Maria-Noelia Herne and Aprina Wang
mherne1@jhu.edu and awang100@jhu.edu

Our project creates a recipe management database that stores recipe information (ingredients used, their nutrition, the steps required) and associated information (recipe reviews, users, dietary restrictions)
https://www.ugrad.cs.jhu.edu/~mherne1/

During this project, we created a database that serves as a recipe management library. At the core of the database is our Recipe entity. In relation to Recipes, we also have Ingredients—containing information about the nutritional contents of the food—and Steps—which, ordered together, compose how to make a recipe—and Interactions—which store reviews that users have left on recipes—and Tags—which store tags associated with each recipe. For ingredients, there is a RecipeIngredientLink table that annotates the amount of an ingredient used. Further, Ingredients may also be Allergens, which indicate if they may trigger a certain type of Allergy.

Database Design

This database design has changed slightly since Phase I. This is mostly due to the nature of the raw data we were able to find and the extent to which we were able to clean and combine it. Details about data collection are listed below. The following DDL specification shows how our database is designed:

DDL Statements:

```
DROP TABLE IF EXISTS User;

CREATE TABLE User (
    user_id INTEGER NOT NULL,
    username VARCHAR(25) NOT NULL,
    email VARCHAR(255) NOT NULL,
    PRIMARY KEY (user_id)
);

DROP TABLE IF EXISTS Recipe;

CREATE TABLE Recipe (
    recipe_name VARCHAR(255) NOT NULL,
    recipe_id INTEGER NOT NULL,
    minutes INTEGER,
    user_id INTEGER NOT NULL,
    date DATETIME,
    description VARCHAR(6000),
    calories INTEGER,
    PRIMARY KEY (recipe_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);

DROP TABLE IF EXISTS Ingredient;

CREATE TABLE Ingredient (
```

```
    ingredient_id INTEGER NOT NULL,
    Energy_Kcal INTEGER,
    Protein FLOAT,
    Carbohydrate FLOAT,
    Fiber FLOAT,
    Sugar FLOAT,
    Calcium FLOAT,
    Iron FLOAT,
    Magnesium FLOAT,
    Sodium FLOAT,
    Cholestrl FLOAT,
    description VARCHAR(100) NOT NULL,
    unit INTEGER,
    volume_quantity INTEGER,
    volume_desc VARCHAR(25),
    weight_quantity INTEGER,
    weight_desc VARCHAR(25),
    PRIMARY KEY (ingredient_id)
);

DROP TABLE IF EXISTS RecipeIngredientLink;

CREATE TABLE RecipeIngredientLink (
    recipe_id INTEGER NOT NULL,
    ingredient_id INTEGER NOT NULL,
    quantity DECIMAL(4,2),
    quantity_desc VARCHAR(25),
    variant VARCHAR(25) NOT NULL,
    FOREIGN KEY (recipe_id) REFERENCES Recipe(recipe_id),
    FOREIGN KEY (ingredient_id) REFERENCES
Ingredient(ingredient_id)
);

DROP TABLE IF EXISTS Interaction;

CREATE TABLE Interaction (
    user_id INTEGER NOT NULL,
    recipe_id INTEGER NOT NULL,
    date DATETIME,
    rating INTEGER,
    review VARCHAR(20000),
```

```
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (recipe_id) REFERENCES Recipe(recipe_id)
);

DROP TABLE IF EXISTS Allergy;

CREATE TABLE Allergy (
    allergy VARCHAR(25) NOT NULL,
    allergy_id INTEGER NOT NULL,
    PRIMARY KEY (allergy_id)
);

DROP TABLE IF EXISTS Allergens;

CREATE TABLE Allergens (
    allergy_id INTEGER NOT NULL,
    ingredient_id INTEGER NOT NULL,
    FOREIGN KEY (allergy_id) REFERENCES Allergy(allergy_id),
    FOREIGN KEY (ingredient_id) REFERENCES
Ingredient(ingredient_id)
);

DROP TABLE IF EXISTS Tag;

CREATE TABLE Tag (
    tag_id INTEGER NOT NULL,
    tag_name VARCHAR(255),
    PRIMARY KEY (tag_id)
);

DROP TABLE IF EXISTS TagLink;

CREATE TABLE TagLink (
    recipe_id INTEGER NOT NULL,
    tag_id INTEGER,
    FOREIGN KEY (recipe_id) REFERENCES Recipe(recipe_id),
    FOREIGN KEY (tag_id) REFERENCES Tag(tag_id)
);

DROP TABLE IF EXISTS Steps;
```

```
CREATE TABLE Steps (
    recipe_id INTEGER NOT NULL,
    steps VARCHAR(255),
    step_order INTEGER,
    FOREIGN KEY (recipe_id) REFERENCES Recipe(recipe_id)
);

DROP TABLE IF EXISTS Conversions;

CREATE TABLE Conversions (
    from_unit VARCHAR(25),
    to_unit VARCHAR(25),
    factor DECIMAL(5,2)
);
```

Data Loading

The data was loaded using datasets from Kaggle. We used Python (pandas and fuzzywuzzy) to clean and make connections between RAW data, as well as doing some web scraping using BeautifulSoup from Food.com to get additional recipe information.

More details about the exact datasets used can be found in `/data/generate_csv.ipynb`, as well as other notebooks found in that folder. Instructions for use are found at the beginning of the notebook. Additionally, information about the specific data used is found in the README.

How To Run

Tech Stack

We stuck with using a MySQL/PHP tech stack since that was what was introduced in class and that was what we were familiar with.

Populating Tables

In order to populate the database, all sql files are located in the `./populate` folder. `/populate/0tables.sql` (contains DDL statements) should be run first to create all the tables. Then the following files (*in the order specified*) can be run to insert the data:

1. `user.sql`
2. `recipe.sql`
3. `ingredient.sql`
4. `recipe_link.sql`
5. `interaction.sql`
6. `allergy.sql`
7. `allergens.sql`
8. `tag.sql`

9. `taglink.sql`
10. `steps.sql`
11. `conversions.sql`

Specializations

The area we decided to focus on was data mining. In particular, we did some extensive data cleaning and joining as well as web scraping. It required us to learn many new Python libraries and the process of web scraping.

Strengths

- Clean recipe and ingredient data
- Over 1000 recipes in the database
- Ability to search for recipes based on ingredients

Limitations + Extensions

- More querying capabilities with the interface
- Some of the data is a bit imperfect due to the way we scraped and clean it