

4 Subconsultas

```
SELECT pelicula.titulo, pelicula.idpelicula, COUNT(peliculainterprete.idinterprete) AS num_interpretes
FROM pelicula INNER JOIN peliculainterprete
ON pelicula.idpelicula = peliculainterprete.idpelicula
WHERE pelicula.idpelicula IN ( SELECT idpelicula
FROM peliculainterprete WHERE idinterprete = 19 )
GROUP BY pelicula.idpelicula
HAVING COUNT(peliculainterprete.idinterprete) >= 5;
```

Links de Interés:

Subqueries

Content reproduced on this site is the property of its respective owners, and this content is not reviewed in advance by MariaDB. The views, information and opinions expressed by

🔗 <https://mariadb.com/kb/en/subqueries/>

Understanding Sql subqueries - w3resource

A subquery is a SQL query nested inside a larger query. A subquery may occur in : - A SELECT clause - A FROM clause - A WHERE clause The

w3r <https://www.w3resource.com/sql/subqueries/understanding-sql-subqueries.php>



Subconsultas (SQL Server) - SQL Server

Se aplica a: SQL Server (todas las versiones admitidas) Azure SQL Database Azure SQL Managed Instance Azure Synapse Analytics

🌐 <https://learn.microsoft.com/es-es/sql/relational-databases/performance/subqueries?view=sql-server-ver16>



Writing Subqueries in SQL | Advanced SQL - Mode

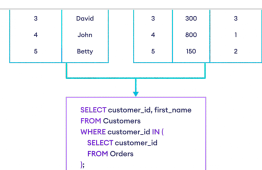
Starting here? This lesson is part of a full-length tutorial in using SQL for Data Analysis. Check out the beginning. In this lesson we'll cover: In this lesson, you will continue to

M <https://mode.com/sql-tutorial/sql-sub-queries/>



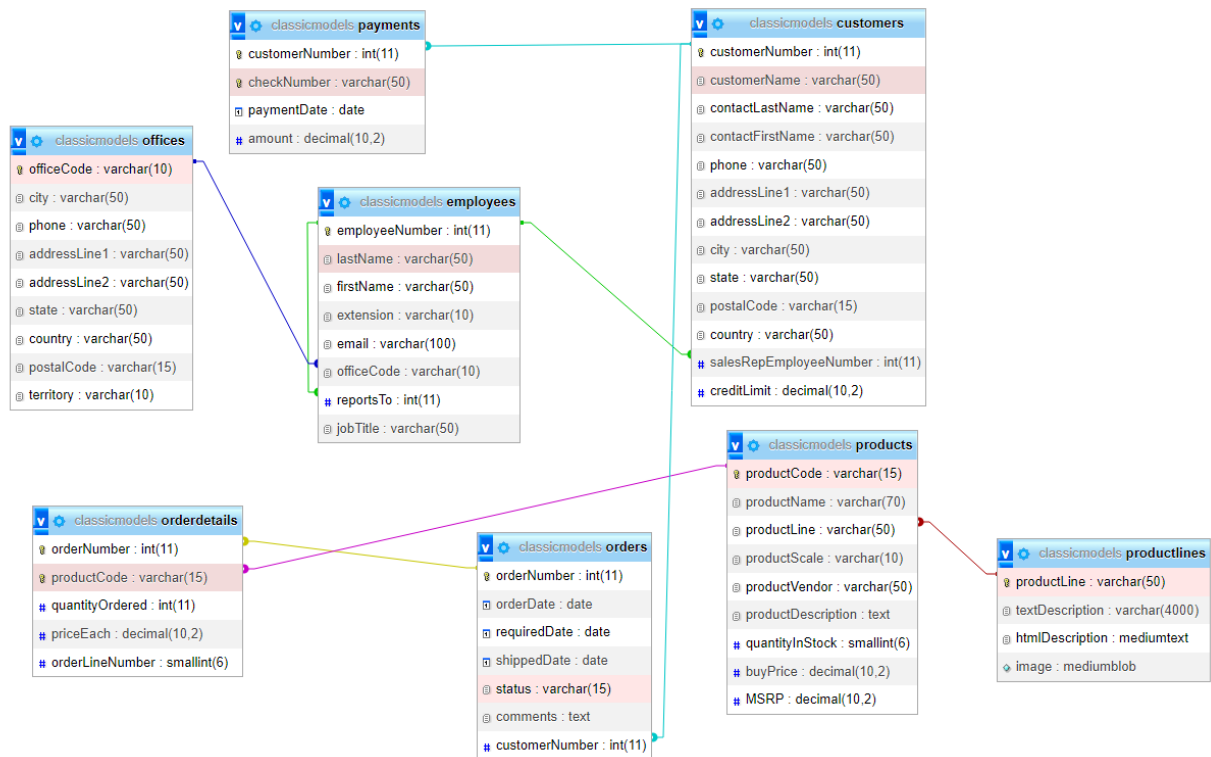
SQL Subquery

In SQL, it's possible to place a SQL query inside another query known as subquery.



<https://www.youtube.com/watch?v=nJIEIzF7tDw>

Diseño de la base de datos:



Existen subconsultas escalonadas, de lista y correlacionadas.

Las escalonadas: El SELECT de la subconsulta devuelve una única columna con un único registro que luego se usa en la consulta padre como criterio o comparación.

Ej: muéstrame el nombre y la sección de los productos con un precio superior a la media. En la subconsulta necesitamos obtener la media para usar luego la media como una comparación en la consulta padre:

La subconsulta sería esta que nos devuelve un único campo, la media: `SELECT AVG(precio) FROM productos`.

La consulta padre sería la siguiente: `SELECT nombreArticulo, seccion FROM productos WHERE precio > (SELECT AVG(precio) FROM productos)`

Aquí como estamos comparando la subconsulta, es en la parte WHERE

Las de lista: Ahora en vez de devolvernos un único registro, nos devuelve una lista de registros. Con esto solemos usar los operadores IN, ANY y ALL.

Ej: Dame los artículos con precio superior a todos los artículos de cerámica. Primero tenemos que hacer una subconsulta para averiguar el precio de todos los artículos de cerámica y con la lista de precios, podemos compararlos en la consulta padre y nos devolverá los artículos que tienen un precio superior a todos ellos:






La subconsulta sería: `SELECT precio FROM productos WHERE seccion='Cerámica'`

La consulta padre sería: `SELECT * FROM productos WHERE precio > ALL (SELECT precio FROM productos WHERE seccion='Cerámica')`

Con el "> ALL" para comparar los precios de la subconsulta, lo que hacemos es que la consulta padre nos de los productos que tienen un precio mayor a todos los productos de cerámica, por lo que todos los que nos devuelva tendrán un precio mayor al producto más caro de cerámica. Si fuera "> ANY" nos devolvería los productos que tuvieran un precio mayor a alguno de los productos, por lo que nos devolvería todos los productos con un precio mayor al producto de cerámica más barato.

SUBCONSULTAS EN LA PARTE FROM

- `SELECT p.customerNumber, SUM(p.amount) FROM payments p GROUP BY p.customerNumber`
(Esta consulta agrupa los pagos de cada cliente)

	customerNumber	SUM(p.amount)
<input type="checkbox"/>   	103	22314.36
<input type="checkbox"/>   	112	80180.98
<input type="checkbox"/>   	114	180585.07
<input type="checkbox"/>   	119	116949.68
<input type="checkbox"/>   	121	104224.79

- `SELECT p.customerNumber, c.customerName, SUM(p.amount) FROM payments p JOIN customers c USING(customerNumber) GROUP BY p.customerNumber` (Esta consulta no usa subconsultas pero nos va a servir en la siguiente que sí las usa. Esta consulta agrupa las sumas de pago por cliente de tal forma que aparece los datos del cliente junto al total de pagos que nos ha realizado, gracias a la cláusula group by)

customerNumber	customerName	SUM(p.amount)
103	Atelier graphique	22314.36
112	Signal Gift Stores	80180.98
114	Australian Collectors, Co.	180585.07
119	La Rochelle Gifts	116949.68
121	Baane Mini Imports	104224.79

- **SELECT MAX(suma) FROM (SELECT p.customerNumber,c.customerName, SUM(p.amount) AS suma FROM payments p JOIN customers c USING(customerNumber) GROUP BY p.customerNumber) AS sumasAgrupadas** (Utiliza la consulta anterior como subconsulta en el FROM de la principal, es decir como si fuera una tabla cualquiera, poniendo alias al campo de suma podemos aplicarle cualquier función de agregado, como por ejemplo en este caso MAX que devuelve el máximo de las sumas de los diferentes grupos de la subconsulta. Dime cual es el cliente que ha gastado más dinero.)

MAX(suma)
715738.98

- **SELECT p.customerNumber,c.customerName, SUM(p.amount) AS suma FROM payments p JOIN customers c USING(customerNumber) GROUP BY p.customerNumber ORDER BY suma DESC** (Sobre la consulta inicial aplicamos un ordenamiento descendente con la cláusula DESC en el order by. Si quisiéramos que fuera ascendente usaremos ASC. Es mejor poner AS que no ponerlo)

customerNumber	customerName	suma ▾ 1
141	Euro+ Shopping Channel	715738.98
124	Mini Gifts Distributors Ltd.	584188.24
114	Australian Collectors, Co.	180585.07
151	Muscle Machine Inc	177913.95
148	Dragon Souvenirs, Ltd.	156251.03
323	Down Under Souvenirs, Inc	154622.08
187	AV Stores Co	148410.09

SUBCONSULTAS EN LA PARTE WHERE

- **SELECT p.* FROM payments p WHERE p.amount >= ALL (SELECT p2.amount FROM payments p2)** (Muestra los datos del pago con la cantidad más alta. La subconsulta devuelve todos los importes, y

con la restricción \geq ALL obligamos a que el registro obtenido en la consulta principal tenga un importe mayor o igual que todos los valores que devuelve la subconsulta. Las subconsultas empeoran el rendimiento comparado con los JOIN. La consulta contraria a esta sería cambiando el \geq ALL por $<$ ALL)

←T→	customerNumber	checkNumber	paymentDate	amount
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	141	JE105477	2005-03-18	120166.58

- **SELECT p.* FROM payments p WHERE p.amount < ALL (SELECT p2.amount FROM payments p2)**
(Muestra los datos del pago de "payments p" que sean (totalmente=ALL) diferentes a los de "payments p2". Al ser comparada una base de datos consigo misma, no nos aparece ningún resultado. Todas las tablas tienen ser diferentes a la que se compara, pero hay un registro igual a si mismo siempre si se compara con la misma base de datos, por eso no da ninguno. (ANY y ALL son casi complementarios porque "ALL" es como decir que "es estrictamente necesario que se cumpla en todos los casos" y "ANY" quiere decir que "por lo menos se tiene que cumplir en algún caso"))

customerNumber	checkNumber	paymentDate	amount
Operaciones sobre los resultados de la consulta			

- **SELECT p.* FROM payments p WHERE p.amount < ANY (SELECT p2.amount FROM payments p2)**
(Muestra los datos del pago de "payments p" que sean (por lo menos en algún caso=ANY) diferentes a los de "payments p2". Al ser comparada una base de datos consigo misma, nos aparecen todos los casos porque aunque haya una coincidencia (consigo mismo) el resto de casos son diferentes y se cumple que por lo menos haya un caso diferente, como vemos, es justo lo contrario que sucede con ALL)

←T→	customerNumber	checkNumber	paymentDate	amount
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	103	HQ336336	2004-10-19	6066.78
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	103	JM555205	2003-06-05	14571.44
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	103	OM314933	2004-12-18	1676.14
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	112	BO864823	2004-12-17	14191.12
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	112	HQ55022	2003-06-06	32641.98
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	112	ND748579	2004-08-20	33347.88
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	114	GG311455	2003-05-20	15864.03

- **SELECT o.* FROM offices o WHERE o.officeCode <> ALL (SELECT o2.officeCode FROM offices o2)** (Muestra los datos de la oficina "o" que sean (totalmente=ALL) diferentes a los de la oficina "o2". Al ser comparada una base de datos consigo misma, no nos aparece ningún resultado. Todas las tablas tienen ser diferentes a la que se compara, pero hay un registro igual a si mismo siempre si se compara con la misma base de datos, por eso no da ninguno como habíamos visto en el ejemplo de payments)
 **Ahora usamos offices en vez de payments porque es una tabla más fácil de usar ya que solo tiene una primary key (officeCode) mientras que payments tiene dos (customerNumber y checkNumber) y para ejemplificar se verá más claro en los siguientes ejemplos:

officeCode	city	phone	addressLine1	addressLine2	state	country	postalCode	territory
Operaciones sobre los resultados de la consulta								

- **SELECT o.* FROM offices o WHERE o.officeCode <> ALL (SELECT o2.officeCode FROM offices o2 WHERE o2.officeCode <> o.officeCode)** (Es la misma consulta que la anterior pero con la condición WHERE en la subconsulta. Ahora mostramos los datos de la oficina "o" que sean (totalmente=ALL) diferentes a los de la oficina "o2" que tengan un officeCode diferente. Con esto conseguimos que nos den de resultado todos los datos porque aunque comparamos una base de datos consigo misma, no estamos comparando cada tabla consigo misma porque hemos puesto el WHERE con la primary key. Cada tabla tiene una única primary key por lo que conseguimos que no se compare consigo misma pero si con las demás y que por ende, cumpla la condición de que sean totalmente(ALL) diferentes. Esto funciona bien porque el WHERE se usa con la primary key, si fuera con cualquier otro atributo como country ya no sería posible ya que varias tablas pueden tener el mismo país y nos podría dar lugar a un error)(A esto se le llama **Consulta relacionada**, y es mejor evitarla porque no es eficiente)

	officeCode	city	phone	addressLine1	addressLine2	state	country	postalCode	territory
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	1	San Francisco	+1 650 219 4782	100 Market Street	Suite 300	CA	USA	94080	NA
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	2	Boston	+1 215 837 0825	1550 Court Place	Suite 102	MA	USA	02107	NA
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	3	NYC	+1 212 555 3000	523 East 53rd Street	apt. 5A	NY	USA	10022	NA
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	4	Paris	+33 14 723 4404	43 Rue Jouffroy D'abbans	NULL	NULL	France	75017	EMEA
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	5	Tokyo	+81 33 224 5000	4-1 Kioicho	NULL	Chiyoda-Ku	Japan	102-8578	Japan
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	6	Sydney	+61 2 9264 2451	5-11 Wentworth Avenue	Floor #2	NULL	Australia	NSW 2010	APAC
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	7	London	+44 20 7877 2041	25 Old Broad Street	Level 7	NULL	UK	EC2N 1HN	EMEA
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	8	Melbourne			NULL	casado	Australia	666	

- **SELECT o.* FROM offices o WHERE o.officeCode = ANY (SELECT o2.officeCode FROM offices o2 WHERE o2.officeCode <> o.officeCode)** (Es la misma que la anterior pero en vez de <>ALL con =ANY. Ahora muestra los datos de la oficina "o" que sean (por lo menos en algún caso=ANY) iguales a los de la oficina "o2" que tengan un officeCode diferente. Ahora no nos sale ningún resultado porque

estamos evitando que una tabla se compare con ella misma, pero si permitimos que se compare con las otras diferentes, al escribir el **WHERE** en la subconsulta)

officeCode	city	phone	addressLine1	addressLine2	state	country	postalCode	territory
------------	------	-------	--------------	--------------	-------	---------	------------	-----------

Operaciones sobre los resultados de la consulta

- **SELECT p.* FROM payments p WHERE p.amount <= ANY (SELECT p2.amount FROM payments p2 WHERE p2.customerNumber > 103)** (Esta consulta hace uso de la cláusula **ANY**. Seleccionará aquellos pagos cuya cantidad sea menor que alguna de las cantidades devuelta por la subconsulta, en este caso las cantidades de los pagos del cliente con clave de cliente 103)

	customerNumber	checkNumber	paymentDate	amount
<input type="checkbox"/> Editar Copiar Borrar	103	HQ336336	2004-10-19	6066.78
<input type="checkbox"/> Editar Copiar Borrar	103	JM555205	2003-06-05	14571.44
<input type="checkbox"/> Editar Copiar Borrar	103	OM314933	2004-12-18	1676.14
<input type="checkbox"/> Editar Copiar Borrar	112	BO864823	2004-12-17	14191.12
<input type="checkbox"/> Editar Copiar Borrar	114	NP603840	2003-05-31	7565.08
<input type="checkbox"/> Editar Copiar Borrar	121	FD317790	2003-10-28	1491.38

- **SELECT e.* FROM employees e WHERE EXISTS (SELECT * FROM offices o WHERE e.officeCode = o.officeCode)** (La cláusula **EXISTS** es boolean, por lo que la subconsulta si se cumple nos da true y si no, false. No tenemos que ocuparnos de la parte **SELECT** de la subconsulta, la podemos dejar con un asterisco. El **EXISTS** va a comprobar si la subconsulta se cumple o no con un valor boolean, por lo que el **WHERE** que importa es el de la subconsulta. En este ejemplo: Nos muestra todos los atributos de la base de datos de employees e, ya que si que existe y se cumple la condición de que haya un empleado con el mismo officeCode que el de su propia oficina. Es lo mismo que un **LEFT JOIN** pero haciendo una consulta relacionada porque lleva mucho más tiempo y es mucho menos eficaz. (Aquí nos aparece hasta el atributo jobTitle y en la siguiente con todos los atributos porque aparece un **LEFT JOIN** y también porque no estamos cogiendo los datos de la oficina en este momento)

	employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
<input type="checkbox"/> Editar Copiar Borrar	1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
<input type="checkbox"/> Editar Copiar Borrar	1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
<input type="checkbox"/> Editar Copiar Borrar	1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
<input type="checkbox"/> Editar Copiar Borrar	1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
<input type="checkbox"/> Editar Copiar Borrar	1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
<input type="checkbox"/> Editar Copiar Borrar	1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep

- **SELECT e.* FROM employees e LEFT JOIN offices o USING(officeCode)** (Es lo mismo que la anterior pero mucho más simple al usar el **LEFT JOIN**)

officeCode	employeeNumber	lastName	firstName	extension	email	reportsTo	jobTitle	city	phone	addressLine1	addressLine2	state	country	postalCode	territory
1	1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	NULL	President	San Francisco	+1 650 219 4782	100 Market Street	Suite 300	CA	USA	94080	NA
1	1056	Patterson	Mary	x4611	mpatterson@classicmodelcars.com	1002	VP Sales	San Francisco	+1 650 219 4782	100 Market Street	Suite 300	CA	USA	94080	NA
1	1076	Firrelli	Jeff	x5273	jfirrelli@classicmodelcars.com	1002	VP Marketing	San Francisco	+1 650 219 4782	100 Market Street	Suite 300	CA	USA	94080	NA
6	1088	Patterson	William	x4871	wopatterson@classicmodelcars.com	1056	Sales Manager	Sydney	+61 2 9264 2451	5-11 Wentworth Avenue	Floor #2	NULL	Australia	NSW 2010	APAC

Cláusula HAVING para GROUP BY

- **SELECT p.customerNumber, c.customerName, SUM(p.amount) total FROM payments p JOIN customers c USING(customerNumber) WHERE p.customerNumber=103 GROUP BY p.customerNumber** (Unimos payments con customers poniéndole una restricción de solo muestre los clientes que tengan un customerNumber de 103 y las agrupe usando la primarykey de payments (que en este caso es la que también se usa para hacer el **JOIN**), ahora nos muestra el customerNumber que es igual a todos los casos, el customerName que es igual también y hace la suma de todos los amounts en donde salga el customerNumber de 103. Aplicamos una restricción en el **WHERE**, y esta se evalúa antes de hacer el grupo. Con el **WHERE** la restricción va a los registros antes de formalizar los grupos, por lo que **el WHERE no afecta al grupo pero si a los registros**. Suma solo los datos del customerNumber 103. Con este ejemplo vemos la diferencia entre **HAVING** y **WHERE**. Este es un ejemplo de que podemos aplicar cualquier restricción usando el **GROUP BY**)

customerNumber	customerName	total
103	Atelier graphique	22314.36

- **SELECT p.customerNumber, c.customerName, SUM(p.amount) total FROM payments p JOIN customers c USING(customerNumber) GROUP BY p.customerNumber HAVING total>100000** (Nos da de resultado todos los customers que tengan un importe total mayor de 100000 (que la suma total de todos sus importes sea mayor a 100000). La cláusula **HAVING** es como un **WHERE** que en vez de ser para registros, es para un grupo. Primero haces un grupo según su "customerNumber" (tiene que ser la primary key de lo que vaya en **FROM**, en este caso payments) sumándolos individualmente, y luego con el **HAVING** se hace una restricción que solo afecta a la suma del grupo. Con **HAVING** hacemos la restricción en el grupo en su atributo "total")

customerNumber	customerName	total
114	Australian Collectors, Co.	180585.07
119	La Rochelle Gifts	116949.68
121	Baane Mini Imports	104224.79
124	Mini Gifts Distributors Ltd.	584188.24
131	Land of Toys Inc.	107639.94

- **SELECT** p.customerNumber, c.customerName, SUM(p.amount) total **FROM** payments p **JOIN** customers c **USING**(customerNumber) **GROUP BY** p.customerNumber **HAVING** total **BETWEEN** 50000 **AND** 60000 (Nos da de resultado todos los customers que tengan un importe total entre 50000 y 60000. Con **GROUP BY** podemos utilizar también **COUNT()**, **MAX()**, **MIN()**, **SUM()**, **AVG()**...)

customerNumber	customerName	total
204	Online Mini Collectables	55577.26
256	Auto Associés & Cie.	58876.41
333	Australian Gift Network, Co	55190.16
339	Classic Gift Ideas, Inc	57939.34
357	GiftsForHim.com	56662.38
450	The Sharp Gifts Warehouse	59551.38
452	Mini Auto Werke	51059.99
484	Iberia Gift Imports, Corp.	50987.85

- **SELECT** p.customerNumber, c.customerName, SUM(p.amount) total **FROM** payments p **JOIN** customers c **USING**(customerNumber) **GROUP BY** p.customerNumber **HAVING** total **BETWEEN** 50000 **AND** 60000 **ORDER BY** total **ASC** (Nos da de resultado todos los customers que tengan un importe total entre 50000 y 60000 por orden ascendente)

customerNumber	customerName	total ▲ 1
484	Iberia Gift Imports, Corp.	50987.85
452	Mini Auto Werke	51059.99
333	Australian Gift Network, Co	55190.16
204	Online Mini Collectables	55577.26
357	GiftsForHim.com	56662.38
339	Classic Gift Ideas, Inc	57939.34
256	Auto Associés & Cie.	58876.41
450	The Sharp Gifts Warehouse	59551.38

- **SELECT** p.customerNumber, c.customerName, SUM(p.amount) total **FROM** payments p **JOIN** customers c **USING**(customerNumber) **GROUP BY** p.customerNumber **HAVING** total **BETWEEN** 50000 **AND** 60000 **ORDER BY** total **DESC** (Nos da de resultado todos los customers que tengan un importe total entre 50000 y 60000 por orden descendente)

customerNumber	customerName	total ▾ 1
450	The Sharp Gifts Warehouse	59551.38
256	Auto Associés & Cie.	58876.41
339	Classic Gift Ideas, Inc	57939.34
357	GiftsForHim.com	56662.38
204	Online Mini Collectables	55577.26
333	Australian Gift Network, Co	55190.16
452	Mini Auto Werke	51059.99
484	Iberia Gift Imports, Corp.	50987.85

- **SELECT** p.customerNumber, c.customerName, SUM(p.amount) total **FROM** payments p **JOIN** customers c **USING**(customerNumber) **GROUP BY** p.customerNumber **HAVING** total **BETWEEN** 50000 **AND** 60000 **ORDER BY** total **DESC LIMIT** 3 (Nos da de resultado los tres primeros customers que tengan un importe total entre 50000 y 60000 por orden descendente. **LIMIT** es para limitar la cantidad de tablas de customers que nos da de solución: **LIMIT** 3 son los tres primeros y **LIMIT** 40 son los cuarenta primeros)

customerNumber	customerName	total ▾ 1
450	The Sharp Gifts Warehouse	59551.38
256	Auto Associés & Cie.	58876.41
339	Classic Gift Ideas, Inc	57939.34

Lo mejor es hacer que **SELECT** esté en la primera línea, **FROM** en la segunda (si haceos una subconsulta, en la línea siguiente), luego **JOIN** aparte, **GROUP BY** aparte y así

- (1)**SELECT** c.customerNumber, c.customerName, COUNT(o.orderNumber) numeroPedidos **FROM** orders o **JOIN** customers c **USING**(customerNumber) **GROUP BY** c.customerNumber **HAVING** numeroPedidos **>=** 4 **ORDER BY** numeroPedidos **DESC** (Selecciona los clientes con el

número (o sea la cuenta) de pedidos que nos han realizado de forma que tengan al menos 4 pedidos y además ordenarlos por la cuenta de pedidos que tengan descendentemente. Era un ejercicio

- Otra forma de hacer esa consulta sería: **SELECT c.customerNumber, c.customerName, COUNT(o.orderNumber) cuenta_de_pedidos FROM customers c JOIN orders o USING(customerNumber) GROUP BY c.customerNumber HAVING cuenta_de_pedidos >= 4 ORDER BY numeroPedidos DESC**

customerNumber	customerName	numeroPedidos ▼ 1
141	Euro+ Shopping Channel	26
124	Mini Gifts Distributors Ltd.	17
114	Australian Collectors, Co.	5
148	Dragon Souvenirs, Ltd.	5
353	Reims Collectables	5

- (2) Primero obtengo lo clientes con lo que me deben totalizado:

SELECT o.customerNumber, SUM(od.priceEach*od.quantityOrdered) debe FROM orders o JOIN orderdetails od USING(orderNumber) GROUP BY o.customerNumber

customerNumber	debe
103	22314.36
112	80180.98
114	180585.07
119	158573.12
121	104224.79
124	591827.34
128	75937.76
129	66710.56
131	149085.15
141	820689.54

Obtengo los que han pagado con una consulta similar a la anterior pero relacionando clientes y pagos

SELECT c.customerNumber, SUM(p.amount) totalPagado FROM customers c JOIN payments p USING(customerNumber) GROUP BY c.customerNumber

customerNumber	totalPagado
103	22314.36
112	80180.98
114	180585.07
119	116949.68
121	104224.79
124	584188.24
128	75937.76
129	66710.56

Solo nos falta tratar estas dos consultas como subconsultas, como si fueran tablas en el FROM. Haríamos JOIN entre las dos mediante el campo customerNumber y en el WHERE de la consulta principal añadiremos una restricción que devuelva los registros en los que lo que debe es mayor que lo que ha pagado.

—

Si existiese una tabla debe y otra tabla pagado

Select customerNumber from debe d join pagado p using(customerNumber)

where d.debe>p.totalPagado

SELECT customerNumber, customerName, pagado.totalPagado, debe.debe **FROM** (**SELECT** o.customerNumber, SUM(od.priceEach*od.quantityOrdered) debe **FROM** orders o **JOIN** orderdetails od **USING**(orderNumber) **GROUP BY** o.customerNumber) debe **JOIN** (**SELECT** c.customerNumber, SUM(p.amount) totalPagado **FROM** customers c **JOIN** payments p **USING**(customerNumber) **GROUP BY** c.customerNumber) pagado **USING**(customerNumber) **JOIN** customers c **USING**(customerNumber) **WHERE** debe.debe>pagado.totalPagado (Ver aquellos clientes que nos deben dinero. Era un ejercicio)

customerNumber	customerName	totalPagado	debe
119	La Rochelle Gifts	116949.68	158573.12
124	Mini Gifts Distributors Ltd.	584188.24	591827.34
131	Land of Toys Inc.	107639.94	149085.15
141	Euro+ Shopping Channel	715738.98	820689.54
144	Volvo Model Replicas, Co	43680.65	66694.82
145	Danish Wholesale Imports	107446.50	129085.12
157	Diecast Classics Inc.	98509.25	104358.69
166	Handii Gifts & Co	105420.57	107746.75

- También se puede hacer así: **SELECT** c.customerNumber, c.customerName **FROM** customers c **WHERE** (**SELECT** SUM(p.amount) **FROM** payments p **WHERE** p.customerNumber=c.customerNumber) < (**SELECT** SUM(od.quantityOrdered*od.priceEach) **FROM** orders o **JOIN** orderdetails od **USING**(orderNumber) **WHERE**

o.customerNumber=c.customerNumber) (Aquí utilizamos una consulta correlacionada. En el *WHERE* es donde ponemos todas las correlaciones. Tienes que tener cuidado de cómo las haces. La desventaja que tiene es que no podemos saber los totales y no podemos saber que está bien, solo podemos confiar en la consulta. Solo nos sale el “customerNumber” y el “customerName”, no podemos saber el “totalPagado” y el “debe”, porque en teoría solo estas en customer. **Las consultas correlacionadas son más pesadas y tardan bastante más.**)

	customerNumber	customerName
<input type="checkbox"/> Editar Copiar Borrar	119	La Rochelle Gifts
<input type="checkbox"/> Editar Copiar Borrar	124	Mini Gifts Distributors Ltd.
<input type="checkbox"/> Editar Copiar Borrar	131	Land of Toys Inc.
<input type="checkbox"/> Editar Copiar Borrar	141	Euro+ Shopping Channel
<input type="checkbox"/> Editar Copiar Borrar	144	Volvo Model Replicas, Co
<input type="checkbox"/> Editar Copiar Borrar	145	Danish Wholesale Imports
<input type="checkbox"/> Editar Copiar Borrar	157	Diecast Classics Inc.
<input type="checkbox"/> Editar Copiar Borrar	166	Handji Gifts& Co
<input type="checkbox"/> Editar Copiar Borrar	201	UK Collectables, Ltd.

- ```
SELECT c. customerName , SUM (od. quantityOrdered * od. quantityOrdered) AS totalDebe,
SUM (p. amount) AS totalPagado, SUM (p. amount - od. quantityOrdered * od. quantityOrdered) AS
debe FROM payments p JOIN customers c USING (customerNumber) JOIN orders o
USING (customerNumber) JOIN orderdetails od USING (orderNumber) GROUP BY p. customerNumber HAVING
totalDebe > totalPagado
```

(Este ejercicio va mal porque debería de resolverse con subconsultas y evitar los JOINS porque aquí, un pago se cuenta varias veces por lo que da un resultado erróneo debido a los JOINS ya que no se ha podido separar cada pago de un objeto o varios bien. Se quería resolver: Ver aquellos clientes que nos deben dinero. Si ves que hay más de un JOIN que hacer (unir dos tablas) siempre haz subconsultas, inclínate a hacer JOINS en distintas subconsultas ya que el JOIN está controlado y no se hace una repetición como en este caso, sobretudo para las SUM() o CONT() en el SELECT o comparar sumas en el WHERE. Tener mucho cuidado con los atributos del SELECT porque si en vez de SUM(p.amount) pongo solo p.amount, me daría todo mal porque no sumaría todos los pagos y es lo que hice mal al principio. Era un ejercicio QUE RESOLVÍ MAL )

| customerName                                    | totalDebe | totalPagado | debe |
|-------------------------------------------------|-----------|-------------|------|
| Operaciones sobre los resultados de la consulta |           |             |      |


- (3) **SELECT** pl.\*, **AVG**(p.buyPrice) media **FROM** productlines pl **JOIN** products p **USING**(productLine) **GROUP BY** pl.productLine **HAVING** media **BETWEEN 40 AND 50** (Selecciona las líneas de producto cuya media de precio esta entre 40 y 50 euros, debe figurar el la clave de línea

de producto y la descripción de la línea. **Cuidado con qué haces el HAVING porque tiene que estar en el SELECT.** Es un ejercicio)

- También se puede hacer: **SELECT** pl.productLine, pl.textDescription, **AVG**(p.buyPrice) Media\_precio **FROM** productLines pl **JOIN** products p **USING**(productLine) **GROUP BY** pl.productLine **HAVING** Media\_precio **BETWEEN** 40 **AND** 50

| productLine  | textDescription                                       | htmlDescription | image | media     |
|--------------|-------------------------------------------------------|-----------------|-------|-----------|
| Planes       | Unique, diecast airplane and helicopter replicas s... | NULL            | NULL  | 49.629167 |
| Ships        | The perfect holiday or anniversary gift for execut... | NULL            | NULL  | 47.007778 |
| Trains       | Model trains are a rewarding hobby for enthusiasts... | NULL            | NULL  | 43.923333 |
| Vintage Cars | Our Vintage Car models realistically portray autom... | NULL            | NULL  | 46.066250 |

- (4) **SELECT** customerNumber, customerName, pagado.totalPagado, debe.debe **FROM** (**SELECT** o.customerNumber, **SUM**(od.priceEach\*od.quantityOrdered) debe **FROM** orders o **JOIN** orderdetails od **USING**(orderNumber) **GROUP BY** o.customerNumber) debe **JOIN** (**SELECT** c.customerNumber, **SUM**(p.amount) totalPagado **FROM** customers c **JOIN** payments p **USING**(customerNumber) **GROUP BY** c.customerNumber) pagado **USING**(customerNumber) **JOIN** customers c **USING**(customerNumber) **WHERE** debe.debe<pagado.totalPagado (Mostrar los clientes que nos han pagado de más por sus pedidos. Sería lo mismo que el ejercicio 2 pero comparamos con < en vez de con >. Es un ejercicio)

| customerNumber                                                                                  | customerName | totalPagado | debe |
|-------------------------------------------------------------------------------------------------|--------------|-------------|------|
| Operaciones sobre los resultados de la consulta                                                 |              |             |      |
|  Crear vista |              |             |      |

- (5) **SELECT** e.firstName nombreEmpleado, e.lastName ap1Empleado, c.contactFirstName nombreCliente, c.contactLastName ap1Cliente **FROM** customers c **JOIN** employees e **ON** (e.firstName=c.contactFirstName **OR** e.lastName=c.contactLastName) (Mostrar los empleados que se llamen igual que alguno de los clientes o que se apellidan igual que alguno de sus clientes. Es un ejercicio)
- También se puede hacer de la siguiente manera: **SELECT** e.firstName nombreEmpleado,e.lastName ap1Empleado, c.contactFirstName nombreCliente, c.contactLastName ap1Cliente **FROM** customers c **JOIN**

employees e ON (e.firstName=c.contactFirstName ) UNION SELECT e.firstName  
 nombreEmpleado,e.lastName ap1Empleado,  
 c.contactFirstName nombreCliente, c.contactLastName ap1Cliente FROM customers c JOIN  
 employees e ON (e.lastName=c.contactLastName)

| nombreEmpleado | ap1Empleado | nombreCliente | ap1Cliente |
|----------------|-------------|---------------|------------|
| Tom            | King        | Jean          | King       |
| Peter          | Marsh       | Peter         | Ferguson   |
| Diane          | Murphy      | Julie         | Murphy     |
| Julie          | Firrelli    | Julie         | Murphy     |
| Marv           | Patterson   | Marv          | Savelev    |

- (6) SELECT c.customerName, YEAR(p.paymentDate) year, SUM(p.amount) total FROM customers c JOIN payments p USING(customerNumber) GROUP BY c.customerName,year HAVING total>1000 (Mostrar los clientes con las suma de sus pagos agrupados por cliente y dáa cuya suma supere los 1000 euros. Es un ejercicio)

| customerName            | year | total    |
|-------------------------|------|----------|
| Alpha Cognac            | 2003 | 48051.04 |
| Alpha Cognac            | 2005 | 12432.32 |
| Amica Models & Co.      | 2004 | 82223.23 |
| Anna's Decorations, Ltd | 2003 | 80101.92 |
| Anna's Decorations Ltd  | 2005 | 56932.30 |

- (7) SELECT e.employeeNumber datosEmpleado, e.firstName, e.lastName, COUNT(c.customerNumber) numeroDeClientes FROM employees e JOIN customers c ON(e.employeeNumber=c.salesRepEmployeeNumber) GROUP BY e.employeeNumber ORDER BY numeroDeClientes DESC LIMIT 5 (Mostrar el top 5 de empleados que mas clientes tienen. Es un ejercicio)



| datosEmpleado | firstName | lastName | numeroDeClientes ▾ 1 |
|---------------|-----------|----------|----------------------|
| 1401          | Pamela    | Castillo | 10                   |
| 1504          | Barry     | Jones    | 9                    |
| 1501          | Larry     | Bott     | 8                    |
| 1323          | George    | Vanauf   | 8                    |
| 1286          | Foon Yue  | Tseng    | 7                    |

También se puede poner esto con dos criterios de order by:

**SELECT** e.employeeNumber datosEmpleado, e.firstName, e.lastName, **COUNT**(c.customerNumber) numeroDeClientes **FROM** employees e **JOIN** customers c **ON**(e.employeeNumber=c.salesRepEmployeeNumber) **GROUP BY** e.employeeNumber **ORDER BY** numeroDeClientes **DESC**, e.employeeNumber **ASC** **LIMIT 5** (Hay que tener cuidado con las comas para los dos criterios del ORDER BY)

| datosEmpleado | firstName | lastName | numeroDeClientes ▾ 1 |
|---------------|-----------|----------|----------------------|
| 1401          | Pamela    | Castillo | 10                   |
| 1504          | Barry     | Jones    | 9                    |
| 1323          | George    | Vanauf   | 8                    |
| 1501          | Larry     | Bott     | 8                    |
| 1286          | Foon Yue  | Tseng    | 7                    |

(8) **SELECT** c.customerNumber, c.customerName, **SUM**(p.amount) total **FROM** customers c **JOIN** payments p **USING**(customerNumber) **WHERE** p.paymentDate **BETWEEN** '2003-01-01' **AND** '2003-01-31' **GROUP BY** c.customerNumber **ORDER BY** total **DESC** **LIMIT 1** (Mostrar los datos del cliente que pagó más dinero en enero de 2003. Es un ejercicio)

| customerNumber | customerName         | total    |
|----------------|----------------------|----------|
| 128            | Blauer See Auto, Co. | 10549.01 |

También podemos hacerlo así:

**SELECT** a.customerNumber, a.customerName, **MAX**(a.total) **FROM** (**SELECT** c.customerNumber, c.customerName, **SUM**(p.amount) total **FROM** customers c **JOIN** payments p **USING**(customerNumber) **WHERE** p.paymentDate **BETWEEN** '2003-01-01' **AND** '2003-01-31' **GROUP BY** c.customerNumber) a

(9) **SELECT \* FROM customers c WHERE (SELECT COUNT(p.customerNumber) FROM payments p WHERE c.customerNumber=p.customerNumber) = (SELECT COUNT(o.customerNumber) FROM orders o WHERE c.customerNumber=o.customerNumber)** (Obtener aquellos clientes con un número de pagos igual al número de pedidos que han realizado. Es un ejercicio)

Opciones extra

|                                                                                                 | customerNumber | customerName         | contactLastName | contactFirstName | phone             | addressLine1              | addressLine2 | city          | state | postalCode | country | salesRepEmployeeNumber | creditLimit |
|-------------------------------------------------------------------------------------------------|----------------|----------------------|-----------------|------------------|-------------------|---------------------------|--------------|---------------|-------|------------|---------|------------------------|-------------|
| <input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar | 103            | Atelier graphique    | Schmitt         | Carine           | 40 32 2555        | 54, rue Royale            | NULL         | Nantes        | NULL  | 44000      | France  | 1370                   | 21000.00    |
| <input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar | 112            | Signal Gift Stores   | King            | Jean             | 7025551838        | 8489 Strong St.           | NULL         | Las Vegas     | NV    | 83030      | USA     | 1166                   | 71800.00    |
| <input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar | 121            | Baane Mini Imports   | Bergulfsen      | Jonas            | 07-98 9555        | Erling Skakkes gate 78    | NULL         | Stavern       | NULL  | 4110       | Norway  | 1504                   | 81700.00    |
| <input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar | 125            | Havel & Zbyszek Co   | Piestrzeniewicz | Zbyszek          | (26) 642-7555     | ul. Filtrawa 68           | NULL         | Warszawa      | NULL  | 01-012     | Poland  | NULL                   | 0.00        |
| <input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar | 128            | Blauer See Auto. Co. | Kettel          | Roland           | +49 69 66 90 2555 | Lyonerstr. 34             | NULL         | Frankfurt     | NULL  | 60528      | Germany | 1504                   | 59700.00    |
| <input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar | 129            | Mini Wheels Co.      | Murphy          | Julie            | 6505555787        | 5557 North Pendale Street | NULL         | San Francisco | CA    | 94217      | USA     | 1165                   | 64600.00    |
| <input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar | 148            | Savaleu & Henriot Co | Savaleu         | Maru             | 78 32 5555        | 2, rue du                 | NULL         | Lyon          | NULL  | 69004      | France  | 1337                   | 123000.00   |

También podemos hacer:

**SELECT customerNumber, customerName, a.numeroPagos, b.numeroPedidos FROM (SELECT p.customerNumber, COUNT(p.customerNumber) numeroPagos FROM payments p GROUP BY p.customerNumber) a JOIN (SELECT o.customerNumber, COUNT(o.customerNumber) numeroPedidos FROM orders o GROUP BY o.customerNumber) b USING(customerNumber) JOIN customers c USING(customerNumber) WHERE a.numeroPagos=b.numeroPedidos** (Al principio no ponemos nada de c. o p. en "customerNumber" o "customerName" porque no es necesario ya que da igual en qué base nos lo de, en las otras como "numeroPagos" o "numeroPedidos" si que lo necesitamos. Es un ejercicio)

(10) **SELECT AVG(debe.debe) FROM (SELECT MONTH(o.orderdate) mes, YEAR(o.orderdate) ano, SUM(od.priceEach\*od.quantityOrdered) debe FROM orders o JOIN orderdetails od USING(orderNumber) GROUP BY mes,ano HAVING mes=12) debe** (Obtener la media de la suma de dinero comprometida en pedidos en diciembre de cualquier año. Por suma comprometida se entiende lo que nos tienen que pagar por los pedidos realizados en el mes indicado. Es un ejercicio)

Dame las películas que tengan como mínimo cinco intérpretes y que uno de estos sea el intérprete con idInterprete=19

```
SELECT pelicula.titulo, pelicula.idpelicula, COUNT(peliculainterprete.idinterprete)
AS num_interpretes FROM pelicula INNER JOIN peliculainterprete
ON pelicula.idpelicula = peliculainterprete.idpelicula
WHERE pelicula.idpelicula
```

```
IN(SELECT idpelicula FROM peliculainterprete WHERE idinterprete = 19)
GROUP BY pelicula.idpelicula HAVING COUNT(peliculainterprete.idinterprete) >= 5;
```

| titulo              | idpelicula | num_interpretes |
|---------------------|------------|-----------------|
| ADAPTATION HOLES    | 3          | 5               |
| CHINATOWN GLADIATOR | 144        | 10              |
| DARES PLUTO         | 208        | 10              |
| DARN FORRESTER      | 212        | 9               |
| DAZED PUNK          | 217        | 6               |
| DYNAMITE TARZAN     | 266        | 5               |
| HOMICIDE PEACH      | 428        | 8               |
| JACKET FRISCO       | 473        | 10              |
| JUMANJI BLADE       | 490        | 6               |

Dame el top 5 de películas con más intérpretes y si tienen el mismo número de intérpretes, ordenarlas alfabéticamente

```
SELECT p.titulo, COUNT(pi.idinterprete) AS cantidad_interpretes FROM pelicula p
INNER JOIN peliculainterprete pi ON p.idpelicula = pi.idpelicula GROUP BY p.idpelicula
ORDER BY cantidad_interpretes DESC, p.titulo ASC LIMIT 5;
```

| titulo            | cantidad_interpretes ▾ 1 |
|-------------------|--------------------------|
| LAMBS CINCINATTI  | 15                       |
| BOONDOCK BALLROOM | 13                       |
| CHITTY LOCK       | 13                       |
| CRAZY HOME        | 13                       |
| DRACULA CRYSTAL   | 13                       |

Dime el intérprete que haya participado en más películas

```
SELECT i.nombre, i.apellido, COUNT(pi.idpelicula) AS cantidad_peliculas
FROM interprete i
INNER JOIN peliculainterprete pi ON i.idinterprete = pi.idinterprete
GROUP BY i.idinterprete
ORDER BY cantidad_peliculas DESC
LIMIT 1;
```

| nombre | apellido  | cantidad_peliculas |
|--------|-----------|--------------------|
| GINA   | DEGENERES | 42                 |

Dame el intérprete que más películas con rating 'R' haya hecho

```
SELECT interprete.nombre, interprete.apellido, COUNT(*) AS num_pelis_R
FROM pelicula
JOIN peliculainterprete ON pelicula.idpelicula = peliculainterprete.idpelicula
JOIN interprete ON peliculainterprete.idinterprete = interprete.idinterprete
WHERE pelicula.rating = 'R'
GROUP BY interprete.idinterprete
ORDER BY num_pelis_R DESC
LIMIT 1;
```

| idinterprete | nombre   | apellido | num_pelis_R |
|--------------|----------|----------|-------------|
| 123          | JULIANNE | DENCH    | 11          |

Dame el intérprete que más películas de Horror y de rating 'R' haya protagonizado

```
SELECT interprete.nombre, interprete.apellido, COUNT(*) as cantidad_peliculas
FROM pelicula JOIN peliculainterprete ON pelicula.idpelicula = peliculainterprete.idpelicula
JOIN interprete ON peliculainterprete.idinterprete = interprete.idinterprete
JOIN peliculacategoria ON pelicula.idpelicula = peliculacategoria.idpelicula
JOIN categoria ON peliculacategoria.idcategoria = categoria.idcategoria
WHERE categoria.idcategoria = 11 AND pelicula.rating = 'R'
GROUP BY interprete.idinterprete ORDER BY cantidad_peliculas DESC LIMIT 1;
```

| nombre | apellido | cantidad_peliculas |
|--------|----------|--------------------|
| JULIA  | MCQUEEN  | 4                  |

Dame todas las películas de Horror con rating 'R' que haya protagonizado JULIA MCQUEEN

```
SELECT pelicula.titulo FROM pelicula
INNER JOIN peliculainterprete ON pelicula.idpelicula = peliculainterprete.idpelicula
INNER JOIN interprete ON peliculainterprete.idinterprete = interprete.idinterprete
INNER JOIN peliculacategoria ON pelicula.idpelicula = peliculacategoria.idpelicula
INNER JOIN categoria ON peliculacategoria.idcategoria = categoria.idcategoria
WHERE interprete.nombre = 'JULIA' AND interprete.apellido = 'MCQUEEN'
AND categoria.idcategoria = 11 AND pelicula.rating = 'R';
```

| titulo             |
|--------------------|
| HIGH ENCINO        |
| SPIRIT FLINTSTONES |
| STRANGERS GRAFFITI |
| TRAIN BUNCH        |

Hacer que los registros de intérprete tengan los nombres y apellidos en minúsculas: (Empezamos con todo en mayus)

| <div>←T→</div>           |                                                                                          |                                                                                          |                                                                                          | idinterprete | nombre   | apellido     | fechamod            |
|--------------------------|------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|--------------|----------|--------------|---------------------|
| <input type="checkbox"/> |  Editar |  Copiar |  Borrar | 1            | PENELOPE | GUINNESS     | 2006-02-15 04:34:33 |
| <input type="checkbox"/> |  Editar |  Copiar |  Borrar | 2            | NICK     | WAHLBERG     | 2006-02-15 04:34:33 |
| <input type="checkbox"/> |  Editar |  Copiar |  Borrar | 3            | ED       | CHASE        | 2006-02-15 04:34:33 |
| <input type="checkbox"/> |  Editar |  Copiar |  Borrar | 4            | JENNIFER | DAVIS        | 2006-02-15 04:34:33 |
| <input type="checkbox"/> |  Editar |  Copiar |  Borrar | 5            | JOHNNY   | LOLLOBRIGIDA | 2006-02-15 04:34:33 |
| <input type="checkbox"/> |  Editar |  Copiar |  Borrar | 6            | BETTE    | NICHOLSON    | 2006-02-15 04:34:33 |
| <input type="checkbox"/> |  Editar |  Copiar |  Borrar | 7            | GRACE    | MOSTEL       | 2006-02-15 04:34:33 |

Ahora lo pasamos todo a minúsculas:

```
UPDATE interprete SET nombre = LOWER(nombre), apellido = LOWER(apellido);
```

| ←T→                      |                                                                                            |                                                                                            |                                                                                            | idinterprete | nombre   | apellido     | fechamod            |
|--------------------------|--------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|--------------|----------|--------------|---------------------|
| <input type="checkbox"/> |  Editar |  Copiar |  Borrar | 1            | penelope | guinness     | 2023-02-27 04:27:49 |
| <input type="checkbox"/> |  Editar |  Copiar |  Borrar | 2            | nick     | wahlberg     | 2023-02-27 04:27:49 |
| <input type="checkbox"/> |  Editar |  Copiar |  Borrar | 3            | ed       | chase        | 2023-02-27 04:27:49 |
| <input type="checkbox"/> |  Editar |  Copiar |  Borrar | 4            | jennifer | davis        | 2023-02-27 04:27:49 |
| <input type="checkbox"/> |  Editar |  Copiar |  Borrar | 5            | johnny   | lollobrigida | 2023-02-27 04:27:49 |
| <input type="checkbox"/> |  Editar |  Copiar |  Borrar | 6            | bette    | nicholson    | 2023-02-27 04:27:49 |

Dame la última película que hizo el actor con idinterprete=122 que tenga un caracteristicasespeciales='Trailer'

```
SELECT p.titulo FROM pelicula p
INNER JOIN peliculainterprete pi ON p.idpelicula = pi.idpelicula
INNER JOIN interprete i ON i.idinterprete = pi.idinterprete
WHERE i.idinterprete = 122 AND p.caracteristicasespeciales LIKE '%Trailer%'
ORDER BY p.anoestreno DESC LIMIT 1;
```

titulo  
ANTITRUST TOMATOES

Dame las películas en las que hayan estado los intérpretes 122 y 123 tanto juntos como separados y que tengan como característica especial Behind the Scenes

```
SELECT DISTINCT p.titulo FROM pelicula p
JOIN peliculainterprete pi ON p.idpelicula = pi.idpelicula
WHERE p.caracteristicasespeciales LIKE '%Behind the Scenes%' AND pi.idinterprete
IN (SELECT idinterprete FROM interprete WHERE idinterprete IN (122, 123))
ORDER BY p.titulo ASC;
```

| titulo             |
|--------------------|
| AMISTAD MIDSUMMER  |
| ATLANTIS CAUSE     |
| BIRDCAGE CASPER    |
| BLUES INSTINCT     |
| CHOCOLATE DUCK     |
| CLOCKWORK PARADISE |
| CLONES PINOCCHIO   |

Dime cuantas veces se alquila de media una película

```
SELECT AVG(total_alquileres) AS media_alquileres FROM (SELECT COUNT(*) AS total_alquileres
FROM alquiler JOIN ejemplar ON alquiler.idejemplar = ejemplar.idejemplar
GROUP BY ejemplar.idpelicula) subconsulta;
```

| media_alquileres |
|------------------|
| 16.7474          |

Dime cuales son las películas más alquiladas que superan la media

```
SELECT p.titulo, COUNT(*) AS num_alquileres FROM pelicula p
JOIN ejemplar e ON p.idpelicula = e.idpelicula
JOIN alquiler a ON e.idejemplar = a.idejemplar
WHERE a.fechadevolucion IS NOT NULL GROUP BY p.idpelicula HAVING COUNT(*) >
(SELECT AVG(num_alquileres) FROM
(SELECT COUNT(*) AS num_alquileres FROM pelicula p
JOIN ejemplar e ON p.idpelicula = e.idpelicula
JOIN alquiler a ON e.idejemplar = a.idejemplar
WHERE a.fechadevolucion IS NOT NULL GROUP BY p.idpelicula) subconsulta)
ORDER BY COUNT(*) DESC;
```

| titulo             | num_alquileres |
|--------------------|----------------|
| BUCKET BROTHERHOOD | 34             |
| ROCKETEER MOTHER   | 33             |
| FORWARD TEMPLE     | 32             |
| SCALAWAG DUCK      | 32             |
| GRIT CLOCKWORK     | 32             |
| TIMBERLAND SKY     | 31             |
| JUGGLER HARDLY     | 31             |
| ZORRO ARK          | 31             |
| ROBBERS JOON       | 31             |