Case Study #1 Danny's Diner

#8WeekSQLChallenge

Prepared by: Angela Boo

Introduction

Danny, a passionate Japanese food enthusiast, opened "Danny's Diner" in early 2021, specializing in sushi, curry, and ramen. To keep the restaurant thriving, he needs help analyzing basic operational data collected over the past few months.

Problem Statement

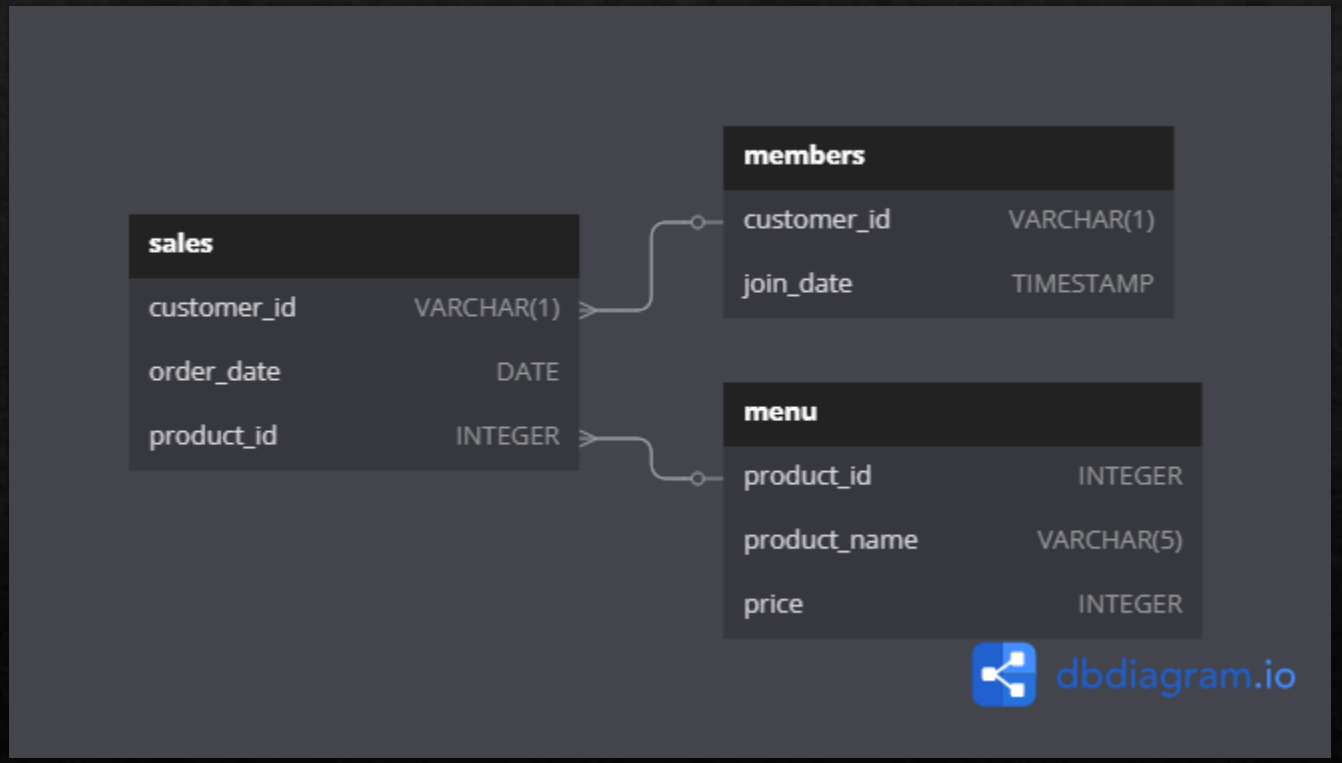
Danny aims to use the data to understand customer visiting patterns, spending habits, and favorite menu items. These insights will help enhance customer experiences and decide whether to expand the loyalty program. He also needs assistance in creating user-friendly datasets for his team to analyze without SQL.

Data Provided

Danny has shared three key datasets:

1. Sales
2. Menu
3. Members

Entity Relationship Diagram



Case Study Questions

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

Query

```
SELECT s.customer_id,
       SUM(m.price) AS
total_amount_spent
FROM sales s
JOIN menu m
ON s.product_id = m.product_id
GROUP BY s.customer_id
ORDER BY s.customer_id;
```

Output

customer_id	total_amount_spent
A	76
B	74
C	36

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

Query

```
SELECT customer_id,
       COUNT(DISTINCT order_date)
AS visit_count
FROM sales
GROUP BY customer_id;
```

Output

customer_id	visit_count
A	4
B	6
C	2

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

Query

```
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 row_num
  FROM sales s
  JOIN menu m
  ON s.product_id = m.product_id
) AS first_purchase
WHERE row_num = 1;
```

Output

customer_id	product_name
A	curry
B	curry
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(s.product_id) AS
total_purchase
FROM menu m
JOIN sales s
ON m.product_id = s.product_id
GROUP BY m.product_name
ORDER BY total_purchase DESC
LIMIT 1;
```

Output

product_name	total_purchase
ramen	8

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

Query

```
WITH customer_purchases AS (
    SELECT
        s.customer_id,
        m.product_name,
        COUNT(*) AS purchase_count,
        RANK() OVER (PARTITION BY customer_id ORDER BY COUNT(*) DESC) AS purchase_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_purchases
    WHERE purchase_rank = 1;
```

Output

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

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

Query

```
WITH first_purchase_after_joining AS (
  SELECT
    sales.customer_id,
    menu.product_name,
    sales.order_date,
    ROW_NUMBER() OVER (PARTITION BY sales.customer_id ORDER BY
sales.order_date) AS row_num
  FROM sales
  JOIN menu
  ON sales.product_id = menu.product_id
  JOIN members
  ON sales.customer_id = members.customer_id
  WHERE sales.order_date ≥ members.join_date
)

SELECT customer_id, product_name
FROM first_purchase_after_joining
WHERE row_num = 1;
```

Output

customer_id	product_name
A	curry
B	sushi

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

Query

```
WITH last_purchase_before_joining AS (
  SELECT
    sales.customer_id,
    menu.product_name,
    sales.order_date,
    ROW_NUMBER() OVER (PARTITION BY sales.customer_id ORDER BY
sales.order_date DESC) AS row_num
  FROM sales
  JOIN menu
  ON sales.product_id = menu.product_id
  JOIN members
  ON sales.customer_id = members.customer_id
  WHERE sales.order_date < members.join_date
)

SELECT customer_id, product_name
FROM last_purchase_before_joining
WHERE row_num = 1;
```

Output

customer_id	product_name
A	sushi
B	sushi

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

Query

```
WITH purchases_before_joining AS (
  SELECT
    sales.customer_id,
    COUNT(sales.product_id) AS total_items,
    SUM(menu.price) AS total_spent
  FROM sales
  JOIN menu
  ON sales.product_id = menu.product_id
  JOIN members
  ON sales.customer_id = members.customer_id
  WHERE sales.order_date < members.join_date
  GROUP BY sales.customer_id
  ORDER BY sales.customer_id
)

SELECT customer_id, total_items, total_spent
FROM purchases_before_joining;
```

Output

customer_id	total_items	total_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?

Query

```
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
ORDER BY s.customer_id;
```

Output

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?

Query

```
WITH customer_points AS (
    SELECT
        sales.customer_id,
        sales.order_date,
        menu.product_name,
        menu.price,
        members.join_date,
        CASE
            WHEN sales.order_date BETWEEN members.join_date AND (members.join_date + INTERVAL '6 days') THEN menu.price * 20
            WHEN menu.product_name = 'sushi' THEN menu.price * 20
            ELSE menu.price * 10
        END AS points
    FROM sales
    JOIN menu
    ON sales.product_id = menu.product_id
    JOIN members
    ON sales.customer_id = members.customer_id
    WHERE sales.order_date ≤ '2021-01-31'
)
SELECT
    customer_id,
    SUM(points) AS total_points
FROM customer_points
WHERE customer_id IN ('A', 'B')
GROUP BY customer_id
ORDER BY customer_id;
```

Output

customer_id	total_points
A	1370
B	820

Bonus question #1: Join All the Things

Query

```
SELECT
     sales.customer_id,
     sales.order_date,
     menu.product_name,
     menu.price,
     CASE
        WHEN members.join_date IS NULL THEN 'N'  -- Customer is not a member
        WHEN sales.order_date < members.join_date THEN 'N'  -- Order placed
before joining
        ELSE 'Y'  -- Order placed on or after joining
     END AS member
FROM sales
JOIN menu
ON sales.product_id = menu.product_id
LEFT JOIN members
ON sales.customer_id = members.customer_id
ORDER BY
    sales.customer_id,
    sales.order_date,
    menu.product_name;
```

Output

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	Y
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	N
C	2021-01-01	ramen	12	N
C	2021-01-07	ramen	12	N

Bonus question #2: Rank All the Things

Query

```
WITH customer_purchases AS (
    SELECT sales.customer_id, sales.order_date, menu.product_name, menu.price,
        CASE
            WHEN members.join_date IS NULL THEN 'N'  -- Customer is not a member
            WHEN sales.order_date < members.join_date THEN 'N'  -- Order placed before joining
            ELSE 'Y'  -- Order placed on or after joining
        END AS member
    FROM sales
    JOIN menu
    ON sales.product_id = menu.product_id
    LEFT JOIN members
    ON sales.customer_id = members.customer_id
)
SELECT
    customer_id, order_date, product_name, price, member,
    CASE
        WHEN member = 'Y' THEN
            RANK() OVER (PARTITION BY customer_id, member ORDER BY order_date)
        ELSE
            NULL
    END AS ranking
FROM customer_purchases
ORDER BY customer_id, order_date, product_name;
```

Output

customer_id	order_date	product_name	price	member	ranking
A	2021-01-01	curry	15	N	null
A	2021-01-01	sushi	10	N	null
A	2021-01-07	curry	15	Y	1
A	2021-01-10	ramen	12	Y	2
A	2021-01-11	ramen	12	Y	3
A	2021-01-11	ramen	12	Y	3
B	2021-01-01	curry	15	N	null
B	2021-01-02	curry	15	N	null
B	2021-01-04	sushi	10	N	null
B	2021-01-11	sushi	10	Y	1
B	2021-01-16	ramen	12	Y	2
B	2021-02-01	ramen	12	Y	3
C	2021-01-01	ramen	12	N	null
C	2021-01-01	ramen	12	N	null
C	2021-01-07	ramen	12	N	null

Key Insights

Customer Preferences

01

RAMEN IS UNIVERSALLY POPULAR

- All three customers (A, B, and C) purchased ramen the most, making it the most popular item across the board.
- Curry and sushi follow in popularity, with curry being slightly more popular than sushi.

02

VARIED PREFERENCES AMONG CUSTOMERS

- Customer A prefers ramen.
- Customer B has a more diverse preference, purchasing ramen, curry, and sushi equally.
- Customer C prefers ramen but has not explored other menu items.

Customer Behavior

01

LOYALTY PROGRAM DRIVES ENGAGEMENT

Members of the loyalty program (A and B) placed more orders and spent more on average compared to non-members (C).

02

FIRST WEEK INCENTIVE

Customers tend to spend more in the first week after joining the loyalty program, likely due to the 2x points incentive.

Business Recommendations

Suggestions for Danny's Diner

01

MENU OPTIMIZATION

Highlight ramen as the restaurant's signature dish, given its popularity among all customers.

02

PROMOTE LOYALTY PROGRAM

Target non-members (like customer C) by promoting the benefits of the loyalty program, such as "earn 2x points in your first week!"

03

CONTINUE 2X POINTS OFFER

Maintain the 2x points incentive in the first week after joining to encourage higher spending and increase customer engagement.

04

CUSTOMER SEGMENTATION

Group customers based on their spending habits and preferences to create tailored marketing strategies and personalized offers (e.g., discounts on favorite items).

Acknowledgments

- Thank you to DataWithDanny.com for the 8-Week SQL Challenge.
- Learn more about **Case Study #1 - Danny's Diner**: https://8weeksqlchallenge.com/case-study-1/