

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 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 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

Entity Relationship Diagram

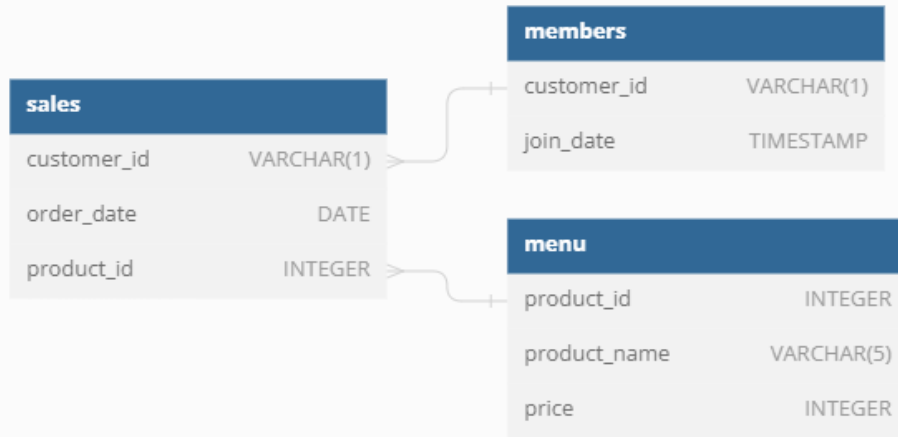


Table 1: sales

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.

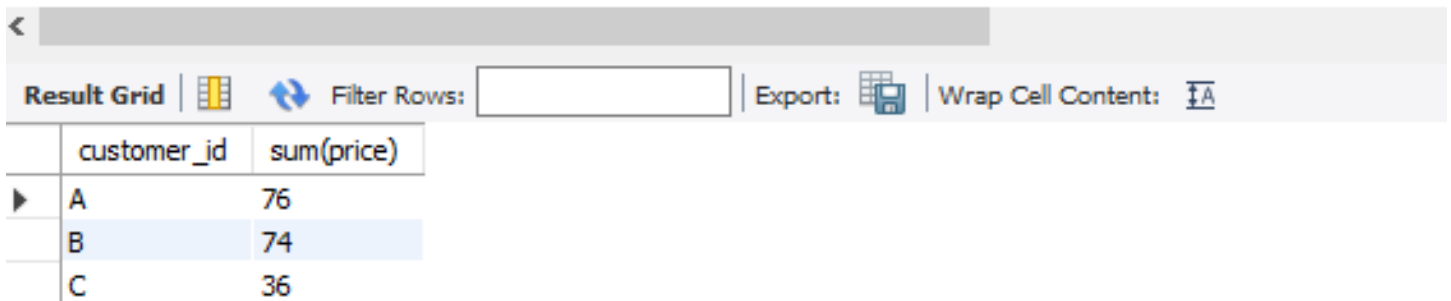
| customer_id | join_date |
|-------------|------------|
| A | 2021-01-07 |
| B | 2021-01-09 |

Case Study Questions

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

2

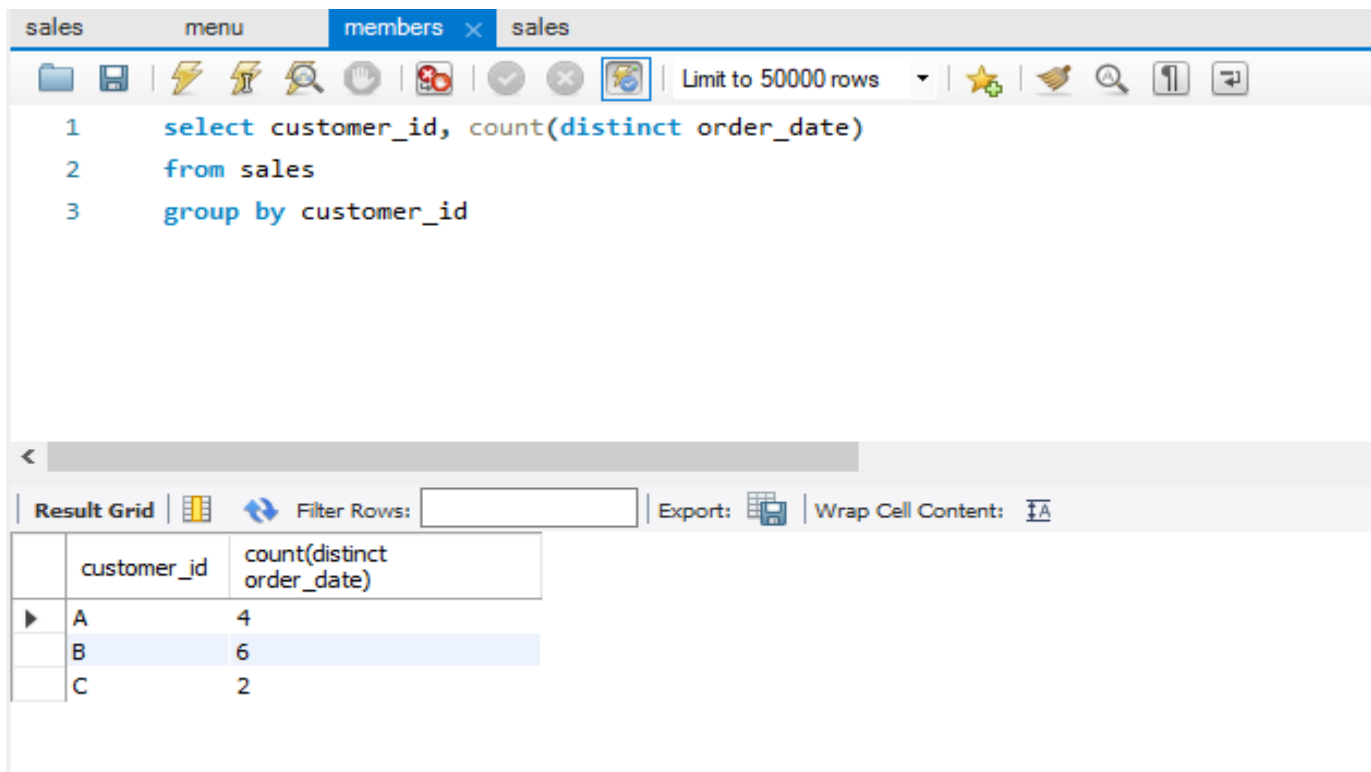
```
3 • select customer_id, sum(price)
4   from sales s
5   join menu m on s.product_id=m.product_id
6   group by customer_id
```



The screenshot shows a database interface with a query editor and a result grid. The query is a SQL statement that joins the 'sales' and 'menu' tables and groups the results by customer_id to calculate the total amount spent. The result grid displays three rows of data.

| | customer_id | sum(price) |
|---|-------------|------------|
| ▶ | A | 76 |
| | B | 74 |
| | C | 36 |

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



The screenshot shows a database interface with a query editor and a result grid. The query is a SQL statement that groups the 'sales' table by customer_id and counts the number of distinct order dates. The result grid displays three rows of data.

| | customer_id | count(distinct order_date) |
|---|-------------|----------------------------|
| ▶ | A | 4 |
| | B | 6 |
| | C | 2 |

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

The screenshot shows a SQL IDE with a query editor and a result grid. The query uses a Common Table Expression (CTE) to rank products by order date for each customer and then selects the first item (rn=1).

```
1 • with cte as
2   (select customer_id, product_id, order_date,
3     row_number() over(partition by customer_id order by order_date) as rn
4   from sales)
5
6   select customer_id, cte.product_id, product_name
7   from cte
8   join menu m on cte.product_id=m.product_id
9   where rn=1
10
```

The result grid displays the following data:

| | customer_id | product_id | product_name |
|---|-------------|------------|--------------|
| ▶ | A | 1 | sushi |
| | B | 2 | curry |
| | C | 3 | ramen |

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

The screenshot shows a SQL IDE with a query editor and a result grid. The query uses a simple SQL statement to count the number of purchases for each product, order by count descending, and limit the result to 1.

```
1 • select s.product_id, product_name, count(s.product_id) as count
2   from sales s
3   join menu m on s.product_id=m.product_id
4   group by 1, 2
5   order by count desc
6   limit 1
7
8
```

The result grid displays the following data:

| | product_id | product_name | count |
|---|------------|--------------|-------|
| ▶ | 3 | ramen | 8 |

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

SQL query editor showing a query to find the most popular item for each customer.

```
1 • with cte as
2   (select customer_id, pid, product_name, count,
3         max(count) over(partition by customer_id) as max_count
4   from
5   (select distinct customer_id, s.product_id as pid, product_name,
6     count(s.product_id) over(partition by customer_id, s.product_id order by customer_id) as count
7   from sales s
8   join menu m on s.product_id=m.product_id
9   order by customer_id, count desc) x )
10  select customer_id, pid, product_name, count
11  from cte
12  where count=max count
```

Result Grid:

| | customer_id | pid | product_name | count |
|---|-------------|-----|--------------|-------|
| ▶ | A | 3 | ramen | 3 |
| | B | 1 | sushi | 2 |
| | B | 2 | curry | 2 |
| | B | 3 | ramen | 2 |
| | C | 3 | ramen | 3 |

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

SQL query editor showing a query to find the first item purchased by a customer after they became a member.

```
1 • with cte as
2   (select s.customer_id as customer_id, product_id, order_date,
3         row_number() over(partition by s.customer_id order by order_date) as rn
4   from sales s
5   join members m on s.customer_id=m.customer_id
6   where order_date>= join_date)
7
8   select cte.customer_id, cte.product_id, product_name, order_date
9   from cte
10  join menu m on cte.product_id=m.product_id
11  where rn=1
12  order by cte.customer id
```

Result Grid:

| | customer_id | product_id | product_name | order_date |
|---|-------------|------------|--------------|------------|
| ▶ | A | 2 | curry | 2021-01-07 |
| | B | 1 | sushi | 2021-01-11 |

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

solution_dannys_diner* menu members sales members

Limit to 50000 rows

```
1 • with cte as
2   (select s.customer_id as customer_id, product_id, order_date,
3         rank() over(partition by s.customer_id order by order_date desc) as rn
4   from sales s
5   join members m on s.customer_id=m.customer_id
6   where order_date<join_date)
7
8   select cte.customer_id, cte.product_id, product_name, order_date
9   from cte
10  join menu m on cte.product_id=m.product_id
11  where rn=1
12  order by cte.customer id
```

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

| | customer_id | product_id | product_name | order_date |
|---|-------------|------------|--------------|------------|
| ▶ | A | 1 | sushi | 2021-01-01 |
| | A | 2 | curry | 2021-01-01 |
| | B | 1 | sushi | 2021-01-04 |

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

solution_dannys_diner* menu members sales members

```
1 • select distinct s.customer_id as customer_id,
2       count(s.product_id) over(partition by s.customer_id) as total_items,
3       sum(price) over(partition by s.customer_id) as total_amt
4   from sales s
5   join members m on s.customer_id=m.customer_id
6   join menu on s.product_id=menu.product_id
7   where order_date<join_date
8
9
10
11
12
```

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

| | customer_id | total_items | total_amt |
|---|-------------|-------------|-----------|
| ▶ | A | 2 | 25 |
| | B | 3 | 40 |

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

SQL query editor showing a query to calculate total points for each customer based on sales and menu items.

```
1 • with cte as
2   (select s.customer_id, s.product_id, product_name, price,
3     case when product_name='sushi' then price*20
4     when product_name!='sushi' then price*10
5     end as points
6   from sales s
7   join menu m on s.product_id=m.product_id)
8
9   select distinct customer_id, sum(points) over(partition by customer_id) as total_points
10  from cte
11
12
```

Result Grid:

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

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?

SQL query editor showing a query to calculate total points for each customer based on sales and menu items, considering a 2x multiplier for the first week after joining.

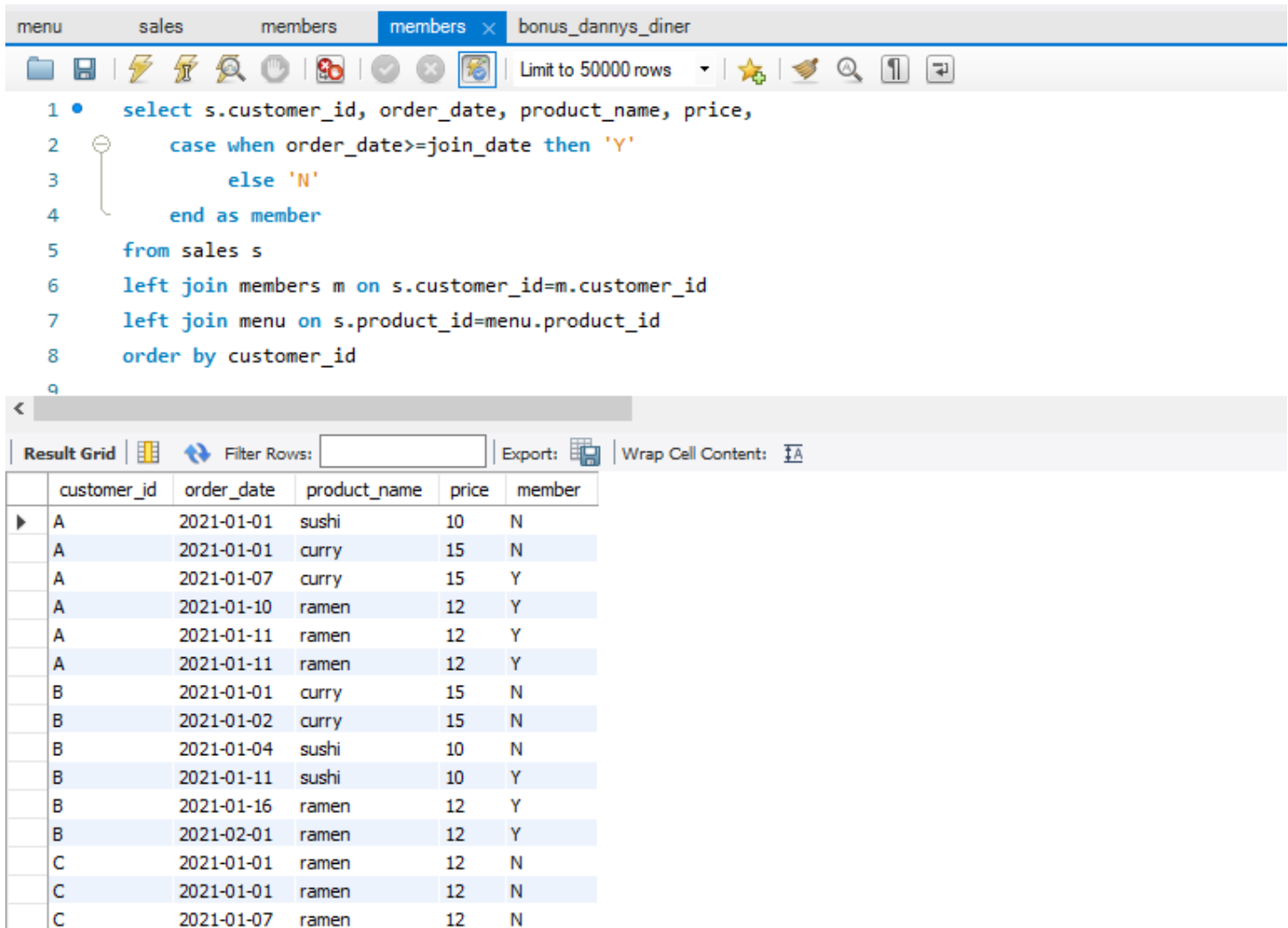
```
1 • select distinct customer_id, sum(points) over(partition by customer_id)
2   from
3   (select s.customer_id as customer_id, order_date, price*20 as points
4     from sales s
5     join members m on s.customer_id=m.customer_id
6     join menu on s.product_id=menu.product_id
7     where order_date>= join_date and month(order_date)=1) x
8
```

Result Grid:

| customer_id | sum(points) over(partition by customer_id) |
|-------------|--|
| A | 1020 |
| B | 440 |

Bonus Questions

Join All The Things



The screenshot shows a database query editor with a tab labeled 'members' selected. The query is as follows:


```
1 • select s.customer_id, order_date, product_name, price,  
2     case when order_date >= join_date then 'Y'  
3         else 'N'  
4     end as member  
5 from sales s  
6 left join members m on s.customer_id=m.customer_id  
7 left join menu on s.product_id=menu.product_id  
8 order by customer_id  
9
```

Below the query editor, the 'Result Grid' is displayed, showing the results of the query. The grid has columns for customer_id, order_date, product_name, price, and member. The results are as follows:




| | 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 |

Rank All The Things

menu sales members members x bonus_dannys_diner

 Limit to 50000 rows

```
1 • with cte as
2 (select s.customer_id, order_date, product_name, price,
3      case when order_date>=join_date then 'Y'
4      else 'N'
5      end as members
6 from sales s
7 left join members m on s.customer_id=m.customer_id
8 left join menu on s.product_id=menu.product_id
9 order by customer_id)
```

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

| | customer_id | order_date | product_name | price | members | 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 |

Insights

The following topics are completely covered in this case study:

- Common Table Expressions
- Group By and Aggregates
- Window Functions for Ranking and Row Number
- Table Joins
- Case When clause
- Subqueries
- Aggregates with Window Functions

The following insights can be gathered for this case study:

- Customer A has spend the maximum total amount i.e., \$76.
- Customer B is one of the most frequent customer.
- Ramen is the most demanded dish in the restaurant.
- Customer B has the maximum points i.e., 940 points in total, followed by customer A having 860 and Customer C having 360 points.