Challenge-1

DANNY'S DINNER

By: Harshita Aswani

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 needs 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 it run the business.

Problem Statement

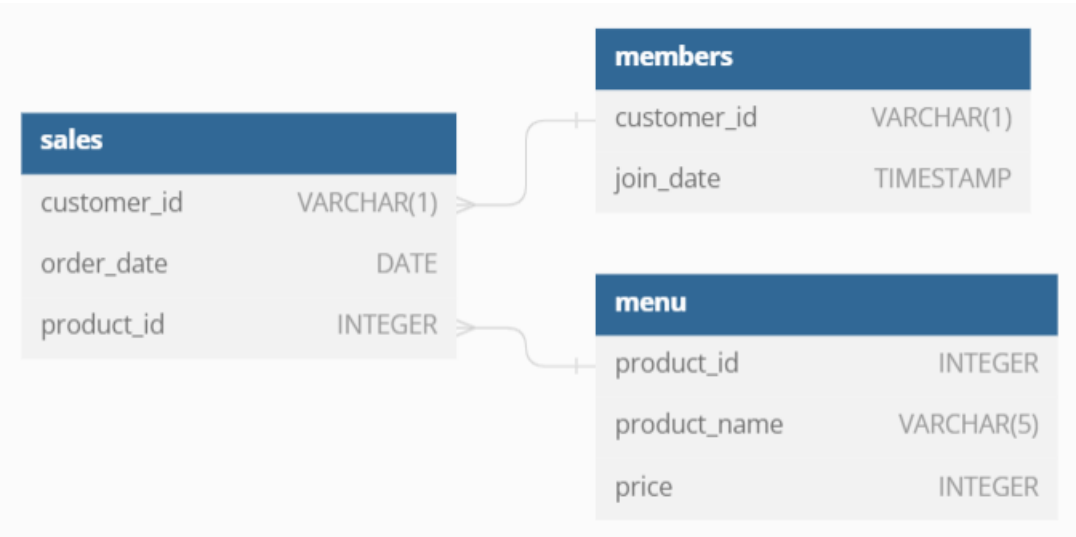
Danny wants to use the data to answer a few simple questions about his customers, especially about their visiting patterns, how much money they've spent and also which menu items are their favorite. Having this deeper connection with his customers will help him deliver a better and more personalized 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 me with a sample of his overall customer data due to privacy issues - but he hopes that these examples are enough for me to write fully functioning SQL queries to help him answer his questions!

Entity Relationship Diagram

Danny has shared with me 3 key datasets for this case study whose entity relationship diagram is given below:



Tables

The sales table captures all customer_id level purchases with a 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

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

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

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

a. Query SQL

```sql
SELECT
    m.customer_id,
    SUM(me.price) AS total_amount_spent
FROM
    members m
JOIN
    sales s ON m.customer_id = s.customer_id
JOIN
    menu me ON s.product_id = me.product_id
GROUP BY
    m.customer_id;
```

b.

customer_id	total_amount_spent
A	76
B	74

II. How many days have each customer visited the restaurant?

a. Query SQL

```sql
SELECT
    customer_id,
    COUNT(DISTINCT order_date) AS total_days_visited
FROM
    sales
GROUP BY
    customer_id;
```

b.

customer_id	total_days_visited
A	4
B	6
C	2

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

a. Query SQL

```sql
WITH first_purchase AS (
    SELECT
        customer_id,
        product_name,
        ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY order_date) AS row_num
    FROM
        sales s
    JOIN
        menu me ON s.product_id = me.product_id
)
SELECT
    customer_id,
    product_name AS first_purchased_item
FROM
    first_purchase
WHERE
    row_num = 1;
```

b.

customer_id	first_purchased_item
A	curry
B	curry
C	ramen

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

Query SQL

a.

```sql
SELECT
    me.product_name,
    COUNT(s.product_id) AS total_purchases
FROM
    menu me
JOIN
    sales s ON me.product_id = s.product_id
GROUP BY
    me.product_name
ORDER BY
    total_purchases DESC
LIMIT 1;
```

b.

product_name	total_purchases
ramen	8

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

Query SQL

a.

```sql
WITH popular_items AS (
    SELECT
        customer_id,
        product_name,
        COUNT(menu.product_id) AS total_purchases
    FROM
        menu
    JOIN
        sales ON menu.product_id = sales.product_id
    GROUP BY
        customer_id, product_name
)
SELECT
    customer_id,
    product_name AS most_popular_item
FROM (
    SELECT
        customer_id,
        product_name,
        ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY total_purchases DESC) AS row_num
    FROM
        popular_items
) ranked_items
WHERE
    row_num = 1;
```

b.

customer_id	most_popular_item
A	ramen
B	ramen
C	ramen

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

Query SQL

```sql
WITH first_purchase_after_membership AS (
    SELECT
        s.customer_id,
        me.product_name,
        s.order_date
    FROM
        sales s
    JOIN
        menu me ON s.product_id = me.product_id
    JOIN
        members m ON s.customer_id = m.customer_id
    WHERE
        s.order_date >= m.join_date
    ORDER BY
        s.order_date
    LIMIT 1
)
SELECT
    customer_id,
    product_name AS first_purchase_after_membership
FROM
    first_purchase_after_membership;
```

a.

b.

customer_id	first_purchase_after_membership
A	curry

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

Query SQL

a.

```sql
WITH last_purchase_before_membership AS (
    SELECT
        s.customer_id,
        me.product_name,
        s.order_date
    FROM
        sales s
    JOIN
        menu me ON s.product_id = me.product_id
    JOIN
        members m ON s.customer_id = m.customer_id
    WHERE
        s.order_date < m.join_date
    ORDER BY
        s.order_date DESC
    LIMIT 1
)
SELECT
    customer_id,
    product_name AS last_purchase_before_membership
FROM
    last_purchase_before_membership;
```

b.

customer_id	last_purchase_before_membership
B	sushi

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

Query SQL

a.

```sql
WITH member_stats AS (
    SELECT
        m.customer_id,
        COUNT(s.product_id) AS total_items_before_membership,
        SUM(me.price) AS total_amount_spent_before_membership
    FROM
        members m
    LEFT JOIN
        sales s ON m.customer_id = s.customer_id
    LEFT JOIN
        menu me ON s.product_id = me.product_id
    WHERE
        s.order_date < m.join_date OR s.order_date IS NULL
    GROUP BY
        m.customer_id
)
SELECT
    m.*,
    COALESCE(total_items_before_membership, 0) AS total_items_before_membership,
    COALESCE(total_amount_spent_before_membership, 0) AS total_amount_spent_before_membership
FROM
    members m
LEFT JOIN
    member_stats ms ON m.customer_id = ms.customer_id;
```

b.

customer_id	join_date	total_items_before_membership	total_amount_spent_before_membership
A	2021-01-07T00:00:00.000Z	2	25
B	2021-01-09T00:00:00.000Z	3	40

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

a. Query SQL

```sql
SELECT
    m.customer_id,
    SUM(CASE WHEN me.product_name = 'sushi' THEN 2 * me.price ELSE me.price END) * 10
AS total_points
FROM
    members m
JOIN
    sales s ON m.customer_id = s.customer_id
JOIN
    menu me ON s.product_id = me.product_id
GROUP BY
    m.customer_id;
```

b.

customer_id	total_points
A	860
B	940

X. 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?

a. Query SQL

```sql
SELECT
    m.customer_id,
    SUM(CASE WHEN s.order_date >= m.join_date AND s.order_date < m.join_date + INTERVAL '7 days' THEN 2 * me.price ELSE me.price END) * 10 AS total_points
FROM
    members m
JOIN
    sales s ON m.customer_id = s.customer_id
JOIN
    menu me ON s.product_id = me.product_id
WHERE
    (m.customer_id = 'A' OR m.customer_id = 'B') AND EXTRACT(MONTH FROM s.order_date) = 1
GROUP BY
    m.customer_id;
```

b.

customer_id	total_points
A	1270
B	720