

Nama: Titik Wihayanti

Link codespace: <https://miniature-zebra-v6wx5jw6wv642w5j7.github.dev/>

Password untuk connect ke RestaurantDB: ABjEKEMzG9eEHN2QWRatWBUX

A. Database Setup

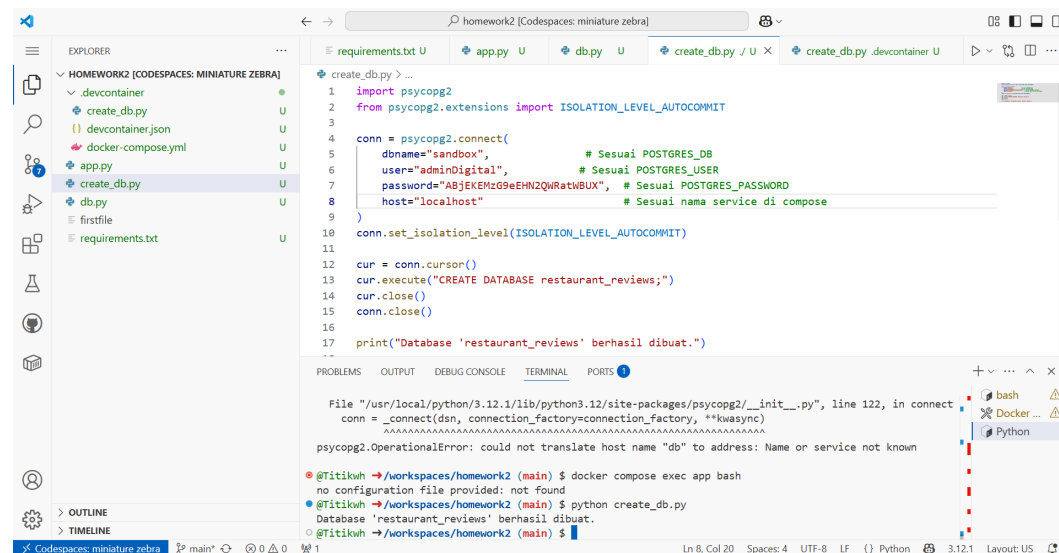
1. Create a new database `restaurant_reviews`

Script:

```
import psycopg2
from psycopg2.extensions import ISOLATION_LEVEL_AUTOCOMMIT

conn = psycopg2.connect(
    dbname="sandbox",           # Sesuai POSTGRES_DB
    user="adminDigital",        # Sesuai POSTGRES_USER
    password="ABjEKEMzG9eEHN2QWRatWBUX", # Sesuai POSTGRES_PASSWORD
    host="db"                   # Sesuai nama service di compose
)
conn.set_isolation_level(ISOLATION_LEVEL_AUTOCOMMIT)
cur = conn.cursor()
cur.execute("CREATE DATABASE restaurant_reviews;")
cur.close()
conn.close()
print("Database 'restaurant_reviews' berhasil dibuat.")
```

Result:

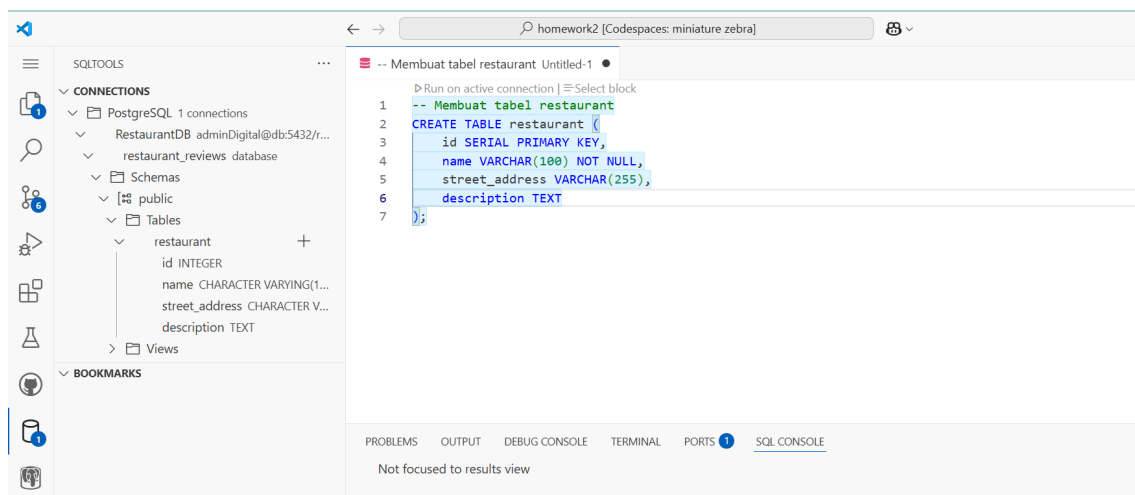


2. Create **Restaurant** table, Column: **id**, **name**, **street_address**, **description**

Script:

```
-- Membuat tabel restaurant
CREATE TABLE restaurant (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    street_address VARCHAR(255),
    description TEXT );
```

Result:

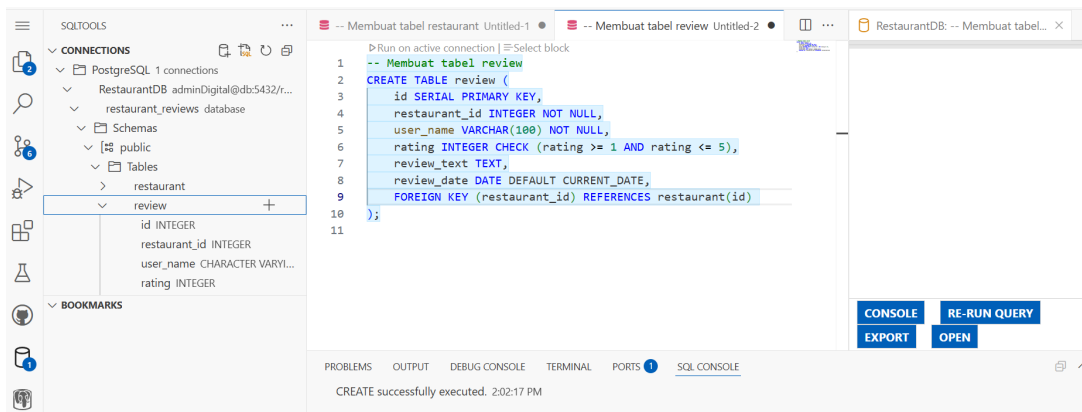


3. Create **Review** table with relation to **restaurant_id**

Script:

```
-- Membuat tabel review
CREATE TABLE review (
    id SERIAL PRIMARY KEY,
    restaurant_id INTEGER NOT NULL,
    user_name VARCHAR(100) NOT NULL,
    rating INTEGER CHECK (rating >= 1 AND rating <= 5),
    review_text TEXT,
    review_date DATE DEFAULT CURRENT_DATE,
    FOREIGN KEY (restaurant_id) REFERENCES restaurant(id));
```

Result:



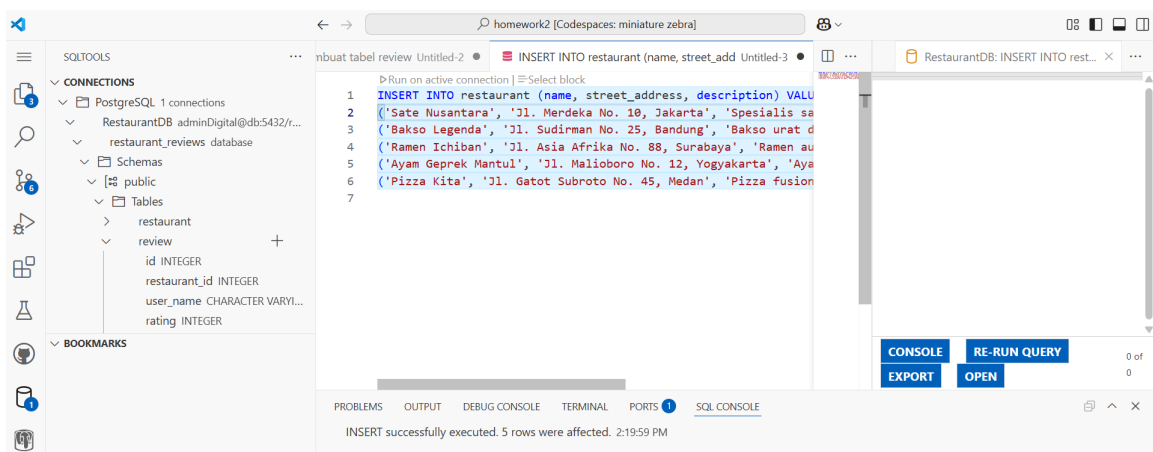
A. Inserting Data

1. Insert 5 data into Restaurant table

Script:

```
INSERT INTO restaurant (name, street_address, description) VALUES
('Sate Nusantara', 'Jl. Merdeka No. 10, Jakarta', 'Spesialis sate dari seluruh nusantara'),
('Bakso Legenda', 'Jl. Sudirman No. 25, Bandung', 'Bakso urat dan bakso isi keju legendaris'),
('Ramen Ichiban', 'Jl. Asia Afrika No. 88, Surabaya', 'Ramen autentik Jepang dengan kuah khas'),
('Ayam Geprek Mantul', 'Jl. Malioboro No. 12, Yogyakarta', 'Ayam geprek super pedas dengan sambal pilihan'),
('Pizza Kita', 'Jl. Gatot Subroto No. 45, Medan', 'Pizza fusion lokal dengan topping unik');
```

Result:



id	name	street_address
1	Sate Nusantara	Jl. Merdeka No. 10, Jakarta
2	Bakso Legenda	Jl. Sudirman No. 25, Bandung
3	Ramen Ichiban	Jl. Asia Afrika No. 88, Surabaya
4	Ayam Geprek Mantul	Jl. Malioboro No. 12, Yogyakarta
5	Pizza Kita	Jl. Gatot Subroto No. 45, Medan
6	Sate Nusantara Baru	Jl. Merdeka No. 11, Jakarta
7	Bakso Legenda Baru	Jl. Sudirman No. 26, Bandung
8	Ramen Ichiban Baru	Jl. Asia Afrika No. 89, Surabaya
9	Ayam Geprek Mantul ...	Jl. Malioboro No. 13, Yogyakarta

2. Insert 5 data into Review table

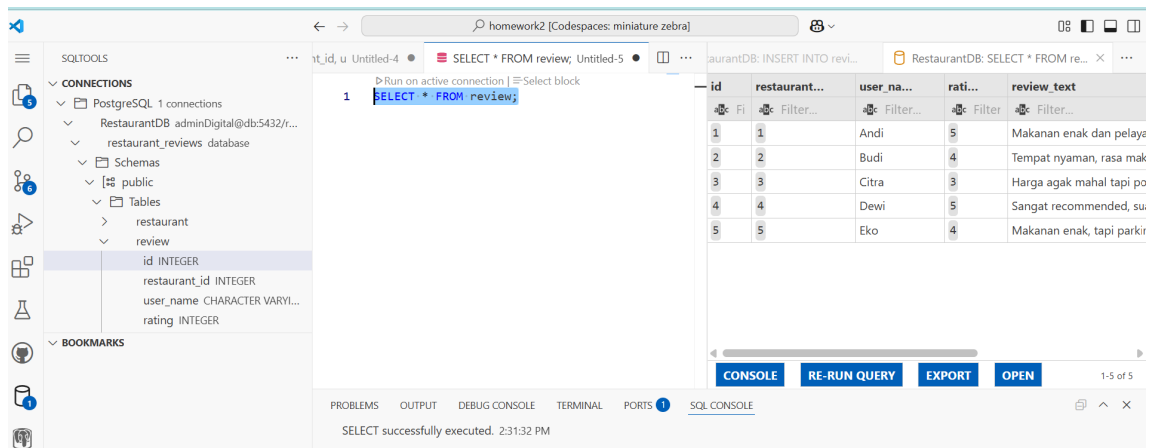
Script:

```
INSERT INTO review (id, restaurant_id, user_name, rating,
review_text, review_date) VALUES
(1, 1, 'Andi', 5, 'Makanan enak dan pelayanan cepat.', '2025-05-01'),
(2, 2, 'Budi', 4, 'Tempat nyaman, rasa makanan cukup lezat.', '2025-05-03'),
(3, 3, 'Citra', 3, 'Harga agak mahal tapi porsi besar.', '2025-05-05'),
(4, 4, 'Dewi', 5, 'Sangat recommended, suasana asik.', '2025-05-07'),
(5, 5, 'Eko', 4, 'Makanan enak, tapi parkir susah.', '2025-05-09');
```

Result:

```
1 INSERT INTO review (id, restaurant_id, user_name, rating, review_text, review_date) VALUES
2 (1, 1, 'Andi', 5, 'Makanan enak dan pelayanan cepat.', '2025-05-01'),
3 (2, 2, 'Budi', 4, 'Tempat nyaman, rasa makanan cukup lezat.', '2025-05-03'),
4 (3, 3, 'Citra', 3, 'Harga agak mahal tapi porsi besar.', '2025-05-05'),
5 (4, 4, 'Dewi', 5, 'Sangat recommended, suasana asik.', '2025-05-07'),
6 (5, 5, 'Eko', 4, 'Makanan enak, tapi parkir susah.', '2025-05-09');
7
```

INSERT successfully executed. 5 rows were affected. 2:29:08 PM



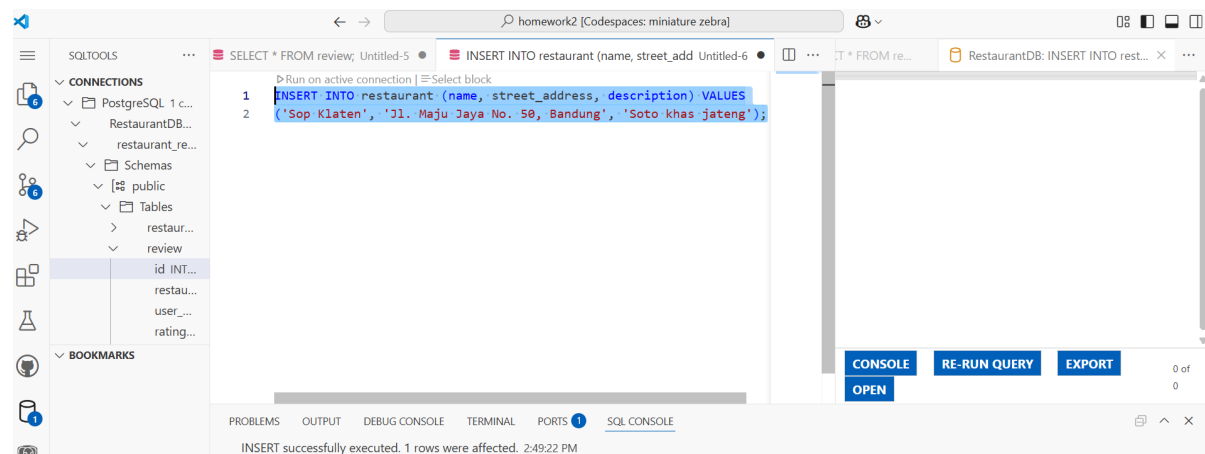
B. Performing CRUD Operations

1. Create (Insert) One Data to Restaurant

Script:

```
INSERT INTO restaurant (name, street_address, description) VALUES
('Sop Klaten', 'Jl. Maju Jaya No.50, Bandung', 'Soto Khas Jawa');
```

Result:

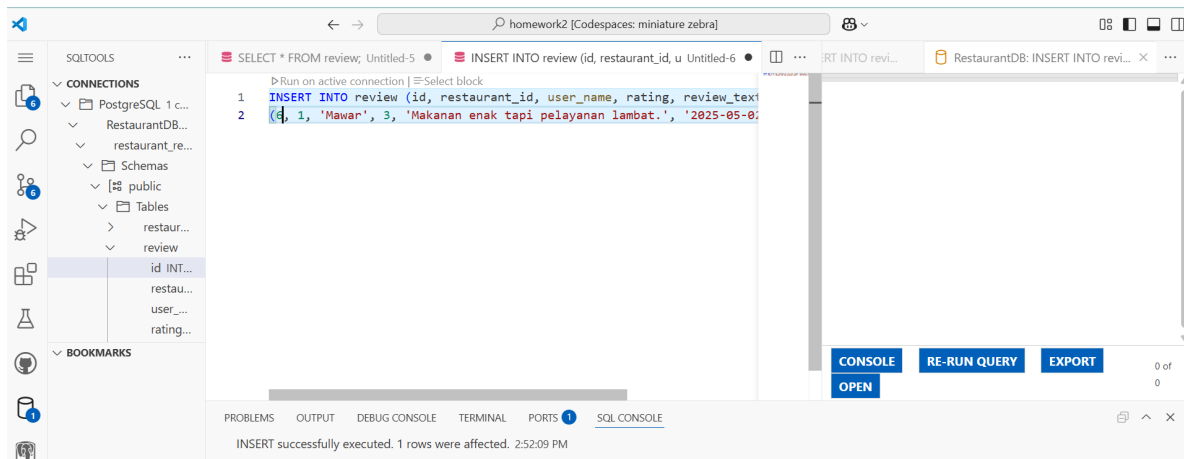


2. Create (Insert) One Data to Review

Script:

```
INSERT INTO review (id, restaurant_id, user_name, rating,
review_text, review_date) VALUES
(6, 1, 'Mawar', 3, 'Makanan enak tapi pelayanan lambat', '2025-05-01'),
```

Result:



C. Read (Select)

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

Script:

```
SELECT * FROM review
WHERE restaurant_id=4;
```

Result:

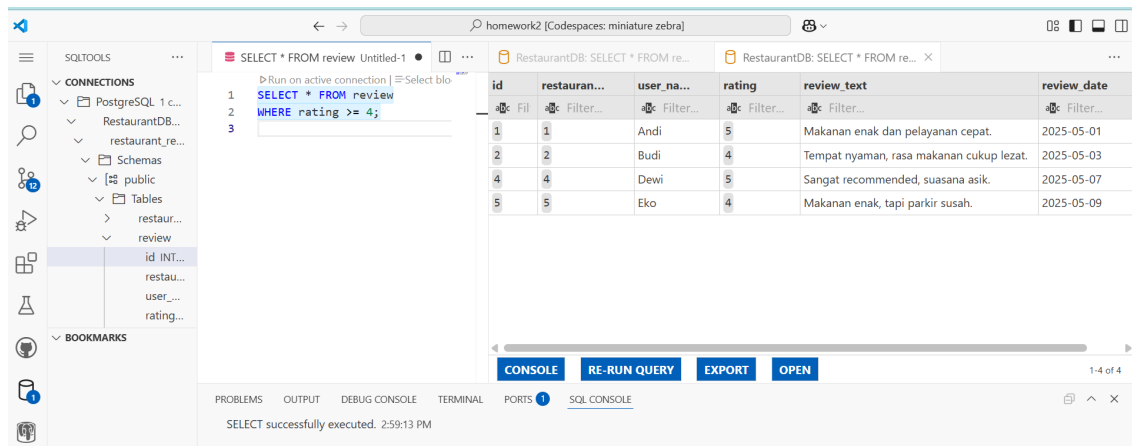
id	resta...	user_name	rat...	review_text	review_date
4	4	Dewi	5	Sangat recommended, suasana asik.	2025-05-07

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

Script:

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

Result:

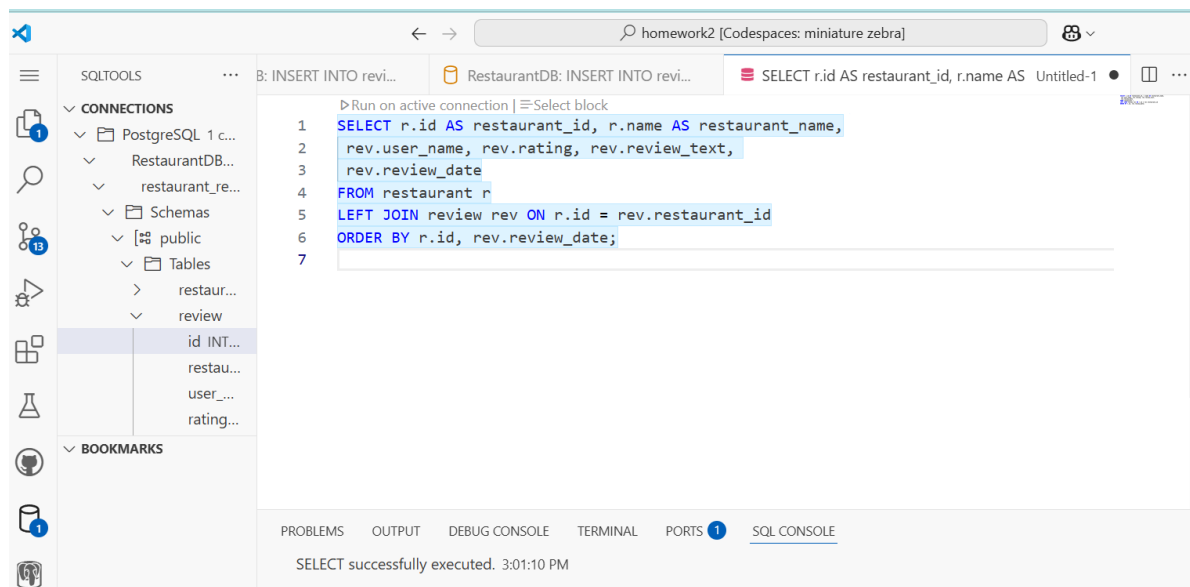


3. Use a **JOIN** to display a list of **restaurants** along with their **reviews**.

Script:

```
SELECT r.id AS restaurant_id, r.name AS restaurant_name,
       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;
```

Result:



rest...	restaurant_name	user_name	rating	review_text	review
1	Sate Nusantara	Andi	5	Makanan enak dan pelayanan cepat.	2025-0
1	Sate Nusantara	Mawar	3	Makanan enak tapi pelayanan lambat.	2025-0
2	Bakso Legenda	Budi	4	Tempat nyaman, rasa makanan cukup lezat.	2025-0
3	Ramen Ichiban	Citra	3	Harga agak mahal tapi porsi besar.	2025-0
4	Ayam Geprek Mantul	Dewi	5	Sangat recommended, suasana asik.	2025-0
5	Pizza Kita	Eko	4	Makanan enak, tapi parkir susah.	2025-0
6	Sate Nusantara Baru	NULL	NULL	NULL	
7	Bakso Legenda Baru	NULL	NULL	NULL	
8	Ramen Ichiban Baru	NULL	NULL	NULL	

D. Update

1. Update the **description** of one restaurant

Script:

```
UPDATE restaurant
SET description = 'Deskripsi baru untuk restoran ini.'
WHERE id = 3;
```

Result:

The screenshot shows a database client interface with a query editor and a console. The query editor contains the following SQL script:

```
1 UPDATE restaurant
2 SET description = 'Deskripsi baru untuk restoran ini.'
3 WHERE id = 3;
```

The console at the bottom displays the message: "UPDATE successfully executed. 1 rows were affected. 3:04:25 PM".

The screenshot shows a database client interface with a query editor and a result set. The query editor contains the following SQL script:

```
1 SELECT description from restaurant WHERE id=3;
```

The result set displays the following data:

description
Deskripsi baru untuk restoran ini.

2. Update the rating of a specific review

Script:

```
UPDATE review
SET rating = 5
WHERE id = 4;
```

Result:

The screenshot shows a database IDE interface. On the left, a tree view displays the database schema, including a table named 'review' with columns 'id', 'rating', 'user', and 'restaurant'. The 'review' table is selected. In the center, a SQL editor shows the following query:

```
1 UPDATE review
2 SET rating = 5
3 WHERE id = 4;
```

Below the editor, a status bar indicates: "UPDATE successfully executed. 1 rows were affected. 3:07:05 PM". On the right, a 'SQL CONSOLE' tab is active, showing the same query. Below it, a table titled 'rating' displays the result of the query:

rating
5

E. Delete

1. Delete one review based on id.

Script:

```
DELETE FROM review
WHERE id = 3;
```

Result:

The screenshot shows the same database IDE interface. The SQL editor now contains the following query:

```
1 DELETE FROM review
2 WHERE id = 3;
```

The status bar indicates: "DELETE successfully executed. 1 rows were affected. 3:09:35 PM". The 'SQL CONSOLE' tab is also active, showing the same query.

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

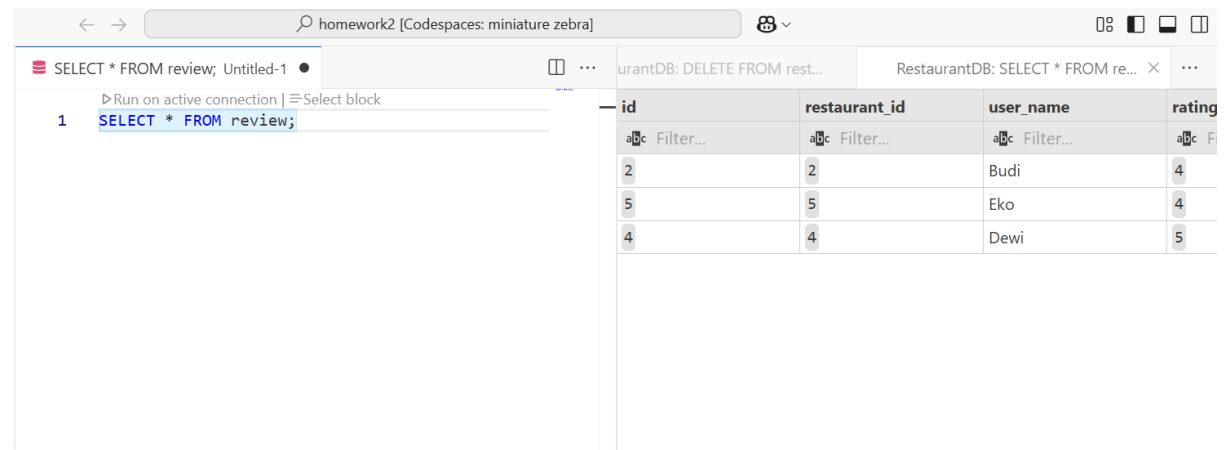
Script:

```
-- 1. Hapus constraint lama
ALTER TABLE review
DROP CONSTRAINT review_restaurant_id_fkey;

-- 2. Tambahkan constraint baru dengan ON DELETE CASCADE
ALTER TABLE review
ADD CONSTRAINT review_restaurant_id_fkey
FOREIGN KEY (restaurant_id) REFERENCES restaurant(id) ON DELETE CASCADE;

-- 3. Tambahkan constraint baru dengan ON DELETE CASCADE
DELETE FROM restaurant WHERE id = 1;
```

Result:



The screenshot shows a database interface with a query editor on the left and a result table on the right. The query editor contains the SQL command `SELECT * FROM review;`. The result table displays the following data:

id	restaurant_id	user_name	rating
2	2	Budi	4
5	5	Eko	4
4	4	Dewi	5

F. Additional Queries

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

Script:

```
SELECT r.id, r.name, AVG(rv.rating) AS avg_rating
FROM restaurant r
JOIN review rv ON r.id = rv.restaurant_id
GROUP BY r.id, r.name
ORDER BY avg_rating DESC
LIMIT 1;
```

Result:

The screenshot shows the VS Code SQL Tools interface. The left sidebar displays the 'CONNECTIONS' panel with a PostgreSQL connection named 'RestaurantDB'. The main editor shows a SQL query: `SELECT r.id, r.name, AVG(rv.rating) AS avg_rating FROM restaurant r JOIN review rv ON r.id = rv.restaurant_id GROUP BY r.id, r.name ORDER BY avg_rating DESC LIMIT 1;`. The right sidebar shows the 'SQL CONSOLE' with the query results: a table with columns 'id', 'name', and 'avg_rating'. The result row is:

id	name	avg_rating
4	Ayam Geprek Mantul	5.0000000000000000

. The status bar at the bottom indicates 'SELECT successfully executed. 3:11:48 PM'.

2. Find the **number of reviews** each **restaurant** has received

Script:

```
SELECT r.id, r.name, COUNT(rv.id) AS review_count
FROM restaurant r
LEFT JOIN review rv ON r.id = rv.restaurant_id
GROUP BY r.id, r.name;
```

Result:

The screenshot shows the VS Code SQL Tools interface. The left sidebar displays the 'CONNECTIONS' panel with a PostgreSQL connection named 'RestaurantDB'. The main editor shows a SQL query: `SELECT r.id, r.name, COUNT(rv.id) AS review_count FROM restaurant r LEFT JOIN review rv ON r.id = rv.restaurant_id GROUP BY r.id, r.name;`. The right sidebar shows the 'SQL CONSOLE' with the query results: a table with columns 'id', 'name', and 'review_count'. The result rows are:

id	name	review_count
4	Ayam Geprek Mantul	1
10	Pizza Kita Baru	0
6	Sate Nusantara Baru	0
2	Bakso Legenda	1
11	Sop Klaten	0
7	Bakso Legenda Baru	0
9	Ayam Geprek Mantul ...	0
3	Ramen Ichiban	0
1	Sate Nusantara	2
5	Pizza Kita	1

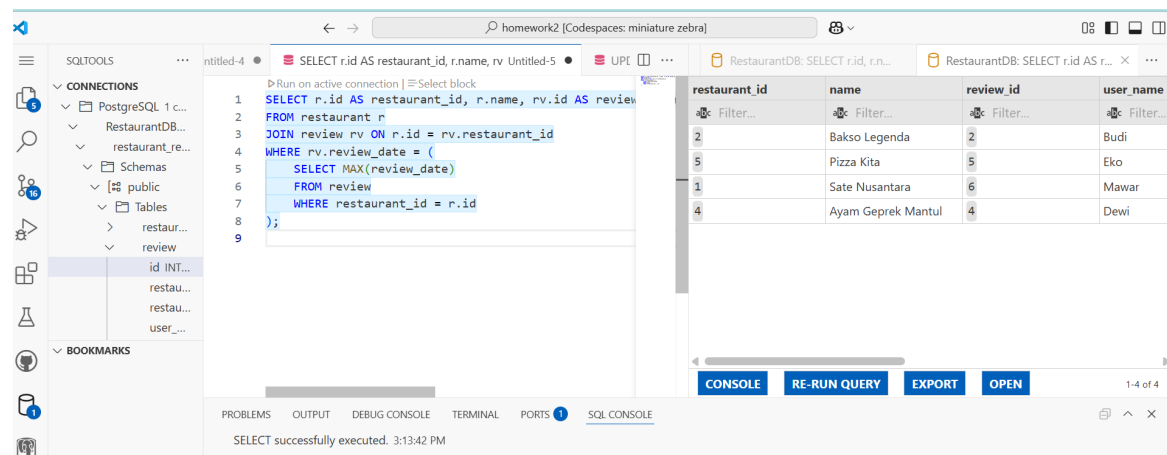
. The status bar at the bottom indicates 'SELECT successfully executed. 3:12:45 PM'.

3. Display the **most recent review** for each **restaurant**

Script:

```
SELECT r.id AS restaurant_id, r.name, rv.id AS review_id, rv.user_name,
rv.rating, rv.review_text, rv.review_date
FROM restaurant r
JOIN review rv ON r.id = rv.restaurant_id
WHERE rv.review_date = (
    SELECT MAX(review_date)
    FROM review
    WHERE restaurant_id = r.id
);
```

Result:



The screenshot shows a SQL IDE interface. On the left, a tree view shows a database connection to 'PostgreSQL 1...' with a schema 'public' containing tables 'restaurant' and 'review'. The main editor displays a SQL query: `SELECT r.id AS restaurant_id, r.name, rv.id AS review_id, rv.user_name, rv.rating, rv.review_text, rv.review_date FROM restaurant r JOIN review rv ON r.id = rv.restaurant_id WHERE rv.review_date = (SELECT MAX(review_date) FROM review WHERE restaurant_id = r.id);`. The bottom right pane shows the query results in a table with 4 rows and 4 columns: 'restaurant_id', 'name', 'review_id', and 'user_name'. The results are: (2, 'Bakso Legenda', 2, 'Budi'), (5, 'Pizza Kita', 5, 'Eko'), (1, 'Sate Nusantara', 6, 'Mawar'), and (4, 'Ayam Geprek Mantul', 4, 'Dewi').

restaurant_id	name	review_id	user_name
2	Bakso Legenda	2	Budi
5	Pizza Kita	5	Eko
1	Sate Nusantara	6	Mawar
4	Ayam Geprek Mantul	4	Dewi

G. Extra Credit

1. Create a menu table, similar to the one used in our class session, and insert at least 3 menu items for each restaurant

Script:

```
SELECT r.id AS restaurant_id, r.name, rv.id AS review_id, rv.user_name,
rv.rating, rv.review_text, rv.review_date
FROM restaurant r
JOIN review rv ON r.id = rv.restaurant_id
WHERE rv.review_date = (
    SELECT MAX(review_date)
    FROM review
    WHERE restaurant_id = r.id
);
```

--Insert 3 menu item untuk setiap restoran

```
INSERT INTO menu (restaurant_id, item_name, description, price) VALUES
(1, 'Nasi Goreng Spesial', 'Nasi goreng dengan telur dan ayam',
25000.00),
```

```
(1, 'Mie Ayam Bakso', 'Mie ayam dengan bakso sapi kenyal', 20000.00),
```

```
(1, 'Es Teh Manis', 'Teh manis dingin segar', 8000.00),
```

```
(2, 'Pizza Margherita', 'Pizza dengan saus tomat dan keju mozzarella',
75000.00),
```

```
(2, 'Pasta Carbonara', 'Pasta dengan saus krim dan bacon', 65000.00),
```

```
(2, 'Tiramisu', 'Dessert khas Italia dengan kopi dan keju mascarpone',
40000.00),
```

```
(3, 'Burger Classic', 'Burger daging sapi dengan selada dan tomat',
45000.00),
```

```
(3, 'Kentang Goreng', 'Kentang goreng renyah', 20000.00),
```

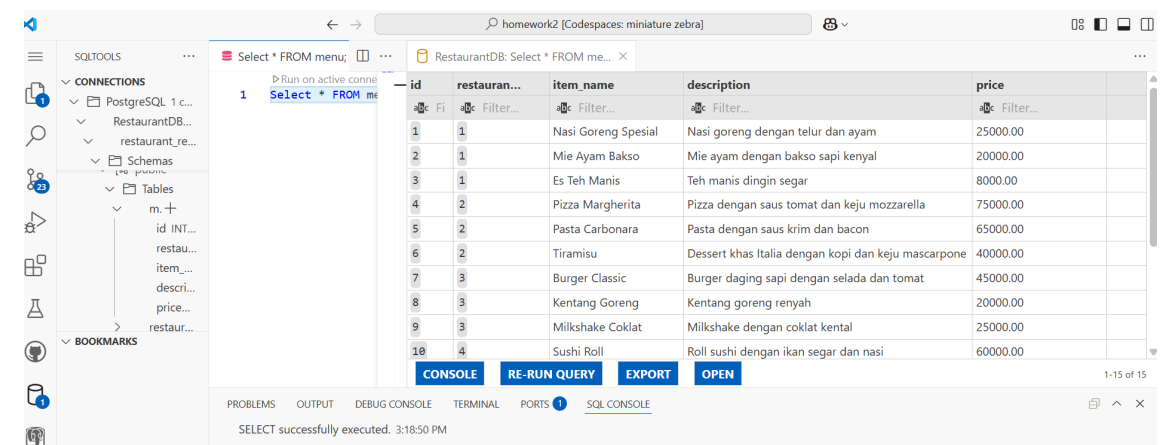
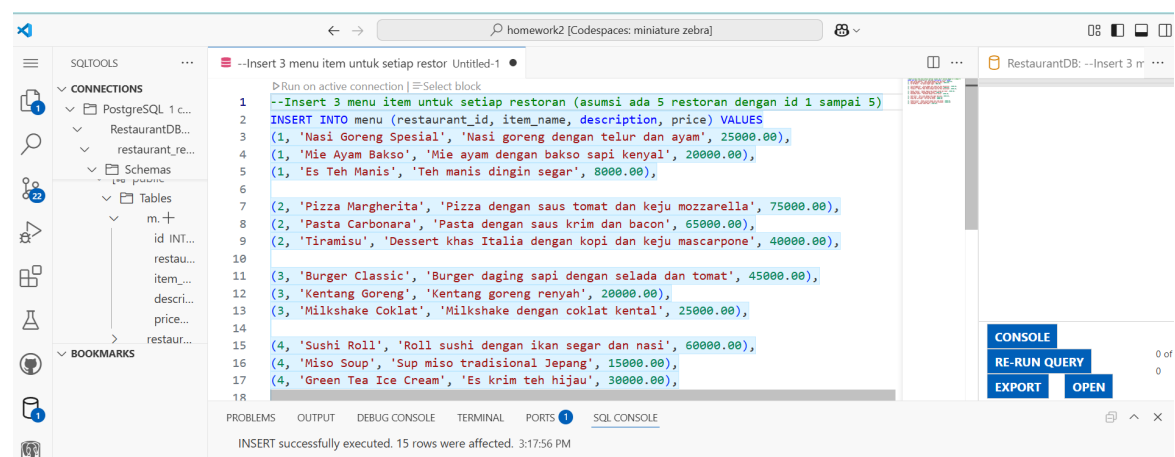
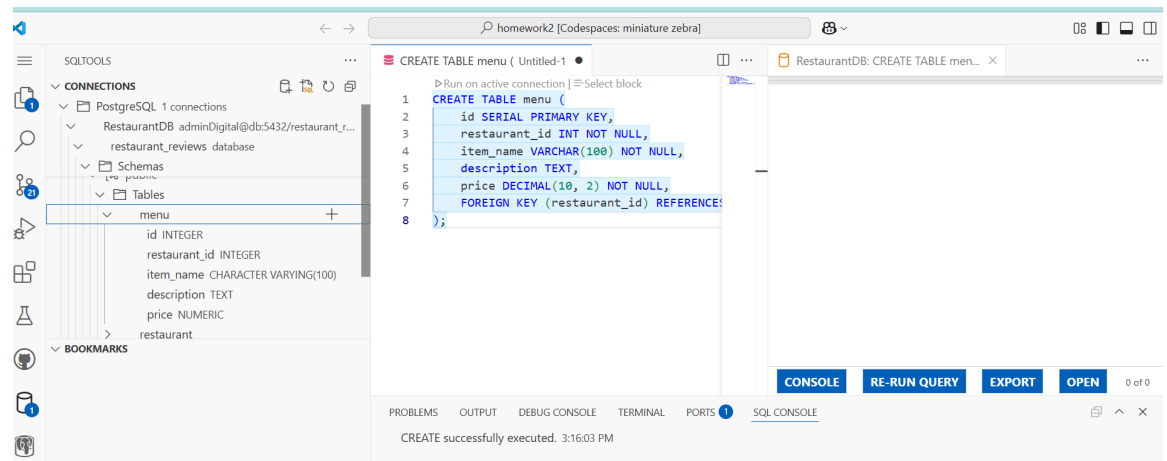
```
(3, 'Milkshake Coklat', 'Milkshake dengan coklat kental', 25000.00),
```

```
(4, 'Sushi Roll', 'Roll sushi dengan ikan segar dan nasi', 60000.00),
```

```
(4, 'Miso Soup', 'Sup miso tradisional Jepang', 15000.00),
(4, 'Green Tea Ice Cream', 'Es krim teh hijau', 30000.00),

(5, 'Steak Sirloin', 'Steak sirloin dengan saus lada hitam', 85000.00),
(5, 'Salad Caesar', 'Salad dengan saus caesar dan croutons', 35000.00),
(5, 'Lemonade', 'Minuman lemon segar', 12000.00);
```

Result:



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

Script:

SELECT

```
r.id AS restaurant_id,  
r.name AS restaurant_name,  
m.item_name AS menu_item,  
m.price AS menu_price,  
COALESCE(AVG(rv.rating), 0) AS average_rating
```

FROM

```
restaurant r
```

LEFT JOIN

```
menu m ON r.id = m.restaurant_id
```

LEFT JOIN

```
review rv ON r.id = rv.restaurant_id
```

GROUP BY

```
r.id, r.name, m.item_name, m.price
```

ORDER BY

```
r.id, m.item_name;
```

Result:

The screenshot shows a SQL IDE interface with a query editor on the left and a results pane on the right. The query is a SELECT statement that joins the restaurant, menu, and review tables. The results pane displays a table with 4 columns: restaurant_id, restaurant_name, menu_item, and menu_price. The data is sorted by restaurant_id and then by menu_item.

restaurant_id	restaurant_name	menu_item	menu_price
1	Sate Nusantara	Es Teh Manis	8000.00
1	Sate Nusantara	Mie Ayam Bakso	20000.00
1	Sate Nusantara	Nasi Goreng Spesial	25000.00
2	Bakso Legenda	Pasta Carbonara	65000.00
2	Bakso Legenda	Pizza Margherita	75000.00
2	Bakso Legenda	Tiramisu	40000.00
3	Ramen Ichiban	Burger Classic	45000.00
3	Ramen Ichiban	Kentang Goreng	20000.00
3	Ramen Ichiban	Milkshake Coklat	25000.00

restaurant_id	restaurant_name	menu_item	menu_price	average_rating	
1	Sate Nusantara	Es Teh Manis	8000.00	4.0000000000000000	
1	Sate Nusantara	Mie Ayam Bakso	20000.00	4.0000000000000000	
1	Sate Nusantara	Nasi Goreng Spesial	25000.00	4.0000000000000000	
2	Bakso Legenda	Pasta Carbonara	65000.00	4.0000000000000000	
2	Bakso Legenda	Pizza Margherita	75000.00	4.0000000000000000	
2	Bakso Legenda	Tiramisu	40000.00	4.0000000000000000	
3	Ramen Ichiban	Burger Classic	45000.00	0	
3	Ramen Ichiban	Kentang Goreng	20000.00	0	
3	Ramen Ichiban	Milkshake Coklat	25000.00	0	
4	Ayam Geprek Mantul	Green Tea Ice Cream	30000.00	5.0000000000000000	