```
/*
  # Wellness Hub Database Schema

  1. New Tables
    - `users`
      - `id` (uuid, primary key) - matches auth.users.id
      - `email` (text, unique)
      - `full_name` (text)
      - `username` (text, unique)
      - `points` (integer, default 0)
      - `created_at` (timestamp)

    - `habits`
      - `id` (uuid, primary key)
      - `title` (text)
      - `description` (text, optional)
      - `user_id` (uuid, foreign key)
      - `created_at` (timestamp)

    - `habit_completions`
      - `id` (uuid, primary key)
      - `habit_id` (uuid, foreign key)
      - `completed_at` (timestamp)
      - `points_earned` (integer)

    - `expenses`
      - `id` (uuid, primary key)
      - `amount` (decimal)
      - `category` (text)
      - `description` (text)
      - `user_id` (uuid, foreign key)
      - `created_at` (timestamp)

    - `mood_entries`
      - `id` (uuid, primary key)
      - `mood` (integer, 1-5 scale)
      - `note` (text, optional)
      - `user_id` (uuid, foreign key)
      - `created_at` (timestamp)

    - `pomodoro_sessions`
      - `id` (uuid, primary key)
      - `duration` (integer, minutes)
      - `completed` (boolean)
      - `user_id` (uuid, foreign key)
      - `points_earned` (integer)
      - `created_at` (timestamp)

  2. Security
    - Enable RLS on all tables
    - Add policies for authenticated users to manage their own data
*/

-- Users table (extends auth.users)
CREATE TABLE IF NOT EXISTS users (
  id uuid PRIMARY KEY REFERENCES auth.users(id) ON DELETE CASCADE,
  email text UNIQUE NOT NULL,
  full_name text NOT NULL,
  username text UNIQUE NOT NULL,
  points integer DEFAULT 0,
  created_at timestamptz DEFAULT now()
);
```

```
-- Habits table
CREATE TABLE IF NOT EXISTS habits (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  title text NOT NULL,
  description text,
  user_id uuid REFERENCES users(id) ON DELETE CASCADE NOT NULL,
  created_at timestamptz DEFAULT now()
);

-- Habit completions table
CREATE TABLE IF NOT EXISTS habit_completions (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  habit_id uuid REFERENCES habits(id) ON DELETE CASCADE NOT NULL,
  completed_at timestamptz DEFAULT now(),
  points_earned integer DEFAULT 10
);

-- Expenses table
CREATE TABLE IF NOT EXISTS expenses (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  amount decimal(10,2) NOT NULL,
  category text NOT NULL,
  description text NOT NULL,
  user_id uuid REFERENCES users(id) ON DELETE CASCADE NOT NULL,
  created_at timestamptz DEFAULT now()
);

-- Mood entries table
CREATE TABLE IF NOT EXISTS mood_entries (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
```

```sql
  mood integer NOT NULL CHECK (mood >= 1 AND mood <= 5),
  note text,
  user_id uuid REFERENCES users(id) ON DELETE CASCADE NOT NULL,
  created_at timestamptz DEFAULT now()
);

-- Pomodoro sessions table
CREATE TABLE IF NOT EXISTS pomodoro_sessions (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  duration integer DEFAULT 25,
  completed boolean DEFAULT false,
  user_id uuid REFERENCES users(id) ON DELETE CASCADE NOT NULL,
  points_earned integer DEFAULT 25,
  created_at timestamptz DEFAULT now()
);

-- Enable Row Level Security
ALTER TABLE users ENABLE ROW LEVEL SECURITY;
ALTER TABLE habits ENABLE ROW LEVEL SECURITY;
ALTER TABLE habit_completions ENABLE ROW LEVEL SECURITY;
ALTER TABLE expenses ENABLE ROW LEVEL SECURITY;
ALTER TABLE mood_entries ENABLE ROW LEVEL SECURITY;
ALTER TABLE pomodoro_sessions ENABLE ROW LEVEL SECURITY;

-- Users policies
CREATE POLICY "Users can read own data"
  ON users
  FOR SELECT
  TO authenticated
  USING (auth.uid() = id);
```

```
CREATE POLICY "Users can update own data"
  ON users
  FOR UPDATE
  TO authenticated
  USING (auth.uid() = id);

CREATE POLICY "Users can insert own data"
  ON users
  FOR INSERT
  TO authenticated
  WITH CHECK (auth.uid() = id);

-- Habits policies
CREATE POLICY "Users can read own habits"
  ON habits
  FOR SELECT
  TO authenticated
  USING (auth.uid() = user_id);

CREATE POLICY "Users can insert own habits"
  ON habits
  FOR INSERT
  TO authenticated
  WITH CHECK (auth.uid() = user_id);

CREATE POLICY "Users can update own habits"
  ON habits
  FOR UPDATE
  TO authenticated
  USING (auth.uid() = user_id);
```

```sql
CREATE POLICY "Users can delete own habits"
  ON habits
  FOR DELETE
  TO authenticated
  USING (auth.uid() = user_id);

-- Habit completions policies
CREATE POLICY "Users can read own habit completions"
  ON habit_completions
  FOR SELECT
  TO authenticated
  USING (EXISTS (
    SELECT 1 FROM habits
    WHERE habits.id = habit_completions.habit_id
    AND habits.user_id = auth.uid()
  ));

CREATE POLICY "Users can insert own habit completions"
  ON habit_completions
  FOR INSERT
  TO authenticated
  WITH CHECK (EXISTS (
    SELECT 1 FROM habits
    WHERE habits.id = habit_completions.habit_id
    AND habits.user_id = auth.uid()
  ));

CREATE POLICY "Users can delete own habit completions"
  ON habit_completions
  FOR DELETE
  TO authenticated
  USING (EXISTS (
    SELECT 1 FROM habits
    WHERE habits.id = habit_completions.habit_id
    AND habits.user_id = auth.uid()
  ));

-- Expenses policies
CREATE POLICY "Users can read own expenses"
  ON expenses
  FOR SELECT
  TO authenticated
  USING (auth.uid() = user_id);

CREATE POLICY "Users can insert own expenses"
  ON expenses
  FOR INSERT
  TO authenticated
  WITH CHECK (auth.uid() = user_id);

CREATE POLICY "Users can update own expenses"
  ON expenses
  FOR UPDATE
  TO authenticated
  USING (auth.uid() = user_id);

CREATE POLICY "Users can delete own expenses"
  ON expenses
  FOR DELETE
  TO authenticated
  USING (auth.uid() = user_id);
```

```
-- Mood entries policies
CREATE POLICY "Users can read own mood entries"
  ON mood_entries
  FOR SELECT
  TO authenticated
  USING (auth.uid() = user_id);

CREATE POLICY "Users can insert own mood entries"
  ON mood_entries
  FOR INSERT
  TO authenticated
  WITH CHECK (auth.uid() = user_id);

CREATE POLICY "Users can update own mood entries"
  ON mood_entries
  FOR UPDATE
  TO authenticated
  USING (auth.uid() = user_id);

CREATE POLICY "Users can delete own mood entries"
  ON mood_entries
  FOR DELETE
  TO authenticated
  USING (auth.uid() = user_id);

-- Pomodoro sessions policies
CREATE POLICY "Users can read own pomodoro sessions"
  ON pomodoro_sessions
  FOR SELECT
  TO authenticated
  USING (auth.uid() = user_id);
```

```
CREATE POLICY "Users can insert own pomodoro sessions"
  ON pomodoro_sessions
  FOR INSERT
  TO authenticated
  WITH CHECK (auth.uid() = user_id);

CREATE POLICY "Users can update own pomodoro sessions"
  ON pomodoro_sessions
  FOR UPDATE
  TO authenticated
  USING (auth.uid() = user_id);

CREATE POLICY "Users can delete own pomodoro sessions"
  ON pomodoro_sessions
  FOR DELETE
  TO authenticated
  USING (auth.uid() = user_id);
```