Personal Fitness & Workout Log: Complete Project Documentation 💪📊

This document provides a comprehensive overview of your **Personal Fitness & Workout Log SQL Project**, detailing every aspect from database design and SQL implementation to the final data visualization. This project showcases your skills in relational database design, SQL (DDL, DML, DQL), and basic data visualization using HTML, CSS, and JavaScript.

🎯 Project Overview

The core objective of this project is to create a robust and efficient database system for individuals to **track their fitness journey**. This includes logging user profiles, specific exercises, workout sessions, and detailed performance metrics such as repetitions, weight lifted, distance covered, and estimated calories burned. The database is designed for easy data entry, retrieval, and analysis to help users understand their workout habits and monitor their progress over time. The project culminates in a simple HTML page that visually presents key fitness insights.

📊 Section 1: Database Design (SQL Schema)

Your database is composed of four main tables, each designed to store specific information and linked through keys to form a cohesive system.

1. Users Table 👩‍💻

- **Purpose:** Stores essential profile information for each individual using the fitness log.
- **Columns:**
  - user_id: **Primary Key (PK)**. Unique identifier for each user.
  - username: Unique name chosen by the user (e.g., 'rutvik', 'vamshi').
  - email: User's email address, unique for each user.
  - join_date: The date the user started logging (YYYY-MM-DD).

2. Exercises Table 🏋️‍♀️

- **Purpose:** Catalogs the different types of exercises that can be performed, standardizing names and classifications.
- **Columns:**
  - exercise_id: **Primary Key (PK)**. Unique identifier for each exercise.
  - exercise_name: The common name of the exercise (e.g., 'Bench Press', 'Running'), unique for each exercise.
  - exercise_type: A broader category (e.g., 'Strength', 'Cardio', 'Flexibility').
  - target_muscle_group: The primary muscle group targeted (e.g., 'Chest', 'Legs',

'Core').

3. Workouts Table 🗓️

- **Purpose:** Stores information about each individual workout session (e.g., a specific gym session on a particular date and time).
- **Columns:**
  - workout_id: **Primary Key (PK)**. Unique identifier for each workout session.
  - user_id: **Foreign Key (FK)**. Links to Users.user_id. This identifies which user performed this workout.
  - workout_date: The specific date of the workout (YYYY-MM-DD).
  - start_time: The start time of the workout (HH:MM:SS).
  - end_time: The end time of the workout (HH:MM:SS).
  - duration_minutes: The total duration of the workout in minutes.

4. Workout_Details Table 📊

- **Purpose:** This is a crucial **associative table** that records specific performance metrics for each exercise performed within a workout. It resolves the **many-to-many relationship** between Workouts and Exercises.
- **Columns:**
  - detail_id: **Primary Key (PK)**. Unique identifier for each specific detail entry (e.g., one set of an exercise).
  - workout_id: **Foreign Key (FK)**. Links to Workouts.workout_id. This identifies which workout session this detail belongs to.
  - exercise_id: **Foreign Key (FK)**. Links to Exercises.exercise_id. This identifies the specific exercise performed.
  - set_number: The sequential number of the set for this exercise within the workout.
  - reps: Number of repetitions performed for this set.
  - weight_kg: Weight lifted in kilograms (for strength exercises).
  - distance_km: Distance covered in kilometers (for cardio exercises).
  - calories_burned: Estimated calories burned for this specific set/exercise instance.
- **Key Detail:** detail_id uniquely identifies each performance record. The combination of workout_id, exercise_id, and set_number is also enforced as **unique** to prevent duplicate entries for the same set of an exercise in a workout.

Relationships Summary

- **Users to Workouts (One-to-Many):** One user can perform many workouts.
- **Workouts to Workout_Details (One-to-Many):** One workout session can contain many individual exercise details (sets).
- **Exercises to Workout_Details (One-to-Many):** One exercise can appear in many workout details (sets).
- **Workouts and Exercises (Many-to-Many):** This relationship is resolved by the Workout_Details table.

🛠️ Section 2: SQL Database Implementation (DDL & DML)

This section provides the complete SQL script to create your database schema and populate it with sample data.

2.1 Database Creation (DDL - Data Definition Language)

These CREATE TABLE statements define the structure of your database.

```
-- 1. Users Table
CREATE TABLE Users (
    user_id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT NOT NULL UNIQUE,
    email TEXT UNIQUE,
    join_date TEXT NOT NULL -- YYYY-MM-DD
);

-- 2. Exercises Table
CREATE TABLE Exercises (
    exercise_id INTEGER PRIMARY KEY AUTOINCREMENT,
    exercise_name TEXT NOT NULL UNIQUE,
    exercise_type TEXT, -- e.g., 'Strength', 'Cardio', 'Flexibility'
    target_muscle_group TEXT -- e.g., 'Chest', 'Legs', 'Core'
);

-- 3. Workouts Table
CREATE TABLE Workouts (
    workout_id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    workout_date TEXT NOT NULL, -- YYYY-MM-DD
    start_time TEXT, -- HH:MM:SS
    end_time TEXT, -- HH:MM:SS
    duration_minutes INTEGER,
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);

-- 4. Workout_Details Table
CREATE TABLE Workout_Details (
    detail_id INTEGER PRIMARY KEY AUTOINCREMENT,
    workout_id INTEGER NOT NULL,
    exercise_id INTEGER NOT NULL,
    set_number INTEGER NOT NULL,
```

```
    reps INTEGER,
    weight_kg REAL,
    distance_km REAL,
    calories_burned REAL, -- Estimated calories burned for this specific set/exercise instance
    UNIQUE(workout_id, exercise_id, set_number), -- Ensures unique set entries per exercise
per workout
    FOREIGN KEY (workout_id) REFERENCES Workouts(workout_id),
    FOREIGN KEY (exercise_id) REFERENCES Exercises(exercise_id)
);
```

2.2 Sample Data Population (DML - Data Manipulation Language)

These INSERT INTO statements populate your tables with sample data for users 'rutvik', 'vamshi', 'priya', 'ramya', various exercises, and their workout details.

```
-- Insert Sample Users
INSERT INTO Users (username, email, join_date) VALUES
('rutvik', 'rutvik@example.com', '2024-01-01'),
('vamshi', 'vamshi@example.com', '2024-02-15'),
('priya', 'priya@example.com', '2024-03-01'),
('ramya', 'ramya@example.com', '2024-04-05');

-- Insert Sample Exercises
INSERT INTO Exercises (exercise_name, exercise_type, target_muscle_group) VALUES
('Bench Press', 'Strength', 'Chest'),
('Squat', 'Strength', 'Legs'),
('Deadlift', 'Strength', 'Full Body'),
('Running', 'Cardio', 'Legs'),
('Plank', 'Flexibility', 'Core'),
('Overhead Press', 'Strength', 'Shoulders');

-- Insert Sample Workouts for 'rutvik'
INSERT INTO Workouts (user_id, workout_date, start_time, end_time, duration_minutes)
VALUES
((SELECT user_id FROM Users WHERE username = 'rutvik'), '2024-08-15', '08:00:00',
'09:00:00', 60),
((SELECT user_id FROM Users WHERE username = 'rutvik'), '2024-08-17', '18:30:00', '19:15:00',
45),
((SELECT user_id FROM Users WHERE username = 'rutvik'), '2024-09-01', '09:00:00',
'09:45:00', 45); -- Latest entry for Rutvik
```

```
-- Insert Sample Workouts for 'vamshi'
INSERT INTO Workouts (user_id, workout_date, start_time, end_time, duration_minutes)
VALUES
((SELECT user_id FROM Users WHERE username = 'vamshi'), '2024-08-16', '17:00:00',
'18:00:00', 60),
((SELECT user_id FROM Users WHERE username = 'vamshi'), '2024-08-18', '06:30:00',
'07:15:00', 45);


-- Insert Sample Workout Details (adjusted for 'calories_burned' column)
-- Rutvik's Workout 1 (2024-08-15)
INSERT INTO Workout_Details (workout_id, exercise_id, set_number, reps, weight_kg,
distance_km, calories_burned) VALUES
((SELECT workout_id FROM Workouts WHERE user_id = (SELECT user_id FROM Users WHERE
username = 'rutvik') AND workout_date = '2024-08-15'), (SELECT exercise_id FROM Exercises
WHERE exercise_name = 'Bench Press'), 1, 8, 60.0, NULL, 50.0),
((SELECT workout_id FROM Workouts WHERE user_id = (SELECT user_id FROM Users WHERE
username = 'rutvik') AND workout_date = '2024-08-15'), (SELECT exercise_id FROM Exercises
WHERE exercise_name = 'Bench Press'), 2, 8, 62.5, NULL, 55.0),
((SELECT workout_id FROM Workouts WHERE user_id = (SELECT user_id FROM Users WHERE
username = 'rutvik') AND workout_date = '2024-08-15'), (SELECT exercise_id FROM Exercises
WHERE exercise_name = 'Squat'), 1, 10, 70.0, NULL, 80.0),
((SELECT workout_id FROM Workouts WHERE user_id = (SELECT user_id FROM Users WHERE
username = 'rutvik') AND workout_date = '2024-08-15'), (SELECT exercise_id FROM Exercises
WHERE exercise_name = 'Running'), 1, NULL, NULL, 5.0, 300.0);

-- Rutvik's Workout 2 (2024-08-17)
INSERT INTO Workout_Details (workout_id, exercise_id, set_number, reps, weight_kg,
distance_km, calories_burned) VALUES
((SELECT workout_id FROM Workouts WHERE user_id = (SELECT user_id FROM Users WHERE
username = 'rutvik') AND workout_date = '2024-08-17'), (SELECT exercise_id FROM Exercises
WHERE exercise_name = 'Deadlift'), 1, 5, 80.0, NULL, 70.0),
((SELECT workout_id FROM Workouts WHERE user_id = (SELECT user_id FROM Users WHERE
username = 'rutvik') AND workout_date = '2024-08-17'), (SELECT exercise_id FROM Exercises
WHERE exercise_name = 'Plank'), 1, NULL, NULL, NULL, 20.0);

-- Rutvik's Workout 3 (2024-09-01) - Latest entry
INSERT INTO Workout_Details (workout_id, exercise_id, set_number, reps, weight_kg,
distance_km, calories_burned) VALUES
((SELECT workout_id FROM Workouts WHERE user_id = (SELECT user_id FROM Users WHERE
username = 'rutvik') AND workout_date = '2024-09-01'), (SELECT exercise_id FROM Exercises
WHERE exercise_name = 'Bench Press'), 1, 8, 60.0, NULL, 50.0),
((SELECT workout_id FROM Workouts WHERE user_id = (SELECT user_id FROM Users WHERE
```

```
username = 'rutvik') AND workout_date = '2024-09-01'), (SELECT exercise_id FROM Exercises
WHERE exercise_name = 'Bench Press'), 2, 6, 65.0, NULL, 55.0);


-- Vamshi's Workout 1 (2024-08-16)
INSERT INTO Workout_Details (workout_id, exercise_id, set_number, reps, weight_kg,
distance_km, calories_burned) VALUES
((SELECT workout_id FROM Workouts WHERE user_id = (SELECT user_id FROM Users WHERE
username = 'vamshi') AND workout_date = '2024-08-16'), (SELECT exercise_id FROM
Exercises WHERE exercise_name = 'Overhead Press'), 1, 8, 40.0, NULL, 45.0),
((SELECT workout_id FROM Workouts WHERE user_id = (SELECT user_id FROM Users WHERE
username = 'vamshi') AND workout_date = '2024-08-16'), (SELECT exercise_id FROM
Exercises WHERE exercise_name = 'Overhead Press'), 2, 8, 42.5, NULL, 50.0),
((SELECT workout_id FROM Workouts WHERE user_id = (SELECT user_id FROM Users WHERE
username = 'vamshi') AND workout_date = '2024-08-16'), (SELECT exercise_id FROM
Exercises WHERE exercise_name = 'Running'), 1, NULL, NULL, 6.5, 400.0);

-- Vamshi's Workout 2 (2024-08-18)
INSERT INTO Workout_Details (workout_id, exercise_id, set_number, reps, weight_kg,
distance_km, calories_burned) VALUES
((SELECT workout_id FROM Workouts WHERE user_id = (SELECT user_id FROM Users WHERE
username = 'vamshi') AND workout_date = '2024-08-18'), (SELECT exercise_id FROM
Exercises WHERE exercise_name = 'Squat'), 1, 10, 60.0, NULL, 70.0),
((SELECT workout_id FROM Workouts WHERE user_id = (SELECT user_id FROM Users WHERE
username = 'vamshi') AND workout_date = '2024-08-18'), (SELECT exercise_id FROM
Exercises WHERE exercise_name = 'Deadlift'), 1, 5, 75.0, NULL, 65.0);
```

📈 Section 3: SQL Analytical Queries (DQL - Data Query Language)

These SELECT queries are used to extract meaningful insights from the database.

1. View All Workouts with User and Basic Details

```
SELECT
    U.username,
    W.workout_date,
    W.start_time,
    W.end_time,
    W.duration_minutes
FROM Workouts AS W
```

```
JOIN Users AS U ON W.user_id = U.user_id
ORDER BY W.workout_date DESC, W.start_time DESC;
```

2. Get Details of a Specific Workout (e.g., Rutvik's workout on '2024-08-15')

```
SELECT
    U.username,
    W.workout_date,
    E.exercise_name,
    WD.set_number,
    WD.reps,
    WD.weight_kg,
    WD.distance_km,
    WD.calories_burned
FROM Workout_Details AS WD
JOIN Workouts AS W ON WD.workout_id = W.workout_id
JOIN Users AS U ON W.user_id = U.user_id
JOIN Exercises AS E ON WD.exercise_id = E.exercise_id
WHERE U.username = 'rutvik' AND W.workout_date = '2024-08-15'
ORDER BY E.exercise_name, WD.set_number;
```

3. Calculate Total Workouts per User

```
SELECT
    U.username,
    COUNT(W.workout_id) AS total_workouts_logged
FROM Users AS U
LEFT JOIN Workouts AS W ON U.user_id = W.user_id
GROUP BY U.username
ORDER BY total_workouts_logged DESC;
```

4. Find Maximum Weight Lifted for 'Bench Press' by Each User (Max per day)

```
SELECT
    U.username,
    W.workout_date,
```

```
    MAX(WD.weight_kg) AS max_weight_kg
FROM Workout_Details AS WD
JOIN Workouts AS W ON WD.workout_id = W.workout_id
JOIN Users AS U ON W.user_id = U.user_id
JOIN Exercises AS E ON WD.exercise_id = E.exercise_id
WHERE U.username = 'rutvik' AND E.exercise_name = 'Bench Press' AND WD.weight_kg IS
NOT NULL
GROUP BY U.username, W.workout_date
ORDER BY W.workout_date;
```

5. Track Running Performance (Total Distance and Calories Burned) for Each User

```
SELECT
    U.username,
    E.exercise_name,
    SUM(WD.distance_km) AS total_distance_km,
    SUM(WD.calories_burned) AS total_calories_burned_running
FROM Workout_Details AS WD
JOIN Exercises AS E ON WD.exercise_id = E.exercise_id
JOIN Workouts AS W ON WD.workout_id = W.workout_id
JOIN Users AS U ON W.user_id = U.user_id
WHERE E.exercise_name = 'Running'
GROUP BY U.username, E.exercise_name
ORDER BY total_distance_km DESC;
```

6. Calculate Total Calories Burned per Workout for Each User

```
SELECT
    U.username,
    W.workout_date,
    SUM(WD.calories_burned) AS total_calories_burned_workout
FROM Workout_Details AS WD
JOIN Workouts AS W ON WD.workout_id = W.workout_id
JOIN Users AS U ON W.user_id = U.user_id
GROUP BY U.username, W.workout_date
ORDER BY U.username, W.workout_date;
```

🌐 Section 4: Data Visualization Page (HTML/CSS/JS) - Idea to be Implemented

This section describes the conceptual index.html page, which would serve as a simple visualization dashboard. It outlines the idea of how a frontend could visually present key fitness metrics (calories burned per workout and Bench Press progression) in an easy-to-understand format.

4.1 Purpose and Implementation Idea

- **Purpose:** To visually present key fitness metrics (calories burned per workout and Bench Press progression) in an easy-to-understand format.
- **Implementation Idea:** This page would be developed using **HTML, CSS, and JavaScript with the Chart.js library**. In a fully dynamic version, this page would connect to a backend API (like a Python Flask server) that queries the fitness_log.db database. For a self-contained demonstration, the JavaScript would contain **hardcoded arrays** that simulate the data fetched from the database. This allows the visualization to function without a live backend, making it easy to demonstrate the visual output of the SQL analysis.
- **Components:**
  - **HTML Structure:** Provides the layout, titles, and <canvas> elements where charts are drawn.
  - **CSS Styling:** Custom styles (without Tailwind) ensure a clean, responsive, and aesthetically pleasing design.
  - **JavaScript Logic:**
    - Would contain hardcoded data for Rutvik's calories and Bench Press progression (as a simulation).
    - Would use the **Chart.js library** (loaded via CDN) to create and manage the bar and line charts.
    - The renderCharts() function would be called when the page loads (DOMContentLoaded) to draw the graphs using the embedded data.

4.2 Conceptual HTML/CSS/JS Code for the Visualization Page

This is the conceptual code for the index.html file, demonstrating the structure and JavaScript logic for rendering the charts with simulated data.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fitness Data Analysis - Latest Data</title>
    <link
```

```
href="https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&display=swap"
rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/chart.js@4.4.3/dist/chart.umd.min.js"></script>
    <style>
        body {
            margin: 0; padding: 1rem; font-family: 'Inter', sans-serif;
            background-color: #f3f4f6; color: #374151;
            display: flex; justify-content: center; align-items: flex-start; min-height: 100vh;
        }
        .container {
            width: 100%; max-width: 64rem; background-color: #fff;
            border-radius: 0.75rem; box-shadow: 0 10px 15px rgba(0,0,0,0.1);
            padding: 1.5rem; margin-bottom: 2rem;
        }
        h1 {
            font-size: 2.25rem; font-weight: 700; text-align: center; color: #4f46e5;
            margin-bottom: 1.5rem;
        }
        p {
            text-align: center; color: #4b5563; margin-bottom: 2rem;
        }
        /* Removed .user-select-block styling as the block is removed */
        .chart-card {
            padding: 1rem; border-radius: 0.5rem; box-shadow: 0 1px 3px rgba(0,0,0,0.1);
            margin-bottom: 2rem;
        }
        .chart-card h2 {
            font-size: 1.25rem; font-weight: 600; text-align: center; margin-bottom: 1rem;
        }
        .chart-card.green { background-color: #ecfdf5; }
        .chart-card.green h2 { color: #059669; }
        .chart-card.purple { background-color: #f5f3ff; }
        .chart-card.purple h2 { color: #7c3aed; }
        .chart-canvas-container {
            position: relative; height: 300px; width: 100%;
        }
        canvas { max-width: 100% !important; height: 100% !important; }
        /* Removed error-message styling as it's no longer needed */
    </style>
</head>
<body>
    <div class="container">
        <h1>🏋️ Personal Fitness Data Analysis 📈</h1>
```

```
        <p>Displaying latest fitness data for Rutvik.</p>

        <!-- User Selection Block Removed -->

        <div class="chart-card green">
            <h2>Calories Burned per Workout Session (Rutvik)</h2>
            <div class="chart-canvas-container"><canvas id="caloriesChart"></canvas></div>
        </div>

        <div class="chart-card purple">
            <h2>Rutvik's Bench Press Weight Progression (Max per day)</h2>
            <div class="chart-canvas-container"><canvas id="benchPressChart"></canvas></div>
        </div>
    </div>

    <script>
        // --- Hardcoded Simulated Data for Rutvik ---
        // This data represents the latest results of your SQL queries for Rutvik.
        const rutvikCaloriesPerWorkoutData = [
            { workout_date: '2024-08-15', total_calories_burned_workout: 485.0 },
            { workout_date: '2024-08-17', total_calories_burned_workout: 90.0 },
            { workout_date: '2024-09-01', total_calories_burned_workout: 50.0 }, // Example latest
entry
        ];

        const rutvikBenchPressProgression = [
            { workout_date: '2024-08-15', max_weight_kg: 62.5 },
            { workout_date: '2024-08-17', max_weight_kg: 60.0 },
            { workout_date: '2024-08-20', max_weight_kg: 32.5 },
            { workout_date: '2024-08-25', max_weight_kg: 42.5 },
            { workout_date: '2024-08-28', max_weight_kg: 45.0 },
            { workout_date: '2024-09-01', max_weight_kg: 60.0 }, // Example latest entry
        ];

        let caloriesChart, benchPressChart;

        function renderCharts() {
            // --- Render Calories Burned Chart ---
            if (caloriesChart) caloriesChart.destroy();
            caloriesChart = new Chart(document.getElementById('caloriesChart').getContext('2d'),
{
                type: 'bar',
                data: {
```

```
                labels: rutvikCaloriesPerWorkoutData.map(d => d.workout_date),
                datasets: [{ label: 'Calories Burned', data: rutvikCaloriesPerWorkoutData.map(d =>
d.total_calories_burned_workout), backgroundColor: '#10b981', borderRadius: 10 }]
            },
            options: {
                responsive: true, maintainAspectRatio: false,
                plugins: { legend: { display: false }, tooltip: { callbacks: { label: c => c.dataset.label +
': ' + c.raw + ' kcal' } } },
                scales: { x: { title: { display: true, text: 'Workout Date' } }, y: { title: { display: true,
text: 'Calories Burned (kcal)' }, beginAtZero: true } }
            }
        });

        // --- Render Bench Press Progression Chart ---
        if (benchPressChart) benchPressChart.destroy();
        benchPressChart = new
Chart(document.getElementById('benchPressChart').getContext('2d'), {
            type: 'line',
            data: {
                labels: rutvikBenchPressProgression.map(d => d.workout_date),
                datasets: [{
                    label: 'Max Weight (kg)', data: rutvikBenchPressProgression.map(d =>
d.max_weight_kg),
                    borderColor: '#8884d8', backgroundColor: 'rgba(136, 132, 216, 0.2)',
                    fill: true, tension: 0.4, pointRadius: 6, pointHoverRadius: 8,
                }]
            },
            options: {
                responsive: true, maintainAspectRatio: false,
                plugins: { legend: { display: false }, tooltip: { callbacks: { label: c => c.dataset.label +
': ' + c.raw + ' kg' } } },
                scales: { x: { title: { display: true, text: 'Workout Date' } }, y: { title: { display: true,
text: 'Max Weight (kg)' }, beginAtZero: true } }
            }
        });
    }

    document.addEventListener('DOMContentLoaded', renderCharts); // Render charts on
page load
  </script>
</body>
</html>
```