

DANNY'S DINER

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.

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.

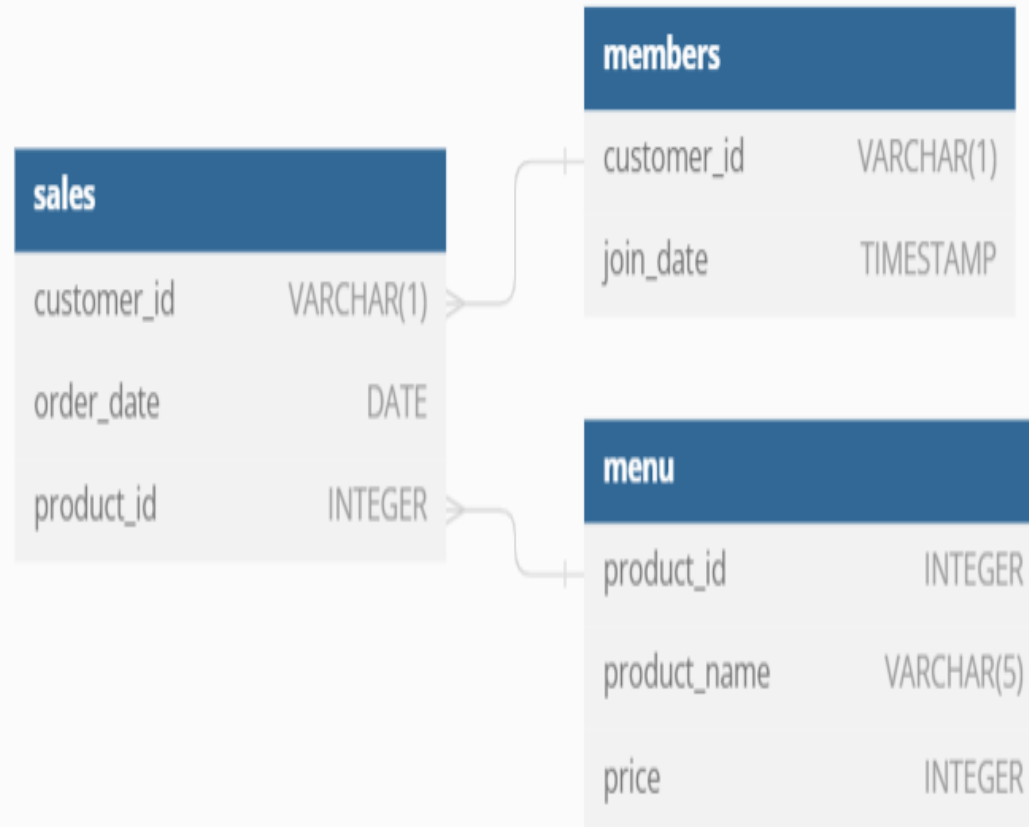
He plans on using these insights to help him decide whether he should expand the existing customer loyalty program - additionally he needs help to generate some basic datasets so his team can easily inspect the data without needing to use SQL.

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

Entity Relationship Diagram



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

QUERY

```
-- What is the total amount each customer spent at the restaurant?  
• SELECT s.customer_id,  
  SUM(m.price) AS Total_amount  
  FROM sales s  
  LEFT JOIN menu m  
  ON s.product_id = m.product_id  
  GROUP BY s.customer_id;
```

OUTPUT

	customer_id	Total_amount
▶	A	76
	B	74
	C	36

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

QUERY

```
67      -- How many days has each customer visited the restaurant?
68 •    SELECT customer_id,
69      COUNT(DISTINCT order_date) AS Total_visits
70      FROM sales
71      GROUP BY customer_id;
```

OUTPUT

Result Grid			Filter Rows
	customer_id	Total_visits	
▶	A	4	
	B	6	
	C	2	

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

QUERY

```
SELECT customer_id AS ID, product_name AS first_order FROM(  
  SELECT s.customer_id, m.product_name,  
    RANK() OVER(PARTITION BY s.customer_id ORDER BY s.order_date) AS rnk FROM sales s  
  LEFT JOIN menu m  
  ON s.product_id = m.product_id) AS Z  
WHERE Z.rnk = 1  
;
```

OUTPUT

Result Grid			Filter Rows:
	ID	first_order	
▶	A	sushi	
	A	curry	
	B	curry	
	C	ramen	
	C	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,
COUNT(m.product_name) AS Most_purchased
FROM sales s
LEFT JOIN menu m ON s.product_id = m.product_id
GROUP BY m.product_name
ORDER BY Most_purchased DESC;
```




##### OUTPUT

| Result Grid |              |                | Filter Rows: |
|-------------|--------------|----------------|--------------|
|             | product_name | Most_purchased |              |
| ▶           | ramen        | 8              |              |
|             | curry        | 4              |              |
|             | sushi        | 3              |              |

## QUERY

- ```
SELECT * FROM
(SELECT *, DENSE_RANK() OVER(ORDER BY Most_purchased DESC) AS RANKING FROM
(SELECT m.product_name, s.customer_id,COUNT(m.product_name) AS Most_purchased
FROM sales s
LEFT JOIN menu m ON s.product_id = m.product_id
GROUP BY m.product_name, s.customer_id
ORDER BY Most_purchased DESC) AS X) AS Subquery
WHERE RANKING =1;
```

OUTPUT

Result Grid			 Filter Rows:	<input type="text"/>	Export: 
	product_name	customer_id	Most_purchased	RANKING	
▶	ramen	A	3	1	
	ramen	C	3	1	

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

QUERY

```
• SELECT * FROM
  (SELECT * FROM
    (SELECT *,
      DENSE_RANK() OVER(PARTITION BY customer_id ORDER BY Total_orders DESC) AS RANKING
    FROM
      (SELECT COUNT(m.product_name) AS Total_orders,m.product_name, s.customer_id
      FROM sales s
      LEFT JOIN menu m ON s.product_id = m.product_id
      GROUP BY s.customer_id,m.product_name) AS Subquery) AS Z
    WHERE RANKING = 1)AS X;
```

OUTPUT

	Total_orders	product_name	customer_id	RANKING
▶	3	ramen	A	1
	2	curry	B	1
	2	sushi	B	1
	2	ramen	B	1
	3	ramen	C	1

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

QUERY

```
• SELECT * FROM
(
  SELECT s.customer_id,s.order_date,m.product_id,m.product_name,m2.join_date,
  DENSE_RANK() OVER(PARTITION BY s.customer_id ORDER BY s.order_date) AS RANKING
  FROM sales s
  LEFT JOIN menu m ON s.product_id = m.product_id
  LEFT JOIN members m2
  ON m2.customer_id = s.customer_id
  WHERE s.order_date >= m2.join_date)
AS Z
WHERE RANKING =1;
```

OUTPUT

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	customer_id	order_date	product_id	product_name	join_date	RANKING
▶	A	2021-01-07	2	curry	2021-01-07	1
	B	2021-01-11	1	sushi	2021-01-09	1

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

QUERY

```
• SELECT * FROM
  (
    SELECT s.customer_id,s.order_date,m.product_id,m.product_name,m2.join_date,
    DENSE_RANK() OVER(PARTITION BY s.customer_id ORDER BY s.order_date DESC) AS RANKING
    FROM sales s
    LEFT JOIN menu m ON s.product_id = m.product_id
    LEFT JOIN members m2
    ON m2.customer_id = s.customer_id
    WHERE s.order_date < m2.join_date)
  AS Z
  WHERE RANKING =1;
```

OUTPUT

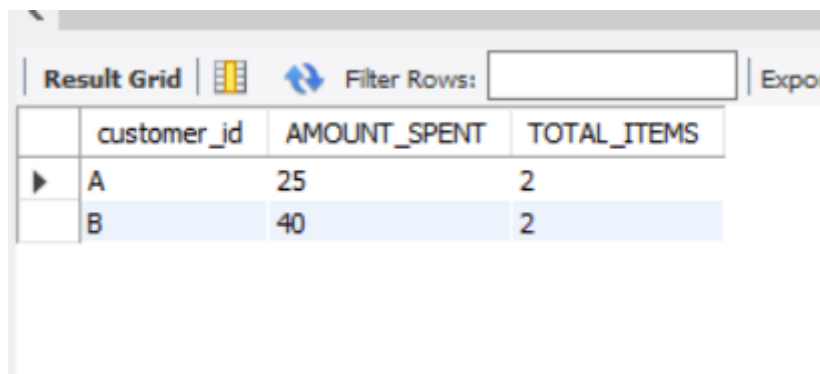
	customer_id	order_date	product_id	product_name	join_date	RANKING
▶	A	2021-01-01	1	sushi	2021-01-07	1
	A	2021-01-01	2	curry	2021-01-07	1
	B	2021-01-04	1	sushi	2021-01-09	1

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

QUERY

- ```
SELECT s.customer_id,SUM(m.price) AS AMOUNT_SPENT,
COUNT(DISTINCT m.product_id) AS TOTAL_ITEMS FROM sales s
LEFT JOIN menu m ON s.product_id = m.product_id
LEFT JOIN members m2
ON m2.customer_id = s.customer_id
WHERE s.order_date < m2.join_date
GROUP BY s.customer_id;
```

### OUTPUT



The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the results of the SQL query, with columns for customer\_id, AMOUNT\_SPENT, and TOTAL\_ITEMS. There are two rows of data: one for customer A and one for customer B. The interface also includes a 'Filter Rows' search bar and an 'Export' button.

|   | customer_id | AMOUNT_SPENT | TOTAL_ITEMS |
|---|-------------|--------------|-------------|
| ▶ | A           | 25           | 2           |
|   | B           | 40           | 2           |

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

## QUERY

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

OUTPUT

Result Grid			Filter Rows:
	customer_id	SUM(points_earned)	
▶	A	860	
	B	940	
	C	360	