8WEEKSQLCHALLENGE.COM

CASE STUDY #1

THE TASTE OF SUCCESS

DATAWITHDANNY.COM

Tung Ho

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.

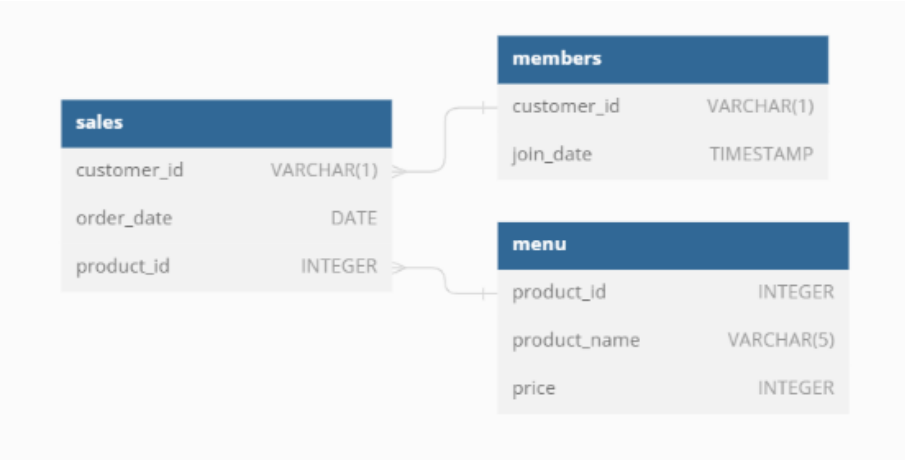
Danny has provided you with a sample of his overall customer data due to privacy issues - but he hopes that these examples are enough for you to write fully functioning SQL queries to help him answer his questions!

Danny has shared with you 3 key datasets for this case study:

- sales
- menu
- members

You can inspect the entity relationship diagram and example data below.

Entity Relationship Diagram



Example Datasets

All datasets exist within the dannys_diner database schema - be sure to include this reference within your SQL scripts as you start exploring the data and answering the case study questions.

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

```sql
SELECT
  S.CUSTOMER_ID,
  SUM(M.PRICE) AS TOTAL_SPEND
FROM DANNYS_DINER.SALES S
JOIN DANNYS_DINER.MENU M
ON S.PRODUCT_ID=M.PRODUCT_ID
GROUP BY S.CUSTOMER_ID
ORDER BY TOTAL_SPEND DESC;
```

customer_id	total_spend
A	76
B	74
C	36

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

```sql
SELECT
  CUSTOMER_ID,
  COUNT(DISTINCT ORDER_DATE) AS TOTAL_VISITED_DAYS
FROM DANNYS_DINER.SALES
GROUP BY CUSTOMER_ID
ORDER BY TOTAL_VISITED_DAYS DESC, CUSTOMER_ID;
```

customer_id	total_visited_days
B	6
A	4
C	2

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

```sql
SELECT
    S.CUSTOMER_ID,
    M.PRODUCT_NAME AS FIRST_ITEM
FROM (SELECT
        CUSTOMER_ID,
        MIN(ORDER_DATE) AS ORDER_DATE
      FROM DANNYS_DINER.SALES
      GROUP BY CUSTOMER_ID) S1
JOIN DANNYS_DINER.SALES S
ON S.CUSTOMER_ID=S1.CUSTOMER_ID AND S.ORDER_DATE=S1.ORDER_DATE
JOIN DANNYS_DINER.MENU M
ON S.PRODUCT_ID=M.PRODUCT_ID
GROUP BY S.CUSTOMER_ID, FIRST_ITEM
ORDER BY S.CUSTOMER_ID;
```

customer_id	first_item
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?

```sql
SELECT
    M.PRODUCT_NAME,
    COUNT(M.PRODUCT_NAME) AS TOTAL_PURCHASED
FROM DANNYS_DINER.SALES S
JOIN DANNYS_DINER.MENU M
ON S.PRODUCT_ID=M.PRODUCT_ID
GROUP BY PRODUCT_NAME
ORDER BY TOTAL_PURCHASED DESC LIMIT 1;
```

product_name	total_purchased
ramen	8

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

```sql
WITH CTE AS (SELECT
                    CUSTOMER_ID,
                    PRODUCT_ID,
                    COUNT(PRODUCT_ID) AS MOST_POP,
                    DENSE_RANK() OVER(PARTITION BY CUSTOMER_ID
                                      ORDER BY COUNT(PRODUCT_ID) DESC) AS RANK
                FROM DANNYS_DINER.SALES
                GROUP BY CUSTOMER_ID, PRODUCT_ID
                ORDER BY CUSTOMER_ID, MOST_POP DESC)
SELECT
    S.CUSTOMER_ID,
    M.PRODUCT_NAME,
    S.MOST_POP
FROM CTE S
JOIN DANNYS_DINER.MENU M
ON S.PRODUCT_ID=M.PRODUCT_ID
WHERE S.RANK=1
ORDER BY S.CUSTOMER_ID;
```

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

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

```sql
WITH CTE AS (SELECT
                  S.CUSTOMER_ID,
                  S.ORDER_DATE,
                  M.PRODUCT_NAME,
                  DENSE_RANK() OVER(PARTITION BY S.CUSTOMER_ID
                                    ORDER BY S.ORDER_DATE ASC) AS FIRST_RANK
      FROM DANNYS_DINER.SALES S
      JOIN DANNYS_DINER.MENU M
      ON S.PRODUCT_ID=M.PRODUCT_ID
      INNER JOIN DANNYS_DINER.MEMBERS MEM
      ON S.CUSTOMER_ID=MEM.CUSTOMER_ID
      WHERE S.ORDER_DATE>=MEM.JOIN_DATE)
SELECT
    CTE.CUSTOMER_ID,
    CTE.PRODUCT_NAME
FROM CTE
WHERE CTE.FIRST_RANK=1;
```

customer_id	product_name
A	curry
B	sushi

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

```sql
WITH CTE AS (SELECT
                S.CUSTOMER_ID,
                S.ORDER_DATE,
                M.PRODUCT_NAME,
                DENSE_RANK() OVER(PARTITION BY S.CUSTOMER_ID
                                  ORDER BY S.ORDER_DATE DESC) AS LAST_RANK
          FROM DANNYS_DINER.SALES S
          JOIN DANNYS_DINER.MENU M
          ON S.PRODUCT_ID=M.PRODUCT_ID
          INNER JOIN DANNYS_DINER.MEMBERS MEM
          ON S.CUSTOMER_ID=MEM.CUSTOMER_ID
          WHERE S.ORDER_DATE<MEM.JOIN_DATE)
SELECT
    CTE.CUSTOMER_ID,
    CTE.PRODUCT_NAME
FROM CTE
WHERE CTE.LAST_RANK=1;
```

customer_id	product_name
A	sushi
A	curry
B	sushi

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

```sql
SELECT
    S.CUSTOMER_ID,
    COUNT(M.PRODUCT_NAME) AS TOTAL_ITEMS,
    SUM(M.PRICE) AS TOTAL_SPEND
FROM DANNYS_DINER.SALES S
JOIN DANNYS_DINER.MENU M
ON S.PRODUCT_ID=M.PRODUCT_ID
INNER JOIN DANNYS_DINER.MEMBERS MEM
ON S.CUSTOMER_ID=MEM.CUSTOMER_ID
WHERE S.ORDER_DATE<MEM.JOIN_DATE
GROUP BY S.CUSTOMER_ID
ORDER BY S.CUSTOMER_ID;
```

customer_id	total_items	total_spend
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?

```sql
--CUST_SPEND_MULTIPLIER CTE
WITH CSM AS (SELECT
    S.CUSTOMER_ID,
    S.ORDER_DATE,
    M.PRODUCT_NAME,
    M.PRICE AS TOTAL_SPEND,
    CASE
        WHEN M.PRODUCT_NAME='sushi' THEN 20
    ELSE 10
    END AS MULTIPLIER
FROM DANNYS_DINER.SALES S
JOIN DANNYS_DINER.MENU M
ON S.PRODUCT_ID=M.PRODUCT_ID)
SELECT
    CSM.CUSTOMER_ID,
    SUM(CSM.TOTAL_SPEND * CSM.MULTIPLIER) AS POINTS
FROM CSM
GROUP BY CSM.CUSTOMER_ID
ORDER BY CSM.CUSTOMER_ID;
```

customer_id	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?

```sql
--CUST_SPEND_MULTIPLIER CTE
WITH CSM AS (SELECT
    S.CUSTOMER_ID,
    S.ORDER_DATE,
    M.PRODUCT_NAME,
    M.PRICE AS TOTAL_SPEND,
    CASE
        WHEN M.PRODUCT_NAME='sushi' THEN 20
        WHEN S.ORDER_DATE BETWEEN MEM.JOIN_DATE AND MEM.JOIN_DATE+6 THEN 20
    ELSE 10
    END AS MULTIPLIER
FROM DANNYS_DINER.SALES S
JOIN DANNYS_DINER.MENU M
ON S.PRODUCT_ID=M.PRODUCT_ID
INNER JOIN DANNYS_DINER.MEMBERS MEM
ON S.CUSTOMER_ID=MEM.CUSTOMER_ID
WHERE DATE_PART('Month', S.ORDER_DATE::DATE)=1)
SELECT
    CSM.CUSTOMER_ID,
    SUM(CSM.TOTAL_SPEND * CSM.MULTIPLIER) AS POINTS
FROM CSM
GROUP BY CSM.CUSTOMER_ID
ORDER BY CSM.CUSTOMER_ID;
```

customer_id	points
A	1370
B	820

BONUS QUESTIONS

Recreate the table output using the available data:

```sql
SELECT
    S.CUSTOMER_ID,
    TO_CHAR(S.ORDER_DATE :: DATE, 'YYYY-MM-DD'),
    M.PRODUCT_NAME,
    M.PRICE,
    CASE
        WHEN MEM.JOIN_DATE IS NULL THEN 'N'
        WHEN S.ORDER_DATE<MEM.JOIN_DATE THEN 'N'
        ELSE 'Y'
        END AS MEMBER
FROM DANNYS_DINER.SALES S
JOIN DANNYS_DINER.MENU M
ON S.PRODUCT_ID=M.PRODUCT_ID
LEFT JOIN DANNYS_DINER.MEMBERS MEM
ON S.CUSTOMER_ID=MEM.CUSTOMER_ID
ORDER BY S.CUSTOMER_ID, S.ORDER_DATE;
```

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

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.

```sql
--RANK TABLE
WITH RT AS (SELECT
                S.CUSTOMER_ID AS CUSTOMER_ID,
                S.ORDER_DATE AS ORDER_DATE,
                M.PRODUCT_NAME AS PRODUCT_NAME,
                DENSE_RANK() OVER(PARTITION BY S.CUSTOMER_ID
                                  ORDER BY S.ORDER_DATE) AS RANKING
            FROM DANNYS_DINER.SALES S
            JOIN DANNYS_DINER.MENU M
            ON S.PRODUCT_ID=M.PRODUCT_ID
            JOIN DANNYS_DINER.MEMBERS MEM
            ON S.CUSTOMER_ID=MEM.CUSTOMER_ID
            WHERE S.ORDER_DATE≥MEM.JOIN_DATE
            GROUP BY S.CUSTOMER_ID, ORDER_DATE, M.PRODUCT_NAME
            ORDER BY S.CUSTOMER_ID, S.ORDER_DATE)
SELECT
    S.CUSTOMER_ID,
    TO_CHAR(S.ORDER_DATE :: DATE, 'YYYY-MM-DD'),
    M.PRODUCT_NAME,
    M.PRICE,
    CASE
        WHEN MEM.JOIN_DATE IS NULL THEN 'N'
        WHEN S.ORDER_DATE<MEM.JOIN_DATE THEN 'N'
        ELSE 'Y'
        END AS MEMBER,
    RT.RANKING
FROM DANNYS_DINER.SALES S
JOIN DANNYS_DINER.MENU M
ON S.PRODUCT_ID=M.PRODUCT_ID
LEFT JOIN DANNYS_DINER.MEMBERS MEM
ON S.CUSTOMER_ID=MEM.CUSTOMER_ID
LEFT JOIN RT
ON S.CUSTOMER_ID=RT.CUSTOMER_ID AND S.ORDER_DATE=RT.ORDER_DATE AND M.PRODUCT_NAME=RT.PRODUCT_NAME
ORDER BY S.CUSTOMER_ID, S.ORDER_DATE;
```

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