1.

a. Crear una vista que devuelva: código y nombre (manu_code, manu_name) de los fabricantes, posean o no productos (en tabla products), cantidad de productos que poseen en tabla products (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 un producto.
- No se permite utilizar funciones definidas por usuario, ni tablas temporales, ni UNION.

```
CREATE VIEW v_punto1 AS

    SELECT m.manu_code,

           m.manu_name,

           COUNT(p.stock_num) AS 'cant_producto',

           (SELECT MAX(o2.order_date)

            FROM orders o2

                JOIN items i2 ON (o2.order_num = i2.order_num)

            WHERE i2.manu_code = m.manu_code) AS 'ult_fecha_orden'

    FROM manufact m
        LEFT JOIN products p ON (m.manu_code = p.manu_code)

    GROUP BY m.manu_code, m.manu_name

    HAVING COUNT(p.stock_num) >= 0
```

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.

```
SELECT manu_code,

       manu_name,

       cant_producto,

       COALESCE(ult_fecha_orden, 'No posee órdenes')

FROM v_punto1
```

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 H y posea 3 letras, y los productos cuya descripción posea el string “tennis” ó el string “ball”.
- Sólo se podrán mostrar los datos de los fabricantes cuyo total sea mayor que el total de ventas promedio por cada fabricante (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) AS 'cantidad_ordenes',

       SUM(i.quantity * i.unit_price) AS 'total_comprado'

FROM manufact m

    JOIN products p ON (m.manu_code = p.manu_code)

    JOIN items i ON (p.stock_num = i.stock_num AND p.manu_code = i.manu_code)

    JOIN product_types tp ON (p.stock_num = tp.stock_num)

WHERE m.manu_code LIKE 'H__'

  AND (tp.description LIKE '%tennis%' OR tp.description LIKE '%ball%')

GROUP BY m.manu_code, manu_name

HAVING (SUM(i.quantity * i.unit_price)) >

       ((SUM(i.quantity * i.unit_price)) / (COUNT(DISTINCT m.manu_code)))

ORDER BY total_comprado
```

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 el total en U$S (total_price) 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_punto3 AS

    SELECT c.customer_num,

           c.lname,

           c.company,

           COUNT(o.order_num) AS 'cant_ordenes',

           (SELECT MAX(o2.order_date)

            FROM orders o2

            WHERE o2.customer_num = c.customer_num) AS 'ult_fecha_orden',

           SUM(i.quantity * i.unit_price) AS 'total_en_dólares',

           (SELECT SUM(i2.quantity * i2.unit_price)

            FROM items i2) AS 'total_general_comprado_por_todos_los_clientes'

    FROM customer c

        LEFT JOIN orders o ON (c.customer_num = o.customer_num)

        JOIN items i ON (i.order_num = o.order_num)

    WHERE stock_num IN (SELECT p.stock_num

                        FROM products p

                        GROUP BY p.stock_num

                        HAVING COUNT(p.manu_code) > 2)

    GROUP BY c.customer_num, c.lname, c.company

    HAVING COUNT(o.order_num) < 5

    ORDER BY cant_ordenes DESC
```

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 precio total, teniendo en cuenta que solo se mostrará el estado en el que tuvo mayor cantidad de ventas de un mismo producto.

- Ordenarlo por la cantidad vendida descendiente.
- No se permite utilizar funciones, ni tablas temporales.

```
CREATE VIEW productMasComprados AS

    SELECT t.description AS tipoProducto,

           c.state AS estado,

           SUM(i.quantity) AS cantVendida,

           SUM(i.unit_price * i.quantity) AS totalVendido

    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)

SELECT TOP 5 *

FROM productMasComprados

ORDER BY cantVendida DESC
```

5. Se quiere averiguar los clientes 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.

a. Realizar la solución utilizando UNION.

- Ordenar por fecha de pago descendiente.
- No se permite utilizar funciones, ni tablas temporales.

```
SELECT c.customer_num,
       c.fname,
       c.lname,
       o.paid_date,
       SUM(i.unit_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.unit_price) >=
    (SELECT AVG(i1.unit_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.unit_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
```

b. Realizar la solución sin implementar UNION.

- Ordenar por fecha de pago descendiente.
- No se permite utilizar funciones, ni tablas temporales.

```
SELECT c.customer_num,

       c.fname,

       c.lname,

       o.paid_date,

       SUM(i.unit_price) AS precioTotal

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.unit_price) >= (SELECT AVG(i1.unit_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.unit_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) cantidad,

       SUM(i.unit_price* i.quantity) totalVendido

FROM manufact m

    JOIN items i ON (m.manu_code = i.manu_code)

    JOIN product_types t ON (i.stock_num=t.stock_num)

GROUP BY m.manu_code, m.manu_name, t.stock_num, t.description

HAVING SUM(i.quantity) >=

       COALESCE((SELECT TOP 1 SUM(i2.quantity)

                 FROM manufact m2

                     JOIN items i2 ON (m2.manu_code = i2.manu_code)

                     JOIN product_types t2 ON (i2.stock_num = t2.stock_num)

                 WHERE t2.stock_num = t.stock_num

                   AND m2.manu_code != m.manu_code

                 GROUP BY m2.manu_code, m2.manu_name, t2.description

                 ORDER BY 1 DESC), 0)

ORDER BY 3
```