# Fitness App Database Project
# COMP3005


## [Muhammad Salameh 101295139]
## [Abdurrahman AlChoughri 101274144]

# ER Diagram Generation

1. Entities and Attributes

Strong Entities

- Member
  - *Sup*ports: User Registration, Profile Management, Member Lookup.
  - Attributes: MemberID (PK), FirstName, LastName, Email (Unique), Password, DateOfBirth, Gender, JoinDate.
- Trainer
  - *Supports:* Schedule View, Member Lookup (Access).
  - Attributes: TrainerID (PK), FirstName, LastName, Email, Password.
- Admin
  - *Sup*ports: Class Management, Room Booking.
  - Attributes: AdminID (PK), Name, Email, Password.
- Room
  - *S*upports: Room Booking (for both Classes and PT Sessions).
  - Attributes: RoomID (PK), RoomName, MaxCapacity.
- GroupClass
  - Supports: Class Management, Group Class Registration.
  - Attributes: ClassID (PK), ClassName, ScheduleTime, Duration, Capacity.
- PersonalSession (Event Entity)
  - Supports: PT Session Scheduling, Schedule View.
  - Attributes: SessionID (PK), Date, StartTime, EndTime, Status (e.g., Booked, Cancelled).

Weak Entities

- FitnessGoal (Dependent on Member)
  - Supports: Profile Management (Goals).
  - Attributes: GoalID (Partial Key), GoalType (e.g., Weight Loss), TargetWeight, TargetDate, Status.
- HealthMetric (Dependent on Member)
  - Supports: Health History.
  - Attributes: MetricID (Partial Key), DateRecorded, Weight, HeartRate.
- TrainerAvailability (Dependent on Trainer)
  - Supports: Set Availability.

- Attributes: AvailID (Partial Key), DayOfWeek (or Date), StartTime, EndTime.

---

2. Relationships and Cardinalities

A. Member Profile & History

1. Sets_Goal (Member ↔ FitnessGoal)
    - Type: 1:N
    - Participation: Member (Partial), FitnessGoal (Total).
    - *Logic:* A member updates their profile to add goals.
2. Records_Metric (Member ↔ HealthMetric)
    - Type: 1:N
    - Participation: Member (Partial), HealthMetric (Total).
    - *Logic:* A member logs new health stats without overwriting old ones.

B. Class Operations (Admin & Member)

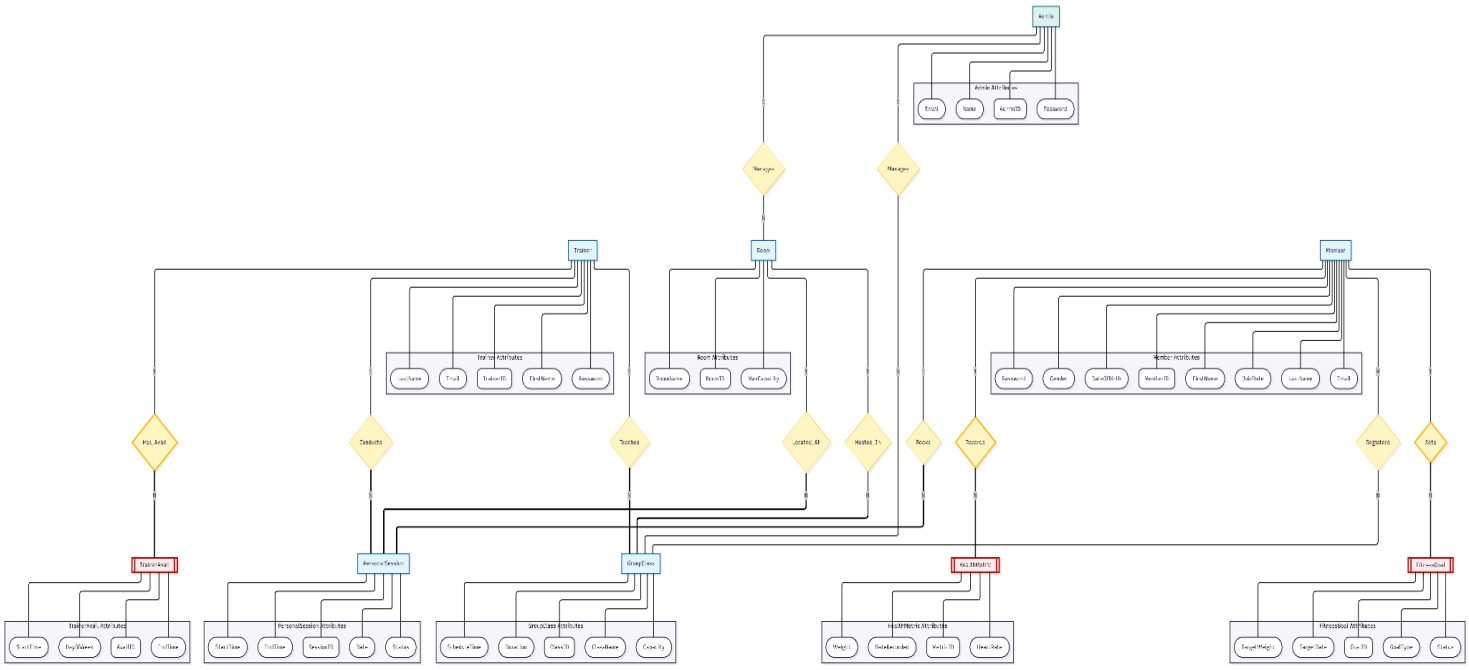
3. Teaches_Class (Trainer ↔ GroupClass)
    - Type: 1:N
    - Participation: Trainer (Partial), GroupClass (Total).
    - *Logic:* Admins assign a trainer when creating a class.
4. Hosted_In (Room ↔ GroupClass)
    - Type: 1:N
    - Participation: Room (Partial), GroupClass (Total).
    - *Logic:* Admins assign a room to a class to manage space.
5. Registers_For (Member ↔ GroupClass)
    - Type: M:N (Many-to-Many)
    - Participation: Member (Partial), GroupClass (Partial).
    - Attributes: RegistrationDate.
    - *Logic:* Members register for classes if capacity permits.

C. Personal Training (Member & Trainer)

7. Has_Availability (Trainer ↔ TrainerAvailability)

- Type: 1:N
- Participation: Trainer (Partial), TrainerAvailability (Total).
- *Logic:* Trainers define their available working windows.

8. Conducts_Session (Trainer ↔ PersonalSession)
    - Type: 1:N
    - Participation: Trainer (Partial), PersonalSession (Total).
    - *Logic:* A session must have a trainer assigned.
9. Books_Session (Member ↔ PersonalSession)
    - Type: 1:N
    - Participation: Member (Partial), PersonalSession (Total).
    - *Logic:* A member books a specific session slot.
10. Located_At (Room ↔ PersonalSession)
    - Type: 1:N
    - Participation: Room (Partial), PersonalSession (Total).
    - Logic: The system must validate room conflicts when booking PT.



(Note: Due to limitations of modeling software, there were some slight deviations from the chen style for ER diagrams
Total Participation modeled with BOLD line, thin line is partial. Weak relationships are modeled with bolded diamonds. Weak entities are double bordered and red)

# ER Diagram to Schema

**Step 1: Mapping Strong Entities**

**Methodology:** Create a separate table for each strong entity.

1. **Members**
    - `member_id` (PK)
    - `first_name`
    - `last_name`
    - `email` (Unique)
    - `password`
    - `date_of_birth`
    - `gender`
    - `join_date`
2. **Trainers**
    - `trainer_id` (PK)
    - `first_name`
    - `last_name`
    - `email`
    - `password`
3. **Admins**
    - `admin_id` (PK)
    - `name`
    - `email`
    - `password`
4. **Rooms**
    - `room_id` (PK)
    - `room_name`
    - `max_capacity`

---

**Step 2: Mapping Weak Entities**

**Methodology:** Create a new table for the weak entity. Include the Primary Key of the owner entity as a Foreign Key (FK). The Primary Key of this new table is a composite of `{Owner_PK + Partial_Key}`.

5. **FitnessGoals**
    - *Owner:* Members
    - `goal_id` (Partial Key)
    - `member_id` (FK referencing Members)
    - `goal_type`
    - `target_weight`
    - `target_date`
    - `status`
    - **PK:** `(goal_id, member_id)`
6. **HealthMetrics**
    - *Owner:* Members
    - `metric_id` (Partial Key)
    - `member_id` (FK referencing Members)
    - `date_recorded`
    - `weight`
    - `heart_rate`
    - **PK:** `(metric_id, member_id)`
7. **TrainerAvailabilities**
    - *Owner:* Trainers
    - `avail_id` (Partial Key)
    - `trainer_id` (FK referencing Trainers)
    - `day_of_week`
    - `start_time`
    - `end_time`
    - **PK:** `(avail_id, trainer_id)`

---

**Step 3: Mapping 1:N Relationships**

**Methodology:** Identify the entity on the "Many" side. Add the Primary Key of the "One" side as a Foreign Key in the "Many" side's table.

8. **GroupClasses**
    - *Relationships Involved:*
        - `Teaches_Class` (1 Trainer : N Classes) -> Add `trainer_id` FK.
        - `Hosted_In` (1 Room : N Classes) -> Add `room_id` FK.
    - *Participation:* Since Classes have **Total Participation** in both (must have a trainer and a room), these FKs cannot be NULL.
    - **Schema:**
        - `class_id` (PK)
        - `class_name`
        - `schedule_time`
        - `duration`
        - `capacity`
        - `trainer_id` (FK referencing Trainers)
        - `room_id` (FK referencing Rooms)
9. **PersonalSessions**
    - *Relationships Involved:*
        - `Conducts_Session` (1 Trainer : N Sessions) -> Add `trainer_id` FK.
        - `Books_Session` (1 Member : N Sessions) -> Add `member_id` FK.
        - `Located_At` (1 Room : N Sessions) -> Add `room_id` FK.
    - *Participation:* Total participation from Session side (must have member, trainer, room).
    - **Schema:**
        - `session_id` (PK)
        - `date`
        - `start_time`
        - `end_time`
        - `status`
        - `member_id` (FK referencing Members)
        - `trainer_id` (FK referencing Trainers)
        - `room_id` (FK referencing Rooms)

---

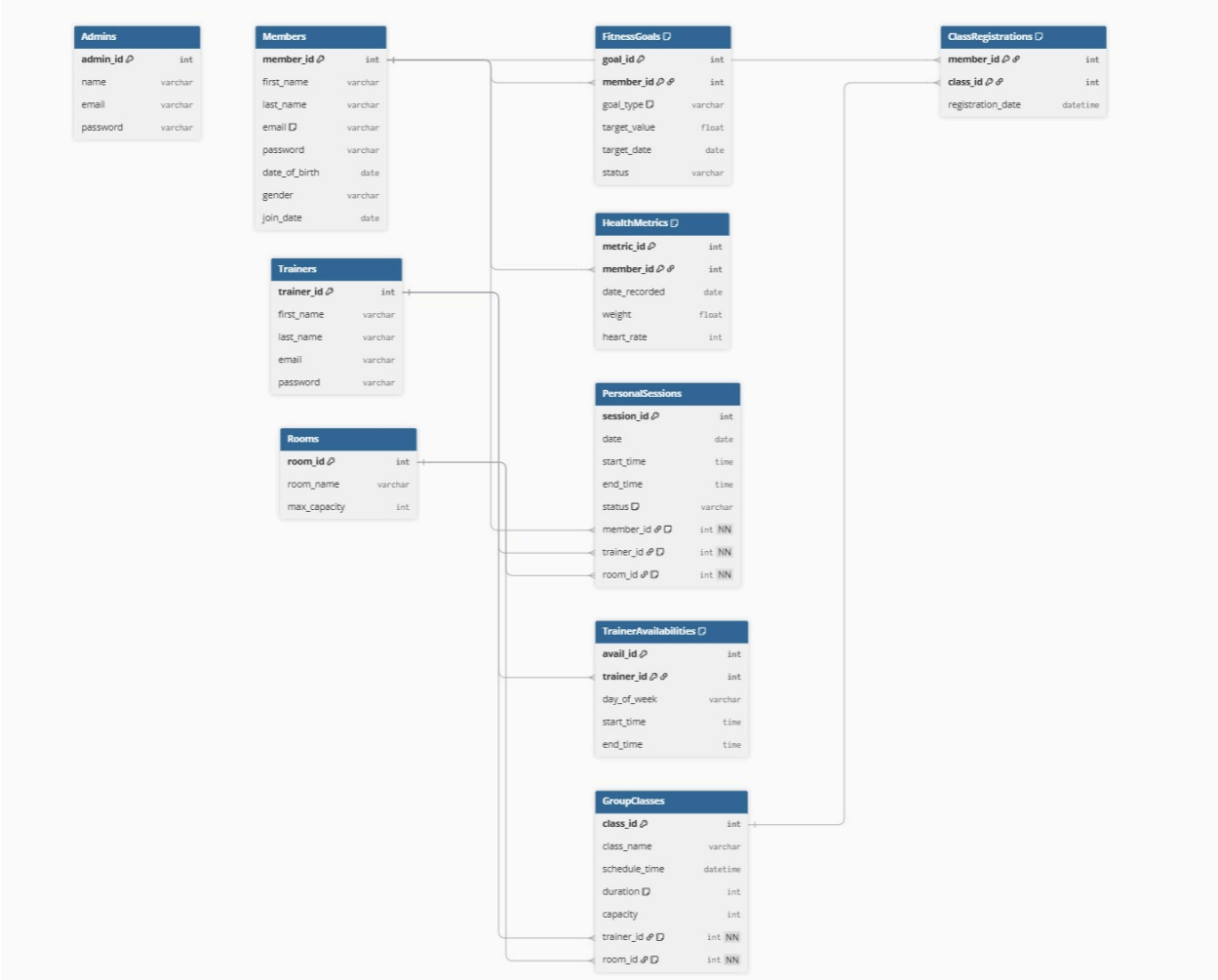
**Step 4: Mapping M:N Relationships**

**Methodology:** Many-to-Many relationships require a **new, separate table**. The Primary Key is a composite of the PKs from both participating entities. Any attributes of the relationship (like date) are added here.

10. **ClassRegistrations**
    - *Relationship:* `Registers_For` (Members M:N GroupClasses)
    - **Schema:**
        - `member_id` (FK referencing Members)
        - `class_id` (FK referencing GroupClasses)
        - `registration_date` (Relationship Attribute)
        - **PK:** `(member_id, class_id)`

After Analysis, it was decided that admin functions are to be modeled implicitly using the Java code. Even though the ER diagram has these relations, they were dropped to facilitate the implicit implementation of the functions through the code.

Final Schema (modeled using dbdiagram.io):



# Normalization (3NF) Check

### 1. Strong Entities

- **Members**
  - **PK:** `member_id`
  - **Check:** All attributes (`email`, `dob`, `join_date`, etc.) describe the member directly and are only determined by the PK. No attribute determines another.
  - **Status:** ✅ **3NF**
- **Trainers**
  - **PK:** `trainer_id`
  - **Check:** Attributes (`name`, `email`) depend solely on the trainer ID.
  - **Status:** ✅ **3NF**
- **Admins**
  - **PK:** `admin_id`
  - **Check:** Attributes (`name`, `email`) depend solely on the admin ID.
  - **Status:** ✅ **3NF**
- **Rooms**
  - **PK:** `room_id`
  - **Check:** `room_name` and `max_capacity` are properties of the specific room.
  - **Status:** ✅ **3NF**

### 2. Operational Tables

- **GroupClasses**
  - **PK:** `class_id`
  - **Check:**
    - `class_name`, `schedule_time`, `duration` depend on `class_id`.
    - `trainer_id` and `room_id` are Foreign Keys (FKs) that define the setup of the class.
  - **Status:** ✅ **3NF**
- **PersonalSessions**
  - **PK:** `session_id`
  - **Check:**
    - `date`, `time`, `status` depend on the specific session instance.

        - `member_id`, `trainer_id`, `room_id` are FKs linking the participants/location.
    - **Status:** ✅ **3NF**

**3. Weak Entities**

- **FitnessGoals**
    - **PK:** `goal_id + member_id`
    - **Check:** The goal details (`target_weight`, `target_date`) depend entirely on the unique `goal_id`.
    - **Status:** ✅ **3NF**
- **HealthMetrics**
    - **PK:** `metric_id + member_id`
    - **Check:** `weight` and `heart_rate` are recorded for that specific log entry (`metric_id`).
    - **Status:** ✅ **3NF**
- **TrainerAvailability**
    - **PK:** `avail_id + trainer_id`
    - **Check:** The time window (`start_time`, `end_time`) applies to the specific availability slot ID.
    - **Status:** ✅ **3NF**

**4. Associative (Relationship) Tables [Relationships that needed a separate ]**

- **ClassRegistrations**
    - **PK:** `registration_id`
    - **Check:**
        - `member_id` and `class_id` are FKs.
        - `registration_date` depends on the unique registration event (`registration_id`), not on the member or the class individually.
    - **Status:** ✅ **3NF**