1. Crear un trigger que ante el intento de borrado de un cliente realice un cambio de estado a ‘I’ Inactivo del mismo. Tener en cuenta que el borrado de los clientes puede ser masivo.

2. Crear un trigger que ante un borrado sobre la tabla ***FACTURAS*** realice un borrado en cascada sobre la tabla ***DETALLE***, validando que sólo se intente borrar 1 factura. Si detecta que están queriendo borrar más de una factura, informe un error y aborte la operación.

3. Crear la vista ProductosFabricanteV que tenga los siguientes atributos: producto_cod, producto_desc, precio_unit, fabricante_cod, fabricante_nom, provincia_cod. Crear un trigger sobre la vista anterior que ante un insert, inserte una fila en la tabla Productos. Si el fabricante no existe, inserte una fila en dicha tabla con el campo lead_time en 1 validando también que exista la provincia del fabricante. Los inserts pueden ser masivos, SI hay un error informarlo.

4. Dada la tabla PRECIOS_HIST realizar un trigger que verifique ante INSERTS en dicha tabla, que no se superpongan los periodos para los precios de los productos. Solo se insertará de a UN registro, si se trata de insertar mas de un registro devolver un error.

5. Crear un trigger que ante un cambio de precios en un producto inserte un nuevo registro con el precio cambiado (el anterior, no el nuevo) en la tabla PRECIOS_HIST. La ***fecha desde*** del nuevo registro será la ***fecha hasta*** del último cambio de precios de ese producto y su ***fecha hasta*** será la fecha del dia. SI no tuviese un registro de precio anterior ingrese como ***fecha desde*** ‘2000-01-01’

1.

```
ALTER trigger BorrarCliente on clientes
INSTEAD OF DELETE AS
begin
    update clientes
          set estado = 'I'
       where cliente_num in (select cliente_num from deleted)
end
```

2.

```
ALTER trigger BorrarFactura on facturas
INSTEAD OF DELETE AS
begin
    if (select count(*) from deleted) > 1
          THROW 50001, 'No puede borrar mas de 1 factura', 1

    DELETE facturas_det
        where factura_num in (select factura_num from deleted)
       --
       DELETE facturas
        where factura_num in (select factura_num from deleted)
   --
End
```

3.

```
Create view ProductosFabricanteV as
   select producto_cod, producto_desc, precio_unit,
          f.fabricante_cod, fabricante_nom, provincia_cod
        from productos p join fabricantes f on p.fabricante_cod = f.fabricante_cod;


CREATE trigger insertProdFabricanteTr on ProductosFabricanteV
INSTEAD OF Insert AS
begin
       declare @producto_cod int, @producto_desc varchar(30),
              @precio_unit decimal(10,2),
                    @fabricante_cod varchar(5), @fabricante_nom varchar(20),
                    @provincia_cod char(2)
       --
       DECLARE ProdFab_c CURSOR FOR
              select producto_cod, producto_desc, precio_unit,
                  fabricante_cod, fabricante_nom, provincia_cod
                     from INSERTED;
       --
       OPEN ProdFab_c
       fetch ProdFab_c into @producto_cod, @producto_desc, @precio_unit,
                         @fabricante_cod, @fabricante_nom, @provincia_cod
       while @@FETCH_STATUS = 0
       begin
           if not exists (select 1 from fabricantes
                           where fabricante_cod = @fabricante_cod)
        begin
```

```
                if not exists (select 1 from provincias
                               where provincia_cod = @provincia_cod)
                               THROW 50001, 'Provincia Inexistente', 1
            insert into fabricantes (fabricante_cod, fabricante_nom,
                                                tiempo_entrega, provincia_cod)
                           values (@fabricante_cod, @fabricante_nom, 1, @provincia_cod);
              end
        insert into productos (producto_cod, producto_desc,
                                       precio_unit, fabricante_cod)
                    values (@producto_cod, @producto_desc, @precio_unit,
                            @fabricante_cod);
           fetch ProdFab_c into @producto_cod, @producto_desc, @precio_unit,
                            @fabricante_cod, @fabricante_nom, @provincia_cod
     end
        CLOSE ProdFab_c
        DEALLOCATE ProdFab_c
--
END
```

4.

```
ALTER trigger InsertPrecioHist on precios_hist
INSTEAD OF INSERT AS
begin
    declare @fechaDde date, @fechaHta date, @producto smallint,
              @precio_unit decimal(10,2)

       select @producto=producto_cod, @fechaDde=fechadde, @fechaHta=fechaHta,
              @precio_unit=precio_unit
         from inserted;
    --
       if (select count(*) from precios_hist h
            where h.producto_cod = @producto
                 and  (@fechaDde <= h.fechaDde and h.fechaDde < @fechaHta
                  or @fechaDde < h.fechaHta and h.fechaHta < @fechaHta
                     or h.fechaDde < @fechaHta and @fechaHta <= h.fechaHta )
           ) > 0
              THROW 50001, 'Periodo superpuesto', 1
    --
       insert into Precios_hist (producto_cod, fechaDde, fechaHta, precio_unit)
                    values (@producto, @fechaDde, @fechaHta, @precio_unit)
    --
END
```

5.

```
ALTER trigger cambiarPrecio on productos
AFTER UPDATE AS
begin
       declare @producto_cod int, @precio_unit decimal(10,2)
       declare @fechaDde date, @fechaHta date
       set @fechaDde = null
       --
       select @producto_cod=producto_cod, @precio_unit=precio_unit
         from DELETED;
```

```
    --
       select @fechaDde=coalesce(max(fechaHta), '2000-01-01')
         from precios_hist
        where producto_cod = @producto_cod;
       --
       insert into Precios_hist (producto_cod, fechaDde, fechaHta, precio_unit)
                    values (@producto_cod, @fechaDde, getDate()+1, @precio_unit)
    --
END
```