

8WEEKSQLCHALLENGE.COM
CASE STUDY #1



THE TASTE OF SUCCESS

DATAWITHDANNY.COM

Made by: *Mai Huy Khang*

Note: *all results use for practicing SQL (MySQL) only*

INTRODUCTION

❖ Context

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 needs 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 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 - additionally he needs help to generate some basic datasets so his team can easily inspect the data without needing to use SQL.

CHAPTER 1: CREATE DATASET AS REQUIRED

❖ Coding part

```
CREATE SCHEMA dannys_diner;  
SET search_path = 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');
```

```
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');
```

```
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');
```

CHAPTER 2: SOLVE CASE STUDY QUESTION

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

```
SELECT customer_id,  
       sum(price) as total_amount  
FROM sales as s  
LEFT JOIN menu as m  
USING(product_id)  
GROUP BY customer_id
```

customer_id	total_amount
A	76
B	74
C	36

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

```
SELECT customer_id,  
       count(distinct order_date) as visited_frequency  
FROM sales  
GROUP BY customer_id
```

customer_id	visited_frequency
A	4
B	6
C	2

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

```
SELECT s.customer_id,  
       min(s.order_date),  
       m.product_name as first_item  
FROM sales as s  
LEFT JOIN menu as m  
USING(product_id)  
GROUP BY s.customer_id
```

customer_id	min(s.order_date)	first_item
A	2021-01-01	sushi
B	2021-01-01	curry
C	2021-01-01	ramen

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

```
SELECT m.product_name,  
       count(s.product_id) as times  
FROM sales as s  
LEFT JOIN menu as m  
USING(product_id)  
GROUP BY m.product_name  
ORDER BY times DESC limit 1
```

product_name	times
ramen	8

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

```
SELECT s.customer_id,  
       m.product_name,  
       count(s.product_id) as times  
FROM sales as s  
LEFT JOIN menu as m  
USING(product_id)  
GROUP BY s.customer_id  
ORDER BY times DESC
```

customer_id	product_name	times
A	sushi	6
B	curry	6
C	ramen	3

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

```
SELECT m.customer_id,  
       m.join_date, min(s.order_date) as first_purchase_members,  
       m2.product_name  
FROM members as m  
LEFT JOIN sales as s  
USING(customer_id)  
LEFT JOIN menu as m2  
USING(product_id)  
WHERE s.order_date >= m.join_date  
GROUP BY m.customer_id, m.join_date
```

customer_id	join_date	first_purchase_members	product_name
A	2021-01-07	2021-01-07	curry
B	2021-01-09	2021-01-11	sushi

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

```
SELECT m.customer_id,
       m.join_date,
       max(s.order_date) as last_purchase_before_member,
       m2.product_name
FROM members as m
LEFT JOIN sales as s
  USING(customer_id)
LEFT JOIN menu as m2
  USING(product_id)
WHERE s.order_date < m.join_date
GROUP BY s.customer_id
```

customer_id	join_date	last_purchase_before_member	product_name
A	2021-01-07	2021-01-01	sushi
B	2021-01-09	2021-01-04	curry

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

```
SELECT s.customer_id,
       count(s.product_id) as total_item,
       sum(m2.price) as amount_spent
FROM sales as s
LEFT JOIN members as m
  USING(customer_id)
LEFT JOIN menu as m2
  USING(product_id)
WHERE s.order_date < m.join_date
GROUP BY s.customer_id
```

customer_id	total_item	amount_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?

```
With points as (
    SELECT product_id ,product_name, price,
           CASE when product_name = 'sushi' then 20
           else 10
           END as point_per_dollar
    FROM menu
)
SELECT s.customer_id,
       sum(price*point_per_dollar) as total_points
FROM sales as s
LEFT JOIN points as p
USING(product_id)
GROUP BY s.customer_id
```

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?

```
With points as (
    SELECT product_id ,
           product_name,
           price,
           CASE when product_name = 'sushi' then 20
           else 10
           END as point_per_dollar
    FROM menu
), points_daily as (
    SELECT s.customer_id,
           sum(price*point_per_dollar) as
total_points
    FROM sales as s
    LEFT JOIN points as p
    USING(product_id)
    LEFT JOIN members as m
```



```

        USING(customer_id)
        WHERE (date(order_date) NOT BETWEEN
date(join_date) AND date_add(join_date, INTERVAL 7 DAY)) AND
month(order_date) = 1
        GROUP BY s.customer_id
    )
    , point_firstweek as (
        SELECT s.customer_id,
            sum(price*20) as total_points
        FROM sales as s
            LEFT JOIN menu as m
            USING(product_id)
            LEFT JOIN members as m2
            USING(customer_id)
        WHERE (date(order_date) BETWEEN
date(join_date) AND date_add(join_date, INTERVAL 7 DAY)) AND
month(order_date) = 1
        GROUP BY s.customer_id
    )
    , final_table as (
        SELECT customer_id,
            total_points
        FROM point_firstweek
        UNION ALL
        SELECT customer_id,
            total_points
        FROM points_daily
    )
    SELECT customer_id,
        sum(total_points) as total_points_Jan
    FROM final_table
    GROUP BY customer_id

```

customer_id	total_points_Jan
A	1,370
B	940

#The following questions are related creating basic data tables that Danny and his team can use to quickly derive insights without needing to join the underlying tables using SQL.

#Recreate the following table output using the available data:

```
CREATE TABLE monitor AS
(
  SELECT s.customer_id,
         s.order_date,
         m.product_name, m.price,
         CASE WHEN date(s.order_date) >= date(m2.join_date) THEN 'Y'
              ELSE 'N'
         END as member_regis
  FROM sales as s
  LEFT JOIN menu as m
  USING(product_id)
  LEFT JOIN members as m2
  USING(customer_id)
)
```

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

#Danny also requires further information about the **ranking** of customer products, but he purposely does not need the ranking for non-member purchases so he expects null **ranking** values for the records when customers are not yet part of the loyalty program.

```
SELECT m.customer_id ,
       m.order_date ,
       m.product_name ,m.price ,
       m.member_regis,
       CASE WHEN m.member_regis = 'Y' THEN DENSE_RANK() OVER
(PARTITION BY customer_id,m.member_regis ORDER BY order_date)
       ELSE 'null'
END as ranking
FROM monitor as m
```

customer_id	order_date	product_name	price	member_regis	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