Project Report

Fitness Club Membership Management

PostgreSQL database design and implementation summary

Scope and Requirements

This report describes a PostgreSQL database created for managing a fitness club. The database is built to store and manage:

- Members and their status
- Membership plans and purchased subscriptions with start and end dates
- Trainers, class types, and scheduled class sessions
- Attendance records linked to scheduled sessions
- Payments linked to both members and subscriptions
- Locker assignments for members (optional operational feature)

The implementation is organized as SQL scripts that separate schema creation, sample data loading, queries, transactions and indexes, and reusable views and functions.

ER Model Summary

The conceptual design is centered around nine entities and their relationships:

- Member
- MembershipPlan
- Subscription
- Trainer
- Class
- ClassSchedule
- Attendance
- Payment
- LockerAssignment

Key relationships:

- Member to Subscription: one member can have multiple subscriptions over time (1:N).
- MembershipPlan to Subscription: one plan can be purchased many times (1:N).
- Subscription to Payment: one subscription can be associated with multiple payments (1:N).
- Member to Payment: one member can have multiple payments (1:N).
- Trainer to ClassSchedule: one trainer can lead many scheduled sessions (1:N).

- Class to ClassSchedule: one class type can appear in the schedule many times (1:N).
- ClassSchedule to Attendance: one scheduled session can have many attendance records (1:N).
- Member to Attendance: one member can attend many scheduled sessions (1:N).
- Member to LockerAssignment: a member can have zero or one locker assignment (1:0..1).

PostgreSQL Implementation and Data Integrity

The schema is implemented with normalized tables and explicit integrity rules. Constraints are used to prevent common operational errors. Member and trainer emails are unique; payments must have a positive amount; scheduled sessions require a positive capacity and a valid time range; and attendance is constrained so the same member cannot be recorded twice for the same scheduled session.

SQL Queries

The query set is split into two groups. Basic queries cover selection, filtering, ordering, and joins to produce operational lists such as active members, plan lists, and schedules with trainer names. Advanced queries provide reporting-style outputs using aggregation (GROUP BY and HAVING), CTEs for structured reports, and window functions for rankings such as identifying the most active members by attendance.

Transactions

Transactions are demonstrated through multi-step operations that must be applied atomically. The first example registers a new member, creates a subscription for a chosen plan, and records a payment within a single transaction. A rollback case illustrates how the database returns to a consistent state if an invalid statement is executed. This matches real workflows where partial updates would be unacceptable.

Indexing

Indexes are defined to support common access patterns. Member lookup fields (such as last name and email), subscription filters (member and status), and schedule time searches are indexed. Foreign key columns used heavily in joins (attendance and payments) are also indexed. These choices reduce scan time for frequent reports and improve performance as the dataset grows.

Views and Functions

Views provide reusable reporting layers, including:

- Active members

- Active subscriptions
- Class occupancy
- Trainer schedule
- Payment summary
- Members with no attendance
- Expired subscriptions

Functions encapsulate small reusable actions such as:

- Returning a member’s full name
- Extending a subscription end date
- Registering attendance while avoiding duplicates
- Calculating total completed payments for a member

Backup and Recovery

A logical backup strategy is documented using pg_dump. The custom dump format supports reliable restoration with pg_restore, while an optional plain SQL dump can be used for readability. The recovery workflow restores into a separate database and verifies key tables with simple count checks before the restored copy is considered valid.

Conclusion

The Fitness Club Membership Management database satisfies the project requirements: a consistent ER-driven schema in PostgreSQL, a structured set of operational and analytical queries, transaction examples that protect data consistency, indexing aligned with expected query patterns, and a practical backup and recovery procedure. The design is clear, extendable, and ready to be used as a foundation for an application layer if required.