

/\*

Write the code to CREATE all tables

* Emphasis on surrogate PKs with auto-increment (done with IDENTITY() keyword
* Recognize there are three methods to code FOREIGN KEY constraint

\*/

CREATE DATABASE OTTER\_gthay

GO

-- Be sure to change the context of the query window to the new database!!!

USE OTTER\_gthay

GO

CREATE TABLE tblCUSTOMER\_TYPE (

CustomerTypeID INT IDENTITY(1,1) primary key not null,

CustomerTypeName varchar(50) not null,

CustomerTypeDescr varchar(500) NOT NULL)

GO

CREATE TABLE tblCUSTOMER

(CustomerID INT IDENTITY(1,1) primary key not null,

CustFname varchar(25) not null,

CustLname varchar(25) not null,

CustBirthDate Date NOT NULL,

CustomerTypeID INT FOREIGN KEY REFERENCES tblCUSTOMER\_TYPE (CustomerTypeID) NOT NULL)

GO

CREATE TABLE tblORDER

(OrderID INT IDENTITY(1,1) primary key not null,

OrderDate Date DEFAULT GetDate() NOT NULL, -- Be aware this is a default value

CustomerID INT FOREIGN KEY REFERENCES tblCUSTOMER (CustomerID) not null)

GO

CREATE TABLE tblPRODUCT

(ProductID INT IDENTITY(1,1) primary key not null,

ProductName varchar(100) not null,

ProductPrice numeric(10,2) not null,

ProductDescr varchar(500) NULL)

GO

CREATE TABLE tblORDER\_PRODUCT

(OrderProdID INT IDENTITY(1,1) primary key not null,

OrderID INT FOREIGN KEY REFERENCES tblORDER (OrderID) not null,

ProductID INT NOT NULL,

Quantity INT not null)

GO

CREATE TABLE tblPRODUCT\_TYPE

(ProductTypeID INT IDENTITY(1,1) NOT NULL,

ProductTypeName varchar(100) not null,

ProductTypeDescr varchar(500) NULL,

PRIMARY KEY (ProductTypeID)) -- Be aware this is a second method to create a PK

GO

-- Be aware the following is a method to add a new column AND make it a FK

ALTER TABLE tblPRODUCT

ADD ProductTypeID INT

FOREIGN KEY REFERENCES tblPRODUCT\_TYPE (ProductTypeID)

GO

-- Be aware the following is a method to make an FK to an existing column

ALTER TABLE tblORDER\_PRODUCT

ADD CONSTRAINT FK\_tblORDER\_PRODUCT\_ProductID

FOREIGN KEY (ProductID) REFERENCES tblPRODUCT (ProductID)

-- Adding a set of values into the first tables (these are look-up tables)

INSERT INTO tblCUSTOMER\_TYPE ([CustomerTypeName],[CustomerTypeDescr])

VALUES ('Retail', 'Anyone older than 45 who is afraid of the internet'),

('Online', 'Most everybody with a cell phone')

GO

-- The following code uses a SELECT statement to quickly capture 2.5 million rows of data from another database

INSERT INTO tblCUSTOMER (CustFname, CustLname, CustBirthDate, CustomerTypeID)

SELECT Top 2500000 [CustomerFname],[CustomerLname],[DateOfBirth],

(SELECT CustomerTypeID FROM tblCUSTOMER\_TYPE WHERE CustomerTypeName = 'Online')

FROM Super\_Store.dbo.tblCUSTOMER

GO

-- The following code is a simple INSERT statement into a look-up table

INSERT INTO tblPRODUCT\_TYPE (ProductTypeName, ProductTypeDescr)

VALUES ('Accessory', 'Any product not considered a snowboard'), ('Snowboard', 'Any product made of fiberglass or carbon and has boot straps')

GO

-- The following code is an example of a nested stored procedure

CREATE PROCEDURE uspGetProdTypeID

@P\_Type\_Name2 varchar(100),

@PT\_ID2 INT OUTPUT

AS

SET @PT\_ID2 = (SELECT ProductTypeID FROM tblPRODUCT\_TYPE WHERE ProductTypeName = @P\_Type\_Name2)

-- The following code is a stored procedure to populate a table that has frequent INSERT, UPDATE or DELETE activity (we call them transactional tables). This stored procedure calls a nested stored procedure leveraging an OUTPUT parameter

CREATE PROCEDURE uspADDProduct

@ProdName varchar(100),

@Price Numeric(10,2),

@P\_Type\_Name varchar(100),

@ProdDescr varchar(500)

AS

DECLARE @PT\_ID INT

-- following is the execution of the nested stored procedure

EXEC uspGetProdTypeID @P\_Type\_Name2 = @P\_Type\_Name, @PT\_ID2 = @PT\_ID OUTPUT

-- following is the first example of error-handling

IF @PT\_ID IS NULL

BEGIN

PRINT 'Hey...your lookup value for ProductTypeID has no value'

RAISERROR ('@PT\_ID cannot be NULL; session terminating', 11,1)

RETURN

END

BEGIN TRAN

INSERT INTO tblPRODUCT (ProductName, ProductPrice, ProductTypeID, ProductDescr)

VALUES (@ProdName, @Price, @PT\_ID, @ProdDescr)

IF @@ERROR <> 0

ROLLBACK TRAN

ELSE

COMMIT TRAN

GO