Ejercicio 1:

TRIGGER	EVENTO	TABLA	TAREA
BEFORE	INSERT	DETALLE	SI f.ESTADO ES 1 O 2 → EXCEPTION
BEFORE	INSERT	DETALLE	SI p.STOCK < NEW.CANTIDAD →EXCEPTION
BEFORE	DELETE	DETALLE	F.ESTADO = 1 O 2 →EXCEPTION
AFTER	INSERT	DETALLE	p.STOCK = p.STOCK – d.CANTIDAD
AFTER	DELETE	DETALLE	p.STOCK = p.STOCK + d.CANTIDAD
AFTER	UPDATE	DETALLE	RECALCULAR MONTO FACTURA
BEFORE	UPDATE	DETALLE	IF NEW.NRO <> OLD.NRO OR NEW.IDP <> OLD.IDP →EXCEPTION IF NEW.CANTIDAD <> OLD.CANTIDAD AND P.STOCK < NEW.CANTIDAD →EXCEPTION If f.estado = 1 o 2 → EXCEPTION
BEFORE	UPDATE	PAGO	IF ESTADO = 1 O 2→EXCEPTION. IF NEW.NRO <> OLD.NRO OR NEW.NRO <> OLD.NRO →EXCEPTION
BEFORE	UPDATE	FACTURA	IF NEW.NRO <> OLD.NRO→ EXCEPTION
After	UPDATE	FACTURA	IF F.ESTADO = 2→ STOCK =STOCK + CANTIDAD

Create exception EX_Factutra_terminada 'La factura se encuentra terminada, error!';
Créate exception EX_primary_key 'No se puede modificar la clave primaria, error!' ;
Créate exception EX_stock_insuficiente ' El stock es insuficiente, error!';

```
Créate  trigger TRG_BIDETALLE for DETALLE
Active before insert position 0
As
Begin
        If((select estado from factura where nro = new.nro) = 1 or  (select estado from factura
        where nro = new.nro)) = 2 then
                EX_Factutra_terminada;

        If(select cantidad from producto where idp = new.idp) < new.cantidad then
                EX_stock_insuficiente;
End^
```

```
Créate trigger TRG_BUDETALLE for DETALLE
Active before update position 0
As
Begin
        If(new.nro<> old.nro or new.idp<>old.idp) then
                EX_primary_key;
```

```
        If((select estado from factura where nro = new.nro) = 1 or (select estado from factura
        where nro=new.nro) = 2 ) then
                EX_Factutra_terminada;

        If(select cantidad from producto where idp = new.idp) < new.cantidad then
                EX_stock_insuficiente;

End^


Créate trigger TRG_BDDETALLE FOR DETALLE
Active before delete postion 0
As
Begin
        If((select estado from factura where nro = old.nro) = 1 or (select estado from factura
        where nro=old.nro) = 2 ) then
                EX_Factutra_terminada;

End^




Créate trigger TRG_AUDETALE for DETALLE
Active after update position 0
As
        Declare resto float;
Begin
        resto = ((old.precio * old.cantidad) – (new.precio * new.cantidad));

        If (resto > 0 ) then;
                Update factura set monto = monto – resto where nro=new.nro;
        Else
                Update factura set monto = monto + abs(resto) where nro = new.nro;
End^



Créate trigger TRG_BUPAGO for PAGO
Active before update position 0
As
Begin
        If(new.nro<> old.nro or new.nropago<>old.nropago )then
                EX_primary_key;
End^
```

```
Créate trigger TRG_BUFACTURA for FACTURA
Active before update position 0
As
Begin
        If(new.nro<> old.nro) then
                EX_primary_key;

End^
```

```
Créate trigger TRG_AUFACTURA for FACUTRA
Active after update position 0
As
Begin
        If(new.estado = 2)then
                Update producto set stock = stock + old.cantidad;

End^
```

Ejercicio 2 .

```
CREATE PROCEDURE P_CIERRE(ANNO SMALLINT ,MES SMALLINT)
RETURNS (FACTURAS_DEL INTEGER)
AS
        DECLARE VARIABLE NRO TYPE OF COLUMN FACTURA.NRO;
BEGIN
FACTURAS_DEL = 0;
        FOR
                SELECT F.NRO
                FROM FACTURA F
                WHERE EXTRACT(YEAR FROM F.FECHA) = : ANNO AND
                 EXTRACT(MONTH FROM F.FECHA) = :MES AND
                ESTADO = 2 INTO :NRO
        DO
        BEGIN
                DELETE FROM PAGO WHERE NRO =:NRO;
                DELETE FROM DETALLE WHERE NRO =:NRO;
                DELETE FROM FACTURA WHERE NRO = :NRO;
                FACTURAS_BORRADAS = FACTURAS_BORRADAS + 1;
        END
END^
```

Ejercicio 3:

```
Créate procedure P_STOTALFAC(anno smallint)
Returns(t1 float, t2 float,t3 float, t4 float, TOTAL float)
As
Begin
        if (exists(select * from factura where ((extract (YEAR from FECHA)= : anno) and (extract
        (MONTH from FECHA)>=1) and (extract (MONTH from FECHA)<=3))))then begin
        :t1=( select SUM(f.monto) from facture f where ((extract (YEAR from FECHA)= : anno)
        and (extract (MONTH from FECHA)>=1) and (extract (MONTH from FECHA)<=3)));
         end
                ELSE :t1=0;


  if (exists (select * from factura where ((extract (YEAR from FECHA)= :anno) and
(extract(MONTH from FECHA)>=4) and (extract (MONTH from FECHA)<=6)))) THEN BEGIN
     :t2=( select SUM(f.monto) from factura f where ((extract (YEAR from FECHA)= :anno) and
(extract (MONTH from FECHA)>=4) and (extract (MONTH from FECHA)<=6)));
  end
        ELSE :t2=0;

  if (exists (select * from factura where ((extract (YEAR from FECHA)= :anno) and
(extract(MONTH from FECHA)>=7) and (extract (MONTH from FECHA)<=9)))) THEN BEGIN
     :t3=( select SUM(f.monto) from factura f where ((extract (YEAR from FECHA)= :anno) and
(extract (MONTH from FECHA)>=7) and (extract (MONTH from FECHA)<=9)));
  end
        ELSE :t3=0;

 if (exists (select * from factura WHERE ((extract(YEAR from FECHA)= :anno) and (extract
(MONTH from FECHA)>=10) and (extract (MONTH from FECHA)<=12)))) THEN BEGIN
     :t4=( select SUM(f.monto) from factura f where ((extract(YEAR from FECHA)= :anno) and
(EXTRACT(MONTH from FECHA)>=10) and (extract (MONTH from FECHA)<=12)));
  end
        else :t4=0;

  :TOTAL= :t1 + :t2 + :t3 + :t4;
  suspend;

END^
```

Ejercicio 4:

```
Public class Util{
*
*
*

        Public static ObjectSet<Cliente> listar(){
                ObjectSet<Cliente> objSet= null;
                Try{
                        Query q = getDB().query();
                        q.constrain(Cliente.class);
                        q.descend("codigo").constrain(0).greater().and
                        (q.descend("codigo").constrain(101).smaller());
                        objSet = q.execute();
                }catch(Exception ex){
                        Ex.error();
                }
        Return objSet;
}
```