

# Práctica 2

## Fundamentos de Bases de Datos

*Práctica realizada por Lorena Almoguera Romero*

# CONSULTAS SQL

1. Obtener el nombre, los dos apellidos y el email de los usuarios de la tienda que se llaman Maite o Ana

```
SELECT nombre, apellido1, apellido2, email FROM usuario WHERE nombre = 'Maite' OR nombre = 'Ana';
```

✓ Showing rows 0 - 1 (2 total, Query took 0.0002 seconds.)

```
SELECT nombre, apellido1, apellido2, email FROM `usuario` WHERE nombre = 'Maite' OR nombre = 'Ana';
```

☐ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

		nombre	apellido1	apellido2	email
<input type="checkbox"/>	Edit	maite	barcelo	garcia	maitebarcelo@gmail.com
<input type="checkbox"/>	Edit	ana	romero	martinez	bernardoromero@gmail.com

Seleccionamos los atributos requeridos (nombre, apellido1, apellido2, email) para la consulta dentro de la tabla en la cual deseamos realizar la query. Además, ya que se nos indica que debemos buscar los usuarios que se llamen Maite o Ana, indicamos que los atributos por los cuales queremos filtrar deben tener el nombre Maite o Ana.

2. Obtener el id del producto, nombre y precio de los artículos que no tienen marca

```
SELECT id_producto, nombre, precio FROM `producto` WHERE id_marca IS NULL;
```

✓ Showing rows 0 - 2 (3 total, Query took 0.0002 seconds.)

```
SELECT id_producto, nombre, precio FROM `producto` WHERE id_marca IS NULL;
```

☐ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by

Extra options

		id_producto	nombre	precio
<input type="checkbox"/>	Edit	2	cucharón de madera de pino	1.00
<input type="checkbox"/>	Edit	5	escoba nylon	1.99
<input type="checkbox"/>	Edit	7	guantes desechables negros	11.49

Seleccionamos los atributos requeridos (id\_producto, nombre, precio, marca) para la consulta dentro de la tabla en la cual deseamos realizar la query. Además, ya que se nos indica que debemos buscar los productos en los cuales marca sea nulo, especificamos esto como WHERE id\_marca IS NULL de esta manera nos proporcionará los resultados.

3. Obtener id, nombre, precio y precio incrementado un 10% de los productos de la tienda.

```
SELECT id_producto, nombre, precio, precio * 1.10 AS precio_incrementado FROM `producto`;
```

Showing rows 0 - 10 (11 total, Query took 0.0002 seconds.)

```
SELECT id_producto, nombre, precio, precio * 1.10 AS precio_incrementado FROM `producto`;
```

☐ Profiling [\[ Edit inline \]](#) [\[ Edit \]](#) [\[ Explain SQL \]](#) [\[ Create PHP code \]](#) [\[ Refresh \]](#)

☐ Show all | Number of rows: 25 | Filter rows:  Search this table | Sort by key: None

Extra options

		id_producto	nombre	precio	precio_incrementado
<input type="checkbox"/>	Edit Copy Delete	1	robot aspirador cecotec conga 1390	200.00	220.0000
<input type="checkbox"/>	Edit Copy Delete	2	cucharón de madera de pino	1.00	1.1000
<input type="checkbox"/>	Edit Copy Delete	3	fregona super	5.00	5.5000
<input type="checkbox"/>	Edit Copy Delete	4	aspiradora ultra	150.00	165.0000
<input type="checkbox"/>	Edit Copy Delete	5	escoba nylon	1.99	2.1890
<input type="checkbox"/>	Edit Copy Delete	6	super glue 3	4.79	5.2690
<input type="checkbox"/>	Edit Copy Delete	7	guantes desechables negros	11.49	12.6390
<input type="checkbox"/>	Edit Copy Delete	8	tiras adhesivas para colgar cuadros	6.49	7.1390
<input type="checkbox"/>	Edit Copy Delete	9	gafas proteccion	3.25	3.5750
<input type="checkbox"/>	Edit Copy Delete	10	taladro easy impact	39.95	43.9450
<input type="checkbox"/>	Edit Copy Delete	11	medidor laser bosch	143.28	157.6080

Seleccionamos los atributos requeridos (id\_producto, nombre, precio). Ya que queremos visualizar el precio incrementado en un 10% de los productos de la tienda, especificaremos dos veces el precio, uno de manera habitual y otra en la cual le indicaremos a la consulta de seleccionarlo COMO precio\_incrementado en el cual incrementaremos el precio en 1.10 (para que se le añada el 10 por ciento). De esta forma obtendremos los resultados deseados.

4. Obtener el código postal, nombre y habitantes de las localidades que tienen más de 1000 habitantes y que además tienen un nombre que empieza por la letra b o la letra c. Ordena los datos por habitantes de forma ascendente.

```
SELECT codigopostal, nombre, habitantes FROM `localidad` WHERE habitantes > 1000 AND nombre LIKE 'B%' OR nombre LIKE 'C%' ORDER BY habitantes ASC;
```

Showing rows 0 - 7 (8 total, Query took 0.0002 seconds.) [habitantes: 2000... - 30000...]

```
SELECT codigopostal, nombre, habitantes FROM 'localidad' WHERE habitantes > 1000 AND nombre LIKE 'B%' OR nombre LIKE 'C%' ORDER BY habitantes ASC;
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	codigopostal	nombre	habitantes
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	00005	bétera	2000
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	00010	caen	2000
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	00009	burriana	3000
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	00006	bocairente	4000
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	00008	benicasim	6000
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	00007	burjasot	8000
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	00002	benisa	20000
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	00003	benidorm	30000

Seleccionamos los atributos requeridos y especificamos que habitantes tiene que ser mayor a 1000. Además, indicamos que nombre debe comenzar por 'B' o 'C'. Finalmente, indicamos que ordene los resultados por habitantes de manera ascendente. De esta manera obtendremos los resultados deseados.

- Obtener el id de usuario, dni, nombre, primer apellido, teléfono y fecha de nacimiento de los usuarios cuyo dni empieza por el número 3 y que NO tienen teléfono y que han nacido en el año 1992. Ordena los datos en primer lugar por primer apellido y después por nombre.

```
SELECT id_usuario, dni, nombre, apellido1, telefono, fechanac FROM usuario WHERE dni LIKE '3%' AND telefono IS NULL AND YEAR(fechanac) = 1992 ORDER BY apellido1 ASC, nombre ASC;
```

Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.) [apellido1: PEREZ... - PEREZ...][nombre: OSCAR... - OSCAR...]

```
SELECT id_usuario, dni, nombre, apellido1, telefono, fechanac FROM usuario WHERE dni LIKE '3%' AND telefono IS NULL AND YEAR(fechanac) = 1992 ORDER BY apellido1 ASC, nombre ASC;
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

	id_usuario	dni	nombre	apellido1	telefono	fechanac
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	30333444W	oscar	perez	NULL	1992-10-03

Check all | With selected: Edit Copy Delete Export

Show all | Number of rows: 25 | Filter rows: Search this table

Seleccionamos los atributos requeridos y especificamos que el dni debe empezar por 3, el año debe ser 1992 y que se deben ordenar los resultados en base a el orden ascendente de los apellidos. Sucesivamente ordenamos los nombres de manera ascendente también.

6. Obtener el nombre de los productos junto con el nombre de la categoría a la que pertenecen.

```
SELECT p.nombre, c.nombre AS categoria FROM producto p JOIN categoria c ON p.id_categoria = c.id_categoria;
```

Showing rows 0 - 10 (11 total, Query took 0.0002 seconds.)

```
SELECT p.nombre, c.nombre AS categoria FROM producto p JOIN categoria c ON p.id_categoria = c.id_categoria;
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

nombre	categoria
robot aspirador ceotec conga 1390	robots aspiradores
cucharón de madera de pino	menaje
fregona super	menaje
aspiradora ultra	menaje
escoba nylon	menaje
super glue 3	bricolaje
guantes desechables negros	bricolaje
tiras adhesivas para colgar cuadros	bricolaje
gafas proteccion	bricolaje
taladro easy impact	bricolaje
medidor laser bosch	bricolaje

Seleccionamos los atributos requeridos y especificamos las tablas de las cuales deseamos realizar las consultas. Ya que queremos obtener el nombre de la categoría de otra tabla, tenemos que especificar de que tabla se trata y el que es lo que queremos obtener, buscando el id de la categoría en una tabla y la otra.

7. Obtener el nombre de las localidades junto con el nombre de provincia y nombre de país al que pertenecen.

```
SELECT l.nombre, prov.nombre AS nombre_provincia, pais.nombre AS nombre_pais FROM localidad l JOIN provincia prov ON l.id_provincia = prov.id_provincia JOIN pais pais ON prov.id_pais = pais.id_pais;
```

Showing rows 0 - 11 (12 total, Query took 0.0003 seconds.)

```
SELECT l.nombre, prov.nombre AS nombre_provincia, pais.nombre AS nombre_pais FROM localidad l JOIN provincia prov ON l.id_provincia = prov.id_provincia JOIN pais pais ON prov.id_pais = pais.id_pais;
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

nombre	nombre_provincia	nombre_pais
eida	alicante	españa
benisa	alicante	españa
benidorm	alicante	españa
villena	alicante	españa
silla	valencia	españa
bétera	valencia	españa
bocairente	valencia	españa
burjasot	valencia	españa
benicasim	castellón	españa
burriana	castellón	españa
madrid	madrid	españa
caen	normandía	francia

Seleccionamos los atributos requeridos y especificamos las tablas de las cuales deseamos realizar las consultas. Observamos que igualamos siempre el id para encontrar el nombre de la provincia o país.

- Obtener el id del producto, nombre, importe y cantidad de los productos solicitados en el pedido número 1 que sean de la familia 'menaje'.

```
SELECT lped.id_producto, prod.nombre, lped.importe, lped.cantidad FROM
lineapedido lped JOIN producto prod ON lped.id_producto = prod.id_producto JOIN
categoria cat ON prod.id_categoria = cat.id_categoria WHERE
lped.id_pedido = 1 AND cat.nombre = 'menaje';
```

✓ Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)

```
SELECT lped.id_producto, prod.nombre, lped.importe, lped.cantidad FROM lineapedido lped JOIN
producto prod ON lped.id_producto = prod.id_producto JOIN categoria cat ON prod.id_categoria =
cat.id_categoria WHERE lped.id_pedido = 1 AND cat.nombre = 'menaje';
```

☐ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

id_producto	nombre	importe	cantidad
2	cucharón de madera de pino	0.50	3
3	fregona super	3.25	200
4	aspiradora ultra	150.00	18
5	escoba nylon	1.99	10

Seleccionamos el id\_producto, y filtramos por el id\_pedido. Además, solo buscamos obtener los productos que son de la familia 'menaje'. Recordamos que queremos imprimir el importe y la cantidad además, del id\_producto y el nombre.

OJO: Si deseamos comprobar si se han pillado los registros correctos lo podemos visualizar con el siguiente query:

```
SELECT lped.id_producto, lped.id_pedido, lped.importe, lped.cantidad,
prod.nombre, cat.nombre FROM lineapedido lped JOIN producto prod ON lped.id_producto = prod.id_producto JOIN
categoria cat ON prod.id_categoria = cat.id_categoria WHERE lped.id_pedido = 1 AND cat.nombre = 'menaje';
```

✓ Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)

```
SELECT lped.id_producto, lped.id_pedido, lped.importe, lped.cantidad, prod.nombre, cat.nombre FROM
lineapedido lped JOIN producto prod ON lped.id_producto = prod.id_producto JOIN categoria cat ON
prod.id_categoria = cat.id_categoria WHERE lped.id_pedido = 1 AND cat.nombre = 'menaje';
```

☐ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

id_producto	id_pedido	importe	cantidad	nombre	nombre
2	1	0.50	3	cucharón de madera de pino	menaje
3	1	3.25	200	fregona super	menaje
4	1	150.00	18	aspiradora ultra	menaje
5	1	1.99	10	escoba nylon	menaje

Como podemos observar estamos filtrando por id\_pedido de manera correcta y obteniendo los productos de la familia menaje. Por lo tanto la query que hemos mencionado al principio está obteniendo los resultados de manera correcta.

9. Obtener toda la información (todas las columnas) de los pedidos realizados en el año 2018, ya que han sido realizados por usuarios del dominio gmail.com

```
SELECT * FROM `pedido` WHERE YEAR(fecha) = 2018;
```

Showing rows 0 - 2 (3 total, Query took 0.0002 seconds.)

SELECT \* FROM `pedido` WHERE YEAR(fecha) = 2018;

☐ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

		id_pedido	fecha	id_usuario	id_estado	id_formapago
<input type="checkbox"/>	Edit Copy Delete	14	2018-01-01 00:00:00	1	5	1
<input type="checkbox"/>	Edit Copy Delete	17	2018-06-15 00:00:00	4	5	3
<input type="checkbox"/>	Edit Copy Delete	23	2018-08-08 00:00:00	2	5	3

Para obtener toda la información (todas las columnas) de los pedidos realizados en el año 2018, únicamente habrá que hacer un SELECT \*, especificando la tabla de pedido e indicar el año que queremos visualizar. Para esto simplemente indicamos WHERE YEAR(nombre atributo) = año que queremos visualizar y por lo tanto WHERE YEAR(fecha) = 2018. De esta manera habremos obtenido los resultados deseados.

10. Obtener dni, nombre de usuario, fecha de nacimiento, nombre de localidad, habitantes, nombre de provincia y nombre de país de los usuarios que cumplen todas estas condiciones:
  - Se llaman pedro, jose o Luis
  - Son de una provincia que NO pertenece a la Comunidad Valenciana
  - El dni contiene el número 7
  - La fecha de nacimiento es mayor o igual que el 1 de Junio del 2000

```
SELECT user.dni, user.nombre AS nombre_usuario, user.fechanac, loc.id_localidad, loc.nombre AS nombre_localidad, loc.habitantes, pais.nombre AS nombre_pais FROM usuario user JOIN localidad loc ON user.id_localidad = loc.id_localidad JOIN provincia prov ON loc.id_provincia = prov.id_provincia JOIN pais pais ON prov.id_pais = pais.id_pais WHERE user.dni LIKE '%7%' AND user.nombre IN ('pedro', 'jose', 'luis') AND user.fechanac >= '2000-06-01' AND prov.id_provincia > '3';
```

Showing rows 0 - 0 (1 total, Query took 0.0008 seconds.)

```
SELECT user.dni, user.nombre AS nombre_usuario, user.fechanac, loc.id_localidad, loc.nombre AS nombre_localidad, loc.habitantes, pais.nombre AS nombre_pais FROM usuario user JOIN localidad loc ON user.id_localidad = loc.id_localidad JOIN provincia prov ON loc.id_provincia = prov.id_provincia JOIN pais pais ON prov.id_pais = pais.id_pais WHERE user.dni LIKE '%7%' AND user.nombre IN ('pedro', 'jose', 'luis') AND user.fechanac >= '2000-06-01' AND prov.id_provincia > '3';
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

dni	nombre_usuario	fechanac	id_localidad	nombre_localidad	habitantes	nombre_pais
7077777E	pedro	2001-01-22	13	caen	2000	francia

Para obtener los registros que cumplen los requerimientos debemos considerar lo siguiente: Que el dni contenga el número 7, que los nombres sean Pedro, Jose o Luis, que la fecha de nacimiento sea  $\geq 01/06/2000$  y que la provincia pertenezca a la comunidad valenciana. Observando los id\_provincia podemos ver que las provincias que forman parte de la comunidad valenciana son: alicante (id\_provincia = 1), valencia (id\_provincia = 2) y Castellón (id\_provincia = 3) Es por esto que establecemos el requerimiento de id\_provincia  $> 3$ .

Tras tener claras las restricciones de la consulta (es decir, por lo que va a filtrar) lo único que quedaba por hacer es realizar los diversos JOIN (localidad loc, provincia pov, pais pais).

- Obtener el id del pedido, fecha de pedido, dni del usuario, nombre del estado en el que se encuentra el pedido y nombre de la forma de pago de los pedidos realizados en el año 2022. Si algún pedido tiene null en la columna formapago debe aparecer.

```
SELECT ped.id_pedido, ped.fecha, user.dni AS dni, stat.nombre AS estado, pag.nombre AS formapago FROM pedido ped JOIN usuario user ON user.id_usuario = ped.id_usuario JOIN estado stat ON stat.id_estado = ped.id_estado LEFT JOIN formapago pag ON pag.id_formapago = ped.id_formapago WHERE YEAR(fecha) = 2022;
```

Showing rows 0 - 8 (9 total, Query took 0.0004 seconds.)

```
SELECT ped.id_pedido, ped.fecha, user.dni AS dni, stat.nombre AS estado, pag.nombre AS formapago FROM pedido ped JOIN usuario user ON user.id_usuario = ped.id_usuario JOIN estado stat ON stat.id_estado = ped.id_estado LEFT JOIN formapago pag ON pag.id_formapago = ped.id_formapago WHERE YEAR(fecha) = 2022;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

id_pedido	fecha	dni	estado	formapago
40	2022-06-16 00:00:00	40555444B	pendiente de pago	NULL
39	2022-05-05 00:00:00	50555444R	pendiente de pago	NULL
38	2022-06-06 00:00:00	60606888Q	procesando	tarjeta de crédito
37	2022-08-07 00:00:00	70777555W	en espera	transferencia bancaria
21	2022-10-25 00:00:00	70777777E	completado	tarjeta de crédito
34	2022-04-04 00:00:00	70777777E	procesando	tarjeta de crédito
35	2022-05-05 00:00:00	70777777E	en espera	transferencia bancaria
33	2022-03-03 00:00:00	88808808R	procesando	tarjeta de crédito
36	2022-09-06 00:00:00	88808808R	en espera	transferencia bancaria

Para obtener los registros que cumplen los requerimientos en primer lugar tendremos que especificar la fecha establecida 2022. Posteriormente indicamos los atributos que de los cuales deseamos visualizar los valores (id\_pedido, fecha, dni, estado, formapago). Y a continuación, juntamos las tablas mediante el JOIN. Para poder visualizar los valores NULOS que encontramos bajo formapago, tendremos que realizar un LEFT JOIN. Esto nos ayudará a visualizar todos los



registros que cumplan las condiciones, incluidos aquellos que tengan la forma de pago establecida como nulo.

12. Obtener dni y una sola columna con el nombre y los apellidos de los usuarios.

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

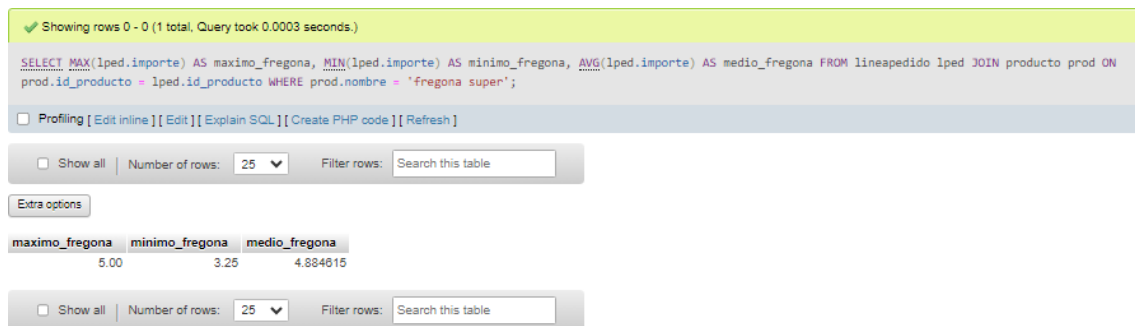


	dni	nombre_completo
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	20111222E	jose garcia gomez
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	30333444W	oscar perez barcelo
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	40555444B	maite barcelo garcia
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	50555444R	luis herrero perez
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	60666888Q	ana romero martinez
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	70777555V	pedro ramirez polo
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	88808808R	pablo gonzalez ortega
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	70777777E	pedro herrero ortega

Para obtener el dni, además de una sola columna que incluya el nombre y los apellidos de los usuarios se hará uso del método CONCAT(atributo1, atributo2, atributo3). Se añaden los , ' ' dentro del concat para añadir un espacio.

13. Obtener el importe máximo, mínimo y medio de las líneas de pedido que tienen el artículo “fregona super”.

```
SELECT MAX(lped.importe) AS maximo_fregona, MIN(lped.importe) AS minimo_fregona, AVG(lped.importe) AS medio_fregona FROM lineapedido lped JOIN producto prod ON prod.id_producto = lped.id_producto WHERE prod.nombre = 'fregona super';
```



maximo_fregona	minimo_fregona	medio_fregona
5.00	3.25	4.884615

Para obtener el precio máximo, mínimo y medio tenemos que emplear las funciones MAX(atributo), MIN(atributo) AVG(atributo). Esto nos imprimirá el máximo, mínimo

y valor medio del atributo seleccionado. Hacemos un JOIN de las tablas, filtramos por el nombre del producto y listo.

14. Obtener id\_producto, nombre de producto, cantidad total que se ha pedido y total de ventas de cada producto.

```
SELECT lp.id_producto, prod.nombre, SUM(lp.cantidad) AS cantidad, SUM(
lp.importe) AS total FROM lineapedido lp JOIN producto prod ON lp.id_p
roducto = prod.id_producto GROUP BY prod.id_producto;
```

SELECT lp.id\_producto, prod.nombre, SUM(lp.cantidad) AS cantidad, SUM(lp.importe) AS total FROM lineapedido lp JOIN producto prod ON lp.id\_producto = prod.id\_producto GROUP BY prod.id\_producto;

☐ Profiling [\[Edit inline\]](#) [\[Edit\]](#) [\[Explain SQL\]](#) [\[Create PHP code\]](#) [\[Refresh\]](#)

☐ Show all | Number of rows: 25 | Filter rows:

Extra options

id_producto	nombre	cantidad	total
1	robot aspirador cecotec conga 1390	377	3500.00
2	oucharón de madera de pino	224	20.90
3	fregona super	454	127.00
4	aspiradora ultra	232	3000.00
5	escoba nylon	208	35.82
6	super glue 3	190	91.01
7	guantes desechables negros	100	160.86
8	tiras adhesivas para colgar cuadros	228	128.80
9	gafas proteccion	231	68.25
10	taladro easy impact	218	838.95
11	medidor laser bosch	204	2722.32

Para obtener el id\_producto, nombre de producto, y la cantidad total que se ha pedido, además del total de ventas de cada uno de ellos. Tenemos que una vez más unir las tablas. Las tablas que uniremos serán la tabla lineapedido y la tabla producto. Lineapedido contendrá las cantidades y los totales vendidos, mientras que producto contendrá el nombre y la información sobre los productos. Para posteriormente organizar / agrupar los productos según el id\_producto utilizaremos GROUP BY y el atributo en cuestión.

15. Obtener id\_categoria, nombre de categoría, mes de venta, año de venta, y total de ventas de cada categoría de producto

```
SELECT c.id_categoria, c.nombre AS nombre_categoria, EXTRACT(MONTH
FROM pe.fecha) AS mes_venta, EXTRACT(YEAR FROM pe.fecha) AS año_ven
ta, SUM(l.cantidad * p.precio) AS ventas_totales FROM categoria c J
OIN producto p ON c.id_categoria = p.id_categoria JOIN lineapedido
l ON p.id_producto = l.id_producto JOIN pedido pe ON l.id_pedido =
pe.id_pedido GROUP BY c.id_categoria, c.nombre, mes_venta, año_vent
a;
```

id_categoria	nombre_categoria	mes_venta	año_venta	ventas_totales
1	robots aspiradores	1	2002	1000.00
1	robots aspiradores	1	2019	4000.00
1	robots aspiradores	1	2021	40000.00
1	robots aspiradores	2	2021	400.00
1	robots aspiradores	3	2022	1400.00
1	robots aspiradores	4	2019	2000.00
1	robots aspiradores	4	2020	2600.00
1	robots aspiradores	4	2021	1200.00
1	robots aspiradores	5	2021	3200.00
1	robots aspiradores	5	2022	4000.00
1	robots aspiradores	6	2018	1400.00
1	robots aspiradores	7	2019	3000.00
1	robots aspiradores	8	2018	200.00
1	robots aspiradores	8	2022	4000.00
1	robots aspiradores	9	2021	6200.00
1	robots aspiradores	10	2021	800.00
2	menaje	1	2002	100.00
2	menaje	1	2016	84.00
2	menaje	1	2018	2109.00
2	menaje	1	2019	25.00
2	menaje	1	2020	688.00
2	menaje	1	2021	52.93
2	menaje	2	2017	52.81
2	menaje	2	2021	146.83
2	menaje	3	2017	904.00

Para obtener el total de ventas de una categoría específica, además de las fechas en las que estas ocurren, se precisa de hacer un JOIN de varias tablas y agrupar los resultados por el id\_categoria. Además, se debe extraer el año y el mes de la fecha que encontramos en el pedido.

- Obtener id\_marca, nombre de marca y la cantidad de artículos vinculados a cada marca.

```
SELECT prod.id_marca, marc.nombre, COUNT(prod.id_producto) AS total_prod_de_marca FROM producto prod JOIN marca marc ON marc.id_marca = prod.id_marca GROUP BY marc.id_marca;
```

id_marca	nombre	total_prod_de_marca
1	hp	1
2	cecotec	1
6	multihogar	1
7	loctite	1
8	arnomed	1
9	atope	1
10	bosch	2

Agrupamos los productos mediante el id de la marca. Además, unimos las tablas producto y marca para contabilizar la cantidad de productos existentes. El uso del COUNT nos ayudará a reflejar la cantidad de productos asociados con cada marca. También se podría haber agrupado por el nombre de la marca, sin embargo al agrupar por el id de la marca, no solo se nos agruparán los datos sino que también se nos ordenará de la tabla.

- Obtener id\_pedido, fecha y cantidad de líneas de cada pedido.

```
SELECT ped.id_pedido, ped.fecha, SUM(lped.cantidad) AS total_cantidad_pedido FROM pedido ped LEFT JOIN lineapedido lped ON ped.id_pedido = lped.id_pedido GROUP BY ped.id_pedido, ped.fecha ORDER BY ped.id_pedido;
```

Showing rows 0 - 24 (40 total, Query took 0.0005 seconds.) [id\_pedido: 1... - 25...]

SELECT ped.id\_pedido, ped.fecha, SUM(lped.cantidad) AS total\_cantidad\_pedido FROM pedido ped LEFT JOIN lineapedido lped ON ped.id\_pedido = lped.id\_pedido GROUP BY ped.id\_pedido, ped.fecha ORDER BY ped.id\_pedido;

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

id_pedido	fecha	total_cantidad_pedido
1	2019-04-28 00:00:00	275
2	2002-01-01 00:00:00	89
3	2021-01-01 00:00:00	232
4	2021-02-02 00:00:00	77
5	2021-03-03 00:00:00	41
6	2021-04-04 00:00:00	68
7	2021-05-05 00:00:00	48
8	2021-06-06 00:00:00	99
9	2020-07-07 00:00:00	42
10	2021-08-08 00:00:00	82
11	2021-09-09 00:00:00	25
12	2021-10-10 00:00:00	59
13	2021-11-11 00:00:00	33
14	2018-01-01 00:00:00	22
15	2019-01-01 00:00:00	57
16	2020-01-01 00:00:00	100
17	2018-06-15 00:00:00	43
18	2019-07-20 00:00:00	89
19	2020-08-21 00:00:00	17
20	2021-09-22 00:00:00	22
21	2022-10-25 00:00:00	75
22	2021-10-24 00:00:00	84
23	2019-08-08 00:00:00	67
24	2019-07-07 00:00:00	36
25	2020-08-08 00:00:00	67

Se realiza un SELECT agrupando los queries por orden de pedido. Además, para obtener la cantidad total pedida (líneas de cada pedido) realizamos un SUM indicando el atributo cantidad. Se hace un join entre las tablas pedido y lineapedido para obtener los atributos fecha desde línea pedido y la cantidad de lineapedido.

18. Obtener id\_usuario y nombre de los usuarios que solo han realizado dos pedidos.

```
SELECT ped.id_usuario, user.nombre FROM pedido ped LEFT JOIN usuario user ON user.id_usuario = ped.id_usuario GROUP BY user.id_usuario HAVING COUNT(ped.id_pedido) = 2;
```

Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)

SELECT ped.id\_usuario, user.nombre FROM pedido ped LEFT JOIN usuario user ON user.id\_usuario = ped.id\_usuario GROUP BY user.id\_usuario HAVING COUNT(ped.id\_pedido) = 2;

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

id_usuario	nombre
1	jose

Se realiza un select en el cual se agrupan dos tablas, la tabla pedido y la tabla usuarios. Buscamos obtener el nombre de la tabla user, la id de usuario (podríamos obtenerla de la tabla user también, sin embargo en este caso lo he hecho desde la tabla pedido). También buscamos contar el id de los pedidos mediante la sentencia HAVING COUNT e igualamos el contador a 2. De esta manera obtenemos únicamente los usuarios que hayan realizado 2 pedidos.

19. Obtener id\_pedido y cantidad de artículos de los pedidos que tienen más de 100 unidades de producto

```
SELECT lp.id_pedido, SUM(lp.cantidad) AS cantidad_total FROM lineapedido lp GROUP BY lp.id_pedido HAVING SUM(lp.cantidad) > 100;
```



Showing rows 0 - 1 (2 total, Query took 0.0003 seconds.)

```
SELECT lp.id_pedido, SUM(lp.cantidad) AS cantidad_total FROM lineapedido lp GROUP BY lp.id_pedido HAVING SUM(lp.cantidad) > 100;
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 | Filter rows: Search this table

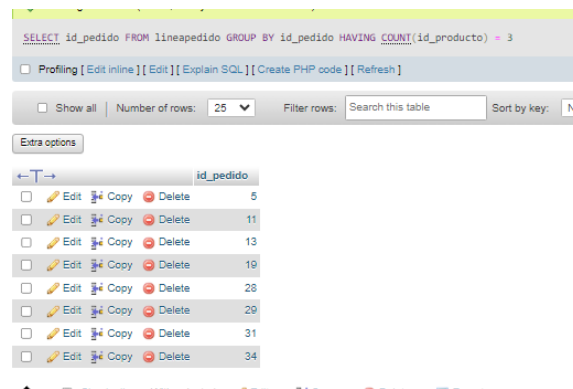
Extra options

	id_pedido	cantidad_total
<input type="checkbox"/>	1	275
<input type="checkbox"/>	3	232

La única tabla a utilizar será la de lineapedido. En esta tabla sumamos la cantidad total realizando la operación SUM(lp.cantidad) y la imprimimos junto a la id del pedido. Al realizar el group por id\_pedido, agrupamos todos los pedidos que comparten id. Finalmente, se realiza una suma de la cantidad de pedidos y obligamos a que solo muestre aquellos resultados que sean mayores a 100.

20. Obtener id\_pedido de los pedidos que tienen 3 productos diferentes en las líneas de pedido.

```
SELECT id_pedido FROM lineapedido GROUP BY id_pedido HAVING COUNT(id_producto) = 3
```



```
SELECT id_pedido FROM lineapedido GROUP BY id_pedido HAVING COUNT(id_producto) = 3
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: No

Extra options

	id_pedido
<input type="checkbox"/>	5
<input type="checkbox"/>	11
<input type="checkbox"/>	13
<input type="checkbox"/>	19
<input type="checkbox"/>	28
<input type="checkbox"/>	29
<input type="checkbox"/>	31
<input type="checkbox"/>	34

Únicamente se observa la tabla lineapedido en la cual agrupamos todos los registros por la id del pedido y contamos la cantidad de pedidos que tienen un contador de id de producto igual a 3 mediante la sentencia HAVING COUNT(id\_producto) = 3;

21. Realiza un informe de ventas por localidades, obteniendo las columnas: id\_localidad, nombre de localidad, total vendido a cada localidad, cantidad de pedidos realizados en cada localidad. Teniendo en cuenta que el año del pedido está comprendido entre 2020 y 2022, las localidades deben tener menos de 50.000 habitantes y deben salir las localidades donde se han

hecho al menos 2 pedidos. El resultado debe ir ordenado por vendido a cada localidad de forma descendente.

```
SELECT loc.id_localidad, loc.nombre, COUNT(DISTINCT ped.id_pedido) AS total_pedidos_localidad, SUM(lp.cantidad * p.precio) AS total_vendido FROM localidad loc JOIN usuario user ON loc.id_localidad = user.id_localidad JOIN pedido ped ON user.id_usuario = ped.id_usuario JOIN lineapedido lp ON ped.id_pedido = lp.id_pedido JOIN producto p ON lp.id_producto = p.id_producto WHERE loc.habitantes < 50000 AND ped.fecha >= '2020-01-01' AND ped.fecha <= '2022-12-31' GROUP BY loc.id_localidad HAVING COUNT(DISTINCT ped.id_pedido) >= 2 ORDER BY total_vendido DESC;
```

Showing rows 0 - 3 (4 total. Query took 0.0008 seconds)

```
SELECT loc.id_localidad, loc.nombre, COUNT(DISTINCT ped.id_pedido) AS total_pedidos_localidad, SUM(lp.cantidad * p.precio) AS total_vendido FROM localidad loc JOIN usuario user ON loc.id_localidad = user.id_localidad JOIN pedido ped ON user.id_usuario = ped.id_usuario JOIN lineapedido lp ON ped.id_pedido = lp.id_pedido JOIN producto p ON lp.id_producto = p.id_producto WHERE loc.habitantes < 50000 AND ped.fecha >= '2020-01-01' AND ped.fecha <= '2022-12-31' GROUP BY loc.id_localidad HAVING COUNT(DISTINCT ped.id_pedido) >= 2 ORDER BY total_vendido DESC;
```

☐ Profiling [\[Edit inline\]](#) [\[Edit\]](#) [\[Explain SQL\]](#) [\[Create PHP code\]](#) [\[Refresh\]](#)

☐ Show all | Number of rows: 25 | Filter rows:

Extra options

id_localidad	nombre	total_pedidos_localidad	total_vendido
2	silla	10	63892.93
3	elida	3	11044.94
7	villena	2	6365.11
13	caen	3	3317.62

En esta sentencia tenemos que tener varias cosas en cuenta.

1. Los datos a obtener
2. Los datos por los que tenemos que filtrar

Primero, observamos que nos piden obtener el id de la localidad, el nombre, el número total de pedidos, y la suma total de lo vendido. Para esto necesitaremos las tablas localidad, usuario (ya que queremos observar los pedidos realizados) y línea pedido (Se realizan los JOIN necesarios). Para el total calculamos la cantidad multiplicado por el precio del producto por lo tanto empleamos un SUM y realizamos la operación dentro de este. Al contar el total de pedidos el DISTINCT lo que hará es evitar registros dobles, encontrando entidades únicas.

Tras haber identificado los atributos procedemos a filtrar. Por lo tanto realizamos lo siguiente:

1. Agrupamos los datos por el id de la localidad (de esta manera se ordenaran en función de el número de la localidad).
2. Especificamos que el orden de registros debe ser descendente en función del total\_vendido (que hemos calculado al indicar que atributos queremos visualizar)
3. Especificamos las fechas de pedidos que queremos obtener y visualizar.
4. Especificamos que las únicas localidades que queremos visualizar son aquellas que tienen menos de 50000 habitantes.
5. Especificamos que queremos obtener las localidades en las cuales se han realizado al menos 2 videos (>=2)

Tras haber implementado esta lógica obtendremos el resultado deseado.