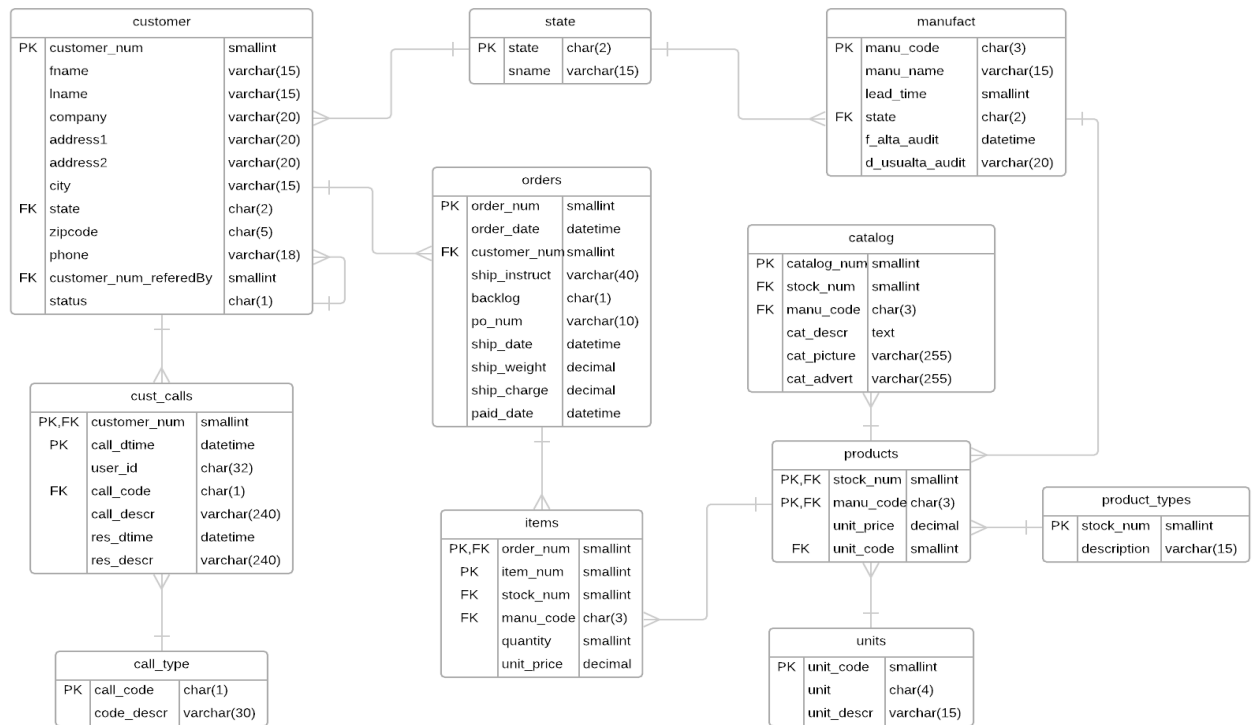


Práctica de SQL



JOINS (Group by, having, Subqueries, subq. correlacionados, outer joins, Temp tables)

1. Mostrar el Código del fabricante, nombre del fabricante, tiempo de entrega y monto Total de productos vendidos, ordenado por nombre de fabricante. En caso que el fabricante no tenga ventas, mostrar el total en NULO.

```

SELECT m.manu_code, m.manu_name, lead_time, SUM(unit_price * quantity) Total
FROM manufact m left join items i ON (i.manu_code=m.manu_code)
GROUP BY m.manu_code,m.manu_name, lead_time
order by m.manu_name
    
```

2. Mostrar una lista de a pares, de todos los fabricantes que fabriquen el mismo producto. En el caso que haya un único fabricante deberá mostrar el Código de fabricante 2 en nulo.

El listado tiene que tener el siguiente formato:

Nro. de Producto (stock_num)	Descripcion (Description)	Cód. de fabric. 1 (manu_code)	Cód. de fabric. 2 (manu_code)
---------------------------------	------------------------------	----------------------------------	----------------------------------

```

SELECT s1.stock_num,tp.description,s1.manu_code, s2.manu_code
FROM products s1 LEFT JOIN products s2
ON (s1.stock_num=s2.stock_num AND s1.manu_code != s2.manu_code)
JOIN product_Types tp ON (s1.stock_num = tp.stock_num)
    
```

```

SELECT s1.stock_num, tp.description, s1.manu_code, s2.manu_code
FROM products s1 LEFT JOIN products s2
      ON (s1.stock_num=s2.stock_num
          AND s1.manu_code != s2.manu_code)
      JOIN product_Types tp ON (s1.stock_num = tp.stock_num)
where s1.manu_code < s2.manu_code or s2.manu_code is NULL

```

3. Listar todos los clientes que hayan tenido más de una orden.
 - a) En primer lugar, escribir una consulta usando una subconsulta.
 - b) Reescribir la consulta usando dos sentencias SELECT y una tabla temporal.
 - c) Reescribir la consulta utilizando GROUP BY y HAVING.

La consulta deberá tener el siguiente formato:

Número_de_Cliente	Nombre	Apellido
(customer_num)	(fname)	(lname)

a.1

```

SELECT customer_num, fname, lname
FROM customer
WHERE customer_num IN (SELECT customer_num FROM orders
                      GROUP BY customer_num HAVING COUNT(order_num)>1)

```

a2

```

SELECT customer_num, fname, lname
FROM customer c
WHERE EXISTS (SELECT customer_num FROM orders o
              WHERE o.customer_num = c.customer_num
              GROUP BY customer_num HAVING COUNT(order_num)>1)

```

a3

```

SELECT customer_num, fname, lname
FROM customer c
WHERE (SELECT COUNT(order_num) FROM orders o
      WHERE o.customer_num = c.customer_num)>1

```

a4

```

SELECT c.customer_num, fname, lname
FROM customer c JOIN (SELECT customer_num FROM orders
                    GROUP BY customer_num HAVING COUNT(order_num)>1) sub1
ON c.customer_num = sub1.customer_num

```

b

```

SELECT customer_num
INTO #clieTemp
FROM orders
GROUP BY customer_num HAVING COUNT(order_num)>1

SELECT c.customer_num, fname, lname
FROM customer c JOIN #clieTemp c2

```

```
ON c.customer_num = c2.customer_num
```

C

```
SELECT c.customer_num, fname, lname
FROM customer c JOIN orders o ON (c.customer_num=o.customer_num)
GROUP BY c.customer_num, fname, lname
HAVING COUNT(order_num)>1
```

4. Seleccionar todas las Órdenes de compra cuyo Monto total (Suma de p x q de sus items) sea menor al precio total promedio (avg p x q) de todos los ítems de todas las ordenes.

Formato de la salida:

Nro. de Orden	Total
(order_num)	(suma)

```
SELECT o.order_num 'Nro. de Orden', SUM(unit_price*quantity) Total
FROM orders o JOIN items I ON (o.order_num = i.order_num)
GROUP BY o.order_num
HAVING SUM(unit_price*quantity) < (SELECT AVG(unit_price*quantity) FROM items)
```

5. Obtener por cada fabricante, el listado de todos los productos de stock con precio unitario (unit_price) mayor que el precio unitario promedio para dicho fabricante.

Los campos de salida serán: manu_code, manu_name, stock_num, description, unit_price.

Por ejemplo:

El precio unitario promedio de los productos fabricados por ANZ es \$180.23. se debe incluir en su lista todos los productos de ANZ que tengan un precio unitario superior a dicho importe.

```
SELECT s.manu_code, manu_name, s.stock_num, description, unit_price
FROM products s JOIN manufact m ON (s.manu_code=m.manu_code) JOIN
product_Types tp ON (tp.stock_num = s.stock_num)
WHERE unit_price > (SELECT AVG(unit_price) FROM products s2 WHERE
s2.manu_code = m.manu_code)
```

6. Usando el operador NOT EXISTS listar la información de órdenes de compra que NO incluyan ningún producto que contenga en su descripción el string ' baseball gloves'. Ordenar el resultado por compañía del cliente ascendente y número de orden descendente.

El formato de salida deberá ser:

Número de Cliente	Compañía	Número de Orden	Fecha de la Orden
(customer_num)	(company)	(order_num)	(order_date)

```
SELECT c.customer_num, company, o.order_num, order_date
FROM orders o JOIN customer c ON (o.customer_num=c.customer_num)
WHERE NOT EXISTS (SELECT item_num FROM items i JOIN product_Types tp
ON (i.stock_num=tp.stock_num)
WHERE description LIKE '%baseball gloves%'
AND i.order_num = o.order_num)
ORDER BY company, order_num desc
```

Operador UNION

7. Reescribir la siguiente consulta utilizando el operador UNION:

```
SELECT * FROM products
WHERE manu_code = 'HRO' OR stock_num = 1
```

```
SELECT * FROM products WHERE manu_code='HRO'
UNION
SELECT * FROM products WHERE stock_num=1
```

8. Desarrollar una consulta que devuelva las ciudades y compañías de todos los Clientes ordenadas alfabéticamente por Ciudad pero en la consulta deberán aparecer primero las compañías situadas en Redwood City y luego las demás.

Formato:

Clave de ordenamiento (sortkey)	Ciudad (city)	Compañía (company)
------------------------------------	------------------	-----------------------

```
SELECT 1 sortkey, city ciudad, Company 'Compañía '
FROM customer
WHERE city ='Redwood City'
UNION
SELECT 2 sortkey, city ciudad, Company 'Compañía '
FROM customer
WHERE city !='Redwood City'
ORDER BY sortkey, city
```

9. Desarrollar una consulta que devuelva los dos tipos de productos más vendidos y los dos menos vendidos en función de las unidades totales vendidas.

Formato

<i>Tipo Producto</i>	<i>Cantidad</i>
101	999
189	888
24	...
4	1

```
SELECT i.stock_num, sum(i.quantity) Total
FROM items i
where i.stock_num in (SELECT top 2 i2.stock_num
FROM items i2
group by i2.stock_num
order by sum(i2.quantity) desc)
group by i.stock_num
UNION
SELECT i.stock_num, sum(i.quantity) Total
FROM items i
where i.stock_num in (SELECT top 2 i2.stock_num
FROM items i2
group by i2.stock_num
order by sum(i2.quantity) ASC)
group by i.stock_num
```

order by 2 desc

VISTAS

10. Crear una Vista llamada ClientesConMultiplesOrdenes basada en la consulta realizada en el punto 3.c con los nombres de atributos solicitados en dicho punto.

```
CREATE VIEW ClientesConMultiplesOrdenes
(numero_cliente, nombre, apellido)
AS
SELECT c.customer_num, fname, lname
FROM customer c JOIN orders o
ON (c.customer_num=o.customer_num)
GROUP BY c.customer_num, lname, fname
HAVING COUNT(order_num)>1
```

11. Crear una Vista llamada Productos_HRO en base a la consulta

```
SELECT * FROM products
WHERE manu_code = "HRO"
```

La vista deberá restringir la posibilidad de insertar datos que no cumplan con su criterio de selección.

- a. Realizar un INSERT de un Producto con manu_code='ANZ' y stock_num=303. Qué sucede?
- b. Realizar un INSERT con manu_code='HRO' y stock_num=303. Qué sucede?
- c. Validar los datos insertados a través de la vista.

```
CREATE VIEW Productos_HRO
AS SELECT * FROM products
WHERE manu_code = 'HRO'
WITH CHECK OPTION ;

INSERT INTO Productos_HRO (stock_num,manu_code)
VALUES (303,'ANZ') ;

INSERT INTO Productos_HRO (stock_num,manu_code)
VALUES (303,'HRO') ;
```

TRANSACCIONES

12. Escriba una transacción que incluya las siguientes acciones:
 - BEGIN TRANSACTION
 - Insertar un nuevo cliente llamado "Fred Flintstone" en la tabla de clientes (customer).

- Seleccionar todos los clientes llamados Fred de la tabla de clientes (customer).
- ROLLBACK TRANSACTION

Luego volver a ejecutar la consulta

- Seleccionar todos los clientes llamados Fred de la tabla de clientes (customer).
 - Completado el ejercicio descripto arriba. Observar que los resultados del segundo SELECT difieren con respecto al primero.
13. Se ha decidido crear un nuevo fabricante AZZ, quién proveerá parte de los mismos productos que provee el fabricante ANZ, los productos serán los que contengan el string 'tennis' en su descripción.
- Agregar las nuevas filas en la tabla manufact y la tabla products.
 - El código del nuevo fabricante será "AZZ", el nombre de la compañía "AZZIO SA" y el tiempo de envío será de 5 días (lead_time).
 - La información del nuevo fabricante "AZZ" de la tabla Products será la misma que la del fabricante "ANZ" pero sólo para los productos que contengan 'tennis' en su descripción.
 - Tener en cuenta las restricciones de integridad referencial existentes, manejar todo dentro de una misma transacción.

```
begin transaction
insert into manufact (manu_code, manu_name, lead_time)
  values ('AZZ', 'AZZIO SA', 5);
insert into products (manu_code, stock_num, unit_price, unit_code)
  select 'AZZ', p.stock_num, p.unit_price, p.unit_code
    from products p join product_types t on p.stock_num = t.stock_num
    where manu_code = 'ANZ' and t.description like '%tennis%';
commit;
select * from products where manu_code = 'AZZ';
```