Fabrice Mikobi - 101196480
Ahmed Abdou - 101176667

COMP 3005 Final Project Report

1. Requirements

FlexTrainer is a database application for a Health and Fitness Club Management System that stores and manages information for three types of users: Members, Trainers, and Staff.

Members of our Health and Fitness Club will have access to the following features:

- **Workouts:**

Members should be able to pick from a list of various workouts designed by the trainers and be able to log a workout. Each workout includes multiple exercises that target specific muscle groups. A collection of exercises constitutes a workout that the application provides to members.

- **Workout history:**

Members should be able to view a history of the past workouts they have performed and track their progress over time

- **Fitness goals:**

Each member should be able to choose from a list of pre-selected fitness goals and personalize them to fit their needs. For instance, fitness goals may include weight loss, muscle gain, body decomposition, etc. Each member will have the freedom to specify the metric for each goal. E.g., losing 20 lbs.

- **Group classes:**

The Fitness club offers a variety of group classes such as pilates, zumba, boxing, cardio, high intensity interval training (HIIT), strength and conditioning which will allow members to work out with a supportive community that pushes them to get better. Members should be able to sign up for any class they are interested in

- **Personal training:**

Personal training sessions are similar to group classes except that they are one-on-one with the trainer and the member.

- **Health metrics:**

Members should be able to update their profile and track their health metric over time such as weight and height.

Trainers are the second category of users in this system. Their features include :

- **Personal training:**

Trainers can help members with proper guidance on workouts and keep them accountable to their fitness goals. They should have access to a list of all the members who signed up for training with them.

- **Group classes:**

Each trainer can teach multiple group classes and the database must keep track of that information. However, only one trainer per class is allowed.

- **Class history:**

Trainers should be able to log how many members attended a class and add notes to describe the progress and effort of members and comment on any additional relevant details.

- **Trainer profile:**

Each trainer must provide personal information about themselves such as their name, phone number, birthday and email address. They will also be required to give a short bio that may include details such as a short description, fitness background, expertise, fun fact, etc.

Staff members should access the following features:

- **Transactions:**

Staff members exclusively have access to this table to oversee the club's activity and to ensure monitor the financial activities of the club.

- **Equipment**

The database should allow the staff to manage the equipment, the quantity, the room where it is stored, and when it was purchased.

- **Rooms:**

The database should also keep track of all the rooms in the fitness facility and their capacity.

- **Bookings:**

The staff is in charge of the bookings and make sure that two classes are never booked at the same time and in the same room. The staff should assign a room to every group class or personal training session.

- **Staff profile:**

Each staff member must provide personal information about themselves for safety purposes. The database will store details such as name, birth date, sex, email address, date hired, and phone number. Furthermore, it will also store the staff's salary and update it as necessary.

2. Assumptions and explanations

We have assumed total participation from the N side of one-to-many relationships and partial participation from the 1 side. For instance, a member may not have a workout history but every workout history must reference a member. The same assumption can be made for all the one-to-many relationships listed below. Furthermore, we ensured referential integrity by adding DELETE constraints for all the foreign keys that reference other relations. When a class is deleted from the classes table, all its bookings get cascade deleted as well. Wherever it is necessary to preserve the record, the foreign key is set to null. The following are all the relationships in the database:

One-to-many:

- Member-Transactions
- Member-Workout history
- Member-Fitness goals
- Workouts-Exercises
- Workout history-Workouts
- Trainers-Classes
- Class-Class history
- Trainer-Class history
- Bookings-Classes
- Rooms-Bookings
- Rooms-Equipment

Many-to-many:

- Member-Classes

The staff table is a single entity and does not have any relationship because no additional information is stored about them. However, if an application were implemented for this database, they would be the only users who would have access to the transactions, bookings, and rooms tables.

3. Normalization of relation schema (into 3NF)

Functional Dependencies:

Member
Member_id -> {first_name, last_name, email, phone, birth_date, sex, height, weight, signup_date}

Trainer
Trainer_id -> {first_name, last_name, email, phone, birth_date, sex, hire_date, bio}

Staff
Staff_id -> {first_name, last_name, email, phone, birth_date, sex, hire_date, salary}

Workouts
Workout_id -> {workout_name, duration}

Workout_history
Workout_history_id -> {workout_id, member_id, workout_date}

Exercises
Exercise_id -> {exercise_id, exercise_name, reps, sets, workouts_id}

Rooms
Room_id -> {room_name, capacity}

Bookings
Booking_id -> {class_id, day_of_week, time_of_day, room_id, duration}

Equipment
Equipment_id -> {equipment_name, room_id, quantity, purchase_date}

Transactions
Transaction_id -> {transaction_date, amount, transaction_type, member_id}

Classes
Class_id -> {class_name, class_type, trainer_id}

Class_history
Class_history_id -> {class_history_id, notes, attendance, class_id, trainer_id}

Member_classes
The set of primary key (member_id, class_id) uniquely identify each tuple in this relation, but there is no other attribute and therefore no functional dependencies.

Fitness_goals
Goal_id -> {member_id, goal_name, goal_description, goal_type, target_metric, target_value, start_date, end_date, achieved}

- This table is not in 2NF because the target metric is partially dependent on the goal type. Therefore, we can decompose this table into 2 to achieve 3NF:

****Fitness_goals (updated)**
Goal_id -> {member_id, goal_name, goal_description, goal_type_id, target_value, start_date, end_date, achieved}

****Goal type goals (new)**
Goal_type_id -> {goal_title, target_metric}

- Now, the tables are fully normalized and there is no partial dependency.

Thus, we can conclude that every table in the database is in Third Normal Form because it is in 2NF (i.e., there is no partial dependency) and there is no transitive dependency. The Primary Key of every relation uniquely determines all the non-prime attributes. The tables are already broken down into normalized tables that ensure referential integrity and prevent update anomalies.