1.c SQL (60 pts)

Seleccionar código de fabricante, nombre fabricante, cantidad de órdenes del fabricante, cantidad total vendida del fabricante, promedio de las cantidades vendidas de todos los Fabricantes de todos aquellos fabricantes cuyas ventas totales sean mayores al PROMEDIO de las ventas de TODOS los fabricantes.

Mostrar el resultado ordenado por cantidad total vendida en forma descendente.
IMPORTANTE: No se pueden usar procedures, ni Funciones de usuario.

manu_code	manu_name	CantOrdenes	Total vendido	promedioDeTodoslosFabricantes
ANZ	Anza	11	11081.80	3972.85
SHM	Shimara	4	5677.91	3972.85

2.a Stored Procedure (45 pts)

Crear un procedimiento procBorraOC que reciba un número de orden de compra por parámetro y realice la eliminación de la misma y sus ítems.

Deberá manejar una transacción y deberá manejar excepciones ante algún error que ocurra.

El procedimiento deberá guardar en una tabla de auditoria auditOC los siguientes datos order_num, order_date, customer_num, cantidad_items, total_orden (SUM(total_price)), cant_productos_comprados (SUM(quantity)), cantidad de ítems.

Ante un error deberá almacenar en una tablas erroresOC, order_num, order_date, customer_num, error_ocurrido VARCHAR(50).

2.e Triggers (55 pts)

En el caso que el cliente (customer_n…

Dada la siguiente tabla CURRENT_STOCK

```
create table CURRENT_STOCK (
       stock_num            smallint not null,
       manu_code            char(3) not null,
       CURRENT_AMOUNT       integer default 0,
       created_date         datetime not null, -- fecha de creación del registro
       updated_date         datetime not null, -- última fecha de actualización del registro
       PRIMARY KEY (stock_num, manu_code)
);
```

Crear un trigger que ante un insert, update o delete en la tabla ITEMS actualice la cantidad CURRENT_AMOUNT de la tabla CURRENT_STOCK de forma tal que siempre contenga el stock actual del par (stock_num, manu_code).

Si la operación es un INSERT se restará la cantidad QUANTITY al CURRENT_AMOUNT.
Si la operación es un DELETE se sumará la cantidad QUANTITY al CURRENT_AMOUNT.
Si la operación es un UPDATE se sumará la cantidad QUANTITY nueva y se restará la anterior al CURRENT_AMOUNT.

Si no existe el par (stock_num, manu_code) en la tabla CURRENT_STOCK debe insertarlo en la tablA CURRENT_STOCK con el valor inicial de 0 (cero) mas/menos la operación a realizar.

Tener en cuenta que las operaciones (INSERTs, DELETEs, UPDATEs) pueden ser masivas.

1.c SQL (60 pts)

- ***Solución 1***

```
SELECT m.manu_code,
       m.manu_name,
       COUNT(distinct i.order_num) CantOrdenes,
       SUM(quantity * total_price) total,
       (SELECT SUM(quantity * total_price) / COUNT(distinct manu_code)
        FROM items i)  promedioDeTodoslosFabricantes
FROM manufact m JOIN items i ON (m.manu_code = i.manu_code)
GROUP BY m.manu_code, m.manu_name
HAVING SUM(quantity * total_price) > (SELECT SUM(quantity * total_price) / COUNT(distinct manu_code)
                                      FROM items i)
```

- ***Solución 2***

```
SELECT m.manu_code,
       m.manu_name,
       COUNT(distinct i.order_num) CantOrdenes,
       SUM(quantity * total_price) total,
       (SELECT AVG(A.total) total
        FROM (SELECT manu_code, SUM(quantity * total_price) total
              FROM items i
              GROUP BY manu_code)a)  promedioDeTodoslosFabricantes
 FROM manufact m JOIN items i ON (m.manu_code = i.manu_code)
 GROUP BY m.manu_code, m.manu_name
 HAVING SUM(quantity * total_price) > (SELECT avg(A.total) total
                                       FROM (SELECT manu_code,
                                                    SUM(quantity * total_price) total
                                             FROM items i
                                             GROUP BY manu_code)
```

2.e Triggers (55 pts)

```
CREATE TRIGGER itemsTR ON items
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
--
    DECLARE @stock_num smallint, @manu_code char(3), @quantityI smallint, @quantityD smallint
    DECLARE Actualizados CURSOR FOR
        SELECT COALESCE(i.stock_num, d.stock_num) stock_num,
               COALESCE(i.manu_code, d.manu_code) manu_code,
               i.quantity, d.quantity
        FROM inserted i FULL JOIN deleted d ON (i.order_num = d.order_num AND i.item_num = d.item_num)
    OPEN Actualizados
    FETCH Actualizados
        INTO @stock_num, @manu_code, @quantityI, @quantityD
    WHILE @@FETCH_STATUS=0
    BEGIN
        IF NOT EXISTS (SELECT 1 FROM CURRENT_STOCK p WHERE p.manu_code = @manu_code AND p.stock_num = @stock_num)
            INSERT INTO CURRENT_STOCK (stock_num, manu_code, Current_Amount, created_date, updated_date)
                VALUES (@stock_num, @manu_code, 0, GETDATE(), GETDATE());
        --
        UPDATE CURRENT_STOCK
            SET Current_Amount = Current_Amount - COALESCE(@quantityI, 0) + COALESCE(@quantityD, 0),
                updated_date = getdate()
            WHERE stock_num = @stock_num AND manu_code = @manu_code;

        FETCH Actualizados
            INTO @stock_num, @manu_code, @quantityI, @quantityD;
   END

   CLOSE Actualizados
   DEALLOCATE Actualizados
END
```