Danny’s Dinner: Case Study

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 favorite 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 questions about his customers, especially about them:

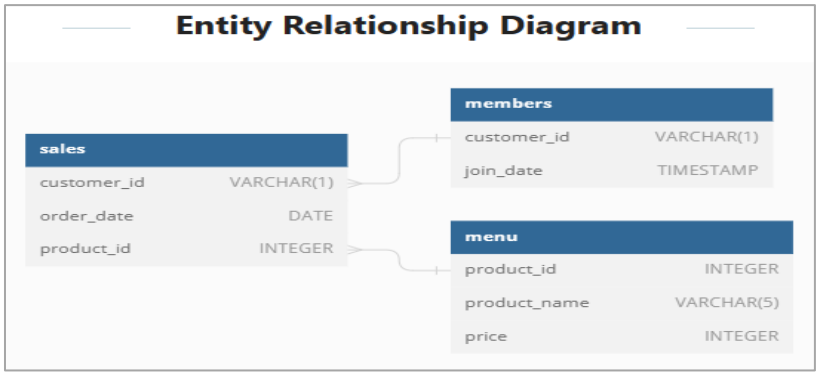
- visiting patterns,
- how much money they’ve spent, and
- which menu items are their favorite.

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.

The data set contains the following 3 tables which refer to the relationship diagram below to understand the connection.

- **Sales –** this dataset holds valuable information about the transactions that take place at Danny’s Diner, including the customer ID, menu items ordered and the order date.
- **Menu –** It encompasses all the delightful culinary creations offered at the restaurant including curry, ramen and sushi. It contains details such as item names, and their prices.
- **Members –** This dataset holds information about when customers joined the beta version of Danny’s loyalty program.



Case Study Analysis

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

```
SELECT
      s.customer_id, sum(m.price) As total_spent
FROM sales s
    JOIN menu m
ON s.product_id=m.product_id
GROUP BY s.customer_id;
```

Answer:

	customer_id	total_spent
1	A	76
2	B	74
3	C	36

- *Customer **A, B and C spent $76, $74 and $36** respectively. This makes **Customer A** the most valuable customer at the moment.*

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

```
SELECT
      customer_id, count(DISTINCT order_date) AS days_visited
FROM sales
GROUP BY customer_id;
```

Answer:

	customer_id	days_visited
1	A	4
2	B	6
3	C	2

- *Customer **A, B and C visited 4, 6 and 2 times** respectively.*

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

```
With Rank as(
    Select s.customer_id, m.product_name, s.order_date,
         DENSE_RANK() OVER (PARTITION BY s.customer_id Order by s.order_date) as rank
From Menu m
    Join Sales s
On m.product_id = s.product_id
Group by S.customer_id, M.product_name,S.order_date)
Select customer_id, product_name, order_date
From Rank
Where rank = 1;
```

Answer:

customer_id	product_name	order_date
A	curry	2021-01-01
A	sushi	2021-01-01
B	curry	2021-01-01
C	ramen	2021-01-01

- Customer **A's first order is curry and sushi.**
- Customer **B's first order is curry.**
- Customer **C's first order is ramen.**

All the three customers A, B & C purchased their first item on **2021-01-01.**

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(*) AS total_purchased
FROM sales s
    JOIN menu m
    ON s.product_id=m.product_id
GROUP BY m.product_name
ORDER BY total_purchased DESC;
```

Answer:

	product_name	total_purchased
1	ramen	8

- *Most purchased item on the menu is ramen which is 8 times.*

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

```
WITH customer_popularity AS (
  SELECT s.customer_id, m.product_name, COUNT(*) AS purchase_count,
        DENSE_RANK() OVER (PARTITION BY s.customer_id ORDER BY COUNT(*) DESC) AS rank
  FROM sales s
  JOIN menu m
    ON s.product_id = m.product_id
  GROUP BY s.customer_id, m.product_name)

SELECT customer_id, product_name, purchase_count
FROM customer_popularity
WHERE rank = 1;
```

Answer:

customer_id	product_name	purchase_count
A	ramen	3
B	sushi	2
B	curry	2
B	ramen	2
C	ramen	3

- *Customer **A and C's favorite item is ramen**.*
- *Customer **B savors** **all items** **sushi, ramen, and curry** on the menu.*

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

```
WITH first_purchase_after_membership AS (
    SELECT s.customer_id, MIN(s.order_date) as first_purchase_date
    FROM sales s
    JOIN members mb ON s.customer_id = mb.customer_id
    WHERE s.order_date >= mb.join_date
    GROUP BY s.customer_id)

SELECT fpam.customer_id, first_purchase_date, m.product_name
FROM first_purchase_after_membership fpam
JOIN sales s ON fpam.customer_id = s.customer_id
AND fpam.first_purchase_date = s.order_date
JOIN menu m ON s.product_id = m.product_id;
```

Answer:

customer_id	first_purchase_date	product_name
A	2021-01-07	curry
B	2021-01-11	sushi

- *After becoming members, the first purchase Customer A and Customer B made were curry and sushi respectively.*

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

```
WITH last_purchase_before_membership AS (
    SELECT s.customer_id, MAX(s.order_date) AS last_purchase_date
    FROM sales s
    JOIN members mb ON s.customer_id = mb.customer_id
    WHERE s.order_date < mb.join_date
    GROUP BY s.customer_id
)
SELECT lpbm.customer_id, last_purchase_date, m.product_name
FROM last_purchase_before_membership lpbm
JOIN sales s ON lpbm.customer_id = s.customer_id
AND lpbm.last_purchase_date = s.order_date
JOIN menu m ON s.product_id = m.product_id;
```

Answer:

customer_id	last_purchase_date	product_name
A	2021-01-01	sushi
A	2021-01-01	curry
B	2021-01-04	sushi

- *Just before becoming members, Customer **A bought sushi and curry** while Customer **B bought sushi.***

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

```
SELECT s.customer_id,
    COUNT(DISTINCT s.product_id) as total_items,
    SUM(m.price) AS total_spent
FROM sales s
    JOIN menu m ON s.product_id = m.product_id
    JOIN members mb ON s.customer_id = mb.customer_id
WHERE s.order_date < mb.join_date
GROUP BY s.customer_id;
```

Answer:

customer_id	total_items	total_spent
A	2	25
B	2	40

- *Before becoming members, Customer A had spent $25 on 2 items and Customer B, $40 on 2 items.*

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

```
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 s
    JOIN menu m ON s.product_id = m.product_id
GROUP BY s.customer_id;
```

Answer:

customer_id	total_points
A	860
B	940
C	360

- *A got 860 points, B got 940 points and C got 360 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?

```
SELECT s.customer_id, SUM(
    CASE
        WHEN s.order_date BETWEEN mb.join_date AND DATEADD(day, 7, mb.join_date) THEN m.price*20
        WHEN m.product_name = 'sushi' THEN m.price*20
        ELSE m.price*10
    END) AS total_points
FROM sales s
JOIN menu m ON s.product_id = m.product_id
LEFT JOIN members mb ON s.customer_id = mb.customer_id
WHERE s.customer_id = mb.customer_id AND s.order_date <= '2021-01-31'
GROUP BY s.customer_id;
```

Answer:

customer_id	total_points
A	1370
B	940

- *A got 1370 points and B got 820 points.*

Bonus Queries from Danny

1. Danny requires us to create basic data tables that his team can use to quickly derive insights without needing to write SQL. We need to recreate the following table output using the available data.

Recreate a Comprehensive Customer Data Table using the available data

	customer_id	order_date	product_name	price	member
1	A	2021-01-01	sushi	10	N
2	A	2021-01-01	curry	15	N
3	A	2021-01-07	curry	15	Y
4	A	2021-01-10	ramen	12	Y
5	A	2021-01-11	ramen	12	Y
6	A	2021-01-11	ramen	12	Y
7	B	2021-01-01	curry	15	N
8	B	2021-01-02	curry	15	N
9	B	2021-01-04	sushi	10	N
10	B	2021-01-11	sushi	10	Y
11	B	2021-01-16	ramen	12	Y
12	B	2021-02-01	ramen	12	Y
13	C	2021-01-01	ramen	12	N
14	C	2021-01-01	ramen	12	N
15	C	2021-01-07	ramen	12	N

Here is the SQL query that will reproduce the above table.

```
SELECT s.customer_id, s.order_date, m.product_name, m.price,
    (CASE
        WHEN s.order_date >= mb.join_date THEN 'Y'
        ELSE 'N'
        END) AS member
FROM sales s
    JOIN menu m ON s.product_id = m.product_id
    LEFT JOIN members mb ON s.customer_id = mb.customer_id
ORDER BY s.customer_id, s.order_date;
```

2. Danny also requires further information about the ranking of products. He purposely does not need the ranking of non-member purchases, so he expects NULL ranking values for customers who are not yet part of the loyalty program.

Rank the products of member customers

	customer_id	order_date	product_name	price	member	ranking
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
13	C	2021-01-01	ramen	12	N	NULL
14	C	2021-01-01	ramen	12	N	NULL
15	C	2021-01-07	ramen	12	N	NULL

Here's the query to rank the products of member customers.

```
WITH customers_data AS (
  SELECT s.customer_id, s.order_date,  m.product_name, m.price,
    CASE
      WHEN s.order_date < mb.join_date THEN 'N'
      WHEN s.order_date >= mb.join_date THEN 'Y'
      ELSE 'N'
      END AS member
  FROM sales s
  LEFT JOIN members mb
    ON s.customer_id = mb.customer_id
  JOIN menu m
    ON s.product_id = m.product_id)
SELECT *,
  CASE
    WHEN member = 'N' THEN NULL
    ELSE RANK () OVER(PARTITION BY customer_id, member ORDER BY order_date)
    END AS ranking
FROM customers_data
ORDER BY customer_id, order_date;
```

Insights from Case Study

From the analysis, I discovered the following interesting insights that would be certainly useful for Danny:

- Customer B is the most frequent visitor with 6 visits in Jan 2021.
- Danny's Diner's most popular item is ramen, followed by curry and sushi.
- Customer A and C loves ramen whereas Customer B seems to enjoy sushi, curry and ramen equally.
- Customer A is the 1st member of Danny's Diner and his first order is curry.
- The last item ordered by Customers A and B before they became members are sushi and curry. Does it mean both of these items are the deciding factor?
- Before they became members, Customer A and Customer B spent $25 and $40 respectively.
- Throughout Jan 2021, their points for Customer A: 860, Customer B: 940 and Customer C: 360.
- Assuming that members can earn 2x points a week from the day they became a member — not just sushi, Customer A has 1370 points and Customer B has 940 points by the end of Jan 2021.

I worked on Microsoft SQL server to solve the case study's queries.