1) **Crear una vista que devuelva:**
   a) **Código y Nombre (manu_code,manu_name) de los fabricante, posean o no productos (en tabla stock) , cantidad de productos que poseen en tabla stock (cant_producto) y la fecha de la última OC que contenga un producto suyo (ult_fecha_orden).**
      - **De los fabricantes que fabriquen productos sólo se podrán mostrar los dr**
      - **No se permite utilizar funciones definidas por usuario, ni tablas temporales, ni UNION.**

   b) **Realizar una consulta sobre la vista que devuelva manu_code, manu_name, cant_producto y si el campo ult_fecha_orden posee un NULL informar 'No Posee Órdenes' si no posee NULL informar el valor de dicho campo.**
      - **No se puede utilizar UNION para el SELECT.**

```sql
-- 1a
-- Opción 1
CREATE VIEW vrecu1 AS
SELECT m.manu_code,m.manu_name,
      count(stock_num) cant_productos,
      (SELECT max(order_date)
         FROM orders o JOIN items i
           ON o.order_num=i.order_num
             AND i.manu_code=m.manu_code) ult_compra
FROM manufact m LEFT JOIN products s
  ON s.manu_code = m.manu_code
GROUP BY m.manu_code,m.manu_name
HAVING count(stock_num)=0 OR count(stock_num)>1

-- Opción 2
DROP VIEW vrecu1
CREATE VIEW vrecu1 AS
SELECT m.manu_code,m.manu_name,
       count(distinct s.stock_num) cant_productos,
       max(o.order_date)ult_compra
FROM manufact m
LEFT JOIN products s ON s.manu_code = m.manu_code
LEFT JOIN items i ON s.manu_code = i.manu_code AND s.stock_num=i.stock_num
LEFT JOIN orders o ON i.order_num = o.order_num
GROUP BY m.manu_code,m.manu_name
HAVING count(distinct s.stock_num)=0
    OR count(distinct s.stock_num)>1

-- Opción 3
DROP VIEW vrecu1
CREATE VIEW vrecu1 AS
SELECT m.manu_code,m.manu_name,
count(distinct s.stock_num) cant_productos,
max(o.order_date)ult_compra
FROM manufact m
LEFT JOIN products s ON s.manu_code = m.manu_code
LEFT JOIN items i ON s.manu_code = i.manu_code AND s.stock_num=i.stock_num
LEFT JOIN orders o ON i.order_num = o.order_num
WHERE m.manu_code IN
(SELECT m2.manu_code
   FROM manufact m2 JOIN products s2
     ON (m2.manu_code = s2.manu_code)
     GROUP BY m2.manu_code
     HAVING COUNT(*) >1 OR COUNT(*) = 0)
GROUP BY m.manu_code,m.manu_name
```

```sql
-- 1b
-- Inserto fila de prueba, la borro al final
INSERT INTO manufact VALUES ('PRU','Prueba',99,'CA',NULL,NULL)

-- Opcion 1 con CASE
select manu_code,manu_name,cant_productos,
case when ult_compra is null then 'No posee Productos'
when ult_compra is not null then cast(ult_compra as char) end
from vrecu1
-- Opcion 2 con COALESCE
-- falla por problemas de Casteo
select manu_code,manu_name,cant_productos,
COALESCE(ult_compra,'No posee Productos'
from vrecu1

-- Opcion 2 con COALESCE
select manu_code,manu_name,cant_productos,
COALESCE(cast(ult_compra as char),'No posee Productos')
from vrecu1

-- Borro la fila dummy
DELETE FROM manufact WHERE manu_code='PRU'
```

2) Desarrollar una consulta muestre un ABC de fabricantes que:

Liste el código de fabricante, el nombre del fabricante, la cantidad de órdenes de compra que contentan sus productos y la suma total los productos vendidos.

Se deberán tener en cuenta sólo los fabricantes cuyo código comience con A ó con N y posea 3 letras, y los productos cuya descripción posea el string "tennis" ó el string "ball" en cualquier parte del nombre.

Sólo se podrán mostrar los datos de los fabricantes cuyo total sea mayor que el total de ventas promedio de todos los fabricantes (Cantidad vendida / Cantidad de fabricantes que tuvieron productos vendidos).

La consulta deberá mostrar los registros ordenados por total vendido de mayor a menor.

```
SELECT m.manu_code,m.manu_name,
       COUNT(DISTINCT i.order_num) cantidadOrdenes,
          SUM(total_price*quantity) totalComprado,
          (SELECT COUNT(*) FROM manufact m2) cantFabricantes
FROM   manufact m JOIN items i ON (m.manu_code=i.manu_code)
JOIN product_types p ON (i.stock_num=p.stock_num)
WHERE (description LIKE '%tennis%' OR description LIKE '%ball%')
AND   m.manu_code LIKE '[AN]__'
GROUP BY m.manu_code,m.manu_name
HAVING SUM(total_price*quantity) >
(select SUM(total_price*quantity)
 /count(DISTINCT i.manu_code) from items i)
       ORDER BY 4 DESC
```

3) Crear una vista que devuelva

Mostrar los datos (customer_num,lname,company) de los clientes, posean o no órdenes de compra y la cantidad de órdenes de compra, la fecha de la última OC y el total en u$s (total_price*quantity)comprado y el total general Comprado por todos los clientes.

De los clientes que posean órdenes sólo se podrán mostrar los clientes que tengan alguna órden que posea productos que son fabricados por más de dos fabricantes. Mostrar los clientes que posean menos de 5 órdenes de compra.

Ordenar el reporte primero por los clientes que tengan órdenes por cantidad de órdenes descendente y luego por los clientes que no tengan órdenes

No se permite utilizar funciones, ni tablas temporales.

```
CREATE VIEW v_parcial AS
select c.customer_num, c.lname, c.company, sname,
        null ultima_compra,0 cantidad_ordenes, 0 total_ordenes,
       (select sum(total_price*quantity)  FROM items) total_general
from customer c
inner join state s on c.state =s.code
where customer_num not in (select distinct customer_num from orders)


UNION

select c.customer_num, c.lname, c.company, sname, MAX(order_date),
       count(distinct o.order_num), sum(i.total_price*quantity),
       (select sum(total_price*quantity) FROM items)
from customer c
join orders o on c.customer_num=o.customer_num
join items i on o.order_num = i.order_num
join state s on c.state =s.code
where c.customer_num in
      (select DISTINCT o2.customer_num from orders o2
      JOIN items i2 ON o2.order_num=i2.order_num
      WHERE i2.stock_num IN (SELECT stock_num FROM products
      GROUP BY stock_num HAVING count(*) >2))
group by c.customer_num,c.lname,c.company,sname
having count(distinct o.order_num) < 5

SELECT * FROM v_parcial
order by 6 DESC, 1
```

4) Crear una vista que devuelva

El top 5 de los productos (description) que fueron más comprados en cada estado (state) con la cantidad vendida y su total vendido, teniendo en cuenta que solo se mostrará el estado en el que tuvo mayor cantidad de ventas ese mismo producto.

Ordenarlo por la cantidad vendida descendente.

No se permite utilizar funciones, ni tablas temporales.

```
CREATE VIEW productMasComprados
(TipoProducto,Estado,CantVendida,TotalVendido)
AS
SELECT t.description, c.state,
       SUM(i.quantity),
       SUM(i.total_price*i.quantity)
FROM products s
JOIN items i ON (s.stock_num = i.stock_num)
JOIN product_types t ON (s.stock_num=t.stock_num)
JOIN orders o ON (i.order_num = o.order_num)
JOIN customer c ON (o.customer_num = c.customer_num)
GROUP BY t.description, c.state
HAVING SUM(i.quantity)
= (SELECT TOP 1 SUM(i1.quantity)
FROM products s1
JOIN product_types t1 ON (s1.stock_num=t1.stock_num)
JOIN items i1 ON (s1.stock_num = i1.stock_num)
                    JOIN orders o1 ON (i1.order_num = o1.order_num)
      JOIN customer c1 ON (o1.customer_num = c1.customer_num)
    WHERE t1.description = t.description
    GROUP BY c1.state, t1.description
    ORDER BY SUM(i1.quantity) DESC)
      ORDER BY 1

SELECT TOP 5 * FROM productMasComprados
order by cantVendida DESC
```

5) Se quiere averiguar los customers que no posean órdenes de compra y aquellos cuyas últimas órdenes de compra superen el promedio de las anteriores. Se pide mostrar customer_num, fname, lname, paid_date y el precio total, de las órdenes que tengan la última fecha más reciente.

Ordenar por fecha de pago descendiente.

No se permite utilizar funciones, ni tablas temporales.

```
VERSION 1:

SELECT c.customer_num, c.fname, c.lname, o.paid_date,
SUM(i.total_price)
FROM customer c JOIN orders o ON (c.customer_num = o.customer_num)
JOIN items i ON (o.order_num = i.order_num)
WHERE o.paid_date IN (SELECT MAX(o1.paid_date) FROM customer c1 JOIN
orders o1 ON (c1.customer_num = o1.customer_num)
                        WHERE c1.customer_num = c.customer_num)
GROUP BY c.customer_num, c.fname, c.lname, o.paid_date
HAVING SUM(i.total_price) >= (SELECT AVG(i1.total_price) FROM customer
c1 JOIN orders o1 ON (c1.customer_num = o1.customer_num)
JOIN items i1 ON (o1.order_num = i1.order_num)
WHERE o.paid_date >= o1.paid_date AND c1.customer_num =
c.customer_num)
UNION
SELECT c.customer_num, c.fname, c.lname, o.paid_date,
SUM(i.total_price)
FROM customer c LEFT JOIN orders o ON (c.customer_num =
o.customer_num)
LEFT JOIN items i ON (o.order_num = i.order_num)
WHERE c.customer_num NOT IN (SELECT customer_num FROM orders)
GROUP BY c.customer_num, c.fname, c.lname, o.paid_date
ORDER BY o.paid_date DESC


VERSION 2:

SELECT c.customer_num, c.fname, c.lname, o.paid_date,
SUM(i.total_price)
FROM customer c LEFT JOIN orders o ON (c.customer_num =
o.customer_num)
LEFT JOIN items i ON (o.order_num = i.order_num)
WHERE (o.paid_date IN (SELECT MAX(o1.paid_date) FROM customer c1 JOIN
orders o1 ON (c1.customer_num = o1.customer_num)
                                    WHERE c1.customer_num =
c.customer_num)) OR c.customer_num NOT IN (SELECT customer_num FROM
orders)
GROUP BY c.customer_num, c.fname, c.lname, o.paid_date
HAVING SUM(i.total_price) >= (SELECT AVG(i1.total_price) FROM customer
c1 JOIN orders o1 ON (c1.customer_num = o1.customer_num)
                                                JOIN items i1 ON
(o1.order_num = i1.order_num)
                                                WHERE o.paid_date >=
o1.paid_date AND c1.customer_num = c.customer_num) OR
SUM(i.total_price) IS NULL
ORDER BY o.paid_date DESC
```

6) Se desean saber los fabricantes que vendieron mayor cantidad de un mismo producto que la competencia con la cantidad vendida y su precio total. Tener en cuenta que puede existir un único producto que no sea fabricado por algún otro.

No se permite utilizar funciones, ni tablas temporales.

```
SELECT m.manu_code, m.manu_name, t.description,
       SUM(i.quantity), SUM(i.total_price)
FROM manufact m JOIN products s ON (m.manu_code = s.manu_code)
JOIN items i ON (s.stock_num = i.stock_num)
JOIN product_types t ON (s.stock_num=t.stock_num)
GROUP BY m.manu_code, m.manu_name, t.description
HAVING SUM(i.quantity) > (SELECT TOP 1 SUM(i1.quantity) FROM manufact m1 JOIN
products s1 ON (m1.manu_code = s1.manu_code)
JOIN items i1 ON (s1.stock_num = i1.stock_num)
JOIN product_types t1 ON (s1.stock_num=t1.stock_num)
WHERE (t1.description = t.description AND m1.manu_code != m.manu_code)
GROUP BY m1.manu_code, m1.manu_name, t1.description
ORDER BY 1 DESC)
OR (SELECT COUNT(*) FROM products s2
JOIN product_types t2 ON (s2.stock_num=t2.stock_num)
WHERE t2.description = t.description) = 1
```