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 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

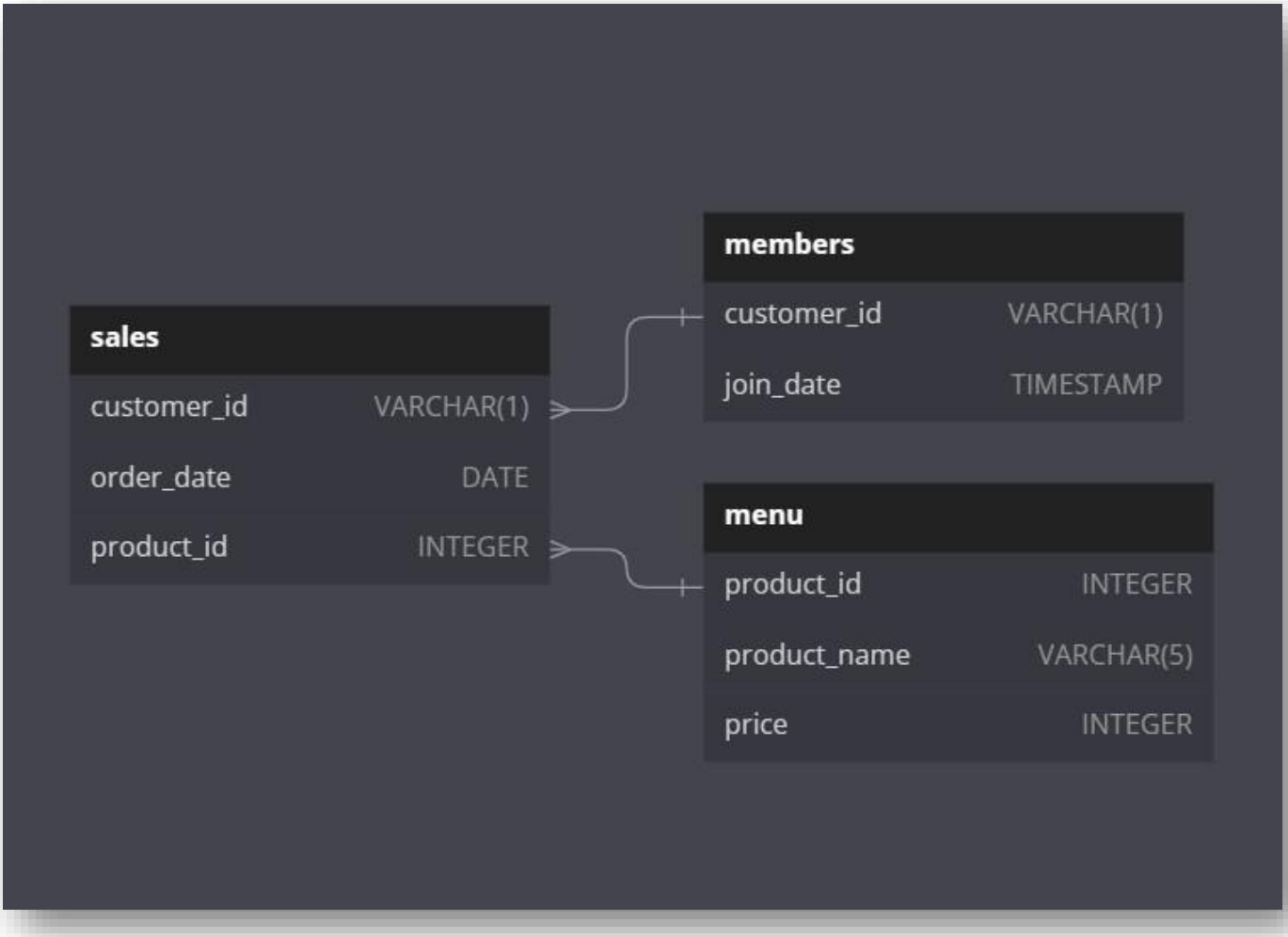


Table 1: sales

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?

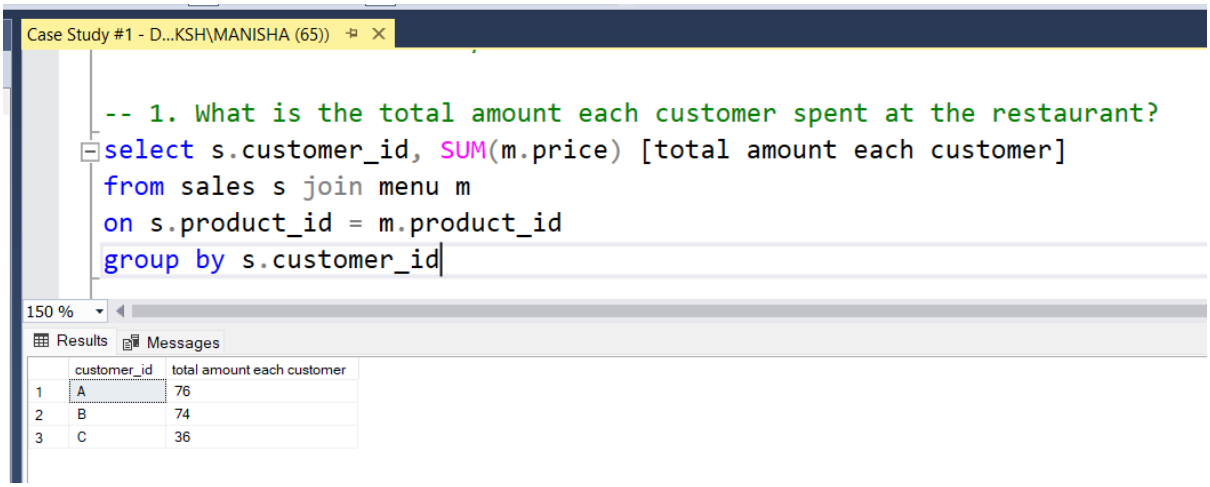
Solution

I used **Microsoft SQL Server Workbench** and these are the particular functions I employed:

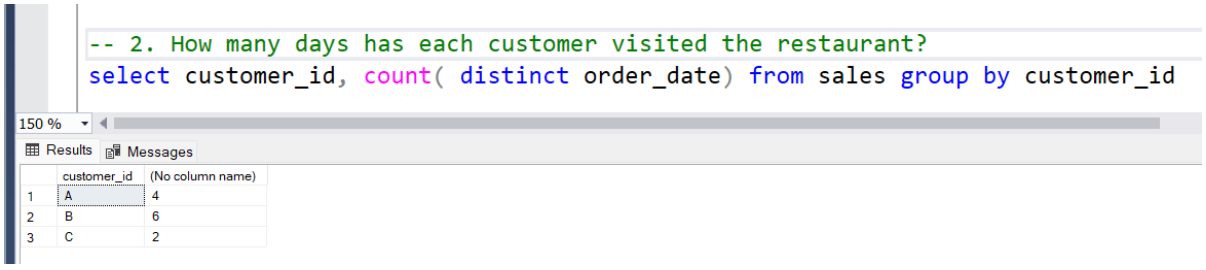
- Aggregate functions (SUM, COUNT)
- Table Joins
- Ranking functions (RANK, DENSE_RANK)

- Date Functions
- Common Table Expressions (CTE)
- Window function

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



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



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

```
-- 3. What was the first item from the menu purchased by each customer?
select s.customer_id, s.product_name from
(
    select customer_id, order_date, m.product_name ,
    row_number() over(partition by customer_id order by order_date) as rn
    from sales s join menu m on s.product_id = m.product_id
) s where rn = 1
```

150 %

Results | Messages

	customer_id	product_name
1	A	sushi
2	B	curry
3	C	ramen

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

```
-- 4. What is the most purchased item on the menu and how many times was it purchased by all customers?
select top 1 s.product_id,m.product_name, COUNT(s.product_id) cnt_prod
from sales s join menu m
on s.product_id = m.product_id
group by s.product_id, m.product_name
order by cnt_prod desc
```

150 %

Results | Messages

	product_id	product_name	cnt_prod
1	3	ramen	8

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

```
-- 5. Which item was the most popular for each customer?
with popular_item_cte as
(
select customer_id, m.product_name, COUNT(s.product_id) as cnt,
ROW_NUMBER() over(partition by s.customer_id order by COUNT(*) desc) as rn
from sales s left join menu m
on s.product_id = m.product_id
group by customer_id, m.product_name
)
select customer_id, product_name
from popular_item_cte
where rn = 1
```

150 %

Results | Messages

	customer_id	product_name
1	A	ramen
2	B	sushi
3	C	ramen

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

```sql
-- 6. Which item was purchased first by the customer after they became a member?
with cte_first_item_after_member
as
(
select s.customer_id, m.product_name , s.order_date, mem.join_date,
DENSE_RANK() over(partition by s.customer_id order by s.order_date) dr
from sales s join menu m
on s.product_id = m.product_id
join members mem on s.customer_id = mem.customer_id
where s.order_date >= mem.join_date
)
select customer_id, product_name as first_product
from cte_first_item_after_member
where dr = 1
```

150 %

Results | Messages

	customer_id	first_product
1	A	curry
2	B	sushi

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

```sql
-- 7. Which item was purchased just before the customer became a member?
with cte_item_before_member
as
(
select s.customer_id, m.product_name , s.order_date, mem.join_date,
DENSE_RANK() over(partition by s.customer_id order by s.order_date) dr
from sales s join menu m
on s.product_id = m.product_id
join members mem on s.customer_id = mem.customer_id
where s.order_date <= mem.join_date
)
select *
from cte_item_before_member
where dr = 1
```

150 %

Results | Messages

	customer_id	product_name	order_date	join_date	dr
1	A	sushi	2021-01-01	2021-01-07	1
2	A	curry	2021-01-01	2021-01-07	1
3	B	curry	2021-01-01	2021-01-09	1

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

```sql
-- 8. What is the total items and amount spent for each member before they became a member?
select s.customer_id, COUNT(s.product_id) as total_items,
'$'+cast(SUM(m.price) as varchar) as amount_spent
from sales s left join menu m
on s.product_id = m.product_id
left join members mem on s.customer_id = mem.customer_id
where s.order_date <= mem.join_date
group by s.customer_id
```

150 %

Results | Messages

	customer_id	total_items	amount_spent
1	A	3	$40
2	B	3	$40

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

```sql
-- 9.  If each $1 spent equates to 10 points and sushi has a 2x points multiplier
-- how many points would each customer have?
with cte_total_points as
(
    select s.customer_id, m.product_name ,
    case
        when product_name = 'sushi' then 20 * m.price
        else m.price*10
        end as points
    from sales s
    left join menu m
    on s.product_id = m.product_id
)
select customer_id, sum(points) as total_points
from cte_total_points
group by customer_id
```

150 %

Results | Messages

	customer_id	total_points
1	A	860
2	B	940
3	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
select s.customer_id,
    sum(case
        when DATEDIFF(day, mem.join_date, s.order_date) BETWEEN 0 AND 6
        then 20 * m.price
        else m.price*10
        end) as points
    from sales s
    left join menu m
    on s.product_id = m.product_id
    left join members mem
    on s.customer_id = mem.customer_id
    where month(s.order_date) = 1 and s.customer_id in ('A','B')
    group by s.customer_id
```

125 %

Results | Messages

	customer_id	points
1	A	1270
2	B	720

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
--Bonus questions
-- Join All The Things
select s.customer_id, s.order_date, m.product_name, m.price,
case
when s.order_date < mem.join_date AND S.customer_id = MEM.customer_id then 'N'
when mem.join_date IS NULL then 'N'
    else 'Y'
    end as [member]
    from sales s
    left join menu m
    on s.product_id = m.product_id
    left join members mem
    on s.customer_id = mem.customer_id
```

125 %

Results | Messages

	customer_id	order_date	product_name	price	member
1	A	2021-01-01	sushi	10	N
2	A	2021-01-01	curry	15	N
3	A	2021-01-07	curry	15	Y
4	A	2021-01-10	ramen	12	Y
5	A	2021-01-11	ramen	12	Y
6	A	2021-01-11	ramen	12	Y
7	B	2021-01-01	curry	15	N
8	B	2021-01-02	curry	15	N
9	B	2021-01-04	sushi	10	N
10	B	2021-01-11	sushi	10	Y
11	B	2021-01-16	ramen	12	Y
12	B	2021-02-01	ramen	12	Y
13	C	2021-01-01	ramen	12	N
14	C	2021-01-01	ramen	12	N
15	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.

```sql
-- Rank All The Things
WITH CTE AS
(
    select s.customer_id, s.order_date, m.product_name, m.price,
    case
    when s.order_date < mem.join_date AND S.customer_id = MEM.customer_id then 'N'
    when mem.join_date IS NULL then 'N'
        else 'Y'
        end as [member]
        from sales s
        left join menu m
        on s.product_id = m.product_id
        left join members mem
        on s.customer_id = mem.customer_id
)
SELECT *,
    CASE WHEN [member] = 'N' THEN NULL
    ELSE RANK() OVER (PARTITION BY CUSTOMER_ID, [MEMBER] ORDER BY ORDER_DATE)
    END AS RANKING
FROM CTE
```

125 %

Results | Messages

	customer_id	order_date	product_name	price	member	RANKING
1	A	2021-01-01	sushi	10	N	NULL
2	A	2021-01-01	curry	15	N	NULL
3	A	2021-01-07	curry	15	Y	1
4	A	2021-01-10	ramen	12	Y	2
5	A	2021-01-11	ramen	12	Y	3
6	A	2021-01-11	ramen	12	Y	3
7	B	2021-01-01	curry	15	N	NULL
8	B	2021-01-02	curry	15	N	NULL
9	B	2021-01-04	sushi	10	N	NULL
10	B	2021-01-11	sushi	10	Y	1
11	B	2021-01-16	ramen	12	Y	2
12	B	2021-02-01	ramen	12	Y	3
13	C	2021-01-01	ramen	12	N	NULL
14	C	2021-01-01	ramen	12	N	NULL
15	C	2021-01-07	ramen	12	N	NULL

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.