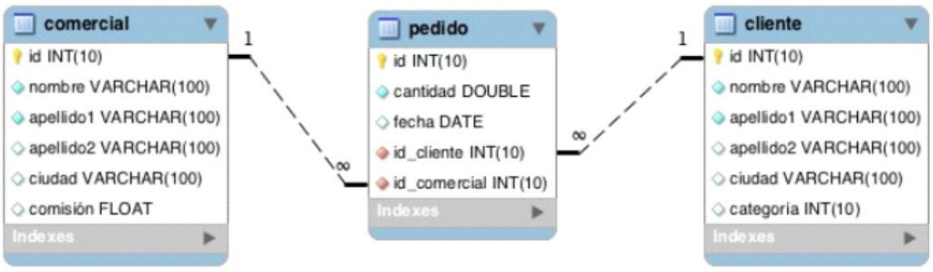
EJERCICIOS BASE DE DATOS

TABLAS: COMERCIAL, PEDIDO, CLIENTE

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;**

Result Grid | Filter Rows: | Edit:

id	total	fecha	id_cliente	id_comercial
15	370.85	2019-03-11	1	5
16	2389.23	2019-03-11	1	5
13	545.75	2019-01-25	6	1
8	1983.43	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	3045.6	2017-04-25	2	1
14	145.82	2017-02-02	6	1
9	2480.4	2016-10-10	8	3
2	270.65	2016-09-10	1	5
4	110.5	2016-08-17	8	3
11	75.29	2016-08-17	3	7
6	2400.6	2016-07-27	7	1
7	5760	2015-09-10	2	1
10	250.45	2015-06-27	8	2
NULL	NULL	NULL	NULL	NULL

2.

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

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

Result Grid | Filter Rows: | Edit:

id	total	fecha	id_cliente	id_comercial
7	5760	2015-09-10	2	1
12	3045.6	2017-04-25	2	1
NULL	NULL	NULL	NULL	NULL

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;

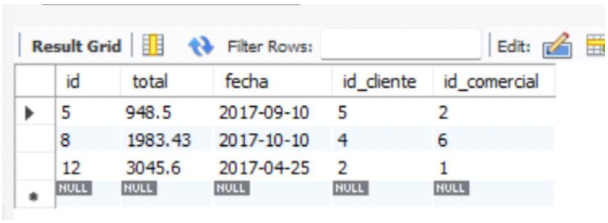
Result Grid

id_cliente
1
2
3
4
5
6
7
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;
```

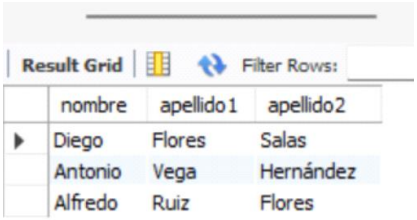


id	total	fecha	id_cliente	id_comercial
5	948.5	2017-09-10	5	2
8	1983.43	2017-10-10	4	6
12	3045.6	2017-04-25	2	1

5.

Devuelve un listado con el nombre y apellido1, apellido2 de los comerciales que tienen un float entre 0.05 y 0.11.

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

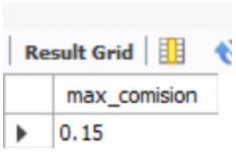


nombre	apellido1	apellido2
Diego	Flores	Salas
Antonio	Vega	Hernández
Alfredo	Ruiz	Flores

6.

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

```
SELECT comisión
FROM comercial
ORDER BY comisión DESC
LIMIT 1;
```

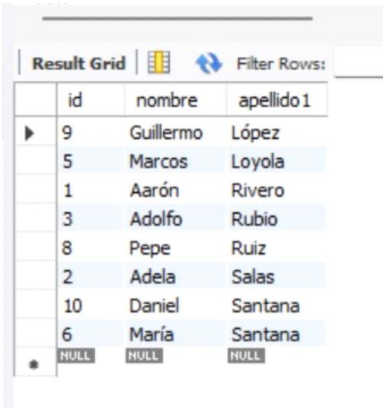


max_comision
0.15

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

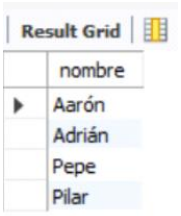


id	nombre	apellido1
9	Guillermo	López
5	Marcos	Loyola
1	Aarón	Rivero
3	Adolfo	Rubio
8	Pepe	Ruiz
2	Adela	Salas
10	Daniel	Santana
6	María	Santana

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 *
FROM cliente
WHERE nombre LIKE 'A%n' OR nombre LIKE 'P%'
ORDER BY nombre ASC;
```

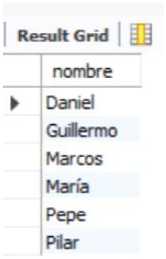


nombre
Aarón
Adrián
Pepe
Pilar

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 nombre NOT LIKE 'A%'
ORDER BY nombre ASC;
```

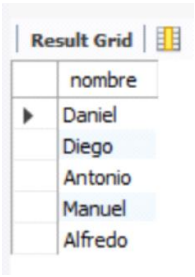


nombre
Daniel
Guillermo
Marcos
María
Pepe
Pilar

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 nombre LIKE "%el" OR nombre LIKE "%o";
```



nombre
Daniel
Diego
Antonio
Manuel
Alfredo

2. Consultas multitabla (Composición interna)

Existen dos tipos de composición externa: la composición externa izquierda (LEFT OUTER JOIN) y la composición externa derecha (RIGHT OUTER JOIN). Estos tipos de JOIN permiten incluir todos los registros de una tabla en la consulta, incluso si no hay coincidencias en la otra tabla.

2.1

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 AS c
INNER JOIN pedido AS p
ON p.id_cliente = c.id
ORDER BY c.nombre ASC;
```

id	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
8	Pepe	Ruiz	Santana
7	Pilar	Ruiz	NULL

DISTINCT para asegurarte de que no se muestren registros duplicados.

Luego, se realiza un INNER JOIN entre las tablas "cliente" y "pedido"utilizando la condición p.id_cliente = c.id para relacionar los clientes con sus pedidos. Por último, la cláusula ORDER BY c.nombre ASC ordena los resultados por el campo "nombre" en orden ascendente.

También se puede realizar de la siguiente manera:

```
SELECT DISTINCT id, nombre, apellido1, apellido2
FROM cliente
WHERE id IN (SELECT DISTINCT id_cliente FROM pedido)
ORDER BY nombre, apellido1, apellido2;
```

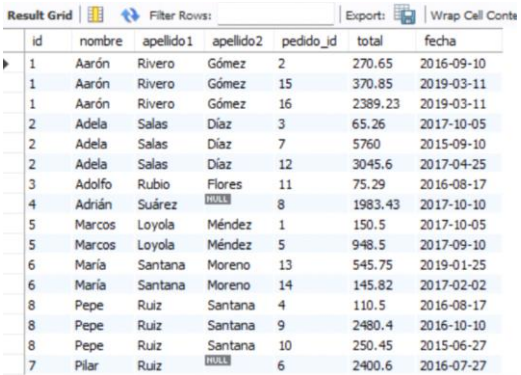
Explicación de la consulta:

- La subconsulta (SELECT DISTINCT id_cliente FROM pedido) selecciona los identificadores de los clientes que han realizado algún pedido sin elementos repetidos.
- La consulta principal SELECT DISTINCT id, nombre, apellido1, apellido2 FROM cliente selecciona los campos id, nombre, apellido1 y apellido2 de la tabla cliente.
- La cláusula WHERE id IN (...) filtra solo los registros de cliente cuyo identificador está presente en la subconsulta, es decir, los clientes que han realizado algún pedido.
- La cláusula ORDER BY nombre, apellido1, apellido2 ordena los resultados alfabéticamente por nombre, apellido1 y apellido2.

2. 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, c.nombre, c.apellido1, c.apellido2,
p.id AS pedido_id, p.total, p.fecha
FROM cliente c
JOIN pedido p ON c.id = p.id_cliente
ORDER BY c.nombre ASC;
```



id	nombre	apellido1	apellido2	pedido_id	total	fecha
1	Aarón	Rivero	Gómez	2	270.65	2016-09-10
1	Aarón	Rivero	Gómez	15	370.85	2019-03-11
1	Aarón	Rivero	Gómez	16	2389.23	2019-03-11
2	Adela	Salas	Díaz	3	65.26	2017-10-05
2	Adela	Salas	Díaz	7	5760	2015-09-10
2	Adela	Salas	Díaz	12	3045.6	2017-04-25
3	Adolfo	Rubio	Flores	11	75.29	2016-08-17
4	Adrián	Suárez	NULL	8	1983.43	2017-10-10
5	Marcos	Loyola	Méndez	1	150.5	2017-10-05
5	Marcos	Loyola	Méndez	5	948.5	2017-09-10
6	María	Santana	Moreno	13	545.75	2019-01-25
6	María	Santana	Moreno	14	145.82	2017-02-02
8	Pepe	Ruiz	Santana	4	110.5	2016-08-17
8	Pepe	Ruiz	Santana	9	2480.4	2016-10-10
8	Pepe	Ruiz	Santana	10	250.45	2015-06-27
7	Pilar	Ruiz	NULL	6	2400.6	2016-07-27

Utilizamos la cláusula SELECT para seleccionar los campos que deseamos mostrar en el resultado. En este caso, seleccionamos los campos id, nombre, apellido1 y apellido2 de la tabla cliente, y los campos id, total y fecha de la tabla pedido.

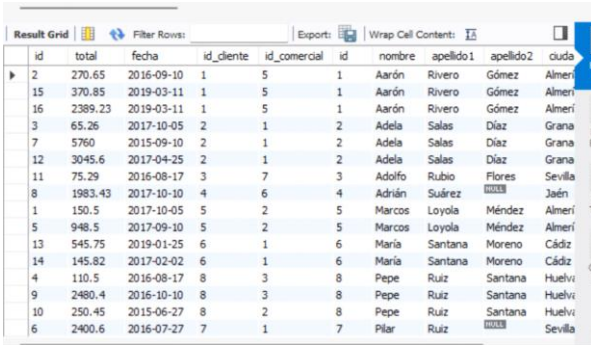
- Utilizamos la cláusula FROM para especificar las tablas involucradas en la consulta, en este caso, la tabla cliente y la tabla pedido.

- Utilizamos la cláusula JOIN para combinar las filas de las tablas cliente y pedido basándonos en la condición de igualdad c.id = p.id_cliente. Esto asegura que cada pedido esté relacionado con el cliente correspondiente.

- Finalmente, utilizamos la cláusula ORDER BY para ordenar los resultados alfabéticamente por el campo nombre de la tabla cliente, especificando ASC para orden ascendente.

También se puede poner de la siguiente manera:

```
SELECT p.*, c.*
FROM pedido p
INNER JOIN cliente c ON p.id_cliente = c.id
ORDER BY c.nombre ASC;
```



id	total	fecha	id_cliente	id_comercial	id	nombre	apellido1	apellido2	ciuda
2	270.65	2016-09-10	1	5	1	Aarón	Rivero	Gómez	Almerí
15	370.85	2019-03-11	1	5	1	Aarón	Rivero	Gómez	Almerí
16	2389.23	2019-03-11	1	5	1	Aarón	Rivero	Gómez	Almerí
3	65.26	2017-10-05	2	1	2	Adela	Salas	Díaz	Grana
7	5760	2015-09-10	2	1	2	Adela	Salas	Díaz	Grana
12	3045.6	2017-04-25	2	1	2	Adela	Salas	Díaz	Grana
11	75.29	2016-08-17	3	7	3	Adolfo	Rubio	Flores	Sevilla
8	1983.43	2017-10-10	4	6	4	Adrián	Suárez	NULL	Jaén
1	150.5	2017-10-05	5	2	5	Marcos	Loyola	Méndez	Almerí
5	948.5	2017-09-10	5	2	5	Marcos	Loyola	Méndez	Almerí
13	545.75	2019-01-25	6	1	6	María	Santana	Moreno	Cádiz
14	145.82	2017-02-02	6	1	6	María	Santana	Moreno	Cádiz
4	110.5	2016-08-17	8	3	8	Pepe	Ruiz	Santana	Huelva
9	2480.4	2016-10-10	8	3	8	Pepe	Ruiz	Santana	Huelva
10	250.45	2015-06-27	8	2	8	Pepe	Ruiz	Santana	Huelva
6	2400.6	2016-07-27	7	1	7	Pilar	Ruiz	NULL	Sevilla

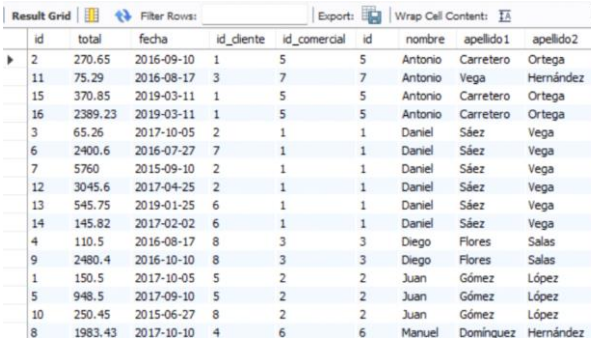
Ahora lo mismo sin utilizar alias

```
SELECT pedido.*, cliente.*
FROM pedido
INNER JOIN cliente ON pedido.id_cliente = cliente.id
ORDER BY cliente.nombre ASC;
```

2.3

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.*, comercial.*
FROM pedido
INNER JOIN comercial ON pedido.id_comercial
= comercial.id
ORDER BY comercial.nombre ASC;
```



id	total	fecha	id_cliente	id_comercial	id	nombre	apellido1	apellido2
2	270.65	2016-09-10	1	5	5	Antonio	Carretero	Ortega
11	75.29	2016-08-17	3	7	7	Antonio	Vega	Hernández
15	370.85	2019-03-11	1	5	5	Antonio	Carretero	Ortega
16	2389.23	2019-03-11	1	5	5	Antonio	Carretero	Ortega
3	65.26	2017-10-05	2	1	1	Daniel	Sáez	Vega
6	2400.6	2016-07-27	7	1	1	Daniel	Sáez	Vega
7	5760	2015-09-10	2	1	1	Daniel	Sáez	Vega
12	3045.6	2017-04-25	2	1	1	Daniel	Sáez	Vega
13	545.75	2019-01-25	6	1	1	Daniel	Sáez	Vega
14	145.82	2017-02-02	6	1	1	Daniel	Sáez	Vega
4	110.5	2016-08-17	8	3	3	Diego	Flores	Salas
9	2480.4	2016-10-10	8	3	3	Diego	Flores	Salas
1	150.5	2017-10-05	5	2	2	Juan	Gómez	López
5	948.5	2017-09-10	5	2	2	Juan	Gómez	López
10	250.45	2015-06-27	8	2	2	Juan	Gómez	López
8	1983.43	2017-10-10	4	6	6	Manuel	Domínguez	Hernández

2.4

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.*, p.*, co.*
FROM cliente c
INNER JOIN pedido p
ON c.id = p.id_cliente
INNER JOIN comercial co
ON co.id = p.id_comercial;
```

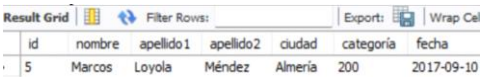
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total	fecha	id_cliente	id_comercial	id	nombre	apellido1	apellido2	comisión
270.65	2016-09-10	1	5	5	Antonio	Carretero	Ortega	0.12
370.85	2019-03-11	1	5	5	Antonio	Carretero	Ortega	0.12
2389.23	2019-03-11	1	5	5	Antonio	Carretero	Ortega	0.12
65.26	2017-10-05	2	1	1	Daniel	Sáez	Vega	0.15
5760	2015-09-10	2	1	1	Daniel	Sáez	Vega	0.15
3045.6	2017-04-25	2	1	1	Daniel	Sáez	Vega	0.15
75.29	2016-08-17	3	7	7	Antonio	Vega	Hernández	0.11
1983.43	2017-10-10	4	6	6	Manuel	Domínguez	Hernández	0.13
150.5	2017-10-05	5	2	2	Juan	Gómez	López	0.13
948.5	2017-09-10	5	2	2	Juan	Gómez	López	0.13
545.75	2019-01-25	6	1	1	Daniel	Sáez	Vega	0.15
145.82	2017-02-02	6	1	1	Daniel	Sáez	Vega	0.15
110.5	2016-08-17	8	3	3	Diego	Flores	Salas	0.11
2480.4	2016-10-10	8	3	3	Diego	Flores	Salas	0.11
250.45	2015-06-27	8	2	2	Juan	Gómez	López	0.13
2400.6	2016-07-27	7	1	1	Daniel	Sáez	Vega	0.15

2.5

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

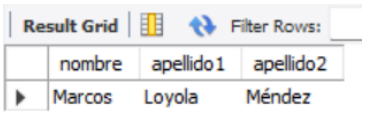
```
SELECT c.*, p.fecha
FROM pedido p
INNER JOIN cliente c
ON p.id_cliente = c.id
WHERE fecha LIKE '%2017%'
AND total BETWEEN 300 AND 1000;
```



Result Grid | Filter Rows: | Export: | Wrap Cel

id	nombre	apellido1	apellido2	ciudad	categoría	fecha
5	Marcos	Loyola	Méndez	Almería	200	2017-09-10

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



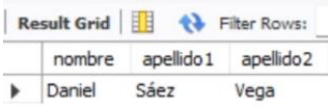
Result Grid | Filter Rows:

nombre	apellido1	apellido2
Marcos	Loyola	Méndez

2.6

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

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



Result Grid | Filter Rows:

nombre	apellido1	apellido2
Daniel	Sáez	Vega

2.7

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

Result Grid

nombre
Adela
Pilar
María

3. Consultas multitabla (Composición externa)

La composición externa en consultas multitabla se refiere a la inclusión de registros que no tienen coincidencia en la condición de unión entre las tablas. Esto significa que se incluirán todos los registros de una tabla, incluso si no hay coincidencia con los registros de la otra tabla.

Existen dos tipos de composición externa: la composición externa izquierda (LEFT OUTER JOIN) y la composición externa derecha (RIGHT OUTER JOIN). Estos tipos de JOIN permiten incluir todos los registros de una tabla en la consulta, incluso si no hay coincidencias en la otra tabla. La palabra OUTER es opcional y no añade ninguna función.

2.8

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.

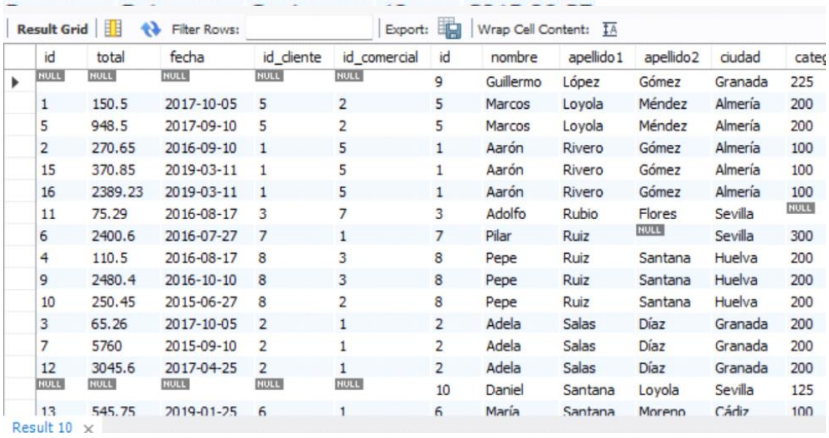
1 manera:

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

nombre	apellido1	apellido2	id	fecha
Aarón	Rivero	Gómez	2	2016-09-10
Aarón	Rivero	Gómez	15	2019-03-11
Aarón	Rivero	Gómez	16	2019-03-11
Adela	Salas	Díaz	3	2017-10-05
Adela	Salas	Díaz	7	2015-09-10
Adela	Salas	Díaz	12	2017-04-25
Adolfo	Rubio	Flores	11	2016-08-17
Adrián	Suárez	NULL	8	2017-10-10
Marcos	Loyola	Méndez	1	2017-10-05
Marcos	Loyola	Méndez	5	2017-09-10
María	Santana	Moreno	13	2019-01-25
María	Santana	Moreno	14	2017-02-02
Pilar	Ruiz	NULL	6	2016-07-27
Pepe	Ruiz	Santana	4	2016-08-17
Pepe	Ruiz	Santana	9	2016-10-10

2 manera:

```
SELECT *
FROM pedido
RIGHT JOIN cliente
ON cliente.id =
pedido.id_cliente
ORDER BY cliente.apellido1
ASC, cliente.apellido2 ASC,
cliente.nombre ASC;
```



id	total	fecha	id_cliente	id_comercial	id	nombre	apellido1	apellido2	ciudad	categ
NULL	NULL	NULL	NULL	NULL	9	Guillermo	López	Gómez	Granada	225
1	150.5	2017-10-05	5	2	5	Marcos	Loyola	Méndez	Almería	200
5	948.5	2017-09-10	5	2	5	Marcos	Loyola	Méndez	Almería	200
2	270.65	2016-09-10	1	5	1	Aarón	Rivero	Gómez	Almería	100
15	370.85	2019-03-11	1	5	1	Aarón	Rivero	Gómez	Almería	100
16	2389.23	2019-03-11	1	5	1	Aarón	Rivero	Gómez	Almería	100
11	75.29	2016-08-17	3	7	3	Adolfo	Rubio	Flores	Sevilla	NULL
6	2400.6	2016-07-27	7	1	7	Pilar	Ruiz	NULL	Sevilla	300
4	110.5	2016-08-17	8	3	8	Pepe	Ruiz	Santana	Huelva	200
9	2480.4	2016-10-10	8	3	8	Pepe	Ruiz	Santana	Huelva	200
10	250.45	2015-06-27	8	2	8	Pepe	Ruiz	Santana	Huelva	200
3	65.26	2017-10-05	2	1	2	Adela	Salas	Díaz	Granada	200
7	5760	2015-09-10	2	1	2	Adela	Salas	Díaz	Granada	200
12	3045.6	2017-04-25	2	1	2	Adela	Salas	Díaz	Granada	200
NULL	NULL	NULL	NULL	NULL	10	Daniel	Santana	Loyola	Sevilla	125
13	545.75	2019-01-25	6	1	6	María	Santana	Moreno	Cádiz	100

Ésta segunda consulta devuelve un listado de clientes con los detalles de los pedidos que han realizado. Si un cliente no ha realizado ningún pedido, aún aparecerá en el listado, pero con valores nulos en las columnas correspondientes a los pedidos. Los resultados se ordenan alfabéticamente por apellido1, apellido2 y nombre del cliente.

2.9

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.

Con un primera consulta el resultado sería un listado que muestra los detalles de los pedidos junto con la información del cliente asociado. Los resultados se ordenan alfabéticamente por el primer apellido (apellido1), segundo apellido (apellido2) y nombre del cliente en orden ascendente.

```
SELECT
a.id,a.apellido1,a.apellido2,
a.nombre,b.*
FROM ventas.pedido b
RIGHT JOIN ventas.cliente a
ON b.id_cliente = a.id
ORDER BY apellido1 ASC,
apellido2 ASC,
nombre ASC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	apellido1	apellido2	nombre	id	total	fecha	id_cliente	id_comercial
9	López	Gómez	Guillermo	NULL	NULL	NULL	NULL	NULL
5	Loyola	Méndez	Marcos	1	150.5	2017-10-05	5	2
5	Loyola	Méndez	Marcos	5	948.5	2017-09-10	5	2
1	Rivero	Gómez	Aarón	2	270.65	2016-09-10	1	5
1	Rivero	Gómez	Aarón	15	370.85	2019-03-11	1	5
1	Rivero	Gómez	Aarón	16	2389.23	2019-03-11	1	5
3	Rubio	Flores	Adolfo	11	75.29	2016-08-17	3	7
7	Ruiz	NULL	Pilar	6	2400.6	2016-07-27	7	1
8	Ruiz	Santana	Pepe	4	110.5	2016-08-17	8	3
8	Ruiz	Santana	Pepe	9	2480.4	2016-10-10	8	3
8	Ruiz	Santana	Pepe	10	250.45	2015-06-27	8	2
2	Salas	Díaz	Adela	3	65.26	2017-10-05	2	1
2	Salas	Díaz	Adela	7	5760	2015-09-10	2	1
2	Salas	Díaz	Adela	12	3045.6	2017-04-25	2	1
10	Santana	Loyola	Daniel	NULL	NULL	NULL	NULL	NULL
6	Santana	Moreno	María	13	545.75	2019-01-25	6	1

```
SELECT a.id, a.apellido1,
a.apellido2, a.nombre, b.*
FROM ventas.cliente a
LEFT JOIN ventas.pedido b
ON b.id_cliente = a.id
ORDER BY a.apellido1 ASC,
a.apellido2 ASC, a.nombre
ASC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	apellido1	apellido2	nombre	id	total	fecha	id_cliente	id_comercial
9	López	Gómez	Guillermo	NULL	NULL	NULL	NULL	NULL
5	Loyola	Méndez	Marcos	1	150.5	2017-10-05	5	2
5	Loyola	Méndez	Marcos	5	948.5	2017-09-10	5	2
1	Rivero	Gómez	Aarón	2	270.65	2016-09-10	1	5
1	Rivero	Gómez	Aarón	15	370.85	2019-03-11	1	5
1	Rivero	Gómez	Aarón	16	2389.23	2019-03-11	1	5
3	Rubio	Flores	Adolfo	11	75.29	2016-08-17	3	7
7	Ruiz	NULL	Pilar	6	2400.6	2016-07-27	7	1
8	Ruiz	Santana	Pepe	4	110.5	2016-08-17	8	3
8	Ruiz	Santana	Pepe	9	2480.4	2016-10-10	8	3
8	Ruiz	Santana	Pepe	10	250.45	2015-06-27	8	2
2	Salas	Díaz	Adela	3	65.26	2017-10-05	2	1
2	Salas	Díaz	Adela	7	5760	2015-09-10	2	1
2	Salas	Díaz	Adela	12	3045.6	2017-04-25	2	1
10	Santana	Loyola	Daniel	NULL	NULL	NULL	NULL	NULL
6	Santana	Moreno	María	13	545.75	2019-01-25	6	1
6	Santana	Moreno	María	14	145.82	2017-02-02	6	1

El resultado de la segunda consulta sería un listado que muestra los detalles de los clientes junto con la información de los pedidos asociados a cada cliente. Si un cliente no tiene ningún pedido asociado, las columnas correspondientes a los pedidos tendrán valores nulos en esas filas.

2.10

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

```
SELECT cliente.*
FROM pedido
RIGHT JOIN cliente ON cliente.id =
pedido.id_cliente
WHERE pedido.id_cliente IS NULL;
```

id	nombre	apellido1	apellido2	ciudad	categoría
9	Guillermo	López	Gómez	Granada	225
10	Daniel	Santana	Loyola	Sevilla	125

2.11

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

```
SELECT comercial.*
FROM pedido
RIGHT JOIN comercial ON comercial.id =
pedido.id_comercial
WHERE pedido.id_comercial IS NULL;
```

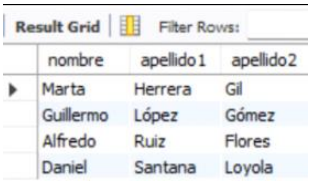
id	nombre	apellido1	apellido2	comisión
4	Marta	Herrera	Gil	0.14
8	Alfredo	Ruiz	Flores	0.05

Esta consulta busca los registros de la tabla "comercial" que no tienen coincidencia en la tabla "pedido". La condición WHERE pedido.id_comercial IS NULL filtra los resultados para mostrar solamente los registros de la tabla "comercial" que no tienen una correspondencia en la tabla "pedido" a través del campo "id_comercial". Esto significa que se devolverán los comerciales que no hayan realizado ningún pedido.

2.12

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 comercial.nombre, comercial.apellido1, comercial.apellido2
FROM pedido
RIGHT JOIN comercial ON pedido.id_comercial = comercial.id
WHERE pedido.id_comercial IS NULL)
UNION
(SELECT cliente.nombre, cliente.apellido1, cliente.apellido2
FROM pedido
RIGHT JOIN cliente ON pedido.id_cliente = cliente.id
WHERE pedido.id_cliente IS NULL)
ORDER BY apellido1 ASC, apellido2 ASC, nombre ASC;
```



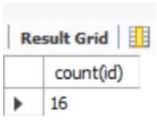
nombre	apellido1	apellido2
Marta	Herrera	Gil
Guillermo	López	Gómez
Alfredo	Ruiz	Flores
Daniel	Santana	Loyola

4. Consultas resumen

2.13

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

```
SELECT count(id)
FROM pedido;
```

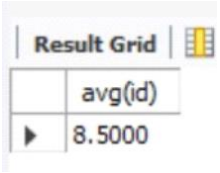


count(id)
16

2.14

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

```
SELECT AVG(id)
FROM pedido;
```

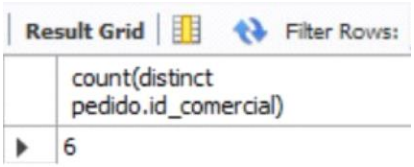


avg(id)
8.5000

2.15

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

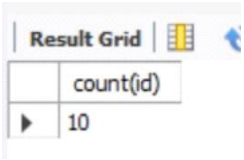
```
SELECT COUNT(distinct pedido.id_comercial)
FROM pedido
JOIN comercial ON pedido.id_comercial = comercial.id;
```



2.16

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

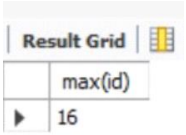
```
SELECT COUNT(id)
FROM cliente;
```



2.17

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

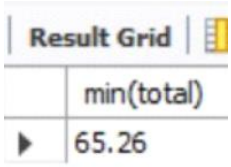
```
SELECT max(id)
FROM pedido;
```



2.18

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

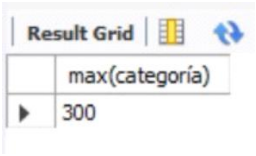
```
SELECT min(total)
FROM pedido;
```



2.19

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

```
SELECT max(categoría)
FROM cliente;
```



2.20

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.

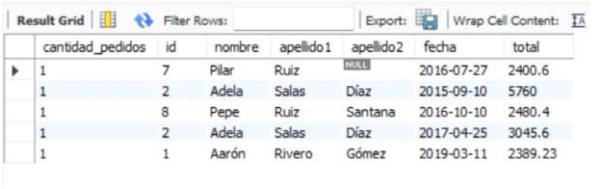
```
SELECT COUNT(pedido.fecha) AS
cantidad_pedidos, cliente.id,
cliente.nombre, cliente.apellido1,
cliente.apellido2, pedido.fecha, pedido.total
FROM pedido
JOIN cliente ON pedido.id_cliente =
cliente.id
GROUP BY cliente.id, cliente.nombre,
cliente.apellido1, cliente.apellido2,
pedido.fecha, pedido.total;
```

cantidad_pedidos	id	nombre	apellido1	apellido2	fecha	total
1	5	Marcos	Loyola	Méndez	2017-10-05	150.5
1	1	Aarón	Rivero	Gómez	2016-09-10	270.65
1	2	Adela	Salas	Díaz	2017-10-05	65.26
1	8	Pepe	Ruiz	Santana	2016-08-17	110.5
1	5	Marcos	Loyola	Méndez	2017-09-10	948.5
1	7	Pilar	Ruiz	NULL	2016-07-27	2400.6
1	2	Adela	Salas	Díaz	2015-09-10	5760
1	4	Adrián	Suárez	NULL	2017-10-10	1983.43
1	8	Pepe	Ruiz	Santana	2016-10-10	2480.4
1	8	Pepe	Ruiz	Santana	2015-06-27	250.45
1	3	Adolfo	Rubio	Flores	2016-08-17	75.29
1	2	Adela	Salas	Díaz	2017-04-25	3045.6
1	6	María	Santana	Moreno	2019-01-25	545.75
1	6	María	Santana	Moreno	2017-02-02	145.82
1	1	Aarón	Rivero	Gómez	2019-03-11	370.85
1	1	Aarón	Rivero	Gómez	2019-03-11	2389.23

2.21

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 COUNT(pedido.fecha), cliente.id,
cliente.nombre, cliente.apellido1,
cliente.apellido2, pedido.fecha, pedido.total
FROM pedido
JOIN cliente ON pedido.id_cliente = cliente.id
WHERE pedido.total > 2000
GROUP BY cliente.nombre, pedido.fecha,
pedido.total;
```

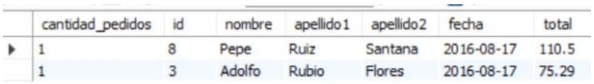


cantidad_pedidos	id	nombre	apellido1	apellido2	fecha	total
1	7	Pilar	Ruiz	NULL	2016-07-27	2400.6
1	2	Adela	Salas	Díaz	2015-09-10	5760
1	8	Pepe	Ruiz	Santana	2016-10-10	2480.4
1	2	Adela	Salas	Díaz	2017-04-25	3045.6
1	1	Aarón	Rivero	Gómez	2019-03-11	2389.23

2.22

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 COUNT(pedido.fecha) AS
cantidad_pedidos, cliente.id, cliente.nombre,
cliente.apellido1, cliente.apellido2,
pedido.fecha, pedido.total
FROM pedido
JOIN cliente ON pedido.id_cliente = cliente.id
WHERE pedido.fecha = '2016-08-17'
GROUP BY cliente.id, cliente.nombre,
cliente.apellido1, cliente.apellido2,
pedido.fecha, pedido.total;
```



cantidad_pedidos	id	nombre	apellido1	apellido2	fecha	total
1	8	Pepe	Ruiz	Santana	2016-08-17	110.5
1	3	Adolfo	Rubio	Flores	2016-08-17	75.29

2.23

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 cliente.id, cliente.nombre,
cliente.apellido1, cliente.apellido2, count
(pedido.id_cliente)
FROM pedido
RIGHT JOIN cliente ON pedido.id_cliente =
cliente.id
GROUP BY pedido.id_cliente;
```

Result Grid | Filter Rows:

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

2.24

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

Result Grid | Filter Rows:

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

2. 25

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

Result Grid | Filter Rows:

id	nombre	apellido1	max_total
1	Aarón	Rivero	2389.23
2	Adela	Salas	5760
3	Adolfo	Rubio	75.29
4	Adrián	Suárez	1983.43
5	Marcos	Loyola	948.5
6	María	Santana	545.75
7	Pilar	Ruiz	2400.6
8	Pepe	Ruiz	2480.4
9	Guillermo	López	0
10	Daniel	Santana	0

2.26

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

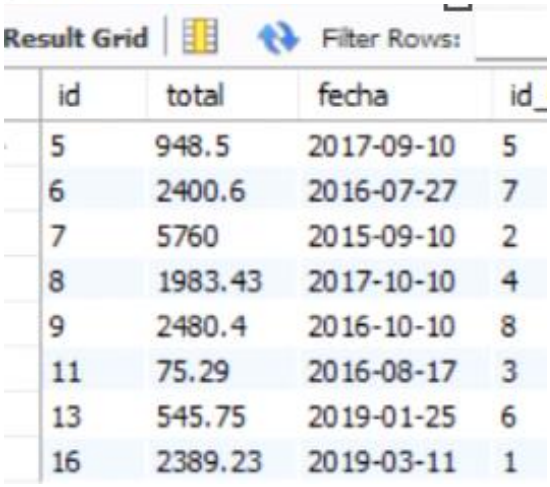
1 MANERA

```
SELECT max(pedido.total)
FROM pedido
JOIN cliente
ON pedido.id_cliente = cliente.id
GROUP BY pedido.id_cliente;
```

max(pedido.total)
948.5
2389.23
5760
2480.4
2400.6
1983.43
75.29
545.75

2 MANERA

```
SELECT p.*
FROM pedido p
INNER JOIN cliente c ON p.id_cliente = c.id
WHERE p.total = (
   SELECT MAX(total)
   FROM pedido
   WHERE id_cliente = c.id
);
```



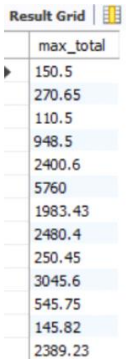
Result Grid | Filter Rows:

id	total	fecha	id_
5	948.5	2017-09-10	5
6	2400.6	2016-07-27	7
7	5760	2015-09-10	2
8	1983.43	2017-10-10	4
9	2480.4	2016-10-10	8
11	75.29	2016-08-17	3
13	545.75	2019-01-25	6
16	2389.23	2019-03-11	1

2. 27

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

```
SELECT max(pedido.total)
FROM pedido JOIN cliente
ON pedido.id_cliente = cliente.id
GROUP BY pedido.fecha;
```



Result Grid

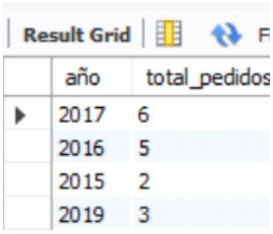
max_total
150.5
270.65
110.5
948.5
2400.6
5760
1983.43
2480.4
250.45
3045.6
545.75
145.82
2389.23

En esta consulta, se obtiene el máximo valor de los pedidos para cada fecha, realizando un JOIN entre las tablas 'pedido' y 'cliente' utilizando la columna 'id_cliente' y 'id' respectivamente. Luego, utilizo la cláusula GROUP BY para agrupar los resultados por fecha.

Sin embargo, este código devuelve el máximo valor de los pedidos para cada fecha individualmente. Si quiero obtener el máximo valor de todos los pedidos en general, sin agrupar por fecha, debo omitir la cláusula GROUP BY

Para conocer el total de pedidos que se han realizado cada año tengo que utilizar YEAR

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



Ahora utilizo la función YEAR() para extraer el año de la columna 'fecha_pedido'. Utilizo la función de agregación COUNT() para contar el número total de pedidos realizados para cada año.

La cláusula GROUP BY se usa para agrupar los resultados por el año extraído.

Se obtiene el año y el número total de pedidos realizados en ese año.