# 8weekSQLCHALLENGE

*Case Study #1: Danny's Dinner*

https://8weeksqlchallenge.com/case-study-1/

*Solved by Alfiana Ramdhan*

# Contents

*Solved by Alfiana Ramdhan*

## Introduction

This is #1 case study of 8weekSQLCHALLENGE by Danny Ma, for details can be found at https://8weeksqlchallenge.com/case-study-1/.

As information, Danny decided to open a small restaurant 'Danny's Dinner' which sells 3 of his favorite foods: sushi, curry and ramen.

In this project, danny needs our help about how to use their data to help them run the business. Danny wants to get insight about Customer visiting patterns, how much money they've spent ,which menu items are their favourite and whether he should expand the existing customer loyalty program or not.

Danny has provided 10 questions for this 'Case Study to be solved using SQL with 2 bonus questions. Also, Danny has shared 3 datasets for this case study : sales, menu, members. All datasets exist within Danny's Dinner database schema.

For this case study, I used PostgreSQL and all the queries done to solve the questions are result of my SQL knowledge.

*Solved by Alfiana Ramdhan*

## Problem Statement

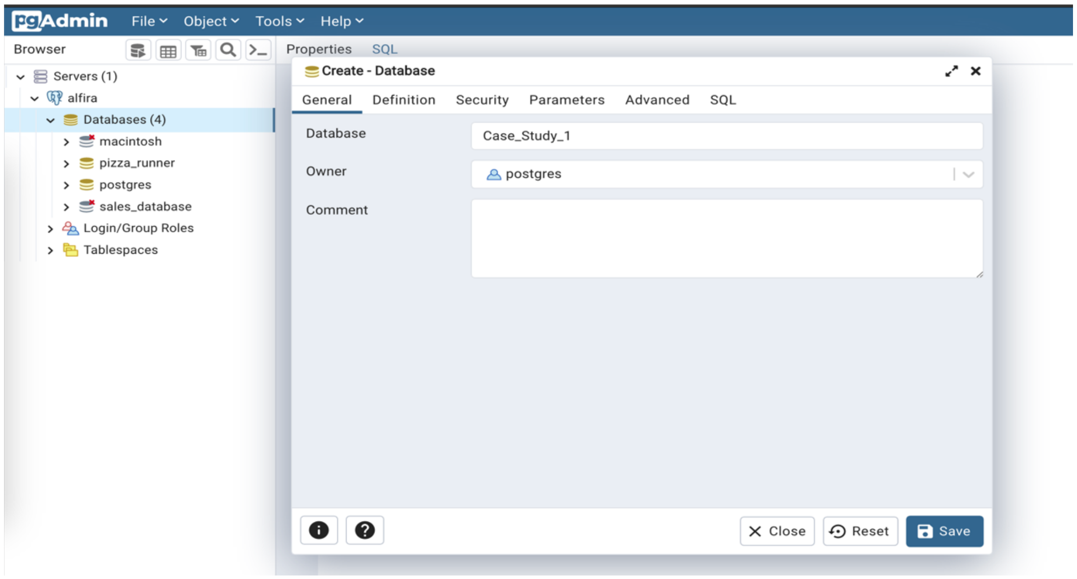
Danny wants to use the data to answer a few simple questions about :

1. How is customer visiting pattern ?
2. How much money customers have spent and also which menu items are their Favourite ?
3. Expansion of existing customer loyalty program
4. Join all tables and about the ranking of customer products so Danny and his team can quickly derive insights

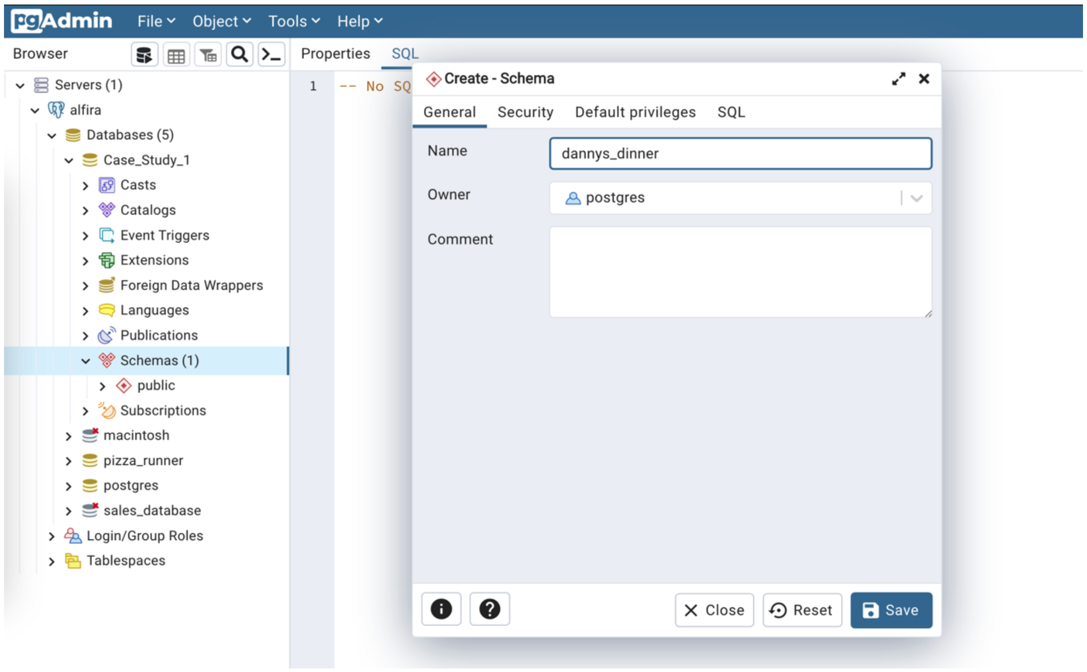
*Solved by Alfiana Ramdhan*

## Create database, schema and tables

For this case study, I used PostgreSQL then I create the database, schema and table by using menu options after right clicking on database as shown in following image:



Database: Case_Study_1
Owner: postgres
Comment:



```
-- No SQ
```

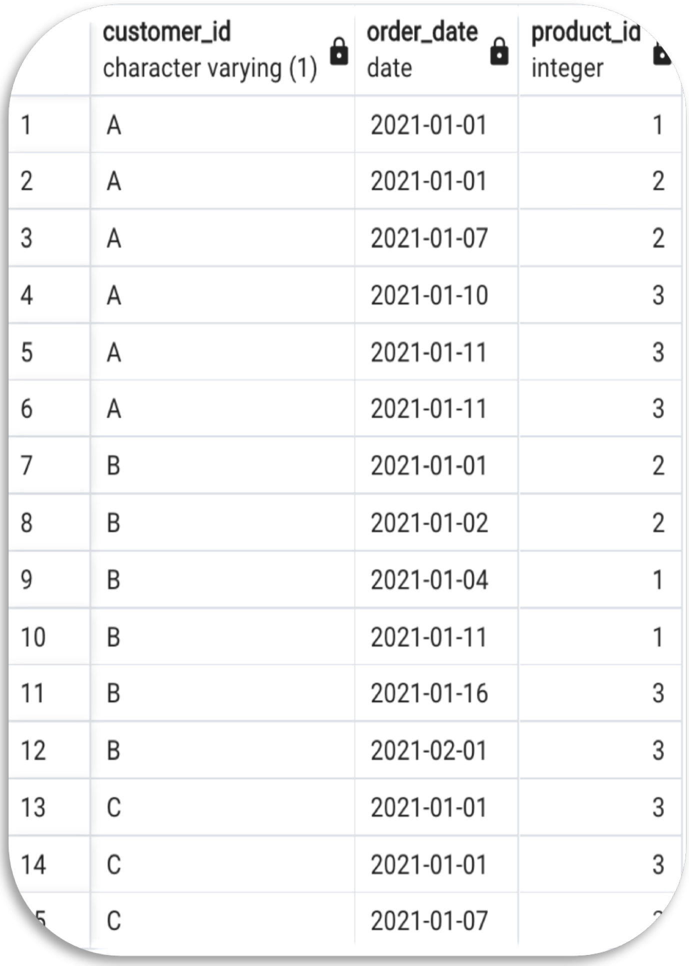
Name: dannys_dinner
Owner: postgres
Comment:

*Solved by Alfiana Ramdhan*

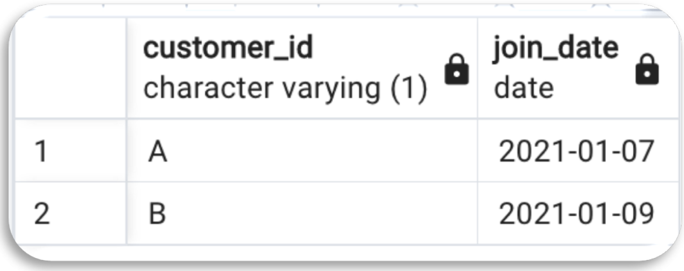
Once database and schema were created, next all three tables were created as per the SQL queries provided by Danny Ma.

```
CREATE SCHEMA dannys_diner;

SET search_path = dannys_diner;


CREATE TABLE sales (

        "customer_id" VARCHAR(1),

        "order_date" DATE,

        "product_id" INTEGER

);

INSERT INTO sales ("customer_id","order_date","product_id")

VALUES

        ('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');

CREATE TABLE menu (
	"product_id" INTEGER,
	"product_name" VARCHAR(5),
	"price" INTEGER
);
INSERT INTO menu ("product_id","product_name","price")
VALUES
 ('1', 'sushi', '10'),
 ('2', 'curry', '15'),
 ('3', 'ramen', '12');

CREATE TABLE members (
	"customer_id" VARCHAR(1),
	"join_date" DATE
);
```

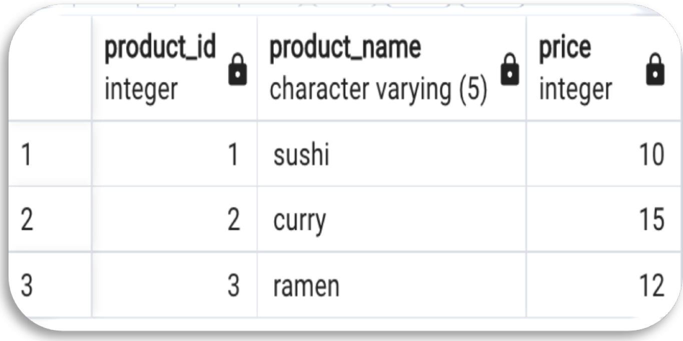
```
INSERT INTO members ("customer_id","join_date")

VALUES

('A', '2021-01-07'),

 ('B', '2021-01-09');
```



| | customer_id character varying (1) | order_date date | product_id integer |
|---|---|---|---|
| 1 | A | 2021-01-01 | 1 |
| 2 | A | 2021-01-01 | 2 |
| 3 | A | 2021-01-07 | 2 |
| 4 | A | 2021-01-10 | 3 |
| 5 | A | 2021-01-11 | 3 |
| 6 | A | 2021-01-11 | 3 |
| 7 | B | 2021-01-01 | 2 |
| 8 | B | 2021-01-02 | 2 |
| 9 | B | 2021-01-04 | 1 |
| 10 | B | 2021-01-11 | 1 |
| 11 | B | 2021-01-16 | 3 |
| 12 | B | 2021-02-01 | 3 |
| 13 | C | 2021-01-01 | 3 |
| 14 | C | 2021-01-01 | 3 |
| 15 | C | 2021-01-07 | 2 |



| | customer_id character varying (1) | join_date date |
|---|---|---|
| 1 | A | 2021-01-07 |
| 2 | B | 2021-01-09 |



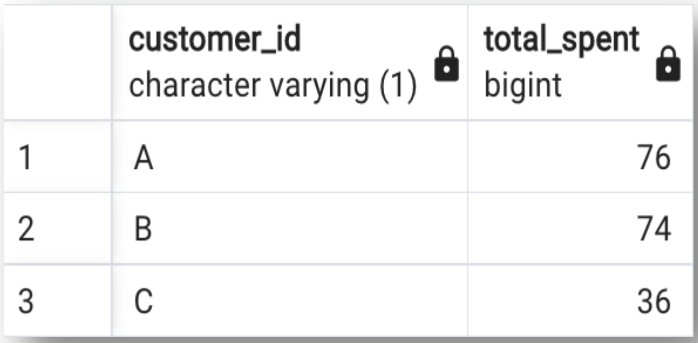
| | product_id integer | product_name character varying (5) | price integer |
|---|---|---|---|
| 1 | 1 | sushi | 10 |
| 2 | 2 | curry | 15 |
| 3 | 3 | ramen | 12 |

*Solved by Alfiana Ramdhan*

## Case Study Questions

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

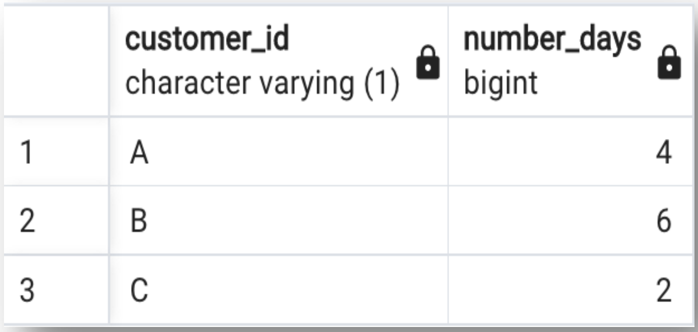
```
SELECT customer_id,
        SUM(price) as total_spent
FROM dannys_dinner.sales t1
JOIN dannys_dinner.menu t2
    ON t1.product_id = t2.product_id
GROUP BY 1
ORDER BY 1;
```



| | customer_id character varying (1) | total_spent bigint |
|---|---|---|
| 1 | A | 76 |
| 2 | B | 74 |
| 3 | C | 36 |

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

```
SELECT customer_id,
        COUNT(DISTINCT order_date) as number_days
FROM dannys_dinner.sales
GROUP BY 1
ORDER BY 1;
```



| | customer_id character varying (1) | number_days bigint |
|---|---|---|
| 1 | A | 4 |
| 2 | B | 6 |
| 3 | C | 2 |

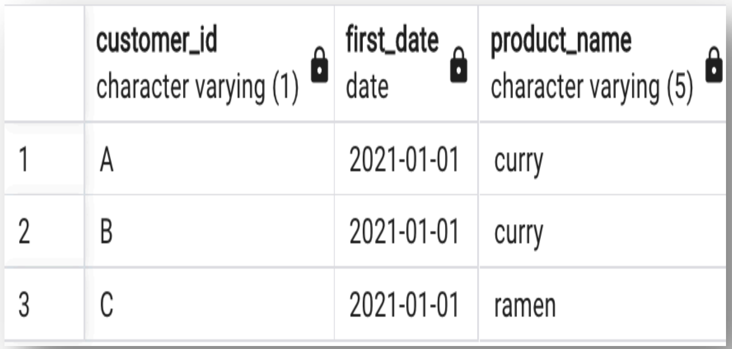
3. **What was the first item from the menu purchased by each customer?**

To answer this question, we can use either subquery or window function. I will try to solve the question in both ways.

*Solved by Alfiana Ramdhan*

**#1 option :**

```
SELECT  t1.customer_id,
            first_date,
            product_name
FROM dannys_dinner.sales t1
JOIN (
    SELECT customer_id,
            MIN(order_date) as first_date
    FROM dannys_dinner.sales
    GROUP BY 1
)t2 ON t1.customer_id = t2.customer_id
JOIN dannys_dinner.menu t3 ON t1.product_id = t3.product_id
WHERE t1.order_date = t2.first_date
ORDER BY 1;
```



| | customer_id character varying (1) | first_date date | product_name character varying (5) |
|---|---|---|---|
| 1 | A | 2021-01-01 | curry |
| 2 | B | 2021-01-01 | curry |
| 3 | C | 2021-01-01 | ramen |

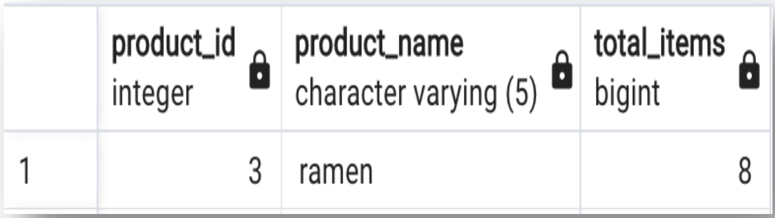
**#2 option :**

```
WITH rank_order AS (
    SELECT customer_id,  order_date as first_date,  t1.product_id, product_name,
            ROW_NUMBER() OVER(PARTITION BY customer_id ORDER BY order_date) as rn
    FROM dannys_dinner.sales t1
    LEFT JOIN dannys_dinner.menu t2
            ON t1.product_id = t2.product_id
    ORDER BY 1,2,3)

            SELECT customer_id,
                    first_date,
                    product_name
            FROM rank_order
            WHERE rn = 1;
```

*Solved by Alfiana Ramdhan*

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

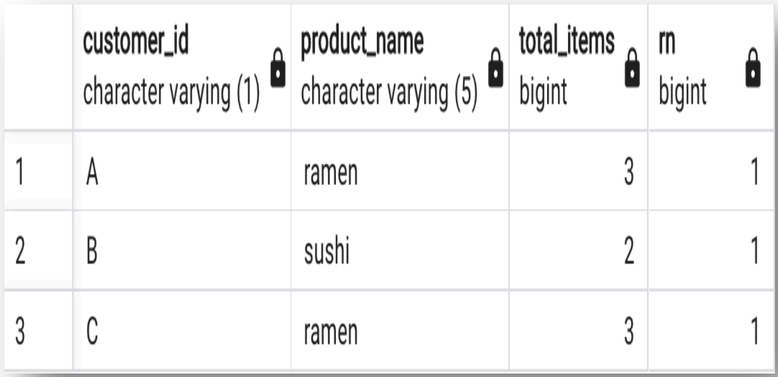
```
SELECT t1.product_id,  product_name, COUNT(t1.product_id) as total_items
FROM dannys_dinner.sales t1
LEFT JOIN dannys_dinner.menu t2
    ON t1.product_id = t2.product_id
GROUP BY 1,2
ORDER BY 3 DESC
LIMIT 1;
```



| | product_id integer | product_name character varying (5) | total_items bigint |
|---|---|---|---|
| 1 | 3 | ramen | 8 |

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

```
WITH rank_order AS (
    SELECT  t1.customer_id,  product_name,  COUNT(t1.product_id) as total_items,
                    ROW_NUMBER () OVER (
                            PARTITION BY t1.customer_id ORDER BY
                            COUNT(t1.product_id) DESC ) as rn
    FROM dannys_dinner.sales t1
    LEFT JOIN dannys_dinner.menu t2
            ON t1.product_id = t2.product_id
    GROUP BY 1,2
    ORDER BY 3 DESC)

SELECT customer_id, product_name,
            total_items, rn
FROM rank_order
    WHERE rn=1
      ORDER BY 1;
```

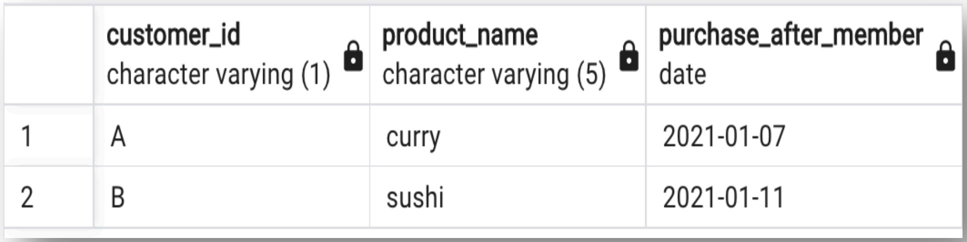


| | customer_id character varying (1) | product_name character varying (5) | total_items bigint | rn bigint |
|---|---|---|---|---|
| 1 | A | ramen | 3 | 1 |
| 2 | B | sushi | 2 | 1 |
| 3 | C | ramen | 3 | 1 |

*Solved by Alfiana Ramdhan*

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

```
WITH rank_order AS (
    SELECT t1.customer_id,
            product_name,
            t1.order_date,
            ROW_NUMBER () OVER(PARTITION BY t1.customer_id order by order_date)as rn
    FROM dannys_dinner.sales t1
    LEFT JOIN dannys_dinner.menu t2 ON t1.product_id = t2.product_id
    LEFT JOIN dannys_dinner.members t3 ON t1.customer_id = t3.customer_id
    WHERE t1.order_date >= t3.join_date
    GROUP BY 1, 2, 3
    ORDER BY 1
)
 SELECT customer_id,
    product_name,
    order_date as purchase_after_member
 FROM rank_order
 WHERE rn = 1
 ORDER BY 1;
```

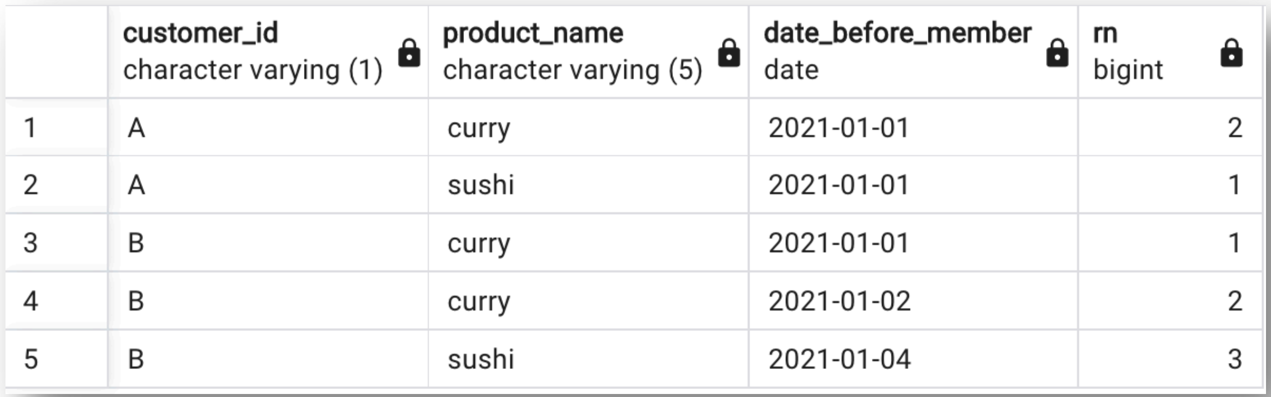


| | customer_id character varying (1) | product_name character varying (5) | purchase_after_member date |
|---|---|---|---|
| 1 | A | curry | 2021-01-07 |
| 2 | B | sushi | 2021-01-11 |

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

```
SELECT t1.customer_id,
            product_name,
            t1.order_date as date_before_member,
            ROW_NUMBER () OVER(PARTITION BY t1.customer_id order by order_date)as rn
FROM dannys_dinner.sales t1
LEFT JOIN dannys_dinner.menu t2 ON t1.product_id = t2.product_id
LEFT JOIN dannys_dinner.members t3 ON t1.customer_id = t3.customer_id
WHERE t1.order_date < t3.join_date
ORDER BY 1,2;
```

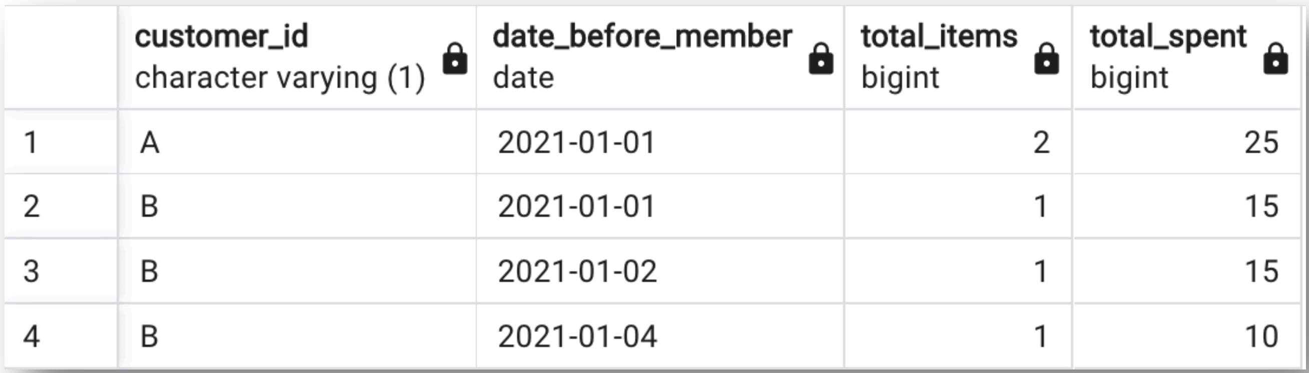
*Solved by Alfiana Ramdhan*



| | customer_id character varying (1) | product_name character varying (5) | date_before_member date | rn bigint |
|---|---|---|---|---|
| 1 | A | curry | 2021-01-01 | 2 |
| 2 | A | sushi | 2021-01-01 | 1 |
| 3 | B | curry | 2021-01-01 | 1 |
| 4 | B | curry | 2021-01-02 | 2 |
| 5 | B | sushi | 2021-01-04 | 3 |

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

```
SELECT t1.customer_id,
        order_date as date_before_member,
        COUNT(t1.product_id)as total_items,
        SUM(price)as total_spent
FROM dannys_dinner.sales t1
LEFT JOIN dannys_dinner.menu t2 ON t1.product_id = t2.product_id
LEFT JOIN dannys_dinner.members t3 ON t1.customer_id = t3.customer_id
WHERE t1.order_date < t3.join_date
GROUP BY 1, 2
ORDER BY 1;
```

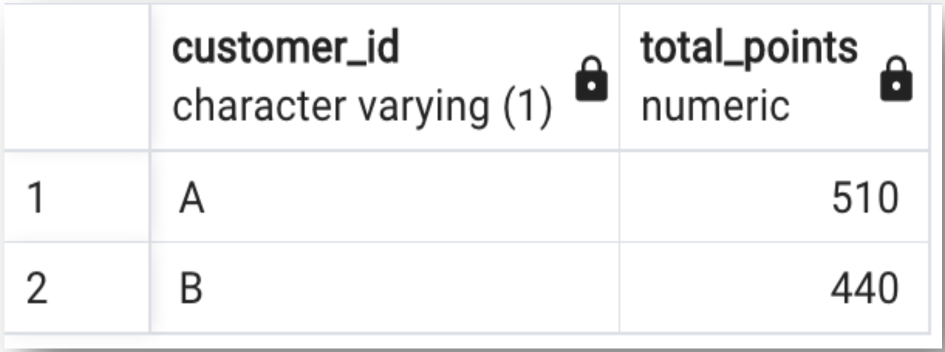


| | customer_id character varying (1) | date_before_member date | total_items bigint | total_spent bigint |
|---|---|---|---|---|
| 1 | A | 2021-01-01 | 2 | 25 |
| 2 | B | 2021-01-01 | 1 | 15 |
| 3 | B | 2021-01-02 | 1 | 15 |
| 4 | B | 2021-01-04 | 1 | 10 |

*Solved by Alfiana Ramdhan*

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

```
WITH spent AS (
    SELECT t1.customer_id,
            product_name,
            order_date,
            SUM(price)as total_spent,
            CASE WHEN product_name = 'sushi' THEN 20
                    ELSE 10
            END as points
    FROM dannys_dinner.sales t1
    LEFT JOIN dannys_dinner.menu t2 ON t1.product_id = t2.product_id
    LEFT JOIN dannys_dinner.members t3 ON t1.customer_id = t3.customer_id
    WHERE order_date >= join_date
    GROUP BY 1, 2, 3
    ORDER BY 1
)
    SELECT customer_id,
            SUM(total_spent * points)as total_points
    FROM spent
    GROUP BY 1
    ORDER BY 1;
```

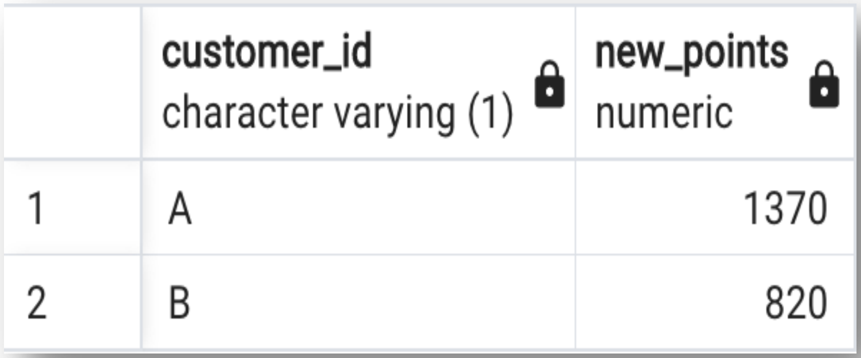


| | customer_id character varying (1) | total_points numeric |
|---|---|---|
| 1 | A | 510 |
| 2 | B | 440 |

*Solved by Alfiana Ramdhan*

**10. <u>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?</u>**

```
WITH count_points AS (
  SELECT t1.customer_id,
         order_date,
         join_date,
         product_name,
         SUM(point) AS point
  FROM dannys_dinner.sales t1
  JOIN (SELECT  product_id,
                product_name,
         CASE WHEN product_name = 'sushi' THEN price * 20
              ELSE price * 10
         END AS point
         FROM dannys_dinner.menu AS m
        ) AS t2 ON t1.product_id = t2.product_id
  JOIN dannys_dinner.members AS t3 ON t1.customer_id = t3.customer_id
  GROUP BY 1,2,3,4
 )
   SELECT customer_id,
          SUM(CASE WHEN order_date >= join_date
                   AND order_date < join_date + (7 * INTERVAL '1 day')
                   AND product_name != 'sushi' THEN point * 2
                   ELSE point END) AS new_points
   FROM count_points
   WHERE  DATE_PART('month', order_date) = 1
   GROUP BY 1
   ORDER BY 1;
```



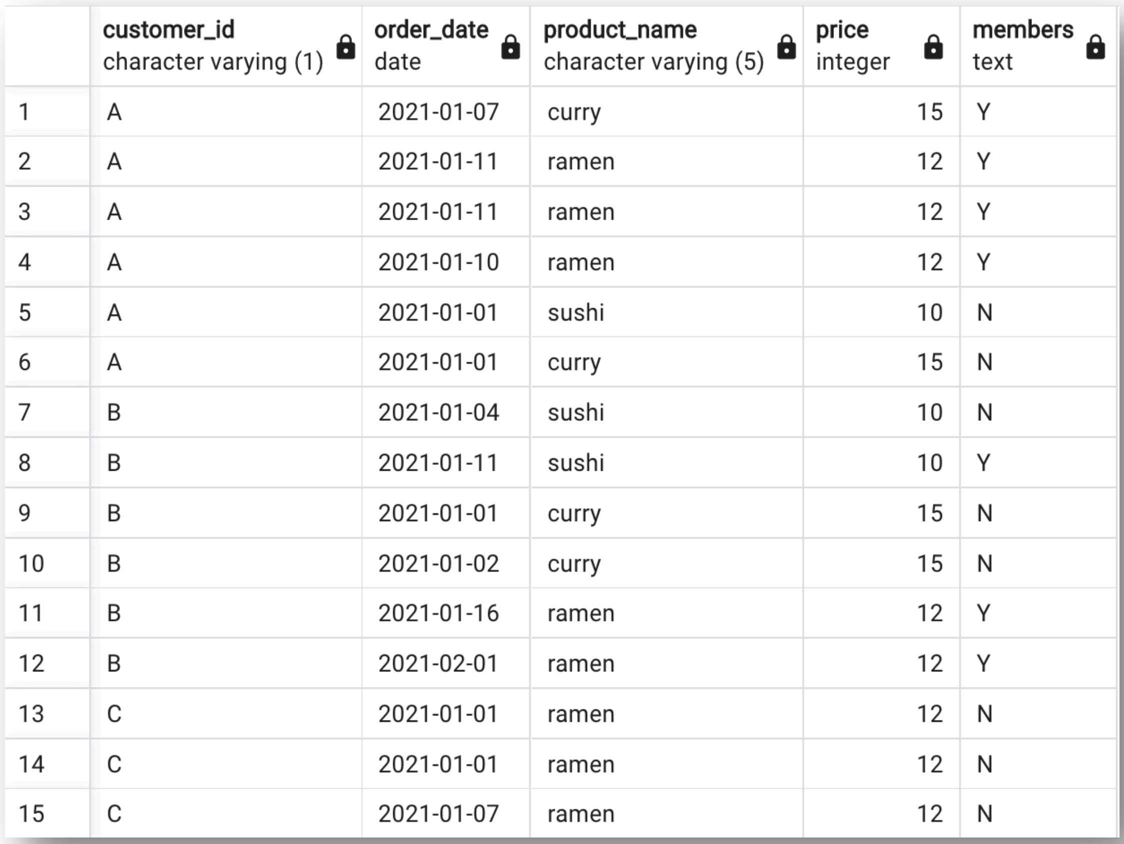
| | customer_id<br>character varying (1) | new_points<br>numeric |
|---|---|---|
| 1 | A | 1370 |
| 2 | B | 820 |

*Solved by Alfiana Ramdhan*

## Bonus questions

### 11. Join all the things

```
SELECT t1.customer_id,
            order_date,
            product_name,
            price,
    CASE WHEN order_date >= join_date THEN 'Y'
            ELSE 'N'
    END as members
FROM dannys_dinner.sales t1
LEFT JOIN dannys_dinner.menu t2 ON t1.product_id = t2.product_id
LEFT JOIN dannys_dinner.members t3 ON t1.customer_id = t3.customer_id;
```



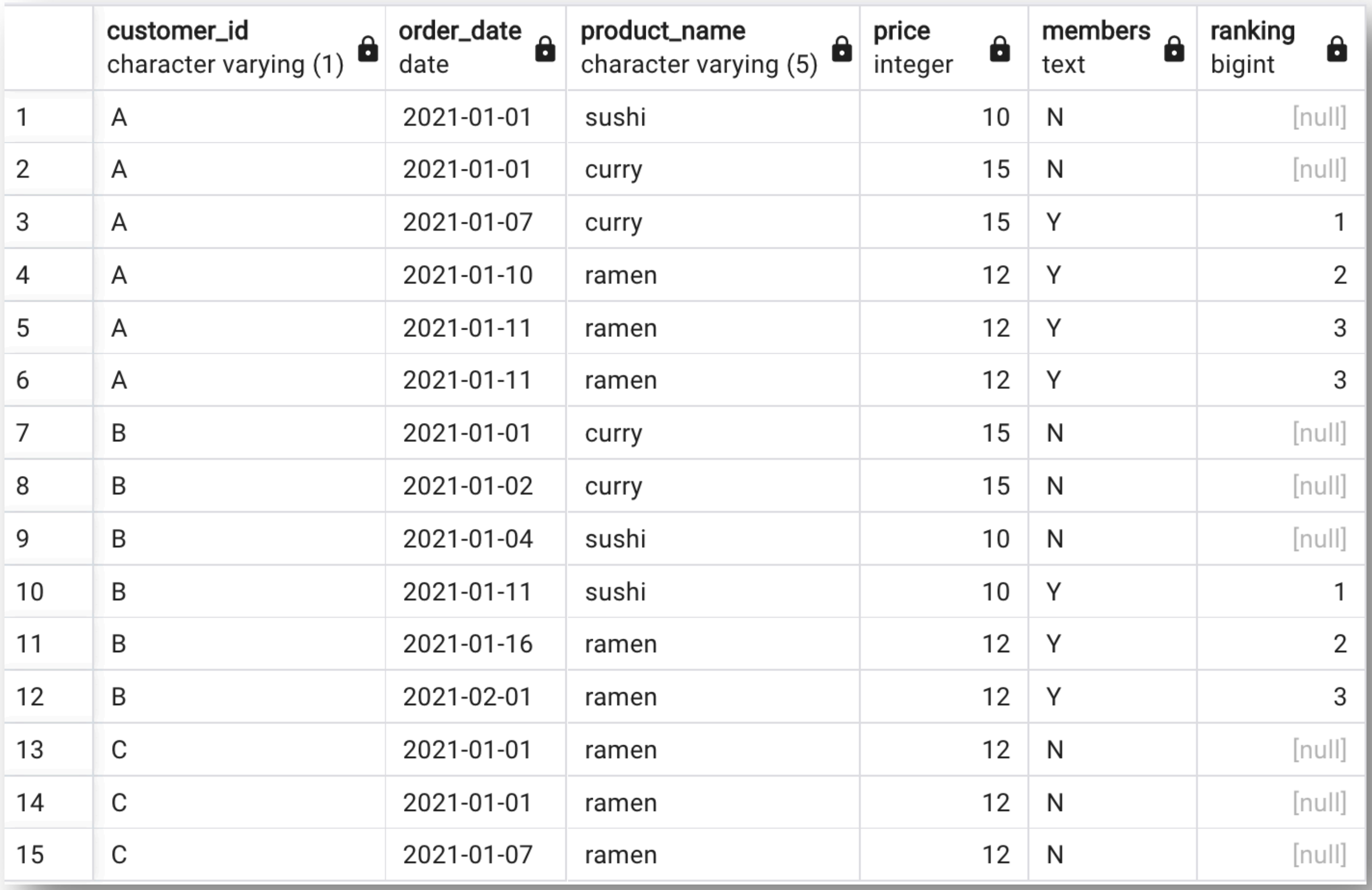
| | customer_id character varying (1) | order_date date | product_name character varying (5) | price integer | members text |
|---|---|---|---|---|---|
| 1 | A | 2021-01-07 | curry | 15 | Y |
| 2 | A | 2021-01-11 | ramen | 12 | Y |
| 3 | A | 2021-01-11 | ramen | 12 | Y |
| 4 | A | 2021-01-10 | ramen | 12 | Y |
| 5 | A | 2021-01-01 | sushi | 10 | N |
| 6 | A | 2021-01-01 | curry | 15 | N |
| 7 | B | 2021-01-04 | sushi | 10 | N |
| 8 | B | 2021-01-11 | sushi | 10 | Y |
| 9 | B | 2021-01-01 | curry | 15 | N |
| 10 | B | 2021-01-02 | curry | 15 | N |
| 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 |

*Solved by Alfiana Ramdhan*

## 12. Rank all the things

```
WITH index_rn AS (
            SELECT t1.customer_id,
                        order_date,
                        product_name,
                        price,
                  CASE WHEN order_date >= join_date THEN 'Y'
                        ELSE 'N' END as members
            FROM dannys_dinner.sales t1
            LEFT JOIN dannys_dinner.menu t2 ON t1.product_id = t2.product_id
            LEFT JOIN dannys_dinner.members t3 ON t1.customer_id = t3.customer_id
            ORDER BY  1
)
  SELECT customer_id,
            order_date,
            product_name,
            price,
            members,
      CASE WHEN members = 'N' THEN null
            ELSE RANK () OVER (PARTITION BY customer_id,members ORDER BY order_date)
      END as ranking
  FROM index_rn ;
```

*Solved by Alfiana Ramdhan*



| | customer_id<br>character varying (1) | order_date<br>date | product_name<br>character varying (5) | price<br>integer | members<br>text | ranking<br>bigint |
|---|---|---|---|---|---|---|
| 1 | A | 2021-01-01 | sushi | 10 | N | [null] |
| 2 | A | 2021-01-01 | curry | 15 | 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] |

## Insights and Recommendation

- The customer who has spent the most money was customer A, at $76, followed by customer B at $74.
- Although customer B was not the one who spends the most money, B was the one who frequents the restaurant the most, 6 times in month.
- The most popular menu for each customer was ramen. It was purchased 8 times which is 53% of whole.
- Ramen is favourite item for customer A and C whereas B likes all three items equally as per the data.
- Even though Ramen was popular but before joining 'Customer loyalty' program A ordered 'sushi' and 'curry' and B ordered 'sushi'.
- Customer A was the first 'Loyal Customer' followed by B .
- Customer C has purchased the lowest out of all three customer and also, he is not a member  of 'loyalty program'.
- Find out what makes sushi their favorite menu for customers made first order and apply the same strategy in other customer cities.
- The restaurant should utilize customer and product information for marketing strategies that will help in get loyal customer.

*Solved by Alfiana Ramdhan*