MyFitnessPal

DATABASE MANAGEMENT

PROJECT REPORT

By Group 7

Debarati Saha

Eyohas Haile

Hanyaqian Yuan

Soumya Chava

## 1.Introduction

Myfitnesspal is an app that solves a real problem and relies strongly on its database. It has several features and is extensive. Having a “structured database” is of utmost importance for the app, and that’s why we chose myfitnesspal.  
We have tried our best to accommodate all of its features in our database.  
  
MyFitnessPal is an online calorie counter which helps users to lose weight by tracking calorie intake and calorie burnt quickly.

We will design a database to facilitate the following functions:

- Create user profile

- Record the food intake and the corresponding calorie intake

- Record the exercise and corresponding calorie burnt

- Record water consumption

- Set Goals for themselves

## 2.Table Normalization

The exercise table is in 2-NF because each non-key column depends on whole candidate keys, not on a subset of any candidate key.

Table1. Exercise Table



However it is not in 3-NF because exercise type is dependent on exercise description. To make it to 3-NF table,then

Table1-1: ExerciseDesc(PK), ExerciseType

Table1-2: ExerciseID(PK), ExerciseDesc(FK), CalBurnedPerMinPerlbs

The goal table is in 2 -NF because each non-key column depends on whole candidate keys, not on a subset of any candidate key.

Table2. Goal Table



However it is not in 3-NF because the daily calorie limit is determined by the combination of goal type, goal level start weight limit and end weight limit. our assumption is that people with the same goal type(lose, maintain or gain weight) , choose the same goal level( easy, medium or hard) and whose weight falls in the same interval(decided by the start weight limit and end weight limit) will have the same daily calorie limit. To make it a 3-NF table, then

Table 2-1: GoalType(PK), GoalLevel(PK), StartWeightLimit(PK), EndWeightLimit(PK), DailyCalorieLimit

Table 2-2: GoalID(PK), GoalType(FK), GoalLevel(FK), StartWeightLimit(FK), EndWeightLimit(FK), WeeklyWeightGainOrLoss

The User table is in 2-NF because each non-key column depends on whole candidate keys, not on a subset of any candidate key.

Table3. User Table



However it is not in 3-NF because the country is determined by zip code. To make it to 3-NF table,then

Table 3-1: ZipCode(PK), Country

Table3-2: UserID(PK), UseName, BirthDATE, StartingWeight, Height, ZipCode(FK), JoinDate

All the tables below are all 3-NF because each non-key column depends on whole candidate keys, not on a subset of any candidate key and also each non-key column depends only on candidate keys, not on other non-key columns.

Table4.Exercise Log Table



Table5. Food Log Table



Table6. Friend List Table



Table7. Goal Log Table



Table8. Food Table



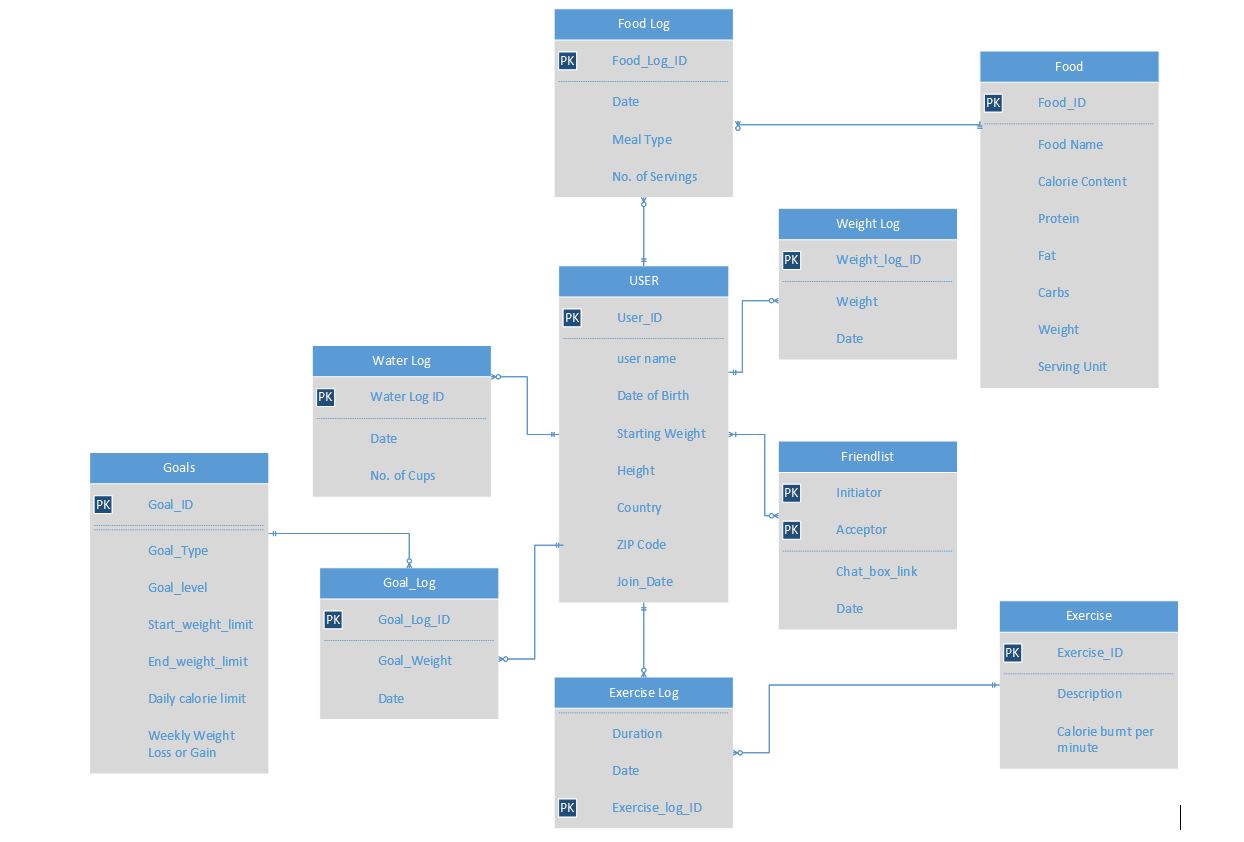
Table9. Water Log Table



Table10. Weight Log Table



## 3.ERD



### ERD Breakdown

#### 1. User

Every user needs to create a profile to use the application. The user database will store the name, date of birth, current weight, height, country of origin, ZIP code Goal Weight and Join Date of the user.

#### 2. Calories Consumed. The Database records the calories consumed by a user. This is facilitated by the User, Food Log & Food entities. The food database will store all different kind of foods a person can eat. It stores details such as Food Name, Calorie Content, Protein, Fat, Carbs, Weight and Serving Unit

User should mention when they had that food, in which meal (Breakfast/lunch/dinner/snacks), and the number of servings they had. This data will be stored in the food log table. The food log table stores details such as Date, Meal type and No. of servings.

* Each User can have 0 to many Food logs and every food log will be associated with one and only one User.
* Each Food log can have one and only one food item and any food item can be used 0 to many food logs.

#### 3. Calories burned

Exercise Database is useful for determining the calories burnt through various forms of exercises.

This is facilitated by the User, Exercise and Exercise Log entities.  
The exercise database will have a unique ID (Exercise\_ID) for each exercise, and will store other details such as description and calories burnt per minute.  
Exercise log database is useful for recording the exercises the user does each day and how long does every exercise last.

* Each User can have 0 to many Exercise logs and every exercise log will be associated with one and only one User.
* Each Exercise log can have one and only one exercise and any exercise can be used 0 to many exercise logs.

#### 4. Water Log The water log database stores the amount of water consumed by a user through the day. Each entry has a unique Water\_log\_ID It stores data such as Date, No. of cups consumed.

* Each user can have 0 to many water logs and each water log will be associated with one and only one user.

#### 5. Weight Log

The weight log database stores the changes in weight that the user updates. Each update of weight will have its own unique weight\_log\_id. It stores further details such as Weight and Date.

* Each user can have 0 to many weight logs and each weight log will be associated with one and only one user.

6. Goals  
Each user in the app can set goals for themselves. They will be advised on how many calories they can consume per day depending on the goal they set for themselves. This is facilitated by the User, Goals and Goal Log entities.   
The Goals database will have a unique ID (Goal\_ID) for each goal, and will store other details such as Goal\_type, Goal\_level, Start\_weight\_limit, end\_weight\_limit, Daily\_calorie\_limit, Weekly\_weight\_loss\_or\_gain  
Goal log database is useful for recording the goals each user has set for themselves throughout their life in the app. It store a unique Goal\_log\_ID and Date.

* Each User can have 0 to many Goal logs and every goal log will be associated with one and only one User.
* Each Goal log can have one and only one goal and any goal can be used 0 to many goal logs.

#### 7. Friendlist Any user on the app can connect their social media profiles to the app which will sync their existing friend base and show the user all their friends who are currently using the app. Now the user can add all these people as their friends in the app, and the app will provide a chatbox link so that they can chat with each other within the app. This is an optional feature that could be used at the users discretion. This is facilitated by the friendlist entity. Which saves data such as initiator, acceptor and Date. It saves the user Id of the intiator and also the user ID of the person who receive the request and accepts the request(Acceptor) and the Date on which they have become friends.

4.New FeaturesFeature 1

One of the new features we are adding to the app is a facility of a user to connect with a trainer that is best suited for them. We will have information about the trainers for example trainer experience, skill set, etc.  
To add this feature we have to store data of the users who wish to opt to get a trainer, and data regarding the trainers.   
  
To achieve this we added 2 new entities to the ERD, namely, User\_trainer, Trainer.  
  
User\_trainer database stores the data of all the users who have opted to get a trainer and the trainer database stores the data regarding trainers which include information such as first\_name, Last\_name, Exp\_years & Expertise. Each trainer will have a unique ID that is the trainer\_id.

* Each User can have 0 to many trainers and every user\_trainer ID will be associated with one and only one User.
* Each User\_trainer ID can have one and only one trainer and any trainer can be associated with 0 to many User\_trainer ID’s.



Feature 2  
Another new feature that we have decided to add is the facility of any two users who are friends with each other to challenge each other to compare what portion of their target weight goal (loss/gain) they have achieved in a certain period of time. These challenges will be pre-established in the app.  
  
To achieve this we added 1 new entities to the ERD that is Challenges.  
  
Each challenge will have a unique challenge ID.This entity will also store user Id of the challenger, the user ID of the opponent, the start\_date and end\_date of the challenge.  


### ERD After New Features Added



## 5.Scenarios

We have 8 scenarios to show how our Database works and its functionality.

## Scenario 1

Calories consumed by users per day, from their food log. **Query:**Select USER.USERID,FOODLOG.FoodLogDATE,

SUM(FOOD.FoodCalories\*foodlog.NoOfServing) AS TOTAL\_CALORIES\_CONSUMED

FROM USER,FOODLOG,FOOD

WHERE USER.USERID=FOODLOG.USERID AND

FOOD.FOODID=FOODLOG.FOODID

GROUP BY USER.USERID,FOODLOG.FoodLogDate

### **Output:**

### Calorie consumed by users per day per meal

**Query:**

Select USER.USERID,FOODLOG.FoodLogDATE,foodlog.TypeOfMeal,

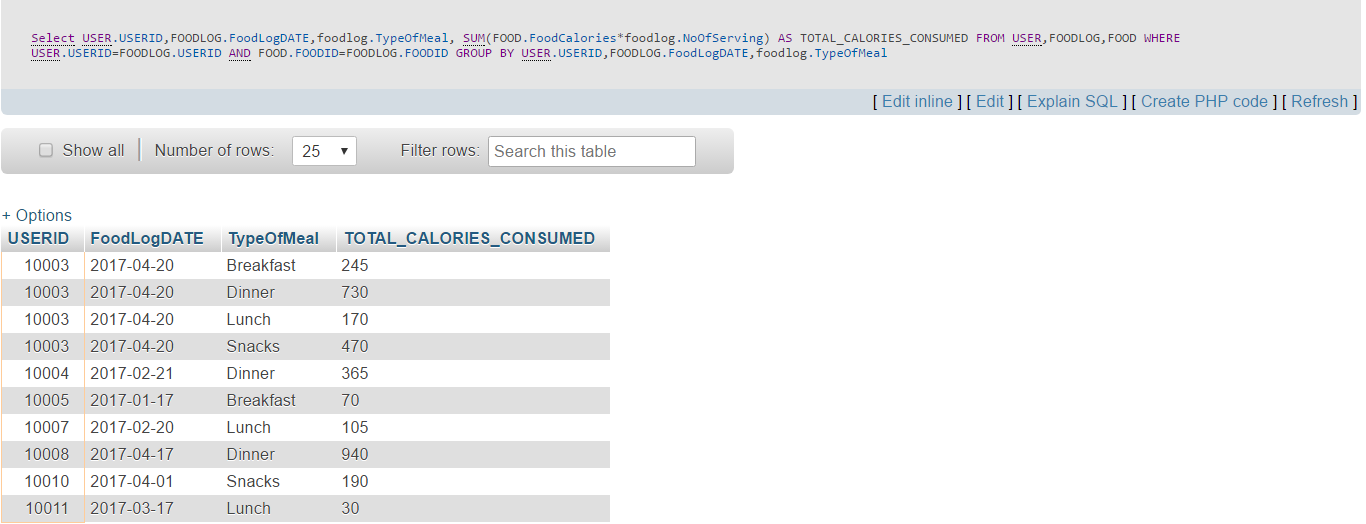
SUM(FOOD.FoodCalories\*foodlog.NoOfServing) AS TOTAL\_CALORIES\_CONSUMED

FROM USER,FOODLOG,FOOD

WHERE USER.USERID=FOODLOG.USERID AND

FOOD.FOODID=FOODLOG.FOODID

GROUP BY USER.USERID,FOODLOG.FoodLogDATE,foodlog.TypeOfMeal

**Output:  
  
  
Scenario 2:**

* Total calories burnt by a user per day per exercise.

**Query:**SELECT userID, exerciselog.ExLogDate, exercise.ExerciseDesc,

SUM(exerciselog.Duration\*EXERCISE.CalBurnedPerMin) AS TotalCalorie

From Exerciselog join exercise

on exerciselog.ExerciseID=exercise.ExerciseID

GROUP BY userid, ExLogDATE, exerciselogID

### **Output:**

### Total calorie burnt by per user per day:

**Query:**SELECT userID, exerciselog.ExLogDate,

SUM(exerciselog.Duration\*EXERCISE.CalBurnedPerMin) AS Total

From Exerciselog join exercise

on exerciselog.ExerciseID=exercise.ExerciseID

GROUP BY userid, ExLogDATE

**Output:**

****

**Scenario 3:**Average daily water consumption per user: **Query:**SELECT w.UserID, AVG(DailyTotal) AS AvgDailyCups

FROM

( SELECT UserID, DATE, SUM(NoOfCups) AS DailyTotal

FROM waterlog

GROUP BY waterlog.DATE, UserID

) w

GROUP BY w.UserID

**Output:  
  
Scenario 4:**When a user wants to set a goal, he will get goal options based on his current body weight and whether he wants to lose or gain or maintain weight  
Below are the examples where two different users gets different daily calorie intake limits based on their body weights:

* For User 10014:

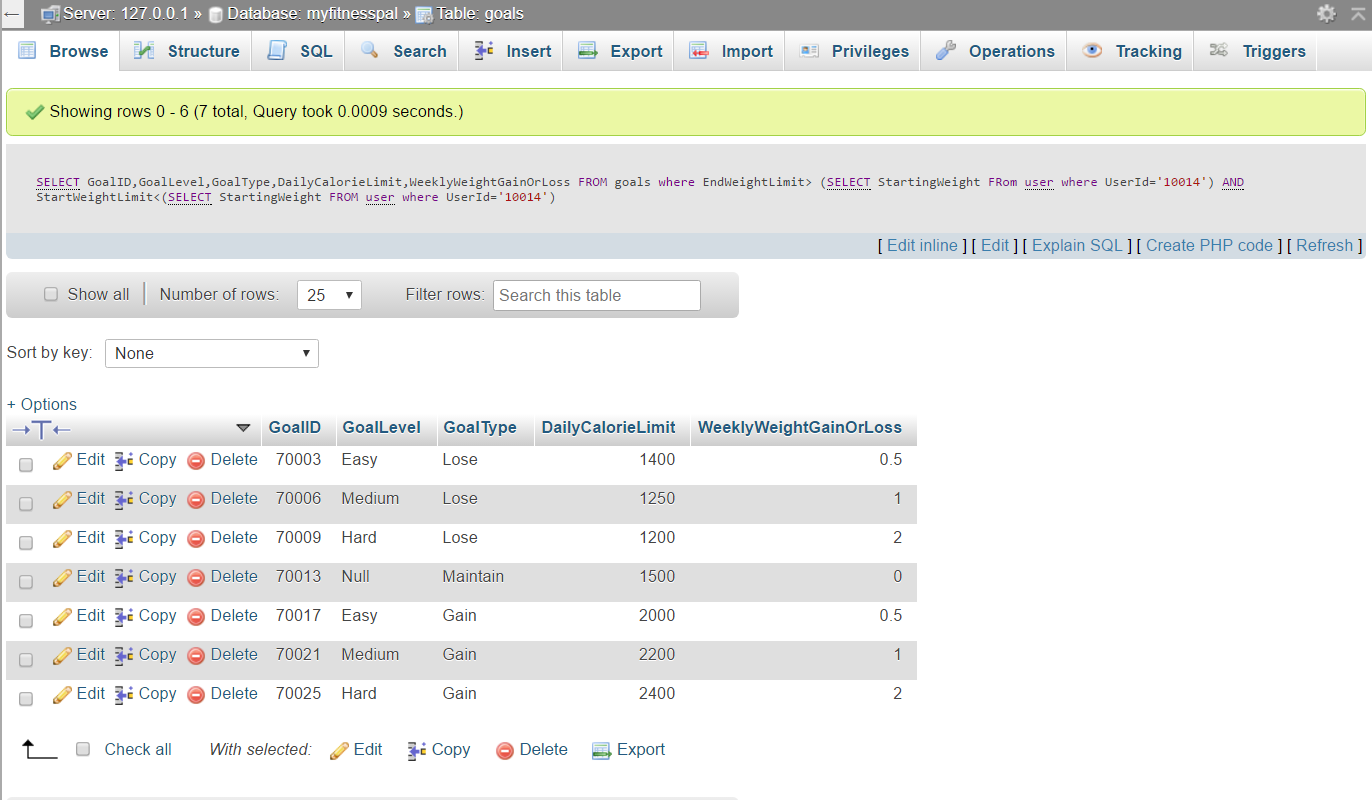
**Query:**SELECT GoalID,GoalLevel,GoalType,DailyCalorieLimit,WeeklyWeightGainOrLoss

FROM goals

where USERID = '10014' AND EndWeightLimit>

(SELECT StartingWeight FRom user where UserId='10014')

AND StartWeightLimit<(SELECT StartingWeight FROM user where UserId='10014')

**Output:**

* For User 10002:

**Query:**

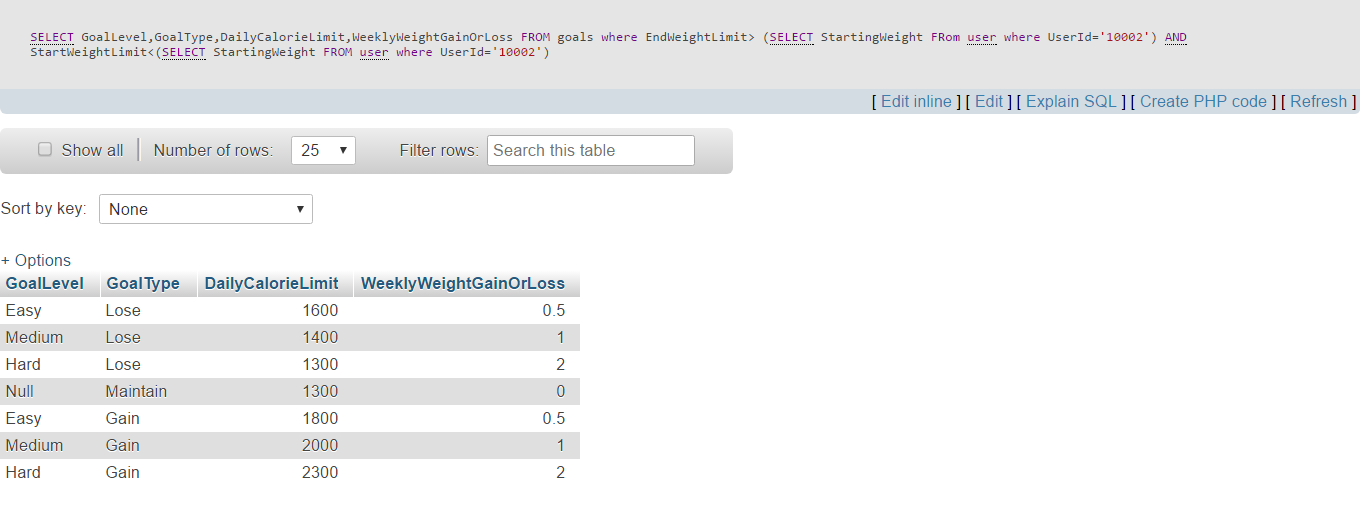
SELECT GoalLevel,GoalType,DailyCalorieLimit,WeeklyWeightGainOrLoss

FROM goals

where EndWeightLimit>

(SELECT StartingWeight FRom user where UserId='10002')

AND StartWeightLimit<(SELECT StartingWeight FROM user where UserId='10002')

**Output:  
  
Scenario 5:**

How long should it take the users to reach their goals? [Only the users who wants to lose or gain weight

**Query:**SELECT user.UserID,

((user.StartingWeight-GoalLog.Goal\_Weight)/Goals.WeeklyWeightGainOrLoss)

AS WeeksNeededToReachGoal

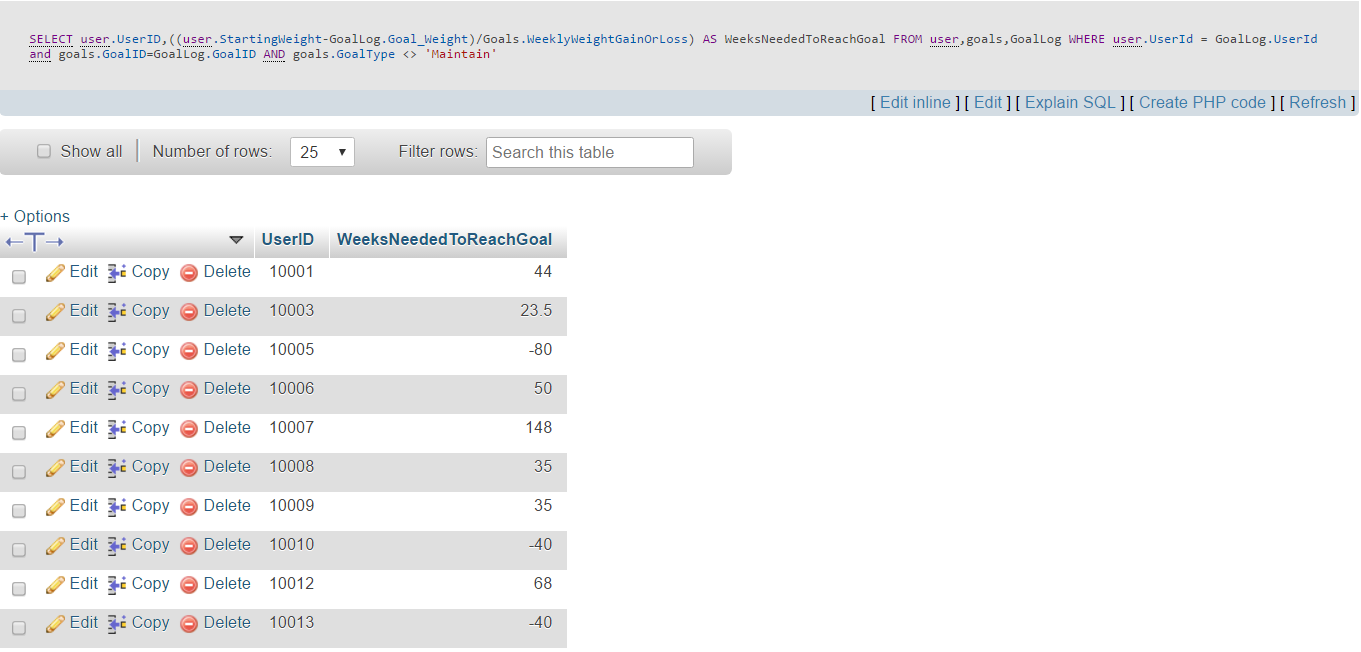
FROM user,goals,GoalLog

WHERE user.UserId = GoalLog.UserId

AND goals.GoalID=GoalLog.GoalID

AND goals.GoalType <> 'Maintain'

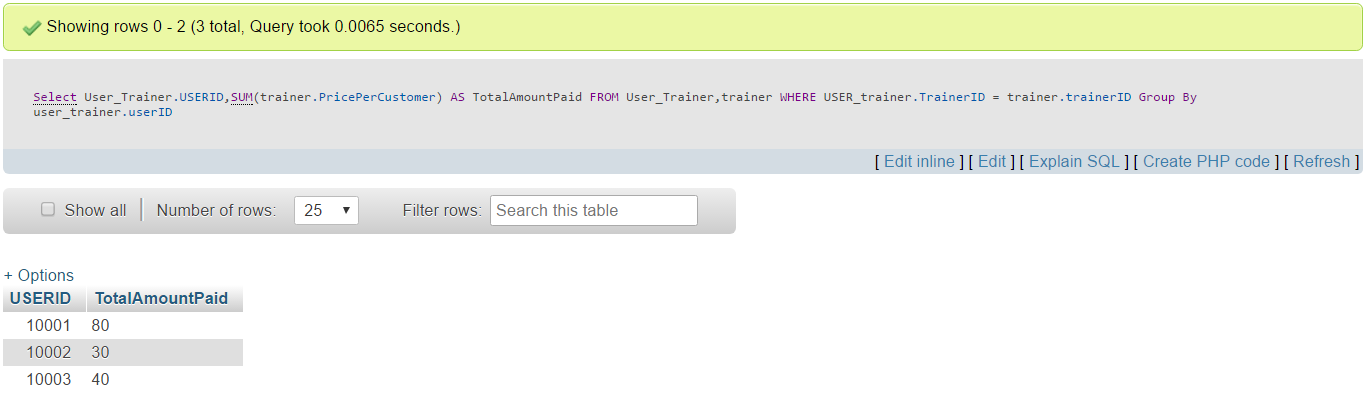
**Output:**

**  
Scenario 6:**How much money have users paid for trainers till now? **Query:**Select User\_Trainer.USERID,SUM(trainer.PricePerCustomer) AS TotalAmountPaid

FROM User\_Trainer,trainer

WHERE USER\_trainer.TrainerID = trainer.trainerID

Group By user\_trainer.userID

**Output:  
  
Scenario 7:**

The weight journey of any user, i.e. how much weight a user has lost or gained till date, since he/she has joined the app [Users who never logged their weight after joining, the weight loss value will show as ‘0’ for them] **Query:**

SELECT u.UserID, u.StartingWeight, gl.Goal\_Weight,

wl.LatestWeightLogDate,wl.Weight AS LatestWeight,

COALESCE((u.StartingWeight-wl.Weight),0) AS WeightLoss

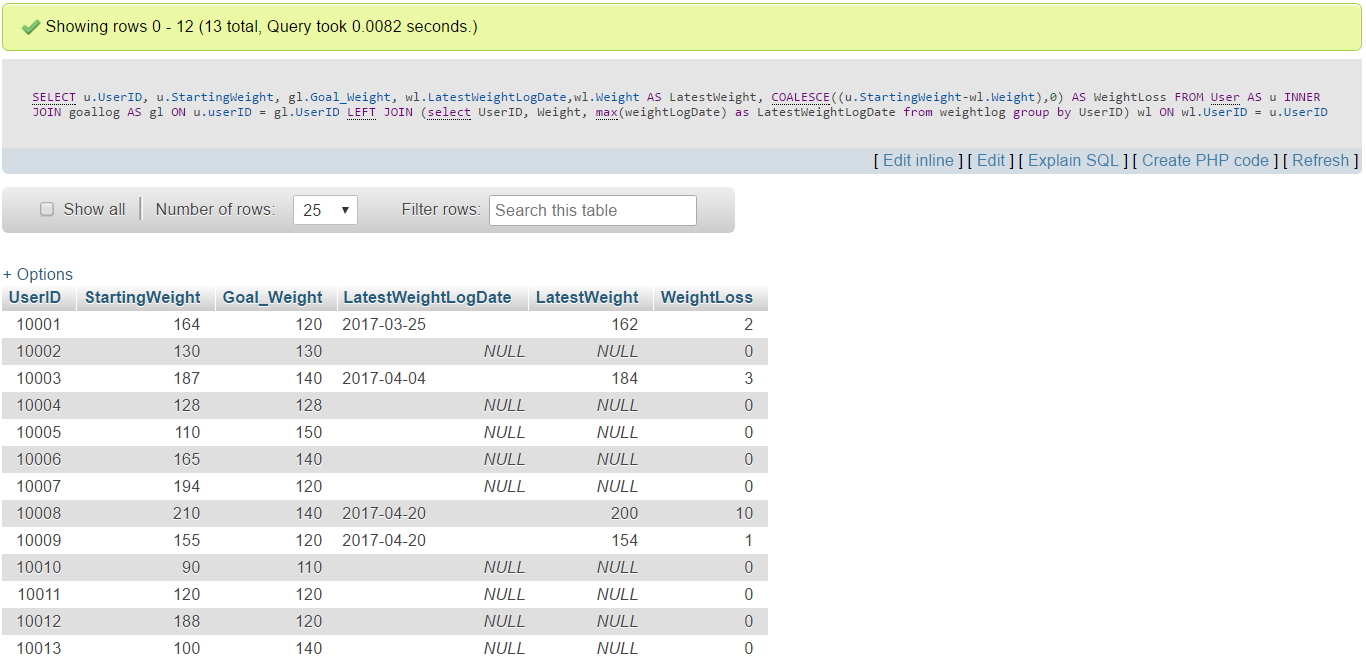
FROM User AS u INNER JOIN goallog AS gl ON u.userID = gl.UserID LEFT JOIN

(select UserID, Weight, max(weightLogDate) as LatestWeightLogDate

from weightlog

group by UserID) wl

ON wl.UserID = u.UserID

**Output  
**

**Scenario 8:**Daily nutrients consumed by any user: **Query:**Select USER.USERID,FOODLOG.FoodLogDATE,

SUM(FOOD.Carbs\*foodlog.NoOfServing) AS TOTAL\_CARBS\_CONSUMED,

SUM(FOOD.FAT\*foodlog.NoOfServing) AS TOTAL\_FAT\_CONSUMED,

SUM(FOOD.Protein\*foodlog.NoOfServing) AS TOTAL\_Protein\_CONSUMED

FROM USER,FOODLOG,FOOD

WHERE USER.USERID=FOODLOG.USERID AND

FOOD.FOODID=FOODLOG.FOODID

GROUP BY USER.USERID,FOODLOG.FoodLogDate

**Output:**

