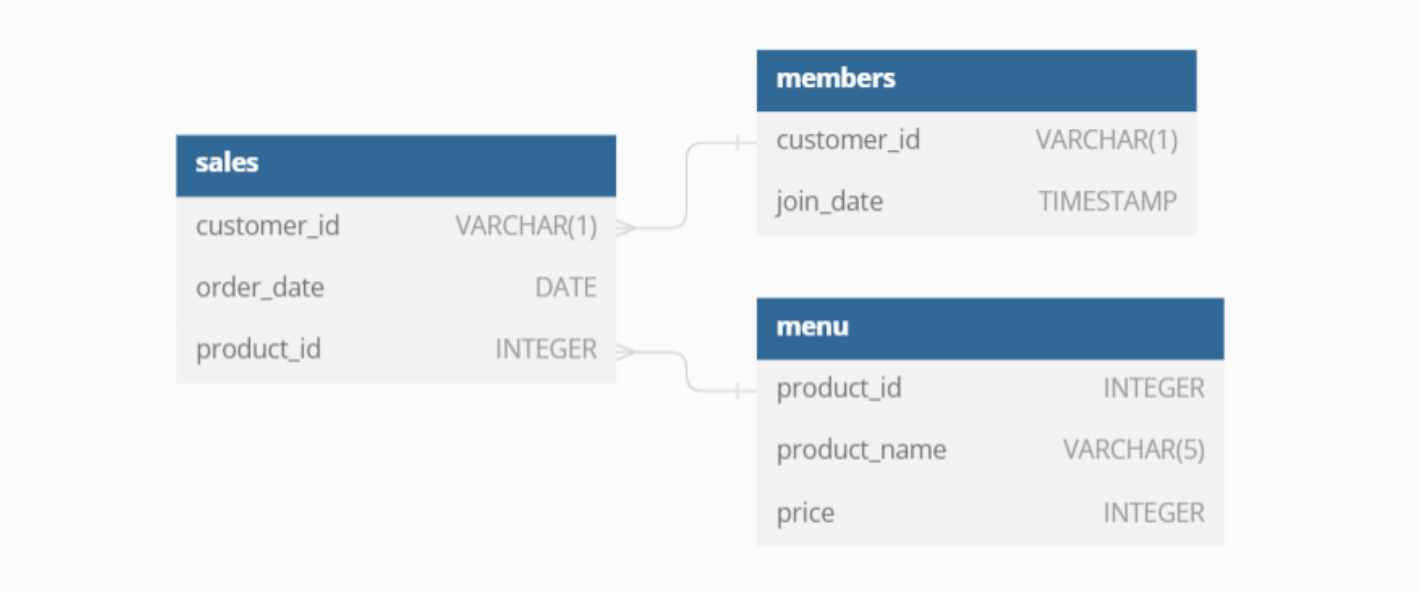
Danny's Diner - Case Study

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.
Danny has shared with use key datasets for this case study:

- `sales`
- `menu`
- `members`

Entity Relationship Diagram



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?

```
SELECT s.customer_id, SUM(m.price) as Total_amount
FROM dannys_diner.sales s
INNER JOIN dannys_diner.menu m on s.product_id = m.product_id
GROUP BY s.customer_id;
```

customer_id	total_amount
B	74
C	36
A	76

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

```
SELECT customer_id, COUNT(order_date) as customer_visited
FROM dannys_diner.sales
GROUP BY customer_id
ORDER BY customer_visited DESC;
```

customer_id	customer_visited
B	6
A	6
C	3

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

```
SELECT
s.customer_id,
m.product_name,
s.order_date
FROM
dannys_diner.sales s
JOIN
dannys_diner.menu m ON s.product_id = m.product_id
WHERE
s.order_date = (
SELECT MIN(order_date)
FROM dannys_diner.sales
WHERE customer_id = s.customer_id
);
```

customer_id	product_name	order_date
A	sushi	2021-01-01T00:00:00.000Z
A	curry	2021-01-01T00:00:00.000Z
B	curry	2021-01-01T00:00:00.000Z
C	ramen	2021-01-01T00:00:00.000Z

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

```
SELECT m.product_name, COUNT(m.product_id) as Most_purchased
FROM dannys_diner.sales s
INNER JOIN dannys_diner.menu m on m.product_id = s.product_id
GROUP BY m.product_name
ORDER BY Most_purchased DESC LIMIT 1;
```

product_name	most_purchased
ramen	8

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

```
WITH most_popular AS (
SELECT
sales.customer_id,
menu.product_name,
COUNT(menu.product_id) AS order_count,
DENSE_RANK() OVER(
PARTITION BY sales.customer_id
ORDER BY COUNT(sales.customer_id) DESC) AS rank
```

```
FROM dannys_diner.menu
JOIN dannys_diner.sales
ON menu.product_id = sales.product_id
GROUP BY sales.customer_id, menu.product_name
)
SELECT
customer_id,
product_name,
order_count
FROM most_popular
WHERE rank = 1;
```

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

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

```
WITH joined_as_member AS (
SELECT
members.customer_id,
sales.product_id,
ROW_NUMBER() OVER(
PARTITION BY members.customer_id
ORDER BY sales.order_date) AS row_num
FROM dannys_diner.members
JOIN dannys_diner.sales
ON members.customer_id = sales.customer_id
AND sales.order_date > members.join_date
)
SELECT
customer_id,
product_name
FROM joined_as_member
JOIN dannys_diner.menu
ON joined_as_member.product_id = menu.product_id
WHERE row_num = 1
ORDER BY customer_id ASC;
```

customer_id	product_name
A	ramen
B	sushi

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

```
WITH purchased_prior_member AS (
SELECT
members.customer_id,
sales.product_id,
ROW_NUMBER() OVER(
PARTITION BY members.customer_id
ORDER BY sales.order_date DESC) AS rank
FROM dannys_diner.members
JOIN dannys_diner.sales
ON members.customer_id = sales.customer_id
AND sales.order_date < members.join_date
)
SELECT
p_member.customer_id,
menu.product_name
FROM purchased_prior_member AS p_member
JOIN dannys_diner.menu
ON p_member.product_id = menu.product_id
WHERE rank = 1
ORDER BY p_member.customer_id ASC;
```

customer_id	product_name
A	sushi
B	sushi

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, SUM(m.price) AS total_sales
FROM dannys_diner.menu m
INNER JOIN dannys_diner.sales s on s.product_id = m.product_id
INNER JOIN dannys_diner.members me on me.customer_id = s.customer_id
WHERE s.order_date < me.join_date
GROUP BY s.customer_id
ORDER BY s.customer_id;
```

customer_id	total_items	total_sales
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?

```
WITH Point_multi AS (
SELECT product_id,
CASE WHEN
product_id = 1 then price * 20
ELSE price* 10
END AS points
FROM dannys_diner.menu
)
SELECT s.customer_id, SUM(p.points) as Total_points
FROM dannys_diner.sales s
JOIN Point_multi p
ON s.product_id = p.product_id
GROUP BY s.customer_id
ORDER BY s.customer_id;
```

customer_id	total_points
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?

```
WITH customer_points AS (
SELECT
s.customer_id,
s.order_date,
m.product_name,
m.price,
CASE
WHEN s.order_date BETWEEN me.join_date AND me.join_date + INTERVAL '6 day' THEN 2 *
m.price
ELSE m.price
END AS points
FROM dannys_diner.sales s JOIN dannys_diner.menu m ON s.product_id = m.product_id
JOIN dannys_diner.members me ON s.customer_id = me.customer_id
WHERE s.order_date BETWEEN '2021-01-01' AND '2021-01-31'
)
SELECT
```

```
customer_id,
SUM(points) AS total_points
FROM customer_points
GROUP BY customer_id;
```

customer_id	total_points
B	72
A	127

Bonus Questions

Join All The Things

11. Recreate the following table output using the available data:

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

```
SELECT
sales.customer_id,
sales.order_date,
menu.product_name,
menu.price,
CASE
WHEN members.join_date > sales.order_date THEN 'N'
WHEN members.join_date <= sales.order_date THEN 'Y'
ELSE 'N' END AS member_status
FROM dannys_diner.sales
LEFT JOIN dannys_diner.members
ON sales.customer_id = members.customer_id
JOIN dannys_diner.menu
ON sales.product_id = menu.product_id
ORDER BY members.customer_id, sales.order_date
```

Rank all the things

12. Put null ranking values for the records when customers are not yet part of the loyalty program.

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

```
WITH customers_data AS (
SELECT
sales.customer_id,
sales.order_date,
menu.product_name,
menu.price,
CASE
WHEN members.join_date > sales.order_date THEN 'N'
WHEN members.join_date <= sales.order_date THEN 'Y'
ELSE 'N' END AS member_status
FROM dannys_diner.sales
LEFT JOIN dannys_diner.members
ON sales.customer_id = members.customer_id
JOIN dannys_diner.menu
ON sales.product_id = menu.product_id
ORDER BY members.customer_id, sales.order_date
)
SELECT
*,
CASE
WHEN member_status = 'N' then NULL
ELSE RANK () OVER(
PARTITION BY customer_id, member_status
ORDER BY order_date) END AS ranking
FROM customers_data;
```