# Danny's Diner

SQL Case Study #1

## Table of Contents

## 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**. However, **Danny's Diner** is facing some challenges in managing its operations and finances. The restaurant has collected some basic data from its first few months of business, but it does not know how to analyze and use the data to improve its performance and profitability.

**Danny's Diner** needs help to turn its data into insights and actions that can help the restaurant grow.

## Problem Statement

**Danny** is interested in learning more about his customers’ behavior and preferences, such as their **visit frequency**, **spending amount**, and **favorite menu items**. He believes that having a deeper understanding of his customers will enable him to provide a better and more personalized service for his loyal patrons.

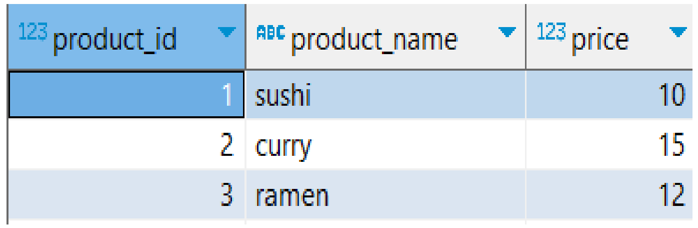
He intends to use these insights to evaluate whether he should expand his existing customer loyalty program - he also requires assistance to create some basic datasets that his team can easily access and examine the data without needing to use SQL.

# Datasets

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

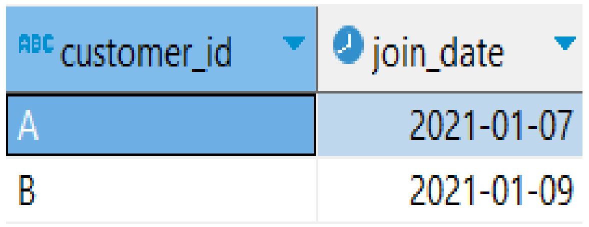
- `sales`
- `menu`
- `members`

Menu



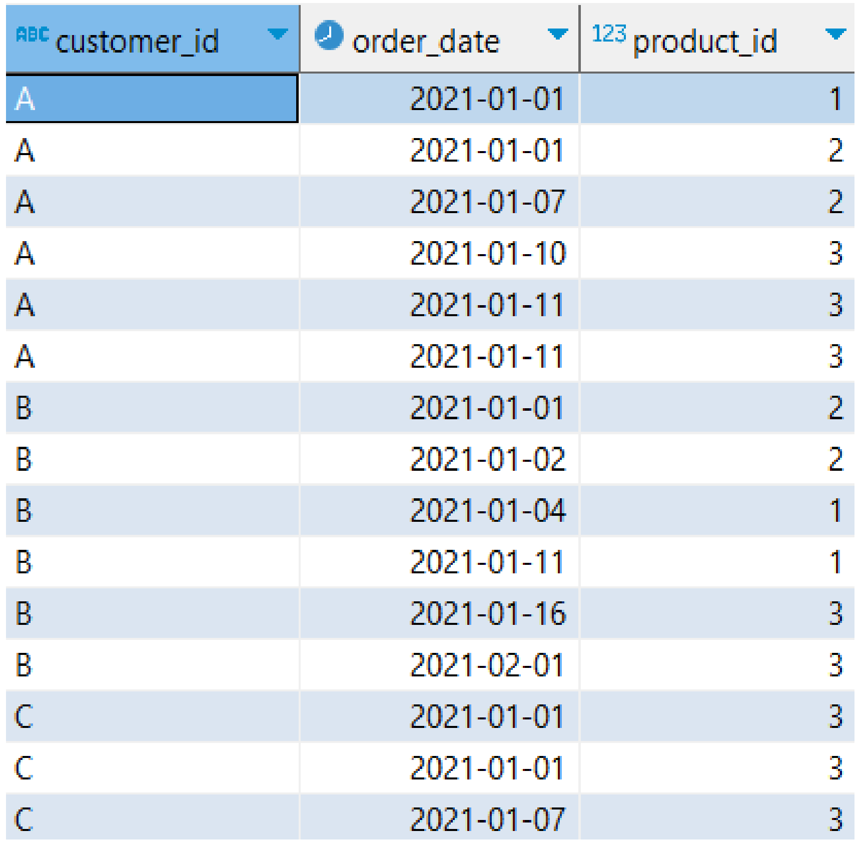
| product_id | product_name | price |
|---|---|---|
| 1 | sushi | 10 |
| 2 | curry | 15 |
| 3 | ramen | 12 |

Members



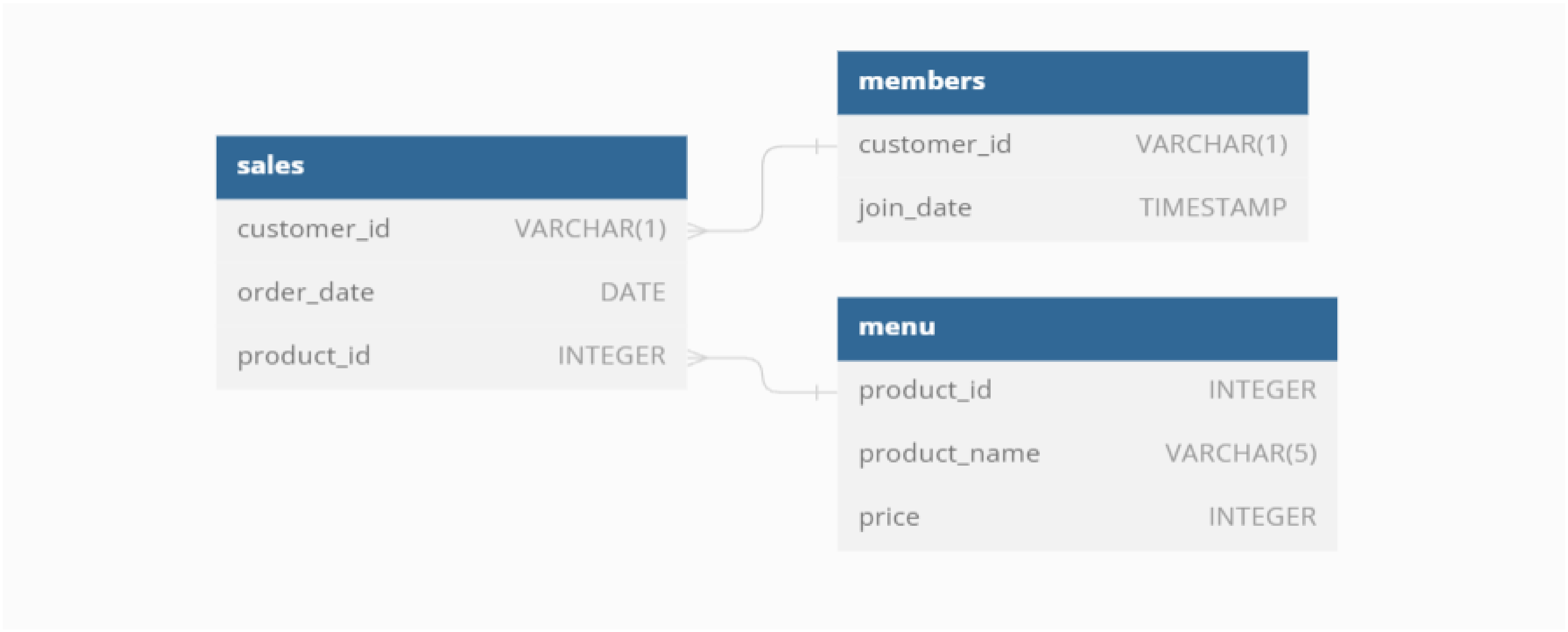
| customer_id | join_date |
|---|---|
| A | 2021-01-07 |
| B | 2021-01-09 |

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 |

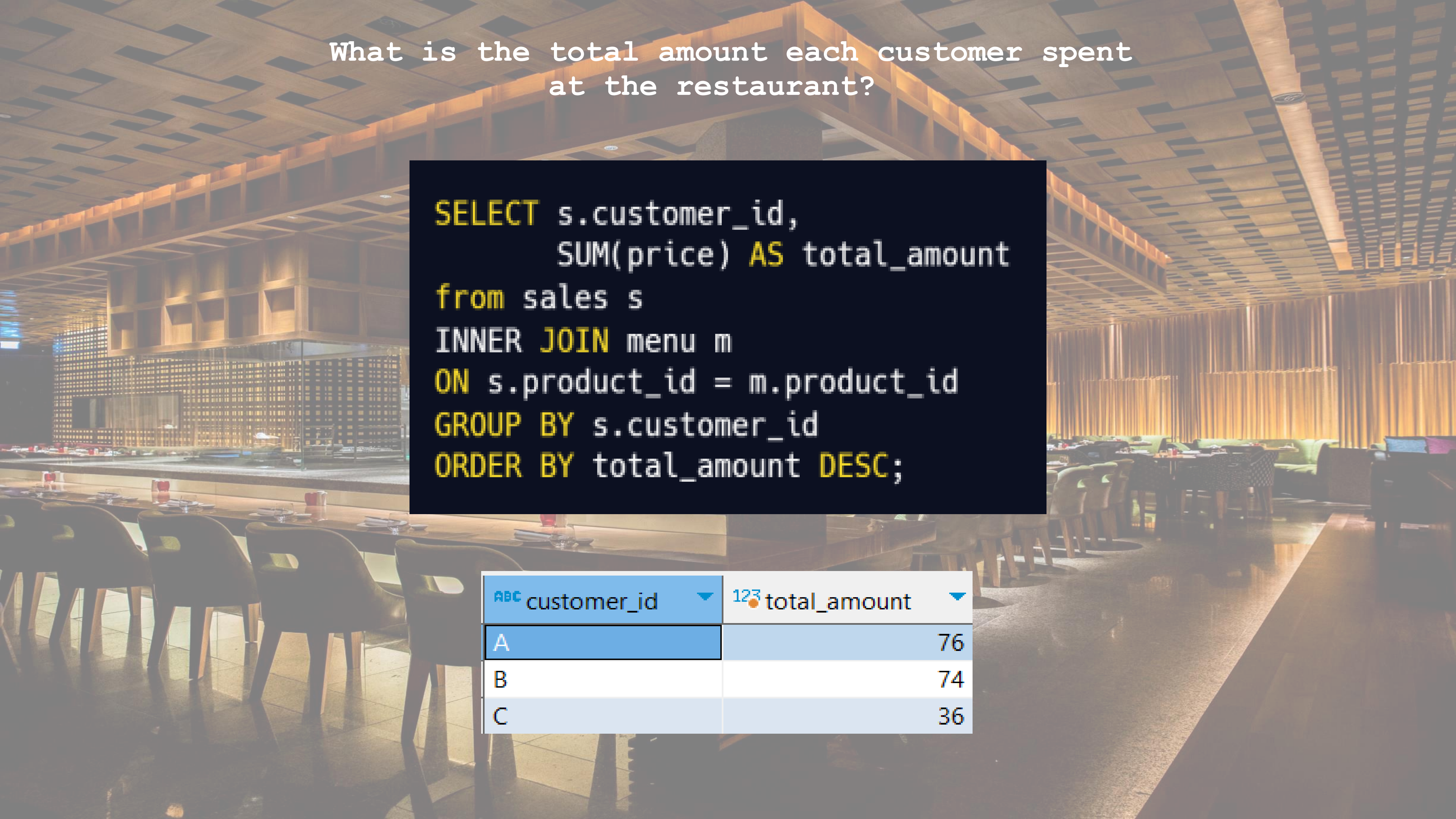
# Entity Relationship Diagram

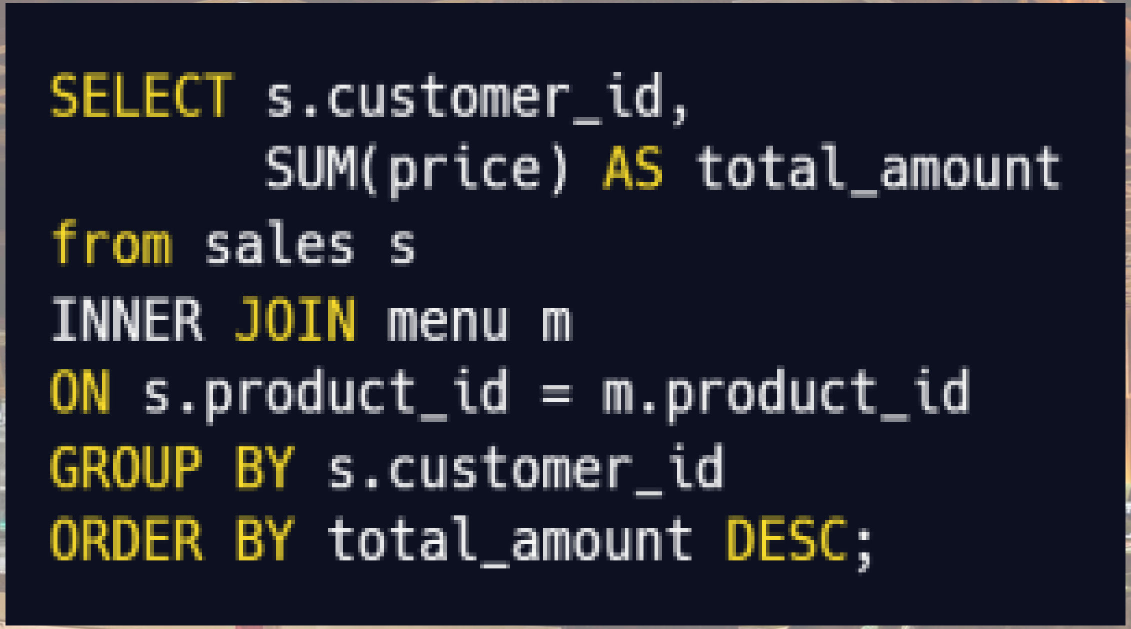


## Deriving Insights

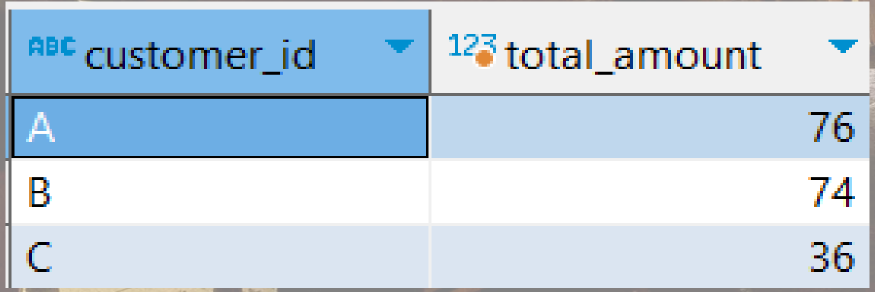
RDBMS used: PostgreSQL

## What is the total amount each customer spent at the restaurant?

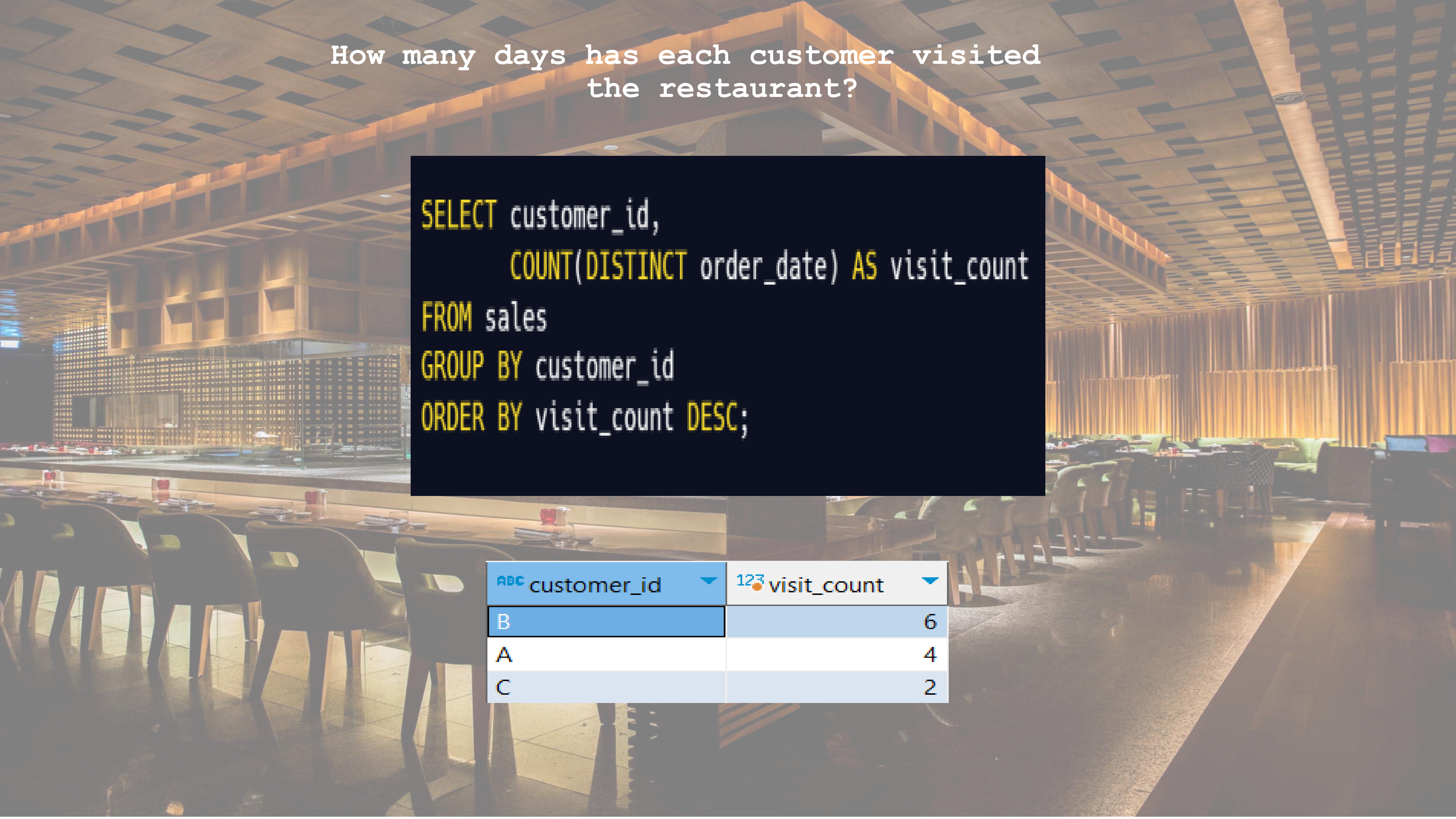




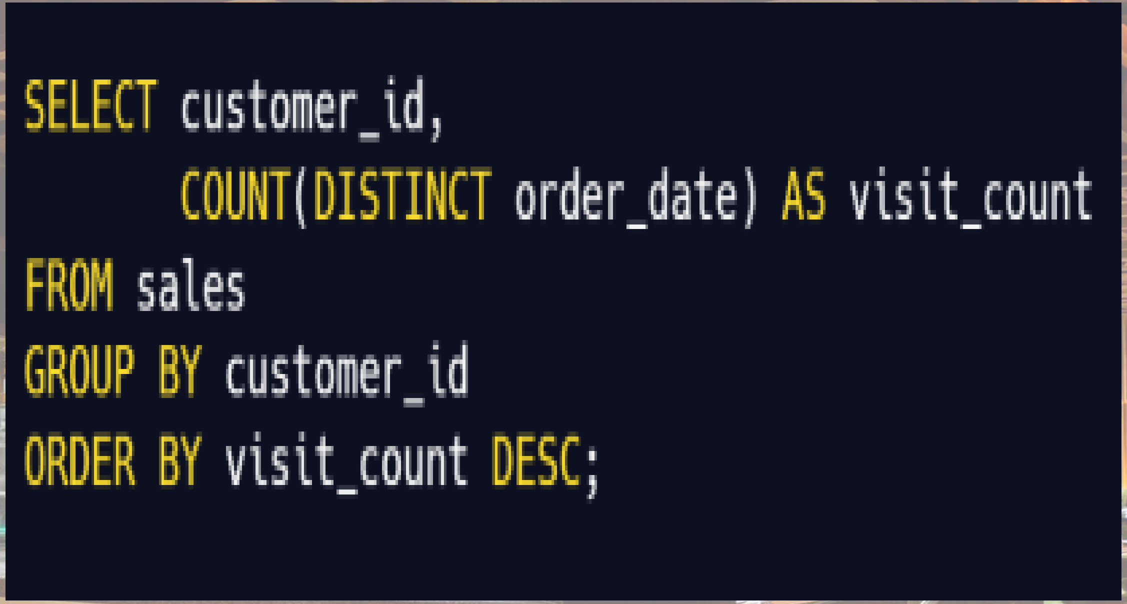
```
SELECT s.customer_id,
       SUM(price) AS total_amount
from sales s
INNER JOIN menu m
ON s.product_id = m.product_id
GROUP BY s.customer_id
ORDER BY total_amount DESC;
```



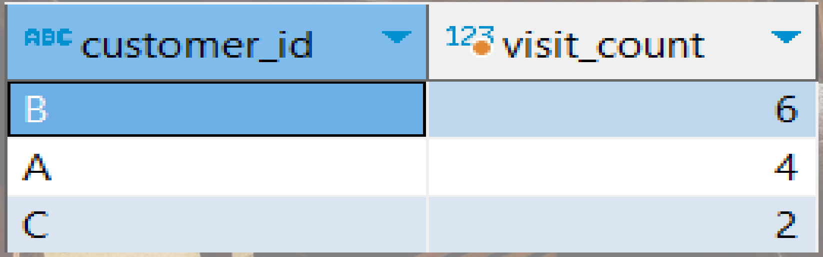
| customer_id | total_amount |
|---|---|
| A | 76 |
| B | 74 |
| C | 36 |



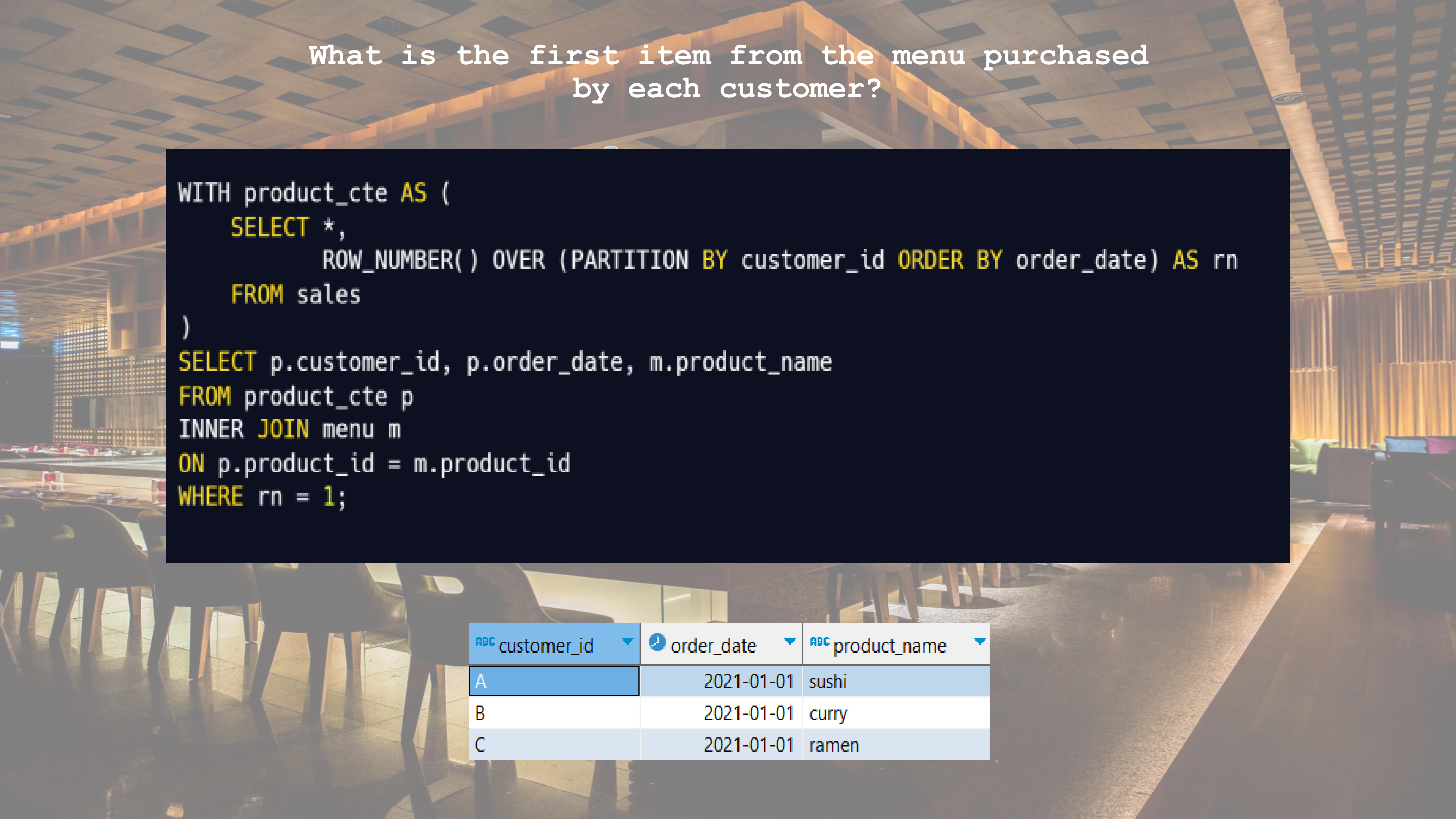
## How many days has each customer visited the restaurant?



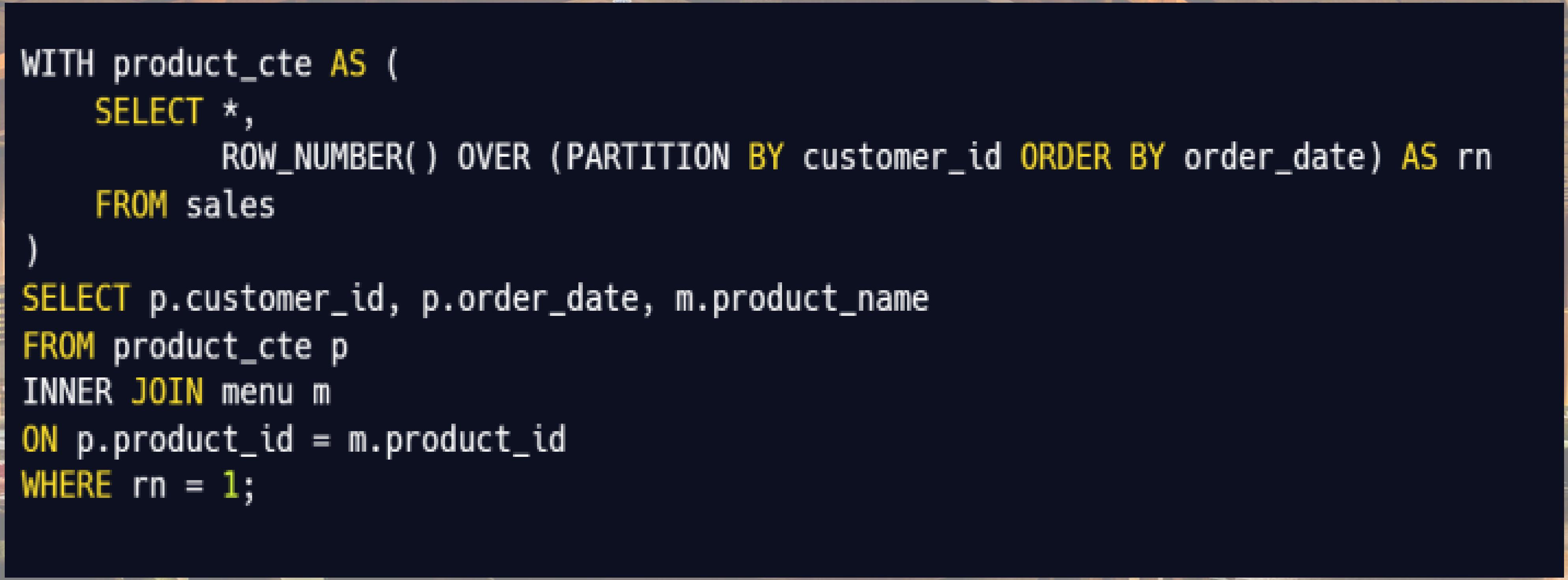
```
SELECT customer_id,
       COUNT(DISTINCT order_date) AS visit_count
FROM sales
GROUP BY customer_id
ORDER BY visit_count DESC;
```



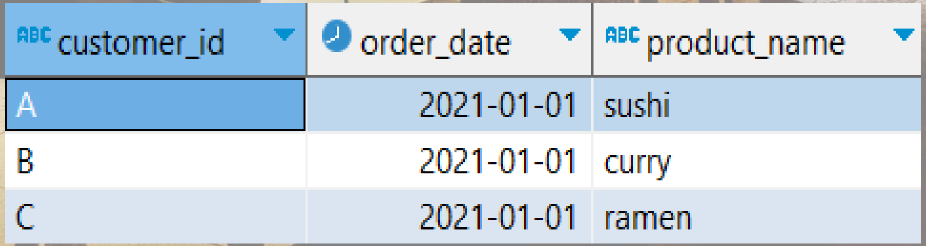
| customer_id | visit_count |
|---|---|
| B | 6 |
| A | 4 |
| C | 2 |



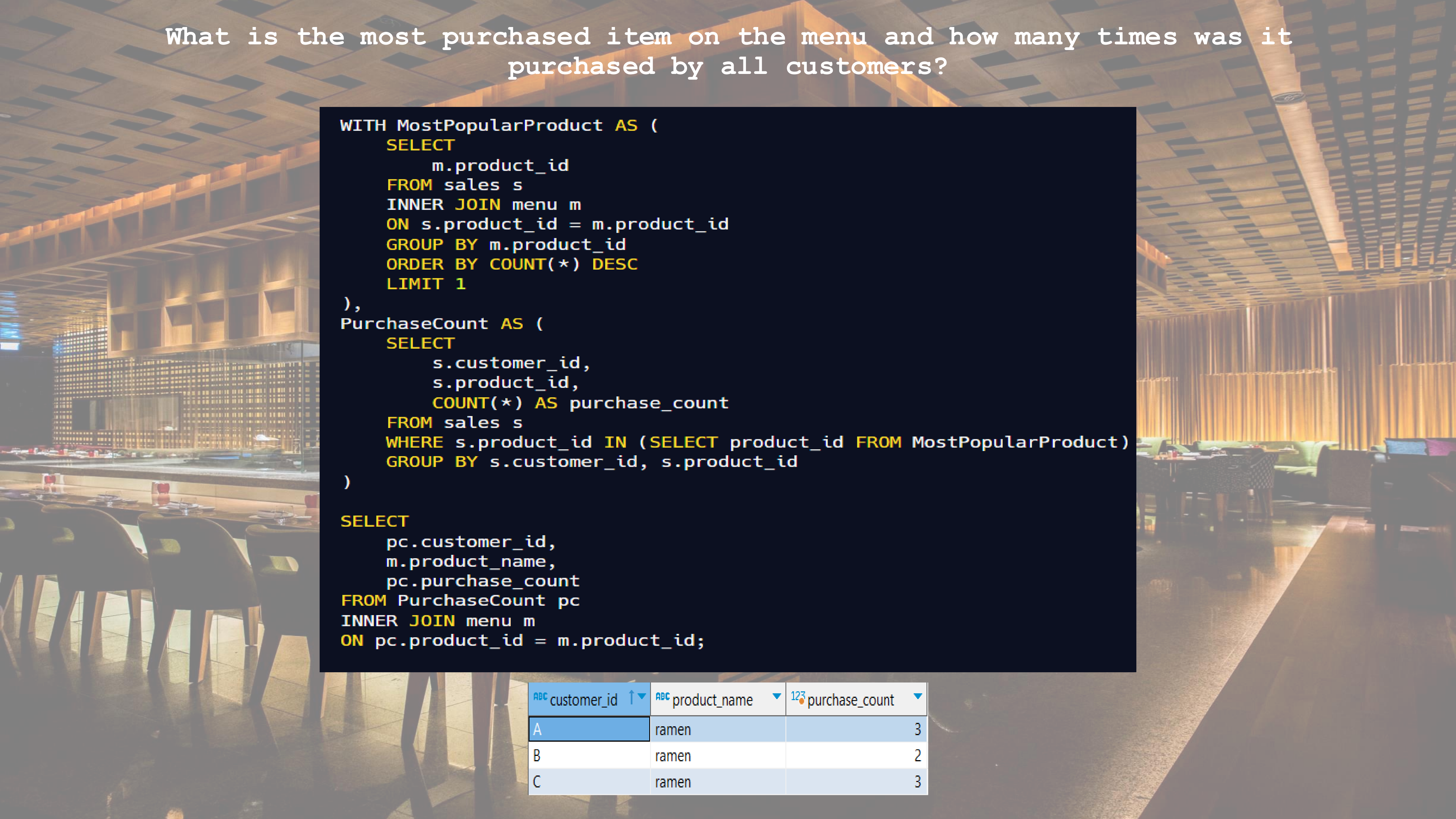
## What is the first item from the menu purchased by each customer?



```sql
WITH product_cte AS (
    SELECT *,
           ROW_NUMBER( ) OVER (PARTITION BY customer_id ORDER BY order_date) AS rn
    FROM sales
)
SELECT p.customer_id, p.order_date, m.product_name
FROM product_cte p
INNER JOIN menu m
ON p.product_id = m.product_id
WHERE rn = 1;
```



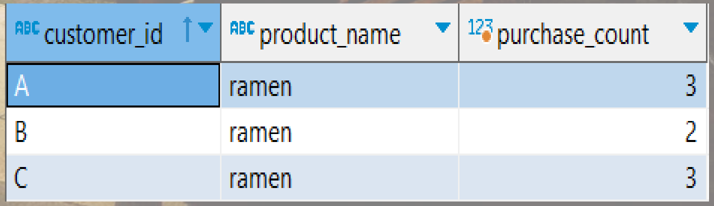
| customer_id | order_date | product_name |
|---|---|---|
| A | 2021-01-01 | sushi |
| B | 2021-01-01 | curry |
| C | 2021-01-01 | ramen |



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

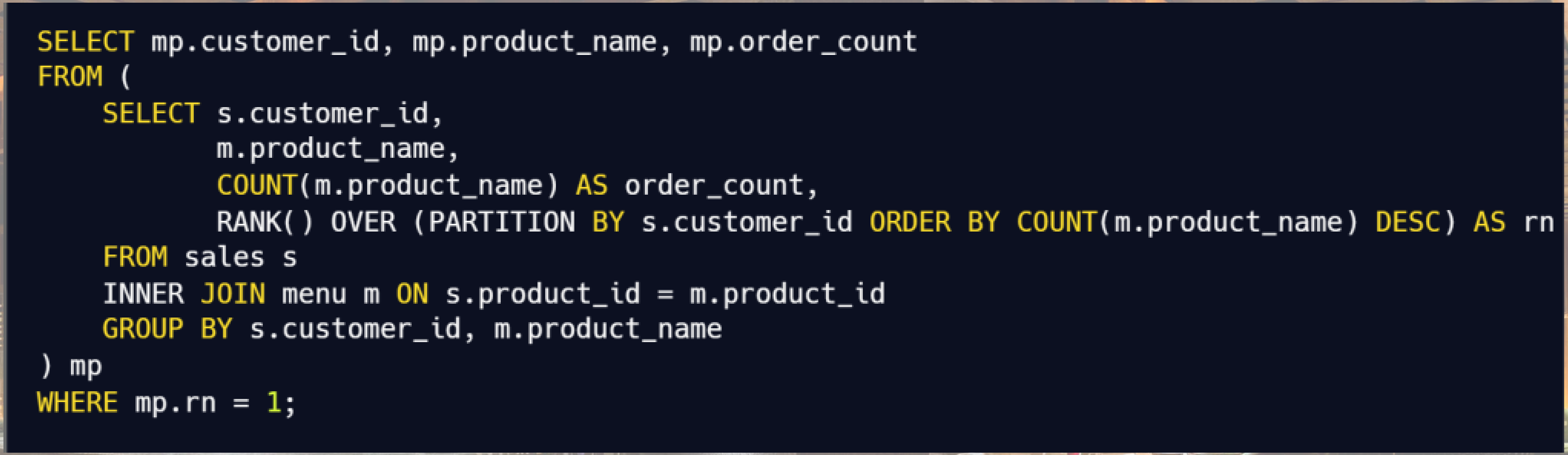
```sql
WITH MostPopularProduct AS (
    SELECT
        m.product_id
    FROM sales s
    INNER JOIN menu m
    ON s.product_id = m.product_id
    GROUP BY m.product_id
    ORDER BY COUNT(*) DESC
    LIMIT 1
),
PurchaseCount AS (
    SELECT
        s.customer_id,
        s.product_id,
        COUNT(*) AS purchase_count
    FROM sales s
    WHERE s.product_id IN (SELECT product_id FROM MostPopularProduct)
    GROUP BY s.customer_id, s.product_id
)

SELECT
    pc.customer_id,
    m.product_name,
    pc.purchase_count
FROM PurchaseCount pc
INNER JOIN menu m
ON pc.product_id = m.product_id;
```

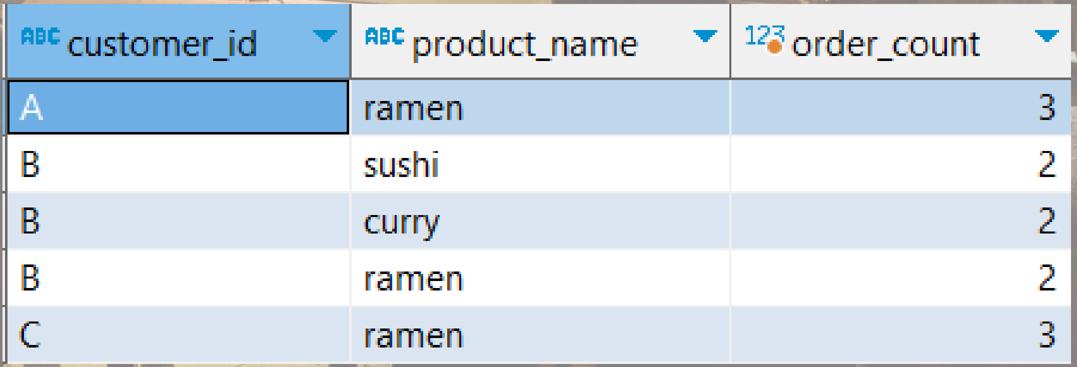


| customer_id | product_name | purchase_count |
|---|---|---|
| A | ramen | 3 |
| B | ramen | 2 |
| C | ramen | 3 |

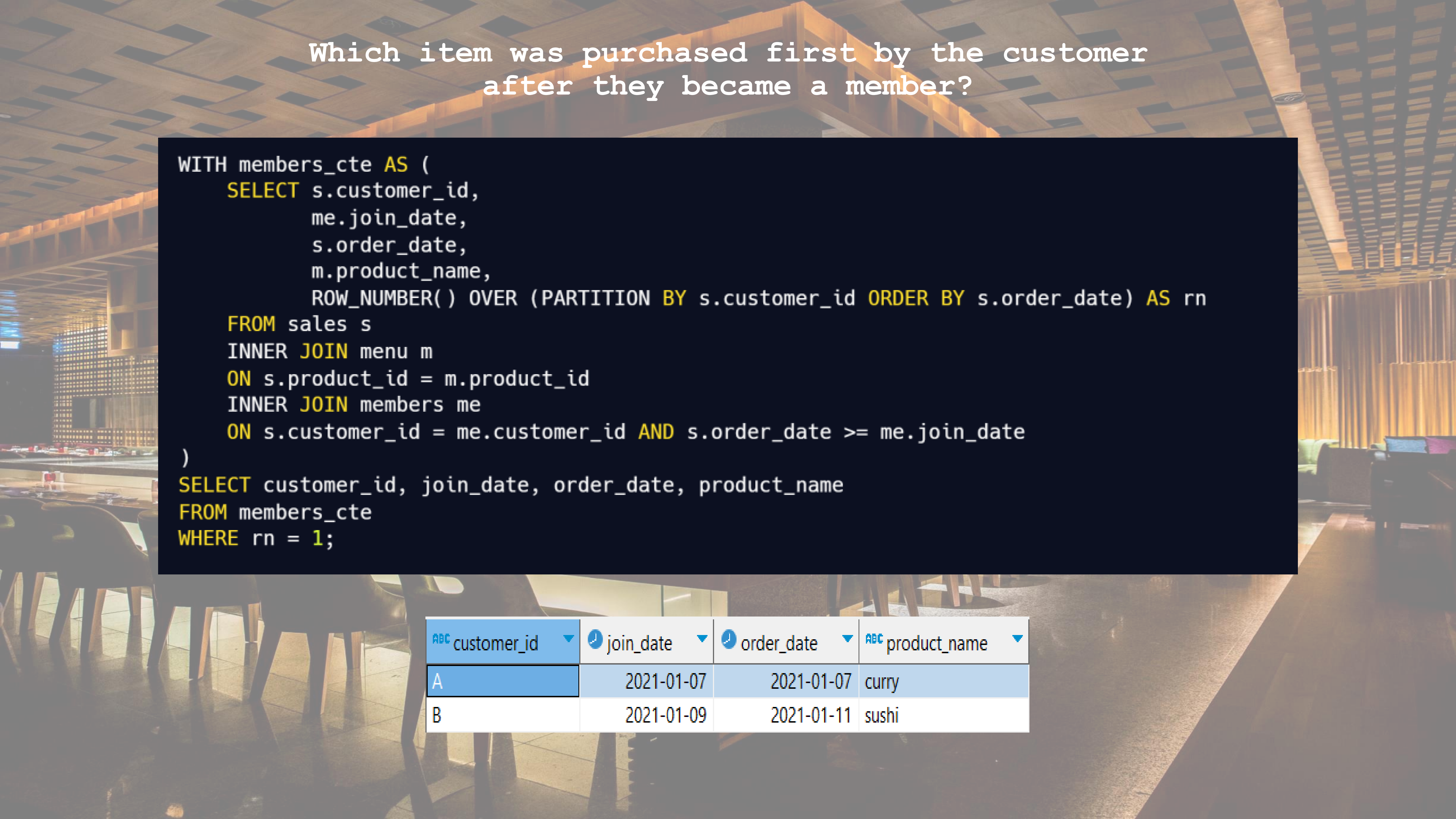
## Which item was the most popular for each customer?



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

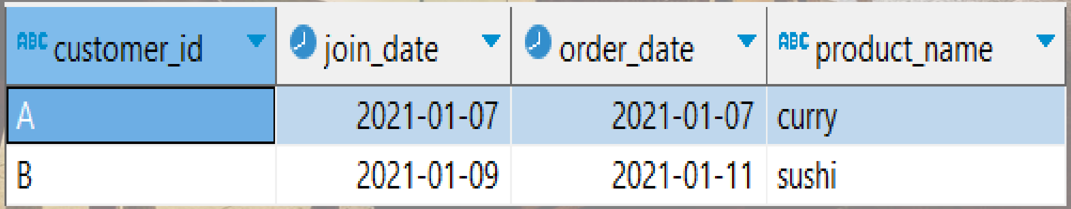


| customer_id | product_name | order_count |
|---|---|---|
| A | ramen | 3 |
| B | sushi | 2 |
| B | curry | 2 |
| B | ramen | 2 |
| C | ramen | 3 |



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

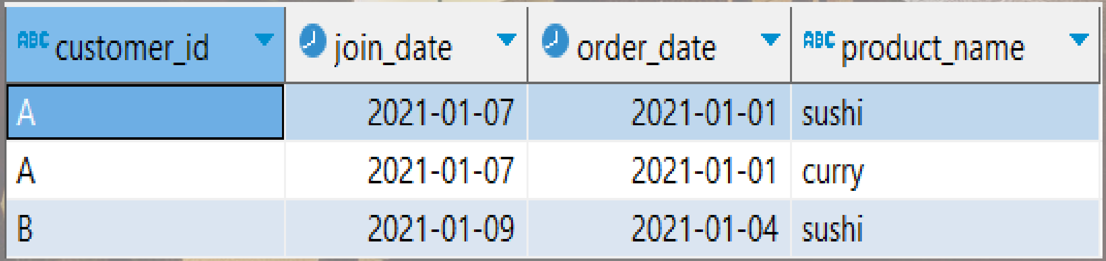
```sql
WITH members_cte AS (
    SELECT s.customer_id,
           me.join_date,
           s.order_date,
           m.product_name,
           ROW_NUMBER() OVER (PARTITION BY s.customer_id ORDER BY s.order_date) AS rn
    FROM sales s
    INNER JOIN menu m
    ON s.product_id = m.product_id
    INNER JOIN members me
    ON s.customer_id = me.customer_id AND s.order_date >= me.join_date
)
SELECT customer_id, join_date, order_date, product_name
FROM members_cte
WHERE rn = 1;
```



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

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

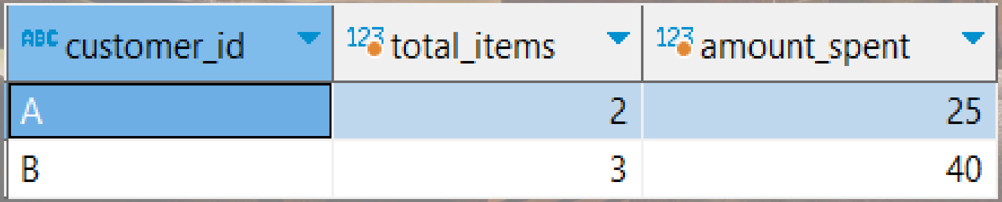
```sql
WITH members_cte AS (
    SELECT s.customer_id,
           me.join_date,
           s.order_date,
           m.product_name,
           DENSE_RANK() OVER (PARTITION BY s.customer_id ORDER BY s.order_date DESC) AS rnk
    FROM sales s
    INNER JOIN menu m
    ON s.product_id = m.product_id
    INNER JOIN members me
    ON s.customer_id = me.customer_id AND s.order_date < me.join_date
)
SELECT customer_id, join_date, order_date, product_name
FROM members_cte
WHERE rnk = 1;
```



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

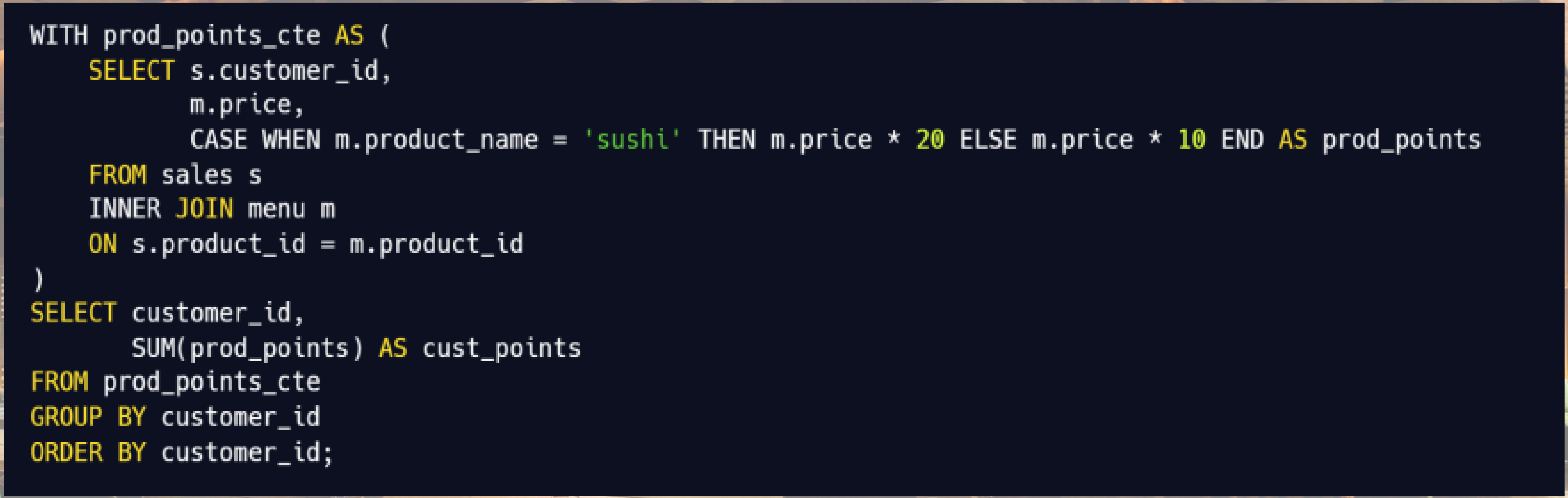
## What is the total items and amount spent for each member before they became a member?

```sql
SELECT s.customer_id,
       COUNT(m.product_name) AS total_items,
       SUM(m.price) AS amount_spent
FROM sales s
INNER JOIN menu m
ON s.product_id = m.product_id
INNER JOIN members me
ON s.customer_id = me.customer_id AND s.order_date < me.join_date
GROUP BY s.customer_id
ORDER BY s.customer_id;
```

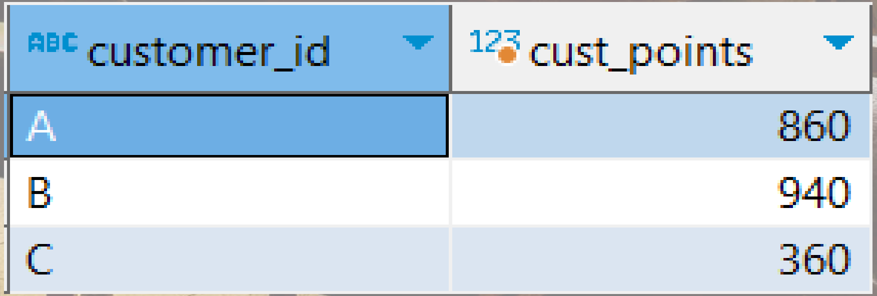


| customer_id | total_items | amount_spent |
|---|---|---|
| A | 2 | 25 |
| B | 3 | 40 |

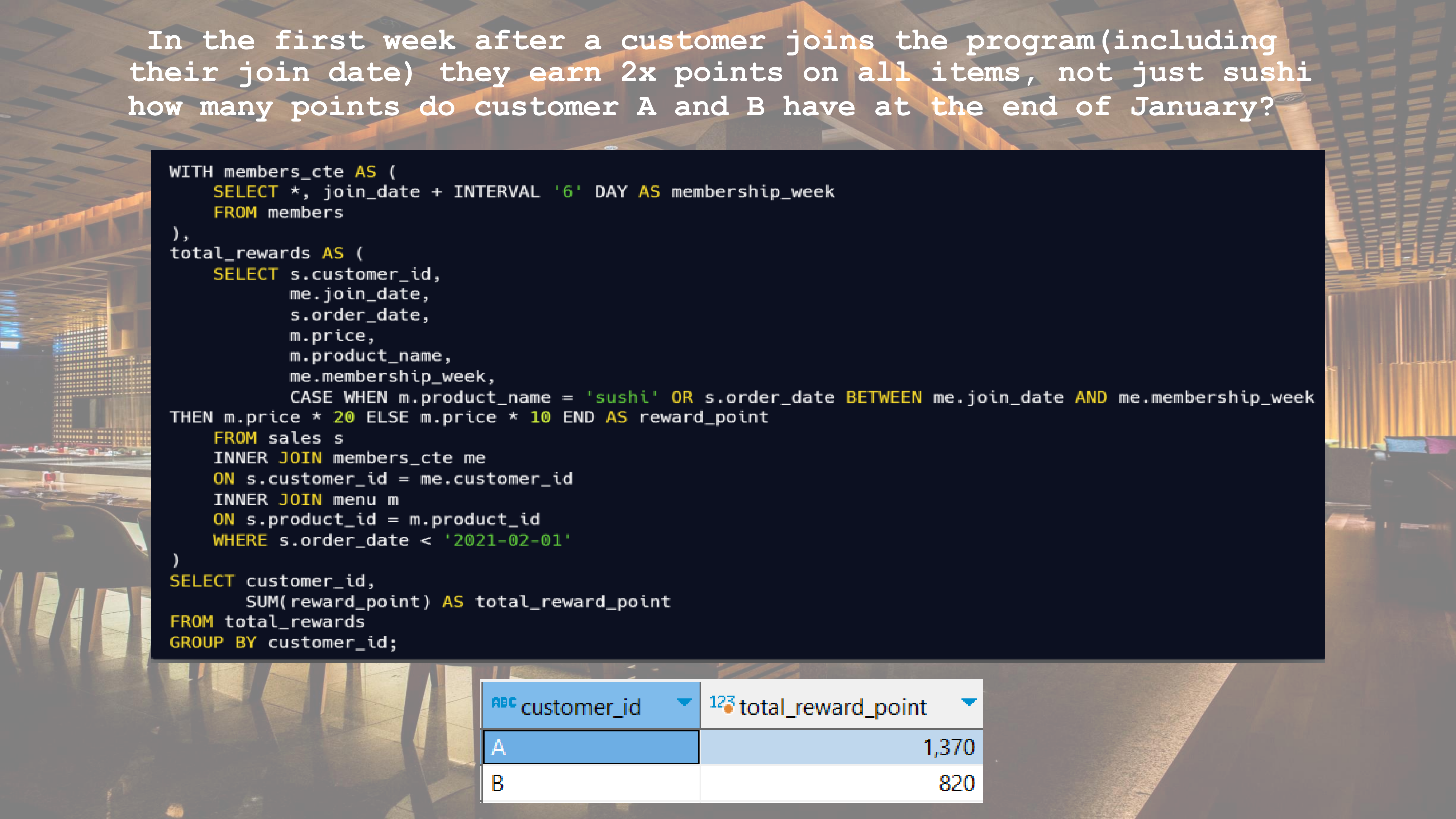
## If each $1 spent equates to 10 points and sushi has a 2x points multiplier, how many points would each customer have?



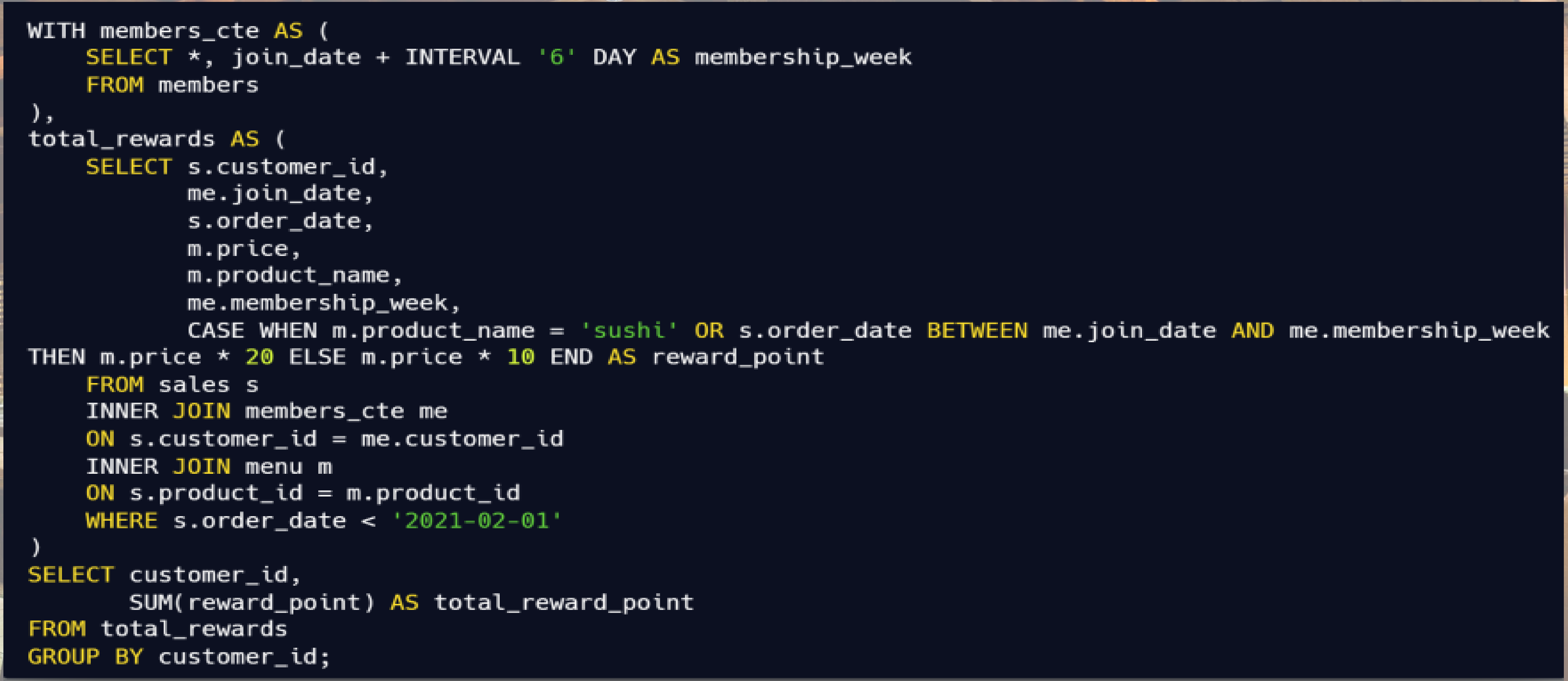
```
WITH prod_points_cte AS (
    SELECT s.customer_id,
           m.price,
           CASE WHEN m.product_name = 'sushi' THEN m.price * 20 ELSE m.price * 10 END AS prod_points
    FROM sales s
    INNER JOIN menu m
    ON s.product_id = m.product_id
)
SELECT customer_id,
       SUM(prod_points) AS cust_points
FROM prod_points_cte
GROUP BY customer_id
ORDER BY customer_id;
```



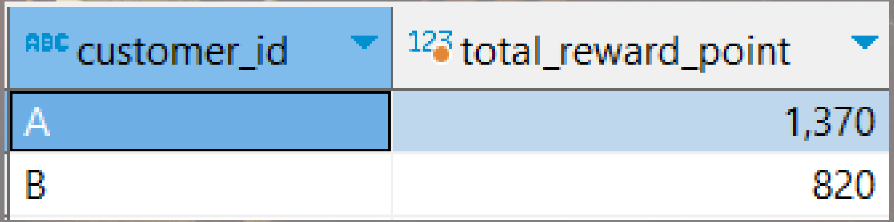
| customer_id | cust_points |
|---|---|
| A | 860 |
| B | 940 |
| C | 360 |



## 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?

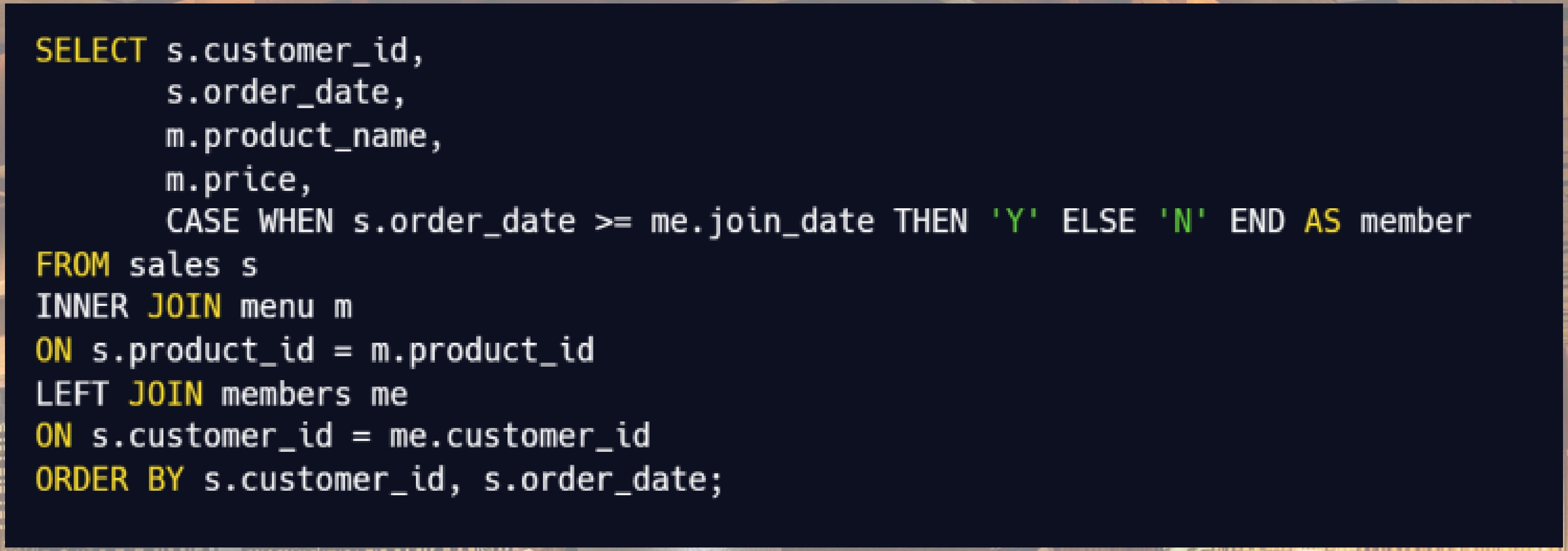


```sql
WITH members_cte AS (
    SELECT *, join_date + INTERVAL '6' DAY AS membership_week
    FROM members
),
total_rewards AS (
    SELECT s.customer_id,
           me.join_date,
           s.order_date,
           m.price,
           m.product_name,
           me.membership_week,
           CASE WHEN m.product_name = 'sushi' OR s.order_date BETWEEN me.join_date AND me.membership_week
THEN m.price * 20 ELSE m.price * 10 END AS reward_point
    FROM sales s
    INNER JOIN members_cte me
    ON s.customer_id = me.customer_id
    INNER JOIN menu m
    ON s.product_id = m.product_id
    WHERE s.order_date < '2021-02-01'
)
SELECT customer_id,
       SUM(reward_point) AS total_reward_point
FROM total_rewards
GROUP BY customer_id;
```

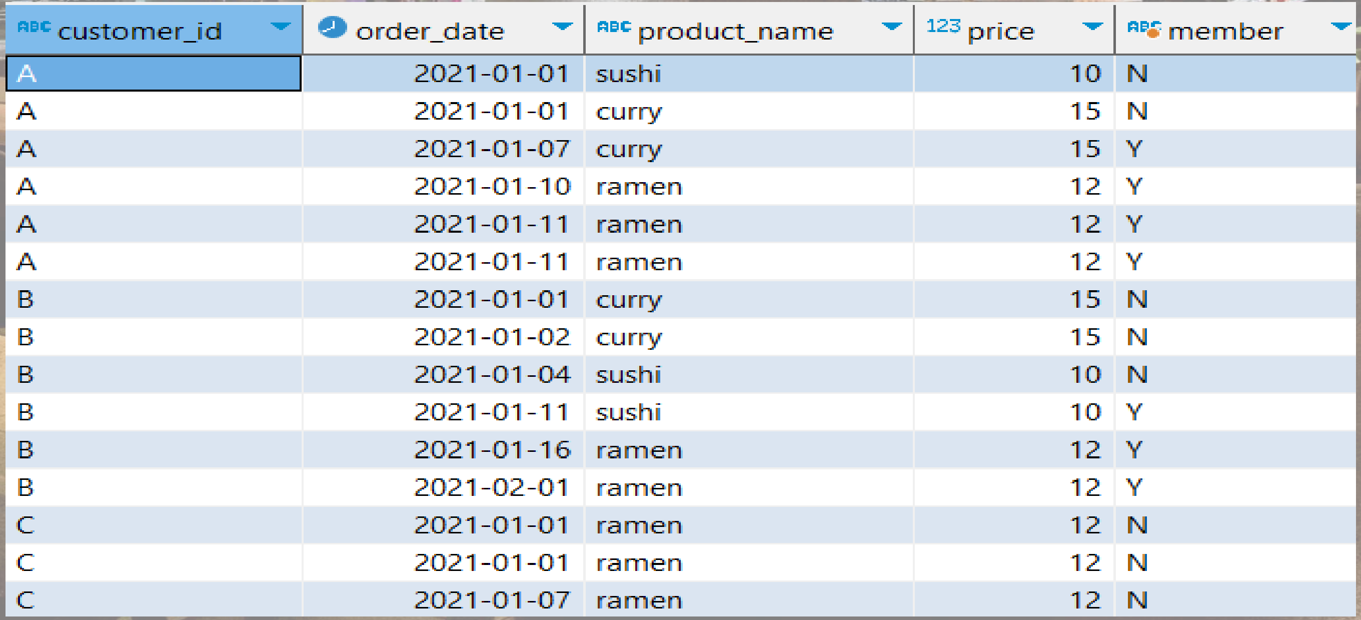


| customer_id | total_reward_point |
|---|---|
| A | 1,370 |
| B | 820 |

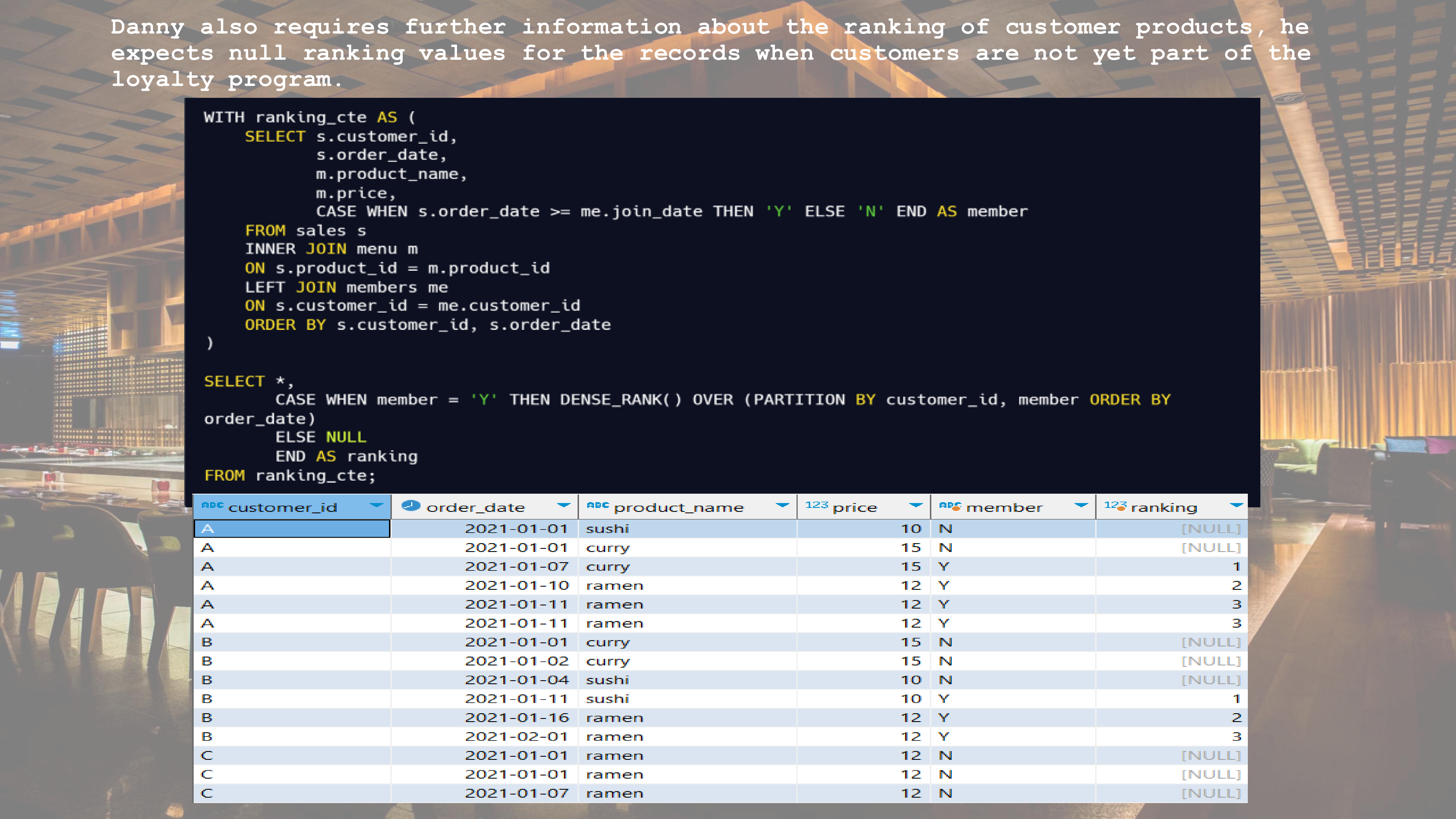
## Recreate the table output using the available data (Output table given @8weeksqlchallenge.com bonus questions)



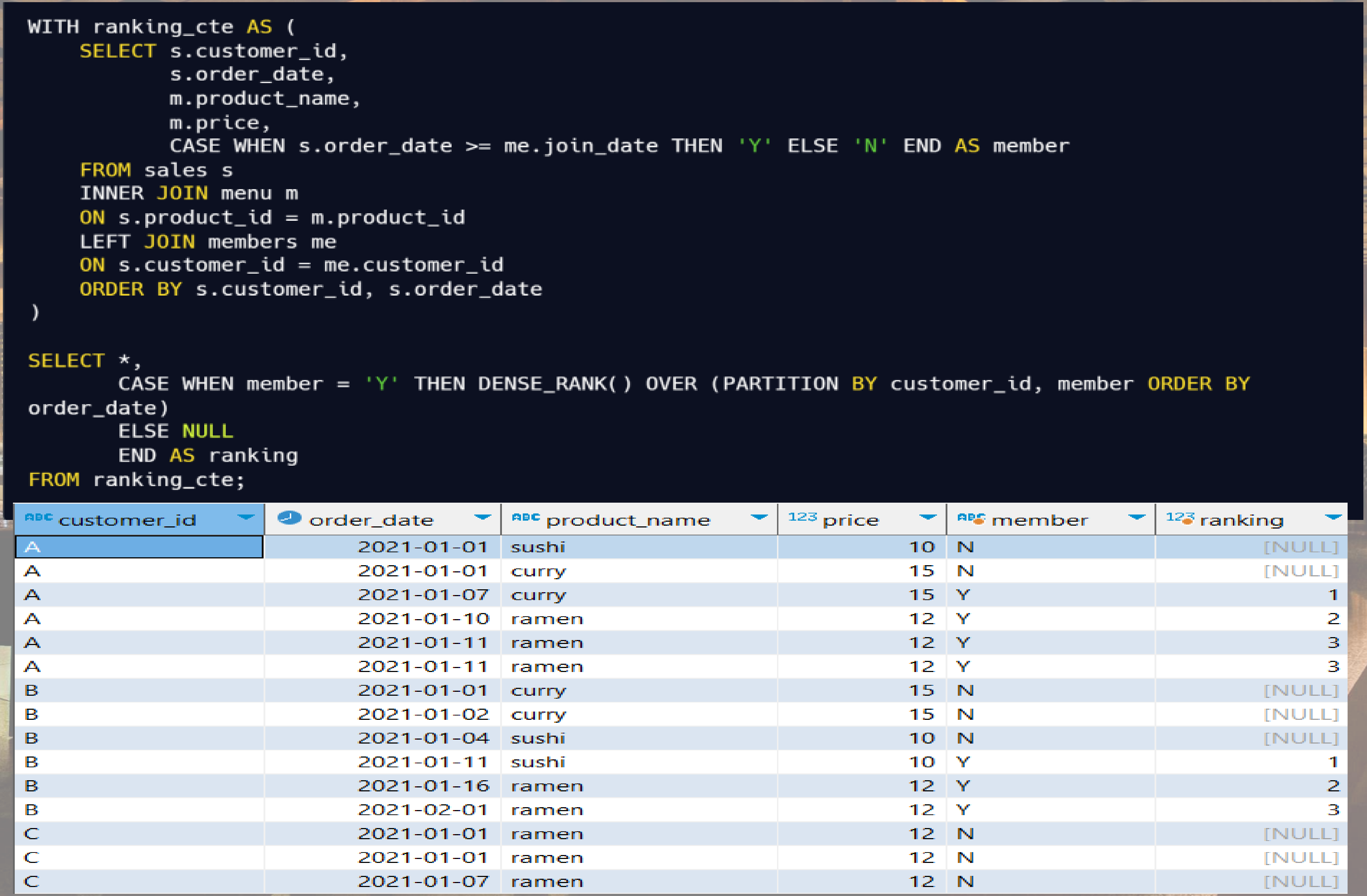
```sql
SELECT s.customer_id,
       s.order_date,
       m.product_name,
       m.price,
       CASE WHEN s.order_date >= me.join_date THEN 'Y' ELSE 'N' END AS member
FROM sales s
INNER JOIN menu m
ON s.product_id = m.product_id
LEFT JOIN members me
ON s.customer_id = me.customer_id
ORDER BY s.customer_id,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



| 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 |



Danny also requires further information about the ranking of customer products, he expects null ranking values for the records when customers are not yet part of the loyalty program.



```
WITH ranking_cte AS (
    SELECT s.customer_id,
           s.order_date,
           m.product_name,
           m.price,
           CASE WHEN s.order_date >= me.join_date THEN 'Y' ELSE 'N' END AS member
    FROM sales s
    INNER JOIN menu m
    ON s.product_id = m.product_id
    LEFT JOIN members me
    ON s.customer_id = me.customer_id
    ORDER BY s.customer_id, s.order_date
)

SELECT *,
       CASE WHEN member = 'Y' THEN DENSE_RANK() OVER (PARTITION BY customer_id, member ORDER BY
order_date)
       ELSE NULL
       END AS ranking
FROM ranking_cte;
```

| 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] |

## Conclusion

- Customer A has made the highest total purchase at the restaurant.
- Customer B was the most frequent visitor to the restaurant.
- The customers' first orders were Sushi, Curry and Ramen for Customer A, Customer B, and Customer C respectively.
- Ramen was the most purchased item on the menu.
- Curry and Sushi  was purchased first by the Customer A and B  respectively after they became a member.
- Before becoming a member, Customer A spent $25 on 2 items whereas Customer B spent $40 on 3 items.

*Thank you!*