

- Ejercicios de los JOIN

Todos los empleados y por cada uno todas las líneas de producto, tengan o no tengan ventas

Solución Juanma:

```
1 • SELECT CONCAT(employees.firstName, employees.lastName) AS empleado,
2     products.productLine,
3     SUM(orderdetails.priceEach*orderdetails.quantityOrdered) AS suma
4 FROM products
5 JOIN orderdetails ON orderdetails.productCode=products.productCode
6 RIGHT JOIN orders ON orderdetails.orderNumber=orders.orderNumber
7 RIGHT JOIN customers ON orders.customerNumber=customers.customerNumber
8 RIGHT JOIN employees ON customers.salesRepEmployeeNumber=employees.employeeNumber
9 GROUP BY empleado, products.productLine;
```

Solución Antonio:

```
• select concat(employees.firstName, ' ', employees.lastName) as nombre
      , products.productLine
      , sum(orderdetails.priceEach*orderdetails.quantityOrdered) as importe
from employees
left join customers
      on employees.employeeNumber = customers.salesRepEmployeeNumber
left join orders
      on customers.customerNumber = orders.customerNumber
left join orderdetails
      on orders.orderNumber = orderdetails.orderNumber
```

y finalizando con GROUP BY 1,2

Ejercicio:

- Query que nos diga cuántos clientes hay por país. (country y un contador de cliente)

```
1 | SELECT offices.country, COUNT(*)
2 | FROM offices
3 | LEFT JOIN employees
4 | ON offices.officeCode=employees.officeCode
5 | LEFT JOIN customers
6 | ON employees.employeeNumber=customers.salesRepEmployeeNumber
7 | GROUP BY 1;
```

OJO

Al hacer el segundo LEFT JOIN, estamos uniendo customers y empleados, y quedándonos con todos los empleados, PERO hay empleados que NO TIENEN CLIENTES, y nos cuenta esos empleados sin clientes como un registro extra. Por lo que para arreglar la query sería así: (eliminamos el segundo LEFT JOIN y dejamos un sólo JOIN)

```
1 SELECT offices.country, COUNT(*)
2 FROM offices
3 LEFT JOIN employees
4 ON offices.officeCode=employees.officeCode
5 JOIN customers
6 ON employees.employeeNumber=customers.salesRepEmployeeNumber
7 GROUP BY offices.country;
```

también podríamos modificar el COUNT y en lugar de asterisco decir COUNT(customers.customerName) también funcionaría, porque estaríamos contando únicamente los clientes.

SUBQUERIES

Se pueden insertar SELECT en el SELECT, en el FROM y en el WHERE.

Si la insertamos dentro del SELECT, la subquery tiene que devolver un campo (puesto que eso es lo que se usa en el SELECT)

Si la insertamos dentro del FROM, la subquery devuelve una tabla, como si creáramos una VIEW al vuelo, que además no se guarda en memoria.

A la hora de insertar la Sub Query en el WHERE aparecen nuevos operadores (ANY,SOME,IN,ALL,EXISTS). Mirar las transparencias y el tutorial de W3Schools.

SUBQUERIES CORRELACIONADAS

Son subqueries que relacionan con la misma tabla sobre la que se está haciendo la query principal. (Mirar transparencias)

-EJERCICIO: (con subqueries)

Query donde obtengamos: cliente y su deuda

Mi forma pero sin JOIN

```
1 SELECT tabla1.customerName, tabla1.Pagado, tabla2.CargoTotal, (tabla1.Pagado-tabla2.CargoTotal) AS "Deuda"
2 FROM (SELECT customers.customerName, SUM(payments.amount) AS "Pagado"
3       FROM customers, payments
4       WHERE customers.customerNumber=payments.customerNumber
5       GROUP BY customers.customerName) AS tabla1,
6       (SELECT customers.customerName, SUM(orderdetails.priceEach*orderdetails.quantityOrdered) AS "CargoTotal"
7       FROM customers, orders, orderdetails
8       WHERE orders.orderNumber=orderdetails.orderNumber
9       AND customers.customerNumber=orders.customerNumber
10      GROUP BY customers.customerName) AS tabla2
11 WHERE tabla1.customerName=tabla2.customerName
12 GROUP BY customerName;
```

```
SELECT tabla1.customerName, tabla1.Pagado, tabla2.CargoTotal,
(tabla1.Pagado-tabla2.CargoTotal) AS "Deuda"
FROM (SELECT customers.customerName, SUM(payments.amount) AS "Pagado"
      FROM customers, payments
      WHERE customers.customerNumber=payments.customerNumber
      GROUP BY customers.customerName) AS tabla1,
      (SELECT customers.customerName,
SUM(orderdetails.priceEach*orderdetails.quantityOrdered) AS "CargoTotal"
      FROM customers, orders, orderdetails
      WHERE orders.orderNumber=orderdetails.orderNumber
      AND customers.customerNumber=orders.customerNumber
      GROUP BY customers.customerName) AS tabla2
WHERE tabla1.customerName=tabla2.customerName
GROUP BY customerName;
```

Query de antes, usando los JOIN

```
1 SELECT customers.customerName, pagos.Pagado, compras.CargoTotal, (compras.CargoTotal-pagos.Pagado) AS "Deuda"
2 FROM (SELECT payments.customerNumber, SUM(payments.amount) AS "Pagado"
3       FROM payments
4       GROUP BY payments.customerNumber) AS pagos
5 RIGHT JOIN (SELECT orders.customerNumber, SUM(orderdetails.priceEach*orderdetails.quantityOrdered) AS "CargoTotal"
6            FROM orders
7            JOIN orderdetails
8            ON orders.orderNumber=orderdetails.orderNumber
9            GROUP BY 1) AS compras
10 ON pagos.customerNumber=compras.customerNumber
11 RIGHT JOIN customers
12 ON compras.customerNumber=customers.customerNumber;
```

OJO

Aquí hacemos la misma query pero no queremos que tengamos datos a NULL, puesto que cualquier operación con NULL (ejemplo: 20000 - NULL) da como resultado NULL y podría jodernos los resultados de la query. Para ello usamos IFNULL y COALESCE.

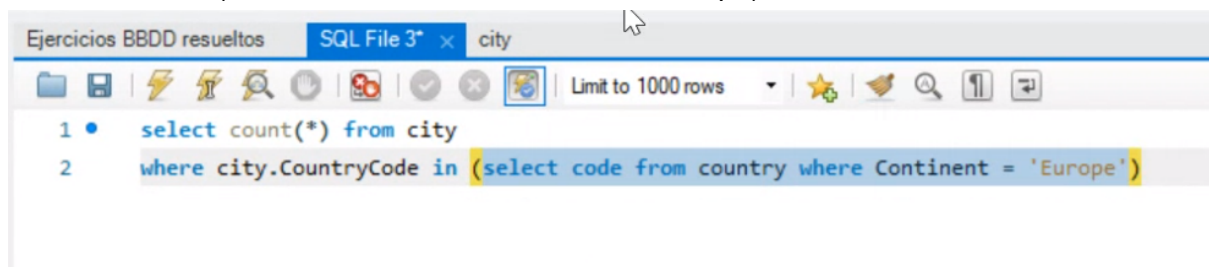
```
1 SELECT customers.customerName, IFNULL(pagos.Pagado,0) AS Pagos, IFNULL(compras.CargoTotal,0) AS Compras, COALESCE((compras.CargoTotal-pagos.Pagado),0) AS "Deuda"
2 FROM (SELECT payments.customerNumber, SUM(payments.amount) AS "Pagado"
3       FROM payments
4       GROUP BY payments.customerNumber) AS pagos
5 RIGHT JOIN (SELECT orders.customerNumber, SUM(orderdetails.priceEach*orderdetails.quantityOrdered) AS "CargoTotal"
6            FROM orders
7            JOIN orderdetails
8            ON orders.orderNumber=orderdetails.orderNumber
9            GROUP BY 1) AS compras
10 ON pagos.customerNumber=compras.customerNumber
11 RIGHT JOIN customers
12 ON compras.customerNumber=customers.customerNumber;
```

```
SELECT customers.customerName, IFNULL(pagos.Pagado,0) AS Pagos,
IFNULL(compras.CargoTotal,0) AS Compras,
COALESCE((compras.CargoTotal-pagos.Pagado),0) AS "Deuda"
FROM (SELECT payments.customerNumber, SUM(payments.amount) AS "Pagado"
      FROM payments
      GROUP BY payments.customerNumber) AS pagos
RIGHT JOIN (SELECT orders.customerNumber,
SUM(orderdetails.priceEach*orderdetails.quantityOrdered) AS "CargoTotal"
      FROM orders
      JOIN orderdetails
      ON orders.orderNumber=orderdetails.orderNumber
      GROUP BY 1) AS compras
ON pagos.customerNumber=compras.customerNumber
RIGHT JOIN customers
ON compras.customerNumber=customers.customerNumber;
```

-Ejercicio

Una query que nos diga las ciudades que hay en Europa (usando subselects, NO JOIN)

Versión Antonio (te da la cantidad de ciudades en Europa):



Versión 2 de Antonio (con JOIN):

```
select count(*)
from city
join country
on city.countryCode = country.Code
and country.Continent = 'Europe';
```

Versión mía:

```
1 SELECT city.Name, city.CountryCode,EuropeanCountries.Continent
2 FROM (SELECT *
3       FROM country
4       WHERE country.Continent="Europe") AS EuropeanCountries,
5       city
6 WHERE city.CountryCode=EuropeanCountries.Code
```

-Ejercicio

Una query de las ciudades de España donde la población sea mayor que la media (de las poblaciones de España)

```
1 SELECT *
2 FROM city
3 WHERE city.Population>(SELECT AVG(city.Population)
4                        FROM city
5                        WHERE city.CountryCode="ESP")
6 AND city.CountryCode="ESP";
```

-Ejercicio

Query de todas las ciudades del mundo donde la media de su población es superior a la media de la población de su país

OJO de ésta forma NO hace la query en condiciones, puesto que no estamos diferenciando el 'CountryCode' en el WHERE interno del city.CountryCode (la forma óptima es la de después)

[illegible]

Forma Óptima (obviar el EXPLAIN)

```
1 EXPLAIN SELECT *
2 FROM city principal
3 WHERE principal.Population > (SELECT AVG(secundaria.Population)
4                               FROM city secundaria
5                               WHERE principal.CountryCode = secundaria.CountryCode);
```

-Ejercicio

Query que nos de los clientes que NO tienen pedidos

Solución Juanma con NOT IN y DISTINCT:

```
1 SELECT customers.customerName
2 FROM customers
3 WHERE customers.customerNumber NOT IN (SELECT DISTINCT customerNumber FROM orders);
```

Solución Raúl (correlada) usando NOT EXISTS:

```
1 SELECT *
2 FROM customers
3 WHERE NOT EXISTS
4     (SELECT orders.customerNumber
5       FROM orders
6       WHERE orders.customerNumber = customers.customerNumber);
```

Solución Marta usando JOIN (por cierto, resulta que ésta es la forma más costosa de procesamiento de las tres):

```
1 SELECT *
2 FROM customers
3 LEFT JOIN orders
4 ON customers.customerNumber = orders.customerNumber
5 WHERE orders.customerNumber IS NULL;
```

-Ejercicio:

Query que nos devuelva países en los que sólo se hable una lengua.

Solucion Antonio 1:

```
select *  
  from country  
 join (select countrycode from countrylanguage  
       group by 1 having count(*) = 1) as cuenta  
 on country.code = cuenta.countrycode;
```

Solucion Antonio 2 (usando el IN y distinta localización del subselect):

```
select * from country  
where country.code in (select countrycode from countrylanguage  
                      group by 1 having count(*) = 1);
```

Solucion Antonio 3, usando JOIN:

```
select country.Name  
from country join countrylanguage  
on country.Code = countrylanguage.CountryCode  
group by 1  
having count(*) = 1;
```

-Ejercicio:

Query que nos devuelva el producto de mayor precio de cada línea de producto.

```
1 | SELECT principal.productName, principal.productLine, principal.buyPrice  
2 | FROM products principal  
3 | WHERE principal.buyPrice=(SELECT MAX(secundaria.buyPrice)  
4 |                           FROM products secundaria  
5 |                           WHERE principal.productLine=secundaria.productLine);
```