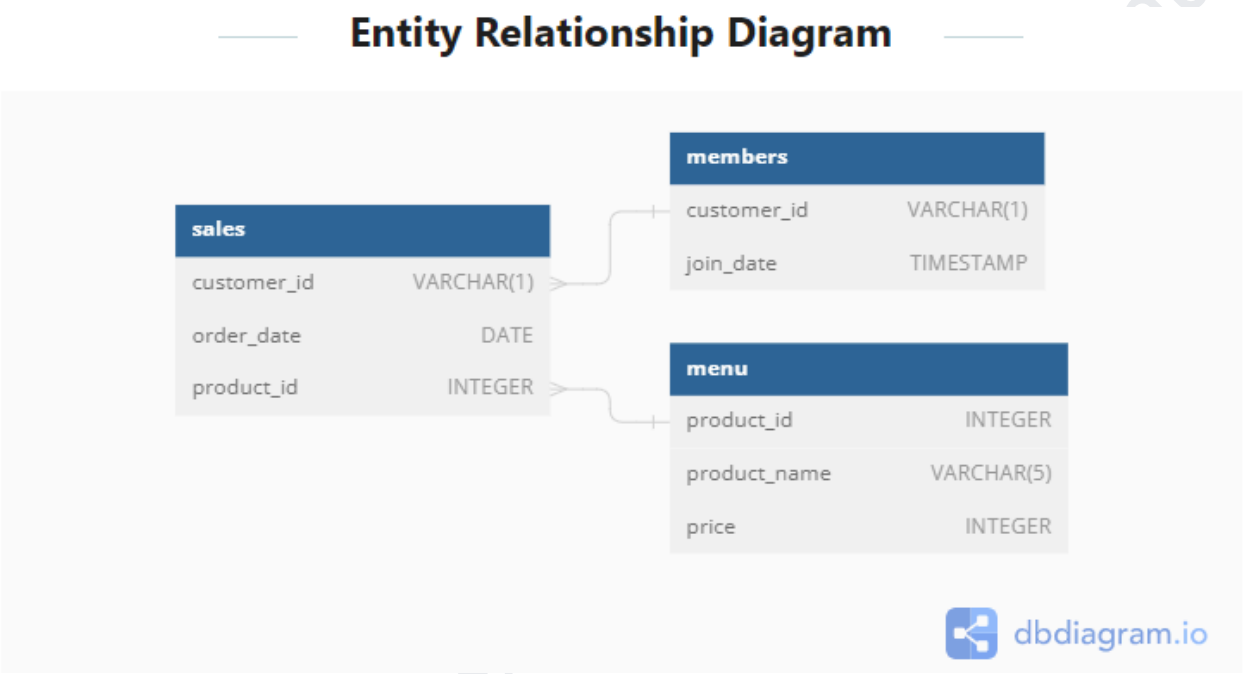
8 Week SQL Challenge - Case Study 1: Danny's Diner

Problem Statement

Danny, the owner of a small restaurant, seeks to understand his customer behavior and sales patterns to optimize his business operations. This analysis will help him make data-driven decisions regarding menu offerings, customer loyalty programs, and overall restaurant performance.

Data Schema

- **Sales:** Contains information about each sale, including customer ID, product ID, order date, and quantity.
- **Menu:** Lists all menu items with their corresponding product IDs and prices.
- **Members:** Contains details about customers who are members of the loyalty program, including customer ID and join date.



Analysis Questions

Customer Spending and Behavior

1. **Total Spending:** Calculate the total amount spent by each customer at the restaurant.
2. **Visit Frequency:** Determine the number of times each customer has visited the restaurant.
3. **First Purchase:** Identify the first item purchased by each customer.
4. **Purchased Items:** Find the most purchased item overall and for each individual customer.
5. **Popular Items:** Which item was the most popular for each customer?

Member Analysis

1. **First Purchase as a Member:** Determine the first item purchased by each customer after becoming a member.
2. **Last Purchase as a Non-Member:** Identify the last item purchased by each customer before becoming a member.
3. **Pre-Membership Spending:** Calculate the total amount spent by each member before joining the loyalty program.

Loyalty Program Evaluation

1. **Loyalty Points:** Calculate the total points earned by each customer based on a point system where $1 spent equals 10 points, with sushi earning double points.
2. **Bonus Points:** Determine the total points earned by customers A and B in the first week after joining the loyalty program, with double points applied to all items during this period.

SQL Queries

The provided SQL queries effectively address the outlined questions.

1. What is the total amount each customer spent at the restaurant?
2. How many days has each customer visited the restaurant?

```
SELECT SA.customer_id, FORMAT(SUM(ME.price), 'C', 'en-US') AS
Total_spent,
COUNT(*) AS No_Of_Visists
FROM Sales SA
LEFT JOIN Menu ME ON
SA.product_id = ME.product_id
GROUP BY SA.customer_id;
```

	customer_id	Total_spent	No_Of_Visists
1	A	$121.00	10
2	B	$128.00	10
3	C	$126.00	10
4	D	$212.00	17
5	E	$197.00	16
6	F	$185.00	15
7	G	$185.00	15

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

```
WITH CTE AS(
SELECT SA.customer_id,ME.product_name,
ROW_NUMBER() OVER(PARTITION BY SA.customer_id ORDER BY (SELECT NULL))
AS RN
FROM Sales SA
LEFT JOIN Menu ME ON
SA.product_id = ME.product_id
)
SELECT customer_id,product_name AS First_Item_Ordered
FROM CTE
WHERE RN = 1
```

	customer_id	First_Item_Ordered
1	A	sushi
2	B	curry
3	C	curry
4	D	sushi
5	E	ramen
6	F	sushi
7	G	curry

4. What is the most purchased item on the menu and how many times was it purchased by all customers?
5. Which item was the most popular for each customer?

```
WITH CTE AS(
SELECT DISTINCT SA.customer_id,ME.product_name,
COUNT(product_name) OVER(PARTITION BY customer_id,product_name) AS
Items
FROM Sales SA
LEFT JOIN Menu ME ON
SA.product_id = ME.product_id

)
```

```
, CTE2 AS(
SELECT customer_id,product_name,Items,
ROW_NUMBER() OVER(PARTITION BY customer_id ORDER BY Items DESC) AS RN
FROM CTE)

SELECT customer_id, Items AS No_Of_Times_Brought, product_name AS
Most_Populat_Item
FROM CTE2
WHERE RN = 1
```

	customer_id	No_Of_Times_Brought	Most_Populat_Item
1	A	4	sushi
2	B	4	curry
3	C	4	curry
4	D	6	curry
5	E	6	ramen
6	F	5	curry
7	G	5	curry

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

```
WITH CTE AS(
SELECT SA.customer_id, product_name, ROW_NUMBER() OVER(PARTITION BY
SA.customer_id ORDER BY order_date) AS RN
FROM Sales SA
LEFT JOIN Menu ME ON
SA.product_id = ME.product_id
LEFT JOIN Members M
ON SA.customer_id = M.customer_id
WHERE order_date > join_date)

SELECT customer_id, product_name AS Foabm
FROM CTE
WHERE RN = 1
```

	customer_id	Foabm
1	A	sushi
2	B	ramen
3	C	curry
4	D	ramen
5	E	curry
6	F	curry
7	G	ramen

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

```
WITH CTE AS(
SELECT SA.customer_id, product_name, ROW_NUMBER() OVER(PARTITION BY
SA.customer_id ORDER BY order_date) AS RN
FROM Sales SA
LEFT JOIN Menu ME ON
SA.product_id = ME.product_id
LEFT JOIN Members M
ON SA.customer_id = M.customer_id
WHERE order_date < join_date)

SELECT customer_id, product_name AS Fobbm
FROM CTE
WHERE RN = 1
```

	customer_id	Fobbm
1	B	curry
2	C	curry
3	D	sushi
4	E	ramen
5	F	sushi

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

```
SELECT SA.Customer_id,  FORMAT(SUM(ME.price), 'C', 'en-US') AS
Total_spent
FROM Sales SA
LEFT JOIN Menu ME ON
SA.product_id = ME.product_id
LEFT JOIN Members M
ON SA.customer_id = M.customer_id
WHERE order_date < join_date
GROUP BY SA.Customer_id
```

	Customer_id	Total_spent
1	B	$15.00
2	C	$15.00
3	D	$10.00
4	E	$12.00
5	F	$10.00

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

```
SELECT SA.customer_id,
SUM(IIF(product_name = 'sushi',(price * 20),(price * 10))) AS
Total_points
FROM Members Me
LEFT JOIN Sales Sa
ON Me.customer_id = Sa.customer_id
LEFT JOIN Menu M ON
M.product_id = Sa.product_id
GROUP BY Sa.customer_id
```

	customer_id	Total_points
1	A	1610
2	B	1480
3	C	1560
4	D	2620
5	E	2470
6	F	2350
7	G	2350

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?

```
SELECT  SA.customer_id,
(price * 20) AS Total_points
FROM Sales SA
LEFT JOIN Menu ME ON
SA.product_id = ME.product_id
LEFT JOIN Members M
ON SA.customer_id = M.customer_id
WHERE order_date > join_date AND DATEDIFF(DAY,join_date, order_date)
<= 7
AND Sa.customer_id IN ('A','B') AND MONTH(order_date) = 1
```

	customer_id	Total_points
1	A	200
2	B	240

Bonus Questions

Creating a Unified View

Join All The Things: This query combines data from the Sales, Menu, and Members tables into a single result set, providing a foundation for further analysis without requiring additional joins.

```
SELECT M.customer_id, order_date, product_name, price,
CASE WHEN order_date >= join_date THEN 'Y' ELSE 'N' END AS member
FROM Members M
JOIN Sales S
ON M.customer_id = S.customer_id
LEFT JOIN Menu Me ON
Me.product_id = S.product_id
```

	customer_id	order_date	product_name	price	member
1	A	2021-01-12	sushi	10	Y
2	A	2021-01-15	curry	15	Y
3	A	2021-01-20	sushi	10	Y
4	A	2021-01-22	ramen	12	Y
5	A	2021-02-01	sushi	10	Y
6	A	2021-02-05	curry	15	Y
7	A	2021-02-08	ramen	12	Y
8	A	2021-02-12	sushi	10	Y
9	A	2021-02-15	curry	15	Y
10	A	2021-02-18	ramen	12	Y
11	B	2021-01-05	curry	15	N
12	B	2021-01-10	ramen	12	Y
13	B	2021-01-20	curry	15	Y
14	B	2021-02-05	sushi	10	Y
15	B	2021-02-10	ramen	12	Y
16	B	2021-02-12	curry	15	Y
17	B	2021-02-14	ramen	12	Y
18	B	2021-02-16	sushi	10	Y
19	B	2021-02-20	curry	15	Y
20	B	2021-02-25	ramen	12	Y
21	C	2021-01-02	curry	15	N
22	C	2021-01-10	curry	15	Y
23	C	2021-01-15	sushi	10	Y
24	C	2021-01-25	ramen	12	Y
25	C	2021-02-01	sushi	10	Y
26	C	2021-02-04	curry	15	Y
27	C	2021-02-07	ramen	12	Y
28	C	2021-02-11	sushi	10	Y
29	C	2021-02-13	curry	15	Y
30	C	2021-02-17	ramen	12	Y
31	D	2021-01-01	sushi	10	N
32	D	2021-01-03	curry	15	Y
33	D	2021-01-04	ramen	12	Y
34	D	2021-01-05	sushi	10	Y
35	D	2021-01-07	curry	15	Y
36	D	2021-01-10	ramen	12	Y
37	D	2021-01-15	curry	15	Y
38	D	2021-01-18	ramen	12	Y
39	D	2021-01-20	sushi	10	Y
40	D	2021-01-25	curry	15	Y
41	D	2021-01-30	ramen	12	Y
42	D	2021-02-02	sushi	10	Y

Query executed successfully.

Adding Ranking Information

Rank All The Things: This query introduces a ranking column for each customer's purchases, considering only purchases made after becoming a member. This can be helpful for identifying top-selling items or customer preferences.

```
WITH CTE AS(
SELECT M.customer_id, order_date, product_name, price, join_date,
CASE WHEN order_date >= join_date THEN 'Y' ELSE 'N' END AS member
FROM Members M
JOIN Sales S
ON M.customer_id = S.customer_id
LEFT JOIN Menu Me ON
Me.product_id = S.product_id)

SELECT customer_id,order_date, product_name,price,member,
CASE WHEN order_date >= join_date THEN DENSE_RANK() OVER(PARTITION BY
customer_id ORDER BY price DESC) ELSE NULL END AS ranking
FROM CTE
ORDER BY customer_id,order_date
```

Results Messages

	customer_id	order_date	product_name	price	member	ranking
1	A	2021-01-12	sushi	10	Y	3
2	A	2021-01-15	curry	15	Y	1
3	A	2021-01-20	sushi	10	Y	3
4	A	2021-01-22	ramen	12	Y	2
5	A	2021-02-01	sushi	10	Y	3
6	A	2021-02-05	curry	15	Y	1
7	A	2021-02-08	ramen	12	Y	2
8	A	2021-02-12	sushi	10	Y	3
9	A	2021-02-15	curry	15	Y	1
10	A	2021-02-18	ramen	12	Y	2
11	B	2021-01-05	curry	15	N	NULL
12	B	2021-01-10	ramen	12	Y	2
13	B	2021-01-20	curry	15	Y	1
14	B	2021-02-05	sushi	10	Y	3
15	B	2021-02-10	ramen	12	Y	2
16	B	2021-02-12	curry	15	Y	1
17	B	2021-02-14	ramen	12	Y	2
18	B	2021-02-16	sushi	10	Y	3
19	B	2021-02-20	curry	15	Y	1
20	B	2021-02-25	ramen	12	Y	2
21	C	2021-01-02	curry	15	N	NULL
22	C	2021-01-10	curry	15	Y	1
23	C	2021-01-15	sushi	10	Y	3
24	C	2021-01-25	ramen	12	Y	2
25	C	2021-02-01	sushi	10	Y	3
26	C	2021-02-04	curry	15	Y	1
27	C	2021-02-07	ramen	12	Y	2
28	C	2021-02-11	sushi	10	Y	3
29	C	2021-02-13	curry	15	Y	1
30	C	2021-02-17	ramen	12	Y	2
31	D	2021-01-01	sushi	10	N	NULL
32	D	2021-01-03	curry	15	Y	1
33	D	2021-01-04	ramen	12	Y	2
34	D	2021-01-05	sushi	10	Y	3
35	D	2021-01-07	curry	15	Y	1
36	D	2021-01-10	ramen	12	Y	2
37	D	2021-01-15	curry	15	Y	1
38	D	2021-01-18	ramen	12	Y	2
39	D	2021-01-20	sushi	10	Y	3
40	D	2021-01-25	curry	15	Y	1
41	D	2021-01-30	ramen	12	Y	2
42	D	2021-02-02	sushi	10	Y	3
43	D	2021-02-05	curry	15	Y	1

Query executed successfully.

The End.

www.linkedin.com/in/viswanadh-gandimenu