Case Study #1 - Danny's Diner

8WEEKSQLCHALLENGE.COM

CASE STUDY #1

THE TASTE OF SUCCESS

DATAWITHDANNY.COM

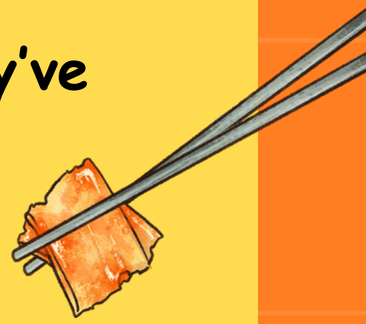
Introduction

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

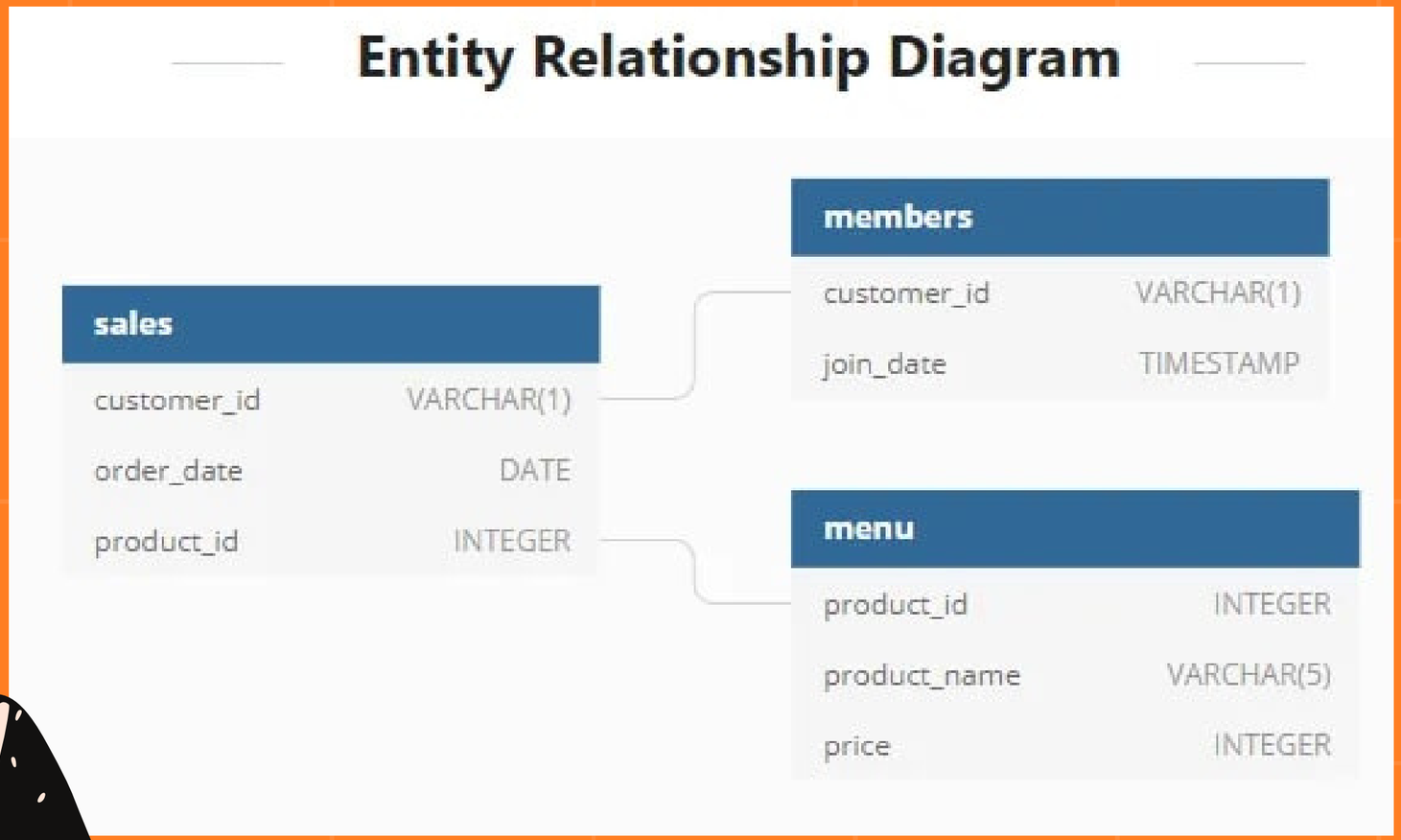
Danny wants to use the data to answer a few simple questions about his customers, especially about their visiting patterns, how much money they've spent and also which menu items are their favourite. Having this deeper connection with his customers will help him deliver a better and more personalised 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!

Entity Relationship Diagram

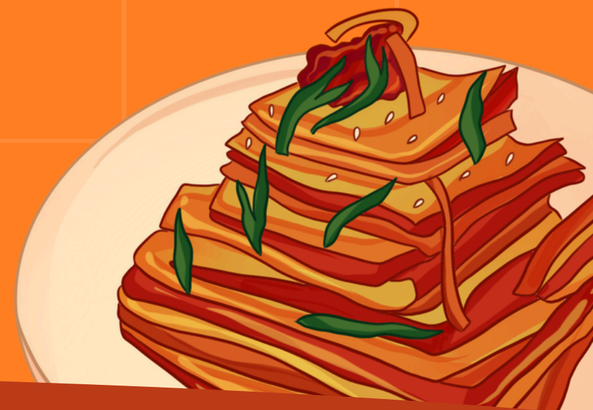


1. What is the total amount each customer spent at the restaurant?

```sql
select s.customer_id,
sum(m.price) as tot_spent
from sales as s
inner join menu m on m.product_id = s.product_id
group by 1;
```

Result Grid | Filter Rows:

customer_id	tot_spent
A	76
B	74
C	36



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

```sql
select customer_id ,
count(distinct order_date) as visited_restaurant
from sales
group by customer_id ;
```

customer_id	visited_restaurant
A	4
B	6
C	2

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

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

customer_id	product_name
A	sushi
B	curry
C	ramen

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

```sql
select m.product_name,
count(m.product_name) as no_of_times
from sales as s
inner join menu as m on m.product_id = s.product_id
group by m.product_name
order by no_of_times desc
limit 1;
```

product_name	no_of_times
ramen	8

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

```sql
SELECT customer_id, product_name
from (
select s.customer_id, m.product_name,
count(*) as order_count,
dense_rank() over (partition by customer_id order by count(*) desc) as rn
from sales s
inner join menu as m on m.product_id = s.product_id
group by s.customer_id, m.product_name
) as purchased_cnt
where rn = 1;
```

customer_id	product_name
A	ramen
B	curry
B	sushi
B	ramen
C	ramen

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

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

customer_id	product_name
A	ramen
B	sushi

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

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

customer_id	product_name
A	sushi
A	curry
B	curry

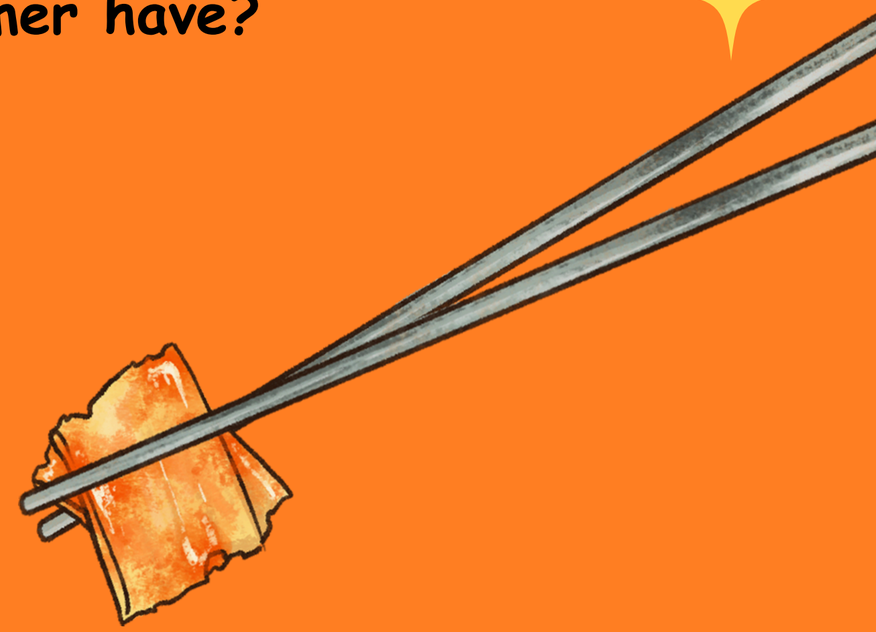
8. What is the total items and amount spent for each member before they became a member?

```sql
select s.customer_id,
count(m.product_id) as tot_items,
sum(m.price) as tot_spent
FROM menu AS m
INNER JOIN sales AS s ON m.product_id = s.product_id
JOIN members AS mb ON mb.customer_id = s.customer_id
WHERE s.order_date < mb.join_date
group by s.customer_id
order by s.customer_id;
```

customer_id	tot_items	tot_spent
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
SELECT s.customer_id,
    SUM(
        CASE
            WHEN m.product_name = 'Sushi' THEN m.price * 20
            ELSE m.price * 10
        END
    ) AS total_points
FROM
    sales AS s
INNER JOIN
    menu AS m ON s.product_id = m.product_id
GROUP BY
    s.customer_id;
```



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?

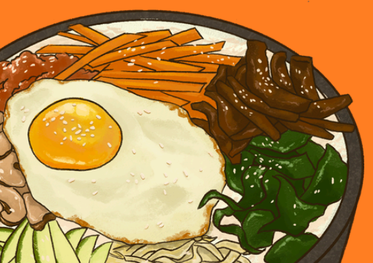
```sql
WITH purchase_points AS (
    SELECT
        s.customer_id,
        CASE
            WHEN s.order_date BETWEEN mb.join_date AND DATE_ADD(mb.join_date, INTERVAL 7 DAY) THEN m.price * 20
            WHEN m.product_name = 'sushi' THEN m.price * 20
            ELSE m.price * 10
        END AS points
     FROM menu AS m INNER JOIN sales AS s ON m.product_id = s.product_id
    INNER JOIN members AS mb ON mb.customer_id = s.customer_id
    WHERE
        s.order_date <= '2024-01-31')
SELECT
    customer_id,
    SUM(points) AS total_points
FROM
    purchase_points
    group by customer_id
    order by customer_id;
```

customer_id	total_points
A	1370
B	1060

11. Bonus Questions :Join All The Things

```sql
SELECT S.CUSTOMER_ID, ORDER_DATE, PRODUCT_NAME, PRICE,
     CASE
       WHEN ORDER_DATE <= JOIN_DATE THEN 'N'
    ELSE 'Y'
   END AS MEMBER
FROM SALES AS S
INNER JOIN MENU AS M ON S.PRODUCT_ID = M.PRODUCT_ID
LEFT JOIN MEMBERS AS MEM ON MEM.CUSTOMER_ID = S.CUSTOMER_ID
ORDER BY 1,2,3,4 DESC;
```

CUSTOMER_ID	ORDER_DATE	PRODUCT_NAME	PRICE	MEMBER
A	2021-01-01	curry	15	N
A	2021-01-01	sushi	10	N
A	2021-01-07	curry	15	N
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	Y
C	2021-01-01	ramen	12	Y
C	2021-01-07	ramen	12	Y



12.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.

```sql
WITH CTE AS (
SELECT S.CUSTOMER_ID, ORDER_DATE, PRODUCT_NAME, PRICE,
     CASE
       WHEN JOIN_DATE <= ORDER_DATE THEN 'Y'
    ELSE 'N'
   END AS MEMBER_STATUS
FROM SALES AS S
JOIN MENU AS M ON S.PRODUCT_ID = M.PRODUCT_ID
LEFT JOIN MEMBERS AS MEM ON MEM.CUSTOMER_ID = S.CUSTOMER_ID
)
SELECT *,
        CASE
            WHEN CTE.MEMBER_STATUS = 'Y'  THEN RANK() OVER(PARTITION BY CUSTOMER_ID, MEMBER_STATUS ORDER BY ORDER_DATE)
            ELSE NULL
       END AS RNK
   FROM CTE;
```

CUSTOMER_ID	ORDER_DATE	PRODUCT_NAME	PRICE	MEMBER
A	2021-01-01	curry	15	N
A	2021-01-01	sushi	10	N
A	2021-01-07	curry	15	N
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	Y
C	2021-01-01	ramen	12	Y

Thank You