SQL CASE STUDY 1

BY MANISHA SASMAL

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. 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!

Entity Relationship Diagram

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

- **Query**

```sql
SELECT
    sales.customer_id,
    SUM(menu.price) as total_money_spend
FROM dannys_diner.sales
JOIN dannys_diner.menu
ON sales.product_id = menu.product_id
GROUP BY customer_id
ORDER BY sales.customer_id ASC;
```

- **Output**

customer_id	total_money_spend
A	76
B	74
C	36

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

- **Query**

```
SELECT
   customer_id,
   COUNT(DISTINCT order_date) as total_days_visited
FROM dannys_diner.sales
GROUP BY customer_id
ORDER BY customer_id ASC;
```

- **Output**

customer_id	total_days_visited
A	4
B	6
C	2

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

- **Query**

```
WITH firstItem as (
  SELECT
      sales.customer_id,
      sales.order_date,
      menu.product_name,
  RANK() OVER( PARTITION BY customer_id ORDER BY order_date ASC ) AS rnk,
  ROW_NUMBER() OVER( PARTITION BY customer_id ORDER BY order_date ASC ) AS rn
  FROM dannys_diner.sales
  JOIN dannys_diner.menu ON sales.product_id = menu.product_id
  )
SELECT
  customer_id,
  product_name AS product
FROM firstItem
WHERE rn = 1;
```

- **Output**

customer_id	product
A	curry
B	curry
C	ramen

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

- **Query**

```sql
SELECT
    s.product_id,
    m.product_name,
    COUNT(s.product_id) AS timePurchased
FROM dannys_diner.sales AS s
JOIN dannys_diner.menu AS m
ON s.product_id = m.product_id
GROUP BY s.product_id, m.product_name
ORDER BY timePurchased DESC LIMIT 1;
```

- **Output**

product_id	product_name	timepurchased
3	ramen	8

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

- **Query**

```sql
WITH items AS(
 SELECT
    s.customer_id,
    m.product_name,
    COUNT(s.product_id) AS popularitem,
    RANK() OVER( PARTITION BY customer_id
                ORDER BY (COUNT(s.product_id)) DESC ) AS rnk
 FROM dannys_diner.sales AS s
 JOIN dannys_diner.menu AS m ON s.product_id = m.product_id
 GROUP BY s.product_id, m.product_name, s.customer_id
  )
SELECT
  customer_id,
  STRING_AGG(product_name,',') AS product
FROM items
WHERE rnk = 1
GROUP BY items.customer_id;
```

- **Output**

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

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

- **Query**

```
WITH firstItem AS(
  SELECT
     s.customer_id,
     s.order_date,
     m.product_name,
     RANK() OVER( PARTITION BY s.customer_id
                  ORDER BY s.order_date ) AS rnk
  FROM dannys_diner.sales s
  JOIN dannys_diner.menu m
     ON s.product_id = m.product_id
  JOIN dannys_diner.members mem
     ON s.customer_id = mem.customer_id
  WHERE s.order_date >= mem.join_date
  )
SELECT
   customer_id,
   product_name
FROM firstItem
WHERE rnk = 1;
```

- **Output**

customer_id	product_name
A	curry
B	sushi

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

- **Query**

```
WITH Item AS(
  SELECT
     s.customer_id,
     s.order_date,
     m.product_name,
     RANK() OVER( PARTITION BY s.customer_id
                  ORDER BY s.order_date DESC ) AS rnk
  FROM dannys_diner.sales s
  JOIN dannys_diner.menu m
     ON s.product_id = m.product_id
  JOIN dannys_diner.members mem
     ON s.customer_id = mem.customer_id
  WHERE s.order_date < mem.join_date
  )
SELECT
   customer_id,
   STRING_AGG(product_name, ',') AS product
FROM Item
WHERE rnk = 1
GROUP BY Item.customer_id;
```

- **Output**

customer_id	product
A	sushi,curry
B	sushi

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

- **Query**

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

- **Output**

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?

- **Query**

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

- **Output**

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?

- **Query**

```
SELECT
  s.customer_id,
  SUM(CASE
      WHEN s.order_date BETWEEN mem.join_date AND (mem.join_date + INTERVAL '6 day') THEN m.price * 20
      ELSE m.price * 10
      END) AS points
FROM dannys_diner.sales s
JOIN dannys_diner.menu m ON s.product_id = m.product_id
JOIN dannys_diner.members mem ON s.customer_id = mem.customer_id
WHERE s.order_date <= '2021-01-31'
AND s.order_date >= mem.join_date
GROUP BY s.customer_id
ORDER BY s.customer_id ASC;
```

- **Output**

customer_id	points
A	1020
B	320

Bonus 1. Join All The Things And Recreate The Table

- **Query**

```sql
SELECT
   s.customer_id,
   s.order_date,
   m.product_name,
   m.price,
  (CASE
    WHEN s.order_date < mem.join_date OR mem.join_date IS NULL THEN 'N'
    ELSE 'Y'
   END) AS member
FROM dannys_diner.sales s
  JOIN dannys_diner.menu m
     ON s.product_id = m.product_id
  LEFT JOIN dannys_diner.members mem
     ON s.customer_id = mem.customer_id
ORDER BY s.customer_id ASC, s.order_date ASC;
```

- **Output**

customer_id	order_date	product_name	price	member
A	2021-01-01T00:00:00.000Z	sushi	10	N
A	2021-01-01T00:00:00.000Z	curry	15	N
A	2021-01-07T00:00:00.000Z	curry	15	Y
A	2021-01-10T00:00:00.000Z	ramen	12	Y
A	2021-01-11T00:00:00.000Z	ramen	12	Y
A	2021-01-11T00:00:00.000Z	ramen	12	Y
B	2021-01-01T00:00:00.000Z	curry	15	N
B	2021-01-02T00:00:00.000Z	curry	15	N
B	2021-01-04T00:00:00.000Z	sushi	10	N
B	2021-01-11T00:00:00.000Z	sushi	10	Y
B	2021-01-16T00:00:00.000Z	ramen	12	Y
B	2021-02-01T00:00:00.000Z	ramen	12	Y
C	2021-01-01T00:00:00.000Z	ramen	12	N
C	2021-01-01T00:00:00.000Z	ramen	12	N
C	2021-01-07T00:00:00.000Z	ramen	12	N

Bonus 2. Rank All The Things And Recreate The Table

- **Query**

```
SELECT *,
  (CASE WHEN member ='Y'
     THEN
     RANK() OVER( PARTITION BY customer_id, member ORDER BY order_date ASC, price DESC)
     ELSE null
   END) AS ranking
FROM
(SELECT
     s.customer_id,
     s.order_date,
     m.product_name,
     m.price,
    (CASE
       WHEN s.order_date < mem.join_date OR mem.join_date IS NULL THEN 'N'
       ELSE 'Y'
     END) AS member
FROM dannys_diner.sales s
  JOIN dannys_diner.menu m
     ON s.product_id = m.product_id
  LEFT JOIN dannys_diner.members mem
     ON s.customer_id = mem.customer_id
ORDER BY s.customer_id ASC, s.order_date ASC) customer;
```

- **Output**

customer_id	order_date	product_name	price	member	ranking
A	2021-01-01T00:00:00.000Z	sushi	10	N	null
A	2021-01-01T00:00:00.000Z	curry	15	N	null
A	2021-01-07T00:00:00.000Z	curry	15	Y	1
A	2021-01-10T00:00:00.000Z	ramen	12	Y	2
A	2021-01-11T00:00:00.000Z	ramen	12	Y	3
A	2021-01-11T00:00:00.000Z	ramen	12	Y	3
B	2021-01-01T00:00:00.000Z	curry	15	N	null
B	2021-01-02T00:00:00.000Z	curry	15	N	null
B	2021-01-04T00:00:00.000Z	sushi	10	N	null
B	2021-01-11T00:00:00.000Z	sushi	10	Y	1
B	2021-01-16T00:00:00.000Z	ramen	12	Y	2
B	2021-02-01T00:00:00.000Z	ramen	12	Y	3
C	2021-01-01T00:00:00.000Z	ramen	12	N	null
C	2021-01-01T00:00:00.000Z	ramen	12	N	null
C	2021-01-07T00:00:00.000Z	ramen	12	N	null