8WEEKSQLCHALLENGE.COM

CASE STUDY #1

THE TASTE OF SUCCESS

DATAWITHDANNY.COM

INTRODUCTION

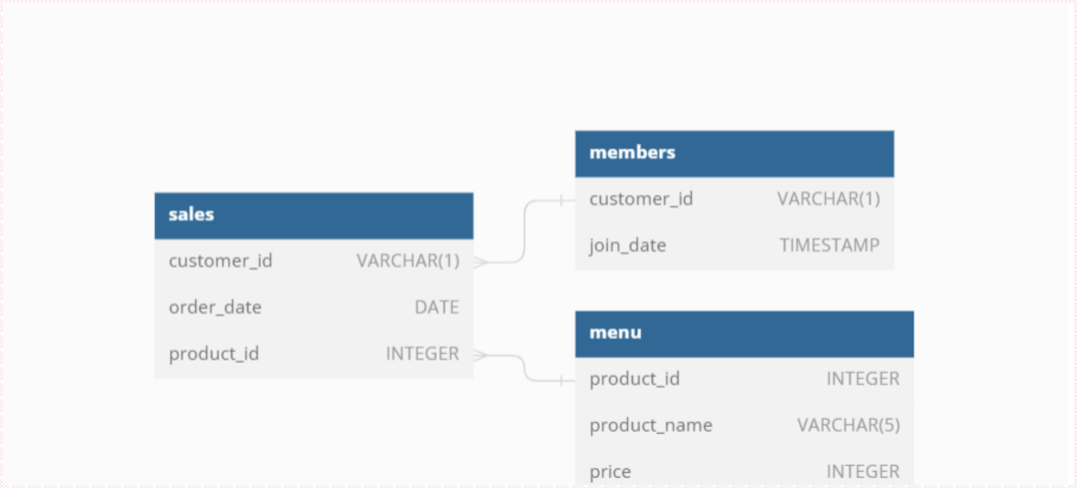
Danny seriously loves Japanese food so in the beginning of 2021, he decides to embark upon a risky venture and opens up a cute little restaurant that sells his 3 favourite foods: sushi, curry and ramen.
Danny's Diner is in need of your assistance to help the restaurant stay afloat - the restaurant has captured some very basic data from their few months of operation but have no idea how to use their data to help them run the business.

PROBLEM STATEMENT

Danny wants to use the data to answer a few simple questions about his customers, especially about their visiting patterns, how much money they've spent and also which menu items are their favourite. Having this deeper connection with his customers will help him deliver a better and more personalised experience for his loyal customers.
He plans on using these insights to help him decide whether he should expand the existing customer loyalty program - additionally he needs help to generate some basic datasets so his team can easily inspect the data without needing to use SQL.
Danny has provided you with a sample of his overall customer data due to privacy issues - but he hopes that these examples are enough for you to write fully functioning SQL queries to help him answer his questions!

Danny has shared with you 3 key datasets for this case study:

- sales
- menu
- members



Example Datasets
All datasets exist within the dannys_diner database schema - be sure to include this reference within your SQL scripts as you start exploring the data and answering the case study questions.

Table 1: sales
The sales table captures all customer_id level purchases with an corresponding order_date and product_id information for when and what menu items were ordered.

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

Table 2: menu
The menu table maps the product_id to the actual product_name and price of each menu item.

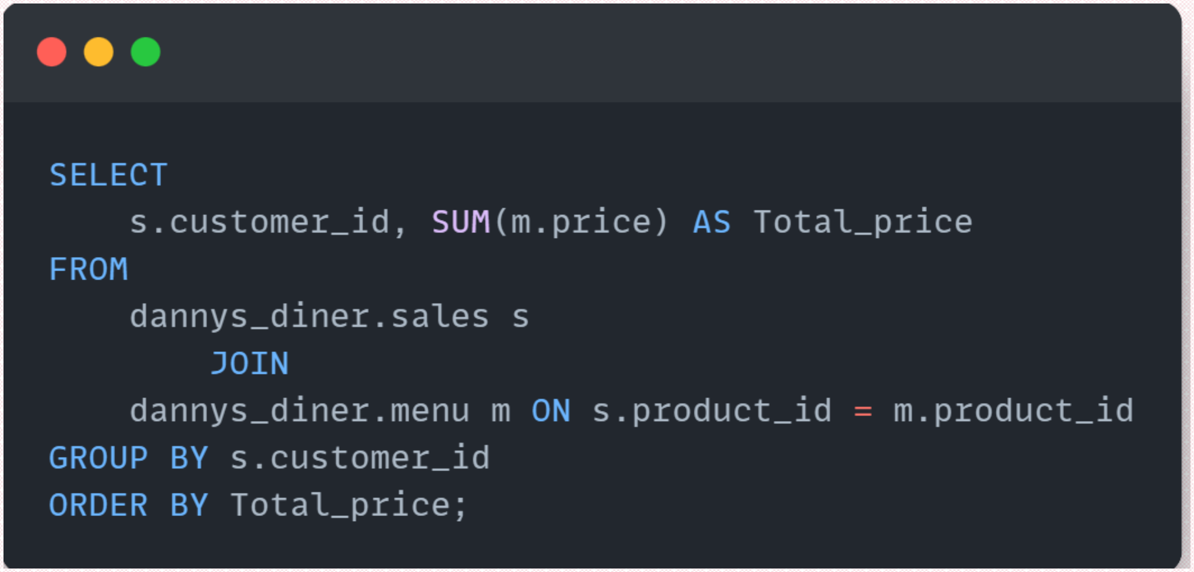
product_id	product_name	price
1	sushi	10
2	curry	15
3	ramen	12

Table 3: members
The final members table captures the join_date when a customer_id joined the beta version of the Danny’s Diner loyalty program.

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

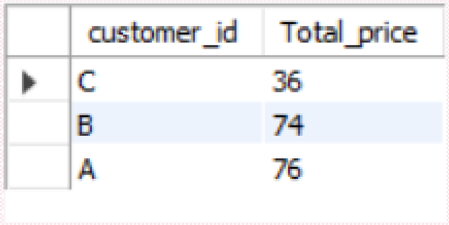
1. What is the total amount each customer spent at the restaurant?

Query



```sql
SELECT
    s.customer_id, SUM(m.price) AS Total_price
FROM
    dannys_diner.sales s
        JOIN
    dannys_diner.menu m ON s.product_id = m.product_id
GROUP BY s.customer_id
ORDER BY Total_price;
```

Output



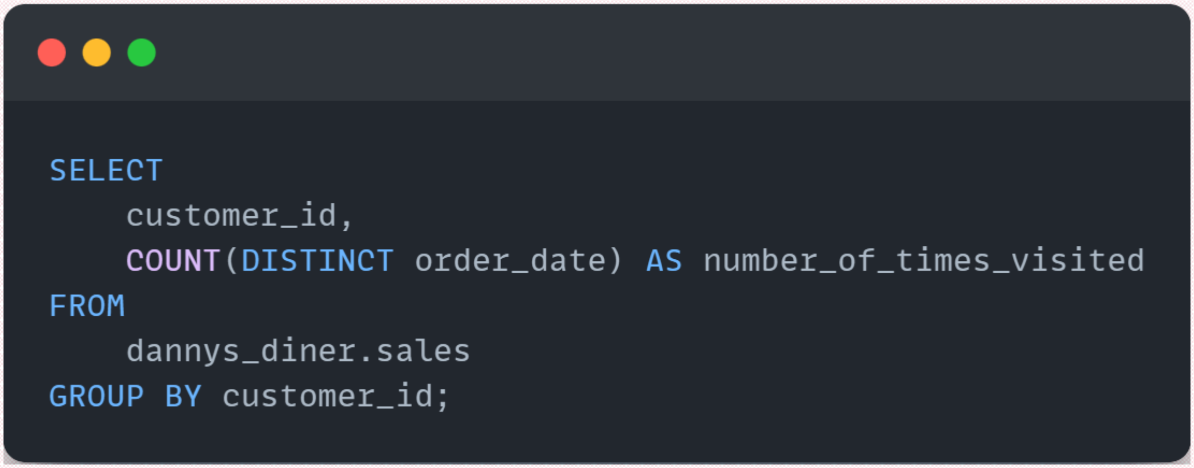
	customer_id	Total_price
▶	C	36
	B	74
	A	76

Analysis

1. Customer A has the highest spending at the restaurant with a spending of $76.
2. Customer B has a spending of $74 and customer C has a spending of only $36

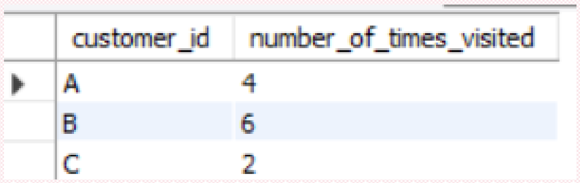
2. How many days has each customer visited the restaurant?

Query



```
SELECT
    customer_id,
    COUNT(DISTINCT order_date) AS number_of_times_visited
FROM
    dannys_diner.sales
GROUP BY customer_id;
```

Output



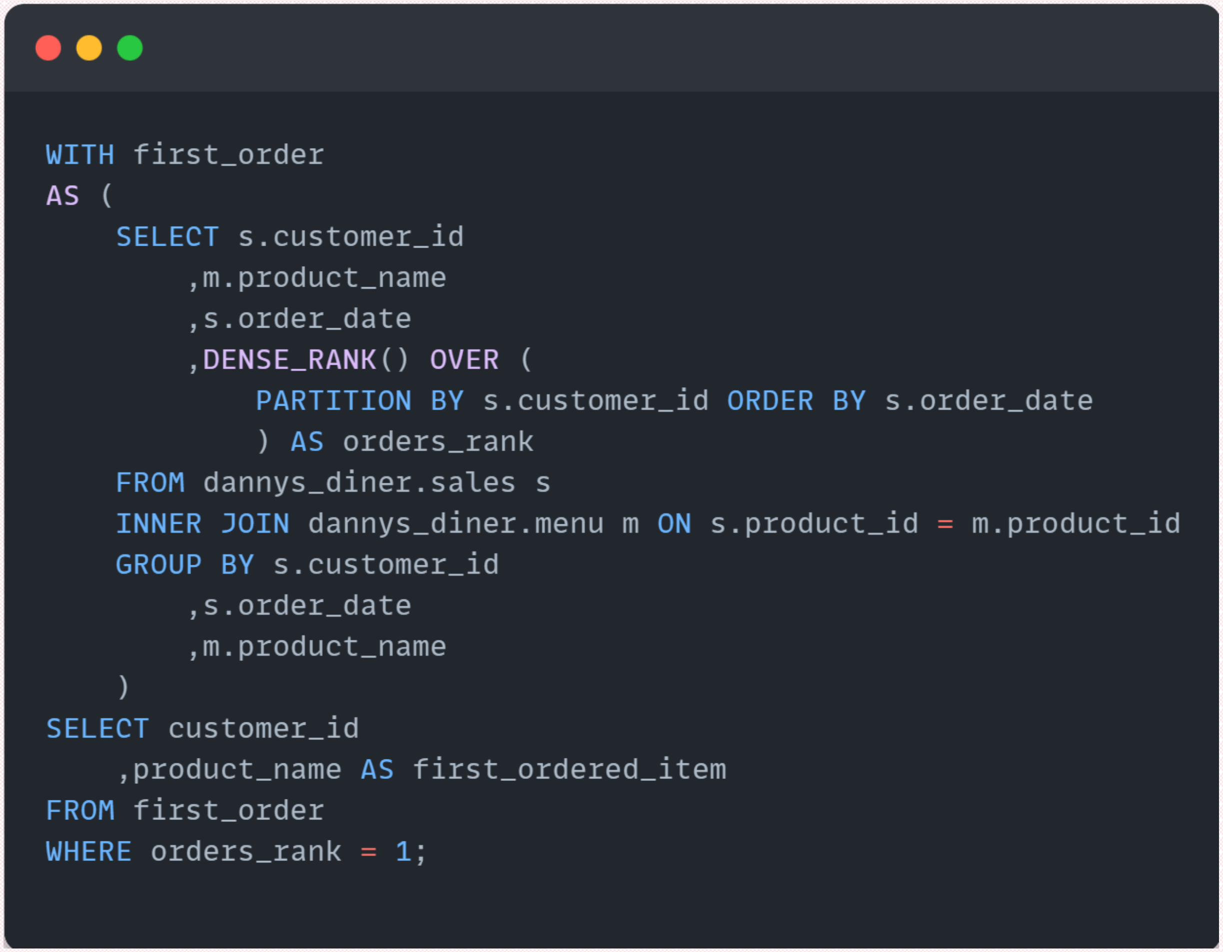
customer_id	number_of_times_visited
A	4
B	6
C	2

Analysis

1. Customer B has visited the restaurant most number of times followed by customer A (4 times) and customer C (2times)

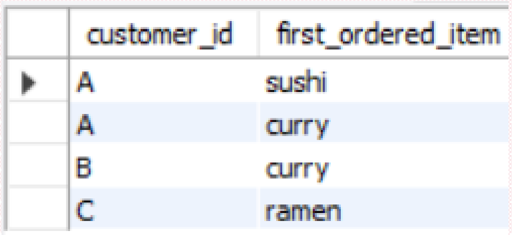
3. What was the first item from the menu purchased by each customer?

Query



```sql
WITH first_order
AS (
    SELECT s.customer_id
        ,m.product_name
        ,s.order_date
        ,DENSE_RANK() OVER (
            PARTITION BY s.customer_id ORDER BY s.order_date
            ) AS orders_rank
    FROM dannys_diner.sales s
    INNER JOIN dannys_diner.menu m ON s.product_id = m.product_id
    GROUP BY s.customer_id
        ,s.order_date
        ,m.product_name
    )
SELECT customer_id
    ,product_name AS first_ordered_item
FROM first_order
WHERE orders_rank = 1;
```

Output



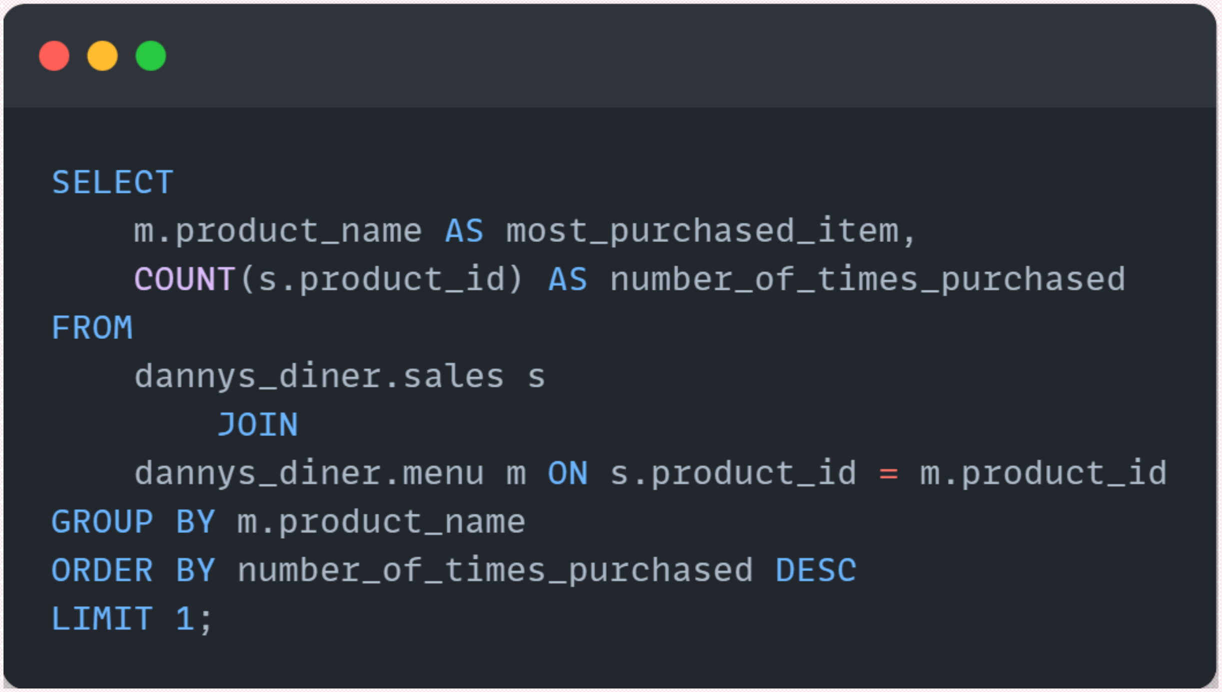
customer_id	first_ordered_item
A	sushi
A	curry
B	curry
C	ramen

Analysis

1. Customer A has ordered sushi and curry as his first items.
2. Customer B has ordered curry
3. Customer C has ordered ramen.

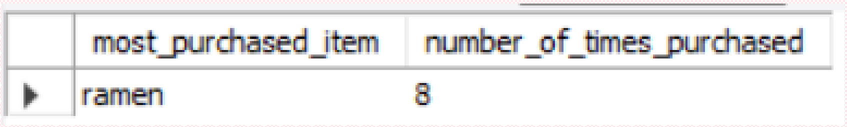
4. What is the most purchased item on the menu and how many times was it purchased by all customers?

Query



```
SELECT
    m.product_name AS most_purchased_item,
    COUNT(s.product_id) AS number_of_times_purchased
FROM
    dannys_diner.sales s
        JOIN
    dannys_diner.menu m ON s.product_id = m.product_id
GROUP BY m.product_name
ORDER BY number_of_times_purchased DESC
LIMIT 1;
```

Output



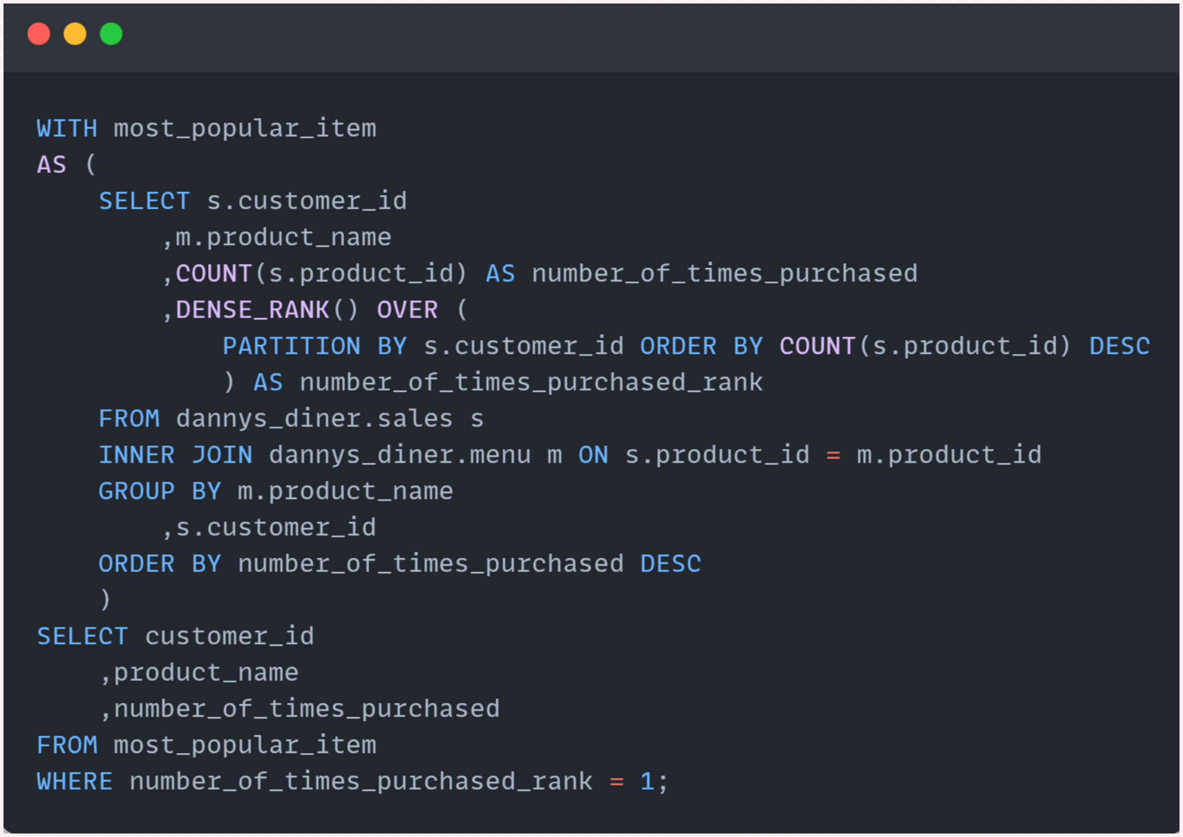
	most_purchased_item	number_of_times_purchased
▶	ramen	8

Analysis

1. Ramen seems to be the most popular item on the menu and was purchased a total of 8 times by all customers.

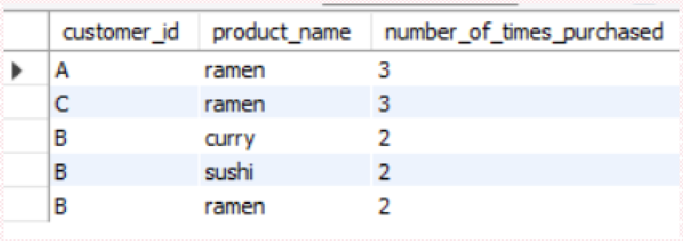
5. Which item was the most popular for each customer?

Query



```sql
WITH most_popular_item
AS (
    SELECT s.customer_id
        ,m.product_name
        ,COUNT(s.product_id) AS number_of_times_purchased
        ,DENSE_RANK() OVER (
            PARTITION BY s.customer_id ORDER BY COUNT(s.product_id) DESC
            ) AS number_of_times_purchased_rank
    FROM dannys_diner.sales s
    INNER JOIN dannys_diner.menu m ON s.product_id = m.product_id
    GROUP BY m.product_name
        ,s.customer_id
    ORDER BY number_of_times_purchased DESC
    )
SELECT customer_id
    ,product_name
    ,number_of_times_purchased
FROM most_popular_item
WHERE number_of_times_purchased_rank = 1;
```

Output



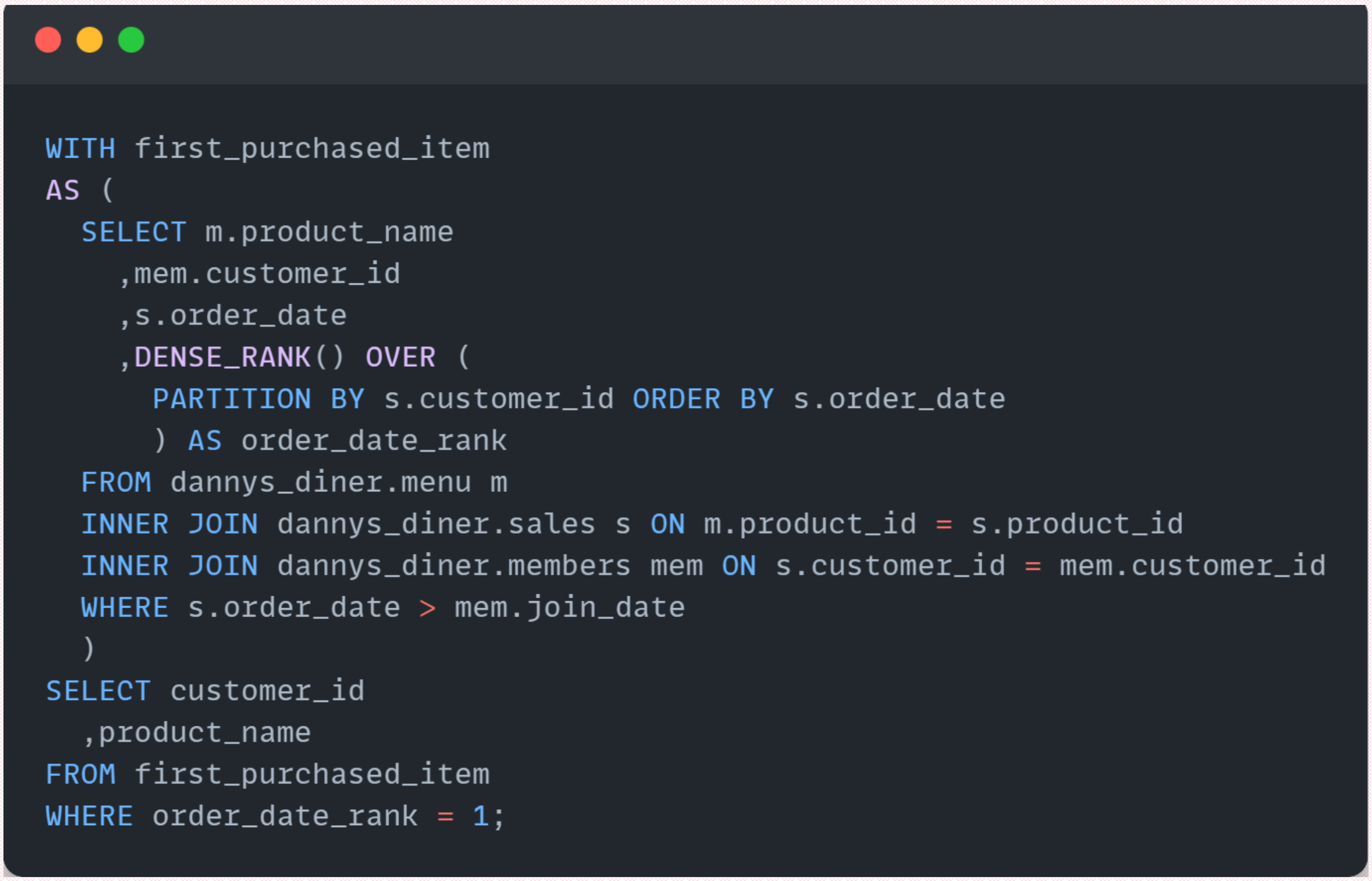
customer_id	product_name	number_of_times_purchased
A	ramen	3
C	ramen	3
B	curry	2
B	sushi	2
B	ramen	2

Analysis

1. Ramen seem to be the most popular item for both customer A and C.
2. However, customer B has ordered all three items (curry, ramen and sushi) twice.

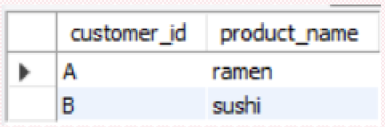
6. Which item was purchased first by the customer after they became a member?

Query



```
WITH first_purchased_item
AS (
  SELECT m.product_name
    ,mem.customer_id
    ,s.order_date
    ,DENSE_RANK() OVER (
      PARTITION BY s.customer_id ORDER BY s.order_date
      ) AS order_date_rank
  FROM dannys_diner.menu m
  INNER JOIN dannys_diner.sales s ON m.product_id = s.product_id
  INNER JOIN dannys_diner.members mem ON s.customer_id = mem.customer_id
  WHERE s.order_date > mem.join_date
  )
SELECT customer_id
  ,product_name
FROM first_purchased_item
WHERE order_date_rank = 1;
```

Output



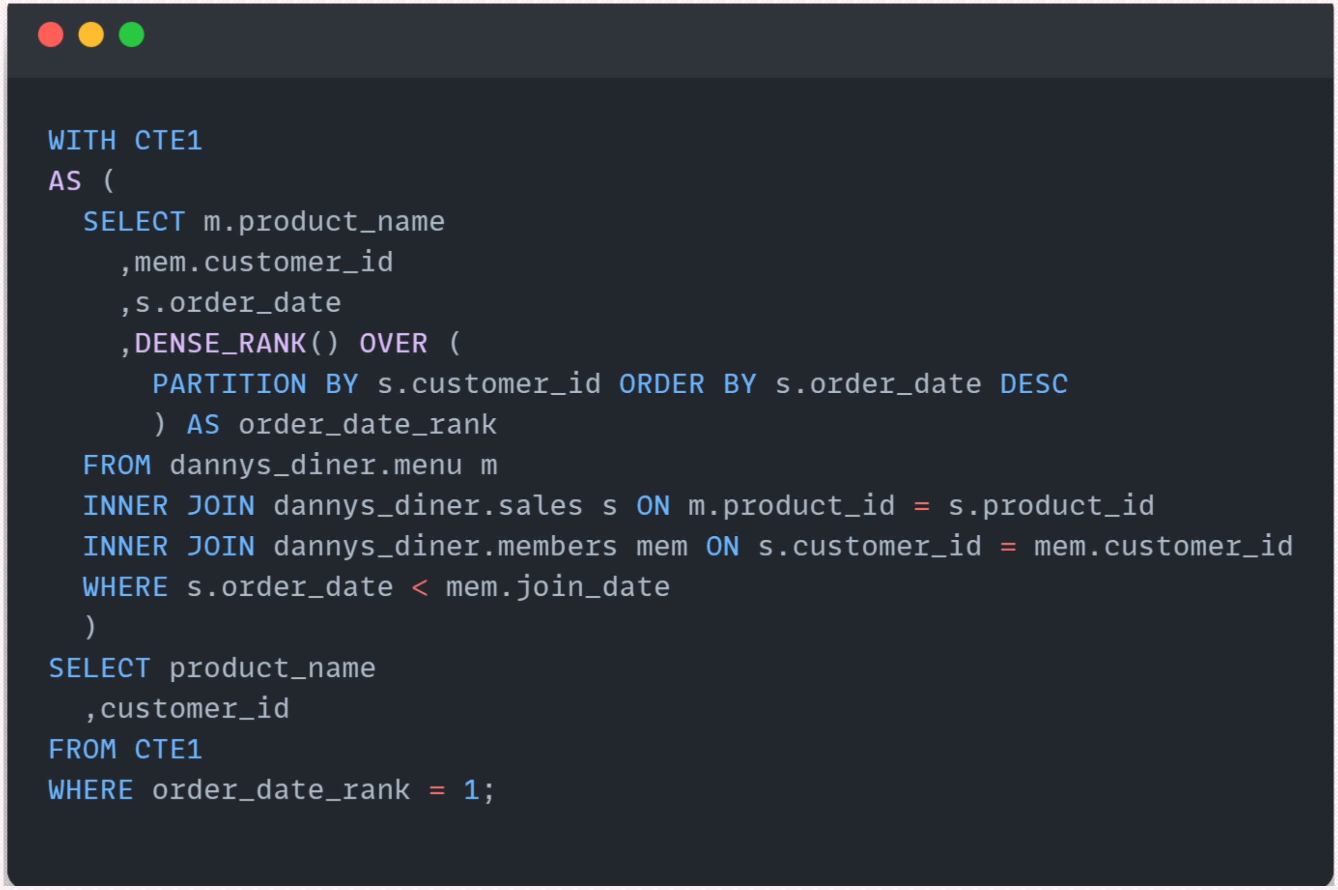
customer_id	product_name
A	ramen
B	sushi

Analysis

1. Once they became members, customer A purchased Ramen and customer B purchased sushi.

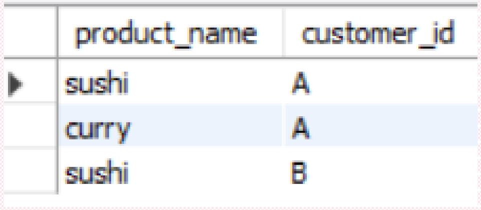
7. Which item was purchased just before the customer became a member?

Query



```
WITH CTE1
AS (
  SELECT m.product_name
    ,mem.customer_id
    ,s.order_date
    ,DENSE_RANK() OVER (
      PARTITION BY s.customer_id ORDER BY s.order_date DESC
      ) AS order_date_rank
  FROM dannys_diner.menu m
  INNER JOIN dannys_diner.sales s ON m.product_id = s.product_id
  INNER JOIN dannys_diner.members mem ON s.customer_id = mem.customer_id
  WHERE s.order_date < mem.join_date
  )
SELECT product_name
  ,customer_id
FROM CTE1
WHERE order_date_rank = 1;
```

Output



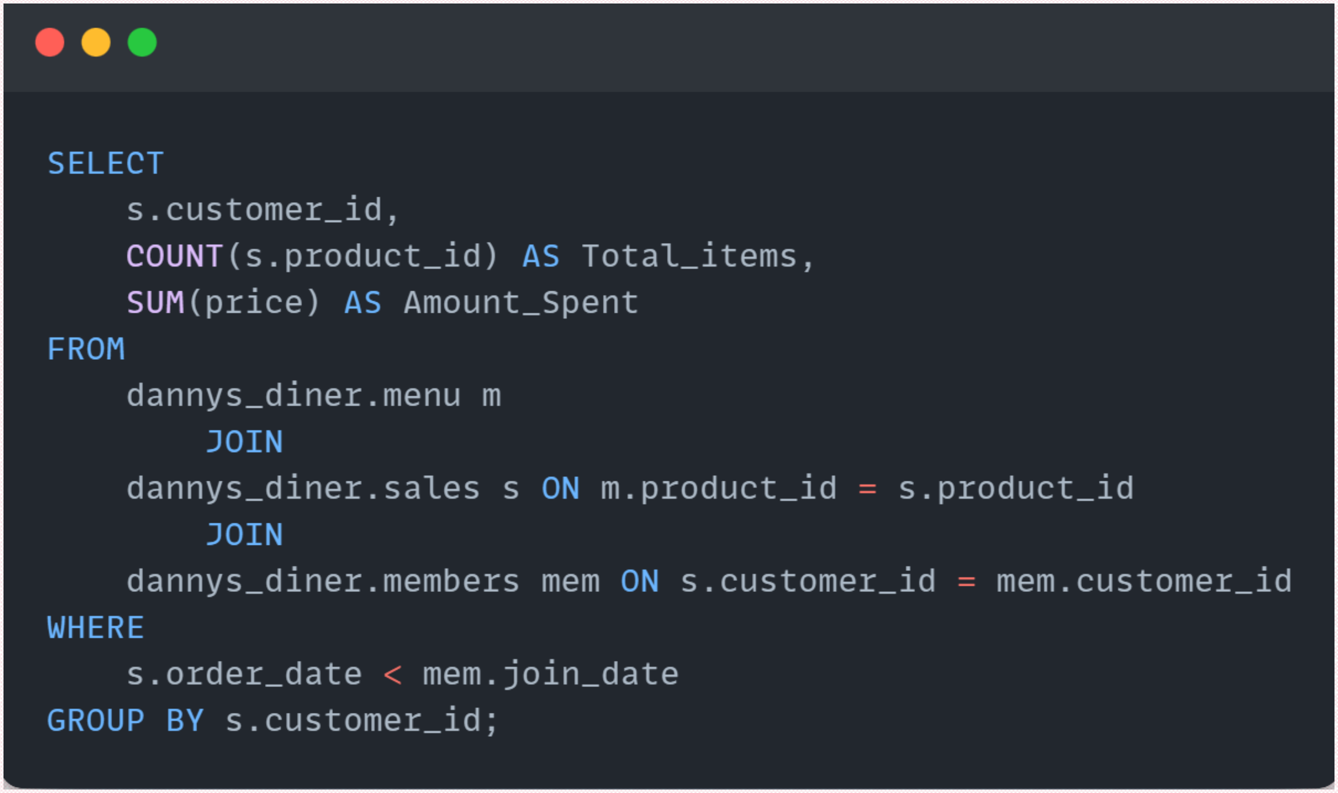
	product_name	customer_id
▶	sushi	A
	curry	A
	sushi	B

Analysis

1. Just before becoming a member, customer A purchased sushi and curry.
2. Customer B purchased sushi.

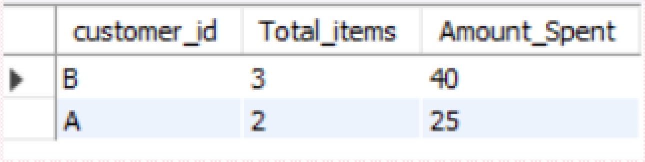
8. What is the total items and amount spent for each member before they became a member?

Query



```
SELECT
    s.customer_id,
    COUNT(s.product_id) AS Total_items,
    SUM(price) AS Amount_Spent
FROM
    dannys_diner.menu m
        JOIN
    dannys_diner.sales s ON m.product_id = s.product_id
        JOIN
    dannys_diner.members mem ON s.customer_id = mem.customer_id
WHERE
    s.order_date < mem.join_date
GROUP BY s.customer_id;
```

Output



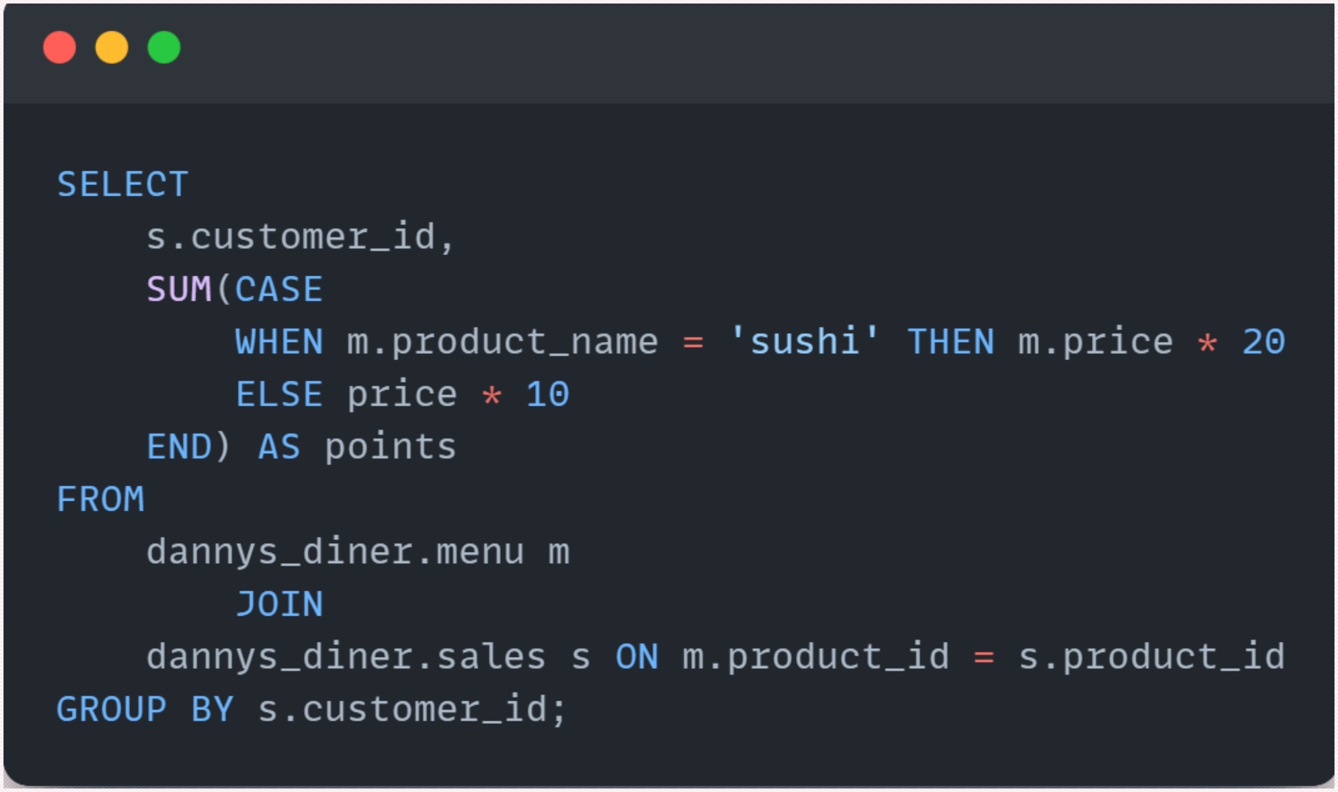
customer_id	Total_items	Amount_Spent
B	3	40
A	2	25

Analysis

1. Customer A has ordered 3 items before becoming a member and spent $40.
2. Customer B has ordered 2 items before becoming a member and spent $40.

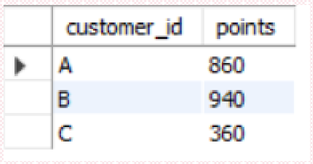
9. If each $1 spent equates to 10 points and sushi has a 2x points multiplier - how many points would each customer have?

Query



```sql
SELECT
    s.customer_id,
    SUM(CASE
        WHEN m.product_name = 'sushi' THEN m.price * 20
        ELSE price * 10
    END) AS points
FROM
    dannys_diner.menu m
        JOIN
    dannys_diner.sales s ON m.product_id = s.product_id
GROUP BY s.customer_id;
```

Output



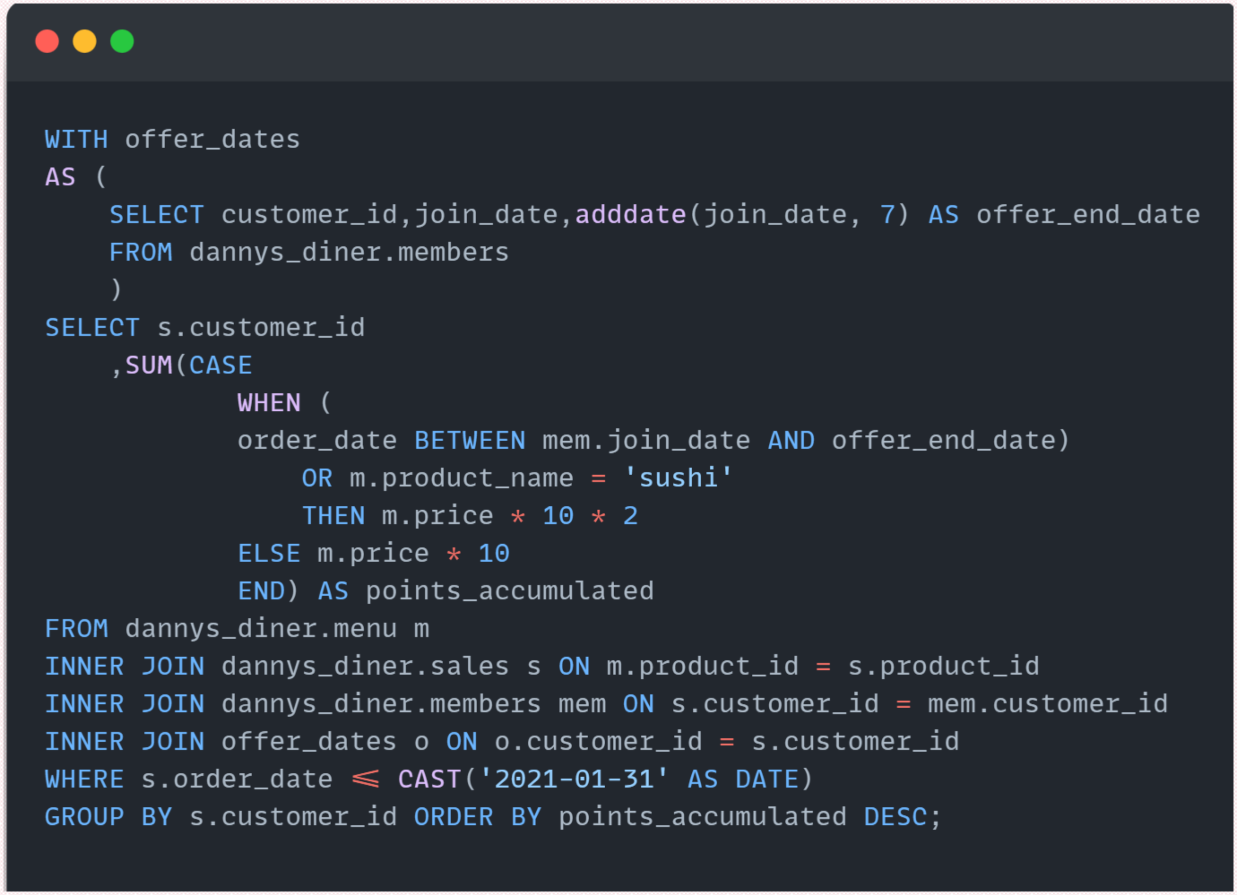
customer_id	points
A	860
B	940
C	360

Analysis

1. Customer B has a total of 940 points followed by customer A with 860 points.
2. Customer C has only 360 points.

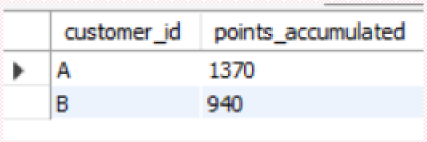
10. In the first week after a customer joins the program (including their join date) they earn 2x points on all items, not just sushi - how many points do customer A and B have at the end of January?

Query



```sql
WITH offer_dates
AS (
    SELECT customer_id,join_date,adddate(join_date, 7) AS offer_end_date
    FROM dannys_diner.members
    )
SELECT s.customer_id
    ,SUM(CASE
            WHEN (
            order_date BETWEEN mem.join_date AND offer_end_date)
                OR m.product_name = 'sushi'
                THEN m.price * 10 * 2
            ELSE m.price * 10
            END) AS points_accumulated
FROM dannys_diner.menu m
INNER JOIN dannys_diner.sales s ON m.product_id = s.product_id
INNER JOIN dannys_diner.members mem ON s.customer_id = mem.customer_id
INNER JOIN offer_dates o ON o.customer_id = s.customer_id
WHERE s.order_date <= CAST('2021-01-31' AS DATE)
GROUP BY s.customer_id ORDER BY points_accumulated DESC;
```

Output



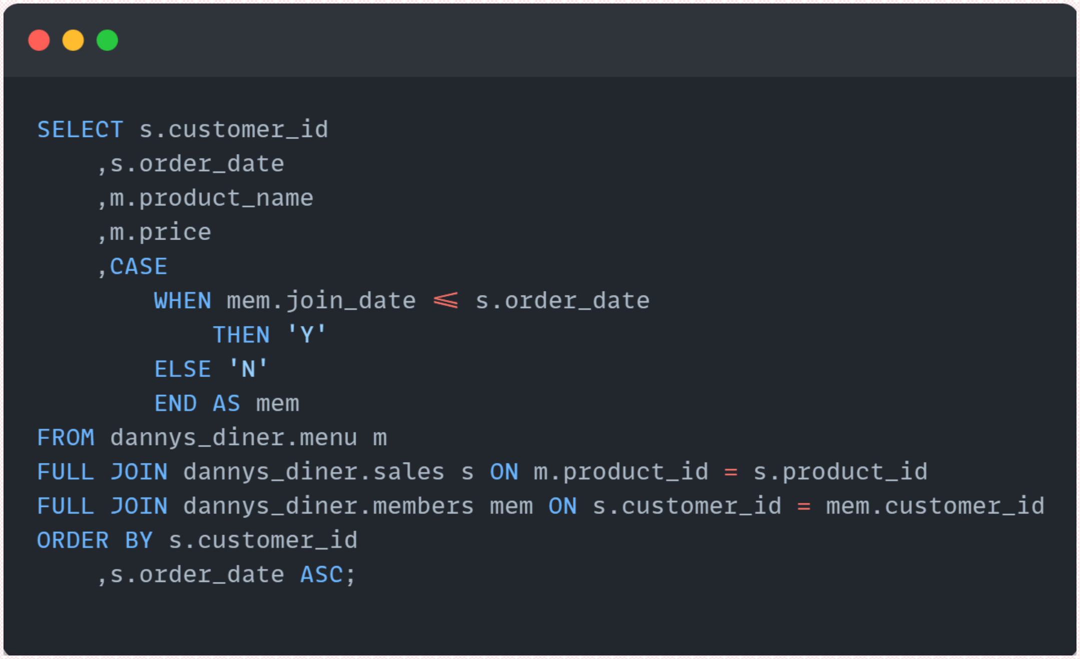
customer_id	points_accumulated
A	1370
B	940

Analysis

1. Customer A has accumulated 1370 points by end of January and customer B has accumulated 940 points.

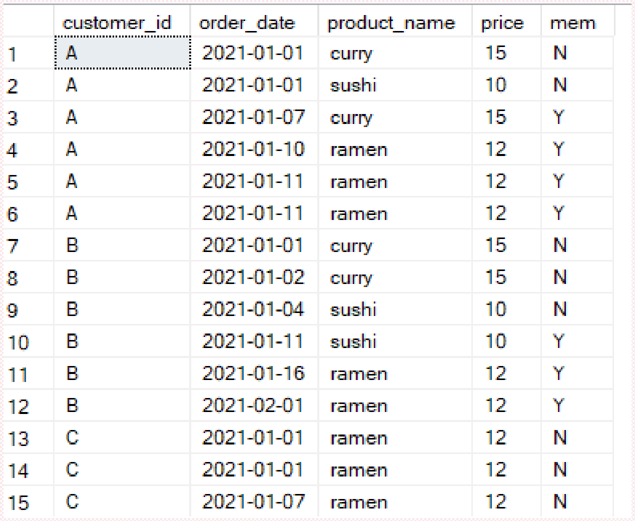
BONUS QUESTIONS
1. Join all tables

Query



```
SELECT s.customer_id
    ,s.order_date
    ,m.product_name
    ,m.price
    ,CASE
        WHEN mem.join_date <= s.order_date
            THEN 'Y'
        ELSE 'N'
        END AS mem
FROM dannys_diner.menu m
FULL JOIN dannys_diner.sales s ON m.product_id = s.product_id
FULL JOIN dannys_diner.members mem ON s.customer_id = mem.customer_id
ORDER BY s.customer_id
    ,s.order_date ASC;
```

Output



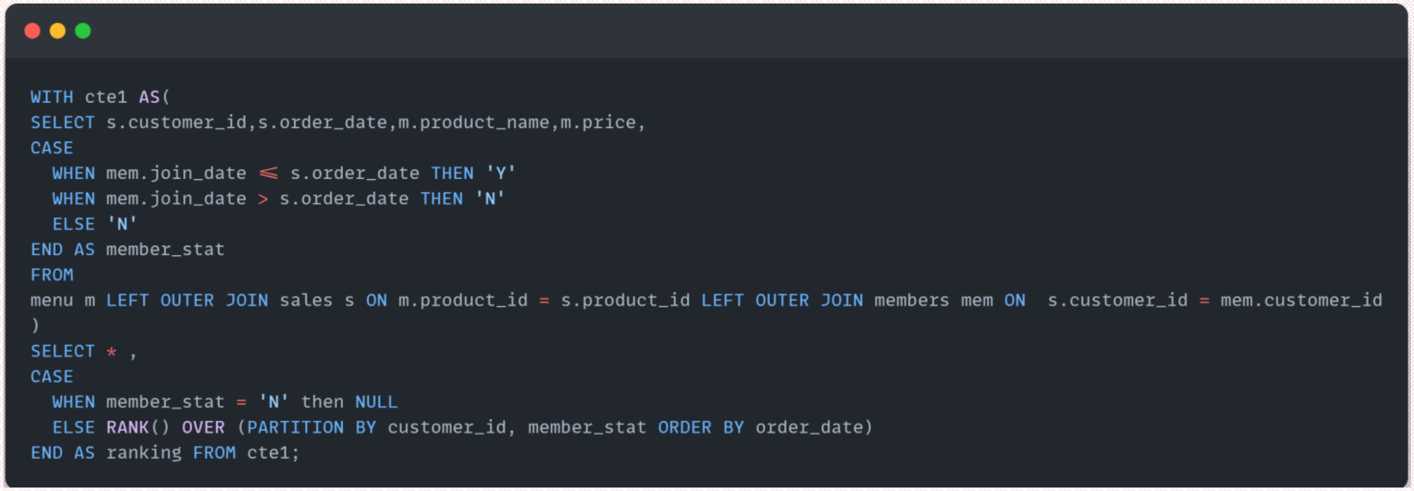
	customer_id	order_date	product_name	price	mem
1	A	2021-01-01	curry	15	N
2	A	2021-01-01	sushi	10	N
3	A	2021-01-07	curry	15	Y
4	A	2021-01-10	ramen	12	Y
5	A	2021-01-11	ramen	12	Y
6	A	2021-01-11	ramen	12	Y
7	B	2021-01-01	curry	15	N
8	B	2021-01-02	curry	15	N
9	B	2021-01-04	sushi	10	N
10	B	2021-01-11	sushi	10	Y
11	B	2021-01-16	ramen	12	Y
12	B	2021-02-01	ramen	12	Y
13	C	2021-01-01	ramen	12	N
14	C	2021-01-01	ramen	12	N
15	C	2021-01-07	ramen	12	N

BONUS QUESTIONS

2. Rank All The Things

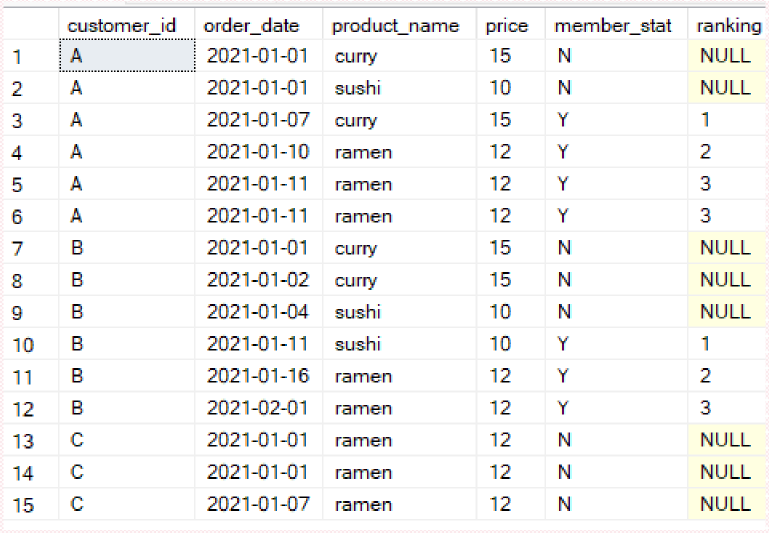
Danny also requires further information about the ranking of customer products, but he purposely does not need the ranking for non-member purchases so he expects null ranking values for the records when customers are not yet part of the loyalty program.

Query



```
WITH cte1 AS(
SELECT s.customer_id,s.order_date,m.product_name,m.price,
CASE
  WHEN mem.join_date <= s.order_date THEN 'Y'
  WHEN mem.join_date > s.order_date THEN 'N'
  ELSE 'N'
END AS member_stat
FROM
menu m LEFT OUTER JOIN sales s ON m.product_id = s.product_id LEFT OUTER JOIN members mem ON  s.customer_id = mem.customer_id
)
SELECT * ,
CASE
  WHEN member_stat = 'N' then NULL
  ELSE RANK() OVER (PARTITION BY customer_id, member_stat ORDER BY order_date)
END AS ranking FROM cte1;
```

Output



	customer_id	order_date	product_name	price	member_stat	ranking
1	A	2021-01-01	curry	15	N	NULL
2	A	2021-01-01	sushi	10	N	NULL
3	A	2021-01-07	curry	15	Y	1
4	A	2021-01-10	ramen	12	Y	2
5	A	2021-01-11	ramen	12	Y	3
6	A	2021-01-11	ramen	12	Y	3
7	B	2021-01-01	curry	15	N	NULL
8	B	2021-01-02	curry	15	N	NULL
9	B	2021-01-04	sushi	10	N	NULL
10	B	2021-01-11	sushi	10	Y	1
11	B	2021-01-16	ramen	12	Y	2
12	B	2021-02-01	ramen	12	Y	3
13	C	2021-01-01	ramen	12	N	NULL
14	C	2021-01-01	ramen	12	N	NULL
15	C	2021-01-07	ramen	12	N	NULL