CPSC 304 Project Cover Page

Milestone #: 2

Date: Feb 21, 2024

Group Number: 52

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
George Song	27473412	t9k1b	georgesong97@gmail.com
Jayden Piao	83679589	k7v0w	jaydenpiao@gmail.com
Kohen Lee	12154647	m3b8	Kohenlee1234@gmail.com

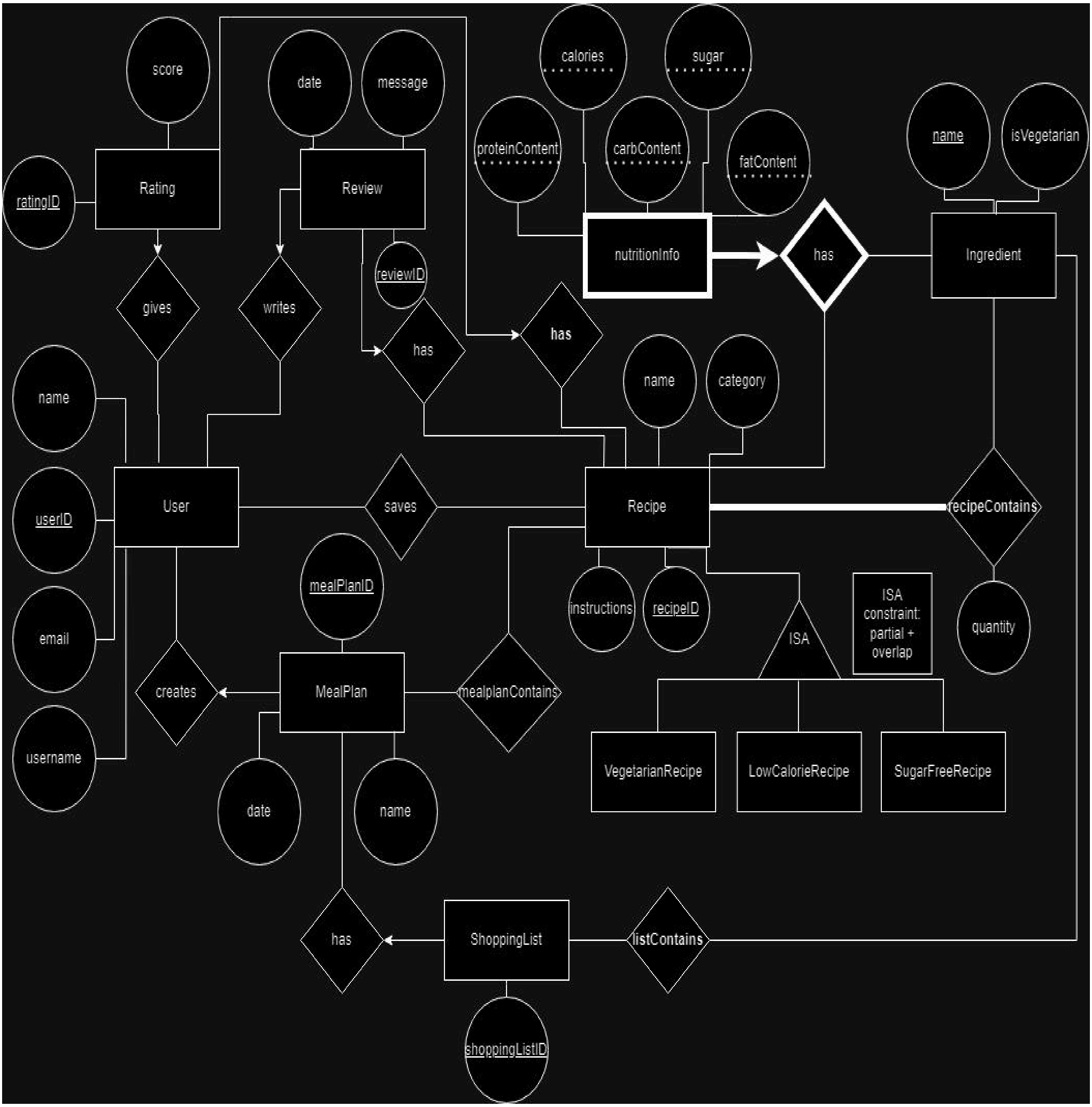
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

1.A brief (~2-3 sentences) summary of your project

Our project idea is creating an application that stores recipes which can be separated into unique categories. Users will be able to create meal plans and keep track of their grocery needs.

2.The ER diagram you are basing your item #3 (below) on.



Changes made to diagram:

- changed repetitive relation names (has, contains) to be more specific (Ex. recipeContains, mealPlanHas)
- Changed review and recipe to many-to-one & review and user to many-to-one
- Changed rating and recipe to many-to-one & rating and user to many-to-one
- Changed meal plan and user to one to many
- Changed meal plan and shopping list to one to many
- Changed the key for shopping list to shoppingListID
- Changed the key for mealPlan to mealplanID
- Added quantity as an attribute to ingredient recipe relation
- Added additional attributes for normalization step in recipe and user

3.The schema derived from your ER diagram (above). For the translation of the ER diagram to the relational model, follow the same instructions as in your lectures. The process should be reasonably straightforward. For each table:

a. List the table definition (e.g., Table1(attr1: domain1, attr2: domain2, ...)). Make sure to include the domains for each attribute

b. Specify the primary key (PK), candidate key, (CK) foreign keys (FK), and other constraints (e.g., not null, unique, etc.) that the table must maintain.

Entities:

- User(userID: int, name: varchar, email: varchar, username: varchar)

  PK: userID

  CK: userID, email

  FK: N/A

  Other: email is UNIQUE and NOT NULL, name is NOT NULL, username is UNIQUE

- Recipe(recipeID: int, name: varchar, category: varchar, instructions: varchar)

  PK: recipeID

  CK: recipeID

  FK: None

  Other: name is NOT NULL

- VegetarianRecipe(recipeID: int)

  PK: recipeID

  CK: recipeID

  FK: recipeID

  Other: ON UPDATE CASCADE and ON DELETE CASCADE on recipeID

- LowCalorieRecipe(recipeID: int)

  PK: recipeID

  CK: recipeID

  FK: recipeID

  Other: ON UPDATE CASCADE and ON DELETE CASCADE on recipeID

- SugarFreeRecipe(recipeID: int)

  PK: recipeID

  CK: recipeID

  FK: recipeID

  Other: ON UPDATE CASCADE and ON DELETE CASCADE on recipeID

- Ingredient(name: varchar, isVegetarian: BOOLEAN)

  PK: name

CK: name

FK: N/A

- nutritionInfoRecipe(calories: int, sugar: int, proteinContent: int, fatContent: int, carbContent: int, recipeID: int)

  PK: all

  CK: all

  FK: recipeID

  Other: calories, sugar, protein, fat, and carb > 0, ON UPDATE CASCADE and ON DELETE CASCADE on name and recipeID

- nutritionInfoIngredient(calories: int, sugar: int, proteinContent: int, fatContent: int, carbContent: int, ingredientName:varchar)

  PK: all

  CK: all

  FK: ingredientName

  Other: calories, sugar, protein, fat, and carb > 0, ON UPDATE CASCADE and ON DELETE CASCADE on ingredientName

- MealPlan(mealPlanID: int, name: varchar, date: date, userID: int)

  PK:mealPlanID

  CK: mealPlanID

  FK: userID

- shoppingList(shoppingListID: int, mealPlanID: int)

  PK: shoppingListID

  CK: shoppingListID

  FK: mealPlanID

Other: ON UPDATE CASCADE and ON DELETE CASCADE mealplanID

- Rating(ratingID: int, score: int, userID: int, recipeID: int)

  PK: ratingID

  CK: ratingID, recipeID + userID

  FK: recipeID, userID

  Other: 0 <= Score <= 5, ON DELETE CASCADE on userID and recipeID

- Review(reviewID: int, date: date, message: varchar, userID: int, recipeID: int)

  PK:reviewID

  CK: reviewID, recipeID + userID

  FK: recipeID, userID

  Other: UNIQUE and NOT NULL user ID, ON DELETE CASCADE and ON UPDATE CASCADE for userID and recipeID

Relationships:

- saves(userID: int, recipeID: int)

  PK: userID + recipeID

  CK: userId + recipeID

  FK: userID, recipeID

  Other: ON UPDATE CASCADE and ON DELETE CASCADE on recipeID and userID

- recipeContains(recipeID: int, ingredientName: varchar)

  PK: recipeID + ingredientName

  CK: recipeID + ingredientName

  FK: recipeID, ingredientName

Other: ON UPDATE CASCADE and ON DELETE CASCADE on recipeID and ingredientName

- mealPlanContains(mealPlanID: int, recipeID:int)

  PK: recipeID + mealPlanID

  CK: recipeID + mealPlanID

  FK: recipeID, mealPlanID

  Other: ON UPDATE CASCADE and ON DELETE CASCADE on recipeID and mealPlanID

- listContains(shoppingListID:int, ingredientName: varchar)

  PK: shoppingListID + ingredientName

  CK: shoppingListID + ingredientName

  FK: shoppingListID, ingredientName

  Other: ON UPDATE CASCADE and ON DELETE CASCADE on shoppingListID and ingredientName

5. Functional Dependencies (FDs) a. Identify the functional dependencies in your relations, including the ones involving all candidate keys (including the primary key). PKs and CKs are considered functional dependencies and should be included in the list of FDs. You do not need to include trivial FDs such as A → A.

Note: In your list of FDs, there must be some kind of valid FD other than those identified by a PK or CK. If you observe that no relations have FDs other than the PK and CK(s), then you will have to intentionally add some (meaningful) attributes to show valid FDs. We want you to get a good

normalization exercise. Your design must go through a normalization process. You do not need to have a non-PK/CK FD for each relation but be reasonable. If your TA feels that some non-PK/CK FDs have been omitted, your grade will be adjusted accordingly.

- User(userID: int, name: varchar, email: varchar, username: varchar)

  userID -> name, email,username

  username -> name

- Recipe(recipeID: int, name: varchar, category: varchar)

  recipeID -> recipe.name

  recipe.name-> category

- Ingredient(name: varchar, isVegetarian: BOOLEAN)

  ingredient.name -> isVegetarian

- nutritionInfoRecipe(calories: int, sugar: int, proteinContent: int, fatContent: int, carbContent: int, recipeID: int)

  recipeID -> calories, sugar, proteinContent, fatContent, carbContent

- nutritionInfoIngredient(calories: int, sugar: int, proteinContent: int, fatContent: int, carbContent: int, ingredientName:varchar)

  ingredientName -> calories, sugar, proteinContent, fatContent, carbContent

- MealPlan(mealPlanID: int, name: varchar, date: date, userID: int) mealPlanID -> name, date, userID, recipeID

- ShoppingList(shoppingListID: int, mealPlanID: int)

  shoppingListID -> mealPlanID

- Rating(ratingID: int, score: int, userID: int, recipeID: int)

  ratingID -> score, userID, recipeID

  recipeID, userID -> ratingID, score

- Review(reviewID: int, date: date, message: varchar, userID: int, recipeID: int)

  reviewID -> date, message, userID, recipeID

  recipeID, userID -> reviewID, date, message

Relationships:

- saves(userID: int, recipeID: int)

  userID -> recipeID

  recipeID -> userID

- recipeContains(recipeID: int, ingredientName: varchar)

  recipeID -> ingredientName

- mealPlanContains(mealPlanID: int, recipeID:int)

  PK: recipeID + mealPlanID

mealPlanID -> recipeID

recipeID -> mealPlanID

- listContains(shoppingListID:int, ingredientName: varchar)

shoppingListID -> ingredientName

ingredientName -> shoppingListID

6. Normalization

a. Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after normalization. You should show the steps taken for the decomposition. Should there be errors, and no work is shown, no partial credit can be awarded without steps shown.

The format should be the same as Step 3, with tables listed similar to Table1(attr1:domain1, attr2:domain2, ...). ALL Tables must be listed, not only theones post normalization

User(userID: int, name: varchar, email: varchar,username: varchar)

PK: userID

CK: userID, email

FK: N/A

FDs:

userID -> name, email,username

username -> name

Closures

userID+ ={userID, name, email,username}

Username+ = {username, name}

Username is violates BCNF, not a key

Decompose into BCNF:

First decompose on :Username -> name

R1(username,name)

Two attribute relation, does not violate BCNF

R2(userID,name,username,email)

userID is super key, this is also in BCNF

Recipe(recipeID: int, name: varchar, category: varchar, instructions: varchar)

PK: recipeID

CK: recipeID

FK: N/A

FDs:

recipeID->recipe.name, instructions

recipe.name->category

Closures:

recipeID += {recipeID,recipe.name, category, instructions} recipe.name += {recipe.name, category}

recipeID is superkey

recipe.name is not in BCNF so decompose into R1 and R2

R1(recipe.name, category)

R2(recipeID, recipe.name, instructions)

R1 and R2 are now in BCNF.

Recipe(recipeID: int, name: varchar, instructions:varchar)

PK: recipeID

CK: recipeID

FK: N/A

Category(name:varchar, category: varchar)

PK: name

CK: name

FK: N/A

VegetarianRecipe(recipeID: int)

PK: recipeID

CK: recipeID

FK: recipeID

In 3NF/BCNF because it only contains recipeID.

LowCalorieRecipe(recipeID: int)

PK: recipeID

CK: recipeID

FK: recipeID

In 3NF/BCNF because it only contains recipeID.

SugarFreeRecipe(recipeID: int)

PK: recipeID

CK: recipeID

FK: recipeID

In 3NF/BCNF because it only contains recipeID.

Ingredient(name: varchar, isVegetarian: BOOLEAN)

PK: name

CK: name

FK: N/A

FDs:

ingredient.name - > isVegetarian

Closures:

ingredient.name += {ingredient.name, isVegetarian}

ingredient.name is superkey, this relationship is in BCNF

nutritionInfoRecipe(calories: int, sugar: int, proteinContent: int, fatContent: int, carbContent: int, recipeID: int)

PK: calories, sugar, proteinContent, fatContent, carbContent, recipeID

CK: calories, sugar, proteinContent, fatContent, carbContent, recipeID

FK: recipeID

FDs:

recipeID -> calories, sugar, proteinContent, fatContent, carbContent

Closures:

recipeID += {recipeID, calories, sugar, proteinContent, fatContent, carbContent}

RecipeID is superkey, relation is in BCNF

nutritionInfoIngredient(calories: int, sugar: int, proteinContent: int, fatContent: int, carbContent: int, ingredientName:varchar)

PK: calories, sugar, proteinContent, fatContent, carbContent,ingredientName

CK: calories, sugar, proteinContent, fatContent, carbContent, ,ingredientName

FK: ingredientName

FDs:

ingredientName -> calories, sugar, proteinContent, fatContent, carbContent

Closures:

ingredientName + = i{ngredientName,calories, sugar, proteinContent, fatContent, carbContent}

ingredientName is superkey, relation is in BCNF

MealPlan(mealPlanID: int, name: varchar, date: date, userID: int)

FDs:

mealPlanID -> name, date, userID, recipeID

No non-trivial, FDs and mealPlanID is superkey, relationship is in BCNF

shoppingList(shoppingListID: int, mealPlanID: int)

PK: shoppingListID

CK: shoppingListID

FK: mealPlanID

FDs:

shoppingListID -> mealPlanID

No non-trivial, FDs and shoppingListID is superkey, relationship is in BCNF

Rating(ratingID: int, score: int, userID: int, recipeID: int)

PK: ratingID

CK: ratingID, recipeID + userID

FK: recipeID, userID

FDs:

ratingID -> score, userID, recipeID

recipeID, userID -> ratingID, score

Closures:

ratingID+ = ratingID,score,userID,recipeID

recipeID,userID+= recipeID,userID,ratingID,score

(ratingID,userID) is a superkey, this relationship is in BCNF

Review(reviewID: int, date: date, message: varchar, userID: int, recipeID: int)

PK:reviewID

CK: reviewID, recipeID + userID

FK: recipeID, userID

FDs:

reviewID -> date, message, userID, recipeID

recipeID, userID -> reviewID, date, message

Closures:

reviewID+ = {reviewID,date,message,userID,recipeID}

recipeID,userID+ = {recipeID,userID,reviewID,date,message}

(recipeID,userID)+ is a superkey, relationship is in BCNF

saves(userID: int, recipeID: int)

PK: userID + recipeID

FDs:

userID -> recipeID

recipeID -> userID

No non-trivial FDs, relationship is in BCNF

recipeContains(recipeID: int, ingredientName: varchar)

FDs:

recipeID -> ingredientName

No non-trivial FDs, relationship is in BCNF

mealPlanContains(mealPlanID: int, recipeID:int)

PK: recipeID + mealPlanID

FDs:

mealPlanID -> recipeID

recipeID -> mealPlanID

No non-trivial FDs, relationship is in BCNF

listContains(shoppingListID:int, ingredientName: varchar)

PK: shoppingListID + ingredientName

FDs:

shoppingListID -> ingredientName

ingredientName -> shoppingListID

No non-trivial FDs, relationship is in BCNF

7. The SQL DDL statements required to create all the tables from item #6. The statements should use the appropriate foreign keys, primary keys, UNIQUE constraints, etc.

Unless you know that you will always have exactly x characters for a given character, it is better to use the VARCHAR data type as opposed to a CHAR(Y). For example, UBC courses always use four characters to represent which department offers a course. In that case, you will want to use CHAR(4) for the department attribute in your SQL DDL statement. If you are trying to represent the name of a UBC course, you will want to use VARCHAR as the number of characters in a course name can vary greatly.

```
CREATE TABLE User (
        userID INTEGER,
        name VARCHAR(50) NOT NULL,
        UNIQUE email VARCHAR(50) NOT NULL,
        UNIQUE username VARCHAR(50) NOT NULL,
        PRIMARY KEY (userID)
)

CREATE TABLE Recipe (
        recipeID INTEGER,
        name VARCHAR(50) NOT NULL,
        instructions VARCHAR(100000),
        PRIMARY KEY (recipeID)
)

CREATE TABLE Category (
        category varchar(50),
	name VARCHAR(50) NOT NULL,
	PRIMARY KEY (name)
)


CREATE TABLE VegetarianRecipe (
	recipeID INTEGER,
	PRIMARY KEY (recipeID),
	FOREIGN KEY (recipeID) REFERENCES Recipe(recipeID)
		ON UPDATE CASCADE
		ON DELETE CASCADE
)

CREATE TABLE LowCalorieRecipe (
	recipeID INTEGER,
	PRIMARY KEY (recipeID),
	FOREIGN KEY (recipeID) REFERENCES Recipe(recipeID)
		ON UPDATE CASCADE
		ON DELETE CASCADE
)

CREATE TABLE SugarFreeRecipe (
	recipeID INTEGER,

	PRIMARY KEY (recipeID),

	FOREIGN KEY (recipeID) REFERENCES Recipe(recipeID)

		ON UPDATE CASCADE

		ON DELETE CASCADE

)


CREATE TABLE Ingredient (

	name VARCHAR(50),

	isVegetarian BOOLEAN,

	PRIMARY KEY (name)

)


CREATE TABLE nutritionInfoRecipe (

	recipeID INTEGER,

	calories INTEGER,

	sugar INTEGER CHECK (sugar >=0),

	proteinContent INTEGER CHECK(proteinContent>=0),

	fatContent INTEGER CHECK(fatContent >=0),

	carbContent INTEGER CHECK (carbContent >=0),

	PRIMARY KEY (recipeID, calories, proteinContent, fatContent, carbContent),

	FOREIGN KEY (recipeID) REFERENCES Recipe(recipeID)
```

```
                ON UPDATE CASCADE

                ON DELETE CASCADE

)


CREATE TABLE nutritionInfoIngredient (

        ingredientName VARCHAR(50),

        calories INTEGER,

        sugar INTEGER CHECK (sugar >=0),

        proteinContent INTEGER CHECK(proteinContent>=0),

        fatContent INTEGER CHECK(fatContent >=0),

        carbContent INTEGER CHECK (carbContent >=0),


        PRIMARY KEY (ingredientName, calories, proteinContent, fatContent,
carbContent),

        FOREIGN KEY (ingredientName) REFERENCES Ingredient(ingredientName)

                ON UPDATE CASCADE

                ON DELETE CASCADE

)


CREATE TABLE MealPlan (

        mealPlanID INTEGER,

        name VARCHAR(50),

        date DATE,
    userID INTEGER,
    PRIMARY KEY (mealPlanID),
    FOREIGN KEY (userID) references User(userID)
)

CREATE TABLE ShoppingList (
    shoppingListID INTEGER,
    mealPlanID INTEGER,
    PRIMARY KEY (shoppingListID)
    FOREIGN KEY (mealPlanID) REFERENCES MealPlan (mealPlanID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
)

CREATE TABLE Rating (
    ratingID INTEGER,
    score INTEGER CHECK (0<=SCORE<=5),
    UNIQUE userID INTEGER,
    UNIQUE recipeID INTEGER,
    PRIMARY KEY (ratingID),
    FOREIGN KEY (userID) REFERENCES User
        ON DELETE CASCADE,
    FOREIGN KEY (recipeID) REFERENCES Recipe
        ON DELETE CASCADE
)

CREATE TABLE Review (
    reviewID INTEGER,
    date DATE,
    message VARCHAR(500),
    UNIQUE userID INTEGER NOT NULL,
    UNIQUE recipeID INTEGER,
    PRIMARY KEY (reviewID),
    FOREIGN KEY (userID) REFERENCES User(userID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
    FOREIGN KEY recipeID REFERENCES recipes(recipeID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
)

** Relationship Tables **

CREATE TABLE saves (
    userID INTEGER,

    recipeID INTEGER,

    PRIMARY KEY (userID, recipeID),

    FOREIGN KEY (userID) REFERENCES User (userID)

        ON UPDATE CASCADE

        ON DELETE CASCADE,

    FOREIGN KEY (recipeID) REFERENCES Recipe (recipeID)

        ON UPDATE CASCADE

        ON DELETE CASCADE

)


CREATE TABLE recipeContains (

    recipeID INTEGER,

    ingredientName VARCHAR(50),

    PRIMARY KEY (recipeID, ingredientName),

    FOREIGN KEY (recipeID) REFERENCES Recipe(recipeID)

        ON UPDATE CASCADE

        ON DELETE CASCADE,

    FOREIGN KEY (ingredientName) REFERENCES Ingredient (ingredientName)

        ON UPDATE CASCADE

        ON DELETE CASCADE

)
```

```
CREATE TABLE mealPlanContains (
        mealPlanID INTEGER,
        recipeID INTEGER,
        PRIMARY KEY (mealPlanID, recipeID),
        FOREIGN KEY (mealPlanID) REFERENCES MealPlan(mealPlanID)
                ON UPDATE CASCADE
                ON DELETE CASCADE,
        FOREIGN KEY (recipeID) REFERENCES RECIPE (recipeID)
                ON UPDATE CASCADE
                ON DELETE CASCADE
)

CREATE TABLE listContains (
        shoppingListID INTEGER,
        ingredientName VARCHAR(50),
        PRIMARY KEY (shoppingListID, ingredientName),
        FOREIGN KEY (shoppinglistID) REFERENCES ShoppingList (shoppinglistID)
                ON UPDATE CASCADE
                ON DELETE CASCADE,
        FOREIGN KEY (ingredientName) REFERENCES Ingredient(ingredientName)
                ON UPDATE CASCADE
                ON DELETE CASCADE

)
```

8. INSERT statements to populate each table with at least 5 tuples. You will likely want to have more than 5 tuples so that you can have meaningful queries later.

Note: Be consistent with the names used in your ER diagram, schema, and FDs. Make a note if the name has been intentionally changed.

```
INSERT INTO User (userID, name, email, username) VALUES

(1, ‘Joe’, ‘joe@email.com’, ‘joe’),

(2, ‘Jay’, ‘jay@email.com’, ‘jay’),

(3, ‘Kohen’, ‘kohen@email.com’, ‘kohen’),

(4, ‘George’, ‘george@email.com’, ‘george’),

(5, ‘Jeff’, ‘jeff@email.com’, ‘jeff’);


INSERT INTO Recipe (recipeID, name, instructions) VALUES

(1, ‘Pizza’, ‘recipe text… we didn’t want to actually make recipes for everything
but here is where the recipe instructions would go’),

(2, ‘Pasta’, ‘recipe text…’),

(3, ‘Burger’, ‘recipe text…’),

(4, ‘Fries’, ‘recipe text…’),

(5, ‘Grilled Cheese Sandwich’, ‘recipe text…’),

(6, ‘Salad’, ‘recipe text…’),

(7, ‘Vegetarian Pasta’, ‘recipe text…’),
(8, ‘Vegetarian Pizza’, ‘recipe text…’),
(9, ‘Vegetarian Burger’, ‘recipe text…’),
(10, ‘Quinoa’, ‘recipe text…’),
(11, ‘Low Calorie Pizza’, ‘recipe text…’),
(12, ‘Low Calorie Pasta’, ‘recipe text…’),
(13, ‘Sugar Free Pizza’, ‘recipe text…’),
(14, ‘Sugar Free Pasta’, ‘recipe text…’),
(15, ‘Sugar Free Burger’, ‘recipe text…’),
(16, Sugar Free ‘Fries’, ‘recipe text…’),
(17, ‘Sugar Free Sandwich’, ‘recipe text…’);

INSERT INTO Category (name, category) VALUES
(1, ‘Pizza’, ‘‘italian’’),
(2, ‘Pasta’, ‘italian’),
(3, ‘Burger’, ‘fast food’),
(4, ‘Fries’, ‘fast food’),
(5, ‘Grilled Cheese Sandwich’, ‘sandwich’),
(6, ‘Salad’, ‘salad’),
(7, ‘Vegetarian Pasta’,‘italian’),
(8, ‘Vegetarian Pizza’, ‘italian’),
(9, ‘Vegetarian Burger’, ‘recipe text…’),
(10, ‘Quinoa’, ‘ ’),
(11, ‘Low Calorie Pizza’, ‘italian’),
(12, ‘Low Calorie Pasta’, ‘italian’),
(13, ‘Sugar Free Pizza’, ‘italian’),
(14, ‘Sugar Free Pasta’, ‘italian’),
(15, ‘Sugar Free Burger’, ‘fast food’),
(16, Sugar Free ‘Fries’, ‘fast food’),
( ‘Sugar Free Sandwich’, ‘sandwich’);

INSERT INTO VegetarianRecipe (recipeID) VALUES
(4), (6), (7), (8), (9);

INSERT INTO LowCalorieRecipe (recipeID) VALUES
(5), (6), (10), (11), (12);

INSERT INTO SugarFreeRecipe (recipeID) VALUES
(13), (14), (15), (16), (17);

INSERT INTO Ingredient (name, isVegetarian) VALUES
(‘Rice’, TRUE),
(‘Noodles’, TRUE),
(‘Salt’, TRUE),
(‘Beef’, FALSE),
(‘Chicken’, FALSE);

INSERT INTO nutritionInfoRecipe (recipeID, calories, sugar, proteinContent,
fatContent, carbContent) VALUES
(1, 100, 10, 10, 10 ,10),
(2, 200, 20, 20, 20 ,20),
(3, 300, 30, 30, 30 ,30),
(4, 400, 40, 40, 40 ,40),
(5, 500, 50, 50, 50 ,50);

INSERT INTO nutritionInfoIngredient (ingredientName, calories, sugar,
proteinContent, fatContent, carbContent) VALUES
(‘Rice’, 100, 10, 10, 10, 10),
(‘Noodles’, 200, 20, 20, 20, 20),
(‘Salt’, 300, 30, 30, 30, 30),
(‘Beef’, 400, 40, 40, 40, 40),
(‘Chicken’, 500, 50, 50, 50, 50),

INSERT INTO MealPlan (mealPlanID, name, date, userID) VALUES
(1, ‘Normal Plan’, ‘2024-03-01’, 1),
(2, ‘Protein Plan’, ‘2024-03-02’, 2),
(3, ‘Vegetarian Plan’, ‘2024-03-03’, 3),
(4, ‘Vegan Plan’, ‘2024-03-04’, 4),
(5, ‘Low Sugar Plan’, ‘2024-03-05’, 5);

INSERT INTO ShoppingList (shoppingListID, mealPlanID) VALUES
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5);

INSERT INTO Rating (ratingID, score, userID, recipeID) VALUES
(1, 1, 1, 1),
(2, 2, 2, 2),
(3, 3, 3, 3),
(4, 4, 4, 4),
(5, 5, 5, 5);

INSERT INTO Review (reviewID, date, message, userID, recipeID) VALUES
(1, ‘2024-03-01’, ‘Good’, 1, 1),
(2, ‘2024-03-02’, ‘Bad’, 2, 2),
(3, ‘2024-03-03’, ‘Amazing’, 3, 3),
(4, ‘2024-03-04’, ‘Terrible’, 4, 4),
(5, ‘2024-03-05’, ‘Spectacular’, 5, 5);

** Relationship Tables **

INSERT INTO saves (userID, recipeID) VALUES
```

```
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5);


INSERT INTO recipeContains (recipeID, ingredientName) VALUES

(1, ‘Salt’), (1, ‘Chicken’), (2, ‘Noodles’), (2, ‘Beef’), (3, ‘Beef’);


INSERT INTO mealPlanContains (mealPlanID, recipeID) VALUES

(1, 1), (2, 3), (3, 6), (4, 10), (5, 11);


INSERT listContains (shoppingListID, ingredientName) VALUES

(1, ‘Salt’), (2, ‘Chicken’), (2, ‘Beef’), (3, ‘Rice’), (4, ‘Noodles’);
```