Case Study #1 - Danny's Diner

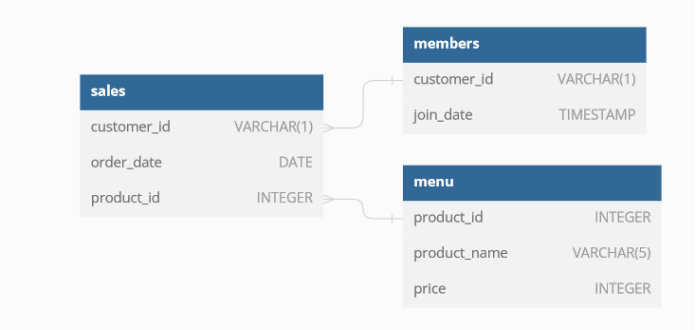
The case study is sourced from the 8-week SQL challenge by Danny Ma - https://8weeksqlchallenge.com/

Introduction: Danny seriously loves Japanese food so in the beginning of 2021, he decides to embark upon a risky venture and opens 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 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 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 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 and members.**

Entity Relationship Diagram



I am using MySQL in DB Fiddle (SQL editor) to answer the below questions.

Questions and Answers:

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

```
SELECT s.customer_id,SUM(b.price) AS total_amount
FROM dannys_diner.sales s JOIN dannys_diner.menu b
ON s.product_id = b.product_id
GROUP BY s.customer_id
ORDER BY s.customer_id
```

Output:

Query #1 Execution time: 2ms

customer_id	total_amount
A	76
B	74
C	36

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

```
SELECT customer_id, COUNT(DISTINCT order_date) AS total_visits
FROM dannys_diner.sales
GROUP BY customer_id
```

Output:

Query #1 Execution time: 2ms

customer_id	total_visits
A	4
B	6
C	2

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

```
WITH ranking AS
(
SELECT customer_id, order_date,product_id,
DENSE_RANK() OVER(PARTITION BY customer_id ORDER BY order_date) AS row_num
FROM dannys_diner.sales
)

SELECT s.customer_id, m.product_name
FROM ranking s JOIN dannys_diner.menu m
ON s.product_id=m.product_id
WHERE row_num =1
ORDER BY s.customer_id
```

Output: Since there is no timestamp mentioned, there is no way to decide what customer A ordered first.

Query #1 Execution time: 2ms

customer_id	product_name
A	sushi
A	curry
B	curry
C	ramen

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

```
WITH most_purchased AS
(
SELECT product_id, COUNT(product_id) AS number_of_times
FROM dannys_diner.sales
GROUP BY product_id
ORDER BY product_id DESC LIMIT 1
)

SELECT s.customer_id, COUNT(m.product_id) AS total_count
FROM most_purchased m JOIN dannys_diner.sales s
ON m.product_id = s.product_id
GROUP BY s.customer_id
```

Most purchased item on the menu is 8

Query #1 Execution time: 1ms

product_id	number_of_times
3	8

Number of times it was purchased by each customer.

Query #1 Execution time: 2ms

customer_id	total_count
A	3
B	2
C	3

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

```
WITH ranking AS
(
SELECT customer_id,product_id, COUNT(product_id) AS total_count
FROM dannys_diner.sales
GROUP BY customer_id,product_id
ORDER BY customer_id,product_id
)

SELECT customer_id,product_id,total_count FROM
(
SELECT customer_id,product_id,total_count,
DENSE_RANK() OVER(PARTITION BY customer_id ORDER BY total_count DESC) AS highest_rank
FROM ranking
)sub
WHERE highest_rank=1
```

Output:

Query #1 Execution time: 1ms

customer_id	product_id	total_count
A	3	3
B	1	2
B	2	2
B	3	2
C	3	3

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

```
WITH members_joindate AS
(
SELECT s.customer_id,s.order_date,s.product_id,m.join_date,d.product_name
FROM dannys_diner.sales s JOIN
dannys_diner.members m ON
s.customer_id = m.customer_id
AND s.order_date >= m.join_date
JOIN  dannys_diner.menu d on
 s.product_id = d.product_id
)

SELECT customer_id,product_name FROM
(
SELECT customer_id,order_date,product_id,join_date,product_name,
DENSE_RANK() OVER(PARTITION BY customer_id ORDER BY order_date) AS ranking
FROM members_joindate
)sub
WHERE ranking = 1
```

Output:

Query #1 Execution time: 2ms

customer_id	product_name
A	curry
B	sushi

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

```
WITH members_joindate AS
(
SELECT s.customer_id,s.order_date,s.product_id,m.join_date,d.product_name
FROM dannys_diner.sales s JOIN
dannys_diner.members m ON
s.customer_id = m.customer_id
AND s.order_date < m.join_date
JOIN  dannys_diner.menu d on
 s.product_id = d.product_id
)

SELECT customer_id,product_name FROM
(
SELECT customer_id,order_date,product_id,join_date,product_name,
DENSE_RANK() OVER(PARTITION BY customer_id ORDER BY order_date DESC) AS ranking
FROM members_joindate
)sub
WHERE ranking = 1
```

Output:

Query #1 Execution time: 1ms

customer_id	product_name
A	sushi
A	curry
B	sushi

Customer A has two orders from the same date just before he/she became a member. Without a timestamp we cannot determine which one was placed first/last.

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 num_of_items, SUM(d.price) AS total_amount
FROM dannys_diner.sales s JOIN
dannys_diner.members m ON
s.customer_id = m.customer_id
AND s.order_date < m.join_date
JOIN  dannys_diner.menu d on
s.product_id = d.product_id
GROUP BY s.customer_id
ORDER BY s.customer_id
```

Output:

Query #1 Execution time: 3ms

customer_id	num_of_items	total_amount
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 points AS
(
SELECT product_id,product_name,price,
CASE WHEN product_name = 'sushi' THEN price*20
WHEN product_name = 'curry' THEN price*10
WHEN product_name = 'ramen' THEN 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 points p
ON s.product_id = p.product_id
GROUP BY s.customer_id
ORDER BY s.customer_id
```

Output:

Query #1 Execution time: 2ms

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?

```sql
WITH members AS
(
SELECT s.customer_id,s.order_date,s.product_id,
m.product_name,m.price,d.join_date
FROM dannys_diner.sales s JOIN dannys_diner.menu m
ON s.product_id = m.product_id
JOIN dannys_diner.members d
ON s.customer_id = d.customer_id
)
,
points AS
(
SELECT customer_id,order_date,product_id,product_name,join_date,
CASE WHEN product_name = 'sushi' THEN price*20
WHEN order_date BETWEEN join_date AND join_date +6 THEN price*20 ELSE price*10
END AS points
FROM members
ORDER BY customer_id,order_date
)

SELECT customer_id,SUM(points) AS total_points
FROM points
WHERE order_date BETWEEN '2021-01-01' AND '2021-01-31'
GROUP BY customer_id
ORDER BY customer_id
```

Output:

Query #1 Execution time: 3ms

customer_id	total_points
A	1370
B	820

BONUS QUESTION:

1)Join All The Things

The following questions are related to creating basic data tables that Danny and his team can use to quickly derive insights without needing to join the underlying tables using SQL.

Recreate the following table output using the available data:

```sql
WITH cte AS
(
SELECT a.customer_id,a.order_date,b.product_name,b.price,c.join_date
FROM dannys_diner.sales a JOIN dannys_diner.menu b
ON a.product_id = b.product_id
LEFT JOIN dannys_diner.members c
ON a.customer_id=c.customer_id
ORDER BY a.customer_id,a.order_date
)

SELECT customer_id,order_date,product_name,price,
CASE WHEN order_date >= join_date THEN 'Y' ELSE 'N' END AS member
FROM cte
```

Output:

customer_id	order_date	product_name	price	member
A	2021-01-01T00:00:00.000Z	sushi	10	N
A	2021-01-01T00:00:00.000Z	curry	15	N
A	2021-01-07T00:00:00.000Z	curry	15	Y
A	2021-01-10T00:00:00.000Z	ramen	12	Y
A	2021-01-11T00:00:00.000Z	ramen	12	Y
A	2021-01-11T00:00:00.000Z	ramen	12	Y
B	2021-01-01T00:00:00.000Z	curry	15	N
B	2021-01-02T00:00:00.000Z	curry	15	N
B	2021-01-04T00:00:00.000Z	sushi	10	N
B	2021-01-11T00:00:00.000Z	sushi	10	Y
B	2021-01-16T00:00:00.000Z	ramen	12	Y
B	2021-02-01T00:00:00.000Z	ramen	12	Y
C	2021-01-01T00:00:00.000Z	ramen	12	N
C	2021-01-01T00:00:00.000Z	ramen	12	N
C	2021-01-07T00:00:00.000Z	ramen	12	N

2) Rank All The Things - 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
WITH cte AS
(
SELECT a.customer_id,a.order_date,b.product_name,b.price,c.join_date
FROM dannys_diner.sales a JOIN dannys_diner.menu b
ON a.product_id = b.product_id
LEFT JOIN dannys_diner.members c
ON a.customer_id=c.customer_id
ORDER BY a.customer_id,a.order_date
)

,member AS
(
SELECT *,
CASE WHEN order_date >= join_date THEN 'Y' ELSE 'N' END AS member_status
FROM cte
)

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 member
```

Output:

customer_id	order_date	product_name	price	join_date	member_status	ranking
A	2021-01-01T00:00:00.000Z	sushi	10	2021-01-07T00:00:00.000Z	N	null
A	2021-01-01T00:00:00.000Z	curry	15	2021-01-07T00:00:00.000Z	N	null
A	2021-01-07T00:00:00.000Z	curry	15	2021-01-07T00:00:00.000Z	Y	1
A	2021-01-10T00:00:00.000Z	ramen	12	2021-01-07T00:00:00.000Z	Y	2
A	2021-01-11T00:00:00.000Z	ramen	12	2021-01-07T00:00:00.000Z	Y	3
A	2021-01-11T00:00:00.000Z	ramen	12	2021-01-07T00:00:00.000Z	Y	3
B	2021-01-01T00:00:00.000Z	curry	15	2021-01-09T00:00:00.000Z	N	null
B	2021-01-02T00:00:00.000Z	curry	15	2021-01-09T00:00:00.000Z	N	null
B	2021-01-04T00:00:00.000Z	sushi	10	2021-01-09T00:00:00.000Z	N	null
B	2021-01-11T00:00:00.000Z	sushi	10	2021-01-09T00:00:00.000Z	Y	1
B	2021-01-16T00:00:00.000Z	ramen	12	2021-01-09T00:00:00.000Z	Y	2
B	2021-02-01T00:00:00.000Z	ramen	12	2021-01-09T00:00:00.000Z	Y	3
C	2021-01-01T00:00:00.000Z	ramen	12	null	N	null
C	2021-01-01T00:00:00.000Z	ramen	12	null	N	null
C	2021-01-07T00:00:00.000Z	ramen	12	null	N	null