Problem Statement:
Danny is deciding whether he should expand the existing customer loyalty program. We are going to create functional SQL queries to aid his decision making.

Entity Relationship Diagram & Dataset:
Source: https://8weeksqlchallenge.com/case-study-1/

Inspiration: https://www.datawithdanny.com/

/* --------------------

  Case Study Questions

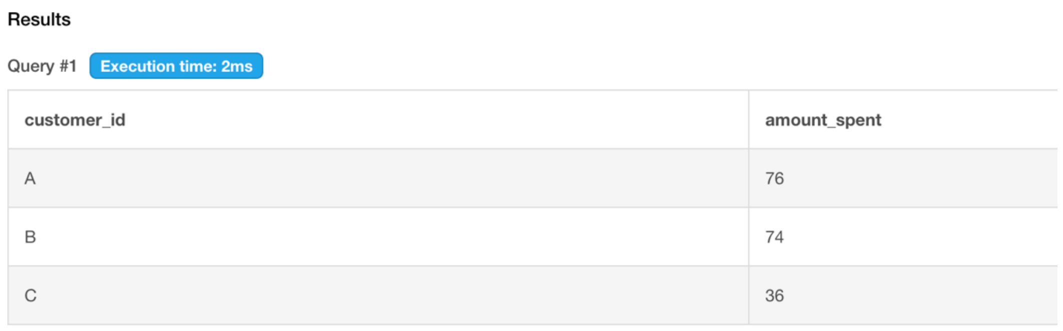
  --------------------*/

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

-- left join sales with menu. We use left join to reserve the left row of sales and keep customer_id who has been making purchases
-- use sum() to compute the total amount each customer spent

```
select s.customer_id, sum(price) as amount_spent
from dannys_diner.sales s
left join dannys_diner.menu m on s.product_id = m.product_id
group by s.customer_id
order by amount_spent desc;
```

Output:


Results

Query #1 Execution time: 2ms

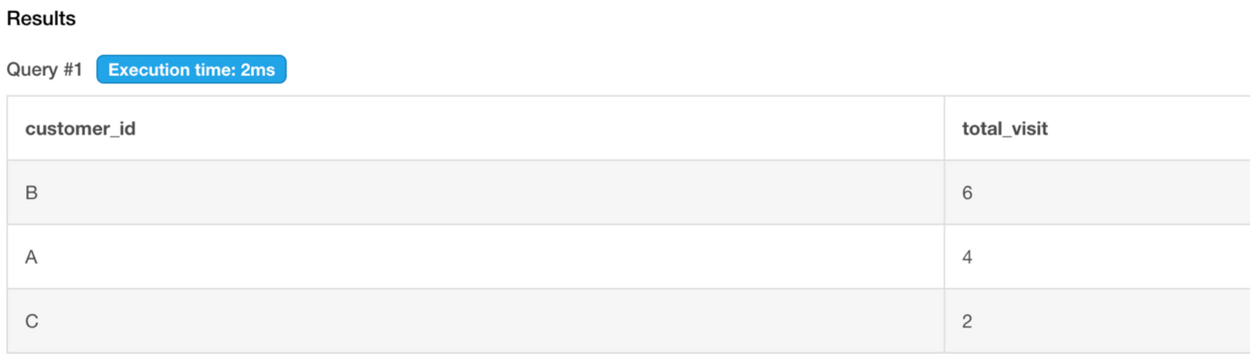
customer_id	amount_spent
A	76
B	74
C	36

Looks like customer A, and customer B spent the most.

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

```
  select s.customer_id, count(distinct order_date) total_visit
  from dannys_diner.sales s

  left join dannys_diner.members m on s.customer_id = m.customer_id

  left join dannys_diner.menu mu on mu.product_id = s.product_id

   group by s.customer_id

   order by total_visit desc;
```

Output:



Results

Query #1 Execution time: 2ms

customer_id	total_visit
B	6
A	4
C	2

Customer B and A visited the restaurant more frequently than others.

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

////
modifying

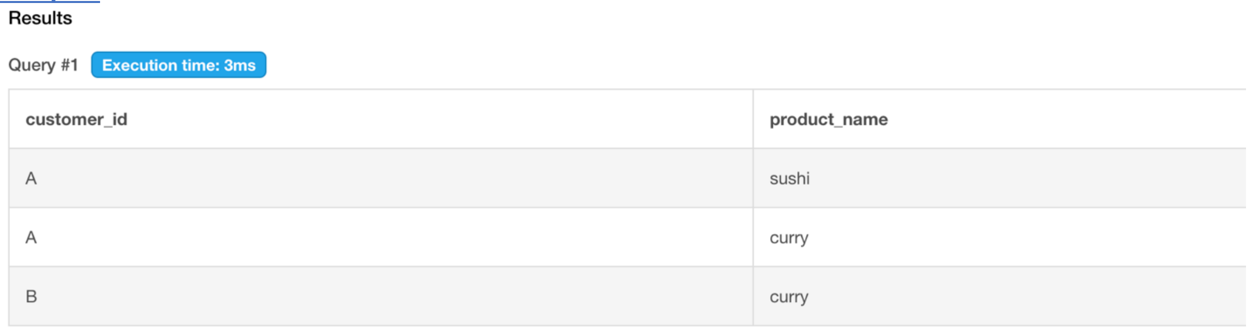
-- 4. What is the most purchased item on the menu and how many times was it purchased by all customers?
/// modifying

```
  with sales_date

  as

  (select customer_id, min(date(order_date)) as dt

  from dannys_diner.sales

  group by 1)

  select s.customer_id, s.order_date
```

```
   from dannys_diner.sales s

   inner join sales_date sd

   on s.customer_id = sd.customer_id

   and s.order_date = sd.dt

   inner join dannys_diner.menu m

   on m.product_id = s.product_id;




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


with

purchase as

(select s.customer_id, m.product_name, count(s.product_id) as total

      from dannys_diner.sales s

      inner join dannys_diner.menu m

      on m.product_id = s.product_id

      group by 1, 2

      order by 1),


pur as

    (select customer_id, max(total) as pop

    from purchase

    group by customer_id),
final as
   (select p.customer_id, p.product_name, pi.pop
   from purchase p
   inner join pur pi
   on p.customer_id = pi.customer_id
   and p.total = pi.pop)

select * from final;



-- 6. Which item was purchased first by the customer after they became a member?
with
cte as
(select s.customer_id, s.order_date, s.product_id, m.join_date,
min(order_date) over(partition by s.customer_id) as first_date
from dannys_diner.sales s
inner join dannys_diner.members m
on s.customer_id = m.customer_id
where m.join_date <= s.order_date)

select c.customer_id, order_date, first_date, product_name
from cte c
left join dannys_diner.menu m
on c.product_id = m.product_id
```

```
where order_date = first_date

order by c.customer_id;


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

-- find the product bought before customer become member

  with cte

  as

  (select s.customer_id, s.order_date, s.product_id,

  min(order_date) over(partition by s.customer_id) as first_order_date

  from dannys_diner.sales s

  inner join dannys_diner.members m  on s.customer_id = m.customer_id

  where order_date < join_date)

  select customer_id, product_name

  from cte c

  left join dannys_diner.menu m on c.product_id = m.product_id

  where order_date = first_order_date

  order by c.customer_id;
```

Output:



Results

Query #1 Execution time: 3ms

customer_id	product_name
A	sushi
A	curry
B	curry

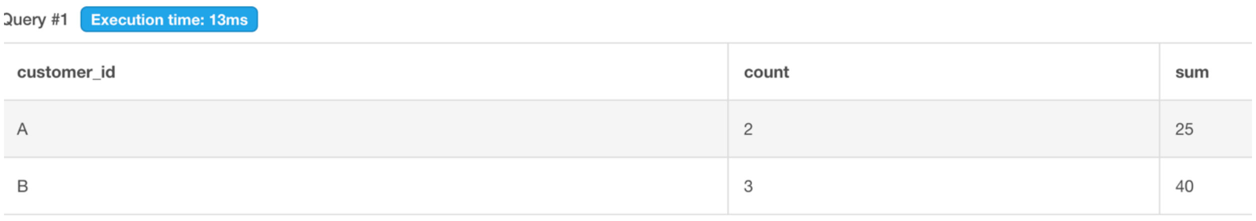
Customer 1 ordered sushi and curry, whereas customer 2 ordered curry before he/she became the member.

Ah ha! Sushi and curry are the favorite items in the restaurant

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

```
with

sales_members as

(select m.customer_id, s.product_id

from dannys_diner.members m

inner join dannys_diner.sales s on m.customer_id = s.customer_id

where order_date < join_date)

select  c.customer_id, count(m.product_id), sum(price)

from sales_members c

left join dannys_diner.menu m on c.product_id = m.product_id

group by c.customer_id

order by c.customer_id;
```

output:



Query #1 Execution time: 13ms

customer_id	count	sum
A	2	25
B	3	40

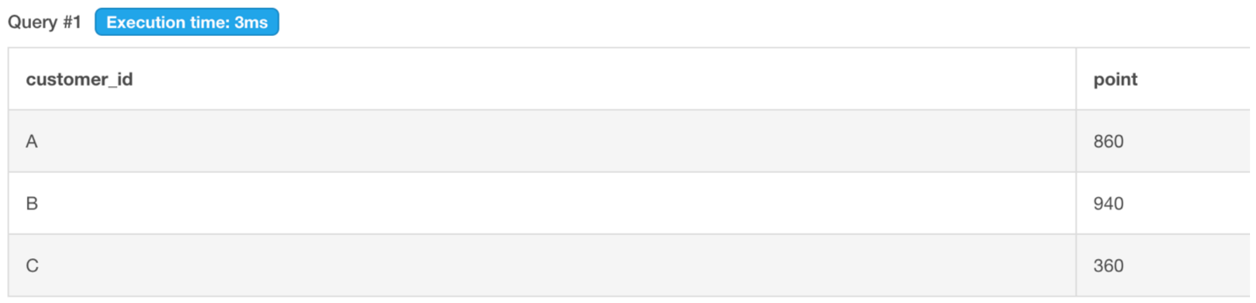
Customer B bought 3 items and spent 40. customer A spent less at 25 with 2 items.

Ah ha! Customer B could be a very potential customer

-- 9. If each $1 spent equates to 10 points and sushi has a 2x points multiplier - how many points would each customer have?
-- create a case statement to compute the point

```
 select s.customer_id,

        sum(case

          when m.product_name = 'sushi' then m.price * 10 * 2

          else m.price *10

          end) as point

  from dannys_diner.sales s

  inner join dannys_diner.menu m  on s.product_id = m.product_id

  group by s.customer_id

  order by customer_id;
```

Output:
We got the total point for client A is 860, client B is 940, client C is 360



Query #1 Execution time: 3ms

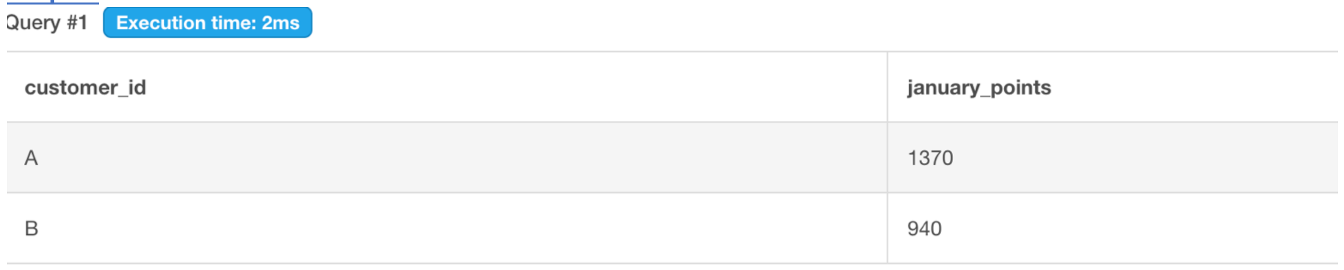
customer_id	point
A	860
B	940
C	360

-- 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:
-- create a timeline table that contains join_date, and first_week(after become member)
-- left join timeline table with sales and menu
-- create a case statement to compute the point

```
with
timeline as
(select
  customer_id,
  join_date,
  join_date + 6 as first_week,
 extract(month from join_date) as month
from dannys_diner.members)

select t.customer_id,
       sum(case
         when product_name = 'sushi' then (price * 2 * 10)
        when order_date between join_date
                                and first_week then (price * 10 * 2)
        else price * 10
        end) as january_points
from timeline t
left join  dannys_diner.sales s on s.customer_id = t.customer_id
left join dannys_diner.menu m on m.product_id = s.product_id
where month = 1
group by t.customer_id
order by t.customer_id;
```

output:



Query #1 Execution time: 2ms

customer_id	january_points
A	1370
B	940