Database Documentation

## Overview

This section covers the database implementation for the project. We use MySQL as our database management system with a single schema db04 (as required by the server) that contains both application data and user-related data.

## Directory Structure

DB/  
├── Data/ **# Contains raw data files**  
│ └── CSV/ **# Contains CSV data files (after processing)**  
├── Migration/ **# Python scripts, Jupyter Notebooks for data migration**  
├── SQL/ **# SQL scripts for database operations**  
│ ├── Initialisation.sql **# Schema initialisation script for the database**  
│ └── Insertion/ **# Data insertion scripts for the database**  
└── Synth/ **# Synthetic data generation scripts and output files**

## Database Schema

### Application Data Tables

Tables containing application data: food and beverage data, nutritional data, activity data, etc…

#### Tables:

1. **Category**
   * Primary food and beverage categories, its goal is to group food and beverages into categories.
   * Fields:
     + ID (SMALLINT UNSIGNED): Category ID
     + Description (VARCHAR(150)): Category description
   * Primary Key: ID
   * Indexes: ID
2. **Food\_Beverages**
   * Main food and beverage items, its goal is to store the food and beverage items in the database.
   * Fields:
     + ID (INT UNSIGNED): Food/Beverage ID
     + Description (VARCHAR(200)): Food/Beverage description
     + Extra\_Description (VARCHAR(450)): Extra description
     + Category\_ID (SMALLINT UNSIGNED): Category ID
     + Score (FLOAT UNSIGNED): Health score
   * Primary Key: ID
   * Foreign Keys: Category\_ID references Category(ID)
   * Indexes: ID
3. **Portions\_Weights**
   * Provided portion sizes for foods, its goal is to store portion sizes for each food and beverage item.
   * Fields:
     + FB\_ID (INT UNSIGNED): Food/Beverage ID
     + Description (VARCHAR(200)): Description of the portion size
     + Weight (FLOAT UNSIGNED): Weight of the portion size
   * Primary Key: FB\_ID, Description, Weight
   * Foreign Keys: FB\_ID references Food\_Beverages(ID)
   * Indexes: FB\_ID
4. **Nutrients**
   * Nutrient definitions and properties.
   * Fields:
     + ID (SMALLINT UNSIGNED): Nutrient ID
     + Name (VARCHAR(50)): Nutrient name
     + Unit (ENUM("kcal", "g", "mg", "mcg")): Unit of the nutrient
     + Importance (FLOAT): Importance of the nutrient, used to calculate the health score and obtained using linear regression (see Jupyter Notebook found in Migration/Nutrient\_Importance.ipynb).
   * Primary Key: ID
   * Indexes: ID
5. **Ingredients**
   * Ingredient table, store ingredient descriptions along with their ID.
   * Fields:
     + ID (INT UNSIGNED): Ingredient ID
     + Description (VARCHAR(150)): Ingredient description
   * Primary Key: ID
   * Indexes: ID
6. **Ingredients\_Values**
   * Nutrient values for each ingredient.
   * Fields:
     + Ingredient\_ID (INT UNSIGNED): Ingredient ID
     + Nutrient\_ID (SMALLINT UNSIGNED): Nutrient ID
     + Value (FLOAT UNSIGNED): Value of the nutrient (i.e. amount of the nutrient per 100g of the ingredient)
   * Primary Key: Ingredient\_ID, Nutrient\_ID
   * Foreign Keys:
     + Ingredient\_ID references Ingredients(ID)
     + Nutrient\_ID references Nutrients(ID)
   * Indexes: Ingredient\_ID, Nutrient\_ID
7. **FB\_Ingredients**
   * Mapping between food and beverage items and the ingredients they contain.
   * Fields:
     + FB\_ID (INT UNSIGNED): Food/Beverage ID
     + Ingredient\_ID (INT UNSIGNED): Ingredient ID
     + Weight (FLOAT UNSIGNED): Weight of the ingredient in the food/beverage item
   * Primary Key: FB\_ID, Ingredient\_ID, Weight
   * Foreign Keys:
     + FB\_ID references Food\_Beverages(ID)
     + Ingredient\_ID references Ingredients(ID)
   * Indexes: FB\_ID, Ingredient\_ID
8. **FB\_Values**
   * Mapping between food and beverage items and their nutrient values.
   * Fields:
     + FB\_ID (INT UNSIGNED): Food/Beverage ID
     + Nutrient\_ID (SMALLINT UNSIGNED): Nutrient ID
     + Value (FLOAT UNSIGNED): Value of the nutrient (i.e. amount of the nutrient per 100g of the food/beverage item)
   * Primary Key: FB\_ID, Nutrient\_ID
   * Foreign Keys:
     + FB\_ID references Food\_Beverages(ID)
     + Nutrient\_ID references Nutrients(ID)
   * Indexes: FB\_ID, Nutrient\_ID
9. **RDI**
   * Recommended Daily Intake values per nutrient.
   * Fields:
     + Nutrient\_ID (SMALLINT UNSIGNED): Nutrient ID
     + Value (FLOAT UNSIGNED): Value of the nutrient (i.e. amount of the nutrient per 100g of the food/beverage item)
   * Primary Key: Nutrient\_ID
   * Foreign Keys: Nutrient\_ID references Nutrients(ID)
   * Indexes: Nutrient\_ID
10. **Activities**
    * Physical activities and their calorie burn rates.
    * Fields:
      + ID (SMALLINT UNSIGNED AUTO\_INCREMENT): Activity ID
      + Description (VARCHAR(150)): Description of the activity
      + CaloriesPKG (FLOAT UNSIGNED): Calories burned per 1kg of body weight
    * Primary Key: ID
    * Indexes: ID

### User Data Tables

Tables containing user data: user information, logged meals.

#### Tables:

1. **User**
   * User profile information.
   * Fields:
     + ID (INT UNSIGNED AUTO\_INCREMENT): User ID
     + Email (VARCHAR(50)): User email
     + First\_Name (VARCHAR(50)): User first name
     + Last\_Name (VARCHAR(50)): User last name
     + Password (VARCHAR(50)): User password
     + DOB (DATE): User date of birth
     + Gender (ENUM("M", "F")): User gender
     + Weight (SMALLINT UNSIGNED): User weight
   * Primary Key: ID
   * Indexes: ID
2. **User\_Food\_Beverages**
   * Logged meals per user.
   * Fields:
     + User\_ID (INT UNSIGNED): User ID
     + FB\_ID (INT UNSIGNED): Food/Beverage ID
     + Portion\_Weight (FLOAT UNSIGNED): Weight of the portion size
     + Date\_Time (DATETIME): Date and time of the meal
   * Primary Key: User\_ID, FB\_ID, Date\_Time
   * Foreign Keys:
     + User\_ID references User(ID)
     + FB\_ID references Food\_Beverages(ID)
   * Indexes: User\_ID, Date\_Time

## Data Sources

### Main Dataset

The main dataset is the 2021-2023 FNDDS dataset, which is given by the US Department of Agriculture and contains food and beverage data, nutritional data, etc…

The dataset is obtained from [here](https://www.ars.usda.gov/northeast-area/beltsville-md-bhnrc/beltsville-human-nutrition-research-center/food-surveys-research-group/docs/fndds-download-databases/), and is a collection of the following Excel files:

* 2021-2023 FNDDS At A Glance - FNDDS Ingredients.xlsx
* 2021-2023 FNDDS At A Glance - FNDDS Nutrient Values.xlsx
* 2021-2023 FNDDS At A Glance - Foods and Beverages.xlsx
* 2021-2023 FNDDS At A Glance - Ingredient Nutrient Values.xlsx
* 2021-2023 FNDDS At A Glance - Portions and Weights.xlsx

### Additional Datasets

* RDI values tables, given by the US Food and Drug Administration (FDA) and obtained from [here](https://www.fda.gov/media/99069/download) and [here](https://www.fda.gov/media/99059/download). The tables were parsed manually to create the RDI table.
* Exercise dataset found in Data/exercise\_dataset.csv and obtained from [here](https://www.kaggle.com/datasets/aadhavvignesh/calories-burned-during-exercise-and-activities).
* OpenFoodFacts dataset compressed using xz and found in Data/en.openfoodfacts.org.products.tsv.xz and obtained from [here](https://www.kaggle.com/datasets/openfoodfacts/world-food-facts).

## Data Migration Scripts

### Main Data Processing Scripts

In the Migration/ folder, you can find the following scripts:

* Category.py – Processes the 2021-2023 FNDDS At A Glance - Foods and Beverages.xlsx file and creates the Category table.
* FB\_Ingredients.py – Processes the 2021-2023 FNDDS At A Glance - FNDDS Ingredients.xlsx file and creates the FB\_Ingredients table.
* FB\_Values.py – Processes the 2021-2023 FNDDS At A Glance - FNDDS Nutrient Values.xlsx file and creates the FB\_Values table.
* Food\_Beverages.py – Processes the 2021-2023 FNDDS At A Glance - Foods and Beverages.xlsx file and creates a part of the Food\_Beverages table.
* Ingredients.py – Processes the 2021-2023 FNDDS At A Glance - FNDDS Ingredients.xlsx file and creates the Ingredients table.
* Ingredients\_Values.py – Processes the 2021-2023 FNDDS At A Glance - Ingredient Nutrient Values.xlsx file and creates the Ingredients\_Values table.
* Nutrients.py – Processes the 2021-2023 FNDDS At A Glance - FNDDS Nutrient Values.xlsx file and creates a part of the Nutrients table.
* Portions\_Weights.py – Processes the 2021-2023 FNDDS At A Glance - Portions and Weights.xlsx file and creates the Portions\_Weights table.

### Additional Data Processing Scripts

In the Migration/ folder, you can find the following files:

* Activities.py – Processes exercise dataset found in Data/exercise\_dataset.csv and creates the Activities table.
* Nutrient\_Importance.ipynb – Processes OpenFoodFacts dataset found in Data/en.openfoodfacts.org.products.tsv.xz and modifies the Nutrients and Food\_Beverages tables to include the nutrient importance values and nutritional scores.

## Synthetic Data Generation

In the Synth/ folder, you can find the following files:

* User.py – Generates synthetic user profiles and outputs them to a CSV file.
* User\_Food\_Beverages.py – Generates synthetic food consumption records and outputs them to a CSV file.

## Database Initialisation

The initialisation process follows these steps:

1. Create schema db04 with appropriate character encoding (utf8\_bin)
2. Run table creation script from SQL/Initialisation.sql
3. Run synthetic data generation scripts from Synth/
4. Run data insertion scripts from SQL/Insertion/
   1. Data insertion script
   2. Synthetic data insertion script

## SQL Queries Documentation

### Food & Beverage Queries

#### Search Foods

SELECT \*  
FROM Food\_Beverages  
WHERE LOWER(Description) LIKE %s

Searches for food items where the description contains the given search term. Before the query is executed, the search term is converted to lowercase to ensure case-insensitive matching.

#### Search Food Category

SELECT Category.Description  
FROM Category, Food\_Beverages  
WHERE Food\_Beverages.ID = %s  
AND Food\_Beverages.Category\_ID = Category.ID

Retrieves the category description for a specific food item.

#### Search Food Name

SELECT Description  
FROM Food\_Beverages  
WHERE Food\_Beverages.ID = %s

Retrieves the description of a specific food item.

#### Calculate Food Calories

SELECT (%s \* FB\_Values.Value / 100) AS Value  
FROM FB\_Values  
JOIN Nutrients ON FB\_Values.Nutrient\_ID = Nutrients.ID  
WHERE FB\_Values.FB\_ID = %s AND Nutrients.Name = 'Energy'

Calculates calories for a given food weight (first parameter) and food ID (second parameter).

#### Get Food Portions

SELECT Description, Weight  
FROM Portions\_Weights  
WHERE FB\_ID = %s

Retrieves standardised portion sizes for a specific food item.

#### Get Food Ingredients

SELECT i.ID, i.Description,  
 %s \* fb.Weight / (SELECT SUM(Weight)  
 FROM FB\_Ingredients  
 WHERE FB\_ID = fb.FB\_ID) AS Normalized\_Weight  
FROM FB\_Ingredients AS fb  
JOIN Ingredients AS i ON fb.Ingredient\_ID = i.ID  
WHERE fb.FB\_ID = %s  
ORDER BY fb.Weight DESC

Retrieves ingredients and their normalised weights for a given food item.

#### Get RDI Percentages

SELECT  
 Nutrients.Name,  
 %s \* (FB\_Values.Value / 100) / RDI.Value AS Percentage  
FROM  
 FB\_Values  
 JOIN Nutrients ON FB\_Values.Nutrient\_ID = Nutrients.ID  
 JOIN RDI ON Nutrients.ID = RDI.Nutrient\_ID  
WHERE  
 FB\_ID = %s AND RDI.Value > 0  
ORDER BY   
 FIELD(Nutrients.Unit, 'kcal', 'g', 'mg', 'mcg') ASC,  
 Percentage DESC

Calculates the percentage of Recommended Daily Intake (RDI) for each nutrient in a food item.

#### Get Nutrient Distribution

SELECT  
 cte.Nutrient\_Name,  
 (  
 cte.total\_std\_amount  
 / SUM(cte.total\_std\_amount)  
 OVER (PARTITION BY cte.FB\_ID)  
 ) as Percentage  
FROM  
(  
 SELECT  
 fb.FB\_ID,  
 iv.Nutrient\_ID,  
 n.Name AS Nutrient\_Name,  
 n.Unit,  
 SUM(  
 CASE n.Unit  
 WHEN 'g' THEN fb.Weight \* iv.Value \* 1000  
 WHEN 'mg' THEN fb.Weight \* iv.Value  
 WHEN 'mcg' THEN fb.Weight \* iv.Value / 1000  
 ELSE 0  
 END  
 ) AS total\_std\_amount  
 FROM FB\_Ingredients AS fb  
 JOIN Ingredients\_Values AS iv  
 ON fb.Ingredient\_ID = iv.Ingredient\_ID  
 JOIN Nutrients AS n  
 ON iv.Nutrient\_ID = n.ID  
 WHERE fb.FB\_ID = %s  
 AND n.Unit <> 'kcal'  
 GROUP BY  
 fb.FB\_ID,  
 iv.Nutrient\_ID,  
 n.Unit  
) AS cte  
ORDER BY  
 cte.total\_std\_amount DESC  
LIMIT %s

Calculates the percentage distribution of nutrients in a food item, standardising units for comparison.

#### Search Nutrients Per Ingredient

SELECT  
 Nutrients.Name,  
 (Value \* %s / 100) AS Value,  
 Unit  
FROM  
 FB\_Values JOIN Nutrients  
 ON FB\_Values.Nutrient\_ID = Nutrients.ID  
WHERE  
 FB\_ID = %s  
ORDER BY   
 FIELD(Nutrients.Unit, 'kcal', 'g', 'mg', 'mcg') ASC,  
 Value DESC

Retrieves nutrient values for a given food item by weight.

#### Get Nutrition Score

SELECT Score \* 100  
FROM Food\_Beverages as fb  
WHERE fb.ID = %s

Retrieves the nutrition score for a given food item.

### User Queries

#### Create User

INSERT INTO User (Email, First\_Name, Last\_Name, Password, DOB, Gender, Weight)  
VALUES (%s, %s, %s, %s, %s, %s, %s)

Creates a new user record.

#### Get User

SELECT \* FROM User WHERE Email = %s AND Password = %s

Retrieves user information with authentication.

#### Get User Weight

SELECT Weight FROM User WHERE ID = %s

Retrieves user weight.

#### Get Full Name

SELECT First\_Name, Last\_Name FROM User WHERE ID = %s

Retrieves user full name.

#### Update User

UPDATE User  
SET First\_Name = %s, Last\_Name = %s, Weight = %s  
WHERE ID = %s

Updates user profile information.

#### Log Meal

INSERT INTO User\_Food\_Beverages (User\_ID, FB\_ID, Portion\_Weight, Date\_Time)  
VALUES (%s, %s, %s, %s)

Records a meal consumed by a user.

#### Get User Meals History

SELECT FB.description, User\_Food\_Beverages.Portion\_Weight, User\_Food\_Beverages.Date\_Time  
FROM User\_Food\_Beverages  
JOIN Food\_Beverages FB ON User\_Food\_Beverages.FB\_ID = FB.ID  
WHERE User\_ID = %s  
ORDER BY Date\_Time DESC

Retrieves a user’s meal history, ordered by date and time.

#### Calculate User Nutrition Score

SELECT AVG(FB.Score \* UFB.Portion\_Weight) / AVG(UFB.Portion\_Weight) AS Score  
FROM User\_Food\_Beverages UFB  
JOIN Food\_Beverages FB ON UFB.FB\_ID = FB.ID  
WHERE UFB.User\_ID = %s

Calculates a user’s nutrition score based on their meal history.

#### Calculate Relative Nutrition Score

WITH User\_Scores AS (  
 SELECT UFB.User\_ID, AVG(FB.Score \* UFB.Portion\_Weight) / AVG(UFB.Portion\_Weight) AS Score  
 FROM User\_Food\_Beverages UFB  
 JOIN Food\_Beverages FB ON UFB.FB\_ID = FB.ID  
 GROUP BY UFB.User\_ID  
),  
User\_Relative\_Scores AS (  
 SELECT User\_ID, Score, PERCENT\_RANK() OVER (ORDER BY Score ASC) AS Relative\_Score  
 FROM User\_Scores  
)  
SELECT Relative\_Score  
FROM User\_Relative\_Scores  
WHERE User\_ID = %s

Calculates a user’s nutrition score percentile rank compared to the score of other users.

### Activity Queries

#### Search Activities by Calories and Weight

SELECT ID, Description, (CaloriesPKG \* %s) AS Calories\_Burned  
FROM Activities  
ORDER BY ABS(CaloriesPKG \* %s - %s) ASC  
LIMIT %s