/\*

Create the database and run script to create all tables.

\*/

USE MASTER

GO

CREATE DATABASE EXAMPLE\_CART2

GO

USE EXAMPLE\_CART2

GO

CREATE TABLE [dbo].[tblCUSTOMER](

[CustID] [int] IDENTITY(1,1) PRIMARY KEY NOT NULL,

[CustomerFname] [nvarchar](35) NOT NULL,

[CustomerLname] [nvarchar](35) NOT NULL,

[CustomerAddress] [nvarchar](100) NOT NULL,

[CustomerCity] [nvarchar](75) NOT NULL,

[CustomerCounty] [nvarchar](75) NOT NULL,

[CustomerState] [nvarchar](30) NOT NULL,

[CustomerZIP] [nvarchar](12) NOT NULL,

[AreaCode] [nvarchar](3) NOT NULL,

[Email] [nvarchar](100) NOT NULL,

[BusinessName] [nvarchar](100) NOT NULL,

[DateOfBirth] [date] NOT NULL,

[PhoneNum] [nvarchar](12) NOT NULL)

CREATE TABLE [dbo].[tblPRODUCT\_TYPE](

[ProdTypeID] [int] IDENTITY(1,1) PRIMARY KEY NOT NULL,

[ProdTypeName] [varchar](50) NOT NULL,

[ProdTypeDescr] varchar(1000) NULL)

CREATE TABLE [dbo].[tblPRODUCT](

[ProdID] [int] IDENTITY(1,1) PRIMARY KEY NOT NULL,

[ProdName] [varchar](50) NOT NULL,

[ProdTypeID] INT FOREIGN KEY REFERENCES tblPRODUCT\_TYPE (ProdTypeID) NOT NULL,

[Price] [numeric](12, 2) NULL)

CREATE TABLE [dbo].[tblORDER](

[OrderID] [int] IDENTITY(1,1) PRIMARY KEY NOT NULL,

[CustID] [int] FOREIGN KEY REFERENCES tblCUSTOMER (CustID) NOT NULL,

[OrderDate] [datetime] DEFAULT GetDate() NULL)

CREATE TABLE [dbo].[tblLINE\_ITEM](

[LineItemID] [int] IDENTITY(1,1) PRIMARY KEY NOT NULL,

[OrderID] [int] FOREIGN KEY REFERENCES tblORDER (OrderID) NULL,

[ProdID] [int] FOREIGN KEY REFERENCES tblPRODUCT (ProdID) NOT NULL,

[Qty] [int] NOT NULL)

CREATE TABLE [dbo].[tblCART](

[CartID] [int] IDENTITY(1,1) PRIMARY KEY NOT NULL,

[CustID] [int] FOREIGN KEY REFERENCES tblCUSTOMER (CustID) Not NULL,

[ProductID] [int] FOREIGN KEY REFERENCES tblPRODUCT (ProdID) NOT NULL,

[CartDate] DATETIME DEFAULT GetDate() NOT NULL,

[Qty] [int] NOT NULL

)

GO

/\*

Create computed columns.

\*/

CREATE FUNCTION fnCalc\_LI\_PriceExtd (@LineItemID INT)

RETURNS numeric(12,2)

AS

BEGIN

DECLARE @RET numeric(12,2) =

(SELECT P.Price \* LI.QTY FROM tblPRODUCT P JOIN tblLINE\_ITEM LI ON P.ProdID = LI.ProdID WHERE LI.LineItemID = @LineItemID)

RETURN @RET

END

GO

ALTER TABLE tblLINE\_ITEM

ADD PriceExtd AS (dbo.fnCalc\_LI\_PriceExtd(LineItemID))

GO

CREATE FUNCTION fnCalc\_ORDER\_TotalAmount (@OrderID INT)

RETURNS numeric(12,2)

AS

BEGIN

DECLARE @RET numeric(12,2) =

(SELECT SUM(PriceExtd) FROM tblLINE\_ITEM WHERE OrderID = @OrderID)

RETURN @RET

END

GO

ALTER TABLE tblORDER

ADD TotalAmount AS (dbo.fnCalc\_ORDER\_TotalAmount(OrderID))

/\*

Populate tables with a few rows each

\*/

USE [EXAMPLE\_CART2]

GO

INSERT INTO [dbo].[tblCUSTOMER] ([CustomerFname],[CustomerLname],[CustomerAddress],[CustomerCity],[CustomerCounty]

,[CustomerState],[CustomerZIP],[AreaCode],[Email],[BusinessName],[DateOfBirth],[PhoneNum])

VALUES ('Tari','Greeb','9116 NW Juneberry Valley Walk','Gratis','Preble','Ohio, OH','45330','937','Tari.Greeb939@bpweakleyconstruction.com','Weakley Construction','1934-05-28','560-4745'),

('Alanna','Aplington','10387 South Magnuson Hill Avenue','New Haven','New Haven','Connecticut, CT','06532','203','Alanna.Aplington076@fdwashingtondesignassociates.com','Washington Design Associates','1988-08-20','243-4244'),

('Phylis','Balich','24764 SW Grove View Alley','Rule','Haskell','Texas, TX','79548','940','Phylis.Balich589@dcgreenwoodgasdiesel.com','Greenwood Gas & Diesel','1956-04-25','374-8627'),

('Celestine','Wildermuth','20980 South Linden Heights Trail','Taylor','Loup','Nebraska, NE','68879','308','Celestine.Wildermuth075@hbkingfishergalleydesigns.com','Kingfisher Galley Designs','1988-12-25','656-5605'),

('Lanell','Adonis','9775 SE Wisconsin Park Place','Bucyrus','Miami','Kansas, KS','66013','913','Lanel.Adonis441@jcalleganymaterials.com','Allegany Materials','1965-11-05','326-0277'),

('Hester','Guess','32292 SE Franklin Beach Sloop','Ellis Grove','Randolph','Illinois, IL','62241','618','Hester.Guess683@zkcusterceramicsandtile.com','Custer Ceramics and Tile','1950-09-04','264-1941'),

('Harriett','Mearing','12443 West Montana View Highway','Leflore','Le Flore','Oklahoma, OK','74942','918','Harriett.Mearing477@lamiamisecuritypatrol.com','Miami Security Patrol','1963-06-05','505-6481'),

('Ninfa','Hockenbrock','10652 E Fieldstone Lake Sloop','Bayonne','Hudson','New Jersey, NJ','07002','551','Ninfa.Hockenbrock592@rsfayetteplating.com','Fayette Plating','1956-02-22','264-8350'),

('Larry','Ochinang','9511 N Tompkins Terrace Sloop','King Of Prussia','Chester','Pennsylvania, PA','19487','610','Larry.Ochinang999@blpauldingcoating.com','Paulding Coating','1930-08-10','161-9631'),

('Renato','Isgrigg','14072 North Cedar Park Hillclimb','Caldwell','Noble','Ohio, OH','43724','740','Renato.Isgrigg797@dvbonnevillejobshop.com','Bonneville Job Shop','1943-04-07','705-5508')

GO

INSERT INTO [dbo].[tblPRODUCT\_TYPE] ([ProdTypeName],[ProdTypeDescr])

VALUES

('Electronics', 'Electronics are products that are plugged into a standard outlet such as stereo equipment, televisions and computers'),

('Housewares', 'Housewares are products designed for consumer use in the home including such items as kitchen utensils, tableware, and decorative objects'),

('Automotive', 'Automotive products are car parts or items associated with maintenance like tires, oil, spark plugs and brake fluid'),

('Food and Beverage', 'Food and Beverage are any products that can be eaten by consumers')

GO

INSERT INTO [dbo].[tblPRODUCT] ([ProdName],[ProdTypeID], [Price])

VALUES

('Super VX-12 Home Stereo' , (SELECT ProdTypeID FROM tblPRODUCT\_TYPE WHERE ProdTypeName = 'Electronics'), 576.32),

('Ultra Max Table Set', (SELECT ProdTypeID FROM tblPRODUCT\_TYPE WHERE ProdTypeName = 'Housewares'), 162.98),

('Premium Radial JT51', (SELECT ProdTypeID FROM tblPRODUCT\_TYPE WHERE ProdTypeName = 'Automotive'), 92.48),

('Extended-Length USB 16 GB Storage Device', (SELECT ProdTypeID FROM tblPRODUCT\_TYPE WHERE ProdTypeName = 'Electronics'), 56.77),

('Standard 2-Quart Mixer', (SELECT ProdTypeID FROM tblPRODUCT\_TYPE WHERE ProdTypeName = 'Housewares'), 71.18),

('Rugged 94-Inch Truck Bed Shell' , (SELECT ProdTypeID FROM tblPRODUCT\_TYPE WHERE ProdTypeName = 'Automotive'), 281.88),

('XL 21-inch High-Definition LCD Monitor', (SELECT ProdTypeID FROM tblPRODUCT\_TYPE WHERE ProdTypeName = 'Electronics'), 230.92),

('Power Master 54 Vacuum', (SELECT ProdTypeID FROM tblPRODUCT\_TYPE WHERE ProdTypeName = 'Housewares'), 312.49),

('24K Bristle Scrub Brush', (SELECT ProdTypeID FROM tblPRODUCT\_TYPE WHERE ProdTypeName = 'Automotive'), 11.79),

('Battery Mate 2000 Power Supply', (SELECT ProdTypeID FROM tblPRODUCT\_TYPE WHERE ProdTypeName = 'Electronics'), 319.23),

('Curly Fries', (SELECT ProdTypeID FROM tblPRODUCT\_TYPE WHERE ProdTypeName = 'Food and Beverage'), 7.89),

('Diet Coke', (SELECT ProdTypeID FROM tblPRODUCT\_TYPE WHERE ProdTypeName = 'Food and Beverage'), 4.90),

('Cheese Burger', (SELECT ProdTypeID FROM tblPRODUCT\_TYPE WHERE ProdTypeName = 'Food and Beverage'), 10.90)

GO

/\*

Create the stored procedure to locate a customerID after being provided first name, last name, date of birth and zip code.

This is an example of a procedure using OUTPUT parameter

\*/

CREATE PROCEDURE [dbo].[uspFINDCustID]

@Fname varchar(35),

@Lname varchar(35),

@DOB Date,

@Zip char(12),

@CustID INT OUT

AS

BEGIN

SET @CustID = (

SELECT CustID

FROM tblCUSTOMER

WHERE CustomerFname = @Fname

AND CustomerLname = @Lname

AND DateOfBirth = @DOB

AND CustomerZIP = @Zip)

END

GO

/\*

Create the stored procedure to populate items/products into the table tblCART.

\*/

CREATE PROCEDURE [dbo].[uspPopCart]

@ProdName varchar(35),

@CustID INT,

@Date DateTime,

@Qty INT

AS

BEGIN

DECLARE @ProdID INT

SET @ProdID = (

SELECT ProdID

FROM tblPRODUCT

WHERE ProdName = @ProdName)

--@CustID, @Date and @Qty will be passed-in from calling stored procedure

INSERT INTO tblCART (CustID, ProductID, CartDate, Qty)

VALUES (@CustID, @ProdID, @Date, @Qty)

END

GO

/\*

Create the stored procedure to process the contents of tblCART.

This is an example of a variable being constructed as a TABLE which resides in memory

\*/

USE EXAMPLE\_CART2

GO

CREATE PROCEDURE uspProcessTMP\_Cart

@CustID INT

AS

--create a temp table to hold tblCART values while processing contents into tblORDER

-- and tblLINE\_ITEM

DECLARE @CART TABLE

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

ProdID INT not null,

Qty INT not null)

/\*

\* populate the temp table @CART with values from tblCART

\* reminder that tblCART is not organized and may include multiple rows of similar products

with different quantities that will need to be consolidated and summarized

\* this process will allow for GROUP BY on ProdID to summarize quantities

\*/

INSERT INTO @CART (ProdID, Qty)

SELECT ProductID, SUM(qty) AS QTY

FROM tblCART

WHERE CustID = @CustID

GROUP BY ProductID

/\*

--now we are ready to 'rip' through the temp table to create a new row in tblORDER

--and add rows to the LINE\_ITEM that are associated to a single ORDER

\*/

--first task is to determine the number of rows to cycle through and process

DECLARE @Count INT

SET @COUNT = (SELECT Count(\*) FROM @CART)

--this variable (@count) is going to be the number of loops we take with WHILE statement

--establish variables that are needed for the INSERT statement

DECLARE @ID INT

DECLARE @ProdID INT

DECLARE @Qty INT

DECLARE @OrderID INT

/\*

--begin WHILE loop with @Count (which reflects the number of rows in tblCART that have

--been copied into a temp table for processing (we are safer processing tblCART rows in

--a temp table as opposed to the actual tblCART table because many users are populating tblCART)

--establish an explicit transaction wrapper to hold insert activity into tblORDER and

tblLINE\_ITEM in a single statement; this allows everything to be rolled-back together

if there is an error

\*/

-- SET XACT\_ABORT ON will render the transaction uncommittable

-- when the constraint violation occurs.

SET XACT\_ABORT ON;

BEGIN TRY

BEGIN TRANSACTION G1

INSERT INTO tblORDER ([CustID],[OrderDate])

VALUES (@CustID, GetDate())

SET @OrderID = (SELECT Scope\_Identity())

WHILE @Count > 0 --begin loop to process all rows from #CART; @Count is number of rows to be processed

BEGIN

SET @ID = (SELECT MIN(tmpCartID) FROM @CART)

SET @ProdID = (SELECT TOP 1 ProdID FROM @CART WHERE tmpCartID = @ID)

SET @Qty = (SELECT Qty FROM @CART WHERE tmpCartID = @ID)

--'old-school'error-handling method

-- sp\_addmessage 50011, 11, 'OrderID cannot be NULL' was previously added to system

IF @OrderID IS NULL

BEGIN

RAISERROR (50011, 11, 1)

END

ELSE

--'New School' error-handling

--need to make sure there is a value in @ProdID variable

IF @ProdID IS NULL

THROW 50012, 'ProdID cannot be NULL.', 1;

--BREAK --exit WHILE loop if there is no value in @ProdID

ELSE

-- if things are good so far, issue INSERT statement in tblLINE\_ITEM

INSERT INTO tblLINE\_ITEM ([OrderID],[ProdID],[Qty])

VALUES (@OrderID, @ProdID, @Qty)

-- we MUST clean-up the row just INSERTed into tblLINE\_ITEM by DELETING it from #CART

-- or else we will see it again and enter duplicate rows in tblLINE\_ITEM

DELETE

FROM @CART

WHERE tmpCartID = @ID --anchoring to @ID ensures we are deleting only one row

-- we must also decrement the boolean variable that keeps the loop alive; if we do not have

-- this line then the loop will never reach zero and run infinitely

SET @Count = @Count -1

END

-- If the INSERT operation into tblORDER and tblLINE\_ITEM succeeds, commit the transaction.

-- The CATCH block will not execute if there are no errors.

IF @@ERROR <> 0 --just looking for any global errors at this time

BEGIN

ROLLBACK TRANSACTION G1

END

ELSE

-- Test whether the transaction is active and valid.

IF (XACT\_STATE()) = 1

BEGIN

PRINT 'The transaction is committable.' +

' Committing transaction.'

COMMIT TRANSACTION G1;

END

--DELETE rows from tbl that have just been processed

-- conduct a count to make sure number of rows processed is equal to the number of rows to be deleted

PRINT @@Rowcount

DELETE

FROM tblCART

WHERE CustID = @CustID

END TRY

BEGIN CATCH

-- Test XACT\_STATE for 0 or -1

-- If -1, the transaction is uncommittable and should be rolled back

-- XACT\_STATE = 0 means there is no transaction and a commit or rollback operation would generate an error

-- Test whether the transaction is uncommittable.

IF (XACT\_STATE()) = -1

BEGIN

PRINT 'The transaction is in an uncommittable state.' +

' Rolling back transaction.'

SELECT

ERROR\_NUMBER() AS ErrorNumber

,ERROR\_MESSAGE() AS ErrorMessage;

ROLLBACK TRANSACTION G1;

END;

END CATCH;

/\*

Example execution to find CustID from [tblCUSTOMER]

\*/

DECLARE @GetDate DateTime = (SELECT GetDate())

DECLARE @CustID\_OUT INT

EXEC uspFINDCustID

@Fname = 'Renato',

@Lname = 'Isgrigg',

@DOB = '1943-04-07',

@Zip = 43724,

@CustID = @CustID\_OUT OUT

PRINT @CustID\_OUT

-- Add items to cart with CustID just captured by running the

-- block of code starting with the two DECLARE statements above

-- and include all of the calls of uspPopCart below

-- Also, after populating tblCART, select \* FROM tblCART to view contents to notice duplicate productIDs in multiple rows

-- as that is how the data was entered

EXEC uspPopCart

@ProdName = 'Premium Radial JT51',

@CustID = @CustID\_OUT,

@Date = @GetDate,

@Qty = 12

EXEC uspPopCart

@ProdName = 'Cheese Burger',

@CustID = @CustID\_OUT,

@Date = @GetDate,

@Qty = 4

EXEC uspPopCart

@ProdName = 'Cheese Burger',

@CustID = @CustID\_OUT,

@Date = @GetDate,

@Qty = 5

EXEC uspPopCart

@ProdName = 'Premium Radial JT51',

@CustID = @CustID\_OUT,

@Date = @GetDate,

@Qty = 20

EXEC uspPopCart

@ProdName = 'Super VX-12 Home Stereo',

@CustID = @CustID\_OUT,

@Date = @GetDate,

@Qty = 11

EXEC uspPopCart

@ProdName = 'Super VX-12 Home Stereo',

@CustID = @CustID\_OUT,

@Date = @GetDate,

@Qty = 10

EXEC uspPopCart

@ProdName = 'Diet Coke',

@CustID = @CustID\_OUT,

@Date = '12-03-2015',

@Qty = 12

EXEC uspPopCart

@ProdName = 'Curly Fries',

@CustID = @CustID\_OUT,

@Date = '1-12-2016',

@Qty = 1

EXEC uspPopCart

@ProdName = 'Diet Coke',

@CustID = @CustID\_OUT,

@Date = '12-03-2015',

@Qty = 1

EXEC uspPopCart

@ProdName = 'Curly Fries',

@CustID = @CustID\_OUT,

@Date = '1-12-2016',

@Qty = 1

/\*

Example execution to process the contents of tblCART for a single customer

Be sure to pass the same columns that were used to populate tblCART

After processing, SELECT \* FROM tblLINE\_ITEM to view processed order that has had productIDs summarized

\*/

SELECT \* FROM tblLINE\_ITEM

SELECT \* FROM tblCART

SELECT \* FROM tblORDER

GO

DECLARE @CustID\_OUT INT

EXEC uspFINDCustID

@Fname = 'Renato',

@Lname = 'Isgrigg',

@DOB = '1943-04-07',

@Zip = 43724,

@CustID = @CustID\_OUT OUT

EXEC uspProcessTMP\_Cart @CustID = @CustID\_OUT

SELECT \* FROM tblLINE\_ITEM

SELECT \* FROM tblCART

SELECT \* FROM tblORDER

GO