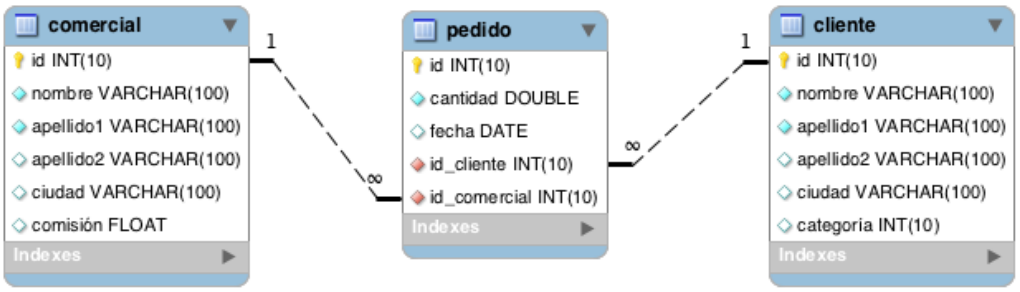
[Ejercicios. Realización de consultas SQL (josejuansanchez.org)]



1.3.3 Consultas sobre una tabla

1. Devuelve un listado con todos los pedidos que se han realizado. Los pedidos deben estar ordenados por la fecha de realización, mostrando en primer lugar los pedidos más recientes.
    a. **SELECT * FROM pedido**
    b. **ORDER BY fecha DESC;**
2. Devuelve todos los datos de los dos pedidos de mayor valor.
    a. **SELECT * FROM pedido**
    b. **ORDER BY total DESC**
    c. **LIMIT 0,2**
3. Devuelve un listado con los identificadores de los clientes que han realizado algún pedido. Tenga en cuenta que no debe mostrar identificadores que estén repetidos.
    a. **SELECT DISTINCT id_cliente FROM pedido**
4. Devuelve un listado de todos los pedidos que se realizaron durante el año 2017, cuya cantidad total sea superior a 500€.
    a. **SELECT * FROM pedido**
    b. **WHERE EXTRACT(YEAR from fecha)=2017 AND**
    c. **total>500**
5. Devuelve un listado con el nombre y los apellidos de los comerciales que tienen una comisión entre 0.05 y 0.11.
    a. **SELECT nombre, apellido1, apellido2 FROM comercial**
    b. **WHERE comisión>0.05**
    c. **AND comisión<0.11**
6. Devuelve el valor de la comisión de mayor valor que existe en la tabla comercial.
    a. **SELECT * FROM comercial**
    b. **ORDER BY comlsión DESC**
    c. **LIMIT 0,1**
7. Devuelve el identificador, nombre y primer apellido de aquellos clientes cuyo segundo apellido no es NULL. El listado deberá estar ordenado alfabéticamente por apellidos y nombre.
    a. **SELECT id, nombre, apellido1, apellido2 FROM cliente**
    b. **WHERE apellido2 IS NOT NULL**
    c. **ORDER BY nombre, apellido1, apellido2**
8. Devuelve un listado de los nombres de los clientes que empiezan por A y terminan por n y también los nombres que empiezan por P. El listado deberá estar ordenado alfabéticamente.
    a. **(SELECT nombre**
    b. **FROM cliente**
    c. **WHERE nombre LIKE 'A%' '%n%')**
    d. **UNION**
    e. **(SELECT nombre**
    f. **FROM cliente**
    g. **WHERE NOMBRE like '%p%')**
    h. **ORDER BY nombre ASC;**
9. Devuelve un listado de los nombres de los clientes que no empiezan por A. El listado deberá estar ordenado alfabéticamente.
    a. **SELECT nombre**
    b. **FROM cliente**
    c. **WHERE nombre NOT LIKE 'A%'**
    d. **ORDER BY nombre ASC;**

10. Devuelve un listado con los nombres de los comerciales que terminan por el o o. Tenga en cuenta que se deberán eliminar los nombres repetidos.
    a. **SELECT DISTINCT nombre FROM comercial**
    b. **WHERE nombre NOT LIKE '%o'**

1.3.4 Consultas multitabla (Composición interna)

Resuelva todas las consultas utilizando la sintaxis de SQL1 y SQL2.

1. Devuelve un listado con el identificador, nombre y los apellidos de todos los clientes que han realizado algún pedido. El listado debe estar ordenado alfabéticamente y se deben eliminar los elementos repetidos.
    a. **SELECT DISTINCT id_cliente,nombre, apellido1, apellido2**
    b. **FROM cliente**
    c. **JOIN pedido**
    d. **ON cliente.id=pedido.id_cliente**
    e. **ORDER BY nombre ASC;**
2. Devuelve un listado que muestre todos los pedidos que ha realizado cada cliente. El resultado debe mostrar todos los datos de los pedidos y del cliente. El listado debe mostrar los datos de los clientes ordenados alfabéticamente.
    a. **SELECT DISTINCT ***
    b. **FROM cliente**
    c. **JOIN pedido**
    d. **ON cliente.id=pedido.id_cliente**
    e. **ORDER BY nombre ASC;**
3. Devuelve un listado que muestre todos los pedidos en los que ha participado un comercial. El resultado debe mostrar todos los datos de los pedidos y de los comerciales. El listado debe mostrar los datos de los comerciales ordenados alfabéticamente.
    a. **SELECT DISTINCT ***
    b. **FROM comercial**
    c. **JOIN pedido**
    d. **ON comercial.id=pedido.id_comercial**
    e. **ORDER BY nombre ASC;**
4. Devuelve un listado que muestre todos los clientes, con todos los pedidos que han realizado y con los datos de los comerciales asociados a cada pedido.
    a. **SELECT ***
    b. **FROM comercial INNER JOIN pedido ON comercial.id=pedido.id_comercial**
    c. **INNER JOIN cliente ON cliente.id=pedido.id_cliente**
    d. **GROUP BY pedido.id**
5. Devuelve un listado de todos los clientes que realizaron un pedido durante el año 2017, cuya cantidad esté entre 300 € y 1000 €.
    a. **SELECT * FROM cliente INNER JOIN pedido ON cliente.id=pedido.id_cliente**
    b. **WHERE EXTRACT(year from fecha)=2017 AND**
    c. **total>300 AND total<1000**
6. Devuelve el nombre y los apellidos de todos los comerciales que han participado en algún pedido realizado por María Santana Moreno.
    a. **SELECT comercial.nombre, comercial.apellido1, comercial.apellido2 FROM comercial**
    b. **INNER JOIN pedido ON comercial.id=pedido.id_comercial**
    c. **INNER JOIN cliente ON cliente.id=pedido.id_cliente**
    d. **WHERE cliente.nombre='María' AND cliente.apellido1='Santana' AND cliente.apellido2='Moreno**
7. Devuelve el nombre de todos los clientes que han realizado algún pedido con el comercial Daniel Sáez Vega.
    a. **SELECT cliente.nombre FROM comercial**
    b. **INNER JOIN pedido ON comercial.id=pedido.id_comercial**
    c. **INNER JOIN cliente ON cliente.id=pedido.id_cliente**
    d. **WHERE comercial.nombre='Daniel' AND comercial.apellido1='Sáez' AND comercial. apellido2='Vega'**

1.3.5 Consultas multitabla (Composición externa)

Resuelva todas las consultas utilizando las cláusulas LEFT JOIN y RIGHT JOIN.

1. Devuelve un listado con todos los clientes junto con los datos de los pedidos que han realizado. Este listado también debe incluir los clientes que no han realizado ningún pedido. El listado debe estar ordenado alfabéticamente por el primer apellido, segundo apellido y nombre de los clientes.
    a. **SELECT * FROM cliente**
    b. **LEFT JOIN pedido**
    c. **ON cliente.id=pedido.id_cliente**
    d. **ORDER BY cliente.apellido1,cliente.apellido2,cliente.nombre**
2. Devuelve un listado con todos los comerciales junto con los datos de los pedidos que han realizado. Este listado también debe incluir los comerciales que no han realizado ningún pedido. El listado debe estar ordenado alfabéticamente por el primer apellido, segundo apellido y nombre de los comerciales.
    a. **SELECT * FROM comercial**
    b. **LEFT JOIN pedido**
    c. **ON comercial.id=pedido.id_comercial**
    d. **ORDER BY comercial.apellido1,comercial.apellido2,comercial.nombre**
3. Devuelve un listado que solamente muestre los clientes que no han realizado ningún pedido.
    a. **SELECT * FROM cliente LEFT JOIN pedido**
    b. **ON cliente.id=pedido.id_cliente**
    c. **WHERE pedido.id IS NULL**
4. Devuelve un listado que solamente muestre los comerciales que no han realizado ningún pedido.
    a. **SELECT * FROM comercial LEFT JOIN pedido**
    b. **ON comercial.id=pedido.id_comercial**
    c. **WHERE pedido.id IS NULL**
5. Devuelve un listado con los clientes que no han realizado ningún pedido y de los comerciales que no han participado en ningún pedido. Ordene el listado alfabéticamente por los apellidos y el nombre. En el listado deberá diferenciar de algún modo los clientes y los comerciales.
    a. **(SELECT cliente.nombre, cliente.apellido1, cliente.apellido2, 'cliente' AS 'stakeholder' FROM cliente LEFT JOIN pedido**
    b. **ON cliente.id=pedido.id_cliente**
    c. **WHERE pedido.id IS NULL**
    d. **ORDER BY cliente.apellido2, cliente.apellido1, cliente.nombre)**
    e. **UNION**
    f. **(SELECT comercial.nombre, comercial.apellido1, comercial.apellido2, 'comercial' FROM comercial LEFT JOIN pedido**
    g. **ON comercial.id=pedido.id_comercial**
    h. **WHERE pedido.id IS NULL**
    i. **ORDER BY comercial.apellido2, comercial.apellido1, comercial.nombre)**
6. ¿Se podrían realizar las consultas anteriores con NATURAL LEFT JOIN o NATURAL RIGHT JOIN? Justifique su respuesta.
    a. **No, porque NATURAL JOIN utiliza los nombres de las columnas para hacer las relaciones. Si los nombres son iguales entonces utilizará esas columnas para hacer las relaciones, lo cual derivaría en error en el caso de las tablas de este ejercicio, que tienen algunas columnas, en cada tabla, con nombres iguales, pero con valores diferentes. En concreto, para este ejercicio, debo hacer relaciones utilizando nombres de columna diferentes, ya que estas contienen los valores a los que quiero hacer referencia, que deben ser iguales tanto en una columna, como en otra. Ejemplo: la tabla "cliente" tiene una columna llamada "id", y para hacer referencia a esta desde la tabla "pedido" no puedo utilizar el nombre "id", sino un nombre específico dentro de "pedido" que hace referencia a la columna "id" de la tabla "cliente", esta columna en la tabla "pedido" es "id_cliente".**

1.3.6 Consultas resumen

1. Calcula la cantidad total que suman todos los pedidos que aparecen en la tabla pedido.
    a. **SELECT SUM(total) FROM pedido**
2. Calcula la cantidad media de todos los pedidos que aparecen en la tabla pedido.
    a. **SELECT AVG(total) FROM pedido**
3. Calcula el número total de comerciales distintos que aparecen en la tabla pedido.
    a. **SELECT DISTINCT COUNT(nombre AND apellido1 AND apellido2) FROM comercial**
4. Calcula el número total de clientes que aparecen en la tabla cliente.
    a. **SELECT DISTINCT COUNT(nombre AND apellido1 AND apellido2) FROM cliente**
5. Calcula cuál es la mayor cantidad que aparece en la tabla pedido.
    a. **SELECT MAX(total) AS CantidadMAYOR FROM pedido**

6. Calcula cuál es la menor cantidad que aparece en la tabla pedido.
    a. **SELECT MIN(total) AS CantidadMENOR FROM pedido**
7. Calcula cuál es el valor máximo de categoría para cada una de las ciudades que aparece en la tabla cliente.
    a. **select ciudad, max(categoría) from cliente group by ciudad**
8. Calcula cuál es el máximo valor de los pedidos realizados durante el mismo día para cada uno de los clientes. Es decir, el mismo cliente puede haber realizado varios pedidos de diferentes cantidades el mismo día. Se pide que se calcule cuál es el pedido de máximo valor para cada uno de los días en los que un cliente ha realizado un pedido. Muestra el identificador del cliente, nombre, apellidos, la fecha y el valor de la cantidad.
    a. **select cliente.id, nombre, apellido1, apellido2, fecha, total, max(total) from cliente inner join pedido on cliente.id=pedido.id_cliente**
    b. **group by fecha**
9. Calcula cuál es el máximo valor de los pedidos realizados durante el mismo día para cada uno de los clientes, teniendo en cuenta que sólo queremos mostrar aquellos pedidos que superen la cantidad de 2000 €.
    a. **select cliente.id, nombre, apellido1, apellido2, fecha, total, max(total) from cliente inner join pedido on cliente.id=pedido.id_cliente**
    b. **where total > 2000**
    c. **group by fecha**
10. Calcula el máximo valor de los pedidos realizados para cada uno de los comerciales durante la fecha 2016-08-17. Muestra el identificador del comercial, nombre, apellidos y total.
    a. **select comercial.id, comercial.nombre, comercial.apellido1, comercial.apellido2, max(total) from comercial inner join pedido on comercial.id=pedido.id_comercial**
    b. **where fecha='20160817'**
    c. **group by nombre**
11. Devuelve un listado con el identificador de cliente, nombre y apellidos y el número total de pedidos que ha realizado cada uno de los clientes. Tenga en cuenta que pueden existir clientes que no han realizado ningún pedido. Estos clientes también deben aparecer en el listado indicando que el número de pedidos realizados es 0.
    a. **select cliente.id, nombre, apellido1, apellido2,**
    b. **count(pedido.id_cliente)**
    c. **as 'pedidos realizados'**
    d. **from cliente left join pedido on cliente.id=pedido.id_cliente**
    e. **group by nombre**
12. Devuelve un listado con el identificador de cliente, nombre y apellidos y el número total de pedidos que ha realizado cada uno de clientes durante el año 2017.
    a. **select cliente.id, nombre, apellido1, apellido2,**
    b. **count(pedido.id_cliente)**
    c. **as 'pedidos realizados'**
    d. **from cliente left join pedido on cliente.id=pedido.id_cliente**
    e. **where extract(year from fecha)='2017'**
    f. **group by nombre**
13. Devuelve un listado que muestre el identificador de cliente, nombre, primer apellido y el valor de la máxima cantidad del pedido realizado por cada uno de los clientes. El resultado debe mostrar aquellos clientes que no han realizado ningún pedido indicando que la máxima cantidad de sus pedidos realizados es 0. Puede hacer uso de la función IFNULL.
    a. **select cliente.id, nombre, apellido1, apellido2, IFNULL(max(total), 0) from cliente left join pedido**
    b. **on cliente.id=pedido.id_cliente**
    c. **group by nombre**
14. Devuelve cuál ha sido el pedido de máximo valor que se ha realizado cada año.
    a. **select id as id_pedido, id_cliente, id_comercial, extract(year from fecha) as año, max(total) as Pedido_Máximo_Valor from pedido**
    b. **group by extract(year from fecha)**
    c. **order by fecha**
15. Devuelve el número total de pedidos que se han realizado cada año.
    a. **select id as id_pedido, id_cliente, id_comercial, extract(year from fecha) as año, count(pedido.id) as Total_Pedidos from pedido**
    b. **group by extract(year from fecha)**
    c. **order by fecha**

1.3.7 Subconsultas

1.3.7.1 Con operadores básicos de comparación

1. Devuelve un listado con todos los pedidos que ha realizado Adela Salas Díaz. (Sin utilizar INNER JOIN).
    a. **select ' ' as 'id_pedido', ' ' as 'total', ' ' as 'fecha', ' ' as 'id_comercial' from cliente**
    b. **where nombre='Adela'**
    c. **union**
    d. **select id, total, fecha, id_comercial from pedido**
    e. **where id_cliente='2'**
2. Devuelve el número de pedidos en los que ha participado el comercial Daniel Sáez Vega. (Sin utilizar INNER JOIN)
    a. **select ' ' as 'Número_Pedidos' from comercial**
    b. **where nombre='Daniel' AND apellido1='Sáez'**
    c. **union**
    d. **select count(id_comercial) from pedido**
    e. **where id_comercial='1'**
3. Devuelve los datos del cliente que realizó el pedido más caro en el año 2019. (Sin utilizar INNER JOIN)
    a. **select '' as 'nombre', '' as ' apellido1', '' as 'apellido2', total as Pedido_Más_Caro_2019 from pedido**
    b. **where total=(select max(total) from pedido where extract(year from fecha)=2019)**
    c. **union**
    d. **select nombre, apellido1, apellido2, '' from cliente**
    e. **where id=1**
4. Devuelve la fecha y la cantidad del pedido de menor valor realizado por el cliente Pepe Ruiz Santana.
    a. **select '', nombre, apellido1 from cliente**
    b. **where nombre='Pepe'**
    c. **union**
    d. **select '', fecha, total from pedido**
    e. **where total=(select min(total) from pedido)**
5. Devuelve un listado con los datos de los clientes y los pedidos, de todos los clientes que han realizado un pedido durante el año 2017 con un valor mayor o igual al valor medio de los pedidos realizados durante ese mismo año.
    a. **select pedido.id, pedido.total, pedido.fecha, pedido.id_cliente, pedido.id_comercial, nombre, apellido1, apellido2 from pedido INNER JOIN cliente on**
    b. **pedido.id_cliente=cliente.id**
    c. **where pedido.total>=(select avg(total) from pedido**
    d. **where extract(year from fecha)='2017') and extract(year from pedido.fecha)='2017'**

1.3.7.2 Subconsultas con ALL y ANY

6. Devuelve el pedido más caro que existe en la tabla pedido sin hacer uso de MAX, ORDER BY ni LIMIT.
    a. **select MIN(-1 * total)* -1 as Pedido_Más_Caro from pedido**
7. Devuelve un listado de los clientes que no han realizado ningún pedido. (Utilizando ANY o ALL).
    a. **select * from cliente**
    b. **where id=any(select cliente.id from cliente left join pedido on cliente.id=pedido.id_cliente where pedido.id IS NULL)**
8. Devuelve un listado de los comerciales que no han realizado ningún pedido. (Utilizando ANY o ALL).
    a. **select * from comercial**
    b. **where id=any(select comercial.id from comercial left join pedido on comercial.id=pedido.id_comercial where pedido.id IS NULL)**

1.3.7.3 Subconsultas con IN y NOT IN

9. Devuelve un listado de los clientes que no han realizado ningún pedido. (Utilizando IN o NOT IN).
    a. **select * from cliente**
    b. **where id IN (select cliente.id from cliente left join pedido on cliente.id=pedido.id_cliente where pedido.id IS NULL)**

10. Devuelve un listado de los comerciales que no han realizado ningún pedido. (Utilizando IN o NOT IN).
    a. **select * from comercial**
    b. **where id NOT IN (select comercial.id from comercial left join pedido on comercial.id=pedido.id_comercial where pedido.id IS NOT NULL)**

1.3.7.4 Subconsultas con EXISTS y NOT EXISTS

11. Devuelve un listado de los clientes que no han realizado ningún pedido. (Utilizando EXISTS o NOT EXISTS).
    a. **select * from cliente**
    b. **where not exists (select id_cliente from pedido where id_cliente=cliente.id)**
12. Devuelve un listado de los comerciales que no han realizado ningún pedido. (Utilizando EXISTS o NOT EXISTS).
    a. **select * from comercial**
    b. **where not exists (select id_comercial from pedido where id_comercial=comercial.id)**

1.3.8 Tarea campus.dicampus

1. Además de los ejercicios expuestos, realizar, una nueva tabla que esté relacionada con alguna de los de los ejercicios y realizar 2 consultas que hagan uso de un join, con esa nueva y otras dos tablas más.

    a. **CREATE TABLE contacto (**
    b. **id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,**
    c. **correo VARCHAR(100) NOT NULL,**
    d. **teléfono INT,**
    e. **id_cliente_c INT UNSIGNED NOT NULL,**
    f. **FOREIGN KEY (id_cliente_c) REFERENCES cliente(id)**
    g. **);**

    I. **INSERT INTO contacto VALUES(1, 'Aarón@gmail.com', '608472265', 1);**
    II. **INSERT INTO contacto VALUES(2, 'Adela@gmail.com', '608472261', 2);**
    III. **INSERT INTO contacto VALUES(3, 'Adolfo@gmail.com', '608472262', 3);**
    IV. **INSERT INTO contacto VALUES(4, 'Adrián@gmail.com', '608472263', 4);**
    V. **INSERT INTO contacto VALUES(5, 'Marcos@gmail.com', '608472264', 5);**
    VI. **INSERT INTO contacto VALUES(6, 'María@gmail.com', '608472266', 6);**
    VII. **INSERT INTO contacto VALUES(7, 'Pilar@gmail.com', '608472267', 7);**
    VIII. **INSERT INTO contacto VALUES(8, 'Pepe@gmail.com', '608472268', 8);**
    IX. **INSERT INTO contacto VALUES(9, 'Guillermo@gmail.com', '608472269', 9);**
    X. **INSERT INTO contact VALUES(10, 'Daniel@gmail.com', '608472260', 10);**

    I. **Consulta I:**
        A. Muestra todos los datos de contacto de los clientes que han hecho una transacción con el comercial Daniel Sáez Vega
            1. **SELECT contacto.id, contacto.correo, contacto.teléfono, contacto.id_clientec FROM contacto**
            2. **INNER JOIN pedido ON contacto.id_clientec=pedido.id_cliente**
            3. **INNER JOIN comercial ON comercial.id=pedido.id_comercial**
            4. **WHERE comercial.nombre='Daniel' AND comercial.apellido1='Sáez' AND comercial.apellido2='Vega'**
    II. **Consulta II:**
        A. Muestra todos los datos de contacto de los clientes cuya ciudad es Sevilla y cuyo comercial con el que han realizado la transacción es Antonio Vega Sánchez
            1. **SELECT contacto.id, contacto.correo, contacto.teléfono, contacto.id_clientec FROM contacto**
            2. **INNER JOIN pedido ON contacto.id_clientec=pedido.id_cliente**
            3. **INNER JOIN comercial ON comercial.id=pedido.id_comercial**
            4. **INNER JOIN cliente ON cliente.id=pedido.id_cliente**
            5. **WHERE cliente.ciudad='Sevilla' AND comercial.nombre='Antonio' AND comercial.apellido1='Vega' AND comercial.apellido2='Hernández'**