8WEEKSQLCHALLENGE.COM

CASE STUDY #1

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 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 very basic data from their few months of operation but have no idea how to use their data to help them run the business.
- Direct link.

Problem Statement

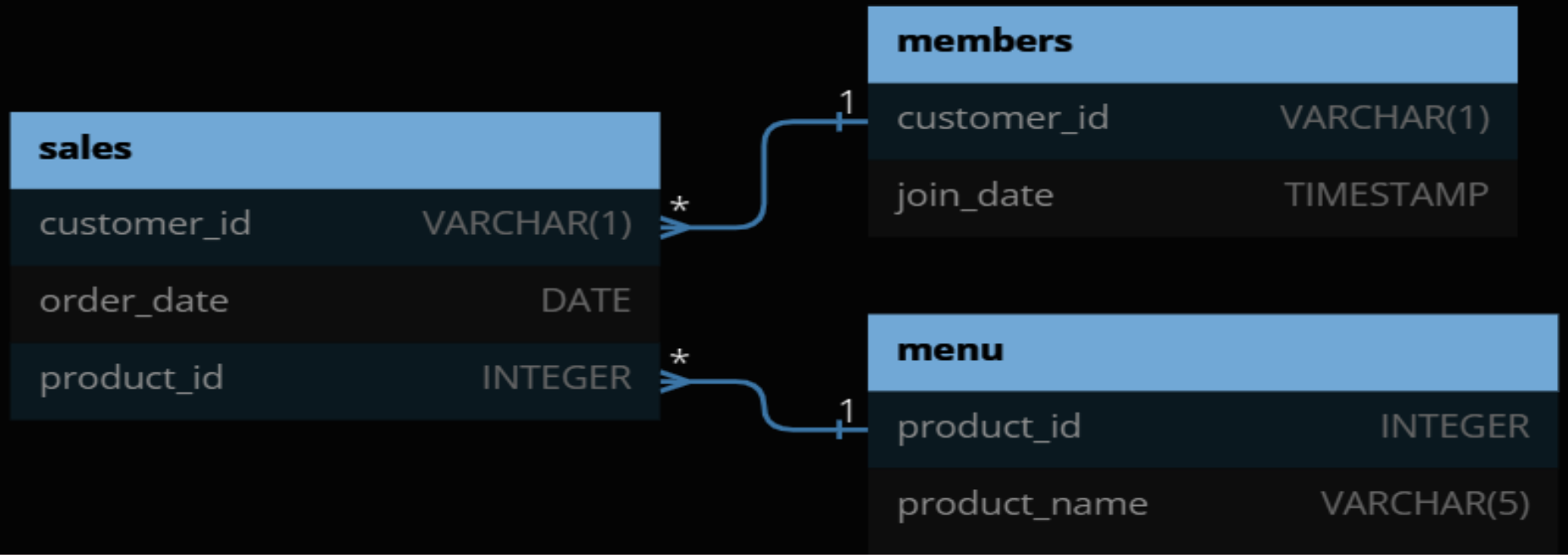
- Danny wants to use the data to answer a few simple questions about his customers, especially about their visiting patterns, how much money they've spent, and which menu items are their favourite. Having this deeper connection with his customers will help him deliver a better and more personalised experience 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



Case Study Questions

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

```
select s.customer_id, sum(m.price) as total_spent
from sales s
inner join menu m
on s.product_id = m.product_id
group by customer_id;
```

customer_id	total_spent
A	76
B	74
C	36

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

customer_id	no_of_days
A	4
B	6
C	2

```
select customer_id,count(distinct(order_date)) as no_of_days
from sales
group by customer_id;
```

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

```sql
with first_order as
(
select s.customer_id, s.product_id, m.product_name,
row_number() over(partition by customer_id) as order_numbering
from sales s
inner join
menu m
on s.product_id = m.product_id
)
select customer_id, product_name
from first_order
where order_numbering = 1;
```

customer_id	product_name
A	sushi
B	curry
C	ramen

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

```sql
with number_sold as
(
select m.product_name, count(s.product_id) as number_of_sale
from sales s
inner join menu m
on s.product_id = m.product_id
group by m.product_name
)
select  product_name, number_of_sale
from number_sold
group by product_name
order by number_of_sale desc
limit 1;

-- how many times it was purchase by all the customers
select product_id from menu
where product_name = "ramen";

select customer_id, count(product_id)
from sales
where product_id = 3
group by customer_id;
```

	product_name	number_of_sale
▶	ramen	8

	customer_id	count(product_id)
▶	A	3
	B	2
	C	3

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

```sql
with popular as
(
select s.customer_id, s.product_id, m.product_name,
dense_rank() over(partition by customer_id  order by product_id) as number_of_times
from sales s
inner join
menu m
on
s.product_id = m.product_id
)
select customer_id ,product_name, count(number_of_times) as no_of_times
from popular
-- where customer_id = "C"
group by customer_id,product_name
order by customer_id,no_of_times desc;
```

customer_id	product_name	no_of_times
A	ramen	3
A	curry	2
A	sushi	1
B	sushi	2
B	curry	2
B	ramen	2
C	ramen	3

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

```sql
with pur_after_mem as(
select s.customer_id,s.order_date,mu.product_id, mu.product_name,
lead(s.product_id) over(partition by customer_id order by s.order_date) as next_id
from sales s
inner join members m on s.customer_id = m.customer_id
inner join menu mu
on s.product_id = mu.product_id
)
(select p.customer_id, p.product_name as purch_product_after_joining
from pur_after_mem p
inner join members mem
on p.customer_id = mem.customer_id
where p.customer_id = 'A'
AND order_date > (select join_date from members where customer_id = 'A')
order by order_date limit 1)
UNION ALL
(select p.customer_id, p.product_name as purch_product_after_joining
from pur_after_mem p
inner join members mem
on p.customer_id = mem.customer_id
where p.customer_id = 'B'
AND order_date > (select join_date from members where customer_id = 'B')
order by order_date limit 1);
```

customer_id	purch_product_after_joining
A	ramen
B	sushi

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

```
with proper_date as(
select s.customer_id,s.order_date,mu.product_id, mu.product_name,
lag(s.product_id) over(partition by customer_id order by s.order_date) as late_id
from sales s
inner join members m on s.customer_id = m.customer_id
inner join menu mu
on s.product_id = mu.product_id)
(select p.customer_id, p.product_name as last_product_before_joining
from proper_date p
inner join members mem on p.customer_id = mem.customer_id
where p.customer_id = 'A'and
(order_date = (select join_date from members where customer_id = 'A') OR
order_date < (select join_date from members where customer_id = 'A'))
order by order_date desc limit 1)
union all
(select p.customer_id, p.product_name as last_product_before_joining
from proper_date p
inner join members mem on p.customer_id = mem.customer_id
where p.customer_id = 'B'and
(order_date = (select join_date from members where customer_id = 'B') OR
order_date < (select join_date from members where customer_id = 'B'))
order by order_date desc limit 1);
```

customer_id	last_product_before_joining
A	curry
B	sushi

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

```
with purchase_before as(
select s.customer_id,s.order_date,m.product_id, m.price
from sales s
inner join
menu m on s.product_id = m.product_id
inner join
members mem on s.customer_id = mem.customer_id
order by s.customer_id, s.order_date
)
(select p.customer_id, count(product_id) as total_items, sum(price) as total_spent
from purchase_before p
inner join
members mem on p.customer_id = mem.customer_id
where p.customer_id = 'A' AND order_date < (select join_date from members where customer_id = 'A')
group by customer_id)
union all
(select p.customer_id, count(product_id) as total_items, sum(price) as total_spent
from purchase_before p
inner join
members mem on p.customer_id = mem.customer_id
where p.customer_id = 'B' AND order_date < (select join_date from members where customer_id = 'B')
group by customer_id)
```

customer_id	total_items	total_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?

```sql
with basic_detail as
(
select s.customer_id, s.product_id, m.price, m.product_name
from sales s
inner join
menu m on s.product_id = m.product_id
)
select customer_id,
-- product_name, price,
sum(case
    when product_name = "sushi" then (price * (2*price))
    -- when product_name = "curry" then price * price
    else price * price
end) as points
from basic_detail
group by customer_id;
```

	customer_id	points
▶	A	1082
	B	1138
	C	432

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?

```sql
with first_week as(
select s.customer_id, s.order_date,s.product_id,mu.product_name,mu.price, mem.join_date
from sales s
inner join menu mu on s.product_id = mu.product_id
inner join members mem on s.customer_id = mem.customer_id
order by s.customer_id
)
select customer_id,
sum(case
    when order_date between (select join_date from members where customer_id = 'A')
                        and (select join_date + interval 6 day from members where customer_id = 'A')
    then (price * (2*price))
    when order_date between (select join_date from members where customer_id = 'B')
                        and (select join_date+ interval 6 day from members where customer_id = 'B')
    then (price * (2*price))
    else price*price
end) as "total point"
from first_week
where month(order_date) = 1
group by customer_id
order by customer_id;
```

	customer_id	total point
▶	A	1639
	B	894