

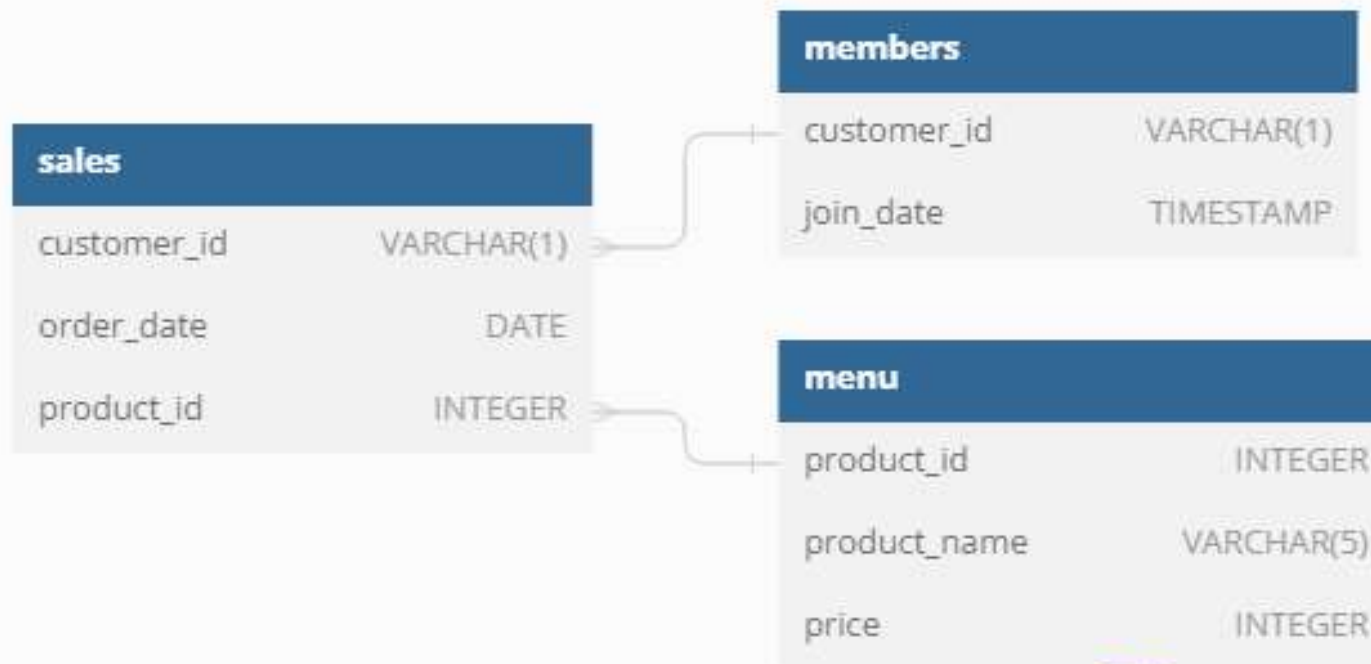
8 Week SQL Challenge

Case Study #1 - Danny's Diner

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

Entity Relationship Diagram



Case Study Questions

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?

ANSWER 1

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

```
2 ✓ SELECT customer_id,  
3       sum(price) as Total_amount  
4 FROM sales as s  
5 Inner Join menu as M  
6       on s.product_id = M.product_id  
7 GROUP BY customer_id  
8 ORDER BY total_amount DESC;  
9
```

Data Output Messages Notifications



	customer_id character varying (1) 🔒	total_amount bigint 🔒
1	A	76
2	B	74
3	C	36

ANSWER 2

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

```
11 SELECT customer_id,  
12        COUNT(DISTINCT(order_date)) as visited_days  
13 FROM sales  
14 GROUP BY customer_id;  
15
```

Data Output Messages Notifications

	customer_id character varying (1)	visited_days bigint
1	A	4
2	B	6
3	C	2

ANSWER 3

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

```
17 WITH CTE as (  
18     SELECT customer_id,  
19     product_name,  
20     ROW_NUMBER () over(partition by customer_id order by order_date ASC) as rn  
21     FROM sales as s  
22     INNER JOIN menu as m  
23     on s.product_id = m.product_id  
24 )  
25 SELECT customer_id,  
26     product_name  
27 FROM CTE  
28 WHERE rn = 1;  
29
```

Data Output Messages Notifications

	customer_id character varying (1) 🔒	product_name character varying (5) 🔒
1	A	curry
2	B	curry
3	C	ramen

ANSWER 4

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

```
32 ▼ SELECT product_name,  
33         count(order_date) as Orders  
34 FROM sales as s  
35 INNER JOIN menu as m  
36 on s.product_id = m.product_id  
37 GROUP BY product_name  
38 LIMIT 1;  
39
```

Data Output Messages Notifications

		
	product_name character varying (5) 	orders bigint 
1	ramen	8

ANSWER 5

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

```
41 WITH CTE as(  
42     SELECT product_name,  
43     customer_id,  
44     count(order_date) as Orders,  
45     RANK() OVER(PARTITION BY customer_id order by COUNT(order_date) DESC) as rnk  
46     FROM sales as s  
47     INNER JOIN menu as m  
48     on s.product_id = m.product_id  
49     GROUP BY product_name,  
50     customer_id  
51     ORDER BY customer_id  
52 )  
53 SELECT product_name,  
54 customer_id,  
55 orders  
56 from CTE  
57 where rnk = 1
```

Data Output Messages Notifications



	product_name character varying (5)	customer_id character varying (1)	orders bigint
1	ramen	A	3
2	sushi	B	2
3	curry	B	2
4	ramen	B	2
5	ramen	C	3

ANSWER 6

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

```
60 WITH CTE as(  
61     SELECT s.customer_id,  
62            order_date,  
63            product_name,  
64            join_date,  
65            RANK() OVER(PARTITION BY s.customer_id ORDER BY order_date) as rnk  
66     FROM sales as s  
67     INNER JOIN members as mem on s.customer_id = mem.customer_id  
68     INNER JOIN menu as m on s.product_id = m.product_id  
69     WHERE order_date >= join_date  
70 )  
71 SELECT customer_id,  
72        product_name  
73 FROM CTE  
74 WHERE rnk = 1;  
75
```

Data Output Messages Notifications

	customer_id character varying (1) 🔒	product_name character varying (5) 🔒
1	A	curry
2	B	sushi

ANSWER 7

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

```
76 -- 7. Which item was purchased just before the customer became a member?
77 WITH CTE as(
78     SELECT s.customer_id,
79     order_date,
80     product_name,
81     join_date,
82     RANK() OVER(PARTITION BY s.customer_id ORDER BY order_date DESC) as rnk
83 FROM sales as s
84 INNER JOIN members as mem on s.customer_id = mem.customer_id
85 INNER JOIN menu as m on s.product_id = m.product_id
86 WHERE order_date < join_date
87 )
88 SELECT customer_id,
89     product_name
90 FROM CTE
91 WHERE rnk = 1;
```

Data Output Messages Notifications

	customer_id character varying (1) 🔒	product_name character varying (5) 🔒
1	A	sushi
2	A	curry
3	B	sushi

ANSWER 8

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

```
92
93 -- 8. What is the total items and amount spent for each member before they became a member?
94 ✓ SELECT s.customer_id,
95        COUNT(product_name) as Total_items,
96        SUM(price) as Total_amount
97        FROM sales as s
98        INNER JOIN members as mem on s.customer_id = mem.customer_id
99        INNER JOIN menu as m on s.product_id = m.product_id
100        WHERE order_date < join_date
101        GROUP BY s.customer_id
102        ORDER BY s.customer_id
```

Data Output Messages Notifications



	customer_id character varying (1) 🔒	total_items bigint 🔒	total_amount bigint 🔒
1	A	2	25
2	B	3	40

ANSWER 9

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

```
106 SELECT customer_id,  
107 SUM(CASE WHEN product_name = 'sushi' THEN price * 10 * 2 ELSE price * 10 END) as points  
108 FROM menu as m  
109 INNER JOIN sales as s on s.product_id = m.product_id  
110 GROUP BY customer_id  
111 ORDER BY customer_id;
```

Data Output Messages Notifications

	customer_id character varying (1) 🔒	points bigint 🔒
1	A	860
2	B	940
3	C	360

ANSWER 10

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?

```
115 SELECT s.customer_id,  
116 SUM(CASE  
117     WHEN order_date BETWEEN mem.join_date AND (SELECT mem.join_date + INT '6') THEN price * 10 * 2  
118     WHEN product_name = 'sushi' THEN price * 10 * 2 ELSE price * 10 END) AS points  
119 FROM menu AS m  
120 INNER JOIN sales AS s ON s.product_id = m.product_id  
121 INNER JOIN members AS mem ON s.customer_id = mem.customer_id  
122 WHERE DATE_TRUNC('month', order_date) = '2021-01-01'  
123 GROUP BY s.customer_id  
124 ORDER BY s.customer_id;
```

Data Output Messages Notifications



	customer_id character varying (1)	points bigint
1	A	1370
2	B	820

BONUS QUESTIONS

Join All The Things

```
SELECT s.customer_id,  
order_date,  
product_name,  
price,  
CASE  
    WHEN join_date is null THEN 'N'  
    WHEN order_date < join_date THEN 'N' ELSE 'Y'  
    END as member  
FROM sales as s  
INNER JOIN menu m on s.product_id = m.product_id  
LEFT JOIN members mem on mem.customer_id = s.customer_id  
ORDER BY s.customer_id, order_date, price DESC
```

	customer_id character varying (1) 🔒	order_date date 🔒	product_name character varying (5) 🔒	price integer 🔒	member text 🔒
1	A	2021-01-01	curry	15	N
2	A	2021-01-01	sushi	10	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

BONUS QUESTIONS

Rank All The Things

```
WITH CTE as(  
  SELECT  
    s.customer_id,  
    order_date,  
    product_name,  
    price,  
    CASE  
      WHEN join_date is NULL THEN 'N'  
      WHEN order_date < join_date THEN 'N' ELSE 'Y'  
    END as member  
  FROM sales as s  
  INNER JOIN menu m on s.product_id = m.product_id  
  LEFT JOIN members mem on mem.customer_id = s.customer_id  
  ORDER BY s.customer_id, order_date, price DESC  
)  
SELECT *,  
CASE  
  WHEN member = 'N' THEN NULL  
  ELSE RANK() OVER(PARTITION BY customer_id, member ORDER by order_date)  
  END as ranking  
FROM CTE;
```

customer_id character varying (1)	order_date date	product_name character varying (5)	price integer	member text	ranking bigint
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]