In this practice set, let's build a relational database for a typical social networking site.

In a social networking site, we have ***user***, ***post***, ***group***, and ***comment*** entities.

Use Case:

- A ***user*** can create multiple posts. Each ***post*** can have only one user.
- A ***user*** can make multiple comments to a post. Each ***comment*** can have only one user.
- A ***post*** can have multiple comments. Each ***comment*** can have only one post.
- A ***user*** can be in multiple groups. Each ***group*** can have multiple users.

Answer all the questions at once. The database resets everytime you navigate out of the practice set

QUESTIONS

1. Write a query to represent the ***user*** entity type in the relational database. Below are the attributes of a user entity type.

attribute	description
id	an integer to uniquely identify a user - key attribute
name	a string of max length 250 characters
gender	a string of max length 50 characters
email_id	a string of max length 500 characters

SHOW ANSWER

CREATE TABLE user (id INTEGER NOT NULL PRIMARY KEY, name VARCHAR(250), gender VARCHAR(50), email_id VARCHAR(500));

2. We have created a ***user*** table in the database.

   Now, let's write a query to represent the ***post*** entity type and its relation with ***user*** entity type.

   Below are the attributes of the post entity type.

attribute	description
post_id	an integer to uniquely identify a post - key attribute
content	a text field
published_at	datetime field

Note:

- Create a table in such a way that if we delete a user from the *user* table, then the related posts in the *post* table must be automatically deleted.

CREATE TABLE post (post_id INTEGER NOT NULL PRIMARY KEY, content TEXT, published_at DATETIME, user_id INTEGER, FOREIGN KEY(user_id) REFERENCES user(id) ON DELETE CASCADE);

3. We have created *user* and *post* tables in the database. Now, users want to comment on the posts. So, let's create a *comment* table.

   Write a query to represent the *comment* entity type, and its relation with *user* and *post* entity types.

   - Attributes of a comment entity type are given below.

attribute	description
comment_id	an integer to uniquely identify a comment - key attribute
content	a text field
commented_at	datetime field

Note:

Create a table in such a way that:

- If we delete a user from the *user* table, then the related comments in the *comment* table must be automatically deleted.
- Similarly, if we delete a *post*, then the comments related to the post must be automatically deleted.

CREATE TABLE COMMENT (comment_id INTEGER NOT NULL PRIMARY KEY, content TEXT, commented_at DATETIME, user_id INTEGER, post_id INTEGER, FOREIGN KEY(user_id) REFERENCES user(id) ON DELETE CASCADE, FOREIGN KEY(post_id) REFERENCES post(post_id) ON DELETE CASCADE);

4. Any social network application has groups with users of similar interests.
   Now, let's create a *group_details* table that stores the information about a group.

   Write a query to represent the *group_details* entity type in the relational database. Below are the attributes of the entity type.

attribute	description
id	an integer to uniquely identify a group - key attribute
name	a string of max length 500 characters

CREATE TABLE group_details (id INTEGER NOT NULL PRIMARY KEY, name VARCHAR(500));

5. A user can be in multiple groups, and a group can contain many users.

   Now, let's create *user_group* table to capture the many-to-many relationship between *user* and *group* entity types.

   Below are the attributes of the relationship.

attribute	description
joined_at	a datetime field
is_admin	a boolean field

Note:

Create this junction table in such a way that if we delete a user/group, then the related data in the *user_group* must be automatically deleted.

CREATE TABLE user_group (joined_at DATETIME, is_admin BOOLEAN, user_id INTEGER, group_id INTEGER, FOREIGN KEY(user_id) REFERENCES user(id) ON DELETE CASCADE, FOREIGN KEY(group_id) REFERENCES group_details(id) ON DELETE CASCADE);