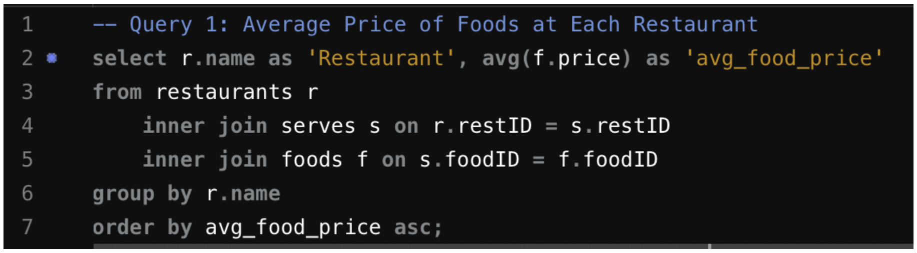
DB Assignment 2
Brittany Klose
9/24/24

Part II: SQL queries and analysis

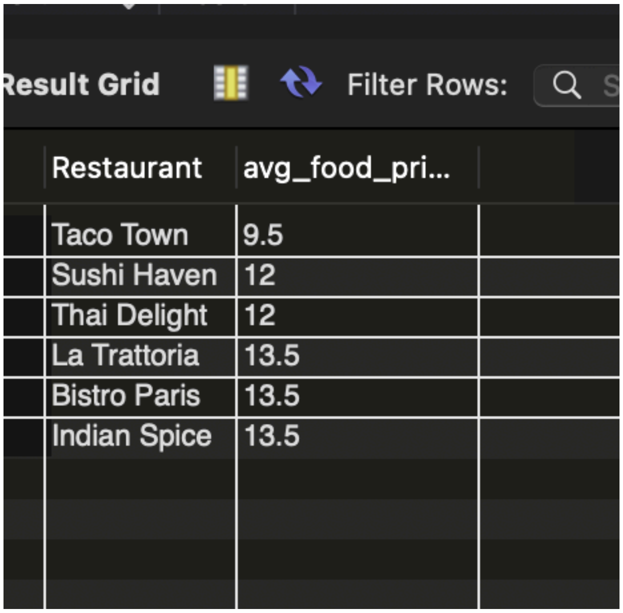
1. **Average Price of Foods at Each Restaurant**
   - For this query I used 2 inner joins to merge columns from tables: serves, foods, and restaurants; as well as utilizing the average math command to find the average price of food at each restaurant. Two inner joins were used between the relationship serves and entities foods and restaurants. This allowed the restaurant table to gain access to food prices. The output displays the restaurants by their name and the average price of food they serve. I used an ascending order for average food price, as typically people want to see the lowest prices first.

   - Query Code :



```sql
-- Query 1: Average Price of Foods at Each Restaurant
select r.name as 'Restaurant', avg(f.price) as 'avg_food_price'
from restaurants r
    inner join serves s on r.restID = s.restID
    inner join foods f on s.foodID = f.foodID
group by r.name
order by avg_food_price asc;
```

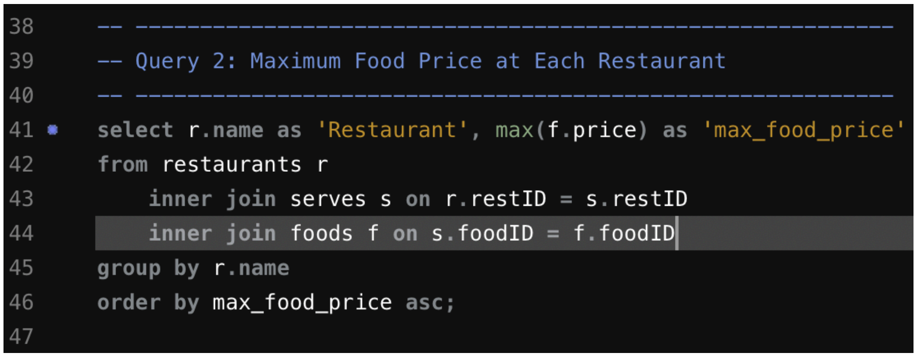
   - Resulted Output::



Restaurant	avg_food_pri...
Taco Town	9.5
Sushi Haven	12
Thai Delight	12
La Trattoria	13.5
Bistro Paris	13.5
Indian Spice	13.5

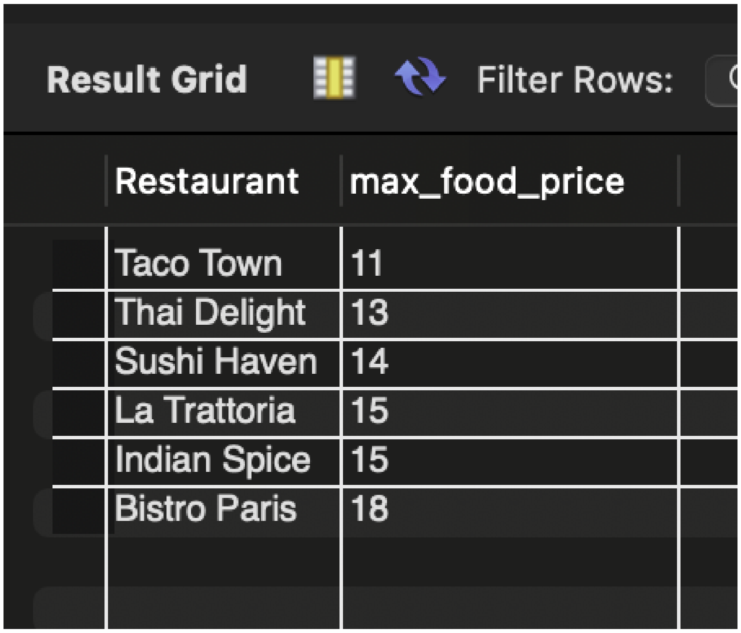
2. **Maximum Food Price at Each Restaurant**
   - This query was very similar to the previous, if not identical. It required the same information as the previous so I used the same inner joins again on the same tables. The only change here was I switched the *avg* command out for *max.* As a result each restaurant is printed with the most expensive food item they serve.

   - Query Code:



```
-- -------------------------------------------------------------
-- Query 2: Maximum Food Price at Each Restaurant
-- -------------------------------------------------------------
select r.name as 'Restaurant', max(f.price) as 'max_food_price'
from restaurants r
    inner join serves s on r.restID = s.restID
    inner join foods f on s.foodID = f.foodID
group by r.name
order by max_food_price asc;
```

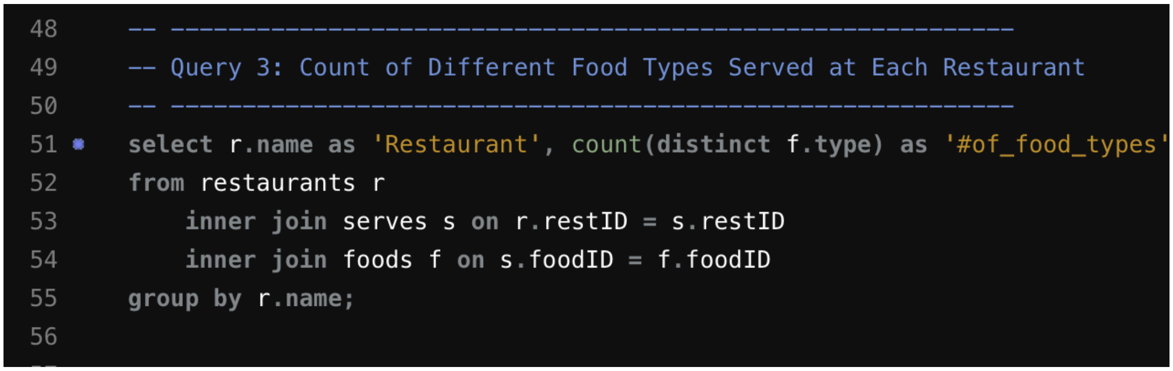
   - Resulted Output:



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

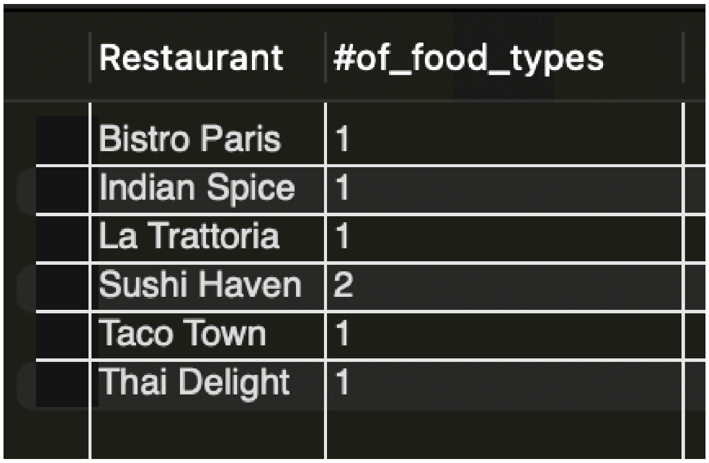
3. **Count of Different Food Types Served at Each Restaurant**
   - This query required two joins as well with the same previous tables: serves, foods, restaurants. A key word I needed to use here was ‘distinct’ when counting the types of food. To explain, each restaurant offers two meal options, all of the same food type, except Sushi Haven. Thus, the first time I ran the query without ‘distinct’ the result was 2 for every restaurant. Once I used distinct it correctly printed out the number of food types; which was 2 for Sushi Haven and 1 for the rest.

   - Query Code:



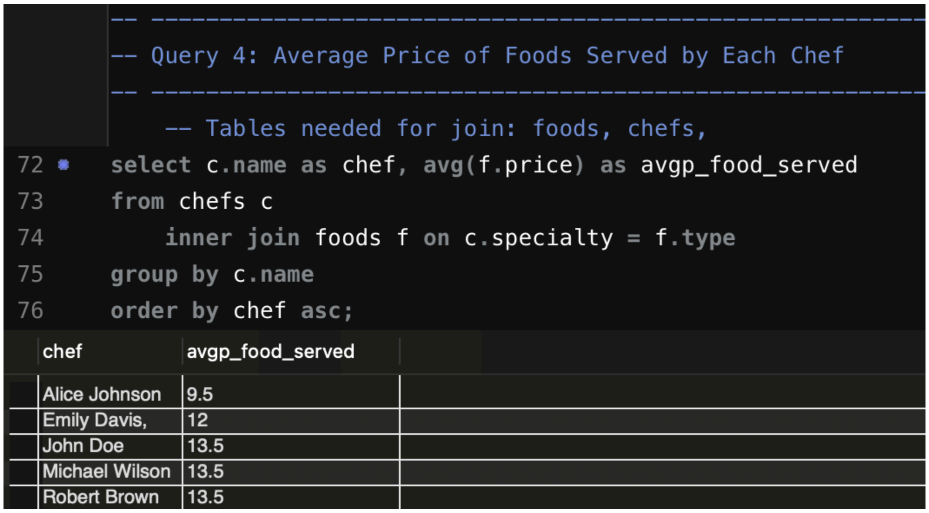
```
-- ------------------------------------------------------------
-- Query 3: Count of Different Food Types Served at Each Restaurant
-- ------------------------------------------------------------
select r.name as 'Restaurant', count(distinct f.type) as '#of_food_types'
from restaurants r
    inner join serves s on r.restID = s.restID
    inner join foods f on s.foodID = f.foodID
group by r.name;
```

   - Resulted Output:



Restaurant	#of_food_types
Bistro Paris	1
Indian Spice	1
La Trattoria	1
Sushi Haven	2
Taco Town	1
Thai Delight	1

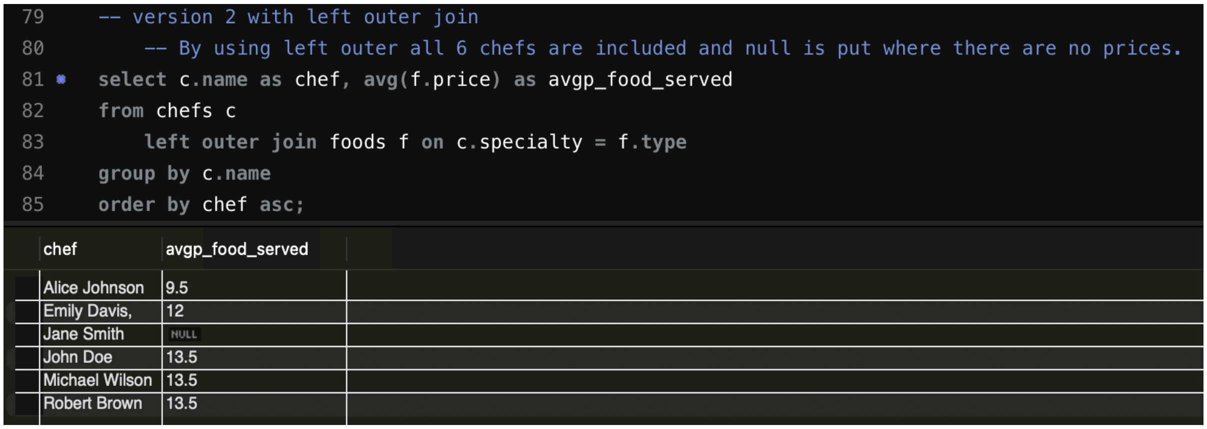
4. **Average Price of Foods Served by Each Chef**
    - To solve this problem my query was very similar to that of problem 1 but instead of inner joins for restaurants, serves, and foods I had one inner join between chefs and foods as they have a direct commonality to join them. This being food type and chef speciality. By joining them I was able to make an output table showing the chefs by name and the average price of meals they make. Notice there are only 5 chefs listed because one chef, Jane Smith specializes in Japanese cuisine and no restaurant in the schema serves a food of that cuisine so there are no prices. I later decided to add another version and use a left outer join so all chefs are represented and have null for avg price where there is no associated value as seen below.
    - Version 1: Inner Join



```
-- -------------------------------------------------------------
-- Query 4: Average Price of Foods Served by Each Chef
-- -------------------------------------------------------------
    -- Tables needed for join: foods, chefs,
select c.name as chef, avg(f.price) as avgp_food_served
from chefs c
    inner join foods f on c.specialty = f.type
group by c.name
order by chef asc;
```

chef	avgp_food_served
Alice Johnson	9.5
Emily Davis,	12
John Doe	13.5
Michael Wilson	13.5
Robert Brown	13.5

    - Version 2: Left Outer Join

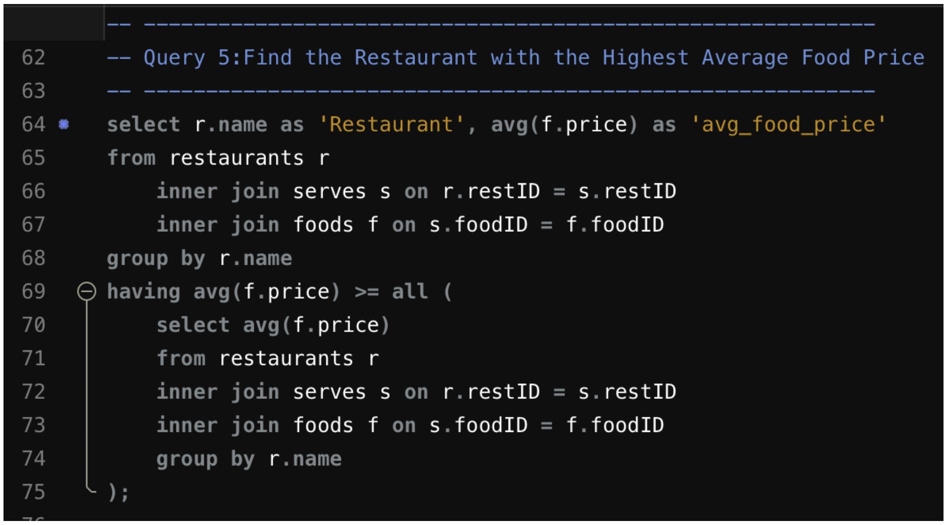


```
-- version 2 with left outer join
    -- By using left outer all 6 chefs are included and null is put where there are no prices.
select c.name as chef, avg(f.price) as avgp_food_served
from chefs c
    left outer join foods f on c.specialty = f.type
group by c.name
order by chef asc;
```

chef	avgp_food_served
Alice Johnson	9.5
Emily Davis,	12
Jane Smith	NULL
John Doe	13.5
Michael Wilson	13.5
Robert Brown	13.5

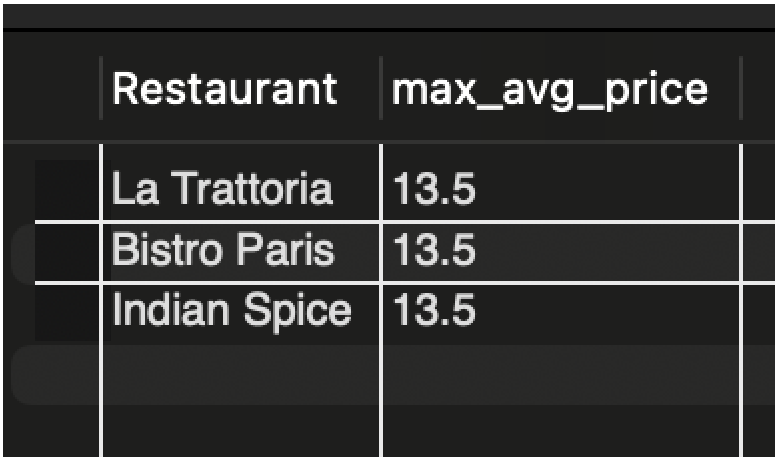
5. **Find the Restaurant with the Highest Average Food Price**
    - To solve this problem I reused my code from query 1 that calculated the average price of food for each restaurant. Then I used the 'having' command to filter the results so it found the maximum and excluded all other averages. The 'having' keyword was especially needed here over say using 'limit 1' and using less code because the maximum average price was shared by 3 restaurants. This could have also been done by using 'as' in addition to 'having' as a subquery. I tried it as well and it gave me the same output so it did not have an affect on my results. As seen on the github code file.

    - Query Code:



```
-- -----------------------------------------------------------
-- Query 5:Find the Restaurant with the Highest Average Food Price
-- -----------------------------------------------------------
select r.name as 'Restaurant', avg(f.price) as 'avg_food_price'
from restaurants r
    inner join serves s on r.restID = s.restID
    inner join foods f on s.foodID = f.foodID
group by r.name
having avg(f.price) >= all (
    select avg(f.price)
    from restaurants r
    inner join serves s on r.restID = s.restID
    inner join foods f on s.foodID = f.foodID
    group by r.name
);
```

    - Resulted Output:



Restaurant	max_avg_price
La Trattoria	13.5
Bistro Paris	13.5
Indian Spice	13.5

6. **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, the average food price, and the names of the restaurants where the chef works. Sort the results by the average food price in descending order.**
   - I ran out of time to attempt this problem properly but I began approaching it by building a table to see how I could connect all chefs to their restaurants, the types of food and meal names served at these restaurants and their associated prices. I had some trouble getting the right restaurant names in the output for all of them, so I would have to go back and look at the connections again. Once I understood the connection I would then use the avg price function I had in the other queries to hopefully get the right output.