-- SQL Code for Fitness Centre Database System

-- This script creates the tables, inserts sample data, and provides various queries

-- based on the "FITNESS CENTRE DATABASE SYSTEM (DBMS)" PDF document.

-- Database Schema:

-- 1. Users table: Stores information about the users of the fitness center.

CREATE TABLE Users (

    User\_ID INT PRIMARY KEY,

    Name VARCHAR(20) NOT NULL,

    Age INT NOT NULL,

    Gender VARCHAR(20) NOT NULL CHECK (Gender IN ('m', 'f', 'F', 'M')), -- Using IN for case-insensitivity or specific values

    Height FLOAT NOT NULL,

    Weight FLOAT NOT NULL,

    Email VARCHAR(30) NOT NULL UNIQUE -- Added UNIQUE constraint for email

);

-- 2. Goals table: Stores fitness goals for each user.

CREATE TABLE Goals (

    Goal\_ID INT PRIMARY KEY,

    User\_ID INT NOT NULL,

    Goal\_name VARCHAR(50) NOT NULL,

    Target\_start\_date DATE NOT NULL,

    Target\_end\_date DATE NOT NULL, -- Renamed from Target\_end\_time for clarity as it's a DATE

    Target\_weight FLOAT NOT NULL,

    Target\_fat\_percentage INT NULL,

    FOREIGN KEY (User\_ID) REFERENCES Users (User\_ID)

);

-- 3. Activities table: Lists various activities available at the fitness center.

CREATE TABLE Activities (

    Activity\_ID INT PRIMARY KEY,

    User\_ID INT NOT NULL, -- Assuming this links an activity to a user who might have defined it or is related to it

    Activity\_name VARCHAR(50) NOT NULL,

    Calories\_burned\_per\_hour INT NOT NULL,

    FOREIGN KEY (User\_ID) REFERENCES Users (User\_ID)

);

-- 4. Workout\_Logs table: Records individual workout sessions for users.

CREATE TABLE Workout\_Logs (

    Log\_ID INT PRIMARY KEY,

    User\_ID INT NOT NULL,

    Activity\_ID INT NOT NULL,

    Start\_time TIME NOT NULL,

    End\_time TIME NOT NULL,

    Calories\_burned INT NOT NULL,

    FOREIGN KEY (User\_ID) REFERENCES Users (User\_ID),

    FOREIGN KEY (Activity\_ID) REFERENCES Activities (Activity\_ID)

);

-- 5. Heart\_Rate\_Logs table: Stores heart rate measurements for users.

CREATE TABLE Heart\_Rate\_Logs (

    Log\_ID INT PRIMARY KEY,

    User\_ID INT NOT NULL,

    Log\_date DATE NOT NULL,

    Heart\_rate INT NOT NULL,

    FOREIGN KEY (User\_ID) REFERENCES Users (User\_ID)

);

-- 6. Blood\_Pressure\_Logs table: Stores blood pressure measurements for users.

CREATE TABLE Blood\_Pressure\_Logs (

    Log\_ID INT PRIMARY KEY,

    User\_ID INT NOT NULL,

    Log\_Date DATE NOT NULL,

    Systolic INT NOT NULL,

    Diastolic INT NOT NULL,

    FOREIGN KEY (User\_ID) REFERENCES Users (User\_ID)

);

-- 7. Food\_Intake\_Logs table: Tracks food consumption for users.

CREATE TABLE Food\_Intake\_Logs (

    Food\_ID INT PRIMARY KEY,

    User\_ID INT NOT NULL,

    Food\_name VARCHAR(30) NOT NULL,

    Calories INT NOT NULL,

    FOREIGN KEY (User\_ID) REFERENCES Users (User\_ID)

);

-- 8. Nutrition\_Plans table: Defines nutrition plans for users.

CREATE TABLE Nutrition\_Plans (

    Plan\_ID INT PRIMARY KEY,

    User\_ID INT NOT NULL,

    Plan\_name VARCHAR(50) NOT NULL,

    Start\_date DATE NOT NULL,

    End\_date DATE NOT NULL,

    FOREIGN KEY (User\_ID) REFERENCES Users (User\_ID)

);

-- 9. Sleep\_Logs table: Records sleep data for users.

CREATE TABLE Sleep\_Logs (

    Log\_ID INT PRIMARY KEY,

    User\_ID INT NOT NULL,

    Start\_time DATETIME NOT NULL, -- Changed to DATETIME as per PDF's "Date time"

    End\_time DATETIME NOT NULL,   -- Changed to DATETIME

    Sleep\_duration TIME NOT NULL,

    FOREIGN KEY (User\_ID) REFERENCES Users (User\_ID)

);

-- 10. Water\_Intake\_Logs table: Tracks daily water intake for users.

CREATE TABLE Water\_Intake\_Logs (

    Log\_ID INT PRIMARY KEY,

    User\_ID INT NOT NULL,

    Log\_date DATE NOT NULL,

    Amount\_ML INT NOT NULL,

    FOREIGN KEY (User\_ID) REFERENCES Users (User\_ID)

);

-- 11. Exercise\_Plans table: Defines exercise plans for users.

CREATE TABLE Exercise\_Plans (

    Plan\_ID INT PRIMARY KEY,

    User\_ID INT NOT NULL,

    Plan\_name VARCHAR(50) NOT NULL,

    Start\_date DATE NOT NULL,

    End\_date DATE NOT NULL,

    FOREIGN KEY (User\_ID) REFERENCES Users (User\_ID)

);

-- 12. Body\_Fat\_Logs table: Stores body fat percentage measurements for users.

CREATE TABLE Body\_Fat\_Logs (

    Log\_ID INT PRIMARY KEY,

    User\_ID INT NOT NULL,

    Log\_date DATE NOT NULL,

    Body\_fat\_percentage FLOAT NOT NULL,

    FOREIGN KEY (User\_ID) REFERENCES Users (User\_ID)

);

-- 13. Body\_Mass\_Index table: Stores BMI calculations for users.

CREATE TABLE Body\_Mass\_Index (

    Body\_Mass\_Index\_ID INT PRIMARY KEY, -- Renamed to be consistent with table name

    User\_ID INT NOT NULL,

    Date DATE NOT NULL,

    Body\_Mass\_Index\_Value FLOAT NOT NULL, -- Renamed to avoid confusion with table name

    FOREIGN KEY (User\_ID) REFERENCES Users (User\_ID)

);

-- Sample Data Insertion (DML - INSERT statements)

-- Inserting data into Users table

INSERT INTO Users VALUES (560, 'Sukesh', 18, 'm', 5.69, 60, 'sukesh@gmail.com');

INSERT INTO Users VALUES (561, 'Mukesh', 18, 'm', 5.00, 50, 'mukesh125@gmail.com');

INSERT INTO Users VALUES (562, 'Sowmya', 18, 'F', 5.00, 45, 'sowmya764@gamil.com');

INSERT INTO Users VALUES (563, 'Sai', 20, 'm', 6.00, 60, 'sai7674@gmail.com');

INSERT INTO Users VALUES (564, 'Ravan', 18, 'm', 5.42, 52, 'ravan123@gmail.com');

INSERT INTO Users VALUES (565, 'Pavani', 17, 'F', 5.12, 54, 'pavani145@gmail.com');

INSERT INTO Users VALUES (566, 'Siva', 18, 'm', 5.69, 65, 'siva432@gmail.com');

INSERT INTO Users VALUES (567, 'Rupa', 19, 'f', 6.00, 59, 'rupa7674@gmail.com');

INSERT INTO Users VALUES (568, 'Guru', 18, 'm', 5.50, 54, 'guru7574@gmail.com');

INSERT INTO Users VALUES (569, 'Mike', 19, 'm', 5.50, 56, 'mike7874@gmail.com');

INSERT INTO Users VALUES (570, 'Rubi', 18, 'f', 5.45, 56, 'rubi794@gmail.com');

INSERT INTO Users VALUES (571, 'Lucky', 18, 'f', 5.35, 56, 'lucky994@gmail.com');

INSERT INTO Users VALUES (572, 'Roja', 18, 'f', 5.25, 56, 'roja904@gmail.com');

-- Inserting data into Goals table

INSERT INTO Goals VALUES (101, 560, 'Weight Loss', '2023-01-01', '2023-07-01', 45, 1500);

INSERT INTO Goals VALUES (102, 561, 'Cardio Fitness', '2023-01-01', '2023-06-20', 40, 1200);

INSERT INTO Goals VALUES (103, 562, 'Muscle Gain', '2023-01-01', '2023-07-24', 45, 2500);

INSERT INTO Goals VALUES (104, 563, 'Active Minutes', '2023-01-01', '2023-08-30', 50, 2300);

INSERT INTO Goals VALUES (105, 564, 'Weight Maintenance', '2023-01-01', '2023-08-30', 50, 2000);

INSERT INTO Goals VALUES (106, 565, 'Step Count', '2023-01-01', '2023-07-25', 40.00, 1400);

INSERT INTO Goals VALUES (107, 566, 'Marathon Training', '2023-01-01', '2023-06-14', 45, 1400);

INSERT INTO Goals VALUES (108, 567, 'Calorie Burn', '2023-01-01', '2023-05-30', 54.00, 2000);

INSERT INTO Goals VALUES (109, 568, 'Overall Fitness', '2023-01-01', '2023-09-01', 45, 1800);

INSERT INTO Goals VALUES (110, 569, 'Heart Rate Zones', '2023-01-01', '2023-06-10', 47, 2000);

INSERT INTO Goals VALUES (111, 569, 'Weight Loss', '2023-02-01', '2023-07-10', 48, 1900);

-- Inserting data into Activities table

INSERT INTO Activities VALUES (111, 560, 'Walking', 200);

INSERT INTO Activities VALUES (112, 561, 'Running', 500);

INSERT INTO Activities VALUES (113, 562, 'Cycling', 400);

INSERT INTO Activities VALUES (114, 563, 'Swimming', 600);

INSERT INTO Activities VALUES (115, 564, 'Weightlifting', 300);

INSERT INTO Activities VALUES (116, 565, 'Yoga', 250);

INSERT INTO Activities VALUES (117, 566, 'Cardio', 450);

INSERT INTO Activities VALUES (118, 567, 'Skipping', 550);

INSERT INTO Activities VALUES (119, 568, 'Climbing', 350);

INSERT INTO Activities VALUES (120, 569, 'Treadmill', 450);

-- Inserting data into Workout\_Logs table

INSERT INTO Workout\_Logs VALUES (121, 560, 111, '08:00:00', '08:30:00', 100);

INSERT INTO Workout\_Logs VALUES (122, 561, 112, '08:00:00', '09:00:00', 450);

INSERT INTO Workout\_Logs VALUES (123, 562, 113, '09:30:00', '10:00:00', 300);

INSERT INTO Workout\_Logs VALUES (124, 563, 114, '10:00:00', '11:00:00', 500);

INSERT INTO Workout\_Logs VALUES (125, 564, 115, '11:00:00', '11:30:00', 200);

INSERT INTO Workout\_Logs VALUES (126, 565, 116, '12:00:00', '12:20:00', 150);

INSERT INTO Workout\_Logs VALUES (127, 566, 117, '01:00:00', '01:30:00', 600);

INSERT INTO Workout\_Logs VALUES (128, 567, 118, '02:00:00', '02:30:00', 700);

INSERT INTO Workout\_Logs VALUES (129, 568, 119, '02:30:00', '03:50:00', 250);

INSERT INTO Workout\_Logs VALUES (130, 569, 120, '03:50:00', '04:30:00', 450);

-- Inserting data into Heart\_Rate\_Logs table

INSERT INTO Heart\_Rate\_Logs VALUES (131, 560, '2023-05-20', 70);

INSERT INTO Heart\_Rate\_Logs VALUES (132, 561, '2023-03-21', 75);

INSERT INTO Heart\_Rate\_Logs VALUES (133, 562, '2023-05-22', 80);

INSERT INTO Heart\_Rate\_Logs VALUES (134, 563, '2023-06-19', 65);

INSERT INTO Heart\_Rate\_Logs VALUES (135, 564, '2023-07-18', 70);

INSERT INTO Heart\_Rate\_Logs VALUES (136, 565, '2023-08-17', 75);

INSERT INTO Heart\_Rate\_Logs VALUES (137, 566, '2023-05-15', 80);

INSERT INTO Heart\_Rate\_Logs VALUES (138, 567, '2023-03-13', 85);

INSERT INTO Heart\_Rate\_Logs VALUES (139, 568, '2023-03-06', 70);

INSERT INTO Heart\_Rate\_Logs VALUES (140, 569, '2023-02-08', 75);

-- Inserting data into Blood\_Pressure\_Logs table

INSERT INTO Blood\_Pressure\_Logs VALUES (141, 560, '2023-05-01', 120, 80);

INSERT INTO Blood\_Pressure\_Logs VALUES (142, 561, '2023-05-02', 122, 78);

INSERT INTO Blood\_Pressure\_Logs VALUES (143, 562, '2023-05-03', 115, 75);

INSERT INTO Blood\_Pressure\_Logs VALUES (144, 563, '2023-05-04', 130, 82);

INSERT INTO Blood\_Pressure\_Logs VALUES (145, 564, '2023-05-05', 135, 88);

INSERT INTO Blood\_Pressure\_Logs VALUES (146, 565, '2023-05-06', 118, 76);

INSERT INTO Blood\_Pressure\_Logs VALUES (147, 566, '2023-05-07', 125, 80);

INSERT INTO Blood\_Pressure\_Logs VALUES (148, 567, '2023-05-08', 128, 85);

INSERT INTO Blood\_Pressure\_Logs VALUES (149, 568, '2023-05-09', 130, 80);

INSERT INTO Blood\_Pressure\_Logs VALUES (150, 569, '2023-05-10', 122, 78);

-- Inserting data into Food\_Intake\_Logs table

INSERT INTO Food\_Intake\_Logs VALUES (151, 560, 'Egg rice', 120);

INSERT INTO Food\_Intake\_Logs VALUES (152, 561, 'Gobi rice', 100);

INSERT INTO Food\_Intake\_Logs VALUES (153, 562, 'Chicken Breast', 165);

INSERT INTO Food\_Intake\_Logs VALUES (154, 563, 'chicken biriyani', 147);

INSERT INTO Food\_Intake\_Logs VALUES (155, 564, 'Egg dosa', 60);

INSERT INTO Food\_Intake\_Logs VALUES (156, 565, 'egg roll', 23);

INSERT INTO Food\_Intake\_Logs VALUES (157, 566, 'Tomato rice', 90);

INSERT INTO Food\_Intake\_Logs VALUES (158, 567, 'chicken lollypop', 234);

INSERT INTO Food\_Intake\_Logs VALUES (159, 568, 'Brown Rice', 215);

INSERT INTO Food\_Intake\_Logs VALUES (160, 569, 'Almonds', 164);

-- Inserting data into Nutrition\_Plans table

INSERT INTO Nutrition\_Plans VALUES (161, 560, 'Weight Loss Plan', '2023-01-01', '2023-05-31');

INSERT INTO Nutrition\_Plans VALUES (162, 561, 'Muscle Gain Plan', '2023-01-01', '2023-06-20');

INSERT INTO Nutrition\_Plans VALUES (163, 562, 'Healthy Eating Plan', '2023-02-01', '2023-07-30');

INSERT INTO Nutrition\_Plans VALUES (164, 563, 'Vegan Diet Plan', '2023-02-15', '2023-07-30');

INSERT INTO Nutrition\_Plans VALUES (165, 564, 'Low Carb Plan', '2023-03-20', '2023-08-20');

INSERT INTO Nutrition\_Plans VALUES (166, 565, 'Gluten-Free Plan', '2023-03-25', '2023-08-25');

INSERT INTO Nutrition\_Plans VALUES (167, 566, 'Dairy-Free Plan', '2023-04-30', '2023-09-30');

INSERT INTO Nutrition\_Plans VALUES (168, 567, 'Balanced Diet Plan', '2023-04-01', '2023-09-01');

INSERT INTO Nutrition\_Plans VALUES (169, 568, 'Intermittent Fasting Plan', '2023-05-05', '2023-10-05');

INSERT INTO Nutrition\_Plans VALUES (170, 569, 'High Protein Plan', '2023-06-10', '2023-11-10');

-- Inserting data into Sleep\_Logs table

INSERT INTO Sleep\_Logs VALUES (171, 560, '2023-05-01 09:00:00', '2023-05-02 06:00:00', '09:00:00');

INSERT INTO Sleep\_Logs VALUES (172, 561, '2023-05-02 10:30:00', '2023-05-03 07:30:00', '10:00:00');

INSERT INTO Sleep\_Logs VALUES (173, 562, '2023-05-03 08:45:00', '2023-05-04 06:15:00', '10:05:00');

INSERT INTO Sleep\_Logs VALUES (174, 563, '2023-05-04 09:30:00', '2023-05-05 06:30:00', '09:00:00');

INSERT INTO Sleep\_Logs VALUES (175, 564, '2023-05-05 09:45:00', '2023-05-06 07:45:00', '10:00:00');

INSERT INTO Sleep\_Logs VALUES (176, 565, '2023-05-06 10:45:00', '2023-05-07 08:15:00', '09:30:00');

INSERT INTO Sleep\_Logs VALUES (177, 566, '2023-05-07 06:30:00', '2023-05-08 07:00:00', '12:30:00');

INSERT INTO Sleep\_Logs VALUES (178, 567, '2023-05-08 08:15:00', '2023-05-09 07:45:00', '11:30:00');

INSERT INTO Sleep\_Logs VALUES (179, 568, '2023-05-09 11:45:00', '2023-05-10 07:15:00', '07:30:00');

INSERT INTO Sleep\_Logs VALUES (180, 569, '2023-05-10 11:10:00', '2023-05-11 07:30:00', '08:20:00');

-- Inserting data into Water\_Intake\_Logs table

INSERT INTO Water\_Intake\_Logs VALUES (191, 560, '2023-05-01', 1200);

INSERT INTO Water\_Intake\_Logs VALUES (192, 561, '2023-05-02', 1800);

INSERT INTO Water\_Intake\_Logs VALUES (193, 562, '2023-05-03', 1500);

INSERT INTO Water\_Intake\_Logs VALUES (194, 563, '2023-05-04', 1500);

INSERT INTO Water\_Intake\_Logs VALUES (195, 564, '2023-05-05', 1200);

INSERT INTO Water\_Intake\_Logs VALUES (196, 565, '2023-05-06', 1900);

INSERT INTO Water\_Intake\_Logs VALUES (197, 566, '2023-05-07', 1700);

INSERT INTO Water\_Intake\_Logs VALUES (198, 567, '2023-05-08', 1000);

INSERT INTO Water\_Intake\_Logs VALUES (199, 568, '2023-05-09', 1200);

INSERT INTO Water\_Intake\_Logs VALUES (200, 569, '2023-05-10', 1600);

-- Inserting data into Exercise\_Plans table

INSERT INTO Exercise\_Plans VALUES (201, 560, 'Weight Loss Plan', '2023-01-01', '2023-05-31');

INSERT INTO Exercise\_Plans VALUES (202, 561, 'Muscle Gain Plan', '2023-01-01', '2023-05-31');

INSERT INTO Exercise\_Plans VALUES (203, 562, 'Strength Training Plan', '2023-01-01', '2023-05-31');

INSERT INTO Exercise\_Plans VALUES (204, 563, 'Cardiovascular Fitness Plan', '2023-01-01', '2023-05-31');

INSERT INTO Exercise\_Plans VALUES (205, 564, 'Endurance Training Plan', '2023-01-01', '2023-05-31');

INSERT INTO Exercise\_Plans VALUES (206, 565, 'Flexibility Plan', '2023-01-01', '2023-05-31');

INSERT INTO Exercise\_Plans VALUES (207, 566, 'HIIT Workout Plan', '2023-01-01', '2023-05-31');

INSERT INTO Exercise\_Plans VALUES (208, 567, 'CrossFit Training Plan', '2023-01-01', '2023-05-31');

INSERT INTO Exercise\_Plans VALUES (209, 568, 'Pilates Workout Plan', '2023-01-01', '2023-05-31');

INSERT INTO Exercise\_Plans VALUES (210, 569, 'Yoga and Meditation Plan', '2023-01-01', '2023-05-31');

-- Inserting data into Body\_Fat\_Logs table

INSERT INTO Body\_Fat\_Logs VALUES (211, 560, '2023-05-01', 22.5);

INSERT INTO Body\_Fat\_Logs VALUES (212, 561, '2023-05-03', 21.8);

INSERT INTO Body\_Fat\_Logs VALUES (213, 562, '2023-05-05', 18.9);

INSERT INTO Body\_Fat\_Logs VALUES (214, 563, '2023-05-07', 25.3);

INSERT INTO Body\_Fat\_Logs VALUES (215, 564, '2023-05-09', 20.1);

INSERT INTO Body\_Fat\_Logs VALUES (216, 565, '2023-05-11', 19.6);

INSERT INTO Body\_Fat\_Logs VALUES (217, 566, '2023-05-13', 24.7);

INSERT INTO Body\_Fat\_Logs VALUES (218, 567, '2023-05-15', 18.5);

INSERT INTO Body\_Fat\_Logs VALUES (219, 568, '2023-05-17', 21.2);

INSERT INTO Body\_Fat\_Logs VALUES (220, 569, '2023-05-19', 24.0);

-- Inserting data into Body\_Mass\_Index table

INSERT INTO Body\_Mass\_Index VALUES (221, 560, '2023-05-01', 22.5);

INSERT INTO Body\_Mass\_Index VALUES (222, 561, '2023-05-02', 23.1);

INSERT INTO Body\_Mass\_Index VALUES (223, 562, '2023-05-01', 21.8);

INSERT INTO Body\_Mass\_Index VALUES (224, 563, '2023-05-02', 25.4);

INSERT INTO Body\_Mass\_Index VALUES (225, 564, '2023-05-01', 20.3);

INSERT INTO Body\_Mass\_Index VALUES (226, 565, '2023-05-02', 24.9);

INSERT INTO Body\_Mass\_Index VALUES (227, 566, '2023-05-01', 26.7);

INSERT INTO Body\_Mass\_Index VALUES (228, 567, '2023-05-02', 22.6);

INSERT INTO Body\_Mass\_Index VALUES (229, 568, '2023-05-01', 19.2);

INSERT INTO Body\_Mass\_Index VALUES (230, 569, '2023-05-02', 21.1);

-- SQL Queries (DML - SELECT statements)

-- Query 1: Retrieve users' ID, name, and target fat percentage for specific target fat percentage.

-- This query joins the Users and Goals tables to get user details along with their target fat percentage.

SELECT u.User\_ID, u.Name, g.Target\_fat\_percentage

FROM Users AS u

JOIN Goals AS g ON g.User\_ID = u.User\_ID

WHERE g.Target\_fat\_percentage = 2000;

-- Query 2: Count the number of goals with a target fat percentage greater than 1500.

-- This query uses an aggregate function (COUNT) and a WHERE clause to filter goals.

SELECT COUNT(Target\_weight) AS target\_weight\_count -- The column name in the original query was Target\_weight, but the condition is on Target\_fat\_percentage. Keeping it as is from the PDF.

FROM Goals

WHERE Target\_fat\_percentage > 1500;

-- Query 3: Calculate the average target fat percentage for users with a BMI greater than 20.

-- This query demonstrates joining multiple tables (Goals, Body\_Mass\_Index) and using an aggregate function (AVG).

SELECT AVG(g.Target\_fat\_percentage) AS avg\_target\_fat\_percentage

FROM Goals AS g

JOIN Body\_Mass\_Index AS b ON b.User\_ID = g.User\_ID

WHERE b.Body\_Mass\_Index\_Value > 20; -- Corrected column name for BMI

-- Query 4: Retrieve all users who have a goal to lose weight.

-- This query uses a subquery with an IN clause to filter users based on their goals.

SELECT \*

FROM Users

WHERE User\_ID IN (SELECT User\_ID FROM Goals WHERE Goal\_name = 'Weight Loss');

-- Query 5: Retrieve the names of users who have logged water intake exceeding 1500 ml.

-- This query joins Users and Water\_Intake\_Logs to filter users based on their water intake.

SELECT u.Name

FROM Users AS u

JOIN Water\_Intake\_Logs AS w ON u.User\_ID = w.User\_ID

WHERE w.Amount\_ML > 1500;

-- Query 6: Retrieve all body fat logs for a specific user (e.g., User\_ID 560) ordered by log date.

-- This query demonstrates filtering by a specific user and ordering the results.

SELECT \*

FROM Body\_Fat\_Logs

WHERE User\_ID = 560 -- Replaced 'user\_id' with a specific ID for demonstration

ORDER BY Log\_date;

-- Query 7: Retrieve the names of users who have a sleep duration greater than 10 hours.

-- This query joins Users and Sleep\_Logs and filters based on sleep duration.

SELECT u.Name, u.User\_ID

FROM Users AS u

JOIN Sleep\_Logs AS S ON u.User\_ID = S.User\_ID

WHERE S.Sleep\_duration > '10:00:00'; -- Using 'HH:MM:SS' format for TIME comparison

-- Query 8: Retrieve the names of users with a BMI greater than 20, weight greater than 50, and height greater than 5.50.

-- This query joins Users and Body\_Mass\_Index and applies multiple conditions.

SELECT u.Name

FROM Users AS u

JOIN Body\_Mass\_Index AS B ON u.User\_ID = B.User\_ID

WHERE B.Body\_Mass\_Index\_Value > 20.00 AND u.Weight > 50 AND u.Height > 5.50;

-- Query 9: Retrieve the name of the user who has a sleep log with log ID 175.

-- This query joins Users and Sleep\_Logs to find the user associated with a specific log.

SELECT u.Name

FROM Users AS u

JOIN Sleep\_Logs AS S ON u.User\_ID = S.User\_ID

WHERE S.Log\_ID = 175;

-- Query 10: Retrieve the average systolic and diastolic blood pressure for a specific user (e.g., 'Sai').

-- This query uses a subquery to find the User\_ID for 'Sai' and then calculates averages.

SELECT AVG(Systolic) AS average\_systolic, AVG(Diastolic) AS average\_diastolic

FROM Blood\_Pressure\_Logs

WHERE User\_ID = (SELECT User\_ID FROM Users WHERE Name = 'Sai');

-- Query 11: Retrieve the names of users with a body fat percentage between 18 and 20.

-- This query joins Users and Body\_Fat\_Logs and uses the BETWEEN operator.

SELECT u.Name

FROM Users AS u

JOIN Body\_Fat\_Logs AS B ON u.User\_ID = B.User\_ID

WHERE B.Body\_fat\_percentage BETWEEN 18 AND 20;

-- Query 12: Retrieve the names of users who have a goal named 'Weight Loss'.

-- This query joins Users and Goals to filter users based on their goal name.

SELECT U.Name

FROM Users AS U

JOIN Goals AS G ON U.User\_ID = G.User\_ID

WHERE G.Goal\_name = 'Weight Loss';

-- Query 13: Calculate the average heart rate for each user.

-- This query uses GROUP BY to calculate the average heart rate for each user.

SELECT U.Name, AVG(H.Heart\_rate) AS average\_heart\_rate

FROM Users U

JOIN Heart\_Rate\_Logs H ON U.User\_ID = H.User\_ID

GROUP BY U.Name;

-- Query 14: Retrieve the names of users who have a goal named 'Muscle Gain' and their weight is less than or equal to the target weight.

-- This query combines conditions from Users and Goals tables.

SELECT U.Name, G.Goal\_name

FROM Users U

JOIN Goals G ON U.User\_ID = G.User\_ID

WHERE G.Goal\_name = 'Muscle Gain' AND U.Weight <= G.Target\_weight;

-- Query 15: Retrieve the user ID and name of users who have goals and workout logs with target weight greater than or equal to their weight.

-- This query joins Users, Goals, and Workout\_Logs to find users meeting the criteria.

SELECT u.User\_ID, u.Name

FROM Users u

JOIN Goals g ON u.User\_ID = g.User\_ID

JOIN Workout\_Logs w ON u.User\_ID = w.User\_ID

WHERE g.Target\_weight >= u.Weight;

-- Query 16: Retrieve all users whose heart rate is greater than the average heart rate of all users.

-- This query uses a subquery to find the average heart rate and then filters users.

SELECT \*

FROM Users

WHERE User\_ID IN (

    SELECT User\_ID

    FROM Heart\_Rate\_Logs

    WHERE Heart\_rate > (SELECT AVG(Heart\_rate) FROM Heart\_Rate\_Logs)

);

-- Query 17: Retrieve the names of users who have a goal named 'Weight Loss' and their weight is less than or equal to the target weight.

-- Similar to Query 14, but specifically for 'Weight Loss' goal.

SELECT Name

FROM Users

JOIN Goals ON Users.User\_ID = Goals.User\_ID

WHERE Goals.Goal\_name = 'Weight Loss' AND Users.Weight <= Goals.Target\_weight;

-- Query 18: Retrieve all users who have logged food intake with calories greater than 150.

-- This query joins Users and Food\_Intake\_Logs and filters based on calories.

SELECT u.\*, f.Food\_ID, f.Food\_name, f.Calories

FROM Users AS u

JOIN Food\_Intake\_Logs AS f ON u.User\_ID = f.User\_ID

WHERE f.Calories > 150;

-- Query 19: Retrieve the average calories burned per hour for each activity.

-- This query uses GROUP BY to calculate average calories burned per activity.

SELECT A.Activity\_name, AVG(A.Calories\_burned\_per\_hour) AS average\_calories\_burned

FROM Activities AS A

GROUP BY A.Activity\_name;

-- Query 20: Retrieve the user ID, name, current weight, and target weight for users whose weight is greater than their target weight.

-- This query joins Users and Goals to identify users who are currently above their target weight.

SELECT u.User\_ID, u.Name, u.Weight, g.Target\_weight

FROM Users AS u

JOIN Goals AS g ON u.User\_ID = g.User\_ID

WHERE u.Weight > g.Target\_weight;

-- Query 21: Retrieve the activity name and the count of unique users who have performed each activity.

-- This query uses multiple joins and COUNT(DISTINCT) to count unique users per activity.

SELECT A.Activity\_name, COUNT(DISTINCT U.User\_ID) AS unique\_user\_count

FROM Activities AS a

JOIN Workout\_Logs AS w ON a.Activity\_ID = w.Activity\_ID

JOIN Users AS u ON w.User\_ID = u.User\_ID

GROUP BY A.Activity\_name;

-- Query 22: Retrieve the user ID, name, height, and workout logs for users who have a workout between 8:00 AM and 9:00 AM.

-- This query filters workout logs based on a specific time range.

SELECT u.User\_ID, u.Name, u.Height, w.Start\_time, w.End\_time

FROM Users AS u

JOIN Workout\_Logs AS w ON u.User\_ID = w.User\_ID

WHERE w.Start\_time >= '08:00:00' AND w.End\_time <= '09:00:00'; -- Adjusted to be a range

-- Query 23: Retrieve the user ID, name, food name, and calories for users who have consumed food with more than 100 calories.

-- This query joins Users and Food\_Intake\_Logs to show food details for users with high calorie intake.

SELECT u.User\_ID, u.Name, f.Food\_name, f.Calories

FROM Users AS u

JOIN Food\_Intake\_Logs AS f ON f.User\_ID = u.User\_ID

WHERE f.Calories > 100;

-- Query 24: Retrieve the user ID, name, and activity name for users who have performed activities with a calories burned per hour greater than 400.

-- This query joins Users and Activities to find users involved in high-calorie-burning activities.

SELECT u.User\_ID, u.Name, a.Activity\_name

FROM Users AS u

JOIN Activities AS a ON a.User\_ID = u.User\_ID

WHERE a.Calories\_burned\_per\_hour > 400;

-- Query 25: Retrieve the user ID and name for users who have consumed food with more than 100 calories, have a weight greater than 40, and have a BMI greater than 20.

-- This query demonstrates joining multiple tables with multiple conditions.

SELECT u.User\_ID, u.Name

FROM Users AS u

JOIN Food\_Intake\_Logs AS f ON f.User\_ID = u.User\_ID

JOIN Body\_Mass\_Index AS b ON b.User\_ID = u.User\_ID

WHERE f.Calories > 100 AND u.Weight > 40 AND b.Body\_Mass\_Index\_Value > 20;

-- Query 26: Retrieve the name of the user who has a sleep log with log ID 175.

-- This is a duplicate of Query 9, but included as per the PDF's repetition.

SELECT u.Name

FROM Users AS u

JOIN Sleep\_Logs AS S ON u.User\_ID = S.User\_ID

WHERE S.Log\_ID = 175;

-- Query 27: Retrieve the average systolic and diastolic blood pressure for a specific user (e.g., 'Siva').

-- Similar to Query 10, but for user 'Siva'.

SELECT AVG(Systolic) AS average\_systolic, AVG(Diastolic) AS average\_diastolic

FROM Blood\_Pressure\_Logs

WHERE User\_ID = (SELECT User\_ID FROM Users WHERE Name = 'Siva');

-- Query 28: Find the booking id who booked roomtype as ac.

-- NOTE: This query refers to 'Room' and 'Booking' tables which are NOT defined in the provided schema.

-- This query seems to be from a different context or an example not fully integrated into the fitness center schema.

-- I will comment it out or provide a placeholder if the tables were not defined.

-- SELECT b.B\_id FROM Room r, Booking b WHERE b.R\_number=r.R\_number AND r.R\_type='A';

-- Query 29: Find cid, cname and from customer whose age is minimum

-- NOTE: This query refers to 'customers' table which is NOT defined in the provided schema.

-- I will comment it out or provide a placeholder if the tables were not defined.

-- SELECT CID, CName, age FROM customers WHERE age=(SELECT MIN(age) FROM customers);

-- Query 30: Display the total no of departments

-- NOTE: This query refers to 'Depart' table which is NOT defined in the provided schema.

-- I will comment it out or provide a placeholder if the tables were not defined.

-- SELECT COUNT(DID) FROM Depart;

-- Query 31: Retrieve the user ID, name, food name, and calories for users who have consumed food with more than 100 calories.

-- This is a duplicate of Query 23, but included as per the PDF's repetition.

SELECT u.User\_ID, u.Name, f.Food\_name, f.Calories

FROM Users AS u

JOIN Food\_Intake\_Logs AS f ON f.User\_ID = u.User\_ID

WHERE f.Calories > 100;

-- Query 32: Retrieve the user ID, name, and activity name for users who have performed activities with a calories burned per hour greater than 400.

-- This is a duplicate of Query 24, but included as per the PDF's repetition.

SELECT u.User\_ID, u.Name, a.Activity\_name

FROM Users AS u

JOIN Activities AS a ON a.User\_ID = u.User\_ID

WHERE a.Calories\_burned\_per\_hour > 400;

-- Query 33: Retrieve the goal ID and the count of users for each unique target weight.

-- This query groups by Goal\_ID and Target\_weight and counts users.

SELECT g.Goal\_ID, COUNT(g.Target\_weight) AS weight\_count

FROM Goals AS g

JOIN Users AS u ON u.User\_ID = g.User\_ID

WHERE Target\_weight >= 50

GROUP BY g.Goal\_ID, g.Target\_weight;

-- Query 34: Retrieve all workout logs where the activity name contains the word "Running".

-- This query uses a subquery and LIKE operator.

SELECT \*

FROM Workout\_Logs

WHERE Activity\_ID IN (

    SELECT Activity\_ID

    FROM Activities

    WHERE Activity\_name LIKE '%Running%'

);

-- Query 35: Retrieve all activities that have "run" anywhere in their name.

-- This query uses the LIKE operator for pattern matching.

SELECT \*

FROM Activities

WHERE Activity\_name LIKE '%run%';

-- Query 36: Get the names and average sleep duration of users who have an average sleep duration greater than 8 hours.

-- This query uses DATEDIFF to calculate duration and HAVING for filtering grouped results.

-- NOTE: DATEDIFF function syntax might vary slightly depending on the SQL dialect (e.g., SQL Server, MySQL, PostgreSQL).

-- The original PDF uses DATEDIFF(MINUTE, '00:00:00', S.Sleep\_duration)) / 60 which implies SQL Server or similar.

-- For standard SQL, you might need to extract hours from TIME or use interval arithmetic.

-- Assuming a SQL Server-like environment for DATEDIFF.

SELECT U.Name, AVG(CAST(strftime('%H', S.Sleep\_duration) AS INTEGER) + CAST(strftime('%M', S.Sleep\_duration) AS INTEGER) / 60.0) AS average\_sleep\_duration

FROM Users

JOIN Sleep\_Logs AS S ON Users.User\_ID = S.User\_ID

GROUP BY Users.Name

HAVING AVG(CAST(strftime('%H', S.Sleep\_duration) AS INTEGER) + CAST(strftime('%M', S.Sleep\_duration) AS INTEGER) / 60.0) > 8;

-- Query 37: Get the names and average heart rates of all users.

-- This query uses GROUP BY to calculate average heart rate for each user.

SELECT Users.Name, AVG(Heart\_Rate\_Logs.Heart\_rate) AS average\_heart\_rate

FROM Users

JOIN Heart\_Rate\_Logs ON Users.User\_ID = Heart\_Rate\_Logs.User\_ID

GROUP BY Users.Name;

-- Query 38: Get the names of users who have logged a food item containing the word 'rice'.

-- This query uses DISTINCT and LIKE for pattern matching in food names.

SELECT DISTINCT u.User\_ID, U.Name, f.Food\_name, f.Calories

FROM Users AS u

JOIN Food\_Intake\_Logs AS f ON U.User\_ID = F.User\_ID

WHERE F.Food\_name LIKE '%rice%';

-- Query 39: Get the names of users who have logged a heart rate greater than the average heart rate of all users.

-- This query uses a subquery to find the overall average heart rate.

SELECT DISTINCT Users.Name

FROM Users

JOIN Heart\_Rate\_Logs ON Users.User\_ID = Heart\_Rate\_Logs.User\_ID

WHERE Heart\_Rate\_Logs.Heart\_rate > (

    SELECT AVG(Heart\_rate)

    FROM Heart\_Rate\_Logs

);

-- Query 40: Retrieve Which users have consumed the highest total calories.

-- This query calculates the total calories consumed by each user.

SELECT Users.Name, SUM(Food\_Intake\_Logs.Calories) AS TotalCaloriesConsumed

FROM Food\_Intake\_Logs

JOIN Users ON Food\_Intake\_Logs.User\_ID = Users.User\_ID

GROUP BY Users.Name

ORDER BY TotalCaloriesConsumed DESC; -- Added ORDER BY to find highest

-- Query 41: Get the names of users who have a username ends with the letter 'h'.

-- This query uses the LIKE operator for pattern matching.

SELECT \*

FROM Users AS u

WHERE Name LIKE '%h';

-- Query 42: Find the users who have achieved their weight loss goal or their muscle gain goal etc.

-- This query uses a subquery with IN clause to filter users based on multiple goal names.

SELECT \*

FROM Users

WHERE User\_ID IN (

    SELECT User\_ID

    FROM Goals

    WHERE Goal\_name IN ('Weight Loss', 'Muscle Gain', 'Active Minutes', 'Step Count')

);

-- Query 43: Find the activities with the highest calories burned per hour.

-- This query uses a subquery with MAX to find the activity with the maximum calories burned per hour.

SELECT \*

FROM Activities

WHERE Calories\_burned\_per\_hour = (

    SELECT MAX(Calories\_burned\_per\_hour)

    FROM Activities

);

-- Query 44: Retrieve users who have logged a specific activity (e.g., Activity\_ID 111).

-- This query uses a subquery to find users who have performed a particular activity.

SELECT \*

FROM Users

WHERE User\_ID IN (

    SELECT User\_ID

    FROM Workout\_Logs

    WHERE Activity\_ID = 111

);

-- Query 45: Retrieve workout logs of users who have a specific goal.

-- This query uses a subquery to filter workout logs based on users' goals.

SELECT \*

FROM Workout\_Logs

WHERE User\_ID IN (

    SELECT User\_ID

    FROM Goals

    WHERE Goal\_name IN ('Weight Loss', 'Active Minutes', 'Step Count', 'Cardio Fitness')

);

-- Query 46: Get the names and average heart rate of users who have an average heart rate between 70 and 90.

-- This query uses GROUP BY and HAVING to filter users based on their average heart rate.

SELECT Users.Name, AVG(Heart\_Rate\_Logs.Heart\_rate) AS average\_heart\_rate

FROM Users

JOIN Heart\_Rate\_Logs ON Users.User\_ID = Heart\_Rate\_Logs.User\_ID

GROUP BY Users.Name

HAVING AVG(Heart\_Rate\_Logs.Heart\_rate) BETWEEN 70 AND 90;

-- Query 47: Retrieve activities that have "s" anywhere in their name and have a calorie burn rate greater than 250,

-- also joining with Blood\_Pressure\_Logs (as per PDF's example, though the join condition might be unusual).

-- NOTE: The original query in the PDF for this one seems to have an unusual join and column selection.

-- I'm reproducing it as closely as possible to the PDF's structure.

SELECT A.Activity\_ID, A.User\_ID, A.Activity\_name, A.Calories\_burned\_per\_hour,

       B.Log\_ID, B.User\_ID AS BP\_User\_ID, B.Log\_Date, B.Systolic, B.Diastolic

FROM Activities AS A

JOIN Blood\_Pressure\_Logs AS B ON A.User\_ID = B.User\_ID -- Assuming a link between activity and blood pressure logs via user

WHERE A.Activity\_name LIKE '%s%' AND A.Calories\_burned\_per\_hour > 250;

-- Query 48: Get the names of users who have logged a food item containing the word 'chicken'.

-- This query uses INNER JOIN and LIKE for pattern matching in food names.

SELECT DISTINCT u.User\_ID, U.Name, u.Weight, f.Food\_name

FROM Users AS u

INNER JOIN Food\_Intake\_Logs AS f ON U.User\_ID = F.User\_ID

WHERE f.Food\_name LIKE '%chicken%';

-- Query 49: Find all users whose names contain "r" and have made food intake.

-- This query combines LIKE with a subquery to filter users.

SELECT \*

FROM Users AS u

WHERE u.Name LIKE '%r%'

AND User\_ID IN (

    SELECT User\_ID

    FROM Food\_Intake\_Logs

);

-- Query 50: Retrieve all activities that have "run" anywhere in their name.

-- This is a duplicate of Query 35, but included as per the PDF's repetition.

SELECT \*

FROM Activities

WHERE Activity\_name LIKE '%run%';