8WEEKSQLCHALLENGE.COM
CASE STUDY #1
DANNY'S DINER
THE TASTE OF SUCCESS
DATAWITHDANNY.COM

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

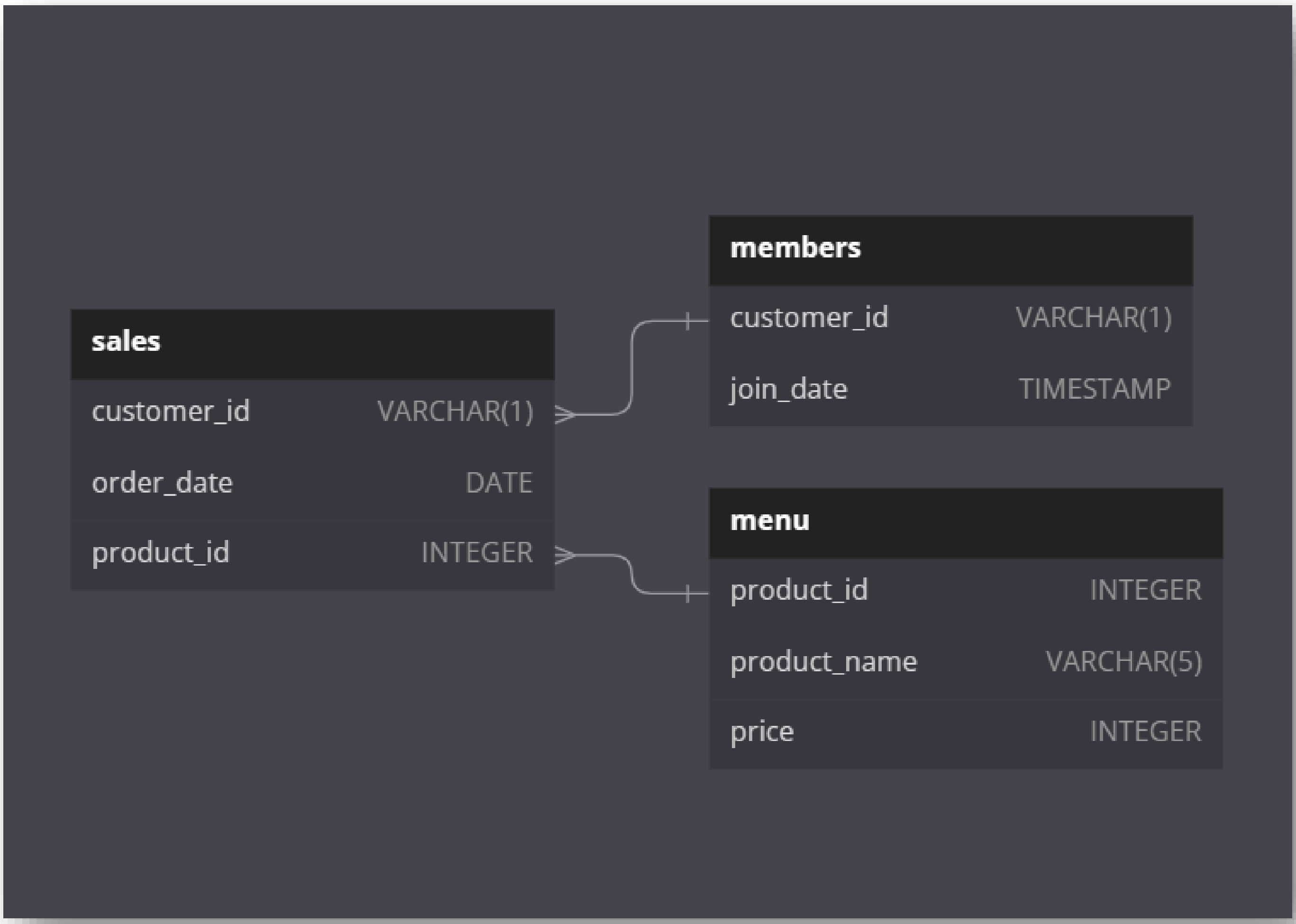


Table 1: sales

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.

customer_id	order_date	product_id
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

Table 2: menu

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

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

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.

customer_id	join_date
A	2021-01-07
B	2021-01-09

Case Study Questions

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

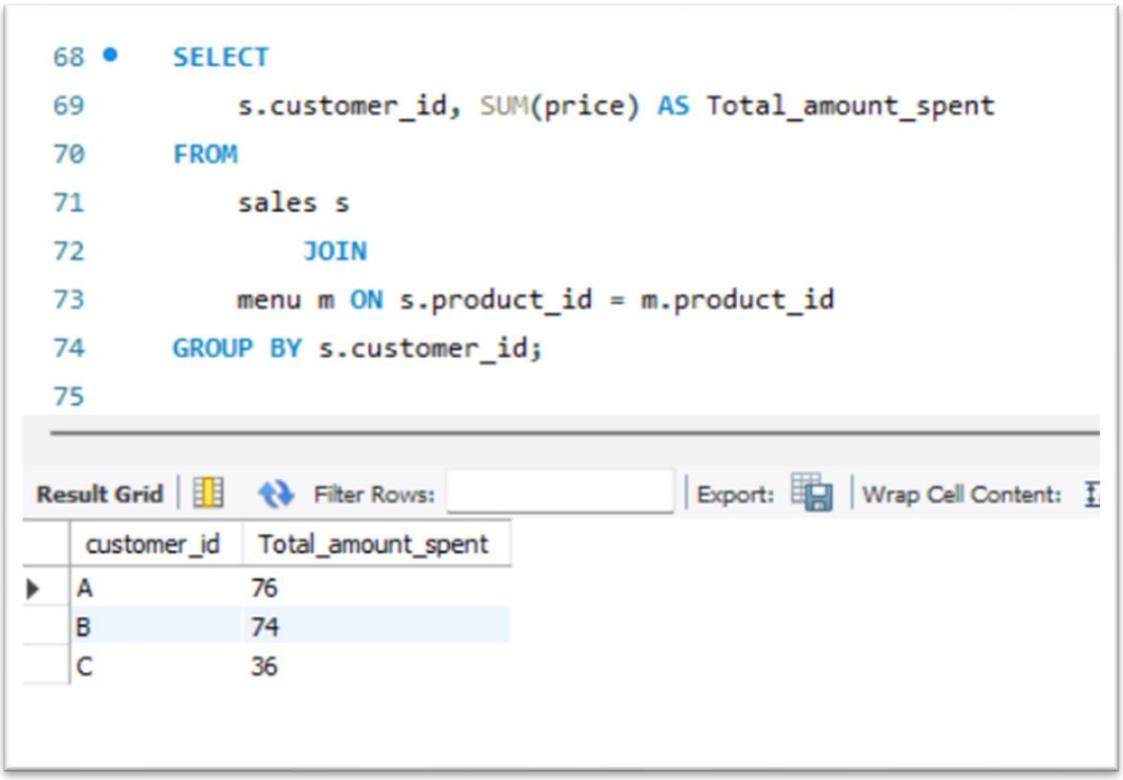
1. What is the total amount each customer spent at the restaurant?
2. How many days has each customer visited the restaurant?
3. What was the first item from the menu purchased by each customer?
4. What is the most purchased item on the menu and how many times was it purchased by all customers?
5. Which 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?

Solution

I used **MySQL Workbench** and these are the particular functions I employed:

- Aggregate functions (SUM, COUNT)
- Table Joins
- Ranking functions (RANK, DENSE_RANK)
- Date Functions (DATE_ADD, LAST_DAY)
- Common Table Expressions (CTE)
- Window function

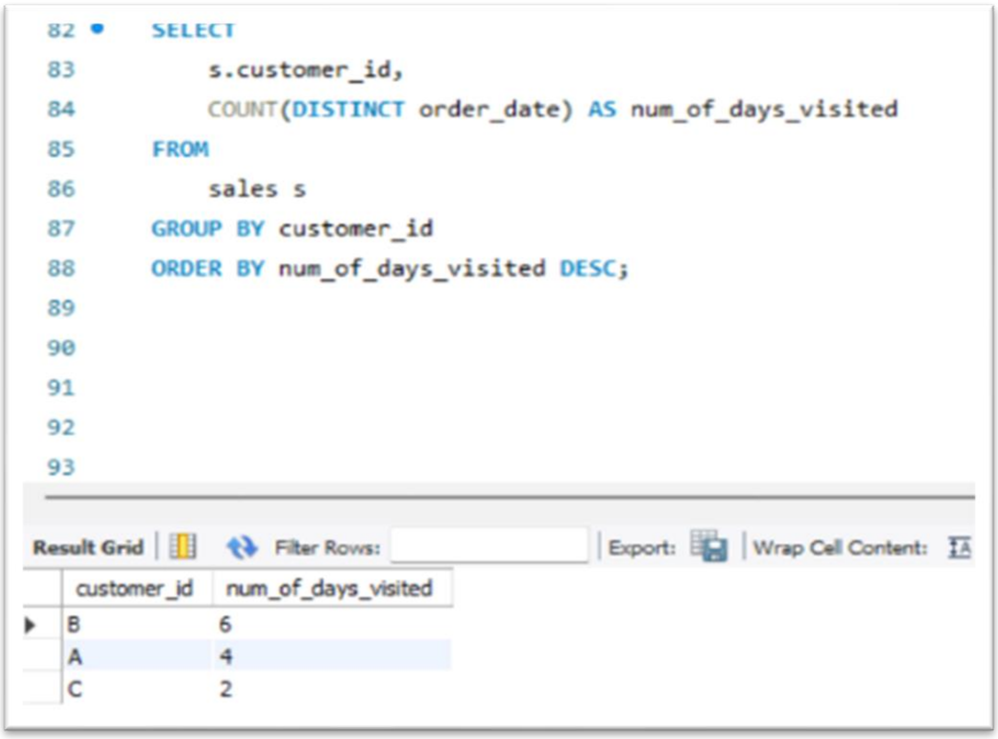
Q1. What is the total amount each customer spent at the restaurant?



```
SELECT
    s.customer_id, SUM(price) AS Total_amount_spent
FROM
    sales s
        JOIN
    menu m ON s.product_id = m.product_id
GROUP BY s.customer_id;
```

customer_id	Total_amount_spent
A	76
B	74
C	36

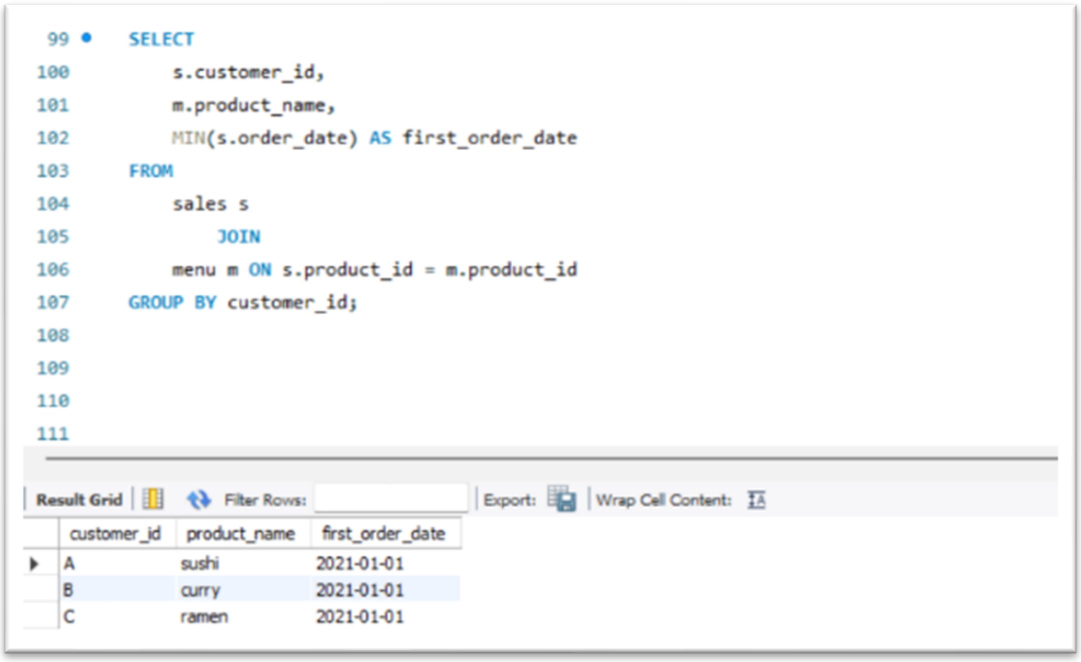
Q2. How many days has each customer visited the restaurant?



```
SELECT
    s.customer_id,
    COUNT(DISTINCT order_date) AS num_of_days_visited
FROM
    sales s
GROUP BY customer_id
ORDER BY num_of_days_visited DESC;
```

customer_id	num_of_days_visited
B	6
A	4
C	2

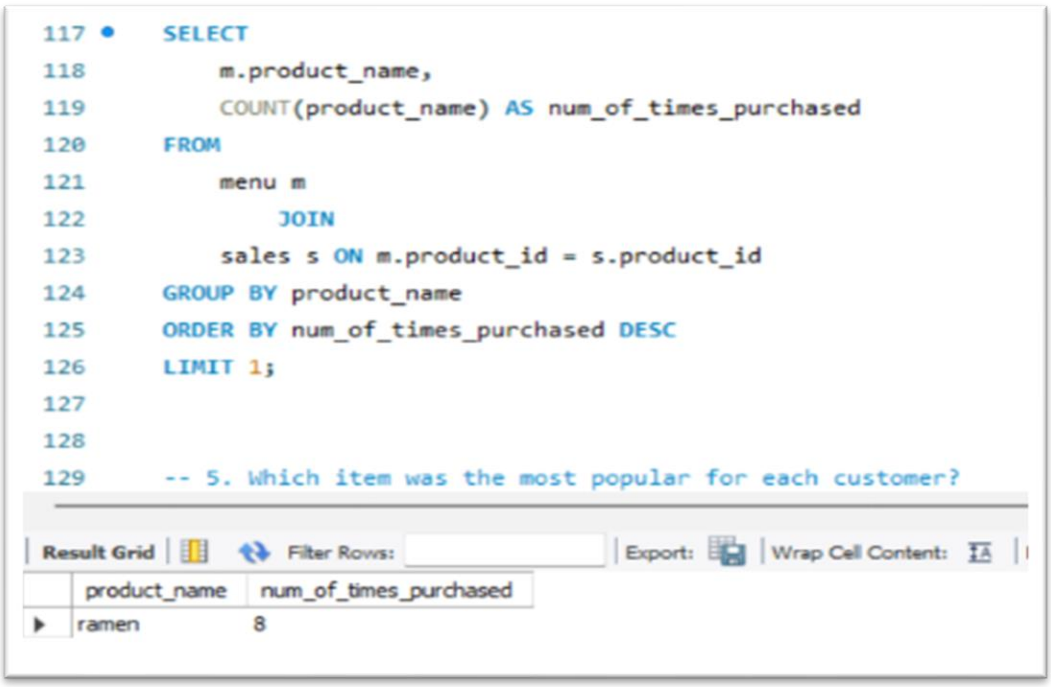
Q3. What was the first item from the menu purchased by each customer?



```
SELECT
    s.customer_id,
    m.product_name,
    MIN(s.order_date) AS first_order_date
FROM
    sales s
        JOIN
    menu m ON s.product_id = m.product_id
GROUP BY customer_id;
```

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

Q4. What is the most purchased item on the menu and how many times was it purchased by all customers?



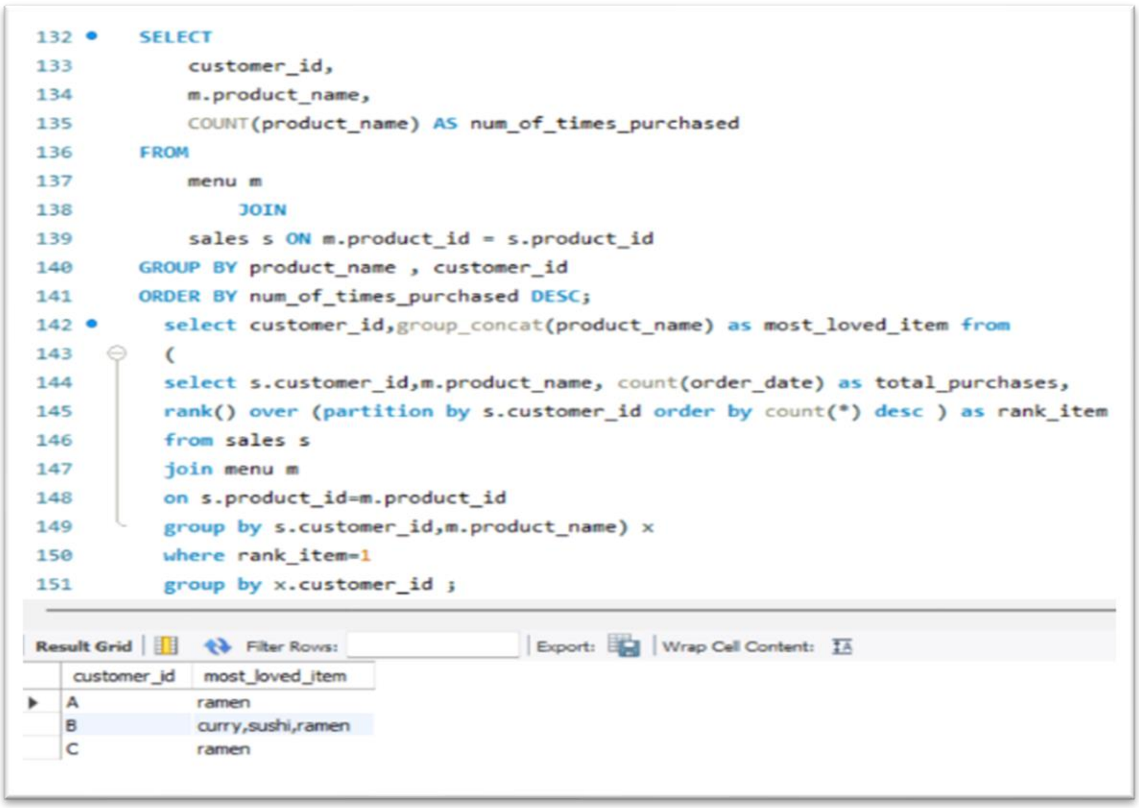
```
SELECT
    m.product_name,
    COUNT(product_name) AS num_of_times_purchased
FROM
    menu m
        JOIN
    sales s ON m.product_id = s.product_id
GROUP BY product_name
ORDER BY num_of_times_purchased DESC
LIMIT 1;


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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
product_name   num_of_times_purchased
ramen          8
```

Q5. Which item was the most popular for each customer?

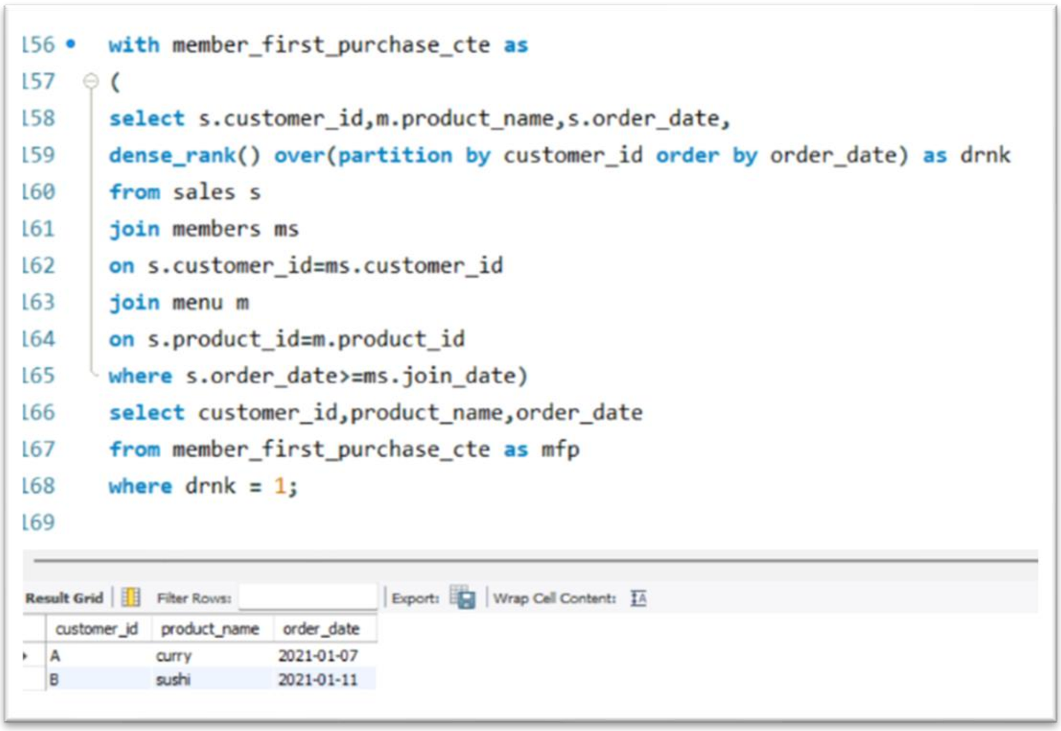


```
SELECT
    customer_id,
    m.product_name,
    COUNT(product_name) AS num_of_times_purchased
FROM
    menu m
        JOIN
    sales s ON m.product_id = s.product_id
GROUP BY product_name , customer_id
ORDER BY num_of_times_purchased DESC;
  select customer_id,group_concat(product_name) as most_loved_item from
  (
  select s.customer_id,m.product_name, count(order_date) as total_purchases,
  rank() over (partition by s.customer_id order by count(*) desc ) as rank_item
  from sales s
  join menu m
  on s.product_id=m.product_id
  group by s.customer_id,m.product_name) x
  where rank_item=1
  group by x.customer_id ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
customer_id   most_loved_item
A             ramen
B             curry,sushi,ramen
C             ramen
```

Q6. Which item was purchased first by the customer after they became a member?

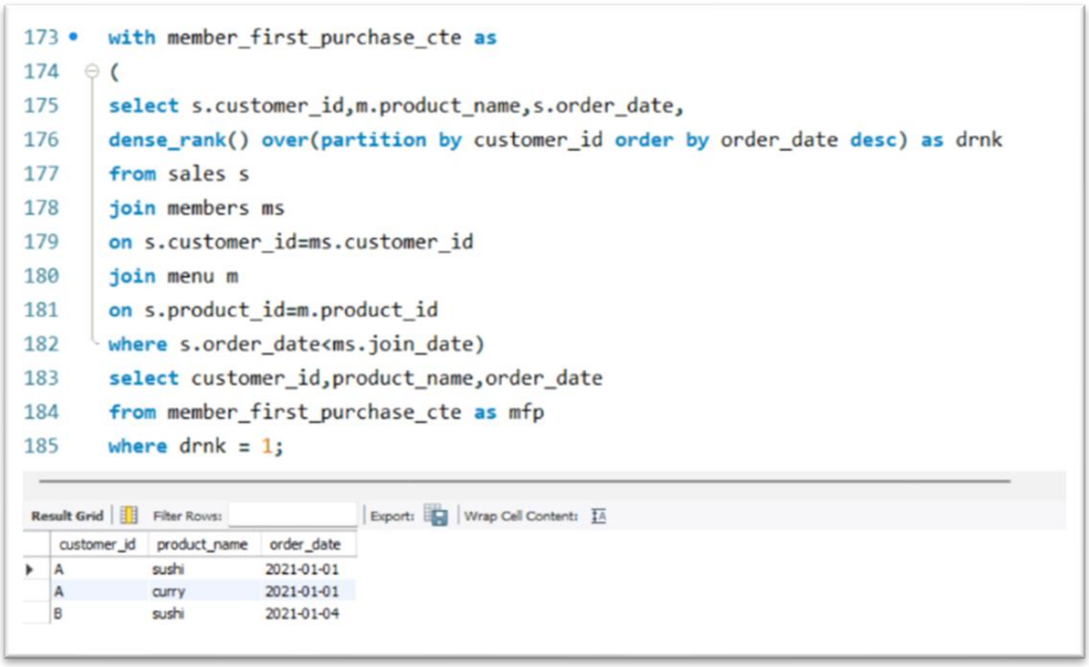


```
with member_first_purchase_cte as
(
select s.customer_id,m.product_name,s.order_date,
dense_rank() over(partition by customer_id order by order_date) as drnk
from sales s
join members ms
on s.customer_id=ms.customer_id
join menu m
on s.product_id=m.product_id
where s.order_date>=ms.join_date)
select customer_id,product_name,order_date
from member_first_purchase_cte as mfp
where drnk = 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

Q7. Which item was purchased just before the customer became a member?

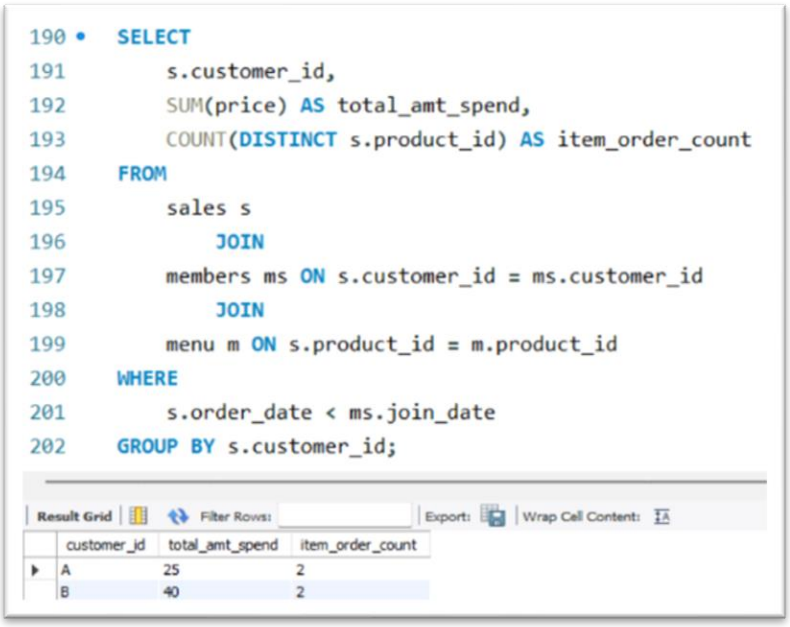


```
with member_first_purchase_cte as
(
select s.customer_id,m.product_name,s.order_date,
dense_rank() over(partition by customer_id order by order_date desc) as drnk
from sales s
join members ms
on s.customer_id=ms.customer_id
join menu m
on s.product_id=m.product_id
where s.order_date<ms.join_date)
select customer_id,product_name,order_date
from member_first_purchase_cte as mfp
where drnk = 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

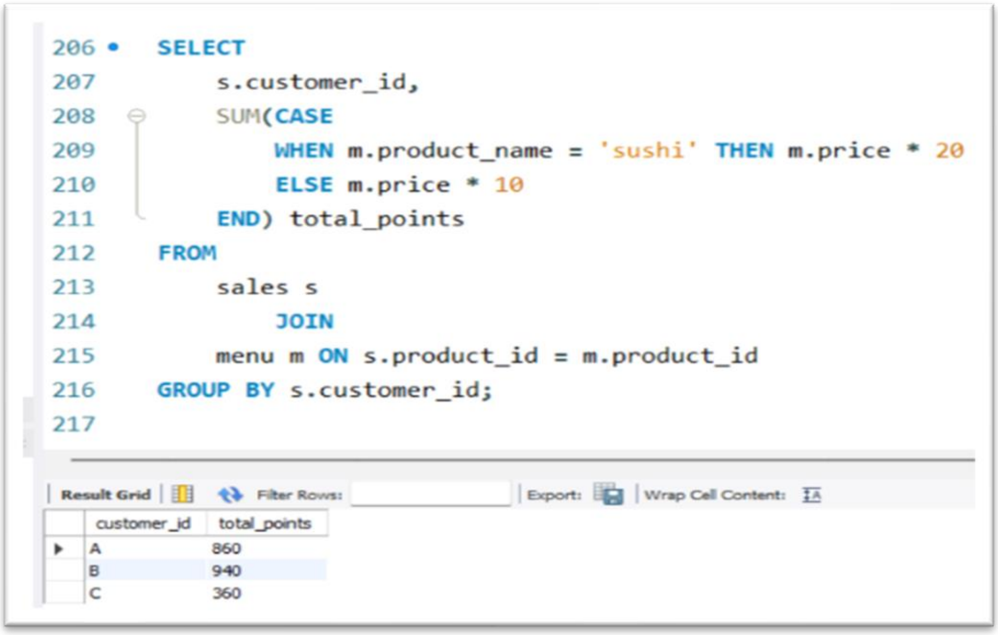
Q8. What are the total items and amount spent for each member before they became a member?



```
SELECT
    s.customer_id,
    SUM(price) AS total_amt_spend,
    COUNT(DISTINCT s.product_id) AS item_order_count
FROM
    sales s
        JOIN
    members ms ON s.customer_id = ms.customer_id
        JOIN
    menu m ON s.product_id = m.product_id
WHERE
    s.order_date < ms.join_date
GROUP BY s.customer_id;
```

customer_id	total_amt_spend	item_order_count
A	25	2
B	40	2

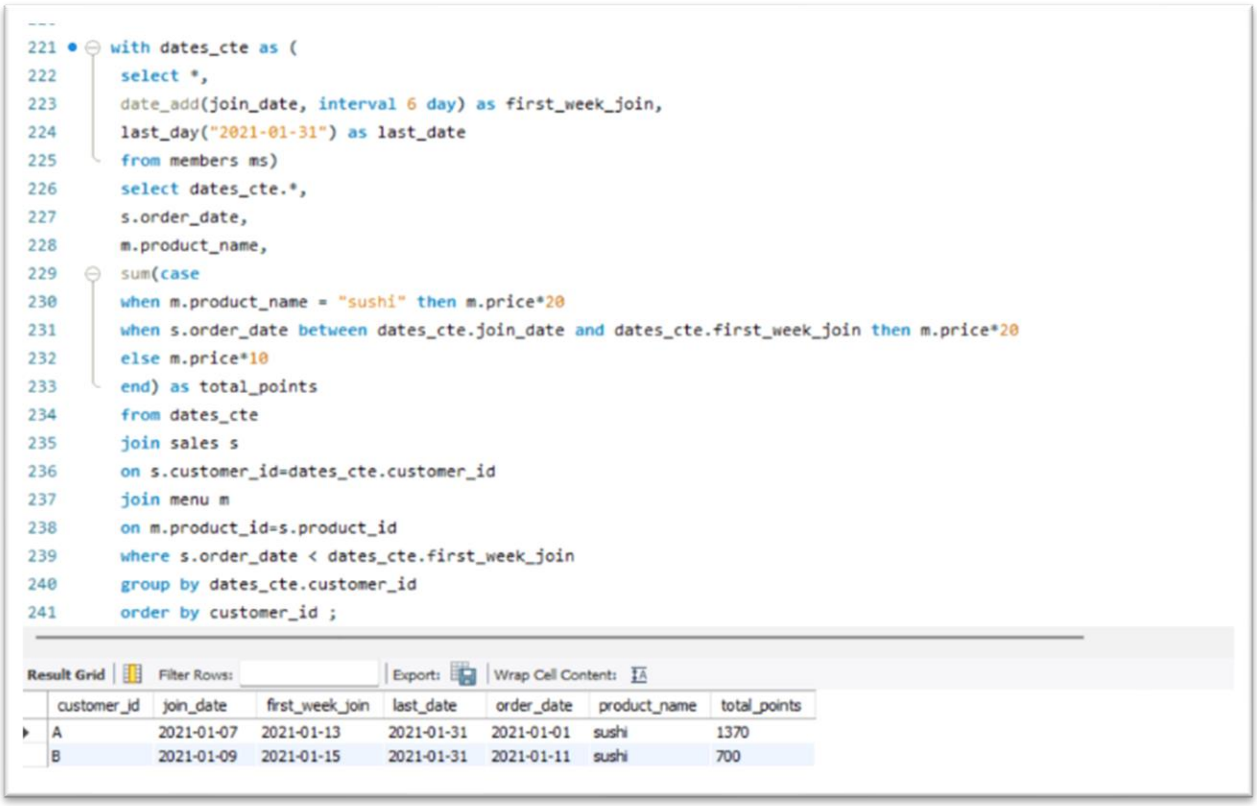
Q9. If each $1 spent equates to 10 points and sushi has a 2x points multiplier — how many points would each customer have?



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

customer_id	total_points
A	860
B	940
C	360

Q10. If 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?



```sql
with dates_cte as (
  select *,
  date_add(join_date, interval 6 day) as first_week_join,
  last_day("2021-01-31") as last_date
  from members ms)
  select dates_cte.*,
  s.order_date,
  m.product_name,
  sum(case
  when m.product_name = "sushi" then m.price*20
  when s.order_date between dates_cte.join_date and dates_cte.first_week_join then m.price*20
  else m.price*10
  end) as total_points
  from dates_cte
  join sales s
  on s.customer_id=dates_cte.customer_id
  join menu m
  on m.product_id=s.product_id
  where s.order_date < dates_cte.first_week_join
  group by dates_cte.customer_id
  order by customer_id ;
```

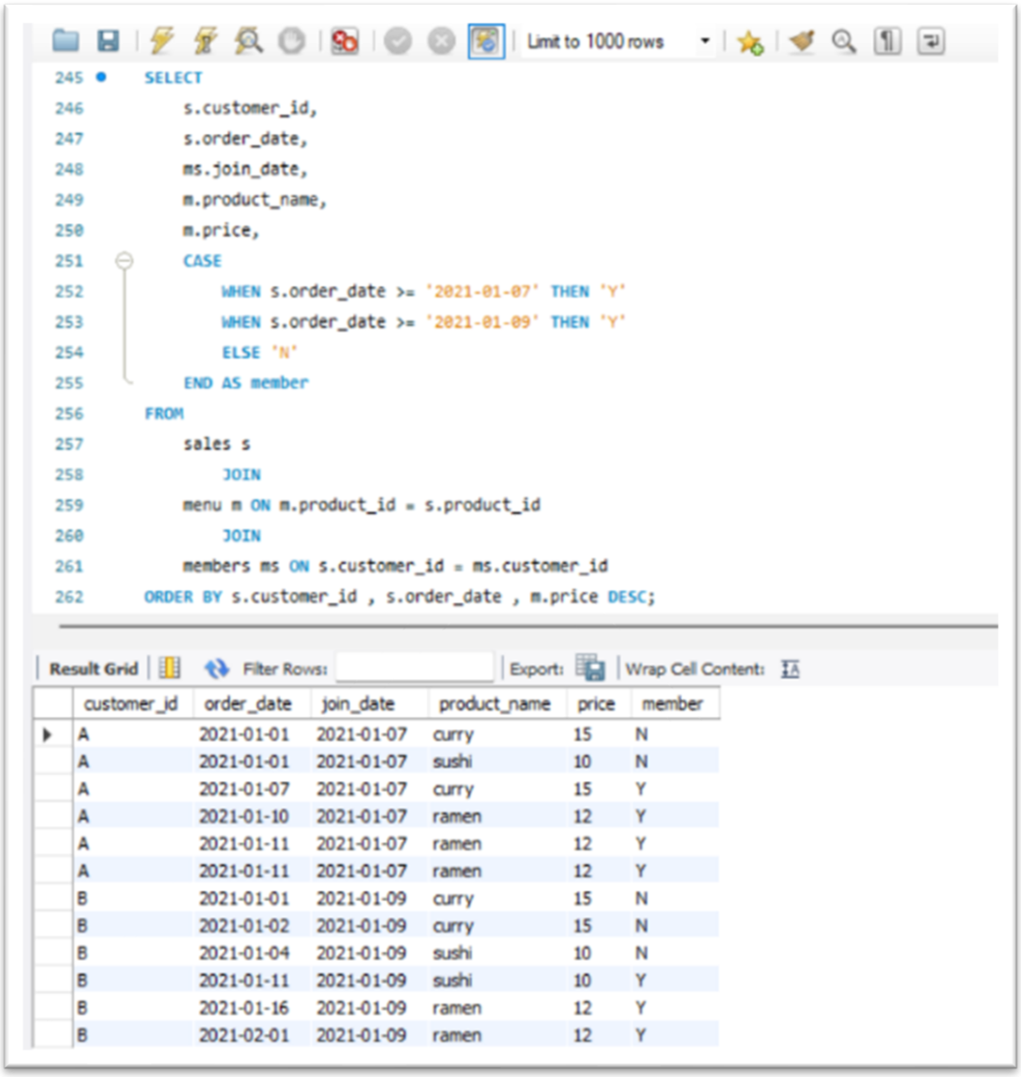
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	join_date	first_week_join	last_date	order_date	product_name	total_points
A	2021-01-07	2021-01-13	2021-01-31	2021-01-01	sushi	1370
B	2021-01-09	2021-01-15	2021-01-31	2021-01-11	sushi	700

Bonus Questions

Join All The Things

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.

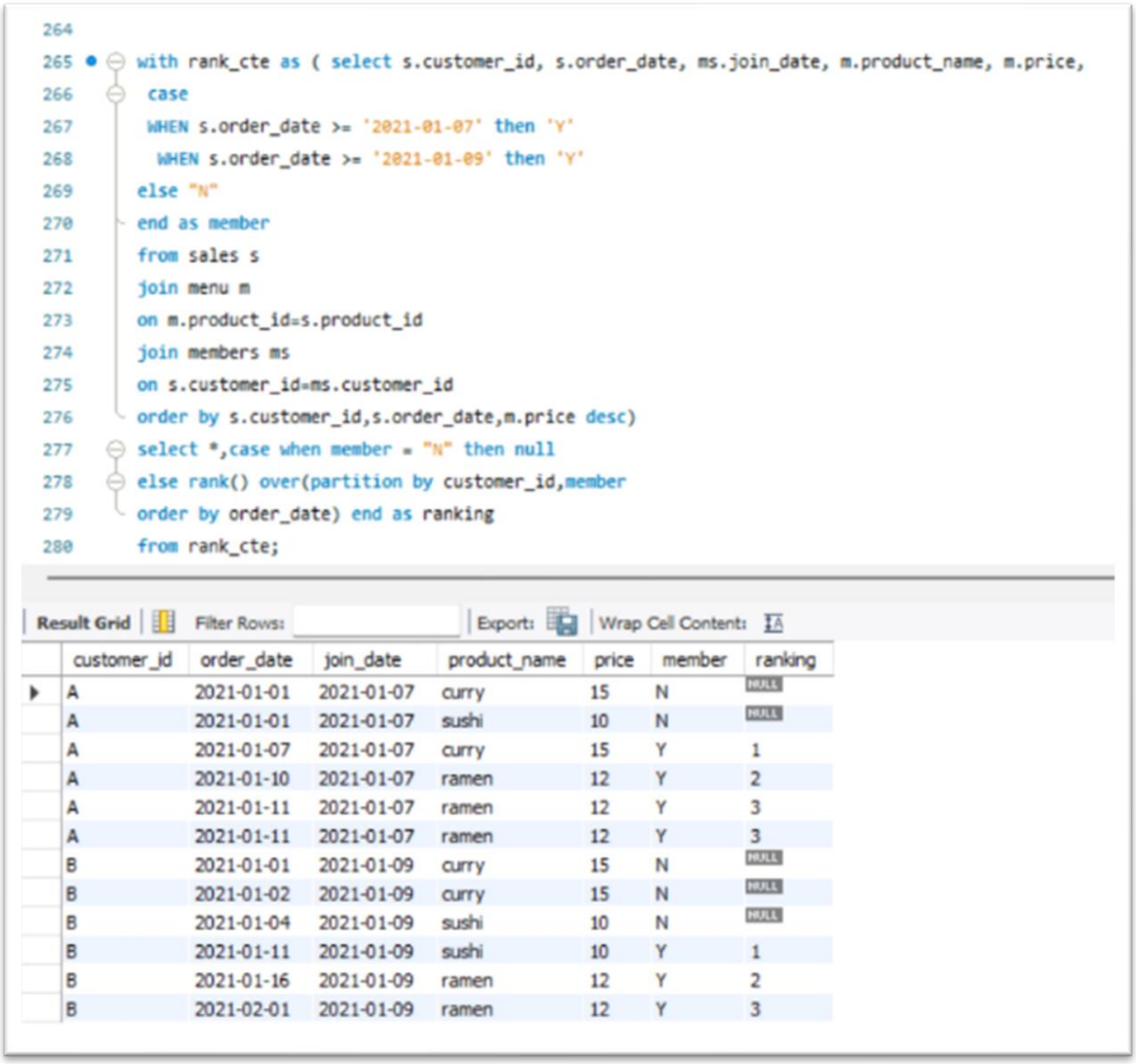


```sql
SELECT
    s.customer_id,
    s.order_date,
    ms.join_date,
    m.product_name,
    m.price,
    CASE
        WHEN s.order_date >= '2021-01-07' THEN 'Y'
        WHEN s.order_date >= '2021-01-09' THEN 'Y'
        ELSE 'N'
    END AS member
FROM
    sales s
        JOIN
    menu m ON m.product_id = s.product_id
        JOIN
    members ms ON s.customer_id = ms.customer_id
ORDER BY s.customer_id , s.order_date , m.price DESC;
```

customer_id	order_date	join_date	product_name	price	member
A	2021-01-01	2021-01-07	curry	15	N
A	2021-01-01	2021-01-07	sushi	10	N
A	2021-01-07	2021-01-07	curry	15	Y
A	2021-01-10	2021-01-07	ramen	12	Y
A	2021-01-11	2021-01-07	ramen	12	Y
A	2021-01-11	2021-01-07	ramen	12	Y
B	2021-01-01	2021-01-09	curry	15	N
B	2021-01-02	2021-01-09	curry	15	N
B	2021-01-04	2021-01-09	sushi	10	N
B	2021-01-11	2021-01-09	sushi	10	Y
B	2021-01-16	2021-01-09	ramen	12	Y
B	2021-02-01	2021-01-09	ramen	12	Y

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.



```sql
with rank_cte as ( select s.customer_id, s.order_date, ms.join_date, m.product_name, m.price,
 case
 WHEN s.order_date >= '2021-01-07' then 'Y'
  WHEN s.order_date >= '2021-01-09' then 'Y'
else "N"
end as member
from sales s
join menu m
on m.product_id=s.product_id
join members ms
on s.customer_id=ms.customer_id
order by s.customer_id,s.order_date,m.price desc)
select *,case when member = "N" then null
else rank() over(partition by customer_id,member
order by order_date) end as ranking
from rank_cte;
```

customer_id	order_date	join_date	product_name	price	member	ranking
A	2021-01-01	2021-01-07	curry	15	N	NULL
A	2021-01-01	2021-01-07	sushi	10	N	NULL
A	2021-01-07	2021-01-07	curry	15	Y	1
A	2021-01-10	2021-01-07	ramen	12	Y	2
A	2021-01-11	2021-01-07	ramen	12	Y	3
A	2021-01-11	2021-01-07	ramen	12	Y	3
B	2021-01-01	2021-01-09	curry	15	N	NULL
B	2021-01-02	2021-01-09	curry	15	N	NULL
B	2021-01-04	2021-01-09	sushi	10	N	NULL
B	2021-01-11	2021-01-09	sushi	10	Y	1
B	2021-01-16	2021-01-09	ramen	12	Y	2
B	2021-02-01	2021-01-09	ramen	12	Y	3

INSIGHTS

- Customer A spent the most money i.e., $76.
- Customer B was the most frequent visitor of Danny’s Diner i.e., 6 times
- The first order of customer A was sushi, customer B was curry and customer C was ramen.
- The most purchased item was ramen and was purchased 8 times in total.
- Customer A and C loves ramen and customer B loves curry, sushi and ramen equally.
- Customer A was the first member of the Danny’s Diner and his first item was curry
- Before becoming a member, customer A and B spent $25 and $40 respectively.
- Throughout Jan 2021, Customer A, Customer B and Customer C gathered 860 points, 940 points and 360 points respectively.
- We assumed that members can earn 2x a week from the day they became a member with bonus 2x points for sushi, Customer A has 660 points and Customer B has 340 by the end of Jan 2021.