Case Study #1 - Danny's Diner

Introduction

Danny seriously loves Japanese food so at the beginning of 2021, he decides to embark upon a risky venture and opens up a cute little restaurant that sells his 3 favorite 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 its few months of operation but has no idea how to use its 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 favorite. Having this deeper connection with his customers will help him deliver a better and more personalized experience for his loyal customers.

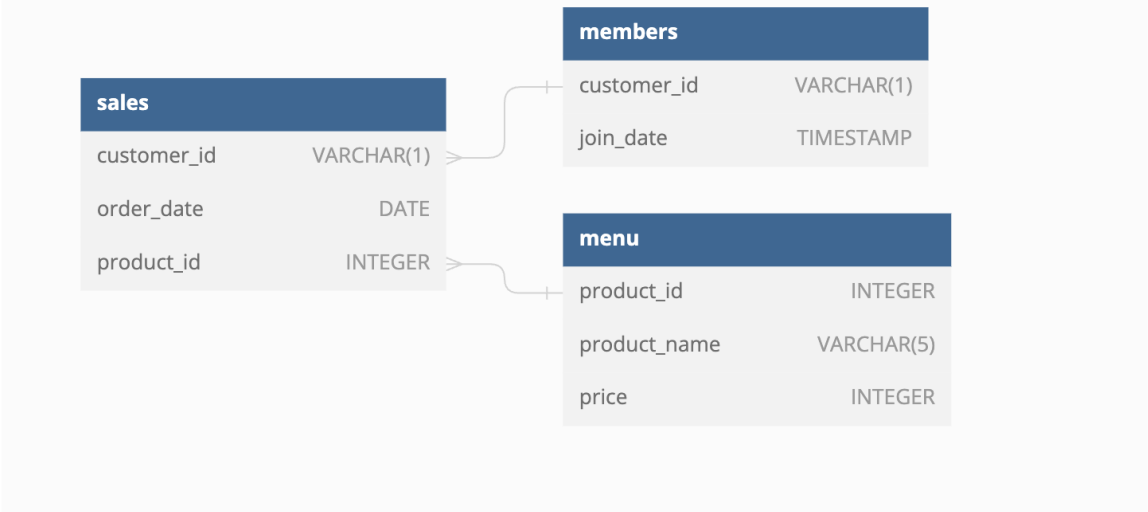
He plans on using these insights to help him decide whether he should expand the existing customer loyalty program - additionally, he needs help to generate some basic datasets so his team can easily inspect the data without needing to use SQL.

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`

Entity Relationship Diagram



Case Study Questions:

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

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

customer_id	total_amt_spent
A	76
B	74
C	36

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

```
select
  customer_id,
  count(
    distinct(order_date)
  ) as num_days_visited
from dannys_diner.sales
group by customer_id;
```

customer_id	num_days_visited
A	4
B	6
C	2

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

```
with cte(
        select customer_id, product_id
          from (select customer_id, product_id,
                       rank() over(partition by customer_id
                       order by order_date) as rnk
                from  dannys_diner.sales
                ) x where  rnk = 1)
select
  distinct customer_id,
  product_name as first_item_ordered
from dannys_diner.menu m
inner join cte c on m.product_id = c.product_id;
```

customer_id	first_item_ordered
A	curry
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?**

```
with cte(
        select product_id,
            count(product_id) as cnt
        from dannys_diner.sales
        group by product_id)
select m.product_name as most_purchased_item,
        x.cnt as cnt_purchases
from dannys_diner.menu as m
inner join cte x on m.product_id = x.product_id
order by x.cnt desc limit 1;
```

most_purchased_item	cnt_purchases
ramen	8

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

```
with grouped as (
    select customer_id, product_id, count(product_id) as cnt
    from dannys_diner.sales
    group by customer_id, product_id),
cte as (
    select  customer_id,  product_id, cnt,  rank() over(
        partition by customer_id order by cnt desc) as rnk
    from grouped),
cte_1 as (
    select customer_id,  product_id, cnt as times_purchased
```

```
        from cte
        where rnk = 1),
    cte_2 as (
        select customer_id, m.product_name as product_name, times_purchased
        from cte_1
        inner join dannys_diner.menu m on m.product_id = cte_1.product_id)
    select * from cte_2 order by customer_id;
```

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

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

```
    with cte as (
            select customer_id,product_id, order_date
            from (
               select s.customer_id as customer_id,
                 product_id as product_id,
                 s.order_date as order_date,
                 rank() over(partition by s.customer_id
                   order by order_date) as rnk
               from dannys_diner.sales s
               inner join dannys_diner.members m
               on m.customer_id = s.customer_id
               and s.order_date >= m.join_date) x
            where  rnk = 1),
    cte_1 as (
            select  customer_id, m.product_name, order_date
            from cte
            inner join dannys_diner.menu as m
            on m.product_id = cte.product_id)
    select * from cte_1 order by customer_id;
```

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

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

```
with cte as (
        select s.customer_id as customer_id,
                s.order_date as order_date,
                s.product_id as product_id,
                m.join_date, join_date-order_date as diff
        from dannys_diner.sales s
        inner join dannys_diner.members m
        on m.customer_id=s.customer_id
        and s.order_date<m.join_date),
cte2 as (
        select customer_id,
        min(join_date-order_date) as difference
        from cte
        group by customer_id)
select x1.customer_id,m.product_name ,x1.order_date
from cte x1
inner join cte2 x2 on x1.customer_id=x2.customer_id and x2.difference=x1.diff
inner join dannys_diner.menu as m on m.product_id=x1.product_id
order by x1.customer_id;
```

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

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

```
with cte1 as (
        select s.customer_id as customer_id,
                s.product_id as product_id
        from dannys_diner.sales s
```

```
        join dannys_diner.members m on m.customer_id=s.customer_id
        and s.order_date<m.join_date),
cte2 as (
        select  distinct customer_id,product_id,count(product_id) as cnt
        from cte1
        group by customer_id, product_id)
select cte2.customer_id,
        count(cte2.cnt) as num_prod_ordered,
        sum(cte2.cnt*m.price) as total_price
from cte2 inner join dannys_diner.menu m
on cte2.product_id=m.product_id
group by cte2.customer_id
order by cte2.customer_id;
```

customer_id	num_prod_ordered	total_price
A	2	25
B	2	40

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 as (
        select
          s.customer_id,
          m.product_name,
          count(m.product_name) as cnt,
          min(price) as price
        from dannys_diner.sales s
        inner join dannys_diner.menu m
        on s.product_id = m.product_id
        group by s.customer_id, m.product_name),
cte_1 as (
        select customer_id, product_name, cnt * price * 10 as total
        from cte)
select customer_id, sum(
                case when product_name = 'sushi'
                        then total * 2 else total end) as points
from cte_1 group by customer_id;
```

customer_id	points
B	940
C	360
A	860

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 customers A and B have at the end of January?

```
with cte as (
        select
          s.customer_id as customer_id,
          s.order_date as order_date,
          me.product_id as product_id,
          m.join_date as join_date,
          me.price as price,
          case when order_date between join_date
                    and join_date + interval '6day' or me.product_name = 'sushi'
               then me.price * 2 else me.price end as new_price
        from dannys_diner.sales s
        inner join dannys_diner.members m on m.customer_id = s.customer_id
        inner join dannys_diner.menu me on me.product_id = s.product_id
        where order_date between '2021-01-01' and '2021-01-31')
select customer_id, sum(new_price * 10) as points
from cte
group by customer_id
order by customer_id;
```

customer_id	points
A	1370
B	820

Bonus Question

11. **Danny and his team can use quickly derive insights without needing to join the underlying tables using SQL Recreate the following table output using the available data.**

```
select
  s.customer_id as customer_id,
  s.order_date as order_date,
  me.product_name as product_name,
  me.price as price,
  case when s.order_date < m.join_date or m.join_date is null
        then 'N' when s.order_date >= m.join_date then 'Y' end as member
from dannys_diner.sales s
left join dannys_diner.members m on m.customer_id = s.customer_id
left join dannys_diner.menu me on s.product_id = me.product_id
order by customer_id, price desc;
```

customer_id	order_date	product_name	price	member
A	2021-01-01	curry	15	N
A	2021-01-01	sushi	10	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

12. **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 cte as (
        select  s.customer_id as customer_id,
                s.order_date as order_date,
                me.product_name as product_name,
                me.price as price,
                case when s.order_date<m.join_date or  m.join_date is null
                        then 'N'
                        when s.order_date>=m.join_date then 'Y' end as member
        from dannys_diner.sales s
        left join dannys_diner.members m on m.customer_id=s.customer_id
        left join dannys_diner.menu me on s.product_id=me.product_id)
select customer_id, order_date, product_name,
        case when member ='Y'
                then dense_rank() over(partition by customer_id,
                        member order by order_date)
                when member ='N' then null end as ranking
from cte order by customer_id;
```

customer_id	order_date	product_name	ranking
A	2021-01-01	sushi	*null*
A	2021-01-01	curry	*null*
A	2021-01-07	curry	1
A	2021-01-10	ramen	2
A	2021-01-11	ramen	3
A	2021-01-11	ramen	3
B	2021-01-01	curry	*null*
B	2021-01-02	curry	*null*
B	2021-01-04	sushi	*null*
B	2021-01-11	sushi	1
B	2021-01-16	ramen	2
B	2021-02-01	ramen	3
C	2021-01-01	ramen	*null*
C	2021-01-01	ramen	*null*
C	2021-01-07	ramen	*null*