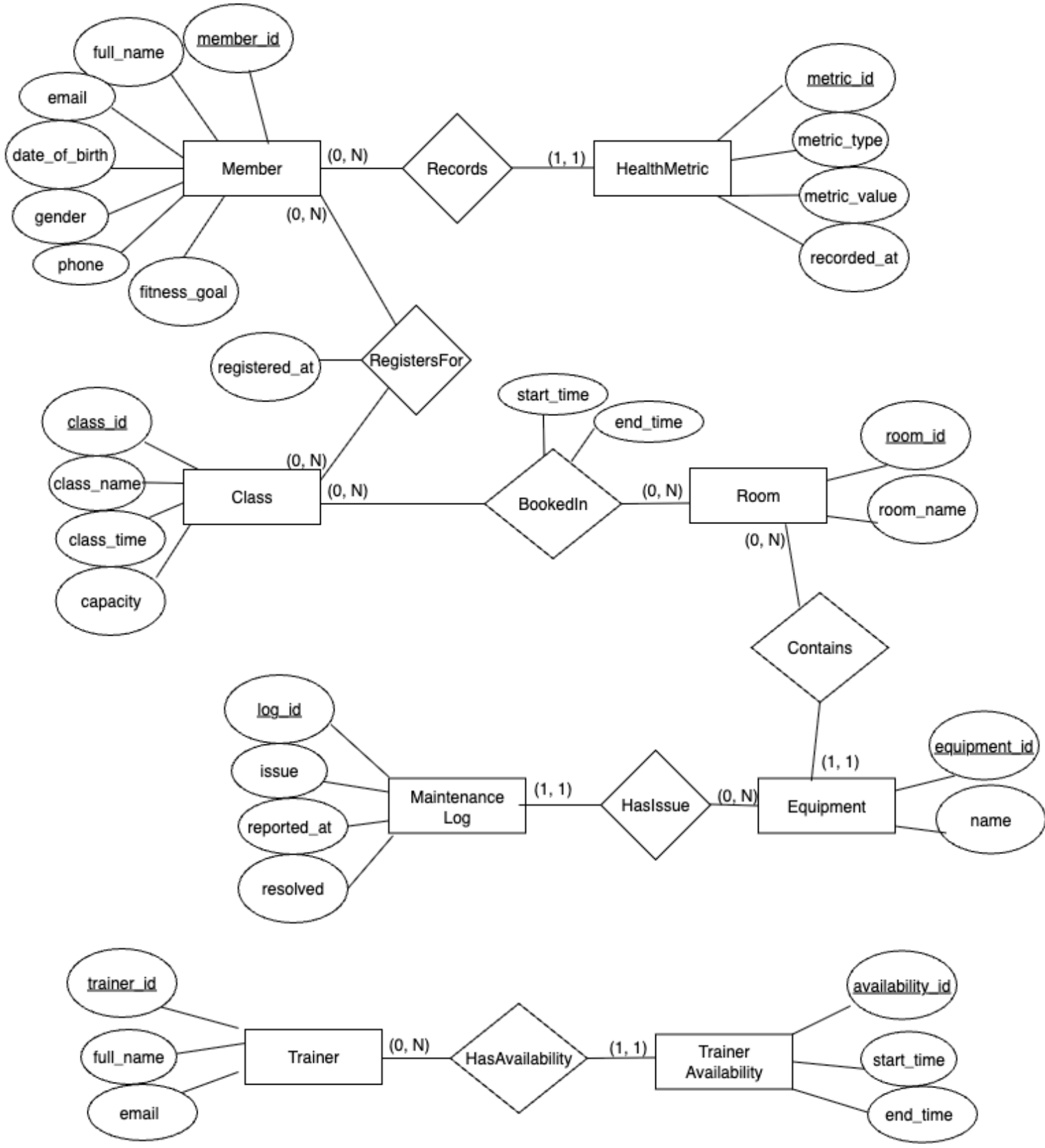
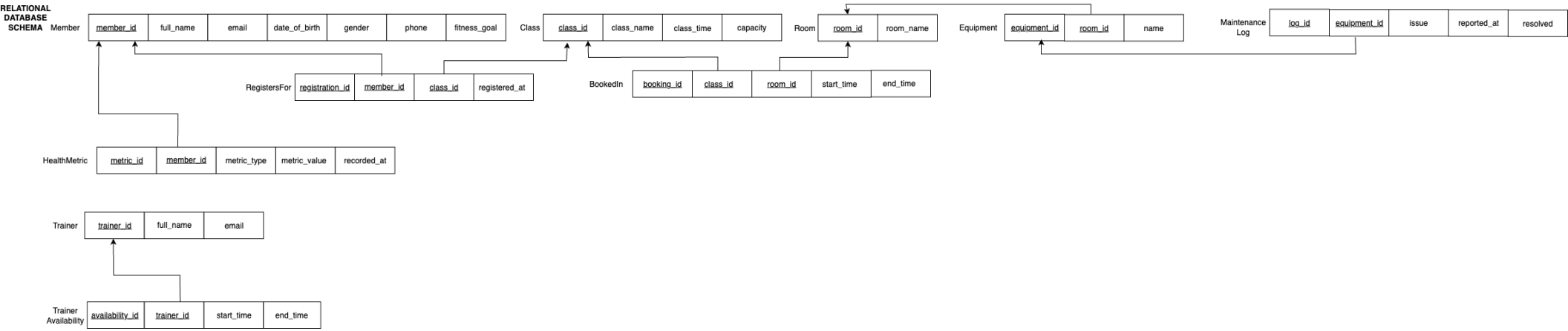
ER MODEL


full_name
member_id
email
date_of_birth
Member
gender
phone
fitness_goal
(0, N)
Records
(1, 1)
HealthMetric
metric_id
metric_type
metric_value
recorded_at
(0, N)
RegistersFor
registered_at
class_id
class_name
class_time
capacity
Class
(0, N)
(0, N)
start_time
end_time
BookedIn
(0, N)
Room
room_id
room_name
(0, N)
Contains
log_id
issue
reported_at
resolved
Maintenance Log
(1, 1)
HasIssue
(0, N)
Equipment
(1, 1)
equipment_id
name
trainer_id
full_name
email
Trainer
(0, N)
HasAvailability
(1, 1)
Trainer Availability
availability_id
start_time
end_time


RELATIONAL DATABASE SCHEMA
Member
member_id
full_name
email
date_of_birth
gender
phone
fitness_goal
Class
class_id
class_name
class_time
capacity
Room
room_id
room_name
Equipment
equipment_id
room_id
name
Maintenance Log
log_id
equipment_id
issue
reported_at
resolved
RegistersFor
registration_id
member_id
class_id
registered_at
BookedIn
booking_id
class_id
room_id
start_time
end_time
HealthMetric
metric_id
member_id
metric_type
metric_value
recorded_at
Trainer
trainer_id
full_name
email
Trainer Availability
availability_id
trainer_id
start_time
end_time

Mapping Table

Requirement	Assumption	Representation in ER Model/ Relational DB schema
A member should be able to register by providing personal information such as name, date of birth, gender, and contact details, and later manage or update these details as needed.	Member profile data is stored in a single member entity with unique emails.	Member Entity: Attributes include member_id(PK), full_name, email (unique), date_of_birth, gender, phone Schema: members(member_id PK, full_name, email, date_of_birth, gender, phone, fitness_goal)
Members will have the ability to establish and track personalized fitness goals—such as achieving a target body weight or reducing body fat percentage—and store health metrics like height, weight, heart rate, and other measurable indicators of physical performance. These metrics should not overwrite previous entries but instead be recorded historically so that trends and progress can be analyzed over time.	Each metric entry is stored separately and linked to the member	Health_Metrics Entity: Attributes include metric_id (PK), member_id(FK), metric_type, metric_value, recorded_at. 1-to-Many: Member → Health Metrics Schema: health_metrics(metric_id PK, member_id FK → members, metric_type, metric_value, recorded_at)
Members should be able… to register for group fitness classes, subject to class capacity and schedule constraints.	A member can join many classes, and a class can have many members	Class Entity: Attributes include class_id PK, class_name, class_time, capacity) + Class_Registrations

		relationship: registration_id (PK), member_id, class_id, registered_at. Unique(member_id, class_id) Schema: class_registrations(egistration_id PK, member_id FK, class_id FK)
Trainers should be able to specify their availability periods	Availability stored as individual intervals	Trainer Availability Entity: Attributes include availability_id (PK), start_time, end_time. 1-to-Many: Trainer → Availability slots Constraint: end_time > start_time. Schema: trainer_availability(availability_id PK, trainer_id FK, start_time, end_time)
Trainers must have controlled access to member profiles to review relevant information such as health metrics, progress toward goals.	Trainers can read member data but not update it	No relationship needed. Looking up is handled at an application level action using queries to Member + Health Metrics.
Administrators will have the capability to manage room bookings to ensure that physical spaces—such as studios or training rooms—are properly	Rooms are separate entities, bookings reference both rooms and classes	Room Bookings relationship: Attributes include Booking_id(PK), room_id, class_id, start_time, end_time. Schema:

allocated for classes and personal sessions, while avoiding conflicts in scheduling.		booked_in(booking_id PK, room_id FK, class_id FK, start_time, end_time)
Administrators should also be able to oversee equipment management, including tracking the operational status of machines, logging maintenance issues, assigning repair tasks, and updating maintenance records once issues are resolved.	Equipment may optionally be assigned to a room. Maintenance logs track issues until resolved	Equipment Entity: Attributes include equipment_id(PK), name, room_id (FK). Schema: equipment(equipment_id PK, name, room_id FK) + MaintenanceLog Entity: Attributes include log_id (PK), equipment_id, issue, reported_at, resolved Relationship: Equipment 1-to-many MaintenanceLog Schema: maintenance_logs(log_id PK, equipment_id FK, issue, reported_at, resolved)

Normalization Justification

The Health and Fitness Club database is structured so that all tables follow the principles of 3NF. Each table uses a single-attribute primary key (eg, member_id, class_id, or metric_id), ensuring every record is uniquely identifiable. All non-key attributes in each table depend directly on the primary key and nothing else. For example, in the health_metrics table, attributes like metric_type, metric_value, and recorded_at depend solely on metric_id, meaning no partial dependencies. The design also prevents redundancy by placing repeating or multi-valued information into separate tables. Member health measurements are stored in health_metric, class registrations in class_registrations, trainer schedules in trainer_availability, and equipment issues in maintenance_logs. This prevents duplicate information and keeps data consistent.