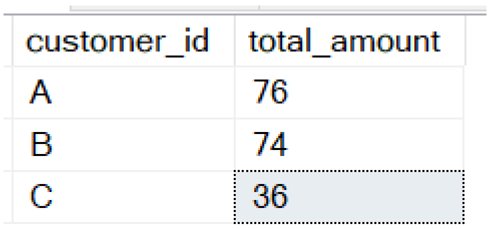
Case Study #1: Danny's Diner

Solution

View the complete syntax here

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

```
SELECT s.customer_id,
       SUM(m.price) as total_amount
FROM dannys_diner.sales s
JOIN dannys_diner.menu m
ON s.product_id = m.product_id
GROUP BY s.customer_id
ORDER BY total_amount DESC;
```



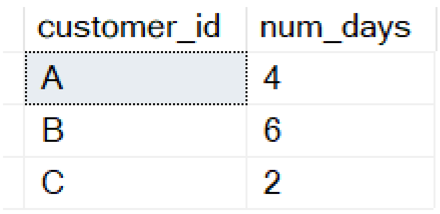
customer_id	total_amount
A	76
B	74
C	36

- Customer A spent $76.
- Customer B spent $74.
- Customer C spent $36.

---

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

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



customer_id	num_days
A	4
B	6
C	2

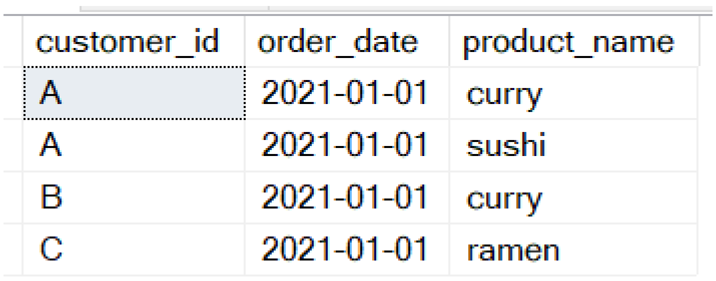
- Customer A visited 4 times.
- Customer B visited 6 times.
- Customer C visited 2 times.

---

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

```sql
WITH first_item
AS
(
    SELECT customer_id,
           product_id,
           order_date,
           RANK() OVER(PARTITION BY customer_id ORDER BY order_date) as rk
    FROM dannys_diner.sales
)

SELECT DISTINCT fi.customer_id,
       fi.order_date,
       m.product_name
FROM first_item fi
JOIN dannys_diner.menu m
ON fi.product_id = m.product_id
WHERE fi.rk = 1
ORDER BY fi.customer_id;
```



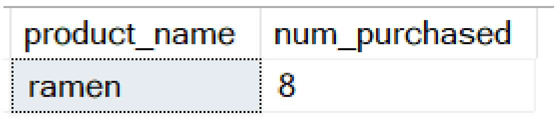
customer_id	order_date	product_name
A	2021-01-01	curry
A	2021-01-01	sushi
B	2021-01-01	curry
C	2021-01-01	ramen

- Customer A first orders are curry and sushi.
- Customer B first order is curry.
- Customer C first order is ramen.

---

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

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



product_name	num_purchased
ramen	8

- Most purchased item on the menu is ramen which is 8 times.

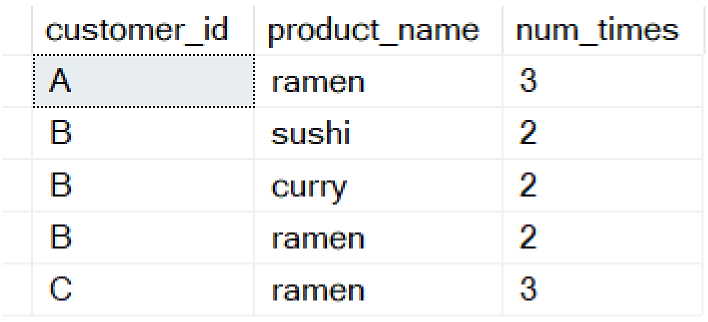
---

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

```
WITH customer_sales
AS(
    SELECT  s.customer_id,
            m.product_name,
            COUNT(*) as num_times
    FROM dannys_diner.sales s
    JOIN dannys_diner.menu m
    ON s.product_id = m.product_id
    GROUP BY s.customer_id,m.product_name

),
most_popular_item
AS
(
    SELECT customer_id,
           product_name,
           num_times,
           RANK() OVER(PARTITION BY customer_id ORDER BY num_times DESC) rk
    FROM customer_sales
)

SELECT customer_id,
       product_name,
       num_times
FROM most_popular_item
WHERE rk = 1;
```



customer_id	product_name	num_times
A	ramen	3
B	sushi	2
B	curry	2
B	ramen	2
C	ramen	3

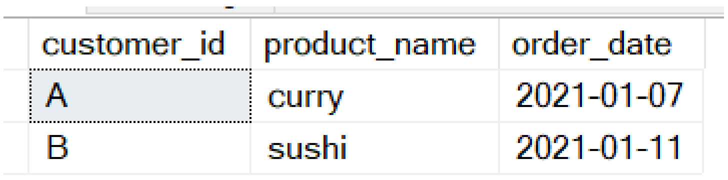
- Customer A and C favourite item is ramen.
- Customer B enjoys all items on the menu.

---

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

```
WITH customer_member
AS(
    SELECT s.customer_id,
           mn.product_name,
           s.order_date,
           RANK() OVER(PARTITION BY s.customer_id ORDER BY order_date) as rk
    FROM dannys_diner.sales s
    JOIN dannys_diner.members m
    ON s.customer_id = m.customer_id
    JOIN dannys_diner.menu mn
    ON mn.product_id = s.product_id
    WHERE s.order_date >= m.join_date
)

SELECT customer_id,
       product_name,
       order_date
FROM customer_member
WHERE rk = 1;
```



customer_id	product_name	order_date
A	curry	2021-01-07
B	sushi	2021-01-11

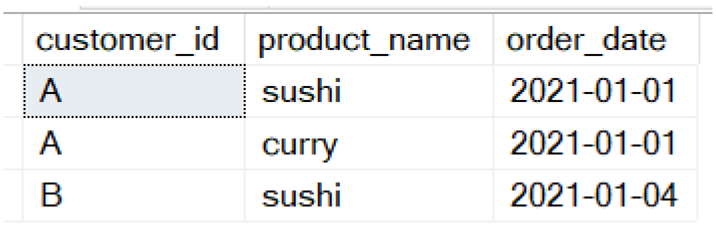
- Customer A first order as member is curry.
- Customer B first order as member is sushi.

---

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

```
WITH customer_member
AS(
    SELECT s.customer_id,
           mn.product_name,
           s.order_date,
           RANK() OVER(PARTITION BY s.customer_id ORDER BY s.order_date DESC) as
rk
    FROM dannys_diner.sales s
    JOIN dannys_diner.members m
    ON s.customer_id = m.customer_id
    JOIN dannys_diner.menu mn
    ON mn.product_id = s.product_id
    WHERE s.order_date < m.join_date
)
```

```
SELECT customer_id,
       product_name,
       order_date
FROM customer_member
WHERE rk = 1;
```



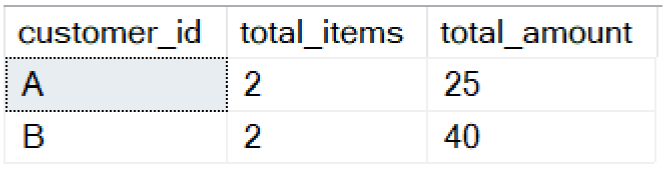
customer_id	product_name	order_date
A	sushi	2021-01-01
A	curry	2021-01-01
B	sushi	2021-01-04

- Customer A's first order as member is curry.
- Customer B's first order as member is sushi.

---

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

```
SELECT s.customer_id,
       COUNT(DISTINCT s.product_id) as total_items,
       SUM(mn.price)       as total_amount
FROM dannys_diner.sales s
JOIN dannys_diner.members m
ON s.customer_id = m.customer_id
JOIN dannys_diner.menu mn
ON mn.product_id = s.product_id
WHERE s.order_date < m.join_date
GROUP BY s.customer_id;
```



customer_id	total_items	total_amount
A	2	25
B	2	40

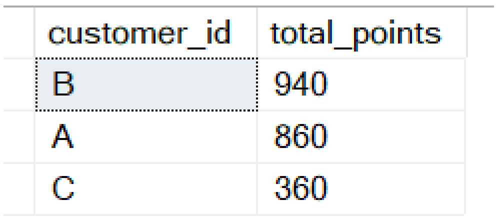
- Before becoming members,
    - Customer A spent $25 on 2 items.
    - Customer B spent $40 on 2 items.

---

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 2*10*m.price
                ELSE 10*m.price END
```

```
          )as total_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 total_points DESC;
```



customer_id	total_points
B	940
A	860
C	360

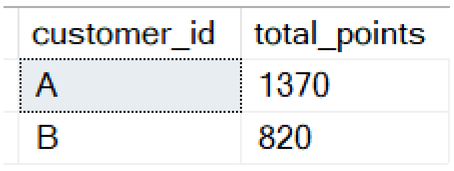
- Total points for Customer A is 860.
- Total points for Customer B is 940.
- Total points for Customer C is 360.

---

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 m.product_name = 'sushi' THEN 2*10*m.price
                WHEN s.order_date >= mb.join_date AND
                     s.order_date < DATEADD(WEEK , 1 , mb.join_date)
                THEN 2*10*m.price
                ELSE 10*m.price END
          )as total_points

FROM dannys_diner.sales s
JOIN dannys_diner.menu m
ON s.product_id = m.product_id
JOIN dannys_diner.members mb
ON mb.customer_id = s.customer_id
WHERE s.order_date <= '2021-01-31'
GROUP BY s.customer_id;
```



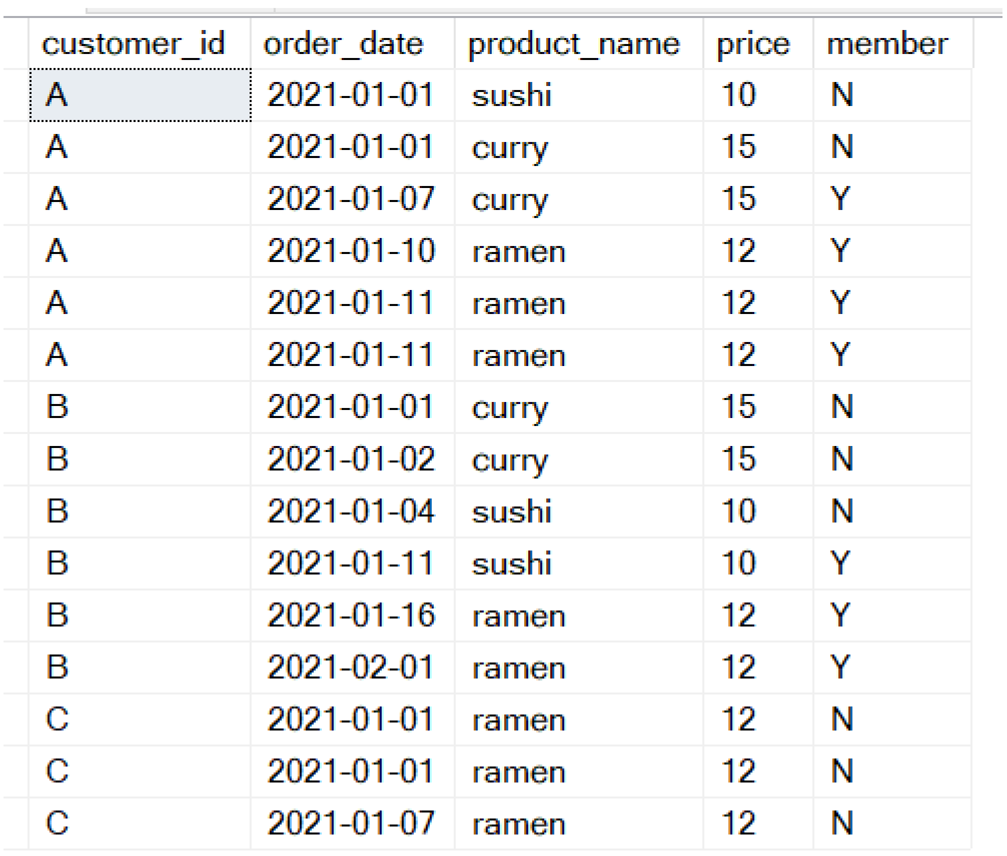
customer_id	total_points
A	1370
B	820

- Total points for Customer A is 1,370.
- Total points for Customer B is 820.

BONUS QUESTIONS

Join All The Things - Recreate the table with: customer_id, order_date, product_name, price, member (Y/N)

```
SELECT s.customer_id,
       s.order_date,
       m.product_name,
       m.price,
       CASE WHEN mb.join_date IS NULL THEN 'N'
            WHEN mb.join_date IS NOT NULL AND mb.join_date > s.order_date
            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 mb
ON mb.customer_id = s.customer_id;
```

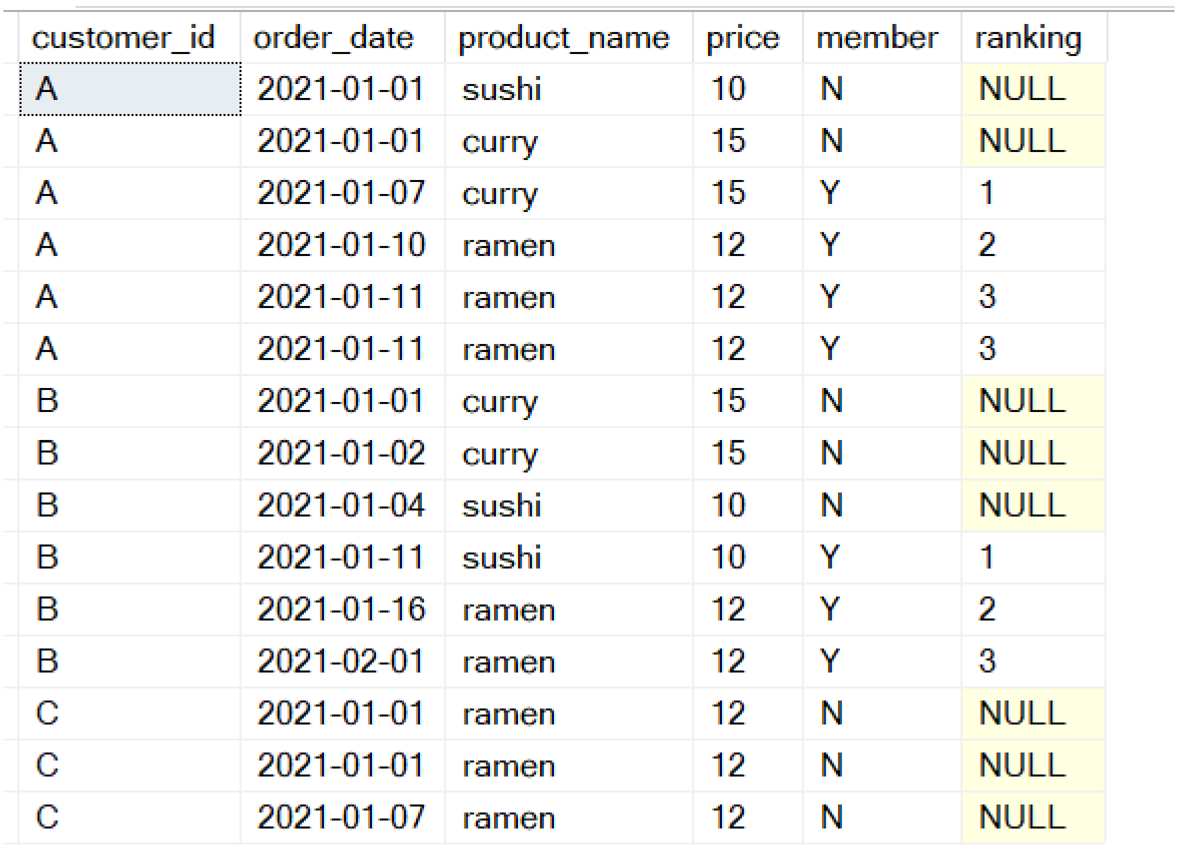


customer_id	order_date	product_name	price	member
A	2021-01-01	sushi	10	N
A	2021-01-01	curry	15	N
A	2021-01-07	curry	15	Y
A	2021-01-10	ramen	12	Y
A	2021-01-11	ramen	12	Y
A	2021-01-11	ramen	12	Y
B	2021-01-01	curry	15	N
B	2021-01-02	curry	15	N
B	2021-01-04	sushi	10	N
B	2021-01-11	sushi	10	Y
B	2021-01-16	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

Rank All The Things 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.

```
WITH join_all_data
AS(
    SELECT s.customer_id,
           s.order_date,
           m.product_name,
           m.price,
           CASE WHEN mb.join_date IS NULL THEN 'N'
                WHEN mb.join_date IS NOT NULL AND mb.join_date > s.order_date
                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 mb
    ON mb.customer_id = s.customer_id
)

SELECT * ,CASE
        WHEN member = 'Y' THEN RANK() OVER(PARTITION BY customer_id,member ORDER
BY order_date)
        ELSE NULL END as ranking
FROM join_all_data;
```



customer_id	order_date	product_name	price	member	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
B	2021-02-01	ramen	12	Y	3
C	2021-01-01	ramen	12	N	NULL
C	2021-01-01	ramen	12	N	NULL
C	2021-01-07	ramen	12	N	NULL
