Express.js and PostgreSQL CRUD API

This is a mini-project to demonstrate a simple RESTful API built with Express.js that performs CRUD (Create, Read, Update, Delete) operations on a PostgreSQL database.

Objectives
- Create a simple Express.js API.
- Connect to a PostgreSQL database.
- Implement basic CRUD database operations.

Prerequisites
- [Node.js](https://nodejs.org/) (which includes npm)
- [PostgreSQL](https://www.postgresql.org/download/)

Setup and Running the Project

1. Clone the Repository
```bash
git clone <your-github-repository-url>
cd express-postgres-crud
```

2. Install Dependencies
```bash
npm install
```

3. Set up the PostgreSQL Database
  - Make sure your PostgreSQL server is running.
  - Open a terminal and connect to PostgreSQL using `psql`.
  ```bash
  psql -U your_postgres_username
  ```
  - Create a new database.
  ```sql
  CREATE DATABASE your_database_name;
  ```
  - Connect to your newly created database.
  ```sql
  \c your_database_name
  ```
  - Run the following SQL command to create the `users` table.
  ```sql
  CREATE TABLE users (
  ```

```
    id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100),
    age INTEGER
  );
  ```

4. Configure the Application
  - Open the `db.js` file in your code editor.
  - Update the `Pool` configuration with your own PostgreSQL username, database name, and password.
  ```javascript
  const pool = new Pool({
    user: 'your_postgres_username',      // Replace with your username
    host: 'localhost',
    database: 'your_database_name',     // Replace with your database name
    password: 'your_postgres_password', // Replace with your password
    port: 5432,
  });
  ```

5. Run the Server
```bash
npm start
```
The server should now be running on `http://localhost:3000`.

API Endpoints

You can test the following endpoints using a tool like [Postman](https://www.postman.com/) or `curl`.

`GET /users`
- **Description:** Retrieves a list of all users.
- **Method:** `GET`
- **URL:** `http://localhost:3000/users`
- **Success Response (200 OK):**
  ```json
  [
    {
      "id": 1,
      "name": "John Doe",
      "email": "john.doe@example.com",
      "age": 30
```

```
  },
  {
    "id": 2,
    "name": "Jane Smith",
    "email": "jane.smith@example.com",
    "age": 25
  }
]
```

`GET /users/:id`
- **Description:** Retrieves a single user by their ID.
- **Method:** `GET`
- **URL:** `http://localhost:3000/users/1`
- **Success Response (200 OK):**
```json
{
  "id": 1,
  "name": "John Doe",
  "email": "john.doe@example.com",
  "age": 30
}
```

`POST /users`
- **Description:** Creates a new user.
- **Method:** `POST`
- **URL:** `http://localhost:3000/users`
- **Request Body (JSON):**
```json
{
  "name": "Peter Jones",
  "email": "peter.jones@example.com",
  "age": 42
}
```
- **Success Response (201 Created):**
```json
{
  "id": 3,
  "name": "Peter Jones",
  "email": "peter.jones@example.com",
  "age": 42
}
```

  ```

#### `PUT /users/:id`
- **Description:** Updates an existing user's information.
- **Method:** `PUT`
- **URL:** `http://localhost:3000/users/3`
- **Request Body (JSON):**
  ```json
  {
    "name": "Peter Jones Jr.",
    "email": "peter.jones.jr@example.com",
    "age": 43
  }
  ```
- **Success Response (200 OK):**
  ```json
  {
    "id": 3,
    "name": "Peter Jones Jr.",
    "email": "peter.jones.jr@example.com",
    "age": 43
  }
  ```

`DELETE /users/:id`
- **Description:** Deletes a user by their ID.
- **Method:** `DELETE`
- **URL:** `http://localhost:3000/users/3`
- **Success Response (200 OK):**
  ```json
  {
    "message": "User deleted successfully"
  }
  ```
  ```