

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.

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 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.

Danny has shared 3 key datasets & SQL Scripts for this case study:

Sales

Menu

Members

SQL SCRIPT

```
CREATE dannys_diner;  
USE dannys_diner;
```

```
CREATE TABLE sales ( "customer_id" VARCHAR(1), "order_date"  
DATE, "product_id" INTEGER);
```

```
INSERT INTO sales
```

```
  ("customer_id", "order_date", "product_id")
```

```
VALUES
```

```
  ('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');
```

SQL SCRIPT CONTD.

```
CREATE TABLE menu (  
    "product_id" INTEGER,  
    "product_name" VARCHAR(5),  
    "price" INTEGER  
);
```

```
INSERT INTO menu  
    ("product_id", "product_name", "price")  
VALUES  
    ('1', 'sushi', '10'),  
    ('2', 'curry', '15'),  
    ('3', 'ramen', '12');
```

SQL SCRIPT CONTD.

```
CREATE TABLE members (  
    "customer_id" VARCHAR(1),  
    "join_date" DATE  
);
```

```
INSERT INTO members  
    ("customer_id", "join_date")  
VALUES  
    ('A', '2021-01-07'),  
    ('B', '2021-01-09');
```

CASE STUDY QUESTIONS

- 1) What is the total amount each customer spent at the restaurant?
- 2) How many days has each customer visited the restaurant?
- 3) What was the first item from the menu purchased by each customer?
- 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?
- 6) Which item was purchased first by the customer after they became a member?
- 7) Which item was purchased just before the customer became a member?
- 8) What is the total items and amount spent for each member before they became a member?
- 9) If each \$1 spent equates to 10 points and sushi has a 2x points multiplier - how many points would each customer have?
- 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?
- 11) Determine the name and price of the product ordered by each customer on all order date and find out whether the customer was a member on the order date or not.
- 12) Rank the previous output from Q. 11 based on the order date for each customer. Display NULL if customer was not a member on the ordered date.

Q1: WHAT IS THE TOTAL AMOUNT EACH CUSTOMER SPENT AT THE RESTAURANT?

--What is the total amount each customer spent at the restaurant?

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

100 %



Results



Messages

	customer_id	total_amount
1	A	76
2	B	74
3	C	36

Q2: HOW MANY DAYS HAS EACH CUSTOMER VISITED THE RESTAURANT?

--How many days has each customer visited the restaurant?

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

100 %



Results



Messages

	customer_id	num_days
1	A	4
2	B	6
3	C	2

Q3: WHAT WAS THE FIRST ITEM FROM THE MENU PURCHASED BY EACH CUSTOMER?

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

```
WITH CTE AS
(
  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 AS s
  JOIN menu AS m
    ON s.product_id = m.product_id
)
SELECT customer_id, product_name
FROM CTE WHERE row_num = 1;
```

100 %



Results



Messages

	customer_id	product_name
1	A	sushi
2	B	curry
3	C	ramen

Q4:
WHAT IS THE MOST PURCHASED ITEM ON THE MENU AND
HOW MANY TIMES WAS IT PURCHASED BY ALL CUSTOMERS?

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

100 %

Results Messages

	product_name	order_count
1	ramen	8

Q5: WHICH ITEM WAS THE MOST POPULAR FOR EACH CUSTOMER?

--Which item was the most popular for each customer?

```
WITH item_count AS
(
    SELECT
        s.customer_id,
        m.product_name,
        COUNT(*) AS order_count,
        DENSE_RANK() OVER(PARTITION BY s.customer_id ORDER BY COUNT(*) DESC) AS ranking
    FROM
        sales AS s
        JOIN menu AS m
        ON s.product_id = m.product_id
    GROUP BY s.customer_id, m.product_name)
SELECT
    customer_id,
    product_name
FROM
    item_count
WHERE ranking = 1
```

100 %

Results Messages

	customer_id	product_name
1	A	ramen
2	B	sushi
3	B	curry
4	B	ramen
5	C	ramen

Q6: WHICH ITEM WAS PURCHASED FIRST BY THE CUSTOMER AFTER THEY BECAME A MEMBER?

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

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

100 %



Results



Messages

	customer_id	product_name
1	A	ramen
2	B	sushi

Q7: WHICH ITEM WAS PURCHASED JUST BEFORE THE CUSTOMER BECAME A MEMBER?

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

```
SELECT
    customer_id,
    product_name
FROM
    (SELECT
        s.customer_id,
        m.product_name,
        mb.join_date,
        s.order_date,
        DENSE_RANK() OVER(PARTITION BY s.customer_id ORDER BY order_date DESC) AS row_num
    FROM
        sales AS s
        JOIN menu AS m ON s.product_id = m.product_id
        JOIN members AS mb ON mb.customer_id = s.customer_id
        WHERE mb.join_date > s.order_date) AS pre_mb_orders
WHERE row_num = 1;
```

100 %

Results

Messages

	customer_id	product_name
1	A	sushi
2	A	cunyu
3	B	sushi

Q8:
WHAT IS THE TOTAL ITEMS AND AMOUNT SPENT FOR EACH
MEMBER BEFORE THEY BECAME A MEMBER?

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

```
SELECT
    s.customer_id,
    COUNT(m.product_name) num_of_items,
    SUM(m.price) AS amount_spent
FROM
    sales AS s
    JOIN menu AS m ON s.product_id = m.product_id
    JOIN members AS mb ON s.customer_id = mb.customer_id
WHERE s.order_date < mb.join_date
GROUP BY s.customer_id
```

100 %



Results



Messages

	customer_id	num_of_items	amount_spent
1	A	2	25
2	B	3	40

Q9:
IF EACH \$1 SPENT EQUATES TO 10 POINTS AND SUSHI HAS A
2X POINTS MULTIPLIER - HOW MANY POINTS WOULD EACH
CUSTOMER HAVE?

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

```
SELECT  
    customer_id,  
    SUM(points) AS total_point  
FROM  
    (  
        SELECT  
            s.customer_id,  
            m.product_name,  
            m.price,  
            CASE  
                WHEN m.product_name = 'sushi' THEN m.price*10*2  
                ELSE m.price*10  
            END AS points  
        FROM  
            sales AS s  
            JOIN menu AS m  
            ON s.product_id = m.product_id) AS points  
GROUP BY customer_id
```

100 %



Results



Messages

	customer_id	total_point
1	A	860
2	B	940
3	C	360

Q10:
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?

```
--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
    customer_id,
    SUM(points) AS total_point
FROM
    (SELECT
        s.customer_id,
        m.product_name,
        m.price,
        s.order_date,
        mb.join_date,
        CASE
            WHEN s.order_date BETWEEN mb.join_date AND DATEADD(day,7,mb.join_date) THEN m.price*10*2
            WHEN m.product_name = 'sushi' THEN m.price*10*2
            ELSE m.price*10
        END AS points
    FROM
        sales AS s
        JOIN menu AS m ON s.product_id = m.product_id
        JOIN members AS mb ON mb.customer_id = s.customer_id
    WHERE s.order_date < '2021-02-01') AS points
GROUP BY customer_id
```

100 %

Results Messages

	customer_id	total_point
1	A	1370
2	B	940

Q11:
DETERMINE THE NAME AND PRICE OF THE PRODUCT
ORDERED BY EACH CUSTOMER ON ALL ORDER DATE AND
FIND OUT WHETHER THE CUSTOMER WAS A MEMBER ON THE
ORDER DATE OR NOT.

```
--Determine the name and price of the product ordered by each customer on all order date and  
--find out whether the customer was a member on the order date or not.  
SELECT  
    s.customer_id,  
    m.product_name,  
    m.price,  
    s.order_date,  
    CASE  
        WHEN s.order_date >= mb.join_date THEN 'Yes'  
        ELSE 'No'  
    END AS [member]  
FROM  
    sales AS s  
    JOIN menu AS m ON s.product_id = m.product_id  
    LEFT JOIN members AS mb ON s.customer_id = mb.customer_id
```

100 %

Results

Messages

	customer_id	product_name	price	order_date	member
1	A	sushi	10	2021-01-01	No
2	A	curry	15	2021-01-01	No
3	A	curry	15	2021-01-07	Yes
4	A	ramen	12	2021-01-10	Yes
5	A	ramen	12	2021-01-11	Yes
6	A	ramen	12	2021-01-11	Yes
7	B	curry	15	2021-01-01	No
8	B	curry	15	2021-01-02	No
9	B	sushi	10	2021-01-04	No
10	B	sushi	10	2021-01-11	Yes
11	B	ramen	12	2021-01-16	Yes

Q12:
RANK THE PREVIOUS OUTPUT FROM Q. 11 BASED ON THE
ORDER DATE FOR EACH CUSTOMER. DISPLAY NULL IF
CUSTOMER WAS NOT A MEMBER ON THE ORDERED DATE.

```
--Rank the previous output from Q. 11 based on the order date for each customer.
--Display NULL if customer was not a member on the ordered date.

WITH member_status AS

    (SELECT s.customer_id, m.product_name, m.price, s.order_date,
        CASE
            WHEN s.order_date >= mb.join_date THEN 'Yes'
            ELSE 'No'
        END AS [member]
    FROM
        sales AS s
        JOIN menu AS m ON s.product_id = m.product_id
        LEFT JOIN members AS mb ON s.customer_id = mb.customer_id)

SELECT *,
    CASE
        WHEN member_status.[member] = 'Yes' THEN RANK() OVER(PARTITION BY customer_id, [member] ORDER BY order_date)
        ELSE NULL
    END AS ranking
FROM member_status
```

100 %

Results Messages

	customer_id	product_name	price	order_date	member	ranking
1	A	sushi	10	2021-01-01	No	NULL
2	A	curry	15	2021-01-01	No	NULL
3	A	curry	15	2021-01-07	Yes	1
4	A	ramen	12	2021-01-10	Yes	2
5	A	ramen	12	2021-01-11	Yes	3
6	A	ramen	12	2021-01-11	Yes	3
7	B	curry	15	2021-01-01	No	NULL
8	B	curry	15	2021-01-02	No	NULL

THANK YOU