Homework: Managing SQL Data

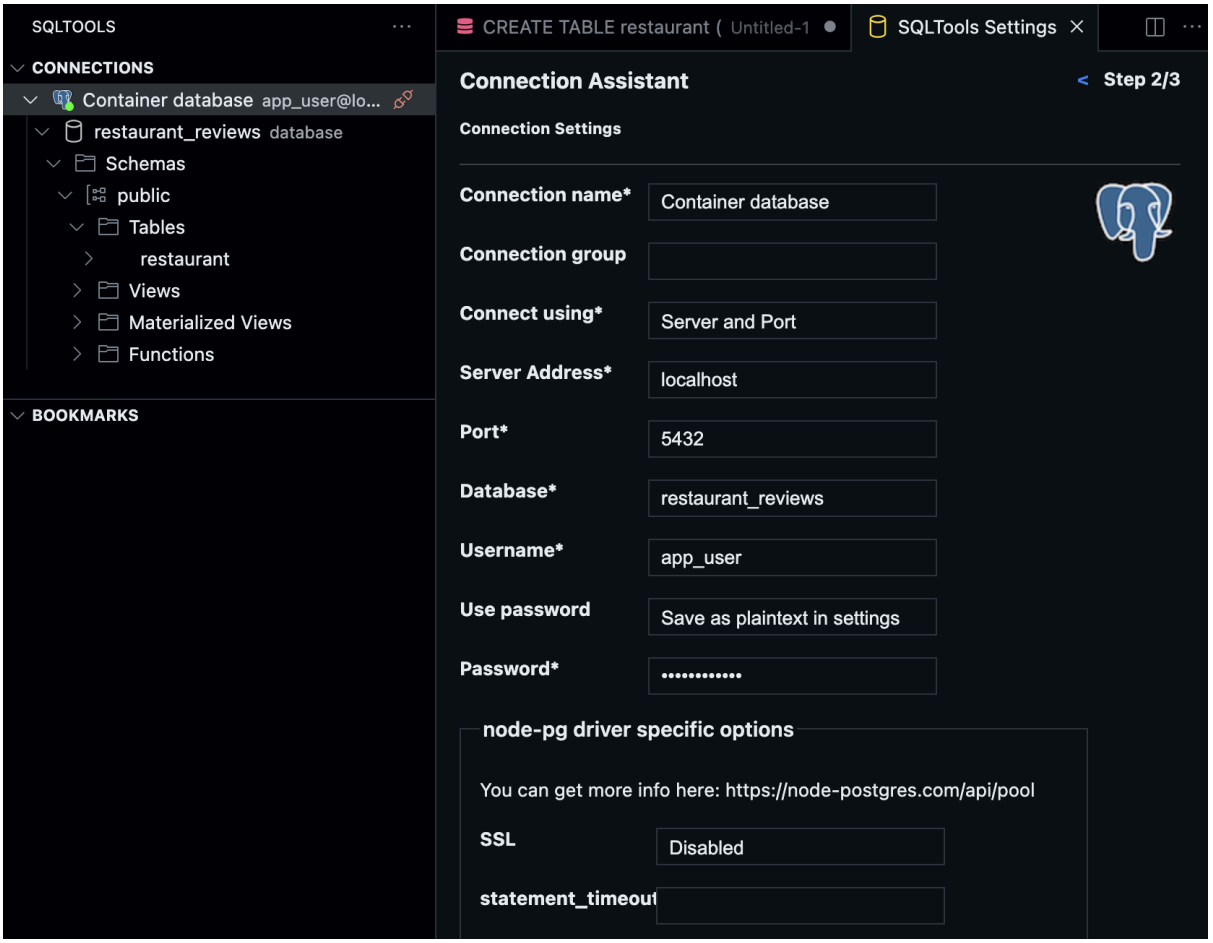
Database Setup

Create a new database:

Name the database **`restaurant_reviews`**

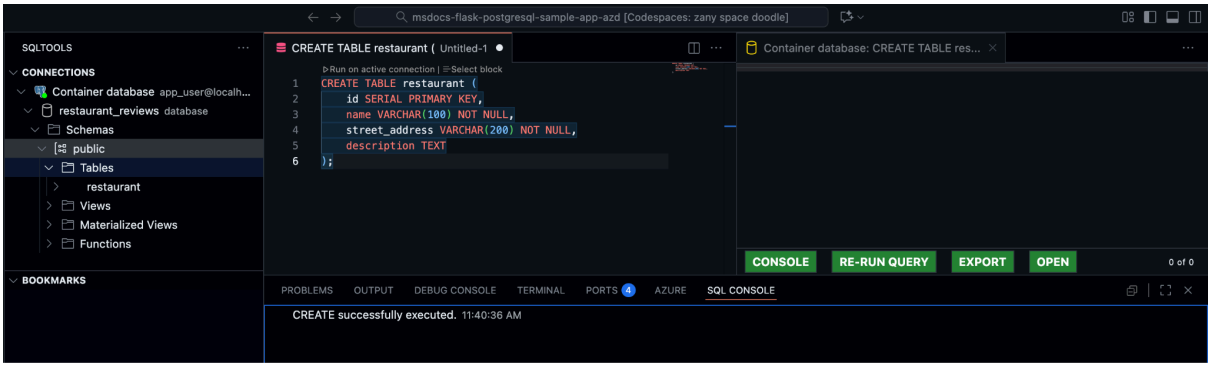
```
@marulityson → /workspaces/msdocs-flask-postgresql-sample-app-azd (main) $ psql -h localhost -U app_user -d app
psql (17.6 (Debian 17.6-0+deb13u1), server 18.0 (Debian 18.0-1.pgdg13+3))
WARNING: psql major version 17, server major version 18.
         Some psql features might not work.
Type "help" for help.

app=# CREATE DATABASE restaurant_reviews;
CREATE DATABASE
app=# \l
                                                    List of databases
        Name        |  Owner   | Encoding | Locale Provider |  Collate   |   Ctype    | Locale | ICU Rules |   Access privileges
--------------------+----------+----------+-----------------+------------+------------+--------+-----------+-----------------------
 app                | app_user | UTF8     | libc            | en_US.utf8 | en_US.utf8 |        |           |
 postgres           | app_user | UTF8     | libc            | en_US.utf8 | en_US.utf8 |        |           |
 restaurant_reviews | app_user | UTF8     | libc            | en_US.utf8 | en_US.utf8 |        |           |
 template0          | app_user | UTF8     | libc            | en_US.utf8 | en_US.utf8 |        |           | =c/app_user          +
                    |          |          |                 |            |            |        |           | app_user=CTc/app_user
 template1          | app_user | UTF8     | libc            | en_US.utf8 | en_US.utf8 |        |           | =c/app_user          +
                    |          |          |                 |            |            |        |           | app_user=CTc/app_user
(5 rows)
```

Connect to the database

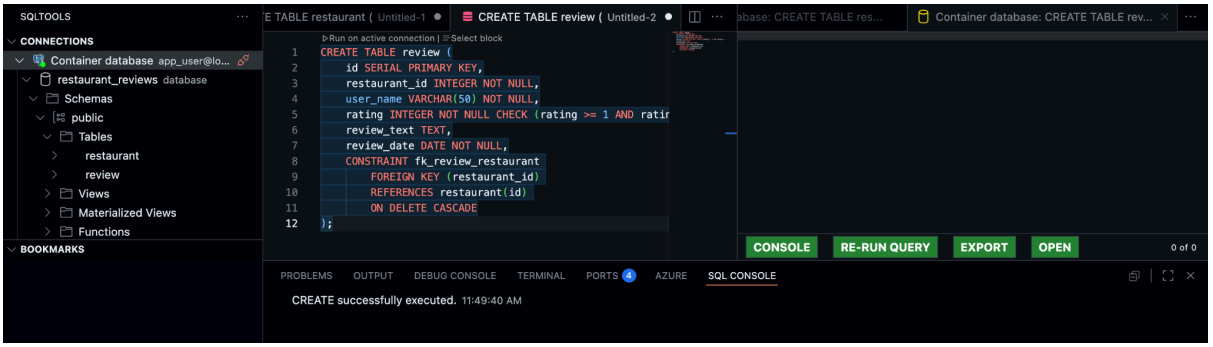


Create two tables:

restaurant table: `id, name, street_address, description`.

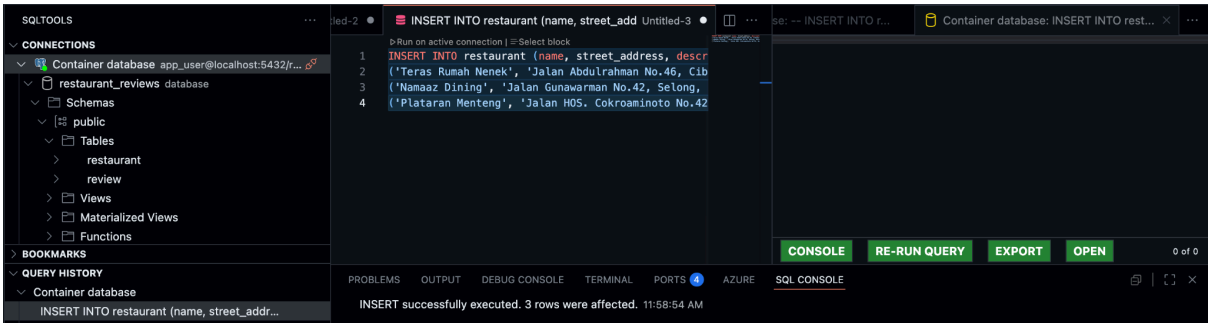


review table: `id, restaurant_id, user_name, rating, review_text, review_date`.



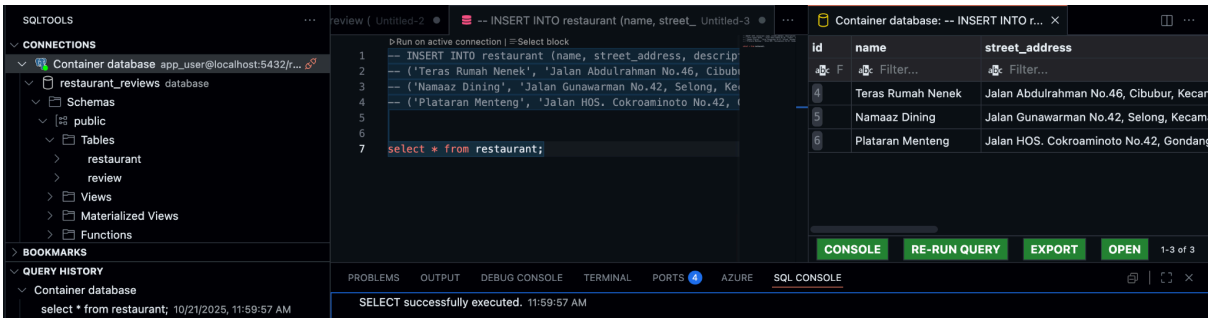
Inserting Data

Insert **at least 3 restaurants** in the restaurant table



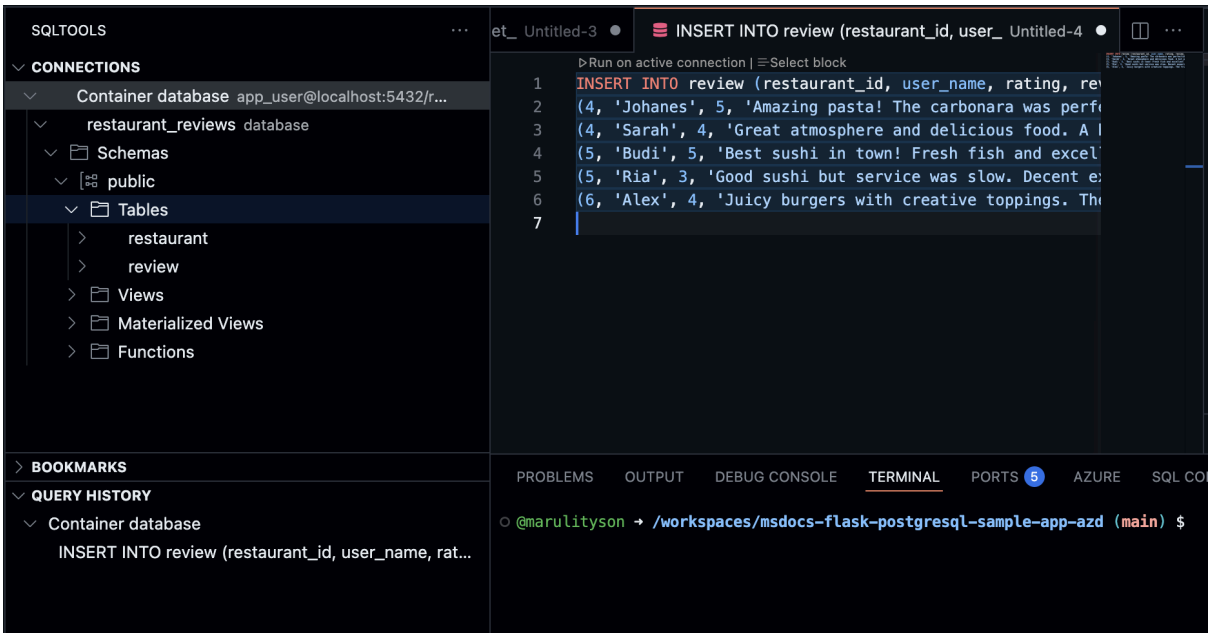
Check the data successfully inserted

*Select * from restaurant;*



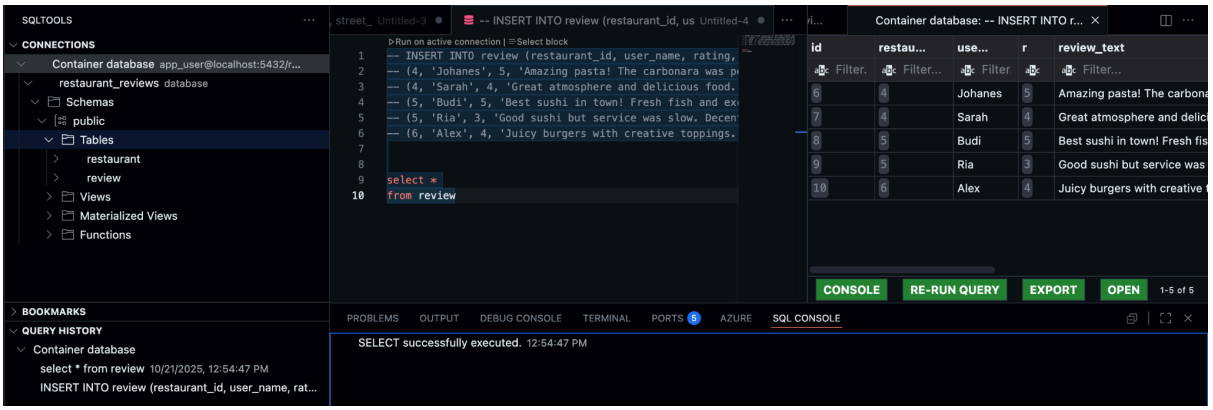
Insert **at least 5 reviews** in the review table, ensuring they reference the correct restaurants via `restaurant_id`.

```
INSERT INTO review (restaurant_id, user_name, rating, review_text, review_date)
VALUES
(4, 'Johanes', 5, 'Amazing pasta! The carbonara was perfection. Highly recommend!', '2025-10-15'),
(4, 'Sarah', 4, 'Great atmosphere and delicious food. A bit pricey but worth it.', '2025-10-18'),
(5, 'Budi', 5, 'Best sushi in town! Fresh fish and excellent presentation.', '2025-10-10'),
(5, 'Ria', 3, 'Good sushi but service was slow. Decent experience overall.', '2025-10-20'),
(6, 'Alex', 4, 'Juicy burgers with creative toppings. The fries were crispy and perfect!', '2025-10-12');
```



Check the data successfully inserted

```
Select * from review;
```

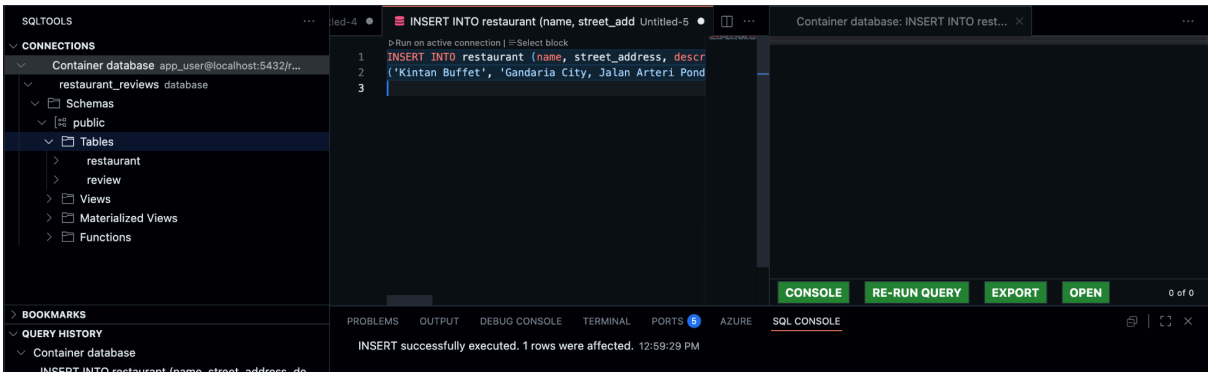


Performing CRUD Operations

Create (Insert):

1. Insert a new restaurant into the restaurant table

INSERT INTO restaurant (name, street_address, description)
VALUES ('Kintan Buffet', 'Gandaria City, Jalan Arteri Pondok Indah, Kebayoran Lama Utara, Kecamatan Kebayoran Lama, Kota Jakarta Selatan, Daerah Khusus Ibukota Jakarta', 'Menu rekomendasinya ada banyak. Salah satu menu favoritnya adalah karubi garlic yang punya tekstur lembut saat dikunyah. Soal rasa gak perlu diragukan karena dijamin enak dan sedap di lidah.');

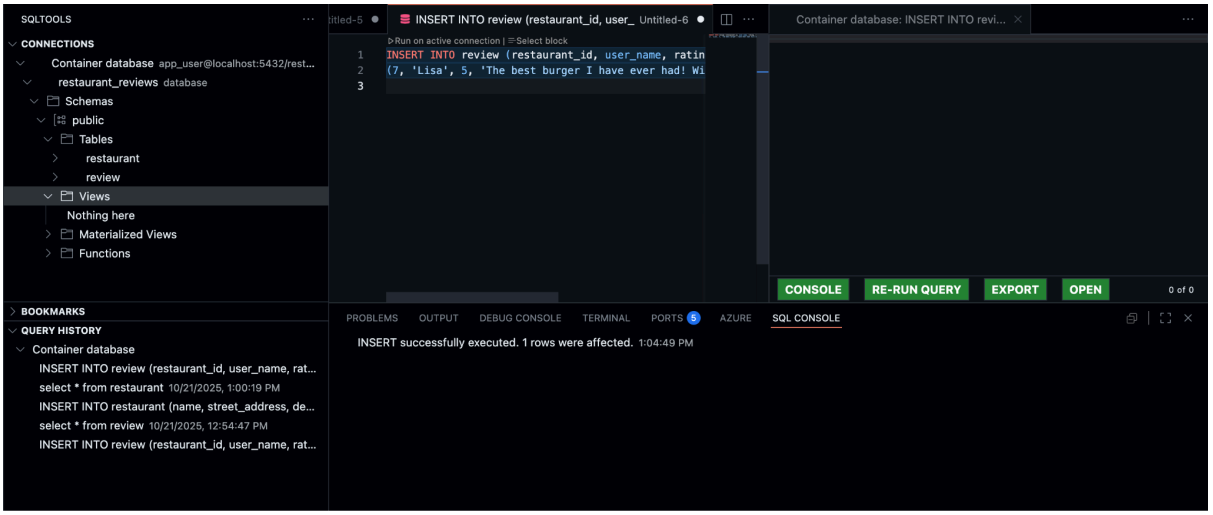


Check the data successfully inserted

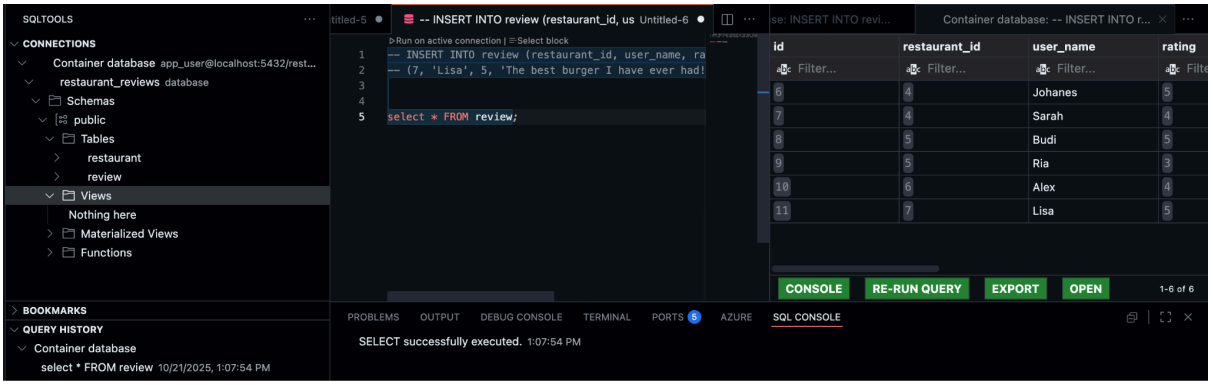
*Select * from restaurant;*

2. Insert a new review for an existing restaurant

INSERT INTO review (restaurant_id, user_name, rating, review_text, review_date) VALUES (7, 'Lisa', 5, 'The best burger I have ever had! Will definitely come back!', '2025-10-21');



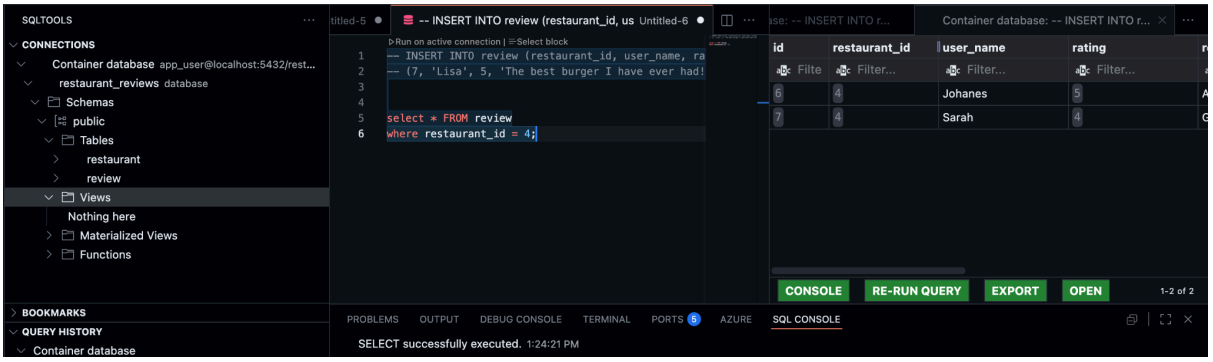
Check the data successfully inserted



Read (Select):

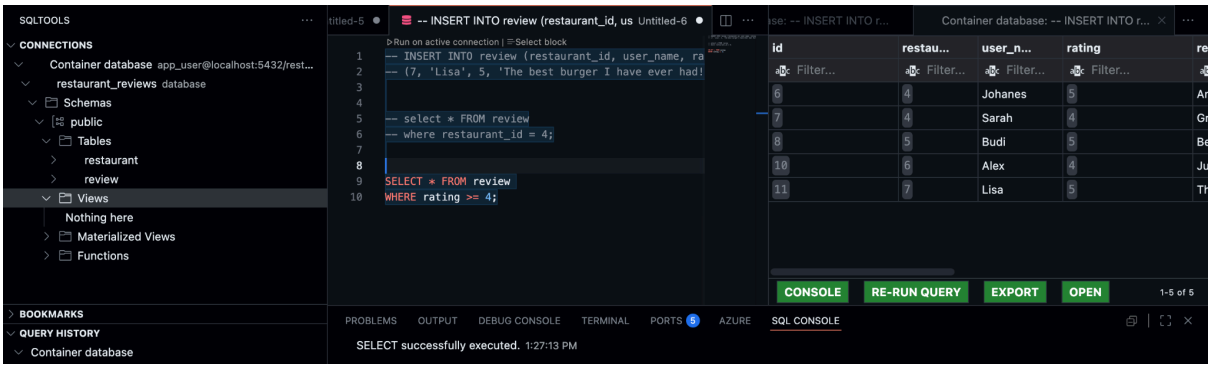
1. Retrieve all reviews for a specific restaurant using the `restaurant_id`

   *select * FROM review*
   where restaurant_id = 4;



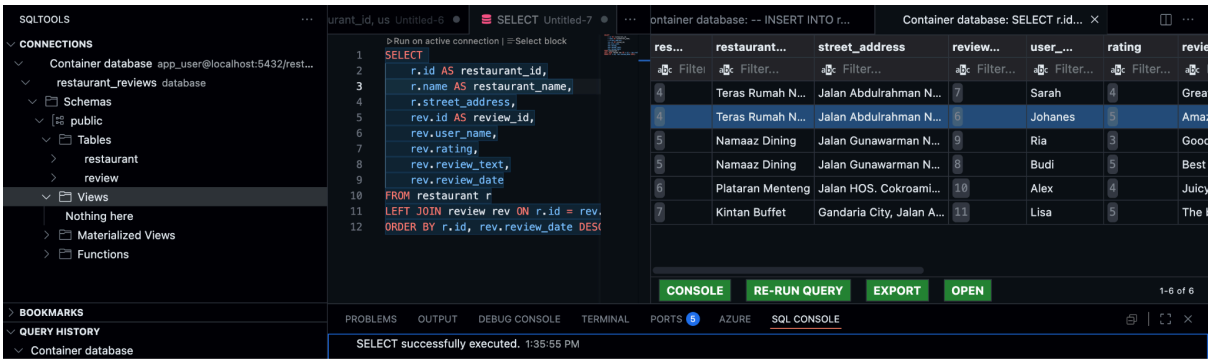
2. Retrieve all reviews with a rating of 4 or higher

   *SELECT * FROM review*
   WHERE rating >= 4;



3. Use a JOIN to display a list of restaurants along with their reviews

```
SELECT
    r.id AS restaurant_id,
    r.name AS restaurant_name,
    r.street_address,
    rev.id AS review_id,
    rev.user_name,
    rev.rating,
    rev.review_text,
    rev.review_date
FROM restaurant r
LEFT JOIN review rev ON r.id = rev.restaurant_id
ORDER BY r.id, rev.review_date DESC;
```

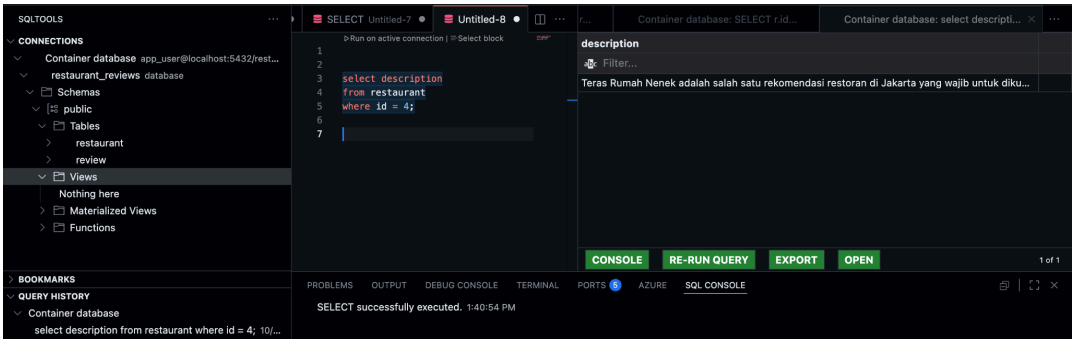


Update:

1. Update the description of one restaurant.

Before

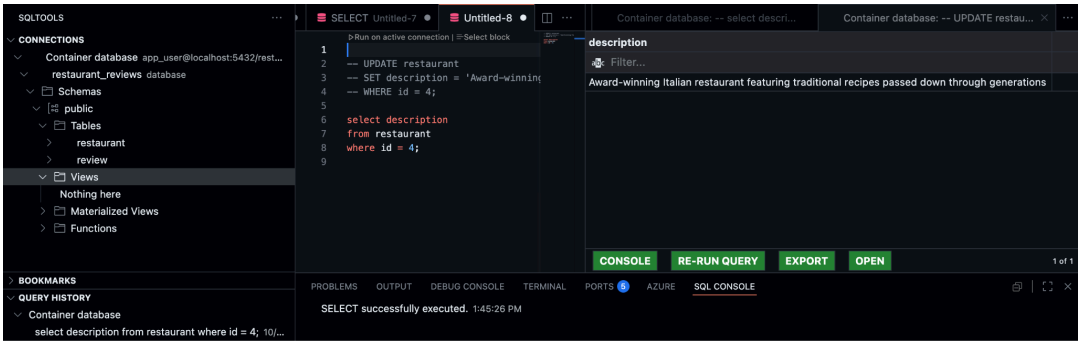
select description
from restaurant
where id = 4;



Updating data

UPDATE restaurant
SET description = 'Award-winning Italian restaurant featuring traditional recipes passed down through generations'
WHERE id = 4;

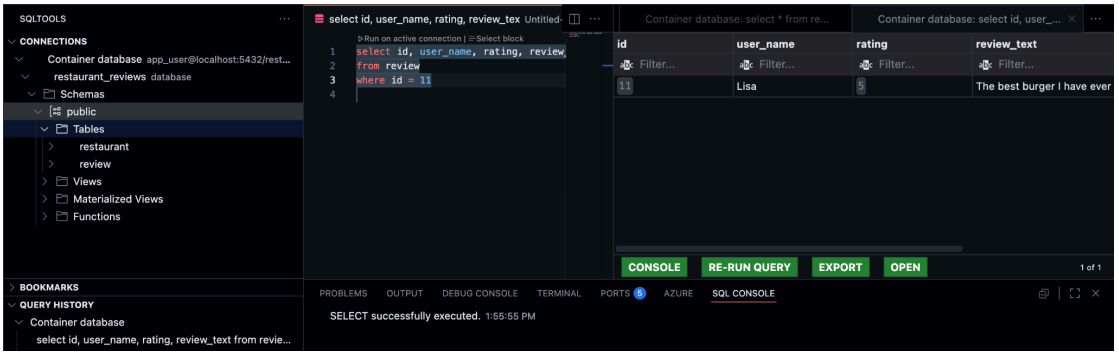
After
select description
from restaurant
where id = 4;



2. Update the rating of a specific review.

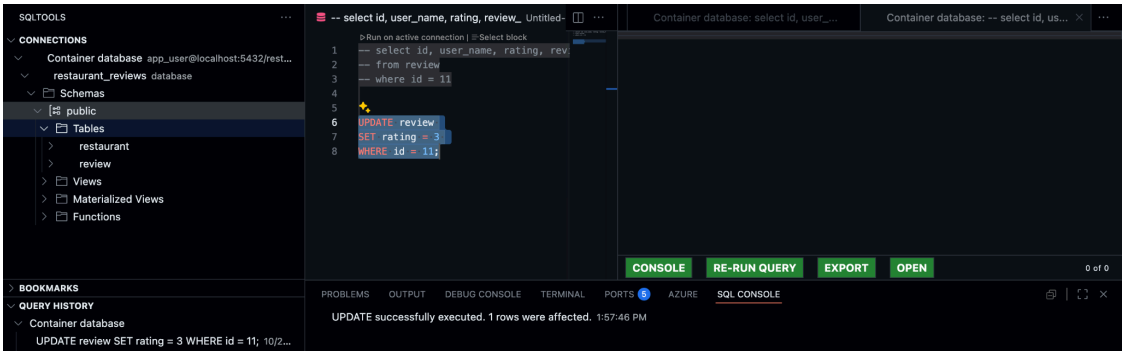
Before

```
select id, user_name, rating, review_text
from review
where id = 11;
```



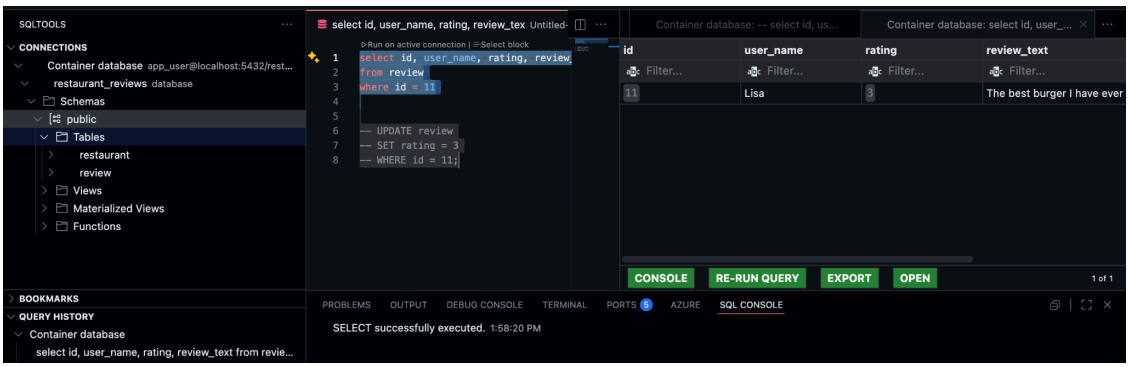
Updating data

```
UPDATE review
SET rating = 3
WHERE id = 11;
```



After

```
select id, user_name, rating, review_text
from review
where id = 11;
```

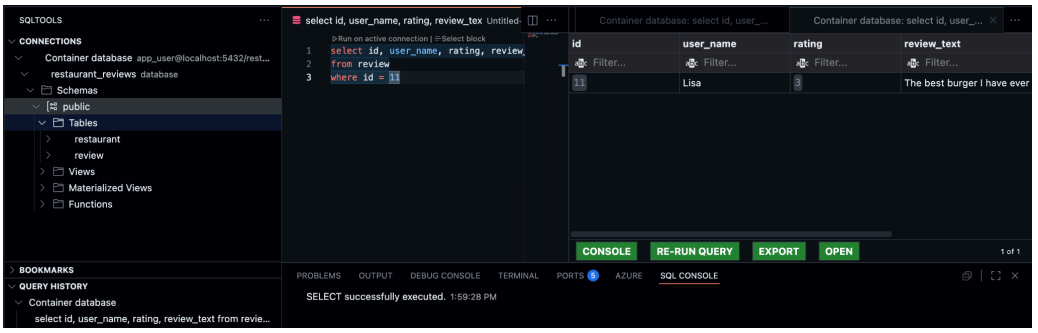


Delete:

1. Delete one review based on id

Before

```
select id, user_name, rating, review_text
from review
where id = 11;
```

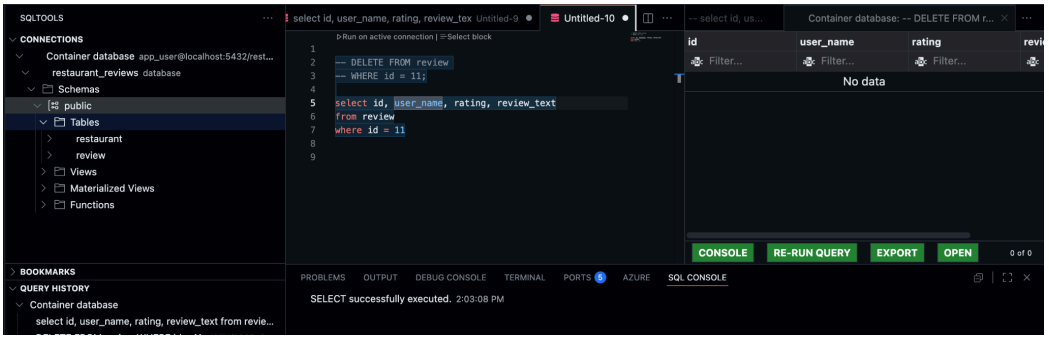


Delete

```
DELETE FROM review
WHERE id = 11;
```

After

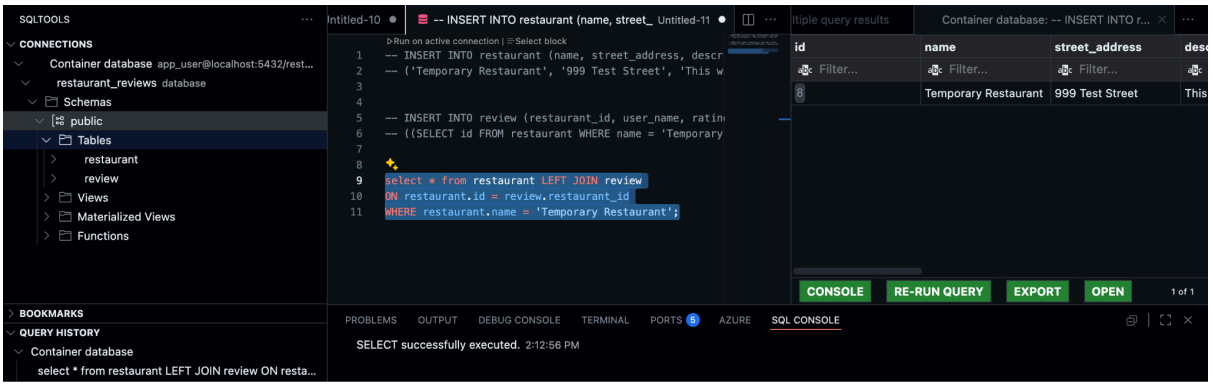
```
select id, user_name, rating, review_text
from review
where id = 11;
```



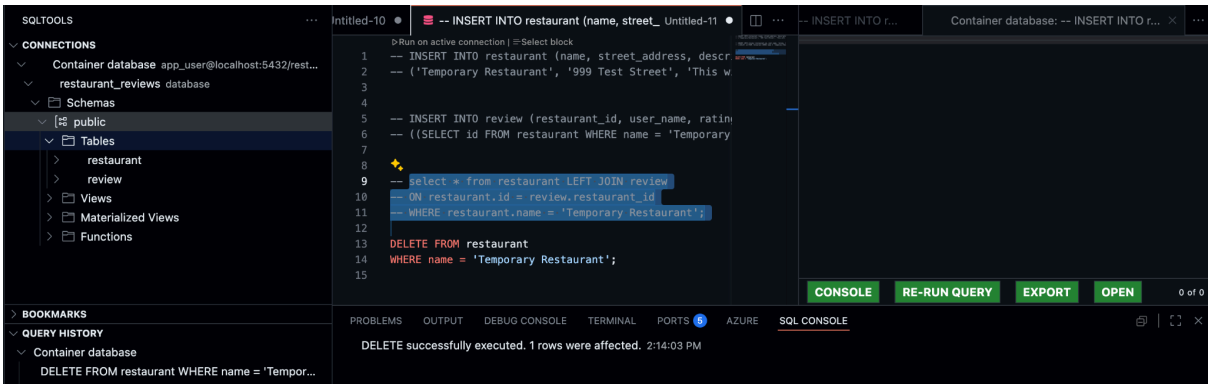
2. Delete a restaurant and ensure its associated reviews are also deleted (using cascade)

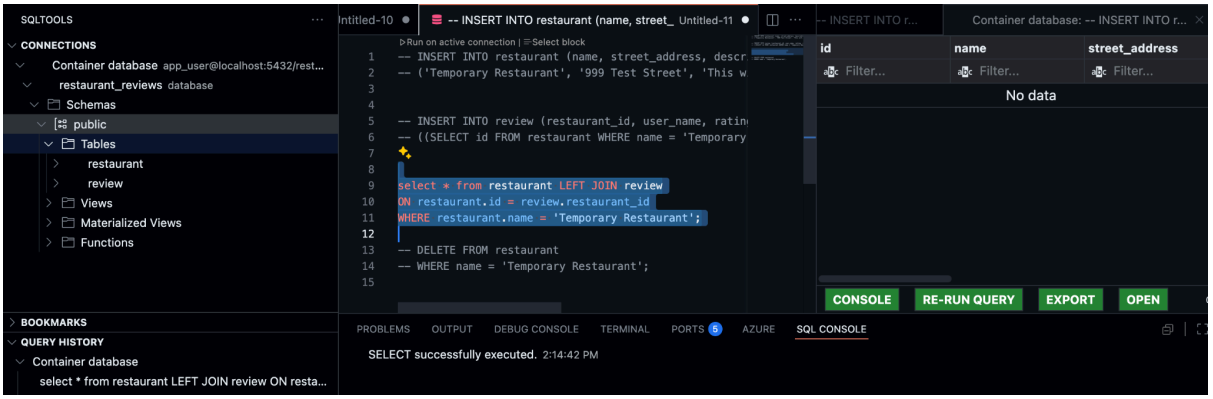
Insert new temporary data in table restaurant and review

Data inserted successfully



Deleting data

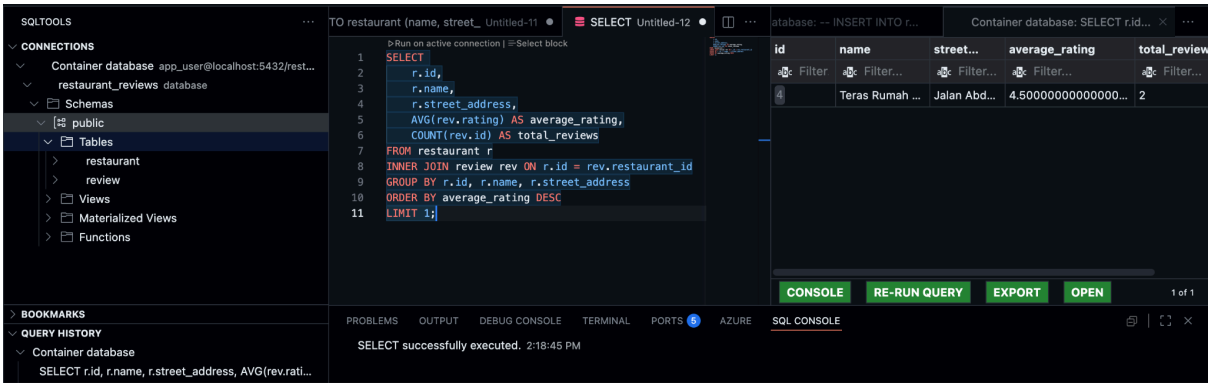



SQLTOOLS
CONNECTIONS
Container database app_user@localhost:5432/rest...
restaurant_reviews database
Schemas
public
Tables
restaurant
review
Views
Materialized Views
Functions
BOOKMARKS
QUERY HISTORY
Container database
select * from restaurant LEFT JOIN review ON resta...
Untitled-10
-- INSERT INTO restaurant (name, street_ Untitled-11
Run on active connection | Select block
-- INSERT INTO restaurant (name, street_address, descr
-- ('Temporary Restaurant', '999 Test Street', 'This w
-- INSERT INTO review (restaurant_id, user_name, ratin
-- ((SELECT id FROM restaurant WHERE name = 'Temporary
select * from restaurant LEFT JOIN review
ON restaurant.id = review.restaurant_id
WHERE restaurant.name = 'Temporary Restaurant';
-- DELETE FROM restaurant
-- WHERE name = 'Temporary Restaurant';
-- INSERT INTO r...
Container database: -- INSERT INTO r...
id
name
street_address
Filter...
No data
CONSOLE
RE-RUN QUERY
EXPORT
OPEN
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS 5
AZURE
SQL CONSOLE
SELECT successfully executed. 2:14:42 PM

Additional Queries

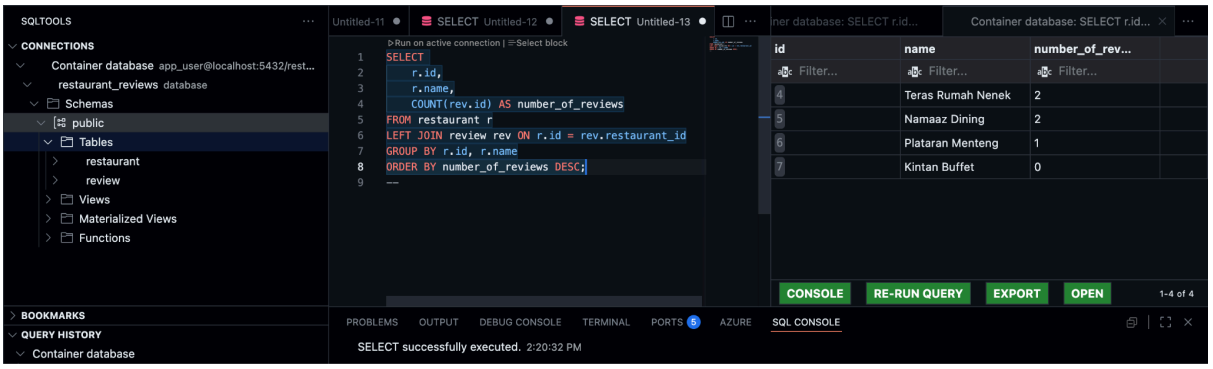
Find the highest-rated restaurant based on the average rating of all its reviews.

```
SELECT
    r.id,
    r.name,
    r.street_address,
    AVG(rev.rating) AS average_rating,
    COUNT(rev.id) AS total_reviews
FROM restaurant r
INNER JOIN review rev ON r.id = rev.restaurant_id
GROUP BY r.id, r.name, r.street_address
ORDER BY average_rating DESC
LIMIT 1;
```



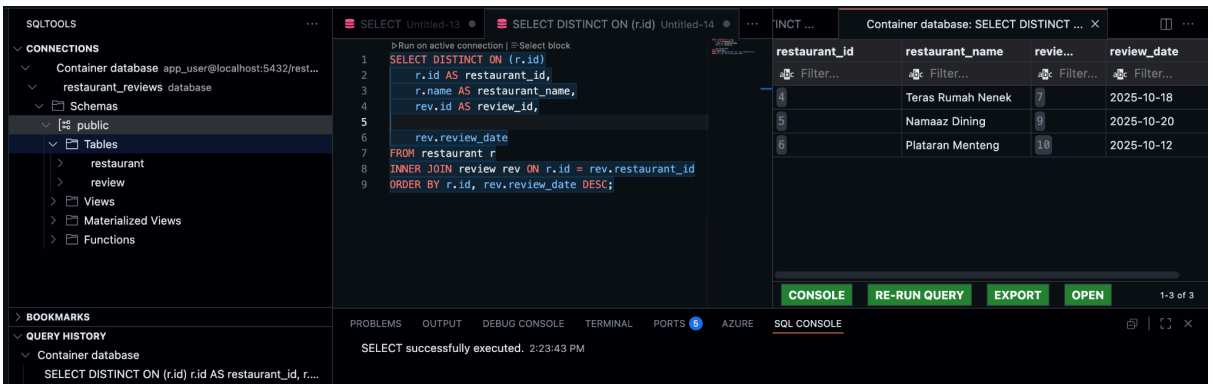
Find the number of reviews each restaurant has received.

```
SELECT
    r.id,
    r.name,
    COUNT(rev.id) AS number_of_reviews
FROM restaurant r
LEFT JOIN review rev ON r.id = rev.restaurant_id
GROUP BY r.id, r.name
ORDER BY number_of_reviews DESC;
```



Display the most recent review for each restaurant.

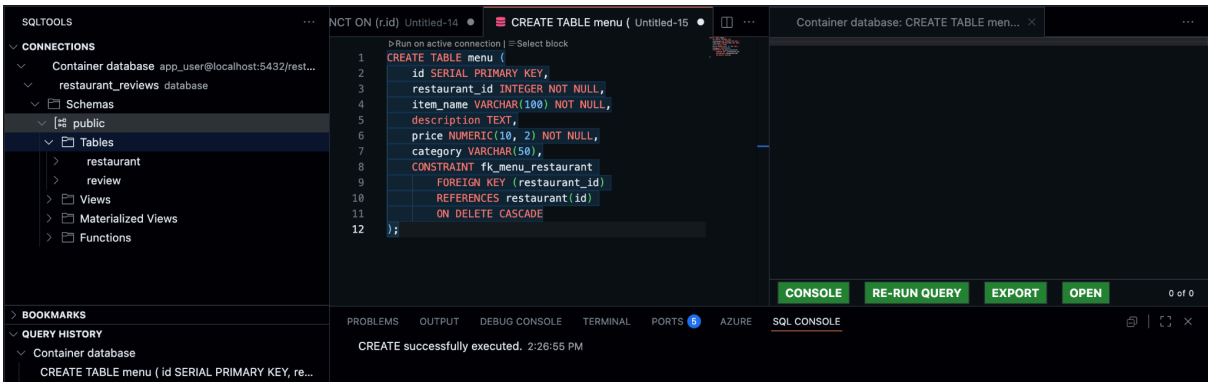
```
SELECT DISTINCT ON (r.id)
    r.id AS restaurant_id,
    r.name AS restaurant_name,
    rev.id AS review_id,
    rev.review_date
FROM restaurant r
INNER JOIN review rev ON r.id = rev.restaurant_id
ORDER BY r.id, rev.review_date DESC;
```



Extra Credit (Optional)

Create a menu table

```
CREATE TABLE menu (
    id SERIAL PRIMARY KEY,
    restaurant_id INTEGER NOT NULL,
    item_name VARCHAR(100) NOT NULL,
    description TEXT,
    price NUMERIC(10, 2) NOT NULL,
    category VARCHAR(50),
    CONSTRAINT fk_menu_restaurant
        FOREIGN KEY (restaurant_id)
        REFERENCES restaurant(id)
        ON DELETE CASCADE
);
```



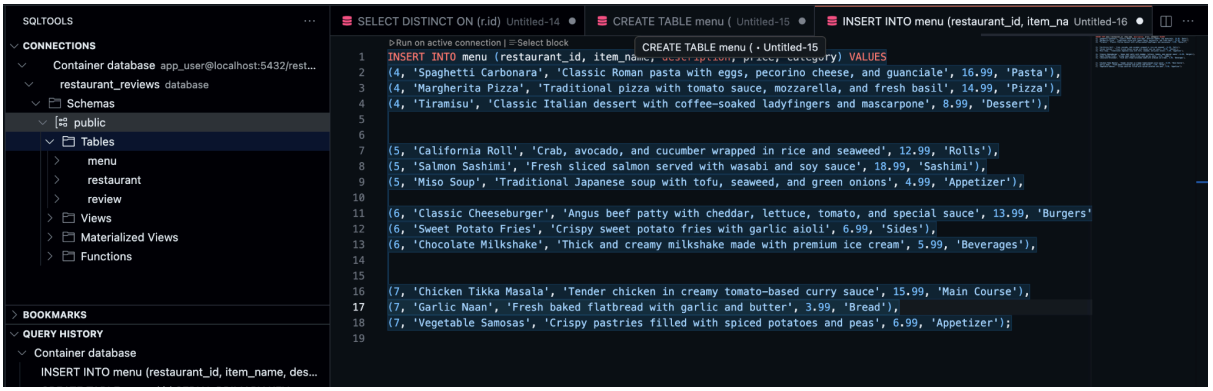
Insert at least 3 menu items for each restaurant.

INSERT INTO menu (restaurant_id, item_name, description, price, category) VALUES
(4, 'Spaghetti Carbonara', 'Classic Roman pasta with eggs, pecorino cheese, and guanciale', 16.99, 'Pasta'),
(4, 'Margherita Pizza', 'Traditional pizza with tomato sauce, mozzarella, and fresh basil', 14.99, 'Pizza'),
(4, 'Tiramisu', 'Classic Italian dessert with coffee-soaked ladyfingers and mascarpone', 8.99, 'Dessert'),

(5, 'California Roll', 'Crab, avocado, and cucumber wrapped in rice and seaweed', 12.99, 'Rolls'),
(5, 'Salmon Sashimi', 'Fresh sliced salmon served with wasabi and soy sauce', 18.99, 'Sashimi'),
(5, 'Miso Soup', 'Traditional Japanese soup with tofu, seaweed, and green onions', 4.99, 'Appetizer'),

(6, 'Classic Cheeseburger', 'Angus beef patty with cheddar, lettuce, tomato, and special sauce', 13.99, 'Burgers'),
(6, 'Sweet Potato Fries', 'Crispy sweet potato fries with garlic aioli', 6.99, 'Sides'),
(6, 'Chocolate Milkshake', 'Thick and creamy milkshake made with premium ice cream', 5.99, 'Beverages'),

(7, 'Chicken Tikka Masala', 'Tender chicken in creamy tomato-based curry sauce', 15.99, 'Main Course'),
(7, 'Garlic Naan', 'Fresh baked flatbread with garlic and butter', 3.99, 'Bread'),
(7, 'Vegetable Samosas', 'Crispy pastries filled with spiced potatoes and peas', 6.99, 'Appetizer');



Write a query to display each restaurant with its menu and the average rating from its reviews.

```
SELECT
    r.id AS restaurant_id,
    r.name AS restaurant_name,
    m.item_name AS menu_item,
    m.description AS menu_description,
    m.price,
    ROUND(AVG(rev.rating), 2) AS average_rating,
    COUNT(DISTINCT rev.id) AS total_reviews
FROM restaurant r
LEFT JOIN menu m ON r.id = m.restaurant_id
LEFT JOIN review rev ON r.id = rev.restaurant_id
GROUP BY r.id, r.name, r.street_address,
r.description, m.id, m.item_name, m.description, m.price, m.category
ORDER BY r.id, m.category, m.item_name;
```