CASE STUDY DANNY'S DINNER

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 shared with you 3 key datasets for this case study:

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

Entity Relation Diagram

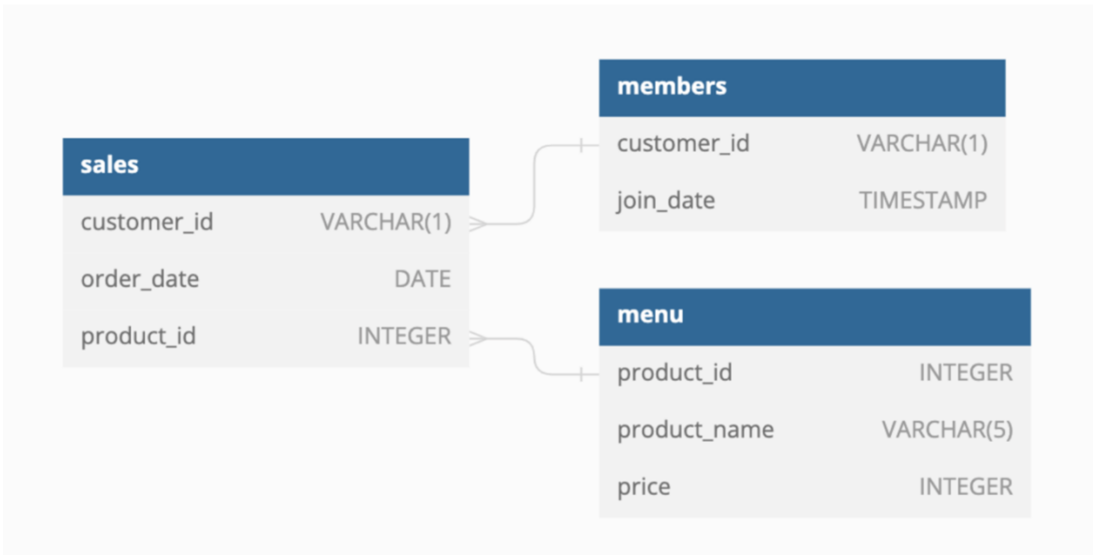
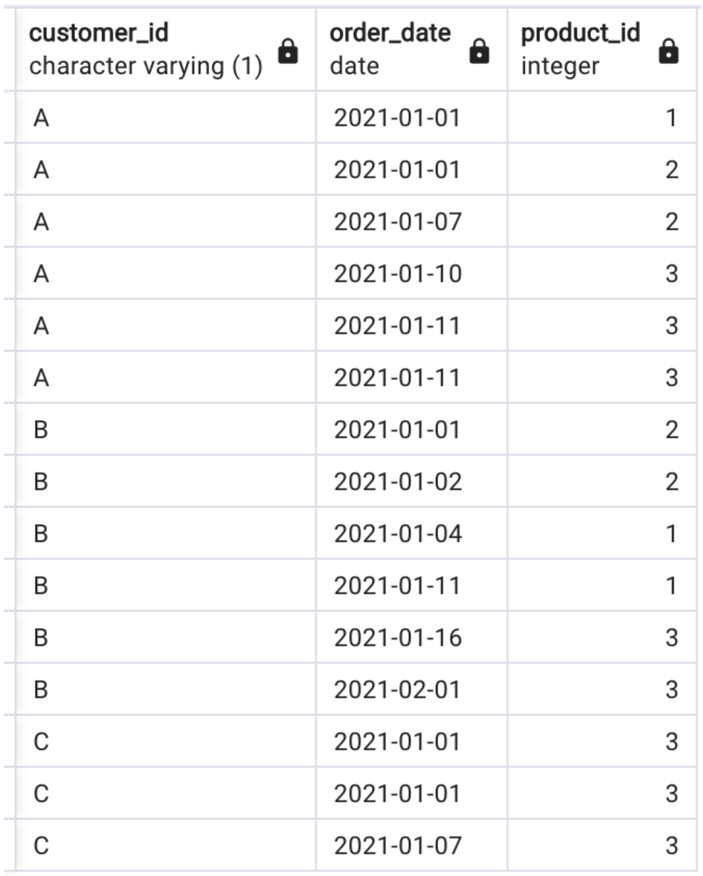


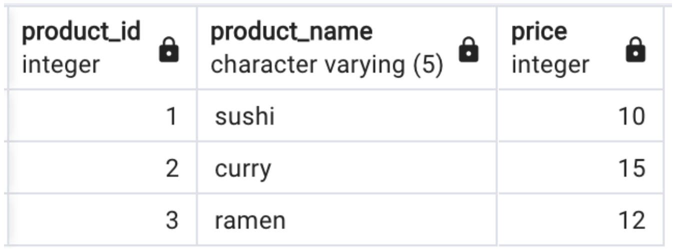
Table 1: Sales
The `sales` table captures all `customer_id` level purchases with an corresponding `order_date` and `product_id` information for when and what menu items were ordered.



customer_id character varying (1)	order_date date	product_id integer
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

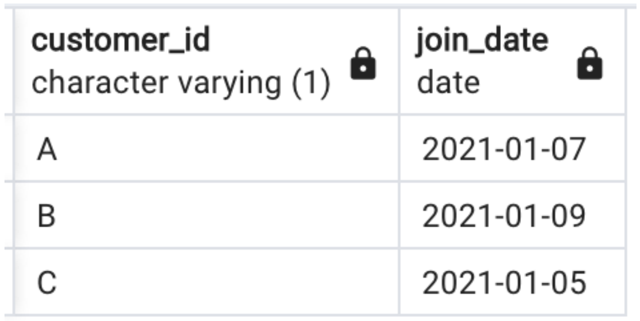
The `menu` table maps the `product_id` to the actual `product_name` and `price` of each menu item.



product_id integer	product_name character varying (5)	price integer
1	sushi	10
2	curry	15
3	ramen	12

Table 3: Members

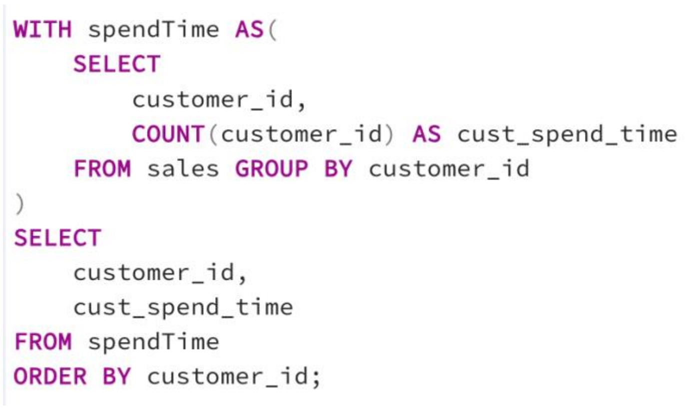
The final `members` table captures the `join_date` when a `customer_id` joined the beta version of the Danny's Diner loyalty program.



customer_id character varying (1)	join_date date
A	2021-01-07
B	2021-01-09
C	2021-01-05

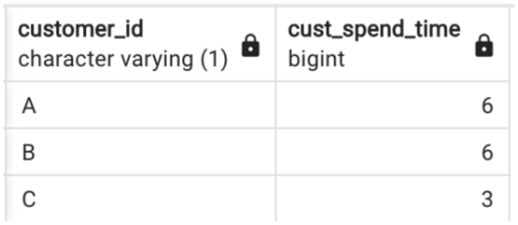
Case Study Questions

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



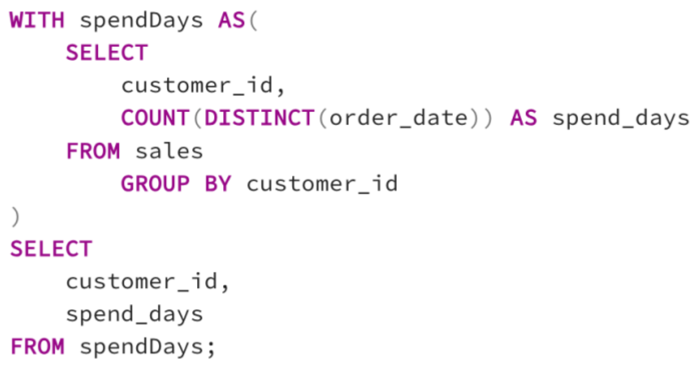
```
WITH spendTime AS(
    SELECT
        customer_id,
        COUNT(customer_id) AS cust_spend_time
    FROM sales GROUP BY customer_id
)
SELECT
    customer_id,
    cust_spend_time
FROM spendTime
ORDER BY customer_id;
```

   Result:



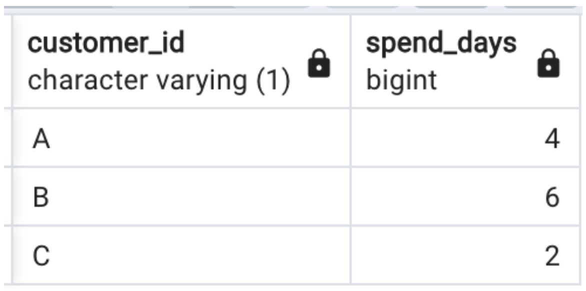
customer_id character varying (1)	cust_spend_time bigint
A	6
B	6
C	3

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



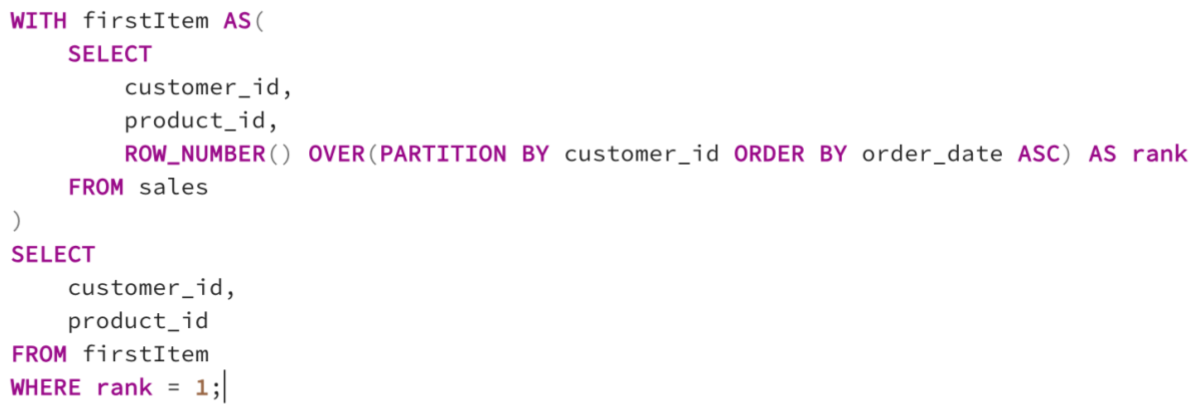
```
WITH spendDays AS(
    SELECT
        customer_id,
        COUNT(DISTINCT(order_date)) AS spend_days
    FROM sales
        GROUP BY customer_id
)
SELECT
    customer_id,
    spend_days
FROM spendDays;
```

   Result:

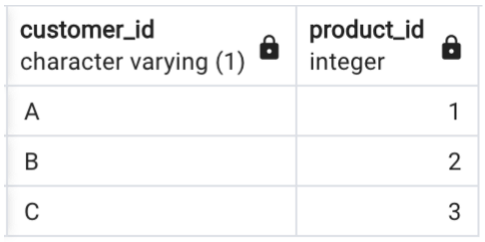


customer_id character varying (1)	spend_days bigint
A	4
B	6
C	2

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

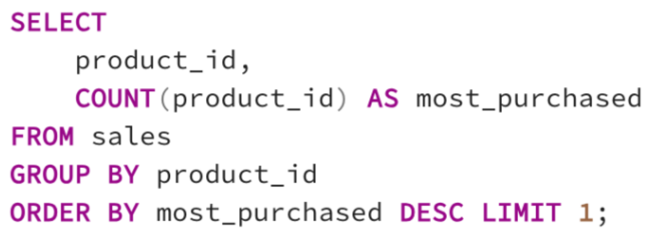

```
WITH firstItem AS(
    SELECT
        customer_id,
        product_id,
        ROW_NUMBER() OVER(PARTITION BY customer_id ORDER BY order_date ASC) AS rank
    FROM sales
)
SELECT
    customer_id,
    product_id
FROM firstItem
WHERE rank = 1;
```

Result:

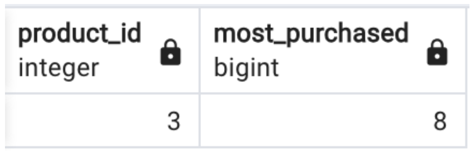


customer_id character varying (1)	product_id integer
A	1
B	2
C	3

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


```
SELECT
    product_id,
    COUNT(product_id) AS most_purchased
FROM sales
GROUP BY product_id
ORDER BY most_purchased DESC LIMIT 1;
```

Result:

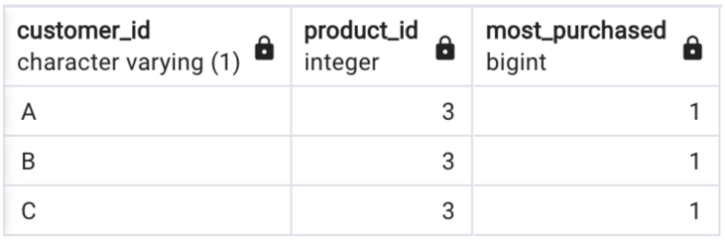


product_id integer	most_purchased bigint
3	8

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

```
WITH rankedProducts AS(
    SELECT
        customer_id,
        product_id,
        ROW_NUMBER() OVER(PARTITION BY customer_id ORDER BY COUNT(*) DESC) AS rank
    FROM sales
    GROUP BY customer_id, product_id
)
SELECT
    customer_id,
    product_id,
    count(*) as most_purchased
FROM rankedProducts
WHERE rank = 1
GROUP BY customer_id, product_id;
```

Result:



customer_id character varying (1)	product_id integer	most_purchased bigint
A	3	1
B	3	1
C	3	1

6. Which item and how many items was purchased first by the customer after they became a member?
Query:
Show All

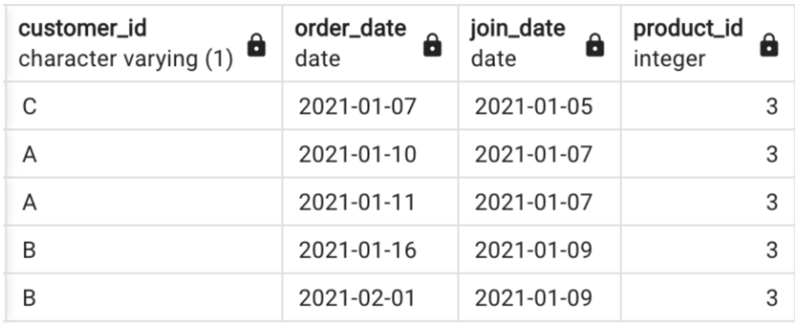
```
WITH firstPurchased AS(
    SELECT
        sales.customer_id,
        sales.order_date,
        members.join_date,
        sales.product_id,
        ROW_NUMBER() OVER (PARTITION BY sales.order_date ORDER BY members.join_date)AS rnk
    FROM sales
    INNER JOIN members
    ON sales.customer_id = members.customer_id
    WHERE sales.order_date > members.join_date
)
SELECT
    customer_id, order_date, join_date, product_id
FROM firstPurchased WHERE rnk=1;
```

Counted

```sql
WITH afterPurchase AS(
    SELECT
        sales.customer_id,
        sales.order_date,
        members.join_date,
        sales.product_id
    FROM
        sales INNER JOIN members
        ON sales.customer_id = members.customer_id
        LEFT JOIN menu ON sales.product_id = menu.product_id
    WHERE order_date > join_date ORDER BY customer_id
)
SELECT
    customer_id,
    COUNT (*) AS total
FROM afterPurchase GROUP BY customer_id ORDER BY customer_id;
```

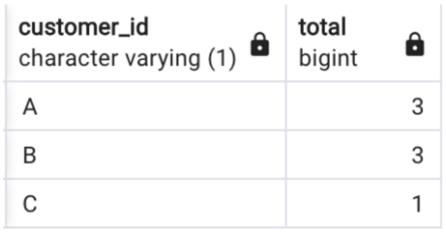
Result:

Show All



customer_id character varying (1)	order_date date	join_date date	product_id integer
C	2021-01-07	2021-01-05	3
A	2021-01-10	2021-01-07	3
A	2021-01-11	2021-01-07	3
B	2021-01-16	2021-01-09	3
B	2021-02-01	2021-01-09	3

Counted

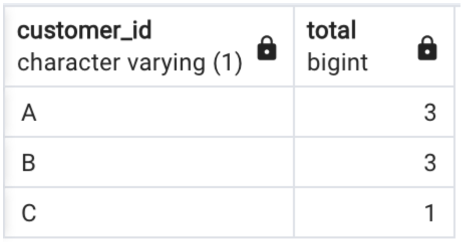


customer_id character varying (1)	total bigint
A	3
B	3
C	1

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

```sql
WITH afterPurchase AS(
    SELECT
        sales.customer_id,
        sales.order_date,
        members.join_date,
        sales.product_id
    FROM
        sales INNER JOIN members
        ON sales.customer_id = members.customer_id
        LEFT JOIN menu ON sales.product_id = menu.product_id
    WHERE order_date < join_date ORDER BY customer_id
)
SELECT
    customer_id,
    COUNT (*) AS total
FROM afterPurchase
GROUP BY customer_id
ORDER BY customer_id;
```

Result:

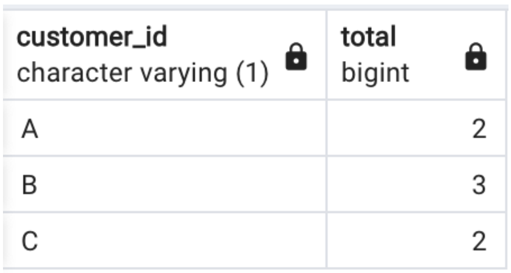


customer_id character varying (1)	total bigint
A	3
B	3
C	1

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

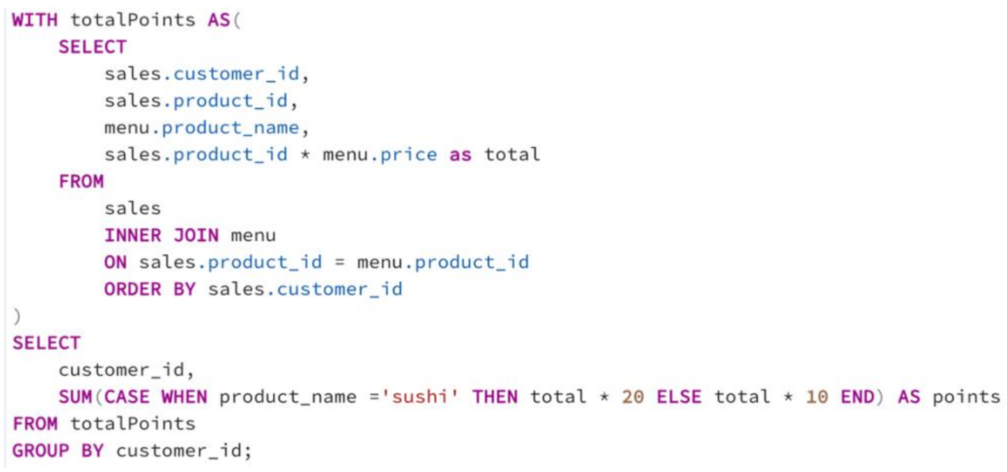
```
WITH beforePurchase AS(
    SELECT
        sales.customer_id,
        sales.order_date,
        members.join_date,
        sales.product_id
    FROM
        sales INNER JOIN members
        ON sales.customer_id = members.customer_id
        LEFT JOIN menu ON sales.product_id = menu.product_id
    WHERE order_date < join_date ORDER BY customer_id
)
SELECT
    customer_id,
    COUNT (*) AS total
FROM beforePurchase GROUP BY customer_id ORDER BY customer_id;
```

   Result:



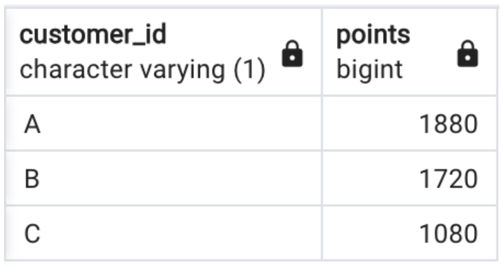
customer_id character varying (1)	total bigint
A	2
B	3
C	2

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



```sql
WITH totalPoints AS(
    SELECT
        sales.customer_id,
        sales.product_id,
        menu.product_name,
        sales.product_id * menu.price as total
    FROM
        sales
        INNER JOIN menu
        ON sales.product_id = menu.product_id
        ORDER BY sales.customer_id
)
SELECT
    customer_id,
    SUM(CASE WHEN product_name ='sushi' THEN total * 20 ELSE total * 10 END) AS points
FROM totalPoints
GROUP BY customer_id;
```

Result:



customer_id character varying (1)	points bigint
A	1880
B	1720
C	1080