ONLINE FITNESS PLATFORM

MSIS 2613 (Database Management Systems - Design, Development & Administration)

Presented by:

Mehak Agrawal

Shruthi Suresh

Padma Rishitha Pusapati

Samhita Komandur Sreenivasa

TABLE OF CONTENTS

TABLE OF FIGURES

BUSINESS DESCRIPTION OVERVIEW

More people are focusing on their health these days, thanks to the COVID-19 pandemic. This has created a demand for new fitness programs that work for different people's needs and interests. The Online Fitness Platform has new exciting features and cutting-edge tools that aim to shake up the fitness world. We're building this online fitness platform because more and more people are looking for easy and personalized ways to get healthy and fit in our busy lives. Our goal is to create a one-stop complete fitness experience. Trainers and clients can access the platform's functionalities with restricted access thanks to this table, which acts as a secure gateway.

Our inspiration for developing this innovative online fitness platform stemmed from our admiration for Chloe Ting's exceptional work and dedication in the realm of fitness. Upon exploring Chloe Ting's website and witnessing the impactful features she offers, such as personalized workout plans, tailored nutrition guidance, and a vast array of exercises, we were deeply inspired. Recognizing the transformative potential of such offerings, we were motivated to create a platform that encapsulates the essence of Chloe Ting's approach, while also incorporating our own unique features and enhancements. Drawing from her commitment to personalized fitness solutions, we aim to provide users with a comprehensive and intuitive platform that caters to their individual needs and empowers them to achieve a healthy lifestyle.

Nature and Scope

Our platform offers two main sections:

- Online: This section provides workout plans, exercise routines focused on specific muscle groups (like abs, legs, biceps) and individual exercises. Everything is tailored to your preferences and fitness goals.
- Offline: For our esteemed Gold members, we introduce an exclusive Offline experience. This premium tier offers unparalleled access to personalized fitness sessions conducted both individually and in groups, facilitated by certified trainers. Through these sessions, members can receive expert guidance and support, tailored to their unique needs and objectives, elevating their fitness journey to new heights.

We have many other features like:

- To help us improve, we have a "Feedback" system where you can leave reviews for your trainers. This helps us maintain a high standard of service and keeps the trainer community accountable.
- We make paying for your membership and classes easy through our secure "Payment" system.
- The platform also includes a "BMI Calculator" to help you understand your health better. It uses your weight and height to calculate your Body Mass Index (BMI), which can be a helpful tool for tracking your fitness journey.

In addition to drawing inspiration from Chloe Ting's impactful workout exercises and features, we've innovatively expanded our platform to include customer referral options and comprehensive nutrition plans. Understanding the significance of community support in fitness journeys, we've integrated a referral system, empowering users to share their positive experiences with others and foster a supportive network. Furthermore, recognizing the critical role of nutrition in achieving holistic wellness, we've incorporated detailed nutrition plans into our platform, ensuring that users receive expert guidance on healthy eating habits to complement their workout routines effectively. By combining these elements with our commitment to personalized fitness solutions, we strive to provide users with a truly comprehensive and transformative online fitness experience.

Purpose:

- Unified Information Hub: All user information (client and trainer), membership choices, workout details, trainer schedules, user reviews, and maybe fitness content (such exercise routines) are stored in one single spot via the database. This guarantees consistency across the system and removes the need for data to be dispersed across several places.
- Organized Data Management: This information can be efficiently arranged and managed thanks to the database's structure. It is simpler to add new information, change old data, and retrieve information as needed since it is stored in a well-defined format with obvious links between various pieces.

Benefits:

- Customized Workouts: Using information about your past workouts, progress, and fitness objectives, the app generates customized training schedules that are tailored to your requirements.
- Easy Interactions: The database makes sure that information is accessible for important functions without much effort. Training plans tailored to your membership level may be found, trainer sessions can be scheduled with ease, and payments are processed quickly.
- Friendly Network: With such functionalities, the database can facilitate elements that promote a friendly atmosphere, such as user reviews for trainers and feedbacks as well.
- Safeguarding Your Data: Consider the database to be your information's high-tech locker room. It employs unique strategies to safeguard your login credentials, allowing you to worry-free concentrate on your exercise.
- Getting Payments Done Quickly: Additionally, the database serves as the platform's extremely well-organized wallet. You may easily do in-app membership purchases because it securely remembers your payment information.
- Prepared for Any Future Developments: Like your fitness journey, the online Fitness Platform is made to evolve and change with you. Increased user capacity is incorporated into the database.
- Keeping Things Organized: Picture yourself strewn all over the gym with your exercise regimens, personal trainer information. For the model, the database serves as a central filing cabinet, keeping everything tidy and accessible. This guarantees that the information you have is always current and accessible.
- Better platform, Wiser Decisions: The information you produce from your exercises isn't private. It also aids in the decision-making process for the team. They are able to see the most popular workouts, user behavior on the app, and even pinpoint areas in need of the customer.

Anticipated Features and Functionalities

- Personalized workout plans: We'll create custom workout plans based on your fitness level and goals.
- Nutrition plans: Get advice on healthy eating to fuel your workouts and see results.
- Connect with trainers: Get guidance and support from certified trainers.
- Huge library of exercises: Find all kinds of workouts to keep your routine fresh and fun
- We offer different membership plans to fit your needs:
    - Silver membership: This plan gives you access to all our online workout classes.
    - Gold membership: This plan gives you everything in the silver plan, plus the benefit of working with a certified trainer for personalized guidance.
- The app is easy to use: We designed the app to be clear and simple to navigate. You'll be able to find everything you need quickly, so you can focus on your workout.
- Build a support network: The app helps you connect with other users and trainers. You can share tips, ask questions, and get motivated by the community.
- BMI calculator: See your Body Mass Index (BMI) and track your progress over time.
- Get personalized tips: The app will learn from your workouts and suggest exercises or plans that are right for you based on feedback from other users.
- The app will grow with you: We designed the app to be flexible and keep up with the latest fitness trends. This means you'll always have access to the newest features and workouts.
- We're always improving: We're constantly listening to user feedback and adding new features based on the latest research and technology. This ensures the app stays ahead of the curve in the fitness world.

CONCEPTUAL DESIGN - ENHANCED ENTITY RELATIONSHIP (EER) MODEL

The objective of this section is to specify the data elements essential for efficiently managing and streamlining business processes while creating a comprehensive representation of the data structure**.** The EER model provides a visual representation of the conceptual design, illustrating entities, attributes, and relationships comprehensively.

Below are the main components of the EER model:

- **Entity Name**: **Login**

  The "Login" entity represents the users logging into the system. It stores information related to user credentials and login timestamps. This entity is essential for managing user access and security within the system.

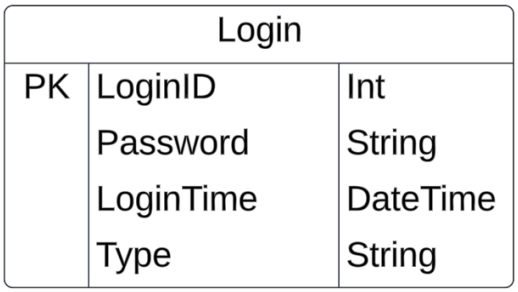

Fig 1: Login Entity

  Attributes:

  - **loginId** (Primary Key): The "loginId" attribute serves as the primary key for the "Login" table, ensuring that each person has a login_id to access the portal that is uniquely identifiable within the database.
  - **password**: A string field storing the hashed or encrypted password associated with the user account. It ensures the security of user credentials by storing them in a protected format.
  - **login_time**: A datetime field capturing the timestamp when the user logged into the system. It records the date and time of each login event, providing insights into user activity and login patterns.
  - **type**: A string field indicating the type or category of the user account. This attribute can be used to differentiate between two types of users, such as customers or trainers.

- **Entity Name: Customer**

  The "Customer" table stores information about the customers registered in the system.

Customer		
PK,FK	CustomerID	Int
	Name	String
	Referred MemberID	Int
	Referral Discount	Int
	BMI	Int

Fig 2: Customer Entity

Attributes:

  - **CustomerID** (Primary Key, Foreign Key): An integer value serving as both the primary key and foreign key for the "Customer" table. It uniquely identifies each customer record in the database and also serves as a reference to the login identifier (loginId) from the "Login" table.
  - **Name**: A string field storing the name of the customer. It represents the personal name or full name of the customer.
  - **Referred MemberID**: An integer value representing the identifier of the member who referred the customer, if applicable. It establishes a connection between customers and the members who referred them to the platform.
  - **ReferralDiscount**: An integer field indicating the discount or benefit provided to the customer as part of a referral program. It represents the incentive offered to customers for being referred by existing members.
  - **BMI** (Body Mass Index): An integer field storing the Body Mass Index (BMI) of the customer, if applicable. BMI is a measure of body fat based on height and weight, often used in health and fitness contexts.

- **Entity Name: Trainer**

  The "Trainer" table stores information about the trainers registered in the system. It includes details such as trainer identifiers, login information, specialization, ratings provided by clients, and feedback received from clients.

Trainer		
PK,FK	TrainerID	Int
	Name	String
	Specialization	String
FK	FeedbackID	String

Fig 3: Trainer Entity

Attributes:

- **TrainerID** (Primary Key, Foreign Key): An integer value serving as both the primary key and foreign key for the "Trainer" table. It uniquely identifies each trainer record in the database and also serves as a reference to the login identifier (loginId) from the "Login" table.
- **Name**: A string field storing the name of the customer. It represents the personal name or full name of the customer.
- **Specialization**: A string field storing the area of specialization or expertise of the trainer. It represents the specific domain or field in which the trainer provides services or instruction.
- **FeedbackID** (Foreign Key): An integer value referencing the feedback identifier (FeedbackID) from the "Feedback" table. It establishes a relationship between the "Trainer" table and the "Feedback" table, allowing each trainer to be associated with feedback provided by clients.

- **Entity Name: Feedback**

The "Feedback" entity stores feedback provided by customers regarding their experience with trainers. It includes details such as a unique identifier for each feedback entry, the identifiers of the customer and trainer involved, the feedback message, and the corresponding rating given by the customer.

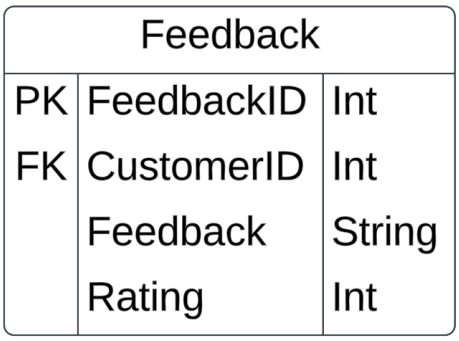

Fig 4: Feedback Entity

Attributes:

- **FeedbackID** (Primary Key): The "FeedbackID" attribute serves as the primary key for the "Feedback" entity, ensuring that each feedback entry is uniquely identifiable within the database.
- **CustomerID** (Foreign Key): The "CustomerID" attribute serves as a foreign key referencing the identifier of the customer providing the feedback, establishing a relationship between the "Feedback" entity and the "Customer" entity.

- **TrainerID** (Foreign Key): The "TrainerID" attribute serves as a foreign key referencing the identifier of the trainer who is the subject of the feedback, establishing a relationship between the "Feedback" entity and the "Trainer" entity.
- **Feedback**: A string field containing the feedback message provided by the customer. It includes comments, testimonials, or suggestions based on the customer's experience with the trainer.
- **Rating**: An integer field representing the rating given by the customer to the trainer. It serves as a quantitative measure of the customer's satisfaction or perception of the trainer's performance.

- **Entity Name: BMI**

The "BMI" entity stores information related to Body Mass Index (BMI) calculations for customers. It includes details such as a unique identifier for each BMI calculation entry, the identifier of the customer for whom the BMI and ultimately, the resulting BMI value.

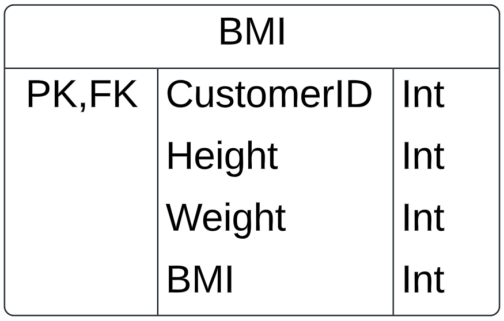

Fig 5: BMI Entity

Attributes:

- **CustomerID** (Primary Key, Foreign Key): An integer value serving as both the primary key and foreign key for the "BMI_Calc" entity. It uniquely identifies each BMI calculation record within the database and also serves as a reference to the identifier of the customer for whom the BMI is calculated in the "Customer" table.
- **Height**: An integer field representing the height measurement (in centimeters or inches) used in the BMI calculation.
- **Weight**: An integer field representing the weight measurement (in kilograms or pounds) used in the BMI calculation.
- **BMI**: An integer field representing the calculated Body Mass Index (BMI) value. BMI is calculated using the formula: BMI = (weight / (height * height)) * 703 (for measurements in inches and pounds) or BMI = (weight / (height * height)) (for measurements in meters and kilograms).

- **Entity Name: Membership**

  The "Membership" entity stores information related to the membership plans subscribed by customers. It includes details like the customer identifier linked with the membership, the type of membership plan, etc.

Membership		
	MembershipType	String
PK,FK	CustomerID	Int
	BaseAmount	Int
FK	PlanID	Int

Fig 6: Membership Entity

  Attributes:

  - **CustomerID** (PrimaryKey,Foreign Key): The "CustomerID" attribute serves as both primary key and foreign key referencing the identifier of the customer who subscribed to the membership plan in the "Customer" table.
  - **MembershipType**: A string field representing the type or category of the membership plan subscribed by the customer. It may include options such as "Basic," "Premium," or "Gold."
  - **BaseAmount**: An integer field representing the base amount or cost of the membership plan. It indicates the initial fee required for subscription to the membership.
  - **PlanId** (Foreign Key): The "PlanId" attribute serves as a foreign key referencing the plan identifier from the "Workout" table, indicating the specific workout plan associated with the membership.

- **Entity Name: WorkoutPlan**

  The "WorkoutPlan" entity stores information related to different workout plans offered within the system. It includes details such as a unique identifier for each plan, the name of the plan, the type of plan, the identifier of the trainer associated with the plan, and the amount or cost of the plan.

WorkoutPlan		
PK	PlanID	Int
	PlanName	String
	PlanType	String
FK	TrainerID	Int
	PlanAmount	Int

Fig 7: WorkoutPlan Entity

Attributes:

- **PlanId** (Primary Key): The "PlanId" attribute serves as the primary key for the "WorkoutPlan" entity, ensuring that each workout plan is uniquely identifiable within the database.
- **PlanName**: A string field representing the name or title of the workout plan. It describes the specific regimen or program included in the plan.
- **PlanType**: A string field representing the type or category of the workout plan. It may include options such as "Strength Training," "Cardiovascular," "Yoga," etc.
- **TrainerId** (Foreign Key): The "TrainerId" attribute serves as a foreign key referencing the identifier of the trainer associated with the workout plan in the "Trainer" table, establishing a relationship between the "WorkoutPlan" entity and the "Trainer" table.
- **PlanAmount**: An integer field representing the amount or cost associated with the workout plan. It indicates the price charged to customers who subscribe to the plan.

- **Entity Name: ExercisePlan**

The "ExercisePlan" entity stores information related to exercise plans tailored for individual customers within the system. It includes details such as a unique identifier for each exercise plan, references to the plan, customers associated with the exercise plan, name of the exercise plan, etc.

ExercisePlan		
PK	ExercisePlanID	Int
FK	PlanID	Int
FK	CustomerID	Int
	Sets	Int
	Repetitions	Int
	ExercisePlanName	String

Fig 8: ExercisePlan Entity

Attributes:

- **ExercisePlanId** (Primary Key): The "ExercisePlanId" attribute serves as the primary key for the "ExercisePlan" entity, ensuring that each exercise plan is uniquely identifiable within the database.
- **PlanId** (Foreign Key): The "PlanId" attribute serves as a foreign key referencing the identifier of the "WorkoutPlan" table. It establishes a relationship between the "ExercisePlan" and the "WorkoutPlan" table.

- **CustomerId** (Foreign Key): The "CustomerId" attribute serves as a foreign key referencing the identifier of the customer from the "Customer" table.It establishes a relationship between the "ExercisePlan" entity and the "Customer" table, indicating which customer the exercise plan is designed for.
- **ExercisePlanName**: A string field representing the name or title of the exercise plan. It describes the specific routine or program included in the exercise plan.
- **Sets**: An integer field representing the number of sets prescribed for each exercise within the exercise plan.
- **Repetitions**: An integer field representing the number of repetitions prescribed for each set within the exercise plan.

- **Entity Name: Exercises**

  The "Exercises" entity stores information related to individual exercises included in exercise plans. It includes details such as a unique identifier for each exercise, a reference to the exercise plan it belongs to, the name and description of the exercise, the difficulty level, and a video representation of the exercise stored as a Binary Large Object (BLOB).

Exercise		
PK	ExerciseID	Int
FK	ExercisePlanID	Int
	Name	String
	Description	String
	DiifficultyLevel	Int
	Video	BLOB

Fig 9: Exercise Entity

  Attributes:

  - **ExerciseID** (Primary Key): The "ExerciseID" attribute serves as the primary key for the "Exercises" entity, ensuring that each exercise is uniquely identifiable within the database.
  - **ExercisePlanId** (Foreign Key):The "ExercisePlanId" attribute serves as a foreign key referencing the identifier of the exercise plan from the "ExercisePlan" table.. It establishes a relationship between the "Exercises" entity and the "ExercisePlan" table, indicating which exercise plan the exercise belongs to.
  - **Name**: A string field representing the name or title of the exercise.
  - **Description**: A string field providing a brief description or instructions for performing the exercise.

- **Difficulty Level**: An integer field representing the difficulty level of the exercise. It may be represented on a scale (e.g., 1 for beginner, 2 for intermediate, 3 for advanced).
- **Video**: A Binary Large Object (BLOB) field storing the video representation of the exercise. It allows users to view visual demonstrations of how to perform the exercise correctly.

- **Entity Name: Session**

  The "Session" entity stores information related to workout sessions scheduled for individual customers within the system and details like: a unique identifier for each session, references to the customer and workout plan associated with the session.

Session		
PK	SessionID	Int
FK	CustomerID	Int
FK	PlanID	Int

Fig 10: Session Entity

  Attributes:

  - **SessionID** (Primary Key): The "SessionID" attribute serves as the primary key for the "Session" entity, ensuring that each workout session is uniquely identifiable.
  - **CustomerID** (Foreign Key): The "CustomerID" attribute serves as a foreign key referencing the identifier of the customer from the "Customer" table. It establishes a relationship between the "Session" entity and the "Customer" table, indicating which customer the session is scheduled for.
  - **PlanID** (Foreign Key): The "PlanID" attribute serves as a foreign key referencing the identifier of the workout plan from the "WorkoutPlan" table. It establishes a relationship between the "Session" entity and the "WorkoutPlan" table, indicating which workout plan the session is associated with.

- **Entity Name: NutritionPlan**

  The "NutritionPlan" entity stores information related to nutrition plans within the system along with a description outlining the details and guidelines of the nutrition plan.

NutritionPlan		
PK, FK	PlanID	Int
	Description	String

Fig 11: NutritionPlan Entity

Attributes:

- **PlanID** (Primary Key, Foreign Key): The "PlanID" attribute serves as the primary key as well as foreign key for the "NutritionPlan" entity, ensuring that each nutrition plan is uniquely identifiable within the database.It uniquely identifies each nutrition plan within the database and also serves as a reference to the identifier of the nutrition plan from other related entities, if applicable.
- **Description**: A string field containing a detailed description of the nutrition plan. It outlines the dietary guidelines, meal plans, nutritional requirements, and any other relevant information for following the plan.

- **Entity Name: Payment**

The "Payment" entity stores information related to payments made by customers for their membership plans within the system. It includes details such as a unique identifier for each payment, references to the customer, membership plan amount, base amount, referral discount, and the calculated final amount of the payment.

Payment		
PK	PaymentID	Int
FK	CustomerID	String
	PlanAmount	Int
	BaseAmount	Int
	Referral Discount	Int
	FinalAmount	Int

Fig 12: Payment Entity

Attributes:

- **PaymentId** (Primary Key): The "PaymentId" attribute serves as the primary key for the "Payment" entity, ensuring that each payment is uniquely identifiable within the database.
- **CustomerId** (Foreign Key): The "CustomerId" attribute serves as a foreign key referencing the identifier of the customer from the "Customer" entity.It establishes a relationship between the "Payment" entity and the "Customer" entity, indicating which customer made the payment.
- **PlanAmount** (Foreign Key):The "PlanAmount" attribute serves as a foreign key referencing the amount of the workout plan from the "WorkoutPlan" entity. It establishes a relationship between the "Payment" entity and the "WorkoutPlan" entity, indicating the amount associated with the workout plan for which the payment is made.

- **BaseAmount** (Foreign Key):The "BaseAmount" attribute serves as a foreign key referencing the base amount from the "Membership" entity. It establishes a relationship between the "Payment" entity and the "Membership" entity, indicating the base amount of the membership plan for which the payment is made.
- **ReferralDiscount** (Foreign Key):The "ReferralDiscount" attribute serves as a foreign key referencing the referral discount from the "Customer" entity. It establishes a relationship between the "Payment" entity and the "Customer" entity, indicating the discount received by the customer through a referral program.
- **FinalAmount**: An integer field representing the calculated final amount of the payment. It is calculated based on the sum of the plan amount, base amount, and referral discount.

- **Entity Name: Feedback_Trainer_Association**

The “Feedback_Trainer_Association” entity is a junction table that facilitates the many-to-many relationship between Feedback and Trainer tables by storing associations between specific feedback entries and trainers.

Feedback_Trainer_Association		
PK, FK	FeedbackID	Int
PK, FK	TrainerID	Int

Fig 13: The Associative Entity

Attributes:

- **FeedbackID, TrainerID** (Primary Key): The primary key constraint is applied to a combination of columns which ensures that each combination of FeedbackID and TrainerID is unique within the table.
- **FeedbackID, TrainerID** (Foreign Key): Two foreign key constraints are defined, one for FeedbackID referencing the Feedback table, and another for TrainerID referencing the Trainer table. These foreign key constraints ensure referential integrity, meaning that the values in the FeedbackID and TrainerID columns must exist in the respective referenced tables and enforce the relationship between Feedback_Trainer_Association and the Feedback and Trainer tables.

Relationships:

- **Mandatory One-to-Mandatory Many Relationship between Customer and Payment**

  Each customer can make one or multiple payments.

  Each payment is made by one customer.

- **Mandatory One-to-Optional Many Relationship between Customer and Feedback**

  Each customer can provide zero or multiple feedback.

  Each feedback is provided by one customer only.

- **Mandatory One-to-Mandatory One Relationship between Customer and Membership**

  Each customer can have only one membership.

  Each membership is associated with one customer.

- **Mandatory One-to-Mandatory One Relationship between Customer and BMI**

  Each customer can have multiple BMI calculations over time.

  Each BMI calculation is associated with only one customer.

- **Mandatory one -to- Mandatory Many Relationship between WorkoutPlan and Membership**

  Each workout plan can include multiple membership plans.

  Each membership plan can be associated with only one workout plan.

- **Mandatory One-to-Mandatory Many Relationship between WorkoutPlan and ExercisePlan**

  Each workout plan can have multiple exercise plans.

  Each exercise plan belongs to one workout plan.

- **Many-to-Many Relationship between ExercisePlan and Exercises**

  Each exercise plan can consist of multiple exercises.

  Each exercise belongs to multiple exercise plans.

- **Mandatory One-to-Mandatory Many Relationship between WorkoutPlan and Session**

  Each workout plan can have multiple sessions.

  Each session is associated with one workout plan.

- **Optional Many-to-Optional Many Relationship between Customer and Trainer (Indirectly through Feedback)**

  Each customer can provide feedback for zero or multiple trainers.

  Each trainer can receive feedback from zero or multiple customers.
- **Mandatory One-to-Mandatory Many Relationship between Trainer and WorkoutPlan**

  Each trainer can be associated with multiple workout plans.

  Each workout plan can be associated with one trainer.
- **Mandatory Many-to-Mandatory One Relationship between NutritionPlan and WorkoutPlan**

  Each workout plan can have one associated nutrition plan.

  Each nutrition plan can be associated with one or more workout plans.
- **Unary Relationship between CustomerID and ReferralID**

  CustomerID: This attribute represents the unique identifier of a customer within the "Customer" entity. It serves as the primary key for identifying individual customers.

  ReferralID: This attribute represents the unique identifier of the referring customer within the "Customer" entity. It refers to another instance of the "Customer" entity, indicating that one customer can refer another customer to the system.
- **Disjoint Rule between Login and Customer or Trainer**

  In our database, the relationship between individuals and their respective roles as customers or trainers is defined by a disjoint rule. It ensures that each individual in the system is uniquely classified as either a customer or a trainer, but not both, based on the "type" attribute in the login table. This organization maintains clarity and prevents ambiguity in the system's data model.
- **Associative entity between Feedback and Trainer**

  The "Feedback_Trainer_Association" entity serves as an essential bridge between customers providing feedback and trainers receiving it within our database. Acting as an associative entity, it establishes a many-to-many relationship between trainer and feedback.

- **Entity Clustering**

  In our database design, we implemented entity clustering to effectively organize related tables and optimize access patterns for different membership tiers gold and silver. For enhancing the data management we have divided the entities into separate clusters based on offline and online activities.

  - Sessions (Personal Cluster) - Only for Gold Membership
  - ExercisePlan and Exercise (Online Cluster) - For both Gold and Silver Memberships

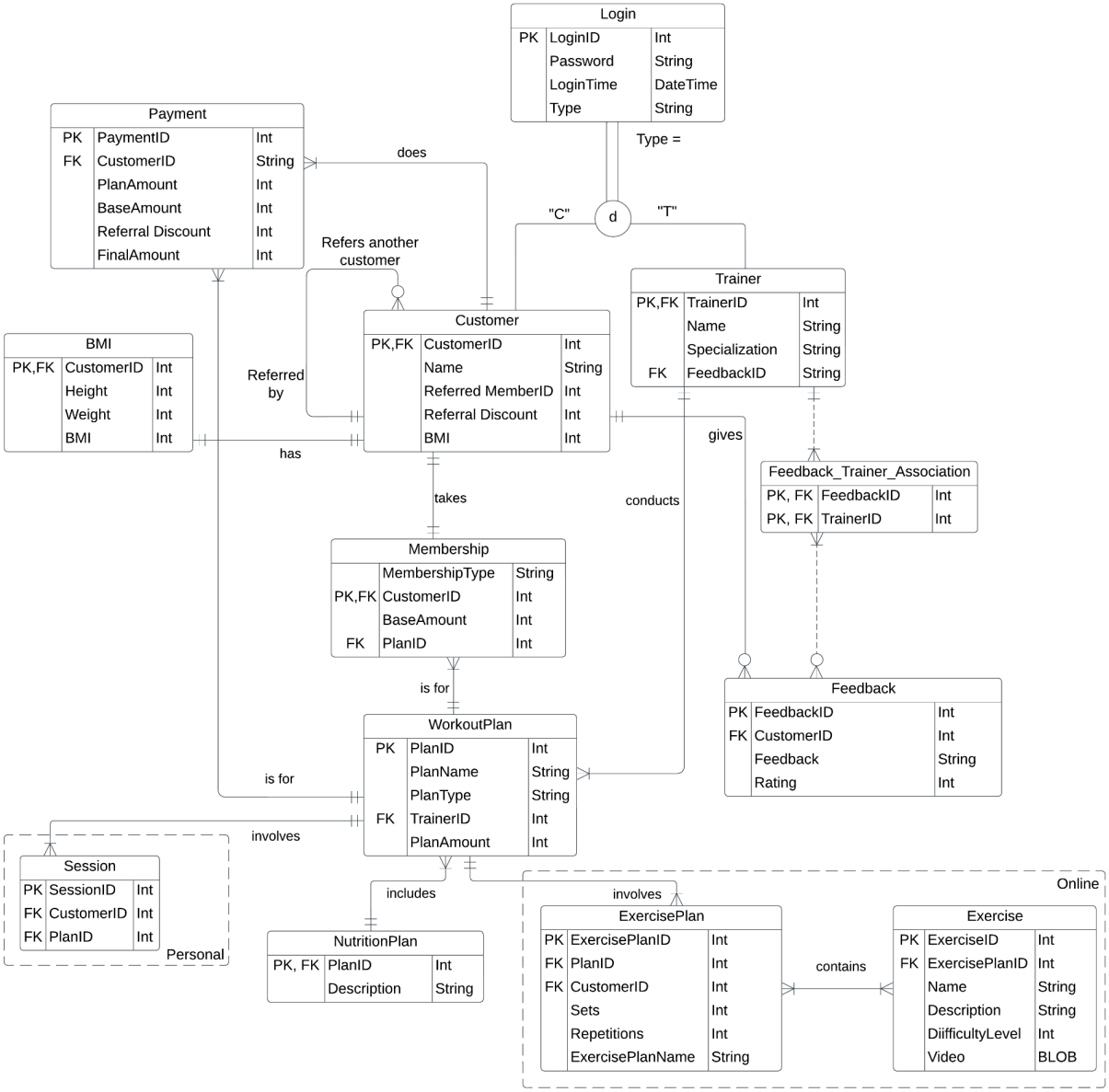

Fig 14: The EER Model

TRANSFORMATION TO RELATIONAL MODEL

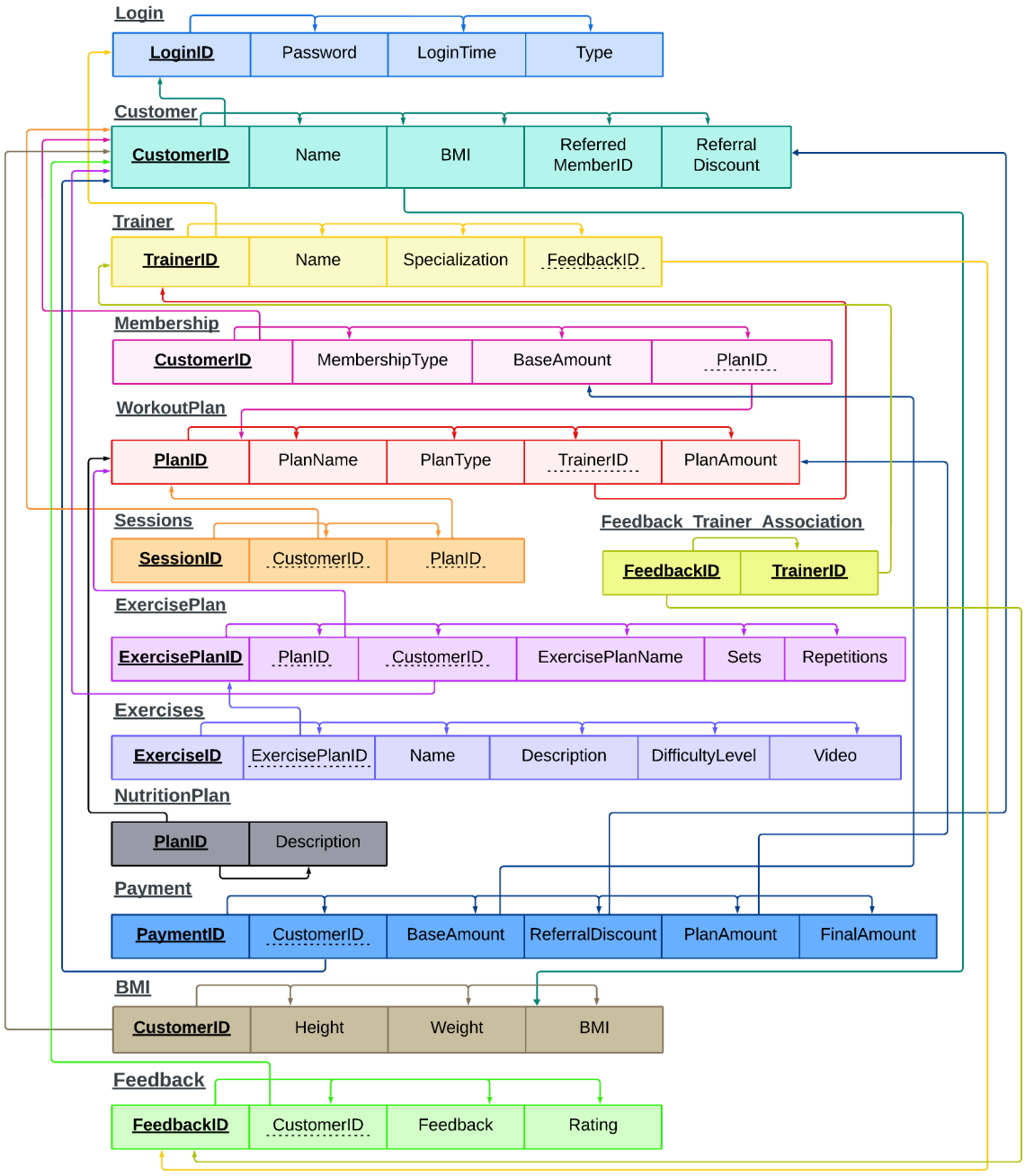

Fig 15: The Relational Model

Steps for Normalization:

- Converting into 1NF: The goal of 1NF is to ensure that the values in each column of a table are atomic, meaning they cannot be divided further. In 1NF, a table is organized so that it contains only simple, indivisible data values, and each column represents a single attribute.

1) In the original WorkoutPlan table, a variety of information related to workout plans, sessions, individual exercises and nutrition plans were stored together. By decomposing the original table into five separate tables, each with a specific focus on one aspect of the data—WorkoutPlan, Sessions, ExercisePlan, Exercise and NutritionPlan—it became possible to eliminate redundant information and organize the data more efficiently.

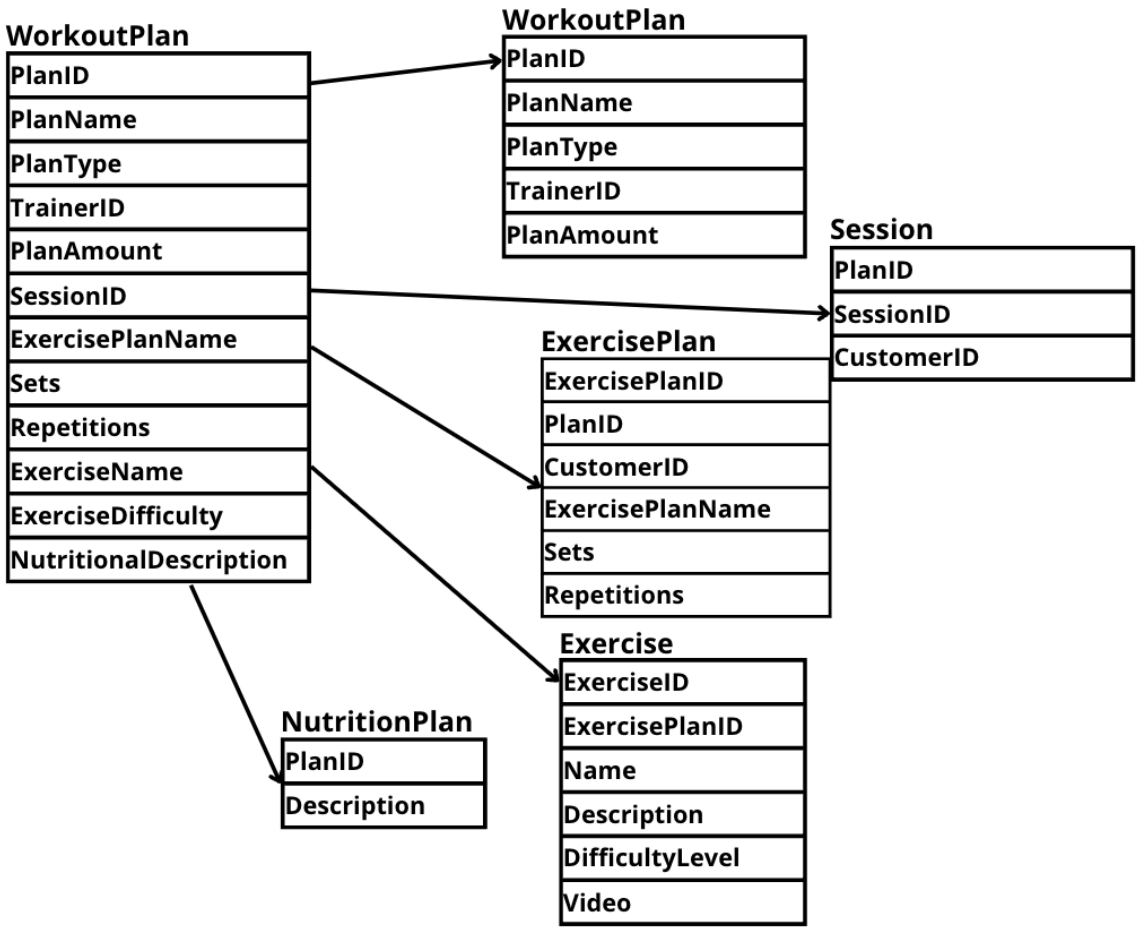

Fig 16: WorkoutPlan 1NF

2) In the original schema, where BMI was a non-atomic attribute within the Customer table, the BMI value was derived using Height and Weight. However, this violated the principles of 1NF, which requires that each column contains atomic, indivisible values. To address this, a new table named BMI was introduced, with individual attributes such as CustomerID, Height, Weight, and BMI.

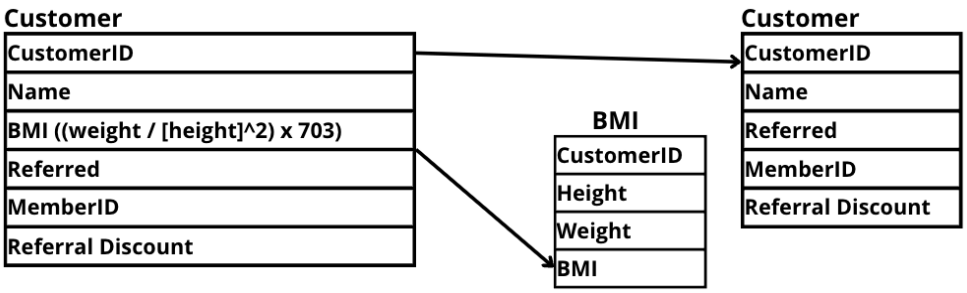

Fig 17: BMI 1NF

- Converting into 2NF: The goal of Second Normal Form (2NF) involves ensuring that all non-prime attributes are fully functionally dependent on the primary key. The primary key should uniquely identify each record, and no partial dependencies should exist.
  1) Initially, the relational schema included the "Payments" information as part of the "Customer" table. However, to enhance the design and adhere to the principles of Second Normal Form (2NF), we introduced a separate entity named "Payment."

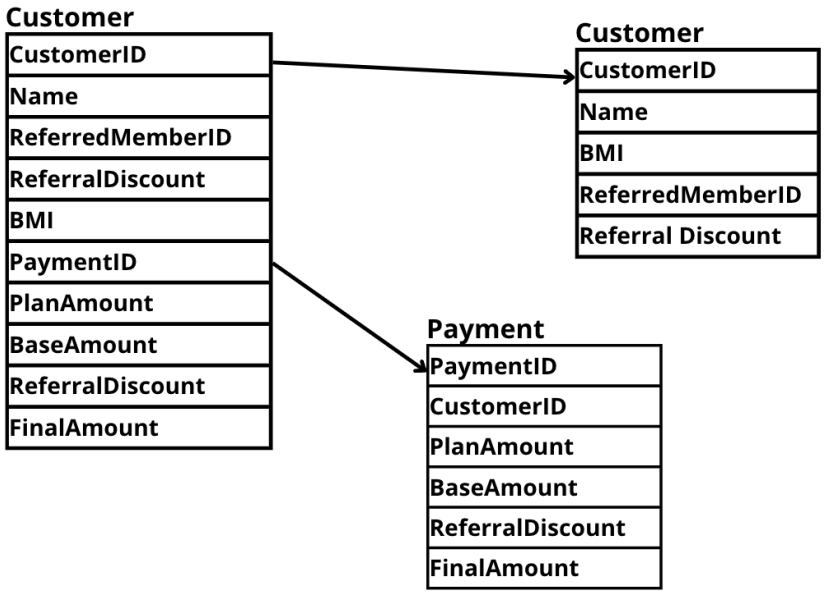

Fig 18: Payment 2NF

  2) By creating a separate Membership table, redundancy is eliminated. Membership information is now stored in one place instead of being duplicated in both the Customer and WorkoutPlan tables. Foreign key relationships ensure that each Membership entry is associated with a valid customer and workout plan.

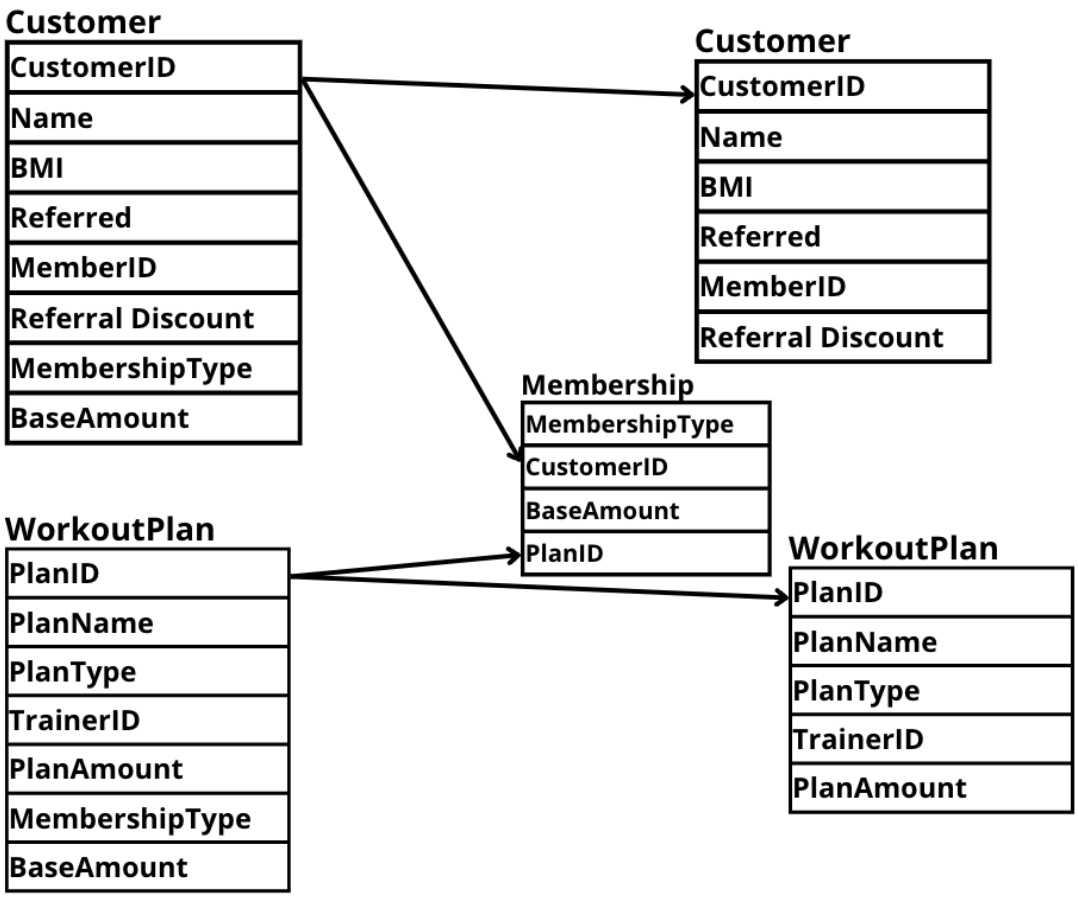

Fig 19: Membership 2NF

- Converting into 3NF: Third Normal Form (3NF) is used in database design to further refine the structure of a relational database and eliminate transitive dependencies. The primary goals of achieving 3NF are to minimize data redundancy, enhance data integrity, and promote a more maintainable and flexible database schema.

  1) In our original schema, the Customers table included the Feedback attribute, which caused deletion anomalies. So, if a customer provides feedback but if that customer leaves (deleting a customer entry) the associated feedback would also get deleted. To solve this, we created a new table called feedback where all the feedback attributes are stored together separately. However, due to this, now the trainer table and feedback table were directly referencing each other's foreign keys which led to data inconsistency. Hence, to resolve this issue, we introduced an associative entity called Feedback_Trainer_Association to manage the relationship between feedback and trainers more efficiently.

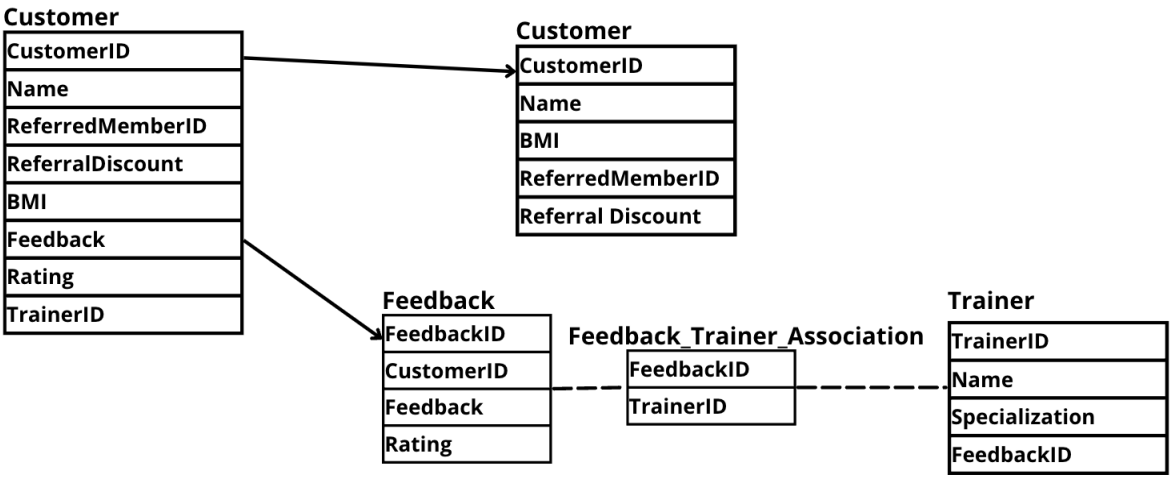

Fig 20: Feedback 3NF

IMPLEMENTATION

The SQL file attached in the folder contains comprehensive implementations for all the necessary queries related to our tables. Not only have we included commands for creating and populating the tables, but we've also incorporated various queries to view feedback specific to particular trainers, top 3 trainers with the highest average rating, highest-rated feedback, etc.

APPLICATION

Potential Issues:

- Data Integrity: There might be instances where data integrity could be compromised, such as entering invalid customer IDs or incorrect references between tables.
- Security: Storing passwords as plain text in the `Login` table could pose a security risk if the database is breached.
- Incomplete Data: Some tables, like the `Trainer` table, allow inserting records without specifying a `FeedbackID`, which could lead to incomplete data or referential integrity issues.
- Scalability: The database design might face scalability issues if the volume of data increases significantly over time, affecting query performance and system responsiveness.

Measures or Features to Address Issues:

- Data Validation: Implement strict data validation rules at the application level to ensure that only valid data is entered into the database, reducing the risk of data integrity issues.
- Hashed Passwords: Store passwords securely by hashing them before storing in the database. Additionally, consider implementing encryption for sensitive data.
- Foreign Key Constraints: Enforce foreign key constraints in all related tables to maintain referential integrity and prevent incomplete data entries.
- Indexes and Query Optimization: Utilize appropriate indexing strategies and optimize queries to enhance the database's performance, particularly concerning scalability concerns.
- Role-Based Access Control (RBAC): Introduce RBAC to manage user roles and permissions effectively, ensuring that users only have access to the data and functionalities they require.
- Regular Backups: Implement automated backup routines to regularly backup the database, reducing the risk of data loss in case of system failures or security breaches.
- Logging and Auditing: Implement logging and auditing mechanisms to track user activities and changes made to the database, aiding in troubleshooting and ensuring accountability.

Administration Functions and Processes:

- Database Monitoring: Regularly monitor database performance and usage metrics to identify potential issues and optimize resource utilization.
- Security Patching: Stay up-to-date with security patches and updates for the database management system and associated software to mitigate security vulnerabilities.
- Backup and Recovery Procedures: Establish clear backup and recovery procedures to restore data in case of data loss or corruption, including testing backup integrity regularly.
- Routine Maintenance: Schedule routine maintenance tasks such as index rebuilds, statistics updates, and database reorganizations to optimize performance and ensure data integrity.