Sample SQL Code *Mock* Petco Project

INFO 330

Team Lead Role

```
use INFO_330_Proj_2


Create table tblCUSTOMER
(CustomerID Integer Identity(1,1) Primary Key,
Fname varchar(30) not null,
Lname varchar(30) not null,
Birthdate Date NOT NULL,
CustAddress varchar(80) not null,
CustCity varchar(50) not null,
CustState varchar(30) not null,
AreaCode varchar(5) not null,
Phone varchar(20) not null)
go


--foreign key: CustomerID
Create table tblORDER
(OrderID Integer Identity(1,1) Primary Key,
OrderDate Date not null,
CustomerID int not null)
GO


alter table tblORDER
ADD CONSTRAINT FK_tblORDER_CustomerID
FOREIGN KEY (CustomerID) references tblCUSTOMER(CustomerID)
```

```
go

--foreign key: OrderID, ProductID
Create table tblORDER_PRODUCT
(OrderProductID integer Identity(1,1) Primary Key,
OrderID int not null,
ProductID int not null,
Amount Numeric(8,2) not null)
go

ALTER TABLE tblORDER_PRODUCT
ADD CONSTRAINT FK_tblORDER_PRODUCT_OrderID
FOREIGN KEY (OrderID)
REFERENCES tblORDER (OrderID)
GO

ALTER TABLE tblORDER_PRODUCT
ADD CONSTRAINT FK_tblORDER_PRODUCT_ProductID
FOREIGN KEY (ProductID)
REFERENCES tblProduct (ProductID)
GO

--foreign key: OrderProductID, RatingID
Create table tblREVIEW
(ReviewID integer Identity(1,1) Primary Key,
ProdReview varchar(500) not null,
RatingID int not null,
OrderProductID int not null)
```

```
go

ADD CONSTRAINT FK_tblREVIEW_OrderProductID
FOREIGN KEY (OrderProductID)
REFERENCES tblORDER_PRODUCT (OrderProductID)
GO

ALTER TABLE tblREVIEW
ADD CONSTRAINT FK_tblREVIEW_RatingID
FOREIGN KEY (RatingID)
REFERENCES tblRATING (RatingID)
GO

create table tblRATING
(RatingID integer Identity(1,1) Primary Key,
Rating Numeric(2,2))
go

--foreign key: ProductTypeID, SupplierID
Create table tblPRODUCT
(ProductID integer Identity(1,1) Primary Key,
ProdName varchar(50) not null,
ProdPrice Numeric(7,2),
ProdDescr varchar(500) null,
SupplierID int not null,
ProductTypeID int not null,
ProdStock int not null)
```

```
Go

ALTER TABLE tblPRODUCT
ADD CONSTRAINT FK_tblPRODUCT_ProductTypeID
FOREIGN KEY (ProductTypeID)
REFERENCES tblPRODUCT_TYPE (ProductTypeID)
GO

ALTER TABLE tblPRODUCT
ADD CONSTRAINT FK_tblPRODUCT_SupplierID
FOREIGN KEY (SupplierID)
REFERENCES tblSUPPLIER (SupplierID)
GO

Create table tblPRODUCT_TYPE
(ProductTypeID integer Identity(1,1) primary key
ProdTypeName varchar(50) not null,
ProdTypeDescr varchar (500) null)
go

--Foreign id: ProductID, IngredientID
Create table tblPROD_INGREDIENT
(ProductIngredientID integer Identity(1,1) primary key,
ProductID int not null,
IngredientID int not null,
IngredientAmount int not null)
go
```

```
ALTER TABLE tblPROD_INGREDIENT
ADD CONSTRAINT FK_tblPROD_INGREDIENT_ProductID
FOREIGN KEY (ProductID)
REFERENCES tblPRODUCT (ProductID)
GO

ALTER TABLE tblPROD_INGREDIENT
ADD CONSTRAINT FK_tblPROD_INGREDIENT_IngredientID
FOREIGN KEY (IngredientID)
REFERENCES tblINGREDIENT (IngredientID)
GO

Create table tblINGREDIENT
(IngredientID integer Identity(1,1) primary key,
IngredientName varchar(60))
go
--foreign key: ProductID, SpeciesID
Create table tblPRODUCT_SPECIES
(ProductSpeciesID integer Identity(1,1) primary key,
ProductID int not null,
SpeciesID int  not null)
Go

ALTER TABLE tblPRODUCT_SPECIES
ADD CONSTRAINT FK_tblPROD_SPECIES_ProductID
FOREIGN KEY (ProductID)
```

```
REFERENCES tblPRODUCT (ProductID)

GO


ALTER TABLE tblPRODUCT_SPECIES

ADD CONSTRAINT FK_tblPROD_SPECIES_SpeciesID

FOREIGN KEY (SpeciesID)

REFERENCES tblSPECIES (SpeciesID)

GO


Create table tblSPECIES

(SpeciesID integer Identity(1,1) primary key,

SpeciesName varchar(50) not null)

go


--foreign key: ProductID, HealthIssueID

Create table tblPRODUCT_HEALTH_ISSUE

(ProductHealthIssueID integer identity(1,1) primary key,

HealthIssueID int not null,

ProductID int not null)

go


ALTER TABLE tblPRODUCT_HEALTH_ISSUE

ADD CONSTRAINT FK_tblPRODUCT_HEALTH_ISSUE_ProductID

FOREIGN KEY (ProductID)

REFERENCES tblPRODUCT(ProductID)

GO
```

```
ALTER TABLE tblPRODUCT_HEALTH_ISSUE

ADD CONSTRAINT FK_tblPRODUCT_HEALTH_ISSUE_HealthIssueID

FOREIGN KEY (HealthIssueID)

REFERENCES tblHEALTH_ISSUE (HealthIssueID)

GO


Create table tblHEALTH_ISSUE

(HealthIssueID integer identity(1,1) primary key,

add HealthIssueName varchar(50) not null)

go


Create table tblSUPPLIER

(SupplierID integer identity(1,1) primary key,

SupplierName varchar(60) not null)

go


--VICTORIA:


--STORED PROCEDURES
--1)Populate Species


GO
CREATE or alter  PROCEDURE vmInsert_Spec
@S_Name varchar(50)
AS


BEGIN TRANSACTION G1
INSERT INTO tblSPECIES (SpeciesName)
VALUES (@S_Name)


IF @@ERROR <> 0
   BEGIN
```

```
      PRINT 'rollback'
   ROLLBACK TRANSACTION G1
END
COMMIT TRANSACTION G1
GO


EXEC vmInsert_Spec
@S_Name = 'Dog'


EXEC vmInsert_Spec
@S_Name = 'Cat'


EXEC vmInsert_Spec
@S_Name = 'Fish'


EXEC vmInsert_Spec
@S_Name = 'Small Pet'


EXEC vmInsert_Spec
@S_Name = 'Bird'


EXEC vmInsert_Spec
@S_Name = 'Reptile'


GO


--2) Populate Health_Issue


GO
CREATE or alter PROCEDURE vmInsert_HealthIssue
@Health_Name varchar(50)
AS



BEGIN TRANSACTION G1
INSERT INTO tblHEALTH_ISSUE (HealthIssueName)
VALUES (@Health_Name)
```

```
IF @@ERROR <> 0
  BEGIN
     PRINT 'rollback'
  ROLLBACK TRANSACTION G1
END
COMMIT TRANSACTION G1
GO

EXEC vmInsert_HealthIssue
@Health_Name = 'Flea and Tick'


EXEC vmInsert_HealthIssue
@Health_Name = 'Arthritis'


EXEC vmInsert_HealthIssue
@Health_Name = 'Heart Strength'


EXEC vmInsert_HealthIssue
@Health_Name = 'Stress and Anxiety'


EXEC vmInsert_HealthIssue
@Health_Name = 'Coat Maintenance'


GO


/* BUSINESS RULES


1) No Residents living in California, CA are able to
buy over 10 Bird products */


GO
CREATE or alter  FUNCTION vm_ProductSpecies()
RETURNS INT
AS
BEGIN
```

```
DECLARE @RET INT = 0
IF EXISTS (SELECT *
FROM tblCUSTOMER C
  JOIN tblORDER O ON C.CustomerID = O.CustomerID
  JOIN tblORDER_PRODUCT OP ON O.OrderID = OP.OrderProductID
  JOIN tblPRODUCT P ON OP.ProductID = P.ProductTypeID
  JOIN tblPRODUCT_SPECIES PS ON P.ProductID = P.ProductID
  JOIN tblSPECIES S ON PS.ProductSpeciesID = S.SpeciesID
WHERE C.CustState = 'California, CA'
  AND P.ProductID <= 10
  AND S.SpeciesName = 'Bird'
  )


SET @RET = 1
RETURN @RET
END
GO


ALTER TABLE tblPRODUCT_TYPE WITH Nocheck
ADD CONSTRAINT CK_Product_Species_Num
CHECK (dbo.vm_ProductSpecies() = 0)


--2) Customer can't purchase more products than whatever's in stock.


GO
CREATE or alter FUNCTION vm_InStockPurchases()
RETURNS INT
AS
BEGIN


DECLARE @RET INT = 0
IF EXISTS (SELECT *
FROM tblCUSTOMER C
  JOIN tblORDER O ON C.CustomerID = O.CustomerID
  JOIN tblORDER_PRODUCT OP ON O.OrderProductID = OP.OrderProductID
  JOIN tblPRODUCT P ON OP.ProductID = P.ProductID
WHERE OP.Amount < P.ProdStock
)


SET @RET = 1
RETURN @RET
```

```
END
GO


ALTER TABLE tblCUSTOMER WITH Nocheck
ADD CONSTRAINT CK_NoStock_Purch
CHECK (dbo.vm_InStockPurchases() = 0)


--COMPUTED COLUMNS
--1) Average rating for one product_type.


GO
CREATE or alter FUNCTION vmAverageRating(@PK INT)
RETURNS NUMERIC (10,2)
AS


BEGIN
DECLARE @RET NUMERIC(10,2) = (
  SELECT AVG(Rating)
  FROM tblRATING R
     JOIN tblREVIEW RV ON R.RatingID = RV.RatingID
     JOIN tblORDER_PRODUCT OP ON RV.OrderProductID = OP.OrderProductID
     JOIN tblPRODUCT P ON OP.ProductID = P.ProductID
     JOIN tblPRODUCT_TYPE PT ON P.ProductTypeID = PT.ProductTypeID
  WHERE R.RatingID = @PK)
  RETURN @RET
  END
  GO


ALTER TABLE tblRATING
ADD vmCalc_Reg AS (dbo.vmAverageRating(RatingID))




--2) Number of order_products per species.


GO
CREATE or alter FUNCTION vmOrder_ProdSpecies(@PK INT)
RETURNS NUMERIC (10,2)
AS
```

```
BEGIN
DECLARE @RET NUMERIC(10,2) = (
  SELECT SUM(OrderProductID)
  FROM tblORDER_PRODUCT OP
    JOIN tblPRODUCT P ON OP.ProductID = P.ProductID
    JOIN tblPRODUCT_SPECIES PS ON P.ProductSpeciesID = PS.ProductSpeciesID
    JOIN tblSPECIES S ON PS.SpeciesID = S.SpeciesID
  WHERE R.RatingID = @PK)
  RETURN @RET
  END
  GO


ALTER TABLE tblRATING
ADD vmCalc_Reg AS (dbo.vmAverageRating(RatingID))


--COMPLEX QUERIES


--1) Return CustomerID, First and Last name, and OrderID


SELECT C.CustomerID, C.Fname, C.Lname, O.OrderID
FROM tblCUSTOMER C
  JOIN tblORDER O ON C.CustomerID = O.CustomerID
  JOIN tblORDER_PRODUCT OP ON O.OrderID = OP.OrderID
  JOIN tblPRODUCT P ON OP.ProductID = P.ProductID
  JOIN tblREVIEW R ON OP.OrderProductID = R.OrderProductID
  JOIN tblRATING RT ON R.RatingID = RT.RatingID
GROUP BY C.CustomerID, C.Fname, C.Lname, O.OrderID


--2) Return Product Names with less than 50 Ingredients


SELECT P.ProductID, PT.ProdTypeName, SUM(IngredientAmount) AS TotalIngredients
FROM tblPRODUCT_TYPE PT
  JOIN tblPRODUCT P ON PT.ProductTypeID = P.ProductTypeID
  JOIN tblPROD_INGREDIENT PI ON P.ProductID = PI.ProductID
  JOIN tblINGREDIENT I ON PI.IngredientID = I.IngredientID
GROUP BY P.ProductID, PT.ProdTypeName
HAVING SUM(IngredientAmount) < 50
ORDER BY TotalIngredients
```

Erica:

```
--Stored procedure to populate customer table
Create or alter procedure eluo_insertCustomer
@FName varchar(60),
@LName varchar(60),
@Birthdate Date,
@CustAddress varchar(80),
@CustCity varchar(50),
@CustState varchar(30),
@ZipCode varchar(5),
@Phone varchar(20)

as

begin TRANSACTION t1
insert into tblCUSTOMER(Fname, Lname, Birthdate, CustAddress, CustCity, CustState, ZipCode, Phone)
values(@FName, @LName, @Birthdate, @CustAddress, @CustCity, @CustState, @ZipCode, @Phone)
if @@error <> 0
   begin
      print 'Missing or misspelled value. Could not insert.'
      rollback TRANSACTION t1
   end
commit TRANSACTION t1

go

Execute eluo_insertCustomer
@FName = 'Bob',
@LName = 'Ross',
@Birthdate = 'October 29, 1942',
@CustAddress = '13454 Rosewood St',
@CustCity = 'Orlando',
@CustState = 'Florida, FL',
@ZipCode = '32789',
@Phone = '409-110-2340'
go

Execute eluo_insertCustomer
@FName = 'Ally',
@LName = 'Gator',
@Birthdate = 'October 1, 2010',
@CustAddress = '14049 Swamp St',
@CustCity = 'Tampa Bay',
@CustState = 'Florida, FL',
@ZipCode = '33084',
```

```
@Phone = '402-441-4705'
go

Execute eluo_insertCustomer
@FName = 'Crocko',
@LName = 'Dyle',
@Birthdate = 'October 9, 2010',
@CustAddress = '14041 Swamp St',
@CustCity = 'Tampa Bay',
@CustState = 'Florida, FL',
@ZipCode = '33084',
@Phone = '402-455-4790'
go

Execute eluo_insertCustomer
@FName = 'Cody',
@LName = 'Fish',
@Birthdate = 'March 1, 1980',
@CustAddress = '14041 Swamp St',
@CustCity = 'Cape Cod',
@CustState = 'Florida, FL',
@ZipCode = '14084',
@Phone = '902-485-6890'
go

Execute eluo_insertCustomer
@FName = 'Cody',
@LName = 'Fish',
@Birthdate = 'June 30, 1992',
@CustAddress = '12091 Ocean St',
@CustCity = 'Boston',
@CustState = 'Massachusetts, MA',
@ZipCode = '58324',
@Phone = '337-805-1267'
go

Execute eluo_insertCustomer
@FName = 'Guy',
@LName = 'Eldoon',
@Birthdate = 'July 10, 1972',
@CustAddress = '11111 Noodle St',
@CustCity = 'Los Angeles',
@CustState = 'California, CA',
@ZipCode = '78224',
@Phone = '228-395-9067'
go
```

```
--Stored procedure to populate product table

create or alter procedure eluo_getPTID
@PTName2 varchar(50),
@PT_ID2 int output
as

set @PT_ID2 = (Select pt.ProductTypeID
            from tblPRODUCT_TYPE pt
            where pt.ProdTypeName = @PTName2)
go

create or alter procedure eluo_getSupID
@SName2 varchar(50),
@S_ID2 int output
as

set @S_ID2 = (Select SupplierID
            from tblSUPPLIER
            where SupplierName = @SName2)
go


Create or alter procedure eluo_insertProduct
@ProductTypeN2 varchar(50),
@SupplierN2 varchar(60),
@ProdName2 varchar(50),
@ProdPrice2 Numeric(7,2),
@ProdDescr2 varchar(500),
@ProdStock2 int

as

declare @PT_ID int, @S_ID int
Execute eluo_getPTID
@PTName2 = @ProductTypeN2,
@PT_ID2 = @PT_ID output

execute eluo_getSupID
@SName2 = @SupplierN2,
@S_ID2 = @S_ID output

begin transaction e2
insert into tblPRODUCT (ProductTypeID, SupplierID, ProdName, ProdPrice, ProdDescr, ProdStock)
   values(@PT_ID, @S_ID, @ProdName2, @ProdPrice2, @ProdDescr2, @ProdStock2)
   if @PT_ID is null or @S_ID is null
      begin
```

```
        print 'missing values. please retry.'
        rollback TRANSACTION e2
      end
   else
commit transaction e2
go

execute eluo_insertProduct
@ProductTypeN2 = 'Wet Food',
@SupplierN2 = 'Gentle Pet',
@ProdName2 = 'Natural Digestive Care Adult Wet Dog Food Chicken',
@ProdPrice2=  35.00,
@ProdDescr2 = 'sensitive stomach dog food',
@ProdStock2=  50

execute eluo_insertProduct
@ProductTypeN2 = 'Wet Food',
@SupplierN2 = 'Gentle Pet',
@ProdName2 = 'Natural Digestive Care Adult Wet Dog Food Beef',
@ProdPrice2=  37.00,
@ProdDescr2 = 'sensitive stomach dog food',
@ProdStock2= 69

execute eluo_insertProduct
@ProductTypeN2 = 'Wet Food',
@SupplierN2 = 'Gentle Pet',
@ProdName2 = 'Natural Digestive Care Adult Wet Dog Food Bacon',
@ProdPrice2=  31.00,
@ProdDescr2 = 'sensitive stomach dog food',
@ProdStock2= 20

execute eluo_insertProduct
@ProductTypeN2 = 'Dry Food',
@SupplierN2 = 'Tail Treats',
@ProdName2 = 'Adult Alaskan Pollock & Brown Rice',
@ProdPrice2=  45.00,
@ProdDescr2 = '15lb bag of sustainably-sourced Alaskan pollock & gently cooked grains for targeted skin and coat
support',
@ProdStock2= 19

execute eluo_insertProduct
@ProductTypeN2 = 'Vitamins',
@SupplierN2 = 'Tail Treats',
@ProdName2 = 'Nu Cat Chewable Tablets Multivitamin for Cats',
@ProdPrice2=  9.00,
@ProdDescr2 = 'All in one multivitamin great for cats of all ages with key minerals, fish oil omegas and taurine,
essential for maintaining healthy vision.',
```

```
@ProdStock2= 190

execute eluo_insertProduct
@ProductTypeN2 = 'Wet Food',
@SupplierN2 = 'Fine Pet Diner Inc',
@ProdName2 = 'Adult Perfect Weight Roasted Vegetable & Chicken Medley Canned Cat Food',
@ProdPrice2=  25.00,
@ProdDescr2 = 'A good source of protein to help your grown cat maintain lean muscle Scrumptious wet food made
with natural ingredients',
@ProdStock2= 54
go


--Buisiness rule: no products with chocolate can be added
Create or alter function eluo_noChoc()
Returns int
as

begin
Declare @Ret int = 0
If
Exists(Select p.ProductID
      from tblPRODUCT p
         join tblPROD_INGREDIENT pi on p.ProductID = pi.ProductID
         join tblINGREDIENT i on pi.IngredientID = i.IngredientID
      where i.ingredientName like '%Chocolate%')
      set @Ret = 1
Return @Ret
end
go

alter table tblPRODUCT
add constraint ck_noChoc
check(dbo.eluo_noChoc() = 0)

go

--Business rule: Reviews must have ratings between 1 and 5

create function eluo_reviewScale1to5()
boooReturns int
as

begin
declare @Ret int = 0
if
exists(Select r.ReviewID
```

```
        from tblREVIEW r
            join tblRating rate on rate.RatingID = r.RatingID
        where rate.Rating > 5
            or rate.Rating < 1)
        set @Ret = 1
return @Ret
end
go

alter table tblREVIEW
add constraint ck_RateValid
check(dbo.eluo_reviewScale1to5() = 0)
go

--Complex query: toy with the highest ratings
Select top 10 with ties p.ProductID, p.ProdName, AVG(rate.Rating) as 'AvgRating'
from tblPRODUCT p
    join tblPRODUCT_TYPE pt on p.ProductTypeID = pt.ProductTypeID
    join tblORDER_PRODUCT op on p.ProductID = op.ProductID
    join tblREVIEW r on op.OrderProductID = r.OrderProductID
    join tblRATING rate on r.RatingID = rate.RatingID
where pt.ProdTypeName like '%Toy%'
group by p.ProductID, p.ProdName
order by AVG(rate.RatingID) ASC
go

--Complex query: species with the most products aimed at its health issues health issues
Select top 1 with ties s.SpeciesID, s.SpeciesName, count(distinct hi.HealthIssueID) as 'NumHealthIssues'
from tblSPECIES s
    join tblProduct_Species ps on s.SpeciesID = ps.SpeciesID
    join tblProduct p on ps.ProductID = p.ProductID
    join tblProduct_Health_Issue phi on p.ProductID = phi.ProductID
    join tblHealth_Issue hi on phi.HealthIssueID = hi.HealthIssueID
group by s.SpeciesID, s.SpeciesName
order by count(distinct hi.HealthIssueID) desc
go

--computed column: how many different products one supplier supplies
Create function eluo_NumDiffProducts(@PK int)
returns int
as
begin
declare @Ret int
    set @Ret = (Select count(distinct p.ProductID)
                from tblSupplier s
                    join tblProduct p on s.SupplierID = p.SupplierID
                where s.SupplierID = @PK)
```

```
return @Ret
end
go

alter table tblSUPPLIER
add NumProducts as (dbo.eluo_NumDiffProducts(SupplierID))
go

--computed column: num reviews a product has
Create function eluo_numRev(@PK int)
returns int
as
begin
declare @Ret int
   set @Ret = (Select count(r.ReviewID)
            from tblProduct p
               join tblReview r on p.ProductID = r.reviewID
            where p.ProductID = @PK)
return @Ret
end
go

alter table tblPRODUCT
add NumReviews as (dbo.eluo_numRev(ProductID))
go
```

Zoe Steers:

```
-- Populate Ingrdient Table
GO
CREATE PROCEDURE zsPopulateIngredient
@IngredientName varchar(60)
AS
DECLARE @I_ID INT
SET @I_ID = (
  SELECT IngredientID
  FROM tblINGREDIENT
  WHERE IngredientName = @IngredientName
)
BEGIN TRANSACTION
INSERT INTO tblINGREDIENT (IngredientName)
VALUES (@IngredientName)


IF @@ERROR <> 0
 BEGIN
```

```
    PRINT 'rollback'
  ROLLBACK TRANSACTION
END
COMMIT TRANSACTION
GO


EXEC zsPopulateIngredient
@IngredientName = 'Chicken'


EXEC zsPopulateIngredient
@IngredientName = 'Beef'


EXEC zsPopulateIngredient
@IngredientName = 'Pork'


EXEC zsPopulateIngredient
@IngredientName = 'Corn'


EXEC zsPopulateIngredient
@IngredientName = 'Fish'


EXEC zsPopulateIngredient
@IngredientName = 'Soybean'


EXEC zsPopulateIngredient
@IngredientName = 'Eggs'


GO


--- Populate Supplier Table
CREATE PROCEDURE zsPopulateSupplier
@SupplierName varchar(60)
AS
DECLARE @SU_ID INT
SET @SU_ID = (
  SELECT SupplierID
  FROM tblSUPPLIER
  WHERE SupplierName = @SupplierName
```

```
)
BEGIN TRANSACTION
INSERT INTO tblSUPPLIER (SupplierName)
VALUES (@SupplierName)


IF @@ERROR <> 0
  BEGIN
     PRINT 'rollback'
  ROLLBACK TRANSACTION
END
COMMIT TRANSACTION
GO


EXEC zsPopulateSupplier
@SupplierName = 'Paws Dearie Meal'


EXEC zsPopulateSupplier
@SupplierName = 'Gentle Pet'


EXEC zsPopulateSupplier
@SupplierName = 'Precious Paw Diner'


EXEC zsPopulateSupplier
@SupplierName = 'Deary Spot Foods'


EXEC zsPopulateSupplier
@SupplierName = 'Fine Pet Diner Inc'


EXEC zsPopulateSupplier
@SupplierName = 'Tail Treats'


EXEC zsPopulateSupplier
@SupplierName = 'Furry Pet Food'


EXEC zsPopulateSupplier
@SupplierName = 'BFF Raw Food'
```

```
GO


-- No Younger Than 18 Customer
GO
CREATE FUNCTION zsNoBelow18()
RETURNS INT
AS
BEGIN


DECLARE @RET INT = 0
IF EXISTS (
  SELECT *
  FROM tblCUSTOMER C
  WHERE C.Birthdate > DateAdd(YEAR, -18, GetDate())
)
SET @RET = 1
RETURN @RET
END
GO

ALTER table tblORDER
ADD CONSTRAINT Order_No18_Purchase
CHECK(dbo.zsNoBelow18() = 0)


-- No more than 15 items in dog from customer with name starting with S
GO
CREATE FUNCTION zsNoSNameDogItem()
RETURNS INT
AS
BEGIN
DECLARE @RET INT = 0
IF EXISTS (
  SELECT *
  FROM tblCUSTOMER C
    JOIN tblORDER O ON C.CustomerID = O.CustomerID
    JOIN tblORDER_PRODUCT OP ON O.OrderID = OP.OrderID
    JOIN tblPRODUCT P ON OP.ProductID = P.ProductID
    JOIN tblPRODUCT_SPECIES PS ON P.ProductID = P.ProductID
    JOIN tblSPECIES S ON PS.ProductSpeciesID = S.SpeciesID
  WHERE C.Fname LIKE 'S%'
    AND S.SpeciesName = 'Dog'
    AND P.ProductID <= 15
)
SET @RET = 1
```

```
RETURN @RET
END
GO
ALTER table tblORDER
ADD CONSTRAINT Order_No15SName_Item
CHECK(dbo.zsNoSNameDogItem() = 0)


-- Order Total
GO
CREATE FUNCTION zsOrderTotal(@PK INT)
RETURNS NUMERIC(5,2)
AS
BEGIN
DECLARE @RET NUMERIC(5,2) = (
  SELECT SUM(ProdPrice)
  FROM tblPRODUCT P
    JOIN tblORDER_PRODUCT OP ON P.ProductID = OP.ProductID
    JOIN tblORDER O ON OP.OrderID = O.OrderID
  WHERE OP.OrderID = @PK
)
RETURN @RET
END
GO


ALTER TABLE tblORDER
ADD zsOrder_Calc AS (dbo.zsOrderTotal(OrderID))


-- Num of orders per customer
GO
CREATE or alter FUNCTION zsNumCustOrder(@PK INT)
RETURNS NUMERIC(4,2)
AS
BEGIN
DECLARE @RET NUMERIC(4,2) = (
  SELECT SUM(DISTINCT OrderID)
  FROM tblCUSTOMER C
    JOIN tblORDER O ON C.CustomerID = O.CustomerID
  WHERE O.CustomerID = @PK
)
RETURN @RET
END
GO
```

```
ALTER TABLE tblCUSTOMER
ADD zsNumOrder_Calc AS (dbo.zsNumCustOrder(CustomerID))


-- Return products for arthritis
SELECT P.ProdName, P.ProdPrice
FROM tblPRODUCT P
  JOIN tblPRODUCT_HEALTH_ISSUE PHI ON P.ProductID = PHI.ProductID
  JOIN tblHEALTH_ISSUE HI ON PHI.HealthIssueID = HI.HealthIssueID
WHERE HI.HealthIssueName = 'Arthritis'
GROUP BY P.ProdName, P.ProdPrice
ORDER BY ProdPrice DESC
-- Return foods with lowest ratings
SELECT P.ProdName, P.ProductID, AVG(R.Rating) AS AvgRating
FROM tblRATING R
    JOIN tblREVIEW RV ON R.RatingID = RV.RatingID
    JOIN tblORDER_PRODUCT OP ON RV.OrderProductID = OP.OrderProductID
    JOIN tblPRODUCT P ON OP.ProductID = P.ProductID
    join tblPRODUCT_TYPE pt on p.ProductTypeID = pt.ProductTypeID
WHERE Pt.ProdTypeName LIKE '%Food%'
GROUP BY P.ProdName, P.ProductID
ORDER BY AVG(R.Rating) ASC
```