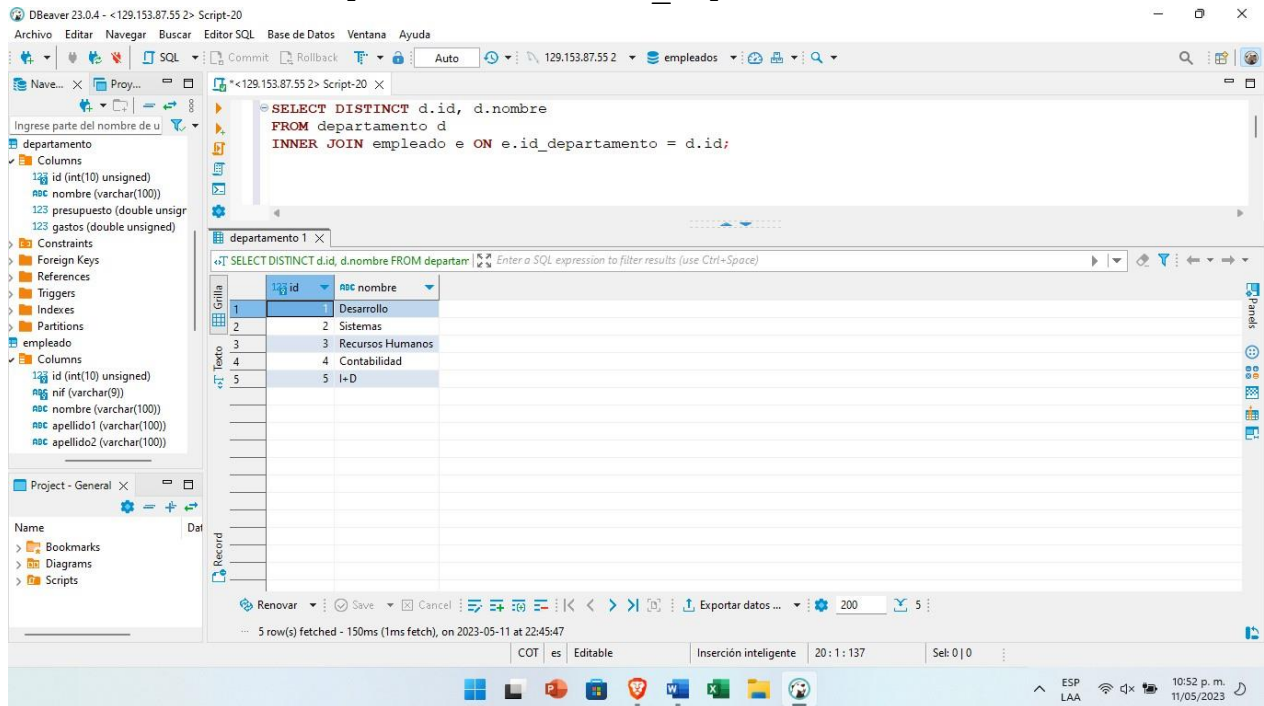
FROM empleado e
JOIN departamento d ON e.id_departamento = d.id

ORDER BY d.nombre, e.apellido1, e.apellido2, e.nombre;
```

	ABC nombre	ABC apellido1	ABC apellido2	ABC nombre	123 presupuesto	123 gastos
1	Adrián	Suárez	[NULL]	Contabilidad	110.000	3.000
2	Marta	Herrera	Gil	Desarrollo	120.000	6.000
3	Aarón	Rivero	Gómez	Desarrollo	120.000	6.000
4	María	Santana	Moreno	Desarrollo	120.000	6.000
5	Diego	Flores	Salas	I+D	375.000	380.000
6	Marcos	Loyola	Méndez	I+D	375.000	380.000
7	Adolfo	Rubio	Flores	Recursos Humanos	280.000	25.000
8	Pepe	Ruiz	Santana	Recursos Humanos	280.000	25.000
9	Juan	Gómez	López	Sistemas	150.000	21.000
10	Pilar	Ruiz	[NULL]	Sistemas	150.000	21.000
11	Adela	Salas	Díaz	Sistemas	150.000	21.000

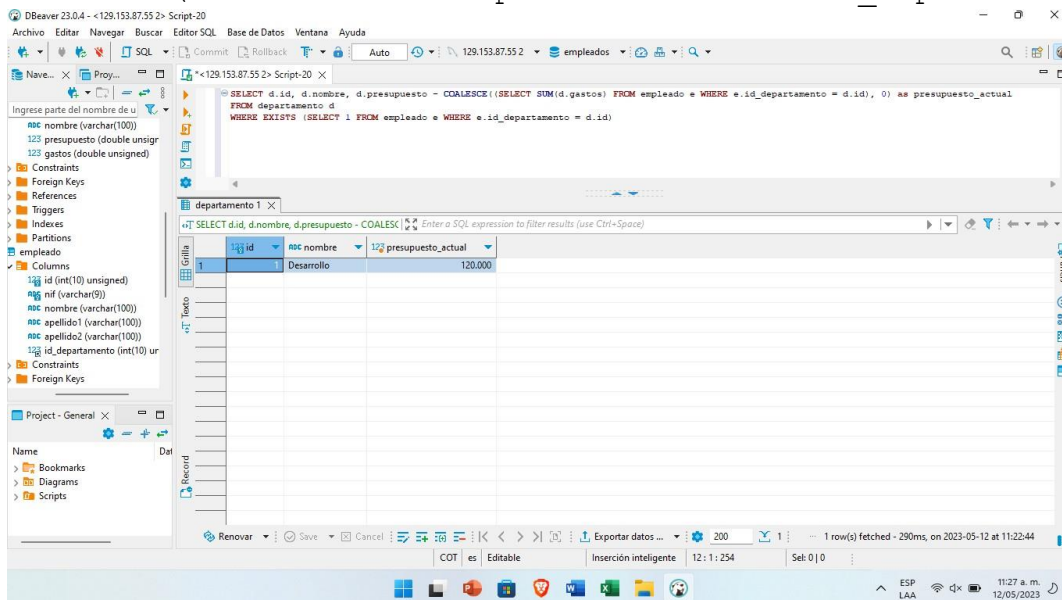
3. Devuelve un listado con el identificador y el nombre del departamento, solamente de aquellos departamentos que tienen empleados.

```
SELECT DISTINCT d.id, d.nombre
FROM departamento d
INNER JOIN empleado e ON e.id_departamento = d.id;
```



4. Devuelve un listado con el identificador, el nombre del departamento y el valor del presupuesto actual del que dispone, solamente de aquellos departamentos que tienen empleados. El valor del presupuesto actual lo puede calcular restando al valor del presupuesto inicial (columna presupuesto) el valor de los gastos que ha generado (columna gastos).

```
SELECT d.id, d.nombre, d.presupuesto - COALESCE((SELECT SUM(d.gastos)
FROM empleado e WHERE e.id_departamento = d.id), 0) as
presupuesto_actual
FROM departamento d
WHERE EXISTS (SELECT 1 FROM empleado e WHERE e.id_departamento = d.id)
```



5. Devuelve el nombre del departamento donde trabaja el empleado que tiene el nif 38382980M.

```
SELECT d.nombre  
FROM empleado e  
INNER JOIN departamento d ON e.id_departamento = d.id  
WHERE e.nif = '38382980M';
```

	ABC nombre ▼
1	Desarrollo

6. Devuelve el nombre del departamento donde trabaja el empleado Pepe Ruiz Santana.

```
SELECT d.nombre  
FROM empleado e  
INNER JOIN departamento d ON e.id_departamento = d.id  
  
WHERE e.nombre = 'Pepe' AND e.apellido1 = 'Ruiz' AND e.apellido2 =  
'Santana';
```

	ABC nombre ▼
1	Recursos Humanos

7. Devuelve un listado con los datos de los empleados que trabajan en el departamento de I+D. Ordena el resultado alfabéticamente.

```
SELECT *
FROM empleado e
INNER JOIN departamento d ON e.id_departamento = d.id
WHERE d.nombre = 'I+D'
ORDER BY e.apellido1, e.apellido2, e.nombre;
```

	id	nif	nombre	apellido1	apellido2	id_departamento	id	nombre	presupuesto	gastos
1	10	46384486H	Diego	Flores	Salas	5	5	I+D	375.000	380.000
2	5	17087203C	Marcos	Loyola	Méndez	5	5	I+D	375.000	380.000

8. Devuelve un listado con los datos de los empleados que trabajan en el departamento de Sistemas, Contabilidad o I+D. Ordena el resultado alfabéticamente.

```
SELECT *
FROM empleado e

JOIN departamento d ON e.id_departamento = d.id
WHERE d.nombre IN ('Sistemas', 'Contabilidad', 'I+D')
ORDER BY e.apellido1, e.apellido2, e.nombre;
```

	id	nif	nombre	apellido1	apellido2	id_departamento	id	nombre	presupuesto	gastos
1	10	46384486H	Diego	Flores	Salas	5	5	I+D	375.000	380.000
2	9	56399183D	Juan	Gómez	López	2	2	Sistemas	150.000	21.000
3	5	17087203C	Marcos	Loyola	Méndez	5	5	I+D	375.000	380.000
4	7	80576669X	Pilar	Ruiz	[NULL]	2	2	Sistemas	150.000	21.000
5	2	Y5575632D	Adela	Salas	Díaz	2	2	Sistemas	150.000	21.000
6	4	77705545E	Adrián	Suárez	[NULL]	4	4	Contabilidad	110.000	3.000

9. Devuelve una lista con el nombre de los empleados que tienen los departamentos que no tienen un presupuesto entre 100000 y 200000 euros.

```
SELECT e.nombre  
FROM empleado e  
WHERE e.id_departamento NOT IN (  
    SELECT d.id  
    FROM departamento d  
    WHERE d.presupuesto BETWEEN 100000 AND 200000  
)
```

	ABC nombre ▼
1	Adolfo
2	Marcos
3	Pepe
4	Diego

10. Devuelve un listado con el nombre de los departamentos donde existe algún empleado cuyo segundo apellido sea NULL. Tenga en cuenta que no debe mostrar nombres de departamentos que estén repetidos.

```
SELECT DISTINCT d.nombre  
FROM empleado e  
  
INNER JOIN departamento d  
ON e.id_departamento = d.id  
WHERE e.apellido2 IS NULL;
```

	ABC nombre ▼
1	Contabilidad
2	Sistemas

11. Devuelve un listado con todos los empleados junto con los datos de los departamentos donde trabajan. Este listado también debe incluir los empleados que no tienen ningún departamento asociado.

```
SELECT e.*, d.nombre
FROM empleado e
LEFT JOIN departamento d ON e.id_departamento = d.id;
```

	123 id	ABC nif	ABC nombre	ABC apellido1	ABC apellido2	123 id_departamento	ABC nombre
1	1	32481596F	Aarón	Rivero	Gómez	1	Desarrollo
2	2	Y5575632D	Adela	Salas	Díaz	2	Sistemas
3	3	R6970642B	Adolfo	Rubio	Flores	3	Recursos Humanos
4	4	77705545E	Adrián	Suárez	[NULL]	4	Contabilidad
5	5	17087203C	Marcos	Loyola	Méndez	5	I+D
6	6	38382980M	María	Santana	Moreno	1	Desarrollo
7	7	80576669X	Pilar	Ruiz	[NULL]	2	Sistemas
8	8	71651431Z	Pepe	Ruiz	Santana	3	Recursos Humanos
9	9	56399183D	Juan	Gómez	López	2	Sistemas
10	10	46384486H	Diego	Flores	Salas	5	I+D
11	11	67389283A	Marta	Herrera	Gil	1	Desarrollo
12	12	41234836R	Irene	Salas	Flores	[NULL]	[NULL]
13	13	82635162B	Juan Antonio	Sáez	Guerrero	[NULL]	[NULL]

12. Devuelve un listado donde sólo aparezcan aquellos empleados que no tienen ningún departamento asociado.

```
SELECT *
FROM empleado e
LEFT JOIN departamento d ON e.id_departamento = d.id

WHERE e.id_departamento IS NULL;
```

SELECT * FROM empleado e LEFT JOIN departam

	123 id	ABC nif	ABC nombre	ABC apellido1	ABC apellido2	123 id_departamento	123 id	ABC nombre	123 presupuesto	123 gastos
1	12	41234836R	Irene	Salas	Flores	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
2	13	82635162B	Juan Antonio	Sáez	Guerrero	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]

13. Devuelve un listado donde sólo aparezcan aquellos departamentos que no tienen ningún empleado asociado.

```
SELECT d.id, d.nombre
FROM departamento d
LEFT JOIN empleado e ON d.id = e.id_departamento
WHERE e.id IS NULL;
```

	123 id	ABC nombre
1	6	Proyectos
2	7	Publicidad

14. Devuelve un listado con todos los empleados junto con los datos de los departamentos donde trabajan. El listado debe incluir los empleados que no tienen ningún departamento asociado y los departamentos que no tienen ningún empleado asociado. Ordene el listado alfabéticamente por el nombre del departamento.

```
SELECT e.*, d.*
FROM empleado e
LEFT JOIN departamento d ON e.id_departamento = d.id

ORDER BY d.nombre;
```

	123 id	ABC nif	ABC nombre	ABC apellido1	ABC apellido2	123 id_departamento	123 id	ABC nombre	123 presupuesto	123 gastos
1	13	82635162B	Juan Antonio	Sáez	Guerrero	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
2	12	41234836R	Irene	Salas	Flores	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
3	4	77705545E	Adrián	Suárez	[NULL]	4	4	Contabilidad	110.000	3.000
4	11	67389283A	Marta	Herrera	Gil	1	1	Desarrollo	120.000	6.000
5	6	38382980M	María	Santana	Moreno	1	1	Desarrollo	120.000	6.000
6	1	32481596F	Aarón	Rivero	Gómez	1	1	Desarrollo	120.000	6.000
7	5	17087203C	Marcos	Loyola	Méndez	5	5	I+D	375.000	380.000
8	10	46384486H	Diego	Flores	Salas	5	5	I+D	375.000	380.000
9	8	71651431Z	Pepe	Ruiz	Santana	3	3	Recursos Humanos	280.000	25.000
10	3	R6970642B	Adolfo	Rubio	Flores	3	3	Recursos Humanos	280.000	25.000
11	7	80576669X	Pilar	Ruiz	[NULL]	2	2	Sistemas	150.000	21.000
12	9	56399183D	Juan	Gómez	López	2	2	Sistemas	150.000	21.000
13	2	Y5575632D	Adela	Salas	Díaz	2	2	Sistemas	150.000	21.000

15. Devuelve un listado con los empleados que no tienen ningún departamento asociado y los departamentos que no tienen ningún empleado asociado. Ordene el listado alfabéticamente por el nombre del departamento.

```
SELECT e.*, d.*  
FROM empleado e  
LEFT JOIN departamento d  
ON e.id_departamento = d.id  
WHERE e.id_departamento IS NULL OR d.id IS NULL  
ORDER BY d.nombre;
```

Enter a SQL expression to filter results (use Ctrl+Space)

	123 id	ABC nif	ABC nombre	ABC apellido1	ABC apellido2	123 id_departamento	123 id	ABC nombre	123 presupuesto	123 gastos
1	13	82635162B	Juan Antonio	Sáez	Guerrero	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
2	12	41234836R	Irene	Salas	Flores	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]

• Consultas resumen

1. Calcula la suma del presupuesto de todos los departamentos.

```
SELECT SUM(presupuesto) AS suma_presupuesto  
FROM departamento;
```

	123 suma_presupuesto
1	1.035.000

2. Calcula la media del presupuesto de todos los departamentos.

```
SELECT AVG(presupuesto) as media_presupuesto  
FROM departamento;
```

3. Calcula el valor mínimo del presupuesto de todos los departamentos.

```
SELECT MIN(presupuesto) AS presupuesto_minimo  
FROM departamento;
```


4. Calcula el nombre del departamento y el presupuesto que tiene asignado, del departamento con menor presupuesto.

```
SELECT d.nombre, d.presupuesto
FROM departamento d
INNER JOIN (
    SELECT MIN(presupuesto) as min_presupuesto
    FROM departamento
) t
ON d.presupuesto = t.min_presupuesto
```

5. Calcula el valor máximo del presupuesto de todos los departamentos.

```
SELECT MAX(presupuesto) AS max_presupuesto
FROM departamento;
```

6. Calcula el nombre del departamento y el presupuesto que tiene asignado, del departamento con mayor presupuesto.

```
SELECT nombre, presupuesto
FROM departamento
WHERE presupuesto = (
    SELECT MAX(presupuesto)
    FROM departamento
)
```

7. Calcula el número total de empleados que hay en la tabla empleado.

```
SELECT COUNT(*) AS total_empleados FROM empleado;
```

8. Calcula el número de empleados que no tienen NULL en su segundo apellido.

```
SELECT COUNT(*) as Num_Empleados_Con_Segundo_Apellido  
FROM empleado  
WHERE apellido2 IS NOT NULL;
```

9. Calcula el número de empleados que hay en cada departamento. Tienes que devolver dos columnas, una con el nombre del departamento y otra con el número de empleados que tiene asignados.

```
SELECT d.nombre, COUNT(e.id) AS num_empleados  
FROM departamento d  
LEFT JOIN empleado e ON d.id = e.id_departamento  
GROUP BY d.nombre;
```

10. Calcula el nombre de los departamentos que tienen más de 2 empleados. El resultado debe tener dos columnas, una con el nombre del departamento y otra con el número de empleados que tiene asignados.

```
SELECT d.nombre, COUNT(*) as num_empleados
FROM empleado e
RIGHT JOIN departamento d ON e.id_departamento = d.id
GROUP BY d.nombre
HAVING COUNT(*) > 2
```

11. Calcula el número de empleados que trabajan en cada uno de los departamentos. El resultado de esta consulta también tiene que incluir aquellos departamentos que no tienen ningún empleado asociado.

```
SELECT d.nombre, COUNT(e.id) as numero_empleados
FROM departamento d
LEFT JOIN empleado e ON d.id = e.id_departamento
GROUP BY d.nombre
```

12. Calcula el número de empleados que trabajan en cada uno de los departamentos que tienen un presupuesto mayor a 200000 euros.

```
SELECT d.nombre, COUNT(e.id) AS num_empleados
FROM departamento d
LEFT JOIN empleado e ON d.id = e.id_departamento
WHERE d.presupuesto > 200000
GROUP BY d.nombre;
```

• Subconsultas

1. Devuelve un listado con todos los empleados que tiene el departamento de Sistemas. (Sin utilizar INNER JOIN).

```
SELECT id, nif, nombre, apellido1, apellido2, id_departamento
FROM empleado
WHERE id_departamento = (SELECT id FROM departamento WHERE nombre =
'Sistemas');
```

2. Devuelve el nombre del departamento con mayor presupuesto y la cantidad que tiene asignada.

```
SELECT nombre, presupuesto  
FROM departamento  
WHERE presupuesto = (SELECT MAX(presupuesto) FROM departamento) ;
```

3. Devuelve el nombre del departamento con menor presupuesto y la cantidad que tiene asignada.

```
SELECT nombre, presupuesto  
FROM departamento  
WHERE presupuesto = (SELECT MIN (presupuesto) FROM departamento) ;
```

4. Devuelve el nombre del departamento con mayor presupuesto y la cantidad que tiene asignada. Sin hacer uso de MAX, ORDER BY ni LIMIT.

```
SELECT nombre, presupuesto  
FROM departamento  
WHERE presupuesto = (SELECT presupuesto FROM departamento WHERE  
presupuesto >= ALL(SELECT presupuesto FROM departamento));
```

5. Devuelve el nombre del departamento con menor presupuesto y la cantidad que tiene asignada. Sin hacer uso de MIN, ORDER BY ni LIMIT.

```
SELECT nombre, presupuesto  
FROM departamento  
WHERE presupuesto = (SELECT presupuesto FROM departamento WHERE  
presupuesto >= ALL(SELECT presupuesto FROM departamento));
```

6. Devuelve los nombres de los departamentos que tienen empleados asociados. (Utilizando ALL o ANY).

```
SELECT nombre  
FROM departamento  
WHERE id = ANY (SELECT id_departamento FROM empleado)
```

7. Devuelve los nombres de los departamentos que no tienen empleados asociados. (Utilizando ALL o ANY).

```
SELECT nombre  
FROM departamento  
WHERE 0 = ALL (SELECT COUNT(*)  
                FROM empleado  
                WHERE empleado.id_departamento = departamento.id);
```


8. Devuelve los nombres de los departamentos que tienen empleados asociados. (Utilizando IN o NOT IN).

```
SELECT nombre  
FROM departamento  
WHERE id IN (SELECT id_departamento FROM empleado);
```

9. Devuelve los nombres de los departamentos que no tienen empleados asociados. (Utilizando IN o NOT IN).

```
SELECT nombre  
FROM departamento  
WHERE id NOT IN (SELECT id_departamento FROM empleado);
```

10. Devuelve los nombres de los departamentos que tienen empleados asociados. (Utilizando EXISTS o NOT EXISTS).

```
SELECT nombre
FROM departamento d
WHERE EXISTS (
    SELECT *
    FROM empleado e
    WHERE e.id_departamento = d.id
);
```

11. Devuelve los nombres de los departamentos que no tienen empleados asociados. (Utilizando EXISTS o NOT EXISTS).

```
SELECT nombre
FROM departamento d
WHERE NOT EXISTS (
    SELECT *
    FROM empleado e
    WHERE e.id_departamento = d.id
);
```

TIENDA INFORMATICA

- Consultas sobre una tabla

1. Lista el nombre de todos los productos que hay en la tabla producto.

SELECT nombre **FROM** producto;

2. Lista los nombres y los precios de todos los productos de la tabla producto.

SELECT nombre, precio **FROM** producto;

3. Lista todas las columnas de la tabla producto.

```
DESCRIBE producto;
```

4. Lista el nombre de los productos, el precio en euros y el precio en dólares estadounidenses (USD).

```
SELECT nombre, precio, precio * 1.21 AS precio_usd FROM producto;
```

5. Lista el nombre de los productos, el precio en euros y el precio en dólares estadounidenses (USD). Utiliza los siguientes alias para las columnas: nombre de producto, euros, dólares.

```
SELECT nombre AS 'nombre de producto', precio AS 'euros', precio/1.2  
AS 'dólares' FROM producto;
```

6. Lista los nombres y los precios de todos los productos de la tabla producto, convirtiendo los nombres a mayúscula.

```
SELECT UPPER(nombre) AS nombre, precio FROM producto;
```

7. Lista los nombres y los precios de todos los productos de la tabla producto, convirtiendo los nombres a minúscula.

```
SELECT LOWER(nombre) AS nombre, precio FROM producto;
```

8. Lista el nombre de todos los fabricantes en una columna, y en otra columna obtenga en mayúsculas los dos primeros caracteres del nombre del fabricante.

```
SELECT nombre AS 'Nombre del fabricante', UPPER(SUBSTRING(nombre,1,2))  
AS '2 primeras letras' FROM fabricante;
```

9. Lista los nombres y los precios de todos los productos de la tabla producto, redondeando el valor del precio.

```
SELECT nombre, ROUND(precio) AS precio_redondeado FROM producto;
```

10. Lista los nombres y los precios de todos los productos de la tabla producto, truncando el valor del precio para mostrarlo sin ninguna cifra decimal.

```
SELECT nombre, TRUNCATE(precio, 0) AS precio_truncado FROM producto;
```

11. Lista el identificador de los fabricantes que tienen productos en la tabla producto.

```
SELECT DISTINCT id_fabricante FROM producto;
```

12. Lista el identificador de los fabricantes que tienen productos en la tabla producto, eliminando los identificadores que aparecen repetidos.

```
SELECT DISTINCT f.id FROM fabricante f  
INNER JOIN producto p ON f.id = p.id_fabricante;
```


13. Lista los nombres de los fabricantes ordenados de forma ascendente.

```
SELECT nombre FROM fabricante ORDER BY nombre ASC;
```

14. Lista los nombres de los fabricantes ordenados de forma descendente.

```
SELECT nombre FROM fabricante ORDER BY nombre DESC;
```

15. Lista los nombres de los productos ordenados en primer lugar por el nombre de forma ascendente y en segundo lugar por el precio de forma descendente.

```
SELECT nombre, precio FROM producto ORDER BY nombre ASC, precio DESC;
```

16. Devuelve una lista con las 5 primeras filas de la tabla fabricante.

```
SELECT * FROM fabricante LIMIT 5;
```

17. Devuelve una lista con 2 filas a partir de la cuarta fila de la tabla fabricante. La cuarta fila también se debe incluir en la respuesta.

```
SELECT * FROM fabricante LIMIT 2 OFFSET 3;
```

18. Lista el nombre y el precio del producto más barato. (Utilice solamente las cláusulas ORDER BY y LIMIT)

```
SELECT nombre, precio FROM producto ORDER BY precio ASC LIMIT 1;
```

19. Lista el nombre y el precio del producto más caro. (Utilice solamente las cláusulas ORDER BY y LIMIT)

```
SELECT nombre, precio FROM producto ORDER BY precio DESC LIMIT 1;
```

20. Lista el nombre de todos los productos del fabricante cuyo identificador de fabricante es igual a 2.

```
SELECT nombre FROM producto WHERE id_fabricante = 2;
```

21. Lista el nombre de los productos que tienen un precio menor o igual a 120€.

```
SELECT nombre FROM producto WHERE precio <= 120;
```

22. Lista el nombre de los productos que tienen un precio mayor o igual a 400€.

```
SELECT nombre FROM producto WHERE precio >= 400;
```

23. Lista el nombre de los productos que no tienen un precio mayor o igual a 400€.

```
SELECT nombre FROM producto WHERE precio >= 400;
```

24. Lista todos los productos que tengan un precio entre 80€ y 300€. Sin utilizar el operador BETWEEN.

```
SELECT * FROM producto WHERE precio >= 80 AND precio <= 300;
```

25. Lista todos los productos que tengan un precio entre 60€ y 200€. Utilizando el operador BETWEEN.

```
SELECT * FROM producto WHERE precio BETWEEN 60 AND 200;
```

26. Lista todos los productos que tengan un precio mayor que 200€ y que el identificador de fabricante sea igual a 6.

```
SELECT * FROM producto WHERE precio > 200 AND id_fabricante = 6;
```

27. Lista todos los productos donde el identificador de fabricante sea 1, 3 o 5. Sin utilizar el operador IN.

```
SELECT nombre, precio  
FROM producto  
WHERE id_fabricante = 1 OR id_fabricante = 3 OR id_fabricante = 5;
```

28. Lista todos los productos donde el identificador de fabricante sea 1, 3 o 5. Utilizando el operador IN.

```
SELECT nombre, precio  
FROM producto  
WHERE id_fabricante IN (1, 3, 5);
```


29. Lista el nombre y el precio de los productos en céntimos (Habrá que multiplicar por 100 el valor del precio).

Cree un alias para la columna que contiene el precio que se llame céntimos.

```
SELECT nombre AS "nombre del producto", precio * 100 AS "céntimos"  
FROM producto;
```

30. Lista los nombres de los fabricantes cuyo nombre empiece por la letra S.

```
SELECT nombre  
FROM fabricante  
WHERE nombre LIKE 'S%';
```

31. Lista los nombres de los fabricantes cuyo nombre termine por la vocal e.

```
SELECT nombre  
FROM fabricante  
WHERE nombre LIKE '%e';
```

32. Lista los nombres de los fabricantes cuyo nombre contenga el carácter w.

```
SELECT nombre FROM fabricante WHERE nombre LIKE '%w%';
```

33. Lista los nombres de los fabricantes cuyo nombre sea de 4 caracteres.

```
SELECT nombre  
FROM fabricante  
WHERE LENGTH(nombre) = 4;
```

34. Devuelve una lista con el nombre de todos los productos que contienen la cadena Portátil en el nombre.

```
SELECT nombre  
FROM producto  
WHERE nombre LIKE '%Portátil%';
```

35. Devuelve una lista con el nombre de todos los productos que contienen la cadena Monitor en el nombre y tienen un precio inferior a 215 €.

```
SELECT nombre  
FROM producto  
WHERE nombre LIKE '%Monitor%' AND precio < 215;
```

36. Lista el nombre y el precio de todos los productos que tengan un precio mayor o igual a 180€. Ordene el resultado en primer lugar por el precio (en orden descendente) y en segundo lugar por el nombre (en orden ascendente).

```
SELECT nombre, precio  
FROM producto  
WHERE precio >= 180  
ORDER BY precio DESC, nombre ASC;
```

- Consultas multitabla

1. Devuelve una lista con el nombre del producto, precio y nombre de fabricante de todos los productos de la base de datos.

```
SELECT producto.nombre, producto.precio, fabricante.nombre  
FROM producto  
INNER JOIN fabricante  
ON producto.id_fabricante = fabricante.id;
```

2. Devuelve una lista con el nombre del producto, precio y nombre de fabricante de todos los productos de la base de datos. Ordene el resultado por el nombre del fabricante, por orden alfabético.

```
SELECT producto.nombre, producto.precio, fabricante.nombre  
FROM producto  
INNER JOIN fabricante  
ON producto.id_fabricante = fabricante.id  
ORDER BY fabricante.nombre;
```

3. Devuelve una lista con el identificador del producto, nombre del producto, identificador del fabricante y nombre del fabricante, de todos los productos de la base de datos.

```
SELECT producto.id, producto.nombre, producto.id_fabricante,  
fabricante.nombre  
FROM producto  
INNER JOIN fabricante  
ON producto.id_fabricante = fabricante.id;
```

4. Devuelve el nombre del producto, su precio y el nombre de su fabricante, del producto más barato.

```
SELECT producto.nombre, producto.precio, fabricante.nombre  
FROM producto  
INNER JOIN fabricante  
ON producto.id_fabricante = fabricante.id  
WHERE producto.precio = (SELECT MIN(precio) FROM producto);
```

5. Devuelve el nombre del producto, su precio y el nombre de su fabricante, del producto más caro.

```
SELECT producto.nombre, producto.precio, fabricante.nombre  
FROM producto  
INNER JOIN fabricante  
ON producto.id_fabricante = fabricante.id  
WHERE producto.precio = (SELECT MAX(precio) FROM producto);
```

6. Devuelve una lista de todos los productos del fabricante Lenovo.

```
SELECT producto.nombre, producto.precio, fabricante.nombre  
FROM producto  
INNER JOIN fabricante  
ON producto.id_fabricante = fabricante.id  
WHERE fabricante.nombre = 'Lenovo';
```

7. Devuelve una lista de todos los productos del fabricante Crucial que tengan un precio mayor que 200€.

```
SELECT producto.nombre, producto.precio, fabricante.nombre  
FROM producto  
INNER JOIN fabricante  
ON producto.id_fabricante = fabricante.id  
WHERE fabricante.nombre = 'Crucial' AND producto.precio > 200;
```

8. Devuelve un listado con todos los productos de los fabricantes Asus, Hewlett-Packard y Seagate. Sin utilizar el operador IN.

```
SELECT producto.nombre, producto.precio, fabricante.nombre  
FROM producto  
INNER JOIN fabricante  
ON producto.id_fabricante = fabricante.id  
WHERE fabricante.nombre = 'Asus' OR fabricante.nombre = 'Hewlett-  
Packard' OR fabricante.nombre = 'Seagate';
```


9. Devuelve un listado con todos los productos de los fabricantes Asus, Hewlett-Packard y Seagate. Utilizando el operador IN.

```
SELECT producto.nombre, producto.precio, fabricante.nombre  
FROM producto  
INNER JOIN fabricante  
ON producto.id_fabricante = fabricante.id  
WHERE fabricante.nombre IN ('Asus', 'Hewlett-Packard', 'Seagate');
```

10. Devuelve un listado con el nombre y el precio de todos los productos de los fabricantes cuyo nombre termine por la vocal e.

```
SELECT p.nombre, p.precio, f.nombre AS nombre_fabricante  
FROM producto p  
INNER JOIN fabricante f ON p.id_fabricante = f.id  
WHERE f.nombre LIKE '%e';
```

11. Devuelve un listado con el nombre y el precio de todos los productos cuyo nombre de fabricante contenga el carácter w en su nombre.

```
SELECT p.nombre, p.precio, f.nombre AS nombre_fabricante
FROM producto p
INNER JOIN fabricante f ON p.id_fabricante = f.id
WHERE f.nombre LIKE '%w%';
```

12. Devuelve un listado con el nombre de producto, precio y nombre de fabricante, de todos los productos que tengan un precio mayor o igual a 180€. Ordene el resultado en primer lugar por el precio (en orden descendente) y en segundo lugar por el nombre (en orden ascendente)

```
SELECT p.nombre, p.precio, f.nombre AS nombre_fabricante
FROM producto p
INNER JOIN fabricante f ON p.id_fabricante = f.id
WHERE p.precio >= 180
ORDER BY p.precio DESC, p.nombre ASC;
```

13. Devuelve un listado con el identificador y el nombre de fabricante, solamente de aquellos fabricantes que tienen productos asociados en la base de datos.

```
SELECT f.id, f.nombre  
FROM fabricante f  
INNER JOIN producto p ON f.id = p.id_fabricante  
GROUP BY f.id, f.nombre;
```

14. Devuelve un listado de todos los fabricantes que existen en la base de datos, junto con los productos que tiene cada uno de ellos. El listado deberá mostrar también aquellos fabricantes que no tienen productos asociados.

```
SELECT f.id, f.nombre, COUNT(p.id) AS cantidad_productos  
FROM fabricante f  
LEFT JOIN producto p ON f.id = p.id_fabricante  
GROUP BY f.id, f.nombre;
```

15. Devuelve un listado donde sólo aparezcan aquellos fabricantes que no tienen ningún producto asociado.

```
SELECT f.id, f.nombre  
FROM fabricante f  
LEFT JOIN producto p ON f.id = p.id_fabricante  
WHERE p.id IS NULL;
```

- Consultas resumen

1. Calcula el número total de productos que hay en la tabla productos.

```
SELECT COUNT(*) AS total_productos FROM producto;
```

2. Calcula el número total de fabricantes que hay en la tabla fabricante.

```
SELECT COUNT(*) AS total_fabricantes FROM fabricante;
```

3. Calcula el número de valores distintos de identificador de fabricante aparecen en la tabla productos.

```
SELECT COUNT(DISTINCT id_fabricante) AS num_fabricantes FROM producto;
```

4. Calcula la media del precio de todos los productos.

```
SELECT AVG(precio) AS media_precio FROM producto;
```

5. Calcula el precio más barato de todos los productos.

```
SELECT MIN(precio) AS precio_minimo FROM producto;
```

6. Calcula el precio más caro de todos los productos.

```
SELECT MAX(precio) AS precio_maximo FROM producto;
```

7. Lista el nombre y el precio del producto más barato.

```
SELECT nombre, precio FROM producto WHERE precio = (SELECT MIN(precio)  
FROM producto);
```

8. Lista el nombre y el precio del producto más caro.

```
SELECT nombre, precio FROM producto WHERE precio = (SELECT MAX(precio)  
FROM producto) ;
```

9. Calcula la suma de los precios de todos los productos.

```
SELECT SUM(precio) AS total_precios FROM producto;
```


10. Calcula el número de productos que tiene el fabricante Asus.

```
SELECT COUNT(*) AS num_productos FROM producto p JOIN fabricante f ON  
p.id_fabricante = f.id WHERE f.nombre = 'Asus';
```

11. Calcula la media del precio de todos los productos del fabricante Asus.

```
SELECT AVG(precio) FROM producto WHERE id_fabricante = (SELECT id_fabricante FROM  
fabricante WHERE nombre = 'Asus');
```

12. Calcula el precio más barato de todos los productos del fabricante Asus.

```
SELECT MIN(precio) FROM producto WHERE id_fabricante = (SELECT id_fabricante FROM fabricante WHERE nombre = 'Asus');
```

13. Calcula el precio más caro de todos los productos del fabricante Asus.

```
SELECT MAX(precio) FROM producto WHERE id_fabricante = (SELECT id_fabricante FROM fabricante WHERE nombre = 'Asus');
```

14. Calcula la suma de todos los productos del fabricante Asus.

```
SELECT SUM(precio) FROM producto WHERE id_fabricante = (SELECT id_fabricante FROM fabricante WHERE nombre = 'Asus');
```

15. Muestra el precio máximo, precio mínimo, precio medio y el número total de productos que tiene el fabricante Crucial.

```
SELECT MAX(precio) AS maximo, MIN(precio) AS minimo, AVG(precio) AS promedio, COUNT(*) AS total FROM producto WHERE id_fabricante = (SELECT id_fabricante FROM fabricante WHERE nombre = 'Crucial');
```

16. Muestra el número total de productos que tiene cada uno de los fabricantes. El listado también debe incluir los fabricantes que no tienen ningún producto. El resultado mostrará dos columnas, una con el nombre del fabricante y otra con el número de productos que tiene. Ordene el resultado descendientemente por el número de productos.

```
SELECT fabricante.nombre, COUNT(producto.id) AS num_productos FROM fabricante LEFT  
JOIN producto ON fabricante.id = producto.id_fabricante GROUP BY fabricante.nombre  
ORDER BY num_productos DESC;
```

17. Muestra el precio máximo, precio mínimo y precio medio de los productos de cada uno de los fabricantes. El resultado mostrará el nombre del fabricante junto con los datos que se solicitan.

```
SELECT fabricante.nombre, MAX(producto.precio) AS maximo, MIN(producto.precio) AS  
minimo, AVG(producto.precio) AS promedio FROM fabricante INNER JOIN producto ON  
fabricante.id = producto.id_fabricante GROUP BY fabricante.nombre;
```

18. Muestra el precio máximo, precio mínimo, precio medio y el número total de productos de los fabricantes que tienen un precio medio superior a 200€. No es necesario mostrar el nombre del fabricante, con el identificador del fabricante es suficiente.

```
SELECT fabricante.nombre, MAX(producto.precio) AS maximo, MIN(producto.precio) AS minimo, AVG(producto.precio) AS promedio FROM fabricante INNER JOIN producto ON fabricante.id = producto.id_fabricante GROUP BY fabricante.nombre;
```

19. Muestra el nombre de cada fabricante, junto con el precio máximo, precio mínimo, precio medio y el número total de productos de los fabricantes que tienen un precio medio superior a 200€. Es necesario mostrar el nombre del fabricante.

```
SELECT id_fabricante, MAX(precio) AS maximo, MIN(precio) AS minimo, AVG(precio) AS promedio, COUNT(*) AS total FROM producto GROUP BY id_fabricante HAVING promedio > 200;
```

20. Calcula el número de productos que tienen un precio mayor o igual a 180€.

```
SELECT COUNT(*) FROM producto WHERE precio >= 180;
```

21. Calcula el número de productos que tiene cada fabricante con un precio mayor o igual a 180€.

```
SELECT fabricante.nombre, COUNT(producto.id) AS num_productos  
FROM fabricante  
LEFT JOIN producto ON fabricante.id = producto.id_fabricante  
WHERE precio >= 180  
GROUP BY fabricante.nombre;
```

22. Lista el precio medio los productos de cada fabricante, mostrando solamente el identificador del fabricante.

```
SELECT id_fabricante, AVG(precio) AS precio_medio
FROM producto
GROUP BY id_fabricante;
```

23. Lista el precio medio los productos de cada fabricante, mostrando solamente el nombre del fabricante.

```
SELECT f.nombre, AVG(p.precio) AS precio_medio
FROM producto p
JOIN fabricante f ON p.id_fabricante = f.id
GROUP BY f.nombre;
```

24. Lista los nombres de los fabricantes cuyos productos tienen un precio medio mayor o igual a 150€.

```
SELECT f.nombre, AVG(p.precio) AS precio_medio
FROM producto p
JOIN fabricante f ON p.id_fabricante = f.id
GROUP BY f.nombre
HAVING precio_medio >= 150;
```

25. Devuelve un listado con los nombres de los fabricantes que tienen 2 o más productos.

```
SELECT f.nombre
FROM fabricante f
JOIN producto p ON f.id = p.id_fabricante
GROUP BY f.nombre
HAVING COUNT(*) >= 2;
```


26. Devuelve un listado con los nombres de los fabricantes y el número de productos que tiene cada uno con un precio superior o igual a 220 €. No es necesario mostrar el nombre de los fabricantes que no tienen productos que cumplan la condición.

```
SELECT f.nombre, COUNT(*) AS num_productos
FROM fabricante f
JOIN producto p ON f.id = p.id_fabricante
WHERE p.precio >= 220
GROUP BY f.nombre;
```

27. Devuelve un listado con los nombres de los fabricantes y el número de productos que tiene cada uno con un precio superior o igual a 220 €. El listado debe mostrar el nombre de todos los fabricantes, es decir, si hay algún fabricante que no tiene productos con un precio superior o igual a 220€ deberá aparecer en el listado con un valor igual a 0 en el número de productos.

```
SELECT f.nombre, COUNT(p.id) AS num_productos
FROM fabricante f
LEFT JOIN producto p ON f.id = p.id_fabricante AND p.precio >= 220
GROUP BY f.nombre;
```

28. Devuelve un listado con los nombres de los fabricantes donde la suma del precio de todos sus productos es superior a 1000 €.

```
SELECT f.nombre, SUM(p.precio) AS total_precio
FROM fabricante f
JOIN producto p ON f.id = p.id_fabricante
GROUP BY f.nombre
HAVING total_precio > 1000;
```

29. Devuelve un listado con el nombre del producto más caro que tiene cada fabricante. El resultado debe tener tres columnas: nombre del producto, precio y nombre del fabricante. El resultado tiene que estar ordenado alfabéticamente de menor a mayor por el nombre del fabricante.

```
SELECT p.nombre AS nombre_producto, p.precio, f.nombre AS nombre_fabricante
FROM producto p
JOIN fabricante f ON p.id_fabricante = f.id
WHERE p.precio = (
    SELECT MAX(p2.precio)
    FROM producto p2
    WHERE p2.id_fabricante = p.id_fabricante
)
ORDER BY f.nombre ASC;
```

- Subconsultas

1. Devuelve todos los productos del fabricante Lenovo. (Sin utilizar INNER JOIN).

```
SELECT *  
FROM producto  
WHERE id_fabricante = (SELECT id_fabricante FROM fabricante WHERE nombre =  
'Lenovo');
```

2. Devuelve todos los datos de los productos que tienen el mismo precio que el producto más caro del fabricante Lenovo. (Sin utilizar INNER JOIN).

```
SELECT *  
FROM producto  
WHERE precio = (SELECT MAX(precio) FROM producto WHERE id_fabricante = (SELECT  
id_fabricante  
FROM fabricante WHERE nombre = 'Lenovo'));
```

3. Lista el nombre del producto más caro del fabricante Lenovo.

```
SELECT nombre
FROM producto
WHERE precio = (SELECT MAX(precio) FROM producto
WHERE id_fabricante = (SELECT id_fabricante
FROM fabricante WHERE nombre = 'Lenovo'));
```

4. Lista el nombre del producto más barato del fabricante Hewlett-Packard.

```
SELECT nombre
FROM producto
WHERE precio = (SELECT MIN(precio)
FROM producto
WHERE id_fabricante = (SELECT id_fabricante
FROM fabricante
WHERE nombre = 'Hewlett-Packard'));
```

5. Devuelve todos los productos de la base de datos que tienen un precio mayor o igual al producto más caro del fabricante Lenovo.

[illegible]

6. Lista todos los productos del fabricante Asus que tienen un precio superior al precio medio de todos sus productos.

[illegible]

7. Devuelve el producto más caro que existe en la tabla producto sin hacer uso de MAX, ORDER BY ni LIMIT.

```
SELECT *  
FROM producto  
WHERE precio >= ALL (SELECT precio FROM producto);
```

8. Devuelve el producto más barato que existe en la tabla producto sin hacer uso de MIN, ORDER BY ni LIMIT.

```
SELECT *  
FROM producto  
WHERE precio <= ALL (SELECT precio FROM producto);
```

9. Devuelve los nombres de los fabricantes que tienen productos asociados. (Utilizando ALL o ANY).

```
SELECT nombre  
FROM fabricante  
WHERE id = ANY (SELECT id_fabricante  
FROM producto);
```

10. Devuelve los nombres de los fabricantes que no tienen productos asociados. (Utilizando ALL o ANY).

```
SELECT nombre  
FROM fabricante  
WHERE id <> ALL (SELECT id_fabricante FROM producto);
```

11. Devuelve los nombres de los fabricantes que tienen productos asociados. (Utilizando IN o NOT IN).

```
SELECT nombre  
FROM fabricante  
WHERE id IN (SELECT id_fabricante FROM producto) ;
```

12. Devuelve los nombres de los fabricantes que no tienen productos asociados. (Utilizando IN o NOT IN).

```
SELECT nombre  
FROM fabricante  
WHERE id NOT IN (SELECT id_fabricante FROM producto)
```


13. Devuelve los nombres de los fabricantes que tienen productos asociados. (Utilizando EXISTS o NOT EXISTS).

```
SELECT nombre  
FROM fabricante  
WHERE EXISTS (SELECT id_fabricante FROM producto WHERE  
producto.id_fabricante = fabricante.id)
```

14. Devuelve los nombres de los fabricantes que no tienen productos asociados. (Utilizando EXISTS o NOT EXISTS).

```
SELECT nombre  
FROM fabricante  
WHERE NOT EXISTS (SELECT id_fabricante FROM producto WHERE  
producto.id_fabricante = fabricante.id)
```

15. Lista el nombre de cada fabricante con el nombre y el precio de su producto más caro.

```
SELECT f.nombre, p.nombre, p.precio
FROM fabricante f
JOIN producto p ON f.id = p.id_fabricante
WHERE p.precio = (
SELECT MAX(precio)
FROM producto
WHERE id_fabricante = f.id
);
```

16. Devuelve un listado de todos los productos que tienen un precio mayor o igual a la media de todos los productos de su mismo fabricante.

```
SELECT p.nombre, p.precio
FROM producto p
WHERE p.precio >= (
SELECT AVG(p2.precio)
FROM producto p2
WHERE p2.id_fabricante = p.id_fabricante
)
```

17. Lista el nombre del producto más caro del fabricante Lenovo.

```
SELECT nombre
FROM producto
WHERE id_fabricante = (SELECT id_fabricante FROM fabricante WHERE nombre = 'Lenovo')
AND precio = (SELECT MAX(precio) FROM producto WHERE id_fabricante = (SELECT
id_fabricante FROM fabricante WHERE nombre= 'Lenovo'));
```

18. Devuelve un listado con todos los nombres de los fabricantes que tienen el mismo número de productos que el fabricante Lenovo.

```
SELECT f.nombre
FROM fabricante f
WHERE (SELECT COUNT(*) FROM producto p WHERE p.id_fabricante = f.id) =
(SELECT COUNT(*) FROM producto p JOIN fabricante f2 ON p.id_fabricante = f2.id WHERE
f2.nombre = 'Lenovo')
```