- **Cree la siguiente Base de datos con sus dos tablas, e introduzca los datos indicados:**

```
CREATE DATABASE tienda;
USE tienda;

CREATE TABLE fabricante (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100)
);

CREATE TABLE producto (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100),
  precio DECIMAL(10,2),
  id_fabricante INT,
  FOREIGN KEY (id_fabricante) REFERENCES fabricante(id)
);

INSERT INTO fabricante VALUES(1, 'Asus');
INSERT INTO fabricante VALUES(2, 'Lenovo');
INSERT INTO fabricante VALUES(3, 'Hewlett-Packard');
INSERT INTO fabricante VALUES(4, 'Samsung');
INSERT INTO fabricante VALUES(5, 'Seagate');
INSERT INTO fabricante VALUES(6, 'Crucial');
INSERT INTO fabricante VALUES(7, 'Gigabyte');
INSERT INTO fabricante VALUES(8, 'Huawei');
INSERT INTO fabricante VALUES(9, 'Xiaomi');

INSERT INTO producto VALUES(1, 'Disco duro SATA3 1TB', 86.99, 5);
INSERT INTO producto VALUES(2, 'Memoria RAM DDR4 8GB', 120, 6);
INSERT INTO producto VALUES(3, 'Disco SSD 1 TB', 150.99, 4);
INSERT INTO producto VALUES(4, 'GeForce GTX 1050Ti', 185, 7);
INSERT INTO producto VALUES(5, 'GeForce GTX 1080 Xtreme', 755, 6);
INSERT INTO producto VALUES(6, 'Monitor 24 LED Full HD', 202, 1);
INSERT INTO producto VALUES(7, 'Monitor 27 LED Full HD', 245.99, 1);
INSERT INTO producto VALUES(8, 'Portátil Yoga 520', 559, 2);
INSERT INTO producto VALUES(9, 'Portátil Ideapd 320', 444, 2);
INSERT INTO producto VALUES(10, 'Impresora HP Deskjet 3720', 59.99, 3);
```

- **Realice las siguientes consultas:**

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`.

```
SELECT *
FROM producto;
```

4. Lista el nombre de los productos y el precio con IVA incluido.

```
SELECT nombre, precio * 0.85 AS precio_en_euros
FROM producto;
```

5. Lista el nombre de los productos, el precio en euros. Utiliza los siguientes alias para las columnas: `nombre de producto`, `euros`.

```
SELECT nombre AS 'nombre de producto', precio * 0.85 AS euros
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_mayuscula, 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_minuscula, precio
FROM producto;
```

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

```
SELECT nombre AS fabricante, UPPER(SUBSTRING(nombre, 1, 2)) AS
primeros_dos_caracteres
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 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
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 3, 2;
```

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
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 nombre
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 *
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 *
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, 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 p.nombre, p.precio, f.nombre AS nombre_fabricante
FROM producto p, fabricante f
WHERE 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, p.precio, f.nombre AS nombre_fabricante
FROM producto p, fabricante f
WHERE 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.id_fabricante, f.nombre AS nombre_fabricante
FROM producto p, fabricante f
WHERE 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 AS nombre_fabricante
FROM producto p, fabricante f
WHERE p.id_fabricante = f.id AND 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, p.precio, f.nombre AS nombre_fabricante
FROM producto p, fabricante f
WHERE p.id_fabricante = f.id AND p.precio = (SELECT MAX(precio)
FROM producto);
```

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

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

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

```
SELECT *
FROM producto
WHERE id_fabricante = (SELECT id FROM fabricante WHERE
nombre = 'Crucial') AND 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
WHERE id_fabricante = (SELECT id FROM fabricante WHERE
nombre = 'Asus')
  OR id_fabricante = (SELECT id FROM fabricante WHERE nombre
= 'Hewlett-Packard')
  OR id_fabricante = (SELECT id FROM fabricante WHERE nombre
= 'Seagate');
```

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

```
SELECT *
FROM producto
WHERE id_fabricante IN (SELECT id FROM fabricante WHERE
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, fabricante f
WHERE p.id_fabricante = f.id AND 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, fabricante f
WHERE p.id_fabricante = f.id AND 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
FROM producto p, fabricante f
WHERE p.id_fabricante = f.id AND 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
WHERE EXISTS (SELECT 1 FROM producto p WHERE
p.id_fabricante = f.id);
```

Resuelva las siguientes consultas utilizando las cláusulas `LEFT JOIN` y `RIGHT JOIN`.

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

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

```
SELECT f.nombre
FROM fabricante f
LEFT JOIN producto p ON f.id = p.id_fabricante
```

```
WHERE p.id IS NULL;
```

3. ¿Pueden existir productos que no estén relacionados con un fabricante? Justifique su respuesta.

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 total_valores_distintos
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_mas_barato
FROM producto;
```

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

```
SELECT MAX(precio) AS precio_mas_caro
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 suma_precios
FROM producto;
```

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

```
SELECT COUNT(*) AS cantidad_productos
FROM producto
WHERE id_fabricante = (SELECT id FROM fabricante WHERE nombre = 'Asus');
```

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

```
SELECT AVG(precio) AS media_precios
```

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

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

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

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

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

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

```
SELECT SUM(precio) AS suma_precios
FROM producto
WHERE id_fabricante = (SELECT id 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 precio_maximo, MIN(precio) AS precio_minimo, AVG(precio) AS
precio_medio, COUNT(*) AS cantidad_productos
FROM producto
WHERE id_fabricante = (SELECT id 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 descendentemente por el número de productos.

```
SELECT f.nombre AS fabricante, COUNT(p.id) AS numero_productos
FROM fabricante f
LEFT JOIN producto p ON f.id = p.id_fabricante
GROUP BY f.nombre
ORDER BY numero_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.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 id_fabricante, MAX(precio) AS precio_maximo,
MIN(precio) AS precio_minimo, AVG(precio) AS precio_medio,
COUNT(id) AS numero_productos
```

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

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

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

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

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

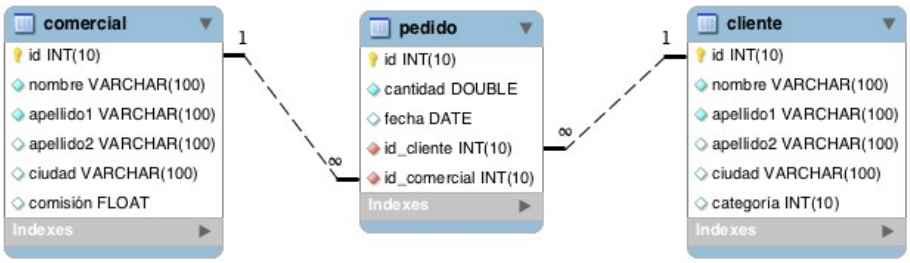
24. Lista los nombres de los fabricantes cuyos productos tienen un precio medio mayor o igual a 150.

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

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

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

- **Cree la siguiente Base de datos "Ventas" con sus tres tablas, e introduzca los datos indicados:**




```
CREATE DATABASE ventas;
USE ventas;

CREATE TABLE cliente (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100),
  apellido1 VARCHAR(100),
  apellido2 VARCHAR(100),
  ciudad VARCHAR(100),
  categoria INT
);

CREATE TABLE comercial (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100),
  apellido1 VARCHAR(100),
  apellido2 VARCHAR(100),
  comision FLOAT
);

CREATE TABLE pedido (
  id INT AUTO_INCREMENT PRIMARY KEY,
  total DOUBLE NOT NULL,
  fecha DATE,
  id_cliente INT,
  id_comercial INT,
  FOREIGN KEY (id_cliente) REFERENCES cliente(id),
  FOREIGN KEY (id_comercial) REFERENCES comercial(id)
);
```

```
INSERT INTO cliente (nombre, apellido1, apellido2, ciudad, categoria)
VALUES('Aarón', 'Rivero', 'Gómez', 'Almería', 100),
INSERT INTO cliente VALUES('Adela', 'Salas', 'Díaz', 'Granada', 200),
INSERT INTO cliente VALUES('Adolfo', 'Rubio', 'Flores', 'Sevilla', NULL),
INSERT INTO cliente VALUES('Adrián', 'Suárez', NULL, 'Jaén', 300),
INSERT INTO cliente VALUES('Marcos', 'Loyola', 'Méndez', 'Almería', 200),
INSERT INTO cliente VALUES('María', 'Santana', 'Moreno', 'Cádiz', 100),
INSERT INTO cliente VALUES('Pilar', 'Ruiz', NULL, 'Sevilla', 300),
INSERT INTO cliente VALUES('Pepe', 'Ruiz', 'Santana', 'Huelva', 200),
INSERT INTO cliente VALUES('Guillermo', 'López', 'Gómez', 'Granada', 225),
INSERT INTO cliente VALUES('Daniel', 'Santana', 'Loyola', 'Sevilla', 125);
```

```
INSERT INTO comercial(nombre, apellido1, apellido2, comision)
VALUES('Daniel', 'Sáez', 'Vega', 0.15),
 ('Juan', 'Gómez', 'López', 0.13),
 ('Diego','Flores', 'Salas', 0.11),
 ('Marta','Herrera', 'Gil', 0.14),
 ('Antonio','Carretero', 'Ortega', 0.12),
 ('Manuel','Domínguez', 'Hernández', 0.13),
 ('Antonio','Vega', 'Hernández', 0.11),
 ('Alfredo','Ruiz', 'Flores', 0.05);
```

```
INSERT INTO pedido(total, fecha, id_cliente, id_comercial)
VALUES(150.5, '2017-10-05', 5, 2),
 (270.65, '2016-09-10', 1, 5),
 (65.26, '2017-10-05', 2, 1),
 (110.5, '2016-08-17', 8, 3),
 (948.5, '2017-09-10', 5, 2),
 (2400.6, '2016-07-27', 7, 1),
 (5760, '2015-09-10', 2, 1),
 (1983.43, '2017-10-10', 4, 6),
 (2480.4, '2016-10-10', 8, 3),
 (250.45, '2015-06-27', 8, 2),
 (75.29, '2016-08-17', 3, 7),
 (3045.6, '2017-04-25', 2, 1),
 (545.75, '2019-01-25', 6, 1),
 (145.82, '2017-02-02', 6, 1),
 (370.85, '2019-03-11', 1, 5),
 (2389.23, '2019-03-11', 1, 5);
```

- **Realice las siguientes consultas:**

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;
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Consultas multitabla (Composición interna)

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
INNER JOIN pedido p ON c.id = p.id_cliente
ORDER BY c.apellido1, c.nombre;
```

2. Devuelve un listado que muestre todos los pedidos que ha realizado cada cliente. El resultado debe 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, p.fecha,
    com.id AS comercial_id, com.nombre AS comercial_nombre, com.apellido1
AS comercial_apellido1, com.apellido2 AS comercial_apellido2
FROM cliente c
INNER JOIN pedido p ON c.id = p.id_cliente
LEFT JOIN comercial com ON p.id_comercial = com.id
ORDER BY c.apellido1, c.nombre;
```

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 com.id AS comercial_id, com.nombre AS comercial_nombre,
com.apellido1 AS comercial_apellido1, com.apellido2 AS comercial_apellido2,
    p.id AS pedido_id, p.total, p.fecha,
    c.id AS cliente_id, c.nombre AS cliente_nombre, c.apellido1 AS
cliente_apellido1, c.apellido2 AS cliente_apellido2
FROM comercial com
INNER JOIN pedido p ON com.id = p.id_comercial
LEFT JOIN cliente c ON p.id_cliente = c.id
ORDER BY com.apellido1, com.nombre;
```

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

```
SELECT c.id AS cliente_id, c.nombre AS cliente_nombre, c.apellido1 AS
cliente_apellido1, c.apellido2 AS cliente_apellido2,
    p.id AS pedido_id, p.total, p.fecha,
```

```
    com.id AS comercial_id, com.nombre AS comercial_nombre, com.apellido1
AS comercial_apellido1, com.apellido2 AS comercial_apellido2
FROM cliente c
LEFT JOIN pedido p ON c.id = p.id_cliente
LEFT JOIN comercial com ON p.id_comercial = com.id
ORDER BY c.apellido1, c.nombre, p.fecha;
```

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

```
SELECT 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;
```

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

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

Consultas multitabla (Composición externa)

1. Devuelve un listado con todos los clientes junto con los datos de los pedidos que han realizado. Este 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.id, c.nombre, c.apellido1, c.apellido2, p.id AS pedido_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;
```

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

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

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

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

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

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

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

```
SELECT 'Cliente' AS tipo, c.id, c.nombre, c.apellido1, c.apellido2, NULL AS comercial_id, NULL AS comercial_nombre, NULL AS comercial_apellido1, NULL AS comercial_apellido2
```

```
FROM cliente c
LEFT JOIN pedido p ON c.id = p.id_cliente
WHERE p.id IS NULL
UNION
SELECT 'Comercial' AS tipo, NULL AS cliente_id, NULL AS cliente_nombre,
NULL AS cliente_apellido1, NULL AS cliente_apellido2, com.id, com.nombre,
com.apellido1, com.apellido2
FROM comercial com
LEFT JOIN pedido p ON com.id = p.id_comercial
WHERE p.id IS NULL
ORDER BY apellido1, apellido2, nombre;
```

Consultas resumen

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

11. Devuelve un listado con el identificador de cliente, nombre y apellidos y el número total de pedidos que ha realizado cada uno de clientes. Tenga en cuenta que pueden existir clientes que no han realizado 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 total_pedidos
FROM cliente c
LEFT JOIN pedido p ON c.id = p.id_cliente
GROUP BY c.id, c.nombre, c.apellido1, c.apellido2;
```

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

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

```
WHERE YEAR(p.fecha) = 2017 OR p.fecha IS NULL
GROUP BY c.id, c.nombre, c.apellido1, c.apellido2;
```

13. Devuelve un listado que muestre el identificador de cliente, nombre, primer apellido y el valor de la 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;
```

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

```
SELECT YEAR(fecha) AS año, MAX(total) AS max_valor_pedido
FROM pedido
GROUP BY YEAR(fecha);
```

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);
```