Brainstorming

Tables:

//this table will store user information
Users: email, password, id, is_viewable
//this table will store recipe information
Recipes: id, instructions, ingredients, users_id, is_public, name
//this table will store ingredients from recipes
Grocery list: ingredients_id

//this table will store information about users assigning certain recipes to occasions and creating their own occasions
Occasions: id, name, recipes_id, users_id

Relationships:
User: One to many, one user can access all recipes, grocery list, and occasions
Recipes: Many to many, many users have recipes and many recipes can have multiple users
Grocery List: Association table, the 'glue' between two tables
Occasions: Many to many, many users can have many occasions, many occasions to many users

Columns:

Users

- id , to reference in recipes, grocery list, occasions
- email , for people to log into , varchar to write down unique emails
- password , to keep secure from other users, varchar to write passwords
- is_viewable , if users can view other people's recipes , boolean true or false

Recipes

- id , to reference in other tables, and we chose int because it will be a number that increments
- name , to name the recipe. We chose varchar because it will be text
- instructions , to describe how to make the recipe. We chose varchar because it will be text
- Ingredients, to list out what the recipe needs, we chose varchar because it will be text
- is_public , to let any user use the recipe. We chose boolean to set equal to true or false
- user_id , to reference what user is using what recipe, we chose int to list the id

Grocery List

- id , to reference in other tables if needed
- ingredient_list , to get the ingredients from recipes.

- user_id , to reference what user is using a grocery list

Occasions

- id , to set id's for occasions , int serial primary key to increment the number & keep unique id
- name , to name unique occasions , varchar to write
- recipes_id , to reference all recipes , int because it's an id
- user_id , to reference all the users , int because it's an id

Steps in Posgres:

```
//creating tables of users, recipes, grocery_list
CREATE TABLE users (
--         id SERIAL PRIMARY KEY,
--   email VARCHAR(35) UNIQUE NOT NULL,
--   password VARCHAR(35) UNIQUE NOT NULL,
--   is_viewable BOOLEAN NOT NULL
-- );

-- CREATE TABLE recipes (
--         id SERIAL PRIMARY KEY,
--   name VARCHAR(35) UNIQUE NOT NULL,
--   instructions VARCHAR(255) NOT NULL,
--   is_public BOOLEAN NOT NULL,
--   ingredients VARCHAR(255) NOT NULL,
--   user_id INT NOT NULL REFERENCES users(id)
-- );

-- ALTER TABLE recipes DROP COLUMN ingredients;

-- CREATE TABLE grocery_list (
--         id SERIAL PRIMARY KEY,
--   user_id INT NOT NULL REFERENCES users(id)
-- );

//adding items to grocery_list and putting ingredients back into recipes
-- ALTER TABLE grocery_list ADD items_name VARCHAR(255);

-- ALTER TABLE recipes ADD ingredients VARCHAR(255) NOT NULL;

-- CREATE TABLE occasions (
--         id SERIAL PRIMARY KEY,
--   name VARCHAR(255) NOT NULL,
--   user_id INT NOT NULL REFERENCES users(id),
```

```
--   recipes_id INT NOT NULL REFERENCES recipes(id)
-- );

CREATE TABLE occasions_recipes (
          id SERIAL PRIMARY KEY,
  recipes_id INT NOT NULL REFERENCES recipes(id),
  occasions_id INT NOT NULL REFERENCES occasions(id)
);

//insert info into email
INSERT INTO users (email, password, is_viewable)
VALUES ('joeshmow@theemail.com', 'password', TRUE),
('myemail@email.com', 'ABC123', FALSE);
```