ASSIGNMENT – 5

CSA0593 – DBMS

NAME – S.DEEPIKA

REG NO : 192324269

1. Build a database to manage workouts, exercises, trainers, and user progress.

* Model tables for exercises, workouts, trainers, and user progress.
* Write stored procedures for adding exercises to workouts and updating user workout logs.
* Implement triggers to update workout completion stats and track user progress milestones.
* Write SQL queries to analyze workout popularity, trainer effectiveness, and user fitness improvements.

ANSWER :

CONCEPTUA ER DIAGRAM:

* **Users** are linked to **UserProgress** through the **UserID**, where each progress record represents a user's performance in a **Workout**.
* **Workouts** are associated with **Trainers** by the **TrainerID**, with each workout containing multiple **Exercises**.
* **Exercises** are related to **Workouts** via a many-to-many relationship, where exercises are added to workouts.

CODE :

CREATE DATABASE WorkoutManagement;

USE WorkoutManagement;

CREATE TABLE Exercises (

ExerciseID INT AUTO\_INCREMENT PRIMARY KEY,

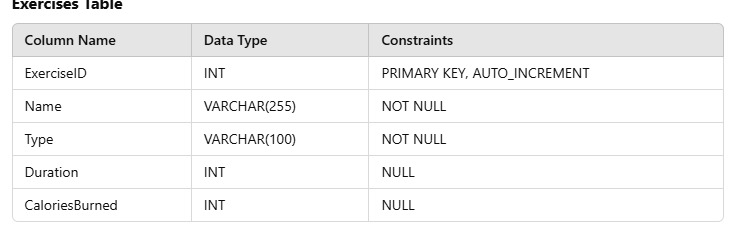
Name VARCHAR(255) NOT NULL,

Type VARCHAR(100) NOT NULL,

Duration INT, -- in minutes

CaloriesBurned INT -- Estimated calories burned

);



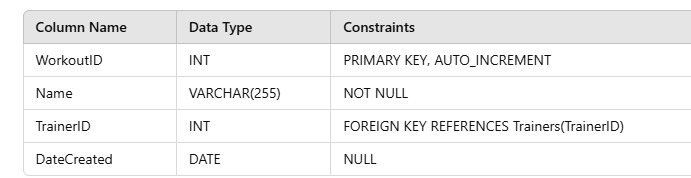
CREATE TABLE Trainers (

TrainerID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(255) NOT NULL,

Specialization VARCHAR(100) NOT NULL

);



CREATE TABLE Workouts (

WorkoutID INT AUTO\_INCREMENT PRIMARY KEY,

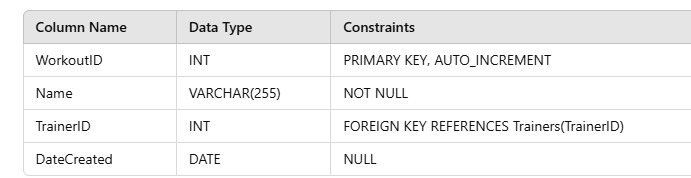
Name VARCHAR(255) NOT NULL,

TrainerID INT NOT NULL,

DateCreated DATE,

FOREIGN KEY (TrainerID) REFERENCES Trainers(TrainerID)

);



CREATE TABLE Users (

UserID INT AUTO\_INCREMENT PRIMARY KEY,

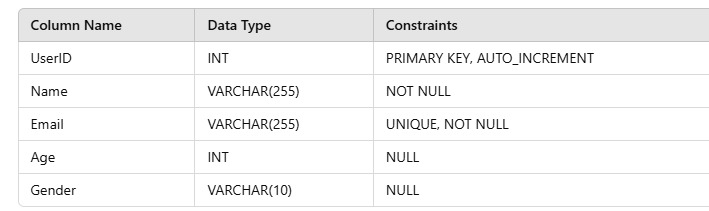
Name VARCHAR(255) NOT NULL,

Email VARCHAR(255) UNIQUE NOT NULL,

Age INT,

Gender VARCHAR(10)

);



CREATE TABLE UserProgress (

ProgressID INT AUTO\_INCREMENT PRIMARY KEY,

UserID INT NOT NULL,

WorkoutID INT NOT NULL,

Completed BOOLEAN DEFAULT FALSE,

DurationSpent INT, -- in minutes

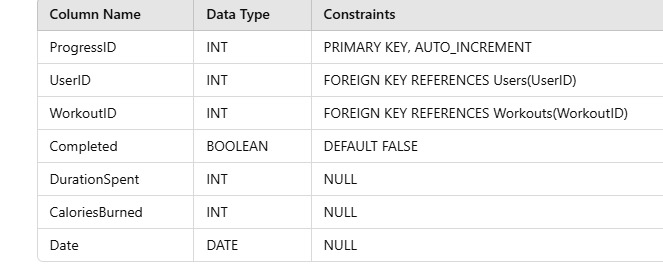
CaloriesBurned INT,

Date DATE,

FOREIGN KEY (UserID) REFERENCES Users(UserID),

FOREIGN KEY (WorkoutID) REFERENCES Workouts(WorkoutID)

);



DELIMITER //

CREATE PROCEDURE AddExerciseToWorkout (

IN p\_WorkoutID INT,

IN p\_ExerciseID INT

)

BEGIN

INSERT INTO WorkoutExercises (WorkoutID, ExerciseID)

VALUES (p\_WorkoutID, p\_ExerciseID);

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE UpdateUserProgress (

IN p\_UserID INT,

IN p\_WorkoutID INT,

IN p\_Completed BOOLEAN,

IN p\_DurationSpent INT,

IN p\_CaloriesBurned INT

)

BEGIN

INSERT INTO UserProgress (UserID, WorkoutID, Completed, DurationSpent, CaloriesBurned, Date)

VALUES (p\_UserID, p\_WorkoutID, p\_Completed, p\_DurationSpent, p\_CaloriesBurned, CURDATE());

END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER UpdateCompletionStats

AFTER INSERT ON UserProgress

FOR EACH ROW

BEGIN

IF NEW.Completed = TRUE THEN

UPDATE Workouts

SET TotalCompleted = (SELECT COUNT(\*) FROM UserProgress WHERE WorkoutID = NEW.WorkoutID AND Completed = TRUE)

WHERE WorkoutID = NEW.WorkoutID;

END IF;

END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER TrackUserMilestones

AFTER INSERT ON UserProgress

FOR EACH ROW

BEGIN

UPDATE Users

SET TotalCaloriesBurned = TotalCaloriesBurned + NEW.CaloriesBurned,

TotalDuration = TotalDuration + NEW.DurationSpent

WHERE UserID = NEW.UserID;

END //

DELIMITER ;

-- Analyzing Workout Popularity

SELECT w.Name AS Workout, COUNT(up.WorkoutID) AS CompletionCount

FROM Workouts w

JOIN UserProgress up ON w.WorkoutID = up.WorkoutID

WHERE up.Completed = TRUE

GROUP BY w.WorkoutID

ORDER BY CompletionCount DESC;

-- Analyzing Trainer Effectiveness

SELECT t.Name AS Trainer, COUNT(up.WorkoutID) AS CompletedWorkouts

FROM Trainers t

JOIN Workouts w ON t.TrainerID = w.TrainerID

JOIN UserProgress up ON w.WorkoutID = up.WorkoutID

WHERE up.Completed = TRUE

GROUP BY t.TrainerID

ORDER BY CompletedWorkouts DESC;

-- Analyzing User Fitness Improvements

SELECT u.Name AS User, SUM(up.CaloriesBurned) AS TotalCaloriesBurned, SUM(up.DurationSpent) AS TotalDuration

FROM Users u

JOIN UserProgress up ON u.UserID = up.UserID

GROUP BY u.UserID

ORDER BY TotalCaloriesBurned DESC, TotalDuration DESC;

CONCLUSION

The database effectively tracks and manages workouts, exercises, trainers, and user progress. Stored procedures automate tasks like adding exercises and updating user progress, while triggers dynamically update statistics like workout completions and user milestones. The SQL queries provide valuable insights into workout popularity, trainer effectiveness, and user fitness improvements, allowing for better management and analysis.