University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: 10.01.2024

Group Number: 107

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Keisuke Yamamoto	39088984	g1d4h	ykei2356@gmail.com
Yuto Kikuta	32572265	v4k9h	jordan2002222@gmail.com
Seokhee Hong	91166660	t9z3e	tjrgml1207@naver.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.)

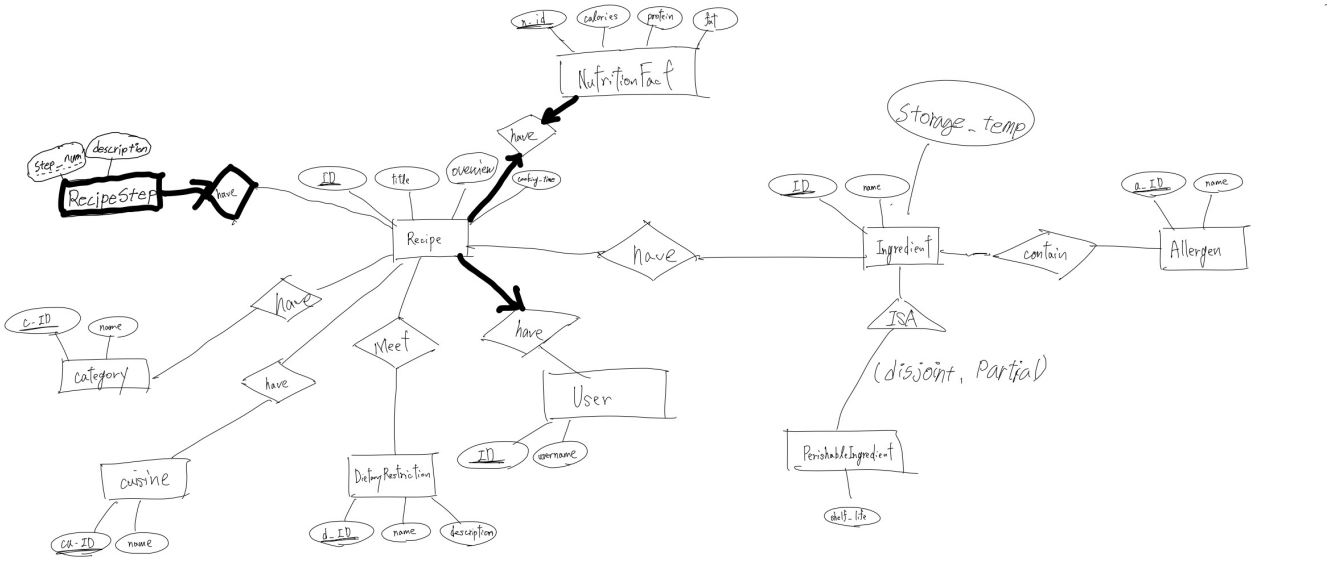
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Milestone2

Brief Summary of Project

The database will provide functionality to store, retrieve, and manage a wide variety of recipe-related data. Users will be able to look up histories for recipes that have been searched using AI and saved to the database, filter recipes by cuisine, category, or dietary restriction, and receive suggestions that meet their health and allergen requirements.

Some changes in ER diagram



• We removed the “Nonperishable” subclass, leaving only one subclass under ISA. Since the ‘storage_temp’ attribute is shared by both subclasses, we moved it to the superclass, making the “Nonperishable” subclass redundant without any unique attributes.

• We revised the relationship between “Recipe” and “User” from many-to-many to many-to-one with total participation on the “Recipe” side. This reflects that users can generate multiple recipes using AI, but each recipe is unique to the generation process.

• We eliminated the “description” attribute from several entities, including “Category,” “Cuisine,” “Ingredient,” and “Allergen,” as we determined this attribute is unnecessary for the app.

• We removed the “servings” and “image” attributes from the “Recipe” entity, and also deleted the “email” and “password_hash” attributes from the “User” entity, as we found these attributes were not useful for the app.

Schema

1. Recipe Table

Definition

Recipe (
ID INT,
title VARCHAR(255),
overview TEXT,
cooking_time INT
)

Constraints

- Primary Key: ID
- Not Null: ID, title, overview, cooking_time

2. RecipeStep Table (Weak Entity)

Definition

RecipeStep (
recipe_ID INT,
step_num INT,
description TEXT
)

Constraints

- Primary Key: (recipe_ID, step_num)
- Foreign Key: recipe_ID references Recipe(ID)
- Not Null: recipe_ID, step_num, description

3. NutritionFact Table (Updated with recipe ID as the foreign key)

Definition

NutritionFact (
ID INT,
calories INT,
protein INT,
fat INT
)

Constraints

- Primary Key: ID
- Foreign Key: ID references Recipe(ID)

- Not Null: ID, calories, protein, fat

4. Ingredient Table

Definition

Ingredient (
ID INT,
name VARCHAR(255),
storage_temp FLOAT
)

Constraints

- Primary Key: ID
- Not Null: ID, name, storage_temp

5. PerishableIngredient Table

Definition

PerishableIngredient (
ID INT,
shelf_life INT
)

Constraints

- Primary Key: ID
- Foreign Key: ID references Ingredient(ID)
- Not Null: ID, shelf_life

6. Allergen Table

Definition

Allergen (
ID INT,
name VARCHAR(255)
)

Constraints

- Primary Key: ID
- Not Null: ID, name

7. IngredientAllergen Table (Many-to-Many relationship between Ingredient and Allergen)

Definition

IngredientAllergen (
ingredient_ID INT,
allergen_ID INT
)

Constraints

- Primary Key: (ingredient_ID, allergen_ID)
- Foreign Keys:
- ingredient_ID references Ingredient(ID)
- allergen_ID references Allergen(ID)
- Not Null: ingredient_ID, allergen_ID

8. Category Table

Definition

Category (
ID INT,
name VARCHAR(255)
)

Constraints

- Primary Key: ID
- Not Null: ID, name

9. RecipeCategory Table

Definition

RecipeCategory (
recipe_ID INT,
category_ID INT
)

Constraints

- Primary Key: (recipe_ID, category_ID)
- Foreign Keys:
- recipe_ID references Recipe(ID)
- category_ID references Category(ID)
- Not Null: recipe_ID, category_ID

10. Cuisine Table

Definition

Cuisine (
ID INT,
name VARCHAR(255)
)

Constraints

- Primary Key: ID
- Not Null: ID, name

11. RecipeCuisine Table

Definition

RecipeCuisine (
recipe_ID INT,
cuisine_ID INT
)

Constraints

- Primary Key: (recipe_ID, cuisine_ID)
- Foreign Keys:
- recipe_ID references Recipe(ID)
- cuisine_ID references Cuisine(ID)
- Not Null: recipe_ID, cuisine_ID

12. DietaryRestriction Table

Definition

DietaryRestriction (
ID INT,
name VARCHAR(255),
description TEXT
)

Constraints

- Primary Key: ID
- Not Null: ID, name, description

13. RecipeDietaryRestriction Table

Definition

RecipeDietaryRestriction (
recipe_ID INT,

dietary_ID INT
)

Constraints

- Primary Key: (recipe_ID, dietary_ID)
- Foreign Keys:
- recipe_ID references Recipe(ID)
- dietary_ID references DietaryRestriction(ID)
- Not Null: recipe_ID, dietary_ID

14. User Table

Definition

User (
ID INT,
username VARCHAR(255)
)

Constraints

- Primary Key: ID
- Not Null: ID, username

15. UserRecipe Table (One-to-Many relationship between User and Recipe)

Definition

UserRecipe (
user_ID INT,
recipe_ID INT
)

Constraints

- Primary Key: (user_ID, recipe_ID)
- Foreign Keys:
- user_ID references User(ID)
- recipe_ID references Recipe(ID)
- Not Null: user_ID, recipe_ID

Functional Dependencies (FDs)

1. Recipe Table

Attributes: ID, title, overview, cooking_time

- Primary Key: ID
- Functional Dependencies:
- ID → title, overview, cooking_time (PK functional dependency)
- Note: No non-PK FDs in this case since we assume title doesn't determine any other attributes.

2. RecipeStep Table

Attributes: recipe_ID, step_num, description

- Primary Key: (recipe_ID, step_num)
- Functional Dependencies:
- (recipe_ID, step_num) → description (PK functional dependency)
- Note: No non-PK FDs in this case since the relationship is weak and dependent on both keys.

3. NutritionFact Table

Attributes: ID, calories, protein, fat

- Primary Key: ID
- Functional Dependencies:
- ID → calories, protein, fat (PK functional dependency)
- Note: No non-PK FDs, as each recipe ID uniquely determines its nutrition facts.

4. Ingredient Table

Attributes: ID, name, storage_temp

- Primary Key: ID
- Functional Dependencies:
- ID → name, storage_temp (PK functional dependency)
- name → storage_temp (Assuming that a specific ingredient name determines the required storage temperature)

5. PerishableIngredient Table

Attributes: ID, shelf_life, storage_temp

- Primary Key: ID
- Functional Dependencies:
- ID → shelf_life (PK functional dependency)
- Note: No non-PK FDs, as PerishableIngredient inherits from Ingredient and is uniquely determined by its ID.

6. Allergen Table

Attributes: ID, name

- Primary Key: ID
- Functional Dependencies:
- ID → name (PK functional dependency)
- Note: No non-PK FDs, as the allergen's ID determines its name.

7. IngredientAllergen Table

Attributes: ingredient_ID, allergen_ID

- Primary Key: (ingredient_ID, allergen_ID)
- Functional Dependencies:
- (ingredient_ID, allergen_ID) → No additional attributes, just the composite PK functional dependency.
- Note: No non-PK FDs, since this is an associative table.

8. Category Table

Attributes: ID, name

- Primary Key: ID
- Functional Dependencies:
- ID → name (PK functional dependency)
- Note: No non-PK FDs, as the category ID uniquely determines its name.

9. RecipeCategory Table

Attributes: recipe_ID, category_ID

- Primary Key: (recipe_ID, category_ID)
- Functional Dependencies:
- (recipe_ID, category_ID) → No additional attributes, just the composite PK functional dependency.
- Note: No non-PK FDs, since this is an associative table.

10. Cuisine Table

Attributes: ID, name

- Primary Key: ID
- Functional Dependencies:
- ID → name (PK functional dependency)
- Note: No non-PK FDs, as the cuisine ID uniquely determines its name.

11. RecipeCuisine Table

Attributes: recipe_ID, cuisine_ID

- Primary Key: (recipe_ID, cuisine_ID)
- Functional Dependencies:
- (recipe_ID, cuisine_ID) → No additional attributes, just the composite PK functional dependency.
- Note: No non-PK FDs, since this is an associative table.

12. DietaryRestriction Table

Attributes: ID, name, description

- Primary Key: ID
- Functional Dependencies:
- ID → name, description (PK functional dependency)
- name → description (Assuming that the name of the dietary restriction uniquely determines the description)

13. RecipeDietaryRestriction Table

Attributes: recipe_ID, dietary_ID

- Primary Key: (recipe_ID, dietary_ID)
- Functional Dependencies:
- (recipe_ID, dietary_ID) → No additional attributes, just the composite PK functional dependency.
- Note: No non-PK FDs, since this is an associative table.

14. User Table

Attributes: ID, username

- Primary Key: ID
- Functional Dependencies:
- ID → username (PK functional dependency)
- Note: No non-PK FDs, as the user's ID determines their username.

15. UserRecipe Table

Attributes: user_ID, recipe_ID

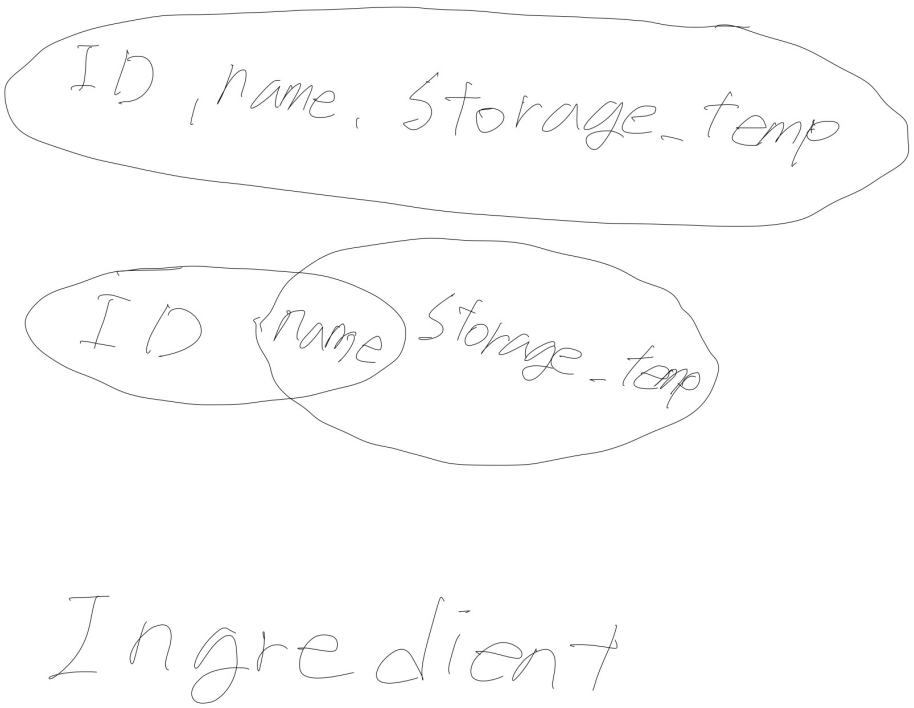
- Primary Key: (user_ID, recipe_ID)
- Functional Dependencies:
- (user_ID, recipe_ID) → No additional attributes, just the composite PK functional

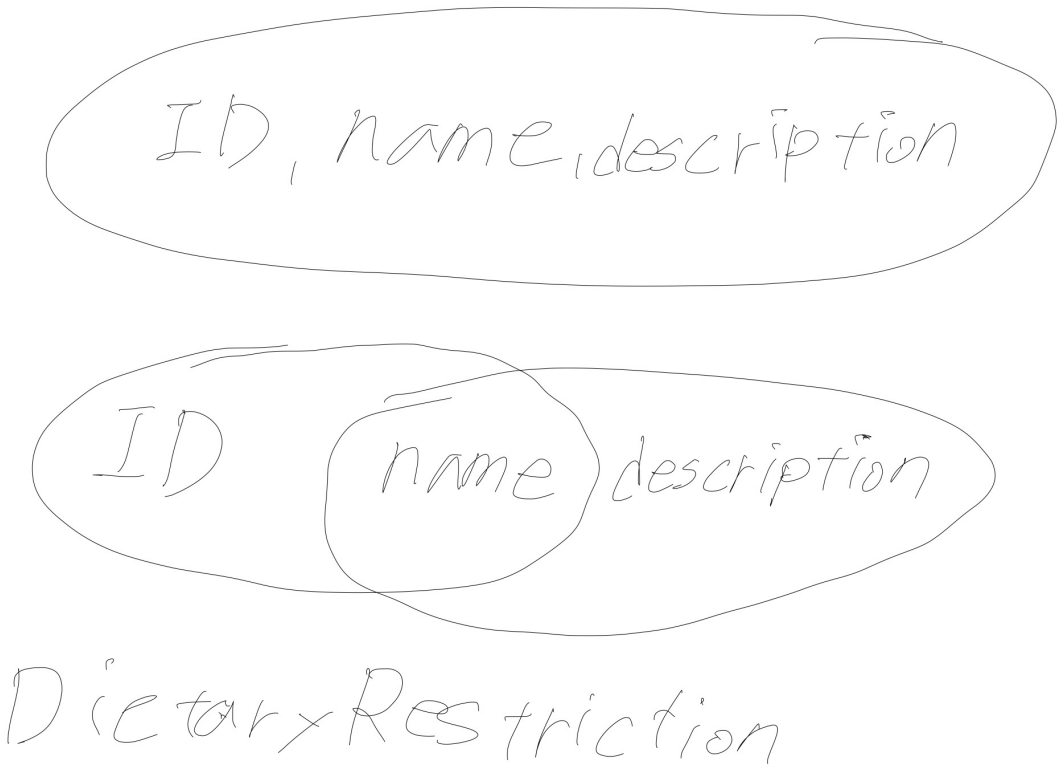
dependency.

- Note: No non-PK FDs, since this is an associative table.

Normalization

We broke the two table to normalize "ingredient" and "dietary restriction" entity. Here are the corrected tables.





1. Recipe Table

Definition

Recipe (
ID INT,
title VARCHAR(255),
overview TEXT,
cooking_time INT
)

Constraints

- Primary Key: ID

2. RecipeStep Table (Weak Entity)

Definition

RecipeStep (
recipe_ID INT,
step_num INT,
description TEXT
)

Constraints

- Primary Key: (recipe_ID, step_num)
- Foreign Key: recipe_ID references Recipe(ID)

3. NutritionFact Table

Definition

NutritionFact (
ID INT,
calories INT,
protein INT,
fat INT
)

Constraints

- Primary Key: ID
- Foreign Key: ID references Recipe(ID)

4. Ingredient Table (After BCNF Decomposition)

Ingredient Table

Ingredient (
ID INT,
name VARCHAR(255)
)

- Primary Key: ID

5. StorageTempByIngredient Table

StorageTempByIngredient (
name VARCHAR(255),
storage_temp FLOAT
)

- Primary Key: name

6. PerishableIngredient Table

Definition

PerishableIngredient (
ID INT,
shelf_life INT
)

Constraints

- Primary Key: ID
- Foreign Key: ID references Ingredient(ID)

7. Allergen Table

Definition

Allergen (
ID INT,
name VARCHAR(255)
)

Constraints

- Primary Key: ID

8. IngredientAllergen Table

Definition

IngredientAllergen (
ingredient_ID INT,
allergen_ID INT
)

Constraints

- Primary Key: (ingredient_ID, allergen_ID)
- Foreign Keys:
- ingredient_ID references Ingredient(ID)
- allergen_ID references Allergen(ID)

9. Category Table

Definition

Category (
ID INT,
name VARCHAR(255)
)

Constraints

- Primary Key: ID

10. RecipeCategory Table

Definition

RecipeCategory (
recipe_ID INT,
category_ID INT
)

Constraints

- Primary Key: (recipe_ID, category_ID)
- Foreign Keys:
- recipe_ID references Recipe(ID)
- category_ID references Category(ID)

11. Cuisine Table

Definition

Cuisine (
ID INT,
name VARCHAR(255)
)

Constraints

- Primary Key: ID

12. RecipeCuisine Table

Definition

RecipeCuisine (
recipe_ID INT,
cuisine_ID INT
)

Constraints

- Primary Key: (recipe_ID, cuisine_ID)
- Foreign Keys:
- recipe_ID references Recipe(ID)
- cuisine_ID references Cuisine(ID)

13. DietaryRestriction Table (After BCNF Decomposition)

DietaryRestriction Table

DietaryRestriction (
ID INT,

name VARCHAR(255)
)

- Primary Key: ID

14. DescriptionByDietaryRestriction Table

DescriptionByDietaryRestriction (
name VARCHAR(255),
description TEXT
)

- Primary Key: name

15. RecipeDietaryRestriction Table

Definition

RecipeDietaryRestriction (
recipe_ID INT,
dietary_ID INT
)

Constraints

- Primary Key: (recipe_ID, dietary_ID)
- Foreign Keys:
- recipe_ID references Recipe(ID)
- dietary_ID references DietaryRestriction(ID)

16. User Table

Definition

User (
ID INT,
username VARCHAR(255)
)

Constraints

- Primary Key: ID

17. UserRecipe Table

Definition

UserRecipe (
user_ID INT,
recipe_ID INT
)

Constraints

- Primary Key: (user_ID, recipe_ID)
- Foreign Keys:
- user_ID references User(ID)
- recipe_ID references Recipe(ID)

SQL DDL Statements

1. Recipe Table

```
CREATE TABLE Recipe (
ID INT PRIMARY KEY,
title VARCHAR(255) NOT NULL,
overview TEXT NOT NULL,
cooking_time INT NOT NULL
);
```

2. RecipeStep Table (Weak Entity)

```
CREATE TABLE RecipeStep (
recipe_ID INT NOT NULL,
step_num INT NOT NULL,
description TEXT NOT NULL,
PRIMARY KEY (recipe_ID, step_num),
FOREIGN KEY (recipe_ID) REFERENCES Recipe(ID)
);
```

3. NutritionFact Table

```
CREATE TABLE NutritionFact (
ID INT NOT NULL PRIMARY KEY,
calories INT NOT NULL,
protein INT NOT NULL,
fat INT NOT NULL,
FOREIGN KEY (ID) REFERENCES Recipe(ID)
);
```

4. Ingredient Table

```
CREATE TABLE Ingredient (
ID INT NOT NULL PRIMARY KEY,
name VARCHAR(255) NOT NULL UNIQUE
);
```

5. StorageTempByIngredient Table

```
CREATE TABLE StorageTempByIngredient (
name VARCHAR(255) NOT NULL PRIMARY KEY,
storage_temp FLOAT NOT NULL,
FOREIGN KEY (name) REFERENCES Ingredient(name)
);
```

6. PerishableIngredient Table

```
CREATE TABLE PerishableIngredient (
ID INT PRIMARY KEY,
shelf_life INT NOT NULL,
FOREIGN KEY (ID) REFERENCES Ingredient(ID)
);
```

7. Allergen Table

```
CREATE TABLE Allergen (
ID INT PRIMARY KEY,
name VARCHAR(255) NOT NULL UNIQUE
);
```

8. IngredientAllergen Table

```
CREATE TABLE IngredientAllergen (
ingredient_ID INT NOT NULL,
allergen_ID INT NOT NULL,
PRIMARY KEY (ingredient_ID, allergen_ID),
FOREIGN KEY (ingredient_ID) REFERENCES Ingredient(ID),
FOREIGN KEY (allergen_ID) REFERENCES Allergen(ID)
);
```

9. Category Table

```
CREATE TABLE Category (
ID INT NOT NULL PRIMARY KEY,
name VARCHAR(255) NOT NULL UNIQUE
);
```

10. RecipeCategory Table

```
CREATE TABLE RecipeCategory (
recipe_ID INT NOT NULL,
category_ID INT NOT NULL,
PRIMARY KEY (recipe_ID, category_ID),
FOREIGN KEY (recipe_ID) REFERENCES Recipe(ID),
FOREIGN KEY (category_ID) REFERENCES Category(ID)
);
```

11. Cuisine Table

```
CREATE TABLE Cuisine (
ID INT NOT NULL PRIMARY KEY,
name VARCHAR(255) NOT NULL UNIQUE
);
```

12. RecipeCuisine Table

```
CREATE TABLE RecipeCuisine (
recipe_ID INT NOT NULL,
cuisine_ID INT NOT NULL,
PRIMARY KEY (recipe_ID, cuisine_ID),
FOREIGN KEY (recipe_ID) REFERENCES Recipe(ID),
FOREIGN KEY (cuisine_ID) REFERENCES Cuisine(ID)
);
```

13. DietaryRestriction Table

```
CREATE TABLE DietaryRestriction (
ID INT NOT NULL PRIMARY KEY,
name VARCHAR(255) NOT NULL UNIQUE
);
```

14. DescriptionByDietaryRestriction Table

```
CREATE TABLE DescriptionByDietaryRestriction (
name VARCHAR(255) NOT NULL PRIMARY KEY,
description TEXT NOT NULL,
FOREIGN KEY (name) REFERENCES DietaryRestriction(name)
);
```

15. RecipeDietaryRestriction Table

```
CREATE TABLE RecipeDietaryRestriction (
recipe_ID INT NOT NULL,
dietary_ID INT NOT NULL,
PRIMARY KEY (recipe_ID, dietary_ID),
FOREIGN KEY (recipe_ID) REFERENCES Recipe(ID),
FOREIGN KEY (dietary_ID) REFERENCES DietaryRestriction(ID)
);
```

16. User Table

```
CREATE TABLE User (
ID INT NOT NULL PRIMARY KEY,
username VARCHAR(255) NOT NULL UNIQUE
);
```

17. UserRecipe Table

```
CREATE TABLE UserRecipe (
user_ID INT NOT NULL,
```

```
recipe_ID INT NOT NULL,
PRIMARY KEY (user_ID, recipe_ID),
FOREIGN KEY (user_ID) REFERENCES User(ID),
FOREIGN KEY (recipe_ID) REFERENCES Recipe(ID)
);
```

INSERT Statements

1. Recipe Table

```
INSERT INTO Recipe (ID, title, overview, cooking_time) VALUES
(1, 'Spaghetti Bolognese', 'A classic Italian pasta dish with meat sauce.', 45),
(2, 'Chicken Curry', 'A spicy and flavorful Indian curry.', 60),
(3, 'Vegetable Stir Fry', 'A quick and healthy stir-fried vegetables.', 20),
(4, 'Beef Tacos', 'Mexican tacos with seasoned beef.', 30),
(5, 'Grilled Salmon', 'Simple and delicious grilled salmon fillets.', 25);
```

2. RecipeStep Table

```
INSERT INTO RecipeStep (recipe_ID, step_num, description) VALUES
(1, 1, 'Cook spaghetti according to package instructions.'),
(1, 2, 'Prepare Bolognese sauce by sautéing ground beef with tomato sauce.'),
(2, 1, 'Cook chicken pieces until browned.'),
(2, 2, 'Add curry powder and simmer with spices.'),
(3, 1, 'Chop vegetables into bite-sized pieces.'),
(3, 2, 'Stir-fry vegetables in a wok with soy sauce.'),
(4, 1, 'Cook seasoned beef in a skillet.'),
(4, 2, 'Assemble tacos with beef and toppings.'),
(5, 1, 'Season salmon fillets with salt and pepper.'),
(5, 2, 'Grill salmon until cooked through.');
```

3. NutritionFact Table

```
INSERT INTO NutritionFact (ID, calories, protein, fat) VALUES
(1, 600, 25, 20),
(2, 550, 30, 15),
(3, 200, 5, 10),
(4, 450, 20, 25),
(5, 350, 30, 15);
```

4. Ingredient Table

```
INSERT INTO Ingredient (ID, name) VALUES
(1, 'Spaghetti'),
(2, 'Ground Beef'),
(3, 'Tomato Sauce'),
(4, 'Onion'),
(5, 'Garlic'),
(6, 'Chicken'),
(7, 'Curry Powder'),
```

(8, 'Bell Pepper'),
(9, 'Soy Sauce'),
(10, 'Salmon'),
(11, 'Peanuts'),
(12, 'Shrimp');

5. StorageTempByIngredient Table

INSERT INTO StorageTempByIngredient (name, storage_temp) VALUES
('Spaghetti', 25),
('Ground Beef', 4),
('Tomato Sauce', 4),
('Onion', 25),
('Garlic', 25),
('Chicken', 4),
('Curry Powder', 25),
('Bell Pepper', 4),
('Soy Sauce', 25),
('Salmon', 4),
('Peanuts', 25),
('Shrimp', 4);

6. PerishableIngredient Table

INSERT INTO PerishableIngredient (ID, shelf_life, storage_temp) VALUES
(2, 2, 4), -- Ground Beef
(3, 5, 4), -- Tomato Sauce
(6, 2, 4), -- Chicken
(8, 7, 4), -- Bell Pepper
(10, 3, 4), -- Salmon
(12, 2, 4); -- Shrimp

7. Allergen Table

INSERT INTO Allergen (ID, name) VALUES
(1, 'Gluten'),
(2, 'Soy'),
(3, 'Fish'),
(4, 'Shellfish'),
(5, 'Milk'),
(6, 'Peanuts'),
(7, 'Tree Nuts');

8. IngredientAllergen Table

INSERT INTO IngredientAllergen (ingredient_ID, allergen_ID) VALUES
(1, 1), -- Spaghetti contains Gluten
(9, 2), -- Soy Sauce contains Soy
(10, 3), -- Salmon contains Fish
(11, 6), -- Peanuts contain Peanuts allergen
(12, 4); -- Shrimp contains Shellfish

9. Category Table

```
INSERT INTO Category (ID, name) VALUES
(1, 'Main Course'),
(2, 'Appetizer'),
(3, 'Dessert'),
(4, 'Salad'),
(5, 'Soup');
```

10. RecipeCategory Table

```
INSERT INTO RecipeCategory (recipe_ID, category_ID) VALUES
(1, 1), -- Spaghetti Bolognese is a Main Course
(2, 1), -- Chicken Curry is a Main Course
(3, 1), -- Vegetable Stir Fry is a Main Course
(4, 1), -- Beef Tacos is a Main Course
(5, 1); -- Grilled Salmon is a Main Course
```

11. Cuisine Table

```
INSERT INTO Cuisine (ID, name) VALUES
(1, 'Italian'),
(2, 'Indian'),
(3, 'Chinese'),
(4, 'Mexican'),
(5, 'Japanese');
```

12. RecipeCuisine Table

```
INSERT INTO RecipeCuisine (recipe_ID, cuisine_ID) VALUES
(1, 1), -- Spaghetti Bolognese is Italian cuisine
(2, 2), -- Chicken Curry is Indian cuisine
(3, 3), -- Vegetable Stir Fry is Chinese cuisine
(4, 4), -- Beef Tacos are Mexican cuisine
(5, 5); -- Grilled Salmon is Japanese cuisine
```

13. DietaryRestriction Table

```
INSERT INTO DietaryRestriction (ID, name) VALUES
(1, 'Vegetarian'),
(2, 'Vegan'),
(3, 'Gluten-Free'),
(4, 'Halal'),
(5, 'Kosher');
```

14. DescriptionByDietaryRestriction Table

```
INSERT INTO DescriptionByDietaryRestriction (name, description) VALUES
('Vegetarian', 'Excludes meat and fish products'),
('Vegan', 'Excludes all animal products'),
('Gluten-Free', 'Excludes gluten-containing grains'),
```

('Halal', 'Permissible under Islamic law'),
('Kosher', 'Prepared according to Jewish dietary laws');

15. RecipeDietaryRestriction Table

INSERT INTO RecipeDietaryRestriction (recipe_ID, dietary_ID) VALUES
(3, 1), -- Vegetable Stir Fry is Vegetarian
(3, 2), -- Vegetable Stir Fry is Vegan
(5, 3); -- Grilled Salmon is Gluten-Free
(4, 4); -- and so on.
(2, 5);

16. User Table

INSERT INTO User (ID, username) VALUES
(1, 'alice'),
(2, 'bob'),
(3, 'charlie'),
(4, 'diana'),
(5, 'eve');

17. UserRecipe Table

INSERT INTO UserRecipe (user_ID, recipe_ID) VALUES
(1, 1), -- Alice saved Spaghetti Bolognese
(1, 3), -- Alice saved Vegetable Stir Fry
(2, 2), -- Bob saved Chicken Curry
(2, 3), -- Bob saved Vegetable Stir Fry
(3, 4), -- Charlie saved Beef Tacos
(4, 5), -- Diana saved Grilled Salmon
(5, 3); -- Eve saved Vegetable Stir Fry