**STORED PROCEDURES**

--ADD NEW LOCATION NAME

CREATE PROCEDURE [dbo].[usp\_AddNewLocationName]

@LocationName varchar(35),

@LocationDescription varchar(50)

AS

BEGIN TRAN T1

INSERT INTO tblLocation\_Name (LocationDescription, LocationName) VALUES (@LocationDescription, @LocationName)

IF @@ERROR <> 0

ROLLBACK TRAN T1

ELSE

COMMIT TRAN

--ADD MENU ITEM INGREDIENT

CREATE PROCEDURE [dbo].[usp\_AddMenuItemIngredient]

@MenuItemName varchar(35),

@IngredientName varchar(35),

@IngredientDescription varchar(70),

@SupplierName varchar(75),

@SupplierDescription varchar(70),

@Quantity INT,

@Calories INT

AS

DECLARE @IngredientID INT, @MenuItemID INT

EXEC [uspGetIngredientID]

@IngredientDescription = @IngredientDescription,

@SupplierName = @SupplierName,

@SupplierDescription = @SupplierDescription,

@IngredientName = @IngredientName,

@IngredientID = @IngredientID OUTPUT

EXEC [GetMenuItemID]

@MenuItemName = @MenuItemName,

@MenuItemID = @MenuItemID OUTPUT

IF @IngredientID IS NULL

BEGIN

PRINT '@IngredientID IS NULL and will fail on insert statement; process terminated'

RAISERROR ('IngredientID variable @IngredientID cannot be NULL',11,1)

RETURN END

IF @MenuItemID IS NULL

BEGIN

PRINT '@MenuItemID IS NULL and will fail on insert statement; process terminated'

RAISERROR ('MenuItemID variable @MenuItemID cannot be NULL',11,1)

RETURN END

BEGIN TRAN T1

INSERT INTO [dbo].[tblMenuItem\_Ingredient] (MenuItemID,IngredientID, Quantity, Calories)

VALUES (@MenuItemID, @IngredientID, @Quantity, @Calories)

IF @@ERROR <> 0

ROLLBACK TRAN T1

ELSE

COMMIT TRAN T1

**BUSINESS RULES**

--TRUCK LOCATION ONLY IN SEATTLE

CREATE FUNCTION [dbo].[fn\_seattleTruckLocation]()

RETURNS INT

as

begin

declare @ret int = 0

if exists (

select \* from tblLocation l

where l.Zip < 98101 OR l.Zip > 98199

) set @ret = 1

return @ret

end

ALTER TABLE tblLocation

ADD CONSTRAINT locationCheck

CHECK (dbo.fn\_seattleTruckLocation() = 0)

--NO CUSTOMER CAN HAVE OVER 50 ORDERS IN ONE DAY

CREATE FUNCTION [dbo].[fn\_customerOrderLimit]()

RETURNS INT

as

begin

DECLARE @ret int = 0

if exists (

select \* from tblCustomer c

join tblOrder o on o.CustomerID = c.CustomerID

where o.OrderDate = GETDATE()

GROUP BY o.CustomerID

having count(\*) > 50

) set @ret = 1

return @ret

end

ALTER TABLE tblOrder

ADD CONSTRAINT maxOrders

CHECK (dbo.fn\_customerOrderLimit () = 0)

**COMPUTED COLUMNS**

**--**EMPLOYEE AGE

CREATE FUNCTION [dbo].[fn\_employeeAge](@EmployeeID int)

RETURNS int

AS

BEGIN

DECLARE @ret int = (

SELECT DateDiff(

year,

(SELECT DOB FROM tblEmployee WHERE EmployeeID = @EmployeeID),

GETDATE()))

RETURN @ret

END

ALTER TABLE tblEmployee

ADD EmployeeAge

AS dbo.fn\_seattleTruckLocation(EmployeeID)

--NUMBER OF EMPLOYEES PER TRUCK

CREATE FUNCTION [dbo].[fn\_employeePerTruck](@TruckID int)

RETURNS int

AS

BEGIN

DECLARE @ret int= (

SELECT count(\*)

FROM tblTruck t

JOIN tblEmployee e ON e.TruckID = t.TruckID

GROUP BY t.TruckID

)

RETURN @ret

END

ALTER TABLE tblTruck

ADD NumEmployees

AS dbo.fn\_employeePerTruck(TruckID)

**VIEWS**

--LIST MENU PER TRUCK

--AGE DEMOGRAPHICS OF EACH TYPE OF TRUCK