Case Study #1

Danny's Dinner

DANNY'S DINER

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 favorite 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 Statements

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 favorite. Having this deeper connection with his customers will help him deliver a better and more personalized 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 with a sample of his overall customer data due to privacy issues - but he hopes that these examples are enough 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

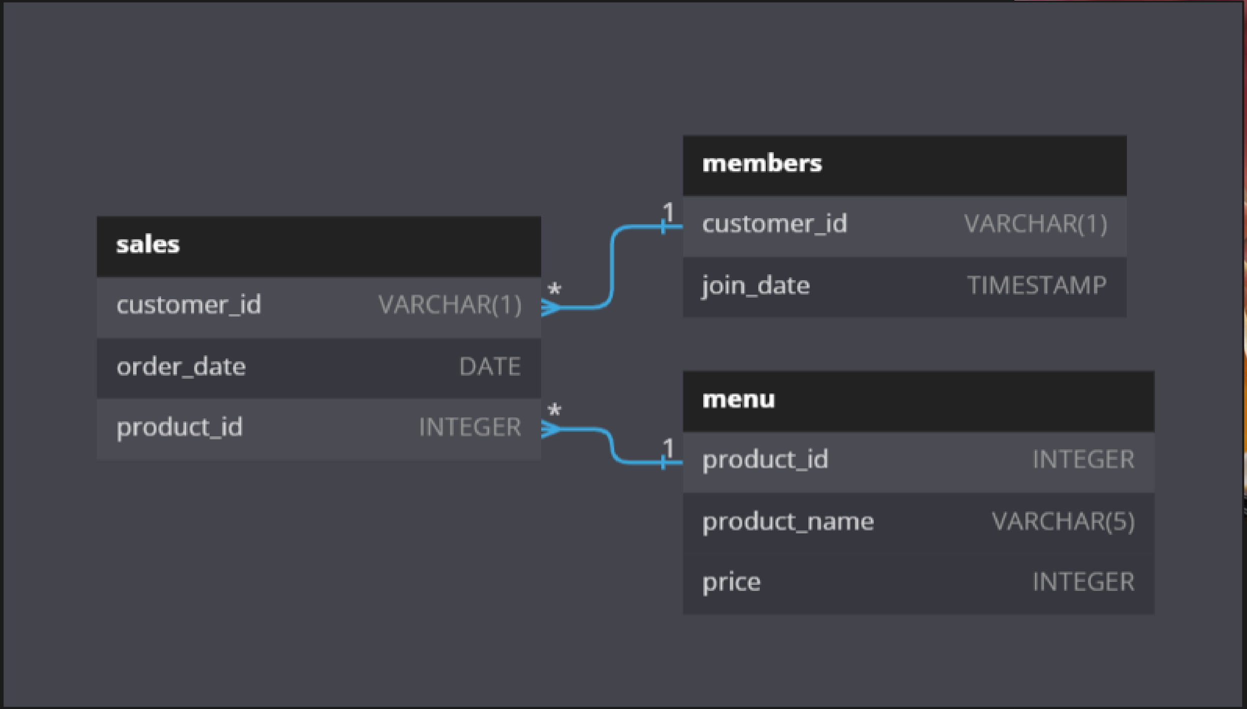
All datasets exist within the dannys_diner database schema

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.

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

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

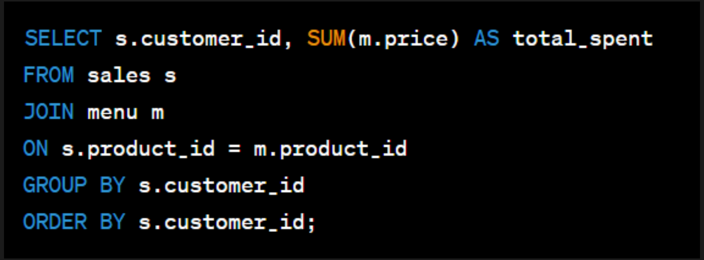
Entity Relationship Diagram



Case Study Questions

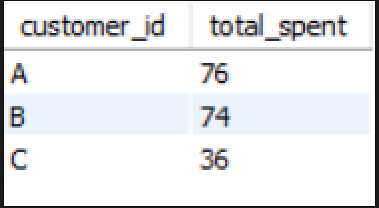
Q1. What is the total amount each customer spent at the restaurant?

Query



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

Output

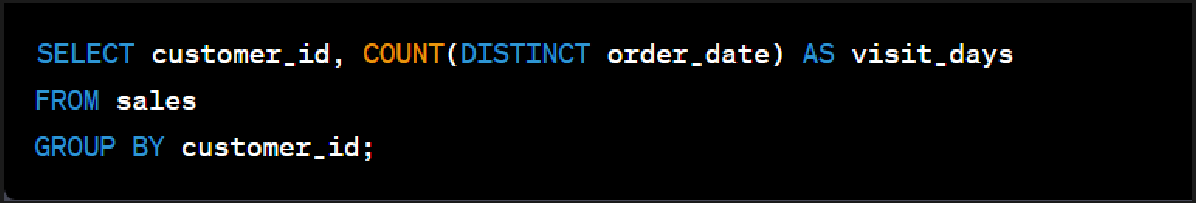


customer_id	total_spent
A	76
B	74
C	36

- Customer A spent $76.
- Customer B spent $74.
- Customer C spent $36

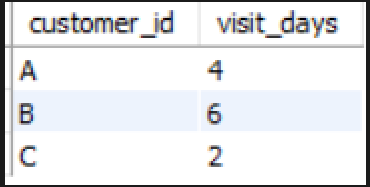
Q2. How many days has each customer visited the restaurant?

Query



```sql
SELECT customer_id, COUNT(DISTINCT order_date) AS visit_days
FROM sales
GROUP BY customer_id;
```

Output

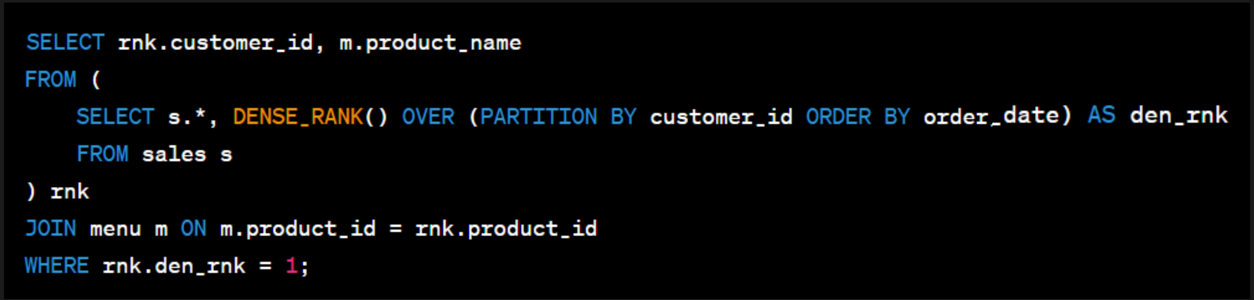


customer_id	visit_days
A	4
B	6
C	2

- Customer A visited 4 times.
- Customer B visited 6 times.
- Customer C visited 2 times.

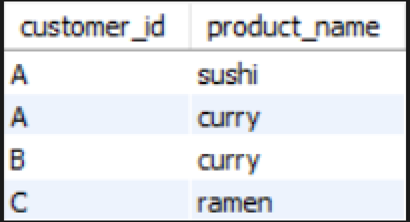
Q3. What was the first item from the menu purchased by each customer?

Query



```sql
SELECT rnk.customer_id, m.product_name
FROM (
    SELECT s.*, DENSE_RANK() OVER (PARTITION BY customer_id ORDER BY order_date) AS den_rnk
    FROM sales s
) rnk
JOIN menu m ON m.product_id = rnk.product_id
WHERE rnk.den_rnk = 1;
```

Output



customer_id	product_name
A	sushi
A	curry
B	curry
C	ramen

- Customer A's first order are curry and sushi, B's is curry and C's is ramen

Q4. What is the most purchased item on the menu and how many times was it purchased by all customers?

Query

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

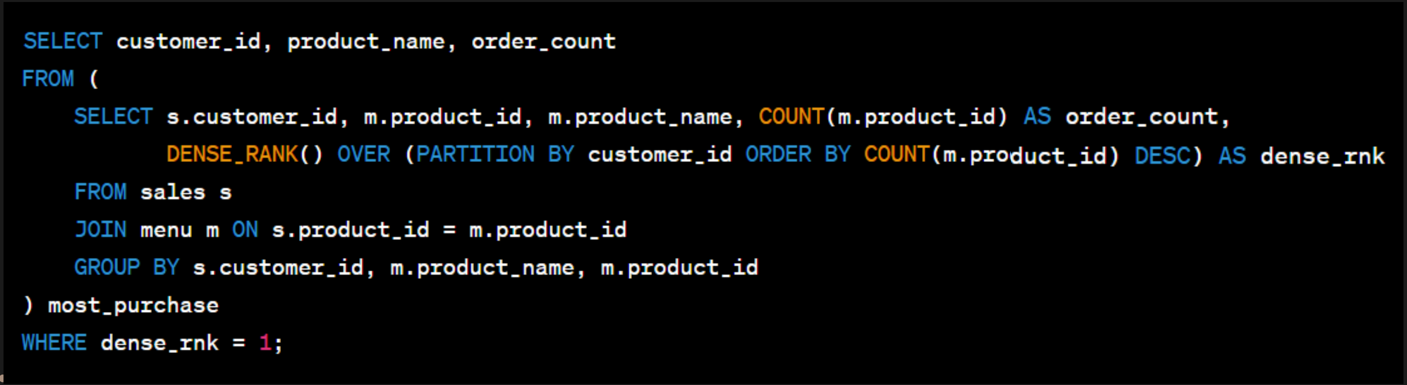
Output

product_name	purchased_times
ramen	8

- The most purchased item on the menu is ramen which is 8 times.

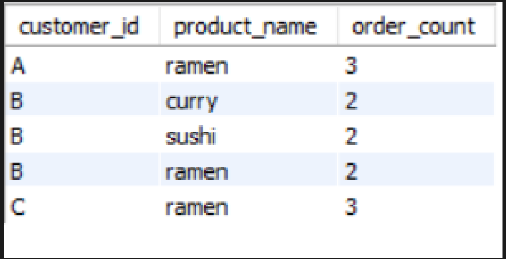
Q5. Which item was the most popular for each customer?

Query



```sql
SELECT customer_id, product_name, order_count
FROM (
    SELECT s.customer_id, m.product_id, m.product_name, COUNT(m.product_id) AS order_count,
           DENSE_RANK() OVER (PARTITION BY customer_id ORDER BY COUNT(m.product_id) DESC) AS dense_rnk
    FROM sales s
    JOIN menu m ON s.product_id = m.product_id
    GROUP BY s.customer_id, m.product_name, m.product_id
) most_purchase
WHERE dense_rnk = 1;
```

Output

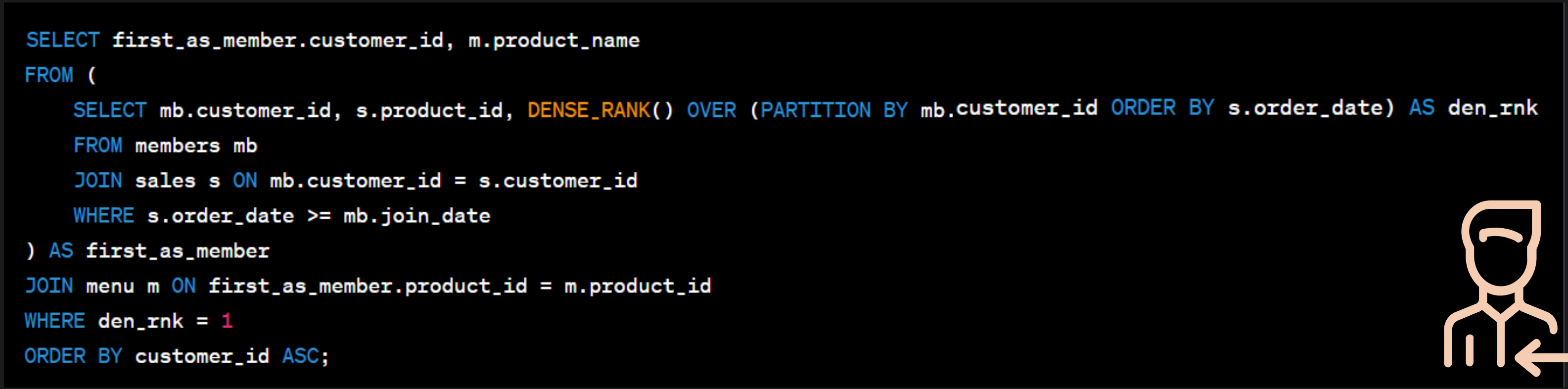


customer_id	product_name	order_count
A	ramen	3
B	curry	2
B	sushi	2
B	ramen	2
C	ramen	3

- Customer A and C's favorite item is ramen and Customer B enjoys all items on the menu.

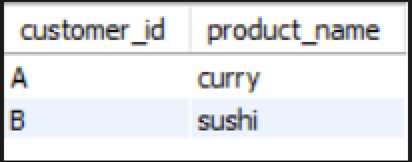
Q6. Which item was purchased first by the customer after they became a member?

Query



```
SELECT first_as_member.customer_id, m.product_name
FROM (
    SELECT mb.customer_id, s.product_id, DENSE_RANK() OVER (PARTITION BY mb.customer_id ORDER BY s.order_date) AS den_rnk
    FROM members mb
    JOIN sales s ON mb.customer_id = s.customer_id
    WHERE s.order_date >= mb.join_date
) AS first_as_member
JOIN menu m ON first_as_member.product_id = m.product_id
WHERE den_rnk = 1
ORDER BY customer_id ASC;
```

Output



customer_id	product_name
A	curry
B	sushi

- Customer A’s first order as a member is ramen and customer B’s is sushi.

Q7. Which item was purchased just before the customer became a member?

Query

```
SELECT just_before_member.customer_id, m.product_name
FROM (
    SELECT mb.customer_id, s.product_id, DENSE_RANK() OVER (PARTITION BY mb.customer_id ORDER BY s.order_date DESC) AS den_rnk
    FROM dannys_diner.members mb
    JOIN sales s ON mb.customer_id = s.customer_id
    WHERE s.order_date < mb.join_date
) AS just_before_member
JOIN menu m ON just_before_member.product_id = m.product_id
WHERE den_rnk = 1
ORDER BY just_before_member.customer_id ASC;
```

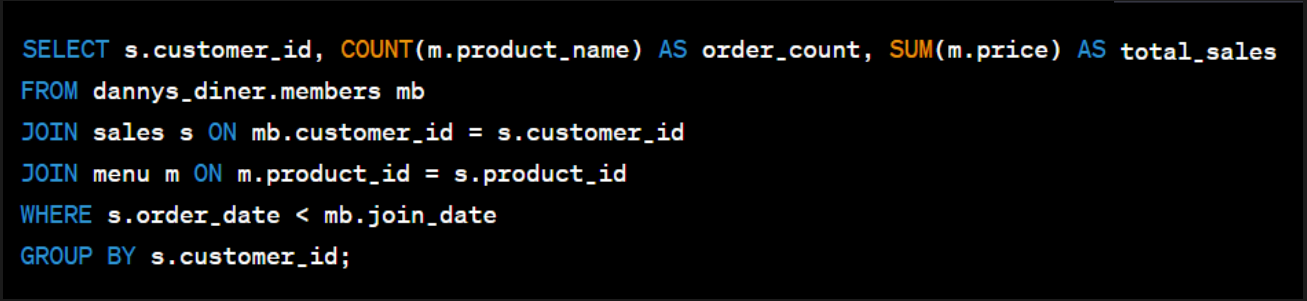
Output

customer_id	product_name
A	sushi
A	curry
B	sushi

- Both customers purchase sushi for there last order before becoming members are sushi.

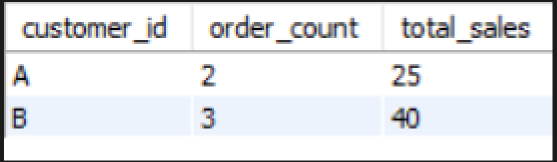
Q8. What is the total items and amount spent for each member before they became a member?

Query



```sql
SELECT s.customer_id, COUNT(m.product_name) AS order_count, SUM(m.price) AS total_sales
FROM dannys_diner.members mb
JOIN sales s ON mb.customer_id = s.customer_id
JOIN menu m ON m.product_id = s.product_id
WHERE s.order_date < mb.join_date
GROUP BY s.customer_id;
```

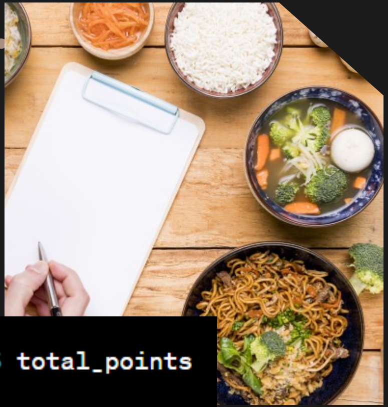
Output



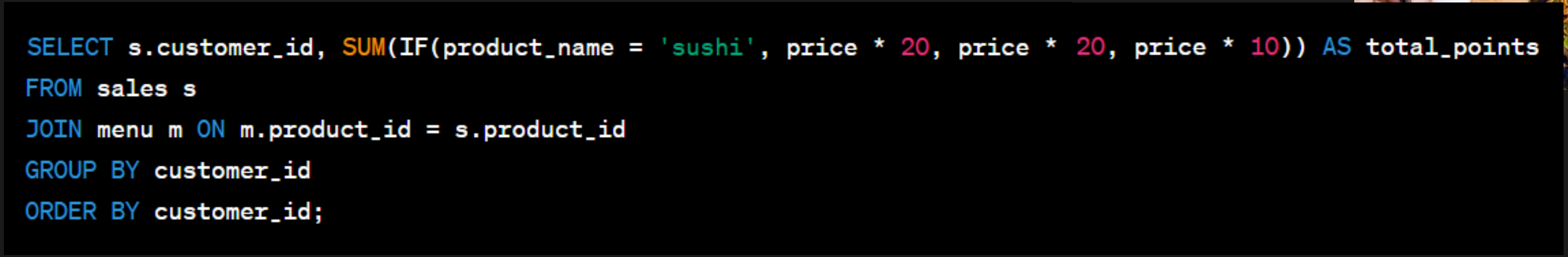
customer_id	order_count	total_sales
A	2	25
B	3	40

- Before becoming members, Customer A spent $25 on 2 items, Customer B spent $40 on 3 items.

Q9. If each $1 spent equates to 10 points and sushi has a 2x points multiplier - how many points would each customer have?

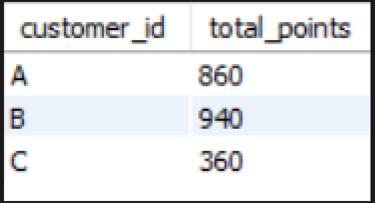


Query



```
SELECT s.customer_id, SUM(IF(product_name = 'sushi', price * 20, price * 20, price * 10)) AS total_points
FROM sales s
JOIN menu m ON m.product_id = s.product_id
GROUP BY customer_id
ORDER BY customer_id;
```

Output

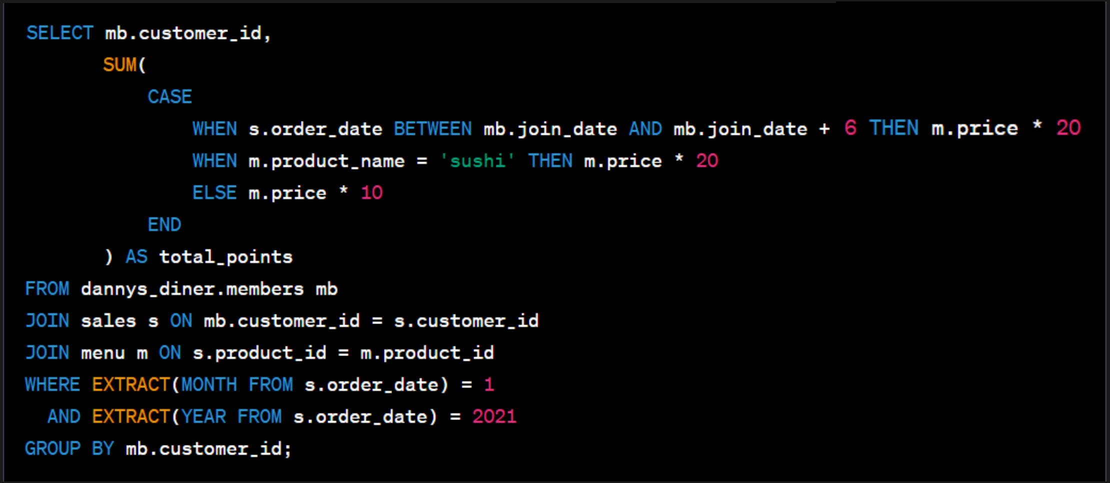


customer_id	total_points
A	860
B	940
C	360

- The total points for Customers A, B and C are $860, $940 and $360.

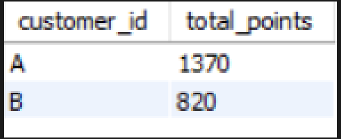
Q10. In the first week after a customer joins the program (including their join date) they earn 2x points on all items, not just sushi - how many points do customer A and B have at the end of January?

Query



```
SELECT mb.customer_id,
       SUM(
           CASE
               WHEN s.order_date BETWEEN mb.join_date AND mb.join_date + 6 THEN m.price * 20
               WHEN m.product_name = 'sushi' THEN m.price * 20
               ELSE m.price * 10
           END
       ) AS total_points
FROM dannys_diner.members mb
JOIN sales s ON mb.customer_id = s.customer_id
JOIN menu m ON s.product_id = m.product_id
WHERE EXTRACT(MONTH FROM s.order_date) = 1
  AND EXTRACT(YEAR FROM s.order_date) = 2021
GROUP BY mb.customer_id;
```

Output

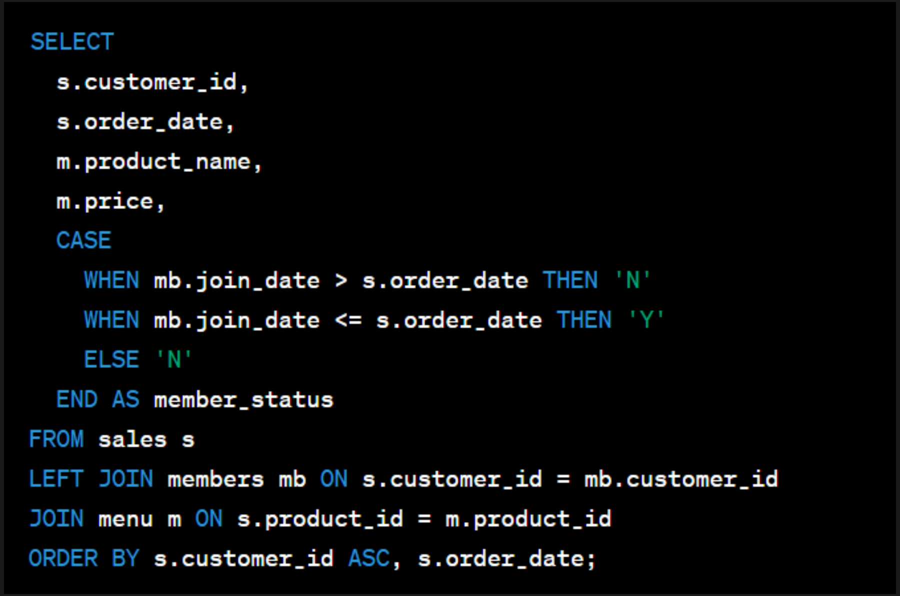


customer_id	total_points
A	1370
B	820

- Customer A has 1,370 points and customer B has 820 points.

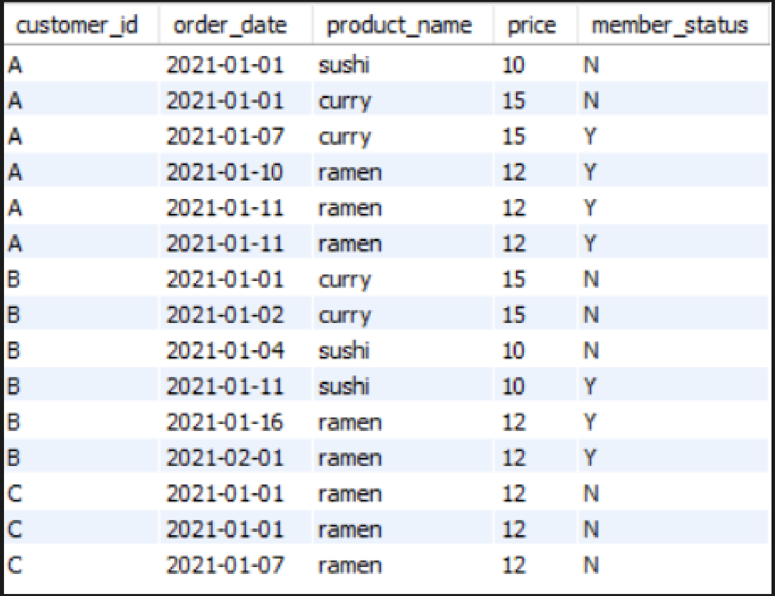
We decided to recreate a basic data table that Danny and his team can use to quickly derive insights without needing to join the underlying tables using SQL

Query



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

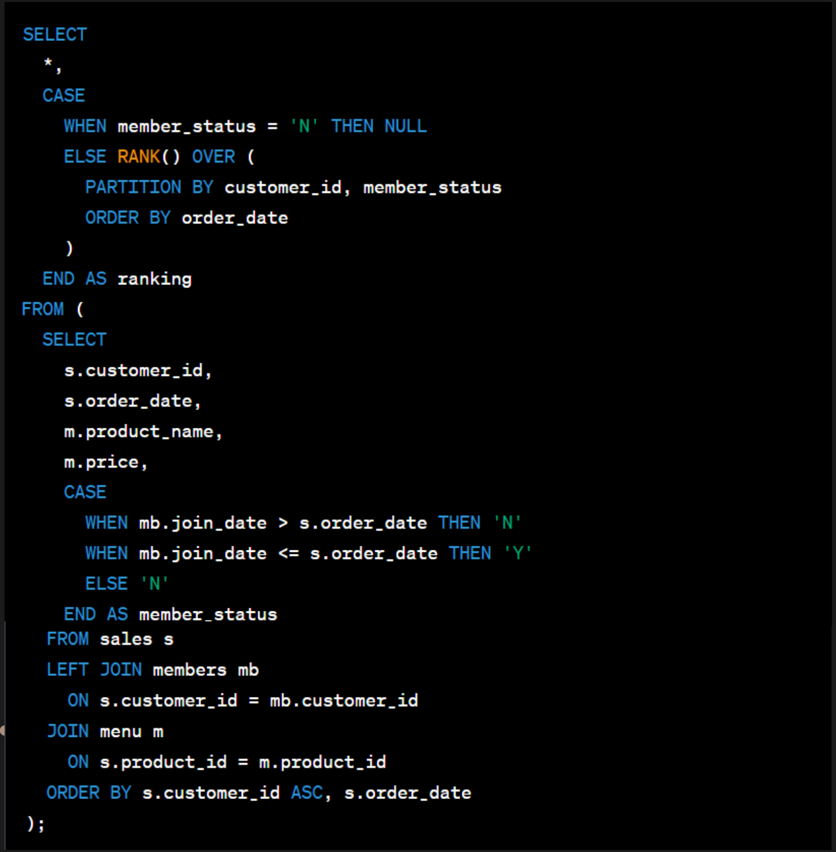
Output



customer_id	order_date	product_name	price	member_status
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

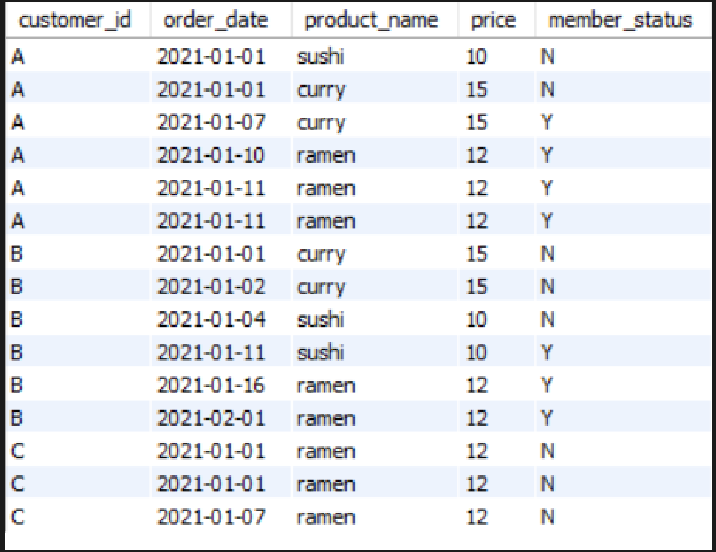
Danny also requires further information about the ranking of customer products, but he purposely does not need the ranking for non-member purchases so he expects null ranking values for the records when customers are not yet part of the loyalty program.

Query



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

Output



customer_id	order_date	product_name	price	member_status
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

Insights from the analysis

- Customer A is the person who spent the most money to purchase food from Danny's Diner.
- Customer B is the most frequent visitor with 6 visits.
- Danny's Diner's most popular item is ramen, followed by curry and sushi.
- Customer A and C ordered ramen mostly whereas Customer B seems to enjoy sushi, curry and ramen equally.
- Customer A became 1st member of Danny's Diner and his first order is curry.
- Secondly, B become a member and C has not become a member until the end of January 2021.
- The last item ordered by Customers A and B before they became members are sushi and curry.
- Before they became members, both Customers A and B spent $25 and $40.
- Throughout Jan 2021, their points for Customer A: 860, Customer B: 940 and Customer C: 360.
- By the end of January 2021, Customer A received 1370 points and Customer B received 820 points, assuming that members can earn 2x a week starting on the day they joined, with an additional 2x points for sushi