<u>Operations:</u>

Register a new user:
First check if the user already exists by their email address (*user_id*): EXISTS (SELECT * FROM Users WHERE user_id="input")
If it doesn't exist, then run:
INSERT INTO Users VALUES (*user_id*, *password*)

Log in:
SELECT password FROM Users WHERE user_id = email_input

Create an Account:
The user_id is the email address.
check if the user already has an account: EXISTS (SELECT * FROM Users WHERE user_id="input")
otherwise: INSERT INTO Users VALUES (*user_id*, *password*)

Get a user's groups (dashboard):
SELECT group_id FROM UserGroups WHERE user_id = email_input

Get all of a user's bills (across all groups):
SELECT bill_id, bill_name, amount, due_date, description FROM Bills WHERE user_id = email_input

Get all of a user's bills in a given group:
SELECT bill_id, bill_name, amount, due_date, description FROM (Bills AS B INNER JOIN (SELECT bill_id FROM GroupBills WHERE group_id="INPUT") AS G ON B.bill_id=G.bill_id) WHERE user_id="INPUT";

Add a Bill:
The program needs to query the database to construct a unique *bill_id* string. Create some *bill_id* string (limit of 20 characters) and check if it already has been used by using an if statement for EXISTS (SELECT * FROM Bills WHERE bill_id="input"). Change the *bill_id* string until that evaluates to False.

The other values included below are inputted by the user. If multiple users are given, repeat the insert queries for each user_id.

When a unique bill_id has been established:
INSERT INTO Bills VALUES (*bill_id*, *bill_name*, *user_id*, *amount*, *due_date*, *description*);
INSERT INTO GroupBills VALUES (*group_id*, *bill_id*)

Delete a Bill:
The program should be given a *user_id*, *group_id*, and *bill_id*.

I’m imagining the user can select bills from a drop down menu, where they may not see the *bill_id*, but that value is still stored by the program so it can be used when the user picks a bill.

DELETE FROM GroupBills WHERE group_id="input" AND bill_id="input";
DELETE FROM Bills WHERE bill_id="input" AND user_id="input";

Change Bill Due Date:
The program should be given the *bill_id* and new *due_date*.
UPDATE Bills SET due_date="input" WHERE bill_id="input";

Change Bill Amount:
The program should be given the *bill_id* and the new *amount*.
UPDATE Bills SET amount=input WHERE bill_id="input";

Create a Group:
The program should be given a *user_id* (email), and the user should input the desired *group_name*.

As with adding a bill, the the program needs to create a unique *group_id* for the group. Do this by creating one, and editing it until if (EXISTS (SELECT * FROM Groups WHERE group_id="input")) evaluates to False.

When a unique group_id has been established:
INSERT INTO Groups VALUES (*group_id*, *group_name*);
INSERT INTO UserGroups VALUES (*user_id*, *group_id*);

Add a User to a Group:
The program needs the new user’s email (their *user_id*) and the *group_id* for the group to be added to (from a dropdown menu of the current user’s groups)

INSERT INTO UserGroups VALUES (*user_id*, *group_id*)
Also, check if the invited user already exists in Users by: if (EXISTS (SELECT * FROM Users WHERE user_id="input")) — if False, send an email inviting them to the app. Users will be updated to include the user when they create an account with the app.

Leave a Group:
The program needs the *group_id* and *user_id*.

Perform the aforementioned query to get all the bills for a given user in a given group. For each of those bills’ *bill_id*:

    DELETE FROM GroupBills WHERE group_id="input" AND bill_id="input";
    DELETE FROM Bills WHERE bill_id="input" AND user_id="input";

DELETE FROM UserGroups WHERE user_id="input" AND group_id="input";

Check if a group is now empty: IF (EXISTS (SELECT * FROM UserGroups WHERE group_id="input"))
if False: DELETE FROM Groups WHERE group_id = "input";