

-- Database + Tables Creation

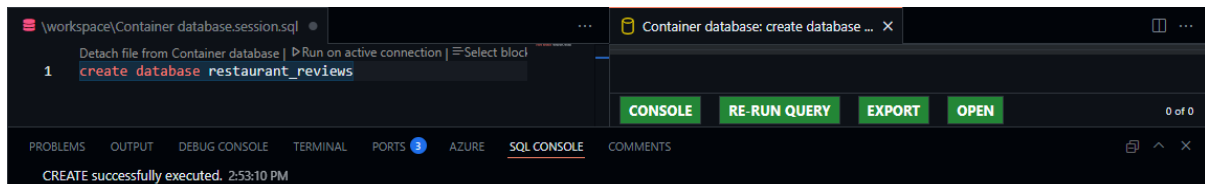
### A. Database Setup

#### 1. Create a new database:

- Name the database **restaurant\_reviews**

create database restaurant\_reviews;

result:



#### 2. Create two tables:

- restaurant table:
  - Columns: **id, name, street\_address, description.**
- review table:
  - Columns: **id, restaurant\_id, user\_name, rating, review\_text, review\_date.**
- Ensure **restaurant\_id** in the review table is a foreign key referencing the restaurant table.

```
create table restaurant (  
    id INT NOT NULL,  
    name VARCHAR(255),  
    street_address VARCHAR(255),  
    description VARCHAR(255),  
    primary key (id)  
);
```

```
create table review (  
    id INT NOT NULL,  
    restaurant_id INT NOT NULL,  
    user_name VARCHAR(255),  
    rating INT,  
    review_text VARCHAR(255),
```

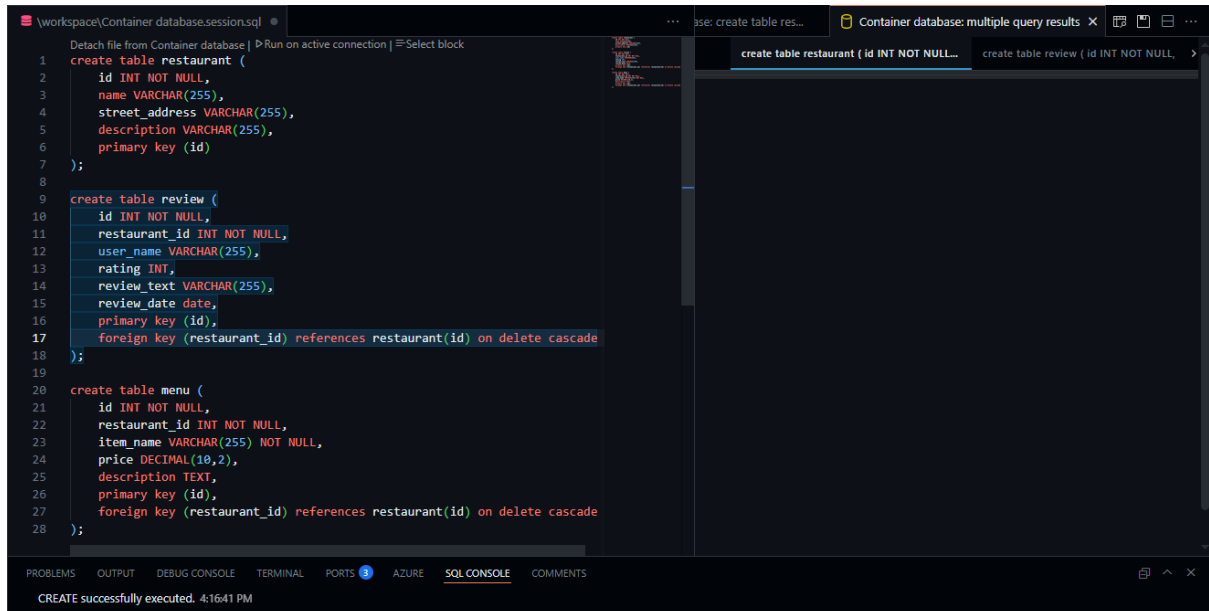
review\_date date,

primary key (id),

foreign key (restaurant\_id) references restaurant(id) on delete cascade

);

Result:



```
1 create table restaurant (  
2   id INT NOT NULL,  
3   name VARCHAR(255),  
4   street_address VARCHAR(255),  
5   description VARCHAR(255),  
6   primary key (id)  
7 );  
8  
9 create table review (  
10  id INT NOT NULL,  
11  restaurant_id INT NOT NULL,  
12  user_name VARCHAR(255),  
13  rating INT,  
14  review_text VARCHAR(255),  
15  review_date date,  
16  primary key (id),  
17  foreign key (restaurant_id) references restaurant(id) on delete cascade  
18 );  
19  
20 create table menu (  
21  id INT NOT NULL,  
22  restaurant_id INT NOT NULL,  
23  item_name VARCHAR(255) NOT NULL,  
24  price DECIMAL(10,2),  
25  description TEXT,  
26  primary key (id),  
27  foreign key (restaurant_id) references restaurant(id) on delete cascade  
28 );
```

CREATE successfully executed. 4:16:41 PM

-- Sample Data Insertion

## B. Inserting Data

Insert sample data into both tables:

1. Insert **at least 3 restaurants** in the restaurant table.
2. Insert **at least 5 reviews** in the review table, ensuring they reference the correct restaurants via **restaurant\_id**.

INSERT INTO restaurant

VALUES

('1','McD','jl. jangan lupa bahagia','nomer 1 ayam'),

('2','KFC','jl. napas manual','nomer 1 kentaki'),

('3','CFC','jl. kedip manual','nomer 1 root beer');

Result:

The screenshot shows a SQL Studio interface with a query editor on the left and a results pane on the right. The query editor contains the following SQL code:

```
1 INSERT INTO restaurant
2 VALUES
3 ('1','McD','jl. jangan lupa bahagia','nomer 1 ayam'),
4 ('2','KFC','jl. napas manual','nomer 1 kentaki'),
5 ('3','CFC','jl. kedip manual','nomer 1 root beer');
6
7 select * from restaurant
```

The results pane displays the output of the second query, showing a table with 4 columns: id, name, street\_address, and description. The data is as follows:

id	name	street_address	description
1	McD	jl. jangan lupa bahagia	nomer 1 ayam
2	KFC	jl. napas manual	nomer 1 kentaki
3	CFC	jl. kedip manual	nomer 1 root beer

INSERT INTO review

VALUES

('1','1','ki joko pinter','4','pembohok','01/01/2001'),

('2','2','jaka sembung','8','wenakk','02/02/2002'),

('3','3','reyna','9','mang eaaakkk','03/03/2003'),

('4','3','sova','3','g enak','04/04/2004'),

('5','3','jilljoy','10','b aja','05/05/2005');

Result:

The screenshot shows a SQL Studio interface with a query editor on the left and a results pane on the right. The query editor contains the following SQL code:

```
1 INSERT INTO review
2 VALUES
3 ('1','1','ki joko pinter','4','pembohok','01/01/2001'),
4 ('2','2','jaka sembung','8','wenakk','02/02/2002'),
5 ('3','3','reyna','9','mang eaaakkk','03/03/2003'),
6 ('4','3','sova','3','g enak','04/04/2004'),
7 ('5','3','jilljoy','10','b aja','05/05/2005');
8
9 SELECT * FROM review
```

The results pane displays the output of the second query, showing a table with 5 columns: id, restaurant\_id, user\_name, rating, and review. The data is as follows:

id	restaurant_id	user_name	rating	review
1	1	ki joko pinter	4	pembo
2	2	jaka sembung	8	wenakl
3	3	reyna	9	mang i
4	3	sova	3	g enak
5	3	jilljoy	10	b aja

-- Queries for CRUD

-- Create

-- Case1

## C. Performing CRUD Operations

Perform the following operations on your database:

### 1. Create (Insert):

- Insert a new restaurant into the restaurant table.

INSERT INTO restaurant

VALUES

('4','A&W','jl. jalan','nomer 1 diseluruh dunia');

Result:

The screenshot shows the SQL Studio interface. The left pane displays the SQL script:

```
1 INSERT INTO restaurant
2 VALUES
3 ('4','A&W','jl. jalan','nomer 1 diseluruh dunia');
4
5 SELECT * FROM restaurant
```

The right pane shows the results of the queries. The first query, "INSERT INTO restaurant VALUES ('4','A&W','jl. jalan','nomer 1 diseluruh dunia');", was executed successfully. The second query, "SELECT \* FROM restaurant", returned the following data:

id	name	street_address	description
1	McD	jl. jangan lupa bahagia	nomer 1 ayam
2	KFC	jl. napas manual	nomer 1 kentaki
3	CFC	jl. kedip manual	nomer 1 root beer
4	A&W	jl. jalan	nomer 1 diseluruh dunia

-- Case2

- Insert a new review for an existing restaurant.

INSERT INTO review

VALUES

('6','4','Agoosh','1','ga enak bjr','06/06/2006');

Result:

The screenshot shows the SQL Studio interface. The left pane displays the SQL script:

```
1 INSERT INTO review
2 VALUES
3 ('6','4','Agoosh','1','ga enak bjr','06/06/2006');
4
5 SELECT * FROM review
```

The right pane shows the results of the queries. The first query, "INSERT INTO review VALUES ('6','4','Agoosh','1','ga enak bjr','06/06/2006');", was executed successfully. The second query, "SELECT \* FROM review", returned the following data:

id	restaurant_id	user_name	rating	review
1	1	ki joko pinter	4	pembo
2	2	jaka sembung	8	wenakl
3	3	reyna	9	mang i
4	4	sova	5	g enak
5	5	jilljoy	10	b aja
6	6	Agoosh	1	ga ena

-- Read

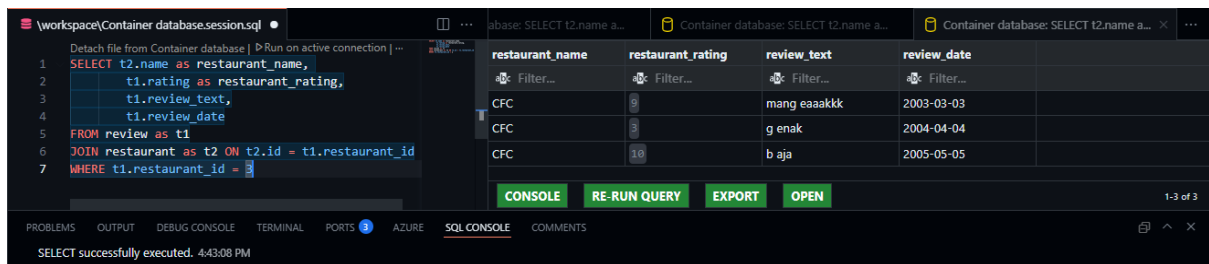
## 2. Read (Select):

- Retrieve all reviews for a specific restaurant using the **restaurant\_id**.
- Retrieve all reviews with a rating of 4 or higher.
- Use a JOIN to display a list of restaurants along with their reviews.

-- Case1

```
SELECT t2.name as restaurant_name,  
  
       t1.rating as restaurant_rating,  
  
       t1.review_text,  
  
       t1.review_date  
  
FROM review as t1  
  
JOIN restaurant as t2 ON t2.id = t1.restaurant_id  
  
WHERE t1.restaurant_id = 3
```

Result:



The screenshot shows a SQL IDE interface. On the left, a query editor displays the following SQL code:

```
1 SELECT t2.name as restaurant_name,  
2       t1.rating as restaurant_rating,  
3       t1.review_text,  
4       t1.review_date  
5 FROM review as t1  
6 JOIN restaurant as t2 ON t2.id = t1.restaurant_id  
7 WHERE t1.restaurant_id = 3
```

On the right, the results pane shows a table with the following data:

restaurant_name	restaurant_rating	review_text	review_date
CFC	9	mang eaaakkk	2003-03-03
CFC	3	g enak	2004-04-04
CFC	10	b aja	2005-05-05

At the bottom of the results pane, there are buttons for 'CONSOLE', 'RE-RUN QUERY', 'EXPORT', and 'OPEN'. The status bar at the bottom indicates 'SELECT successfully executed. 4:43:08 PM'.

-- Case2

```
SELECT t2.name as restaurant_name,  
  
       t1.rating as restaurant_rating,  
  
       t1.review_text,  
  
       t1.review_date  
  
FROM review as t1  
  
JOIN restaurant as t2 ON t2.id = t1.restaurant_id  
  
WHERE t1.rating >= 4
```

Result:

The screenshot shows the VS Code SQL Console with a query window on the left and a results table on the right. The query is a JOIN between 'review' and 'restaurant' tables. The results table has columns: restaurant\_name, restaurant\_rating, review\_text, and review\_date.

```

1 SELECT t2.name as restaurant_name,
2       t1.rating as restaurant_rating,
3       t1.review_text,
4       t1.review_date
5 FROM review as t1
6 JOIN restaurant as t2 ON t2.id = t1.restaurant_id
7 WHERE t1.rating >= 4
  
```

restaurant_name	restaurant_rating	review_text	review_date
McD	4	pembohok	2001-01-01
KFC	8	wenakk	2002-02-02
CFC	10	b aja	2005-05-05
CFC	9	mang eaaakkk	2003-03-03

At the bottom, it says "SELECT successfully executed. 4:45:16 PM".

-- Case3

SELECT \*

FROM review as t1

JOIN restaurant as t2 ON t2.id = t1.restaurant\_id

Result:

The screenshot shows the VS Code SQL Console with a query window on the left and a results table on the right. The query is a simple SELECT \* FROM review JOIN restaurant. The results table has columns: id, res..., user\_name, rating, review\_text, review\_date, id (1), name, street\_address, and description.

```

1 SELECT *
2 FROM review as t1
3 JOIN restaurant as t2
4 ON t2.id = t1.restaurant_id
  
```

id	res...	user_name	rating	review_text	review_date	id (1)	name	street_address	description
1	1	ki joko pinter	4	pembohok	2001-01-01	1	McD	jl. jangan lupa bahagia	nomer 1 ayam
2	2	jaka sembung	8	wenakk	2002-02-02	2	KFC	jl. napas manual	nomer 1 kentaki
3	3	reyna	9	mang eaaakkk	2003-03-03	3	CFC	jl. kedip manual	nomer 1 root beer
4	4	sova	5	g enak	2004-04-04	4	CFC	jl. kedip manual	nomer 1 root beer
5	5	jilljoy	10	b aja	2005-05-05	5	CFC	jl. kedip manual	nomer 1 root beer
6	6	Agoosh	1	ga enak bjir	2006-06-06	6	A&W	jl. jalan	nomer 1 diseluruh d

At the bottom, it says "SELECT successfully executed. 4:47:03 PM".

-- Update

### 3. Update:

- Update the description of one restaurant.
- Update the rating of a specific rating.

-- Case1

update restaurant

SET description = 'nomer 1 aja'

where id = 4

result:

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays a script with the following SQL commands:

```
1 update restaurant
2 SET description = 'nomer 1 aja'
3 where id = 4;
4
5
6 SELECT * FROM restaurant;
```

The right pane shows the results of the query "SELECT \* FROM restaurant". The table has the following data:

id	name	street_address	description
1	McD	jl. jangan lupa bahagia	nomer 1 ayam
2	KFC	jl. napas manual	nomer 1 kentaki
3	CFC	jl. kedip manual	nomer 1 root beer
4	A&W	jl. jalan	nomer 1 aja

The bottom status bar indicates "SELECT successfully executed. 6:25:51 PM".

-- Case2

update review

SET rating = '2'

where id = 4

result:

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays a script with the following SQL commands:

```
1 update review
2 SET rating = '2'
3 where id = 4;
4
5 SELECT * FROM review;
```

The right pane shows the results of the query "SELECT \* FROM review". The table has the following data:

id	restaurant_id	user_name	rating	review
1	1	ki joko pinter	4	pembo
2	2	jaka sembung	8	wenaki
3	3	reyna	9	mang e
5	3	jilljoy	10	b aja
6	4	Agoosh	1	ga ena
4	3	sova	2	g enak

The bottom status bar indicates "SELECT successfully executed. 6:26:38 PM".

-- Delete

#### 4. Delete:

- Delete one review based on id.
- Delete a restaurant and ensure its associated reviews are also deleted (using cascade).

-- Case1

delete from review

where id = 1

Result:

The screenshot shows the SQL Server Enterprise Manager interface. The query window on the left contains the following SQL code:

```
1 delete from review
2 where id = 1;
3
4 SELECT * FROM review;
```

The results window on the right displays the output of the SELECT statement. The table has the following columns: id, restaurant\_id, user\_name, rating, and review. The data is as follows:

id	restaurant_id	user_name	rating	review
2	2	jaka sembung	8	wenaki
3	3	reyna	9	mang e
5	5	jilljoy	10	b aja
6	6	Agoosh	1	ga ena
4	4	sova	2	g enak

-- Case2

delete from restaurant

where id = 1

Result:

The screenshot shows the SQL Server Enterprise Manager interface. The query window on the left contains the following SQL code:

```
1 delete from restaurant
2 where id = 1;
```

The status bar at the bottom indicates: "DELETE successfully executed. 1 rows were affected. 5:00:37 PM".

Before

The screenshot shows the SQL Server Enterprise Manager interface. The query window on the left contains the following SQL code:

```
1 select * from restaurant
```

The results window on the right displays the output of the SELECT statement. The table has the following columns: id, name, street\_address, and description. The data is as follows:

id	name	street_address	description
1	McD	jl. jangan lupa bahagia	nomer 1 ayam
2	KFC	jl. napas manual	nomer 1 kentaki
3	CFC	jl. kedip manual	nomer 1 root beer
4	A&W	jl. jalan	nomer 1 aja



After

The screenshot shows two SQL query results in a dark-themed IDE. The first query, 'select \* from restaurant', returns a table with 4 rows: KFC, CFC, A&W, and another KFC entry. The second query, 'select \* from review', returns a table with 6 rows of reviews for various restaurants.

id	name	street_address	description
1	KFC	jl. napas manual	nomer 1 kentaki
2	CFC	jl. kedip manual	nomer 1 root beer
3	A&W	jl. jalan	nomer 1 aja
4	KFC		

id	restaurant_id	user_name	rating	review_text	review_date
1	1	jaka sembung	8	wenakk	2002-02-02
2	2	reyna	9	mang eaaakkk	2003-03-03
3	3	jilljoy	10	b aja	2005-05-05
4	4	Agoosh	1	ga enak bjir	2006-06-06
5	1	sova	2	g enak	2004-04-04
6	2				

-- Additional Queries

#### D. Additional Queries

1. Find the highest-rated restaurant based on the average rating of all its reviews.
2. Find the number of reviews each restaurant has received.
3. Display the most recent review for each restaurant.

-- Case1

select t1.name as restaurant\_name, avg(t2.rating) as restaurant\_rating

from restaurant as t1

join review as t2 on t1.id = t2.restaurant\_id

GROUP BY t1.id, t1.name

ORDER BY restaurant\_rating DESC

limit 1

Result:

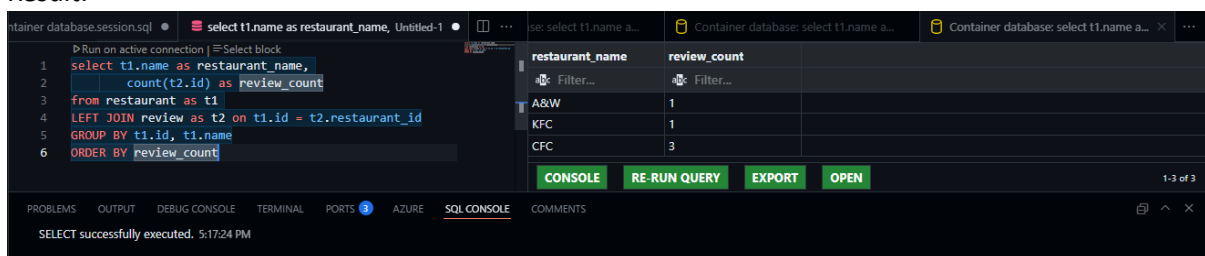
The screenshot shows a SQL query result in a dark-themed IDE. The query is 'select t1.name as restaurant\_name, avg(t2.rating) as restaurant\_rating from restaurant as t1 join review as t2 on t1.id = t2.restaurant\_id GROUP BY t1.id, t1.name ORDER BY restaurant\_rating DESC limit 1'. The result shows KFC with a rating of 8.000000000000000.

restaurant_name	restaurant_rating
KFC	8.000000000000000

-- Case2

```
select t1.name as restaurant_name,  
       count(t2.id) as review_count  
from restaurant as t1  
LEFT JOIN review as t2 on t1.id = t2.restaurant_id  
GROUP BY t1.id, t1.name  
ORDER BY review_count
```

Result:



The screenshot shows a SQL IDE interface. On the left, a query editor displays the following SQL code:

```
1 select t1.name as restaurant_name,  
2       count(t2.id) as review_count  
3 from restaurant as t1  
4 LEFT JOIN review as t2 on t1.id = t2.restaurant_id  
5 GROUP BY t1.id, t1.name  
6 ORDER BY review_count
```

On the right, a results pane shows the output of the query as a table:

restaurant_name	review_count
A&W	1
KFC	1
CFC	3

Below the table are buttons for 'CONSOLE', 'RE-RUN QUERY', 'EXPORT', and 'OPEN'. The status bar at the bottom indicates 'SELECT successfully executed. 5:17:24 PM'.

-- Case3

```
SELECT t1.name as restaurant_name,  
       t2.rating as restaurant_rating,  
       t2.review_text,  
       t2.review_date  
FROM restaurant as t1  
JOIN review as t2 ON t1.id = t2.restaurant_id  
WHERE t2.review_date IN (  
    SELECT max(review_date)  
    FROM review  
    GROUP BY restaurant_id  
)  
ORDER BY t1.name
```

Result:

The screenshot shows a SQL IDE interface. On the left, a query is written in a dark-themed editor. The query is a subquery that selects the maximum review date for each restaurant. On the right, the results of the query are displayed in a table with four columns: restaurant\_name, restaurant\_rating, review\_text, and review\_date. The results show three rows: A&W, CFC, and KFC. At the bottom of the IDE, there is a status bar that says 'SELECT successfully executed, 5:25:47 PM'.

```
1 SELECT t1.name as restaurant_name,
2       t2.rating as restaurant_rating,
3       t2.review_text,
4       t2.review_date
5 FROM restaurant as t1
6 JOIN review as t2 ON t1.id = t2.restaurant_id
7 WHERE t2.review_date IN (
8     SELECT max(review_date)
9     FROM review
10    GROUP BY restaurant_id
11 )
12 ORDER BY t1.name
```

restaurant_name	restaurant_rating	review_text	review_date
A&W	4	ga enak bjir	2006-06-06
CFC	10	b aja	2005-05-05
KFC	8	wenakk	2002-02-02

1-3 of 3

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1 AZURE SQL CONSOLE COMMENTS

SELECT successfully executed, 5:25:47 PM

-- Extra Credits

## Extra Credit (Optional)

1. Create a menu table, similar to the one used in our class session, and insert at least 3 menu items for each restaurant.
2. Write a query to display each restaurant with its menu and the average rating from its reviews.

-- Case1

create table menu (

id INT NOT NULL,

restaurant\_id INT NOT NULL,

item\_name VARCHAR(255) NOT NULL,

price DECIMAL(10,2),

description TEXT,

primary key (id),

foreign key (restaurant\_id) references restaurant(id) on delete cascade

);

INSERT INTO menu

VALUES

('1','2','babi kecap','50000','babi dikecapin'),

('2','2','babi panggang','45000','babi dipanggang'),

('3','2','babi goreng','30000','babi digoreng garing'),

```

('4','3','babi sambal matah','55000','babi dikasi sambel matah'),
('5','3','babi guling','75000','babi diguling'),
('6','3','sate babi','60000','babi disate'),
('7','4','sate kelelawar','20000','kelelawar disate'),
('8','4','rawon babi','25000','babi rawon item'),
('9','4','sop monyet','100000','monyet disop');

```

SELECT \* FROM menu

Result:

The screenshot shows a SQL IDE with a dark theme. On the left, a SQL editor contains an `INSERT INTO menu` statement with 11 rows of data. On the right, a table viewer displays the result of a `SELECT * FROM menu` query. The table has five columns: `id`, `restaurant_id`, `item_name`, `price`, and `description`. The data is as follows:

id	restaurant_id	item_name	price	description
1	2	babi kecap	50000.00	babi dikecapin
2	2	babi panggang	45000.00	babi dipanggang
3	2	babi goreng	30000.00	babi digoreng garing
4	3	babi sambal matah	55000.00	babi dikasi sambel matah
5	3	babi guling	75000.00	babi diguling
6	3	sate babi	60000.00	babi disate
7	4	sate kelelawar	20000.00	kelelawar disate
8	4	rawon babi	25000.00	babi rawon item
9	4	sop monyet	100000.00	monyet disop

-- Case2

```

SELECT t2.name as restaurant_name,
       t3.item_name as restaurant_menu,
       AVG(t1.rating) as restaurant_rating
FROM review as t1
JOIN restaurant as t2 ON t1.restaurant_id = t2.id
JOIN menu as t3 ON t2.id = t3.restaurant_id
GROUP BY t2.id, t2.name, t3.item_name
ORDER BY restaurant_rating DESC

```

Result:

workspace\Container database.session.sql

Detach file from Container database | Run on active connection | Select block

1

2

3

4

5

6

7

8

SELECT t2.name as restaurant\_name,

t3.item\_name as restaurant\_menu,

AVG(t1.rating) as restaurant\_rating

FROM review as t1

JOIN restaurant as t2 ON t1.restaurant\_id = t2.id

JOIN menu as t3 ON t2.id = t3.restaurant\_id

GROUP BY t2.id, t2.name, t3.item\_name

ORDER BY restaurant\_rating DESC

Container database: SELECT t2.name a...

Container database: SELECT t2.name a...

restaurant_name	restaurant_menu	restaurant_rating
KFC	babi kecap	8.0000000000000000
KFC	babi goreng	8.0000000000000000
KFC	babi panggang	8.0000000000000000
CFC	babi sambal matah	7.0000000000000000
CFC	babi guling	7.0000000000000000
CFC	sate babi	7.0000000000000000
A&W	sop monyet	1.0000000000000000
A&W	rawon babi	1.0000000000000000
A&W	sate kelelawar	1.0000000000000000

CONSOLE RE-RUN QUERY EXPORT OPEN

1-9 of 9

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1 AZURE SQL CONSOLE COMMENTS

SELECT successfully executed. 5:42:44 PM