Final Report (Phase One and Phase Two)

CSE334s – Spring 2025

May 17, 2025

Team Members

Name	ID – Section
Shams El-Din Mohamed Abdel-Monem	2101442 – S3
Ahmed Said Sayed	2101546 – S3
Abdelrahman Sherif Hassan	2100735 – S3
Fares Khalaf Salman Sultan	2101371 – S3
Kareem Gaber El Halaby	2101545 – S3
Omar Tamer Mohamed	2100528 – S3
Youssof Waleed Fathi	2101734 – S3
Yousef Mohammed Hassan Abouelela	2101076 – S2

Contents

List of Figures

1 Domain

We chose a **Fitness Tracker** app as our domain, which allows users to track calorie, water and food intakes, as well as biometric data like heart rate, and so. We also incorporated **Gamification** elements, such as points to motivate users to reach their goals.

2 Domain Description (Functional Requirements)

- **User Management:** Register, authenticate, and manage user profiles with personal and health-related attributes.
- **Social Interaction:** Enable users to follow and befriend other users.
- **Diet Tracking and Logging:** Manage diet plans, log food and drink intake, and track hydration and nutrition details.
- **Biometric Tracking:** Log and monitor health metrics like BMI, weight, and heart rate.
- **Exercise Tracking:** Create and track exercise routines with details like duration and muscle groups.
- **Goal Setting:** Set and track user goals with start/end dates and progress points.
- **Challenges:** Allow users to join and track challenges with descriptions and durations, track progress.
- **Gamification:** Implement a points system to reward users for completing tasks and challenges.

3 Assumptions and Constraints

- User can follow exactly one diet plan or none. A diet plan is custom to a single user.
  - **One-to-one** with **total participation of Diet_Plan** and **partial participation of User**
- User can join as many challenges as needed at same time or none. A challenge can only exist if at least one user joins it.
  - **Many-to-many** with **total participation of Challenge** and **partial participation of User**
- User can do as many exercises as needed at same time or none. An exercise can exist even if no user does it.
  - **Many-to-many** with **partial participation of User, Exercise**
- User can target as many goals as needed at same time or none. A goal is custom to a single user.
  - **Many-to-many** with **partial participation of User, Goal**
- User can record logs like water, in-body measurements, biometric and food intake
  - **Disjointness** with **total participation of child entity**
- Goal is a gamification element, to give points to User, User can only have one goal at a time, to motivate and concentrate on it.
  - **One-to-one** with **partial participation of Goal, User**
- Many **Exercise**s can be done by a **User**. **Exercise** can exist even if User didn't do it, by being a part of a **Goal**.
  - **Many-to-many** with **partial participation of User, Exercise**

- Challenge is a gamification element, to give more points to User, combines many Goals together, User can join as many challenges as needed at same time or none. A challenge can only exist if at least one user joins it.
  - **Many-to-many** with **total participation of Challenge** and **partial participation of User**
  - **Many-to-many** with **partial participation of Challenge** and **total participation of Goal**
- Logs can be identified by a unique ID (no need for part of it's ID to be with User relationship), and can be recorded by a user at any time.
  - **Log** is not a weak entity.

4 EER Diagram

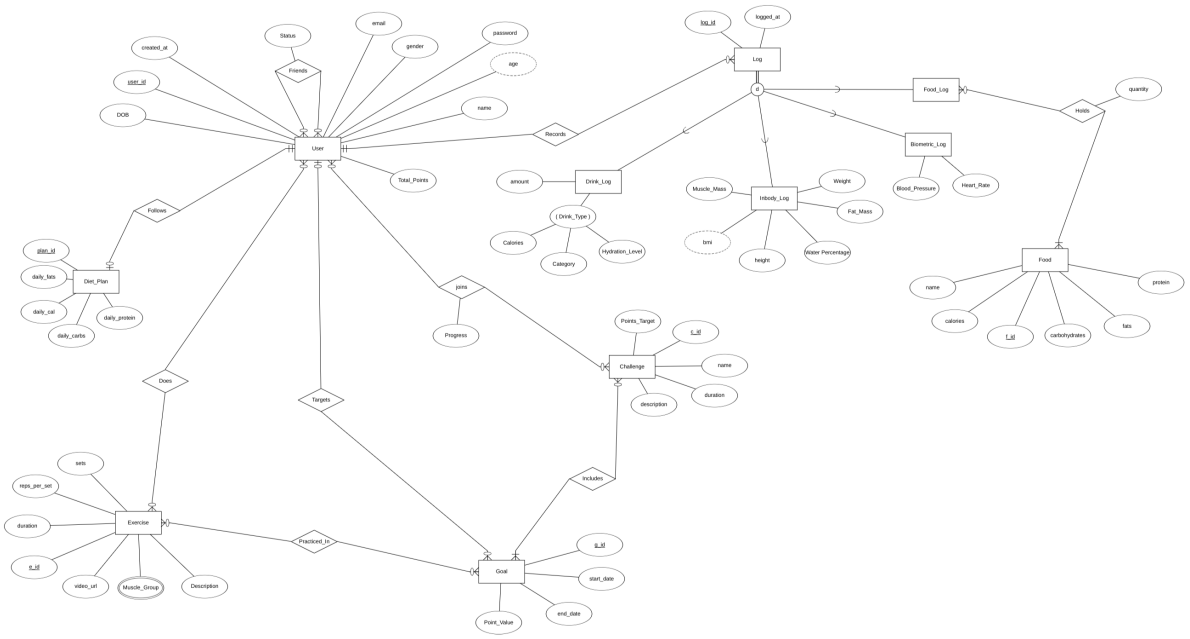

Fig. 1: EER Diagram

5 Relational Schema

6 Important Reports

- *Report 1*
- *Report 2*
- *Report 3*
- *Report 4*
- *Report 5*
- *Report 6*
- *Report 7*
- *Report 8*

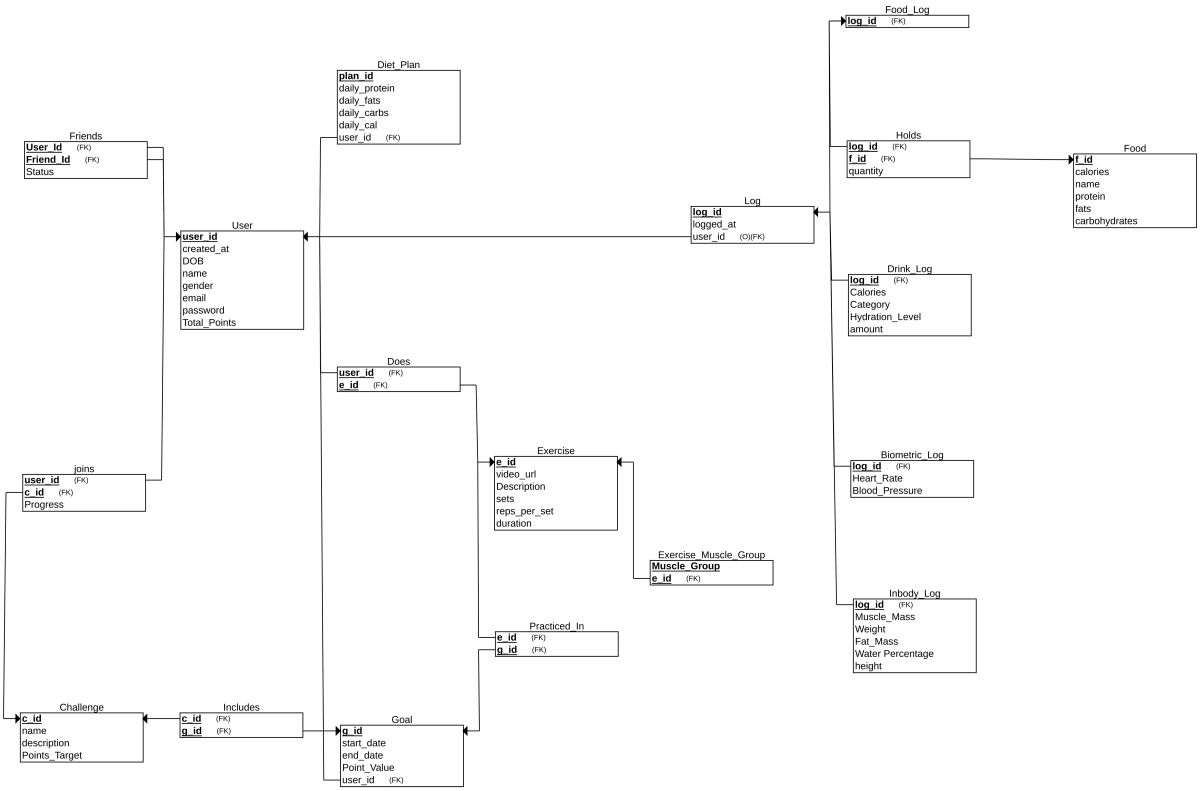

Fig. 2: Relational Schema

- *Report 9*
- *Report 10*
- *Report 11*

7 SQL Differences from EER

The constraint of `Goal` to be bind to one user had to be broken, as it is a mistake, we solved in SQL by dropping `NOT NULL` constraint on this attribute, id it is `NULL` then is bound to a Challenge if it is `NOT NULL` then it is specific to a single User.

8 DBMS Software (PostgreSQL)

8.1 Why PostgreSQL?

We used this DMBS software in Parallel and Distributed Systems, so we are already familiar with it, it allows creating `Domains` which we rely on, implements most `SQL` syntax that is including in `DDL` and `DML`. We have a `docker-compose` file (see *Metabase and Docker*) that automates creating PostgreSQL instance and populating it with database definitions and data.

8.2 DDL

We used domains for stuff used a lot and for enums. We tried putting `ON DELETE` and `ON UPDATE` referential constraints to match correct logical description. We had to declare `User` table with literal quotation mark `"User"` because `User` was a reserved word in PostgreSQL.

```
CREATE DOMAIN Points AS INTEGER CHECK (VALUE >= 0);
CREATE DOMAIN UINT AS INTEGER CHECK (VALUE >= 0);
CREATE TYPE gender_enum AS ENUM ('Male', 'Female');
CREATE TYPE category_enum AS ENUM ('WATER', 'JUICE', 'SODA');
CREATE TYPE status_enum AS ENUM ('Pending', 'Accepted', 'Rejected',
 ↪'Blocked');

CREATE TABLE "User"
(
    created_at INT NOT NULL,
    DOB DATE NOT NULL,
    name Varchar(50) NOT NULL,
    gender gender_enum NOT NULL,
    email Varchar(50) NOT NULL,
    password Varchar(50) NOT NULL,
    user_id INT NOT NULL,
    Total_Points Points NOT NULL,
    PRIMARY KEY (user_id),
    CONSTRAINT email_unique UNIQUE (email),
    CONSTRAINT chk_dob CHECK (DOB >= '1900-01-01' AND DOB <=␣
 ↪CURRENT_DATE)
);

CREATE TABLE Challenge (
    c_id INT NOT NULL,
    name VARCHAR(100) NOT NULL,
    description TEXT NOT NULL,
    Points_Target Points NOT NULL,
    PRIMARY KEY (c_id)
);

CREATE TABLE Goal (
    g_id INT NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    Point_Value Points NOT NULL,
    user_id INT,
    PRIMARY KEY (g_id),
    FOREIGN KEY (user_id) REFERENCES "User"(user_id) ON DELETE␣
 ↪CASCADE ON UPDATE CASCADE,
    CONSTRAINT chk_start_date CHECK (start_date >= CURRENT_DATE),
    CONSTRAINT chk_date_range CHECK (end_date > start_date)
);

CREATE TABLE Diet_Plan (
    daily_protein UINT NOT NULL,
    plan_id INT NOT NULL,
    daily_fats UINT NOT NULL,
    daily_carbs UINT NOT NULL,
    daily_cal UINT NOT NULL,
```

(continues on next page)

(continued from previous page)

```
    user_id INT NOT NULL,
    PRIMARY KEY (plan_id),
    FOREIGN KEY (user_id) REFERENCES "User"(user_id) ON DELETE␣
↪CASCADE ON UPDATE CASCADE
);

CREATE TABLE Exercise (
  e_id INT NOT NULL,
  video_url VARCHAR(255) NOT NULL,
  Description TEXT NOT NULL,
  sets UINT NOT NULL,
  reps_per_set INT NOT NULL CHECK(reps_per_set > 5 AND reps_per_set
↪< 20),
  duration FLOAT NOT NULL CHECK(duration > 0),
  PRIMARY KEY (e_id)
);

CREATE TABLE Log (
  log_id INT NOT NULL,
  logged_at TIMESTAMP NOT NULL,
  user_id INT,
  PRIMARY KEY (log_id),
  FOREIGN KEY (user_id) REFERENCES "User"(user_id) ON DELETE␣
↪CASCADE ON UPDATE CASCADE
);

CREATE TABLE Food_Log (
  log_id INT NOT NULL,
  PRIMARY KEY (log_id),
  FOREIGN KEY (log_id) REFERENCES Log(log_id) ON DELETE CASCADE ON␣
↪UPDATE CASCADE
);

CREATE TABLE Drink_Log (
  Calories UINT,
  Category category_enum NOT NULL,
  Hydration_Level UINT NOT NULL,
  amount UINT NOT NULL,
  log_id INT NOT NULL,
  PRIMARY KEY (log_id),
  FOREIGN KEY (log_id) REFERENCES Log(log_id) ON DELETE CASCADE ON␣
↪UPDATE CASCADE
);

CREATE TABLE Food (
  calories UINT NOT NULL,
  name VARCHAR(100) NOT NULL,
  protein UINT NOT NULL,
  fats UINT NOT NULL,
```

(continues on next page)

(continued from previous page)

```
  carbohydrates UINT NOT NULL,
  f_id INT NOT NULL,
  PRIMARY KEY (f_id)
);

CREATE TABLE Inbody_Log (
  Muscle_Mass UINT NOT NULL,
  Weight UINT NOT NULL,
  Fat_Mass UINT NOT NULL,
  Water_Percentage UINT NOT NULL,
  height UINT NOT NULL,
  log_id INT NOT NULL,
  PRIMARY KEY (log_id),
  FOREIGN KEY (log_id) REFERENCES Log(log_id) ON DELETE CASCADE ON␣
↪UPDATE CASCADE,
  CONSTRAINT chk_totality CHECK (Muscle_Mass + Fat_Mass + Water_
↪Percentage = 100)
);

CREATE TABLE Biometric_Log (
  Heart_Rate UINT NOT NULL CHECK ( Heart_Rate > 20 AND Heart_Rate <␣
↪200 ),
  Blood_Pressure_upper UINT NOT NULL,
  log_id INT NOT NULL,
  PRIMARY KEY (log_id),
  FOREIGN KEY (log_id) REFERENCES Log(log_id) ON DELETE CASCADE ON␣
↪UPDATE CASCADE
);

CREATE TABLE joins (
  Progress INT NOT NULL,
  user_id INT NOT NULL,
  c_id INT,
  PRIMARY KEY (user_id, c_id),
  FOREIGN KEY (user_id) REFERENCES "User"(user_id) ON DELETE␣
↪CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (c_id) REFERENCES Challenge(c_id) ON DELETE SET NULL␣
↪ON UPDATE CASCADE
);

CREATE TABLE Includes (
  c_id INT NOT NULL,
  g_id INT NOT NULL,
  PRIMARY KEY (c_id, g_id),
  FOREIGN KEY (c_id) REFERENCES Challenge(c_id) ON DELETE CASCADE␣
↪ON UPDATE CASCADE ,
  FOREIGN KEY (g_id) REFERENCES Goal(g_id) ON DELETE CASCADE ON␣
↪UPDATE CASCADE
);
```

(continues on next page)

(continued from previous page)

```
CREATE TABLE Does (
  user_id INT NOT NULL,
  e_id INT NOT NULL,
  PRIMARY KEY (user_id, e_id),
  FOREIGN KEY (user_id) REFERENCES "User"(user_id) ON DELETE␣
↪CASCADE ON UPDATE CASCADE ,
  FOREIGN KEY (e_id) REFERENCES Exercise(e_id) ON DELETE RESTRICT␣
↪ON UPDATE CASCADE
);

CREATE TABLE Practiced_In (
  e_id INT NOT NULL,
  g_id INT NOT NULL,
  PRIMARY KEY (e_id, g_id),
  FOREIGN KEY (e_id) REFERENCES Exercise(e_id) ON DELETE CASCADE ON␣
↪UPDATE CASCADE ,
  FOREIGN KEY (g_id) REFERENCES Goal(g_id) ON DELETE CASCADE ON␣
↪UPDATE CASCADE
);

CREATE TABLE Holds (
  quantity UINT NOT NULL,
  log_id INT NOT NULL,
  f_id INT NOT NULL,
  PRIMARY KEY (log_id, f_id),
  FOREIGN KEY (log_id) REFERENCES Food_Log(log_id) ON DELETE␣
↪CASCADE ON UPDATE CASCADE ,
  FOREIGN KEY (f_id) REFERENCES Food(f_id) ON DELETE RESTRICT ON␣
↪UPDATE CASCADE
);

CREATE TABLE Friends (
  "Status" status_enum NOT NULL,
  User_Id INT NOT NULL,
  Friend_Id INT NOT NULL,
  PRIMARY KEY (User_Id, Friend_Id),
  FOREIGN KEY (User_Id) REFERENCES "User"(user_id) ON DELETE␣
↪CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (Friend_Id) REFERENCES "User"(user_id) ON DELETE␣
↪CASCADE ON UPDATE CASCADE
);

CREATE TABLE Exercise_Muscle_Group (
  Muscle_Group VARCHAR(50) NOT NULL,
  Description TEXT NOT NULL,
  e_id INT NOT NULL,
  PRIMARY KEY (Muscle_Group, e_id),
  FOREIGN KEY (e_id) REFERENCES Exercise(e_id) ON DELETE CASCADE ON␣
```

(continues on next page)

(continued from previous page)

```
 ↪UPDATE CASCADE
);
```

8.3 DML

We inserted some data

```
-- User Table
INSERT INTO "User" (user_id, name, gender, email, password, DOB,␣
 ↪created_at, Total_Points) VALUES
(6, 'Emma Wilson', 'Female', 'emma.wilson@example.com', 'pass7890',
 ↪'1990-04-12', 1617235200, 600),
(7, 'Michael Lee', 'Male', 'michael.lee@example.com', 'pass1234',
 ↪'1988-08-25', 1625097600, 400),
(8, 'Sarah Kim', 'Female', 'sarah.kim@example.com', 'pass5678',
 ↪'1997-01-30', 1632960000, 250),
(9, 'David Chen', 'Male', 'david.chen@example.com', 'pass9012',
 ↪'1983-06-15', 1640822400, 800),
(10, 'Lisa Taylor', 'Female', 'lisa.taylor@example.com', 'pass3456',
 ↪ '1992-11-20', 1648684800, 150);

-- Challenge Table
INSERT INTO Challenge (c_id, name, description, Points_Target)␣
 ↪VALUES
(4, 'Hydration Challenge', 'Drink 2L of water daily for 30 days',␣
 ↪600),
(5, 'Strength Challenge', 'Increase bench press weight by 10kg',␣
 ↪900),
(6, 'Flexibility Challenge', 'Complete daily yoga for 1 month',␣
 ↪400);

-- Goal Table
INSERT INTO Goal (g_id, start_date, end_date, Point_Value, user_id)␣
 ↪VALUES
(4, '2025-05-18', '2025-06-18', 250, 6),
(5, '2025-05-20', '2025-07-20', 400, 7),
(6, '2025-06-01', '2025-08-01', 300, 8);

-- Diet_Plan Table
INSERT INTO Diet_Plan (plan_id, daily_protein, daily_fats, daily_
 ↪carbs, daily_cal, user_id) VALUES
(4, 130, 45, 190, 1900, 6),
(5, 160, 55, 220, 2200, 7),
(6, 110, 35, 170, 1700, 8);

-- Exercise Table
INSERT INTO Exercise (e_id, video_url, Description, sets, reps_per_
 ↪set, duration) VALUES
(4, 'http://example.com/deadlifts.mp4', 'Deadlifts', 3, 8, 2.5),
```

(continues on next page)

(continued from previous page)

```
(5, 'http://example.com/plank.mp4', 'Plank', 3, 12, 1.5),
(6, 'http://example.com/pullups.mp4', 'Pull-ups', 4, 10, 2.0);

-- Log Table
INSERT INTO Log (log_id, logged_at, user_id) VALUES
(7, '2025-05-15 08:00:00', 6),
(8, '2025-05-16 10:30:00', 6),
(9, '2025-05-16 14:00:00', 7),
(10, '2025-05-15 16:00:00', 7),
(11, '2025-05-16 09:00:00', 8),
(12, '2025-05-16 11:00:00', 8);

-- Adding corresponding entries in the Log table
INSERT INTO Log (log_id, logged_at, user_id) VALUES
(13, '2025-05-17 08:00:00', 6),
(14, '2025-05-17 09:00:00', 7),
(15, '2025-05-17 10:00:00', 8),
(16, '2025-05-17 11:00:00', 9),
(17, '2025-05-17 12:00:00', 10);

-- Food_Log Table
INSERT INTO Food_Log (log_id) VALUES
(7),
(11);

-- Drink_Log Table
INSERT INTO Drink_Log (log_id, Calories, Category, Hydration_Level,␣
↪amount) VALUES
(8, 0, 'WATER', 100, 750),
(12, 120, 'JUICE', 80, 250);

-- Adding more water logs
INSERT INTO Drink_Log (log_id, Calories, Category, Hydration_Level,␣
↪amount) VALUES
(13, 0, 'WATER', 100, 500),
(14, 0, 'WATER', 90, 750),
(15, 0, 'WATER', 85, 1000),
(16, 0, 'WATER', 95, 600),
(17, 0, 'WATER', 80, 800);

INSERT INTO Drink_Log (log_id, Calories, Category, Hydration_Level,␣
↪amount) VALUES
(7, 50,   'SODA',    15, 200),
(10, 50,   'SODA',       50, 300);

-- Food Table
INSERT INTO Food (f_id, name, calories, protein, fats,␣
↪carbohydrates) VALUES
(4, 'Salmon', 206, 22, 13, 0),
```

(continues on next page)

(continued from previous page)

```
(5, 'Quinoa', 222, 8, 4, 39),
(6, 'Banana', 90, 1, 0, 23);

-- Inbody_Log Table
INSERT INTO Inbody_Log (log_id, Muscle_Mass, Weight, Fat_Mass,␣
↪Water_Percentage, height) VALUES
(9, 38, 65, 17, 45, 170);

-- Biometric_Log Table
INSERT INTO Biometric_Log (log_id, Heart_Rate, Blood_Pressure_
↪upper) VALUES
(10, 65, 115);

-- joins Table
INSERT INTO joins (user_id, c_id, Progress) VALUES
(6, 4, 30),
(7, 5, 40),
(8, 6, 10);

-- Includes Table
INSERT INTO Includes (c_id, g_id) VALUES
(4, 4),
(5, 5),
(6, 6);

-- Does Table
INSERT INTO Does (user_id, e_id) VALUES
(6, 4),
(7, 5),
(8, 6);

-- Practiced_In Table
INSERT INTO Practiced_In (e_id, g_id) VALUES
(4, 4),
(5, 5),
(6, 6);

-- Holds Table
INSERT INTO Holds (log_id, f_id, quantity) VALUES
(7, 4, 1),
(7, 5, 1),
(11, 6, 2);

-- Friends Table
INSERT INTO Friends (User_Id, Friend_Id, "Status") VALUES
(6, 7, 'Accepted'),
(7, 6, 'Accepted'),
(8, 9, 'Pending'),
(9, 8, 'Pending'),
```

(continues on next page)

(continued from previous page)

```
(6, 10, 'Rejected');

-- Exercise_Muscle_Group Table
INSERT INTO Exercise_Muscle_Group (Muscle_Group, Description, e_id)␣
↪VALUES
('Back', 'Lats, traps, lower back', 4),
('Core', 'Abdominals, obliques', 5),
('Back', 'Lats, biceps', 6);
```

8.4 Reports

We tried visualizing all queries to be more interesting (see *Metabase and Docker*).

Report 1

Select the user with the highest total points.

```
SELECT u.name, u.email, u.Total_Points
FROM "User" as u
ORDER BY Total_Points DESC
Limit 1 ;
```

Result:

```
    name     |          email          | total_points
-------------+-------------------------+--------------
 David Chen | david.chen@example.com |          800
```

Report 2

Select the 5 users with the most Logs (Most active).

```
SELECT u.name, COUNT(l.log_id) AS log_count
FROM "User" u
JOIN Log l ON u.user_id = l.user_id
GROUP BY u.name
ORDER BY log_count DESC
LIMIT 5;
```

Result:

```
    name      | log_count
--------------+-----------
 Sarah Kim    |         3
 Michael Lee  |         3
 Emma Wilson  |         3
 Lisa Taylor  |         1
 David Chen   |         1
```

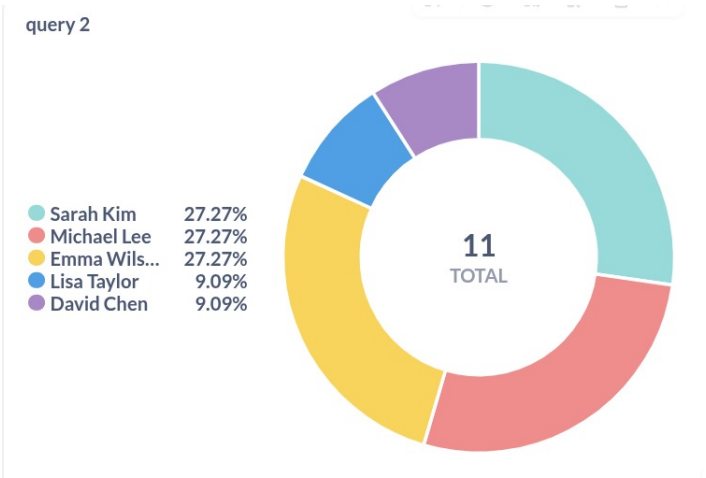

Fig. 3: Report 2 – Visualiztion (Pie)

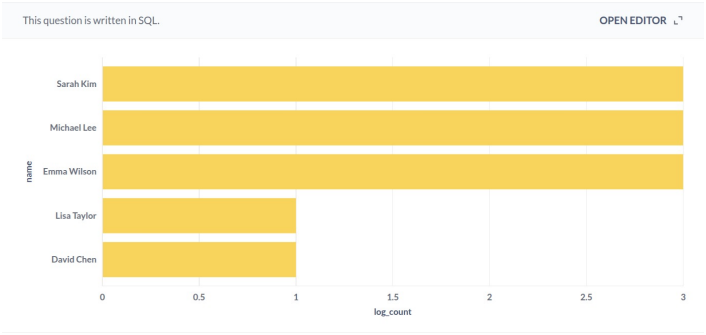

Fig. 4: Report 2 – Visualiztion (Bar)

Report 3

Getting the average protein intake for each gender (interesting statistic).

```
SELECT u.gender, AVG(dp.daily_protein) AS avg_protein
FROM "User" u
JOIN Diet_Plan dp ON u.user_id = dp.user_id
GROUP BY u.gender;
```

Result:

```
 gender |      avg_protein
--------+-----------------------
 Male   | 160.0000000000000000
 Female | 120.0000000000000000
```

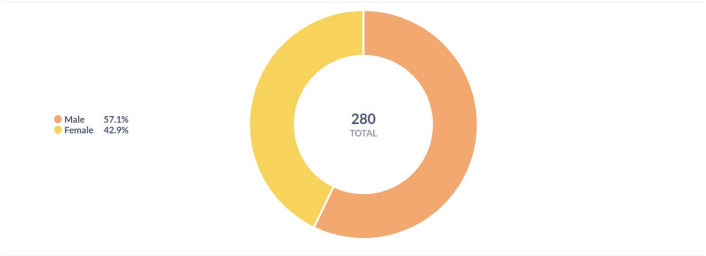

Fig. 5: Report 3 – Visualiztion (Pie)

Report 4

Select the users who have not logged any activity (to send promoting notifications).

```
SELECT name
FROM "User"
WHERE user_id NOT IN (
  SELECT user_id FROM Log
);
```

Result:

```
 name
------
```

No users with zero logs.

Report 5

Select the users who have logged soda drinks (to export to some soda drink company for generating leads).

```
SELECT DISTINCT u.name, d.Category
FROM "User" u
JOIN Log l ON u.user_id = l.user_id
```

(continues on next page)

(continued from previous page)

```
JOIN Drink_Log d ON d.log_id = l.log_id
WHERE d.Category = 'SODA';
```

Result:

```
    name      | category
--------------+----------
 Emma Wilson | SODA
 Michael Lee | SODA
```

Report 6

Select the 2 users with the least daily calories (to send notifications to warn or motivate).

```
SELECT u.name, dp.daily_cal
FROM "User" u
JOIN Diet_Plan dp ON u.user_id = dp.user_id
ORDER BY dp.daily_cal
LIMIT 2;
```

Result:

```
    name      | daily_cal
--------------+-----------
 Sarah Kim    |      1700
 Emma Wilson |      1900
```

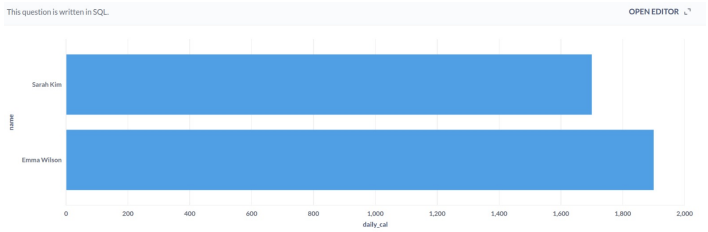

Fig. 6: Report 6 – Visualiztion (Bar)

Report 7

Rank users by their hydration level.

```
SELECT u.name, AVG(d.Hydration_Level) AS avg_hydration
FROM "User" u
JOIN Log l ON u.user_id = l.user_id
JOIN Drink_Log d ON d.log_id = l.log_id
WHERE d.Category = 'WATER'
GROUP BY u.name
ORDER BY avg_hydration DESC;
```

Result:

```
    name      |     avg_hydration
--------------+----------------------
 Emma Wilson  | 100.0000000000000000
 David Chen   |  95.0000000000000000
 Michael Lee  |  90.0000000000000000
 Sarah Kim    |  85.0000000000000000
 Lisa Taylor  |  80.0000000000000000
```

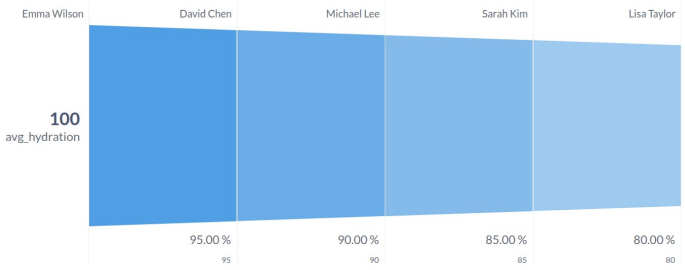

Fig. 7: Report 7 – Visualiztion

Report 8

Select challenges with a target of more than 500 points.

```
SELECT
    name,
    description,
    Points_Target
FROM
    Challenge
WHERE
    Points_Target > 500;
```

Result:

```
        name          |              description              | points_
↪target
----------------------+---------------------------------------+--------
↪-------
 Hydration Challenge | Drink 2L of water daily for 30 days |       ␣
↪    600
 Strength Challenge  | Increase bench press weight by 10kg |       ␣
↪    900
```

Report 9

Select users with more than 2000 calories in their diet plan.

```
SELECT u.name, d.daily_cal
FROM Diet_Plan d
JOIN "User" u ON d.user_id = u.user_id
WHERE d.daily_cal > 2000;
```

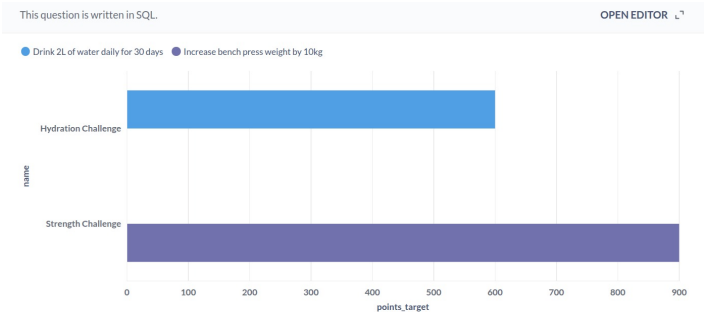

Fig. 8: Report 8 – Visualiztion (Bar)

Result:

```
    name      | daily_cal
--------------+-----------
 Michael Lee  |      2200
```

Report 10

Get average calories for each drink category.

```
SELECT Category, AVG(Calories) AS avg_calories
FROM Drink_Log
GROUP BY Category;
```

Result:

```
 category |       avg_calories
----------+-------------------------
 SODA     |     50.0000000000000000
 WATER    | 0.00000000000000000000
 JUICE    |    120.0000000000000000
```

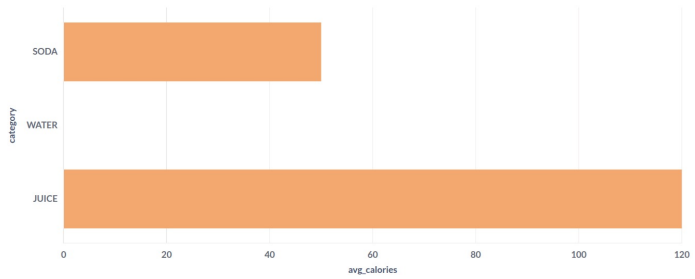

Fig. 9: Report 10 – Visualiztion (Bar)

Report 11

Get the average daily protein, fats, and carbs for each user (Very useful in a dashboard).

```
SELECT u.name, AVG(d.daily_protein) AS avg_protein, AVG(d.daily_
↪fats) AS avg_fats, AVG(d.daily_carbs) AS avg_carbs
```

(continues on next page)

(continued from previous page)

```
FROM "User" u
JOIN Diet_Plan d ON u.user_id = d.user_id
GROUP BY u.name
ORDER BY avg_protein DESC;
```

Result:

```
    name      |      avg_protein       |       avg_fats        |       
↪avg_carbs
--------------+------------------------+-----------------------+---------
↪-------------
 Michael Lee | 160.0000000000000000 | 55.0000000000000000 | 220.
↪0000000000000000
 Emma Wilson | 130.0000000000000000 | 45.0000000000000000 | 190.
↪0000000000000000
 Sarah Kim   | 110.0000000000000000 | 35.0000000000000000 | 170.
↪0000000000000000
```

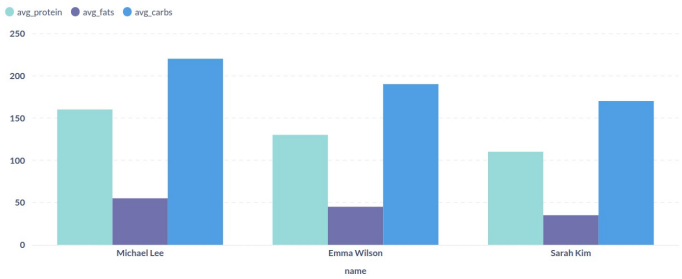

Fig. 10: Report 11 – Visualiztion (Bar)

9 Metabase and Docker

We use a simple `docker-compose` that creates `Metabase` instance to allow visualizing SQL queries, allows dropping database, running it on cross-platforms easily and reliably.

```
services:
  db:
    image: citusdata/citus:13.0.3
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: secret
      POSTGRES_DB: tracker
    volumes:
      - ./sql/init:/docker-entrypoint-initdb.d
  analytics:
    image: metabase/metabase
    ports:
      - "3000:3000"
```