

# DANNY MA's

## Case Study #1 - Danny's Diner

8WEEKSQLCHALLENGE.COM  
**CASE STUDY #1**



**THE TASTE OF SUCCESS**

**DATAWITHDANNY.COM**

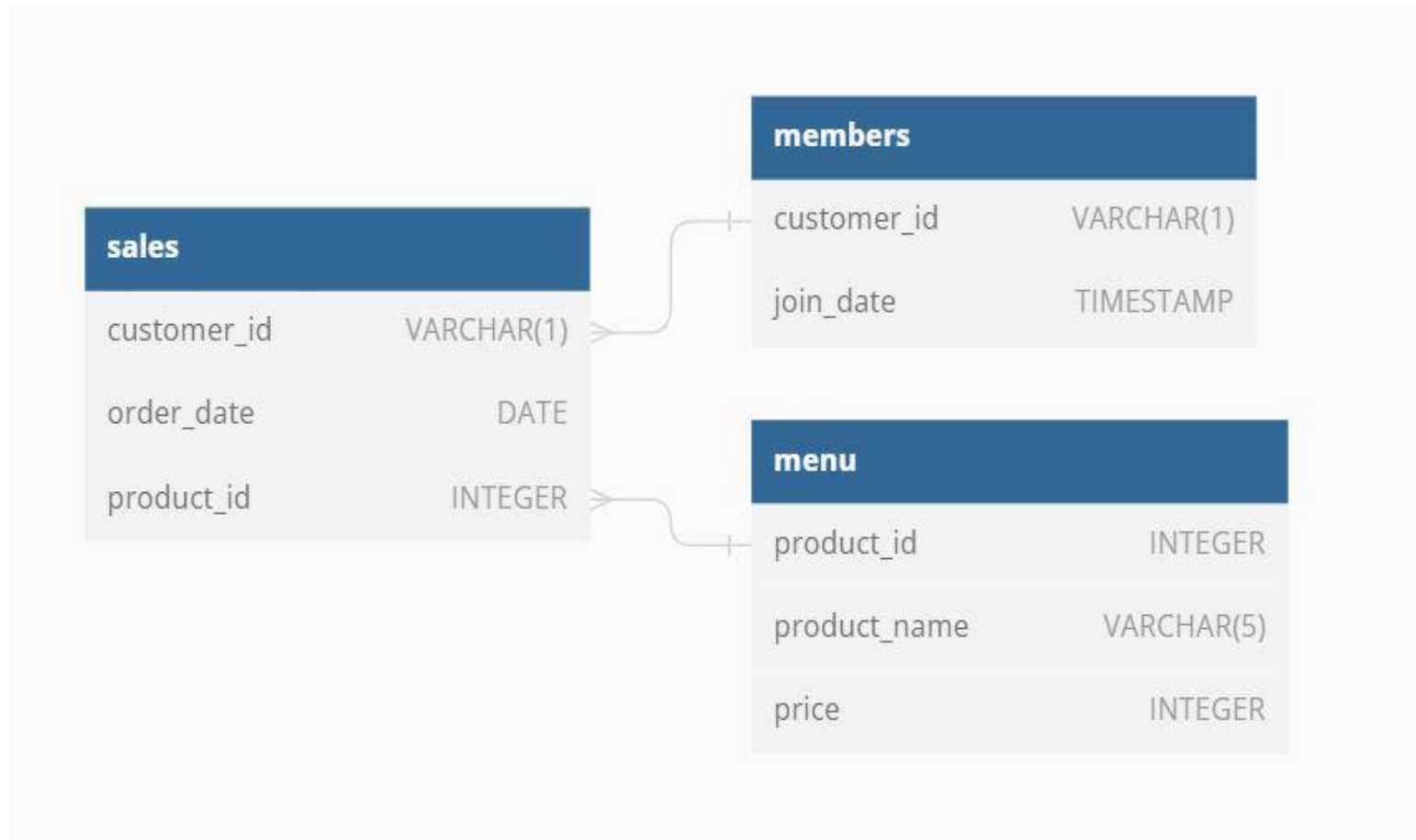
# Introduction

- 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.
- 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 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 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
- You can inspect the entity relationship diagram and example data below.

# Entity Relationship Diagram



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

Customer A spent the most (76)



```
--1) What is the total amount each customer spent at the restaurant?  
SELECT s.customer_id, sum(m.price) AS Total_Amount_per_Customer  
FROM sales s  
INNER JOIN menu m  
ON s.product_id = m.product_id  
GROUP BY s.customer_id;
```

Results Messages

	customer_id	Total_Amount_per_Customer
1	A	76
2	B	74
3	C	36

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

Customer A and B spent 6 days each while customer C spent 3days

```
--2) How many days has each customer visited the restaurant?  
SELECT s.customer_id, CONCAT(COUNT(s.order_date), ' days') AS  
       No_of_Days_Spent_at_the_restaurant  
GROUP BY customer_id;
```

Results		Messages
	customer_id	No_of_Days_Spent_at_the_restaurant
1	A	6 days
2	B	6 days
3	C	3 days

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

Customer A first purchased Curry

Customer B first purchased Curry

Customer C first purchased Ramen

```
--3) What was the first item from the menu purchased by each customer?
```

```
SELECT DISTINCT s.customer_id, FIRST_VALUE(m.product_name) OVER(PARTITION BY s.customer_id ORDER BY  
customer_id) AS first_item_ordered  
FROM  
    sales s  
INNER JOIN  
    menu m ON s.product_id = m.product_id  
GROUP BY  
    s.customer_id, m.product_name;
```

100 %

Results Messages

	customer_id	first_item_ordered
1	A	curry
2	B	curry
3	C	ramen

4. What is the most purchased item on the menu and how many times was it purchased by all customers?  
Ramen was the most ordered (8)

```
--4) What is the most purchased item on the menu and how many times was it purchased by all customers?  
SELECT TOP 1  
    m.product_name, COUNT(s.product_id) AS No_of_times_purchased  
FROM  
    sales s  
INNER JOIN  
    menu m ON s.product_id = m.product_id  
GROUP BY  
    s.product_id, m.product_name  
ORDER BY  
    No_of_times_purchased DESC;
```

Results Messages	
	product_name No_of_times_purchased
1	ramen 8



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

Customer A loved Ramen (ordered 3 times)

Customer B loved Sushi (ordered 2 times)

Customer C loved ramen (ordered 3 times)

```
--5) Which item was the most popular for each customer?
-- CTE to assign row_numbers according to the count of products, ordered in descending order
WITH count_of_product AS (
    SELECT s.customer_id, m.product_name, COUNT(s.product_id) AS product_count,
           ROW_NUMBER() OVER (PARTITION BY s.customer_id ORDER BY COUNT(s.product_id) 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
)
--selecting the item with rank 1
SELECT customer_id, product_count, product_name
FROM count_of_product
WHERE rn = 1;
```

	customer_id	product_count	product_name
1	A	3	ramen
2	B	2	sushi
3	C	3	ramen

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

Since Customers A and B are the only members:

Customer A first ordered

Curry.

while customer B first ordered Sushi after joining.

```
--6) Which item was purchased first by the customer after they became a member?
WITH customer_product_rank AS (
    SELECT s.customer_id, m.product_name, RANK() OVER(PARTITION BY s.customer_id ORDER BY
s.order_date) AS ranks
    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
    WHERE s.order_date >= me.join_date
    GROUP BY s.customer_id, s.order_date, m.product_name
)

SELECT customer_id, product_name
FROM customer_product_rank
WHERE ranks =1
```

	customer_id	product_name
1	A	curry
2	B	sushi

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

Customer A- sushi on 2021-01-01

customer B- sushi on 2021-01-04

```
--7) Which item was purchased just before the customer became a member?
SELECT
  s.customer_id,
  MAX(s.order_date) AS last_purchase_date,
  MAX(m.product_name) AS last_purchased_item,
  me.join_date
FROM
  sales AS s
INNER JOIN
  menu AS m ON s.product_id = m.product_id
INNER JOIN
  members AS me ON s.customer_id = me.customer_id
WHERE
  s.order_date < me.join_date
GROUP BY
  s.customer_id, me.join_date;
```

	customer_id	last_purchase_date	last_purchased_item	join_date
1	A	2021-01-01	sushi	2021-01-07
2	B	2021-01-04	sushi	2021-01-09

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

Customer A spent 25\$ on 2 products

Customer B spent 40\$ on 3 products

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

```
SELECT
  s.customer_id, COUNT(s.product_id) AS total_items, CONCAT(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
WHERE
  s.order_date < me.join_date
GROUP BY s.customer_id
```

	customer_id	total_items	amount_spent
1	A	2	25 \$
2	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?  
Customer A will have 860 points.  
Customer B will have 940 points.  
Customer A will have 360 points.

```
--9) If each $1 spent equates to 10 points and sushi has a 2x points multiplier
-- how many points would each customer have?
WITH customer_summary AS (
  SELECT
    s.customer_id,
    COUNT(s.product_id) AS total_items,
    SUM(m.price) AS amount_spent,
    (10 * SUM(m.price)) AS points,
    (
      SELECT CASE WHEN EXISTS (
        SELECT 1
        FROM menu
        WHERE product_id = s.product_id
        AND product_name = 'sushi'
      ) THEN 2 * (10 * SUM(m.price))
      ELSE 10 * SUM(m.price)
      END
    ) AS updated_points_with_sushi
  FROM
    sales s
  INNER JOIN
    menu m ON s.product_id = m.product_id
  GROUP BY
    s.customer_id, s.product_id
)
SELECT customer_id, SUM(updated_points_with_sushi) AS points_for_each_customer
FROM customer_summary
GROUP BY customer_id
```

	customer_id	points_for_each_customer
1	A	860
2	B	940
3	C	360

Activate Windows

- 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?

Customer A – 910 points

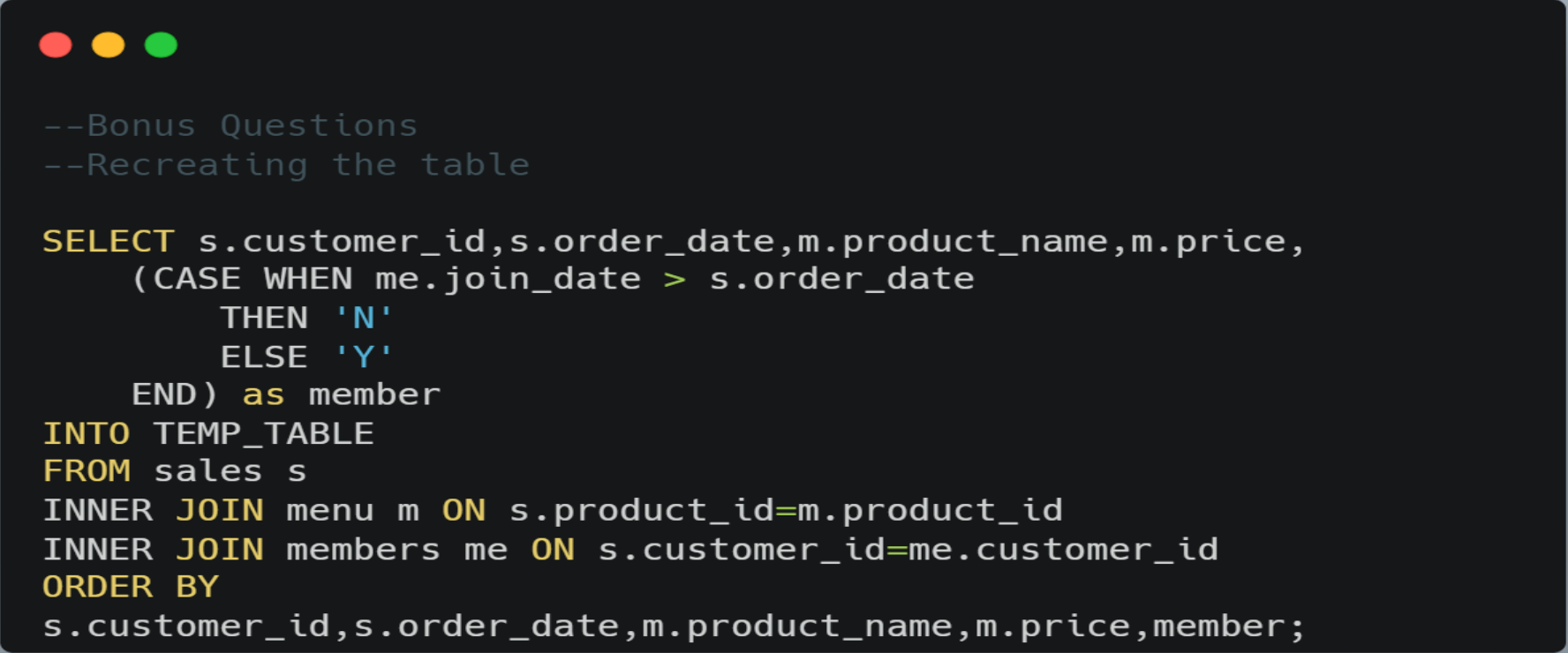
Customer B – 740 points

```
--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?

WITH points_ctes AS (
  SELECT
    s.customer_id,
    (
      CASE WHEN me.join_date <= s.order_date AND DATEPART(WEEK, me.join_date) = DATEPART(WEEK,
s.order_date)
        THEN 2 * (10 * SUM(m.price))
        ELSE (10*SUM(m.price))
      END
    ) AS points_during_join_week
  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
  group by s.customer_id, me.join_date ,s.order_date
)
SELECT customer_id, SUM(points_during_join_week) AS summed_points
FROM points_ctes
GROUP BY customer_id;
```

	customer_id	summed_points
1	A	910
2	B	740

--Bonus Questions  
--Recreating the table



```
--Bonus Questions
--Recreating the table

SELECT s.customer_id,s.order_date,m.product_name,m.price,
       (CASE WHEN me.join_date > s.order_date
            THEN 'N'
            ELSE 'Y'
            END) as member
INTO TEMP_TABLE
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
ORDER BY
s.customer_id,s.order_date,m.product_name,m.price,member;
```

## Rank All The Things

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.

```
--Rank All The Things
SELECT *, (
  CASE WHEN member = 'Y' THEN RANK() OVER (PARTITION BY customer_id,member ORDER BY
order_date)
  ELSE NULL
  END) as rankings
FROM TEMP_TABLE
```

	customer_id	order_date	product_name	price	member	rankings
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