University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: 19 Jul 2024

Group Number: 20

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Abdulrhman Mohamed	36901149	w3x9a	abdulubc@yahoo.ca
Atta Faiz	57572885	z8o6j	afschool21@gmail.com
Vi Do	79982799	v1t4y	haviid.06@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

University of British Columbia, Vancouver
Department of Computer Science

1. A completed cover page (template on canvas).
2. A brief (~2-3 sentences) summary of your project. Many of your TAs are managing multiple projects so this will help them remember details about your project.

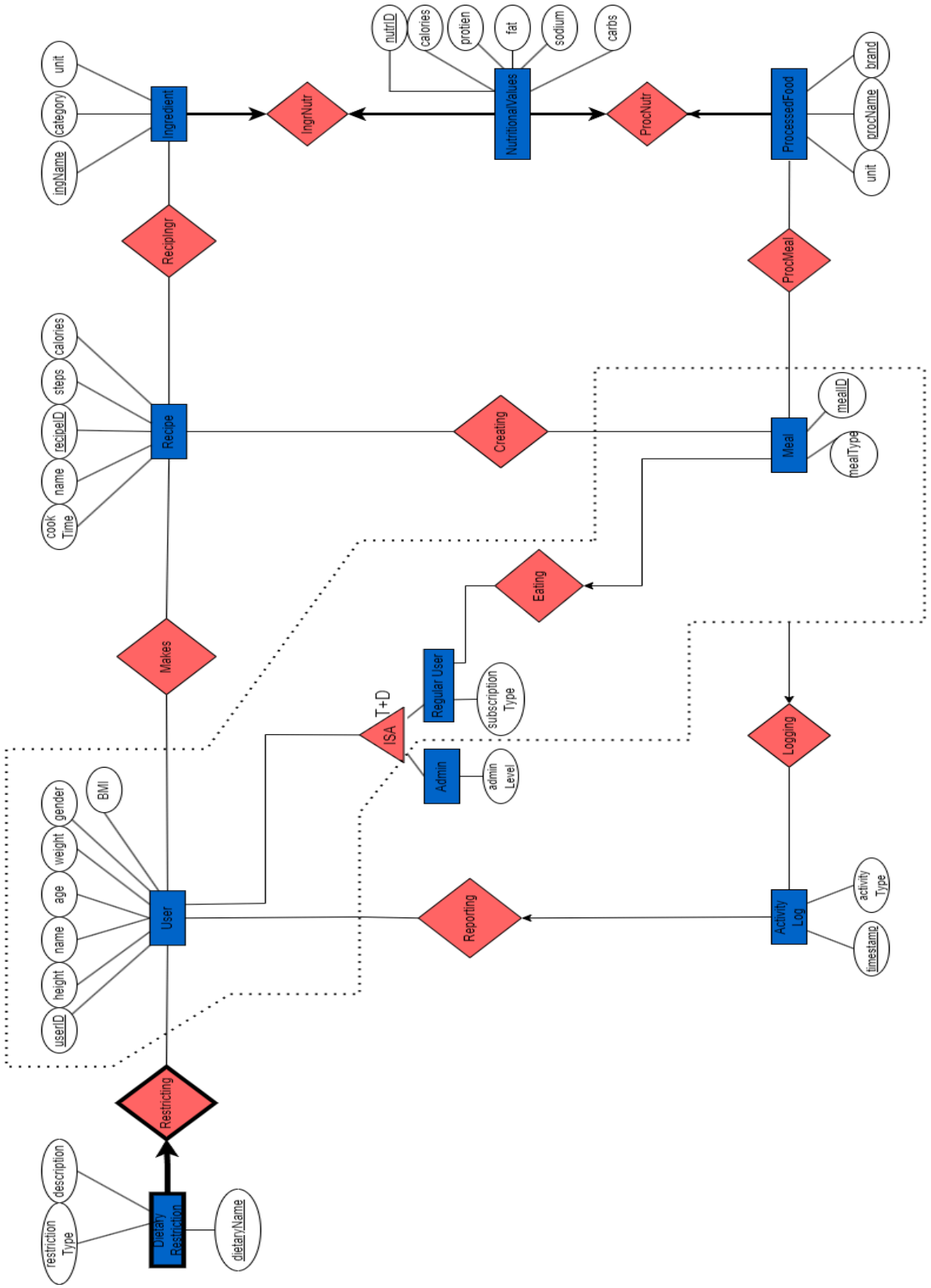
   Our project is a health and wellness application that helps users manage their nutrition and diet. Users can create personal profiles, set health goals through completing activities, browse and save recipes, track their daily food intake, and monitor their progress. The app is designed to be easy to use and supports a variety of dietary needs and preferences, helping users achieve their health and dietary goals.

3. The ER diagram you are basing your item #3 (below) on. This ER diagram may be the same as your milestone 1 submission or it might be different. If you have made changes from the version submitted in milestone 1, attach a note indicating what changes have been made and why. If you have decided not to implement the suggestions given by your project mentor, please be sure to leave a note stating why. This is not to say that you must do everything that your project mentor says. In many instances, there are trade-offs between design choices and your decision may be influenced by different factors. Your TAs will often leave suggestions that are meant to help massage your project into a form that will fit with the requirements in future project milestones. If you choose not to take their advice, it would be helpful for them to know why to better assist the group moving forward.

   We made several updates to the schema based on the TA's feedback. First, we added MealID as the primary key for the Meal entity, which better identifies each meal uniquely. We also removed the mealName attribute, as it was unnecessary considering the new unique key. We also modified the NutritionalValues entity to have total participation in all its relationships with processed food as it was mentioned by the TA the initial relationship was not accurate. Additionally, we incorporated the total and disjoint characteristics for the ISA relationship to accurately reflect the hierarchical structure. Additionally, the admin subclass was removed from the aggregate relationship, as it is not relevant to the meal logging functionality of our program. Finally, we added BMI for users to help assist in meeting the necessary functional dependency requirements.

   The new ER diagram with the recommended TA recommendations is on the next page.

University of British Columbia, Vancouver
Department of Computer Science



4. 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) NOT NULL, foreign keys (FK), and other constraints (e.g., not null, unique, etc.) that the table must maintain.

Ingredient (ingName VARCHAR(99), **nutrID** CHAR(8) NOT NULL + UNIQUE, ingUnit VARCHAR(99) NOT NULL, category VARCHAR(99))

RecipIngr(**recipeID** CHAR(8), **ingName** VARCHAR(99))

Recipe(recipeID CHAR(8), cookTime INTEGER NOT NULL, recipeName VARCHAR(99) NOT NULL, steps VARCHAR(999) CK NOT NULL, recipeCalories DECIMAL(7, 2) NOT NULL)

ProcessedFood(procName VARCHAR(50), brand VARCHAR(50), **nutrID** CHAR(8) NOT NULL + UNIQUE, pfUnit VARCHAR(20) NOT NULL)

NutrionalValue(nutrID CHAR(8), **ingName** VARCHAR(99) UNIQUE, **procName** CHAR(50) UNIQUE, **brand** VARCHAR(50) UNIQUE, nutrCalories DECIMAL(7, 2) NOT NULL, protein DECIMAL(7, 2), fat DECIMAL(7, 2), sodium DECIMAL(7, 2), carbs DECIMAL(7, 2))

ActivityLogReporting(timestamp DATETIME, **userID** CHAR(8), activityType VARCHAR(50))

ProcMeal (**brand** VARCHAR(50), **procName** VARCHAR(50), **mealID** CHAR(8))

User (userID CHAR(8), height DECIMAL(3, 2), weight DECIMAL(5, 2), username CHAR(99) NOT NULL, age INTEGER, BMI DECIMAL(4, 2), gender VARCHAR(50))

UserDietaryRestriction (dietaryName VARCHAR(50), **userID** CHAR(8), restrictionType VARCHAR(50), description VARCHAR(999))

Makes (**userID** CHAR(8), **recipeID** CHAR(8))

Admin (**userID** CHAR(8), adminLevel VARCHAR(50) NOT NULL)

RegularUser (**userID** CHAR(8), subscriptionType VARCHAR(50) NOT NULL)

Creating (**recipeID** CHAR(8), **mealID** CHAR(8))

MealEating(mealID CHAR(8), **userID** CHAR(8), mealType VARCHAR(50) NOT NULL)

UserMealLogging(userID CHAR(8), mealID CHAR(8), **timestamp** DATETIME)

University of British Columbia, Vancouver
Department of Computer Science

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.

Ingredient FD:
ingName -> **nutrID,** ingUnit, category

Recipe FD:
recipeID -> cookTime, recipeName, steps, recipeCalories

ProcessedFood FD:
procName , brand -> **nutrID,** pfUnit

NutrionalValue FD:
nutrID -> **ingName**, **procName**, **brand**, protein, fat, sodium, carbs, nutrCalories
protein, fat, sodium, carbs -> nutrCalories

ActivityLogReporting FD:
timestamp, **userID** -> activityType

User FD:
userID, height, weight, username, age, BMI, gender
height, weight -> BMI

UserDietaryRestriction FD:
dietaryName, **userID**, restrictionType, description

Admin FD:
userID -> adminLevel

RegularUser FD:
userID -> subscriptionType

MealEating FD:
mealID -> **userID**, mealType

University of British Columbia, Vancouver
Department of Computer Science

UserMealLogging FD:

userID, mealID -> **timestamp**

University of British Columbia, Vancouver
Department of Computer Science

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 the ones post normalization.

Ingredient

FD:

ingName -> **nutrID,** ingUnit, category => already in BCNF.

Relation Final:

Ingredient (ingName VARCHAR(99), **nutrID** CHAR(8), ingUnit VARCHAR(99), category VARHAR(99))

Recipe

FD:

recipeID -> cookTime, recipeName, steps, recipeCalories => already in BCNF

Relation Final:

Recipe(recipeID CHAR(8), cookTime INTEGER, recipeName VARCHAR(99), steps VARCHAR(999), recipeCalories DECIMAL(7, 2))

ProcessedFood

FD:

procName , brand -> **nutrID,** pfUnit => already in BCNF

Relation Final:

ProcessedFood(procName VARCHAR(50), brand VARCHAR(50), **nutrID** CHAR(8), pfUnit VARCHAR(20))

NutrionalValue

FD:

nutrID -> **ingName**, **procName**, **brand**, protein, fat, sodium, carbs, nutrCalories

=> already in BCNF

protein, fat, sodium, carbs -> nutrCalories => Not in BCNF

Decompose on the second FD:

NutritionalValue1 (protein, fat, sodium, carbs, nutrCalories) => already in BCNF

NutritionalValue2 (nutrID, **ingName**, **procName**, **brand**, protein, fat, sodium, carbs) => already in BCNF

Relation Final:

NutrionalValue1(protein DECIMAL(7, 2), fat DECIMAL(7, 2), sodium DECIMAL(7, 2), carbs DECIMAL(7, 2), nutrCalories DECIMAL(7, 2))

NutrionalValue2(nutrID CHAR(8), **ingName** VARCHAR(99), **procName** VARCHAR(50), **brand** VARCHAR(50), protein DECIMAL(7, 2), fat DECIMAL(7, 2), sodium DECIMAL(7, 2), carbs DECIMAL(7, 2))

ActivityLogReporting

FD:

timestamp, **userID** -> activityType => already in BCNF

Relation Final:

ActivityLogReporting(timestamp DATETIME, **userID** CHAR(8), activityType VARCHAR(50))

User

FD:

userID -> height, weight, username, age, BMI, gender => already in BCNF
height, weight -> BMI => not in BCNF

Decompose on the second fd:
User1 (height, weight, BMI) => in BCNF
User2 (userID, height, weight, username, age, gender) => in BCNF

Relation Final:
User1 (height DECIMAL(3, 2), weight DECIMAL(5, 2), BMI DECIMAL(4, 2))
User2 (userID CHAR(8), height DECIMAL(3, 2), weight DECIMAL(5, 2), username CHAR(99), age INTEGER, gender VARCHAR(50))

UserDietaryRestriction

FD:

dietaryName, **userID**, restrictionType, description => already in BCNF

Relation Final:
UserDietaryRestriction (dietaryName VARCHAR(99), **userID** CHAR(8), restrictionType VARCHAR(50), description VARCHAR(999))

University of British Columbia, Vancouver
Department of Computer Science

Admin

FD:

userID -> adminLevel => already in BCNF

Relation Final:

Admin (**userID** CHAR(8), adminLevel VARCHAR(50))

RegularUser

FD:

userID -> subscriptionType => already in BCNF

Relation Final:

RegularUser (**userID** CHAR(8), subscriptionType VARCHAR(50))

MealEating

FD:

mealID -> **userID**, mealType => already in BCNF

Relation Final:

MealEating(mealID CHAR(99), **userID** CHAR(8), mealType VARCHAR(50))

UserMealLogging

FD:

userID, mealID -> **timestamp** => already in BCNF

Relation Final:

UserMealLogging(userID CHAR(8), mealID CHAR(8), **timestamp** DATETIME)

University of British Columbia, Vancouver
Department of Computer Science

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.
8.

```
CREATE TABLE Ingredient (
        ingName VARCHAR(99),
        nutrID CHAR(8) NOT NULL UNIQUE,
        ingUnit VARCHAR(99) NOT NULL,
        category VARCHAR(99),
        PRIMARY KEY (ingName)
        FOREIGN KEY (nutrID)
                REFERENCES NutritionalValue2 (nutrID)
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
```

```
CREATE TABLE RecipIngr (
        recipeID CHAR(8),
        ingName VARCHAR(99),
        PRIMARY KEY (recipeID, ingName),
        FOREIGN KEY (recipeID)
                REFERENCES Recipe(recipeID),
                ON UPDATE CASCADE
                ON DELETE CASCADE
        FOREIGN KEY (ingName)
                REFERENCES Ingredient(ingName)
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
```

```
CREATE TABLE ProcessedFood (
        procName VARCHAR(50),
        brand VARCHAR(50),
        nutrID CHAR(8) NOT NULL UNIQUE,
        pfUnit VARCHAR(20) NOT NULL,
        PRIMARY KEY (procName, brand),
        FOREIGN KEY (nutrID)
                REFERENCES NutritionalValue2(nutrID)
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
```

```
CREATE TABLE Recipe (
        recipeID CHAR(8),
        cookTime INTEGER NOT NULL,
        recipeName VARCHAR(99) NOT NULL,
        steps VARCHAR(999) CK NOT NULL,
        recipeCalories DECIMAL(7,2) NOT NULL,
        PRIMARY KEY (recipeID)
);
```

```
CREATE TABLE ProcMeal (
        brand VARCHAR(50),
        procName VARCHAR(50),
        mealID CHAR(8),
        PRIMARY KEY (brand, procName, mealID),
        FOREIGN KEY (brand, procName)
                REFERENCES ProcessedFood(brand, procName)
                ON UPDATE CASCADE
                ON DELETE CASCADE
        FOREIGN KEY (mealID)
                REFERENCES Meal (mealID)
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
```

```
CREATE TABLE ActivityLogReporting (
        timestamp DATETIME,
        userID CHAR(8),
        activityType VARCHAR(50),
        PRIMARY KEY (timestamp),
        FOREIGN KEY (userID)
                REFERENCES User2(userID)
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
```

University of British Columbia, Vancouver
Department of Computer Science

```
CREATE TABLE Admin (
        userID CHAR(8),
        adminLevel VARCHAR(50) NOT NULL,
        PRIMARY KEY (userID),
        FOREIGN KEY (userID)
                REFERENCES User2(userID)
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
```

```
CREATE TABLE RegularUser (
        userID CHAR(8),
        subscriptionType VARCHAR(50) NOT NULL,
        PRIMARY KEY (userID),
        FOREIGN KEY (userID)
                REFERENCES User2(userID)
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
```

```
CREATE TABLE UserDietaryRestriction (
        dietaryName VARCHAR(50),
        userID CHAR(8),
        restrictionType VARCHAR(50),
        description VARCHAR(999),
        PRIMARY KEY (dietaryName, userID),
        FOREIGN KEY (userID)
                REFERENCES User2(userID)
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
```

```
CREATE TABLE Creating (
        recipeID CHAR(8),
        mealID CHAR(8),
        PRIMARY KEY (recipeID, mealID),
        FOREIGN KEY (recipeID)
                REFERENCES Recipe(recipeID),
                ON UPDATE CASCADE
                ON DELETE CASCADE
        FOREIGN KEY (mealID)
                REFERENCES Meal(mealID)
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
```

```
CREATE TABLE Makes (
        userID CHAR(8),
        recipeID CHAR(8),
        PRIMARY KEY (userID, recipeID),
        FOREIGN KEY (userID)
                REFERENCES User2(userID),
                ON UPDATE CASCADE
                ON DELETE CASCADE
        FOREIGN KEY (recipeID)
                REFERENCES Recipe(recipeID)
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
```

```
CREATE TABLE MealEating (
        mealID CHAR(8),
        userID CHAR(8),
        mealType VARCHAR(50) NOT NULL,
        PRIMARY KEY (mealID, userID),
        FOREIGN KEY (userID)
                REFERENCES User2(userID)
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
```

```
CREATE TABLE UserMealLogging (
        userID CHAR(8),
        mealID CHAR(8),
        timestamp DATETIME,
        PRIMARY KEY (userID, mealID),
        FOREIGN KEY (timestamp)
                REFERENCES ActivityLog(timestamp)
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
```

University of British Columbia, Vancouver
Department of Computer Science

```
CREATE TABLE User1 (
        height DECIMAL(3,2),
        weight DECIMAL(5,2),
        BMI DECIMAL(4,2),
        PRIMARY KEY (height, weight)

);
```

```
CREATE TABLE User2(
        userID CHAR(8),
        username VARCHAR(99) NOT NULL,
        age INTEGER,
        gender VARCHAR(50),
        height DECIMAL(3,2),
        weight DECIMAL(5,2),
        PRIMARY KEY (userID)

);
```

```
CREATE TABLE NutritionalValue1(
        protein DECIMAL(7,2),
        fat DECIMAL(7,2),
        sodium DECIMAL(7,2),
        carbs DECIMAL(7,2),
        nutrCalories DECIMAL(7,2) NOT NULL,
        PRIMARY KEY (protein, fat, sodium, carbs)
);
```

```
CREATE TABLE NutritionalValue2(
        nutrID CHAR(8),
        UNIQUE ingName VARCHAR(99),
        UNIQUE procName VARCHAR(50),
        UNIQUE brand VARCHAR(50),
        protein DECIMAL(7,2),
        fat DECIMAL(7,2),
        sodium DECIMAL(7,2),
        carbs DECIMAL(7,2),
        PRIMARY KEY (nutrID),
        FOREIGN KEY (ingName)
                REFERENCES Ingredient (ingName)
                ON UPDATE CASCADE
                ON DELETE SET NULL
        FOREIGN KEY (procName, brand)
                REFERENCES ProcessedFood (procName, brand)
                ON UPDATE CASCADE
                ON DELETE SET NULL
);
```

9. 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    Ingredient (ingName, nutrID, ingUnit, category)
VALUES ('chicken', 'N000001', '100 gram', 'poultry'),
       ('rice', 'N0000002', '100 gram', 'grains'),
       ('carrot', 'N0000003', '100 gram', 'root vegetable'),
       ('lettuce', 'N0000004', '1 leaf', 'leaf vegetable'),
       ('egg', 'N0000005', '1 egg', 'poultry');

INSERT
INTO    ProcessedFood (procName, brand, nutrID, pfUnit)
VALUES ('Strawberry Probiotic Yogurt', 'Activia', 'N0000006', 'pack'),
       ('Chicken Pot Pie', 'Marie Callender's', 'N000007', 'box'),
       ('Medium Iced Capp', 'Tim Hortons', 'N000008', 'cup'),
       ('Small Iced Capp', 'Tim Hortons', 'N0000009', 'cup'),
       ('Butter Chicken', 'President's Choice', 'N000010', 'box');

INSERT
INTO    Recipe (recipeID, cookTime, recipeName, steps, recipeCalories)
VALUES ('R0000001', 10, 'Sunny Side Up Omelet', 'Step 1: Omelet \r\n End recipe', 100),
       ('R0000002', 40, 'Butter Chicken', 'Step 1: Butter chicken \r\n End recipe', 400),
       ('R0000003', 20, 'Mixed Caesar Salad', 'Step 1: MCS \r\n End recipe', 100),
       ('R0000004', 60, 'Chicken Stew', 'Step 1: Chicken stew \r\n End recipe', 300),
       ('R0000005', 25, 'Egg Fried Rice', 'Step 1: Egg fried rice \r\n End recipe', 300);

INSERT
INTO    RecipIngr (recipeID, ingName)
VALUES ('R0000001', 'egg'),
       ('R0000002', 'rice'),
       ('R0000002', 'chicken'),
       ('R0000003', 'lettuce'),
       ('R0000003', 'carrot'),
       ('R0000004', 'chicken'),
       ('R0000004', 'carrot'),
       ('R0000005', 'rice'),
       ('R0000005', 'egg');
```

University of British Columbia, Vancouver
Department of Computer Science

```
INSERT
INTO    NutritionalValue1 (protein, fat, sodium, carbs, nutrCalories)
VALUES (30.2, 3.6, 0.8, 0.0, 153.0),
       (2.7, 0.3, 0.001, 28.0, 130.0),
       (0.8, 0.2, 0.058, 8.2, 35.0),
       (0.3, 0.0, 6.7, 0.7, 4.0),
       (6.0, 5.0, 0.062, 0.6, 78.0),
       (4.0, 1.5, 0.055, 15.0, 90.0),
       (17.0, 36.0, 0.95, 55.0, 610.0),
       (3.0, 15.0, 0.07, 48.0, 360.0),
       (2.0, 11.0, 0.05, 33.0, 250.0),
       (20.0, 18.0, 1.05, 53.0, 450.0);

INSERT
INTO    NutritionalValue2 (nutrID, ingName, procName, brand, protein, fat, sodium, carbs)
VALUES ('N0000001', 'chicken', NULL, NULL, 30.2, 3.6, 0.8, 0.0),
       ('N0000002', 'rice', NULL, NULL, 2.7, 0.3, 0.001, 28.0),
       ('N0000003', 'carrot', NULL, NULL, 0.8, 0.2, 0.058, 8.2),
       ('N0000004', 'lettuce', NULL, NULL, 0.3, 0.0, 6.7, 0.7),
       ('N0000005', 'egg', NULL, NULL, 6.0, 5.0, 0.062, 0.6),
       ('N0000006', NULL, 'Strawberry Probiotic Yogurt', 'Activia', 4.0, 1.5, 0.055, 15.0),
       ('N0000007', NULL, 'Chicken Pot Pie', 'Marie Callender's', 17.0, 36.0, 0.95, 55.0),
       ('N0000008', NULL, 'Medium Iced Capp', 'Tim Hortons', 3.0, 15.0, 0.07, 48.0),
       ('N0000009', NULL, 'Small Iced Capp', 'Tim Hortons', 2.0, 11.0, 0.05, 33.0),
       ('N0000010', NULL, 'Butter Chicken', 'President's Choice', 20.0, 18.0, 1.05, 53.0);


INSERT
INTO    User1 (height, weight, BMI)
VALUES (1.70, 68.0, 23.5),
       (1.80, 75.0, 23.1),
       (1.60, 55.0, 21.5),
       (1.75, 70.0, 22.9),
       (1.65, 62.0, 22.8),
       (1.65, 70.2, 25.8),
       (1.70, 80.1, 27.7),
       (1.49, 35.7, 16.1),
       (2.15, 92.0, 19.9),
       (1.52, 50.1, 21.7);
```

University of British Columbia, Vancouver
Department of Computer Science

```
INSERT
INTO    User2 (userID, username, age, gender, height, weight)
VALUES ('U0000001', 'Grace', 35, 'female', 1.70, 68.0),
        ('U0000002', 'Henry', 40, 'male', 1.80, 75.0),
        ('U0000003', 'Isabel', 28, 'female', 1.60, 55.0),
        ('U0000004', 'Jack', 33, 'male', 1.75, 70.0),
        ('U0000005', 'Karen', 45, 'female', 1.65, 62.0),
        ('U0000010', 'Alice', 30, 'female', 1.65, 70.2),
        ('U0000020', 'Bob', 25, 'male', 1.70, 80.1),
        ('U0000030', 'Charlie', 19, NULL, NULL, NULL),
        ('U0000040', 'Dan', 51, 'non-binary', 1.49, 35.7),
        ('U0000050', 'Erikson', 42, 'male', 2.15, 92.0),
        ('U0109050', 'FeralLion', 60, 'female', 1.52, 50.1);

INSERT
INTO    Admin (userID, adminLevel)
VALUES ('U0000001', 'Beta Tester'),
        ('U0000002', 'Recipe Writer'),
        ('U0000003', 'Community Moderator'),
        ('U0000004', 'Security Administrator'),
        ('U0000005', 'Database Manager');

INSERT
INTO    RegularUser (userID, subscriptionType)
VALUES ('U0000010', 'Free'),
       ('U0000020', 'Premium'),
       ('U0000030', 'Free'),
       ('U0000040', 'Free'),
       ('U0000050', 'Premium'),
       ('U0109050', 'Premium');

INSERT
INTO    UserDietaryRestriction (dietaryName, userID, restrictionType, description)
VALUES ('SeafoodDairyAllergy', 'U0000010', 'Seafood, Dairy', 'Cannot eat seafood and cheese'),
       ('LacGluIntolerance', 'U0000020', 'Lactose, Gluten', 'Lactose and gluten intolerance'),
       ('VegetarianDiet', 'U0000030', 'Vegetarian', 'No meat, fish, or poultry'),
       ('LactoOvoVegetarianDiet', 'U0000040', 'Lacto-Ovo Vegetarian', 'Includes dairy and eggs
       but no meat, fish, or poultry'),
       ('KosherDiet', 'U0000050', 'Pork', 'Follows Jewish dietary laws regarding avoiding pork');
```

```
INSERT
INTO     Makes (userID, recipeID)
VALUES (‘U0000010’, ‘R0000001’),
          (‘U0000020’, ‘R0000005’),
          (‘U0000030’, ‘R0000003’),
          (‘U0000040’, ‘R0000004’),
          (‘U0000050’, ‘R0000002’);

INSERT
INTO     ProcMeal (brand, procName, mealID)
VALUES ('Activia', 'Strawberry Probiotic Yogurt', 'M000001'),
          ('Marie Callender’s', 'Chicken Pot Pie', 'M000002'),
          ('Tim Hortons', 'Medium Iced Capp', 'M000003'),
          ('Tim Hortons', 'Small Iced Capp', 'M0000004'),
          ('President’s Choice', 'Butter Chicken', 'M000005');

INSERT
INTO     ActivityLogReporting (timestamp, userID, activityType)
VALUES ('2024-07-21 08:30:00', 'U0000010', ‘Ran 10km’),
           ('2024-07-24 12:00:00', 'U0000020', “Cycling 5km’),
           ('2024-07-26 15:30:00', 'U0000030', ‘Swimming 5km’),
           ('2024-07-29 16:45:00', 'U0000040', ‘Rock Climbing 50m’),
           ('2024-07-30 06:30:00', 'U0000050', ‘Hiking 3km’);

INSERT
INTO Creating (userID, recipeID)
VALUES ('U0000010',  'R0000001'),
          ('U0000020', 'R0000005'),
          ('U0000030', 'R0000003'),
          ('U0000040', 'R0000004'),
          ('U0000050', 'R0000002');

INSERT
INTO MealEating (mealID, userID)
VALUES ('M0000001', 'U0000010'),
          ('M0000002', 'U0000020'),
          ('M0000004', 'U0000030'),
          ('M0000001', 'U0000040'),
          ('M0000005', 'U0000050');
```

University of British Columbia, Vancouver
Department of Computer Science

```
INSERT
INTO UserMealLogging (logID, userID, mealID, timestamp)
VALUES ('L00000001', 'U0000010', 'M0000001', '2024-07-21 08:00:00'),
        ('L0000002', 'U0000020', 'M0000002', '2024-07-23 20:30:00'),
        ('L0000003', 'U0000030', 'M0000004', '2024-07-25 16:00:00'),
        ('L0000004', 'U0000040', 'M0000001', '2024-07-26 11:30:00'),
        ('L0000005', 'U0000050', 'M0000005', '2024-07-30 20:00:00');
```