2 -A1

// skapar en lagrad procedur som lägger till ny artikel

CREATE PROCEDURE ArticleAdd

@Artnamn varchar(30),

@Antal int,

@Pris decimal(6,2),

@Rabatt decimal(2,2),

@Plats char(10)

AS

BEGIN

BEGIN TRY

INSERT INTO Artikel (Artnamn, Antal, Pris, Rabatt, Plats)

VALUES (@Artnamn, @Antal, @Pris, @Rabatt, @Plats)

END TRY

BEGIN CATCH

RAISERROR('Fel inträffade!, artikel har inte lagts till', 16, 1)

END CATCH

END

EXEC ArticleAdd 'Ipad 3D', 30, 3000.50, 0.00, 'Förråd 10'

SELECT \* FROM Artikel

2 -A2

// skapar en lagrad procedur som hittar och ändrar artikel i tabellen

CREATE PROCEDURE ArticleUpdate

@ArtikelID int = 0,

@Artnamn varchar(30),

@Antal int,

@Pris decimal(6,2),

@Rabatt decimal(6,2),

@Plats char(10)

AS

BEGIN

IF EXISTS(SELECT ArtikelID FROM Artikel WHERE ArtikelID = @ArtikelID)

BEGIN

BEGIN TRY

UPDATE Artikel

SET Artnamn = @Artnamn,

Antal = @Antal,

Pris = @Pris,

Rabatt = @Rabatt,

Plats = @Plats

WHERE Artikel.ArtikelID = @ArtikelID

END TRY

BEGIN CATCH

RAISERROR('Fel inträffade!', 16, 1)

END CATCH

END

ELSE

BEGIN

RAISERROR('Valda ArtikelID finns inte!, 16, 1)

END

END

EXEC ArticleUpdate 9, 'iphone 8', 20, 5000.80, 0.00, 'Förråd 05'

SELECT \* FROM Artikel

2 -A3

// skapar en lagrad procedur som tar bort artikel

CREATE PROCEDURE ArticleDelete

@ArtikelID int = 0

AS

BEGIN

IF EXISTS(SELECT ArtikelID FROM Artikel WHERE ArtikelID = @ArtikelID)

BEGIN

DELETE Artikel

WHERE Artikel.ArtikelID = @ArtikelID

END

ELSE

BEGIN

RAISERROR('Valda ArtikelID finns inte! gårt int att ta bort', 16, 1)

END

END

EXEC ArticleDelete 8

SELECT \* FROM Artikel

2 - B

// skapar en lagrad procedur som visar alla artiklar i en lista och där möjlighet att bar välja ut en artikel finns

CREATE PROCEDURE ArticleList

@ArtikelID int = 0

AS

BEGIN

IF EXISTS(SELECT ArtikelID FROM Artikel WHERE ArtikelID = @ArtikelID)

BEGIN

SELECT ArtikelID, Artnamn, Antal, Pris, (Antal\*Pris) AS Totalpris

FROM Artikel

WHERE ArtikelID = @ArtikelID

END

ELSE

BEGIN

IF (@ArtikelID = 0)

SELECT ArtikelID, Artnamn, Antal, Pris, (Antal\*Pris) AS Totalpris

FROM Artikel

ORDER BY Artnamn ASC

ELSE

RAISERROR('Valda ArtikelID finns inte', 16, 1)

END

END

EXEC ArticleList

SELECT \* FROM Artikel

2 - C

// skapar en lagrad procedur som skapar en temporär tabell och fyller den med kund och kontakt information, man kan välja enskild kund också eller alla kunder

CREATE PROCEDURE TelefonList

@KundID int = 0

AS

BEGIN

CREATE TABLE #Temp

(

Namn varchar(40) NOT NULL,

Ort varchar(25) NOT NULL,

Telefon varchar(15),

Telefontyp varchar(10)

)

IF EXISTS(SELECT KundID FROM Kund WHERE KundID = @KundID)

BEGIN

INSERT INTO #Temp (Namn, Ort, Telefon, Telefontyp)

SELECT k.Namn, k.Ort, t.Telenr, typ.Teltyp

FROM Kund AS K left JOIN Telefon AS t

ON k.KundID=t.KundID left JOIN Teltyp as typ

ON t.TeltypID=typ.TelTypID

WHERE k.KundID = @KundID

ORDER BY typ.Teltyp ASC

END

ELSE

BEGIN

IF (@KundID = 0)

INSERT INTO #Temp (Namn, Ort, Telefon, Telefontyp)

SELECT k.Namn, k.Ort, t.Telenr, typ.Teltyp

FROM Kund AS k LEFT JOIN Telefon AS t

ON k.KundID=t.KundID LEFT JOIN Teltyp as typ

ON t.TeltypID=typ.TelTypID

ORDER BY k.Namn, typ.Teltyp ASC

ELSE

RAISERROR('Valda KundID finns inte', 16, 1)

END

SELECT \* FROM #Temp

DROP TABLE #Temp

END

EXEC TelefonList

2 - D

// skapar en lagrad procedur för statistik sammanställning av försäljningen under önskad period

CREATE PROCEDURE SalesList

@StartDate Date,

@EndDate Date

AS

BEGIN

IF (@StartDate > @EndDate)

RAISERROR('Startdatumet måste vara innan slutdatumet.', 16, 1)

ELSE

SELECT Fr.ArtikelID, Artnamn, Datum, Fr.Antal, Fr.Pris, TotalPris

FROM Faktura AS F

INNER JOIN Fakturad AS Fr ON F.FakturaID = Fr.FakturaID

INNER JOIN Artikel AS A ON Fr.ArtikelID = A.ArtikelID

WHERE Datum BETWEEN @StartDate AND @EndDate

ORDER BY Artnamn ASC

END

EXEC SalesList '2015-03-01', '2015-02-28'

3 - A

// skapar en lagrad procedur som läger upp faktura och uppdaterar artikel antal

CREATE PROCEDURE Fakturera

@FakturaID int = 0,

@ArtikelID int,

@Antal int,

@Rabatt decimal(2,2),

@MomsID int

AS

BEGIN

IF EXISTS(SELECT ArtikelID FROM Artikel WHERE ArtikelID = @ArtikelID)

BEGIN

BEGIN TRY

DECLARE

@Pris decimal(6,2),

@Error varchar(40)

SET @Pris = (SELECT Pris FROM Artikel WHERE ArtikelID = @ArtikelID);

BEGIN TRAN

SET @Error = 'Det gick inte att skapa en ny faktura!'

INSERT INTO Fakturad (FakturaID, ArtikelID, Antal, Pris, Rabatt, MomsID)

VALUES (@FakturaID, @ArtikelID, @Antal, @Pris, @Rabatt, @MomsID);

SET @Error = 'Uppdatering av artikel fungerar inte!'

UPDATE Artikel SET Antal = Antal - @Antal

WHERE ArtikelID = @ArtikelID;

COMMIT TRAN

END TRY

BEGIN CATCH

ROLLBACK TRAN

RAISERROR(@Error, 16, 1)

END CATCH

END

ELSE

BEGIN

RAISERROR('Artikeln finns inte!!!', 16, 1)

END

END

EXEC Fakturera 4, 3, 10, 0.00, 4

3 - B

// skapar 2 lagrad procedur, AddTelefon användas av Addkund för att lägga till telefonnummer och kontakt

CREATE PROCEDURE AddTelefon

@KundID int = @@IDENTITY,

@Telnr varchar(15),

@TeltypID int

AS

BEGIN

BEGIN TRY

INSERT INTO Telefon (KundID, Telenr, TeltypID)

VALUES (@KundID, @Telnr, @TeltypID)

END TRY

BEGIN CATCH

RAISERROR('Fel inträffade med telefonnummer/telefontyp', 16, 1)

END CATCH

END

CREATE PROCEDURE Addkund

@Namn varchar(40),

@Adress varchar(30),

@Postnr varchar(6),

@Ort varchar(25),

@Rabatt decimal(2,2),

@KategoriID int,

@Telnr varchar(15),

@TeltypID int

AS

BEGIN

BEGIN TRY

DECLARE @Error varchar(40)

BEGIN TRAN

SET @Error = 'kunde inte lägga till Kunden'

INSERT INTO Kund (Namn, Adress, Postnr, Ort, Rabatt, KategoriID)

VALUES (@Namn, @Adress, @Postnr, @Ort, @Rabatt, @KategoriID);

SET @Error = 'kunde inte läggas till Telefonen'

EXEC AddTelefon @@IDENTITY, @Telnr, @TeltypID

COMMIT TRAN

END TRY

BEGIN CATCH

ROLLBACK TRAN

RAISERROR(@Error, 16, 1)

END CATCH

END

EXEC Addkund 'Marco Villegas', 'Terapivägen', '14156', 'Huddinge', 0.00, 1, '0706880589', 1

3 - C

// skapar 2 lagrad procedur, DeleteFakturarad användas av DeleteFaktura för att ta bort rad och återställa artikel mängd

CREATE PROCEDURE DeleteFakturarad

@FakturaID int,

@ArtikelID int

AS

BEGIN

BEGIN TRY

DECLARE @Error varchar(40)

BEGIN TRAN

SET @Error = 'Kunde inte återställa mängd'

UPDATE Artikel

SET Antal = Antal + (SELECT SUM(Antal)

FROM Fakturad

WHERE FakturaID = @FakturaID

AND ArtikelID = @ArtikelID)

WHERE ArtikelID = @ArtikelID;

SET @Error = 'Kunde inte radera fakturaraden'

DELETE FROM Fakturad

WHERE FakturaID = @FakturaID AND ArtikelID = @ArtikelID;

COMMIT TRAN

END TRY

BEGIN CATCH

ROLLBACK TRAN

RAISERROR(@Error, 16, 1)

RETURN (1)

END CATCH

END

CREATE PROCEDURE DeleteFaktura

@FakturaID int

AS

BEGIN

BEGIN TRY

DECLARE @Error varchar(40)

DECLARE fakturaradCursor CURSOR

FOR SELECT ArtikelID

FROM Fakturad

WHERE FakturaID = @FakturaID

FOR UPDATE;

OPEN fakturaradCursor

DECLARE @ArtikelID int

FETCH fakturaradCursor INTO @ArtikelID

WHILE @@FETCH\_STATUS = 0

BEGIN

EXEC DeleteFakturarad @FakturaID, @ArtikelID

FETCH fakturaradCursor INTO @ArtikelID

END

CLOSE fakturaradCursor

DEALLOCATE fakturaradCursor

SET @Error = 'Kunde inte radera fakturan.'

BEGIN TRAN

DELETE FROM Faktura

WHERE FakturaID = @FakturaID;

COMMIT TRAN

END TRY

BEGIN CATCH

ROLLBACK TRAN

RAISERROR(@Error, 16, 1)

RETURN (1)

END CATCH

END

EXEC DeleteFaktura 42

4 - A

// skapar tabell

CREATE TABLE Logging

(

LoggingID int NOT NULL IDENTITY(1,1) PRIMARY KEY,

tbl varchar(25) NOT NULL,

tblID int NOT NULL,

kol varchar(25) NOT NULL,

Old varchar(100) NOT NULL,

New varchar(100) NOT NULL,

Datum date DEFAULT GetDate(),

Usr varchar(10) NOT NULL

);

4 - B

// skapar trigger kopplad till artikel tabellen, vissar när ändringar händer i logging tabellen

CREATE TRIGGER Artikeltrigger ON Artikel

AFTER UPDATE

AS

BEGIN

IF UPDATE (Artnamn)

BEGIN

INSERT INTO Logging (tblID, tbl, kol, Old, New, Datum, Usr)

SELECT

i.ArtikelID,

'Artikel',

'Artnamn',

d.Artnamn,

i.Artnamn,

GETDATE(),

SYSTEM\_USER

FROM inserted as i INNER JOIN deleted as d

ON i.ArtikelID = d.ArtikelID;

END

END