Devon Reing

Dr. Forouraghi

Database Management Systems

26 September 2024

DB Assignment 2

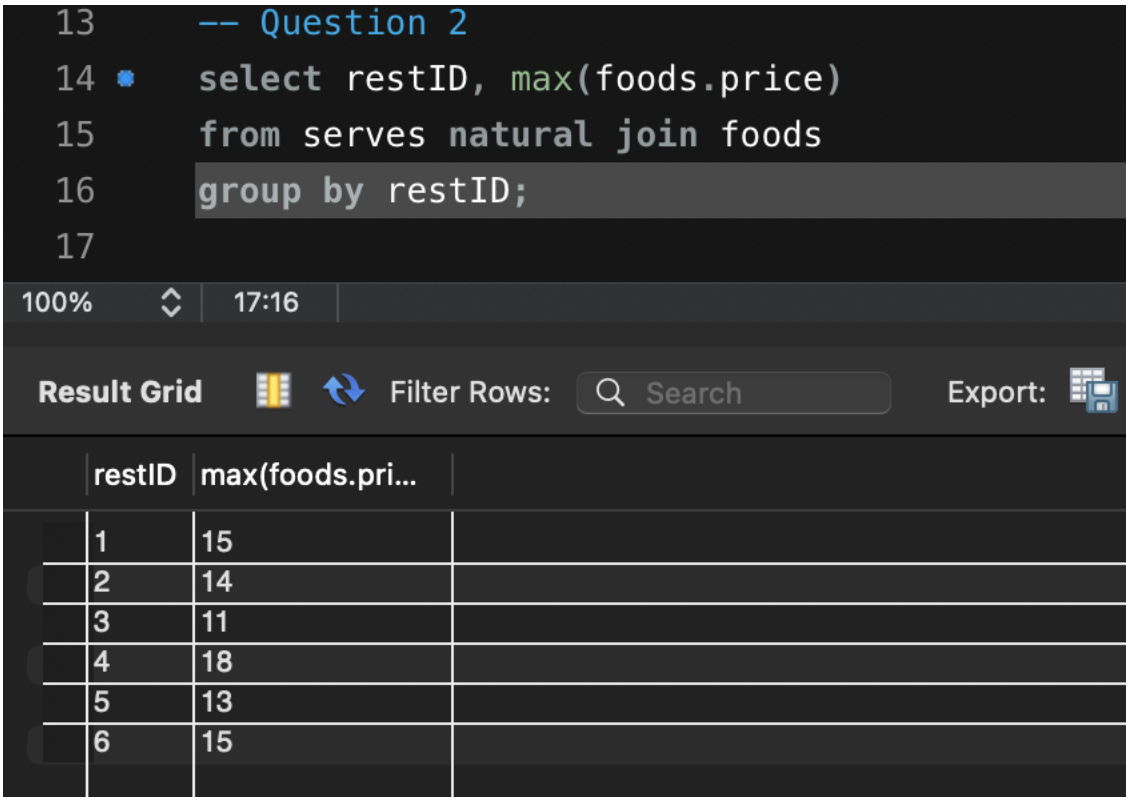
```
7    -- Question 1
8    select restID, avg(foods.price) as foodprice
9    from serves natural join foods
10   group by restID
11   order by foodprice;
12
```

100% 20:11

Result Grid | Filter Rows: Search | Export:

restID	foodprice
3	9.5
2	12
5	12
1	13.5
4	13.5
6	13.5

The first query aims to find the average food price at each restaurant. This query works by doing a natural join on the serves and foods tables, which share only a foodID attribute/column. This allows for the natural join to be easily done instead of an inner join where the column to join the tables on would have to be specified. After these tables are joined together, the restaurant ID primary key and price attribute from the foods table are available to be used in combination. For each restaurant, the average food price is then calculated for each restaurant, which is specified by the ‘group by’ statement. Finally, once this data is found, it is formatted in ascending order according to the average food price for each restaurant.



The second query aims to find the maximum food price at each restaurant. The same as above, this query works by doing a natural join on the serves and foods tables, which share only a foodID attribute/column. The same reason applies for why a natural join is possible instead of the more specific inner join. After these tables are joined together, the restaurant ID primary key and price attribute from the foods table are available to be used in combination again. For each restaurant, the maximum food price is then found for each restaurant, which is specified by the 'group by' statement.

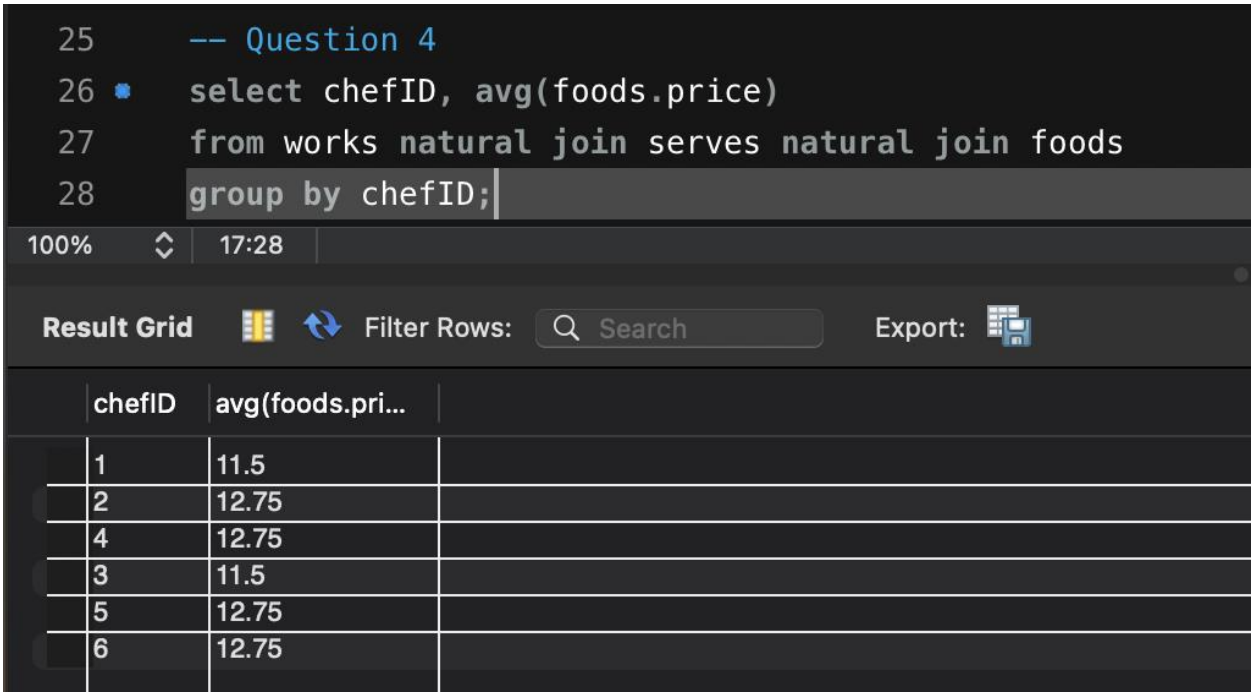
```
18      -- Question 3
19      select restID, count(DISTINCT foods.type)
20      from serves inner join foods
21      on foods.foodID = serves.foodID
22      group by restID;
23
```

100% 17:22

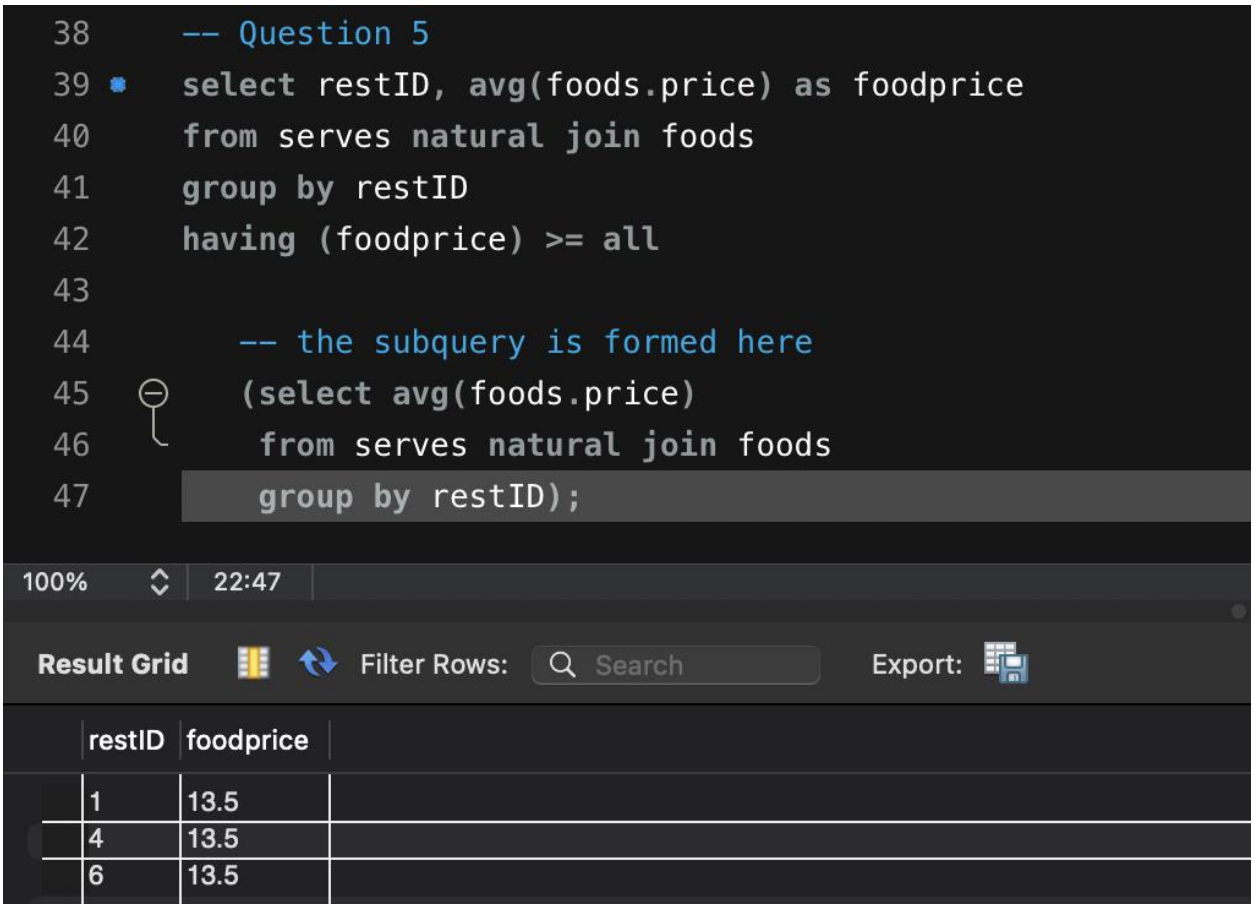
Result Grid | Filter Rows: Search | Export:

restID	count(DISTINCT foods.ty...
1	1
2	2
3	1
4	1
5	1
6	1

The third query aims to find the number of food types served at each restaurant. This query uses an inner join instead of the natural join used before in order to combine the foods and serves table. While it is not necessary to do an inner join on these tables as explained before because of the only shared attribute being foodID, this is another option for performing the join in a more specific manner. After the join is performed, the restaurant ID and food type served at each restaurant are available in the same place. The ‘select’ statement counts the distinct number of different food types at each restaurant. The query know to group the count by restaurant instead of counting the total number of distinct food types as a result of the ‘group by’ statement at the end of the query.



The fourth query aims to find the average price of foods served by each chef across all the different restaurants they work at. This is accomplished by combining the works, serves, and foods tables using natural joins. Works and serves perform a natural join using the restID attribute, which is the only one shared between those two tables allowing for the natural join instead of inner join. The combined serves and works table and foods table are naturally joined using the foodID attribute, which is shared between serves and foods and is the only commonality. This once again allows for the natural join instead of inner join. Once all the tables are combined, chefID and the price attribute from the foods table are available in the same table. In the ‘select’ statement the average of the food prices are performed for each chef. The query knows to perform the average for each chef instead of the overall average because of the ‘group by’ statement in the query.



```
38    -- Question 5
39    select restID, avg(foods.price) as foodprice
40    from serves natural join foods
41    group by restID
42    having (foodprice) >= all
43
44       -- the subquery is formed here
45       (select avg(foods.price)
46        from serves natural join foods
47        group by restID);
```

100% 22:47

Result Grid Filter Rows: Search Export:

restID	foodprice
1	13.5
4	13.5
6	13.5

The fifth query aims to find the restaurant with the highest average food price. First a natural join between serves and foods needs to be performed using the one column in common, foodID. After that, both the restID and foods.price attributes are available in the same place to be used in a query together. The average of the foods price is performed for each restaurant, which is able to be done due to the ‘group by’ statement. This statement ensures that the average is performed for each restaurant and not overall. Those averages are then saved under the alias foodprice. A ‘having’ statement is then performed which limits the output to only the restaurants with the highest average food price. The subquery formed below once again goes through a similar process as described above by performing a natural join on serves and foods and then finding the average price for each restaurant. By using a having statement and a subquery, the

output is not limited to one result, which is helpful when it is possible for multiple restaurants to all share the highest value as is the case here.