Gym Membership & Workout Tracking System - Database Report

1. Introduction

1.1 Project Overview

The Gym Membership & Workout Tracking System is designed to efficiently manage gym operations, including member registrations, trainer schedules, workout tracking, and payment processing. The database ensures streamlined operations, data integrity, and insightful analytics for decision-making.

1.2 Objectives

- Provide a structured system to manage memberships, payments, and trainer assignments.
- Track workout logs, class attendance, and trainer-member relationships.
- Automate billing and payment tracking.
- Ensure data consistency and optimize performance.

2. Business/Organization Description

2.1 Purpose of the Database

The gym database system is designed to enhance operational efficiency by automating key processes. Its primary objectives are:

- Automate Gym Membership Management: Ensure smooth onboarding of new members, track membership renewals, and monitor active subscriptions.
- Track Workout Progress and Trainer Assignments: Maintain records of individual workouts, trainer-member relationships, and customized training plans.
- Improve Customer Experience: Offer personalized workout suggestions, enable members to book sessions with trainers, and keep track of progress reports.

Done by: Alekhya Nalla, Aryan Puranik, Manish Shah

- Enhance Data-Driven Decision Making: Generate detailed reports on gym performance, such as revenue trends, popular workout programs, and peak gym usage times.
- Optimize Resource Allocation: Efficiently manage gym facilities, schedule classes, and allocate trainers based on demand.

2.2 Functionalities Provided by the Database

The database system incorporates several functionalities to enhance the overall management of gym operations. These include:

1. Member Management

- Maintain comprehensive member profiles with personal details and contact information.
- Track active/inactive members, ensuring timely follow-ups for renewals.
- Associate members with specific membership plans, ensuring accurate billing.
- Store and update members' workout logs, progress reports, and attendance history.

2. Trainer Management

- Assign trainers to members based on specialization and availability.
- Maintain trainer schedules, ensuring efficient class and session allocation.
- Track trainer performance and member feedback to optimize service quality.
- Manage trainer availability, allowing members to book sessions online.

3. Class & Workout Tracking

- Maintain an interactive class schedule that tracks member participation.
- Store detailed workout logs, tracking calories burned, session duration, and workout difficulty levels.
- Enable automated reminders for scheduled classes and personal training sessions.
- Analyze popular workout programs to optimize gym services.

4. Payment & Invoicing

- Manage all financial transactions, including membership fees and personal training payments.
- Generate invoices and receipts, ensuring proper financial records.
- Enable multiple payment methods (credit card, cash, online payment portals).
- Implement an automated billing system for seamless subscription renewals.

5. Attendance Tracking

- Monitor check-ins and check-outs for the members.
- Generate reports on peak hours to optimize gym resources.
- Track attendance for group classes, ensuring proper capacity management.

6. Feedback & Review System

- Allow members to rate trainers and classes based on their experience.
- Collect valuable feedback to improve services and optimize workout programs.
- Store detailed review data to help trainers enhance their training approaches.

By integrating these functionalities, the gym ensures efficient management of operations, improved customer satisfaction, and data-driven decision-making for long-term growth.

3. Conceptual Design - Enhanced Entity-Relationship (EER) Model

3.1 Overview of the EER Model

The Enhanced Entity-Relationship (EER) Model provides a detailed representation of how different entities in the gym database interact with each other. It extends the traditional ER model by incorporating specialization, generalization, weak entities, multivalued attributes, and complex relationships.

3.2 Key Components of the EER Model

- **Entities and Attributes**: Representation of different elements in the gym system.
- **Relationships**: Defines how entities interact (e.g., Member **attends** Class, Trainer **supervises** Members).
- **Cardinality Constraints**: Specifies the **number of occurrences** between relationships.
- **Generalization & Specialization**: Groups entities into **supertypes and subtypes**.
- **Weak Entities**: Entities dependent on a **strong entity** for identification.
- **Multivalued & Derived Attributes**: Attributes that can **store multiple values** or be **computed**.

3.3 Key Entities & Attributes

Core Entities

Member (M) - Represents gym members who have active or inactive subscriptions.

- member_id (PK): Unique identifier for each member.
- name: Full name of the member.
- age: Age of the member.
- gender: Member's gender.
- contact_info: Email or phone number for communication.
- membership_type (FK → Membership_Plan): Links the member to a specific membership plan.
- registration_date: Date the member joined the gym.
- status: Indicates whether the member is active, inactive, or suspended.

Trainer (T) - Represents trainers who conduct classes and provide personal training sessions.

- trainer_id (PK): Unique identifier for each trainer.
- name: Full name of the trainer.
- specialization: Trainer's area of expertise (e.g., strength training, cardio, yoga).
- experience: Years of experience in the fitness industry.
- contact_info: Email or phone number for communication.
- availability: Days and times when the trainer is available.

Membership Plan (MP) - Defines different membership tiers offered by the gym.

- membership_type (PK): Unique identifier for each membership type.
- name: Name of the membership plan (e.g., Basic, Premium, VIP).
- duration: Length of the membership (e.g., 1 month, 6 months, 1 year).
- cost: Price of the membership.
- features: List of benefits included in the plan (e.g., free classes, personal training sessions).

Workout Program (WP) - Predefined workout plans available for members.

- WP_id (PK): Unique identifier for each workout program.
- T_id (FK): Foreign key referencing the trainer who created or leads the workout program.
- M_id (FK): Foreign key referencing the member assigned to or following the workout program.
- WP_date: The date on which the workout program was scheduled or assigned.

- WP_duration: Duration of the workout program in minutes.
- WP_name: Name of the workout program
- Time_in_gym: Total hours a member spent in the gym per month.

Operational Entities

Class (C) - Manages group fitness classes.

- class_id (PK): Unique identifier for each class.
- name: Name of the class.
- member_id (FK → Member): Members attending the class.
- trainer_id (FK → Trainer): Trainer conducting the class.
- date: Date the class is held.
- time: Start time of the class.
- location: Venue or room number for the class.

Attendance (A) - Tracks attendance for classes and training sessions.

- member_id (FK → Member): Member checking in.
- class_id (FK → Class_Schedule): Class being attended.
- trainer_id (FK → Trainer): Trainer overseeing the session.
- check_in_time: Time the member checked in.
- check_out_time: Time the member checked out.

Financial Entities

Payment (P) - Records payments made by members.

- payment_id (PK): Unique identifier for each payment.
- member_id (FK → Member): Member making the payment.
- amount: Payment amount.
- date: Date the payment was made.
- method: Payment method (e.g., credit card, cash, online transfer).

Invoice (I) - Stores billing details.

- invoice_id (PK): Unique identifier for each invoice.
- member_id (FK → Member): Member billed.
- amount: Invoice amount.
- due_date: Payment deadline.

- status: Payment status (e.g., Paid, Pending, Overdue).

Administrative Entities

User Account (UA) - Manages login credentials.

- member_id (FK → Member): Links account to a member.
- trainer_id (FK → Trainer): Links account to a trainer.
- username: Unique login identifier.
- password_hash: Encrypted password.
- role: Defines if the user is a member, trainer, or admin.

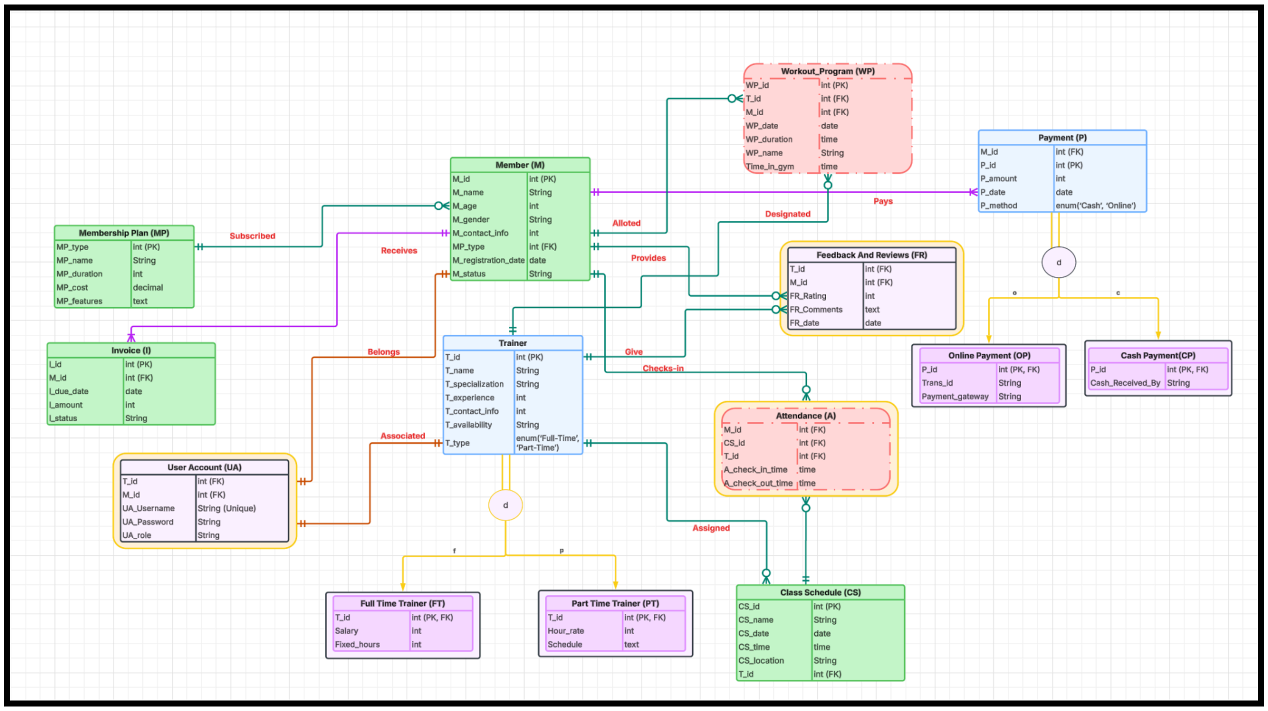
Feedback & Reviews (FR) - Collects feedback from members.

- member_id (FK → Member): Member providing feedback.
- trainer_id (FK → Trainer): Trainer being reviewed.
- rating: Score given to the trainer.
- comments: Member's feedback.
- date: Date of the review.

3.4 Entity Relationships & Cardinality

Entity 1	Entity 2	Relationship Type
Member ↔ Membership Plan	1:M	One plan can be assigned to many members
Member ↔ Payment	1:M	A member can make multiple payments
Member ↔ Invoice	1:M	A member can have multiple invoices
Member ↔ Class Schedule	M:N	A member can attend multiple classes

Member ↔ Feedback & Reviews	1:M	A member can submit multiple reviews
Trainer ↔ Trainer Assignment	1:M	A trainer can be assigned multiple members
Trainer ↔ Class Schedule	1:M	A trainer can conduct multiple classes
Trainer ↔ Feedback & Reviews	1:M	A trainer can receive multiple reviews
Class Schedule ↔ Attendance	1:M	A class schedule can have multiple attendance logs
User Account ↔ Member	1:1	A user account belongs to a member
User Account ↔ Trainer	1:1	A user account belongs to a trainer



4. Relational Model

The **relational model** for the **Gym Management System** was carefully designed by transforming the **Enhanced Entity-Relationship (EER) model** into a structured database schema. This transformation ensures that all relationships are properly mapped, foreign key constraints are enforced, and specialization is handled efficiently. The model maintains **referential integrity, minimizes redundancy, and enhances query efficiency**, making it scalable and easy to manage for real-world applications.

Key Design Considerations

1. Data Integrity and Normalization

The relational model follows the principles of **Third Normal Form (3NF)** to maintain data consistency and eliminate redundancy. By ensuring that each table contains only relevant data and using **foreign keys for relationships**, the design prevents anomalies during insertions, updates, and deletions.

- **Primary keys (PK)** uniquely identify each record, ensuring no duplicate entries.
- **Foreign keys (FK)** establish strong **referential integrity** between related entities.
- **Constraints like NOT NULL and CHECK** ensure data validity in critical fields, such as enforcing a **rating range of 1 to 5** for feedback or allowing only **specific payment methods**.

2. Specialization and Subtypes

To improve **data organization and efficiency**, certain entities were specialized into subtypes:

- The **Trainer** entity was divided into Full-Time Trainer and Part-Time Trainer to reflect different work structures.
- The **Payment** entity was further specialized into Online_Payment and Cash_Payment, allowing for **granular tracking of transactions** based on payment type.

Both specializations use **primary key inheritance**, ensuring that each subtype retains the core attributes of its parent entity while introducing **unique attributes relevant to its type**.

3. Relationship Mapping and Optimization

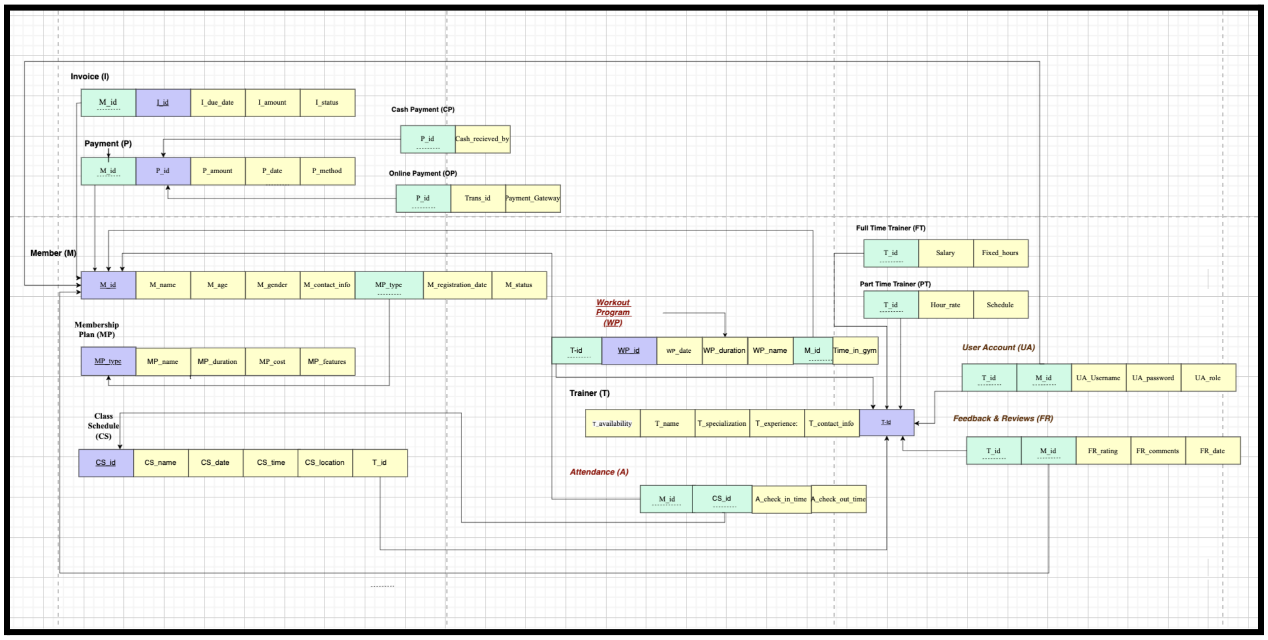
The relational model effectively maps all relationships from the EER diagram into **foreign key constraints and associative tables** to maintain efficiency:

One-to-Many (1:M) Relationships

- A **member** can make multiple **payments**, but each payment belongs to only one member.
- A **trainer** can conduct multiple **classes**, but each class is assigned to only one trainer.
- A **member** can submit multiple **feedback entries**, but each feedback entry belongs to only one member.

Many-to-Many (M:N) Relationships

- Members and trainers have a **many-to-many relationship**, managed via the **Trainer_Assignment** table. This allows **multiple trainers to be assigned to a member and vice versa**.
- Members and class schedules have an **M:N relationship**, handled using the **Attendance** table, which tracks check-ins and check-outs for each session.

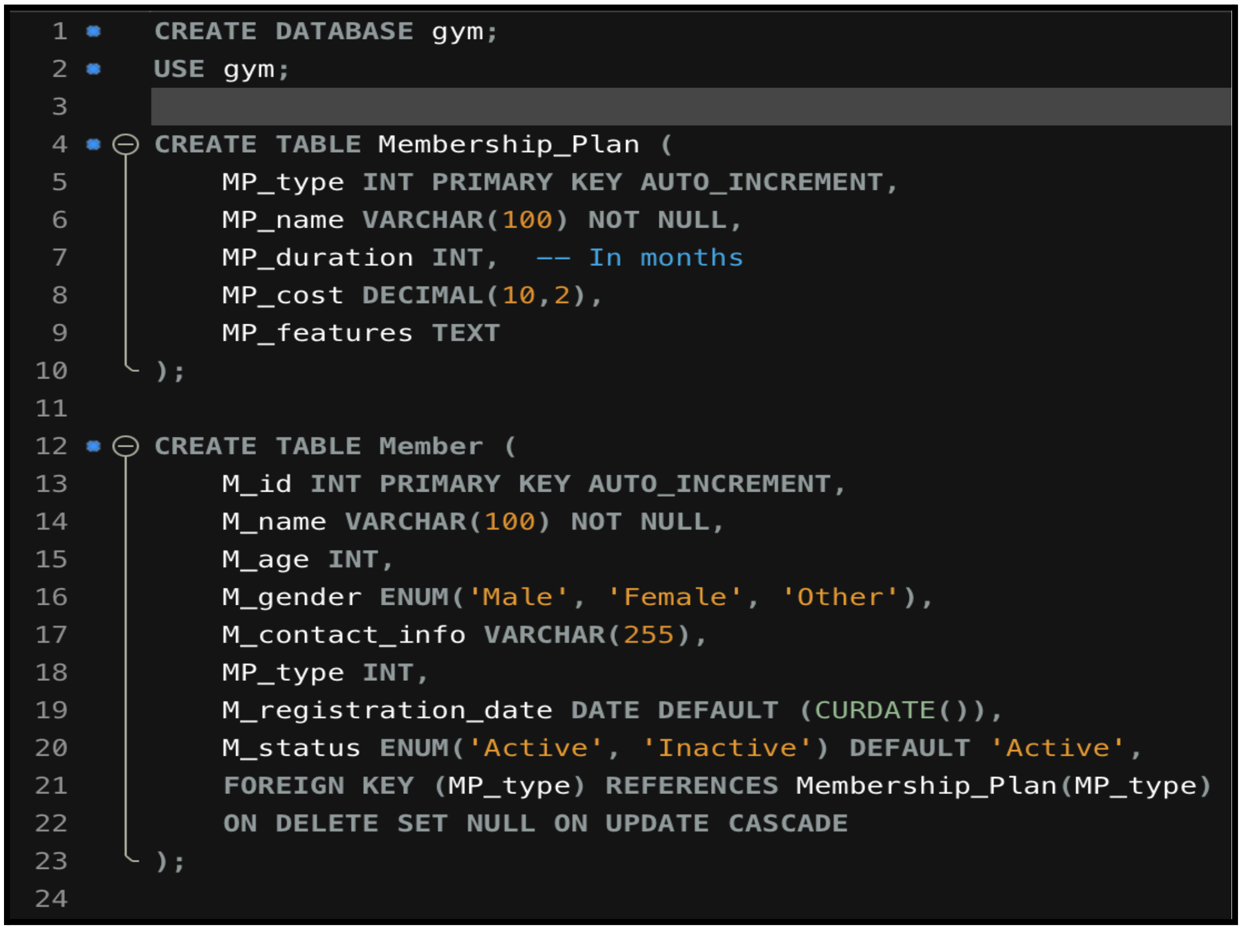


5. Implementation & Application

The implementation phase of the **Gym Management System** involved setting up the database schema, inserting sample data, executing key queries, and addressing challenges to ensure a fully functional system. This section outlines the steps taken to bring the relational model to life in **MySQL Workbench**, along with considerations for data integrity, query optimization, and system validation.

5.1 Database Schema Implementation

The relational schema was implemented using **structured SQL commands**, ensuring that all relationships and constraints were correctly defined. The schema follows best practices by applying **referential integrity, foreign key constraints, and cascade operations** to manage data dependencies efficiently.



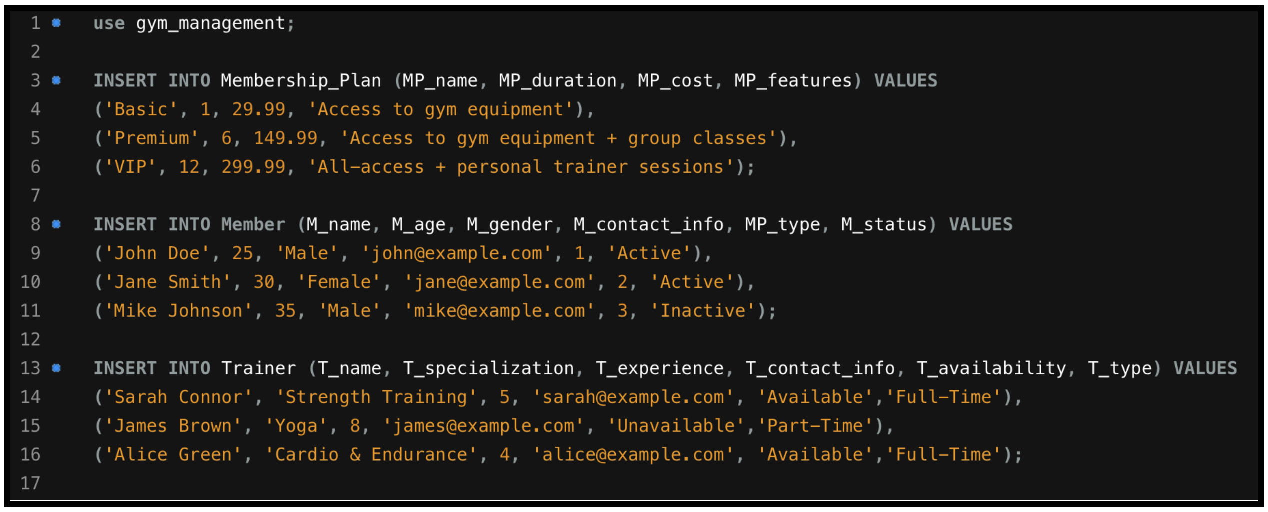
```sql
CREATE DATABASE gym;
USE gym;

CREATE TABLE Membership_Plan (
    MP_type INT PRIMARY KEY AUTO_INCREMENT,
    MP_name VARCHAR(100) NOT NULL,
    MP_duration INT,  -- In months
    MP_cost DECIMAL(10,2),
    MP_features TEXT
);

CREATE TABLE Member (
    M_id INT PRIMARY KEY AUTO_INCREMENT,
    M_name VARCHAR(100) NOT NULL,
    M_age INT,
    M_gender ENUM('Male', 'Female', 'Other'),
    M_contact_info VARCHAR(255),
    MP_type INT,
    M_registration_date DATE DEFAULT (CURDATE()),
    M_status ENUM('Active', 'Inactive') DEFAULT 'Active',
    FOREIGN KEY (MP_type) REFERENCES Membership_Plan(MP_type)
    ON DELETE SET NULL ON UPDATE CASCADE
);

```

5.2. Sample Data Population

To validate the functionality of the database, sample data was inserted into all tables, covering different scenarios such as:

- Active vs. Inactive members
- Different membership plans (Basic, Premium, VIP)
- Trainers with varying experience levels and specializations

- Diverse payment methods (Cash, Online)
- Workout logs tracking different types of exercises

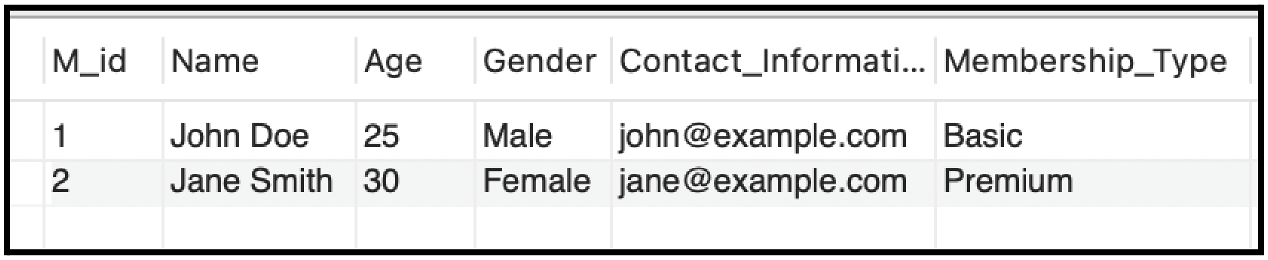


```sql
use gym_management;

INSERT INTO Membership_Plan (MP_name, MP_duration, MP_cost, MP_features) VALUES
('Basic', 1, 29.99, 'Access to gym equipment'),
('Premium', 6, 149.99, 'Access to gym equipment + group classes'),
('VIP', 12, 299.99, 'All-access + personal trainer sessions');

INSERT INTO Member (M_name, M_age, M_gender, M_contact_info, MP_type, M_status) VALUES
('John Doe', 25, 'Male', 'john@example.com', 1, 'Active'),
('Jane Smith', 30, 'Female', 'jane@example.com', 2, 'Active'),
('Mike Johnson', 35, 'Male', 'mike@example.com', 3, 'Inactive');

INSERT INTO Trainer (T_name, T_specialization, T_experience, T_contact_info, T_availability, T_type) VALUES
('Sarah Connor', 'Strength Training', 5, 'sarah@example.com', 'Available','Full-Time'),
('James Brown', 'Yoga', 8, 'james@example.com', 'Unavailable','Part-Time'),
('Alice Green', 'Cardio & Endurance', 4, 'alice@example.com', 'Available','Full-Time');

```

5.3. Query Execution for Business Operations

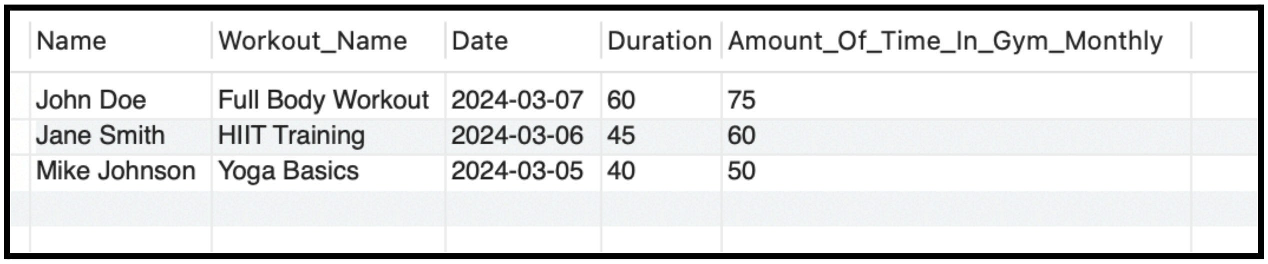
To test the system's functionality, a set of SQL queries were executed to retrieve **critical business insights**. These queries validated data integrity and ensured that relationships between tables were properly established.

1) Retrieve All Active Members



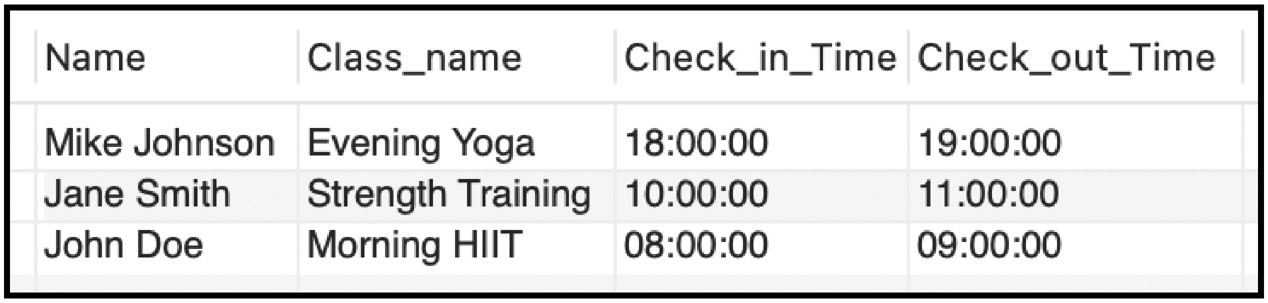
M_id	Name	Age	Gender	Contact_Informati...	Membership_Type
1	John Doe	25	Male	john@example.com	Basic
2	Jane Smith	30	Female	jane@example.com	Premium

2) Track Workout Progress of Each Member



Name	Workout_Name	Date	Duration	Amount_Of_Time_In_Gym_Monthly
John Doe	Full Body Workout	2024-03-07	60	75
Jane Smith	HIIT Training	2024-03-06	45	60
Mike Johnson	Yoga Basics	2024-03-05	40	50

3) Monitor Attendance (Class Check-Ins)



Name	Class_name	Check_in_Time	Check_out_Time
Mike Johnson	Evening Yoga	18:00:00	19:00:00
Jane Smith	Strength Training	10:00:00	11:00:00
John Doe	Morning HIIT	08:00:00	09:00:00

6. UI Development with Streamlit

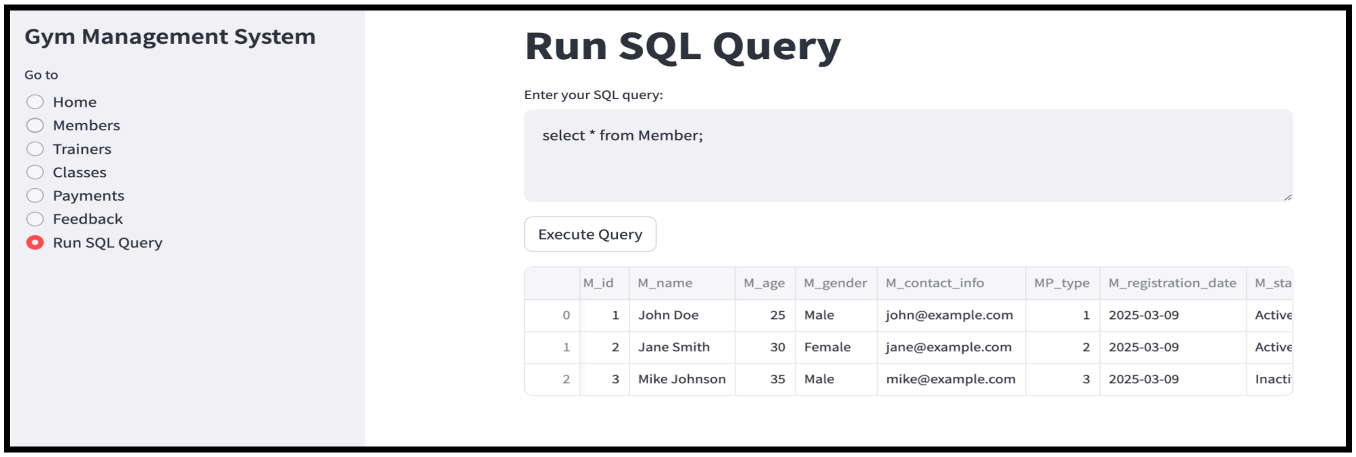
To enhance usability, a **Graphical User Interface (GUI)** was developed using **Streamlit**, allowing users to interact with the database more intuitively. The UI provides:

Done by: Alekhya Nalla, Aryan Puranik, Manish Shah

- **Member Management**
- **Trainer & Class Scheduling**
- **Payments & Invoices**
- **Feedback Collection**
- **Custom SQL Query Execution**

This UI integration bridges the gap between the **backend database** and **end-users**, ensuring a seamless experience for gym administrators and members.

The below is the UI we implemented:



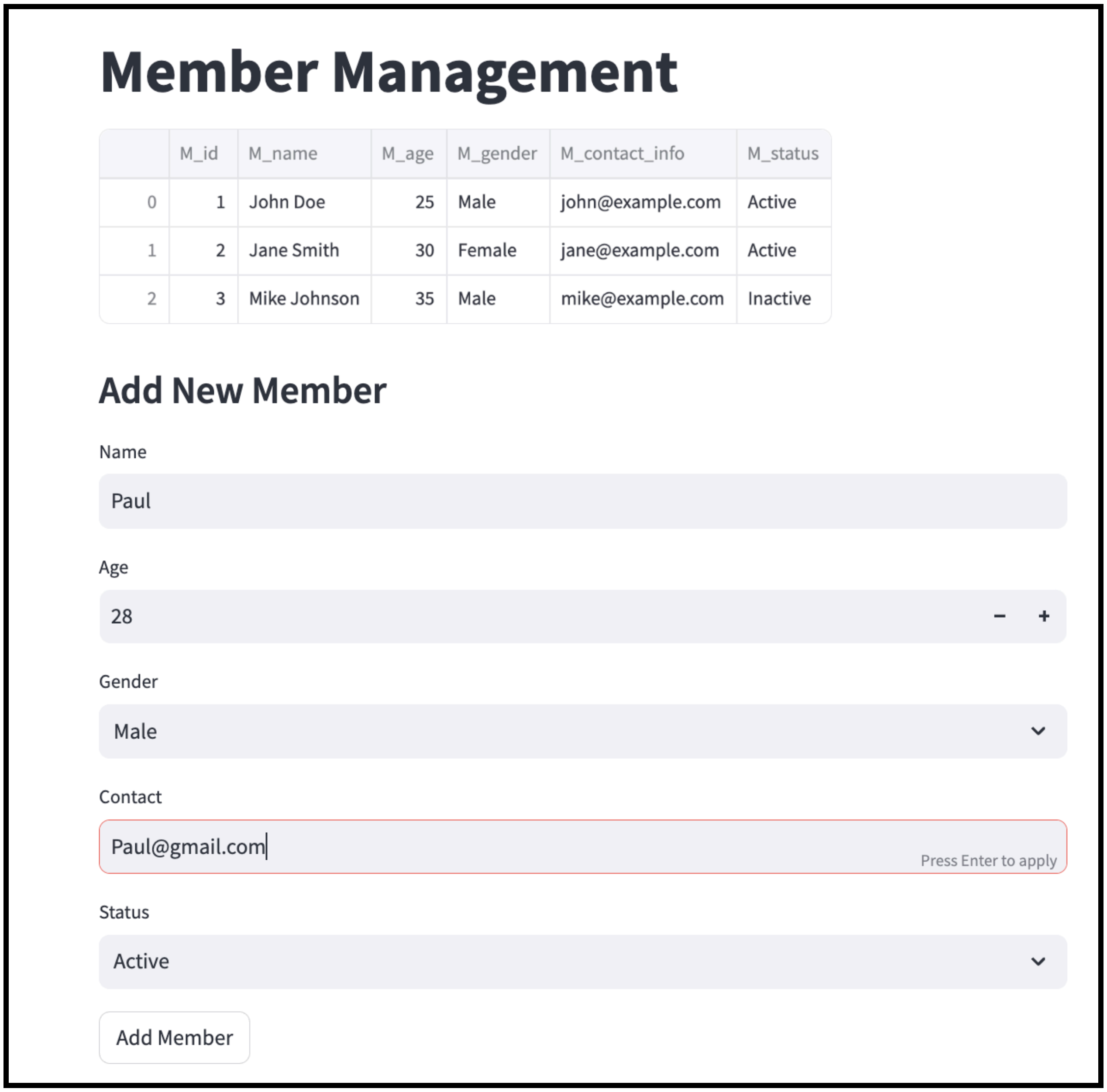
Run SQL Query

Enter your SQL query:

```
select * from Member;
```

	M_id	M_name	M_age	M_gender	M_contact_info	MP_type	M_registration_date	M_sta
0	1	John Doe	25	Male	john@example.com	1	2025-03-09	Active
1	2	Jane Smith	30	Female	jane@example.com	2	2025-03-09	Active
2	3	Mike Johnson	35	Male	mike@example.com	3	2025-03-09	Inacti

Done by: Alekhya Nalla, Aryan Puranik, Manish Shah



Member Management

	M_id	M_name	M_age	M_gender	M_contact_info	M_status
0	1	John Doe	25	Male	john@example.com	Active
1	2	Jane Smith	30	Female	jane@example.com	Active
2	3	Mike Johnson	35	Male	mike@example.com	Inactive

Add New Member

Name

Paul

Age

28

Gender

Male

Contact

Paul@gmail.com

Status

Active

Done by: Alekhya Nalla, Aryan Puranik, Manish Shah