Nutrition and Wellness Tracking Application

(Satya Sai Srinivas Yendru)

College Of Professional Studies, Northeastern University

ITC 6000: Database Management Systems

Final Project Report

Prof: Sammuel Li

December 12,2024

Introduction

The Nutrition and Wellness Tracking App is designed to help users maintain a healthy lifestyle by tracking their dietary intake, exercise activities, and wellness goals. The primary use case for this application involves users logging their daily food intake, water consumption, exercise routines, and tracking progress toward wellness goals such as weight loss, improved fitness, or better nutrition.

The app employs a "freemium" cost model, allowing basic functionalities such as daily logging and progress tracking to be free, while premium features like personalized recommendations from nutrition coaches and advanced analytics are available for a subscription fee.

My personal connection to this app stems from an interest in leveraging data to promote healthier lifestyles. The app's use of structured data to provide actionable insights aligns well with my career focus in database design and cloud computing. This project offers an opportunity to integrate my technical expertise with a real-world application.

Business Analysis

User Personas

1. **Primary User (Health Enthusiast):** Health enthusiasts are individuals aged 25-40 who are highly motivated to maintain or improve their health. They actively use the app to log meals, track calories, monitor exercise routines, and review progress reports. This persona is highly tech-savvy and appreciates features like personalized insights, visual data summaries, and goal-setting capabilities. They rely on the app to gain a structured and data-driven approach to wellness.
2. **Nutrition Coach:** Nutrition coaches are certified dietitians or fitness experts who provide personalized guidance to users. They use the app to review user logs, analyze trends in dietary and exercise data, and offer tailored recommendations to improve health outcomes. Coaches also track user progress and update their guidance based on logged data. This persona needs robust data visualization and reporting tools to make informed recommendations.
3. **Admin:** Admins are technical or managerial stakeholders responsible for maintaining the app's infrastructure. They ensure the smooth functioning of the database and application, resolve technical issues, and monitor system performance. Admins interact with the app primarily through backend tools to manage user accounts, enforce data security, and optimize system performance.
4. **Casual User (Beginner):** Casual users are individuals aged 18-25 who are just beginning their journey toward a healthier lifestyle. They may log data sporadically and explore various features of the app, such as tracking calorie intake and receiving basic

recommendations. This persona values simplicity and ease of use, as they are less experienced with health-tracking tools and may require motivational prompts to remain engaged.

Business Rules and Logic:

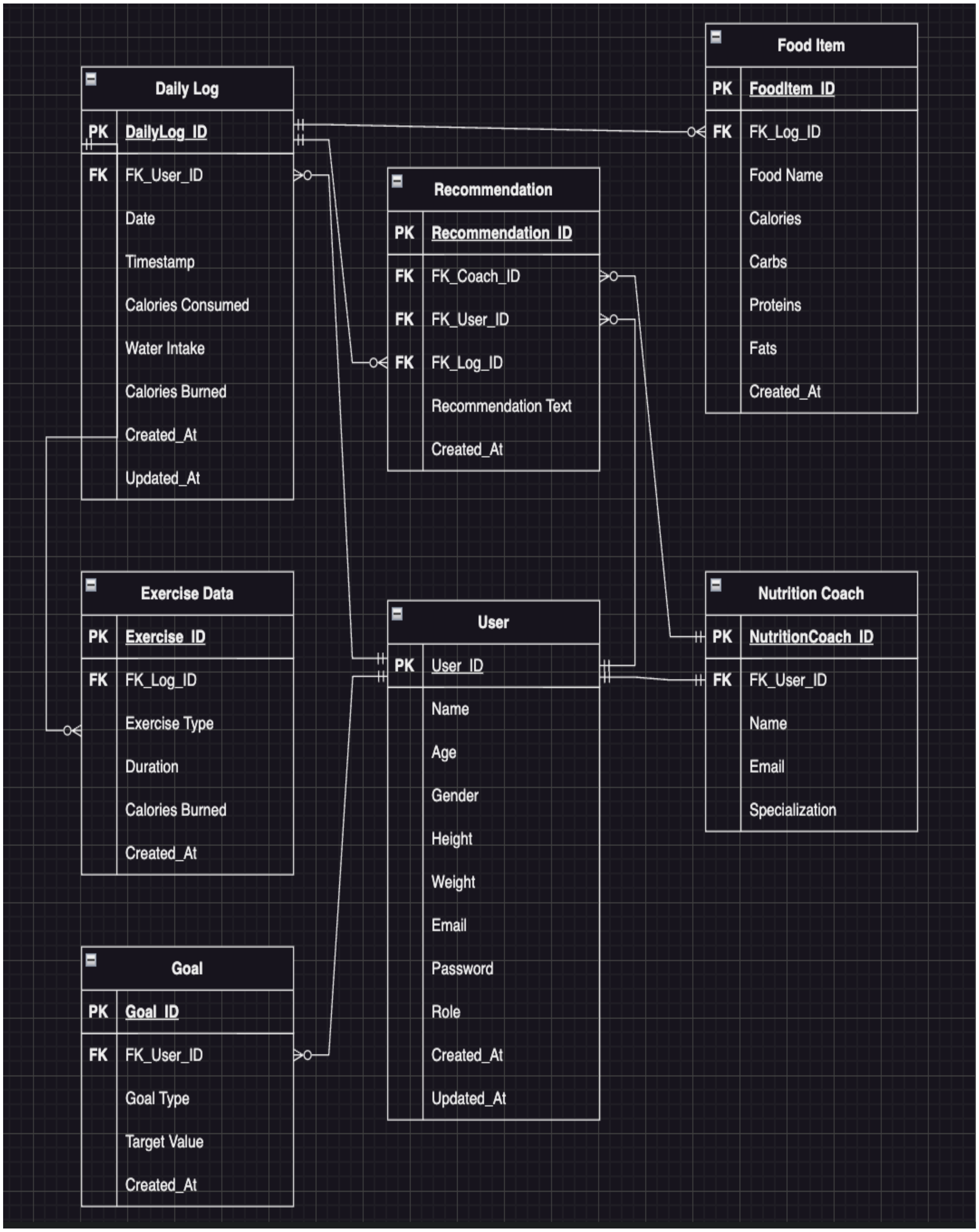
To ensure a seamless user experience and maintain data integrity, the following business rules are implemented:

- User Registration:
  Users must register with a unique email and password to access the app.
  Basic profile information such as name, age, gender, height, and weight is required for personalized insights.
- Daily Logs:
  Users can create daily logs to record food items, water intake, and exercise activities.
  Each log is timestamped to associate it with a specific date.
- Food Items and Exercises:
  Users can log multiple food items and exercises per day.
  Each food item entry includes details such as calories, macronutrients (carbs, proteins, fats), and serving size.
  Exercise entries track activity type, duration, and calories burned.
- Goal Setting:
  Users can set goals for metrics like weight, water intake, and calories burned.
  Goals are linked to users and are tracked over time.
- Recommendations:
  Nutrition coaches can provide recommendations based on user data.
  Recommendations are linked to specific logs, users, and coaches.
- Data Access Control:
  Users can view and update their own data.
  Nutrition coaches can access assigned user logs but cannot edit user data directly.
  Admins have system-wide access for maintenance and support purposes.
- Reporting and Analytics:
  The app generates progress reports, goal achievement summaries, and insights into dietary and exercise trends.
  Analytics features include aggregating data across logs to calculate averages, trends, and adherence rates.
- Data Integrity and Security:
  Triggers ensure timestamps are updated whenever records are modified.
  Foreign keys enforce relationships between tables, preventing orphaned records.
  Indexed fields improve query performance for common operations, such as retrieving logs by user ID.

By adhering to these rules, the app ensures a reliable and user-friendly experience while maintaining the integrity of stored data.

Table Design and Analysis

Entity Relationship Diagram (ERD)



The database comprises the following tables:

1. User Table: Captures user demographics and credentials.
2. NutritionCoach Table: Stores information about nutrition coaches linked to users.
3. DailyLog Table: Records daily user activities, including food intake and exercises.
4. FoodItem Table: Logs individual food items consumed by users.
5. ExerciseData Table: Logs exercise activities performed by users.
6. Goal Table: Tracks wellness goals set by users.
7. Recommendation Table: Stores coach recommendations based on user activities.

Relationships between the Entities:

1. User to NutritionCoach:
   1-to-1 relationship (via User_ID as FK in NutritionCoach).
2. User to DailyLog:
   1-to-Many relationship (via User_ID as FK in DailyLog).
3. User to Goal:
   1-to-Many relationship (via User_ID as FK in Goal).
4. User to Recommendation:
   1-to-Many relationship (via User_ID as FK in Recommendation).
5. NutritionCoach to Recommendation:
   1-to-Many relationship (via NutritionCoach_ID as FK in Recommendation).
6. DailyLog to FoodItem:
   1-to-Many relationship (via DailyLog_ID as FK in FoodItem).
7. DailyLog to ExerciseData:
   1-to-Many relationship (via DailyLog_ID as FK in ExerciseData).
8. DailyLog to Recommendation:
   1-to-Many relationship (via DailyLog_ID as FK in Recommendation).

Table1: User

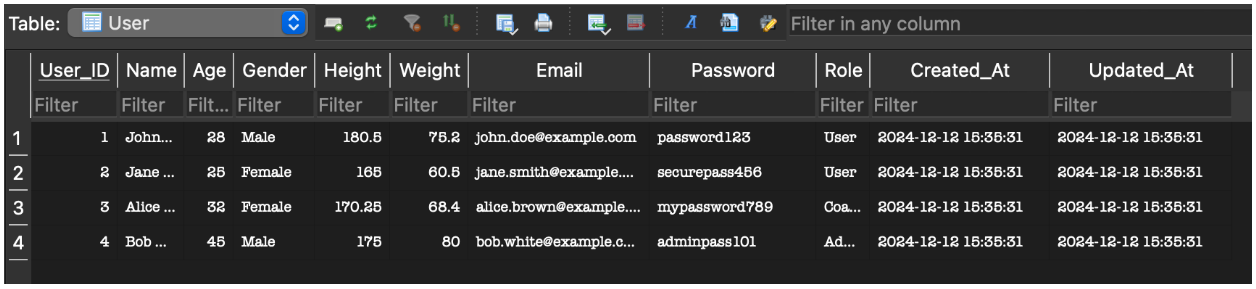


Table: User

Filter in any column

	User_ID	Name	Age	Gender	Height	Weight	Email	Password	Role	Created_At	Updated_At
	Filter	Filter	Filt...	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	John...	28	Male	180.5	75.2	john.doe@example.com	password123	User	2024-12-12 15:35:31	2024-12-12 15:35:31
2	2	Jane ...	25	Female	165	60.5	jane.smith@example....	securepass456	User	2024-12-12 15:35:31	2024-12-12 15:35:31
3	3	Alice ...	32	Female	170.25	68.4	alice.brown@example....	mypassword789	Coa...	2024-12-12 15:35:31	2024-12-12 15:35:31
4	4	Bob ...	45	Male	175	80	bob.white@example.c...	adminpass101	Ad...	2024-12-12 15:35:31	2024-12-12 15:35:31

Table2: Nutrition Coach

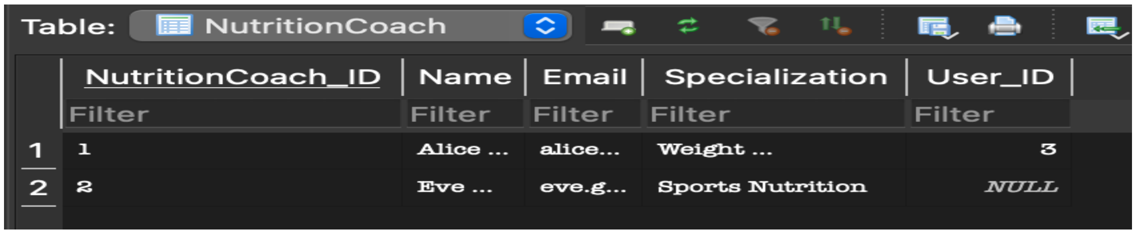
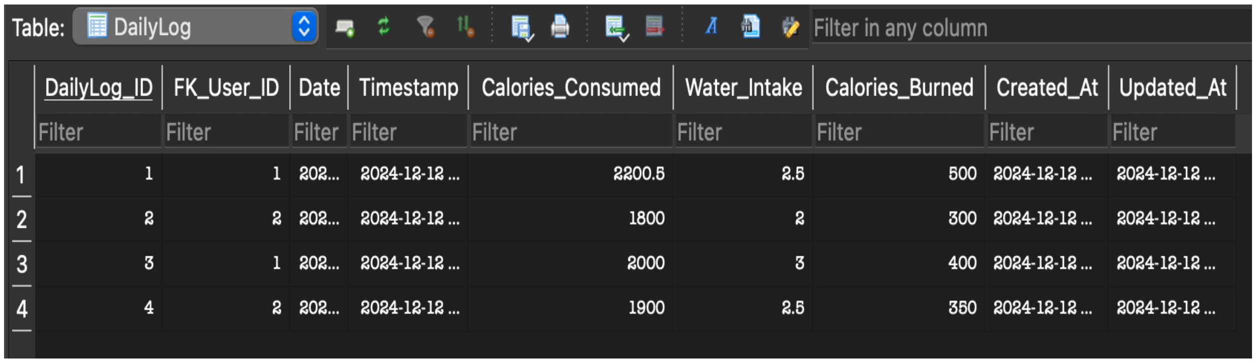


Table: NutritionCoach

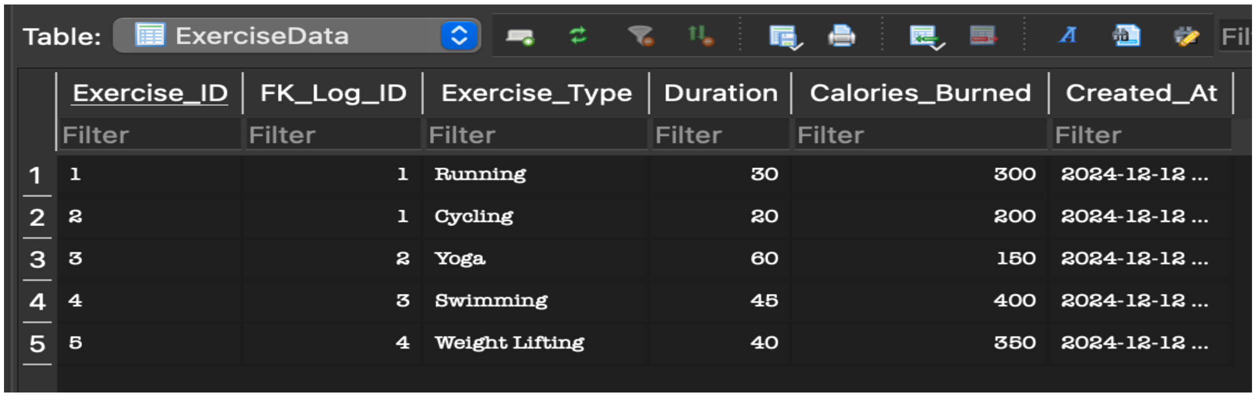
	NutritionCoach_ID	Name	Email	Specialization	User_ID
	Filter	Filter	Filter	Filter	Filter
1	1	Alice ...	alice...	Weight ...	3
2	2	Eve ...	eve.g...	Sports Nutrition	*NULL*

Table2: DailyLog



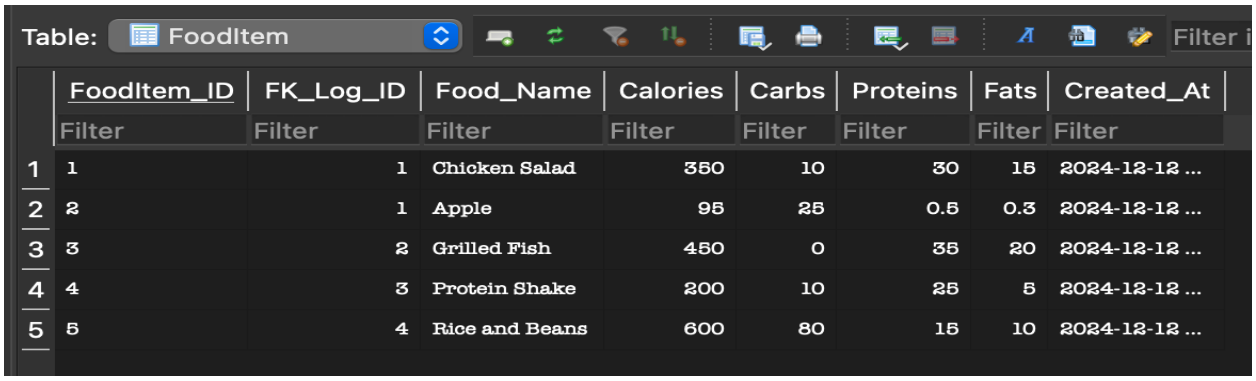
	DailyLog_ID	FK_User_ID	Date	Timestamp	Calories_Consumed	Water_Intake	Calories_Burned	Created_At	Updated_At
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	202...	2024-12-12 ...	2200.5	2.5	500	2024-12-12 ...	2024-12-12 ...
2	2	2	202...	2024-12-12 ...	1800	2	300	2024-12-12 ...	2024-12-12 ...
3	3	1	202...	2024-12-12 ...	2000	3	400	2024-12-12 ...	2024-12-12 ...
4	4	2	202...	2024-12-12 ...	1900	2.5	350	2024-12-12 ...	2024-12-12 ...

Table4: Food Item



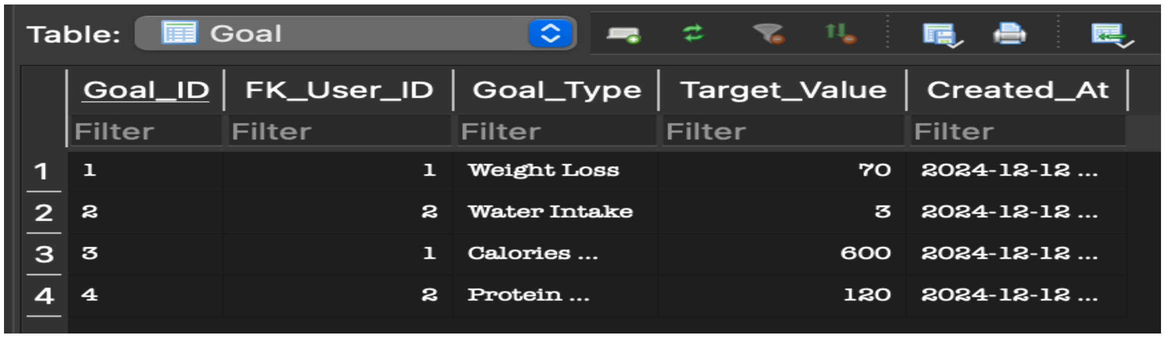
	Exercise_ID	FK_Log_ID	Exercise_Type	Duration	Calories_Burned	Created_At
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	Running	30	300	2024-12-12 ...
2	2	1	Cycling	20	200	2024-12-12 ...
3	3	2	Yoga	60	150	2024-12-12 ...
4	4	3	Swimming	45	400	2024-12-12 ...
5	5	4	Weight Lifting	40	350	2024-12-12 ...

Table5: Exercise Data



	FoodItem_ID	FK_Log_ID	Food_Name	Calories	Carbs	Proteins	Fats	Created_At
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	Chicken Salad	350	10	30	15	2024-12-12 ...
2	2	1	Apple	95	25	0.5	0.3	2024-12-12 ...
3	3	2	Grilled Fish	450	0	35	20	2024-12-12 ...
4	4	3	Protein Shake	200	10	25	5	2024-12-12 ...
5	5	4	Rice and Beans	600	80	15	10	2024-12-12 ...

Table6: Goal



	Goal_ID	FK_User_ID	Goal_Type	Target_Value	Created_At
	Filter	Filter	Filter	Filter	Filter
1	1	1	Weight Loss	70	2024-12-12 ...
2	2	2	Water Intake	3	2024-12-12 ...
3	3	1	Calories ...	600	2024-12-12 ...
4	4	2	Protein ...	120	2024-12-12 ...

Table7: Recommendation

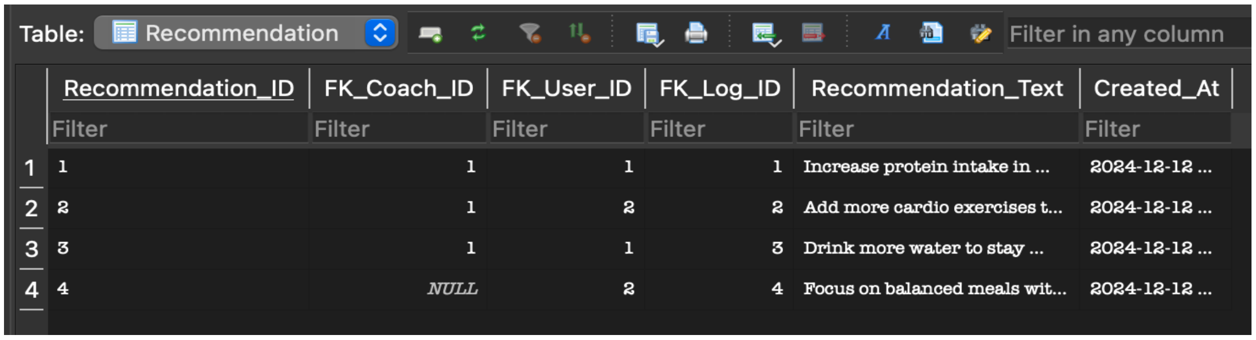


Table: Recommendation | Filter in any column

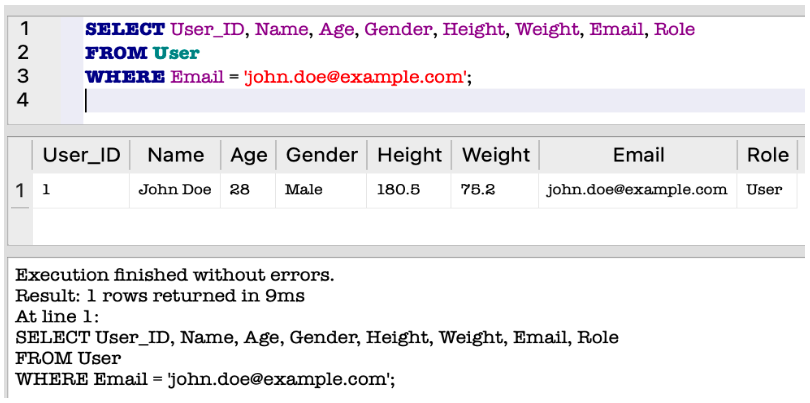
	Recommendation_ID	FK_Coach_ID	FK_User_ID	FK_Log_ID	Recommendation_Text	Created_At
1	1	1	1	1	Increase protein intake in ...	2024-12-12 ...
2	2	1	2	2	Add more cardio exercises t...	2024-12-12 ...
3	3	1	1	3	Drink more water to stay ...	2024-12-12 ...
4	4	NULL	2	4	Focus on balanced meals wit...	2024-12-12 ...

Database Implementation

User Story 1: A User Registers for the App

Description: A new user registers and starts logging their daily activities. This involves inserting user information, creating a daily log, and adding food and exercise data.

Query 1: Retrieve User Details After Registration



```
SELECT User_ID, Name, Age, Gender, Height, Weight, Email, Role
FROM User
WHERE Email = 'john.doe@example.com';
```

	User_ID	Name	Age	Gender	Height	Weight	Email	Role
1	1	John Doe	28	Male	180.5	75.2	john.doe@example.com	User

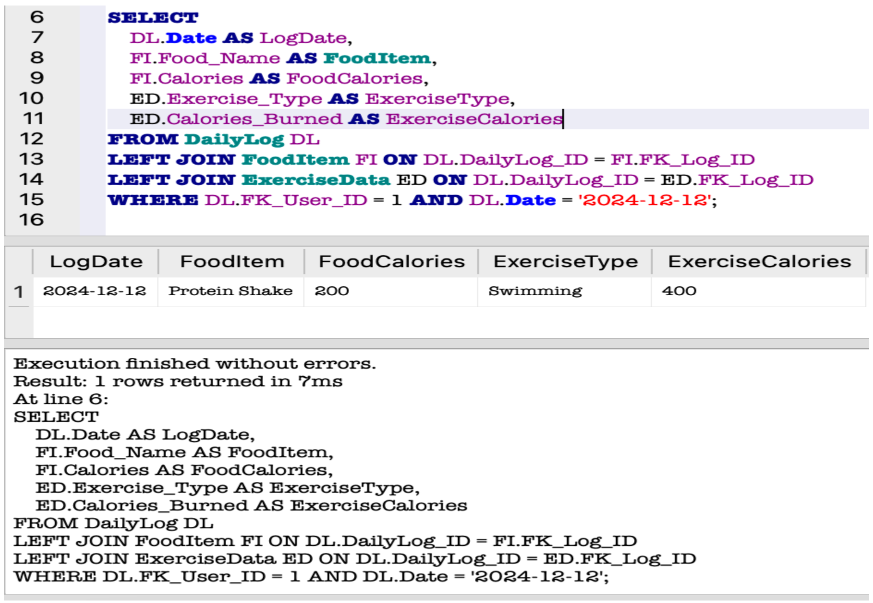
```
Execution finished without errors.
Result: 1 rows returned in 9ms
At line 1:
SELECT User_ID, Name, Age, Gender, Height, Weight, Email, Role
FROM User
WHERE Email = 'john.doe@example.com';
```

Explanation: This query fetches details of a user who recently registered using their email. The email serves as a unique identifier to verify registration.

User Story 2: A User Logs Their Daily Activities

Description: The user logs daily food items and exercises. The following queries demonstrate retrieving detailed logs and joining multiple tables to provide a comprehensive overview.

Query 2: Retrieve Comprehensive Daily Log for a User



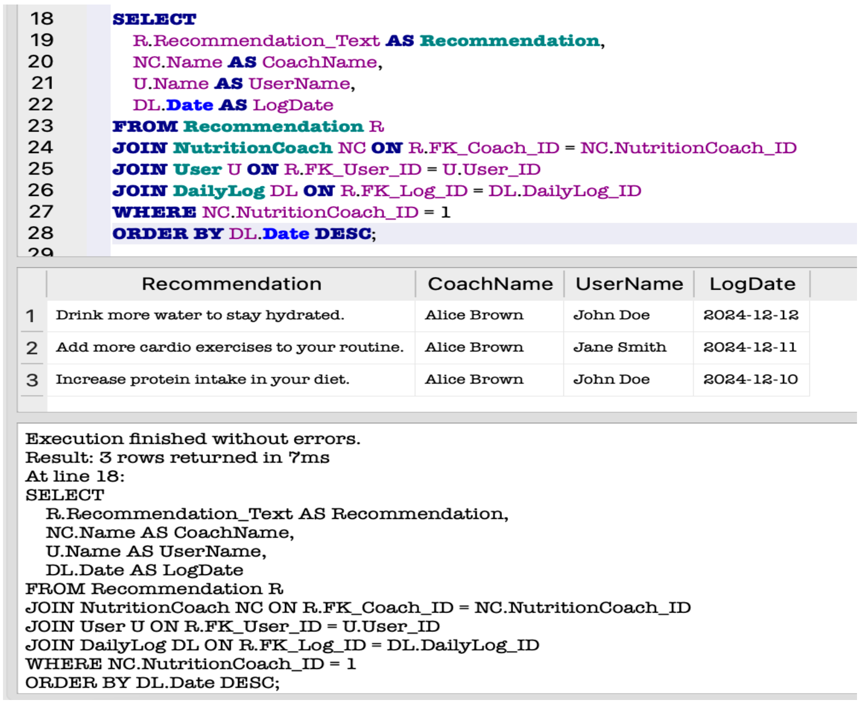
```sql
SELECT
   DL.Date AS LogDate,
   FI.Food_Name AS FoodItem,
   FI.Calories AS FoodCalories,
   ED.Exercise_Type AS ExerciseType,
   ED.Calories_Burned AS ExerciseCalories
FROM DailyLog DL
LEFT JOIN FoodItem FI ON DL.DailyLog_ID = FI.FK_Log_ID
LEFT JOIN ExerciseData ED ON DL.DailyLog_ID = ED.FK_Log_ID
WHERE DL.FK_User_ID = 1 AND DL.Date = '2024-12-12';
```

	LogDate	FoodItem	FoodCalories	ExerciseType	ExerciseCalories
1	2024-12-12	Protein Shake	200	Swimming	400

```
Execution finished without errors.
Result: 1 rows returned in 7ms
At line 6:
SELECT
   DL.Date AS LogDate,
   FI.Food_Name AS FoodItem,
   FI.Calories AS FoodCalories,
   ED.Exercise_Type AS ExerciseType,
   ED.Calories_Burned AS ExerciseCalories
FROM DailyLog DL
LEFT JOIN FoodItem FI ON DL.DailyLog_ID = FI.FK_Log_ID
LEFT JOIN ExerciseData ED ON DL.DailyLog_ID = ED.FK_Log_ID
WHERE DL.FK_User_ID = 1 AND DL.Date = '2024-12-12';
```

Explanation: This query joins the DailyLog, FoodItem, and ExerciseData tables to retrieve all logged activities for a specific user on a given date.

User Story 3: A Coach Provides Recommendations
Description: Nutrition coaches analyze user data and offer personalized recommendations.
Query 3: Fetch Recommendations by a Coach



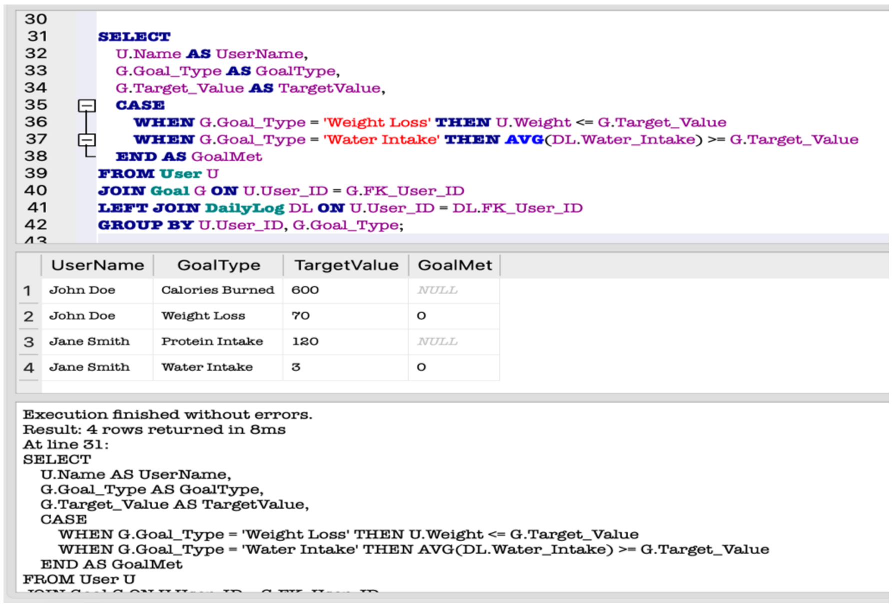
```sql
SELECT
   R.Recommendation_Text AS Recommendation,
   NC.Name AS CoachName,
   U.Name AS UserName,
   DL.Date AS LogDate
FROM Recommendation R
JOIN NutritionCoach NC ON R.FK_Coach_ID = NC.NutritionCoach_ID
JOIN User U ON R.FK_User_ID = U.User_ID
JOIN DailyLog DL ON R.FK_Log_ID = DL.DailyLog_ID
WHERE NC.NutritionCoach_ID = 1
ORDER BY DL.Date DESC;
```

	Recommendation	CoachName	UserName	LogDate
1	Drink more water to stay hydrated.	Alice Brown	John Doe	2024-12-12
2	Add more cardio exercises to your routine.	Alice Brown	Jane Smith	2024-12-11
3	Increase protein intake in your diet.	Alice Brown	John Doe	2024-12-10

```
Execution finished without errors.
Result: 3 rows returned in 7ms
At line 18:
SELECT
   R.Recommendation_Text AS Recommendation,
   NC.Name AS CoachName,
   U.Name AS UserName,
   DL.Date AS LogDate
FROM Recommendation R
JOIN NutritionCoach NC ON R.FK_Coach_ID = NC.NutritionCoach_ID
JOIN User U ON R.FK_User_ID = U.User_ID
JOIN DailyLog DL ON R.FK_Log_ID = DL.DailyLog_ID
WHERE NC.NutritionCoach_ID = 1
ORDER BY DL.Date DESC;
```

Explanation: This query retrieves all recommendations provided by a specific coach, including details about the user and log associated with each recommendation.

User Story 4: A User Checks Progress Toward Goals
Description: Users track their progress against set goals, such as weight loss or water intake.
Query 4: Check Goal Achievement



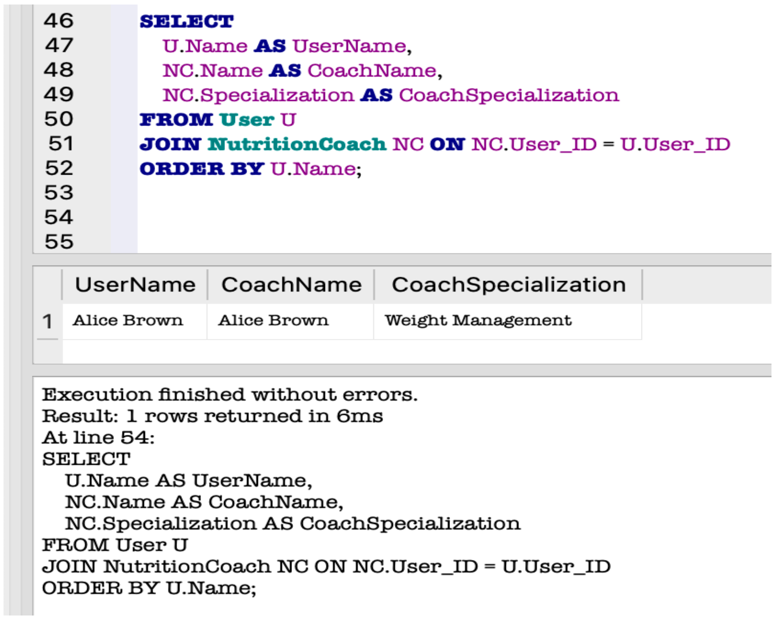
```
SELECT
  U.Name AS UserName,
  G.Goal_Type AS GoalType,
  G.Target_Value AS TargetValue,
  CASE
    WHEN G.Goal_Type = 'Weight Loss' THEN U.Weight <= G.Target_Value
    WHEN G.Goal_Type = 'Water Intake' THEN AVG(DL.Water_Intake) >= G.Target_Value
  END AS GoalMet
FROM User U
JOIN Goal G ON U.User_ID = G.FK_User_ID
LEFT JOIN DailyLog DL ON U.User_ID = DL.FK_User_ID
GROUP BY U.User_ID, G.Goal_Type;
```

	UserName	GoalType	TargetValue	GoalMet
1	John Doe	Calories Burned	600	NULL
2	John Doe	Weight Loss	70	0
3	Jane Smith	Protein Intake	120	NULL
4	Jane Smith	Water Intake	3	0

```
Execution finished without errors.
Result: 4 rows returned in 8ms
At line 31:
SELECT
  U.Name AS UserName,
  G.Goal_Type AS GoalType,
  G.Target_Value AS TargetValue,
  CASE
    WHEN G.Goal_Type = 'Weight Loss' THEN U.Weight <= G.Target_Value
    WHEN G.Goal_Type = 'Water Intake' THEN AVG(DL.Water_Intake) >= G.Target_Value
  END AS GoalMet
FROM User U
```

Explanation: This query evaluates whether users have met their goals by comparing current data with target values. It uses conditional logic to handle different goal types.

User Story 5: Admin Reviews User-Assigned Coaches
Description: Admins review which users are assigned to which coaches.
Query 5: Users and Their Assigned Coaches



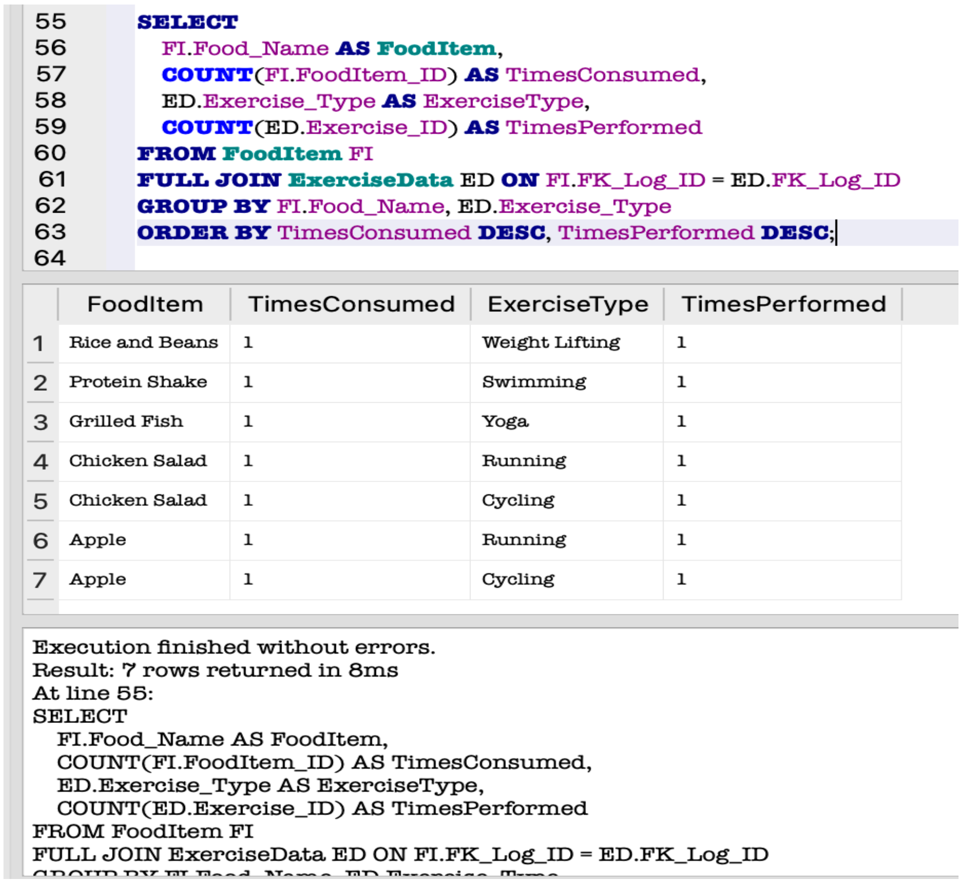
```
SELECT
  U.Name AS UserName,
  NC.Name AS CoachName,
  NC.Specialization AS CoachSpecialization
FROM User U
JOIN NutritionCoach NC ON NC.User_ID = U.User_ID
ORDER BY U.Name;
```

	UserName	CoachName	CoachSpecialization
1	Alice Brown	Alice Brown	Weight Management

```
Execution finished without errors.
Result: 1 rows returned in 6ms
At line 54:
SELECT
  U.Name AS UserName,
  NC.Name AS CoachName,
  NC.Specialization AS CoachSpecialization
FROM User U
JOIN NutritionCoach NC ON NC.User_ID = U.User_ID
ORDER BY U.Name;
```

Explanation: This query displays the coach assigned to each user along with the coach's specialization, aiding admins in overseeing user-coach relationships.

User Story 6: Analyze Food and Exercise Trends
Description: Admins and coaches analyze trends in food consumption and exercise across users.
Query 6: Most Popular Food Items and Exercises



```
SELECT
  FI.Food_Name AS FoodItem,
  COUNT(FI.FoodItem_ID) AS TimesConsumed,
  ED.Exercise_Type AS ExerciseType,
  COUNT(ED.Exercise_ID) AS TimesPerformed
FROM FoodItem FI
FULL JOIN ExerciseData ED ON FI.FK_Log_ID = ED.FK_Log_ID
GROUP BY FI.Food_Name, ED.Exercise_Type
ORDER BY TimesConsumed DESC, TimesPerformed DESC;
```

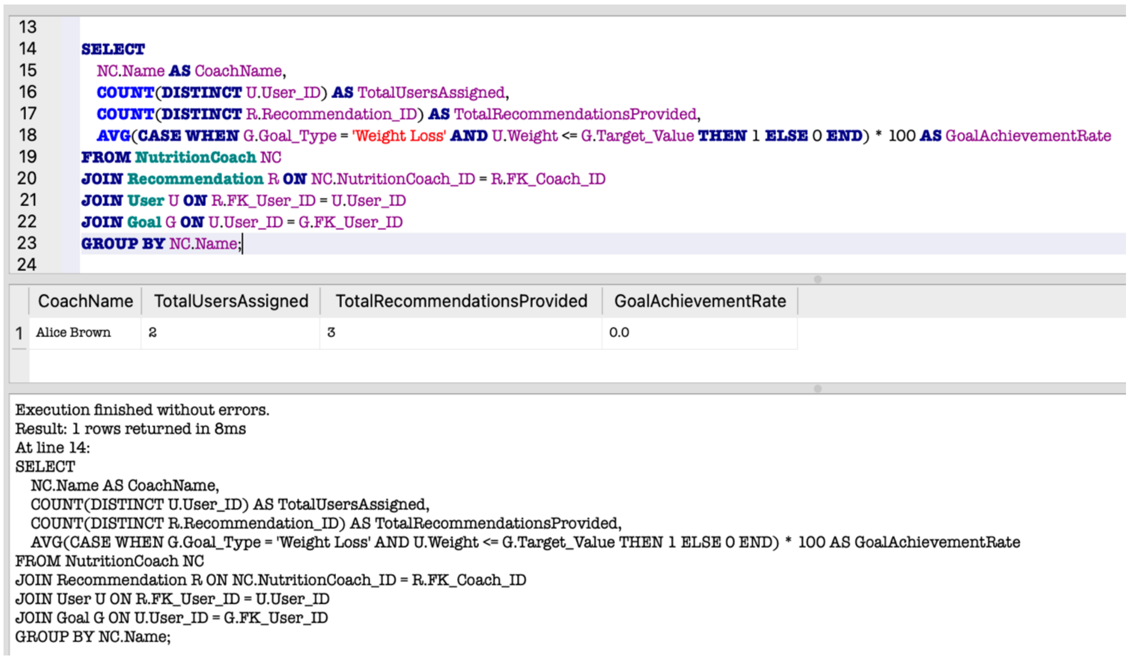
	FoodItem	TimesConsumed	ExerciseType	TimesPerformed
1	Rice and Beans	1	Weight Lifting	1
2	Protein Shake	1	Swimming	1
3	Grilled Fish	1	Yoga	1
4	Chicken Salad	1	Running	1
5	Chicken Salad	1	Cycling	1
6	Apple	1	Running	1
7	Apple	1	Cycling	1

```
Execution finished without errors.
Result: 7 rows returned in 8ms
At line 55:
SELECT
  FI.Food_Name AS FoodItem,
  COUNT(FI.FoodItem_ID) AS TimesConsumed,
  ED.Exercise_Type AS ExerciseType,
  COUNT(ED.Exercise_ID) AS TimesPerformed
FROM FoodItem FI
FULL JOIN ExerciseData ED ON FI.FK_Log_ID = ED.FK_Log_ID
```

Explanation: This query provides insights into the most frequently logged food items and exercises, helping nutrition coaches refine their recommendations.

Analytics, Reports, and Metrics

1. Coach Performance Dashboard
Purpose: Evaluate the effectiveness of nutrition coaches by analyzing the adherence and progress of their assigned users.
End-User Goal: System administrators and coaches can track the impact of their recommendations.
SQL Query: Coach Performance Summary



```sql
SELECT
   NC.Name AS CoachName,
   COUNT(DISTINCT U.User_ID) AS TotalUsersAssigned,
   COUNT(DISTINCT R.Recommendation_ID) AS TotalRecommendationsProvided,
   AVG(CASE WHEN G.Goal_Type = 'Weight Loss' AND U.Weight <= G.Target_Value THEN 1 ELSE 0 END) * 100 AS GoalAchievementRate
FROM NutritionCoach NC
JOIN Recommendation R ON NC.NutritionCoach_ID = R.FK_Coach_ID
JOIN User U ON R.FK_User_ID = U.User_ID
JOIN Goal G ON U.User_ID = G.FK_User_ID
GROUP BY NC.Name;
```

	CoachName	TotalUsersAssigned	TotalRecommendationsProvided	GoalAchievementRate
1	Alice Brown	2	3	0.0

```
Execution finished without errors.
Result: 1 rows returned in 8ms
At line 14:
SELECT
   NC.Name AS CoachName,
   COUNT(DISTINCT U.User_ID) AS TotalUsersAssigned,
   COUNT(DISTINCT R.Recommendation_ID) AS TotalRecommendationsProvided,
   AVG(CASE WHEN G.Goal_Type = 'Weight Loss' AND U.Weight <= G.Target_Value THEN 1 ELSE 0 END) * 100 AS GoalAchievementRate
FROM NutritionCoach NC
JOIN Recommendation R ON NC.NutritionCoach_ID = R.FK_Coach_ID
JOIN User U ON R.FK_User_ID = U.User_ID
JOIN Goal G ON U.User_ID = G.FK_User_ID
GROUP BY NC.Name;
```

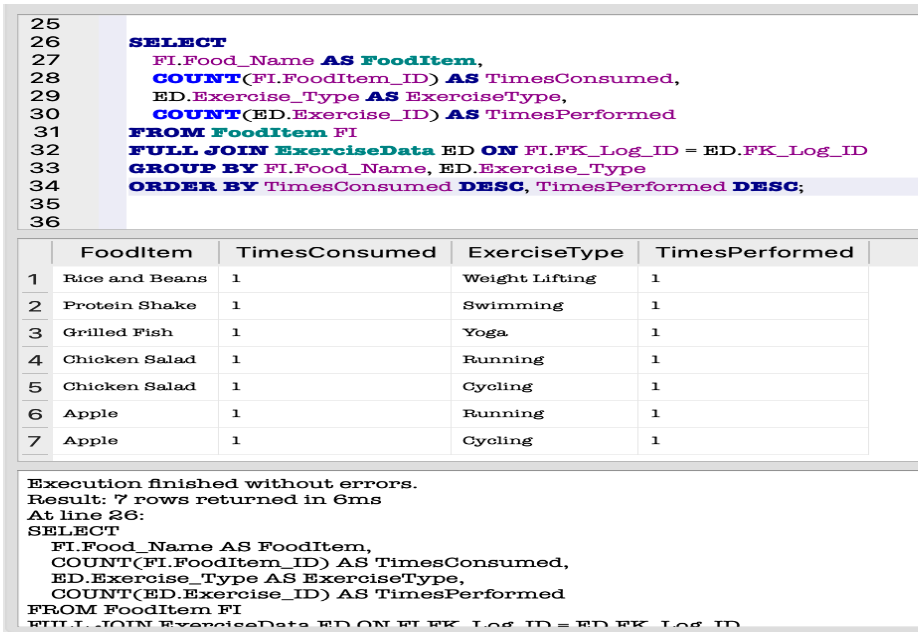
Insight Provided: This query shows the number of users each coach manages, the number of recommendations they've given, and the percentage of users who meet their goals, offering a performance snapshot.

2. Most Popular Food and Exercise Trends

Purpose: Identify the most commonly consumed food items and performed exercises across all users.

End-User Goal: Coaches use this data to provide recommendations, and admins can analyze user behavior trends.

SQL Query: Popular Food Items and Exercises



```sql
SELECT
   FI.Food_Name AS FoodItem,
   COUNT(FI.FoodItem_ID) AS TimesConsumed,
   ED.Exercise_Type AS ExerciseType,
   COUNT(ED.Exercise_ID) AS TimesPerformed
FROM FoodItem FI
FULL JOIN ExerciseData ED ON FI.FK_Log_ID = ED.FK_Log_ID
GROUP BY FI.Food_Name, ED.Exercise_Type
ORDER BY TimesConsumed DESC, TimesPerformed DESC;
```

	FoodItem	TimesConsumed	ExerciseType	TimesPerformed
1	Rice and Beans	1	Weight Lifting	1
2	Protein Shake	1	Swimming	1
3	Grilled Fish	1	Yoga	1
4	Chicken Salad	1	Running	1
5	Chicken Salad	1	Cycling	1
6	Apple	1	Running	1
7	Apple	1	Cycling	1

```
Execution finished without errors.
Result: 7 rows returned in 6ms
At line 26:
SELECT
   FI.Food_Name AS FoodItem,
   COUNT(FI.FoodItem_ID) AS TimesConsumed,
   ED.Exercise_Type AS ExerciseType,
   COUNT(ED.Exercise_ID) AS TimesPerformed
FROM FoodItem FI
FULL JOIN ExerciseData ED ON FI.FK_Log_ID = ED.FK_Log_ID
```

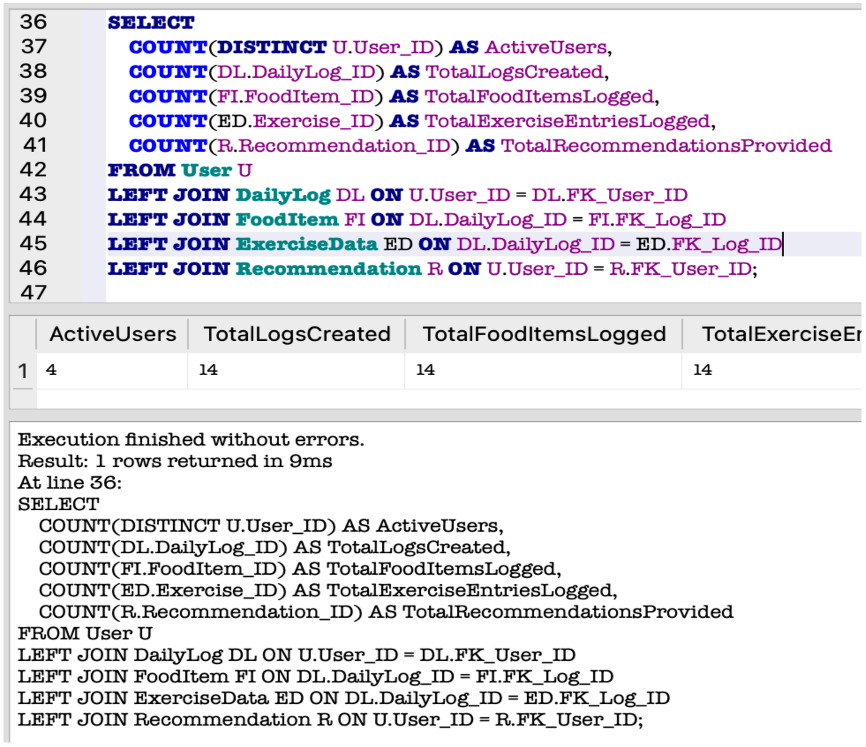
Insight Provided: This query provides a clear picture of the most logged foods and exercises, helping in understanding user preferences.

3. System Usage Metrics

Purpose: Provide system administrators with insights into app usage, such as the number of active users and logs created over time.

End-User Goal: Track user engagement and ensure system optimization.

SQL Query: System Usage Report



```sql
SELECT
   COUNT(DISTINCT U.User_ID) AS ActiveUsers,
   COUNT(DL.DailyLog_ID) AS TotalLogsCreated,
   COUNT(FI.FoodItem_ID) AS TotalFoodItemsLogged,
   COUNT(ED.Exercise_ID) AS TotalExerciseEntriesLogged,
   COUNT(R.Recommendation_ID) AS TotalRecommendationsProvided
FROM User U
LEFT JOIN DailyLog DL ON U.User_ID = DL.FK_User_ID
LEFT JOIN FoodItem FI ON DL.DailyLog_ID = FI.FK_Log_ID
LEFT JOIN ExerciseData ED ON DL.DailyLog_ID = ED.FK_Log_ID
LEFT JOIN Recommendation R ON U.User_ID = R.FK_User_ID;
```

	ActiveUsers	TotalLogsCreated	TotalFoodItemsLogged	TotalExerciseEr
1	4	14	14	14

```
Execution finished without errors.
Result: 1 rows returned in 9ms
At line 36:
SELECT
   COUNT(DISTINCT U.User_ID) AS ActiveUsers,
   COUNT(DL.DailyLog_ID) AS TotalLogsCreated,
   COUNT(FI.FoodItem_ID) AS TotalFoodItemsLogged,
   COUNT(ED.Exercise_ID) AS TotalExerciseEntriesLogged,
   COUNT(R.Recommendation_ID) AS TotalRecommendationsProvided
FROM User U
LEFT JOIN DailyLog DL ON U.User_ID = DL.FK_User_ID
LEFT JOIN FoodItem FI ON DL.DailyLog_ID = FI.FK_Log_ID
LEFT JOIN ExerciseData ED ON DL.DailyLog_ID = ED.FK_Log_ID
LEFT JOIN Recommendation R ON U.User_ID = R.FK_User_ID;
```

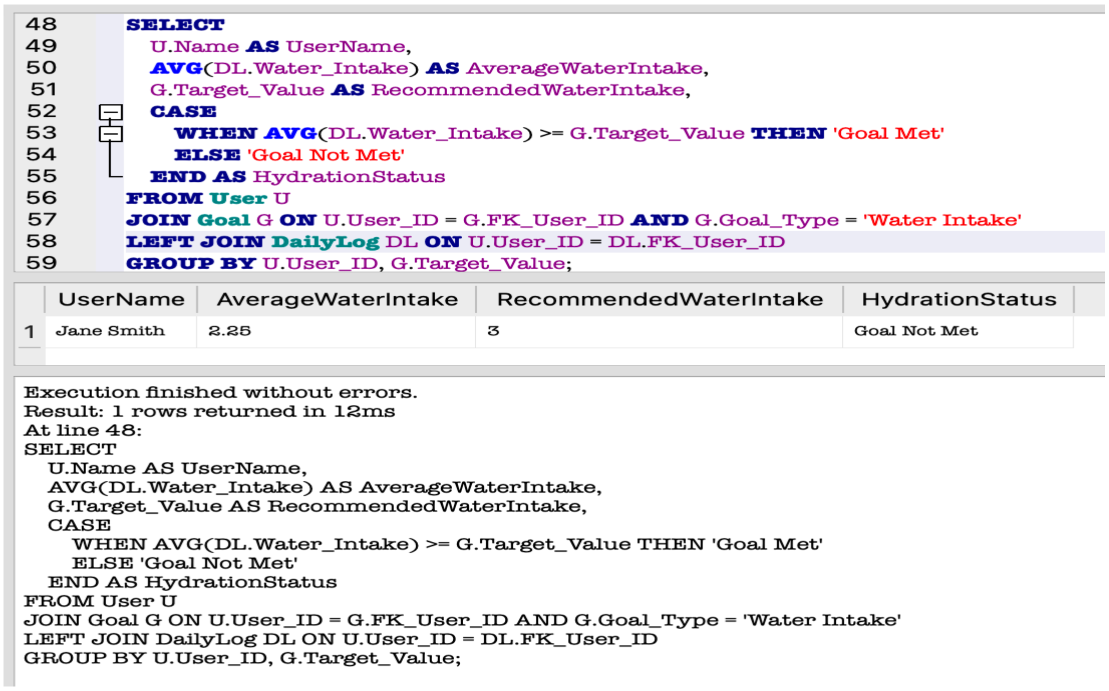
Insight Provided: A summary of system activity, including user engagement and data entry metrics, to evaluate the app's overall performance.

4. Water Intake Analysis

Purpose: Generate insights into users' hydration habits and adherence to recommended water intake goals.

End-User Goal: Assist users in identifying hydration patterns and improving water intake.

SQL Query: Water Intake Report



```
SELECT
  U.Name AS UserName,
  AVG(DL.Water_Intake) AS AverageWaterIntake,
  G.Target_Value AS RecommendedWaterIntake,
  CASE
    WHEN AVG(DL.Water_Intake) >= G.Target_Value THEN 'Goal Met'
    ELSE 'Goal Not Met'
  END AS HydrationStatus
FROM User U
JOIN Goal G ON U.User_ID = G.FK_User_ID AND G.Goal_Type = 'Water Intake'
LEFT JOIN DailyLog DL ON U.User_ID = DL.FK_User_ID
GROUP BY U.User_ID, G.Target_Value;
```

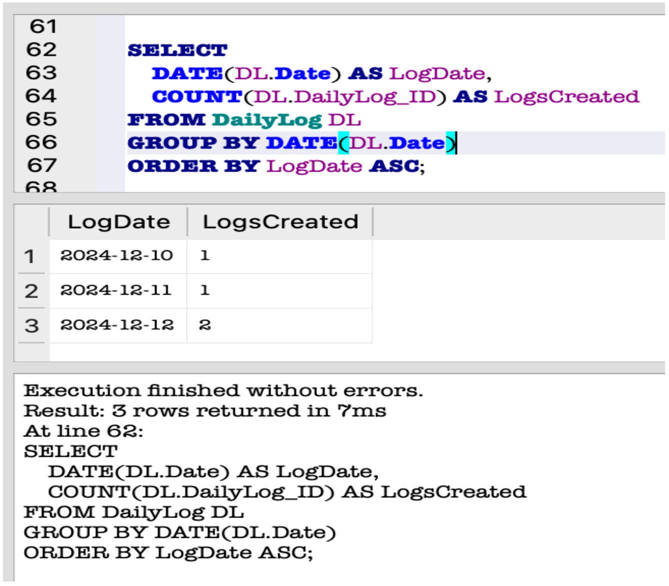
	UserName	AverageWaterIntake	RecommendedWaterIntake	HydrationStatus
1	Jane Smith	2.25	3	Goal Not Met

```
Execution finished without errors.
Result: 1 rows returned in 12ms
At line 48:
SELECT
  U.Name AS UserName,
  AVG(DL.Water_Intake) AS AverageWaterIntake,
  G.Target_Value AS RecommendedWaterIntake,
  CASE
    WHEN AVG(DL.Water_Intake) >= G.Target_Value THEN 'Goal Met'
    ELSE 'Goal Not Met'
  END AS HydrationStatus
FROM User U
JOIN Goal G ON U.User_ID = G.FK_User_ID AND G.Goal_Type = 'Water Intake'
LEFT JOIN DailyLog DL ON U.User_ID = DL.FK_User_ID
GROUP BY U.User_ID, G.Target_Value;
```

Insight Provided: Identifies users meeting their hydration goals and provides data for personalized recommendations.

5. Weekly Engagement Trends

Purpose: Analyze user engagement trends, including the number of logs created per day.
End-User Goal: Help system administrators understand peak usage periods.
SQL Query: Weekly Engagement Analysis



```
SELECT
  DATE(DL.Date) AS LogDate,
  COUNT(DL.DailyLog_ID) AS LogsCreated
FROM DailyLog DL
GROUP BY DATE(DL.Date)
ORDER BY LogDate ASC;
```

	LogDate	LogsCreated
1	2024-12-10	1
2	2024-12-11	1
3	2024-12-12	2

```
Execution finished without errors.
Result: 3 rows returned in 7ms
At line 62:
SELECT
  DATE(DL.Date) AS LogDate,
  COUNT(DL.DailyLog_ID) AS LogsCreated
FROM DailyLog DL
GROUP BY DATE(DL.Date)
ORDER BY LogDate ASC;
```

Insight Provided: Tracks daily app usage patterns to identify high and low engagement days.

Security Concerns

The Nutrition and Wellness Tracking App processes and stores sensitive user data, making data privacy and security critical components of its design and implementation. Names, email addresses, and demographic information (e.g., age, height, weight) are among the personally identifiable information (PII) that the app gathers. It also records sensitive health-related data, including dietary intake, exercise routines, and wellness goals. The handling of such data raises privacy concerns that must be addressed to protect users and ensure compliance with data protection regulations like GDPR or HIPAA (if applicable in the jurisdiction).

Unauthorized access to user data is a major worry. Role-based access control (RBAC) is implemented by the app to lessen this, restricting access to data according to user roles (e.g., regular users, nutrition coaches, admins). For instance, nutrition coaches can view logs and provide recommendations for their assigned users but cannot access other users' data. Similarly, admins have access to system-level functionalities but are restricted from viewing personal user logs unnecessarily. These measures help minimize the risk of data exposure to unauthorized parties. Another significant concern is ensuring data integrity and security in transit and at rest.

Sensitive information, such as user credentials and health data, should be stored using strong encryption mechanisms. Passwords must be hashed using secure algorithms like bcrypt or Argon2 to prevent exposure in case of a data breach. Additionally, all communication between the client and server should be encrypted using TLS (Transport Layer Security) to protect data from being intercepted during transmission. These practices ensure that even if the system is compromised, sensitive data remains inaccessible and unusable to attackers.

Lastly, the app addresses data minimization and retention policies to reduce unnecessary storage of sensitive data. Users' data is periodically reviewed, and any outdated or redundant information is securely deleted. This not only reduces the storage footprint but also minimizes the risk of sensitive data being compromised in the future. These measures demonstrate a proactive approach to privacy and security, ensuring user trust and compliance with industry standards.

Architecture

Database Architecture Requirements

1. Solution Client/Server Architecture

The proposed system will follow a client-server architecture. The client application will serve as the interface for end-users, including features for logging daily activity, viewing personalized recommendations, and setting goals. The database server will manage and store all data required by the system.

- Client Layer: The client will be a web or mobile application developed with a user-friendly interface. It will interact with the database server via secure API endpoints. This layer will handle data input, visualization, and communication with the backend.

- Server Layer: The server will house the business logic and database. It will process user requests, execute SQL queries, and return appropriate responses to the client. SQLite is used for prototyping but can be replaced with a scalable relational database like PostgreSQL or MySQL for production.

2. Cloud Hosting

The database is not cloud-hosted in this scenario but can be hosted on cloud platforms. Cloud hosting offers scalability, availability, and ease of management compared to on-premise solutions.

- Cloud Service Provider: Microsoft Azure, GCP(Google Cloud) or AWS(Amazon Cloud) are viable options. For example, AWS RDS (Relational Database Service) can host the database, while the application can be deployed on AWS Elastic Beanstalk or a similar platform.
- Advantages: Cloud hosting ensures high availability, automated backups, disaster recovery, and global accessibility. It also supports elastic scaling, which is crucial for handling increased user demand.
- Justification: Hosting in the cloud aligns with modern best practices for applications that require reliable and scalable solutions, especially when users may access the system from various geographic locations.

3. Storage Requirements

The database storage requirements will depend on the size of the user base and the volume of data collected daily. Initial estimates are as follows:

- User Data: Assuming 50 users with fields such as name, email, and personal metrics, this will require minimal storage (approximately 10 MB).
- Daily Logs: With 50 users logging daily data (calories, water intake, etc.), and each log occupying around 1 KB, this translates to ~18 MB annually.
- Food and Exercise Data: Assuming 50 food items and exercises, each occupying ~0.5 KB, the total storage is negligible (~50 KB).
- Goals and Recommendations: Storing 50 records for goals and recommendations will require ~25 KB.
- Total: The database will initially require less than 50 MB but should support scalable growth up to several GB as the user base and data volume grow.

Project Wrap-up and Future Considerations

This project provided valuable insights into database design and practical application development. Key takeaways include:

- The importance of understanding user personas to design effective databases.
- Leveraging ER diagrams to simplify complex data relationships.
- Ensuring database security and scalability for future growth.

Future considerations involve integrating machine learning for predictive analytics and expanding the app’s functionality to include wearable device integrations.

References

Cloud hosting best practices and client-server architecture:
Amazon Web Services. (n.d.). *Overview of Amazon Web Services*. Retrieved from https://aws.amazon.com/whitepapers/overview-of-amazon-web-services/

Database security and encryption:
OWASP Foundation. (n.d.). *OWASP Top Ten: Sensitive Data Exposure*. Retrieved from https://owasp.org/www-project-top-ten

Role-based access control (RBAC):
Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D., & Chandramouli, R. (2001). Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3), 224-274. https://doi.org/10.1145/501978.501980

Data privacy and security compliance:
European Union. (2016). *General Data Protection Regulation (GDPR)*. Retrieved from https://gdpr-info.eu/

SQL optimization and relational database design:
Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of database systems* (7th ed.). Pearson.

Nutrition and wellness application features:
Fitbit Inc. (n.d.). *How Fitbit works*. Retrieved from https://www.fitbit.com/global/us/technology

Data analytics and reporting:
Few, S. (2012). *Show me the numbers: Designing tables and graphs to enlighten*. Analytics Press.

ER diagram best practices:
Chen, P. P. (1976). The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1(1), 9-36. https://doi.org/10.1145/320434.320440

Password hashing and encryption:
Schneier, B. (2015). *Applied cryptography: Protocols, algorithms, and source code in C* (2nd ed.). Wiley.