CREATE DATABASE ShoppingCart

GO

/\* CREATE TABLES \*/

CREATE TABLE Customers

(

CustomerID INT NOT NULL PRIMARY KEY Identity (1,1),

FirstName VARCHAR(30) NOT NULL,

LastName VARCHAR(30) NOT NULL,

Telephone VARCHAR(15) NULL DEFAULT ‘(xxx) xxx-xxxx’,

Email NVARCHAR(255) NOT NULL

)

CREATE TABLE Login

(

LoginID INT NOT NULL PRIMARY KEY Identity (1,1),

CustomerID INT REFERENCES Customers(CustomerID) NOT NULL,

Username VARCHAR(50) NOT NULL DEFAULT ‘user’,

Pword VARCHAR(50) NOT NULL

)

CREATE TABLE Orders

(

OrderID INT NOT NULL PRIMARY KEY Identity (1,1),

CustomerID INT REFERENCES Customers(CustomerID) NOT NULL,

PaymentType INT NOT NULL DEFAULT 1,

Date DATETIME NOT NULL

)

CREATE TABLE Products

(

ProductID INT NOT NULL PRIMARY KEY Identity (1,1),

Name VARCHAR(100) NOT NULL,

Description TEXT NOT NULL,

Price FLOAT NOT NULL DEFAULT 0.00

)

CREATE TABLE OrderItems

(

OrderItemsID INT NOT NULL PRIMARY KEY Identity (1,1),

OrderID INT REFERENCES Orders(OrderID) NOT NULL,

ProductID INT REFERENCES Products(ProductID) NOT NULL,

Quantity INT NOT NULL DEFAULT 0

)

/\* ENSURING REFERENTIAL INTEGRITY \*/

USE MyShoppingCart;

IF OBJECT\_ID('LoginCopy') IS NOT NULL

DROP TABLE LoginCopy;

IF OBJECT\_ID('CustomersCopy') IS NOT NULL

DROP TABLE CustomersCopy;

SELECT \* INTO LoginCopy from Login;

SELECT \* INTO CustomersCopy from Customers;

GO

CREATE TRIGGER trg\_Login\_UPDATE\_DELETE

ON LoginCopy

AFTER DELETE

AS

IF EXISTS (Select \* FROM Deleted JOIN CustomersCopy

ON Deleted.CustomerID = CustomersCopy.CustomerID)

BEGIN;

THROW 50002, 'CustomerID in use.', 1;

ROLLBACK TRAN;

END;

/\* STORED PROCS \*/

CREATE PROCEDURE sp\_Customers\_InsertUpdateDelete

(

@customer\_id INTEGER,

@first\_name VARCHAR(30),

@last\_name VARCHAR(30),

@telephone VARCHAR(15),

@email NVARCHAR(255),

@statement\_type NVARCHAR(20) = ''

)

AS

BEGIN

IF @statement\_type = 'Insert'

BEGIN

INSERT INTO Customers

(CustomerID, FirstName, LastName, Telephone, Email)

VALUES( @customer\_id, @first\_name, @last\_name, @telephone, @email)

END

IF @statement\_type = 'Update'

BEGIN

UPDATE Customers SET

FirstName = @first\_name,

LastName = @last\_name,

Telephone = @telephone,

Email = @email

WHERE CustomerID = @customer\_id

END

ELSE IF @statement\_type = 'Delete'

BEGIN

DELETE FROM Customers WHERE CustomerID = @customer\_id

END

END

GO

CREATE PROCEDURE sp\_Login\_InsertUpdateDelete

(

@login\_id INTEGER,

@customer\_id INTEGER,

@username VARCHAR(50),

@password VARCHAR(50),

@statement\_type NVARCHAR(20)

)

AS

BEGIN

IF @statement\_type = 'Insert'

BEGIN

INSERT INTO Login

(LoginID, CustomerID, Username, Pword)

VALUES( @login\_id, @customer\_id, @username, @password)

END

IF @statement\_type = 'Update'

BEGIN

UPDATE Login SET

Username = @username,

Pword = @password

WHERE LoginID = @login\_id

END

ELSE IF @statement\_type = 'Delete'

BEGIN

DELETE FROM Login WHERE LoginID = @login\_id

END

END

GO

CREATE PROCEDURE sp\_Orders\_InsertUpdateDelete

(

@order\_id INTEGER,

@customer\_id INTEGER,

@payment\_type INTEGER,

@date DATETIME,

@status INTEGER,

@statement\_type NVARCHAR(20) = ''

)

AS

BEGIN

IF @statement\_type = 'Insert'

BEGIN

INSERT INTO Orders

(OrderID, CustomerID, PaymentType, Date, Status)

VALUES( @order\_id, @customer\_id, @payment\_type, @date, @status)

END

IF @statement\_type = 'Update'

BEGIN

UPDATE Orders SET

PaymentType = @payment\_type,

Date = @date,

Status = @status

WHERE CustomerID = @customer\_id

END

ELSE IF @statement\_type = 'Delete'

BEGIN

DELETE FROM Orders WHERE OrderID = @order\_id

END

END

GO

CREATE PROCEDURE sp\_Products\_InsertUpdateDelete

(

@product\_id INTEGER,

@name VARCHAR(100),

@description TEXT,

@price FLOAT,

@statement\_type NVARCHAR(20) = ''

)

AS

BEGIN

IF @statement\_type = 'Insert'

BEGIN

INSERT INTO Products

(ProductID, Name, Description, Price)

VALUES( @product\_id, @name, @description, @price)

END

IF @statement\_type = 'Update'

BEGIN

UPDATE Products SET

Name = @name,

Description = @description,

Price = @price

WHERE ProductID = @product\_id

END

ELSE IF @statement\_type = 'Delete'

BEGIN

DELETE FROM Products WHERE ProductID = @product\_id

END

END

GO

CREATE PROCEDURE sp\_OrderItems\_InsertUpdateDelete

(

@orderitems\_id INTEGER,

@order\_id INTEGER,

@product\_id INTEGER,

@quantity INTEGER,

@statement\_type NVARCHAR(20) = ''

)

AS

BEGIN

IF @statement\_type = 'Insert'

BEGIN

INSERT INTO OrderItems

(OrderItemsID, OrderID, ProductID, Quantity)

VALUES( @orderitems\_id, @order\_id, @product\_id, @quantity)

END

IF @statement\_type = 'Update'

BEGIN

UPDATE OrderItems SET

Quantity = @quantity

WHERE OrderItemsID = @orderitems\_id

END

ELSE IF @statement\_type = 'Delete'

BEGIN

DELETE FROM OrderItems WHERE OrderItemsID = @orderitems\_id

END

END

GO

/\* Adding columns for trigger for modifying \*/

ALTER TABLE Customers

ADD LastModifiedBy VARCHAR(30), DateTimeModified DATETIME

ALTER TABLE Login

ADD LastModifiedBy VARCHAR(30), DateTimeModified DATETIME

ALTER TABLE Orders

ADD LastModifiedBy VARCHAR(30), DateTimeModified DATETIME

ALTER TABLE Products

ADD LastModifiedBy VARCHAR(30), DateTimeModified DATETIME

ALTER TABLE OrderItems

ADD LastModifiedBy VARCHAR(30), DateTimeModified DATETIME

/\* Triggers for Modifying with date time columns \*/

CREATE TRIGGER trg\_CustomersModify

ON Customers

AFTER INSERT, UPDATE

AS

DECLARE @user varchar(30), @dt datetime = GETDATE()

BEGIN

UPDATE Customers

SET LastModifiedBy = @User,

DateTimeModified = @dt

WHERE CustomerID IN (Select CustomerID FROM Inserted)

END

GO

CREATE TRIGGER trg\_LoginModify

ON Login

AFTER INSERT, UPDATE

AS

DECLARE @user varchar(30), @dt datetime = GETDATE()

BEGIN

UPDATE Login

SET LastModifiedBy = @User,

DateTimeModified = @dt

WHERE LoginID IN (Select LoginID FROM Inserted)

END

GO

CREATE TRIGGER trg\_OrdersModify

ON Orders

AFTER INSERT, UPDATE

AS

DECLARE @user varchar(30), @dt datetime = GETDATE()

BEGIN

UPDATE Orders

SET LastModifiedBy = @User,

DateTimeModified = @dt

WHERE OrderID IN (Select OrderID FROM Inserted)

END

GO

CREATE TRIGGER trg\_ProductsModify

ON Products

AFTER INSERT, UPDATE

AS

DECLARE @user varchar(30), @dt datetime = GETDATE()

BEGIN

UPDATE Products

SET LastModifiedBy = @User,

DateTimeModified = @dt

WHERE ProductID IN (Select ProductID FROM Inserted)

END

GO

CREATE TRIGGER trg\_OrderItemsModify

ON OrderItems

AFTER INSERT, UPDATE

AS

DECLARE @user varchar(30), @dt datetime = GETDATE()

BEGIN

UPDATE OrderItems

SET LastModifiedBy = @User,

DateTimeModified = @dt

WHERE OrderItemsID IN (Select OrderItemsID FROM Inserted)

END

GO

/\* Indexes \*/

CREATE INDEX IX\_Username

ON Login (Username)

CREATE INDEX IX\_Password

ON Login (Pword)

CREATE INDEX IX\_CustomersName

ON Customers (LastName)

CREATE INDEX IX\_OrderDate

ON Orders (Date DESC)

CREATE INDEX IX\_Quantity

ON OrderItems (Quantity)

CREATE INDEX IX\_Price

ON Products (Price)

/\* Function for Average Price above $100 \*/

CREATE FUNCTION fn\_AvgPrice1

(@price\_threshold money = 100.00)

RETURNS table

RETURN

(SELECT ProductID, Name, AVG(Price) AS AveragePrice

FROM Products

WHERE Price > @price\_threshold

GROUP BY Name, ProductID

)