Consultas gestión

Ángel Leonardo Cáceres Quintero 02210131007

Sistema gestión de base de datos.

Facultad Ing. De sistemas

Universidad De Santander

1. 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 p
ORDER BY fecha DESC;

CREATE PROCEDURE listado_pedidos()
BEGIN
    SELECT *
    FROM pedido p
    ORDER BY fecha DESC;
END
```

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

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

CREATE PROCEDURE pedidos_mayor_valor()
BEGIN
    SELECT *
FROM pedido p
ORDER BY total DESC
LIMIT 2;

END
```

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(c.id)

FROM cliente c

JOIN pedido p ON p.id_cliente = c.id
```

```
CREATE PROCEDURE listado_clientes()

BEGIN

    SELECT DISTINCT(c.id)

FROM cliente c

JOIN pedido p ON p.id_cliente = c.id

END
```

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 p
WHERE YEAR(fecha) = '2017' AND total > 500
ORDER BY total DESC;

CREATE PROCEDURE pedidos_2017_sobre_500()
BEGIN
    SELECT *
FROM pedido p
WHERE YEAR(fecha) = '2017' AND total > 500
ORDER BY total DESC;

END
```

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

```
SELECT CONCAT(c.nombre, ' ', apellido1, ' ', apellido2) as Comerciales, c.comisión
FROM comercial c
WHERE comisión BETWEEN 0.05 AND 0.11;

CREATE PROCEDURE comerciantes_comision_rango()

BEGIN
    SELECT CONCAT(c.nombre, ' ', apellido1, ' ', apellido2) as Comerciales, c.comisión
FROM comercial c
WHERE comisión BETWEEN 0.05 AND 0.11;

END
```

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

```
SELECT comisión
FROM comercial c
ORDER BY comisión DESC
Limit 1;

CREATE PROCEDURE obtener_comision_maxima()
BEGIN
  SELECT comisión
  FROM comercial c
  ORDER BY comisión DESC
  LIMIT 1;
END
```

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 c.id, CONCAT(c.nombre, ' ', c.apellido1, ' ', c.apellido2) as Nombre_Cliente
FROM cliente c
WHERE apellido2 IS NOT NULL
ORDER BY c.nombre ASC, apellido1 ASC, apellido2 ASC;

CREATE PROCEDURE listado_clientes_not_null()
BEGIN
    SELECT c.id, CONCAT(c.nombre, ' ', c.apellido1, ' ', c.apellido2) as Nombre_Cliente
FROM cliente c
WHERE apellido2 IS NOT NULL
ORDER BY c.nombre ASC, apellido1 ASC, apellido2 ASC;

END
```

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 c.nombre
FROM cliente c
WHERE c.nombre LIKE 'A%' AND c.nombre LIKE '%n'
ORDER BY c.nombre ASC;
```

```
CREATE PROCEDURE listado_nombres_A_N()
BEGIN
SELECT c.nombre
FROM cliente c
WHERE c.nombre LIKE 'A%' AND c.nombre LIKE '%n'
ORDER BY c.nombre ASC;

END
```

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

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

CREATE PROCEDURE listado_clientes_no_A()
BEGIN
    SELECT c.nombre
FROM cliente c
WHERE c.nombre NOT LIKE 'A%'
ORDER BY c.nombre;

END
```

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 c

WHERE nombre like '%o';


CREATE PROCEDURE listado_comerciantes_no_O_ni_repetidos ()

BEGIN

    SELECT DISTINCT(nombre)

FROM comercial c

WHERE nombre like '%o';

END
```

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 repetidos.

SELECT DISTINCT(c.id), c.nombre, **CONCAT**(apellido1, ' ', apellido2)

FROM cliente c

JOIN pedido p **ON** c.id = p.id_cliente

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

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.*

FROM pedido p

JOIN cliente c **ON** c.id = p.id_cliente

ORDER BY c.id **ASC**, c.nombre **ASC**, c.apellido1 **ASC**, c.apellido2 **ASC**, c.ciudad **ASC**, c.categoría **ASC**;

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.*

FROM pedido p

JOIN comercial c **ON** c.id = p.id_comercial

ORDER BY c.id **ASC**, c.nombre **ASC**, c.apellido1 **ASC**, c.apellido2 **ASC**, c.comisión **ASC**;

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 *, p.*, c2.*
FROM cliente c
JOIN pedido p ON p.id_cliente = c.id
JOIN comercial c2 ON c2.id = p.id_comercial
ORDER BY c.id;
```

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

```
SELECT *, p.total
FROM cliente c
JOIN pedido p ON p.id_cliente = c.id
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 c.nombre, CONCAT(c.apellido1, ' ', c.apellido2) as apellidos
FROM comercial c
JOIN pedido p ON p.id_comercial = c.id
JOIN cliente c2 ON c2.id = p.id_cliente
WHERE c2.nombre = 'María' AND c2.apellido1 = 'Santana' AND c2.apellido2 = 'Moreno';
```

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 p.id_cliente = c.id
JOIN comercial c2 ON c2.id = p.id_comercial
WHERE c2.nombre = 'Daniel' AND c2.apellido1 = 'Sáez' AND c2.apellido2 = 'Vega';
```

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.*, p.*
FROM cliente c
LEFT JOIN pedido p ON p.id_cliente = c.id
ORDER BY c.apellido1 ASC, c.apellido2 ASC, c.nombre ASC;
```

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.*, p.*
FROM comercial c
LEFT JOIN pedido p ON p.id_comercial = c.id
ORDER BY c.apellido1 ASC, c.apellido2 ASC, c.nombre ASC;
```

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

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

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

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

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 en listado deberá diferenciar de algún modo los clientes y los comerciales.

```
SELECT 'Cliente' AS tipo, c.nombre AS cliente, CONCAT(c.apellido1, ' ', c.apellido2) AS apellidos
FROM cliente c
WHERE c.id NOT IN (SELECT id_cliente FROM pedido)
UNION ALL
SELECT 'Comercial' AS tipo, co.nombre AS comercial, CONCAT(co.apellido1, ' ', co.apellido2) AS apellidos
FROM comercial co
WHERE co.id NOT IN (SELECT id_comercial FROM pedido)
ORDER BY apellidos, cliente;
```

Consultas resumen

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

```
SELECT CONCAT('$ ', FORMAT(SUM(p.total),2) ) as total_pedidos
FROM pedido p;
```

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

```
SELECT CONCAT('$ ', FORMAT(AVG(total),2) ) as Promedio_pedidos
FROM pedido p;
```

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

```
SELECT COUNT(DISTINCT(p.id_comercial)) as total_comerciales
FROM pedido p;
```

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

```
SELECT COUNT(DISTINCT(c.id)) as total_clientes
FROM cliente c;
```

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

```
SELECT MAX(p.total) as Mayor_cantidad
FROM pedido p
limit 1;
```

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

```
SELECT MIN(p.total) as Mayor_cantidad
FROM pedido p
limit 1;
```

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

```
SELECT c.ciudad, MAX(c.categoría) as categoría_maxima
FROM cliente c
group by c.ciudad;
```

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, c.nombre, CONCAT(c.apellido1, ' ', c.apellido2) as apellidos, p.fecha, MAX(p.total) AS Total_maximo
FROM cliente c
JOIN pedido p ON p.id_cliente = c.id
GROUP BY c.id, DAY(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 c.id, c.nombre, CONCAT(c.apellido1, ' ', c.apellido2) as apellidos, p.fecha, MAX(p.total) AS Total_maximo
FROM cliente c
JOIN pedido p ON p.id_cliente = c.id
GROUP BY c.id, DAY(p.fecha)
HAVING Total_maximo > 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 c.id, c.nombre, CONCAT(c.apellido1, ' ',c.apellido2) as apellidos, MAX(p.total) as total_maximo
FROM comercial c
JOIN pedido p ON p.id_comercial = c.id
WHERE p.fecha = '2016-08-17'
GROUP BY c.id;
```

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, CONCAT(c.apellido1, ' ', c.apellido2) as apellidos, COUNT(p.id) as total_pedidos
FROM cliente c
LEFT JOIN pedido p ON p.id_cliente = c.id
GROUP BY c.id;
```

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, CONCAT(c.apellido1, ' ', c.apellido2) as apellidos, COUNT(p.id) as total_pedidos
FROM cliente c
LEFT JOIN pedido p ON p.id_cliente = c.id
WHERE YEAR(p.fecha) = '2017'
GROUP BY c.id;
```

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, IFNULL(CONCAT(c.apellido1, ' ', c.apellido2), '-----') as apellidos, IFNULL(MAX(p.total), 0) as Maximo_registrado
FROM cliente c
LEFT JOIN pedido p ON p.id_cliente = c.id
GROUP BY c.id;
```

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

```
SELECT YEAR(p.fecha) as Año, IFNULL(MAX(p.total), 0) as valor_maximo
FROM pedido p
GROUP BY Año
Order by valor_maximo;
```

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

```
SELECT YEAR(p.fecha) as Año, COUNT(p.id) as Pedidos_realizados
FROM pedido p
GROUP BY Año
ORDER BY Año DESC;
```

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(p.id) AS Cantidad_pedidos_Daniel
FROM pedido p
WHERE p.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 c.*
FROM cliente c
WHERE c.id = (SELECT p.id_cliente FROM pedido p WHERE YEAR(p.fecha) = '2019' ORDER BY p.total DESC LIMIT 1);
```

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

```
SELECT p.fecha , MIN(p.total) as pedido_minimo
FROM pedido p
WHERE id_cliente = ( SELECT id  FROM cliente  WHERE nombre = 'Pepe'  AND apellido1 = 'Ruiz' AND apellido2 = 'Santana');
```

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.*, p.*
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);
```

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 = (SELECT MAX(total) FROM pedido);
```

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

SELECT *

FROM cliente

WHERE NOT id = **ANY**(**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 c

WHERE NOT id = **ANY**(**SELECT** id_comercial **FROM** pedido)

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

SELECT *

FROM cliente **c**

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 c.*

FROM comercial c

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

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

Gestión empleados

- **Consultas sobre una tabla**

1. Lista el primer apellido de todos los empleados.

```
SELECT e.apellido1 as primer_apellido
FROM empleado e;
```

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

```
SELECT DISTINCT(e.apellido1) as primer_apellido
FROM empleado e;
```

3. Lista todas las columnas de la tabla empleado.

```
SELECT *
FROM empleado e;
```

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

```
SELECT e.nombre, IFNULL(CONCAT(e.apellido1, ' ', e.apellido2), '----') as apellidos
FROM empleado e;
```

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

```
SELECT IFNULL(e.id_departamento, 0) as id_departamento
FROM empleado e;
```

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

```
SELECT DISTINCT(e.id_departamento) as id_departamento
FROM empleado e;
```

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

```
SELECT IFNULL(CONCAT(e.nombre, ' ', e.apellido1, ' ', e.apellido2), '----') as Nombre_completo
FROM empleado e;
```

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

```
SELECT IFNULL(UPPER(CONCAT(e.nombre, ' ', e.apellido1, ' ', e.apellido2)),'----') as
Nombre_completo
FROM empleado e;
```

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

SELECT IFNULL(**LOWER**(**CONCAT**(e.nombre, ' ', e.apellido1, ' ', e.apellido2)),'----') **as** Nombre_completo

FROM empleado e;

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 e.id , **SUBSTRING**(nif, 1, **LENGTH**(nif)-1) **AS** nif_digitos, **RIGHT**(nif, 1) **AS** nif_letra

FROM empleado e

Order by e.id;

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 d.nombre, (d.presupuesto - d.gastos) **as** presupuesto_actual

FROM departamento d;

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

SELECT d.nombre, (d.presupuesto - d.gastos) **as** presupuesto_actual

FROM departamento d

order by presupuesto_actual **ASC**;

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

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

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

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

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 IFNULL(CONCAT(e.apellido1, ' ', e.apellido2), '---') as apellidos, e.nombre
FROM empleado e
ORDER BY apellidos ASC, e.nombre ASC;
```

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

```
SELECT d.nombre, d.presupuesto
FROM departamento d
ORDER BY d.presupuesto DESC
limit 3;
```

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

```
SELECT d.nombre, d.presupuesto as menor_presupuesto
FROM departamento d
WHERE d.presupuesto > 0
ORDER BY menor_presupuesto ASC
LIMIT 3;
```

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

```
SELECT d.nombre, d.gastos
FROM departamento d
WHERE gastos > 0
ORDER BY d.gastos ASC
LIMIT 2;
```

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

```
SELECT d.nombre, d.gastos
FROM departamento d
WHERE gastos > 0
ORDER BY d.gastos DESC
LIMIT 2;
```

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 5 OFFSET 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 d.nombre, d.presupuesto
FROM departamento d
WHERE d.presupuesto >= 150000;
```

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

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

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

```
SELECT d.nombre, d.presupuesto
FROM departamento d
WHERE presupuesto >= 100000 AND presupuesto <= 200000;
```

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 d.nombre, d.presupuesto

FROM departamento d

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

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

SELECT d.nombre, d.presupuesto

FROM departamento d

WHERE presupuesto **BETWEEN** 100000 **AND** 200000;

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 d.nombre, d.presupuesto

FROM departamento d

WHERE presupuesto **NOT BETWEEN** 100000 **AND** 200000;

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

SELECT d.nombre, d.gastos, d.presupuesto

FROM departamento d

WHERE d.gastos > d.presupuesto;

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 d.nombre, d.gastos, d.presupuesto
FROM departamento d
WHERE d.gastos < d.presupuesto;
```

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 d.nombre, d.gastos, d.presupuesto
FROM departamento d
WHERE d.gastos = d.presupuesto;
```

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

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

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

```
SELECT *
FROM empleado e
WHERE e.apellido2 IS NOT NULL;
```

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

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

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

```
SELECT *
FROM empleado e
WHERE e.apellido2 = 'López' OR 'Moreno';
```

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

```
SELECT *
FROM empleado e
WHERE e.apellido2 = 'López' OR 'Moreno';
```

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

```
SELECT e.nombre, CONCAT(e.apellido1, ' ', e.apellido2) as apellidos, e.nif
FROM empleado e
WHERE e.id_departamento = 3;
```

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

```
SELECT e.nombre, CONCAT(e.apellido1, ' ', e.apellido2) as apellidos, e.nif
FROM empleado e
WHERE e.id_departamento = 2 OR 4 OR 5;
```

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 d.id = e.id_departamento
```

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 d.nombre, e.apellido1 as apellido_1, e.apellido2 as apellido_2, e.nombre
FROM departamento d
JOIN empleado e ON e.id_departamento = d.id
ORDER BY d.nombre ASC, apellido_1 ASC, apellido_2 ASC, e.nombre ASC;
```

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
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 DISTINCT(d.id), d.nombre, (d.presupuesto - d.gastos) as presupuesto_actual
FROM departamento d
JOIN empleado e ON e.id_departamento = d.id;
```

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

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

6. 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 departamento d
JOIN empleado e ON e.id_departamento = d.id
WHERE e.apellido2 IS NULL;
```

7. 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.*
FROM empleado e
LEFT JOIN departamento d ON d.id = e.id_departamento
```

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

```
SELECT e.*
FROM empleado e
WHERE e.id_departamento IS NULL;
```

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

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

10. 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

INNER JOIN departamento d **ON** e.id_departamento = d.id

WHERE d.id **NOT IN** (**SELECT** id **FROM** departamento **WHERE** presupuesto **BETWEEN** 100000 **AND** 200000);

11. 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 departamento d

JOIN empleado e **ON** d.id = e.id_departamento

WHERE e.apellido2 **IS NULL**;

12. 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.*

FROM empleado e

LEFT JOIN departamento d

ON e.id_departamento = d.id;

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

SELECT *

FROM empleado

WHERE id **NOT IN** (**SELECT** id_empleado **FROM** empleado_departamento);

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.nombre **AS** departamento_nombre, d.presupuesto, d.gastos

FROM empleado e

LEFT JOIN departamento d **ON** e.id_departamento = d.id

ORDER BY d.nombre;

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 d.nombre **AS** nombre_departamento**,** e.nombre **AS** nombre_empleado**,** e.apellido1**,** e.apellido2**,** e.nif

FROM departamento **d**

LEFT JOIN empleado **e ON d.id = e.id_**departamento **OR e.id_**departamento **IS NULL**

WHERE d.id IS NULL OR e.id_departamento **IS NULL**

ORDER BY d.nombre**;**

- **Consultas resumen**

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

SELECT SUM(d.presupuesto) **as** suma_presupuesto

FROM departamento d;

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

```
SELECT FORMAT(AVG(d.presupuesto), 2) as media_presupuesto
FROM departamento d;
```

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

```
SELECT MIN(d.presupuesto) as minimo_presupuesto
FROM departamento d;
```

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

```
SELECT d.nombre ,MIN(d.presupuesto) as minimo_presupuesto
FROM departamento d
GROUP BY d.id;
```

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

```
SELECT MAX(d.presupuesto) as minimo_presupuesto
FROM departamento d;
```

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

```
SELECT d.nombre, MAX(d.presupuesto) AS presupuesto_maximo
FROM departamento d;
```

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

```
SELECT COUNT(e.id) as total_empleados
FROM empleado e
```

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

```
SELECT COUNT(e.id) as total_empleados
FROM empleado e
WHERE e.apellido2 IS 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 COUNT(e.id) as total_empleados, d.nombre
FROM empleado e
JOIN departamento d 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 COUNT(e.id) as total_empleados, d.nombre
FROM empleado e
JOIN departamento d ON d.id = e.id_departamento
GROUP BY d.nombre
HAVING total_empleados > 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 COUNT(e.id) as total_empleados, d.nombre
FROM empleado e
LEFT JOIN departamento d ON d.id = e.id_departamento
GROUP BY d.nombre;
```

12. Calcula el número de empleados que trabajan en cada unos 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 *
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 d.nombre, MAX(d.presupuesto)

FROM d.departamento

ORDER BY d.presupuesto **DESC**

LIMIT 1;

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

SELECT d.nombre, d.presupuesto

FROM departamento d

WHERE d.presupuesto = (**SELECT MIN**(d.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 d.nombre, **SUM**(d.presupuesto) **as** presupuesto_total

FROM departamento d

WHERE (**SELECT SUM**(presupuesto) **FROM** departamento) = (**SELECT SUM**(presupuesto) **FROM** departamento **WHERE** presupuesto <= d.presupuesto)

GROUP BY d.nombre;

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 d.nombre, d.presupuesto **as** presupuesto_total

FROM departamento d

WHERE d.presupuesto = (**SELECT MIN**(presupuesto) **FROM** departamento)

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

```
SELECT d.nombre
FROM departamento d
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 nombre NOT IN (SELECT d.nombre
FROM departamento d
INNER JOIN empleado e ON d.id = e.id_departamento)
```

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

```
SELECT d.nombre
FROM departamento d
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 d.nombre
FROM departamento d
WHERE id NOT IN (SELECT DISTINCT id_departamento FROM empleado)
```

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

SELECT d.nombre

FROM departamento d

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

11. Devuelve los nombres de los departamentos que 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 *

FROM producto p

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

SELECT p.nombre, p.precio

FROM producto p

3. Lista todas las columnas de la tabla producto.

SELECT *

FROM producto p

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

```
SELECT p.nombre, CONCAT('$ ', p.precio) AS precio_USD, CONCAT('€ ', FORMAT(p.precio*0.92, 2)) as precio_EUR
FROM producto p
```

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 p.nombre as nombre_de_producto, CONCAT('$ ', p.precio) AS dólares, CONCAT('€ ', FORMAT(p.precio*0.92, 2)) as euro
FROM producto p
```

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

```
SELECT UPPER(p.nombre) , p.precio
FROM producto p
```

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

```
SELECT LOWER(p.nombre) , p.precio
FROM producto p
```

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 f.nombre, UPPER(CONCAT(LEFT(f.nombre, 2)))
FROM fabricante f
```

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

SELECT nombre, **CONCAT**('$ ', **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 id_fabricante

FROM producto;

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 nombre, precio

FROM producto

WHERE precio >= 80 **AND** precio <= 300;

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

SELECT nombre, precio

FROM producto

WHERE precio **BETWEEN** 60 **AND** 200;

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

SELECT nombre, precio

FROM producto

WHERE precio > 200 **AND** id_fabricante = 6;

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

SELECT nombre

FROM producto

WHERE id_fabricante = 1 **OR** id_fabricante = 3 **OR** id_fabricante = 5;

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

SELECT nombre

FROM producto

WHERE id_fabricante **IN** (1, 3, 5);

28. 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, precio*100 **AS** centimos

FROM producto;

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

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

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

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

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

SELECT nombre

FROM fabricante

WHERE nombre **LIKE** '%w%'

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

SELECT nombre

FROM fabricante

WHERE LENGTH(nombre) = 4;

33. 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%'

34. 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

35. 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 p.nombre, p.precio, f.nombre **AS** fabricante

FROM producto p

JOIN fabricante f **ON** p.id_fabricante = f.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 p.nombre **AS** producto, p.precio, f.nombre **AS** fabricante

FROM producto p

JOIN fabricante f **ON** p.id_fabricante = f.id

ORDER BY f.nombre **ASC**;

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 p.id, p.nombre, p.precio, f.id, f.nombre

FROM producto p

INNER JOIN fabricante f

ON p.id_fabricante = f.id;

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

SELECT p.nombre, p.precio, f.nombre

FROM producto p

JOIN fabricante f **ON** p.id_fabricante = f.id

WHERE p.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 p.nombre **AS** 'Nombre del producto', p.precio **AS** 'Precio', f.nombre **AS** 'Nombre del fabricante'

FROM producto p

JOIN fabricante f **ON** p.id_fabricante = f.id

WHERE p.precio = (**SELECT MAX**(precio) **FROM** producto);

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

```
SELECT *
FROM producto p
JOIN fabricante f ON p.id_fabricante = f.id
WHERE f.nombre = 'Lenovo';
```

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

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

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

```
SELECT *
FROM producto p
JOIN fabricante f ON p.id_fabricante = f.id
WHERE f.nombre = 'Asus' OR f.nombre = 'Hewlett-Packard' OR f.nombre = 'Seagate';
```

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

```
SELECT *
FROM producto p
JOIN fabricante f ON p.id_fabricante = f.id
WHERE f.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
FROM producto p
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
FROM producto p
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 AS 'Nombre del producto', p.precio AS 'Precio', f.nombre AS 'Nombre del fabricante'

FROM producto p

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 AS 'Identificador', f.nombre AS 'Nombre del fabricante'

FROM fabricante f

JOIN producto p ON f.id = p.id_fabricante

GROUP BY f.id;
```

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.nombre AS 'Nombre del fabricante', p.nombre AS 'Nombre del producto'

FROM fabricante f

LEFT JOIN producto p ON f.id = p.id_fabricante

ORDER BY f.nombre ASC;
```

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

SELECT f.nombre **AS** 'Nombre del fabricante'

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** 'Número total de productos'

FROM producto;

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

SELECT COUNT(*) **AS** 'Número total de 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** 'Número de valores distintos de identificador de fabricante'

FROM producto;

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

SELECT AVG(precio) **AS** 'Media del precio de todos los productos'

FROM producto;

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

```
SELECT MIN(precio) AS 'Precio más barato de todos los productos'
FROM producto;
```

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

```
SELECT MAX(precio) AS 'Precio más caro de todos los productos'
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) FROM producto;
```

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

SELECT COUNT(*) **FROM** producto **WHERE** fabricante = 'Asus';

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

SELECT AVG(precio) **FROM** producto **WHERE** fabricante = 'Asus';

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

SELECT MIN(precio) **FROM** producto **WHERE** fabricante = 'Asus';

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

SELECT MAX(precio) **FROM** producto **WHERE** fabricante = 'Asus';

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

SELECT SUM(precio) **FROM** producto **WHERE** fabricante = 'Asus';

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

SELECT MAX(p.precio) **AS** precio_maximo, **MIN**(p.precio) **AS** precio_minimo, **AVG**(p.precio) **AS** precio_medio, **COUNT**(p.id) **AS** total_productos

FROM producto p

JOIN fabricante f **ON** p.id_fabricante = f.id

WHERE f.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 descendentemente 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.id

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 f.nombre **as** Fabricante, **MAX**(p.precio) **as** Precio_Maximo, **MIN**(p.precio) **as** Precio_Minimo, **AVG**(p.precio) **as** Precio_Medio

FROM fabricante f

LEFT JOIN producto p **ON** f.id = p.id_fabricante

GROUP BY f.id

ORDER BY Precio_Maximo **DESC**;

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 p.id_fabricante **as** Fabricante_ID, **MAX**(p.precio) **as** Precio_Maximo, **MIN**(p.precio) **as** Precio_Minimo, **AVG**(p.precio) **as** Precio_Medio, **COUNT**(p.id) **as** Total_Productos

FROM producto p

GROUP BY p.id_fabricante

HAVING Precio_Medio > 200

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 f.nombre as nombre_fabricante, MAX(p.precio) as precio_maximo, MIN(p.precio) as precio_minimo, AVG(p.precio) as precio_medio, COUNT(p.id) as numero_total_productos
FROM fabricante f
LEFT JOIN producto p ON f.id = p.id_fabricante
GROUP BY f.id
HAVING precio_medio > 200;
```

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

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

21. Calcula el número de productos que tiene cada fabricante con un precio mayor o igual a 180€. 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
WHERE precio >= 180
GROUP BY id_fabricante;
```

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

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

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

```
SELECT nombre
FROM fabricante
WHERE id IN (
SELECT id_fabricante
FROM producto
GROUP BY id_fabricante
HAVING AVG(precio) >= 150
);
```

24. 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.id
HAVING COUNT(p.id) >= 2;
```

25. 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 fabricante.nombre, COUNT(*) as num_productos
FROM fabricante
INNER JOIN producto ON fabricante.id = producto.id_fabricante
WHERE producto.precio >= 220
GROUP BY fabricante.nombre
HAVING COUNT(*) >= 2;
```

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 AS fabricante, 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.id
ORDER 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 fabricante.nombre, SUM(producto.precio) as total_precios
FROM fabricante
LEFT JOIN producto ON fabricante.id = producto.id_fabricante
GROUP BY fabricante.id
HAVING total_precios > 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.id_fabricante, p.precio) IN (
SELECT id_fabricante, MAX(precio)
FROM producto
GROUP BY id_fabricante
)
ORDER BY nombre_fabricante ASC;
```

• **Subconsultas**

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

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

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

SELECT nombre

FROM producto

WHERE id_fabricante = (**SELECT** id **FROM** fabricante **WHERE** nombre = 'Lenovo')

AND precio = (**SELECT MAX**(precio) **FROM** producto **WHERE** id_fabricante = (**SELECT** id **FROM** fabricante **WHERE** nombre = 'Lenovo'))

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

SELECT nombre

FROM producto

WHERE id_fabricante = (**SELECT** id **FROM** fabricante **WHERE** nombre = 'Hewlett-Packard')

ORDER BY precio **ASC LIMIT** 1;

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

SELECT *

FROM producto

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

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

```
SELECT *
FROM producto
WHERE id_fabricante = (SELECT id
FROM fabricante
WHERE nombre = 'Asus') AND precio > (SELECT AVG(precio)
FROM producto
WHERE id_fabricante = (SELECT id
FROM fabricante
WHERE nombre = 'Asus'
)
)
```

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 = (SELECT precio FROM producto ORDER BY precio DESC LIMIT 1);
```

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

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

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

SELECT nombre

FROM fabricante

WHERE id = **ALL** (**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 **NOT IN** (**SELECT DISTINCT** 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 DISTINCT 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 *
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 * FROM producto WHERE 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 AS fabricante_nombre, p.nombre AS producto_nombre, p.precio AS 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, f.nombre AS fabricante
FROM producto p
JOIN fabricante f ON p.id_fabricante = f.id
WHERE p.precio >= (SELECT AVG(precio) FROM producto WHERE 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 FROM fabricante WHERE nombre = 'Lenovo') ORDER BY precio
DESC LIMIT 1;
```

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