8 WEEKSQL CHALLENGE

8WEEKSQLCHALLENGE.COM

CASE STUDY #1

DANNY'S DINER

THE TASTE OF SUCCESS

DATAWITHDANNY.COM

TABLE OF CONTENTS

INTRODUCTION

Danny loves Japanese food and in early 2021, he opened a small restaurant called Danny's Diner. He serves his three favorite dishes: sushi, curry, and ramen. However, running the restaurant has been challenging, and Danny needs help to keep it going. Although he has collected some basic data from the first few months of operation, he isn't sure how to use it to improve his business. That's where we come in! Our goal is to analyze this data to provide insights and strategies that will help Danny's Diner thrive.

PROBLEM STATEMENT

Danny wants to leverage his restaurant's data to better understand his customers. Specifically, he aims to uncover insights about their visiting patterns, spending habits, and favorite menu items. This understanding will enable Danny to enhance customer experiences with personalized services.

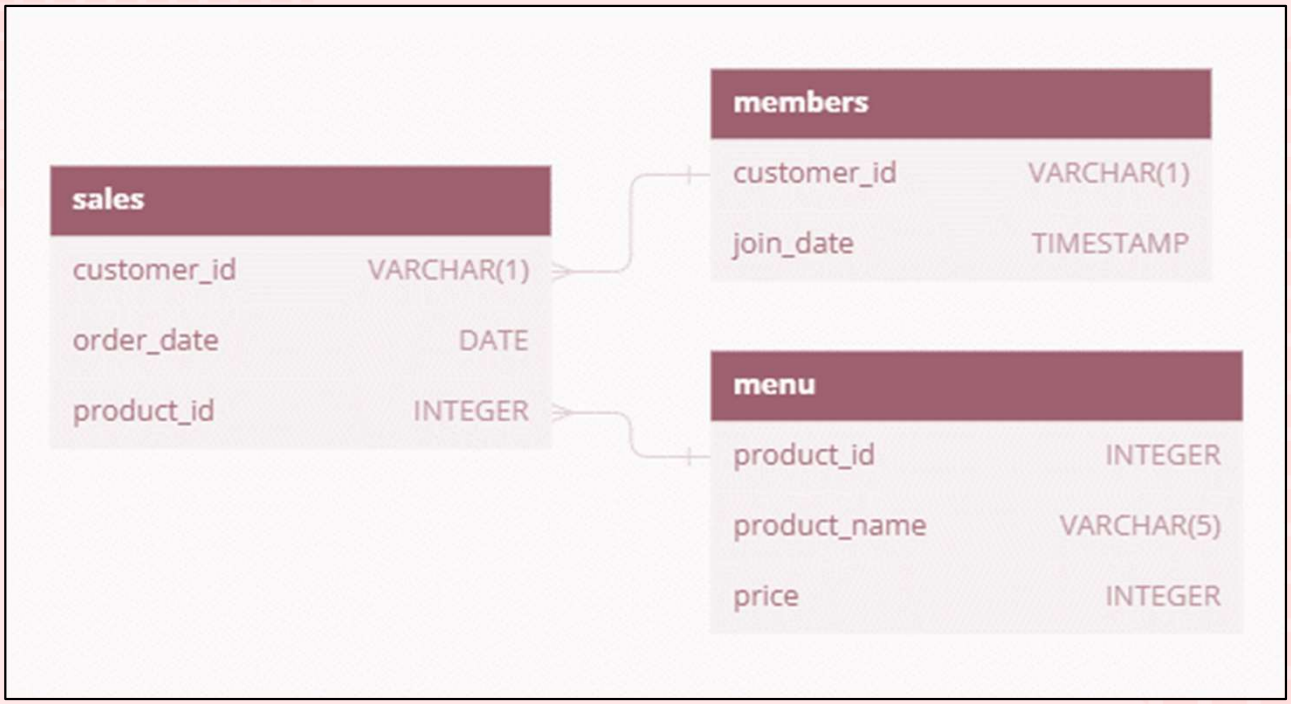
Furthermore, Danny seeks assistance in evaluating whether to expand his current customer loyalty program based on these insights. To facilitate easy data analysis for his team, he needs basic datasets prepared without requiring SQL expertise.

For this case study, Danny has provided three essential datasets:

i. sales
ii. menu
iii. members

These datasets offer a snapshot of customer behavior and preferences, which will be used to formulate SQL queries to address Danny's questions effectively.

ENTITY RELATIONSHIP DIAGRAM



DATASETS

Table 1: sales

The `sales` table captures all `customer_id` level purchases with an corresponding `order_date` and `product_id` information for when and what menu items were ordered.

customer_id	order_date	product_id
A	2021-01-01	1
A	2021-01-01	2
A	2021-01-07	2
A	2021-01-10	3
A	2021-01-11	3
A	2021-01-11	3
B	2021-01-01	2
B	2021-01-02	2
B	2021-01-04	1
B	2021-01-11	1
B	2021-01-16	3
B	2021-02-01	3
C	2021-01-01	3
C	2021-01-01	3
C	2021-01-07	3

Table 2: menu

The `menu` table maps the `product_id` to the actual `product_name` and `price` of each menu item.

product_id	product_name	price
1	sushi	10
2	curry	15
3	ramen	12

Table 3: members

The final `members` table captures the `join_date` when a `customer_id` joined the beta version of the Danny's Diner loyalty program.

customer_id	join_date
A	2021-01-07
B	2021-01-09

FINDING INSIGHTS USING SQL QUERIES

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

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

Results | Messages

customer_id	totalprice
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 days_visited
FROM sales
GROUP BY customer_id
ORDER BY customer_id ASC
```

Results | Messages

customer_id	days_visited
A	4
B	6
C	2

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

```sql
WITH first_item AS
(
  SELECT customer_id, order_date, product_id, DENSE_RANK() OVER(PARTITION BY customer_id ORDER BY order_date) AS rank_num
  FROM sales
)

SELECT f.customer_id, m.product_name
FROM first_item f
INNER JOIN .menu m
ON f.product_id = m.product_id
WHERE rank_num = 1
```

Results | Messages

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

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

```
SELECT TOP 1 s.product_id, m.product_name, COUNT(s.product_id) AS purchase_count
FROM sales s
JOIN menu m
ON s.product_id = m.product_id
GROUP BY m.product_name, s.product_id
ORDER BY purchase_count DESC
```

Results | Messages

product_id	product_name	purchase_count
3	ramen	8

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

```sql
WITH cte AS
(SELECT s.customer_id, s.product_id, m.product_name, COUNT(s.product_id) AS product_count,
        DENSE_RANK() OVER(PARTITION BY s.customer_id ORDER BY COUNT(s.product_id) DESC) AS rank_num
FROM sales s
JOIN menu m
ON s.product_id = m.product_id
GROUP BY s.customer_id, s.product_id, m.product_name)

SELECT customer_id, product_id, product_name, product_count
FROM cte
WHERE rank_num = 1
```

Results | Messages

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

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

```
WITH cte AS
(
  SELECT s.customer_id, s.order_date, s.product_id,
  DENSE_RANK() OVER(PARTITION BY  s.customer_id ORDER BY s.order_date)
  AS rank_num
  FROM sales s
  INNER JOIN members m
  ON s.customer_id = m.customer_id
  WHERE s.order_date >= m.join_date
)

SELECT c.customer_id, m.product_name
FROM cte c
JOIN menu m
ON c.product_id = m.product_id
WHERE rank_num = 1
ORDER BY c.customer_id
```

Results | Messages

customer_id	product_name
A	curry
B	sushi

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

```
WITH orderbeforemember AS
(
  SELECT s.customer_id, s.order_date, s.product_id,
  RANK() OVER(PARTITION BY s.customer_id ORDER BY s.order_date) AS
  rank_num
  FROM sales s
  JOIN members m
  ON s.customer_id = m.customer_id
  WHERE s.order_date < m.join_date
)

SELECT o.customer_id, m.product_name
FROM orderbeforemember o
JOIN menu m
ON o.product_id = m.product_id
WHERE o.rank_num = 1
ORDER BY o.customer_id
```

Results | Messages

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?

```
WITH totalorderbeforemember AS
(
  SELECT s.customer_id, s.order_date, s.product_id
  FROM sales s
  JOIN members m
  ON s.customer_id = m.customer_id
  WHERE s.order_date < m.join_date
)


SELECT t.customer_id, COUNT(*) AS total_items, SUM(m.price) AS amount_spend
FROM totalorderbeforemember t
JOIN menu m
ON t.product_id = m.product_id
GROUP BY t.customer_id
ORDER BY t.customer_id
```

Results | Messages

customer_id	total_items	amount_spend
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?

```
SELECT s.customer_id,
       SUM(CASE WHEN m.product_name = 'sushi' THEN m.price*20
       ELSE m.price*10
       END) AS points
FROM sales s
JOIN menu m
ON s.product_id = m.product_id
GROUP BY s.customer_id
```

Results | Messages

customer_id	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
SELECT s.customer_id,SUM( mu.price * 20) AS points
FROM sales s
JOIN menu mu
ON s.product_id = mu.product_id
JOIN members m
ON s.customer_id = m.customer_id
WHERE s.order_date >= m.join_date AND MONTH(s.order_date) = 1
GROUP BY s.customer_id, m.join_date
```

Results | Messages

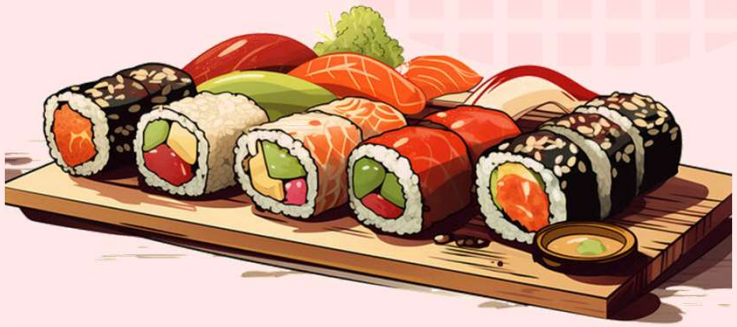
customer_id	points
A	1020
B	440

BONUS QUESTIONS – JOIN ALL THE THINGS

```sql
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
LEFT 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 IN ('A','B','C')
ORDER BY s.customer_id, s.order_date
```

Results | Messages

customer_id	order_date	product_name	price	member
A	2021-01-01	sushi	10	N
A	2021-01-01	curry	15	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 QUESTIONS – RANK ALL THE THINGS

```sql
WITH productrank AS
( 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
  LEFT 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 IN ('A','B','C')),

productrank1 AS
( SELECT customer_id, order_date, product_name, price, member,
       DENSE_RANK() OVER(PARTITION BY customer_id, member ORDER BY order_date) AS rank_num
  FROM productrank )

SELECT customer_id, order_date, product_name, price, member,
       CASE WHEN member = 'N' THEN NULL
       ELSE rank_num
       END AS ranking
FROM productrank1
ORDER BY customer_id, order_date;
```

customer_id	order_date	product_name	price	member	ranking
A	2021-01-01	sushi	10	N	NULL
A	2021-01-01	curry	15	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 from the Case Study

1. Customer A spent the most at the restaurant, totaling $76, followed by Customer B with $74, and Customer C with $36.

2. Customer B visited the restaurant the most frequently.

3. Ramen is the most purchased item on the menu.

4. Customer B has the highest loyalty points with 940, followed by Customer A with 860, and Customer C with 360.

THANKS!