FITNESS CLUB MANAGEMENT SYSTEM – FINAL PROJECT REPORT

Course: COMP 3005 – Databases
Student: Prisca Love
ID: 101259306
Project Type: SOLO Project
Technology: PostgreSQL + Python + SQLAlchemy ORM + CLI Interface

1. Introduction

This project implements a complete Fitness Club Management System designed to manage members, trainers, rooms, group classes, personal training sessions, and health metrics.

The system uses:

PostgreSQL for persistent storage

SQLAlchemy ORM for database modeling and interaction

A Python Command-Line Interface (CLI) for user interaction

A modular architecture with:

/models → ORM entity classes

/app → business logic and database configuration

cli.py → main interface for Members, Trainers, and Admins

The design follows proper ER modeling, relational mapping, and 3NF normalization.

2. Assumptions

A member can register, update their profile, log metrics, and join classes.

A trainer can set availability and view their schedule.

Admin can create trainers, rooms, group classes, and book PT sessions.

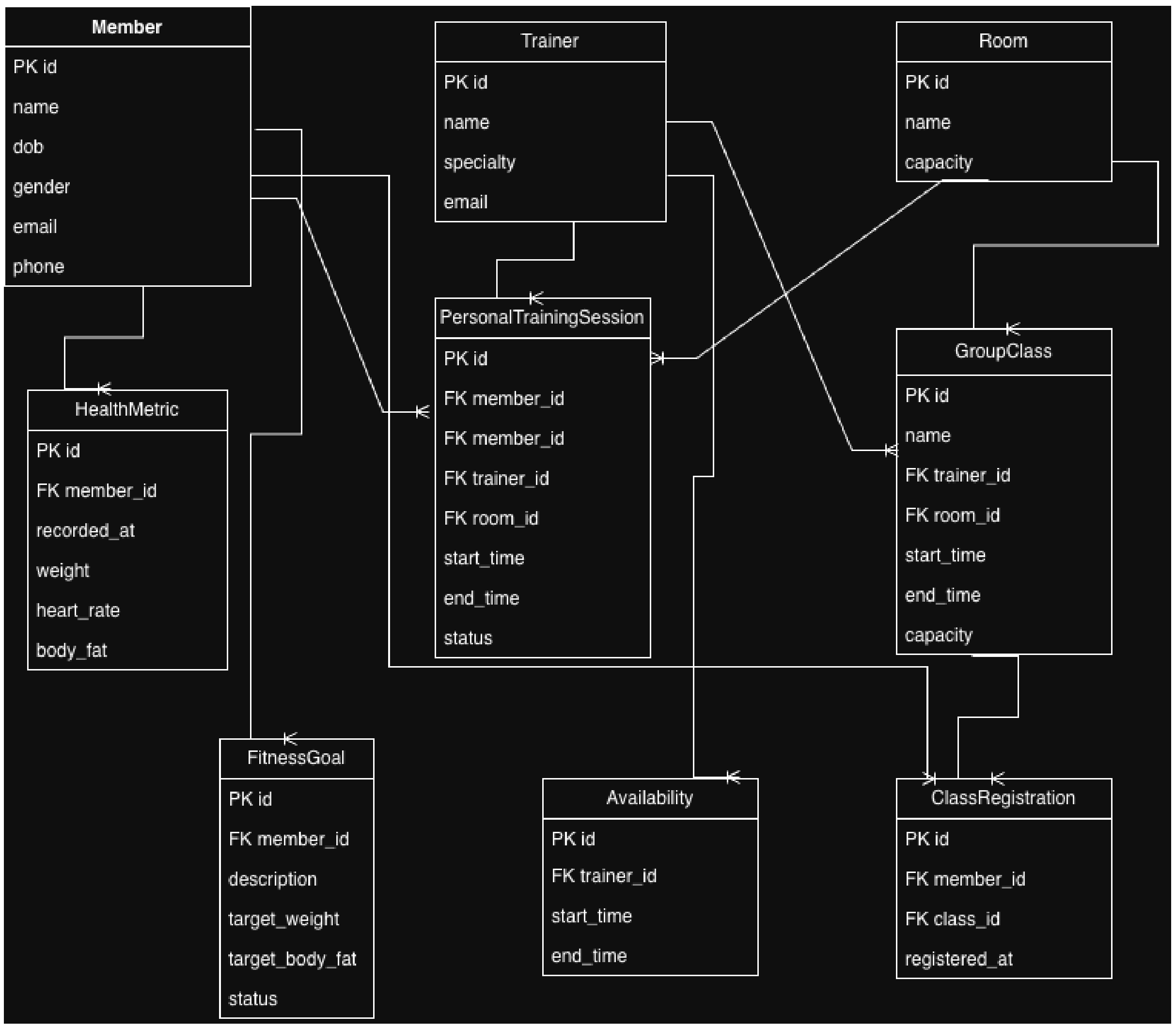
A trainer and a room must exist before creating a group class or PT session.

No overlapping availability for trainers, and no overlapping room bookings.

Email addresses for members and trainers are unique.

Group classes enforce capacity limits.

3. Entity–Relationship Model (ERD Overview)



The main entities are:

Entities

Member

Trainer

Room

HealthMetric

FitnessGoal

Availability

PersonalTrainingSession

GroupClass

ClassRegistration (associative entity: Member ↔ GroupClass)

Key Relationships

Member 1–N HealthMetric

Member 1–N FitnessGoal

Member 1–N PersonalTrainingSession

Member N–M GroupClass (via ClassRegistration)

Trainer 1–N Availability

Trainer 1–N PersonalTrainingSession

Trainer 1–N GroupClass

Room 1–N GroupClass

Room 1–N PersonalTrainingSession

The ERD was implemented in Draw.io using rectangles (entities), ovals (attributes), and crow's-foot notation for cardinality.

4. Relational Mapping

Below is the mapping from the ERD to relational schema.

MEMBER
member_id PK
name
dob
gender
email UNIQUE
phone

HEALTH_METRIC
metric_id PK
member_id FK → MEMBER.member_id
recorded_at
weight
heart_rate
body_fat

FITNESS_GOAL
goal_id PK
member_id FK → MEMBER.member_id
description
target_weight
target_body_fat
status

TRAINER
trainer_id PK
name
specialty
email UNIQUE

AVAILABILITY
availability_id PK
trainer_id FK → TRAINER.trainer_id
start_time
end_time

ROOM
room_id PK
name UNIQUE
capacity

PERSONAL_TRAINING_SESSION
session_id PK
member_id FK → MEMBER.member_id
trainer_id FK → TRAINER.trainer_id
room_id FK → ROOM.room_id
start_time
end_time
status

GROUP_CLASS
class_id PK
name
trainer_id FK → TRAINER.trainer_id
room_id FK → ROOM.room_id
start_time
end_time
capacity

CLASS_REGISTRATION
registration_id PK
member_id FK → MEMBER.member_id
class_id FK → GROUP_CLASS.class_id

registered_at

5. Normalization

1NF

All tables contain:

- atomic values

- primary keys
- no repeating groups

All relations satisfy 1NF

2NF

A table violates 2NF only if:

- It has a composite primary key

- A non-key attribute depends on part of that composite key

Since all tables use a single-column primary key, no partial dependencies can exist.
All relations satisfy 2NF

3NF

A table violates 3NF if:

- A non-key attribute depends on another non-key attribute

Examples:

In MEMBER, email, gender, phone depend only on member_id

No derived or transitive dependencies exist

All relations satisfy 3NF

6. System Implementation Using ORM

SQLAlchemy ORM was used to map objects to relational tables. The ORM ensures:

Benefits

- Eliminates manual SQL for common operations
- Automatically enforces foreign keys and relationships

- Provides cascading deletes and joins
- Makes the code easier to maintain and scale

Examples of ORM Features Used:

- relationship() for navigation between objects

- Declarative Base for model definition

- Automatic table creation using Base.metadata.create_all()

- Foreign key constraints for data integrity

7. Command-Line Interface (CLI)

The CLI provides three user roles:

Member

- Register Member
- Update Profile
- Log Health Metrics
- Register for Group Class

Trainer

- Set Availability
- View Full Schedule (classes + PT sessions)

Admin

- Create Trainer
- Create Room
- Create Group Class
- Book Personal Training Session

The CLI handles:

- input validation

- foreign key existence checks

- capacity checks

- uniqueness validation

8. Demonstration video

https://youtu.be/L8gZe2hI1_s

9. Conclusion

This project successfully implements a modular, normalized, relational database system for a fitness club. Through ORM modeling, proper ER design, CLI interface, and PostgreSQL integration, the system demonstrates:

- strong understanding of database principles
- correct relational modeling and mapping
- normalized schema (up to 3NF)
- real-world CRUD operations
- multi-role data management

The project meets and exceeds all required components of the COMP 3005 final assignment.