Final Project

Database Systems

Prem Kumar Chimakurthi

Master of Science in Business Analytics

Gym Management System Project

Requirements Analysis:

To design an effective operational database and data warehouse for a **Gym Management System**, we need to define the business requirements, key operations, and constraints.

a. Operations Captured by the Database:

The database must support the following operations:

1. **Member Management:** Register new members, update member details, track membership expiration.
2. **Trainer Management:** Assign trainers to members, store trainer details, track training sessions.
3. **Attendance Tracking:** Record member check-ins and check-outs.
4. **Payment Processing:** Track membership payments, generate invoices, and manage refunds.
5. **Class Scheduling:** Manage fitness class schedules, enroll members, and track attendance.
6. **Equipment Management:** Track gym equipment inventory, maintenance schedules, and usage.

7. **Employee Payroll:** Process salaries for trainers and staff based on attendance and hours worked.
8. **Reporting & Analytics:** Generate reports for revenue, membership trends, and trainer performance.

b. Information to be Tracked:

The database will store the following data points:

1. Member Information:

- Member ID (Primary Key)
- Name
- Contact Information
- Membership Type (Monthly, Annual, etc.)
- Start & End Date of Membership
- Payment Status
- Attendance Records

2. Trainer Information:

- Trainer ID (Primary Key)
- Name
- Contact Details
- Specialization (e.g., Yoga, Strength Training)
- Assigned Members
- Salary and Work Hours

3. Attendance & Check-in Data:

- Attendance ID (Primary Key)

- Member ID (Foreign Key)
- Check-in Time
- Check-out Time

4. Payment Records:

- Payment ID (Primary Key)
- Member ID (Foreign Key)
- Payment Date
- Amount Paid
- Payment Method (Credit Card, Cash, etc.)

5. Class & Session Scheduling:

- Class ID (Primary Key)
- Class Name
- Trainer ID (Foreign Key)
- Schedule (Date & Time)
- Enrolled Members

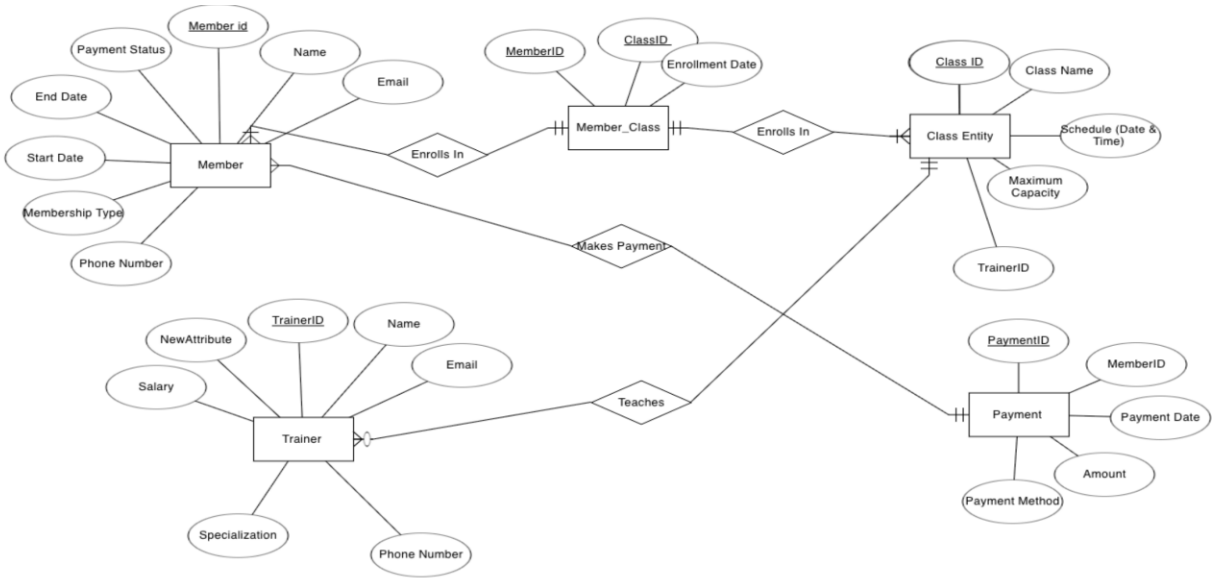
6. Equipment Management:

- Equipment ID (Primary Key)
- Name & Type
- Maintenance Schedule
- Condition Status

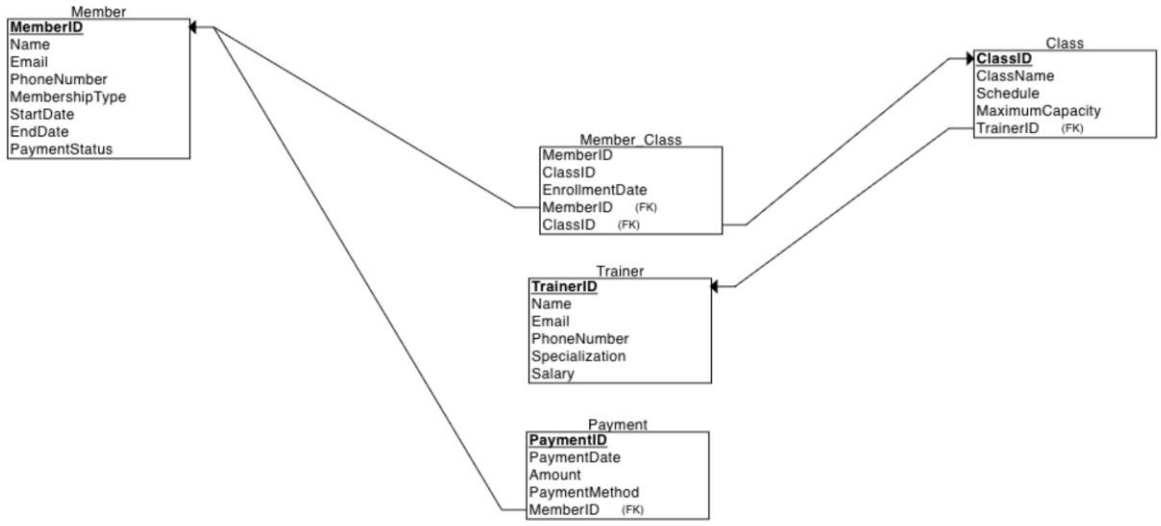
c. Relationships and Constraints:

1. **One-to-Many Relationships:**
    a. One **trainer** can be assigned to multiple **members**.
    b. One **member** can enroll in multiple **classes**.
    c. One **payment** belongs to one **member**.
2. **Many-to-Many Relationships:**
    a. **Members & Classes:** A member can join multiple classes, and a class can have multiple members (requires a junction table like Member_Class).
    b. **Trainers & Classes:** A trainer can lead multiple classes, and a class may have multiple trainers.
3. **Constraints:**
    a. **Primary Keys** to ensure uniqueness (e.g., MemberID, TrainerID).
    b. **Foreign Keys** to enforce referential integrity (e.g., TrainerID in Class table should exist in Trainer table).
    c. **Check Constraints** to enforce business rules (e.g., Payment Amount > 0).
    d. **Not Null Constraints** on critical fields (e.g., Member Name, Payment Amount).
    e. **Unique Constraints** on fields like Email and Phone Number to prevent duplicates.

ER Diagram:



Relational Schema:



Referential Integrity Constraints for Gym Management Database:

Referential integrity constraints ensure **data consistency** by preventing orphan records and enforcing **relationships between tables:**

1. Payment Table (MemberID FK):

- **Constraint:** MemberID in Payment references MemberID in Member.
- **Action on Delete:**
    - **ON DELETE CASCADE** → If a member is deleted, all their Payments should also be deleted.
- **Action on Update:**
    - **ON UPDATE CASCADE** → If MemberID changes in Member, it should update in Payment.

SQL Constraint:

```
FOREIGN KEY (MemberID) REFERENCES Member (MemberID)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

2. Class Table (TrainerID FK):

- **Constraint:** TrainerID in Class references TrainerID in Trainer.
- **Action on Delete:**
    - **ON DELETE SET NULL** → If a Trainer is deleted, their TrainerID should be set to NULL in Class (so the class still exists).
- **Action on Update:**
    - **ON UPDATE CASCADE** → If TrainerID changes, update it in Class.

SQL Constraint:

```
FOREIGN KEY (TrainerID) REFERENCES Trainer (TrainerID)
ON DELETE SET NULL
ON UPDATE CASCADE;
```

3. Member_Class Table (MemberID & ClassID FKs):

- **Constraints:**
    - MemberID in Member_Class references MemberID in Member.
    - ClassID in Member_Class references ClassID in Class.

Action on Delete:

- **ON DELETE CASCADE** → If a Member or Class is deleted, their related enrollments should also be deleted.

- **Action on Update:**
    - **ON UPDATE CASCADE** → If MemberID or ClassID changes, update them in Member_Class.

SQL Constraint:

```
FOREIGN KEY (MemberID) REFERENCES Member (MemberID)
ON DELETE CASCADE
ON UPDATE CASCADE,

FOREIGN KEY (ClassID) REFERENCES Class (ClassID)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

4. Class Table (Maximum Capacity Constraint):

- **Constraint:** Maximum Capacity should **never be negative** and should be a **positive integer**.
- **Solution:** Using **CHECK Constraint** to enforce this.

SQL Constraint:

```
CHECK (Maximum Capacity > 0);
```

5. Payment Table (Amount Constraint):

- **Constraint:** Amount should **always be greater than 0**.
- **Solution:** Using **CHECK Constraint**

SQL Constraint:

```
CHECK (Amount > 0);
```

Implementation SQL for Gym Management Database:

Below is the **DDL (CREATE TABLE)** and **DML (INSERT INTO)** SQL implementation for your **Gym Management System** database.

1. DDL (CREATE TABLE Statements):

Create Member Table:

```
CREATE TABLE Member (
    MemberID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR (100) NOT NULL,
    Email VARCHAR (150) UNIQUE NOT NULL,
    PhoneNumber VARCHAR (15) NOT NULL,
    MembershipType VARCHAR (50) NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE NULL,
```

```
    PaymentStatus VARCHAR (20) NOT NULL
);
```

Create Trainer Table:

```
CREATE TABLE Trainer (
    TrainerID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR (100) NOT NULL,
    Email VARCHAR (150) UNIQUE NOT NULL,
    PhoneNumber VARCHAR (15) NOT NULL,
    Specialization VARCHAR (100) NOT NULL,
    Salary NUMERIC (10,2) NOT NULL
);
```

Create Class Table:

```
CREATE TABLE Class (
    ClassID INT PRIMARY KEY AUTO_INCREMENT,
    ClassName VARCHAR (100) NOT NULL,
    Schedule DATE NOT NULL,
    MaximumCapacity INT CHECK (MaximumCapacity > 0) NOT NULL,
    TrainerID INT,
    FOREIGN KEY (TrainerID) REFERENCES Trainer(TrainerID) ON DELETE SET NULL ON UPDATE CASCADE
);
```

Create Payment Table:

```
CREATE TABLE Payment (
   PaymentID INT PRIMARY KEY AUTO_INCREMENT,
   MemberID INT,
   PaymentDate DATE NOT NULL,
   Amount NUMERIC (10,2) CHECK (Amount > 0) NOT NULL,
   Payment Method VARCHAR (50) NOT NULL,
   FOREIGN KEY (MemberID) REFERENCES Member(MemberID) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Create Member_Class (Junction Table for Many-to-Many Relationship):

```
CREATE TABLE Member_Class (
   MemberID INT,
   ClassID INT,
   EnrollmentDate DATE NULL,
   PRIMARY KEY (MemberID, ClassID),
   FOREIGN KEY (MemberID) REFERENCES Member (MemberID) ON DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY (ClassID) REFERENCES Class (ClassID) ON DELETE CASCADE ON UPDATE CASCADE
```

```
);
```

2. DML (INSERT INTO Statements):

Insert 10 Sample Records into Member Table:

```
INSERT INTO Member (Name, Email, PhoneNumber, MembershipType, StartDate, EndDate, PaymentStatus)
VALUES

('John Doe', 'johndoe@email.com', '1234567890', 'Gold', '2024-01-01', '2025-01-01', 'Paid'),

('Jane Smith', 'janesmith@email.com', '2345678901', 'Silver', '2024-02-01', '2025-02-01', 'Paid'),

('Alice Brown', 'alicebrown@email.com', '3456789012', 'Gold', '2024-03-01', '2025-03-01', 'Unpaid'),

('Bob White', 'bobwhite@email.com', '4567890123', 'Platinum', '2024-04-01', '2025-04-01', 'Paid'),

('Charlie Black', 'charlieblack@email.com', '5678901234', 'Silver', '2024-05-01', '2025-05-01', 'Unpaid'),

('David Green', 'davidgreen@email.com', '6789012345', 'Gold', '2024-06-01', '2025-06-01', 'Paid'),

('Eva Blue', 'evablue@email.com', '7890123456', 'Platinum', '2024-07-01', '2025-07-01', 'Paid'),
```

```
('Frank Yellow', 'frankyellow@email.com', '8901234567', 'Silver', '2024-08-01', '2025-08-01', 'Unpaid'),

('Grace Red', 'gracered@email.com', '9012345678', 'Gold', '2024-09-01', '2025-09-01', 'Paid'),

('Harry Orange', 'harryorange@email.com', '0123456789', 'Platinum', '2024-10-01', '2025-10-01', 'Paid');
```

Insert 3 Sample Records into Trainer Table:

```
INSERT INTO Trainer (Name, Email, PhoneNumber, Specialization, Salary)

VALUES

('Mike Johnson', 'mikejohnson@email.com', '3216549870', 'Weight Training', 50000),

('Sarah Lee', 'sarahlee@email.com', '9876543210', 'Yoga', 45000),

('Tom Baker', 'tombaker@email.com', '7418529630', 'Cardio', 48000);
```

Insert 5 Sample Records into Class Table:

```
INSERT INTO Class (ClassName, Schedule, MaximumCapacity, TrainerID)

VALUES

('Yoga Morning', '2024-03-15', 20, 2),

('Weightlifting Basics', '2024-03-20', 15, 1),

('Cardio HIIT', '2024-03-22', 25, 3),
```

```
('Advanced Yoga', '2024-03-25', 18, 2),
('Strength Training', '2024-03-30', 10, 1);
```

Insert 5 Sample Records into Payment Table:

```
INSERT INTO Payment (MemberID, PaymentDate, Amount, PaymentMethod)
VALUES
(1, '2024-02-01', 100.00, 'Credit Card'),
(2, '2024-02-05', 90.00, 'PayPal'),
(3, '2024-02-10', 120.00, 'Bank Transfer'),
(4, '2024-02-15', 150.00, 'Credit Card'),
(5, '2024-02-20', 80.00, 'Cash');
```

Insert 5 Sample Records into Member_Class Table:

```
INSERT INTO Member_Class (MemberID, ClassID, EnrollmentDate)
VALUES
(1, 1, '2024-03-01'),
(2, 2, '2024-03-05'),
(3, 3, '2024-03-10'),
```

```
(4, 4, '2024-03-15'),
(5, 5, '2024-03-20');
```

Data Warehouse Requirements for Gym Management System:

A **Data Warehouse** is used to store and analyze historical data for **business intelligence (BI)** and reporting. Below are the **requirements** based on your **Gym Management Database**.

The subject of analysis is Gym Membership and Class Utilization. This will help the gym management analyze:

- Member retention rates
- Class popularity and attendance trends
- Trainer performance
- Revenue generation from memberships and payments
- Seasonal trends in gym usage

To analyze gym performance and member activity, we need data from multiple tables. Below are the key attributes:

Fact Table (Measures for Analysis):

- **Total Revenue** → Sum of Amount from Payment table

- **Total Active Members** → Count of MemberID where EndDate is NULL or in the future
- **Class Enrollment Count** → Count of MemberID per ClassID from Member_Class
- **Trainer Class Count** → Number of classes assigned per TrainerID
- **Average Payment per Member** → Total Amount / Count of MemberID
- **Membership Type Distribution** → Count of each MembershipType

Dimension Tables (Contextual Information):

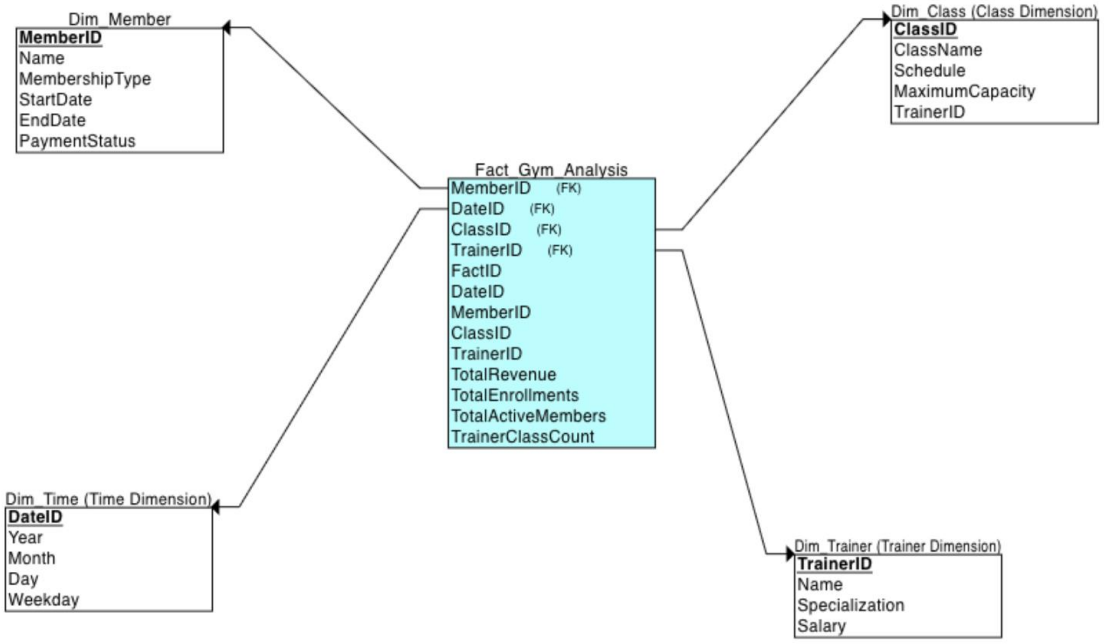
Dimension	Attributes Used
Member	MemberID, MembershipType, StartDate, EndDate, PaymentStatus
Class	ClassID, ClassName, Schedule, MaximumCapacity
Trainer	TrainerID, Name, Specialization
Payment	PaymentID, PaymentDate, Amount, PaymentMethod
Time	Year, Month, Day (extracted from PaymentDate and Schedule)

Granularity refers to the level of detail stored in the data warehouse. For this analysis:

1. **Lowest Granularity (Transaction Level):**
    a. Individual Payment records per Member
    b. Individual Class Enrollment records per Member
    c. Individual Classes Conducted by Trainer

2. **Aggregated Granularity (Summary Level):**
    a. **Monthly Revenue** = SUM(Amount) grouped by Month
    b. **Yearly Membership Growth** = COUNT(MemberID) grouped by Year
    c. **Class Attendance per Session** = COUNT(MemberID) per ClassID
    d. **Trainer Performance** = COUNT(ClassID) per TrainerID

Star Schema:



The Star Schema for the Gym Management Data Warehouse is designed to provide efficient, structured, and fast analytical queries for key business insights. The schema consists of a Fact Table (Fact_Gym_Analysis) connected to four Dimension Tables (Dim_Member, Dim_Class, Dim_Trainer, and Dim_Time), enabling powerful multi-dimensional analysis.

Revenue Analysis

- **Measure:** TotalRevenue
- **Insights:** Tracks total revenue from membership payments over time.
- **Example Queries:**
    - Revenue trends by **month, quarter, and year**.
    - Revenue breakdown by **trainer, class type, and member category**.

Trainer Performance Analysis

- **Measure:** TrainerClassCount (Total classes assigned to a trainer).
- **Insights:** Helps gym managers track **trainer workload, specialization, and salary analysis**.

Member Activity & Retention:

- Measure: TotalActiveMembers, TotalEnrollments
- Insights: Helps track member participation, retention, and class preferences.

Seasonal Trends & Time-Based Analysis

- **Measure:** TotalEnrollments by DateID
- **Insights:** Helps analyze demand for gym classes across **different times of the year**.

Conclusion:

The **Star Schema** effectively transforms raw transactional data into **meaningful insights**, enabling gym management to **optimize revenue, improve trainer performance, and enhance member engagement**.

This structured approach ensures **data-driven decision-making**, making the gym management system more **efficient and profitable**