DB Assignment 2

Giannina Flamiano

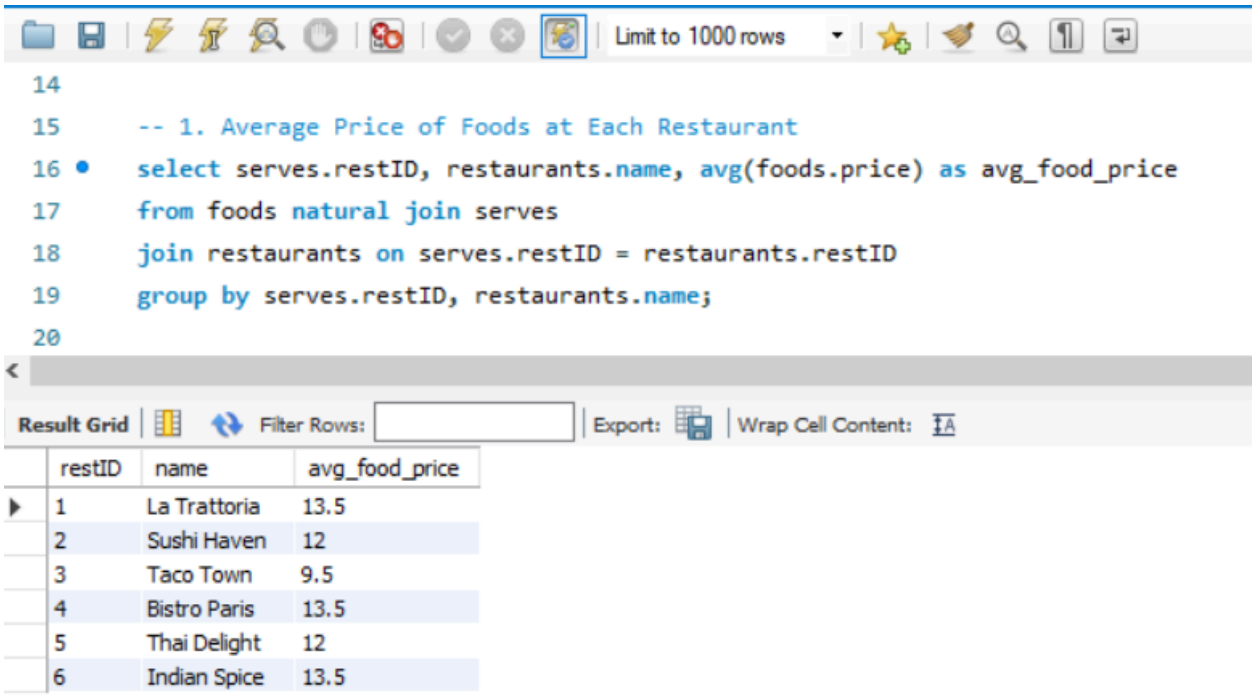
September 26, 2024

1. Average Price of Foods at Each Restaurant

SQL QUERY

```
select serves.restID, restaurants.name, avg(foods.price) as avg_food_price
from foods natural join serves
join restaurants on serves.restID = restaurants.restID
group by serves.restID, restaurants.name;
```

SCREENSHOT



restID	name	avg_food_price
1	La Trattoria	13.5
2	Sushi Haven	12
3	Taco Town	9.5
4	Bistro Paris	13.5
5	Thai Delight	12
6	Indian Spice	13.5

EXPLANATION

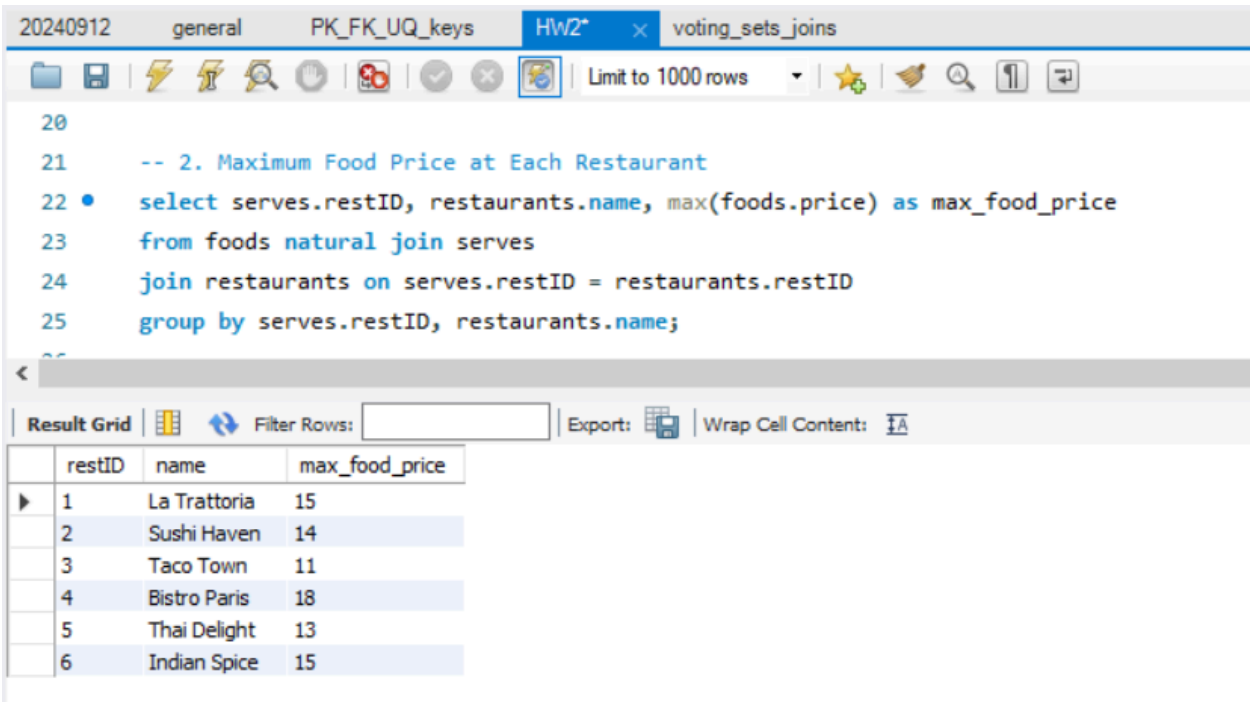
This query looks at foods, serves, and restaurants tables. It combines data from foods and serves using a natural join since it shares a common field, foodID. Then, it joins the restaurants table to get the restaurant details based on matching restIDs. In the results table, the query will select the restID from the serves table, the restaurant's name from the restaurants table, and the average price of foods using an aggregate function. The results are grouped by restID and the restaurant's name so that we get one row for each restaurant showing the average price of its food.

2. Maximum Food Price at Each Restaurant

SQL QUERY

```
select serves.restID, restaurants.name, max(foods.price) as max_food_price
from foods natural join serves
join restaurants on serves.restID = restaurants.restID
group by serves.restID, restaurants.name;
```

SCREENSHOT



restID	name	max_food_price
1	La Trattoria	15
2	Sushi Haven	14
3	Taco Town	11
4	Bistro Paris	18
5	Thai Delight	13
6	Indian Spice	15

EXPLANATION

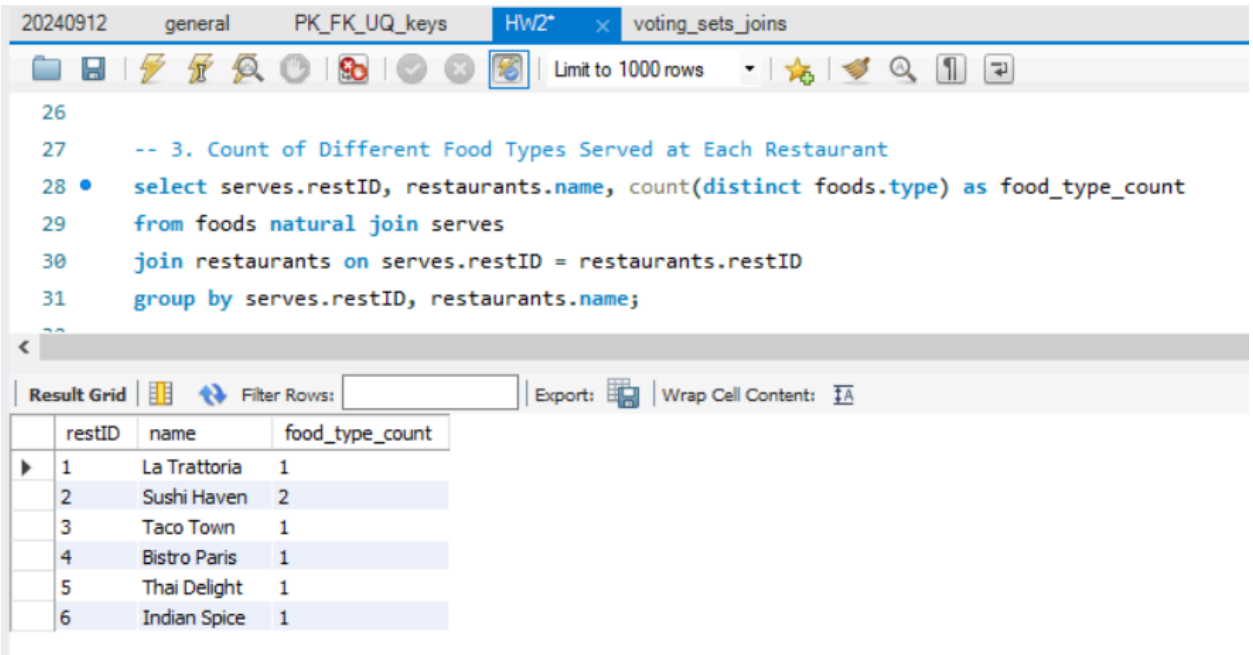
This query looks at foods, serves, and restaurants tables. It combines data from foods and serves using a natural join since it shares a common field, foodID. Then, it joins the restaurants table to get the restaurant details based on matching restIDs. In the results table, the query will select the restID from the serves table, the restaurant name from the restaurants table, and the maximum food price using an aggregate function. The results are grouped by restID and the restaurant's name so that we get one row for each restaurant showing the maximum food price.

3. Count of Different Food Types Served at Each Restaurant

SQL QUERY

```
select serves.restID, restaurants.name, count(distinct foods.type) as food_type_count
from foods natural join serves
join restaurants on serves.restID = restaurants.restID
group by serves.restID, restaurants.name;
```

SCREENSHOT



restID	name	food_type_count
1	La Trattoria	1
2	Sushi Haven	2
3	Taco Town	1
4	Bistro Paris	1
5	Thai Delight	1
6	Indian Spice	1

EXPLANATION

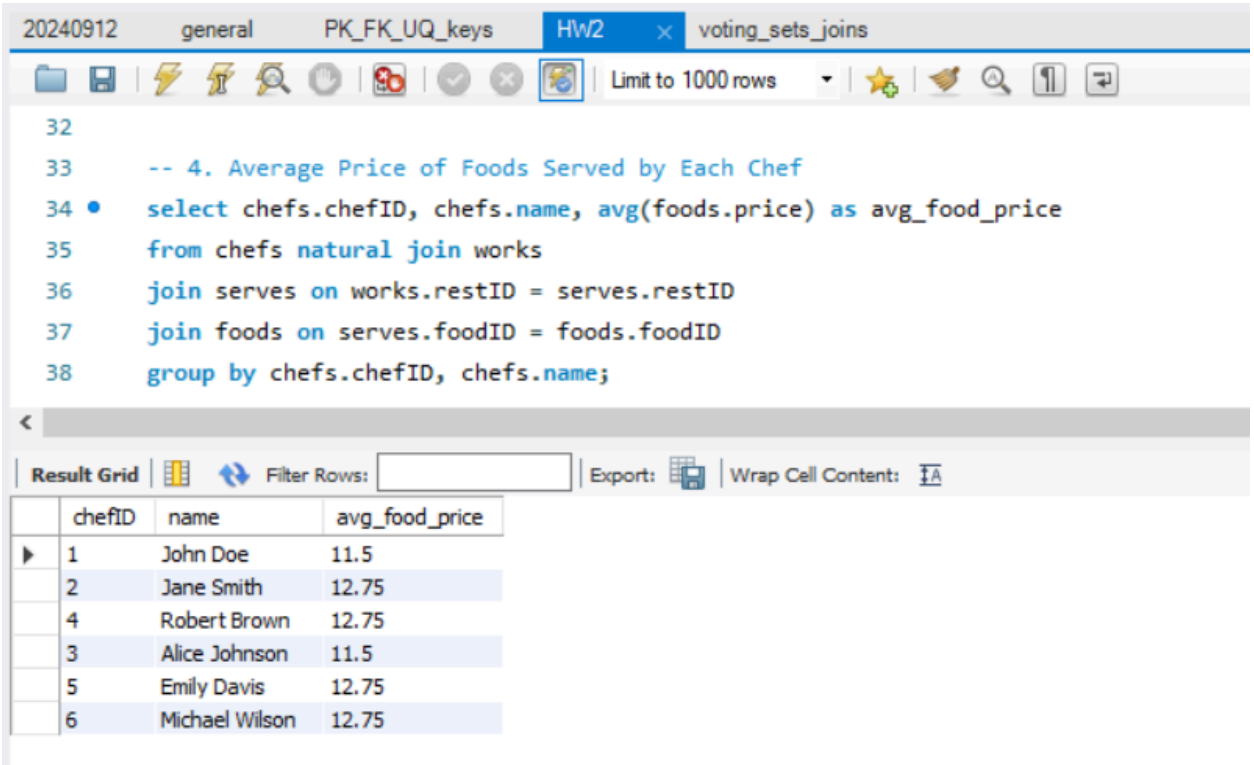
This query looks at foods, serves, and restaurants tables. It combines data from foods and serves using a natural join since it shares a common field, foodID. Then, it joins the restaurants table to get the restaurant details based on matching restIDs. In the results table, the query will select the restID from the serves table, the restaurant name from the restaurants table, and the number (aka count) of distinct food types using an aggregate function. We use distinct so that a single food type will only get counted once. The results are then grouped by restID and the restaurant's name so that we get one row for each restaurant showing the count of different foods types served.

4. Average Price of Foods Served by Each Chef

SQL QUERY

```
select chefs.chefID, chefs.name, avg(foods.price) as avg_food_price
from chefs natural join works
join serves on works.restID = serves.restID
join foods on serves.foodID = foods.foodID
group by chefs.chefID, chefs.name;
```

SCREENSHOT



chefID	name	avg_food_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

EXPLANATION

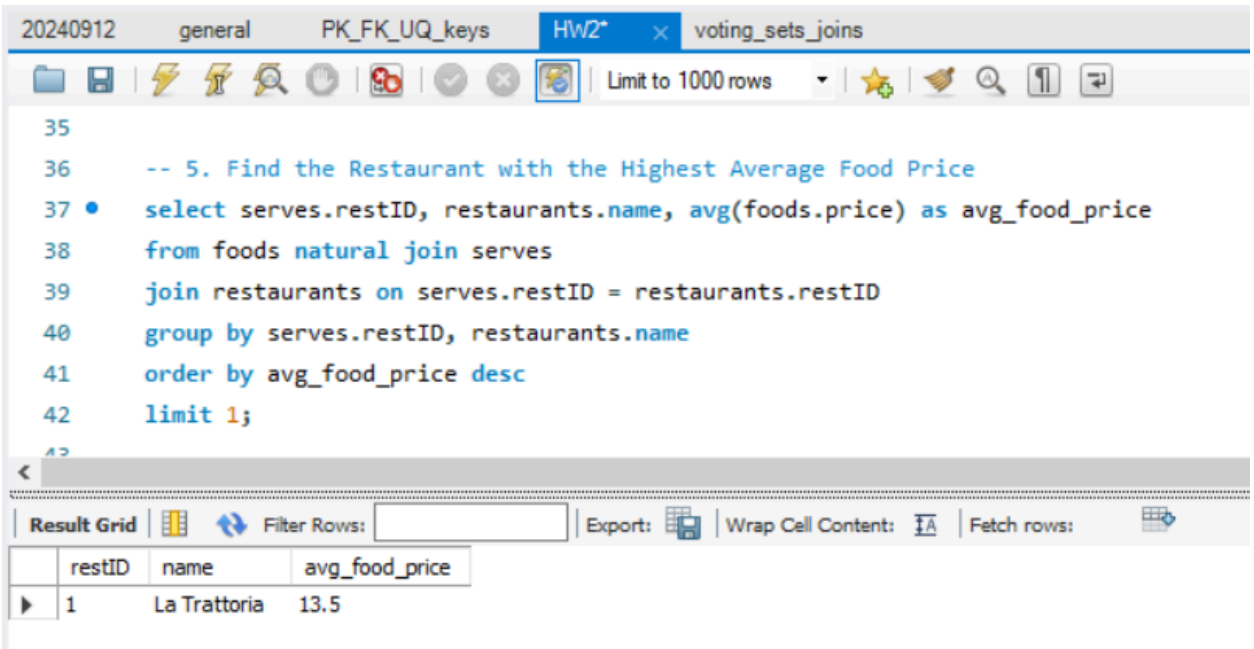
This query looks at chefs, works, serves, and foods tables. It combines data from chefs and works using a natural join since it shares a common field, chefID. Then, it joins the serves table to get the serves details based on matching restIDs. Next, it joins the foods table to get the foods details based on matching foodIDs. In the results table, the query will select the chefID and chef's name from the chefs table and the average food price using an aggregate function. The results are grouped by chefsID and chef's name so that we get one row for each chef showing their respective average price of food.

5. Find the Restaurant with the Highest Average Food Price

SQL QUERY

```
select serves.restID, restaurants.name, avg(foods.price) as avg_food_price
from foods natural join serves
join restaurants on serves.restID = restaurants.restID
group by serves.restID, restaurants.name
order by avg_food_price desc
limit 1;
```

SCREENSHOT



EXPLANATION

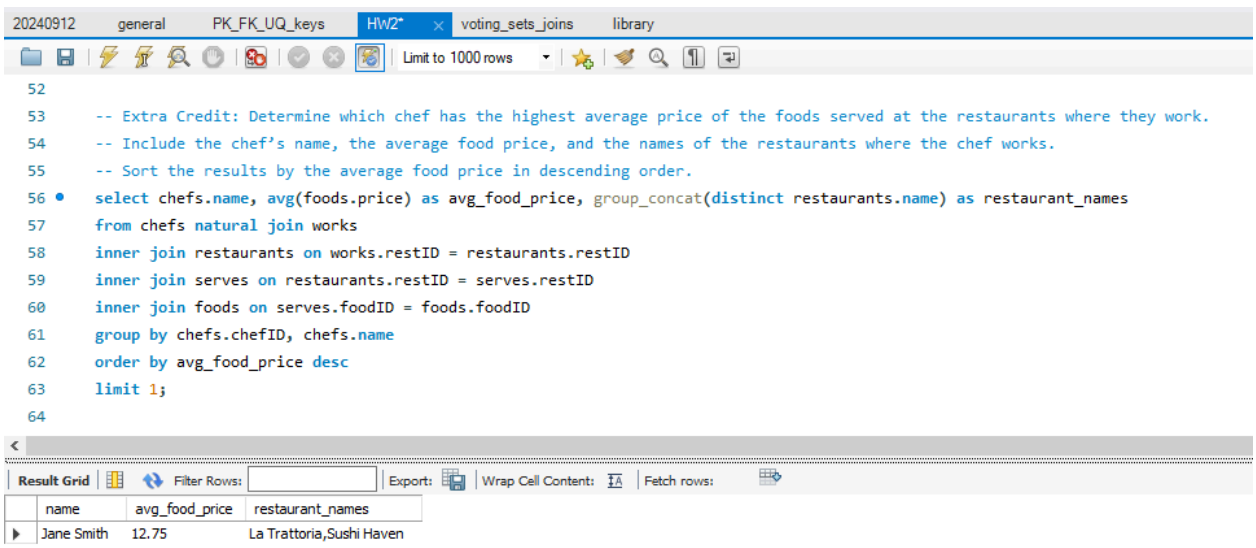
This query looks at foods, serves, and restaurants tables. It combines data from foods and serves using a natural join since it shares a common field, foodID. Then, it joins the restaurants table to get the restaurant details based on matching restIDs. In the results table, the query will select the restID from the serves table, the restaurant's name from the restaurants table, and the average food price using an aggregate function. The results are then grouped by restID and the restaurant's name and displayed in descending order by average food price. We limit the results table by 1 so that we get the top row to get the restaurant with the highest average food price.

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.

SQL QUERY

```
select chefs.name, avg(foods.price) as avg_food_price, group_concat(distinct
restaurants.name) as restaurant_names
from chefs natural join works
inner join restaurants on works.restID = restaurants.restID
inner join serves on restaurants.restID = serves.restID
inner join foods on serves.foodID = foods.foodID
group by chefs.chefID, chefs.name
order by avg_food_price desc
limit 1;
```

SCREENSHOT



EXPLANATION

This query looks at chefs, works, restaurants, serves and foods tables. It combines data from chefs and works using a natural join since they share a common field, chefID. Then it joins the restaurants table to get the restaurant details based on matching restIDs. Then it joins the serves table to get the serves details based on matching restIDs. Then it joins the foods table to get the foods details based on matching foodIDs. In the results table, the query selects the chef's name from the chefs table, the average food price using an aggregate function and the distinct restaurant names using a group concatenation, which combines data from multiple rows into one column. The results are then grouped by chefID and chef's name then displayed in descending order by average food price. We limit the results table by 1 so that we get the top row to get the chef that has the highest average price of the foods served at the restaurants where they work.