

## DB Assignment 2

**Name:** Sean Gor

**Date:** 9.26.24

1.)

```
2  -- query 1: finds average price at each restaurant
3  • SELECT r.name AS restaurant_name, avg(price) AS avg_price
4  FROM restaurants r
5  JOIN serves s ON r.restID = s.restID -- links the serves table
6  JOIN foods f ON s.foodID = f.foodID -- links the foods table
7  GROUP BY r.name;
```

	restaurant_name	avg_price
▶	La Trattoria	13.5000
	Sushi Haven	12.0000
	Taco Town	9.5000
	Bistro Paris	13.5000
	Thai Delight	12.0000
	Indian Spice	13.5000


This query extracts all restaurant names and average prices from the database. It then uses JOIN statements to group the stats together (restaurant ID's from the restaurant table, food ID's from the food table).

2.)

```

9      -- query 2: finds maximum price of foods at each restaurant
10
11  •   SELECT r.name AS restaurant_name, MAX(price) AS max_food_price -- using aggregate function MAX
12      FROM restaurants r
13      JOIN serves s ON r.restID = s.restID -- linking serves table
14      JOIN foods f ON s.foodID = f.foodID -- linking foods table
15      GROUP BY r.name;

```

Result Grid |  Filter Rows:  | Export:  | Wrap Cell Content: 

	restaurant_name	max_food_price
▶	La Trattoria	15
	Sushi Haven	14
	Taco Town	11
	Bistro Paris	18
	Thai Delight	13
	Indian Spice	15

This query extracts all restaurant names and maximum prices (found by using the MAX function) from the database. Like query 1, it uses join statements in order to obtain information from the foods table (via the serves table). This helps get the maximum price.

3.)

```

21      -- query 3: finds the count of each food type at each restaurant.
22
23  •   SELECT r.name as restaurant_name, count(distinct f.type) as num_of_food_types
24      from restaurants r
25      join serves s ON r.restID = s.restID -- linking the serves table
26      join foods f ON s.foodID = f.foodID -- linking the food table
27      GROUP BY r.name; -- organizing data by the restaurant's name
28

```

Result Grid |  Filter Rows:  | Export:  | Wrap Cell Content: 

	restaurant_name	num_of_food_types
▶	Bistro Paris	1
	Indian Spice	1
	La Trattoria	1
	Sushi Haven	2
	Taco Town	1
	Thai Delight	1

This query uses join statements again to get information from the foods table; however, It also uses the count aggregate function to obtain the frequencies of each food type and the distinct keyword to ensure that there are no duplicates.

4.)

```
29      -- query 4: finds average food price for each chef
30
31  ●    SELECT c.name AS chef_name, avg(price) AS avg_food_price
32      FROM chefs c
33      JOIN works w ON c.chefID = w.chefID -- linking the works table to get chef ID
34      JOIN serves s ON w.restID = s.restID -- linking the serves table to get restaurant ID
35      JOIN foods f ON s.foodID = f.foodID -- linking the food table
36      GROUP BY c.name;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
chef_name	avg_food_price			
John Doe	11.5000			
Jane Smith	12.7500			
Robert Brown	12.7500			
Alice Johnson	11.5000			
Emily Davis	12.7500			
Michael Wilson	12.7500			

This query extracts the data headings from the appropriate tables, calculates the average price through the aggregate function, and uses several join statements to obtain the information about food through the works and serves tables.

5.)

```

31 -- query 5: finds restaurants with highest average food price
32 • SELECT r.name AS restaurant_name, avg(f.price) AS avg_price -- aggregate function max being used
33 FROM restaurants r
34 JOIN serves s ON r.restID = s.restID -- serves table is linked
35 JOIN foods f ON s.foodID = f.foodID -- foods table is linked
36 GROUP BY r.name
37 HAVING AVG(f.price) = ( -- Using having statement to include only restaurants with the highest price
38     SELECT MAX(sub.avg_price) -- subquery to find the maximum average price
39     FROM (
40         SELECT r.restID, AVG(f.price) AS avg_price
41         FROM restaurants r
42         JOIN serves s ON r.restID = s.restID
43         JOIN foods f ON s.foodID = f.foodID
44         GROUP BY r.restID
45     ) AS sub
46 );

```

Result Grid		
Filter Rows:		
Export: Wrap Cell Content:		
	restaurant_name	avg_price
▶	La Trattoria	13.5000
	Bistro Paris	13.5000
	Indian Spice	13.5000

This query calculates the average values of the food prices (similar to problem 1). It then uses a subquery to calculate the maximum food price from all the restaurants, and then checks if the price for each restaurant is equal to that maximum value (with a Having clause). If so, the restaurant name is included in the table.

### Extra Credit:

```

48 -- Extra Credit: Determine which chef has the highest average price of the foods served at the restaurants where they work. Include the chef's name,
49 -- the average food price, and the names of the restaurants where the chef works.
50 -- Sort the results by the average food price in descending order.
51 • SELECT
52     c.name AS chef_name,
53     GROUP_CONCAT(DISTINCT r.name) AS rest_name, -- grouping all restaurant names associated with the chef in one cell
54     AVG(price) AS avg_food_price
55 FROM
56     chefs c
57 JOIN
58     works w ON c.chefID = w.chefID -- linking works table
59 JOIN
60     restaurants r ON w.restID = r.restID -- linking restaurants table
61 JOIN
62     serves s ON r.restID = s.restID -- linking serves table
63 JOIN
64     foods f ON s.foodID = f.foodID -- linking foods table
65 GROUP BY
66     c.name

```

Result Grid		
Filter Rows:		
Export: Wrap Cell Content:		
	chef_name	rest_name avg_food_price
▶	Emily Davis	Indian Spice,Thai Delight 12.7500
	Jane Smith	La Trattoria,Sushi Haven 12.7500
	Michael Wilson	Indian Spice,Thai Delight 12.7500
	Robert Brown	Bistro Paris,Sushi Haven 12.7500
	Alice Johnson	Bistro Paris,Taco Town 11.5000
	John Doe	La Trattoria,Taco Town 11.5000

This query extracts each chef's name, stores the average price in a variable called `avg_food_price`, and groups all distinct restaurant names together using the `Group_concat()` function. It uses join statements to receive necessary information from the restaurant and food tables. Finally, it sorts the results from the highest to lowest price with the `Order by` function.