**Database Design Guide: Fitness Tracking System**

This guide will help the Tracker to create a database on the Fitness Tracking System. It will help to manage the below functionalities.

* Manage users (athletes)
* Track Workouts
* Log nutrition intake
* Record Progress
* Provide feedback

We will use MySQL as the DBMS to create the database and its related operations.

1. **Introduction to MySQL**: MySQL is an open-source relational database management system (RDBMS) that uses structured query language (SQL) to manage and manipulate data in a database. It is widely used for various applications, from small web applications to large enterprise systems.

MySQL's key features include:

● Scalability: Capable of handling large amounts of data and concurrent connections.

● Flexibility: Supports various data types and storage engines.

● Performance: Optimized for speed and efficiency.

● Reliability: Known for its stability and robustness.

**2**. **Installation of MySQL**: MySQL can be installed on various operating systems, including Windows, macOS, and Linux. Here are the general steps to install MySQL: Windows:

● Download the MySQL installer from the official website.

<https://dev.mysql.com/downloads/installer/>

● Run the installer and follow the on-screen instructions.

● Choose the installation type (Typical, Complete, or Custom). Recommended Custom.

● Set a root password for the MySQL server.

**3**. **E-R Diagram (ERD):** An Entity-Relationship Diagram (ERD) is a visual representation of the data model that shows the entities, attributes, relationships between entities, and cardinality. ERDs are commonly used in database design to help developers and stakeholders understand the structure and relationships within a database.

**Identify Entities** : Start by identifying the main entities in your system. These are the objects or concepts about which you want to store data. Each entity should correspond to a table in your database.

**Define Attributes** : For each entity, list the attributes (properties or fields) that describe it. These attributes will become columns in the corresponding database table.

**Cardinality Notation** : Cardinality represents the number of times an entity of an entity set participates in a relationship set. Or we can say that the cardinality of a relationship is the number of tuples (rows) in a relationship. ● Use notation (such as Crow's Foot Notation or Chen Notation) to indicate the cardinality of each relationship. ● Cardinality describes how many instances of one entity are related to how many instances of another entity. ● Common notations include: ★ One (1) ★ Zero or one (0..1) ★ Many (N) ★ Zero or many (0..N).

Let’s identify the entities of Fitness tracking system.

1. User
2. Workout
3. Nutrition
4. Progress
5. Feedback

\*\*\* Now let’s identify the attributes and relationships of each entity for the Student Management System.

**1.USER:**

* Attributes:
  + UserID (Primary Key)
  + FirstName
  + LastName
  + DateOfBirth
  + Gender
  + Email
  + Phone
* Relationships:
  + One User can have multiple Workouts (One-to-Many)

**2.Workout**

* Attributes:
  + WorkoutID (Primary Key)
  + UserID (Foreign Key)
  + WorkoutType
  + Duration
  + Date
* Relationships:
  + Many Workouts for one User (Many-to-One)

**3.Nutrition**

* Attributes:
  + NutritionID (Primary Key)
  + UserID (Foreign Key)
  + MealType
  + MealDescription
  + Date
* Relationships:
  + Many Nutrition logs for one User (Many-to-One)

**4.Progress**

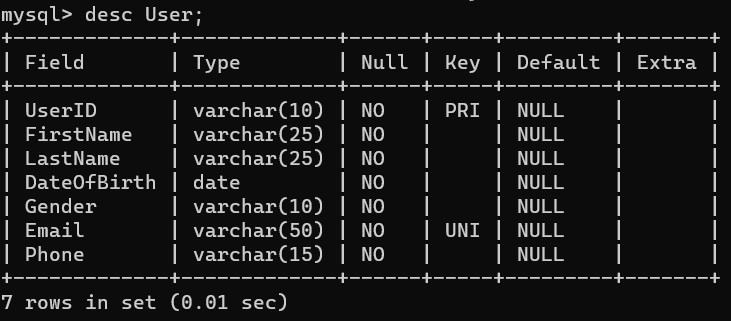
* Attributes:
  + ProgressID (Primary Key)
  + UserID (Foreign Key)
  + Weight
  + BodyFatPercentage
  + Date
* Relationships:
  + Many Progress records for one User (Many-to-One)

**5.Feedback**

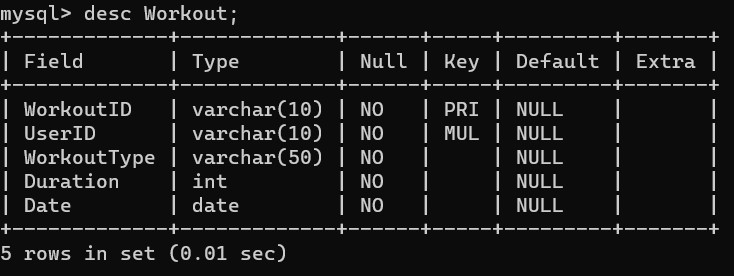
* Attributes:
  + FeedbackID (Primary Key)
  + UserID (Foreign Key)
  + Date
  + FeedbackMessage
* Relationships:
  + One User can provide multiple Feedbacks (One-to-Many)

**TABLE STRUCTURE:**

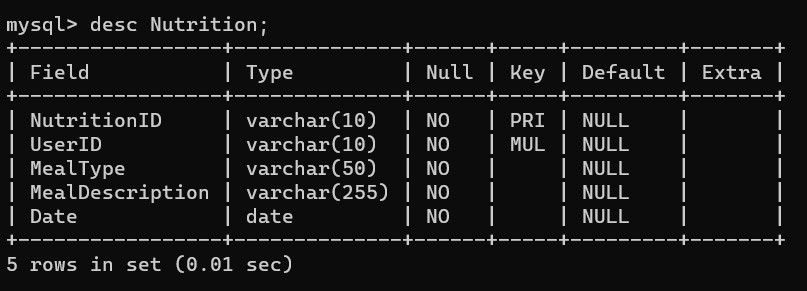
**USER:**



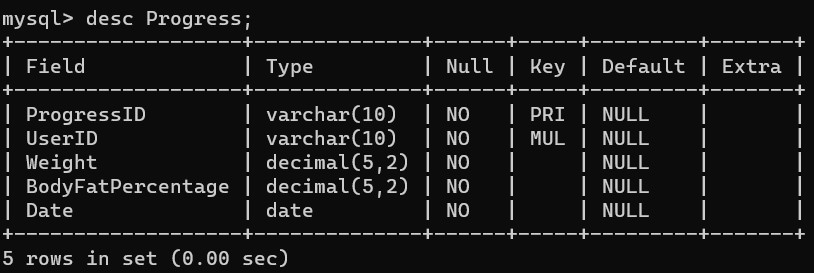
**Workout:**



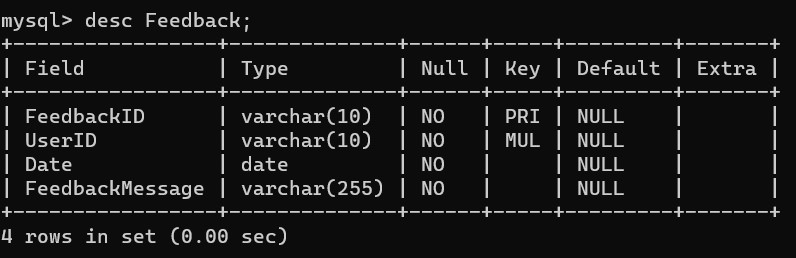
**Nutrition:**



**Progress:**



**Feedback:**



**4. Creating a Database :**

Using MySQL server, create a new database for your student management system. You can do this with SQL commands or through the graphical interface.

CREATE DATABASE FitnessTrackingSystem;

**5.Using a Database :**

Before performing any operations on a database, you need to select it using the

USE FitnessTrackingSystem;

**6. CREATING TABLES FOR EACH ENTITY:**

**->User table:**

CREATE TABLE User (

UserID VARCHAR(10) PRIMARY KEY,

FirstName VARCHAR(25) NOT NULL,

LastName VARCHAR(25) NOT NULL,

DateOfBirth DATE NOT NULL,

Gender VARCHAR(10) NOT NULL,

Email VARCHAR(50) UNIQUE NOT NULL,

Phone VARCHAR(15) NOT NULL

);

**->Workout table:**

CREATE TABLE Workout (

WorkoutID VARCHAR(10) PRIMARY KEY,

UserID VARCHAR(10) NOT NULL,

WorkoutType VARCHAR(50) NOT NULL,

Duration INT NOT NULL,

Date DATE NOT NULL,

FOREIGN KEY (UserID) REFERENCES User(UserID)

);

->**Nutrition table:**

CREATE TABLE Nutrition (

NutritionID VARCHAR(10) PRIMARY KEY,

UserID VARCHAR(10) NOT NULL,

MealType VARCHAR(50) NOT NULL,

MealDescription VARCHAR(255) NOT NULL,

Date DATE NOT NULL,

FOREIGN KEY (UserID) REFERENCES User(UserID)

);

**->Progress table:**

CREATE TABLE Progress (

ProgressID VARCHAR(10) PRIMARY KEY,

UserID VARCHAR(10) NOT NULL,

Weight DECIMAL(5,2) NOT NULL,

BodyFatPercentage DECIMAL(5,2) NOT NULL,

Date DATE NOT NULL,

FOREIGN KEY (UserID) REFERENCES User(UserID)

);

**->Feedback table:**

CREATE TABLE Feedback (

FeedbackID VARCHAR(10) PRIMARY KEY,

UserID VARCHAR(10) NOT NULL,

Date DATE NOT NULL,

FeedbackMessage VARCHAR(255) NOT NULL,

FOREIGN KEY (UserID) REFERENCES User(UserID)

);

**7. INSERT RECORDS:**

Add data to your tables to work with. This step helps you test your database.

**->insert user**

INSERT INTO User (UserID, FirstName, LastName, DateOfBirth, Gender, Email, Phone)

VALUES

('U101', 'vaidya', 'Divija', '2000-01-11', 'Female', 'vaidyadivija@email.com', '8763671282'),

('U102', 'Jakkula', 'Priyanka', '2002-09-20', 'Female', 'jakkulapriyanka@email.com', '9465243786');

**->insert workout**

INSERT INTO Workout (WorkoutID, UserID, WorkoutType, Duration, Date)

VALUES

('W101', 'U101', 'Running', 30, '2024-03-10'),

('W102', 'U101', 'Weightlifting', 60, '2024-03-12');

**->insert Nutrition**

INSERT INTO Nutrition (NutritionID, UserID, MealType, MealDescription, Date)

VALUES

('N101', 'U101', 'Breakfast', 'Oatmeal with fruits', '2024-03-10'),

('N102', 'U101', 'Lunch', 'Grilled chicken salad', '2024-03-10');

**->insert Progress**

INSERT INTO Progress (ProgressID, UserID, Weight, BodyFatPercentage, Date)

VALUES

('P101', 'U101', 75.5, 20.0, '2024-03-10'),

('P102', 'U101', 74.0, 18.5, '2024-03-15');

**->insert Feedback**

INSERT INTO Feedback (FeedbackID, UserID, Date, FeedbackMessage)

VALUES

('F101', 'U101', '2024-03-10', 'Great progress! Keep it up.'),

('F102', 'U101', '2024-03-15', 'You're doing amazing!');

**8. Select records :**

Write SQL queries to retrieve and manage data.

For example: Retrieve all user

Select \* FROM User;

**Retrieve a user’s worksout:**

SELECT WorkoutID, Date, Duration

FROM Workout

WHERE UserID = 'user\_id';

**\*Now try similar Select queries with other tables**

**Retrieve a user's exercises in a workout:**

SELECT Exercise.ExerciseName, ExerciseLog.Sets, ExerciseLog.Reps, ExerciseLog.Weight

FROM ExerciseLog

INNER JOIN Exercise ON ExerciseLog.ExerciseID = Exercise.ExerciseID

WHERE ExerciseLog.WorkoutID = 'workout\_id';

**Update a user's weight:**

UPDATE User

SET Weight = 'new\_weight'

WHERE UserID = 'user\_id';

**Delete a specific workout:**

DELETE FROM Workout

WHERE WorkoutID = 'workout\_id';

PN: Ideally no data should be deleted from any tables. You can use an additional column to set the status of that record to ‘Active/Inactive’, etc. Or you can use an Archive table to move the unecessary records out of the main table