Case Study #1 - Danny's Diner

8WEEKSQLCHALLENGE.COM

CASE STUDY #1

THE TASTE OF SUCCESS

DATAWITHDANNY.COM

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.

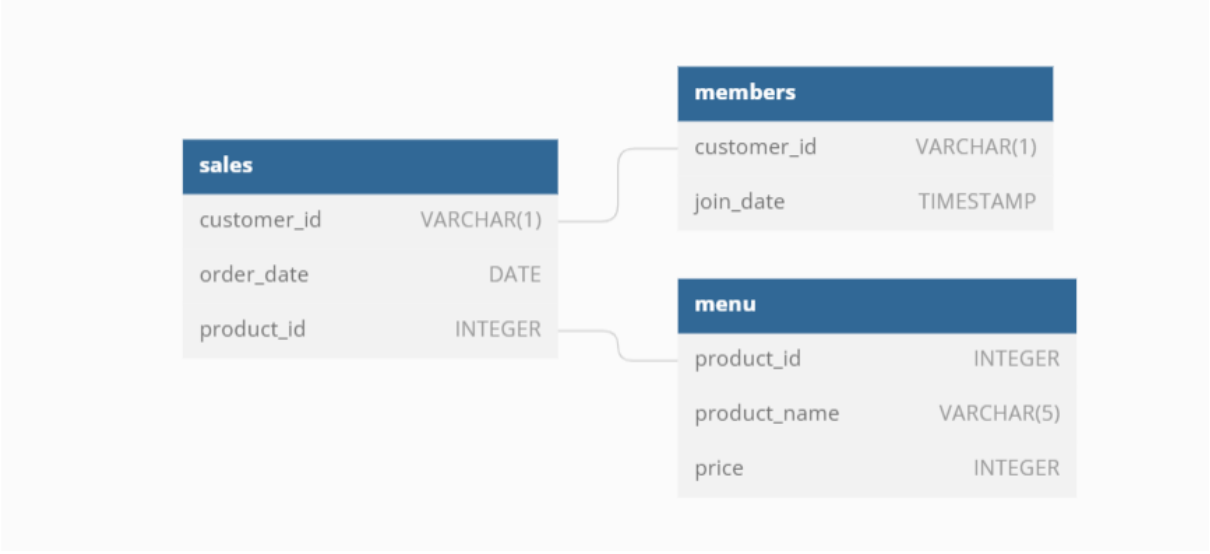
Danny has provided you with a sample of his overall customer data due to privacy issues - but he hopes that these examples are enough for you to write fully functioning SQL queries to help him answer his questions!

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.

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	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

Table 2: menu

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

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	join_date
A	2021-01-07
B	2021-01-09

Case Study Questions

Each of the following case study questions can be answered using a single SQL statement:

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

```
SELECT
        s.customer_id, SUM(me.price) AS total_price
FROM dannys_diner.sales AS s
INNER JOIN dannys_diner.menu AS me ON me.product_id=s.product_id
GROUP BY 1
ORDER BY customer_id;
```

customer_id	total_price
A	76
B	74
C	36

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

```
SELECT
        customer_id, COUNT(order_date) AS number_of_days
FROM (SELECT DISTINCT
        s.customer_id,s.order_date
FROM dannys_diner.sales AS s
ORDER BY s.customer_id) AS subt
GROUP BY 1;
```

customer_id	number_of_days
A	4
B	6
C	2

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

```
SELECT DISTINCT s.customer_id, s.order_date, me.product_name
FROM dannys_diner.sales AS s
        JOIN
(SELECT
        s.customer_id, MIN(s.order_date) AS order_date
FROM dannys_diner.sales AS s
INNER JOIN dannys_diner.menu AS me ON me.product_id=s.product_id
GROUP BY 1) AS b ON s.customer_id=b.customer_id
JOIN dannys_diner.menu AS me ON s.product_id=me.product_id
WHERE s.order_date=b.order_date
ORDER BY s.customer_id;
```

customer_id	order_date	product_name
A	2021-01-01T00:00:00.000Z	curry
A	2021-01-01T00:00:00.000Z	sushi
B	2021-01-01T00:00:00.000Z	curry
C	2021-01-01T00:00:00.000Z	ramen

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

```
SELECT me.product_name,COUNT(s.product_id) AS
number_of_occurence
FROM dannys_diner.sales AS s
JOIN dannys_diner.menu AS me ON s.product_id=me.product_id
GROUP BY 1
LIMIT 1;
```

product_name	number_of_occurence
ramen	8

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

```
SELECT customer_id, product_name FROM
(SELECT s.customer_id,me.product_name,
DENSE_RANK() OVER(PARTITION BY s.customer_id ORDER BY
COUNT(s.product_id) DESC) AS ranking
FROM dannys_diner.sales AS s
JOIN dannys_diner.menu AS me ON s.product_id=me.product_id
GROUP BY 1,2) AS subt
WHERE ranking=1;
```

customer_id	product_name
A	ramen
B	ramen
B	curry
B	sushi
C	ramen

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

```
SELECT customer_id,product_name FROM (SELECT DISTINCT
*,DENSE_RANK() OVER(PARTITION BY customer_id ORDER BY
order_date) RankING FROM
(SELECT s.customer_id,me.product_name,s.order_date
FROM dannys_diner.sales AS s
JOIN dannys_diner.members AS mem ON
s.customer_id=mem.customer_id
JOIN dannys_diner.menu AS me ON me.product_id=s.product_id
WHERE s.order_date>=mem.join_date
ORDER BY customer_id) AS subT
```

```
ORDER BY customer_id)AS subTT
WHERE ranking=1;
```

customer_id	product_name
A	curry
B	sushi

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

```
SELECT customer_id,product_name FROM(SELECT DISTINCT
*,DENSE_RANK() OVER(PARTITION BY customer_id ORDER BY
order_date DESC) RankING FROM
(SELECT s.customer_id,me.product_name,s.order_date
FROM dannys_diner.sales AS s
JOIN dannys_diner.members AS mem ON
s.customer_id=mem.customer_id
JOIN dannys_diner.menu AS me ON me.product_id=s.product_id
WHERE s.order_date<mem.join_date
ORDER BY customer_id) AS subT
ORDER BY customer_id)AS subTT
WHERE ranking=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?**

```
SELECT
        s.customer_id, COUNT(s.product_id) as total_items, sum(me.price)
as total_price
```

```
FROM dannys_diner.sales as s
JOIN dannys_diner.members AS mem ON
s.customer_id=mem.customer_id
JOIN dannys_diner.menu AS me ON me.product_id=s.product_id
WHERE s.order_date<mem.join_date
GROUP BY s.customer_id
ORDER BY s.customer_id;
```

customer_id	total_items	total_price
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?**

```
SELECT s.customer_id, sum(
CASE
WHEN me.product_name = 'sushi' THEN me.price*20
ELSE me.price*10
END ) AS total_points
FROM dannys_diner.sales AS s
INNER JOIN  dannys_diner.menu AS me ON s.product_id =
me.product_id
GROUP BY 1
ORDER BY 1;
```

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?

```
SELECT customer_id,SUM(total_points)AS total_points
FROM((SELECT customer_id,SUM(points) AS total_points
FROM
(SELECT s.customer_id,
CASE
WHEN me.product_name='sushi' THEN me.price*10*2
ELSE me.price*10
END AS points
FROM dannys_diner.sales AS s
JOIN dannys_diner.members AS mem ON
s.customer_id=mem.customer_id
JOIN dannys_diner.menu AS me ON me.product_id=s.product_id
WHERE s.order_date<mem.join_date OR s.order_date>mem.join_date+7
ORDER BY customer_id) AS t
GROUP BY 1)
UNION
(SELECT s.customer_id,sum(me.price*10*2) AS first_week_points
FROM dannys_diner.sales AS s
JOIN dannys_diner.members AS mem ON
s.customer_id=mem.customer_id
JOIN dannys_diner.menu AS me ON me.product_id=s.product_id
WHERE s.order_date>=mem.join_date AND
s.order_date<=mem.join_date+7
GROUP BY 1
ORDER BY customer_id))AS finalt
GROUP BY 1
```

customer_id	total_points
A	1370
B	1060