**Group Project – Skills Demonstration**

Team Name 1:\_Kailynn Liebau\_\_\_\_\_\_\_\_ Date: \_8/2/2023\_\_

Team Name 2:\_Nathan Richardson\_\_\_\_

Team Name 3:\_Nathaniel Bagwell\_\_\_\_\_

Team Name 4:\_Brett Fowler\_\_\_\_\_\_\_\_\_\_

# **Group Project Instructions**

1. Type the names of team members that participated and the date in the spaces provided.
2. Use the SQL Server Management Studio and create and use a new database named xxGPDatabase (where the xx are the initials of one of your team members).
3. Complete all 10 exercises in this group project.
4. Upload and submit a completed copy of this group project worksheet before the due date. Each member of the team that participated is to upload and submit a copy of this document.

**Exercises:**

# **Group Project Exercise 1(Database Design & Creation)**

Design and create a normalized database named xxGPDatabase where the xx are the initials of one of your team members. In this database create tables that a coffee shop can use to manage their Menu, Recipes, Inventory, Employees, and Work Schedules.

The Coffee Shop is open 24 hours a day. Each employee can work a max of 40 hours a week. Each shift is to be staffed with three employees. Each menu item has a recipe that specifies the ingredients and directions. All ingredients except for water are to be tracked in an inventory table.

The database needs to be normalized (3NF / Boyce-Codd Normal Form and no Multi-Valued Dependency). Each table must have a primary key and other appropriate attributes. The attributes on the tables must have appropriate datatypes, lengths, and constraints defined. At least 3 check constraints must be present. In addition, constraints to create appropriate relationships between the tables and enforce referential integrity must be present.

**--- Paste below this line a script file that when executed creates the xxGPDatabase with all the structural components described above. --**

Paste here

USE [BFGPDatabase]

GO

CREATE TABLE Menu (

MenuID int not null,

MenuName varchar (40) not null,

MenuPrice decimal (10,2) not null,

RecipeID varchar (10) not null,

CONSTRAINT PK\_Menu PRIMARY KEY (MenuID)

)

GO

CREATE TABLE Recipes (

RecipeID int not null,

MenuID int not null,

RecipeDirections varchar(MAX) null,

NumberOfServings int not null,

CONSTRAINT PK\_Recipes PRIMARY KEY (RecipeID),

CONSTRAINT FK\_Recipes\_Menu FOREIGN KEY (MenuID) REFERENCES Menu(MenuID)

)

GO

CREATE TABLE Inventory (

InventoryID int not null,

ItemName varchar (40) not null,

InvQuantity decimal (10,2) not null,

InvUnitOFMeasure varchar (4) not null,

InvCostPerUnit decimal (10,2) not null,

CONSTRAINT PK\_Inventory PRIMARY KEY (InventoryID)

)

GO

CREATE TABLE IngredientsToRecipe (

RecipeID int not null,

InventoryID int not null,

IngAmount decimal (10,2) not null,

IngUnitOfMeasure varchar (4) not null,

CONSTRAINT PK\_IngredientsToRecipe PRIMARY KEY (RecipeID, InventoryID),

CONSTRAINT FK\_IngredientsToRecipe\_Recipes FOREIGN KEY (RecipeID) REFERENCES Recipes(RecipeID),

CONSTRAINT FK\_IngredientsToRecipe\_Inventory FOREIGN KEY (InventoryID) REFERENCES Inventory(InventoryID)

)

GO

CREATE TABLE Employees (

  EmployeeID INT PRIMARY KEY NOT NULL,

  FirstName VARCHAR(50) NOT NULL,

  LastName VARCHAR(50) NOT NULL,

  DateOfBirth DATE NOT NULL,

  Email VARCHAR(100),

  Phone VARCHAR(20) NOT NULL,

  Department VARCHAR(100) NOT NULL,

  [Pay] DECIMAL (10, 2) NOT NULL

);

GO

CREATE TABLE WorkSchedule (

  ScheduleID INT PRIMARY KEY NOT NULL,

  EmployeeID INT NOT NULL,

  ScheduleDate DATE,

  StartTime INT,

  EndTime INT,

  ShiftType VARCHAR(3),

  FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID),

);

GO

**--- Paste below this line an image of a database diagram that shows the relationships between the tables --**

Paste here

A diagram of a company

Description automatically generated

# **Group Project Exercise 2 (Database Design & Creation)**

Code and execute Insert Statements to insert six or seven records into each of the tables in the xxGPDatabase Database. Make sure to use each team members name once when coding attribute values for first and last name attributes. Make up appropriate demonstration values for other attribute values.

**--- Paste below this line the insert statements your team coded. –**

Paste here  
  
USE [BFGPDatabase]

GO

INSERT INTO Menu VALUES

(10000, 'Cappuccino', 3.45, 20000),

(10001, 'Caffe Mocha', 4.15, 20001),

(10002, 'Coffee', 2.25, 20002),

(10003, 'Chocolate Croissant', 3.75, 20003),

(10004, 'Chai Tea Latte', 3.95, 20004),

(10005, 'Lemon Loaf Cake', 3.75, 20005)

GO

INSERT INTO Recipes VALUES

(20000, 10000, 'Warm the espresso machine. Put the coffee in the filter basket of the portafilter, gently but firmly tamp it down with a tamper, and secure the portafilter in the brew head. Place a 5- or 6-ounce cup directly under the brew head.

Brew just long enough to yield 1 to 1 1/2 ounces (30 to 44 ml) of espresso. Turn your machine off just as it finishes pouring the last of the pale-colored crema and before the espresso becomes watery, which you’ll learn with practice. The exact time will depend on your machine.

Turn on the steamer and let it warm up. Pour the milk into a stainless steel pitcher. Submerge the tip of the steam wand at least half an inch below the milk surface and turn on the steam switch. As the foam rises, gradually lower the pitcher so the tip of the wand remains just below the milk surface. As the milk begins to heat up, tilt the pitcher slightly to swirl the milk and continue to steam until the side of the pitcher becomes almost too hot to hold. Turn off the steam while still holding the tip of the wand under the milk surface.

Very slowly pour the milk into the cappuccino in one of two ways:

For a smooth cappuccino in which both the warmed and foamed milk on top is evenly integrated throughout the beverage, gently pour the milk over the espresso in a thin, steady stream.', 1),

(20001, 10001, 'Place the filter in the coffeemaker’s basket.

Add your ground coffee beans and then pour in some of the hot water. Use just enough to soak the grounds without pouring through.

Wait 45-90 seconds, permitting the grinds to settle, and then simply operate the automatic coffeemaker as you normally would using 36 ounces of water.', 1),

(20002, 10002, 'Place the filter in the coffeemaker’s basket.

Add your ground coffee beans and then pour in some of the hot water. Use just enough to soak the grounds without pouring through.

Wait 45-90 seconds, permitting the grinds to settle, and then simply operate the automatic coffeemaker as you normally would using 36 ounces of water. Stir cocoa, sugar, and milk into the finished coffee.', 2),

(20003, 10003, 'Make the dough: Cut the butter in four 1-Tablespoon slices and place in the bowl of an electric stand mixer fitted with the dough hook attachment (or you can use a handheld mixer or no mixer, but a stand mixer is ideal). Add the flour, sugar, salt, and yeast. Turn the mixer on low-medium speed to gently combine the ingredients for 1 minute. With the mixer running, slowly pour in the milk. Once all of the milk is added, turn the mixer up to medium speed and beat the dough for at least 5 full minutes. (If you don’t have a mixer, knead by hand for 5 minutes.) The dough will be soft. It will (mostly) pull away from the sides of the bowl and if you poke it with your finger, it will bounce back. If after 5 minutes the dough is too sticky, keep the mixer running until it pulls away from the sides of the bowl.

Remove dough from the bowl and, with floured hands, work it into a ball. Place the dough on a lightly floured silicone baking mat lined, lightly floured parchment paper lined, or lightly floured baking sheet. (I recommend a silicone baking mat because you can roll the dough out in the next step directly on top and it won’t slide all over the counter.) Gently flatten the dough out, and cover with plastic wrap or aluminum foil. Place the entire baking sheet in the refrigerator and allow the covered dough to rest in the refrigerator for 30 minutes.

Shape the dough: Remove the dough from the refrigerator. I like to keep the dough on the silicone baking mat when I’m rolling it in this step because the mat is nonstick and it’s a handy guide for the exact measurement. Begin flattening out the dough with your hands. You’re rolling it out into a rectangle in this step, so shaping it with your hands first helps the stretchy dough. Roll it into a 14×10-inch rectangle. The dough isn’t extremely cold after only 30 minutes in the refrigerator, so it will feel more like soft play-doh. Be precise with the measurement. The dough will want to be oval shaped, but keep working the edges with your hands and rolling pin until you have the correct size rectangle.

Long rest: Place the rolled out dough back onto the baking sheet (this is why I prefer a silicone baking mat or parchment because you can easily transfer the dough). Cover the rolled out dough with plastic wrap or aluminum foil, place the entire baking sheet in the refrigerator and allow the covered dough to rest in the refrigerator for 4 hours or overnight. (Up to 24 hours is ok.)

Butter layer (begin this 35 minutes before the next step so the butter can chill for 30 minutes): In a large bowl using a hand-held mixer or stand mixer fitted with a paddle or whisk attachment, beat the butter and flour together until smooth and combined. Transfer the mixture to a silicone baking mat lined or parchment paper lined baking sheet. (Silicone baking mat is preferred because you can easily peel the butter off in the next step.) Using a spoon or small spatula, smooth out into a 7×10-inch rectangle. Be as precise as you can with this measurement. Place the entire baking sheet in the refrigerator and chill the butter layer for 30 minutes. (No need to cover it for only 30 minutes.) You want the butter layer firm, but still pliable. If it gets too firm, let it sit out on the counter for a few minutes to gently soften. The more firm the butter layer is the more difficult it will be to laminate the dough in the next step.

Laminate the dough: In this next step, you will be rolling out the dough into a large rectangle. Do this on a lightly floured counter instead of rolling out on your silicone baking mat. The counter is typically a little cooler (great for keeping the dough cold) and the silicone baking mat is smaller than the measurement you need. Remove both the dough and butter layers from the refrigerator. Place the butter layer in the center of the dough and fold each end of the dough over it. If the butter wasn’t an exact 7×10-inch rectangle, use a pizza cutter or sharp knife to even out the edges. Seal the dough edges over the butter layer as best you can with your fingers. On a lightly floured counter, roll the dough into a 10×20-inch rectangle. It’s best to roll back and forth with the shorter end of the dough facing you. Use your fingers if you need to. The dough is very cold, so it will take a lot of arm muscle to roll. Again, the dough will want to be oval shaped, but keep working it with your hands and rolling pin until you have the correct size rectangle. Fold the dough lengthwise into thirds as if you were folding a letter. This was the 1st turn.

If the dough is now too warm to work with, place folded dough on the baking sheet, cover with plastic wrap or aluminum foil, and refrigerate for 30 minutes before the 2nd turn. I usually don’t have to.

2nd turn: Turn the dough so the short end is facing you. Roll the dough out once again into a 10×20-inch rectangle, then fold the dough lengthwise into thirds as if you were folding a letter. The dough must be refrigerated between the 2nd and 3rd turn because it has been worked with a lot by this point. Place the folded dough on the baking sheet, cover with plastic wrap or aluminum foil, and refrigerate for 30 minutes before the 3rd turn.

3rd turn: Roll the dough out once again into a 10×20-inch rectangle. Fold the dough lengthwise into thirds as if you were folding a letter.

Long rest: Place the folded dough on the lined baking sheet, cover with plastic wrap or aluminum foil, and refrigerate for 4 hours or overnight. (Up to 24 hours is ok.)

At the end of the next step, you’ll need 2 baking sheets lined with silicone baking mats or parchment paper. The dough is currently on a lined baking sheet in the refrigerator, so you already have 1 prepared!

Shape the croissants: Remove the dough from the refrigerator. On a lightly floured counter, roll the dough out into an 8×20-inch rectangle. Use your fingers if you need to. Once again, the dough is very cold, so it will take a lot of arm muscle to roll. The dough will want to be oval shaped, but keep working it with your hands and rolling pin until you have the correct size rectangle. Using a pizza cutter or sharp knife, slice the dough in half vertically. Each skinny rectangle will be 4 inches wide. Then cut 3 even slices horizontally, yielding 8 4×5-inch rectangles. See photo above for a visual. Cut each rectangle in half lengthwise so you have 16 2×5-inch rectangles. Work with one rectangle at a time. Using your fingers or a rolling pin, stretch it to be about 8 inches long. Do this gently as you do not want to flatten the layers. Place a few small pieces of chocolate in a single layer at one end and tightly roll the dough up around the chocolate. Make sure the end is on the bottom. Repeat with remaining dough, placing the shaped croissants on 2 lined baking sheets, 8 per sheet. Loosely cover with plastic wrap or aluminum foil and allow to rest at room temperature (no warmer! I suggest just keeping on the counter) for 1 hour, then place in the refrigerator to rest for 1 hour or up to 12 hours. (Or freeze, see freezing instructions. I prefer the shaped croissants to be cold going into the oven.

Preheat oven to 400°F (204°C).

Egg wash: Whisk the egg wash ingredients together. Remove the croissants from the refrigerator. Brush each lightly with egg wash.

Bake the croissants: Bake until croissants are golden brown, about 20 minutes. Rotate the pans halfway through baking. If croissants show signs of darkening too quickly, reduce the oven to 375°F (190°C).

Remove chocolate croissants from the oven and place on a wire rack to cool for a few minutes before serving. They will slightly deflate as they cool. If desired, dust with confectioners’ sugar before serving.

Croissants taste best the same day they’re baked. Cover any leftover croissants and store at room temperature for a few days or in the refrigerator for up to 1 week. You can also freeze for up to 3 months, then thaw on the counter or overnight in the refrigerator. Warm up to your liking.', 16),

(20004, 10004, 'Bring 1 cup of water to a boil.

Add the chai tea bags, cinnamon, nutmeg and maple syrup. Cover and let steep for 10 minutes.', 1),

(20005, 10005, 'Heat oven to 400 degrees Fahrenheit. Line 10 muffin cups with paper liners.Whisk the flour, sugar, baking powder, and salt in a large bowl.Add oil to a measuring jug that holds at least 1 cup. Add the egg then fill the jug to the 1-cup line with milk (1/3 to 1/2 cup milk). Add vanilla and whisk to combine.Add milk mixture to the bowl with dry ingredients then use a fork to combine. Do not over mix. (The muffin batter will be quite thick — see note below for more details). Fold in the blueberries.Divide the batter between muffin cups. (If making big-topped muffins, the batter will come to the tops of the paper liners). Sprinkle a little sugar on top of each muffin.Bake muffins 15 to 20 minutes or until tops are no longer wet and a toothpick inserted into the middle of a muffin comes out with crumbs, not wet batter. Transfer to a cooling rack.', 10)

GO

INSERT INTO Inventory VALUES

(30000, 'Ground Coffee', 100, 'OZ', .60),

(30001, 'Whole Milk', 640, 'OZ', .02),

(30002, 'Unsweetened Cocoa Powder', 80, 'OZ', .43),

(30003, 'Granulated White Sugar', 112, 'OZ', .31),

(30004, 'Unsalted Butter', 160, 'OZ', .31),

(30005, 'All-Purpose Flour', 256, 'OZ', .16),

(30006, 'Salt', 52, 'OZ', .77),

(30007, 'Dry Yeast', 18, 'OZ', .51),

(30008, 'Semi-Sweet Chocolate Chips', 48, 'OZ', .79),

(30009, 'Large Eggs', 120, 'CT', .35),

(30010, 'Chai Tea Bags', 100, 'CT', .20),

(30011, 'Ground Cinnamon', 20, 'OZ', .73),

(30012, 'Ground Nutmeg', 20, 'OZ', .88),

(30013, 'Honey', 80, 'OZ', .29),

(30014, 'Baking Powder', 70, 'OZ', .43),

(30015, 'Vegetable Oil', 256, 'OZ', .09),

(30016, 'Vanilla Extract', 16, 'OZ', 1.75),

(30017, 'Frozen Blueberries', 48, 'OZ', 3.00)

GO

INSERT INTO IngredientsToRecipe VALUES

(20000, 30000, .5, 'OZ'),

(20000, 30001, 2, 'OZ'),

(20001, 30000, 1, 'OZ'),

(20001, 30001, 1, 'OZ'),

(20001, 30002, .5, 'OZ'),

(20001, 30003, .5, 'OZ'),

(20002, 30000, 2.25, 'OZ'),

(20003, 30001, 13, 'OZ'),

(20003, 30003, 2, 'OZ'),

(20003, 30004, 14, 'OZ'),

(20003, 30005, 33, 'OZ'),

(20003, 30006, .33, 'OZ'),

(20003, 30007, .5, 'OZ'),

(20003, 30008, 4, 'OZ'),

(20003, 30009, 1, 'CT'),

(20004, 30001, 8, 'OZ'),

(20004, 30010, 2, 'CT'),

(20004, 30011, .05, 'OZ'),

(20004, 30012, .05, 'OZ'),

(20004, 30013, 1, 'OZ'),

(20005, 30001, 4, 'OZ'),

(20005, 30003, 6, 'OZ'),

(20005, 30005, 12, 'OZ'),

(20005, 30006, .04, 'OZ'),

(20005, 30009, 1, 'CT'),

(20005, 30014, .33, 'OZ'),

(20005, 30015, 2.64, 'OZ'),

(20005, 30016, .25, 'OZ'),

(20005, 30017, 8, 'OZ')

INSERT INTO Employees (EmployeeID, FirstName, LastName, DateOfBirth, Email, Phone, Department, [Pay])

VALUES (026580000, 'Ethan', 'Anderson', '1990-02-12', 'EthanAnderson37@gmail.com', '8642019375', 'Management', '30'),

(026580001, 'Ava', 'Mitchell', '2003-07-02', 'AvaMitchell35@yahoo.com', '8642035350', 'Shift Supervisor', '18.75'),

(026580002, 'Liam', 'Thompson', ' 2000-04-30', 'LiamThompson38@gmail.com', '8642038380', 'Shift Supervisor', '18'),

(026580003, 'Isabella', 'Rodriguez', '2002-02-12', 'IsabellaRodriguez37@gmail.com', '8642037370', 'Shift Supervisor', '18'),

(026580004, 'Noah', 'Johnson', '2004-09-27', 'NoahJohnson27@gmail.com', '8642027270', 'Shift Supervisor', '18'),

(026580005, 'Aria', 'Patel', '1989-08-07', 'AriaPatel25@gmail.com', '8642025250', 'Baker', '15'),

(026580006, 'Sebastian', 'Wright', '2005-03-18', 'SebastianWright41@gmail.com', '8642041410', 'Baker', '15'),

(026580007, 'Aurora', 'Ramirez', '1999-11-23', 'AuroraRamirez30@gmail.com', '8642030300', 'Baker', '15.3'),

(026580009, 'Mateo', 'Gonzalez', '2003-09-06', 'MateoGonzalez32@example.com', '8642032320', 'Bookkeeper', '20'),

(026580010, 'Esmeralda ', 'Silva', '1998-12-11', 'EsmeraldaSilva46@gmail.com', '8642046460', 'Trainee', '15'),

(026580011, 'Kailynn', 'Liebau', '2004-07-22', 'Kailynn@gmail.com', '8642046460', 'Barista', '15'),

(026580012, 'Brett', 'Fowler', '2000-01-01', 'Brett@yahoo.com', '8642026260', 'Barista', '15'),

(026580013, 'Nathaniel', 'Bagwell', '2000-01-01', 'Nathaniel@yahoo.com', '8642042420', 'Barista', '15'),

(026580014, 'Nathan', ' Richardson', '2000-01-01', 'Nathan@gmail.com', '8642031310', 'Barista', '15'),

(026580015, 'Benjamin', 'Wilson', '2001-10-15', 'BenjaminWilson46@gmail.com', '8642036360', 'Barista', '15'),

(026580016, 'Maya', 'Lopez', '1999-06-29', 'MayaLopez26@gmail.com', '8642029290', 'Barista', '15'),

(026580017, 'Elijah', 'Smith', '2002-01-08', 'ElijahSmith42@yahoo.com', '8642042420', 'Barista', '15'),

(026580018, 'Suri', 'Davis', '2001-11-09', 'SuriDavis31@gmail.com', '8642027270', 'Barista', '15'),

(026580019, 'Lucas', 'Brown', '1981-03-19', 'LucasBrown36@gmail.com', '8645965853', 'Barista', '15'),

(026580020, 'Zahara', 'Khan', '2004-05-20', 'ZaharaKhan42@gmail.com', '8642043430', 'Barista', '15')

GO

--  0 - 4AM 4 TO 8AM 8AM T 16AM 16TO 24A

INSERT INTO WorkSchedule (ScheduleID, EmployeeID, ScheduleDate, StartTime, EndTime, ShiftType)

VALUES

(0001, 20, '2023-07-17', 9, 14, 'MN'),

(0002, 20, '2023-07-18', 12, 14, 'AFT'),

(0003, 20, '2023-07-19', 12, 14, 'AFT'),

(0004, 20, '2023-07-20', 8, 12, 'MN'),

(0005, 20, '2023-07-21', 13, 17, 'AFT'),

--Bookkeeper

(0006, 9, '2023-07-17', 10, 17, 'AFT'),

(0007, 9, '2023-07-20', 10, 17, 'AFT'),

--Monday +baker, shift lead

(0011, 1, '2023-07-17', 0, 4, 'EMN'),

(0012, 5, '2023-07-17', 0, 4, 'EMN'),

(0013, 8, '2023-07-17', 0, 4, 'EMN'),

(0014, 12, '2023-07-17', 4, 8, 'MN'),

(0015, 3, '2023-07-17', 4, 8, 'MN'),

(0016, 10, '2023-07-17', 4, 8, 'MN'),

(0017, 11, '2023-07-17', 8, 16, 'AFT'),

(0018, 3, '2023-07-17', 8, 16, 'AFT'),

(0019, 13, '2023-07-17', 8, 16, 'AFT'),

(0020, 14, '2023-07-17', 16, 24, 'NGT'),

(0021, 4, '2023-07-17', 16, 24, 'NGT'),

(0022, 15, '2023-07-17', 16, 24, 'NGT'),

--Tuesday +baker

(0023, 2, '2023-07-18', 0, 4, 'EMN'),

(0024, 16, '2023-07-18', 0, 4, 'EMN'),

(0025, 17, '2023-07-18', 0, 4, 'EMN'),

(0026, 1, '2023-07-18', 4, 8, 'MN'),

(0027, 18, '2023-07-18', 4, 8, 'MN'),

(0028, 19, '2023-07-18', 4, 8, 'MN'),

(0029, 20, '2023-07-18', 8, 16, 'AFT'),

(0030, 2, '2023-07-18', 8, 16, 'AFT'),

(0031, 6, '2023-07-18', 8, 16, 'AFT'),

(0032, 7, '2023-07-18', 16, 24, 'NGT'),

(0033, 8, '2023-07-18', 16, 24, 'NGT'),

(0034, 15, '2023-07-18', 16, 24, 'NGT'),

--Wed

(0035, 1, '2023-07-19', 0, 4, 'EMN'),

(0036, 5, '2023-07-19', 0, 4, 'EMN'),

(0037, 10, '2023-07-19', 0, 4, 'EMN'),

(0038, 11, '2023-07-19', 4, 8, 'MN'),

(0039, 12, '2023-07-19', 4, 8, 'MN'),

(0040, 13, '2023-07-19', 4, 8, 'MN'),

(0041, 4, '2023-07-19', 8, 16, 'AFT'),

(0042, 7, '2023-07-19', 8, 16, 'AFT'),

(0043, 14, '2023-07-19', 8, 16, 'AFT'),

(0044, 3, '2023-07-19', 16, 24, 'NGT'),

(0045, 15, '2023-07-19', 16, 24, 'NGT'),

(0046, 16, '2023-07-19', 16, 24, 'NGT'),

--thrus

(0047, 17, '2023-07-20', 0, 4, 'EMN'),

(0048, 5, '2023-07-20', 0, 4, 'EMN'),

(0049, 18, '2023-07-20', 0, 4, 'EMN'),

(0050, 19, '2023-07-20', 4, 8, 'MN'),

(0051, 1, '2023-07-20', 4, 8, 'MN'),

(0052, 20, '2023-07-20', 4, 8, 'MN'),

(0053, 5, '2023-07-20', 8, 16, 'AFT'),

(0054, 4, '2023-07-20', 8, 16, 'AFT'),

(0055, 6, '2023-07-20', 8, 16, 'AFT'),

(0056, 7, '2023-07-20', 16, 24, 'NGT'),

(0057, 6, '2023-07-20', 16, 24, 'NGT'),

(0058, 8, '2023-07-20', 16, 24, 'NGT'),

--fri

(0059, 9, '2023-07-21', 0, 4, 'EMN'),

(0060, 1, '2023-07-21', 0, 4, 'EMN'),

(0061, 10, '2023-07-21', 0, 4, 'EMN'),

(0062, 11, '2023-07-21', 4, 8, 'MN'),

(0063, 2, '2023-07-21', 4, 8, 'MN'),

(0064, 5, '2023-07-21', 4, 8, 'MN'),

(0065, 12, '2023-07-21', 8, 16, 'AFT'),

(0066, 13, '2023-07-21', 8, 16, 'AFT'),

(0067, 14, '2023-07-21', 8, 16, 'AFT'),

(0068, 9, '2023-07-21', 16, 24, 'NGT'),

(0069, 15, '2023-07-21', 16, 24, 'NGT'),

(0070, 16, '2023-07-21', 16, 24, 'NGT'),

--sat

(0071, 12, '2023-07-22', 0, 4, 'EMN'),

(0072, 17, '2023-07-22', 0, 4, 'EMN'),

(0073, 18, '2023-07-22', 0, 4, 'EMN'),

(0074, 19, '2023-07-22', 4, 8, 'MN'),

(0075, 2, '2023-07-22', 4, 8, 'MN'),

(0076, 20, '2023-07-22', 4, 8, 'MN'),

(0077, 7, '2023-07-22', 8, 16, 'AFT'),

(0078, 5, '2023-07-22', 8, 16, 'AFT'),

(0079, 4, '2023-07-22', 8, 16, 'AFT'),

(0080, 11, '2023-07-22', 16, 24, 'NGT'),

(0081, 3, '2023-07-22', 16, 24, 'NGT'),

(0082, 10, '2023-07-22', 16, 24, 'NGT'),

--sun

(0083, 1, '2023-07-23', 0, 4, 'EMN'),

(0084, 5, '2023-07-23', 0, 4, 'EMN'),

(0085, 17, '2023-07-23', 0, 4, 'EMN'),

(0086, 18, '2023-07-23', 4, 8, 'MN'),

(0087, 20, '2023-07-23', 4, 8, 'MN'),

(0089, 19, '2023-07-23', 8, 16, 'AFT'),

(0090, 3, '2023-07-23', 8, 16, 'AFT'),

(0091, 14, '2023-07-23', 8, 16, 'AFT'),

(0092, 6, '2023-07-23', 16, 24, 'NGT'),

(0093, 4, '2023-07-23', 16, 24, 'NGT'),

(0094, 7, '2023-07-23', 16, 24, 'NGT');

GO

**---** **Paste below this line the completion status messages from executing your team’s insert statements-**

Paste here

(6 rows affected)

(6 rows affected)

(18 rows affected)

(29 rows affected)

(20 rows affected)

(90 rows affected)

Completion time: 2023-07-24T12:27:29.1208223-04:00

# **Group Project Exercise 3 (View s)**

Create a view named GPViewXXXXXXXXXXXX where the XXXXXXXXXXXX is the last name of one of your team members. This view must use at least two of your tables and the result set needs to be meaningful.

**--- Paste below this line your team’s view creation code. –**

Paste here

A screenshot of a computer

Description automatically generated

--create a view with the employee and workschedule table to find how many hour each employee works.

CREATE VIEW GPViewLiebau  AS

SELECT

    w.EmployeeID,

    e.FirstName,

    e.LastName,

    SUM(EndTime - StartTime) AS TotalHoursWorked

FROM

    Employees e JOIN WorkSchedule w ON e.EmployeeID = w.EmployeeID

WHERE

    w.ScheduleDate BETWEEN '2023-07-17' AND '2023-07-23'

GROUP BY

    w.EmployeeID,

    e.FirstName,

    e.LastName;

**---** **Paste below this line the completion status messages from executing your team’s view creation code--**

Paste here

A screen shot of a computer

Description automatically generated

**---** **Paste below this line the result set returned by running a SELECT \* from the view your team created. --**

Paste here

A screenshot of a computer

Description automatically generated

# **Group Project Exercise 4 (Stored Procedures)**

Create a stored procedure named GPProcXXXXXXXXXXXX where the XXXXXXXXXXXX is the last name of one of your team members. This stored procedure must contain logic that involves at least two of your tables and it must produce a meaningful report when executed.

**--- Paste below this line your team’s stored procedure creation code. –**

Paste here

USE BFGPDatabase;

GO

CREATE PROC GPProcFowler

@RecipeItem varchar(60) = '%',

@ServingsNeeded decimal = 0

AS

IF @ServingsNeeded <= 0

BEGIN

SELECT MenuName, Recipes.RecipeID, Inventory.InventoryID, ItemName,

CONVERT(decimal(10,2), (IngredientsToRecipe.IngAmount / Recipes.NumberOfServings)) AS TotalIngredient,

IngUnitOfMeasure,

CONVERT(decimal(10,2), (Inventory.InvCostPerUnit \* (IngredientsToRecipe.IngAmount / Recipes.NumberOfServings))) AS TotalCost

FROM Menu JOIN Recipes

ON Menu.MenuID = Recipes.MenuID

JOIN IngredientsToRecipe

ON Recipes.RecipeID = IngredientsToRecipe.RecipeID

JOIN Inventory

ON IngredientsToRecipe.InventoryID = Inventory.InventoryID

WHERE MenuName LIKE @RecipeItem

ORDER BY IngAmount DESC;

END;

ELSE

BEGIN

SELECT MenuName, Recipes.RecipeID, Inventory.InventoryID, ItemName,

CONVERT(decimal(10,2), (IngredientsToRecipe.IngAmount / Recipes.NumberOfServings) \* @ServingsNeeded) AS TotalIngredient,

IngUnitOfMeasure,

CONVERT(decimal(10,2), (Inventory.InvCostPerUnit \* ((IngredientsToRecipe.IngAmount / Recipes.NumberOfServings) \* @ServingsNeeded))) AS TotalCost

FROM Menu JOIN Recipes

ON Menu.MenuID = Recipes.MenuID

JOIN IngredientsToRecipe

ON Recipes.RecipeID = IngredientsToRecipe.RecipeID

JOIN Inventory

ON IngredientsToRecipe.InventoryID = Inventory.InventoryID

WHERE MenuName LIKE @RecipeItem

ORDER BY IngAmount DESC;

END;

**---** **Paste below this line the completion status messages from executing your team’s stored procedure creation code--**

Paste here

Commands completed successfully.

Completion time: 2023-07-26T08:15:11.4962186-04:00

**---** **Paste below the execute statement your team used to test the stored procedure --**

Paste here

USE BFGPDatabase;

EXEC GPProcFowler @RecipeItem = 'Chocolate Croissant', @ServingsNeeded = 20;

**---** **Paste below this line the run results from the statement used to test the stored procedure--**

Paste here

A screenshot of a menu

Description automatically generated

# **Group Project Exercise 5 (Triggers)**

Code a trigger named TR\_GP\_DML\_XXXXXXXXXXXXXX where the XXXXXXXXXXXXXX is the last name of one of your team members. This trigger must respond to inserts and/or updates to one of your tables and the trigger must perform a meaningful purpose.

**--- Paste below this line your team’s trigger creation code. –**

Paste here

USE BFGPDatabase

GO

CREATE TRIGGER TR\_GP\_DML\_Fowler

ON WorkSchedule

AFTER INSERT, UPDATE AS

BEGIN

IF

(SELECT COUNT(\*)

FROM WorkSchedule JOIN Inserted

ON WorkSchedule.EmployeeID = Inserted.EmployeeID

WHERE WorkSchedule.ScheduleDate = Inserted.ScheduleDate) > 1

BEGIN

ROLLBACK TRAN;

THROW 50001, 'This employee is already scheduled for this date.', 1;

END;

END;

**---** **Paste below this line the completion status messages from executing your team’s trigger creation code--**

Paste here

Commands completed successfully.

Completion time: 2023-07-31T11:40:32.2818885-04:00

**---** **Paste below this line the insert and/or update statements your team used to test the trigger --**

Paste here

USE BFGPDatabase;

INSERT INTO WorkSchedule

VALUES (9, 26580000, '2023-07-17', '01:00:00', '02:00:00');

USE BFGPDatabase;

INSERT INTO WorkSchedule

VALUES (9, 26580000, '2023-08-17', '01:00:00', '02:00:00');

**---** **Paste below this line the run results from executing the statements used to test the trigger (include comments explaining what each test demonstrated) --**

Paste here

The trigger checks to see if the employee schedule date includes a date that that employee has already been scheduled for.

To test that the trigger will throw an error if the scheduled date is the same, the first insert statement was used with a date that is already on the work schedule table for that employee. This test threw the following message:

Msg 50001, Level 16, State 1, Procedure TR\_GP\_DML\_Fowler, Line 13 [Batch Start Line 0]

This employee is already scheduled for this date.

Completion time: 2023-07-31T11:44:10.3073986-04:00

To test that the trigger will allow an insert statement for a date that is not already on the schedule for the employee, the second statement was used.  
This test threw the following message:

(1 row affected)

Completion time: 2023-07-31T11:51:42.1795467-04:00

# **Group Project Exercise 6 (Security)**

Write a script that creates a user-defined database role in the XXGPDatabase database (reminder replace the XX with the appropriate initials). Name this new role ProductManagementXX where the XX are your initials. Give UPDATE permission to the new role for the tables that contain menu, recipe, and inventory data, and SELECT permission for all other user tables. Make sure to include a comment at the top of the script with your name and the date you coded the script. Format this comment like the following sample:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* ProductManagementXX Role Creation Script \*/

/\* Coded by: Your Name \*/

/\* Coded on: MM/DD/YY \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**---** **Paste below this line the Script your team wrote for this exercise --**

Paste here

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* ProductManagementNR Role Creation Script                    \*/

/\* Coded by: Nathan Richardson                                 \*/

/\* Coded on: 07/12/23                                          \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

USE BFGPDatabase;

CREATE ROLE ProductManagementNR;

GRANT UPDATE ON dbo.Menu TO ProductManagementNR;

GRANT UPDATE ON dbo.Recipes TO ProductManagementNR;

GRANT UPDATE ON dbo.Inventory TO ProductManagementNR;

GRANT SELECT ON SCHEMA::dbo TO ProductManagementNR;

**---** **Paste below this line the run results from executing the script your team wrote--**

Paste here

A screenshot of a computer

Description automatically generated

# **Group Project Exercise 7 (Security)**

Use the XXGPDatabase database and write a script that:

1. Creates a login ID named “GPDLOG01XX” where the XX are your initials
2. Assigns this login an initial password of “GPDr33333”
3. Sets the default database for this login to the XXGPDatabase database
4. Creates a user named “GPDLOG01XX” where the XX are your initials
5. Assigns the “GPDLOG01XX” user to the ProductManagementXX role you created in exercise 6
6. Includes a comment at the top of the script with your name and the date you coded the script. Format this comment like the following sample:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* GPDLOG01XX Login and User Creation Script \*/

/\* Coded by: Your Name \*/

/\* Coded on: MM/DD/YY \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**--- Paste below this line the Script your team wrote for this exercise --**

Paste here

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* GPDLOG01NR Login and User Creation Script                  \*/

/\* Coded by: Nathan Richardson                                \*/

/\* Coded on: 07/12/23                                         \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

USE BFGPDatabase;

CREATE LOGIN GPDLOG01NR WITH PASSWORD = 'GPDr33333';

ALTER LOGIN GPDLOG01NR WITH DEFAULT\_DATABASE = BFGPDatabase;

CREATE USER GPDLOG01NR FOR LOGIN GPDLOG01NR;

ALTER ROLE ProductManagementNR ADD MEMBER GPDLOG01NR;

**---** **Paste below this line the run results from executing the script your team wrote--**

Paste here

A screenshot of a computer

Description automatically generated

# **Group Project Exercise 8 (Security)**

Write a script that uses a table, dynamic SQL, and a cursor to create login IDs in the XXGPDatabase database based on the contents of the table.

1. Get started by first creating the table. Below is a create statement that you can modify and execute to create this table. Make sure to change the XX in the table name to your initials:

CREATE TABLE NewLoginsXX (LoginName varchar(128));

1. After the table is created. Insert four rows in it. Below is an insert statement that you can modify and execute to put the rows in your NewLoginsXX table. In addition to modifying the table name, make sure to change the XX at the start of each login name to your initials before running this insert.

INSERT NewLoginsXX VALUES ('XXFFlake'), ('XXGGlute'), ('XXMMrit'), ('XXPPert');

1. Start coding the script with a comment block. Format this comment like the following sample (make sure to revise the sample to reflect your name and the date you coded it):

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Login Generator Script \*/

/\* This Script uses dynamic SQL and a cursor to process the \*/

/\* NEWLoginsXX table one row at a time. For each row on this \*/

/\* table the script will perform four actions. \*/

/\* \*/

/\* Coded by: Your Name \*/

/\* Coded on: MM/DD/YY \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. Code the rest of your script using dynamic SQL and a cursor to perform four actions for each row in this table:

(1) create a login with a temporary password that's based on the first four letters of the login name followed by “33333”

(2) set the default database to the XXGPDatabase database

(3) create a user for the login with the same name as the login

(4) assign the user to the ProductManagementXX role you created in exercise 6.

**--- Paste below this line the Script your team wrote for this exercise --**

Paste here

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Login Generator Script                                     \*/

/\* This Script uses dynamic SQL and a cursor to process the   \*/

/\* NEWLoginsNR table one row at a time. For each row on this  \*/

/\* table the script will perform four actions.                \*/

/\*                                                            \*/

/\* Coded by: Nathan Richardson                                \*/

/\* Coded on: 08/01/23                                         \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE TABLE NewLoginsNR (LoginName varchar(128));

INSERT NewLoginsNR VALUES ('NRFFlake'), ('NRGGlute'), ('NRMMrit'), ('NRPPert');

DECLARE @LoginName varchar(128);

DECLARE @SQL nvarchar(max);

DECLARE LoginCursor CURSOR FOR

SELECT LoginName

FROM NewLoginsNR;

OPEN LoginCursor;

FETCH NEXT FROM LoginCursor INTO @LoginName;

WHILE @@FETCH\_STATUS = 0

BEGIN

    DECLARE @TempPassword varchar(128) = LEFT(@LoginName, 4) + '33333';

    SET @SQL = 'CREATE LOGIN ' + QUOTENAME(@LoginName) + ' WITH PASSWORD = ''' + @TempPassword + ''';';

    EXEC sp\_executesql @SQL;

    SET @SQL = 'ALTER LOGIN ' + QUOTENAME(@LoginName) + ' WITH DEFAULT\_DATABASE = BFGPDatabase;';

    EXEC sp\_executesql @SQL;

    SET @SQL = 'CREATE USER ' + QUOTENAME(@LoginName) + ' FOR LOGIN ' + QUOTENAME(@LoginName) + ';';

    EXEC sp\_executesql @SQL;

    SET @SQL = 'ALTER ROLE ProductManagementNR ADD MEMBER ' + QUOTENAME(@LoginName) + ';';

    EXEC sp\_executesql @SQL;

    FETCH NEXT FROM LoginCursor INTO @LoginName;

END

CLOSE LoginCursor;

DEALLOCATE LoginCursor;

DROP TABLE NewLoginsNR;

**---** **Paste below this line the run results from executing the script your team wrote—**

Paste here

A screenshot of a computer

Description automatically generated

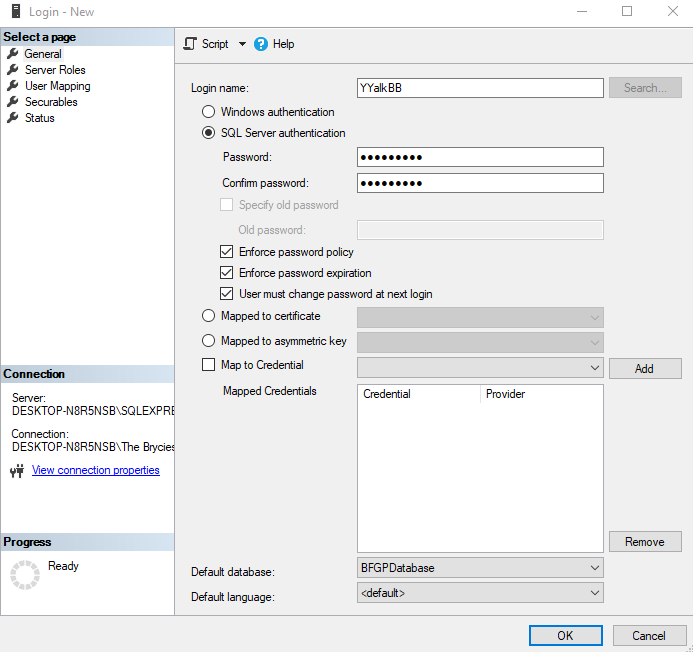
# **Group Project Exercise 9 (Security)**

Using the Management Studio, create a login ID named “YYalkXX” where the XX are your initials. Assign this login ID a password “YYal33333,” and set the default database to the XXGPDatabase database. Then, grant the login ID access to the XXGPDatabase database, create a user for the login ID named “YYalkXX”, and assign the user to the PaymentEntry role you created in exercise 6.

Note: If you get an error that says, “The MUST\_CHANGE option is not supported”, you can deselect the “Enforce password policy” option for the login ID.

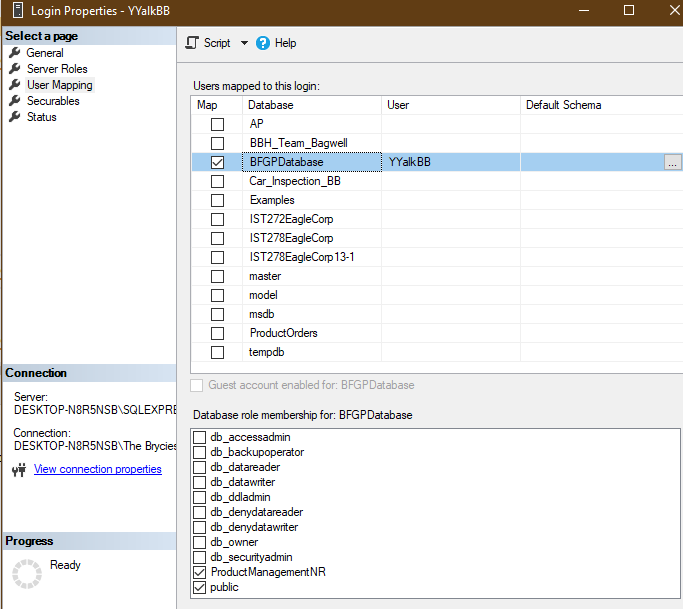
**---** **Paste below this line a screen shot that shows the new login creation window with the YYalkXX information showing—**

Paste here



**---** **Paste below a screen shot that shows the login properties window with the YYalkXX’s mapping to the XXGPDatabase database and membership in the ProductManagementXX role your team created in exercise 6.--**

Paste here



# **Group Project Exercise 10 (DDL / XML Trigger)**

* 1. Create a table named AuditDDLXXXXXXXXXXX (where the XXXXXXXXXXXX is the last name of one of your team members). Create this table with the following columns:

AuditEventID INT IDENTITY(1,1) PRIMARY KEY,

AuditEventType varchar(20),

AuditEventData xml

**--- Paste below this line the create table code your team wrote and executed for this exercise --**

Paste here

A screenshot of a computer

Description automatically generated

CREATE TABLE AuditDDL\_LIEBAU (

  AuditEventID INT IDENTITY(1,1) PRIMARY KEY,

  AuditEventType VARCHAR(20),

  AuditEventData XML

);

* 1. Create a trigger named TR\_DB\_DDL\_XXXXXXXXXXXXXX where the XXXXXXXXXXXXXX is the last name of one of your team members. Code this trigger so that it automatically inserts a row into the AuditDDLXXXXXXXXXXX table every time a create table or drop table statement is executed in the xxGPDatabase. Hint: The trigger can use the EVENTDATA() function to get the values to put in the AuditEventType and the AuditEventData attributes.

**--- Paste below this line the create trigger code your team wrote and executed for this exercise --**

Paste here

A screenshot of a computer program

Description automatically generated

* 1. Create a table named MyTeamXXXXXXXXXXX (where the XXXXXXXXXXXX is the last name of one of your team members). Create this table with the following columns:

MemberID INT Identity(1,10) PRIMARY KEY,

MemberFirstName VARCHAR(15),

MemberLastName VarChar(20)

**--- Paste below this line the create table code your team wrote and executed for this exercise --**

Paste here

A screenshot of a computer

Description automatically generated

* 1. Code and execute a SELECT \* FROM AuditDDLXXXXXXXXXXX table.

**---** **Paste below this line the run results from executing the SELECT \* FROM AuditDDLXXXXXXXXXXX —**

Paste here

A close-up of a web page

Description automatically generated

* 1. Click on the AuditEventData value present in the result set (this should cause the Management Studio to show the value in an XML editor)

**--- Paste below this line the AuditEventData displayed in the XML editor —**

Paste here

A screenshot of a computer code

Description automatically generated