Triggers

a. Se pide: Crear un trigger que valide que ante un insert (de una o más filas) en la tabla ítems, realice la siguiente validación.

- **Si la órden de compra a la que pertenecen los ítems ingresados corresponde a clientes del estado de California, se deberá validar que las órdenes pueden contar con hasta 5 registros en la tabla ítem.**
- **Si se insertan más ítems de los definidos, el resto de los ítems se deberán insertar en la tabla items_error la cuál contiene la misma estructura que la tabla ítems más un atributo fecha que deberá contener la fecha del día en que se trató de insertar.**

Si por ejemplo la OC cuenta con 3 items y se realiza un insert masivo de 3 ítems más, el trigger deberá insertar los 2 primeros en la tabla ítems y el restante en la tabla ítems_error.

Supuesto: En el caso de un insert masivo los ítems pertenecen siempre a la misma órden.

```
create trigger Tr_temaA
on items
instead of insert
AS
BEGIN
    declare @stock_num  smallint, @order_num smallint, @item_num
smallint,
                @quantity smallint
    declare @total_price decimal(8,2)
    declare @manu_code char(3),@state char(2)

  declare c_call cursor
  for select i.*,state from inserted i
                                                    JOIN orders o
ON (i.order_num=o.order_num)
                                                    JOIN customer c
ON (o.customer_num=c.customer_num)

  open c_call
  fetch from c_call into
@item_num,@order_num,@stock_num,@manu_code,
               @quantity,@total_price,@state

  while @@fetch_status=0
           BEGIN
              if @state='CA'
          begin
            if (select COUNT(*) FROM items where
order_num=@order_num) < 5
               begin
                    INSERT INTO items

VALUES(@item_num,@order_num,@stock_num,@manu_code,

@quantity,@total_price)

               end
            else
               begin
                    INSERT INTO items_error

VALUES(@item_num,@order_num,@stock_num,@manu_code,
```

```
@quantity,@total_price,getDate())

                end

           end
        else
           begin
                     INSERT INTO items

VALUES(@item_num,@order_num,@stock_num,@manu_code,

@quantity,@total_price)
           end

  fetch from c_call into
@item_num,@order_num,@stock_num,@manu_code,
                @quantity,@total_price,@state

      END
  close c_call
  deallocate c_call
END


----pruebas del trigger

CREATE TABLE [dbo].[items_error](
  [item_num] [smallint] NOT NULL,
  [order_num] [smallint] NOT NULL,
  [stock_num] [smallint] NOT NULL,
  [manu_code] [char](3) COLLATE Traditional_Spanish_CI_AS NOT
NULL,
  [quantity] [smallint] NULL DEFAULT ((1)),
  [total_price] [decimal](8, 2) NULL,
  [fecha] [datetime] NULL
)

select * from items where order_num in(
select order_num from orders o, customer c
where o.customer_num=c.customer_num
and c.state='CA')

insert into items values (14,1003,9,'ANZ',1,10)
insert into items values (15,1003,9,'ANZ',1,10)
insert into items values (16,1003,9,'ANZ',1,10)
insert into items values (17,1003,9,'ANZ',1,10)
insert into items values (18,1003,9,'ANZ',1,10)

select * from items_error
```

Triggers Dada la siguiente vista (20 pts)

```
CREATE VIEW ProdPorFabricante AS
SELECT manu_code, manu_name, COUNT(*)
FROM manufact m INNER JOIN stock s ON (m.manu_code = s.manu_code)
GROUP BY manu_code, manu_name
```

Crear un trigger que permita ante un insert en la vista ProdPorFabricante insertar una en la tabla manufact.

Observaciones: **el atributo leadtime deberá insertarse con un valor default 10**
El trigger deberá contemplar inserts de varias filas, ante un INSERT / SELECT.

```
insert into ProdPorFabricante (manu_code, manu_name)
values ('ABC','ABCDE XXX')

CREATE TRIGGER insFabric
ON ProdPorFabricante
INSTEAD OF INSERT
AS
BEGIN
  INSERT INTO manufact
      select manu_code,manu_name,100
      from inserted
END
```

b. Se pide: Crear un trigger que valide que ante un update (de una o más filas) en la tabla customer, realice la siguiente validación.

- **La cuota de clientes correspondientes al estado de California es de 20, si se supera dicha cuota se deberán grabar el resto de los clientes en la tabla customer_update_pend.**
- **Validar que si de los clientes a modificar se modifica el Estado, no se puede superar dicha cuota.**

Si por ejemplo el estado de CA cuenta con 18 clientes y se realiza un update masivo de 5 clientes con estrado de CA, el trigger deberá modificar los 2 primeros en la tabla customer y los restantes grabarlos en la tabla customer_updates_pend.
La tabla customer_updates_pend tendrá la misma estructura que la tabla customer con un atributo adicional fecha que deberá actualizarse con la fecha y hora del día.

```
create trigger temaB
on customer
instead of update
AS
BEGIN
    declare @customer_num  smallint
    declare @fname varchar(15), @lname varchar(15),@city
varchar(15)
    declare @company varchar(20),@address1 varchar(20),@address2
varchar(20)
    declare @state char(2), @state_old char(2)
    declare @zipcode char(18)
    declare @phone varchar(18)

      declare c_call cursor
      for select i.*,d.state
                  from inserted I join deleted d
                     on (i.customer_num=d.customer_num)

      open c_call
      fetch from c_call into
@customer_num,@fname,@lname,@company,

@address1,@address2,@city,@state,@zipcode,@phone,@state_old
      while @@fetch_status=0
      BEGIN
        if @state='CA' and @state!=@state_old
           begin
             if (select COUNT(*) FROM customer where state='CA')
< 20
                begin
                     UPDATE customer
                        SET
fname=@fname,lname=@lname,company=@company,

address1=@address1,address2=@address2,city=@city,

state=@state,zipcode=@zipcode,phone=@phone
                      WHERE customer_num=@customer_num

                end
             else
                begin
                     INSERT INTO customer_updates_pend
```

```
                                VALUES (@customer_num,@fname,
@lname,@company,
                                                      @address1,
@address2,

@city,@state,@zipcode,@phone,getDate())

                  end

            end
         else
            begin
              UPDATE customer
                           SET
fname=@fname,lname=@lname,company=@company,

address1=@address1,address2=@address2,city=@city,

state=@state,zipcode=@zipcode,phone=@phone
                        WHERE customer_num=@customer_num
            end

           fetch NEXT from c_call into
@customer_num,@fname,@lname,@company,

@address1,@address2,@city,@state,@zipcode,@phone,@state_old

        END
       close c_call
       deallocate c_call
END


-------  Pruebas

CREATE TABLE customer_updates_pend(
   customer_num smallint NOT NULL,
   fname varchar(15) ,
   lname varchar(15),
   company varchar(20),
   address1 varchar(20),
   address2 varchar(20)
   city varchar(15),
   state char(2),
   zipcode char(5),
   phone varchar(18),
   fecha datetime )

select count(*) from customer where state='CA'

select customer_num,state from customer where customer_num between
123 and 126

update customer set state='CA' where customer_num between 122 and
126

select * from customer_updates_pend
```

c. Dada la siguiente vista

```
CREATE VIEW ProdPorFabricanteDet AS
SELECT m.manu_code, manu_name, stock_num, description
FROM manufact m LEFT OUTER JOIN stock s ON (m.manu_code = s.manu_code)
```

Se pide: Crear un trigger que permita ante un DELETE en la vista ProdPorFabricante borrar los datos en la tabla manufact pero sólo de los fabricantes cuyo campo description sea NULO (o sea que no tienen stock).
Observaciones: El trigger deberá contemplar borrado de varias filas, ante un DELETE masivo. En ese caso sólo borrará de la tabla los fabricantes que no tengan productos en stock.

```
CREATE TRIGGER delFabric
ON ProdPorFabricanteDet
INSTEAD OF DELETE
AS
BEGIN
  DELETE FROM manufact WHERE manu_code IN (select manu_code from
deleted where description IS NULL)
END
```