```
-- BACKUP DATABASE INFO_430_Proj_02 TO DISK = 'C:\SQL\INFO_430_Proj_02.BAK'

-- DROP DATABASE INFO_430_Proj_02

-- BACKUP DATABASE INFO_430_Proj_02 to DISK =
'C:\sql\INFO_430_Group2_BackupDB.bak'
-- BACKUP DATABASE INFO_430_Proj_02 to DISK =
'C:\sql\INFO_430_Group2_BackupDB.bak' with differential
-- RESTORE FILELISTONLY from DISK = 'C:\sql\INFO_430_Group2_BackupDB.bak'
-- RESTORE HEADERONLY from DISK = 'C:\sql\INFO_430_Group2_BackupDB.bak'

-- RESTORE DATABASE Group_2DB_Backup from DISK =
'C:\sql\INFO_430_Group2_BackupDB.bak'
-- WITH
-- MOVE 'INFO_430_Proj_02' to 'C:\sql\Group_2DB_Backup.mdf',
-- MOVE 'INFO_430_Proj_02_log' to 'C:\sql\Group_2DB_Backup.ldf'

-- CREATE NONCLUSTERED INDEX IX_tblCUSTOMER_Birth_Lname_Fname ON
[dbo].[CUSTOMER]
-- (
--       [CustomerBirth] ASC,
--       [CustomerLName] ASC,
--       [CustomerFName] ASC
-- )
-- GO


CREATE DATABASE INFO_430_Proj_02
GO

USE INFO_430_Proj_02
GO

-- Abdiwahid Hajir
CREATE TABLE COURSE (
  CourseID INTEGER IDENTITY(1,1) Primary key,
  CourseName varchar(60) not null,
  CourseDescr varchar(500) NULL
)

-- Abdiwahid Hajir
CREATE TABLE CLASS (
  ClassID INTEGER IDENTITY(1,1) Primary key,
  InstructorID INTEGER not null,
  CourseID INTEGER not null,
  SchoolID INTEGER not null,
  ClassName varchar(60) not null
)

-- Abdiwahid Hajir
CREATE TABLE SCHOOL (
  SchoolID INTEGER IDENTITY(1,1) Primary key,
  SchoolName varchar(60) not null,
  SchoolDescr varchar(500) null,
  SchoolAddress varchar(250) not null
)

-- Abdiwahid Hajir
CREATE TABLE CLASS_DETAIL (
  ClassDetailID INTEGER IDENTITY(1,1) Primary key,
  ClassID INTEGER not null,
  DetailID INTEGER not null,
)

-- Abdiwahid Hajir
CREATE TABLE DETAIL (
  DetailID INTEGER IDENTITY(1,1) Primary key,
  ClassTime VARCHAR(50) not null,
  EndTime VARCHAR(50) not null,
  BeginTime VARCHAR(50) not null,
  ClassDuration VARCHAR(50) not null,
  ClassRoomNumber varchar(20) not null,
  ClassDescr varchar(500) null
)

-- Abdiwahid Hajir
CREATE TABLE [CLASSIFICATION](
  ClassificationID INTEGER IDENTITY(1,1) Primary key,
  ClassificationName varchar(60) not null,
  ClassificationDescr varchar(500) null
)

-- Abdiwahid Hajir
CREATE TABLE SEVERITY (
   SeverityID INT IDENTITY(1, 1) PRIMARY KEY,
   SeverityName VARCHAR(60) NOT NULL,
   SeverityDescr VARCHAR(255) NULL
```

```
)
GO

-- Russell Eng
CREATE TABLE CUSTOMER_ALLERGY (
   CustomerAllergyID INT IDENTITY(1, 1) PRIMARY KEY,
   AllergyID INT NOT NULL,
   SeverityID INT NOT NULL,
   CustomerID INT NOT NULL
)
GO

-- Russell Eng
CREATE TABLE ALLERGY (
   AllergyID INT IDENTITY(1, 1) PRIMARY KEY,
   AllergyName VARCHAR(60) NOT NULL,
   AllergyDescr VARCHAR(255) NULL
)
GO

-- Russell Eng
CREATE TABLE INGREDIENT (
   IngredientID INT IDENTITY(1, 1) PRIMARY KEY,
   IngredientName VARCHAR(60) NOT NULL,
   IngredientTypeID INT NOT NULL
)
GO

-- Russell Eng
CREATE TABLE INGREDIENT_TYPE (
   IngredientTypeID INT IDENTITY(1, 1) PRIMARY KEY,
   IngredientTypeName VARCHAR(60) NOT NULL,
   IngredientTypeDescr VARCHAR(255) NULL
)
GO

-- Russell Eng
CREATE TABLE INGREDIENT_RECIPE (
   IngredientRecipeID INT IDENTITY(1, 1) PRIMARY KEY,
   IngredientID INT NOT NULL,
   RecipeID INT NOT NULL
)
GO
```

```
-- Russell Eng
CREATE TABLE TASK (
   TaskID INT IDENTITY(1, 1) PRIMARY KEY,
   TaskName VARCHAR(255) NOT NULL,
   TaskDescr VARCHAR(255) NULL,
   TaskTime INT NULL
)
GO

 -- Russell Eng
CREATE TABLE CITY(
CityID Integer identity(1,1) primary key not null,
StateID INT NOT NULL,
CityName varchar(55) not null,
CityDescr varchar(80)
)
GO

-- Claire Li
CREATE TABLE [STATE](
StateID Integer identity(1,1) primary key not null,
StateName varchar(55) not null
)
GO

-- Claire Li
CREATE TABLE CUSTOMER(
CustomerID Integer identity(1,1) PRIMARY KEY not null,
CustomerFName varchar(55) not null,
CustomerLName varchar(55) not null,
CityID INT not null,
CustomerBirth DATE not null,
CustomerEmail varchar(55) not null,
CustomerDescr varchar(88) NULL
)
GO

-- Claire Li
CREATE TABLE REGISTRATION(
  RegisterID integer identity(1,1) PRIMARY KEY,
  RegistrationDATE DATE not null,
  GRADE Numeric(10,2) not null,
  RegstrationFEE Numeric(10,2) not null,
  CustomerID INT not null,
   ClassID INT not null

)
GO

-- Claire Li
CREATE TABLE COMMENT(
CommentID integer identity(1,1) PRIMARY KEY,
RecipeID INT NOT NULL,
CommentContent varchar(500) not null
)
GO

-- Lei Lei
CREATE TABLE DIFFICULTY(
   DifficultyID integer identity(1,1) PRIMARY KEY,
   DifficultyLevel  INTEGER not null,
   DifficultyLevelDecr varchar(250) NULL
)
GO

-- Claire Li
CREATE TABLE INSTRUCTOR (
   InstructorID INT IDENTITY(1,1) PRIMARY KEY,
   InstructorTypeID INT NOT NULL,
   InstrFname VARCHAR(50) NOT NULL,
   InstrLname VARCHAR(50) NOT NULL,
   InstrBirth DATE NOT NULL,
   InstrEmail VARCHAR(50) NOT NULL
)
GO

-- Lei Lei
CREATE TABLE INSTRUCTOR_TYPE (
   InstructorTypeID INT IDENTITY(1,1) PRIMARY KEY,
   InstructorTypeName VARCHAR(50) NOT NULL,
   InstructorTypeDescr VARCHAR(500)  NULL
)
GO

-- Lei Lei
CREATE TABLE CLASS_DISH (
   ClassDishID INT IDENTITY(1,1) PRIMARY KEY,
   ClassID INT NOT NULL,
  DishID INT NOT NULL
)

-- Lei Lei
CREATE TABLE DISH (
  DishID INT IDENTITY(1,1) PRIMARY KEY,
  DishTypeID INT NOT NULL,
  DishName VARCHAR(50) NOT NULL
)

-- Lei Lei
CREATE TABLE DISH_TYPE (
  DishTypeID INT IDENTITY(1,1) PRIMARY KEY,
  DishType VARCHAR(50) NOT NULL
)

-- Lei Lei
CREATE TABLE RECIPE (
  RecipeID INT IDENTITY(1,1) PRIMARY KEY,
  DishID INT NOT NULL,
  ClassificationID INT NOT NULL,
  DifficultyID INT NOT NULL,
  RecipeName VARCHAR(50) NOT NULL,
  TimeNeed TIME NOT NULL,
)

-- Lei Lei
CREATE TABLE RECIPE_TASK (
  RecipeTaskID INT IDENTITY(1,1) PRIMARY KEY,
  RecipeID INT NOT NULL,
  TaskID INT NOT NULL,
  [Sequence] VARCHAR(50) NOT NULL
)



-- Lei Lei
ALTER TABLE CLASS
ADD CONSTRAINT FK_CourseID
FOREIGN KEY (CourseID)
REFERENCES COURSE(CourseID)
GO

-- Lei Lei
```

```
ALTER TABLE CLASS
ADD CONSTRAINT FK_InstructorID
FOREIGN KEY (InstructorID)
REFERENCES INSTRUCTOR(InstructorID)
GO

 -- Lei Lei
ALTER TABLE CLASS
ADD CONSTRAINT FK_SchoolID
FOREIGN KEY (SchoolID)
REFERENCES SCHOOL(SchoolID)
GO

-- Lei Lei
ALTER TABLE CLASS_DETAIL
ADD CONSTRAINT FK_ClassID
FOREIGN KEY (ClassID)
REFERENCES CLASS(ClassID)
GO

-- Lei Lei
ALTER TABLE CLASS_DETAIL
ADD CONSTRAINT FK_DetailID
FOREIGN KEY (DetailID)
REFERENCES DETAIL(DetailID)
GO

-- Claire Li
ALTER TABLE CUSTOMER_ALLERGY
ADD CONSTRAINT FK_AllergyID
FOREIGN KEY (AllergyID)
REFERENCES ALLERGY (AllergyID)
GO

-- Claire Li
ALTER TABLE CUSTOMER_ALLERGY
ADD CONSTRAINT FK_SeverityID
FOREIGN KEY (SeverityID)
REFERENCES SEVERITY (SeverityID)
GO

 -- Claire Li
ALTER TABLE CUSTOMER_ALLERGY
ADD CONSTRAINT FK_CustomerID_Allergy
```

```
FOREIGN KEY (CustomerID)
REFERENCES CUSTOMER (CustomerID)
GO

-- Claire Li
ALTER TABLE INGREDIENT
ADD CONSTRAINT FK_IngredTypeID
FOREIGN KEY (IngredientTypeID)
REFERENCES INGREDIENT_TYPE(IngredientTypeID)
GO

-- Claire Li
ALTER TABLE INGREDIENT_RECIPE
ADD CONSTRAINT FK_IngredientID
FOREIGN KEY (IngredientID)
REFERENCES INGREDIENT(IngredientID)
GO

-- Claire Li
ALTER TABLE INGREDIENT_RECIPE
ADD CONSTRAINT FK_RecipeID_INGREDIENT
FOREIGN KEY (RecipeID)
REFERENCES RECIPE(RecipeID)
GO

-- Claire Li
ALTER TABLE INSTRUCTOR
ADD CONSTRAINT FK_InstructorTypeID
FOREIGN KEY (InstructorTypeID)
REFERENCES INSTRUCTOR_TYPE (InstructorTypeID)
GO

-- Russell Eng
ALTER TABLE CLASS_DISH
ADD CONSTRAINT FK_ClassID_23
FOREIGN KEY (ClassID)
REFERENCES CLASS (ClassID)
GO

-- Russell Eng
ALTER TABLE CLASS_DISH
ADD CONSTRAINT FK_DishID
FOREIGN KEY (DishID)
REFERENCES DISH (DishID)
```

```
GO

-- Russell Eng
ALTER TABLE DISH
ADD CONSTRAINT FK_DishTypeID
FOREIGN KEY (DishTypeID)
REFERENCES DISH_TYPE (DishTypeID)
GO

-- Russell Eng
ALTER TABLE RECIPE
ADD CONSTRAINT FK_DishID_33
FOREIGN KEY (DishID)
REFERENCES DISH (DishID)
GO

-- Russell Eng
ALTER TABLE RECIPE
ADD CONSTRAINT FK_ClassificationID
FOREIGN KEY (ClassificationID)
REFERENCES [CLASSIFICATION] (ClassificationID)
GO

-- Russell Eng
ALTER TABLE RECIPE
ADD CONSTRAINT FK_DifficultyID_33
FOREIGN KEY (DifficultyID)
REFERENCES DIFFICULTY (DifficultyID)
GO

-- Russell Eng
ALTER TABLE RECIPE_TASK
ADD CONSTRAINT FK_RecipeID_RECIPE_TASK
FOREIGN KEY (RecipeID)
REFERENCES RECIPE (RecipeID)
GO

-- Abdiwahid Hajir
ALTER TABLE RECIPE_TASK
ADD CONSTRAINT FK_TaskID
FOREIGN KEY (TaskID)
REFERENCES TASK (TaskID)
GO
```

```
-- Abdiwahid Hajir
ALTER TABLE CUSTOMER
ADD CONSTRAINT CityID
FOREIGN KEY(CityID) REFERENCES CITY(CityID)
GO

-- Abdiwahid Hajir
ALTER TABLE REGISTRATION
ADD CONSTRAINT CustomerID
FOREIGN KEY(CustomerID) REFERENCES CUSTOMER(CustomerID)
GO

 -- Abdiwahid Hajir
ALTER TABLE REGISTRATION
ADD CONSTRAINT ClassID_FK
FOREIGN KEY (ClassID)  REFERENCES CLASS(ClassID)
GO

-- Abdiwahid Hajir
ALTER TABLE CITY
ADD CONSTRAINT FK_StateID_2
FOREIGN KEY(StateID) REFERENCES [STATE](StateID)
GO

-- Abdiwahid Hajir
ALTER TABLE COMMENT
ADD CONSTRAINT FK_RecipeIDD
FOREIGN KEY (RecipeID) REFERENCES RECIPE(RecipeID)
GO



------------ GetID Procedures
-- Russell Eng
CREATE PROCEDURE Russ_Get_SeverityID
@Sev_Name VARCHAR(60),
@Sev_ID INT OUTPUT

AS

SET @Sev_ID = (SELECT SeverityID FROM SEVERITY WHERE SeverityName =
@Sev_Name)
GO
```

```
-- Russell Eng
CREATE PROCEDURE Russ_Get_AllergyID
@Allergy_Name VARCHAR(60),
@Allergy_ID INT OUTPUT

AS

SET @Allergy_ID = (SELECT AllergyID FROM ALLERGY WHERE AllergyName =
@Allergy_Name)
GO

 -- Russell Eng
CREATE PROCEDURE Russ_Get_Ingred_TypeID
@IngredType_Name VARCHAR(60),
@IngredType_ID INT OUTPUT

AS

SET @IngredType_ID = (SELECT IngredientTypeID FROM INGREDIENT_TYPE WHERE
IngredientTypeName = @IngredType_Name)
GO

 -- Russell Eng
CREATE PROCEDURE Russ_Get_IngredID
@Ingred_Name VARCHAR(60),
@Ingred_ID INT OUTPUT

AS

SET @Ingred_ID = (SELECT IngredientID FROM INGREDIENT WHERE IngredientName =
@Ingred_Name)
GO

 -- Russell Eng
CREATE PROCEDURE Russ_Get_TaskID
@Task_Name VARCHAR(255),
@Time INT,
@Task_ID INT OUTPUT

AS

SET @Task_ID = (SELECT TaskID FROM TASK WHERE TaskName = @Task_Name AND
TaskTime = @Time)
GO
```

```
---- Claire Li
CREATE PROCEDURE zixinl07_Get_InstructorID
@Fname VARCHAR(50),
@Lname VARCHAR(50),
@Birthy DATE,
@InstrID INT OUTPUT

AS

SET @InstrID = (SELECT InstructorID FROM INSTRUCTOR WHERE InstrFname = @Fname
AND InstrLname = @Lname AND InstrBirth = @Birthy)
GO

---- Claire Li
CREATE PROCEDURE zixinl07_Get_InstrTypeID
@Name VARCHAR(50),
@TypeID INT OUTPUT

AS

SET @TypeID = (SELECT InstructorTypeID FROM INSTRUCTOR_TYPE WHERE
InstructorTypeName = @Name)
GO

---- Claire Li
CREATE PROCEDURE zixinl07_Get_DishID
@DName VARCHAR(50),
@DID INT OUTPUT

AS

SET @DID = (SELECT DishID FROM DISH WHERE DishName = @DName)
GO

---- Claire Li
CREATE PROCEDURE zixinl07_Get_DishTypeID
@DName VARCHAR(50),
@DTID INT OUTPUT

AS

SET @DTID = (SELECT DishTypeID FROM DISH_TYPE WHERE DishType = @DName)
GO
```

```
--- Lei Lei
CREATE PROCEDURE leil_Get_CourseID
@Cou_Name VARCHAR(60),
@Cou_ID INT OUTPUT
AS
SET @Cou_ID = (SELECT CourseID FROM COURSE WHERE CourseName = @Cou_Name)
GO

CREATE PROCEDURE leil_Get_SchoolID
@Sch_Name VARCHAR(60),
@Sch_ID INT OUTPUT
AS
SET @Sch_ID = (SELECT SchoolID FROM SCHOOL WHERE SchoolName = @Sch_Name)
GO

--- Lei Lei
CREATE PROCEDURE leil_Get_DetailID
@Cla_Time VARCHAR(60),
@End_Time VARCHAR(60),
@Room_Number varchar(20),
@Det_ID INT OUTPUT
AS
SET @Det_ID = (SELECT DetailID FROM DETAIL WHERE ClassTime = @Cla_Time AND
EndTime = @End_Time AND ClassRoomNumber = @Room_Number)
GO

--- Lei Lei
CREATE PROCEDURE leil_Get_ClassificationID
@Classif_Name VARCHAR(60),
@Classif_ID INT OUTPUT
AS
SET @Classif_ID = (SELECT ClassificationID FROM CLASSIFICATION WHERE
ClassificationName = @Classif_Name)
GO

--- Lei Lei
CREATE PROCEDURE leil_Get_ClassID
@Ins_get_Fname VARCHAR(50),
@Ins_get_Lname VARCHAR(50),
@Ins_get_Birthy DATE,

@Cou_get_Name VARCHAR(60),
@Sch_get_Name VARCHAR(60),

@Class_get_Name VARCHAR(60),
@C_ID INT OUTPUT

AS

DECLARE @I_ID INT, @Co_ID INT, @S_ID INT

EXEC zixinl07_Get_InstructorID
@Fname = @Ins_get_Fname,
@Lname = @Ins_get_Lname ,
@Birthy = @Ins_get_Birthy,
@InstrID = @I_ID OUTPUT
IF @I_ID IS NULL
  BEGIN
     PRINT '@InstructorID is null. Check spelling.';
     THROW 53100, '@InstructorID cannot be null. Process Terminating', 1;
  END
EXEC leil_Get_CourseID
@Cou_Name = @Cou_get_Name,
@Cou_ID = @Co_ID OUTPUT
IF @Co_ID IS NULL
  BEGIN
     PRINT '@CourseID is null. Check spelling.';
     THROW 53101, '@CourseID cannot be null. Process Terminating', 1;
  END

EXEC leil_Get_SchoolID
@Sch_Name = @Sch_get_Name ,
@Sch_ID = @S_ID OUTPUT
IF @S_ID IS NULL
  BEGIN
     PRINT '@SchoolID is null. Check spelling.';
     THROW 53102, '@SchoolID cannot be null. Process Terminating', 1;
  END

SET @C_ID = (SELECT ClassID FROM CLASS
        WHERE CourseID = @Co_ID
        AND ClassName = @Class_get_Name
        AND InstructorID = @I_ID
        AND SchoolID=@S_ID)
GO
```

```
-- Abdiwahid Hajir
CREATE PROCEDURE get_State_Name
@State_Name2 varchar(45),
@State_ID INTEGER OUTPUT
AS
SET @State_ID = (Select StateID FROM [STATE] WHERE StateName = @State_Name2)
GO

CREATE PROCEDURE get_CityID
@City_Name2 varchar(60),
@State_Name VARCHAR(60),
@City_ID Integer OUTPUT
AS
SET @City_ID = (Select CityID FROM CITY C
                JOIN [STATE] S ON S.StateID = C.StateID
                WHERE CityName = @City_Name2
                AND StateName = @State_Name)
GO

-- Abdiwahid Hajir
CREATE PROCEDURE Get_RecipeID
@Recipe_Name2 varchar(70),
@RECIPE_ID2 INT OUTPUT
AS
SET @RECIPE_ID2 = (Select RecipeID FROM  RECIPE WHERE RecipeName =
@Recipe_Name2)
GO

-- Claire Li
CREATE PROCEDURE zixinl07_Get_DiffID
@Level INTEGER,
@DiffID INT OUTPUT
AS
SET @DiffID = (SELECT DifficultyID FROM DIFFICULTY WHERE DifficultyLevel = @Level)
GO

-- Abdiwahid Hajir
CREATE PROCEDURE Look_Up_Instructor_ID
@InstrFname_2 varchar(80),
@InstrLname_2 varchar(80),
@InstrBirth_2 DATE,
@Inst_ID_2 INT OUTPUT
AS
```

```
Set @Inst_ID_2 = (Select InstructorID FROM INSTRUCTOR WHERE InstrFname =
@InstrFname_2
                AND InstrLname = @InstrLname_2 AND InstrBirth = @InstrBirth_2)

 GO

CREATE PROCEDURE Russ_Get_CustID
@F VARCHAR(60),
@L VARCHAR(60),
@Birth DATE,
@CID INT OUTPUT
AS
SET @CID = (SELECT CustomerID FROM CUSTOMER WHERE CustomerFName = @F AND
CustomerLName = @L AND CustomerBirth = @Birth)
GO

-- Abdiwahid Hajir
CREATE PROCEDURE Look_Up_SchoolID
@SchoolName_2 varchar(60),
@School_ID2 INT OUTPUT
AS
SET @School_ID2 = (Select SchoolID FROM SCHOOL WHERE SchoolName =
@SchoolName_2)

 GO

-- Abdiwahid Hajir
CREATE PROCEDURE Look_Up_CourseID
@Course_Name_2 varchar(80),
@Course_ID_2 INT OUTPUT
AS
SET @Course_ID_2 = (Select CourseID FROM COURSE WHERE CourseName =
@Course_Name_2)
 GO

 -- Abdiwahid Hajir
CREATE PROCEDURE Get_Detail_ID
@Class_Time_2 varchar(80),
@END_TIME_2 varchar(80),
@Begin_Time_2 varchar(80),
@Class_Room_Number_2 varchar(80),
@DETAIL_ID_2 INT OUTPUT
AS
SET @DETAIL_ID_2 = (Select DetailID FROM DETAIL WHERE BeginTime = @Begin_Time_2
                 AND EndTime = @END_TIME_2 AND ClassRoomNumber =
@Class_Room_Number_2)
 GO

 -- Abdiwahid Hajir
CREATE PROCEDURE Look_UP_CLASS_Tables
@ClassName_2 varchar(60),
@InstrFnamy varchar(80),
@InstrLnamy varchar(80),
@InstrBirthy DATE,
@Course_Namey varchar(80),
@Class_IDDS INTEGER OUTPUT,
@Course_IDDS INTEGER
AS
DECLARE @Instructor_IDD INT
EXEC Look_Up_Instructor_ID
@InstrFname_2 = @InstrFnamy,
@InstrLname_2 = @InstrLnamy,
@InstrBirth_2 = @InstrBirthy,
@Inst_ID_2 = @Instructor_IDD OUTPUT
   IF @Instructor_IDD IS NULL
      BEGIN
         PRINT '@Instructor_IDD is null Check spelling';
         THROW 59099, '@Instructor_IDD cannot be null. WE are
         Terminating the process',1;
      END
EXEC Look_Up_CourseID
@Course_Name_2 = @Course_Namey,
@Course_ID_2 = @Course_IDDS OUTPUT
      IF @Instructor_IDD IS NULL
            BEGIN
               PRINT '@Instructor_IDD is null Check spelling';
               THROW 56777, '@Instructor_IDD cannot be null. WE are
               Terminating the process',1;
            END
SET @Class_IDDS = (Select ClassID FROM CLASS WHERE InstructorID = @Instructor_IDD
                 AND ClassName = @ClassName_2 AND CourseID = @Course_IDDS)

 GO

-- Russell Eng
CREATE PROCEDURE Russ_Get_ClassID
 @Class_Name VARCHAR(255),
 @Class_ID INT OUTPUT
 AS
 SET @Class_ID = (SELECT ClassID FROM CLASS WHERE ClassName = @Class_Name)
 GO




------ Insert Stored Procedure
-- Russell Eng
CREATE PROCEDURE Russ_Insert_Severity
@Severity_Name VARCHAR(60),
@Severity_Descr VARCHAR(255)
AS
BEGIN TRANSACTION T1
INSERT INTO SEVERITY(SeverityName, SeverityDescr)
VALUES (@Severity_Name, @Severity_Descr)
IF @@ERROR <> 0
   BEGIN
      PRINT 'Insert into SEVERITY failed... Rolling back'
      ROLLBACK TRANSACTION T1
   END
COMMIT TRANSACTION
GO

-- Russell Eng
CREATE PROCEDURE Russ_Insert_Allergy
@A_Name VARCHAR(60),
@A_Descr VARCHAR(255)
AS
BEGIN TRANSACTION T1
INSERT INTO ALLERGY(AllergyName, AllergyDescr)
VALUES(@A_Name, @A_Descr)
IF @@ERROR <> 0
   BEGIN
      PRINT 'Insert into ALLERGY failed... Rolling back'
      ROLLBACK TRANSACTION T1
   END
COMMIT TRANSACTION
GO

-- Russell Eng
CREATE PROCEDURE Russ_Insert_IngreType
@IngreTypeName VARCHAR(60),
@IngreTypeDescr VARCHAR(255)
```

```
AS
BEGIN TRANSACTION T1
INSERT INTO INGREDIENT_TYPE(IngredientTypeName, IngredientTypeDescr)
VALUES(@IngreTypeName, @IngreTypeDescr)
IF @@ERROR <> 0
   BEGIN
      PRINT 'Insert into INGREDIENT_TYPE failed... Rolling back'
      ROLLBACK TRANSACTION T1
   END
COMMIT TRANSACTION
GO

-- Russell Eng
CREATE PROCEDURE Russ_Insert_Ingre
@Ingre_Name VARCHAR(60),
@Ingre_Type_Name VARCHAR(60)
AS
DECLARE @IngreTypeID INT
EXEC Russ_Get_Ingred_TypeID
@IngredType_Name = @Ingre_Type_Name,
@IngredType_ID = @IngreTypeID OUTPUT
IF @IngreTypeID IS NULL
   BEGIN
      PRINT '@IngreTypeID is null. Check spelling.';
      THROW 53100, '@IngreTypeID cannot be null. Process Terminating', 1;
   END

BEGIN TRANSACTION T1
INSERT INTO INGREDIENT(IngredientName, IngredientTypeID)
VALUES(@Ingre_Name, @IngreTypeID)
IF @@ERROR <> 0
   BEGIN
      PRINT 'Insert into INGREDIENT failed... Rolling back'
      ROLLBACK TRANSACTION T1
   END
COMMIT TRANSACTION T1
GO

-- Russell Eng
CREATE PROCEDURE Russ_Insert_Task
@Task_Name VARCHAR(255),
@Task_Descr VARCHAR(255),
@Task_Time INT
AS
```

```
BEGIN TRANSACTION T1
INSERT INTO TASK(TaskName, TaskDescr, TaskTime)
VALUES(@Task_Name, @Task_Descr,@Task_Time)
IF @@ERROR <> 0
   BEGIN
      PRINT 'Insert into TASK failed... Rolling back'
      ROLLBACK TRANSACTION T1
   END
COMMIT TRANSACTION T1
GO

-- Claire Li
CREATE PROCEDURE zixinl07_Insert_Instructor_Type
@TypeName VARCHAR(50),
@TypeDescr VARCHAR(255)
AS
BEGIN TRANSACTION T1
INSERT INTO INSTRUCTOR_TYPE(InstructorTypeName, InstructorTypeDescr)
VALUES(@TypeName, @TypeDescr)
IF @@ERROR <> 0
  BEGIN
     PRINT 'Insert into INSTRUCTOR_TYPE failed... Rolling back'
     ROLLBACK TRANSACTION T1
  END
COMMIT TRANSACTION
GO

-- Abdiwahid Hajir
CREATE PROCEDURE Update_Comment
@Recipe_Name varchar(65),
@Comment_Content varchar(65)
AS
DECLARE @Recipe_ID INTEGER
EXEC Get_RecipeID
@Recipe_Name2 = @Recipe_Name,
@RECIPE_ID2 = @Recipe_ID  OUTPUT
        IF @Recipe_ID IS NULL
                BEGIN
                        PRINT '@Recipe_ID is null Check spelling';
                        THROW 58767, '@Recipe_ID cannot be null. WE are
                        Terminating the process',1;
                END
BEGIN TRANSACTION AH
INSERT INTO COMMENT(CommentContent, RecipeID)
```

```
VALUES(@Comment_Content, @Recipe_ID)
COMMIT TRANSACTION AH
GO

 -- Claire Li
CREATE PROCEDURE zixinl07_Insert_Instructor
@Fname VARCHAR(50),
@Lname VARCHAR(50),
@Birth DATE,
@Email VARCHAR(50),
@InstTypeName VARCHAR(50)
AS
DECLARE @InstrTypeID INT
EXEC zixinl07_Get_InstrTypeID
@Name = @InstTypeName,
@TypeID = @InstrTypeID OUTPUT

IF @InstrTypeID IS NULL
  BEGIN
     PRINT '@InstrTypeID is null. Check spelling.';
     THROW 54100, '@InstrTypeID cannot be null. Process Terminating', 1;
  END

BEGIN TRANSACTION T1
INSERT INTO INSTRUCTOR(InstructorTypeID, InstrFname, InstrLname, InstrBirth, InstrEmail)
VALUES (@InstrTypeID, @Fname, @Lname, @Birth, @Email)
IF @@ERROR <> 0
  BEGIN
     PRINT 'Insert into INSTRUCTOR failed... Rolling back'
     ROLLBACK TRANSACTION T1
  END
COMMIT TRANSACTION
GO

 -- Claire Li
CREATE PROCEDURE zixinl07_Insert_Dish_Type
@Type VARCHAR(50)
AS
BEGIN TRANSACTION T1
INSERT INTO DISH_TYPE(DishType)
VALUES(@Type)
IF @@ERROR <> 0
  BEGIN
     PRINT 'Insert into DISH_TYPE failed... Rolling back'
```

```
      ROLLBACK TRANSACTION T1
   END
COMMIT TRANSACTION
GO

 -- Claire Li
CREATE PROCEDURE zixinl07_Insert_Dish
@Name VARCHAR(50),
@DishTypeName VARCHAR(50)
AS
DECLARE @DTypeID INT

EXEC zixinl07_Get_DishTypeID
@DName = @DishTypeName,
@DTID = @DTypeID OUTPUT

IF @DTypeID IS NULL
   BEGIN
      PRINT '@DTypeID is null. Check spelling.';
      THROW 54101, '@DTypeID cannot be null. Process Terminating', 1;
   END

BEGIN TRANSACTION T1
INSERT INTO DISH(DishTypeID, DishName)
VALUES (@DTypeID, @Name)
IF @@ERROR <> 0
   BEGIN
      PRINT 'Insert into DISH failed... Rolling back'
      ROLLBACK TRANSACTION T1
   END
COMMIT TRANSACTION
GO

-- Lei Lei
CREATE PROCEDURE leil_Insert_Course
@Course_Name VARCHAR(60),
@Course_Descr VARCHAR(500)
AS
BEGIN TRANSACTION T1
INSERT INTO COURSE(CourseName, CourseDescr)
VALUES(@Course_Name, @Course_Descr)
IF @@ERROR <> 0
   BEGIN
      PRINT 'Insert into Course failed... Rolling back'
```

```
    ROLLBACK TRANSACTION T1
  END
COMMIT TRANSACTION T1
GO
-- insert class
CREATE PROCEDURE leil_Insert_Class
@InsFname VARCHAR(50),
@InsLname VARCHAR(50),
@InsBirthy DATE,

@CouName VARCHAR(60),

@SchName VARCHAR(60),

@Class_Name VARCHAR(60)
AS
DECLARE @In_ID INT, @Course_ID INT, @School_ID INT

EXEC zixinl07_Get_InstructorID
@Fname = @InsFname,
@Lname = @InsLname,
@Birthy = @InsBirthy,
@InstrID = @In_ID OUTPUT
IF @In_ID IS NULL
  BEGIN
    PRINT '@InstructorID is null. Check spelling.';
    THROW 53100, '@InstructorID cannot be null. Process Terminating', 1;
  END
EXEC leil_Get_CourseID
@Cou_Name = @CouName,
@Cou_ID = @Course_ID OUTPUT
IF @Course_ID IS NULL
  BEGIN
    PRINT '@CourseID is null. Check spelling.';
    THROW 53101, '@CourseID cannot be null. Process Terminating', 1;
  END

EXEC leil_Get_SchoolID
@Sch_Name = @SchName,
@Sch_ID = @School_ID OUTPUT
IF @School_ID IS NULL
  BEGIN
    PRINT '@SchoolID is null. Check spelling.';
    THROW 53102, '@SchoolID cannot be null. Process Terminating', 1;
  END

BEGIN TRANSACTION T1
INSERT INTO CLASS(InstructorID,  CourseID, SchoolID, ClassName)
VALUES(@In_ID, @Course_ID, @School_ID, @Class_Name)
IF @@ERROR <> 0
  BEGIN
     PRINT 'Insert into CLASS failed... Rolling back'
     ROLLBACK TRANSACTION T1
  END
COMMIT TRANSACTION T1
GO
-- end of insert class

-- Lei Lei
CREATE PROCEDURE Procedure_Ingredient_Recipe
@Recipe_Name varchar(200),
@Ingrident_Name varchar(100)
AS
DECLARE @Ingreident_ID INT, @Recipe_ID INT
EXEC Get_RecipeID
@Recipe_Name2 = @Recipe_Name,
@RECIPE_ID2 = @Recipe_ID OUTPUT
        IF @Recipe_ID IS NULL
                BEGIN
                PRINT '@RECIPE_ID2 cannot be null terminating process';
                THROW 57000, '@RECIPE_ID2 cannot be null check spelling
                We are termianting this process',1;
                END
EXEC Russ_Get_IngredID
@Ingred_Name = @Ingrident_Name,
@Ingred_ID = @Ingreident_ID OUTPUT
        IF @Ingreident_ID IS NULL
                BEGIN
                PRINT '@Ingreident_ID is null we are terminating this process';
                THROW 57001, '@RECIPE_ID2 cannot be null check spelling
                We are termianting this process',1;
                END
BEGIN TRANSACTION Insert_Recipe_Ingredents
INSERT INTO INGREDIENT_RECIPE(RecipeID, IngredientID)
VALUES( @Recipe_ID, @Ingreident_ID)
COMMIT TRANSACTION Insert_Recipe_Ingredents
 GO
```

```
CREATE PROCEDURE leil_Insert_School
@Sch_Name VARCHAR(60),
@Sch_Descr VARCHAR(500),
@Sch_Address varchar(250)

AS
BEGIN TRANSACTION T1
INSERT INTO SCHOOL(SchoolName, SchoolDescr, SchoolAddress)
VALUES(@Sch_Name, @Sch_Descr, @Sch_Address)
IF @@ERROR <> 0
   BEGIN
       PRINT 'Insert into School failed... Rolling back'
       ROLLBACK TRANSACTION T1
   END
COMMIT TRANSACTION T1
GO

-- Lei Lei
CREATE PROCEDURE leil_Insert_Detail
@Clas_Time VARCHAR(60),
@End_Class_Time VARCHAR(60),
@Begin_Class_Time VARCHAR(60),
@Clas_Duration VARCHAR(60),
@Clas_RoomNumber varchar(20),
@Clas_Descr varchar(500)
AS
BEGIN TRANSACTION T1
INSERT INTO DETAIL(ClassTime, EndTime, BeginTime, ClassDuration, ClassRoomNumber,
ClassDescr)
VALUES(@Clas_Time, @End_Class_Time, @Begin_Class_Time, @Clas_Duration,
@Clas_RoomNumber, @Clas_Descr)
IF @@ERROR <> 0
  BEGIN
      PRINT 'Insert into Detail failed... Rolling back'
      ROLLBACK TRANSACTION T1
  END
COMMIT TRANSACTION T1
GO


-- Abdiwahid Hajir
CREATE PROCEDURE Populate_CITY
@CityName varchar(45),
@StateName varchar(45),
@CityDescr varchar(88)
AS
DECLARE @StateID INT
EXEC get_State_Name
@State_Name2 = @StateName,
@State_ID = @StateID OUTPUT
   IF @StateID IS NULL
      BEGIN
      PRINT 'STATE ID CANNOT BE NULL check spelling';
      THROW 56666, 'StateID cannot be null termiantring process',1
      END


BEGIN TRANSACTION AH
INSERT INTO CITY(StateID, CityName, CityDescr)
VALUES (@StateID, @CityName, @CityDescr)
COMMIT TRANSACTION AH
GO

-- Abdiwahid Hajir
CREATE PROCEDURE Populate_State
@State_Name varchar(70)
AS

BEGIN TRANSACTION Insert_State
Insert into [STATE](StateName)
VALUES(@State_Name)
COMMIT TRANSACTION Insert_State
GO

CREATE PROCEDURE Populate_Customer
@Customer_Fname varchar(50),
@Customer_LName varchar(50),
@Customer_Birth DATE,
@Customer_Email varchar(60),
@Customer_Descr varchar(88),
@CityName varchar(70),
@State_Namy VARCHAR(60)
AS
DECLARE @City_ID INT
EXEC get_CityID
@City_Name2 = @CityName,
@State_Name = @State_Namy,
@City_ID = @City_ID OUTPUT
```

```
    IF @City_ID IS NULL
        BEGIN
        PRINT 'City ID cannot be null check spelling';
        THROW 57000, 'CityID cannot be null terminating process', 1
        END
BEGIN TRANSACTION Insert_Customer
INSERT INTO CUSTOMER(CustomerFName, CustomerLName, CustomerBirth,
CustomerEmail, CustomerDescr, CityID)
VALUES(@Customer_Fname, @Customer_LName, @Customer_Birth,
@Customer_Email,@Customer_Descr,@City_ID)
COMMIT TRANSACTION  Insert_Customer
GO

-- Abdiwahid Hajir
CREATE PROCEDURE populate_Comment
@CommentContent varchar(250),
@Recipe_name varchar(55)
AS
DECLARE @REC_ID INT
EXEC Get_RecipeID
@Recipe_Name2 = @Recipe_name,
@RECIPE_ID2 = @REC_ID OUTPUT
    IF @REC_ID IS NULL
        BEGIN
        PRINT 'Recipe ID cannot be null';
        THROW 578999, 'Cannot continue transaction because recipe id is null try again',1;
        END
BEGIN TRANSACTION AH
INSERT INTO COMMENT(CommentContent, RecipeID)
VALUES (@CommentContent, @REC_ID)
COMMIT TRANSACTION AH
GO

-- Lei Lei
CREATE PROCEDURE Populate_Difficulty
@Diff_Level INTEGER,
@Diff_Descr varchar(200)
AS
BEGIN TRANSACTION Insert_difficulty
insert into DIFFICULTY(DifficultyLevel, DifficultyLevelDecr)
VALUES(@Diff_level, @Diff_Descr)
COMMIT TRANSACTION Insert_difficulty
GO
```

```
-- Claire Li
CREATE PROCEDURE zixinl07_Insert_Recipe
@DishName VARCHAR(50),
@ClassifName VARCHAR(60),
@DiffLevel INT,
@ReName VARCHAR(50),
@Time TIME

AS
DECLARE @Dish_ID INT, @Classification_ID INT, @Diff_ID INT

EXEC zixinl07_Get_DishID
@DName = @DishName,
@DID = @Dish_ID OUTPUT
IF @Dish_ID IS NULL
  BEGIN
      PRINT '@DishID is null. Check spelling.';
      THROW 51600, '@DishID cannot be null. Process Terminating', 1;
  END

EXEC leil_Get_ClassificationID
@Classif_Name = @ClassifName,
@Classif_ID = @Classification_ID OUTPUT
IF @Classification_ID IS NULL
  BEGIN
      PRINT '@Classification_ID is null. Check spelling.';
      THROW 51601, '@Classification_ID cannot be null. Process Terminating', 1;
  END

EXEC zixinl07_Get_DiffID
@Level = @DiffLevel,
@DiffID = @Diff_ID OUTPUT
IF @Diff_ID IS NULL
  BEGIN
      PRINT '@DifficultyID is null. Check spelling.';
      THROW 51602, '@DifficultyID cannot be null. Process Terminating', 1;
  END

BEGIN TRANSACTION T1
INSERT INTO RECIPE(DishID, ClassificationID, DifficultyID, RecipeName, TimeNeed)
VALUES(@Dish_ID, @Classification_ID, @Diff_ID, @ReName, @Time)
IF @@ERROR <> 0
  BEGIN
      PRINT 'Insert into RECIPE failed... Rolling back'
      ROLLBACK TRANSACTION T1
  END
COMMIT TRANSACTION T1

GO

-- Claire Li
CREATE PROCEDURE zixinl07_Insert_Recipe_Task
@R_Name VARCHAR(50),
@T_Name VARCHAR(50),
@T_Time INT,
@S INT

AS
DECLARE @R_ID INT, @T_ID INT

EXEC Get_RecipeID
@Recipe_Name2 = @R_Name,
@RECIPE_ID2 = @R_ID OUTPUT
IF @R_ID IS NULL
 BEGIN
     PRINT '@R_ID is null. Check spelling.';
     THROW 51610, '@R_ID cannot be null. Process Terminating', 1;
 END

EXEC Russ_Get_TaskID
@Task_Name = @T_Name,
@Time = @T_Time,
@Task_ID = @T_ID OUTPUT
IF @T_ID IS NULL
 BEGIN
     PRINT '@T_ID is null. Check spelling.';
     THROW 51611, '@T_ID cannot be null. Process Terminating', 1;
 END

BEGIN TRANSACTION T1
INSERT INTO RECIPE_TASK(RecipeID, TaskID, [Sequence])
VALUES(@R_ID, @T_ID, @S)
IF @@ERROR <> 0
 BEGIN
     PRINT 'Insert into RECIPE_TASK failed... Rolling back'
     ROLLBACK TRANSACTION T1
 END
COMMIT TRANSACTION T1
```

```
GO

-- Russell Eng
CREATE PROCEDURE Russ_Insert_CUSTOMER_ALLERGY
@Allergy_N VARCHAR(60),
@Sever_N VARCHAR(60),
@Cust_F VARCHAR(60),
@Cust_L VARCHAR(60),
@Birthy DATE

AS

DECLARE @Aller_ID INT, @S_ID INT , @C_ID INT

EXEC Russ_Get_AllergyID
@Allergy_Name =@Allergy_N,
@Allergy_ID = @Aller_ID OUTPUT
IF @Aller_ID IS NULL
 BEGIN
     PRINT '@Aller_ID is null. Check spelling.';
     THROW 51602, '@Aller_ID cannot be null. Process Terminating', 1;
 END


EXEC Russ_Get_SeverityID
@Sev_Name = @Sever_N,
@Sev_ID = @S_ID OUTPUT
IF @S_ID IS NULL
 BEGIN
     PRINT '@S_ID is null. Check spelling.';
     THROW 51602, '@S_ID cannot be null. Process Terminating', 1;
 END

EXEC Russ_Get_CustID
@F = @Cust_F,
@L  = @Cust_L,
@Birth = @Birthy,
@CID  = @C_ID OUTPUT
IF @C_ID IS NULL
 BEGIN
     PRINT '@C_ID is null. Check spelling.';
     THROW 51602, '@C_ID cannot be null. Process Terminating', 1;
 END
```

```
BEGIN TRANSACTION Insert_Aller_Customer
INSERT INTO CUSTOMER_ALLERGY(AllergyID, SeverityID, CustomerID)
VALUES(@Aller_ID, @S_ID, @C_ID)
IF @@ERROR <> 0
 BEGIN
     PRINT 'Insert into CUSTOMER_ALLERGY failed... Rolling back'
     ROLLBACK TRANSACTION T1
 END
COMMIT TRANSACTION Insert_Aller_Customer
GO


-- Claire Li
CREATE PROCEDURE zixinl07_Insert_Class_Dish
@Ins_Fname VARCHAR(50),
@Ins_Lname VARCHAR(50),
@Ins_Birthy DATE,
@Cou_Name VARCHAR(60),
@Sch_Name VARCHAR(60),
@Class_Name VARCHAR(60),
@Dish_Name VARCHAR(50)

AS
DECLARE @CID INT, @D_ID INT

EXEC leil_Get_ClassID
@Ins_get_Fname = @Ins_Fname,
@Ins_get_Lname = @Ins_Lname,
@Ins_get_Birthy = @Ins_Birthy,
@Cou_get_Name = @Cou_Name,
@Sch_get_Name = @Sch_Name,
@Class_get_Name = @Class_Name,
@C_ID = @CID OUTPUT
IF @CID IS NULL
 BEGIN
     PRINT '@ClassID is null. Check spelling.';
     THROW 51612, '@ClassID cannot be null. Process Terminating', 1;
 END

EXEC zixinl07_Get_DishID
@DName = @Dish_Name,
@DID = @D_ID OUTPUT
IF @D_ID IS NULL
```

```
 BEGIN
    PRINT '@DishID is null. Check spelling.';
    THROW 51613, '@DishID cannot be null. Process Terminating', 1;
 END

BEGIN TRANSACTION T1
INSERT INTO CLASS_DISH(ClassID, DishID)
VALUES(@CID, @D_ID)
IF @@ERROR <> 0
 BEGIN
    PRINT 'Insert into CLASS_DISH failed... Rolling back'
    ROLLBACK TRANSACTION T1
 END
COMMIT TRANSACTION T1
GO

-- Abdiwahid Hajir
CREATE PROCEDURE Populate_Class_Table
@InstrFname varchar(80),
@InstrLname varchar(80),
@InstrBirth DATE,
@ClassName varchar(60),
@SchoolName varchar(60),
@CourseName varchar(70)
AS
DECLARE @Inst_ID INT, @School_Id INT, @CourseID INT
EXEC Look_Up_Instructor_ID
@InstrFname_2 = @InstrFname,
@InstrLname_2 = @InstrLname,
@InstrBirth_2 = @InstrBirth,
@Inst_ID_2 = @Inst_ID OUTPUT
   IF @Inst_ID IS NULL
      BEGIN
         PRINT '@Inst_ID is null Check spelling';
         THROW 55555, '@Inst_ID cannot be null. WE are
         Terminating the process',1;
      END
EXEC Look_Up_SchoolID
@SchoolName_2 = @SchoolName,
@School_ID2 = @School_Id OUTPUT
      IF @School_Id IS NULL
            BEGIN
               PRINT '@School_Id is null Check spelling';
               THROW 55588, '@School_Id cannot be null. WE are
```

```
                    Terminating the process',1;
                END
EXEC Look_Up_CourseID
@Course_Name_2 = @CourseName,
@Course_ID_2 = @CourseID OUTPUT
        IF @CourseID IS NULL
                BEGIN
                    PRINT '@CourseID is null Check spelling';
                    THROW 56789, '@CourseID cannot be null. WE are
                    Terminating the process',1;
                END
BEGIN TRANSACTION INSERT_INTO_CLass
Insert into CLASS(ClassName, InstructorID, SchoolID, CourseID)
VALUES(@ClassName, @Inst_ID, @School_Id, @CourseID)
COMMIT TRANSACTION INSERT_INTO_CLass
 GO

-- Lei Lei
CREATE PROCEDURE update_ClassDetail_Table
@InstrFname varchar(80),
@InstrLname varchar(80),
@InstrBirth DATE,
@ClassName varchar(60),
@SchoolName varchar(60),
@CourseName varchar(70),
@Class_Time varchar(80),
@END_TIME varchar(80),
@Begin_Time varchar(80),
@Class_Room_Number varchar(80)
AS
DECLARE @Class_ID INTEGER , @DETAIL_ID INT
EXEC Get_Detail_ID
@Class_Time_2 = @Class_Time,
@END_TIME_2 = @END_TIME,
@Begin_Time_2 = @Begin_Time,
@Class_Room_Number_2 = @Class_Room_Number,
@DETAIL_ID_2 = @DETAIL_ID OUTPUT
EXEC leil_Get_ClassID
@Ins_get_Fname = @InstrFname,
@Ins_get_Lname = @InstrLname,
@Ins_get_Birthy =  @InstrBirth,

@Cou_get_Name = @CourseName,
@Sch_get_Name = @SchoolName,

@Class_get_Name = @ClassName,
@C_ID = @Class_ID OUTPUT


BEGIN TRANSACTION INSERT_INTO_Class_DETAIL
INSERT INTO CLASS_DETAIL(ClassID, DetailID)
VALUES(@Class_ID, @DETAIL_ID)
COMMIT TRANSACTION INSERT_INTO_Class_DETAIL


GO

-- Claire Li
CREATE PROCEDURE zixinl07_insert_registration
@Firsty VARCHAR(60),
@Lasty VARCHAR(60),
@Birthy DATE,
@Ins_Fname VARCHAR(50),
@Ins_Lname VARCHAR(50),
@Ins_Birthy DATE,
@Cou_Name VARCHAR(60),
@Sch_Name VARCHAR(60),
@Class_Name VARCHAR(60),
@DATE DATE,
@Grade Numeric(10,2),
@Fee Numeric(10,2)

AS
DECLARE @Cust_ID INT, @Class_ID INT

EXEC Russ_Get_CustID
@F = @Firsty,
@L = @Lasty,
@Birth = @Birthy,
@CID = @Cust_ID OUTPUT

IF @Cust_ID IS NULL
 BEGIN
    PRINT '@Cust_ID is null. Check spelling.';
    THROW 51615, '@Cust_ID cannot be null. Process Terminating', 1;
 END
```

```
EXEC Ieil_Get_ClassID
@Ins_get_Fname = @Ins_Fname,
@Ins_get_Lname = @Ins_Lname,
@Ins_get_Birthy = @Ins_Birthy,
@Cou_get_Name = @Cou_Name,
@Sch_get_Name = @Sch_Name,
@Class_get_Name = @Class_Name,
@C_ID = @Class_ID OUTPUT

IF @Class_ID IS NULL
 BEGIN
    PRINT '@Class_ID is null. Check spelling.';
    THROW 51616, '@Class_ID cannot be null. Process Terminating', 1;
 END

BEGIN TRANSACTION T1
INSERT INTO REGISTRATION(CustomerID, ClassID, RegistrationDATE, GRADE,
RegstrationFEE)
VALUES(@Cust_ID, @Class_ID, @DATE, @Grade, @Fee)
IF @@ERROR <> 0
 BEGIN
    PRINT 'Insert into CLASS_DETAIL failed... Rolling back'
    ROLLBACK TRANSACTION T1
 END
COMMIT TRANSACTION T1
GO

-- Russell Eng
CREATE PROCEDURE Russ_Insert_Registration
@Firsty VARCHAR(60),
@Lasty VARCHAR(60),
@Birthy DATE,
@Class_Name VARCHAR(60),
@DATE DATE,
@Grade Numeric(10,2),
@Fee Numeric(10,2)
 AS
DECLARE @Cust_ID INT, @Class_ID INT

EXEC Russ_Get_CustID
@F = @Firsty,
@L = @Lasty,
@Birth = @Birthy,
@CID = @Cust_ID OUTPUT

IF @Cust_ID IS NULL
 BEGIN
     PRINT '@Cust_ID is null. Check spelling.';
     THROW 51615, '@Cust_ID cannot be null. Process Terminating', 1;
 END

 Exec Russ_Get_ClassID
 @Class_Name = @Class_Name,
 @Class_ID = @Class_ID OUTPUT
 IF @Class_ID IS NULL
 BEGIN
     PRINT '@Class_ID is null. Check spelling.';
     THROW 51617, '@Class_ID cannot be null. Process Terminating', 1;
 END
 BEGIN TRANSACTION T1
INSERT INTO REGISTRATION(CustomerID, ClassID, RegistrationDATE, GRADE,
RegstrationFEE)
VALUES(@Cust_ID, @Class_ID, @DATE, @Grade, @Fee)
IF @@ERROR <> 0
 BEGIN
     PRINT 'Insert into CLASS_DETAIL failed... Rolling back'
     ROLLBACK TRANSACTION T1
 END
COMMIT TRANSACTION T1
GO


--------------------- Populate tables-------------------------------------------
```

```
-- Russell Eng
INSERT INTO INGREDIENT_TYPE(IngredientTypeName,  IngredientTypeDescr)
VALUES ('Dairy', 'Eggs, milk and milk related products'), ('Oils', 'Including products such as
sunflower oil and peanut oil'),
('Fruits', 'Products such as apple and banana'), ('Nuts', 'Including Products such as almonds,
Brazil nuts, and cashew nuts'),
('Grains', 'Products such as rice and pasta'), ('Seasoning', 'Including products such as sugar and
salts'),
('Meat', 'Inclusing all products related to meat, inclusinf sancages, ham, and fish'), ('Vegetables',
'Products such as onion and pepper'),
('Other', 'All other products')
GO

-- Lei Lei
INSERT INTO DISH_TYPE(DishType)
VALUES ('Appetizers/starters'), ('Beans, Grains & Legumes'), ('Breads, Rolls & Muffins'),
('Burgers'), ('Cakes & Cupcakes'),
('Candy & Sweets'), ('Casseroles & Gratins'), ('Cocktails'), ('Cookies'), ('Desserts'), ('Dips &
Spreads'), ('Dressings'),
('Food Gifts'), ('Ice Cream & Sorbet'), ('Marinades & Rubs'), ('Nonalcoholic Drinks'), ('Pasta &
Noodles'), ('Pies & Tarts'),
('Pizzas'), ('Puddings & Custards'), ('Salads'), ('Sandwiches'), ('Sauces & Condiments'), ('Side
Dishes'), ('Soups & Stews'), ('Stuffings');

-- Claire Li
INSERT INTO TASK(TaskName, TaskDescr, TaskTime)
VALUES ('Slice', 'When a large ingredient — such as potatoes or onions — is cut into large, flat
pieces of a similar size', 3),
('Chop', 'Cut similar sized square pieces that are roughly half an inch in diameter', 3),
('Dice', 'Cut ingredients into small, square-shaped pieces', 3),
('Mince', 'The tiniest cut, run the knife over the ingredient in a back-and-forth motion until very
fine.', 3),
('Seasoning to taste', 'Refers to adjusting salt and pepper since everyone palates differ on how
salty a dish tastes ', 1),
('Add a dash', 'Add  roughly 1/8 teaspoon of seasoning', 1),
('Add a pinch', 'Add  around 1/16 teaspoon', 1),
('Add a smidgen', 'Add approximately 1/32 teaspoon.', 1),
('Bake and roast ', 'When preheating your oven, the air inside warms to a temperature of your
setting. This hot air cooks your food at an even rate by surrounding the roasting pan or baking
dish on all sides.', 10),
('Broil', 'Cooks the food only on one side (the top) at a very high heat inside oven', 10),
('Sauté', 'Quickly cook food over high heat. This cooking method often includes oil or fat to
evenly transfer the heat from the pan into the food.', 5),
('Sear', ' The food is cooked in a pan — often one piece at a time to avoid overcrowding — until
fully browned on each side, with no stirring', 1),
('Char', 'Charring is achieved by cooking in a very hot pan or grill grate on the stovetop.', 1),
('Deep fry', 'when the ingredient is fully submerged in hot oil.', 5),
('Pan fry', 'Pan has enough oil to come halfway up the side of what the chef is frying.', 5),
('Braise', 'Prepare tougher cuts of meat. In a large pot, the meat is browned on all sides. Then it
is covered with liquid and cooked low and slow until fall-off-the-bone tender.', 2),
('Boiling', 'When water is heated to 212 degrees F. This makes the water produce bubbles and
movement', 5),
('Simmering', 'When water, or other cooking liquids such as broth, are just below the boiling
point. ', 5),
('Poaching', 'Gently cooking ingredients in water', 5),
('Steaming', 'The ingredients are placed in a steamer basket held above the boiling water.', 10),
('Blanching', 'The food is dipped into boiling water for a small amount of time', 5)
GO


CREATE TABLE SEASON (
  SeasonID INTEGER IDENTITY(1,1) Primary key,
  SeasonName varchar(60) not null,
)
INSERT INTO SEASON (SeasonName)
VALUES ('Autumn'), ('Winter'), ('Spring'), ('Summer')
GO

CREATE TABLE BUILDING (
  BuildingID INTEGER IDENTITY(1,1) Primary key,
  BuildingName varchar(3) not null,
)
INSERT INTO BUILDING (BuildingName)
VALUES ('AGH'), ('QEH'), ('JDK'), ('PGH'), ('QAQ'), ('KFC')
GO

 -- Claire Li
CREATE PROCEDURE zixinl07_pop_detail
@RUN INT
AS
DECLARE  @RANDOM_season_ID INT, @RANDOM_season_Row INT, @RANDOM_Build_ID
INT, @RANDOM_Build_Row INT
DECLARE @CI_Time VARCHAR(50), @End_Time VARCHAR(50), @Begin_Time
VARCHAR(50), @Duration VARCHAR(50), @RoomNumber varchar(20)
DECLARE @Building VARCHAR(50), @Room VARCHAR(3), @Dur INT, @Begin INT, @End
INT
SET @RANDOM_season_Row = (SELECT COUNT(*) FROM SEASON)
SET @RANDOM_Build_Row = (SELECT COUNT(*) FROM BUILDING)
WHILE @RUN > 0
BEGIN
 SET @RANDOM_season_ID = (SELECT RAND() * @RANDOM_season_Row + 1)
 SET @RANDOM_Build_ID = (SELECT RAND() * @RANDOM_Build_Row + 1)
 SET @Cl_Time = (SELECT SeasonName FROM SEASON WHERE SeasonID =
@RANDOM_season_ID)
 SET @Begin = (SELECT FLOOR(RAND()*(13))+8)
 SET @Dur = (SELECT FLOOR(RAND()*(7))+1)
 SET @End = (SELECT @Begin + @Dur)
 SET @End_Time = (SELECT CAST(@End AS VARCHAR) + ':00')
 SET @Begin_Time = (SELECT CAST(@Begin AS VARCHAR) + ':00')
 SET @Duration = (SELECT CAST(@Dur AS VARCHAR) + ' hr')
 SET @Building = (SELECT BuildingName FROM BUILDING WHERE BuildingID =
@RANDOM_Build_ID)
 SET @Room = (SELECT FLOOR(RAND()*(401))+100)
 SET @RoomNumber = (SELECT @Building + ' ' + @Room)

EXEC leil_Insert_Detail
@Clas_Time = @Cl_Time,
@End_Class_Time = @End_Time,
@Begin_Class_Time = @Begin_Time,
@Clas_Duration = @Duration,
@Clas_RoomNumber = @RoomNumber,
@Clas_Descr = NULL
SET @RUN = @RUN - 1
END
GO

EXEC zixinl07_pop_detail 150
GO

-- Lei Lei
INSERT INTO CLASSIFICATION(ClassificationName, ClassificationDescr)
VALUES ('Dairy free', 'Made purely with non-perishable ingredients from the pantry. No eggs, no
butter.'), ('Fish and shellfish free', 'No seafood.'),
     ('Nut free', 'Does not contain nuts.'), ('Gluten free', 'No gluten.'), ('Vegetarian', 'Excludes
meat, poultry, fish and seafood.'),
     ('Vegans', 'Excludes all meat and animal products (meat, poultry, fish, seafood, dairy and
eggs).'),
     ('Healthy Pregnancy', 'Focus on essential nutrients.'),
     ('Halal', 'Cooking without the use of haram, or impermissible, ingredients according to
Islamic dietary guidelines.'),
    ('No classification', NULL)
GO

-- Lei Lei
INSERT INTO DIFFICULTY(DifficultyLevel, DifficultyLevelDecr)
```

```
VALUES ('1', 'The easiest of the levels. This level does not contain alcohol or require the use of
a heat source
      (stove, grill, etc) making it safe for children to prepare with limited assistance.'),
     ('2', 'Very easy but may require the use of a heat source and may contain alcohol.
      Easy to find ingredients and simple steps involved in making the recipe.
      Recipes can usually be made by people with little to no cooking experience.'),
      ('3', 'Average. Most recipes will require basic cooking skills.'),
      ('4', 'Above-average. These recipes either contain harder to find ingredients,
      more advanced cooking skills, or unusual tools needed to prepare the recipe.'),
      ('5', 'Difficult. These recipes usually contain unusual ingredients and
      require advanced cooking skills or elaborate preparations.')
GO

 -- Russell Eng
BEGIN TRANSACTION Pop_Severity
INSERT INTO  SEVERITY (SeverityName, SeverityDescr)
VALUES ('Mild', null), ('Moderate', null), ('Severe', null)
COMMIT TRANSACTION Pop_Severity
GO

 -- Russell Eng
BEGIN TRANSACTION Pop_Allergy
INSERT INTO ALLERGY(AllergyName, AllergyDescr)
VALUES ('Milk', null), ('Eggs', null), ('Peanuts', null), ('Tree nuts', null), ('Soy', null), ('Wheat',
null), ('Fish', null), ('Shellfish', null),
('Corn', null), ('Gelatin', null), ('Seeds', 'often sesame, sunflower, and poppy'), ('Spices', 'such as
caraway, coriander, garlic, and mustard')
COMMIT TRANSACTION Pop_Allergy
GO

-- Claire Li
INSERT INTO INSTRUCTOR_TYPE(InstructorTypeName, InstructorTypeDescr)
VALUES ('Instructor', 'The entry level rank for those who have recently completed their post
doctoral training'),
     ('Assistant Professor', 'Beginning-level professor'), ('Associate Professor', 'Mid-level
professor'), ('Professor', 'Senior-level professor')
GO

 -- Russell Eng
INSERT INTO SCHOOL(SchoolName, SchoolDescr, SchoolAddress)
VALUES ('Amazing Greg Hay Baking School', 'Amazing culinary school', 'Amazing street 9 3/4
Platform'),
('Seattle Pacific Cooking School', null, '535, Settle Ave'),
('University of Cooking Master', 'Master cooking in the speed of life', '5476 122nd NE Ave'),
('Easy Cooking! Ultimate Guide!', null, '133nd SE Ave')

GO

CREATE TABLE New_State_Table(
New_State_ID Integer identity(1,1) primary key,
New_State_Name varchar(50) not null,
)


INSERT INTO New_State_Table(New_State_Name)
SELECT StateName
FROM PEEPs.dbo.tblCITY_STATE_ZIP
WHERE StateName is not null


CREATE TABLE Update_State(
State_Id_New Integer identity(1,1) primary key,
State_Name_New varchar(60)

)
INSERT INTO [STATE](StateName)
SELECT DISTinct New_State_Name
FROM New_State_Table

CREATE TABLE New_City_Table(
New_City_ID Integer identity(1,1) primary key,
New_City_Name varchar(50) not null,
New_State_Name varchar(60) not null
)


INSERT INTO New_City_Table(New_City_Name, New_State_Name)
SELECT DISTINCT CityName, StateName
FROM PEEPs.dbo.tblCITY_STATE_ZIP

GO

-- Abdiwahid Hajir
CREATE PROCEDURE Update_Our_City_Table
AS
DECLARE @Rip_Data INTEGER, @City_Name_Data varchar(60),
@State_Name_Data varchar(60), @RUN INTEGER, @State_ID INTEGER
SET @Rip_Data =(Select TOP 1 New_City_ID FROM New_City_Table)
SET @RUN = (SELECT TOP 35000 COUNT(*)  FROM New_City_Table)
WHILE @RUN > 0
BEGIN
SET @City_Name_Data = (Select New_City_Name From New_City_Table WHERE
New_City_ID = @Rip_Data)
SET @State_Name_Data = (Select New_State_Name From New_City_Table WHERE
New_City_ID = @Rip_Data)
EXEC Populate_CITY
@CityName = @City_Name_Data,
@StateName = @State_Name_Data,
@CityDescr = NULL
DELETE FROM New_City_Table WHERE New_City_ID = @Rip_Data
SET @Rip_Data = (SELECT TOP 1 New_City_ID FROM New_City_Table)
SET @RUN = @RUN - 1
END


GO

EXEC Update_Our_City_Table

GO




IF EXISTS (SELECT * FROM sys.sysobjects WHERE NAME = '#JUNK_Customer_Table')
   BEGIN
      DROP TABLE WORKING_COPY_Pets
      CREATE TABLE #JUNK_Customer_Table(
      New_Customer_ID Integer identity(1,1) primary key,
      New_CustomerF_Name varchar(200) not null,
      New_CustomerL_Name varchar(200) not null,
      New_Customer_Birth DATE,
      New_Customer_Email varchar(200),
      New_Customer_Descr varchar(200),
      )
   END
GO
INSERT INTO #JUNK_Customer_Table(New_CustomerF_Name, New_CustomerL_Name,
New_Customer_Birth, New_Customer_Email)
SELECT TOP 980000 CustomerFname, CustomerLname, DateOfBirth, Email
FROM PEEPs.dbo.tblCUSTOMER
GO
```

```
 -- Abdiwahid Hajir
CREATE PROCEDURE Update_Customer

AS
DECLARE @Random_Customer_ID INTEGER, @City_ID INteger, @Random_City INTEGER,
@Random_Cust_Row INT, @RUN INT,
@Min_PK INT, @CustomerFname varchar(200), @CustomerLname varchar(200),
@DateOfBirth varchar(200), @Email varchar(200),
@City_Name varchar(200), @City_Random_name varchar(70), @Customer_Descr
varchar(200), @State_Name VARCHAR(60)
SET @Random_City =(SELECT COUNT(*) FROM CITY)
SET @RUN = (SELECT COUNT(*) FROM #JUNK_Customer_Table)
SET @Min_PK = (SELECT TOP 1 New_Customer_ID FROM #JUNK_Customer_Table)

WHILE @Run > 0
BEGIN
   SET @Random_Customer_ID = (Select RAND() * @Random_Cust_Row + 1)
   SET @City_ID = (Select RAND() * @Random_City + 1)
   SET @City_Random_name = (Select CityName from CITY WHERE CityID = @City_ID)
   SET @State_Name = (SELECT StateName FROM [STATE] S
                 JOIN CITY C ON C.StateID = S.StateID
                 WHERE CityID = @City_ID)

   SET @CustomerFname = (Select New_CustomerF_Name FROM #JUNK_Customer_Table
WHERE New_Customer_ID = @Min_PK)
   SET @CustomerLname = (Select New_CustomerL_Name FROM #JUNK_Customer_Table
WHERE New_Customer_ID = @Min_PK)
   SET @DateOfBirth = (Select New_Customer_Birth FROM #JUNK_Customer_Table WHERE
New_Customer_ID = @Min_PK)
   SET @Email = (Select New_Customer_Email FROM #JUNK_Customer_Table WHERE
New_Customer_ID = @Min_PK)
   EXEC Populate_Customer
   @Customer_Fname = @CustomerFname,
   @Customer_LName = @CustomerLname,
   @Customer_Birth = @DateOfBirth,
   @Customer_Email = @Email,
   @Customer_Descr = NULL,
   @CityName = @City_Random_name,
   @State_Namy = @State_Name

   DELETE FROM #JUNK_Customer_Table WHERE New_Customer_ID = @Min_PK
   SET @Min_PK = (SELECT TOP 1 New_Customer_ID FROM #JUNK_Customer_Table)
   SET @Run = @Run - 1
```

```
END
GO


EXEC Update_Customer
GO


CREATE TABLE Course_Name_Data(
   CourseID INT IDENTITY(1, 1) PRIMARY KEY,
   CourseName VARCHAR(60) NOT NULL
)
GO

INSERT INTO Course_Name_Data (CourseName)
SELECT *
FROM Group_2_Data.dbo.Sheet1$
GO

SELECT *
INTO Course_Name_Copy
FROM Course_Name_Data
GO

-- Russell Eng
CREATE PROCEDURE Russ_Pop_Course
AS

DECLARE @MIN INT, @Run INT
DECLARE @Descr VARCHAR(255) = NULL
DECLARE @Course_Name VARCHAR(60)


SET @RUN = (SELECT COUNT(*) FROM Course_Name_Data)
SET @MIN = (SELECT TOP 1 CourseID FROM Course_Name_Copy)
WHILE @RUN > 0
   BEGIN
   SET @MIN = (SELECT TOP 1 CourseID FROM Course_Name_Copy)
   SET @Course_Name = (SELECT CourseName FROM Course_Name_Copy WHERE
CourseID = @MIN)
   EXEC leil_Insert_Course
   @Course_Name = @Course_Name,
   @Course_Descr = @Descr
   DELETE FROM Course_Name_Copy WHERE CourseID = @MIN
   SET @RUN = @RUN - 1
   END
GO

EXEC Russ_Pop_Course
GO

-- Russell Eng
CREATE PROCEDURE Pop_Customer_Allergy
@RUN INT
AS
DECLARE @RandomCustID INT , @RandomAllergyID INT, @RandomSeverityID INT
DECLARE @Aller_Name VARCHAR(60), @Sev_Name VARCHAR(60), @C_F_name
VARCHAR(60),
@C_L_name VARCHAR(60), @Birthday DATE
DECLARE @C_Row INT = (SELECT COUNT(*) FROM CUSTOMER)
DECLARE @S_Row INT = (SELECT COUNT(*) FROM SEVERITY)
DECLARE @A_Row INT = (SELECT COUNT(*) FROM ALLERGY)

WHILE @RUN > 0
BEGIN

   SET @RandomCustID = (SELECT RAND() * @C_Row  + 1)
   SET @RandomAllergyID = (SELECT RAND() * @A_Row + 1)
   SET  @RandomSeverityID = (SELECT RAND() * @S_Row + 1)

   SET @C_F_name = (SELECT CustomerFName FROM CUSTOMER WHERE CustomerID =
@RandomCustID)
   WHILE @C_F_name IS NULL
   BEGIN
      SET @RandomCustID = (SELECT RAND() * @C_Row  + 1)
      SET @C_F_name = (SELECT CustomerFName FROM CUSTOMER WHERE CustomerID
= @RandomCustID)
   END
   SET @C_L_name = (SELECT CustomerLName FROM CUSTOMER WHERE CustomerID =
@RandomCustID)
   SET @Birthday = (SELECT CustomerBirth FROM CUSTOMER WHERE CustomerID =
@RandomCustID)
   SET @Aller_Name = (SELECT AllergyName FROM ALLERGY WHERE AllergyID =
@RandomAllergyID)
   SET @Sev_Name = (SELECT SeverityName FROM SEVERITY WHERE SeverityID =
@RandomSeverityID)
```

```
    EXEC Russ_Insert_CUSTOMER_ALLERGY
    @Allergy_N  = @Aller_Name,
    @Sever_N = @Sev_Name,
    @Cust_F = @C_F_name,
    @Cust_L = @C_L_name,
    @Birthy  = @Birthday


    SET @RUN  = @RUN - 1

END
GO

EXEC Pop_Customer_Allergy 50000

GO

-- Lei Lei
CREATE PROCEDURE lei_pop_ingredient
@RUN INT
AS

DECLARE @RANDOM_IngT_ID INT, @RANDOM_IngT_Row INT
DECLARE @InT_Name VARCHAR(60), @Ing_Name VARCHAR(60)

SET @RANDOM_IngT_Row = (SELECT COUNT(*) FROM INGREDIENT_TYPE)

WHILE @RUN > 0
BEGIN
   SET @RANDOM_IngT_ID = (SELECT RAND() * @RANDOM_IngT_Row + 1)

   SET @InT_Name = (SELECT IngredientTypeName FROM INGREDIENT_TYPE WHERE
IngredientTypeID = @RANDOM_IngT_ID)

   SET @Ing_Name = (Select ingredient From (
                                   Select Row_Number() Over (Order By ingredient) As RowNum, *
                                   From (SELECT DISTINCT(ingredient) FROM
Group_2_Data.dbo.['psrecipe-1$']) A) t2  Where RowNum = @RUN)

   EXEC Russ_Insert_Ingre
   @Ingre_Type_Name = @InT_Name,
   @Ingre_Name = @Ing_Name
```

```
  SET @RUN = @RUN - 1
END
GO

EXEC lei_pop_ingredient 335
GO

-- Lei Lei
CREATE PROCEDURE lei_pop_dish
@RUN INT
AS

DECLARE @RANDOM_DishT_ID INT, @RANDOM_DishT_Row INT
DECLARE @DishT_Name VARCHAR(60), @Dish_Name VARCHAR(60)

SET @RANDOM_DishT_Row = (SELECT COUNT(*) FROM DISH_TYPE)

WHILE @RUN > 0
BEGIN
  SET @RANDOM_DishT_ID = (SELECT RAND() * @RANDOM_DishT_Row + 1)

  SET @DishT_Name = (SELECT DishType FROM DISH_TYPE WHERE DishTypeID =
@RANDOM_DishT_ID)

  SET @Dish_Name = (Select [name] From (
                              Select Row_Number() Over (Order By [name]) As RowNum, *
                              From (SELECT DISTINCT([name]) FROM
Group_2_Data.dbo.['psrecipe-1$']) A) t2  Where RowNum = @RUN)

  EXEC zixinl07_Insert_Dish
  @Name = @Dish_Name,
  @DishTypeName = @DishT_Name

  SET @RUN = @RUN - 1
END
GO
EXEC lei_pop_dish 55
GO

CREATE TABLE EMAIL_COPY(
     EmailID INT IDENTITY(1, 1) PRIMARY KEY,
     EmailName VARCHAR(255)
)
```

```
INSERT INTO EMAIL_COPY(EmailName)
SELECT * FROM Group_2_Data.dbo.FAKE_EMAIL
GO


IF EXISTS (SELECT * FROM sys.sysobjects WHERE NAME = '#JUNK_INSTRUCTOR_DATA')
   BEGIN
      DROP TABLE #JUNK_INSTRUCTOR_DATA
      CREATE TABLE #JUNK_INSTRUCTOR_DATA (
      Inst_ID INT IDENTITY(1, 1) PRIMARY KEY,
      InstFname VARCHAR(60),
      InstLname VARCHAR(60),
      InstrBirth VARCHAR(60)
   )
   END
GO



INSERT INTO #JUNK_INSTRUCTOR_DATA (InstFname, InstLname, InstrBirth)
SELECT InstructorFName, InstructorLName, InstructorBirth
FROM UNIVERSITY.dbo.tblINSTRUCTOR

GO

-- Claire Li
CREATE PROCEDURE zixinl07_pop_instructor
AS

DECLARE @RANDOM_InstrT_ID INT, @RANDOM_InstrT_Row INT, @Random_EmailID INT,
@EmailRow INT, @Instr_Row INT, @Random_Inst_ID INT, @RUN INT, @Min_PK INT
DECLARE @In_Fname VARCHAR(50), @In_Lname VARCHAR(50), @In_Birth DATE,
@In_Email VARCHAR(50), @InT_Name VARCHAR(50)

SET @EmailRow = (SELECT COUNT(*) FROM EMAIL_COPY)
SET @RANDOM_InstrT_Row = (SELECT COUNT(*) FROM  INSTRUCTOR_TYPE)
SET @RUN = (SELECT COUNT(*) FROM #JUNK_INSTRUCTOR_DATA)
SET @Min_PK = (SELECT TOP 1 Inst_ID FROM #JUNK_INSTRUCTOR_DATA)

WHILE @RUN > 0
BEGIN

   SET @Random_EmailID = (SELECT RAND() * @EmailRow + 1)
  SET @RANDOM_InstrT_ID = (SELECT RAND() * @RANDOM_InstrT_Row + 1)
  SET @InT_Name = (SELECT InstructorTypeName FROM INSTRUCTOR_TYPE WHERE
InstructorTypeID = @RANDOM_InstrT_ID)

  SET @In_Fname = (Select InstFname FROM #JUNK_INSTRUCTOR_DATA WHERE Inst_ID
= @Min_PK)
  SET @In_Lname = (Select InstLname FROM #JUNK_INSTRUCTOR_DATA WHERE Inst_ID
= @Min_PK)
  SET @In_Birth = (Select InstrBirth FROM #JUNK_INSTRUCTOR_DATA WHERE Inst_ID =
@Min_PK)
  SET @In_Email = (Select EmailName FROM EMAIL_COPY Where EmailID =
@Random_EmailID)
  EXEC zixinl07_Insert_Instructor
  @Fname = @In_Fname,
  @Lname = @In_Lname,
  @Birth = @In_Birth,
  @Email = @In_Email,
  @InstTypeName = @InT_Name

   DELETE FROM #JUNK_INSTRUCTOR_DATA WHERE Inst_ID = @Min_PK
   SET @Min_PK = (SELECT TOP 1 Inst_ID FROM #JUNK_INSTRUCTOR_DATA)
  SET @RUN = @RUN - 1
END

EXEC zixinl07_pop_instructor
GO


-- Claire Li
CREATE PROCEDURE zixinl07_pop_recipe
@RUN INT
AS

DECLARE @RANDOM_Dish_ID INT, @RANDOM_Dish_Row INT, @RANDOM_Classif_ID INT,
@RANDOM_Classif_Row INT, @RANDOM_Diffic_ID INT, @RANDOM_Diffic_Row INT
DECLARE @D_Name VARCHAR(50), @Clsf_Name VARCHAR(60), @Diff_Level INT,
@Re_Name VARCHAR(50), @Time TIME
DECLARE @FName VARCHAR(50)

SET @RANDOM_Dish_Row = (SELECT COUNT(*) FROM DISH)
SET @RANDOM_Classif_Row = (SELECT COUNT(*) FROM CLASSIFICATION)
SET @RANDOM_Diffic_Row = (SELECT COUNT(*) FROM DIFFICULTY)

WHILE @RUN > 0
```

```
BEGIN
  SET @RANDOM_Dish_ID = (SELECT RAND() * @RANDOM_Dish_Row + 1)
  SET @RANDOM_Classif_ID = (SELECT RAND() * @RANDOM_Classif_Row + 1)
  SET @RANDOM_Diffic_ID = (SELECT RAND() * @RANDOM_Diffic_Row + 1)

  SET @FName = (Select [InstrFName] From (Select Row_Number() Over (Order By
[InstrFName]) As RowNum, *
                                From (SELECT DISTINCT([InstrFName]) FROM [INSTRUCTOR]) A) t2
Where RowNum = @RUN)
  SET @D_Name = (SELECT DishName FROM DISH WHERE DishID = @RANDOM_Dish_ID)
  SET @Clsf_Name = (SELECT ClassificationName FROM CLASSIFICATION WHERE
ClassificationID = @RANDOM_Classif_ID)
  SET @Diff_Level = (SELECT DifficultyLevel FROM DIFFICULTY WHERE DifficultyID =
@RANDOM_Diffic_ID)
  SET @D_Name = (SELECT DishName FROM DISH WHERE DishID = @RANDOM_Dish_ID)
  SET @Re_Name = (SELECT @FName + '''s recipe of ' + @D_Name)
  SET @Time = (SELECT DATEADD(s, ABS(CHECKSUM(NewId()) % 43201), CAST('00:00:00'
AS Time)))
  EXEC zixinl07_Insert_Recipe
   @DishName = @D_Name,
   @ClassifName = @Clsf_Name,
   @DiffLevel = @Diff_Level,
   @ReName = @Re_Name,
   @Time = @Time
  SET @RUN = @RUN - 1
END
GO

EXEC zixinl07_pop_recipe 3000
GO

-- Abdiwahid Hajir
CREATE PROCEDURE INSERT_DATE_COMMENT
AS
DECLARE @Recipe_Name varchar(200), @Random INTEGER,
@Comment_Content varchar(500)

SET @Random = 1
WHILE @Random < 100
BEGIN
SET @Recipe_Name = (Select RecipeName FROM RECIPE WHERE RecipeID = @Random)
SET @Comment_Content = (Select RecipeName FROM RECIPE WHERE RecipeID =
@Random) + ' Is Awesome'
EXEC Update_Comment
```

```
@Recipe_Name = @Recipe_Name,
@Comment_Content = @Comment_Content
SET @Random = @Random + 1
END

EXEC INSERT_DATE_COMMENT
GO

-- Claire Li
CREATE PROCEDURE zixinl07_pop_recipe_task
@RUN INT
AS

DECLARE @RANDOM_Re_ID INT, @RANDOM_Re_Row INT, @RANDOM_Task_ID INT,
@RANDOM_Task_Row INT
DECLARE @Re_Name VARCHAR(50), @Task_Name VARCHAR(50), @Task_Time INT,
@Seq INT
DECLARE @Seq_int INT

SET @RANDOM_Re_Row = (SELECT COUNT(*) FROM RECIPE)
SET @RANDOM_Task_Row = (SELECT COUNT(*) FROM TASK)


WHILE @RUN > 0
BEGIN
  SET @RANDOM_Re_ID = (SELECT RAND() * @RANDOM_Re_Row + 1)
  SET @RANDOM_Task_ID = (SELECT RAND() * @RANDOM_Task_Row + 1)
  SET @Seq_int = (SELECT FLOOR(RAND() * (10) +1 ))
  SET @Re_Name = (SELECT RecipeName FROM RECIPE WHERE RecipeID =
@RANDOM_Re_ID)
  SET @Task_Name = (SELECT TaskName FROM TASK WHERE TaskID =
@RANDOM_Task_ID)
  SET @Task_Time = (SELECT TaskTime FROM TASK WHERE TaskID =
@RANDOM_Task_ID)
 SET @Seq = (SELECT FLOOR(RAND() * (10) +1 ))
  EXEC zixinl07_Insert_Recipe_Task
   @R_Name = @Re_Name,
   @T_Name = @Task_Name,
   @T_Time = @Task_Time,
   @S = @Seq
  SET @RUN = @RUN - 1
END
GO
```

```
EXEC zixinl07_pop_recipe_task 10000
GO

-- Russell Eng
CREATE PROCEDURE INSERT_DATA_INTO_CLASS
@RUN INT
AS
DECLARE @Read_Data INTEGER, @Instrcutor_Data INTEGER, @SCHOOL_ID INT,
                  @INSTRUCTOR_FIRST_NAME varchar(80),
                  @INSTRUCTOR_LAST_NAME varchar(80), @INSTRUCTOR_BIRTHDAY
DATE,
                  @CLASS_NAME varchar(80), @COURSE_NAME varchar(80),
@SCHOOL_NAME varchar(80),
      @RandomClassNum INT
DECLARE  @Random_Instr_ID INT, @Instr_Row INT,
      @Random_Course_ID INT, @Course_Row INT

SET @Course_Row = (SELECT COUNT(*) FROM COURSE)
SET @Instr_Row = (SELECT COUNT(*) FROM INSTRUCTOR)
SET @Random_Instr_ID = (SELECT RAND() * @Instr_Row + 1)
SET @Random_Course_ID = (SELECT RAND() * @Course_Row  + 1)
WHILE @RUN > 0
BEGIN
SET @Random_Instr_ID = (SELECT RAND() * @Instr_Row + 1)
SET @Random_Course_ID = (SELECT RAND() * @Course_Row  + 1)
SET @INSTRUCTOR_FIRST_NAME = (Select InstrFname FROM Instructor WHERE
InstructorID = @Random_Instr_ID)
SET @INSTRUCTOR_LAST_NAME = (Select InstrLname FROM Instructor WHERE
InstructorID = @Random_Instr_ID)
SET @INSTRUCTOR_BIRTHDAY = (Select InstrBirth FROM Instructor WHERE InstructorID =
@Random_Instr_ID)

SET @COURSE_NAME = (Select CourseName FROM COURSE WHERE CourseID =
@Random_Course_ID)
SET @RandomClassNum = (SELECT Rand() * 400 + 100)
SET @CLASS_NAME = @COURSE_NAME + ' ' +CAST(@RandomClassNum AS VARCHAR)
WHILE EXISTS (SELECT * FROM CLASS WHERE ClassName = @CLASS_NAME)
BEGIN
  SET @RandomClassNum = (SELECT Rand() * 400 + 100)
   SET @CLASS_NAME = @COURSE_NAME + ' ' +CAST(@RandomClassNum AS
VARCHAR)
END
SET @SCHOOL_ID = (SELECT RAND() * 4 + 1)
SET @SCHOOL_NAME = (Select SchoolName FROM SCHOOL WHERE SchoolID =
@SCHOOL_ID)
IF NOT EXISTS (SELECT * FROM CLASS WHERE InstructorID = @Random_Instr_ID AND
CourseID = @Random_Course_ID
            AND SchoolID = @SCHOOL_ID)
BEGIN
EXEC Populate_Class_Table
@InstrFname = @INSTRUCTOR_FIRST_NAME,
@InstrLname = @INSTRUCTOR_LAST_NAME,
@InstrBirth = @INSTRUCTOR_BIRTHDAY,
@ClassName = @CLASS_NAME,
@SchoolName = @SCHOOL_NAME,
@CourseName = @COURSE_NAME
END
SET @RUN = @RUN - 1
END


EXEC INSERT_DATA_INTO_CLASS 10000
GO

-- Lei Lei
CREATE PROCEDURE leil_Pop_Class_Detail
@RUN INT
AS
DECLARE @InstrFn varchar(80), @InstrLn varchar(80), @InstrBi DATE, @ClassN varchar(60),
@SchoolN varchar(60)
DECLARE @CourseN varchar(70), @Class_T varchar(80), @END_T varchar(80), @Begin_T
varchar(80), @Class_Room_Nu varchar(80)
DECLARE @RANDOM_Detail_ID INT, @RANDOM_Class_ID INT
DECLARE @RANDOM_Detail_Row INT, @RANDOM_Class_Row INT

SET @RANDOM_Detail_Row = (SELECT COUNT(*) FROM DETAIL)
SET @RANDOM_Class_Row = (SELECT COUNT(*) FROM CLASS)

WHILE @RUN > 0
BEGIN
  SET @RANDOM_Detail_ID = (SELECT RAND() * @RANDOM_Detail_Row + 1)
  SET @RANDOM_Class_ID = (SELECT RAND() * @RANDOM_Class_Row + 1)

  SET @InstrFn = (SELECT INS.InstrFname FROM INSTRUCTOR INS JOIN CLASS C ON
C.InstructorID = INS.InstructorID WHERE C.ClassID = @RANDOM_Class_ID)
  SET @InstrLn = (SELECT INS.InstrLname FROM INSTRUCTOR INS JOIN CLASS C ON
C.InstructorID = INS.InstructorID WHERE C.ClassID = @RANDOM_Class_ID)
```

```
  SET @InstrBi = (SELECT INS.InstrBirth FROM INSTRUCTOR INS JOIN CLASS C ON
C.InstructorID = INS.InstructorID WHERE C.ClassID = @RANDOM_Class_ID)

  SET @ClassN = (SELECT ClassName FROM CLASS WHERE ClassID =
@RANDOM_Class_ID)

  SET @SchoolN = (SELECT S.SchoolName FROM CLASS C JOIN SCHOOL S ON
S.SchoolID = C.SchoolID WHERE C.ClassID = @RANDOM_Class_ID)

  SET @CourseN = (SELECT CO.CourseName FROM CLASS C JOIN COURSE CO ON
CO.CourseID = C.CourseID WHERE C.ClassID = @RANDOM_Class_ID)

  SET @END_T = (SELECT EndTime FROM DETAIL WHERE DetailID =
@RANDOM_Detail_ID)
  SET @Begin_T = (SELECT BeginTime FROM DETAIL WHERE DetailID =
@RANDOM_Detail_ID)
  SET @Class_Room_Nu = (SELECT ClassRoomNumber FROM DETAIL WHERE DetailID =
@RANDOM_Detail_ID)


  IF NOT EXISTS (SELECT * FROM CLASS_DETAIL WHERE ClassID =
@RANDOM_Class_ID AND DetailID  = @RANDOM_Detail_ID)
      BEGIN

      EXEC update_ClassDetail_Table
      @InstrFname = @InstrFn,
      @InstrLname = @InstrLn,
      @InstrBirth = @InstrBi,
      @ClassName = @ClassN,
      @SchoolName = @SchoolN,
      @CourseName = @CourseN,
      @Class_Time = @Class_T,
      @END_TIME = @END_T,
      @Begin_Time = @Begin_T,
      @Class_Room_Number = @Class_Room_Nu
      END
  SET @RUN = @RUN - 1
END

GO

EXEC leil_Pop_Class_Detail 10000
GO
```

```
-- Claire Li
CREATE PROCEDURE zixinl07_pop_registration
@RUN INT
AS

DECLARE @RANDOM_Cust_ID INT, @RANDOM_Cust_Row INT, @RANDOM_Class_ID INT,
        @RANDOM_Cl_ID INT, @RANDOM_Cl_Row INT
DECLARE @F VARCHAR(60), @L VARCHAR(60), @Birth DATE,
@ClassName VARCHAR(60), @Datee DATE, @Gradee Numeric(10,2), @Feee Numeric(10,2)
DECLARE @FromDate DATE = '2020-01-01'
DECLARE @ToDate DATE = '2021-12-31'

SET @RANDOM_Cust_Row = (SELECT COUNT(*) FROM CUSTOMER)
SET @RANDOM_Cl_Row = (SELECT COUNT(*) FROM CLASS)

WHILE @RUN > 0
BEGIN
  SET @RANDOM_Cust_ID = (SELECT RAND() * @RANDOM_Cust_Row + 1)

  SET @RANDOM_Cl_ID = (SELECT RAND() * @RANDOM_Cl_Row + 1)

  SET @F = (SELECT CustomerFName FROM CUSTOMER WHERE CustomerID =
@RANDOM_Cust_ID)
  WHILE @F IS NULL
   BEGIN
    SET @RANDOM_Cust_ID = (SELECT RAND() * @RANDOM_Cust_Row + 1)
    SET @F = (SELECT CustomerFName FROM CUSTOMER WHERE CustomerID =
@RANDOM_Cust_ID)
   END
  SET @L = (SELECT CustomerLName FROM CUSTOMER WHERE CustomerID =
@RANDOM_Cust_ID)
  SET @Birth = (SELECT CustomerBirth FROM CUSTOMER WHERE CustomerID =
@RANDOM_Cust_ID)
  IF (SELECT COUNT(*) FROM CUSTOMER WHERE CustomerFName = @F AND
CustomerLName = @L AND CUSTOMERBirth = @Birth) > 1
  BEGIN
    SET @RANDOM_Cust_ID = 1
    SET @F = (SELECT CustomerFName FROM CUSTOMER WHERE CustomerID =
@RANDOM_Cust_ID)
    SET @L = (SELECT CustomerLName FROM CUSTOMER WHERE CustomerID =
@RANDOM_Cust_ID)
    SET @Birth = (SELECT CustomerBirth FROM CUSTOMER WHERE CustomerID =
@RANDOM_Cust_ID)
  END
  SET @ClassName = (SELECT ClassName FROM CLASS WHERE ClassID =
@RANDOM_Cl_ID)
  SET @Datee = (SELECT GETDATE() - (RAND() * 10000))
 SET @Gradee = (SELECT RAND() * (2) +2 )
 SET @Feee = (SELECT RAND() * (500) +300)

Exec Russ_Insert_Registration
@Firsty = @F,
@Lasty = @L,
@Birthy = @Birth,
@Class_Name = @ClassName,
@DATE = @Datee,
@Grade = @Gradee,
@Fee = @Feee


 SET @RUN = @RUN - 1
END
GO

EXEC zixinl07_pop_registration 2000000
GO

-- Claire Li
CREATE PROCEDURE zixinl07_pop_Class_Dish
@RUN INT
AS
DECLARE @RANDOM_Class_ID INT, @RANDOM_Class_Row INT, @RANDOM_Dish_ID INT,
@RANDOM_Dish_Row INT,
      @RANDOM_Sch_ID INT, @RANDOM_Sch_Row INT

DECLARE @Instr_Fname VARCHAR(50), @Instr_Lname VARCHAR(50), @Instr_Birthy DATE,
@Course_Name VARCHAR(60),
      @School_Name VARCHAR(60), @Cl_Name VARCHAR(60), @D_Name VARCHAR(50)

SET @RANDOM_Class_Row = (SELECT COUNT(*) FROM CLASS)
SET @RANDOM_Dish_Row = (SELECT COUNT(*) FROM DISH)
SET @RANDOM_Sch_Row = (SELECT COUNT(*) FROM SCHOOL)

WHILE @RUN > 0
BEGIN
 SET @RANDOM_Class_ID = (SELECT RAND() * @RANDOM_Class_Row + 1)
 SET @RANDOM_Dish_ID = (SELECT RAND() * @RANDOM_Dish_Row + 1)
 SET @RANDOM_Sch_ID = (SELECT RAND() * @RANDOM_Sch_Row + 1)
 SET @Instr_Fname = (SELECT InstrFname FROM INSTRUCTOR I
                JOIN CLASS C ON C.InstructorID = I.InstructorID WHERE C.ClassID =
@RANDOM_Class_ID)
 SET @Instr_Lname = (SELECT InstrLname FROM INSTRUCTOR I
                JOIN CLASS C ON C.InstructorID = I.InstructorID WHERE C.ClassID =
@RANDOM_Class_ID)
 SET @Instr_Birthy = (SELECT InstrBirth FROM INSTRUCTOR I
                JOIN CLASS C ON C.InstructorID = I.InstructorID WHERE C.ClassID =
@RANDOM_Class_ID)
 SET @Course_Name = (SELECT CourseName FROM COURSE CR
                JOIN CLASS C ON CR.CourseID = C.CourseID WHERE C.ClassID =
@RANDOM_Class_ID)
 SET @School_Name = (SELECT SchoolName FROM SCHOOL WHERE SchoolID =
@RANDOM_Sch_ID)
 SET @Cl_Name = (SELECT ClassName FROM CLASS WHERE ClassID =
@RANDOM_Class_ID)
 SET @D_Name = (SELECT DishName FROM DISH WHERE DishID = @RANDOM_Dish_ID)

 IF NOT EXISTS (SELECT * FROM CLASS_DISH WHERE ClassID = @RANDOM_Class_ID
AND DishID  = @RANDOM_Dish_ID)
 BEGIN
 EXEC zixinl07_Insert_Class_Dish
   @Ins_Fname = @Instr_Fname,
   @Ins_Lname = @Instr_Lname,
   @Ins_Birthy = @Instr_Birthy,
   @Cou_Name = @Course_Name,
   @Sch_Name = @School_Name,
   @Class_Name = @Cl_Name,
   @Dish_Name = @D_Name
 END
 SET @RUN = @RUN - 1
END
GO
EXEC zixinl07_pop_Class_Dish 7000
GO


-- Lei Lei
CREATE PROCEDURE leil_Pop_Ing_Recipe_Warpper
@RUN INT
AS
DECLARE @RANDOM_Ing_ID INT, @RANDOM_Rec_ID INT
DECLARE @RANDOM_Ing_Row INT, @RANDOM_Rec_Row INT
DECLARE @Recipe_N varchar(200), @Ingrident_N varchar(100)


SET @RANDOM_Ing_Row = (SELECT COUNT(*) FROM INGREDIENT)
SET @RANDOM_Rec_Row = (SELECT COUNT(*) FROM RECIPE)

WHILE @RUN > 0
BEGIN
      SET @RANDOM_Ing_ID = (SELECT RAND() * @RANDOM_Ing_Row + 1)
      SET @RANDOM_Rec_ID = (SELECT RAND() * @RANDOM_Rec_Row + 1)

      SET @Recipe_N = (SELECT RECIPE.RecipeName FROM RECIPE WHERE
RECIPE.RecipeID = @RANDOM_Rec_ID)
      SET @Ingrident_N = (SELECT INGREDIENT.IngredientName FROM INGREDIENT
WHERE INGREDIENT.IngredientID =  @RANDOM_Ing_ID)

      IF NOT EXISTS (SELECT * FROM INGREDIENT_RECIPE WHERE IngredientID =
@RANDOM_Ing_ID AND RecipeID = @RANDOM_Rec_ID)
         BEGIN
         EXEC Procedure_Ingredient_Recipe
         @Recipe_Name = @Recipe_N,
         @Ingrident_Name = @Ingrident_N

         END
SET @RUN = @RUN - 1
END
GO

Exec Ieil_Pop_Ing_Recipe_Warpper 9500
GO



-- Business Rules

-- Abdiwahid Hajir
-- No instructor younger than 18 may be allowed to teach
CREATE FUNCTION No_Under_Age_Instructors()
RETURNS INTEGER
AS
BEGIN
DECLARE @Ret_Integer INTEGER = 0
IF
EXISTS(SELECT *
FROM INSTRUCTOR I
WHERE I.InstrBirth > DATEADD(YEAR, -18, GetDate()))
BEGIN
SET @Ret_Integer = 1
END
RETURN @Ret_Integer
END
GO

ALTER TABLE INSTRUCTOR
ADD CONSTRAINT NO_underAGE_INSTRUCTORS
CHECK(dbo.No_Under_Age_Instructors () = 0)
GO

-- No dish may be served if the ingredient requires pork.
CREATE FUNCTION No_Pork_Allowed()
RETURNS INTEGER
AS
BEGIN
DECLARE @RET_INTEGER INTEGER = 0
IF
EXISTS(SELECT *
FROM DISH D
JOIN DISH_TYPE DT ON D.DishTypeID = DT.DishTypeID
JOIN RECIPE RE ON D.DishID = RE.DishID
WHERE CHARINDEX('Pork' ,D.DishName) > 0
OR CHARINDEX('Pork' ,RE.RecipeName) > 0)
BEGIN
SET @RET_INTEGER = 1
END
RETURN @RET_INTEGER
END
GO

ALTER TABLE Dish
ADD CONSTRAINT NO_Pork
CHECK(dbo.No_Pork_Allowed () = 0)
GO

-- Claire Li
-- No instructor younger than 20 can taught class having recipe difficult level higher than 2.
CREATE FUNCTION Zixinl07_NoYoung_Instructor ()
RETURNS INT
AS
```

```
BEGIN
  DECLARE @RET INT = 0
  IF EXISTS (SELECT *
            FROM INSTRUCTOR I JOIN CLASS C ON I.InstructorID = C.InstructorID
            JOIN CLASS_DISH CD ON C.ClassID = CD.ClassID
            JOIN DISH D ON CD.DishID = D.DishID
            JOIN RECIPE R ON D.DishID = R.DishID
            JOIN DIFFICULTY DF ON R.DifficultyID = DF.DifficultyID
            WHERE DF.DifficultyLevel > 2
            AND I.InstrBirth > DATEADD(year, -20, GETDATE()))
  BEGIN
     SET @RET = 1
  END
RETURN @RET
END
GO

ALTER TABLE CLASS
ADD CONSTRAINT No_Instructor_teach_hard_class
CHECK (dbo.Zixinl07_NoYoung_Instructor() = 0)
GO

-- Customer from Washington state can’t register over 5 course in one year.
CREATE FUNCTION Zixinl07_NoWA_Course5 ()
RETURNS INT
AS
BEGIN
  DECLARE @RET INT = 0
  IF EXISTS (SELECT C.CustomerID, COUNT(CR.CourseID) AS CourseCount
            FROM CUSTOMER C JOIN CITY CT ON C.CityID = CT.CityID
            JOIN STATE S ON CT.StateID = S.StateID
            JOIN REGISTRATION R ON C.CustomerID = R.RegisterID
            JOIN CLASS CL ON R.ClassID = CL.ClassID
            JOIN COURSE CR ON CL.CourseID = CR.CourseID
            WHERE S.StateName = 'Washington, WA'
            AND R.RegistrationDATE > DATEADD(year, -1, GETDATE())
            GROUP BY C.CustomerID
            HAVING COUNT(CR.CourseID) > 5)
  BEGIN
     SET @RET = 1
  END
RETURN @RET
END
GO
```

```
ALTER TABLE REGISTRATION
ADD CONSTRAINT NoWA_Course5
CHECk (dbo.Zixinl07_NoWA_Course5 () = 0)
GO


-- Lei Lei
-- No customer younger than 16 can register for class with registration fee over 800
CREATE FUNCTION leil_NoYoung_Take_Class()
RETURNS INT
AS
BEGIN
  DECLARE @RET INT = 0
  IF EXISTS (SELECT *
          FROM CUSTOMER C
             JOIN REGISTRATION R ON R.CustomerID = C.CustomerID
             JOIN CLASS CL ON R.ClassID = CL.ClassID
          WHERE C.CustomerBirth > DateAdd(Year, -16, GetDate())
          AND R.RegstrationFEE > 800)
  BEGIN
     SET @RET = 1
  END
  RETURN @RET
END
GO

ALTER TABLE REGISTRATION WITH NOCHECK
ADD CONSTRAINT Check_Age_Fee
CHECK (dbo.leil_NoYoung_Take_Class() = 0)
GO

-- No Customer from Washington can take class 'Spice Fix 436'
ALTER FUNCTION leil_Wash_Take_Class()
RETURNS INT
AS
BEGIN
  DECLARE @RET INT = 0
  IF EXISTS (SELECT *
          FROM CUSTOMER C
             JOIN REGISTRATION R ON R.CustomerID = C.CustomerID
             JOIN CLASS CL ON R.ClassID = CL.ClassID
             JOIN CITY CI ON C.CityID = CI.CityID
             JOIN [STATE] S ON CI.StateID = S.StateID
```

```
            WHERE S.StateName = 'Washington, WA'
          AND CL.ClassName = 'Spice Fix 436')
   BEGIN
      SET @RET = 1
   END
   RETURN @RET
END
GO

ALTER TABLE REGISTRATION WITH NOCHECK
ADD CONSTRAINT Check_Wash_Spice
CHECK (dbo.Ieil_Wash_Take_Class() = 0)
GO

-- Russell Eng
-- No customer with an allergy “Peanuts” can take the course “Seasonal Cooking & Catering”
CREATE FUNCTION Russ_NoAllergy_Take_Class()
RETURNS INT
AS
BEGIN
    DECLARE @RET INT = 0
    IF EXISTS (SELECT *
            FROM ALLERGY A
                JOIN CUSTOMER_ALLERGY CA ON A.AllergyID = CA.AllergyID
                JOIN CUSTOMER C ON C.CustomerID = CA.CustomerID
                JOIN REGISTRATION R ON R.CustomerID = C.CustomerID
                JOIN CLASS CL ON R.ClassID = CL.ClassID
                JOIN COURSE CR ON CR.CourseID = CL.CourseID
            WHERE CR.CourseName = 'Seasonal Cooking & Catering'
            AND A.AllergyName = 'Peanuts')
    BEGIN
        SET @RET = 1
    END
    RETURN @RET

END
GO

ALTER TABLE REGISTRATION WITH NOCHECK
ADD CONSTRAINT Check_Allergy
CHECK (dbo.Russ_NoAllergy_Take_Class() = 0)
GO
```

```
-- A recipe name that has "chocolate" in it cannot exceed 15 tasks
CREATE FUNCTION Russ_Choco_Task_Limit ()
RETURNS INT
AS
BEGIN
   DECLARE @RET INT = 0
   IF EXISTS (SELECT RecipeTaskID, R.RecipeName
               FROM TASK T
                  JOIN RECIPE_TASK RT ON T.TaskID = RT.TaskID
                  JOIN RECIPE R ON RT.RecipeID = R.RecipeID
               WHERE R.RecipeName LIKE '%chocolate%'
               GROUP BY RecipeTaskID, R.RecipeName
               HAVING COUNT(*) > 15)
   BEGIN
      SET @RET = 1
   END
   RETURN @RET
   END
GO

ALTER TABLE RECIPE_TASK
ADD CONSTRAINT linit_Choco
CHECK (dbo.Russ_Choco_Task_Limit() = 0)

GO

-- Computed Columns

-- Abdiwahid Hajir
-- Write a computed column that calculates the total
-- number of instructors that have taught the class Taste The best
CREATE FUNCTION Total_Ins_Taught_TasteBest(@PK INT)
RETURNS INTEGER
AS
BEGIN
DECLARE @RET INTEGER = (SELECT COUNT(*)
FROM INSTRUCTOR I
JOIN CLASS C ON I.InstructorID = C.InstructorID
WHERE CHARINDEX('Taste The Best' ,C.ClassName) > 0
AND I.InstructorID = @PK)
RETURN @RET
END
GO
```

```
ALTER TABLE INSTRUCTOR
ADD Total_Ins_Taught_TasteBest AS(dbo.Total_Ins_Taught_TasteBest(InstructorID))
GO

--Write a computed column that calculates
--the total number of students that have taken a class about Meat
CREATE FUNCTION Total_Student_Pizza(@PK INT)
RETURNS INTEGER
AS
BEGIN
DECLARE @RET INTEGER = (SELECT COUNT(*)
FROM CUSTOMER C
JOIN REGISTRATION R ON C.CustomerID = R.CustomerID
JOIN CLASS CL ON R.ClassID = CL.ClassID
WHERE CHARINDEX('Meat' ,CL.ClassName) > 0
AND C.CustomerID = @PK)
RETURN @RET
END
GO

ALTER TABLE CUSTOMER
ADD Total_Ins_Taught_TasteBest AS(dbo.Total_Student_Pizza(CustomerID))
GO

-- Claire Li

-- Write a computed column that calculates the average grade of each customer.
CREATE FUNCTION Zixinl07_Calc_Grade(@PK INT)
RETURNS INT
AS
BEGIN
  DECLARE @RET INT = (SELECT AVG(R.GRADE)
                FROM CUSTOMER C JOIN REGISTRATION R ON C.CustomerID =
R.CustomerID
                WHERE C.CustomerID = @PK
  )
RETURN @RET
END
GO

ALTER TABLE CUSTOMER
ADD AvgGrade AS(dbo.Zixinl07_Calc_Grade(CustomerID))
GO
```

```
-- Write a computed column that calculates how many tasks for each recipe.
CREATE FUNCTION Zixinl07_Calc_Task(@PK INT)
RETURNS INT
AS
BEGIN
  DECLARE @RET INT = (SELECT COUNT(T.TaskName)
                FROM RECIPE R JOIN RECIPE_TASK RT ON R.RecipeID = RT.RecipeID
                JOIN TASK T ON RT.TaskID = T.TaskID
                WHERE R.RecipeID = @PK
  )
RETURN @RET
END
GO
ALTER TABLE RECIPE
ADD TaskCount AS(dbo.Zixinl07_Calc_Task(RecipeID))
GO

-- Lei Lei

-- Write a computed column that calculates the number of students with nut allergy within each
class
CREATE FUNCTION leil_Nut_Allergy(@PK INT)
RETURNS INT
AS
BEGIN
  DECLARE @RET INT = (SELECT COUNT(*)
                FROM CUSTOMER C
                JOIN CUSTOMER_ALLERGY CA ON CA.CustomerID = C.CustomerID
                JOIN ALLERGY A ON A.AllergyID = CA.AllergyID
                JOIN REGISTRATION R ON R.CustomerID = C.CustomerID
                JOIN CLASS CL ON CL.ClassID = R.RegisterID
                WHERE A.AllergyName LIKE '%nut%' AND CL.ClassID = @PK
  )
RETURN @RET
END
GO
ALTER TABLE CLASS
ADD NutAllergyCount AS(dbo.leil_Nut_Allergy(ClassID))
GO

-- Write a computed column that calculates the number of customer registered for each Course
CREATE FUNCTION leil_Cus_Num(@PK INT)
RETURNS INT
```

```
AS
BEGIN
  DECLARE @RET INT = (SELECT COUNT(*)
                    FROM CUSTOMER C
                    JOIN REGISTRATION R ON R.CustomerID = C.CustomerID
                    JOIN CLASS CL ON CL.ClassID = R.RegisterID
                    JOIN COURSE CO ON CL.CourseID = CO.CourseID
                    WHERE CO.CourseID = @PK
  )
RETURN @RET
END
GO

ALTER TABLE COURSE
ADD CustomerNum AS(dbo.Ieil_Cus_Num(CourseID))
GO

-- Russell Eng
-- Write a computed column to calculate the number of Customers for each allergy
CREATE FUNCTION Russ_Calc_Allergy(@PK INT)
RETURNS INT
AS
BEGIN
   DECLARE @RET INT = (SELECT COUNT(*)
                    FROM CUSTOMER C
                    JOIN CUSTOMER_ALLERGY CA ON CA.CustomerID = C.CustomerID
                    JOIN ALLERGY A ON A.AllergyID = CA.AllergyID
                    WHERE A.AllergyID = @PK
   )

RETURN @RET
END
GO

ALTER TABLE ALLERGY
ADD CustNum AS(dbo.Russ_Calc_Allergy(AllergyID))
GO

-- Write a computed column to calculate how many classes have each instructor taught?
CREATE FUNCTION Russ_Calc_Course(@PK INT)
RETURNS INT
AS
BEGIN
   DECLARE @RET INT = (SELECT COUNT(*)
                        FROM INSTRUCTOR I
                          JOIN CLASS C ON C.InstructorID = I.InstructorID
                           WHERE I.InstructorID = @PK)

RETURN @RET
END
GO

ALTER TABLE INSTRUCTOR
ADD ClassNum AS (dbo.Russ_Calc_Course(InstructorID))
GO

-- Complex Queries

-- Abdiwahid Hajir
/* Select top 10 students that had once spent more than $600 regstrition fees and
have taken the class Taste the Best 290. And are from the school
Amazing Greg Hay Baking School. And have received  a grade of 3.0 or higher.
And were born after 1980*/
CREATE VIEW [Rich_Students] AS
SELECT TOP 10 with Ties CU.CustomerID, Cu.CustomerFName,Cu.CustomerLName,
R.RegstrationFEE, C.ClassName, SC.SchoolName
FROM CUSTOMER CU
JOIN REGISTRATION R ON CU.CustomerID = R.CustomerID
JOIN CLASS C ON R.ClassID = C.ClassID
JOIN SCHOOL SC ON C.SchoolID = SC.SchoolID
WHERE R.RegstrationFEE > 600
AND R.GRADE > 3.0
AND SC.SchoolName = 'Amazing Greg Hay Baking School'
AND YEAR(CU.CustomerBirth) >= 1980
ORDER BY R.RegstrationFEE DESC
GO

SELECT * FROM [Rich_Students]
GO

-- Write the query to determine the top 1000 customers
-- partitioned by state that spent the most money on culinary school

CREATE VIEW [Partition_State_Top_1000_Cust] AS
WITH Top_1000_Customers (CustomerID, CustomerFName, CustomerLName,
RegstrationFEE, StateName, Most_Fees)
AS (
SELECT C.CustomerID, C.CustomerFName, C.CustomerLName, SUM(REG.RegstrationFEE),
ST.StateName,
RANK() OVER (Partition BY ST.StateName ORDER BY SUM(REG.RegstrationFEE) DESC)
FROM CUSTOMER C
      JOIN CITY CIT ON C.CityID = CIT.CityID
      JOIN REGISTRATION REG ON C.CustomerID = REG.CustomerID
      JOIN [STATE] ST ON CIT.StateID = ST.StateID
   GROUP BY C.CustomerID, C.CustomerFName, C.CustomerLName, ST.StateName
   )
SELECT  *
FROM Top_1000_Customers
WHERE Most_Fees <= 1000
GO

SELECT * FROM [Partition_State_Top_1000_Cust]
GO



-- Claire Li
-- Write the query to find the top 5 percentile of states that recived most registration fee over the
past decade
CREATE VIEW [Top_Income_State_Cooking_School] AS
WITH CTE_State_Fee (StateID, StateName, TotalFee, Ntile_Fee)
AS
  (SELECT S.StateID, S.StateName, SUM(R.RegstrationFEE) AS TotalFee,
  NTILE(100) OVER (ORDER BY SUM(R.RegstrationFEE) DESC) AS Ntile_Fee
  FROM [STATE] S JOIN CITY C ON S.StateID = C.StateID
             JOIN CUSTOMER CT ON C.CityID = CT.CityID
             JOIN REGISTRATION R ON CT.CustomerID = R.CustomerID
     WHERE R.RegistrationDATE > DATEADD(year, -10, GETDATE())
     GROUP BY S.StateID, S.StateName)
SELECT *
FROM CTE_State_Fee
WHERE Ntile_Fee <= 5
GO

SELECT * FROM [Top_Income_State_Cooking_School]
GO

-- Write a query to find the top 10 course with the highest average grade .
CREATE VIEW [Easy_Courses] AS
SELECT TOP 10
CR.CourseID, CR.CourseName, AVG(R.GRADE) AS AvgGrade
```

```sql
FROM CLASS C JOIN REGISTRATION R ON C.ClassID = R.ClassID
JOIN COURSE CR ON CR.CourseID = C.CourseID
GROUP BY CR.CourseID, CR.CourseName
ORDER BY AVG(R.GRADE) DESC
GO

SELECT * FROM [Easy_Courses]
GO


-- Lei Lei
-- Write a query to determine the top 5 classes registered by customers with nut allergy.
CREATE VIEW [Nut_Allergy_Popular_Course] AS
SELECT TOP 5 C.ClassName, COUNT(CU.CustomerID) AS NumberOfCustomers
FROM CLASS C
JOIN REGISTRATION R ON C.ClassID = R.ClassID
JOIN CUSTOMER CU ON R.CustomerID = CU.CustomerID
JOIN CUSTOMER_ALLERGY CA ON CU.CustomerID = CA.CustomerID
JOIN ALLERGY A ON CA.AllergyID = A.AllergyID
WHERE A.AllergyName LIKE '%nut%'
GROUP BY C.ClassName
ORDER BY COUNT(CU.CustomerID) DESC
GO

SELECT * FROM [Nut_Allergy_Popular_Course]
GO

-- Write a query to determine the 15th percentile of customers based on their average Grade in the comfort food course.

CREATE VIEW [Avg_score_comfort food] AS
WITH CTE_Grade_Comfort (CusID, CusFName, CusLName, Grade, Ntile_Grade)
AS
(SELECT C.CustomerID, C.CustomerFName, C.CustomerLName, AVG(R.GRADE) AS Grade,
NTILE(100) OVER (ORDER BY AVG(R.GRADE) DESC) AS Ntile_Grade
FROM CUSTOMER C
JOIN REGISTRATION R ON C.CustomerID = R.CustomerID
JOIN CLASS CL ON R.ClassID = CL.ClassID
JOIN COURSE CO ON CL.CourseID = CO.CourseID
WHERE CO.CourseName LIKE '%Comfort%'
GROUP BY C.CustomerID, C.CustomerFName, C.CustomerLName)

SELECT CusID, CusFName, CusLName, Grade, Ntile_Grade
FROM CTE_Grade_Comfort
```

```
WHERE Ntile_Grade = 15
GO

SELECT * FROM [Avg_score_comfort food]
GO


-- Russell Eng
-- Write a query to determine the customers from the state of Washington have spent more
-- than 4000 dollars on registration fees for the culinary schools.
CREATE VIEW [Cust_Spent_more_than_4000] AS
SELECT C.CustomerID, C.CustomerFName, C.CustomerLName, SUM(R.RegstrationFEE) AS
TotalRegFee
FROM CUSTOMER C
   JOIN CITY CT ON C.CityID = CT.CityID
   JOIN [STATE] S ON CT.StateID = S.StateID
   JOIN REGISTRATION R ON R.CustomerID = C.CustomerID
WHERE S.StateName = 'Washington, WA'
GROUP BY C.CustomerID, C.CustomerFName, C.CustomerLName
HAVING SUM(R.RegstrationFEE) > 4000
GO

SELECT * FROM [Cust_Spent_more_than_4000]
GO

-- Write the query to find the top 10 percentile of instructors that had taught the most classes

CREATE VIEW [Popular_Instructor] AS
WITH CTE_Instructor_class (I_ID, Fname, Lname, TotalClass, PctTotalClass)
AS
   (SELECT I.InstructorID, I.InstrFname, I.InstrLname, COUNT(I.InstructorID),
   NTILE(100) OVER (ORDER BY COUNT(I.InstructorID) DESC)
   FROM INSTRUCTOR I
      JOIN CLASS C ON I.InstructorID = C.InstructorID
      GROUP BY I.InstructorID, I.InstrFname, I.InstrLname)

SELECT I_ID, Fname, Lname, TotalClass, PctTotalClass
FROM CTE_Instructor_class
WHERE PctTotalClass <= 10
GO

SELECT * FROM Popular_Instructor
GO
```