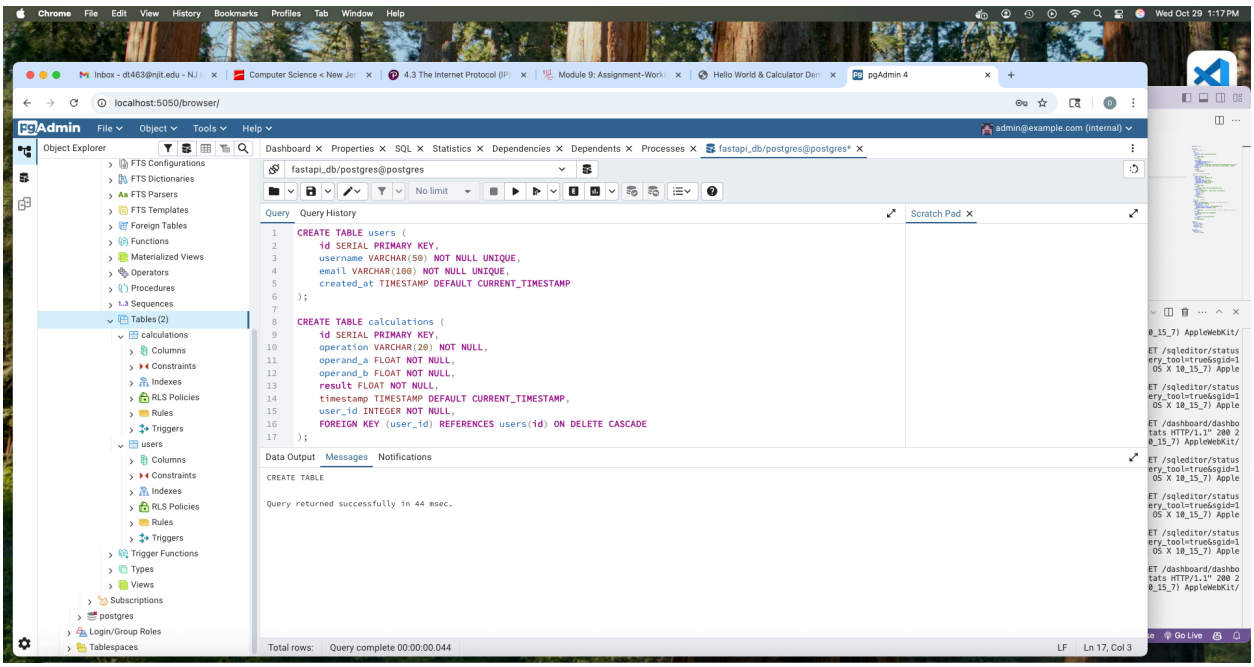
SQL Execution Documentation

(A) Create Tables

Query:

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE calculations (
    id SERIAL PRIMARY KEY,
    operation VARCHAR(20) NOT NULL,
    operand_a FLOAT NOT NULL,
    operand_b FLOAT NOT NULL,
    result FLOAT NOT NULL,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    user_id INTEGER NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE
CASCADE
);
```



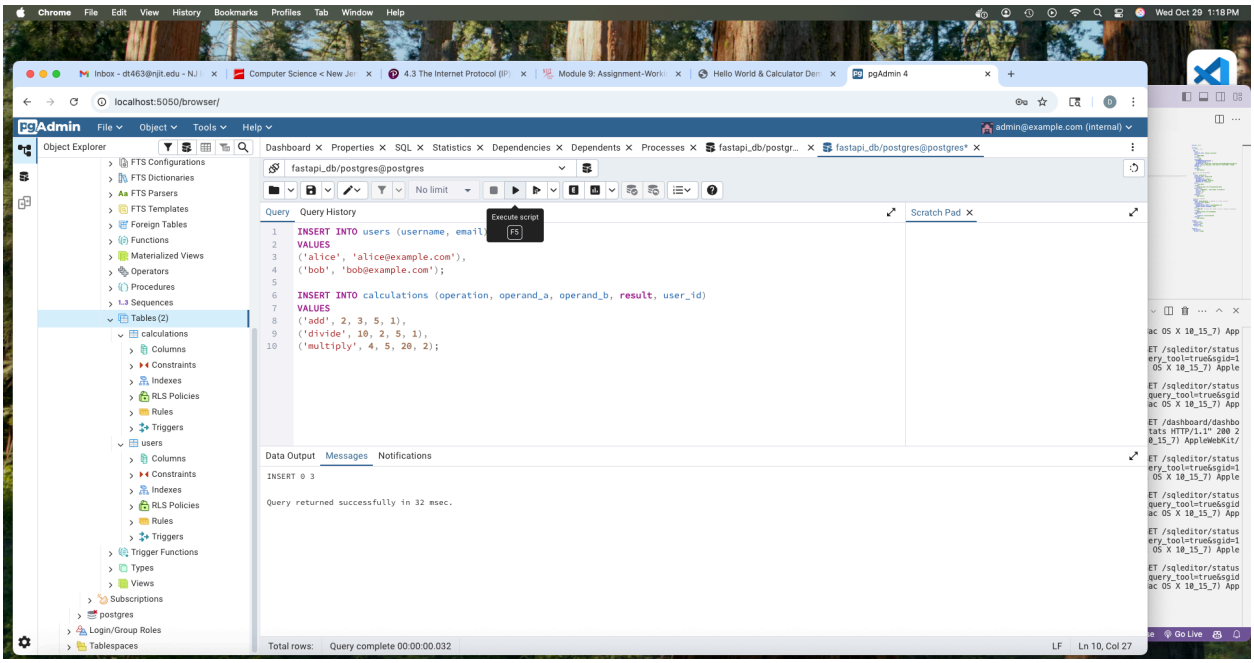
Screenshot Caption:
Tables `users` and `calculations` created successfully in pgAdmin. The output confirms structure and foreign key relationship.

(B) Insert Records

Query:

```
INSERT INTO users (username, email)
VALUES
('alice', 'alice@example.com'),
('bob', 'bob@example.com');

INSERT INTO calculations (operation, operand_a, operand_b,
result, user_id)
VALUES
('add', 2, 3, 5, 1),
('divide', 10, 2, 5, 1),
('multiply', 4, 5, 20, 2);
```



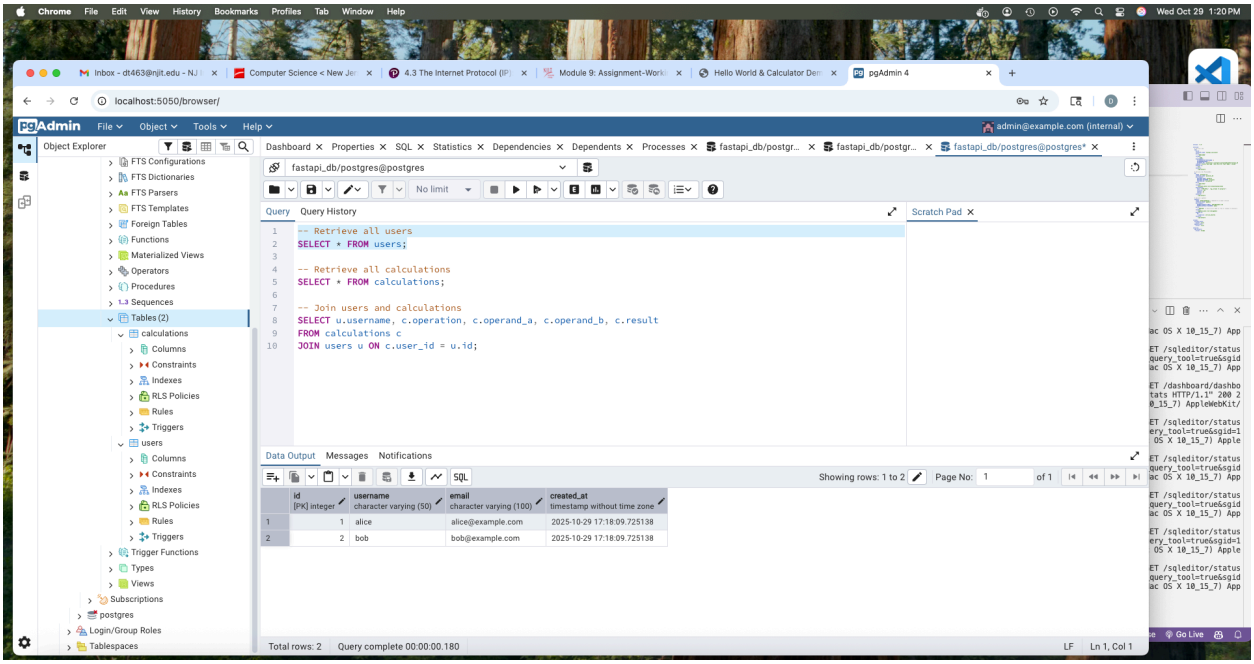
Screenshot Caption:
Records inserted into `users` and `calculations` tables. The output shows "Query returned successfully: X rows affected."

(C) Query Data

1. Retrieve All Users

Query:

```
SELECT * FROM users;
```

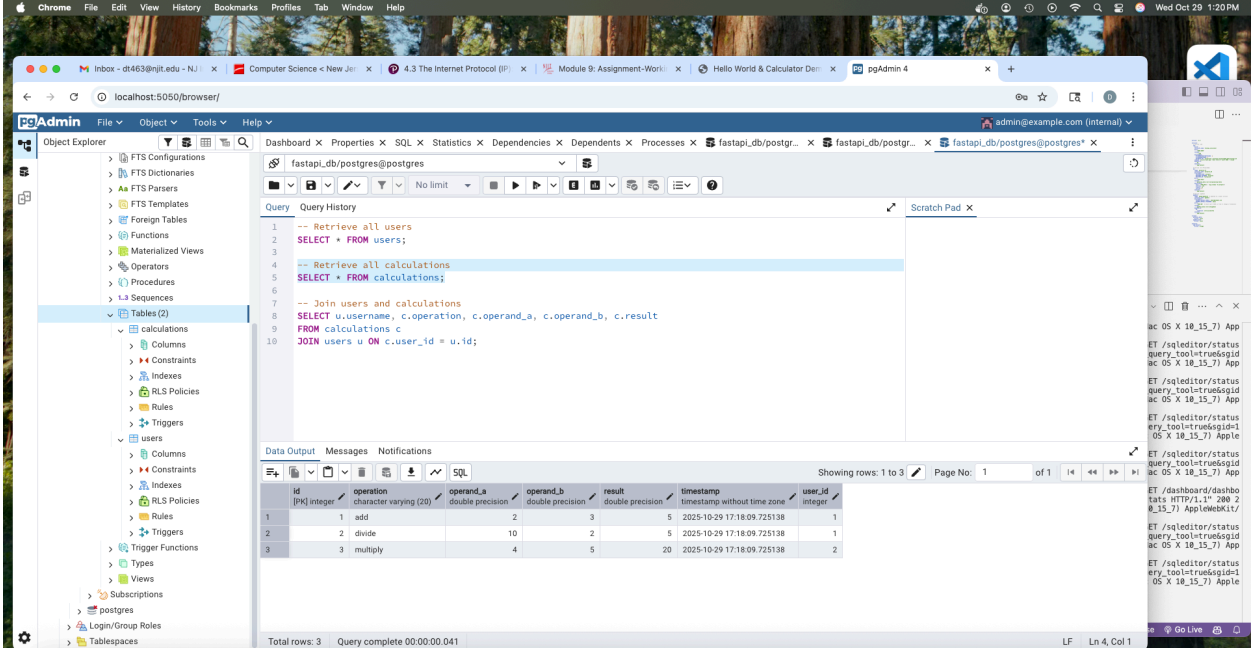


Screenshot Caption:
Displaying all users in the database. Shows `id`, `username`, `email`, and `created_at`.

2. Retrieve All Calculations

Query:

```
SELECT * FROM calculations;
```

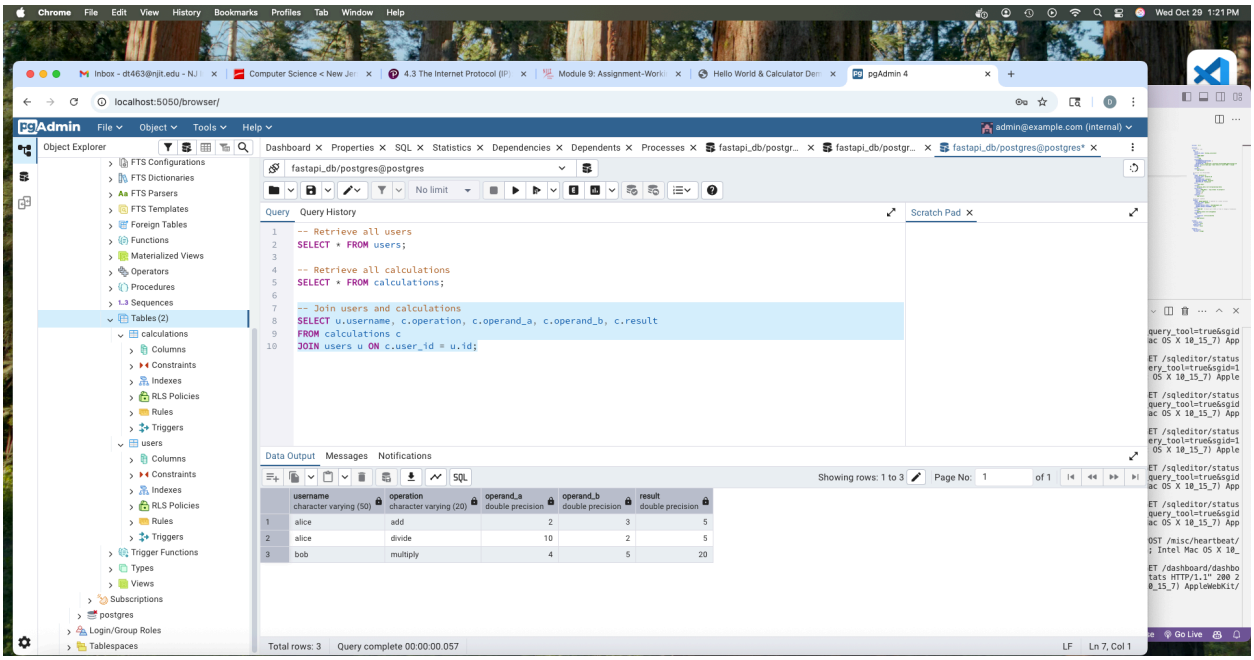


Screenshot Caption:
Displaying all calculations. Includes operation type, operands, result, timestamp, and associated user ID.

3. Join Users and Calculations

Query:

```
SELECT u.username, c.operation, c.operand_a, c.operand_b,
c.result
FROM calculations c
JOIN users u ON c.user_id = u.id;
```

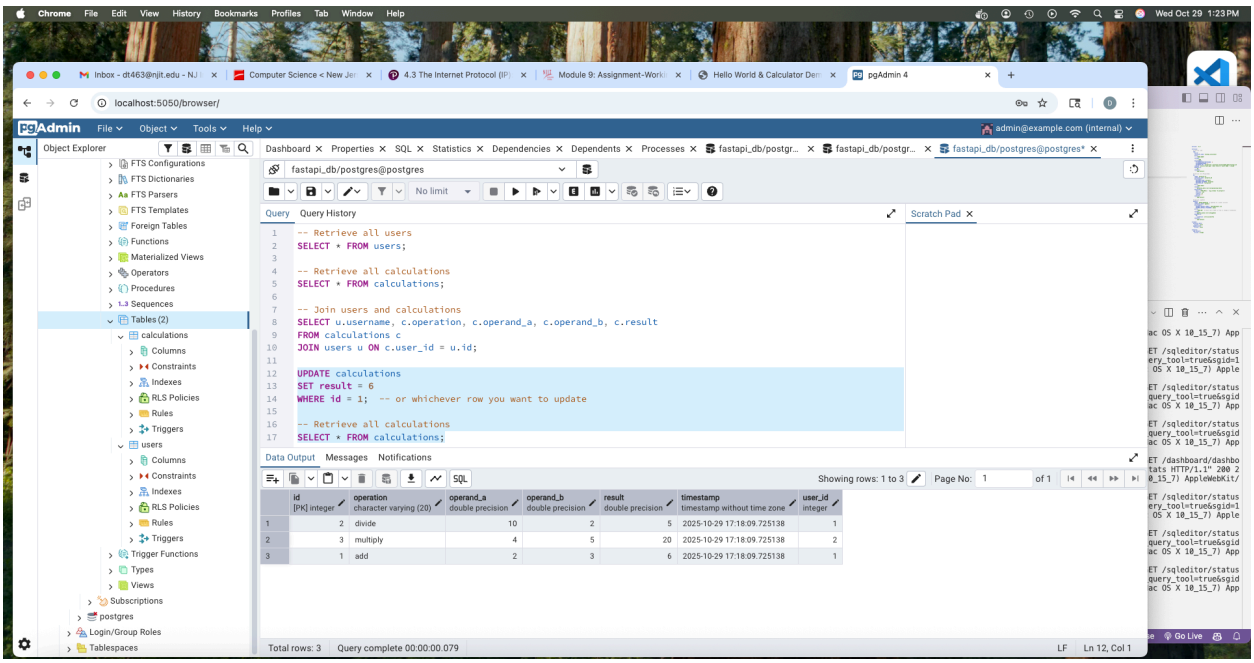


Screenshot Caption:
Joined `users` and `calculations` tables. Shows which user performed each operation and the result.

(D) Update a Record

Query:

```
UPDATE calculations
SET result = 6
WHERE id = 1;
```

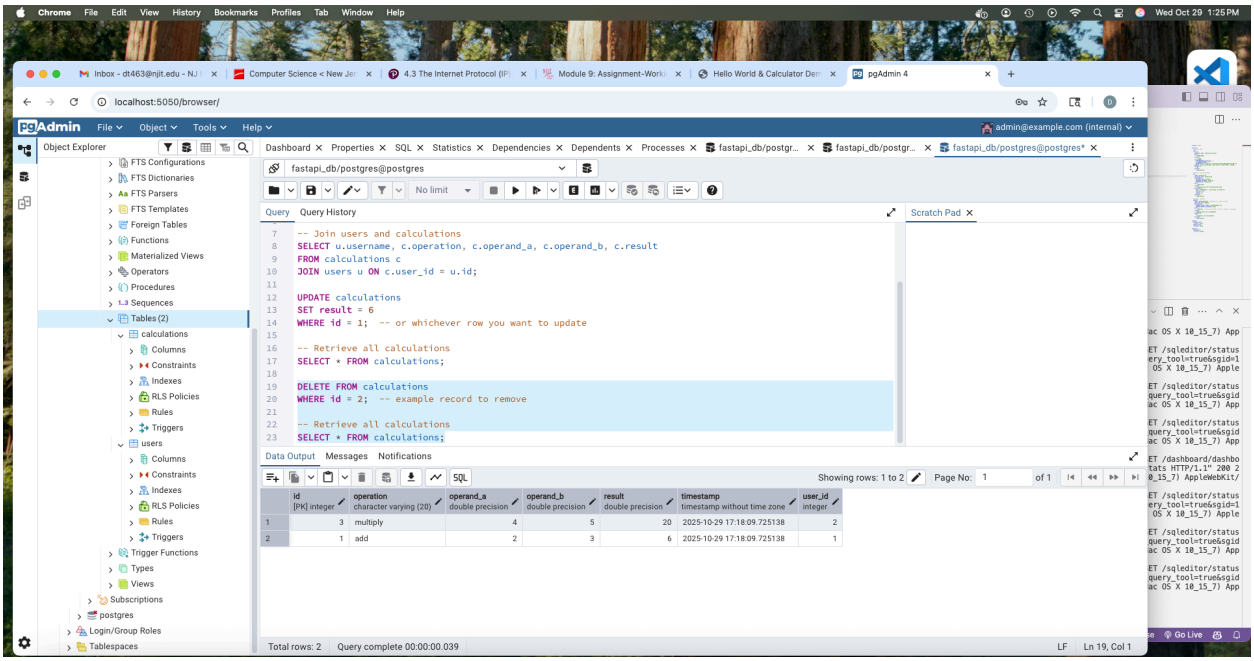


Screenshot Caption:
Updated calculation with `id = 1`. The result value changed from 5 to 6.

(E) Delete a Record

Query:

```
DELETE FROM calculations
WHERE id = 2;
```



Screenshot Caption:
Deleted calculation with `id = 2`. The output confirms successful deletion.

Reflection

In this assignment, I successfully set up a containerized FastAPI environment using Docker Compose, integrated PostgreSQL and pgAdmin, and executed raw SQL commands to manage relational data. I learned how foreign keys enforce relationships and how cascading deletes maintain data integrity. One challenge was resolving Docker volume conflicts, which I overcame by cleaning up old containers and pinning the PostgreSQL version. This experience strengthened my confidence in managing full-stack environments and debugging cross-service issues.