1) Crear una vista que devuelva:
   a) Código y Nombre (manu_code,manu_name) de los fabricante, posean o no productos (en tabla ***Products***), cantidad de productos que fabrican (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 que fabriquen más de 2 productos.
      - 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.

```
-- 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)>2

-- 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)>2

-- 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(*)>2 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') ultcompra
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 ABC de fabricantes que:

Liste el código y nombre del fabricante, la cantidad de órdenes de compra que contengan sus productos y la monto total de los productos vendidos.
Mostrar sólo los fabricantes cuyo código comience con A ó con N y posea 3 letras, y los productos cuya descripción posean el string “tennis” ó el string “ball” en cualquier parte del nombre y cuyo monto total vendido sea mayor que el total de ventas promedio de todos los fabricantes (Cantidad * precio unitario / Cantidad de fabricantes que vendieron sus productos).
Mostrar los registros ordenados por monto total vendido de mayor a menor.

```
SELECT m.manu_code, m.manu_name,
       COUNT(DISTINCT i.order_num) cantidadOrdenes,
       SUM(unit_price*quantity) totalVendido
 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(unit_price*quantity) >
                    (select SUM(unit_price*quantity)
                            /count(DISTINCT i.manu_code) from items i)
                      ORDER BY 4 DESC;
```

3) Crear una vista que devuelva

Para cada cliente mostrar (customer_num, lname, company), cantidad de órdenes de compra, fecha de su última OC, monto total 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 orden que posea productos que son fabricados por más de dos fabricantes y que tengan al menos 3 órdenes de compra.

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

No se permite utilizar funciones, ni tablas temporales.

```
CREATE VIEW v_parcial AS
select 2, c.customer_num, c.lname, c.company,
       0 cantidad_ordenes,
       null ultima_compra,
       0 montoTotal,
       (select sum(unit_price*quantity) FROM items) total_general
  from customer c
 where customer_num not in (select customer_num from orders)
 UNION
select 1, c.customer_num, c.lname, c.company,
       count(distinct o.order_num),
       MAX(order_date),
       sum(i.unit_price*quantity),
       (select sum(unit_price*quantity) FROM items ) total_general
 from customer c
         join orders o on c.customer_num = o.customer_num
         join items i on o.order_num = i.order_num
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
having count(distinct o.order_num) >= 3

SELECT * FROM v_parcial
order by 1, 5 DESC
```

4) Crear una consulta que devuelva los 5 primeros estados y el tipo de producto (description) más comprado en ese estado (state) según la cantidad vendida del tipo de producto.
Ordenarlo por la cantidad vendida en forma descendente.
Nota: No se permite utilizar funciones, ni tablas temporales.

```
SELECT top 5 t.description, c.state,  SUM(i.quantity)
  FROM items i JOIN product_types t ON i.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 i.stock_num, t.description, c.state
 HAVING i.stock_num =
  (SELECT TOP 1 i1.stock_num
     FROM product_types t1 JOIN items i1 ON i1.stock_num = t1.stock_num
                           JOIN orders o1 ON i1.order_num = o1.order_num
                           JOIN customer c1 ON o1.customer_num = c1.customer_num
    WHERE c.state = c1.state
    GROUP BY i1.stock_num, c1.state
    ORDER BY SUM(i1.quantity) DESC)
ORDER BY   SUM(i.quantity) desc
```

5) Listar los customers que no posean órdenes de compra y aquellos cuyas últimas órdenes de compra superen el promedio de todas las anteriores.
Mostrar customer_num, fname, lname, paid_date y el monto total de la orden que supere el promedio de las anteriores. Ordenar el resultado por monto total en forma descendiente.

V1

```
SELECT c.customer_num, c.fname, c.lname, o.paid_date,
       SUM(i.unit_price * i.quantity) Total
  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.order_num = (SELECT MAX(order_num)
                        FROM orders o1
                       WHERE o1.customer_num = c.customer_num)
 GROUP BY c.customer_num, c.fname, c.lname, o.paid_date, o.order_num
HAVING SUM(i.unit_price * i.quantity) >=
           (SELECT SUM(i1.unit_price * i1.quantity)/count(distinct o1.order_num)
              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.order_num > o1.order_num AND c1.customer_num = c.customer_num)
UNION
SELECT c.customer_num, c.fname, c.lname, null, null
  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 5 DESC
```

V2

```
SELECT c.customer_num, c.fname, c.lname, o.paid_date,
       SUM(i.unit_price*i.quantity) total
  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.order_num = (SELECT MAX(order_num) FROM orders
                        WHERE 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.order_num, o.paid_date
HAVING SUM(i.unit_price * i.quantity) >=
                (SELECT sum(i1.unit_price * i1.quantity) /
                        count(distinct o1.order_num)
                   FROM orders o1 JOIN items i1 ON o1.order_num = i1.order_num
                   WHERE o.order_num > o1.order_num
                     AND o1.customer_num = c.customer_num)
        OR SUM(i.unit_price * i.quantity) IS NULL
ORDER BY SUM(i.unit_price * i.quantity) desc
```

6) Se desean saber los fabricantes que vendieron mayor cantidad de un mismo producto que la competencia según la cantidad vendida. Tener en cuenta que puede existir un producto que no sea fabricado por ningún otro fabricante y que puede haber varios fabricantes que tengan la misma cantidad máxima vendida.
Mostrar el código del producto, descripción del producto, código de fabricante, cantidad vendida, monto total vendido. Ordenar el resultado código de producto, por cantidad total vendida y por monto total, ambos en forma decreciente.
Nota: No se permiten utilizar funciones, ni tablas temporales.

```
SELECT i.stock_num, i.manu_code, pt.description,
       SUM(i.quantity) cantidad,
       SUM(i.unit_price* i.quantity) totalVendido
  FROM items i join product_types pt on i.stock_num = pt.stock_num
 GROUP BY i.manu_code, i.stock_num, pt.description
HAVING SUM(i.quantity) >=
          coalesce((SELECT TOP 1 SUM(i2.quantity)
                      FROM items i2
                     WHERE i2.stock_num = i.stock_num
                       AND i2.manu_code != i.manu_code
                     GROUP BY i2.manu_code, i2.stock_num
                      ORDER BY SUM(i2.quantity) DESC), 0)
order by i.stock_num, cantidad DESC, totalVendido DESC
```

Otra posible solución

```
select i.stock_num, i.manu_code, p.description,
       sum(quantity) cantidadVendida, sum(i.unit_price*quantity) totalVendido
  from items i join product_types p ON i.stock_num = p.stock_num
 group by i.stock_num, i.manu_code, p.description
having sum(quantity) = (select top 1 sum(i2.quantity)
                          from items i2
                          where i2.stock_num = i.stock_num
                          group by i2.manu_code
                          order by sum(i2.quantity) desc)
order by 1, 4 DESC, 5 DESC
```