

Case Study #1 - Danny's Diner

Danny Ma · May 1, 2021



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 favourite 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 simple questions about his customers, especially about their visiting patterns, how much money they've spent and also which menu items are their favourite. Having this deeper connection with his customers will help him deliver a better and more personalised 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.

Danny has provided you with a sample of his overall customer data due to privacy issues - but he hopes that these examples are enough for you to write fully functioning SQL queries to help him answer his questions!

Danny has shared with you 3 key datasets for this case study:

- sales
- menu
- members

You can inspect the entity relationship diagram and example data below.

Entity Relationship Diagram

Example Datasets

All datasets exist within the `dannys_diner` database schema - be sure to include this reference within your SQL scripts as you start exploring the data and answering the case study questions.

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.

Table 2: menu

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

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.

Interactive SQL Session

You can use the embedded DB Fiddle below to easily access these example datasets - this interactive session has everything you need to start solving these questions using SQL.

You can click on the [Edit on DB Fiddle](#) link on the top right hand corner of the embedded session below and it will take you to a fully functional SQL editor where you can write your own queries to analyse the data.

You can feel free to choose any SQL dialect you'd like to use, the existing Fiddle is using PostgreSQL 13 as default.

Serious SQL students have access to a dedicated SQL script in the 8 Week SQL Challenge section of the course which they can use to generate relevant temporary tables like we've done throughout the entire course!

Case Study Questions

Each of the following case study questions can be answered using a single SQL statement:

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?

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

Danny's Diner

30 November 2022 12:02

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

```
SELECT s.customer_id,  
       SUM(m.price) total_amount  
FROM sales s  
LEFT JOIN menu m  
ON s.product_id=m.product_idsq  
GROUP BY s.customer_id  
ORDER BY s.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) no_of_days_visited  
FROM sales  
GROUP BY customer_id;
```

customer_id	no_of_days_visited
A	4
B	6
C	2

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

```
SELECT customer_id,  
       STRING_AGG(product_name, ' , ') most_purchased_product  
FROM (SELECT DISTINCT s.customer_id,  
                      m.product_name,  
                      RANK() OVER (PARTITION BY s.customer_id ORDER BY order_date)  
rank  
FROM sales s  
LEFT JOIN menu m  
ON s.product_id=m.product_id  
 ) subquery  
WHERE rank=1  
GROUP BY customer_id;
```

customer_id	most_purchased_product
A	curry , sushi
B	curry
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 m.product_name,
           COUNT(1) number_of_times_purchased
FROM sales s
LEFT JOIN menu m
ON s.product_id=m.product_id
GROUP BY m.product_name
ORDER BY number_of_times_purchased DESC;
```

product_name	number_of_times_purchased
ramen	8

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

```
SELECT customer_id
       ,STRING_AGG(product_name,' , ') popular_dish
FROM (
SELECT  s.customer_id
        ,m.product_name
        ,COUNT(1) no_of_orders
        ,RANK() OVER (PARTITION BY s.customer_id ORDER BY COUNT(1) DESC) rnk
FROM sales s
INNER JOIN menu m
ON s.product_id=m.product_id
GROUP BY s.customer_id
        ,m.product_name
        ) subquery
WHERE rnk=1
GROUP BY customer_id;
```

customer_id	popular_dish
A	ramen
B	sushi , curry , ramen
C	ramen

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

```
WITH rank_cte AS (
SELECT  s.customer_id,
        m.product_name,
        s.order_date,
        DENSE_RANK() OVER (PARTITION BY s.customer_id ORDER BY order_date
ASC) rnk
FROM sales s
INNER JOIN menu m
ON s.product_id=m.product_id
INNER JOIN members mb
ON s.customer_id=mb.customer_id
WHERE s.order_date>=join_date
)
SELECT customer_id,
       product_name
FROM rank_cte
WHERE rnk =1
```

customer_id	product_name
A	curry
B	sushi

```
--7.Which item was purchased just before the customer became a member?
WITH rank_cte AS (
SELECT  s.customer_id,
        m.product_name,
        s.order_date,
        DENSE_RANK() OVER (PARTITION BY s.customer_id ORDER BY order_date
ASC) rnk
FROM sales s
INNER JOIN menu m
ON s.product_id=m.product_id
INNER JOIN members mb
ON s.customer_id=mb.customer_id
WHERE s.order_date<join_date
)
SELECT customer_id,
        STRING_AGG(product_name, ' , ') items_purchased_before_joining
FROM rank_cte
WHERE rnk =1
GROUP BY customer_id;
```

customer_id	items_purchased_before_joining
A	sushi , curry
B	curry

```
--8.What is the total items and amount spent for each member before they
became a member?
SELECT  s.customer_id,
        COUNT(1) total_items,
        SUM(m.price) amount_spent

FROM sales s
INNER JOIN menu m
ON s.product_id=m.product_id
INNER JOIN members mb
ON s.customer_id=mb.customer_id
WHERE s.order_date<join_date
GROUP BY s.customer_id;
```

customer_id	total_items	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?

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

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?

```
SELECT s.customer_id,
       SUM(CASE WHEN s.order_date BETWEEN mb.join_date AND
DATEADD(DAY,DATEDIFF(DAY,0,mb.join_date),6) THEN m.price*20
              WHEN m.product_name='Sushi' THEN m.price*20
              ELSE m.price*10
            END) points
FROM sales s
INNER JOIN members mb
ON s.customer_id=mb.customer_id
INNER JOIN menu m
ON s.product_id=m.product_id
WHERE MONTH(order_date)=1
GROUP BY s.customer_id;
```

customer_id	points
A	1370
B	820