1. **Sobre la bd banco implementa un trigger tal que imprime un mensaje en el cliente cuando alguien intenta cambiar datos en la tabla tiene.**

-- Sobre la bd banco implementa un trigger tal que imprime un mensaje en el cliente cuando alguien intenta cambiar datos en la tabla tiene.

USE banco

GO

CREATE TRIGGER AVISO\_banco111

ON tiene

AFTER UPDATE, DELETE, INSERT

AS

BEGIN

SET NOCOUNT ON;--para que te no te aparezca filas afectadas--

RAISERROR('MODIFICACION DATOSpor datos',16,1)

PRINT 'cliente cambio datos'

END

GO

insert tiene (dni,cod\_cuenta)

values (117,15151515)

1. **Sobre la bd compras, creamos un disparador para que no permita eliminar más de un registro a la vez de la tabla Proveedores.**

--Sobre la bd compras, creamos un disparador para que no permita eliminar más de un registro a la vez de la tabla Proveedores.

use compras;

go

CREATE TRIGGER BORRAR\_REGISTROS

ON Proveedores

FOR DELETE AS --after delete tambien vale

begin

IF (SELECT COUNT(\*) FROM deleted) > 1

BEGIN

PRINT 'No permite borrar más de un registro a la vez'

ROLLBACK-- PARA NO DEJARLE EJECUTAR ESA ORDEN

end

END

GO

DELETE FROM Proveedores

WHERE emailpro IS NULL

1. **Sobre la bd Jardinería crear un *TRIGGER* que bloquee actualizar el Código del producto en el proceso de actualización.**

-- Sobre la bd Jardinería crear un TRIGGER que bloquee actualizar el Código del producto en el proceso de actualización.

use jardineria

go

CREATE TRIGGER bloquearActualizacion

ON Productos

for UPDATE AS --after delete tambien vale

begin

IF UPDATE(CodigoProducto)

BEGIN

PRINT 'No permite actualizar Código del producto en el proceso de actualización'

ROLLBACK transaction-- PARA NO DEJARLE EJECUTAR ESA ORDEN

end

END

GO

1. **Sobre la BD Compras crear un trigger tal que cuando se vaya a introducir o modificar una línea de un artículo descatalogado no lo permita y devuelva el mensaje de error:**

**'No puede seleccionar un artículo descatalogado'**

--Sobre la BD Compras crear un trigger tal que cuando se vaya a introducir o modificar una línea

--de un artículo descatalogado no lo permita y devuelva el mensaje de error:

--'No puede seleccionar un artículo descatalogado'

--si la fechad ee baja de articulos esta con una fehca esta descatalogado

use compras

go

create TRIGGER articulosDESCATALOGADOS

ON Lineas

AFTER UPDATE, INSERT AS

begin

IF EXISTS (SELECT COUNT(\*) FROM Articulos ,inserted i WHERE Articulos.codigart = i.codigart AND fecbaja IS NOT NULL)

BEGIN

declare @erromesage char

set @erromesage = 'No puede seleccionar un artículo descatalogado'

RAISERROR (@erromesage, 16,1)

ROLLBACK transaction-- PARA NO DEJARLE EJECUTAR ESA ORDEN

end

END

GO

1. **Sobre la bd de compras implementar el código necesario tal que cuando se intenten vender más de la mitad de las unidades que tengamos en stock de un artículo de una vez no se permita.**

--Sobre la bd de compras implementar el código necesario tal que cuando se intenten vender más de la mitad

--de las unidades que tengamos en stock de un artículo de una vez no se permita.

use compras

go

create TRIGGER mitadUnidades

ON Articulos

AFTER UPDATE AS

begin

IF EXISTS (SELECT \* FROM inserted i JOIN deleted d ON i.codigart = d.codigart

WHERE(d.stockart - i.stockart)>d.stockart/2 AND d.stockart-i.stockart >0)

BEGIN

SET NOCOUNT ON;--para que te no te aparezca filas afectadas--

RAISERROR('Sobrepasa la mitad de las unidades',16,1)

ROLLBACK transaction-- PARA NO DEJARLE EJECUTAR ESA ORDEN

END

END

GO

--inserted valor nuevo

---deleted valor antiguo

--- valor nuevo comparado entre el valor antiguo dividido por 2

UPDATE Articulos SET stockart = 0 WHERE codigart like '0011'

go

1. **Sobre la bd Compras, crear una tabla eventos con los campos, fecha char(6) y motivo varchar(200). Dicha tabla se rellenará cada vez que en la tabla artículos el stock de un producto alcanza el mínimo establecido.**

--Sobre la bd Compras, crear una tabla eventos con los campos, fecha char(6) y motivo varchar(200).

-- Dicha tabla se rellenará cada vez que en la tabla artículos el stock de un producto alcanza el mínimo establecido.

use compras

GO

CREATE TABLE eventos

(

fecha char(6),

motivo varchar(200)

);

GO

create TRIGGER EVENTOSminimoSTCOK

ON Articulos

AFTER UPDATE AS

begin

IF UPDATE(stockart)

BEGIN

INSERT INTO eventos (fecha,motivo )

SELECT GETDATE(),'Ha alcanzado el stock mínimo establecido', codigart FROM inserted

WHERE stockart <= stockmin

end

END

GO

1. **Sobre la bd compras, cuando se quiera borra un artículo se sustituya por actualizar la fecha de baja en la tabla artículos.**

--Sobre la bd compras, cuando se quiere borrar un artículo se sustituye por actualizar la fecha de baja en la tabla artículos.

--sI HAY ALGUIEN QUE VA AHACER UN DELETED DE UN ART PORQUE ESTA DESCATALAGODAO P, NO LO BORRA SINO QUE PONE COMO FECHA DE BAJA LA FECHA DEL DIIA

USE compras

GO

CREATE TRIGGER actualizarFECHA

ON Articulos

INSTEAD OF DELETE

AS

BEGIN

IF exists(SELECT \* FROM deleted )

BEGIN

SET NOCOUNT ON

UPDATE Articulos

SET fecbaja = GETDATE()

WHERE Articulos.codigart IN (SELECT codigart FROM deleted)

PRINT 'Borrado artículo ,actualizada la fecha de baja en la tabla artículos'

END

END

GO

DELETE FROM Articulos

WHERE descrart like '!Borrador'

1. **En la bd Jardinería trigger que avisa al usuario que se ha eliminado un registro de la tabla pagos.**

--En la bd Jardinería hacer un trigger que avisa al usuario que se ha eliminado un registro de la tabla pagos.

USE jardineria

GO

CREATE TRIGGER ELIMINADOregistro123

ON Pagos

AFTER DELETE

AS

BEGIN

--IF @@ROWCOUNT >= 1

IF (SELECT COUNT(\*) FROM deleted ) >= 1

BEGIN

PRINT 'Se ha eliminado un registro de la tabla pagos'

END

END

GO

DELETE FROM Pagos

WHERE Pagos.IDTransaccion LIKE 'ak-std-000022'

1. **Sobre la bd Banco crear un trigger que cree un registro en la tabla numrojos con los campos id\_fila autoincremental, dni\_cliente, id\_cuenta, fecha y saldo cada vez que algún cliente se quede en números rojos en alguna de sus cuentas.**

**Suponer dos situaciones: que sólo se actualice la tabla cuenta de uno en uno, o que pueda actualizarse con varios a la vez.**

--Sobre la bd Banco crear un trigger que cree un registro en la tabla numrojos con los

--campos id\_fila autoincremental, dni\_cliente, id\_cuenta, fecha y saldo cada vez que algún cliente se quede

--en números rojos en alguna de sus cuentas.

--Suponer dos situaciones: que sólo se actualice la tabla cuenta de uno en uno, o que pueda actualizarse con varios a la vez.

USE banco

GO

CREATE TABLE numrojos

(

id\_fila INT IDENTITY(1,1),

dni\_cliente INT PRIMARY KEY NOT NULL,

id\_cuenta INT NOT NULL ,

fecha DATETIME NOT NULL,

saldo INT NOT NULL,

)

GO

CREATE TRIGGER numerosROJOS

ON Cuenta

AFTER UPDATE, INSERT, DELETE AS

BEGIN

--sólo se actualice la tabla cuenta de uno en uno

IF UPDATE(saldo)

BEGIN

INSERT INTO numrojos (id\_fila,dni\_cliente, id\_cuenta,fecha,saldo )

SELECT tiene.dni, i.cod\_cuenta, GETDATE(), i.saldo FROM inserted i

JOIN tiene ON tiene.cod\_cuenta = i.cod\_cuenta and i.saldo < 0

and COUNT(tiene.dni) <= 1

PRINT 'Un cliente está en números rojos'

END

--pueda actualizarse con varios a la vez

ELSE IF UPDATE(saldo)

BEGIN

INSERT INTO numrojos (id\_fila,dni\_cliente, id\_cuenta,fecha,saldo )

SELECT tiene.dni, i.cod\_cuenta, GETDATE(), i.saldo FROM inserted i

JOIN tiene ON tiene.cod\_cuenta = i.cod\_cuenta and i.saldo < 0

and COUNT(tiene.dni) > 1

PRINT 'Varios clientes está en números rojos'

END

END

GO

alter trigger numerosRojos on cuenta

for update as

declare @id\_cliente int

if exists (select \* from inserted i where i.saldo > 0)

begin

set @id\_cliente = (select dni from tiene join inserted on

tiene.cod\_cuenta = inserted.cod\_cuenta where inserted.saldo > 0)

insert into numrojos values (getdate(), @id\_cliente, (select cod\_cuenta from inserted i where i.saldo > 0), (select saldo from inserted i where i.saldo > 0))

end

--Sobre la bd Banco crear un trigger que cree un registro en la tabla numrojos con los

--campos id\_fila autoincremental, dni\_cliente, id\_cuenta, fecha y saldo cada vez que algún cliente se quede

--en números rojos en alguna de sus cuentas.

--Suponer dos situaciones: que sólo se actualice la tabla cuenta de uno en uno, o que pueda actualizarse con varios a la vez.

USE banco

GO

CREATE TABLE numrojos

(

id\_fila INT IDENTITY(1,1),

dni\_cliente INT PRIMARY KEY NOT NULL,

id\_cuenta INT NOT NULL ,

fecha DATETIME NOT NULL,

saldo INT NOT NULL,

)

GO

CREATE TRIGGER numerosROJOS

ON Cuenta

AFTER UPDATE AS

DECLARE @id\_cliente INT, @saldo INT

SET @id\_cliente = 7

SET @saldo = -525252

BEGIN

--sólo se actualice la tabla cuenta de uno en uno

IF UPDATE(saldo)

BEGIN

INSERT INTO numrojos (dni\_cliente, id\_cuenta,fecha,saldo )

SELECT movimiento.dni, i.cod\_cuenta, GETDATE(), i.saldo FROM inserted i

JOIN movimiento ON i.cod\_cuenta = @id\_cliente and @saldo < 0

PRINT 'Un cliente está en números rojos'

END

END

GO

update cuenta

set saldo = -525252

where cod\_cuenta = 7

select \* from numrojos

1. **Hacer lo necesario para que cada vez que un cliente de la bd banco ingrese más de 1000 euros se le bonifique con 100, sólo para clientes con cuentas que superen los tres años de antigüedad entre el 1 de enero del 2015 y el 31 de Julio del 2015. Hacerlo sobre la tabla movimiento.**

**Para resolverlo creo un trigger y una función( para calcular la antigüedad de la cuenta). Suponer que se produce sólo un movimiento cada vez.**

--Hacer lo necesario para que cada vez que un cliente de la bd banco ingrese más de 1000 euros se le bonifique con 100,

-- sólo para clientes con cuentas que superen los tres años de antigüedad entre el 1 de enero del 2015 y el 31 de Julio del 2015.

-- Hacerlo sobre la tabla movimiento.

--Para resolverlo creo un trigger y una función( para calcular la antigüedad de la cuenta).

--Suponer que se produce sólo un movimiento cada vez.

USE banco

GO

CREATE FUNCTION dbo.ANTIGUEDAD()

RETURNS INT

AS

BEGIN

RETURN (SELECT DATEDIFF(YEAR,fecha\_creacion,GETDATE()) FROM cuenta)

END

GO

CREATE TRIGGER ingresar1000

ON movimiento

AFTER INSERT AS

BEGIN

DECLARE @FECHAINICIAL DATE = '01-01-2015';

DECLARE @FECHAFINAL DATE = '31-07-2015';

DECLARE @cantidad decimal(4, 0) = 1000;

DECLARE @bonificacion decimal(4, 0) = 100;

--Un cliente ingresa más de 1000 euros

IF (SELECT COUNT(\*) from inserted) = 1

BEGIN

SET NOCOUNT ON

INSERT INTO movimiento (fecha, cantidad, dni, cod\_cuenta, id\_movimiento)

SELECT GETDATE() , i.cantidad, i.dni , i.cod\_cuenta, i.id\_movimiento FROM inserted i

JOIN Cuenta ON Cuenta.cod\_cuenta = i.cod\_cuenta AND i.cantidad > @cantidad

PRINT 'Un cliente ha hecho un movimiento de más de 1000 euros'

END

----Un cliente ingresa más de 1000 euros y se le bonifica con 100 euros

IF (SELECT COUNT(\*) from inserted) = 1

BEGIN

SET NOCOUNT ON

INSERT INTO movimiento (fecha, cantidad, dni, cod\_cuenta, id\_movimiento)

SELECT GETDATE() , @bonificacion, i.dni , i.cod\_cuenta, i.id\_movimiento FROM inserted i

JOIN Cuenta ON Cuenta.cod\_cuenta = i.cod\_cuenta AND i.cantidad > @cantidad

AND dbo.ANTIGUEDAD() > 3

AND i.fecha BETWEEN @FECHAINICIAL AND @FECHAFINAL;

PRINT 'Un cliente ha hecho un movimiento de más de 1000 euros y se le ha bonificado

con más de 100 euros'

END

END

GO

INSERT INTO movimiento (fecha, cantidad, dni, cod\_cuenta, id\_movimiento)

VALUES (GETDATE(), 1002, 117, 9, 8)

GO

1. **Crear un desencadenador para la tabla productos en la bd Jardinería, que se active cada vez que se inserte un registro o se actualice la columna precio, la condición para aceptar al inserción o la actualización es que el precio costo no debe ser mayor que el precio venta.**

--Crear un desencadenador para la tabla productos en la bd Jardinería, que se active cada vez que

-- se inserte un registro o se actualice la columna precio, la condición para aceptar al inserción o la

-- actualización es que el precio costo no debe ser mayor que el precio venta.

USE jardineria

GO

CREATE TRIGGER DESENCADENADORproductos

ON Productos

FOR UPDATE, INSERT AS

BEGIN

--CADA VEZ QUE SE actualize LA COLUMNA PRECIO PERO PRECIO NO CORRECTO

IF (SELECT COUNT(\*) FROM inserted i, deleted d WHERE i.PrecioProveedor > d.PrecioVenta)

BEGIN

ROLLBACK

PRINT 'No se puede actualizar el precio,el precio costo no debe ser mayor que el precio venta '

END

--CADA VEZ QUE SE actualize LA COLUMNA PRECIO

ELSE IF (SELECT COUNT(\*) FROM inserted i, deleted d WHERE i.PrecioProveedor > d.PrecioVenta)

BEGIN

SET NOCOUNT ON;--para que te no te aparezca filas afectadas--

PRINT 'Actualizado precio'

END

--CADA VEZ QUE SE INSERTE UN REGISTRO

ELSE IF UPDATE(CodigoProducto)

BEGIN

SET NOCOUNT ON;--para que te no te aparezca filas afectadas--

INSERT INTO Productos (CodigoProducto, Nombre, Gama,Dimensiones,Proveedor

,Descripcion,CantidadEnStock,PrecioVenta,PrecioProveedor)

SELECT \* FROM inserted

PRINT 'Insertado registro'

END

END

GO

1. **Crear trigger asociado a la tabla Pedidos, de manera que cuando queramos borrar un registro de la tabla pedidos elimine todos los registros relacionados de la tabla de detallepedidos. Hay que tener en cuenta que puedo borrar varios pedidos a la vez.**

--Crear trigger asociado a la tabla Pedidos, de manera que cuando queramos borrar un registro de la tabla pedidos

--elimine todos los registros relacionados de la tabla de detallepedidos. Hay que tener en cuenta que puedo borrar

--varios pedidos a la vez.

USE jardineria

GO

CREATE TRIGGER borrarPEDIDOS123

ON Pedidos

AFTER DELETE AS

BEGIN

--BORRAR UN PEDIDO

IF (SELECT COUNT(\*) from deleted) >= 1

BEGIN

SET NOCOUNT ON

DELETE FROM DetallePedidos

WHERE DetallePedidos.CodigoPedido IN (SELECT DISTINCT CodigoPedido FROM deleted)

PRINT 'Borrado un pedido'

END

END

GO

DELETE FROM Pedidos

WHERE CodigoPedido = 128

10

USE banco

GO

CREATE FUNCTION dbo.ANTIGUEDADprueba1(@NumeroCuenta INT)

RETURNS INT

AS

BEGIN

RETURN (SELECT DATEDIFF(YEAR,fecha\_creacion,GETDATE()) FROM cuenta WHERE cod\_cuenta = @NumeroCuenta)

END

GO

SELECT dbo.ANTIGUEDADprueba1(5)

--Hacer lo necesario para que cada vez que un cliente de la bd banco ingrese más de 1000 euros se le bonifique con 100,

-- sólo para clientes con cuentas que superen los tres años de antigüedad entre el 1 de enero del 2015 y el 31 de Julio del 2015.

-- Hacerlo sobre la tabla movimiento.

--Para resolverlo creo un trigger y una función( para calcular la antigüedad de la cuenta).

--Suponer que se produce sólo un movimiento cada vez.

USE banco

GO

CREATE FUNCTION dbo.ANTIGUEDAD()

RETURNS INT

AS

BEGIN

RETURN (SELECT DATEDIFF(YEAR,fecha\_creacion,GETDATE()) FROM cuenta)

END

GO

CREATE TRIGGER ingresar1000bonificacion

ON movimiento

AFTER INSERT AS

BEGIN

DECLARE @FECHAINICIAL DATE = '01-01-2015';

DECLARE @FECHAFINAL DATE = '31-07-2015';

--DECLARE @cantidad decimal(4, 0) = 1000;

DECLARE @bonificacion decimal(4, 0) = 100;

--Un cliente ingresa más de 1000 euros

--IF (SELECT COUNT(\*) from inserted) = 1

-- BEGIN

-- SET NOCOUNT ON

-- INSERT INTO movimiento (fecha, cantidad, dni, cod\_cuenta, id\_movimiento)

-- SELECT GETDATE() , i.cantidad, i.dni , i.cod\_cuenta, i.id\_movimiento FROM inserted i

-- JOIN Cuenta ON Cuenta.cod\_cuenta = i.cod\_cuenta AND i.cantidad > @cantidad

-- PRINT 'Un cliente ha hecho un movimiento de más de 1000 euros'

-- END

----Un cliente ingresa más de 1000 euros y se le bonifica con 100 euros

IF (SELECT COUNT(\*) from inserted) = 1

BEGIN

SET NOCOUNT ON

INSERT INTO cuenta (fecha, cantidad, dni, cod\_cuenta, id\_movimiento)

SELECT GETDATE() , @bonificacion, i.dni , i.cod\_cuenta, i.id\_movimiento FROM inserted i

JOIN Cuenta ON Cuenta.cod\_cuenta = i.cod\_cuenta AND i.cantidad > @cantidad

AND dbo.ANTIGUEDAD() > 3

AND i.fecha BETWEEN @FECHAINICIAL AND @FECHAFINAL;

PRINT 'Un cliente ha hecho un movimiento de más de 1000 euros y se le ha bonificado

con más de 100 euros'

END

END

GO

INSERT INTO movimiento (fecha, cantidad, dni, cod\_cuenta, id\_movimiento)

VALUES (GETDATE(), 1005, 117,2, 8)