Project Report - Health + Fitness Club Management System
COMP3005
Group Members: Alex Bon and Gloria Li

1. Introduction

This project implements a Health and Fitness Club Management System using PostGreSQL and Python. The goal was to design a relational database, implement application operations for 3 user roles (member, trainer, and admin). Then, demonstrate the app using a menu interface.

Our project integrated the following components:

- A fully designed relational schema with constraints
- Sample data covering all entity sets (DML.sql)
- ERD + Relational Mapping + Normalization Documentation
- At least 10 working operations across all roles
- Trigger, View, Index
- A video demonstration of the final system

2. Assumptions

To keep the project focused and aligned with specifications, assumptions were made:

a) Members, trainers, and admins are separate roles, each logged in with an email + password
b) Metrics represent a member's recorded health information (weight, body fat, heart rate) at different times
c) Goals store personal targets for a member and automatically update using a trigger whenever new metrics are added
d) Trainer availability is stored as time ranges (start_time, end_time) and availability slots do not overlap
e) Classes are linked to rooms through room bookings. A class cannot exist without a booking
f) Admin functionality is limited to what the project requires, so billing and payments are not implemented
g) All data shown in the demonstration is sample data loaded from DML.sql

These assumptions guided the ERD structure and the operation in the CLI

3. Entity Sets and Relationships

Regular Entities

- Member
- Metrics
- Goals
- Trainer

- TrainerAvailability
- Admin
- Room
- RoomBooking
- Class
- ClassRegistration

*** All of these are regular bc each has its own primary key + does not depend on another entity's key for identification

We determined that no weak entities were required

Key Relationship

- Member – Metrics: 1-to--Many
- Member – Goals: 1-to-Many
- Trainer–TrainerAvailability: 1 to Many
- Room – RoomBooking: 1 to Many
- RoomBooking – Class: 1-to-1 (each class uses exactly one booked room)
- Member – Class: Many to many
- Trainer – Class: 1 to many

These relationships are represented in our ERD + mapped into the relational schema

4. ER to Relational Mapping Summary

Each entity was mapped into a relational table:

- Member(member_id, fname, lname, email, password, birthday, gender, class_count)
- Metrics(metric_id, member_id(FK), metric_date, weight, body_fat, heart_rate)
- Goals(goal_id, member_id(FK), metric_name, current_metric, goal_metric)
- Trainer(trainer_id, fname, lname, email, password, specialization)
- TrainerAvailability(slot_id, trainer_id(FK), start_time, end_time)
- Admin(admin_id, email, password)
- Room(room_id, room_name, max_capacity)
- RoomBooking(booking_id, room_id(FK), start_time, end_time, purpose)
- Class(class_id, booking_id(FK), trainer_id(FK), attendance)
- ClassRegistration(reg_id, class_id(FK), member_id(FK))

Constraints include:

- FKs enforcing referential integrity
- UNIQUE(member_id, metric_name) to prevent duplicate goals
- CHECK(end_time > start_time) on bookings
- Trigger updating goal progress
- View for available classes
- Index on class attendance (optional but included)

5. Normalization Summary (2NF + 3NF)

We analyzed all relational schemas for normalization

- All tables have single-attribute primary keys, so 2NF is auto satisfied
- No non-key attribute depends on another non-key attribute in any relation, so 3NF is also satisfied

More info is in Normalization.pdf

6. Implemented Application Operations

Member Operations (4+)

- Record new health metrics
- View personal metric history
- View / update personal goals
- Edit profile information
- View dashboard

Trainer Operations (2+)

- Add new availability slots
- View slots
- Look up a member's progress

Admin Operations (2+)

Create room bookings
Create classes linked to trainers and rooms
We exceeded the minimum operations required.

7. Features Using DB Concepts

Trigger: updates goal progress whenever new metrics are inserted
View: available_classes to dynamically compute capacity
Index: supports class attendance queries
Referential Integrity: all tables use appropriate foreign keys

8. Conclusion

This project demonstrates a complete end-to-end implementation of a relational system, from conceptual modelling (ERD) to relational mapping, normalization, SQL schema design, and a working Python application.