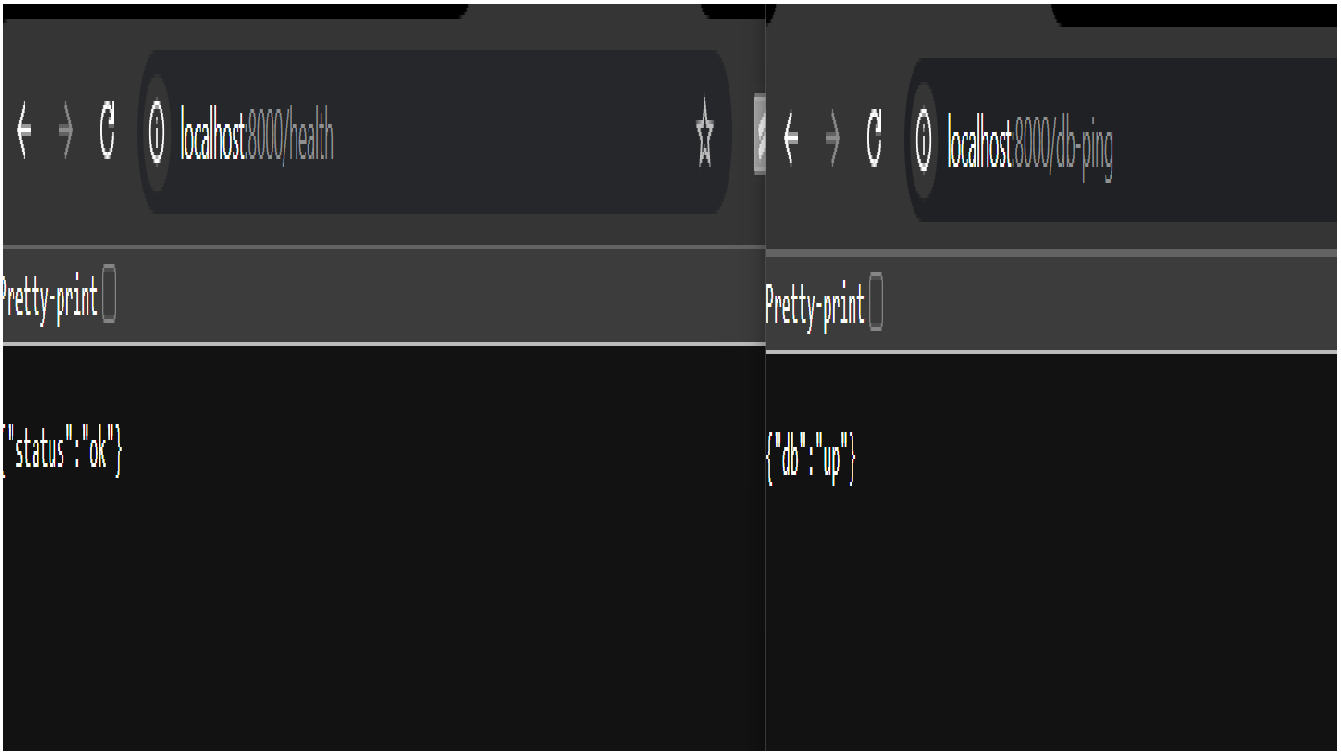
FastAPI + PostgreSQL Docker Compose Assignment

This document shows successful FastAPI and PostgreSQL integration using Docker Compose, with SQL operations executed in pgAdmin.

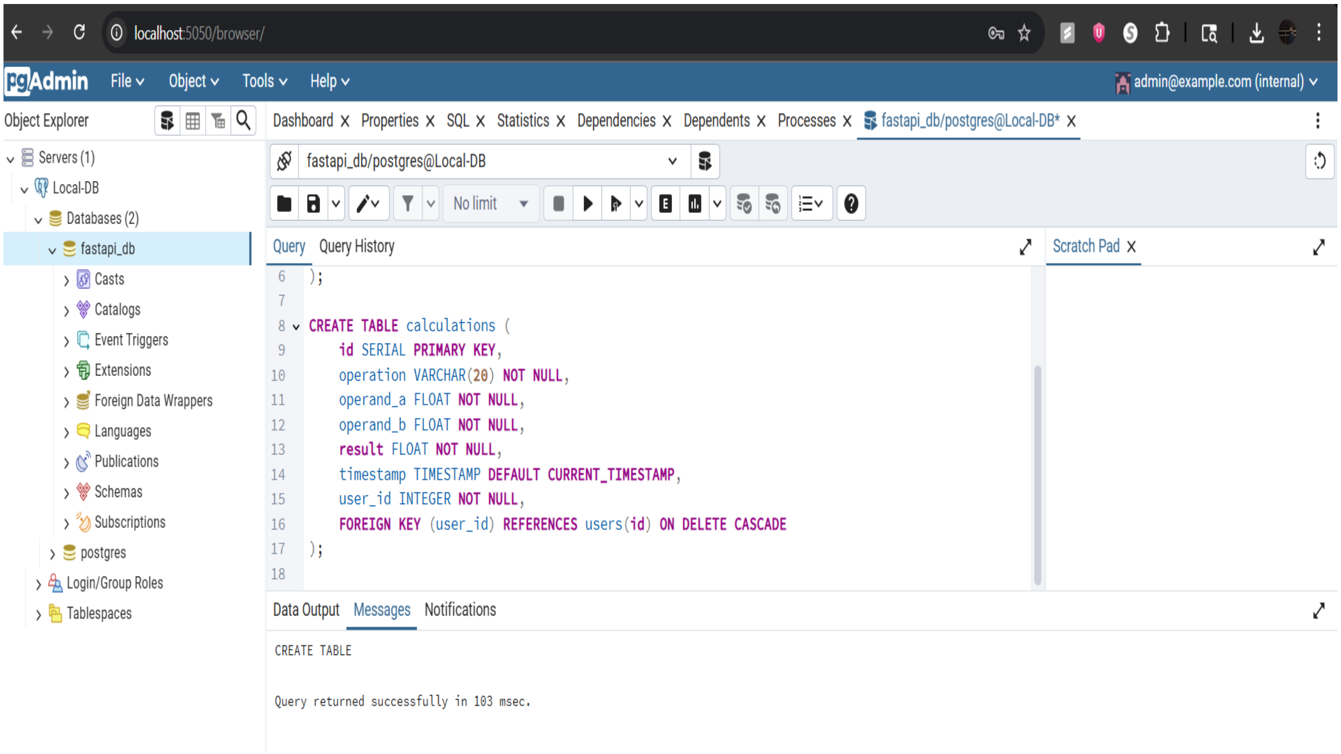
FastAPI Health & DB Ping



```
{"status":"ok"}
```

```
{"db":"up"}
```

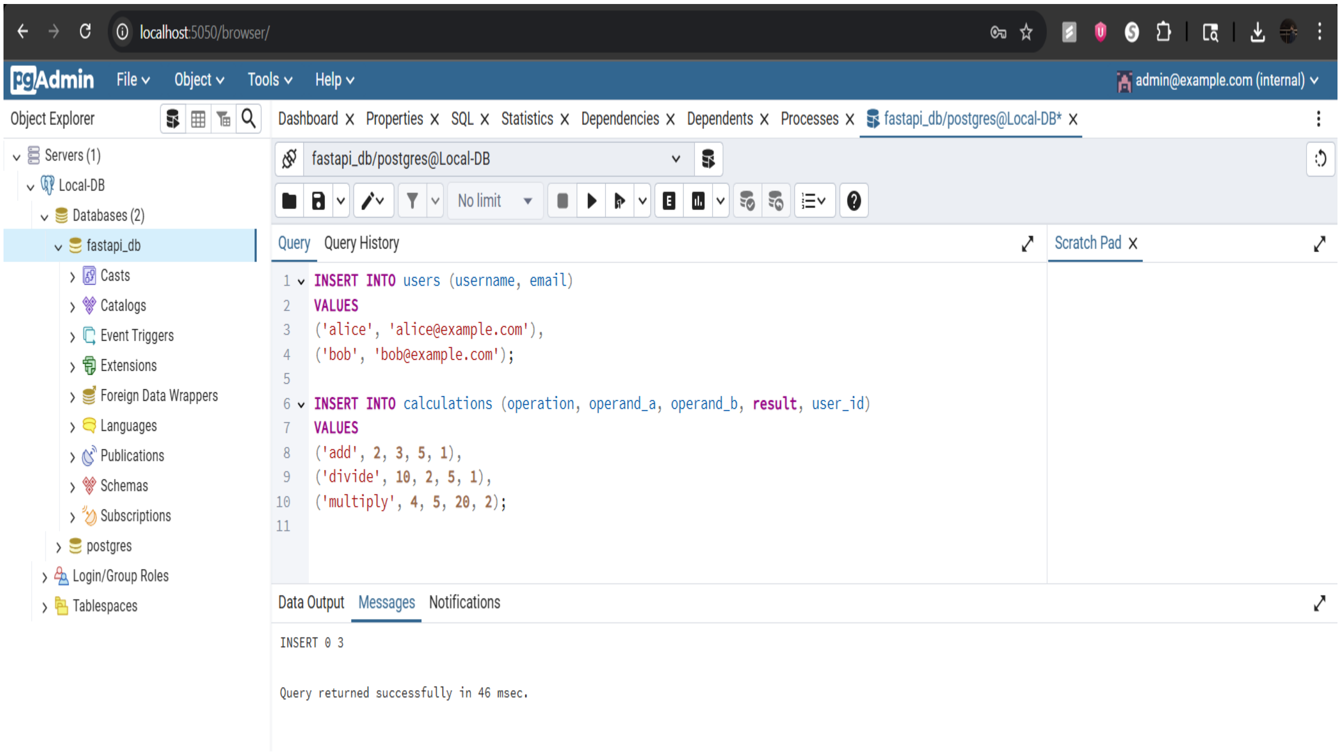
Create Table Query



```
);

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
);

```

```
CREATE TABLE

Query returned successfully in 103 msec.
```

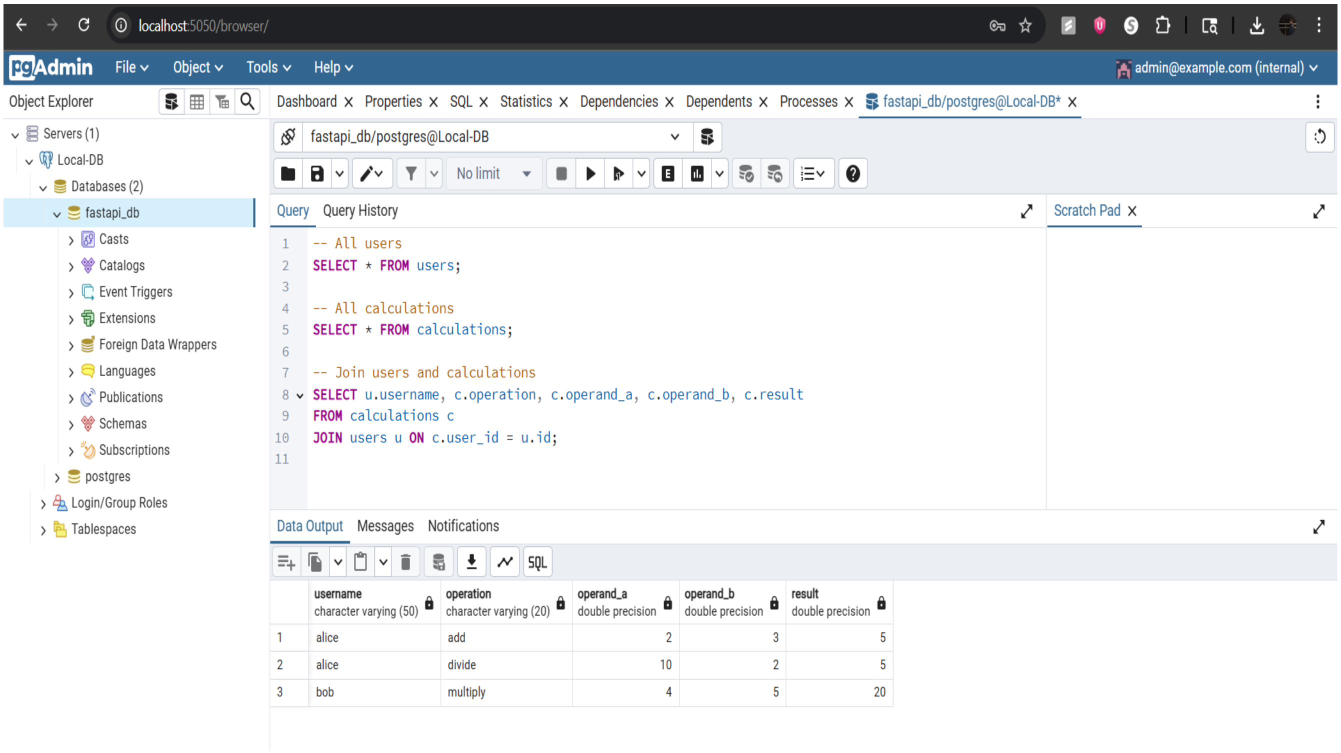
Insert Query Execution



```
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);

```

```
INSERT 0 3

Query returned successfully in 46 msec.
```

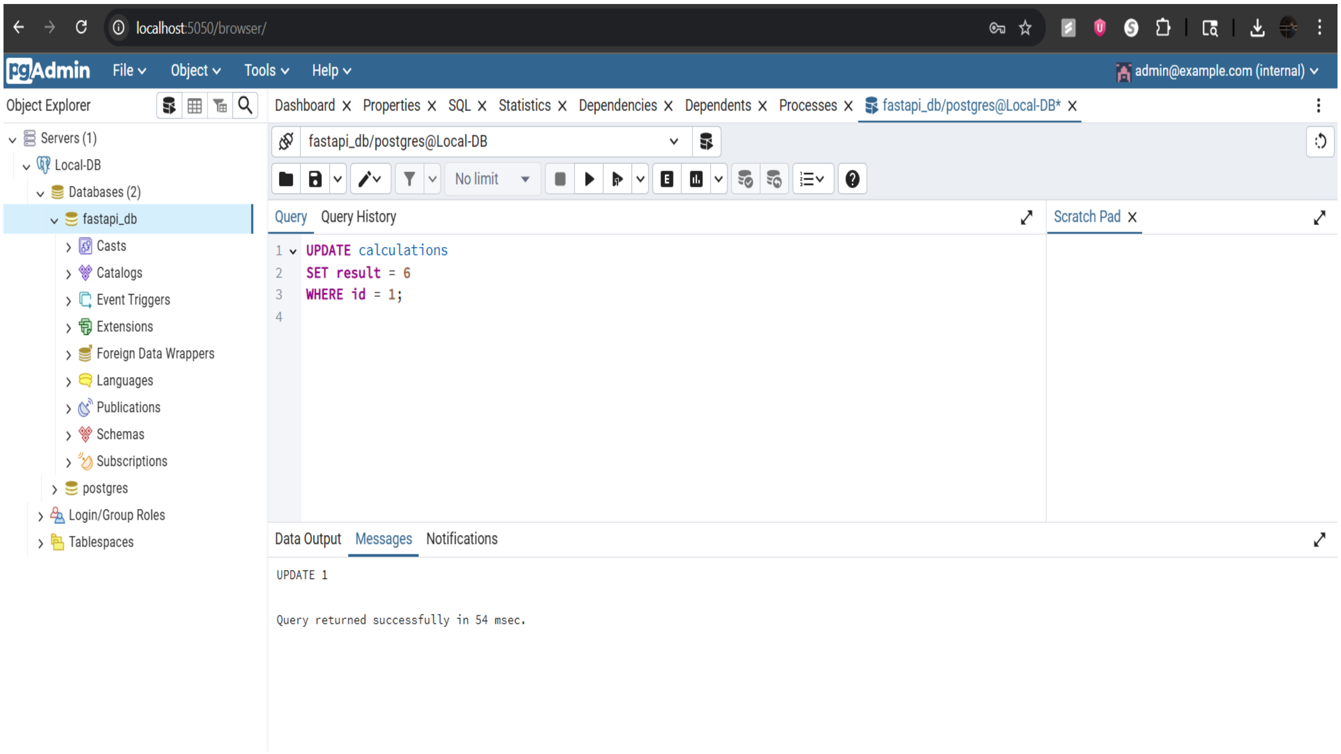
Select Query Results



```sql
-- All users
SELECT * FROM users;

-- All calculations
SELECT * FROM calculations;

-- Join users and calculations
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;
```

	username character varying (50)	operation character varying (20)	operand_a double precision	operand_b double precision	result double precision
1	alice	add	2	3	5
2	alice	divide	10	2	5
3	bob	multiply	4	5	20

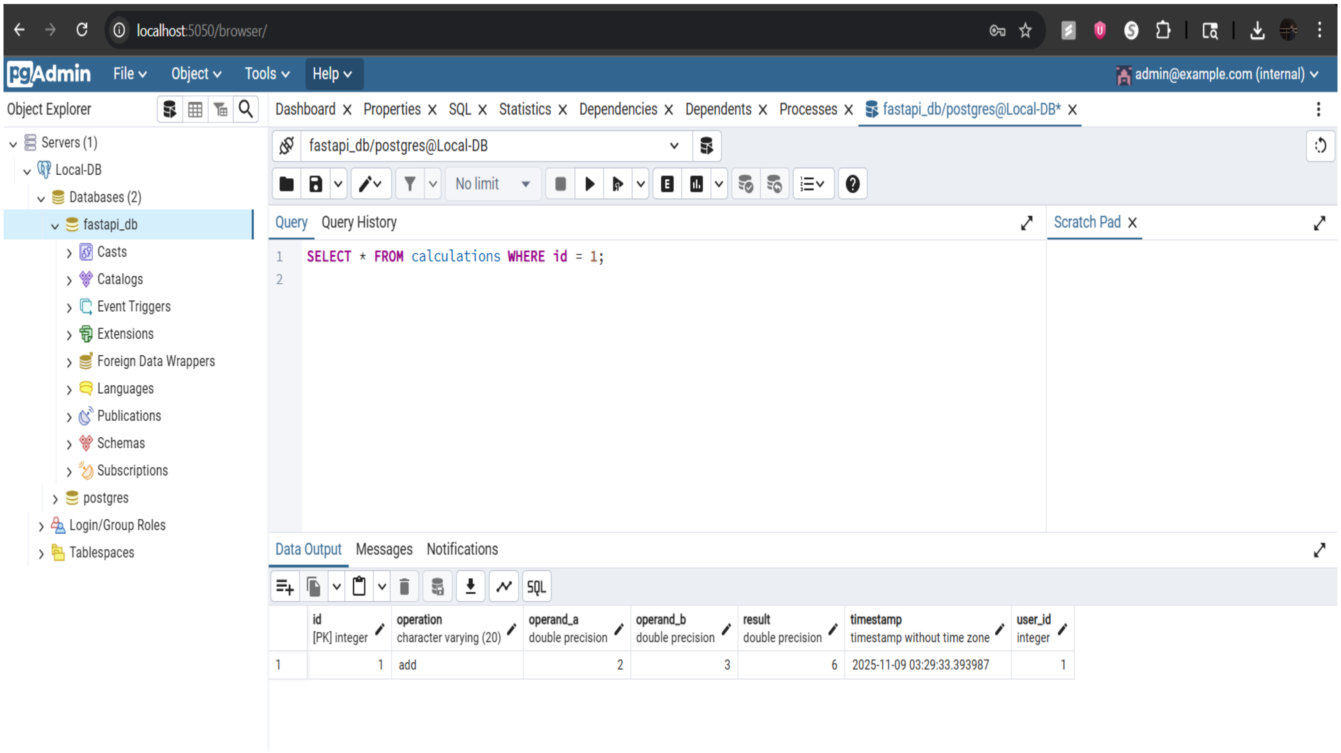
Update Query Execution



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

```
UPDATE 1

Query returned successfully in 54 msec.
```

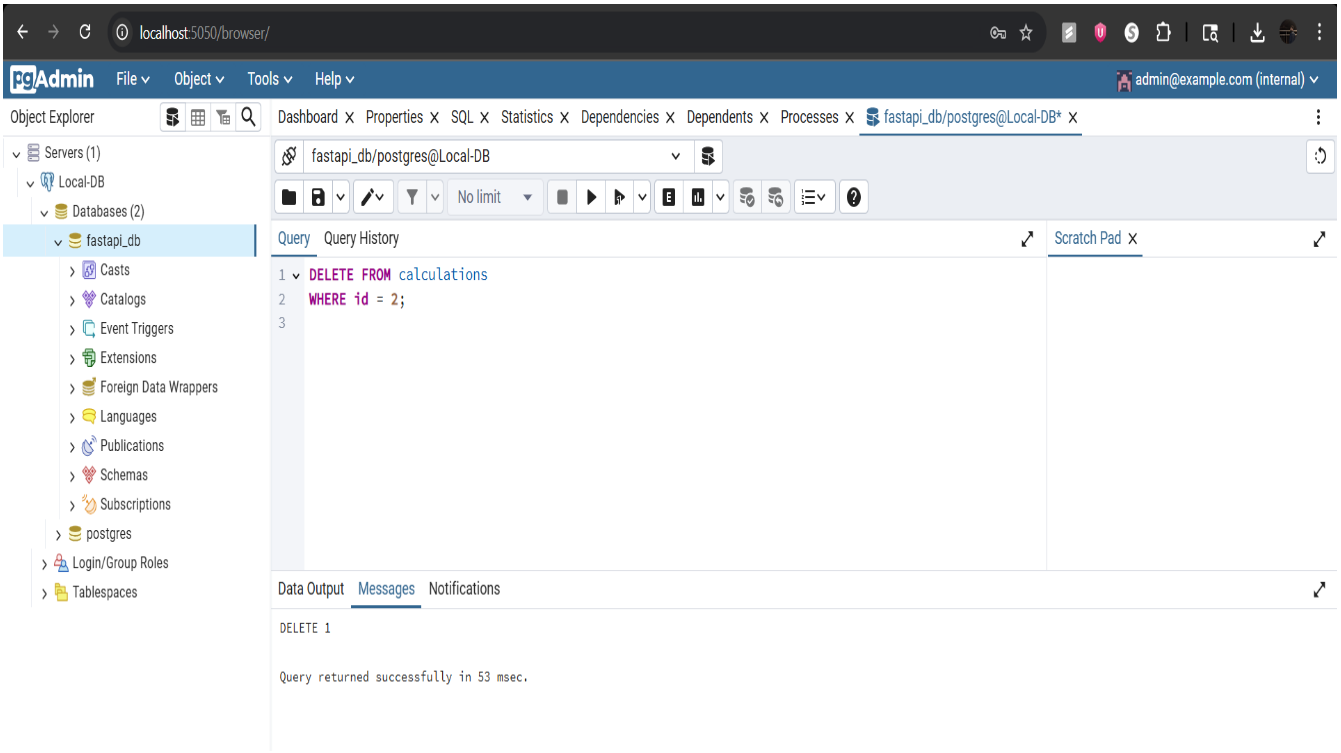
Update Query Results



```sql
SELECT * FROM calculations WHERE id = 1;
```

	id [PK] integer	operation character varying (20)	operand_a double precision	operand_b double precision	result double precision	timestamp timestamp without time zone	user_id integer
1	1	add	2	3	6	2025-11-09 03:29:33.393987	1

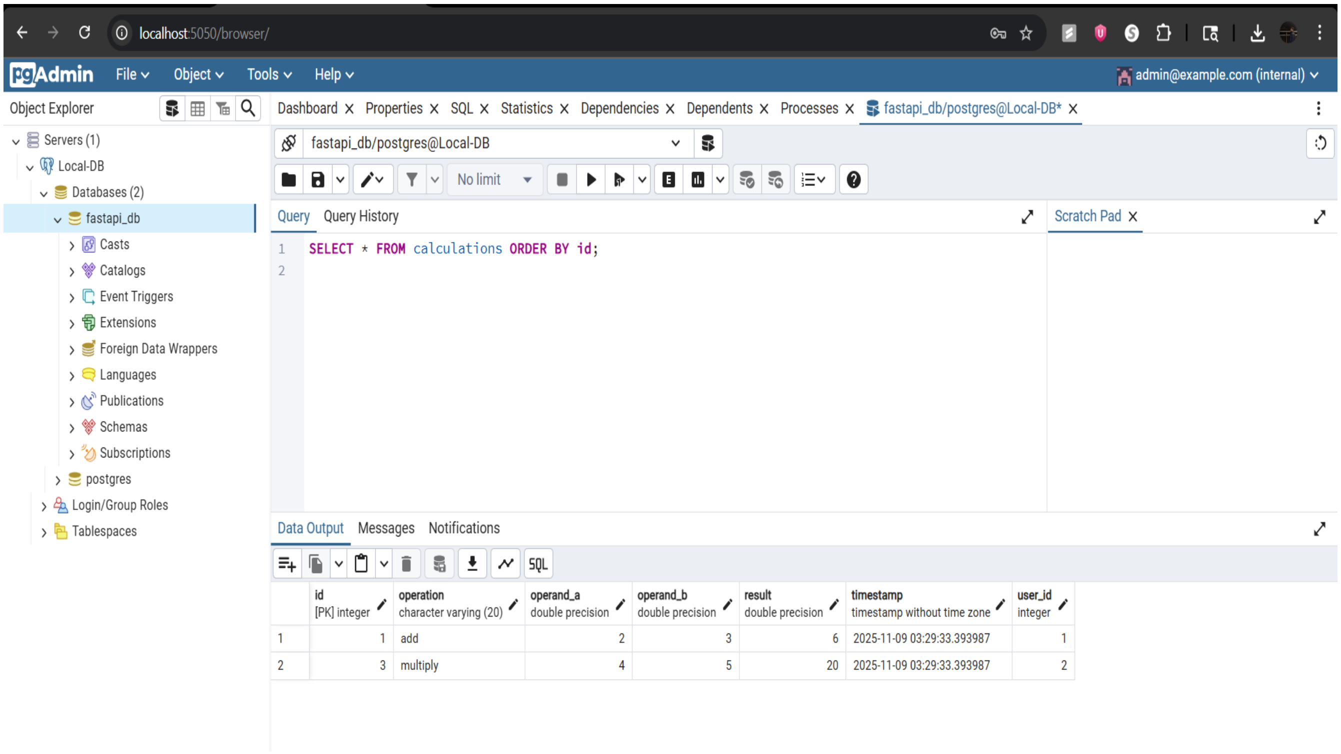
Delete Query Execution



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

```
DELETE 1

Query returned successfully in 53 msec.
```

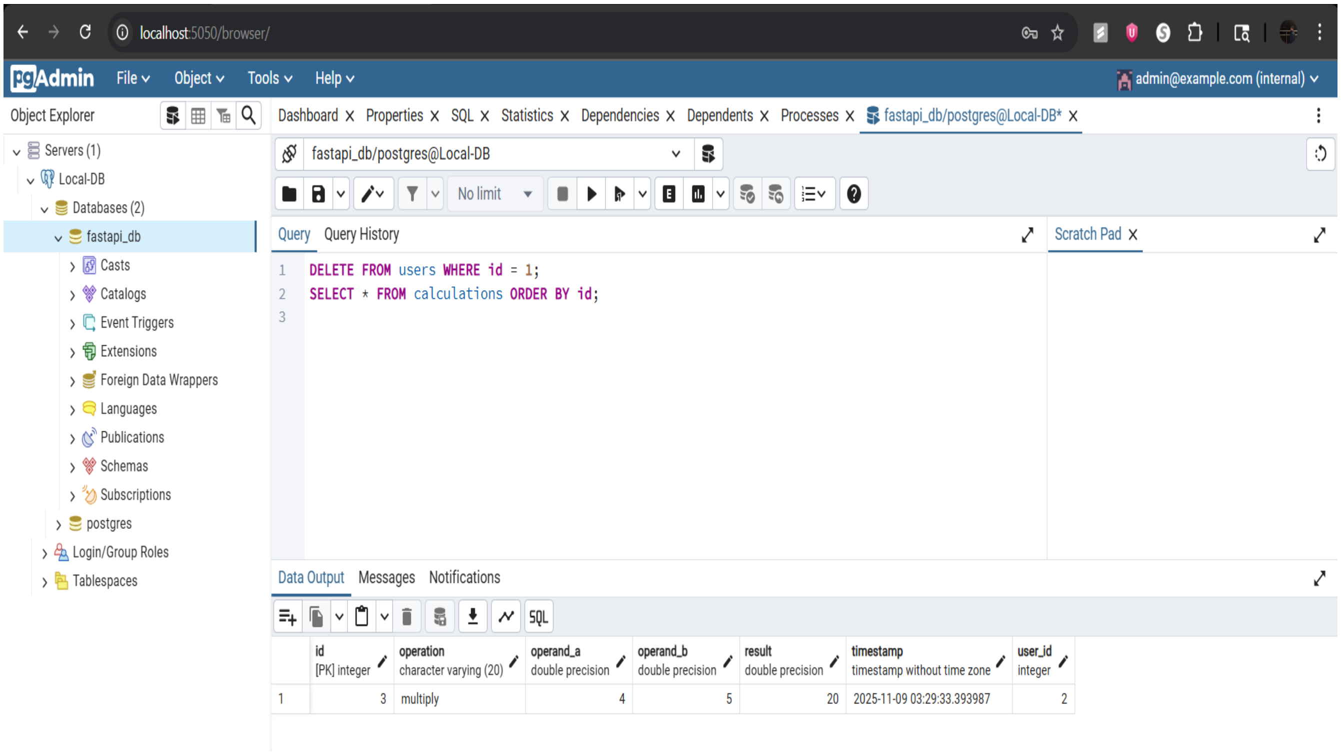
Delete Query Results



```sql
SELECT * FROM calculations ORDER BY id;
```

	id [PK] integer	operation character varying (20)	operand_a double precision	operand_b double precision	result double precision	timestamp timestamp without time zone	user_id integer
1	1	add	2	3	6	2025-11-09 03:29:33.393987	1
2	3	multiply	4	5	20	2025-11-09 03:29:33.393987	2

Delete 2 (Cascade Test)



```sql
DELETE FROM users WHERE id = 1;
SELECT * FROM calculations ORDER BY id;
```

	id [PK] integer	operation character varying (20)	operand_a double precision	operand_b double precision	result double precision	timestamp timestamp without time zone	user_id integer
1	3	multiply	4	5	20	2025-11-09 03:29:33.393987	2