FINAL PROJECT REPORT

Personal Nutrition and Fitness Management System

BUAN 6320

Group 10:

Sneha Priya Polavarapu
Prashanth Thamshetti
Jahanvi Kamleshkumar Patel
Yotesh Jami
Vivek Shanthi Chandra Bollampally
Hari Shanker Reddy Madana

Team Member's Contribution –

- Database Designing & ERR Diagram – Prashanth, Sneha
- .SQL file creation - Hari, Yotesh, Prashanth, Sneha, Shanthi, Jahnavi
- Set 1 table creation and Data Entry – Hari, Yotesh
- Set 2 table creation and Data Entry - Hari, Yotesh
- Set 3 table creation and Data Entry - Shanthi, Jahnavi
- Query creation for Stored Procedure and View – Prashanth, Sneha
- Project Write Up – Sneha, Prashanth
- Report Consolidation - Sneha

Project Write up:

Introduction

The Personal Nutrition and Fitness Management project helps users/people manage their nutrition goals and fitness goals. This involves the use of mobile or web-based applications, wearable devices, and other tools that track and analyze an individual's diet, physical activity, sleep patterns, and other health metrics. The objective is to provide personalized recommendations and tools that assist individuals in achieving their health and fitness goals, including setting and tracking fitness goals, monitoring food intake, receiving personalized meal plans, and tracking progress over time. The project will provide a user-friendly interface which makes entering food intake fast and easy and assists in tracking macronutrients such as carbohydrates, protein, fat, etc and micronutrients such as vitamins and minerals.

Business Scenario

The Nutrition and Fitness management project will target people who are looking to maintain a healthy lifestyle by managing their nutrition and fitness. The target audience will be health-conscious individuals who are looking to monitor their food and exercise intake, set goals, and track their progress.The project will offer several features, including the ability to track food intake, exercise, set goals, monitor progress, and get personalized recommendations. This project will also have a database that will store information about food and exercise, and the user's progress over time. This data will be used to provide users with personalized recommendations based on their goals and progress.Moreover, this system can be marketed to a variety of audiences, from fitness enthusiasts to individuals looking to manage chronic conditions such as diabetes or heart disease. The use of personal nutrition and fitness management has several potential benefits. For individuals, it can help them better understand their health status, track their progress towards fitness and health goals, and make more informed decisions about their diet and exercise routines. Furthermore, healthcare professionals can also benefit from this data, as it can help them personalize treatment plans and provide more accurate diagnoses. Additionally, corporate wellness programs can also benefit from personal nutrition and fitness management software by offering their employees an effective tool to improve their health and productivity.

From a business perspective, personal nutrition and fitness data management can be used to create personalized nutrition and fitness plans and services, which can be marketed to individuals, healthcare providers, and corporate wellness programs. Additionally, the data can be used to inform product development and improve the efficacy of existing products and services.

However, there are potential risks associated with personal nutrition and fitness data management, such as privacy concerns and the possibility of the data being used in

discriminatory ways. As a result, it is critical that appropriate data privacy and security protections are in place to protect individuals' personal information. Personal nutrition and fitness data management has a broad scope, with significant potential for both individual and business benefits.

Scope/Project Plan

Personal nutrition and fitness management system work on a variety of data related to an individual's health and fitness habits.This project will offer a range of features that include:

- Food intake: Many nutrition tracking applications allow users to log the food they eat, either by manually entering the information or by scanning barcodes or taking photos of their meals.
- Physical activity: Fitness tracking tools may collect data on steps taken, distance traveled, and calories burned, either through a wearable device or through the user's phone.
- Health metrics: Some personal nutrition and fitness management tools may also collect data on other health metrics, such as heart rate, blood pressure, and sleep patterns.
- Goal Setting: Users can establish goals for their nutrition and fitness, such as desired calorie intake, exercise duration, and weight loss or gain targets.
- Progress Monitoring: Users can assess their advancement towards their goals through visual feedback and tailored recommendations.
- Personalized Recommendations: The project will provide customized recommendations based on the user's goals and progress, suggesting appropriate foods and exercises that align with their objectives.
- Location data: Location data can be used by personal nutrition and fitness management businesses to provide users with information about nearby fitness facilities, healthy food options, and other resources. This feature differentiates this product from other products in the market.

It's important to note that personal nutrition and fitness management tools should be transparent about the types of data they collect from users and how that data is used. Users should also have the ability to control their data and choose what information they share with the tool or application. Additionally, appropriate data privacy and security measures should be in place to protect users' personal information.

However, the below are some of the challenges that might arise:

- Data quality: Personal nutrition and fitness management businesses rely on accurate data to provide personalized recommendations and advice to users. However, data quality can be a challenge, particularly when relying on user-generated data that may be incomplete or inaccurate.

- Data privacy and security: Personal nutrition and fitness management businesses deal with sensitive user data related to health and wellness habits, which requires appropriate data privacy and security measures to be in place. Business analysts may need to work closely with IT and security teams to ensure that user data is protected.
- Integration with other systems: Personal nutrition and fitness management tools may need to integrate with other systems, such as electronic health records, wearable devices, or other fitness tracking tools. This can be a challenge, particularly if different systems use different data formats or standards.

In conclusion, the Nutrition and Fitness management project is a comprehensive solution for people who want to lead a healthy lifestyle by managing their nutrition and fitness goals. The project targets health-conscious individuals who want to track their food and exercise intake, set goals, and monitor progress towards their goals. The project offers several features, including food and exercise tracking, goal setting, progress monitoring, and personalized recommendations. The project also has a database that stores information about food and exercise, the user's progress over time, and personalized recommendations. It's important to note that personal nutrition and fitness management system should be transparent about the types of data they collect from users and how that data is used. Users should also have the ability to control their data and choose what information they share with the tool or application. Additionally, appropriate data privacy and security measures should be in place to protect users' personal information.If a user interface is built, the project can be a product with a user-friendly interface, a dashboard that displays progress towards goals and personalized recommendations, and notifications that remind users to log their food and exercise. Overall, the Nutrition and Fitness management project is an innovative solution that can help people achieve their fitness and nutrition goals and lead a healthy lifestyle.

Assumptions/Notes About Data Entities and Relationships

Tables and foreign keys

1. **user_profiles table:** This table stores basic information about the users such as their name, password, email, age, gender, height, and weight.
2. **objectives table:** This table stores information about the user's fitness objectives such as start and end dates, target calories, target weight, and target exercise duration. It is linked to the user_profiles table through the user_id foreign key.
3. **meal_items table:** This table stores information about the meal items such as name, calories, protein, carbs, fat, and fiber content.
4. **meal_records table:** This table stores information about the user's meal records such as the item ID, quantity, and record date. It is linked to the user_profiles table through the user_id foreign key and to the meal_items table through the item_id foreign key.
5. **exercise_types table:** This table stores information about the types of exercise such as name and calories expended per minute.
6. **exercise_records table:** This table stores information about the user's exercise records such as the exercise ID, duration, and record date. It is linked to the user_profiles table through the user_id foreign key and to the exercise_types table through the exercise_id foreign key.
7. **wellness_metrics table:** This table stores information about the user's wellness metrics such as pulse rate, blood pressure reading, and sleep length. It is linked to the user_profiles table through the user_id foreign key.
8. **suggestions table:** This table stores information about the suggestions made to the users such as meal and exercise suggestions. It is linked to the user_profiles table through the user_id foreign key, to the meal_items table through the meal_item_id foreign key, and to the exercise_types table through the exercise_type_id foreign key.
9. **place_details table:** This table stores information about the places such as name, address, latitude, longitude, and category.
10. **user_places table:** This table stores information about the places that the users visit such as the visit date, place ID, and user ID. It is linked to the user_profiles table through the user_id foreign key and to the place_details table through the place_id foreign key.
11. **user_milestones table:** This table stores information about the user's milestones such as milestone date, weight, total calorie intake, total calorie expenditure, and total exercise time. It is linked to the user_profiles table through the user_id foreign key.
12. **user_connections table:** This table stores information about the connections between the users such as connected user ID. It is linked to the user_profiles table through the user_id and connected_user_id foreign keys.

13. **user_accomplishments table:** This table stores information about the user's accomplishments such as title, description, and accomplishment date. It is linked to the user_profiles table through the user_id foreign key.
14. **user_alerts table:** This table stores information about the alerts sent to the users such as the alert message and alert date. It is linked to the user_profiles table through the user_id foreign key.
15. **user_meal_storage table:** This table stores information about the user's meal storage such as inventory ID, user ID, meal item ID, item quantity, expiry date, and storage location. It is linked to the user_profiles table through the user_id foreign key and to the meal_items table through the meal_item_id foreign key.
16. **user_membership table:** This table stores information about the user's membership such as membership code, membership plan, membership start date, and membership end date. It is linked to the user_profiles table through the user_id foreign key.

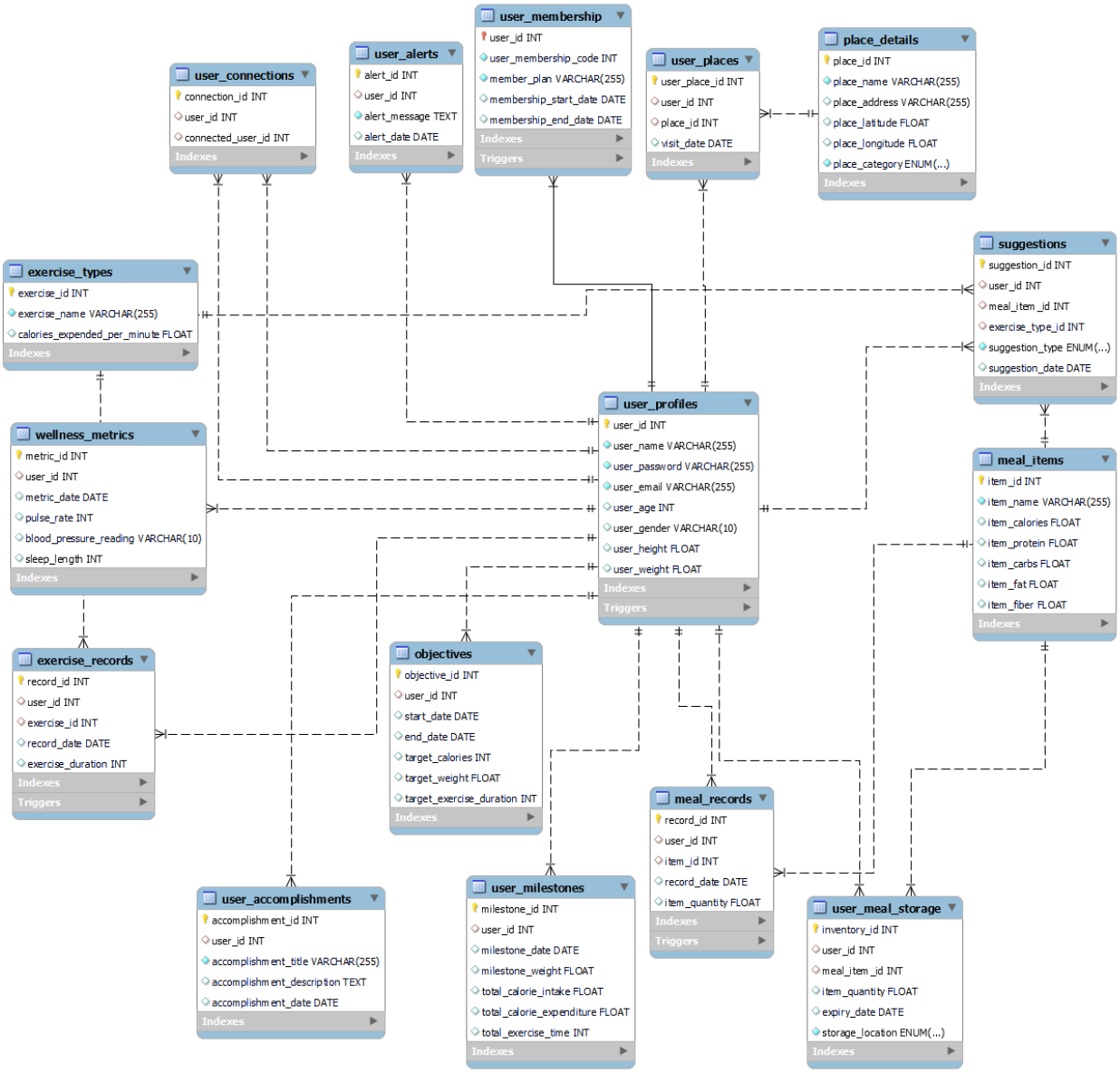
Scenarios covered for the database

The database 'Personal Nutrition and Fitness Management' appears to be designed to manage users nutrition and fitness activities. It includes tables for storing user profiles, objectives, meal items, meal records, exercise types, exercise records, wellness metrics, suggestions, places, milestones, connections, accomplishments, alerts, meal storage, and membership.
Here are some scenarios that the database can cover:

1. **User registration:** Users can create an account with their name, password, email, age, gender, height, and weight. This data can be stored in the 'user_profiles' table.
2. **User login:** An existing user can log in using their username and password.
3. **Setting fitness objectives:** Users can set fitness objectives with start and end dates, target calories, target weight, and target exercise duration. This data can be stored in the 'objectives' table.
4. **Tracking meals:** Users can record their meals by selecting meal items and entering the quantity consumed, date, and user ID. The meal items can be chosen from the 'meal_items' table, and the data can be stored in the 'meal_records' table.
5. **Tracking exercise:** Users can record their exercises by selecting the exercise type and entering the duration, date, and user ID. The exercise types can be chosen from the 'exercise_types' table, and the data can be stored in the 'exercise_records' table.
6. **Tracking wellness metrics:** Users can track their wellness metrics such as pulse rate, blood pressure, and sleep length. This data can be stored in the 'wellness_metrics' table.
7. **Providing personalized suggestions:** Users can receive personalized suggestions for meal items and exercise types based on their preferences, objectives, and past activities. These suggestions can be stored in the 'suggestions' table.
8. **Achieving milestones:** A user can achieve milestones such as milestone weight, total calorie intake, total calorie expenditure, and total exercise time using the "user_milestones" table.

9. **Finding places:** Users can find fitness centers, eateries, recreation areas, and retail shops based on their location and preferences. The places can be stored in the 'place_details' table, and the user visits can be stored in the 'user_places' table.
10. **Celebrating accomplishments:** Users can celebrate their accomplishments such as reaching a fitness milestone or winning a challenge. These accomplishments can be stored in the 'user_accomplishments' table.
11. **Connecting with other users:** A user can connect with other users using the "user_connections" table.
12. **Sending alerts:** Users can receive alerts for important events such as approaching milestones or expired meal items. These alerts can be stored in the 'user_alerts' table.
13. **Managing meal storage:** Users can manage their meal storage by adding, updating, or deleting meal items with their expiry dates and storage locations. This data can be stored in the 'user_meal_storage' table.
14. **Managing membership:** Users can manage their membership by entering their membership code, plan, start date, and end date. This data can be stored in the 'user_membership' table.

ERR Diagram :


user_membership
user_id INT
user_membership_code INT
member_plan VARCHAR(255)
membership_start_date DATE
membership_end_date DATE
Indexes
Triggers
user_alerts
alert_id INT
user_id INT
alert_message TEXT
alert_date DATE
Indexes
user_places
user_place_id INT
user_id INT
place_id INT
visit_date DATE
Indexes
place_details
place_id INT
place_name VARCHAR(255)
place_address VARCHAR(255)
place_latitude FLOAT
place_longitude FLOAT
place_category ENUM(...)
Indexes
user_connections
connection_id INT
user_id INT
connected_user_id INT
Indexes
suggestions
suggestion_id INT
user_id INT
meal_item_id INT
exercise_type_id INT
suggestion_type ENUM(...)
suggestion_date DATE
Indexes
exercise_types
exercise_id INT
exercise_name VARCHAR(255)
calories_expended_per_minute FLOAT
Indexes
user_profiles
user_id INT
user_name VARCHAR(255)
user_password VARCHAR(255)
user_email VARCHAR(255)
user_age INT
user_gender VARCHAR(10)
user_height FLOAT
user_weight FLOAT
Indexes
Triggers
wellness_metrics
metric_id INT
user_id INT
metric_date DATE
pulse_rate INT
blood_pressure_reading VARCHAR(10)
sleep_length INT
Indexes
meal_items
item_id INT
item_name VARCHAR(255)
item_calories FLOAT
item_protein FLOAT
item_carbs FLOAT
item_fat FLOAT
item_fiber FLOAT
Indexes
exercise_records
record_id INT
user_id INT
exercise_id INT
record_date DATE
exercise_duration INT
Indexes
Triggers
objectives
objective_id INT
user_id INT
start_date DATE
end_date DATE
target_calories INT
target_weight FLOAT
target_exercise_duration INT
Indexes
meal_records
record_id INT
user_id INT
item_id INT
record_date DATE
item_quantity FLOAT
Indexes
Triggers
user_accomplishments
accomplishment_id INT
user_id INT
accomplishment_title VARCHAR(255)
accomplishment_description TEXT
accomplishment_date DATE
Indexes
user_milestones
milestone_id INT
user_id INT
milestone_date DATE
milestone_weight FLOAT
total_calorie_intake FLOAT
total_calorie_expenditure FLOAT
total_exercise_time INT
Indexes
user_meal_storage
inventory_id INT
user_id INT
meal_item_id INT
item_quantity FLOAT
expiry_date DATE
storage_location ENUM(...)
Indexes

Design of the Database

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
user_profiles	user_id	None	user_name, user_password, user_email, user_age, user_gender, user_height, user_weight	86

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
objectives	objective_id	user_id references user_profiles(user_id)	start_date, end_date, target_calories, target_weight, target_exercise_dura	47

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
meal_items	item_id	None	item_name, item_calories, item_protein, item_carbs, item_fat, item_fiber	47

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
meal_records	record_id	user_id references user_profiles(user_id) item_id references meal_items(item_id)	record_date, item_quantity	70

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
exercise_types	exercise_id	None	exercise_name, calories_expended_per_minute	55

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
exercise_records	record_id	user_id references user_profiles(user_id) exercise_id references exercise_types(ex	record_date, exercise_duration	60

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
wellness_metrics	metric_id	user_id references user_profiles(user_id)	metric_date, pulse_rate, blood_pressure_reading, sleep_length	56

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
suggestions	suggestion_id	user_id references user_profiles(user_id) meal_item_id references meal_items(item_id) exercise_type_id references exercise_types(ex	suggestion_type, suggestion_date	200

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
place_details	place_id	None	place_name, place_address, place_latitude, place_longitude, place_category	50

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
user_places	user_place_id	user_id references user_profiles(user_id) place_id references place_details(plac	visit_date	45

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
user_milestones	milestone_id	user_id references user_profiles(user_id)	milestone_date, milestone_weight, total_calorie_intake, total_calorie_expenditure, total_exercise_time	49

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
user_connections	connection_id	user_id (references user_profiles.user_id) connected_user_id (references user_profiles.user_id)	None	50

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
user_accomplishments	accomplishment_id	user_id references user_profiles(user_id)	accomplishment_title, accomplishment_description, accomplishment_date	154

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
user_alerts	alert_id	user_id references user_profiles(user	alert_message, alert_date	86

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
user_meal_storage	inventory_id	user_id references user_profiles(user_id) meal_item_id references meal_items(item	item_quantity, expiry_date, storage_location	31

Table Name	Primary Key	Foreign Key	Non-key attributes	# of Rows in Table
user_membership	user_id	user_id references user_profiles(user_id)	user_membership_code, member_plan, membership_start_date, membership_end_date	29

3NF Database :

We have analyzed the dependencies within the tables. 3NF requires that:

The table is in 2NF.

There are no transitive dependencies.

1. user_profiles: This table has a primary key (user_id) and all other columns depend on the user_id. It satisfies 3NF.

2. objectives: The table has a primary key (objective_id) and a foreign key (user_id). It satisfies 3NF.

3. meal_items: This table has a primary key (item_id) and all other columns depend on the item_id. It satisfies 3NF.

4. meal_records: The table has a primary key (record_id) and foreign keys (user_id, item_id). It satisfies 3NF.

5. exercise_types: This table has a primary key (exercise_id) and all other columns depend on the exercise_id. It satisfies 3NF.

6. exercise_records: The table has a primary key (record_id) and foreign keys (user_id, exercise_id). It satisfies 3NF.

7. wellness_metrics: The table has a primary key (metric_id) and a foreign key (user_id). It satisfies 3NF.

8. suggestions: The table has a primary key (suggestion_id) and foreign keys (user_id, meal_item_id, exercise_type_id). It satisfies 3NF.

9. place_details: This table has a primary key (place_id) and all other columns depend on the place_id. It satisfies 3NF.

10. user_places: The table has a primary key (user_place_id) and foreign keys (user_id, place_id). It satisfies 3NF.

11. user_milestones: The table has a primary key (milestone_id) and a foreign key (user_id). It satisfies 3NF.

12. user_connections: The table has a primary key (connection_id) and foreign keys (user_id, connected_user_id). It satisfies 3NF.

13. user_accomplishments: The table has a primary key (accomplishment_id) and a foreign key (user_id). It satisfies 3NF.

14. user_alerts: The table has a primary key (alert_id) and a foreign key (user_id). It satisfies 3NF.

15. user_meal_storage: The table has a primary key (inventory_id) and foreign keys (user_id, meal_item_id). It satisfies 3NF.

16.User_membership: The table has primary key (user_id) referred from user_profiles. It satisfies.

All the tables listed above satisfy the conditions for 3NF.

Views & Stored Procedures created on Database

Stored Procedures :

1)

Stored Procedure 1

/* To calculate the total calorie intake for a user within a specific date range*/
delimiter //
create procedure calculatetotalcalorieintake(in p_user_id int,in p_start_date date,in p_end_date date,out p_total_calories float)
begin
select sum(mr.item_quantity * mi.item_calories)
into p_total_calories
from meal_records mr
join meal_items mi on mr.item_id = mi.item_id
where mr.user_id = p_user_id and mr.record_date between p_start_date and p_end_date;
end // delimiter ;

SQL Call of Stored Procedure

```
call calculatetotalcalorieintake(11, ‘2022-04-10','2022-05-11', @total_calories);
select @total_calories as total_calories;
```

Highlighted below shows the created record

total_calories

'215

2)

Stored Procedure 2

```
/update user objectives based on their latest milestone/

delimiter //

create procedure updateuserobjectives(in p_user_id int )

begin

declare p_latest_milestone_date date;

declare p_latest_milestone_weight float;

select milestone_date, milestone_weight into p_latest_milestone_date, p_latest_milestone_weight

from user_milestones where user_id = p_user_id order by milestone_date desc limit 1;

update objectives set target_weight = p_latest_milestone_weight where user_id = p_user_id and
end_date >= p_latest_milestone_date;

end  // delimiter ;
```

SQL Call of Stored Procedure

```
call updateuserobjectives(1);
```

3)

Stored Procedure 3

```
/transferring membership/

delimiter //

create procedure transfer_membership( in from_user_id int,in  to_user_id int)

begin

declare from_user_membership_code int;

declare from_member_plan varchar(255);

declare  from_membership_start_date date;

declare from_membership_end_date date ;

declare new_membership_code int;

set new_membership_code = (select max(user_membership_code) + 1 from user_membership);

select user_membership_code,member_plan,membership_start_date,membership_end_date  into
from_user_membership_code,from_member_plan,from_membership_start_date,from_membership_
end_date from user_membership  where user_id= from_user_id ;


if (select user_id from user_membership where user_id=to_user_id) is null

then

insert into user_membership
(user_id,user_membership_code,member_plan,membership_start_date,membership_end_date)
values

(to_user_id,new_membership_code,from_member_plan,
from_membership_start_date,from_membership_end_date );

delete from user_membership where user_id=from_user_id;

else

signal sqlstate '45000'

set message_text = 'the user you want to transfer your membership already has a membership';

end if;

end // delimiter ;
```

SQL call of stored procedure

```
call transfer_membership(10, 55);
select * from user_membership where user_id=55;
```

Highlighted below shows the created record

user_id, user_membership_code, member_plan, membership_start_date, membership_end_date

'55', '10028', 'Premium Plan', '2022-02-01', '2023-01-31'

Functions :

1)

Function 1

```
* to calculate the user's body mass index*/
delimiter //
create function calculateuserbmi(
f_user_id int
) returns float reads sql data
begin
declare f_height float;
declare f_weight float;
select user_height, user_weight
into f_height, f_weight
from user_profiles
where user_id = f_user_id;
return (f_weight / ((f_height / 100) * (f_height / 100)));
end // delimiter ;
```

SQL example of Function

```
select user_id, user_name, calculateuserbmi(user_id) as bmi
from user_profiles
where user_id = 1;
```

Highlighted below shows the created record

user_id, user_name, bmi

'1', 'SachinTendulkar', '24.977'

2)

Function 2

```
/calculate the total exercise calories burned by a user within a specific date range/
delimiter //
create function calculateexercisecaloriesburned(
f_user_id int,
f_start_date date,
f_end_date date
) returns float
begin
declare f_total_calories_burned float;
select sum(er.exercise_duration * et.calories_expended_per_minute)
into f_total_calories_burned
from exercise_records er
join exercise_types et on er.exercise_id = et.exercise_id
where er.user_id = f_user_id and er.record_date between f_start_date and f_end_date;
return f_total_calories_burned;
end // delimiter ;
```

SQL example of Function

```
select user_id, user_name, calculateexercisecaloriesburned(user_id, '2022-05-01',
'2022-05-31')
as total_calories_burned
from user_profiles
where user_id = 1;
```

Highlighted below shows the created record

user_id, user_name, total_calories_burned

'1', 'SachinTendulkar', '540'

3)

Function 3

```
/to calculate the user's net calories (calories consumed minus calories burned) within a specific date range/

delimiter //

create function calculatenetcalories(

f_user_id int,

f_start_date date,

f_end_date date

) returns float

begin declare f_total_calories_consumed float;

declare f_total_calories_burned float;

select sum(mr.item_quantity * mi.item_calories)

into f_total_calories_consumed

from meal_records mr join meal_items mi on

mr.item_id = mi.item_id

where mr.user_id = f_user_id and mr.record_date between f_start_date and f_end_date;

select sum(er.exercise_duration * et.calories_expended_per_minute) into f_total_calories_burned

from exercise_records er join exercise_types et on

er.exercise_id = et.exercise_id

where er.user_id = f_user_id and er.record_date between f_start_date and f_end_date;

return (f_total_calories_consumed - f_total_calories_burned);

end // delimiter ;
```

SQL example of Function

```
select user_id, user_name, calculatenetcalories(user_id, '2022-05-01', '2022-05-31') as net_calories

from user_profiles

where user_id = 4;
```

Highlighted below shows the created record

user_id, user_name, net_calories

'4', 'SteveSmith', '491'

Triggers

1)

Trigger 1

```
/trigger to update the total calorie intake in user_milestones after a new meal record is added/
delimiter //
create trigger updatetotalcalorieintake after insert on meal_records for each row
begin update user_milestones
set total_calorie_intake = total_calorie_intake + (new.item_quantity * (select item_calories from meal_items where item_id = new.item_id))
where user_id = new.user_id
and milestone_date = (
select max(milestone_date) from user_milestones where user_id = new.user_id
);
end // delimiter ;

/trigger to delete user data from all related tables when a user profile is deleted/
create table user_deletion_audit (
   audit_id int primary key auto_increment,
   deleted_user_id int,
   deletion_date date,
   deletion_time time
);
delimiter //
create trigger deleteuserdata after delete on user_profiles for each row
begin
   delete from objectives where user_id = old.user_id;
   delete from meal_records where user_id = old.user_id;
   delete from exercise_records where user_id = old.user_id;
   delete from wellness_metrics where user_id = old.user_id;
   delete from suggestions where user_id = old.user_id;
   delete from user_places where user_id = old.user_id;
   delete from user_milestones where user_id = old.user_id;
   delete from user_connections where user_id = old.user_id or connected_user_id = old.user_id;
   delete from user_accomplishments where user_id = old.user_id;
```

2)

Trigger 2

```
create table transfer_membership_audit (
   audit_id int primary key auto_increment,
   from_user_id int,
   to_user_id int,
   transfer_date date,
   foreign key (from_user_id) references user_profiles(user_id),
   foreign key (to_user_id) references user_profiles(user_id)
);
delimiter //
create trigger transfer_membership_audit after update on user_membership for each row
begin
if
new.user_membership_code = old.user_membership_code and
new.member_plan = old.member_plan and
new.membership_start_date = old.membership_start_date and
new.membership_end_date = old.membership_end_date and
new.user_id != old.user_id
then
insert into transfer_membership_audit (from_user_id, to_user_id, transfer_date)
values (old.user_id, new.user_id, current_date);
end if;
end // delimiter ;
```

3)

Trigger 3

```
create table excess_calorie_intake_audit (
    audit_id int primary key auto_increment,
    user_id int,
    record_date date,
    total_calorie_intake float,
    target_calories int,
    foreign key (user_id) references user_profiles(user_id)
);
delimiter //
create trigger check_excess_calorie_intake
after insert on meal_records
for each row
begin
declare total_calorie_intake float;
declare target_calories int;
    select sum(item_calories * item_quantity) into total_calorie_intake
    from meal_records
    join meal_items on meal_records.item_id = meal_items.item_id
    where meal_records.user_id = new.user_id and meal_records.record_date = new.record_date;
    select target_calories into target_calories
    from objectives
    where objectives.user_id = new.user_id and new.record_date between objectives.start_date and
objectives.end_date;
    if total_calorie_intake >= target_calories + 200 then
        insert into excess_calorie_intake_audit (user_id, record_date, total_calorie_intake,
target_calories)
        values (new.user_id, new.record_date, total_calorie_intake, target_calories);
        insert into user_alerts (user_id, alert_message, alert_date)
        values (new.user_id, concat('excess calorie intake alert! your total calorie intake on ',
new.record_date, ' was ', total_calorie_intake, ' which is ', total_calorie_intake - target_calories, '
calories more than your target of ', target_calories, ' calories.'), new.record_date);
end if;
```

Views :

1)

View 1
/daily nutrition summary for each user/ create view daily_nutrition_summary as select user_profiles.user_id, user_name, meal_records.record_date, sum(item_calories * item_quantity) as total_calories, sum(item_protein * item_quantity) as total_protein, sum(item_carbs * item_quantity) as total_carbs, sum(item_fat * item_quantity) as total_fat, sum(item_fiber * item_quantity) as total_fiber from user_profiles join meal_records on user_profiles.user_id = meal_records.user_id join meal_items on meal_records.item_id = meal_items.item_id group by user_profiles.user_id, meal_records.record_date;

Result –

user_id, user_name, record_date, total_calories, total_protein, total_carbs, total_fat, total_fiber

'1', 'SachinTendulkar', '2022-05-01', '215', '35', '10', '4.099999904632568', '2.4000000953674316'

2)

View 2

```
/daily exercise summary for each user/
create view daily_exercise_summary as
select
user_profiles.user_id,
user_name,
exercise_records.record_date,
sum(calories_expended_per_minute * exercise_duration) as total_calories_expended,
sum(exercise_duration) as total_exercise_duration
from
user_profiles
join
exercise_records on user_profiles.user_id = exercise_records.user_id
join
exercise_types on exercise_records.exercise_id = exercise_types.exercise_id
group by
user_profiles.user_id, exercise_records.record_date;
```

Result –

user_id, user_name, record_date, total_calories_expended, total_exercise_duration
'2', 'ViratKohli', '2022-05-01', '360', '60'

3)

View 3

```
/monthly progress for each user/
create view monthly_progress_summary as
select
user_profiles.user_id,
user_name,
year(metric_date) as year,
month(metric_date) as month,
avg(pulse_rate) as average_pulse_rate,
avg(sleep_length) as average_sleep_length
from
user_profiles
join
wellness_metrics on user_profiles.user_id = wellness_metrics.user_id
group by
user_profiles.user_id, year(metric_date), month(metric_date);
```

Result –

user_id, user_name, year, month, average_pulse_rate, average_sleep_length
'1', 'SachinTendulkar', '2022', '5', '70.0000', '8.0000'
'2', 'ViratKohli', '2022', '5', '75.0000', '7.0000'
'3', 'RickyPonting', '2022', '5', '80.0000', '6.0000'
'4', 'SteveSmith', '2022', '5', '72.0000', '8.0000'
'5', 'JoeRoot', '2022', '5', '78.0000', '7.0000'

4)

View 4

```
/meal suggestions for each user/
create view meal_suggestions as
select
user_profiles.user_id,
user_name,
meal_items.item_id,
item_name,
item_calories,
item_protein,
item_carbs,
item_fat,
item_fiber,
suggestion_date
from
user_profiles
join
suggestions on user_profiles.user_id = suggestions.user_id
join
meal_items on suggestions.meal_item_id = meal_items.item_id
where
suggestion_type = 'meal';
```

Result –

user_id, user_name, item_id, item_name, item_calories, item_protein, item_carbs, item_fat, item_fiber, suggestion_date
'1', 'SachinTendulkar', '1', 'Chicken Breast', '165', '31', '0', '3.6', '0', '2022-05-01'
'1', 'SachinTendulkar', '3', 'Brown Rice', '216', '5', '45', '1.8', '3.5', '2022-05-02'
'2', 'ViratKohli', '5', 'Quinoa', '222', '8', '39', '3.6', '5.2', '2022-05-15'
'3', 'RickyPonting', '7', 'Almonds', '576', '21', '22', '49', '12.2', '2022-05-01'
'4', 'SteveSmith', '9', 'Greek Yogurt', '59', '10', '3.9', '0.4', '0', '2022-05-15'

Benefits :

Centralized Data: The platform consolidates users' health data, making it easier for users and administrators to access, update, and manage information with minimal redundancy.

Simplified Data Retrieval: The centralized database allows for quicker and more efficient data retrieval compared to a distributed model, reducing maintenance and software costs.

Enhanced Data Integrity: By storing all data in a single location, the platform minimizes the risk of data integrity loss and ensures that users have access to accurate and up-to-date information.

Personalized User Experience: The platform offers tailored advice based on individual preferences and goals, providing users with a more targeted and effective approach to managing their health.

Conclusions and Future Scope:

In summary, the Nutrition and Fitness Management project offers a complete, easy-to-use platform that helps people efficiently manage their health and wellness. The system is designed to grow and adapt to future changes and advancements.

Future potential for the project includes:

Connecting with wearable devices: The platform could work with fitness trackers and health monitors for real-time data updates and personalized advice.

Covering more health areas: The platform could expand to include mental health, stress management, and other wellness topics for a well-rounded approach.

Better analytics and reports: The platform could use advanced analytics and reporting tools to help users make informed decisions about their health journey.

Teaming up with healthcare providers: Partnering with healthcare professionals could provide specialized guidance for users with unique medical needs, improving user experience.

Adding games and rewards: Introducing fun elements and incentives could motivate users to stay engaged with the platform, leading to better health results.