SQL - CASE STUDY:

1 Danny’s Diner

Created by - Danny Ma

Amol Jalda
@amoljalda

Swipe →

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 needs your assistance to help the restaurant stay afloat - the restaurant has captured some fundamental data from its few months of operation. Still, it has no idea how to use their data to help them run the business.

Amol Jalda
@amoljalda

Swipe →

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:

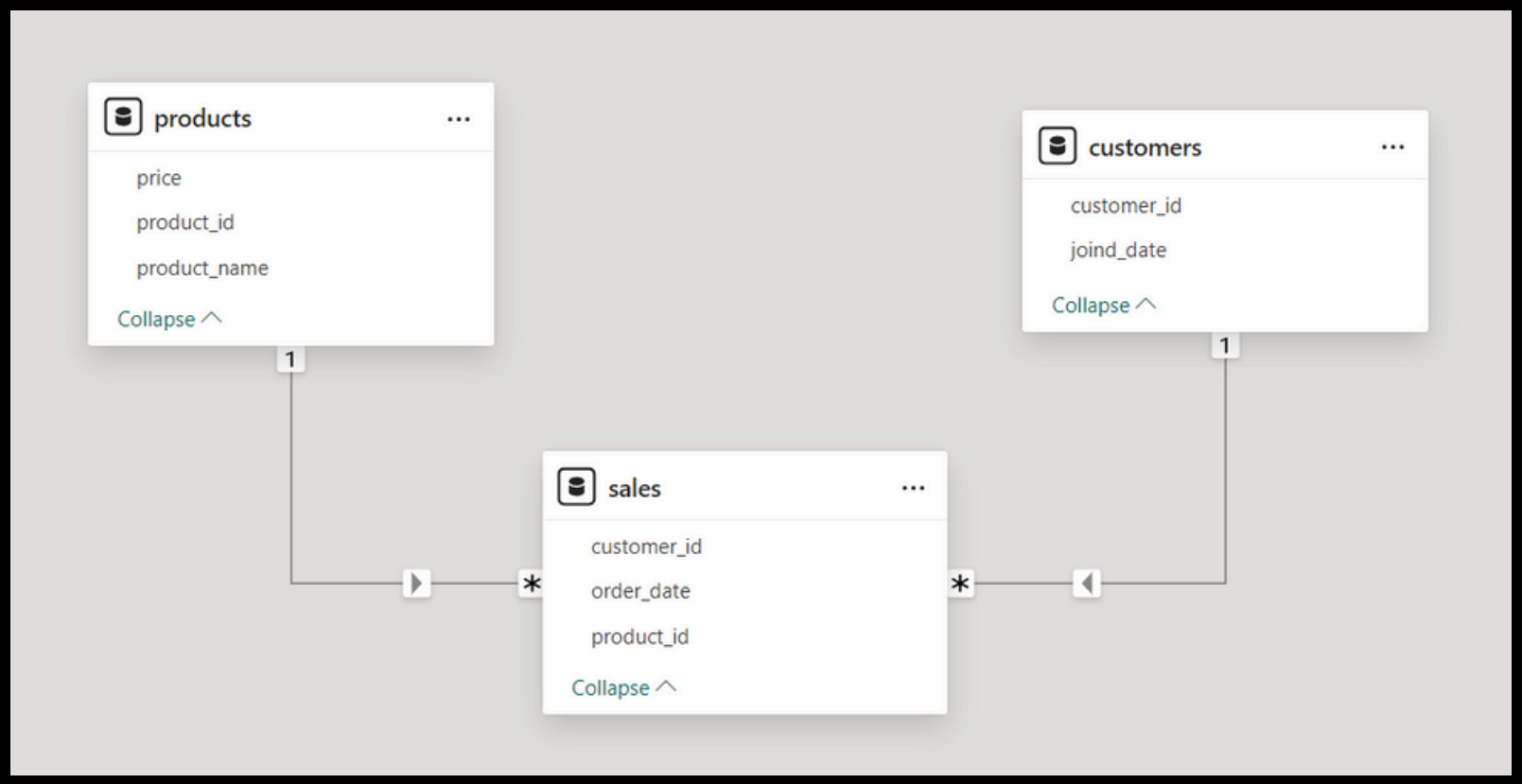
Tables:

- Sales: Dimension
- Customers: Fact
- Products: Fact

Amol Jalda
@amoljalda

Swipe →

Data Schema



Amol Jalda
@amoljalda

Swipe →

Data

```
SELECT * FROM dannys_diner.sales
```

100 %

Results | Messages

	customer_id	order_date	product_id
1	A	2021-01-01	1
2	A	2021-01-01	2
3	A	2021-01-07	2
4	A	2021-01-10	3
5	A	2021-01-11	3
6	A	2021-01-11	3
7	B	2021-01-01	2
8	B	2021-01-02	2
9	B	2021-01-04	1
10	B	2021-01-11	1
11	B	2021-01-16	3
12	B	2021-02-01	3
13	C	2021-01-01	3
14	C	2021-01-01	3
15	C	2021-01-07	3

```
SELECT * FROM dannys_diner.customers
```

100 %

Results | Messages

	customer_id	joind_date
1	A	2021-01-07
2	B	2021-01-09
3	C	2021-01-11

```
SELECT * FROM dannys_diner.products
```

100 %

Results | Messages

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

Amol Jalda
@amoljalda

Swipe →

Query: 1

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

Query & Output:

```
SELECT
    s.customer_id Customers,
    SUM(p.price) Total_Spent
FROM
    dannys_diner.sales s
INNER JOIN
    dannys_diner.products p
ON s.product_id = p.product_id
GROUP BY s.customer_id
```

100 %

Results | Messages

	Customers	Total_Spent
1	A	76
2	B	74
3	C	36

Validatation

```
SELECT * FROM dannys_diner.sales
SELECT * FROM dannys_diner.products
```

100 %

Results | Messages

	customer_id	order_date	product_id
1	A	2021-01-01	1
2	A	2021-01-01	2
3	A	2021-01-07	2
4	A	2021-01-10	3
5	A	2021-01-11	3
6	A	2021-01-11	3
7	B	2021-01-01	2
8	B	2021-01-02	2
9	B	2021-01-04	1
10	B	2021-01-11	1
11	B	2021-01-16	3
12	B	2021-02-01	3
13	C	2021-01-01	3
14	C	2021-01-01	3
15	C	2021-01-07	3

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

Amol Jalda
@amoljalda

Swipe →

Query: 2

How many days has each customer visited the restaurant?

Query & Output:

```
SELECT
    customer_id Customers,
    COUNT(order_date) Visits
FROM dannys_diner.sales
GROUP BY customer_id
```

100 %

Results | Messages | Client Statistics

	Customers	Visits
1	A	6
2	B	6
3	C	3

Validatation

```
SELECT * FROM dannys_diner.sales
SELECT * FROM dannys_diner.products
```

100 %

Results | Messages

	customer_id	order_date	product_id
1	A	2021-01-01	1
2	A	2021-01-01	2
3	A	2021-01-07	2
4	A	2021-01-10	3
5	A	2021-01-11	3
6	A	2021-01-11	3
7	B	2021-01-01	2
8	B	2021-01-02	2
9	B	2021-01-04	1
10	B	2021-01-11	1
11	B	2021-01-16	3
12	B	2021-02-01	3
13	C	2021-01-01	3
14	C	2021-01-01	3
15	C	2021-01-07	3

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

Amol Jalda
@amoljalda

Swipe →

Query: 3

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

Query & Output:

```
WITH RNK_CTE
AS(
SELECT
    s.customer_id,
    p.product_name,
    s.order_date,
    DENSE_RANK() OVER (ORDER BY s.order_date ASC) RNK
FROM dannys_diner.sales s
INNER JOIN dannys_diner.products p
ON s.product_id = p.product_id
)

SELECT
    customer_id customer,
    product_name product
FROM RNK_CTE WHERE RNK = 1
```

100 %

Results | Messages | Client Statistics

	customer	product
1	A	sushi
2	A	curry
3	B	curry
4	C	ramen
5	C	ramen

Validatation

```
SELECT *
FROM dannys_diner.sales
ORDER BY  customer_id ASC, order_date ASC

SELECT * FROM dannys_diner.products
```

100 %

Results | Messages | Client Statistics

	customer_id	order_date	product_id
1	A	2021-01-01	1
2	A	2021-01-01	2
3	A	2021-01-07	2
4	A	2021-01-10	3
5	A	2021-01-11	3
6	A	2021-01-11	3
7	B	2021-01-01	2
8	B	2021-01-02	2
9	B	2021-01-04	1
10	B	2021-01-11	1
11	B	2021-01-16	3
12	B	2021-02-01	3
13	C	2021-01-01	3
14	C	2021-01-01	3
15	C	2021-01-07	3

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

Amol Jalda
@amoljalda

Swipe →

Query: 4

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

Query & Output:

```sql
SELECT TOP 1
    p.product_name products,
    COUNT(s.product_id) purchase_count
FROM
    dannys_diner.sales s
INNER JOIN dannys_diner.products p
ON s.product_id = p.product_id
GROUP BY
    p.product_name
ORDER BY
    purchase_count DESC
```

100 %

Results | Messages | Client Statistics

	products	purchase_count
1	ramen	8

Validatation

```sql
SELECT
    product_id,
    count(customer_id) purchase_count
FROM
    dannys_diner.sales
GROUP BY product_id

SELECT * FROM dannys_diner.products
```

100 %

Results | Messages | Client Statistics

	product_id	purchase_count
1	1	3
2	2	4
3	3	8

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

Amol Jalda
@amoljalda

Swipe →

Query: 5

Which item was the most popular for each customer?

Query & Output:

```
WITH popular_item_row
AS (
SELECT
s.customer_id,
p.product_name products,
COUNT(s.product_id) purchase_count,
DENSE_RANK ()
    OVER(PARTITION BY s.customer_id ORDER BY COUNT(s.product_id) DESC) RNK
FROM dannys_diner.sales s
INNER JOIN dannys_diner.products p
ON s.product_id = p.product_id
GROUP BY s.customer_id, p.product_name
)
SELECT customer_id,
       products,
       purchase_count
FROM popular_item_row
WHERE RNK = 1
```

00 % | Results | Messages | Client Statistics

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

Validatation

```
SELECT *
FROM dannys_diner.sales
ORDER BY customer_id
```

100 % | Results | Messages | Client Statistics

	customer_id	order_date	product_id
1	A	2021-01-01	1
2	A	2021-01-01	2
3	A	2021-01-07	2
4	A	2021-01-10	3
5	A	2021-01-11	3
6	A	2021-01-11	3
7	B	2021-01-01	2
8	B	2021-01-02	2
9	B	2021-01-04	1
10	B	2021-01-11	1
11	B	2021-01-16	3
12	B	2021-02-01	3
13	C	2021-01-01	3
14	C	2021-01-01	3
15	C	2021-01-07	3

DENSE_RANK do not skip the next RANK NUMBER.
Reason to use DENSE_RANK, RANK instead. Because customers have ordered multiple products an equal number of times.

Amol Jalda
@amoljalda

Swipe →

Query: 6

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

Query & Output:

```sql
WITH my_cte
AS (
SELECT
    s.customer_id,
    c.joind_date,
    s.order_date,
    p.product_name,
    RANK()
        OVER(PARTITION BY c.joind_date ORDER BY s.order_date) RNK
FROM dannys_diner.sales s
LEFT JOIN dannys_diner.customers c
ON s.customer_id = c.customer_id
LEFT JOIN dannys_diner.products p
ON s.product_id = p.product_id
WHERE c.joind_date <= s.order_date
)
SELECT
    customer_id,
    joind_date,
    order_date,
    product_name
FROM my_cte
WHERE RNK = 1
```

100 %

Results | Messages | Client Statistics

	customer_id	joind_date	order_date	product_name
1	A	2021-01-07	2021-01-07	curry
2	B	2021-01-09	2021-01-11	sushi

Amol Jalda
@amoljalda

Swipe →

Query: 7

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

Query & Output:

```sql
WITH my_cte AS (
SELECT
    s.customer_id,
    c.joind_date,
    s.order_date,
    p.product_name,
    RANK()
        OVER(PARTITION BY c.joind_date ORDER BY s.order_date) RNK
FROM dannys_diner.sales s
LEFT JOIN dannys_diner.customers c
ON s.customer_id = c.customer_id
LEFT JOIN dannys_diner.products p
ON s.product_id = p.product_id
WHERE c.joind_date >= s.order_date
)
SELECT
    customer_id,
    joind_date,
    order_date,
    product_name
FROM my_cte
WHERE RNK = 1
```

100 %

Results | Messages | Client Statistics

	customer_id	joind_date	order_date	product_name
1	A	2021-01-07	2021-01-01	sushi
2	A	2021-01-07	2021-01-01	curry
3	B	2021-01-09	2021-01-01	curry
4	C	2021-01-11	2021-01-01	ramen
5	C	2021-01-11	2021-01-01	ramen

Amol Jalda
@amoljalda

Swipe →

Query: 8

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

Query & Output:

```sql
WITH my_cte
AS (
SELECT
    c.customer_id,
    c.joind_date,
    s.order_date,
    s.product_id,
    COUNT(s.product_id) product_count,
    p.price
FROM dannys_diner.sales s
INNER JOIN dannys_diner.products p
ON s.product_id = p.product_id
INNER JOIN dannys_diner.customers c
ON s.customer_id = c.customer_id
WHERE c.joind_date > s.order_date
GROUP BY
    c.customer_id,
    c.joind_date,
    s.order_date,
    p.price,
    s.product_id
)
SELECT
    customer_id,
    joind_date,
    MAX(order_date) order_date,
    SUM(product_count) total_products,
    SUM(price) price,
    SUM(product_count * price) total_amount
FROM my_cte
GROUP BY
    customer_id,
    joind_date
ORDER BY customer_id
```

85 %

Results | Messages | Client Statistics

	customer_id	joind_date	order_date	total_products	price	total_amount
1	A	2021-01-07	2021-01-01	2	25	25
2	B	2021-01-09	2021-01-04	3	40	40
3	C	2021-01-11	2021-01-07	3	24	36

Amol Jalda
@amoljalda

Swipe →

Query: 9

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

Query & Output:

```sql
SELECT
    s.customer_id,
    SUM(CASE
            WHEN p.product_name = 'sushi'
                THEN p.price * 20
            ELSE p.price * 10
        END ) AS total_points
FROM dannys_diner.sales s
INNER JOIN dannys_diner.products p
ON s.product_id = p.product_id
GROUP BY
    s.customer_id
```

85 %

Results | Messages | Client Statistics

	customer_id	total_points
1	A	860
2	B	940
3	C	360

Validatation

```sql
SELECT
    s.customer_id,
    s.product_id,
    p.product_name,
    p.price,
    CASE
        WHEN p.product_name = 'sushi' THEN 20 ELSE 10
    END AS points,
    CASE
        WHEN p.product_name = 'sushi'
            THEN p.price * 20
        ELSE p.price * 10
    END AS total_points
FROM dannys_diner.sales s
INNER JOIN dannys_diner.products p
ON s.product_id = p.product_id
```

85 %

Results | Messages | Client Statistics

	customer_id	product_id	product_name	price	points	total_points
1	A	1	sushi	10	20	200
2	A	2	curry	15	10	150
3	A	2	curry	15	10	150
4	A	3	ramen	12	10	120
5	A	3	ramen	12	10	120
6	A	3	ramen	12	10	120
7	B	2	curry	15	10	150
8	B	2	curry	15	10	150
9	B	1	sushi	10	20	200
10	B	1	sushi	10	20	200
11	B	3	ramen	12	10	120
12	B	3	ramen	12	10	120
13	C	3	ramen	12	10	120
14	C	3	ramen	12	10	120
15	C	3	ramen	12	10	120

Amol Jalda
@amoljalda

Swipe →

Query: 10

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 customers A and B have at the end of January?

Query & Output:

```
WITH my_cte
AS (
SELECT
    s.customer_id,
    s.product_id,
    (s.product_id * 2) points,
    s.order_date,
    c.joind_date,
    DATEADD(DAY, 7, c.joind_date) first_week
FROM dannys_diner.sales s
INNER JOIN dannys_diner.customers c
ON s.customer_id = c.customer_id
)
SELECT
    customer_id,
    SUM(points) total_points
FROM my_cte
WHERE
    (order_date BETWEEN joind_date AND first_week)
AND
    customer_id IN ('A', 'B')
GROUP BY
    customer_id
```

85 %

Results | Messages | Client Statistics

	customer_id	total_points
1	A	22
2	B	8

Validatation

```
SELECT
    s.customer_id,
    s.product_id,
    (s.product_id * 2) points,
    c.joind_date, 7 Days , DATEADD(DAY, 7, c.joind_date) first_week,
    s.order_date
FROM dannys_diner.sales s
INNER JOIN dannys_diner.customers c
ON s.customer_id = c.customer_id
WHERE
    s.customer_id IN ('A', 'B') AND s.order_date BETWEEN c.joind_date
        AND DATEADD(DAY, 7, c.joind_date)
```

Results | Messages | Client Statistics

	customer_id	product_id	points	joind_date	Days	first_week	order_date
1	A	2	4	2021-01-07	7	2021-01-14	2021-01-07
2	A	3	6	2021-01-07	7	2021-01-14	2021-01-10
3	A	3	6	2021-01-07	7	2021-01-14	2021-01-11
4	A	3	6	2021-01-07	7	2021-01-14	2021-01-11
5	B	1	2	2021-01-09	7	2021-01-16	2021-01-11
6	B	3	6	2021-01-09	7	2021-01-16	2021-01-16

Amol Jalda
@amoljalda

Swipe →