```
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS password_history;
DROP TABLE IF EXISTS posts;
DROP TABLE IF EXISTS tags;
DROP TABLE IF EXISTS follows;
DROP TABLE IF EXISTS likes;

CREATE TABLE test (col1 INTEGER);
CREATE TABLE users(
  user_id INTEGER PRIMARY KEY AUTOINCREMENT,
  username TEXT UNIQUE,
  first_name TEXT,
  last_name TEXT,
  email_address TEXT UNIQUE,
  password TEXT,
  moderator INTEGER,
  salt TEXT);
CREATE TABLE posts(
  post_id INTEGER PRIMARY KEY,
  user_id INTEGER,
  title TEXT,
  body TEXT,
  FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE);
CREATE TABLE tags(
  post_id INTEGER,
  tag TEXT,
  FOREIGN KEY (post_id) REFERENCES posts(post_id) ON DELETE CASCADE);
CREATE TABLE follows(
  follower_id INTEGER,
  followed_id INTEGER,
  FOREIGN KEY (follower_id) REFERENCES users(user_id) ON DELETE CASCADE,
  FOREIGN KEY (followed_id) REFERENCES users(user_id) ON DELETE CASCADE,
  PRIMARY KEY (follower_id, followed_id));
CREATE TABLE likes(
  user_id INTEGER,
  post_id INTEGER,
  FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE,
  FOREIGN KEY (post_id) REFERENCES posts(post_id) ON DELETE CASCADE,
  PRIMARY KEY(user_id, post_id));
CREATE TABLE password_history(
  user_id INTEGER,
  password TEXT,
  FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE);
```

I included all of a user's info in one table 'users', the same as project 1. All the data points within users are related to each other, and splitting would create redundancy. User_id serves as the primary key that is autoincremented with each user. This ID is unique and referred to by other tables. Username is also unique for security and identification. First_name and last_name store TEXT information. Email_address is also UNQUE so one email is used per user. Password stores a hashed TEXT value, not the plaintext password, for security. Salt stores a value used in password hashing, as an extra layer of security. Moderator stores an INTEGER value representing a Boolean for if a user is a moderator or not.

Post stores the information in a post. Post_id is the primary key, user id is a foreign key pointing to users, to store which user authored the post. Title and Body are both TEXT storing the post's info. A user's posts are deleted when the user is deleted via a deletion cascade.

Tags stores a post's tags. Post_id is a foreign key pointing to posts, to store which post a tag belongs to. Tag stores the TEXT tag for a post. A post's tags are deleted via cascade when the post is deleted

Followers stores the information of which users follow who. Follower_id stores who is following the Followed_id, both are foreign keys referencing valid user_ids. A follow is deleted from the table via cascade if either of the users is deleted. The primary key combination enforces unique pairings, so a user cannot follow another more than once.

The likes table links user_id and post_id to record which users like which post, both foreign keys. Both of these ensure the like is also deleted whenever the user or the post is deleted. Again, the primary key combination enforces unique pairings, so a user cannot like the same post more than once.