CN5000 Group Coursework Report

Group Members

Hard Joshi	*ID: 12345678*
Jayrup Nakawala	*ID: 87654321*
Shyam Jagani	*ID: 11223344*
Yogi Patel	*ID: 44332211*

Table of contents

1 Entities and ERD (10 Marks)

1.1 Identification of Potential Entities

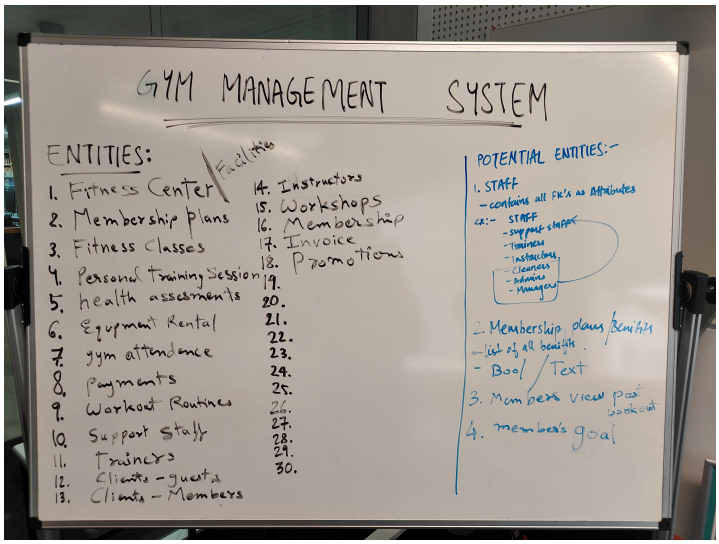


Figure 1: Potential Entities

During the initial phase of the design, we identified **17 entities** as shown in the brainstorming process. These entities were:

1. Fitness Center
2. Membership Plans
3. Fitness Classes
4. Personal Training Sessions
5. Health Assessments
6. Equipment Rentals
7. Gym Attendance
8. Payments
9. Workout Routines
10. Support Staff
11. Trainers
12. Clients - Guests
13. Clients - Members
14. Instructors
15. Workshops
16. Membership
17. Invoice

1.2 Refinement of Entities

The process of refinement involved carefully analyzing the identified entities to eliminate redundancies, merge similar entities, and ensure that only relevant entities were retained to represent the system requirements effectively. The following steps were taken during this refinement process:

1. **Combining Overlapping Entities**:
    - We combined *Clients - Guests* and *Clients - Members* into a single entity **Clients** to avoid duplication and simplify the management of client data.
    - We merged *Support Staff*, *Trainers*, and *Instructors* into a single entity **Staff**. This unified representation simplified the model while allowing roles to be distinguished through attributes.
2. **Eliminating Redundant Entities**:
    - *Fitness Center* was removed as it represented a general concept rather than a specific data entity.
    - *Workshops* were treated as a subset of **Fitness_Classes** or **Personal_Training_Sessions**, depending on the context, and therefore were eliminated.
    - *Payments* and *Invoice* were consolidated into **Billing**, which effectively captured all payment-related activities and attributes.
3. **Creating New Relationships**:
    - To ensure the system captured bookings for classes, we introduced a new entity **Class_Bookings**. This entity connected **Clients** and **Fitness_Classes**, representing many-to-many relationships.
    - We introduced **Discounts** to manage promotional offers and incentives, ensuring flexibility within the billing process.
4. **Streamlining Entity Scope**:
    - We refined entities to focus on core data management processes within the Gym Management System. Entities such as *Workout Routines* were excluded as they could be treated as attributes or separate functionalities within **Personal_Training_Sessions**.

The final streamlined list of **12 entities** was as follows:

1. **Clients**
2. **Membership**
3. **Fitness_Classes**
4. **Class_Bookings**
5. **Personal_Training_Sessions**
6. **Health_Assessments**

7. **Gym_Attendance**
8. **Billing**
9. **Discounts**
10. **Staff**
11. **Facilities**
12. **Equipment_Rentals**

This refined set of entities ensured clarity, reduced redundancy, and effectively supported the required system functionalities.

1.3 Appropriate Naming for Entities

We ensured appropriate and consistent naming conventions for all entities to maintain clarity, ease of understanding, and alignment with database design principles. Entities were represented using singular nouns and underscores to separate words where necessary (e.g., **Fitness_Classes**, **Class_Bookings**). These naming conventions aligned with standard database design practices.

1.4 Entity-Relationship Diagram (ERD)

We created the Entity-Relationship Diagram (ERD) to represent the relationships among the final 12 entities. Key design considerations included:

- **Identification of Major Entities**: We retained all major entities critical to the system in the final design.
- **Weak Entities**: We eliminated weak or redundant entities (e.g., "Support Staff" and "Trainers" merged into **Staff**) to simplify the design.
- **Relationships**: We clearly defined the relationships between entities to ensure a logical flow of data within the system.
- **Multiplicity and Optionality**: We annotated each relationship to indicate the multiplicity (one-to-one, one-to-many) and optionality (mandatory/optional) between entities. For example:
  - A **Client** had one or more **Memberships**.
  - **Fitness_Classes** were booked by multiple **Clients** through **Class_Bookings**.
  - **Billing** records were associated with **Clients** and reflected applicable **Discounts**.

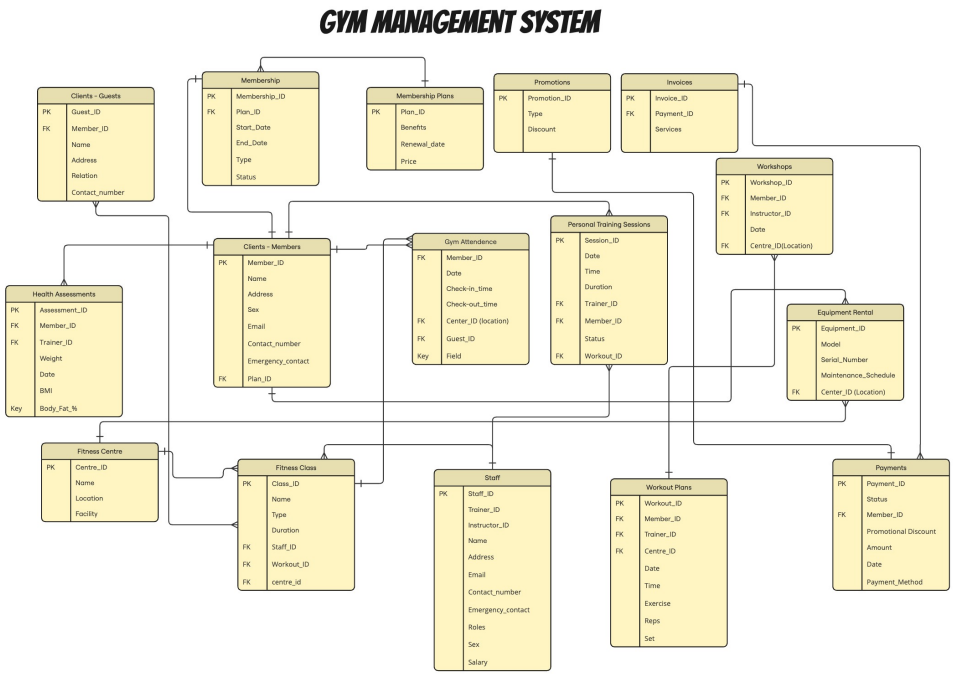


Figure 2: Denormalized ERD

The ERD visually illustrated these entities, their attributes, and the relationships that facilitated data integrity and functionality within the **Gym Management System**. We ensured precision in the design using CASE tools to comply with database modeling standards.

1.5 UseCase Diagram

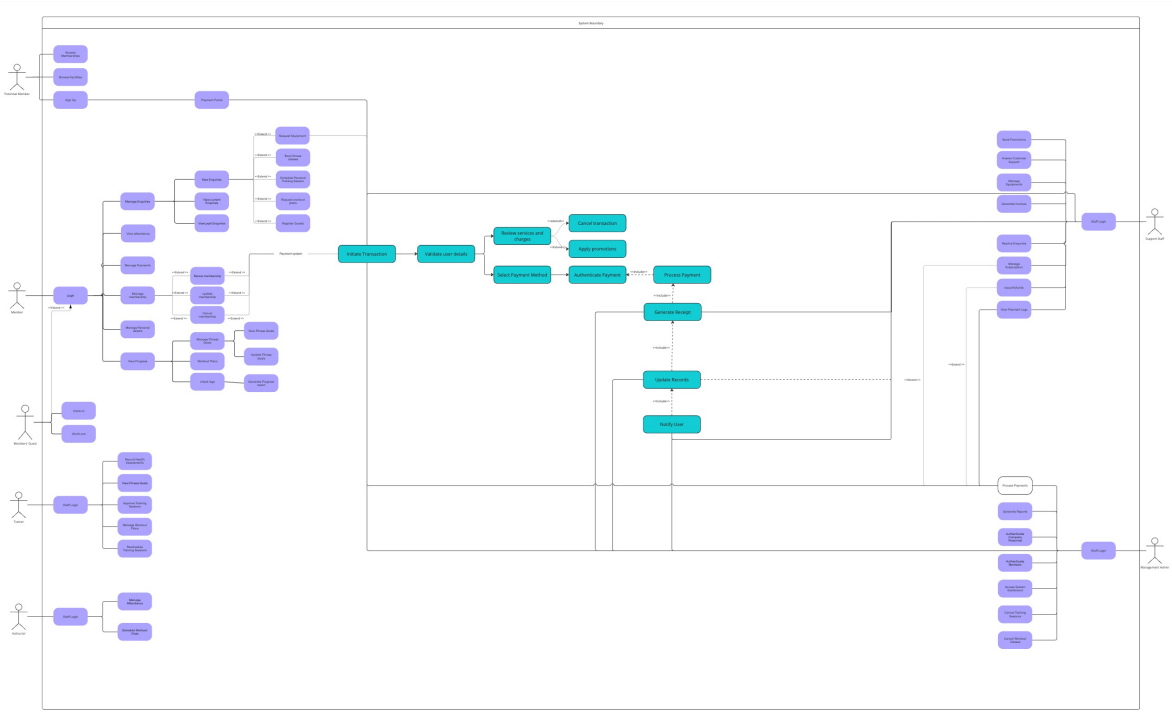


Figure 3: Use Case Diagram

To access the full Use Case Diagram, click here

2 Normalization Process

Normalization is a key process in relational database design that we followed to reduce data redundancy and improve data integrity. It involved organizing tables and attributes into well-defined structures that minimized duplication and eliminated undesirable anomalies (insert, update, and delete anomalies).

The normalization process followed three key stages:

1. **First Normal Form (1NF)**:
    - We ensured all attributes were atomic, i.e., each attribute contained only indivisible values.
    - We removed repeating groups or arrays within tables.
    - Example: In a table with Client and Fitness_Class details, classes attended by a client were represented as separate rows rather than a single row with multiple class values.
2. **Second Normal Form (2NF)**:
    - We ensured that all non-key attributes were fully functionally dependent on the primary key.
    - We removed partial dependencies, where non-key attributes depended on only part of a composite primary key.
    - Example: In a table with Class_Bookings, attributes such as Client_Name were not depending only on Class_ID but rather on the composite key **(Client_ID, Class_ID)**.
3. **Third Normal Form (3NF)**:
    - We ensured that all attributes were only dependent on the primary key and not on other non-key attributes.
    - We removed transitive dependencies, where non-key attributes depended on other non-key attributes.
    - Example: In a Billing table, if Discount_Percentage was depending on Discount_ID, this dependency was moved to a separate **Discounts** table.

2.1 Achieving 3NF in the Gym Management System

We ensured that all tables in the Gym Management System adhered to **Third Normal Form (3NF)**. The steps followed included:

- Ensuring all attributes were atomic (1NF).
- Eliminating partial dependencies (2NF) by clearly defining primary keys.

- Removing transitive dependencies (3NF) to ensure attributes only depended on the primary key. For instance:
  - The **Billing** table referenced **Discount_ID** instead of directly storing discount details.
  - **Class_Bookings** linked **Clients** and **Fitness_Classes** without duplicating client or class data.

By adhering to these normalization principles, we ensured:

- Minimal data redundancy.
- Data integrity through consistent and well-structured relationships.
- Efficient query performance and database maintenance.

2.2 ERD Diagram

The final ERD, which visually depicted the relationships and multiplicity, was provided as part of the **Appendices** section. It showed: - Entities with primary and foreign keys. - Relationships between tables. - Multiplicity and constraints clearly defined.

This comprehensive approach ensured that the database design was normalized, logically structured, and aligned with the requirements of the Gym Management System.

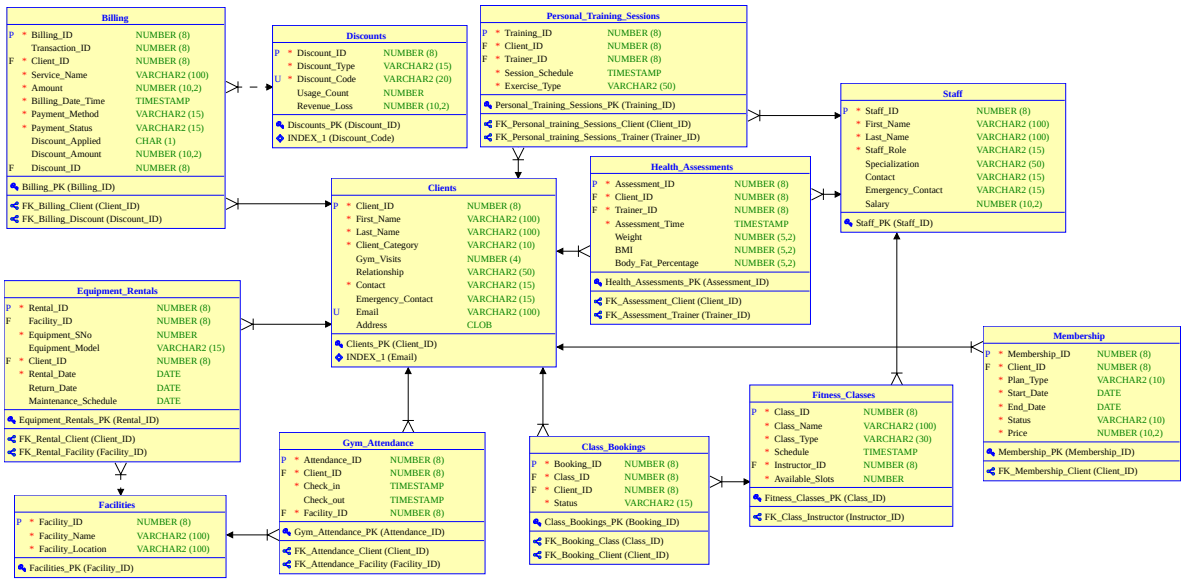


Figure 4: ERD Diagram

---

3 Creation of Tables in SQL and Population of Tables (10 Marks)

3.1 Creating Database in SQL

- **Correct SQL Syntax**:
  - Data types, primary key, foreign key, not null, unique, and check constraints where appropriate.
- **Evidence of Successful Execution**.
- **At Least 10 Records** in each table.

3.2 SQL Queries

- **Correct SQL Syntax**.
- **Evidence of Successful Execution**.

3.3 Stored Procedures and Triggers

- **Correct SQL Syntax** to perform defined actions.
- **Evidence of Successful Execution**.
- Partial attempts can gain marks if effort is evident.

---

4 Report and Presentation (30 Marks)

4.1 Report Reflection (5 Marks)

- **Principles of Database Design**:
  - Application to the case study.
  - Recognition of alternative solutions.
  - Justification of design choices.
- **Discussion**:
  - Well-written and logically coherent.
  - Evaluates design decisions and alternatives.

4.2 Presentation (20 Marks)

- **Design and Implementation in Oracle**.
  - 10-minute presentation.
  - 5-minute Q&A session.

4.3 Gantt Chart (5 Marks)

- **Milestones and Group Involvement**.
- Documentation stating group member participation.

---

5 Sql Queries (15 Marks)

5.1 Get a list of all fitness classes offered at the gym, along with their schedules and the instructor names. (1 mark)

```sql
SELECT
    FC.CLASS_TYPE AS CLASS_LIST,
    S.FIRST_NAME || ' ' || S.LAST_NAME AS INSTRUCTOR_NAME,
    LISTAGG(TO_CHAR(FC.SCHEDULE, 'DD-MON-YYYY HH24:MI'), ', ')
        WITHIN GROUP (ORDER BY FC.SCHEDULE) AS SCHEDULE
FROM FITNESS_CLASSES FC
JOIN STAFF S ON FC.INSTRUCTOR_ID = S.STAFF_ID
GROUP BY FC.CLASS_TYPE, S.FIRST_NAME, S.LAST_NAME
ORDER BY CLASS_LIST;
```

Figure 5: Query 1

5.2 Display all members who have booked a yoga class, including their booking status (Confirmed,Canceled, etc.). (1 mark)

```sql
SELECT
    SUM(CASE WHEN SERVICE_NAME = 'Membership' THEN AMOUNT ELSE 0 END) AS MEMBERSHIP_REVENUE,
    SUM(CASE WHEN SERVICE_NAME = 'Class Booking' THEN AMOUNT ELSE 0 END) AS CLASS_REVENUE,
    SUM(CASE WHEN SERVICE_NAME = 'Personal Training' THEN AMOUNT ELSE 0 END) AS
PERSONAL_TRAINING_SESSION_REVENUE,
    SUM(CASE WHEN SERVICE_NAME IN ('Membership', 'Class Booking', 'Personal Training') THEN AMOUNT ELSE 0 END) AS
TOTAL_REVENUE
FROM BILLING;
```

Figure 6: Query 2

5.3 Calculate the total revenue generated from memberships, personal training sessions, and class bookings for a given month. (2 marks)

```
SELECT PTS.TRAINER_ID, ST.FIRST_NAME || ' ' || ST.LAST_NAME AS TRAINER_NAME, COUNT(PTS.TRAINING_ID) AS
SESSIONS_TAKEN
FROM PERSONAL_TRAINING_SESSIONS PTS
JOIN STAFF ST
ON PTS.TRAINER_ID = ST.STAFF_ID
GROUP BY PTS.TRAINER_ID, ST.FIRST_NAME, ST.LAST_NAME
ORDER BY SESSIONS_TAKEN DESC, TRAINER_NAME
FETCH FIRST 5 ROWS ONLY;
```

Figure 7: Query 4

5.4 List the top 5 trainers who have conducted the most personal training sessions. (2 marks)

```
SELECT
    CL.CLIENT_ID,
    CL.FIRST_NAME || ' ' || CL.LAST_NAME AS CLIENT_NAME,
    COUNT(GA.ATTENDANCE_ID) AS CHECKIN_COUNT
FROM GYM_ATTENDANCE GA
JOIN MEMBERSHIP MEM ON GA.CLIENT_ID = MEM.CLIENT_ID
JOIN CLIENTS CL ON GA.CLIENT_ID = CL.CLIENT_ID
WHERE MEM.END_DATE < SYSDATE -- Membership has expired
  AND GA.CHECK_IN > SYSDATE - 30 -- Checked in within the last 30 days
GROUP BY CL.FIRST_NAME, CL.LAST_NAME, CL.CLIENT_ID
ORDER BY CHECKIN_COUNT DESC;
```

Figure 8: Query 5

5.5 List all members whose membership has expired but who have attended the gym in the past 30 days. (2 marks)

```sql
SELECT
    CL.CLIENT_ID,
    CL.FIRST_NAME || ' ' || CL.LAST_NAME AS CLIENT_NAME,
    COUNT(GA.ATTENDANCE_ID) AS CHECKIN_COUNT
FROM GYM_ATTENDANCE GA
JOIN MEMBERSHIP MEM ON GA.CLIENT_ID = MEM.CLIENT_ID
JOIN CLIENTS CL ON GA.CLIENT_ID = CL.CLIENT_ID
WHERE MEM.END_DATE < SYSDATE -- Membership has expired
  AND GA.CHECK_IN > SYSDATE - 30 -- Checked in within the last 30 days
GROUP BY CL.FIRST_NAME, CL.LAST_NAME, CL.CLIENT_ID
ORDER BY CHECKIN_COUNT DESC;
```

Figure 9: Query 5

5.6 Find the members who have the most active workout plans with at least three different exercises in their current routine. (2 marks)

```sql
SELECT CL.FIRST_NAME || ' ' || CL.LAST_NAME AS CLIENT_NAME, COUNT(PTS.EXERCISE_TYPE) AS EXERCISES
FROM PERSONAL_TRAINING_SESSIONS PTS
JOIN CLIENTS CL ON PTS.CLIENT_ID = CL.CLIENT_ID
WHERE CL.CLIENT_CATEGORY = 'Member'
GROUP BY CL.FIRST_NAME, CL.LAST_NAME
-- using 'HAVING' function instead of 'WHERE' as group by does not support 'WHERE'
HAVING COUNT(PTS.EXERCISE_TYPE) >= 3
ORDER BY EXERCISES DESC, CLIENT_NAME;
```

Figure 10: Query 6

5.7 Calculate the total usage of discount codes and how much revenue was lost due to discounts applied in the last year. (2 marks)

```sql
SELECT DISCOUNT_CODE, USAGE_COUNT, REVENUE_LOSS
FROM DISCOUNTS
ORDER BY REVENUE_LOSS DESC;
```

Figure 11: Query 7

5.8 Identify the progress of members in the last month at the gym, including the number of fitness classes they attended. The result should display the member's name, the specific dates they attended, and the total number of classes attended during that period. (3 marks)

```sql
SELECT
    -- Concatenate the first and last names to create the full name
    CL.FIRST_NAME || ' ' || CL.LAST_NAME AS CLIENT_NAME,

    -- Use GYM_VISITS directly since it's guaranteed to be non-NULL
    CUR.GYM_VISITS AS GYM_VISITS,

    -- Progress percentage capped at 100%
    TRUNC(LEAST(100 * (CUR.GYM_VISITS / 21), 100)) || '%' AS PROGRESS,

    -- Last month's gym visits
    LAST.GYM_VISITS AS LAST_MONTH_GYM_VISITS,

    -- Progress percentage for the last month, capped at 100%
    TRUNC(LEAST(100 * (LAST.GYM_VISITS / 21), 100)) || '%' AS LAST_MONTH_PROGRESS,

    -- If no visit dates, use 'No Visits' explicitly
    CASE WHEN CUR.VISIT_DATES IS NULL THEN 'No Visits' ELSE CUR.VISIT_DATES END AS VISIT_DATES
FROM CLIENTS CL
-- Join for gym visits in the last 30 days
LEFT JOIN (
    SELECT
        GA.CLIENT_ID,
        COUNT(*) AS GYM_VISITS,
        LISTAGG(TO_CHAR(GA.CHECK_IN, 'DD-MM-YYYY'), ', ') WITHIN GROUP (ORDER BY GA.CHECK_IN) AS VISIT_DATES
    FROM GYM_ATTENDANCE GA
    WHERE GA.CHECK_IN >= SYSDATE - 30
    GROUP BY GA.CLIENT_ID
) CUR ON CL.CLIENT_ID = CUR.CLIENT_ID
-- Join for gym visits 31-60 days ago
LEFT JOIN (
    SELECT
        GA.CLIENT_ID,
        COUNT(*) AS GYM_VISITS
    FROM GYM_ATTENDANCE GA
    WHERE GA.CHECK_IN BETWEEN SYSDATE - 60 AND SYSDATE - 31
    GROUP BY GA.CLIENT_ID
) LAST ON CL.CLIENT_ID = LAST.CLIENT_ID
ORDER BY CLIENT_NAME;

```

Figure 12: Query 8

6 Triggers and Store Procedures (5 Marks)

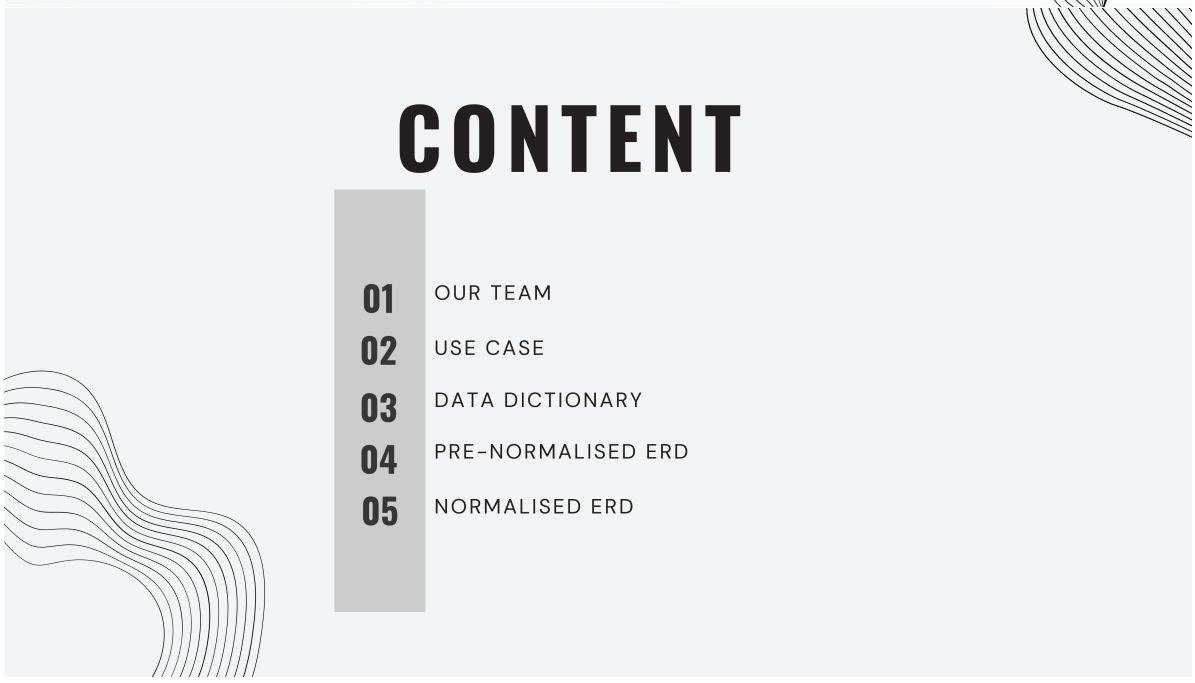
6.1 Create a trigger to automatically set the membership status to "Inactive" when a member's membership has expired (i.e., the current date exceeds the membership's end date). (2 marks)

6.2 Create a trigger to automatically decrease the available spots for a fitness class when a member books a class and ensure the class capacity is not exceeded. (2 marks)

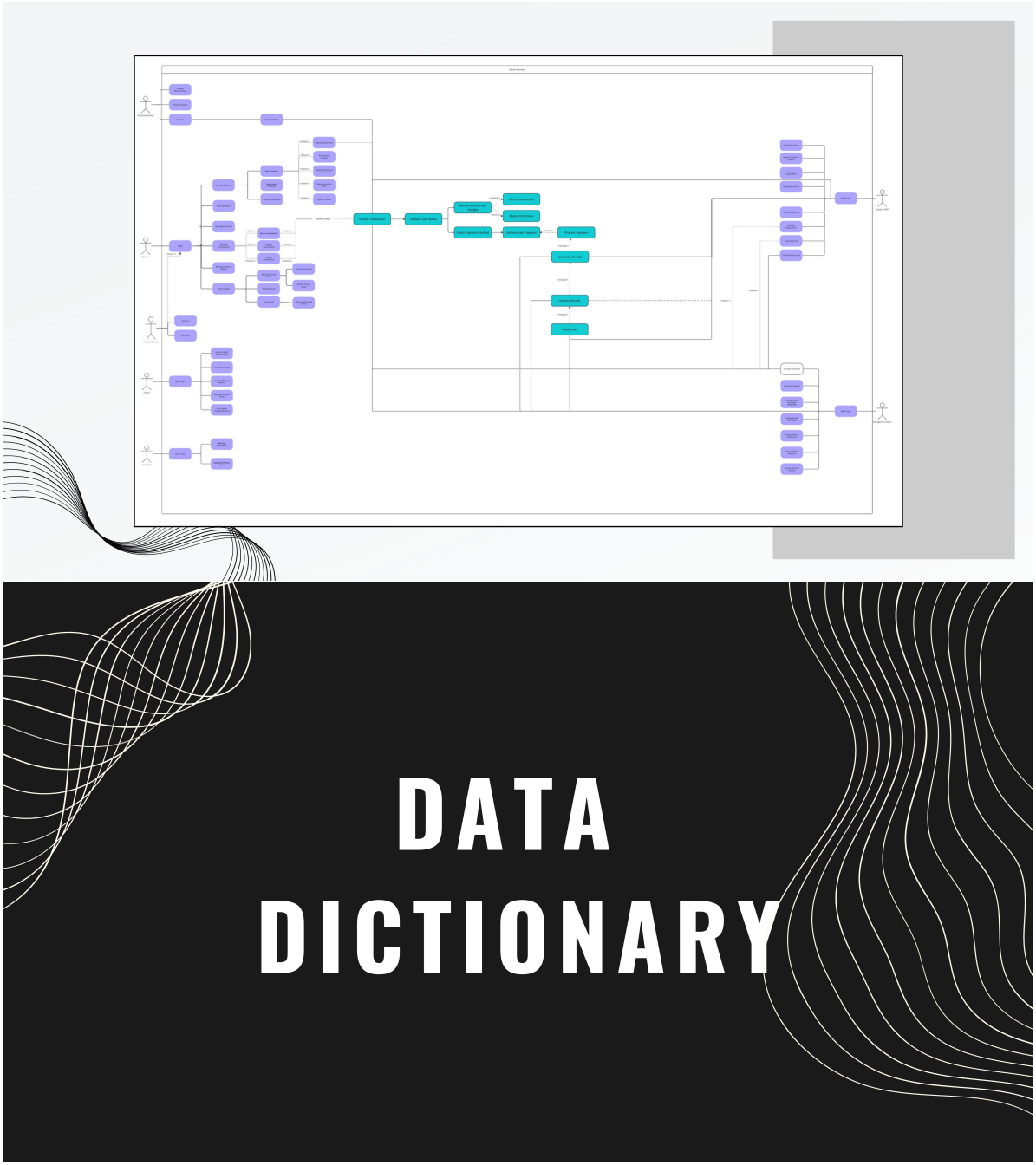
6.3 Create a trigger to automatically notify members when their membership is about to expire (e.g., 7 days before the end date). (1 mark)

7 Presentation (20 Marks)

7.1 Presentation Slides



USE CASE DIAGRAM



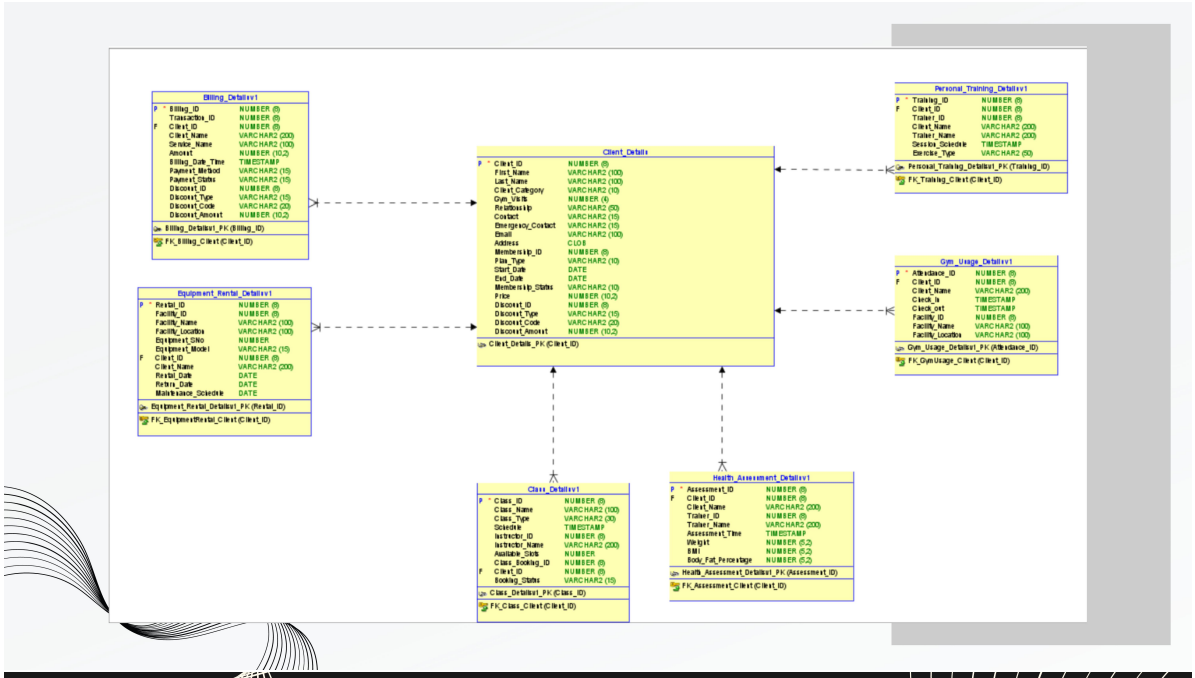
DATA DICTIONARY

	A	B	C	D	E	F
1	Table Name	Column Name	Data Type	Null	PK/FK	Description
2	Clients	Client_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each client
3	Clients	First_Name	VARCHAR2(100)	NOT NULL		Client's first name
4	Clients	Last_Name	VARCHAR2(100)	NOT NULL		Client's last name
5	Clients	Client_Category	VARCHAR2(10)	NOT NULL		Client's category (Member or Guest)
6	Clients	Gym_Visits	NUMBER(4)	NULL		Total number of gym visits
7	Clients	Relationship	VARCHAR2(50)	NULL		Relationship to a member (if applicable)
8	Clients	Contact	VARCHAR2(15)	NOT NULL		Primary contact number
9	Clients	Emergency_Contact	VARCHAR2(15)	NULL		Emergency contact number
10	Clients	Email	VARCHAR2(100)	NOT NULL	UNIQUE	Client's email address
11	Clients	Address	CLOB	NULL		Client's address
12	Membership	Membership_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each membership
13	Membership	Client_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Client table
14	Membership	Plan_Type	VARCHAR2(10)	NOT NULL		Membership plan type (Silver, Gold, Platinum)
15	Membership	Start_Date	DATE	NOT NULL		Membership start date
16	Membership	End_Date	DATE	NOT NULL		Membership end date
17	Membership	Status	VARCHAR2(10)	NOT NULL		Membership status (Active or Inactive)
18	Membership	Price	NUMBER(10,2)	NOT NULL		Membership price
19	Fitness_Classes	Class_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each fitness class
20	Fitness_Classes	Class_Name	VARCHAR2(100)	NOT NULL		Class name
21	Fitness_Classes	Class_Type	VARCHAR2(30)	NOT NULL		Class type (Core Exercises, Yoga, Pilates, Aerobic Fitness)
22	Fitness_Classes	Schedule	TIMESTAMP	NOT NULL		Class schedule
23	Fitness_Classes	Instructor_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Staff table
24	Fitness_Classes	Available_Slots	NUMBER	NOT NULL		Number of available slots in the class

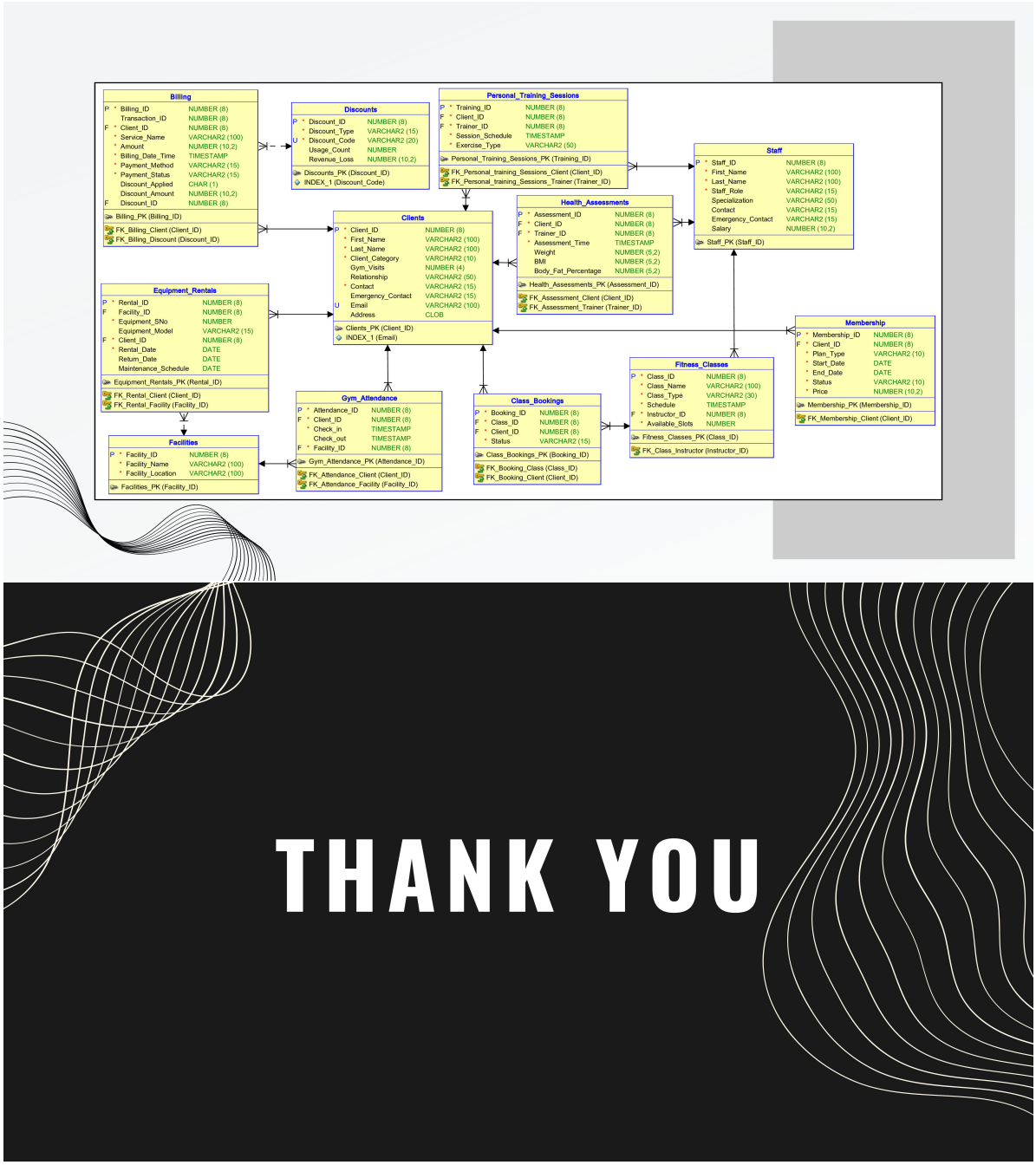
	A	B	C	D	E	F
25	Class_Bookings	Booking_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each class booking
26	Class_Bookings	Class_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Fitness_Classes table
27	Class_Bookings	Client_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Clients table
28	Class_Bookings	Status	VARCHAR2(15)	NOT NULL		Booking status (Confirmed, Canceled, Waitlisted)
29	Personal_Training_Sessions	Training_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each personal training session
30	Personal_Training_Sessions	Client_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Clients table
31	Personal_Training_Sessions	Trainer_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Staff table
32	Personal_Training_Sessions	Session_Schedule	TIMESTAMP	NOT NULL		Session schedule
33	Personal_Training_Sessions	Exercise_Type	VARCHAR2(50)	NOT NULL		Type of exercise (Cardio, Strength Training, Calisthenics, Other)
34	Health_Assessments	Assessment_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each health assessment
35	Health_Assessments	Client_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Clients table
36	Health_Assessments	Trainer_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Staff table
37	Health_Assessments	Assessment_Time	TIMESTAMP	NOT NULL		Assessment time
38	Health_Assessments	Weight	NUMBER(5,2)	NULL		Client's weight
39	Health_Assessments	BMI	NUMBER(5,2)	NULL		Client's BMI
40	Health_Assessments	Body_Fat_Percentage	NUMBER(5,2)	NULL		Client's body fat percentage
41	Gym_Attendance	Attendance_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each gym attendance record
42	Gym_Attendance	Client_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Clients table
43	Gym_Attendance	Check_in	TIMESTAMP	NOT NULL		Check-in time
44	Gym_Attendance	Check_out	TIMESTAMP	NULL		Check-out time
45	Gym_Attendance	Facility_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Facilities table

	A	B	C	D	E	F
46	Billing	Billing_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each billing record
47	Billing	Transaction_ID	NUMBER(8)	NULL		Transaction ID (if applicable)
48	Billing	Client_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Clients table
49	Billing	Service_Name	VARCHAR2(100)	NOT NULL		Service name (Membership, Class Booking, Personal Training, Health Assessment, Other)
50	Billing	Amount	NUMBER(10,2)	NOT NULL		Amount charged
51	Billing	Billing_Date_Time	TIMESTAMP	NOT NULL		Billing date and time
52	Billing	Payment_Method	VARCHAR2(15)	NOT NULL		Payment method (Credit Card, Debit Card, Cash, Direct Debit)
53	Billing	Payment_Status	VARCHAR2(15)	NOT NULL		Payment status (CONFIRMED, PENDING)
54	Billing	Discount_Applied	CHAR(1)	NOT NULL		Whether a discount was applied (Y/N)
55	Billing	Discount_Amount	NUMBER(10,2)	NOT NULL		Discount amount
56	Billing	Discount_ID	NUMBER(8)	NULL	FK	Foreign key referencing the Discounts table
57	Discounts	Discount_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each discount
58	Discounts	Discount_Type	VARCHAR2(15)	NOT NULL		Discount type (Promotion, Referral)
59	Discounts	Discount_Code	VARCHAR2(20)	NOT NULL	UNIQUE	Discount code
60	Discounts	Usage_Count	NUMBER	NOT NULL		Number of times the discount has been used
61	Discounts	Revenue_Loss	NUMBER(10,2)	NOT NULL		Total revenue loss due to the discount
62	Staff	Staff_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each staff member
63	Staff	First_Name	VARCHAR2(100)	NOT NULL		Staff member's first name
64	Staff	Last_Name	VARCHAR2(100)	NOT NULL		Staff member's last name
65	Staff	Staff_Role	VARCHAR2(15)	NOT NULL		Staff role (Trainer, Instructor, Support Staff, Admin)
66	Staff	Specialization	VARCHAR2(50)	NULL		Staff member's specialization (if applicable)
67	Staff	Contact	VARCHAR2(15)	NOT NULL		Staff member's contact number
68	Staff	Emergency_Contact	VARCHAR2(15)	NULL		Staff member's emergency contact number
69	Staff	Salary	NUMBER(10,2)	NOT NULL		Staff member's salary

ERD



NORMALISED ERD



7.2 Data Dictionary

7.2.1 Clients Table

Column Name	Data Type	Null	PK/FK	Description
Client_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each client
First_Name	VARCHAR2(100)	NOT NULL		Client's first name
Last_Name	VARCHAR2(100)	NOT NULL		Client's last name
Client_Category	VARCHAR2(10)	NOT NULL		Client's category (Member or Guest)
Gym_Visits	NUMBER(4)	NULL		Total number of gym visits
Relationship	VARCHAR2(50)	NULL		Relationship to a member (if applicable)
Contact	VARCHAR2(15)	NOT NULL		Primary contact number
Emergency_Contact	VARCHAR2(15)	NULL		Emergency contact number
Email	VARCHAR2(100)	NOT NULL	UNIQUE	Client's email address
Address	CLOB	NULL		Client's address

7.2.2 Membership Table

Column Name	Data Type	Null	PK/FK	Description
Membership_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each membership
Client_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Clients table
Plan_Type	VARCHAR2(10)	NOT NULL		Membership plan type (Silver, Gold, Platinum)
Start_Date	DATE	NOT NULL		Membership start date
End_Date	DATE	NOT NULL		Membership end date
Status	VARCHAR2(10)	NOT NULL		Membership status (Active or Inactive)
Price	NUMBER(10,2)	NOT NULL		Membership price

7.2.3 Fitness_Classes Table

Column Name	Data Type	Null	PK/FK	Description
Class_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each fitness class
Class_Name	VARCHAR2(100)	NOT NULL		Class name
Class_Type	VARCHAR2(30)	NOT NULL		Class type (Core, Yoga, Pilates, Aerobic)
Schedule	TIMESTAMP	NOT NULL		Class schedule
Instructor_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Staff table
Available_Slots	NUMBER	NOT NULL		Number of available slots in the class

7.2.4 Class_Bookings Table

Column Name	Data Type	Null	PK/FK	Description
Booking_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each class booking
Class_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing Fitness_Classes table
Client_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Clients table
Status	VARCHAR2(15)	NOT NULL		Booking status (Confirmed, Canceled)

7.2.5 Health_Assessments Table

Column Name	Data Type	Null	PK/FK	Description
Assessment_ID	NUMBER(8)	NOT NULL	PK	Unique identifier for each health assessment
Client_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Clients table
Trainer_ID	NUMBER(8)	NOT NULL	FK	Foreign key referencing the Staff table
Assessment_Time	TIMESTAMP	NOT NULL		Assessment time
Weight	NUMBER(5,2)	NULL		Client's weight

Column Name	Data Type	Null	PK/FK	Description
BMI	NUMBER(5,2)	NULL		Client's BMI
Body_Fat_Percentage	NUMBER(5,2)	NULL		Client's body fat percentage

8 Appendices

- **ERD and UML Diagrams**
- **SQL Scripts**
- **Gantt Chart**
- **Meeting Minutes and Group Contributions**.