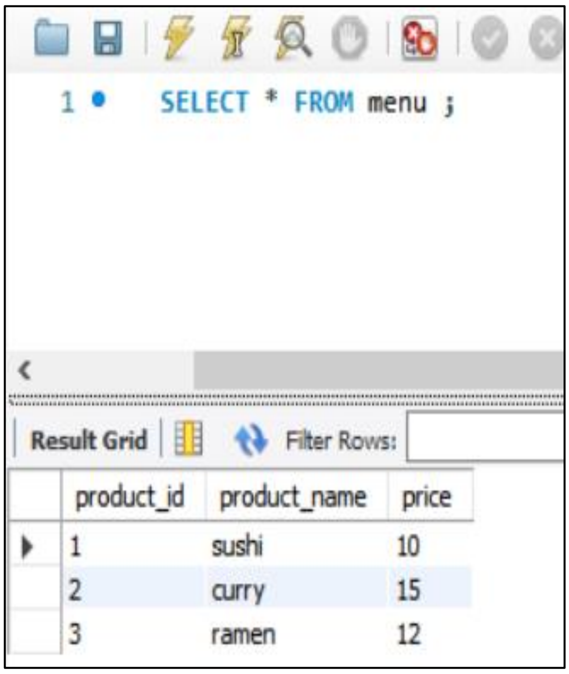
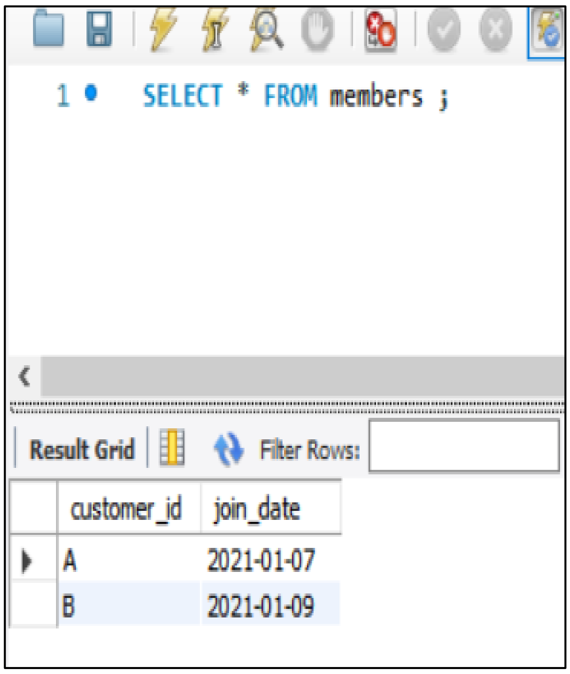
TABLES

- Menu
- Sales
- Members



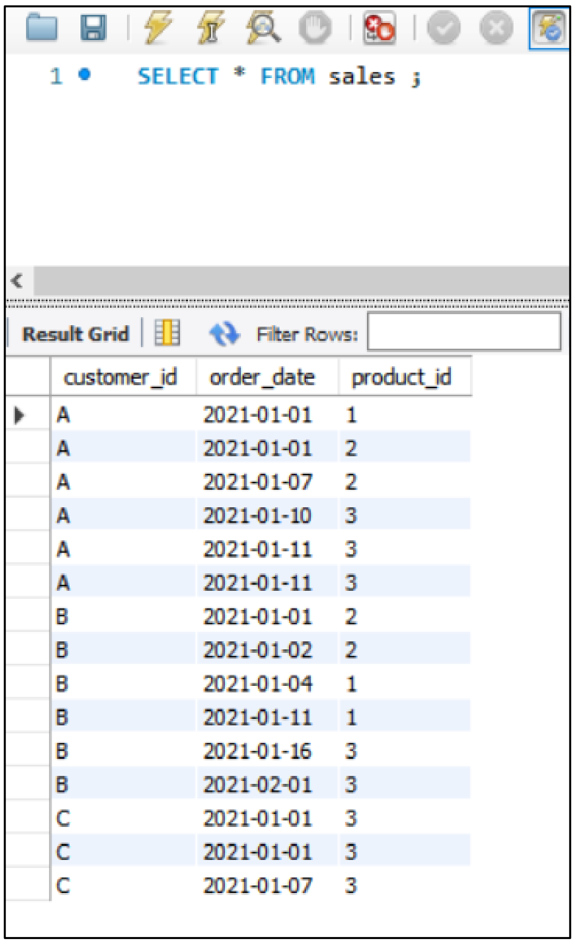
```
SELECT * FROM menu ;
```

product_id	product_name	price
1	sushi	10
2	curry	15
3	ramen	12



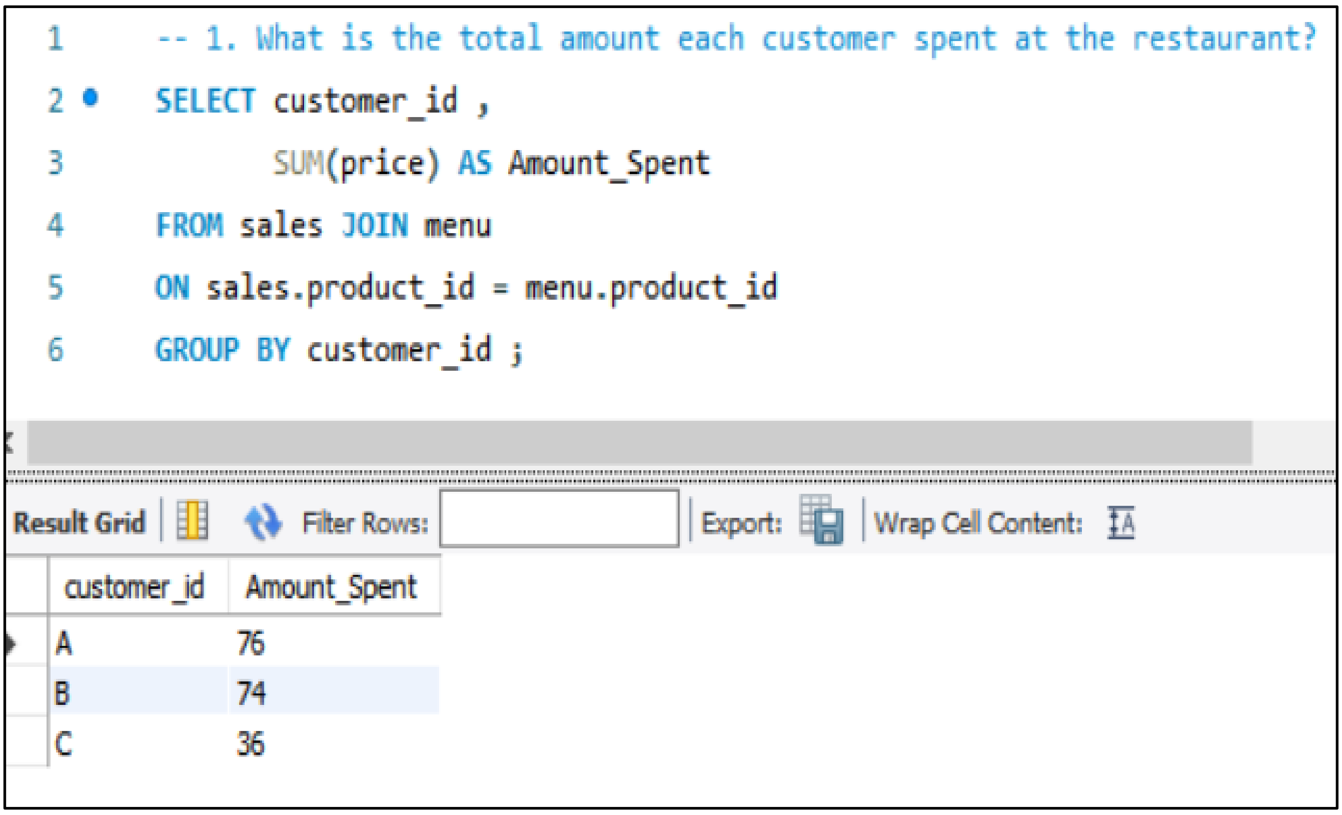
```
SELECT * FROM members ;
```

customer_id	join_date
A	2021-01-07
B	2021-01-09



```
SELECT * FROM sales ;
```

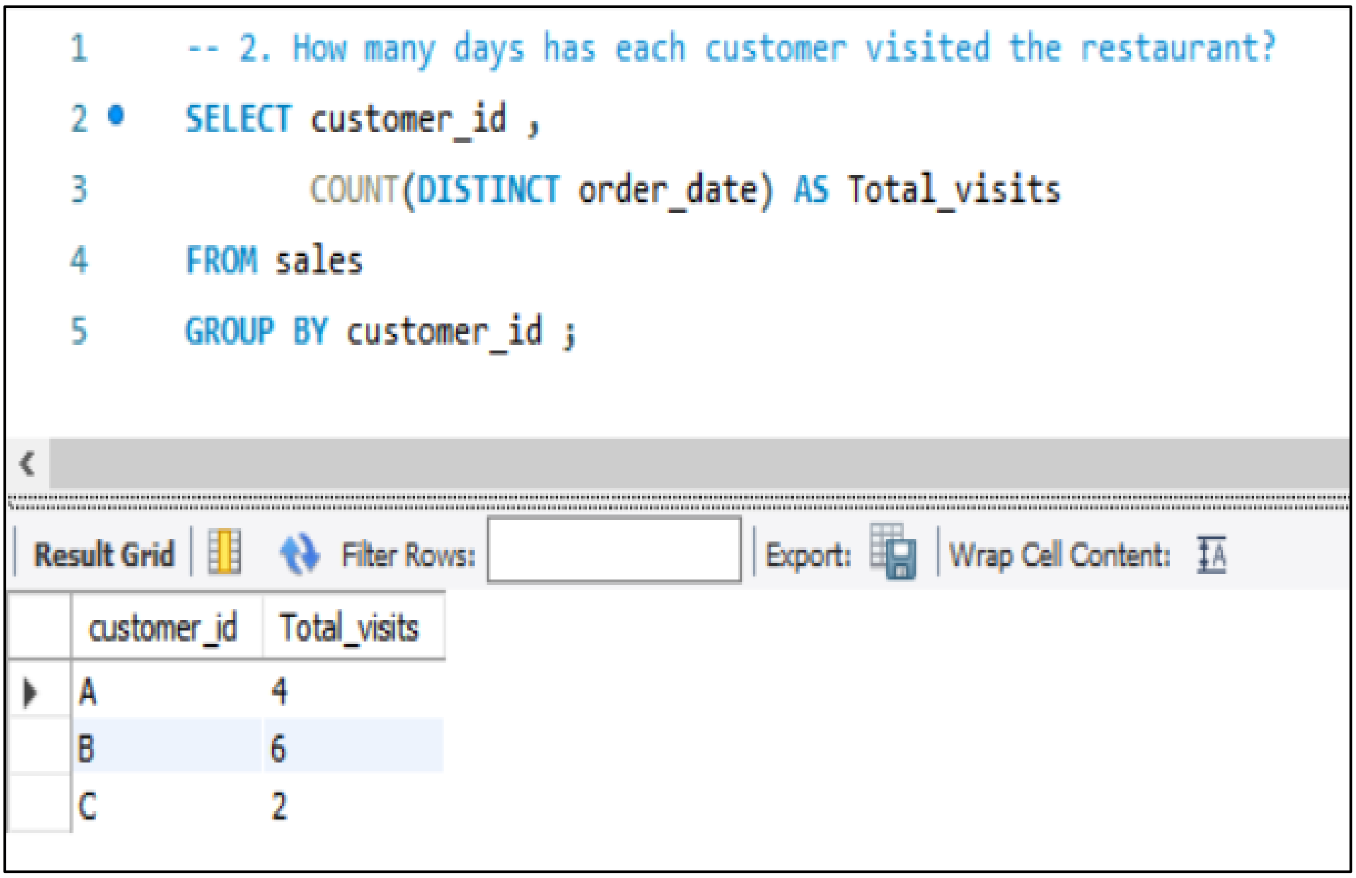
customer_id	order_date	product_id
A	2021-01-01	1
A	2021-01-01	2
A	2021-01-07	2
A	2021-01-10	3
A	2021-01-11	3
A	2021-01-11	3
B	2021-01-01	2
B	2021-01-02	2
B	2021-01-04	1
B	2021-01-11	1
B	2021-01-16	3
B	2021-02-01	3
C	2021-01-01	3
C	2021-01-01	3
C	2021-01-07	3



```sql
-- 1. What is the total amount each customer spent at the restaurant?
SELECT customer_id ,
       SUM(price) AS Amount_Spent
FROM sales JOIN menu
ON sales.product_id = menu.product_id
GROUP BY customer_id ;
```

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

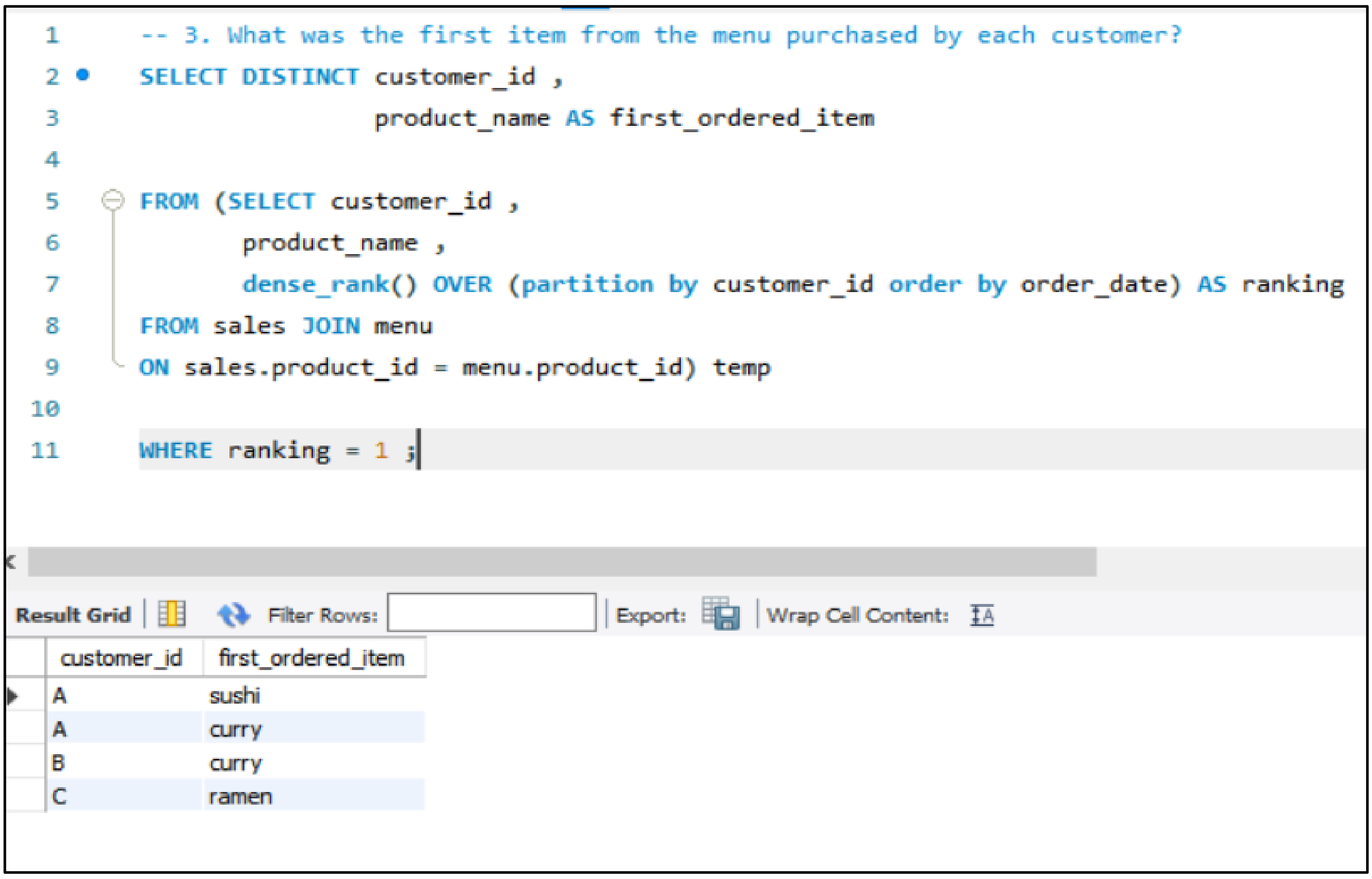
customer_id	Amount_Spent
A	76
B	74
C	36



```sql
-- 2. How many days has each customer visited the restaurant?
SELECT customer_id ,
       COUNT(DISTINCT order_date) AS Total_visits
FROM sales
GROUP BY customer_id ;
```

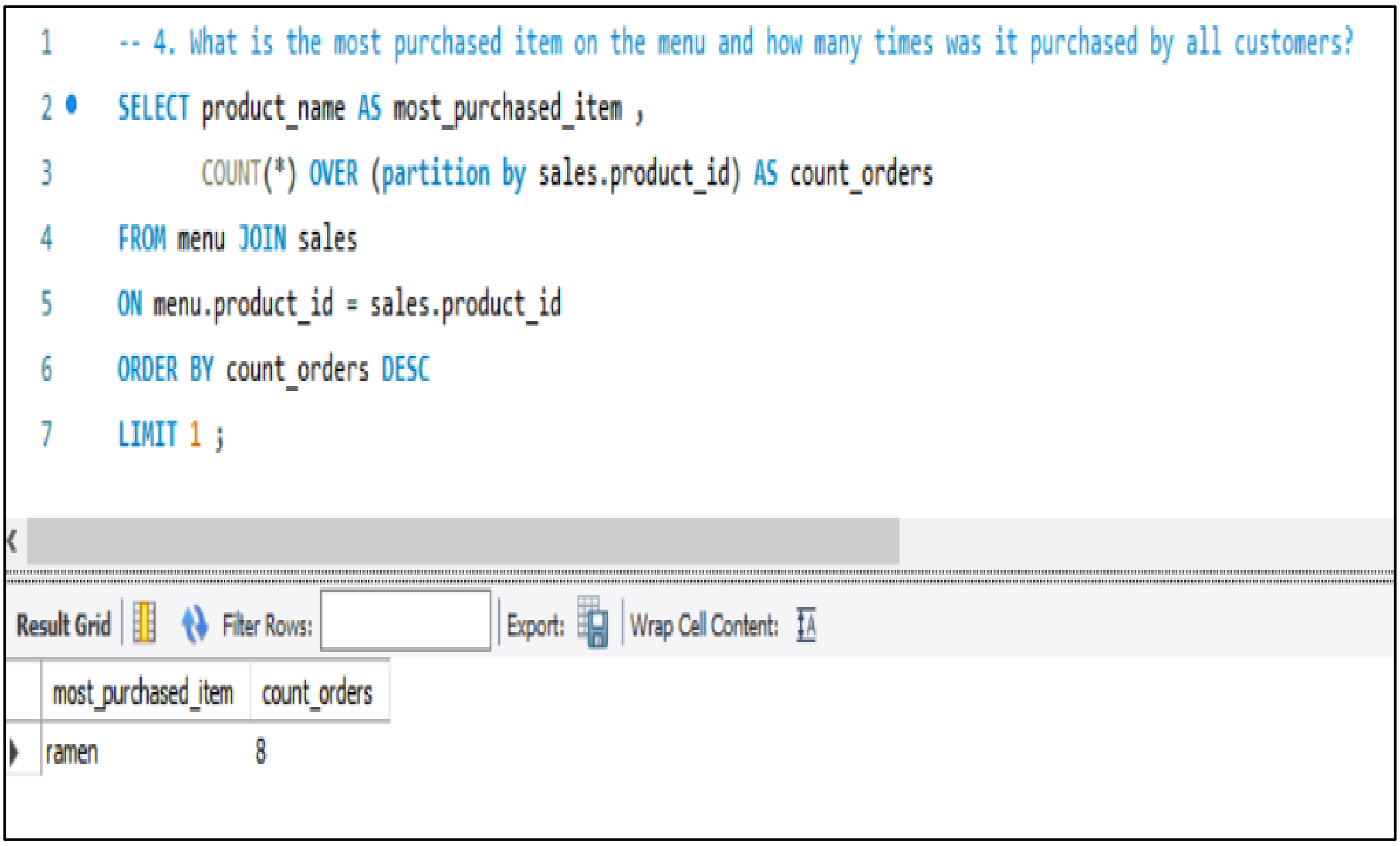
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	Total_visits
A	4
B	6
C	2



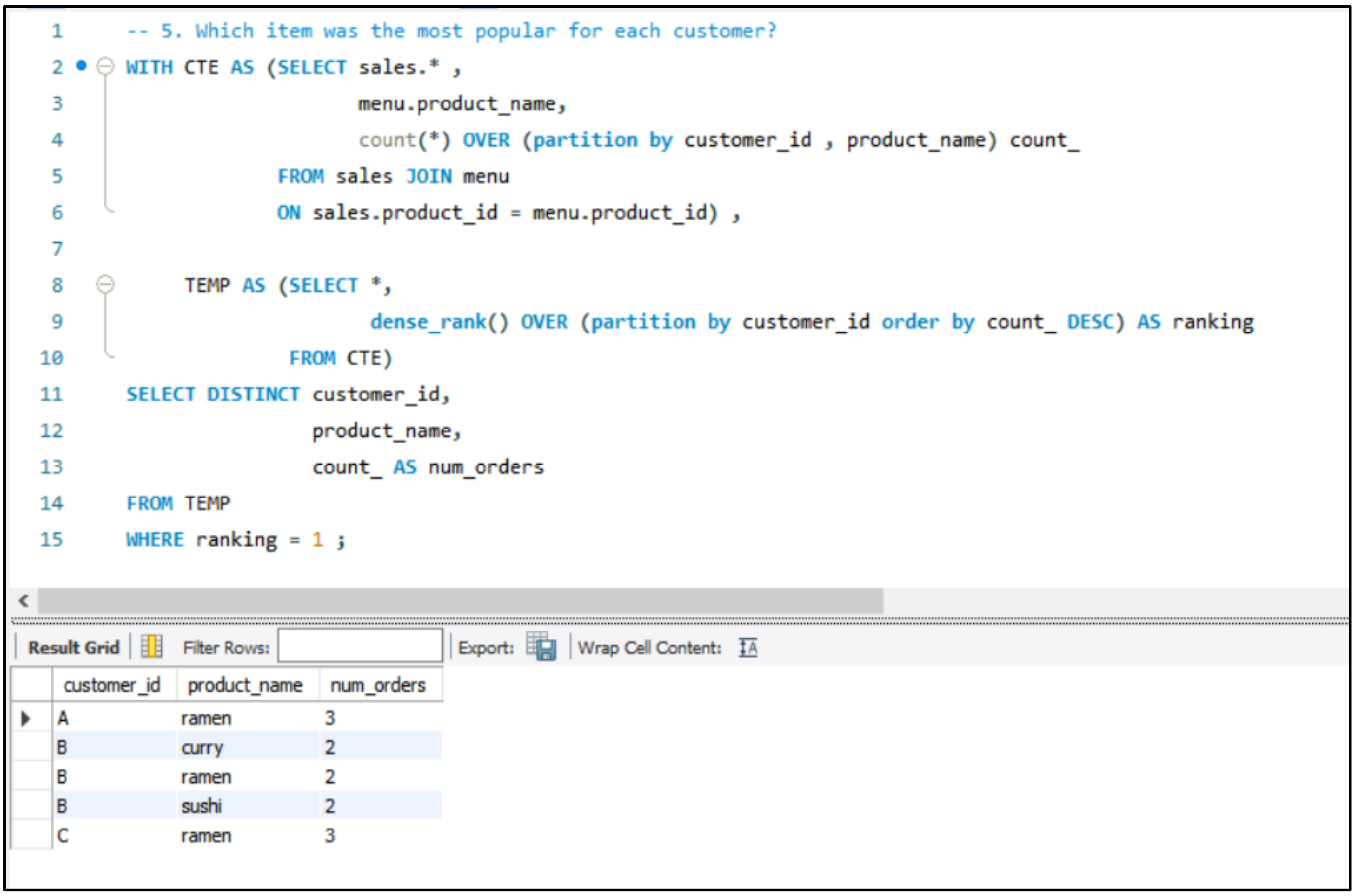
```sql
-- 3. What was the first item from the menu purchased by each customer?
SELECT DISTINCT customer_id ,
                product_name AS first_ordered_item

FROM (SELECT customer_id ,
       product_name ,
       dense_rank() OVER (partition by customer_id order by order_date) AS ranking
FROM sales JOIN menu
ON sales.product_id = menu.product_id) temp

WHERE ranking = 1 ;
```

```
customer_id  first_ordered_item
A            sushi
A            curry
B            curry
C            ramen
```



```sql
-- 4. What is the most purchased item on the menu and how many times was it purchased by all customers?
SELECT product_name AS most_purchased_item ,
       COUNT(*) OVER (partition by sales.product_id) AS count_orders
FROM menu JOIN sales
ON menu.product_id = sales.product_id
ORDER BY count_orders DESC
LIMIT 1 ;
```

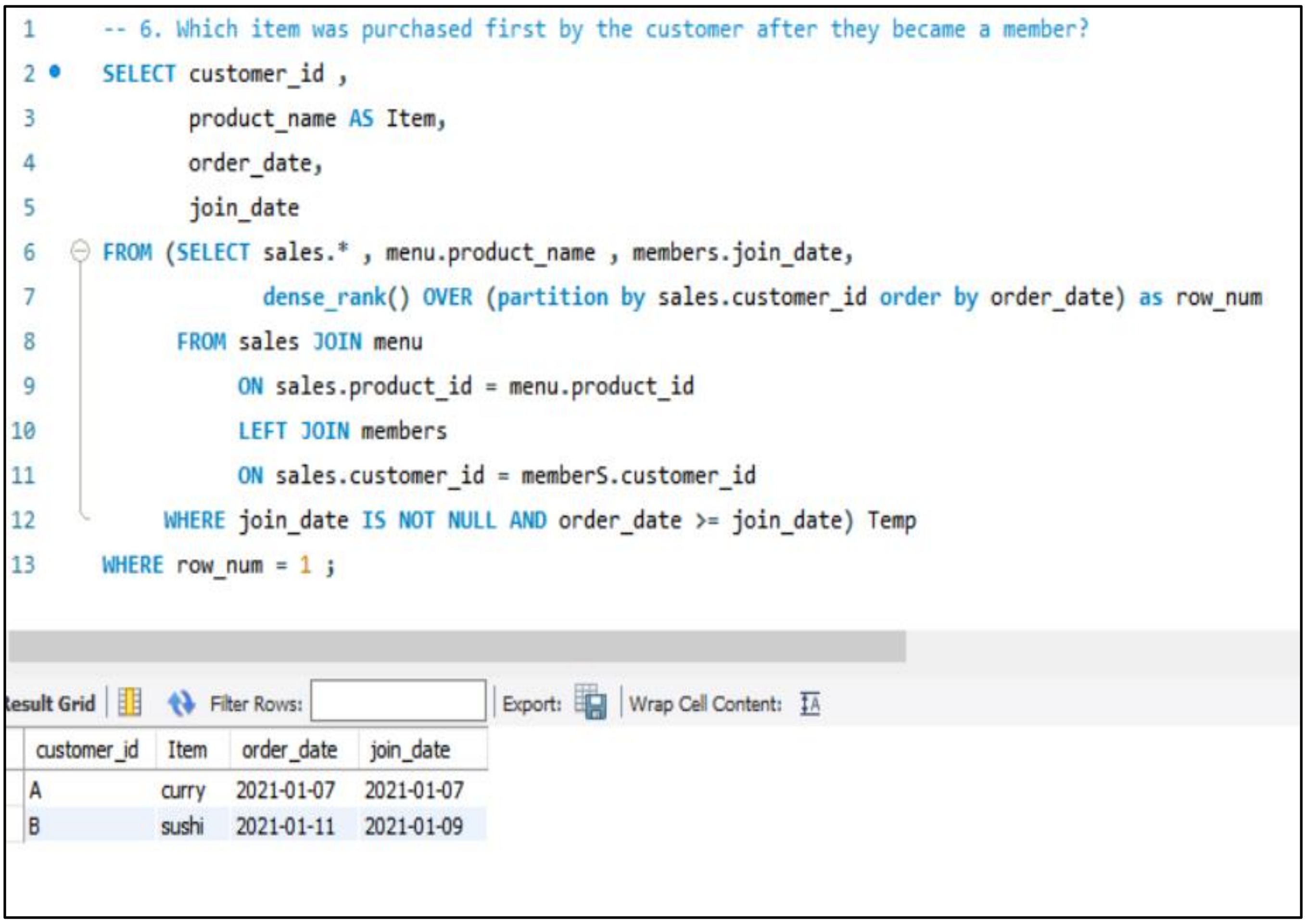
```
most_purchased_item  count_orders
ramen                8
```



```sql
-- 5. Which item was the most popular for each customer?
WITH CTE AS (SELECT sales.* ,
                    menu.product_name,
                    count(*) OVER (partition by customer_id , product_name) count_
             FROM sales JOIN menu
             ON sales.product_id = menu.product_id) ,

       TEMP AS (SELECT *,
                       dense_rank() OVER (partition by customer_id order by count_ DESC) AS ranking
              FROM CTE)
SELECT DISTINCT customer_id,
                product_name,
                count_ AS num_orders
FROM TEMP
WHERE ranking = 1 ;
```

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

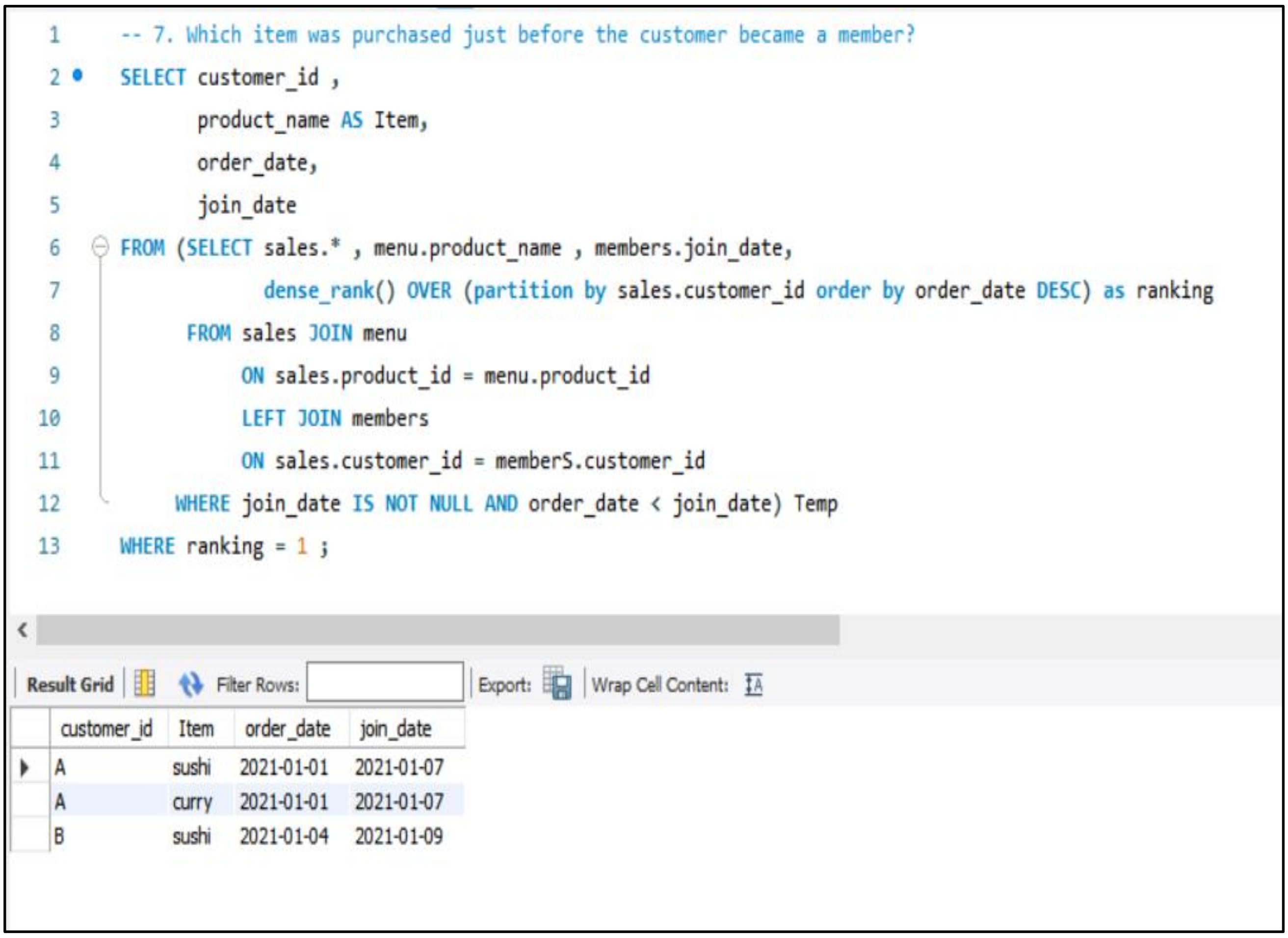
customer_id	product_name	num_orders
A	ramen	3
B	curry	2
B	ramen	2
B	sushi	2
C	ramen	3



```sql
-- 6. Which item was purchased first by the customer after they became a member?
SELECT customer_id ,
       product_name AS Item,
       order_date,
       join_date
FROM (SELECT sales.* , menu.product_name , members.join_date,
              dense_rank() OVER (partition by sales.customer_id order by order_date) as row_num
      FROM sales JOIN menu
          ON sales.product_id = menu.product_id
          LEFT JOIN members
          ON sales.customer_id = memberS.customer_id
     WHERE join_date IS NOT NULL AND order_date >= join_date) Temp
WHERE row_num = 1 ;
```

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

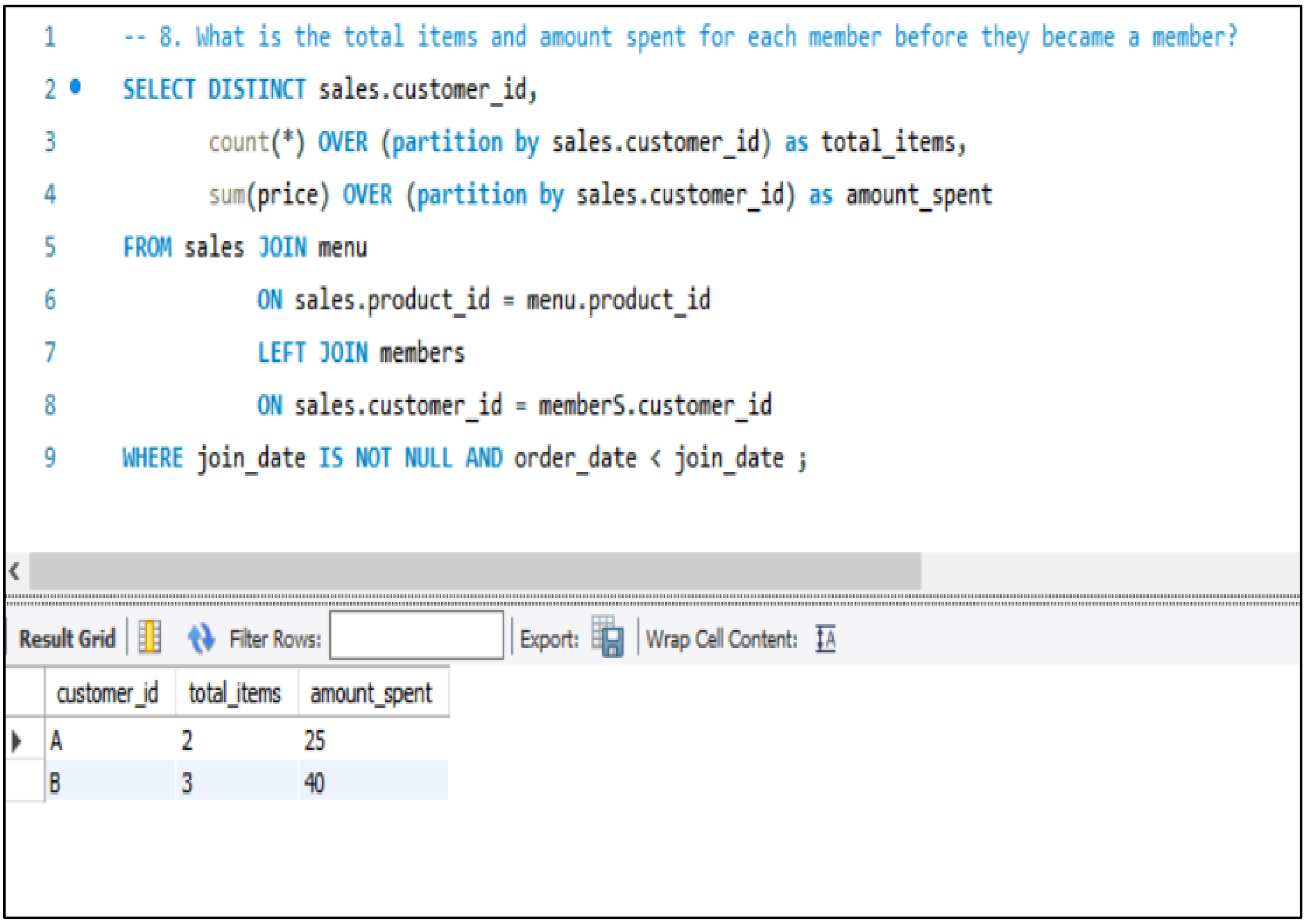
customer_id	Item	order_date	join_date
A	curry	2021-01-07	2021-01-07
B	sushi	2021-01-11	2021-01-09



```sql
-- 7. Which item was purchased just before the customer became a member?
SELECT customer_id ,
        product_name AS Item,
        order_date,
        join_date
FROM (SELECT sales.* , menu.product_name , members.join_date,
                dense_rank() OVER (partition by sales.customer_id order by order_date DESC) as ranking
        FROM sales JOIN menu
            ON sales.product_id = menu.product_id
            LEFT JOIN members
            ON sales.customer_id = memberS.customer_id
        WHERE join_date IS NOT NULL AND order_date < join_date) Temp
WHERE ranking = 1 ;
```

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

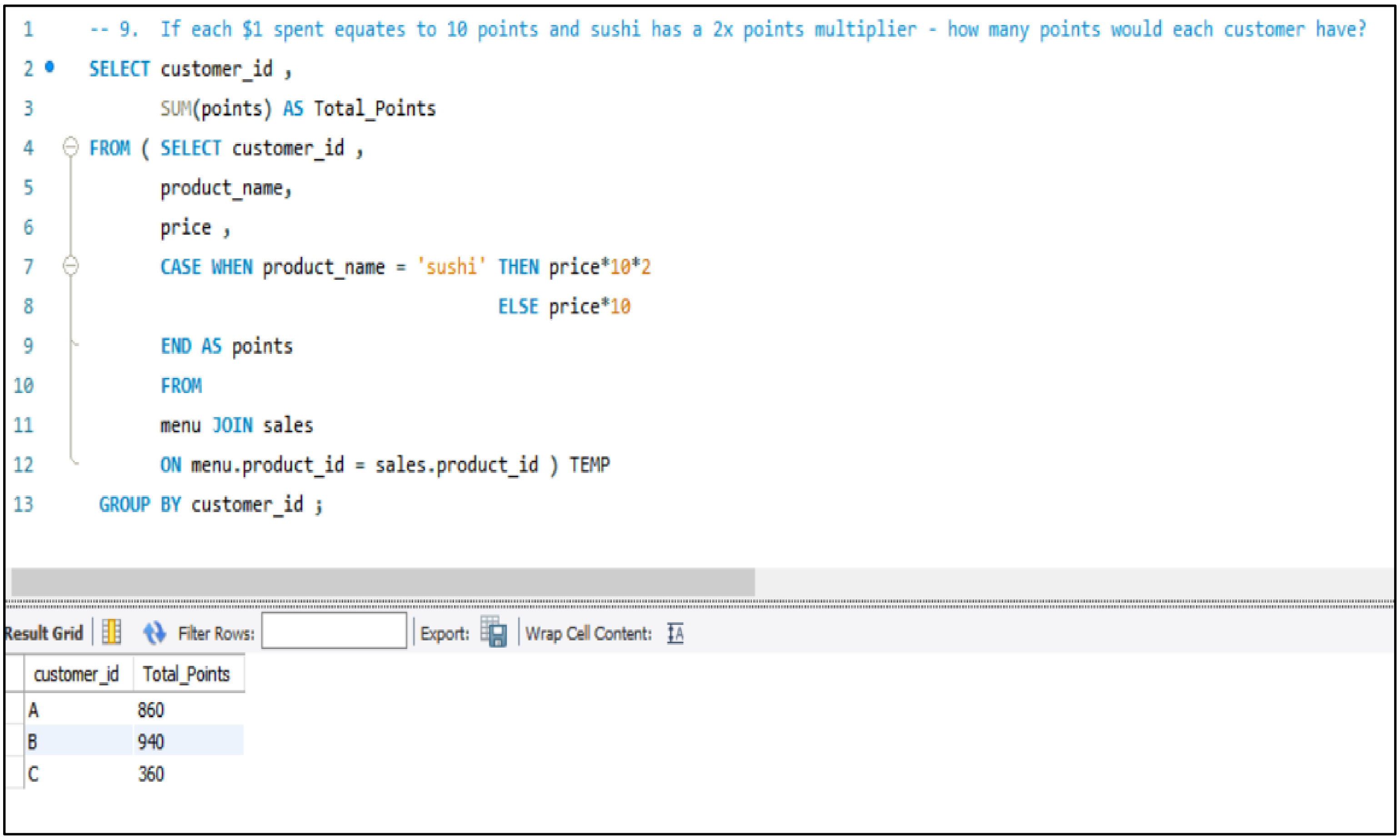
customer_id	Item	order_date	join_date
A	sushi	2021-01-01	2021-01-07
A	curry	2021-01-01	2021-01-07
B	sushi	2021-01-04	2021-01-09



```sql
-- 8. What is the total items and amount spent for each member before they became a member?
SELECT DISTINCT sales.customer_id,
        count(*) OVER (partition by sales.customer_id) as total_items,
        sum(price) OVER (partition by sales.customer_id) as amount_spent
FROM sales JOIN menu
            ON sales.product_id = menu.product_id
            LEFT JOIN members
            ON sales.customer_id = memberS.customer_id
WHERE join_date IS NOT NULL AND order_date < join_date ;
```

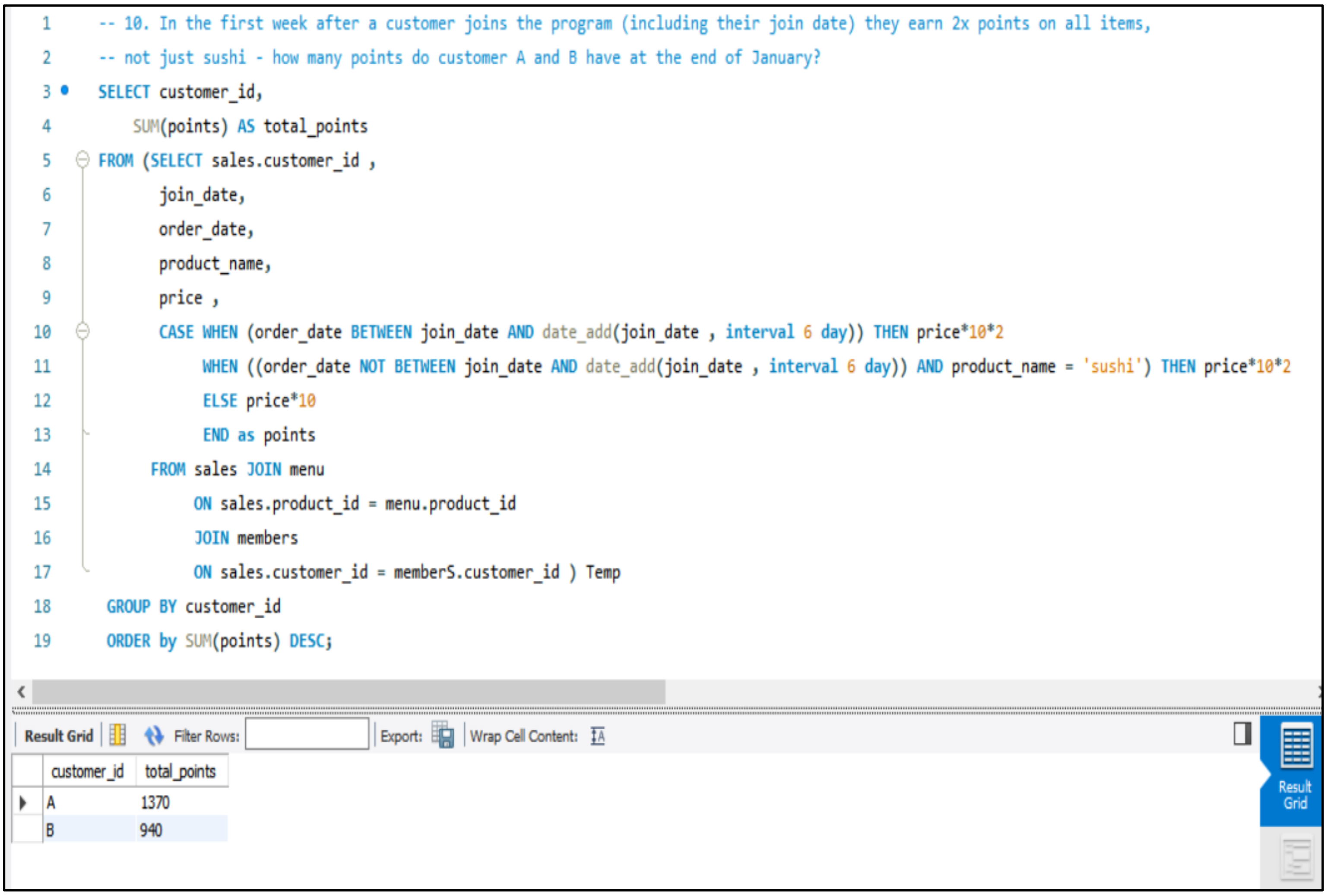
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	total_items	amount_spent
A	2	25
B	3	40



```sql
-- 9.  If each $1 spent equates to 10 points and sushi has a 2x points multiplier - how many points would each customer have?
SELECT customer_id ,
       SUM(points) AS Total_Points
FROM ( SELECT customer_id ,
       product_name,
       price ,
       CASE WHEN product_name = 'sushi' THEN price*10*2
                                        ELSE price*10
       END AS points
       FROM
       menu JOIN sales
       ON menu.product_id = sales.product_id ) TEMP
 GROUP BY customer_id ;
```

```
customer_id  Total_Points
A            860
B            940
C            360
```

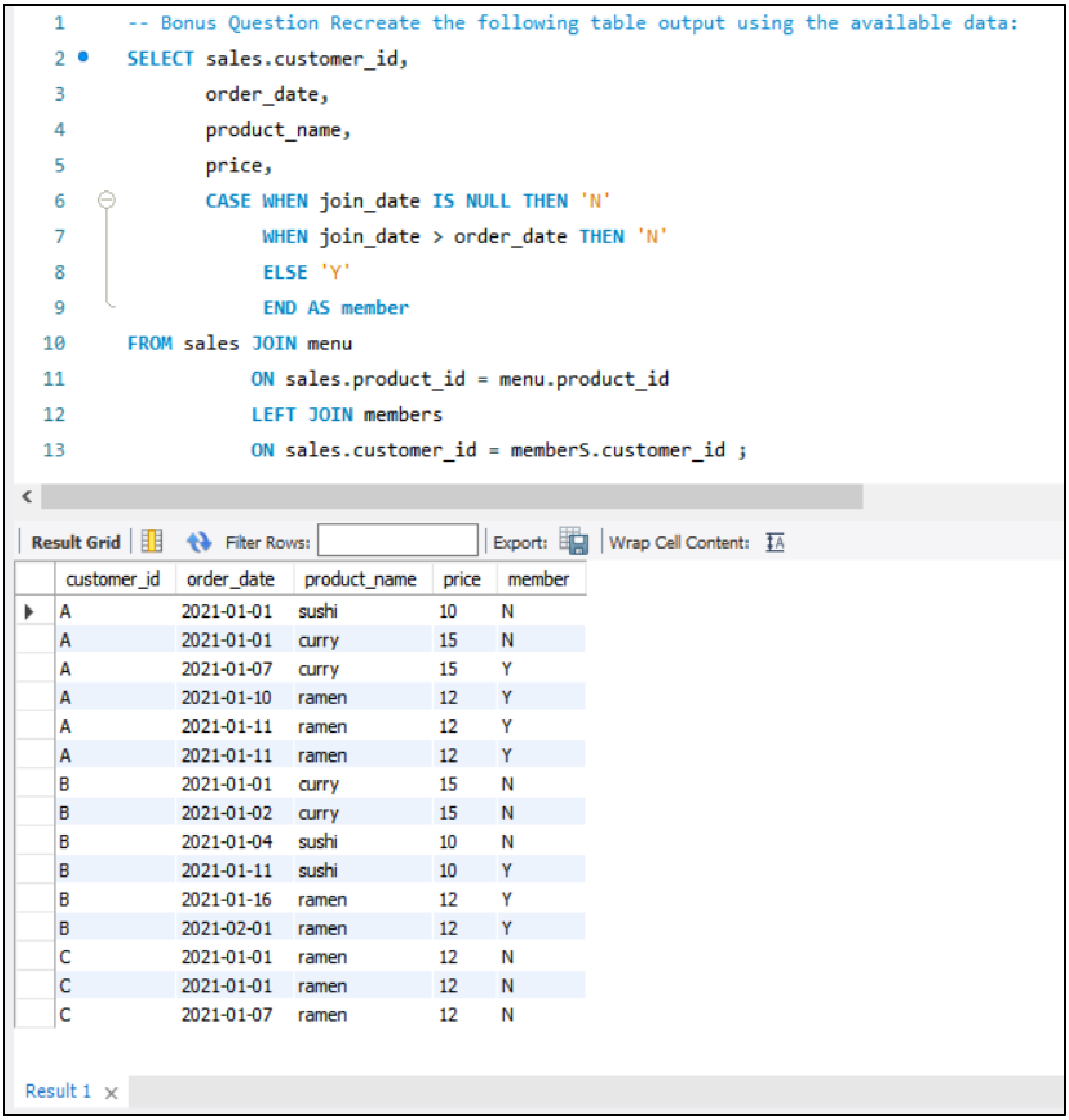


```sql
-- 10. In the first week after a customer joins the program (including their join date) they earn 2x points on all items,
-- not just sushi - how many points do customer A and B have at the end of January?
SELECT customer_id,
    SUM(points) AS total_points
FROM (SELECT sales.customer_id ,
       join_date,
       order_date,
       product_name,
       price ,
       CASE WHEN (order_date BETWEEN join_date AND date_add(join_date , interval 6 day)) THEN price*10*2
            WHEN ((order_date NOT BETWEEN join_date AND date_add(join_date , interval 6 day)) AND product_name = 'sushi') THEN price*10*2
            ELSE price*10
            END as points
      FROM sales JOIN menu
           ON sales.product_id = menu.product_id
           JOIN members
           ON sales.customer_id = memberS.customer_id ) Temp
 GROUP BY customer_id
 ORDER by SUM(points) DESC;
```

```
customer_id  total_points
A            1370
B            940
```

BONUS QUESTION

1. The following questions are related creating basic data tables that Danny and his team can use to quickly derive insights without needing to join the underlying tables using SQL.



```
-- Bonus Question Recreate the following table output using the available data:
SELECT sales.customer_id,
        order_date,
        product_name,
        price,
        CASE WHEN join_date IS NULL THEN 'N'
             WHEN join_date > order_date THEN 'N'
             ELSE 'Y'
             END AS member
FROM sales JOIN menu
            ON sales.product_id = menu.product_id
            LEFT JOIN members
            ON sales.customer_id = memberS.customer_id ;
```

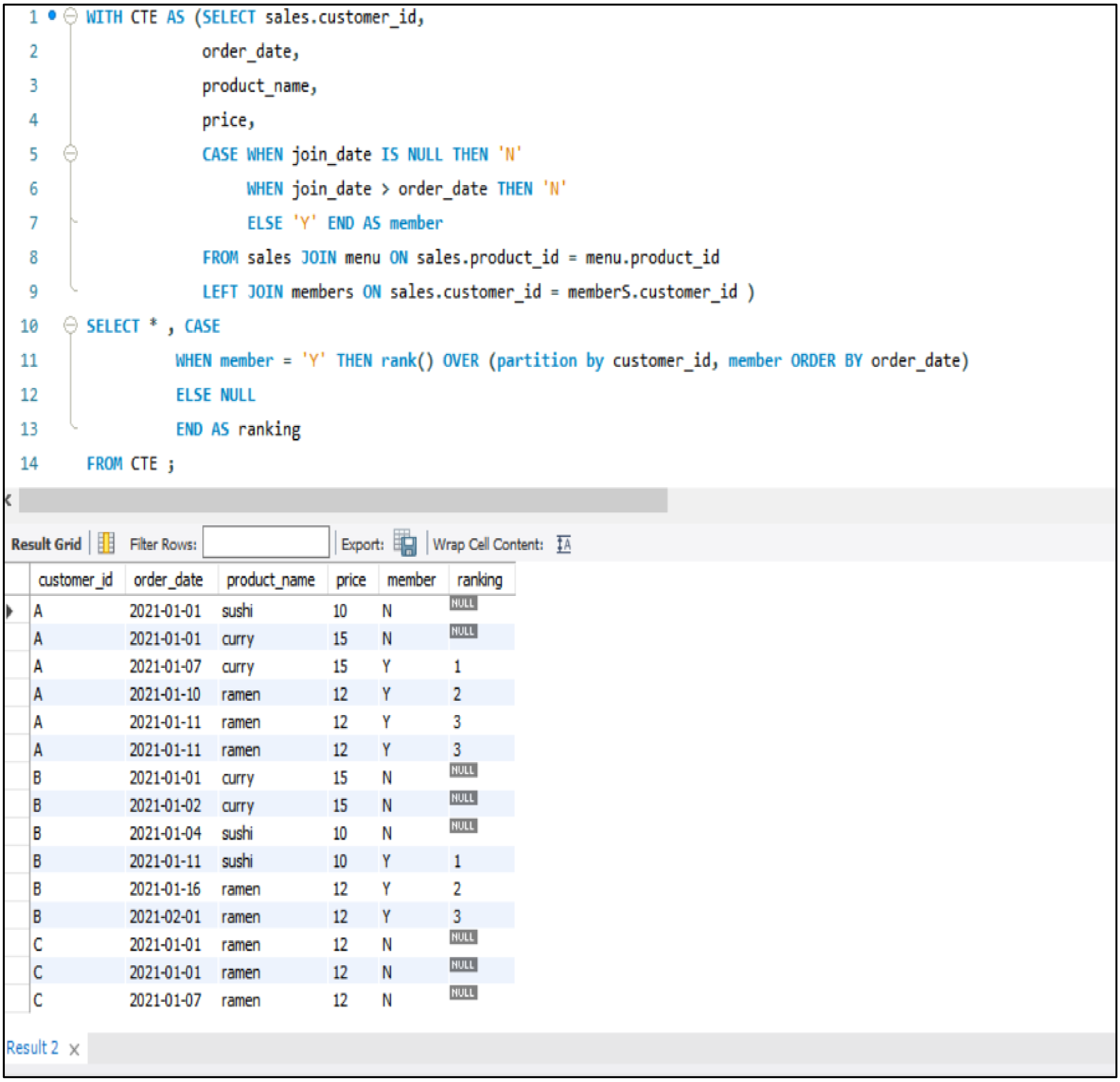
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	order_date	product_name	price	member
A	2021-01-01	sushi	10	N
A	2021-01-01	curry	15	N
A	2021-01-07	curry	15	Y
A	2021-01-10	ramen	12	Y
A	2021-01-11	ramen	12	Y
A	2021-01-11	ramen	12	Y
B	2021-01-01	curry	15	N
B	2021-01-02	curry	15	N
B	2021-01-04	sushi	10	N
B	2021-01-11	sushi	10	Y
B	2021-01-16	ramen	12	Y
B	2021-02-01	ramen	12	Y
C	2021-01-01	ramen	12	N
C	2021-01-01	ramen	12	N
C	2021-01-07	ramen	12	N

Result 1 ×

BONUS QUESTION

2. Danny also requires further information about the ranking of customer products, but he purposely does not need the ranking for non-member purchases so he expects null ranking values for the records when customers are not yet part of the loyalty program.



```sql
WITH CTE AS (SELECT sales.customer_id,
             order_date,
             product_name,
             price,
             CASE WHEN join_date IS NULL THEN 'N'
                  WHEN join_date > order_date THEN 'N'
                  ELSE 'Y' END AS member
             FROM sales JOIN menu ON sales.product_id = menu.product_id
             LEFT JOIN members ON sales.customer_id = memberS.customer_id )
SELECT * , CASE
           WHEN member = 'Y' THEN rank() OVER (partition by customer_id, member ORDER BY order_date)
           ELSE NULL
           END AS ranking
FROM CTE ;
```

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

customer_id	order_date	product_name	price	member	ranking
A	2021-01-01	sushi	10	N	NULL
A	2021-01-01	curry	15	N	NULL
A	2021-01-07	curry	15	Y	1
A	2021-01-10	ramen	12	Y	2
A	2021-01-11	ramen	12	Y	3
A	2021-01-11	ramen	12	Y	3
B	2021-01-01	curry	15	N	NULL
B	2021-01-02	curry	15	N	NULL
B	2021-01-04	sushi	10	N	NULL
B	2021-01-11	sushi	10	Y	1
B	2021-01-16	ramen	12	Y	2
B	2021-02-01	ramen	12	Y	3
C	2021-01-01	ramen	12	N	NULL
C	2021-01-01	ramen	12	N	NULL
C	2021-01-07	ramen	12	N	NULL

Result 2 ×