

**Title: DB Assignment 2**

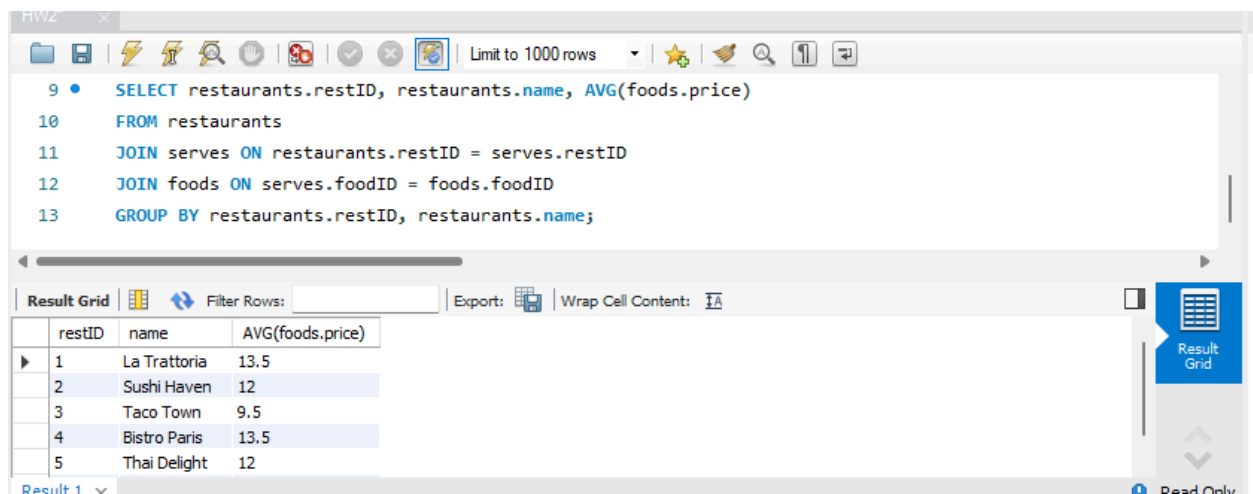
**Name: Aidan Elm**

**Date: 2024-09-26**

## Problem 1

```
SELECT restaurants.restID, restaurants.name, AVG(foods.price)
FROM restaurants
JOIN serves ON restaurants.restID = serves.restID
JOIN foods ON serves.foodID = foods.foodID
GROUP BY restaurants.restID, restaurants.name;
```

In this query, we join serves to restaurants and then foods to serves. Aggregation of the price is done with the AVG function.



The screenshot shows a SQL query editor with the following query:

```
9 • SELECT restaurants.restID, restaurants.name, AVG(foods.price)
10 FROM restaurants
11 JOIN serves ON restaurants.restID = serves.restID
12 JOIN foods ON serves.foodID = foods.foodID
13 GROUP BY restaurants.restID, restaurants.name;
```

Below the query editor, the results are displayed in a grid. The grid has columns for restID, name, and AVG(foods.price). The results are as follows:

restID	name	AVG(foods.price)
1	La Trattoria	13.5
2	Sushi Haven	12
3	Taco Town	9.5
4	Bistro Paris	13.5
5	Thai Delight	12

## Problem 2

```
SELECT restaurants.restID, restaurants.name, MAX(foods.price)
FROM restaurants
JOIN serves ON restaurants.restID = serves.restID
JOIN foods ON serves.foodID = foods.foodID
GROUP BY restaurants.restID, restaurants.name;
```

The same as above, except with MAX.

HW2

```

16 • SELECT restaurants.restID, restaurants.name, MAX(foods.price)
17 FROM restaurants
18 JOIN serves ON restaurants.restID = serves.restID
19 JOIN foods ON serves.foodID = foods.foodID
20 GROUP BY restaurants.restID, restaurants.name;

```

Result Grid

	restID	name	MAX(foods.price)
▶	1	La Trattoria	15
	2	Sushi Haven	14
	3	Taco Town	11
	4	Bistro Paris	18
	5	Thai Delight	13

Result 2 x

Read Only

### Problem 3

```

SELECT restaurants.restID, restaurants.name, COUNT(DISTINCT foods.type)
FROM restaurants
JOIN serves ON restaurants.restID = serves.restID
JOIN foods ON serves.foodID = foods.foodID
GROUP BY restaurants.restID, restaurants.name;

```

The same as above, except with COUNT DISTINCT.

HW2

```

23 • SELECT restaurants.restID, restaurants.name, COUNT(DISTINCT foods.type)
24 FROM restaurants
25 JOIN serves ON restaurants.restID = serves.restID
26 JOIN foods ON serves.foodID = foods.foodID
27 GROUP BY restaurants.restID, restaurants.name;

```

Result Grid

	restID	name	COUNT(DISTINCT foods.type)
▶	1	La Trattoria	1
	2	Sushi Haven	2
	3	Taco Town	1
	4	Bistro Paris	1

Result 4 x

Read Only

### Problem 4

```

SELECT chefs.chefID, chefs.name, AVG(foods.price)
FROM chefs

```

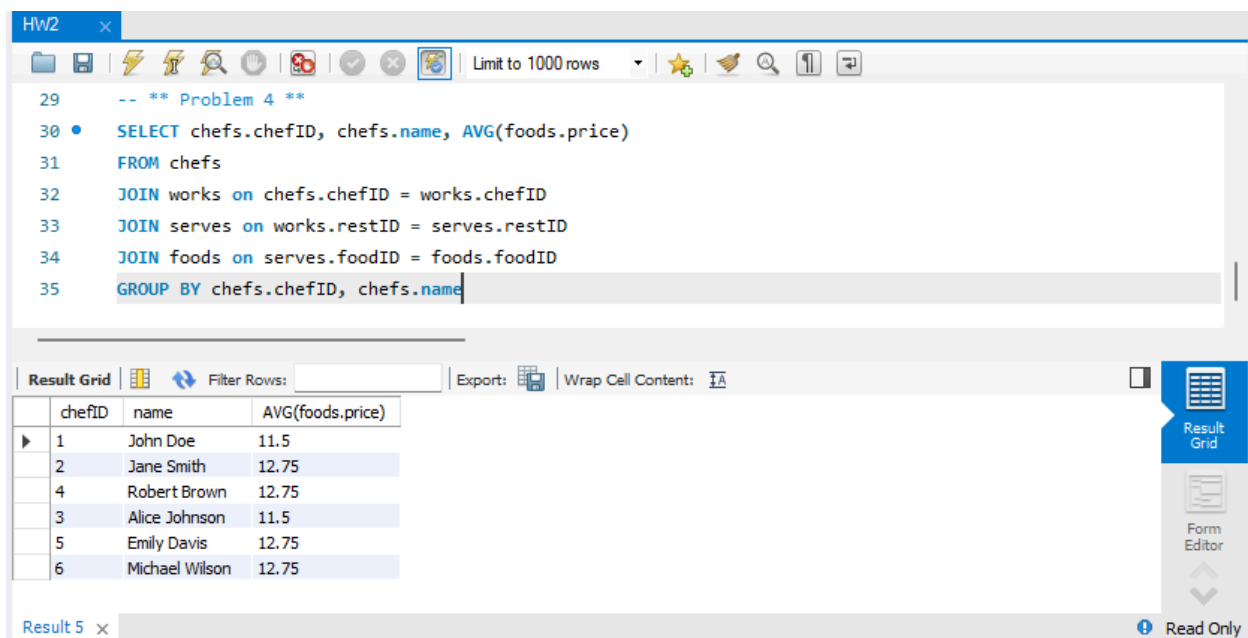
JOIN works on chefs.chefID = works.chefID

JOIN serves on works.restID = serves.restID

JOIN foods on serves.foodID = foods.foodID

GROUP BY chefs.chefID, chefs.name;

In this query, we join chefs -> works -> serves -> foods. From there, we can get the average of the price of foods for each chef.



The screenshot shows a SQL IDE window titled 'Hw2'. The query editor contains the following SQL code:

```
-- ** Problem 4 **
29
30 • SELECT chefs.chefID, chefs.name, AVG(foods.price)
31 FROM chefs
32 JOIN works on chefs.chefID = works.chefID
33 JOIN serves on works.restID = serves.restID
34 JOIN foods on serves.foodID = foods.foodID
35 GROUP BY chefs.chefID, chefs.name
```

Below the query editor, the 'Result Grid' tab is active, displaying the following data:

	chefID	name	AVG(foods.price)
▶	1	John Doe	11.5
	2	Jane Smith	12.75
	4	Robert Brown	12.75
	3	Alice Johnson	11.5
	5	Emily Davis	12.75
	6	Michael Wilson	12.75

The IDE interface includes a toolbar with various icons, a 'Limit to 1000 rows' dropdown, and a 'Read Only' status indicator at the bottom right.

### Problem 5

SELECT restaurants.name, AVG(foods.price)

FROM restaurants

JOIN serves ON restaurants.restID = serves.restID

JOIN foods ON serves.foodID = foods.foodID

GROUP BY restaurants.name

ORDER BY AVG(foods.price) DESC

LIMIT 1;

Here, we are joining restaurants -> serves -> foods and getting the average of foods price. To only get the first, we order by descending and limit to 1.

