Danny’s Diner Case Study

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 favorite 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 favorite. Having this deeper connection with his customers will help him deliver a better and more personalized experience for his loyal customers.

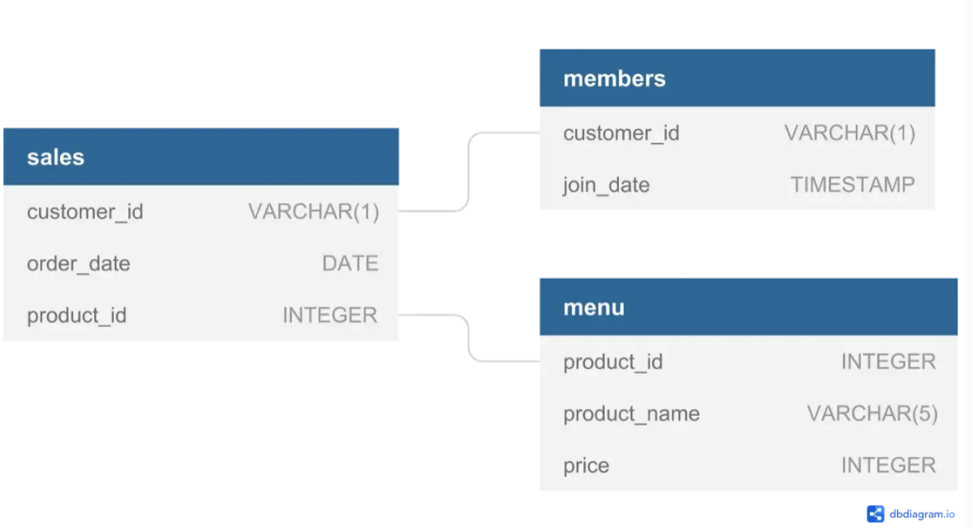
He plans on using these insights to help him decide whether he should expand the existing customer loyalty program – additionally he needs help to generate some basic datasets so his team can easily inspect the data without needing to use SQL.

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
- members

The Entity Relationship Diagram is shown here below:



Entity Relationship Diagram

Case Study Questions

1. what is the total amount each customer spent at the restaurant?
2. How many days has each customer visited the restaurant?
3. What was the first item from the menu purchased by each customer?
4. What is the most purchased item on the menu and how many items was it purchased by all customers?
5. Which item was the most popular for each customer?
6. Which item was purchased first by the customer after they became a member?
7. Which item was purchased just before the customer became a member?
8. What is the total items and amount spent for each member before they became a member?
9. If each $1 spent equates to 10 points and sushi has a 2x points multiplier – how many points would each customer have?
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?

Solution

On investigating the database, no data cleaning/ transformation is required, so I proceeded to answer the case study questions.

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

This required using the JOIN STATEMENT to join the menu and sales tables, then utilizing the SUM function on the price column, grouping by the customer_id.
Total amount spent:

- Customer A spent $76
- Customer B spent $74
- Customer C spent $36

2. **How many days has each customer visited the restaurant?**
   I determined this using the order_date column in the sales table. To avoid repetitive counting of multiple visits on a single day, I used the COUNT DISTINCT function. Customer B had the most visits (6), followed by customer A with 4 visits and customer C with just 2 visits.
3. **What was the first item from the menu purchased by each customer?**
   For this question, I used DENSE_RANK to rank the orders by the order_date column for each customer to create a temporary table ranked_date. The required window is PARTITION BY customer_id ORDER BY order_date. From the ranked_date table, since we only want the very first order, I filtered the results for rank = 1.
   - Customer A's first order was curry and sushi
   - Customer B's first order was curry
   - Customer C's first order was ramen
4. **What is the most purchased item on the menu and how many times was it purchased by all customers?**
   To answer this question, I joined the menu and sales tables, then counted the product_id from the sales table, grouping by the product_name. I ordered the resulting table by descending count. Since only the most purchased item was required, I limited the results to only show the first row.
5. **Which item was the most popular for each customer?**
   For each customer, I aggregated by count the product_id and used DENSE_RANK to rank the counts by descending order. Since we wanted the most popular item, I filtered for rank = 1.
   - Customer A's most popular item was ramen, bought thrice
   - Customer B's most popular items were sushi, curry & ramen, all bought twice
   - Customer C's most popular item on the menu was ramen, bought thrice
6. **Which item was purchased first by the customer after they became a member?**
   For this question, I joined all the tables, used DENSE_RANK to rank the orders by the order_date. To get orders that occurred after the join_date, I filtered for order_date ≥ join_date. The resulting table was then filtered for date_rank =1.
   - Customer A's first order after joining ws curry

- Customer B's first order after joining was sushi
- Customer C did not become a member

7. **Which item was purchased just before the customer became a member?**
   The approach is similar to the previous question. However, for this one, I filtered the tables for only `order_date < join_date`. In the desired window, I ordered the `order_date` in descending order, so the latest order would have a rank of 1.
   - Customer A last purchased sushi and curry before becoming a member
   - Customer B last purchased sushi before becoming a member
8. **What is the total items and amount spent for each member before they became a member?**
   Here, for each customer, I counted the `product_id` and aggregated by summing the price to get the total amount spent. I then filtered for `order_date < join_date`.
9. **If each $1 spent equates to 10 points and sushi has a 2x points multiplier — how many points would each customer have?**
   Here, I used the CASE STATEMENT to create a table including a column `points`, where sushi is assigned 20 points and 10 points for any other product. I then summed the `points` column for each customer.
   - Customer B has 940 points
   - Customer A has 860 points
   - Customer C has 360 points
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?**
    For this question, I had to use multiple temporary tables. In the first temp table, I created a column `one_week_later` to show the date within which the 2X points are gained. Next, I used a CASE statement to calculate the `points` column showing points gained during the promo period by multiplying the price by 20, whereas all purchases made outside this window have the x10 multiplier. Finally, I summed up the points for each customer.
    - Customer A had 1270 points
    - Customer B had 720 points

Bonus Questions

Join all the things to show a table using the available data showing `customer_id`, `order_date`, `product_name`, `price`, `member`.

This required joining all the tables followed by using a CASE statement to create the `member` column. We had to use a LEFT JOIN on sales and members tables because we didn't want to exclude customer C who didn't have a corresponding id in the members table.

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.

With the previous table as `join_cte`, I used the CASE statement to create a column `ranking`, where for each customer, orders are ranked basing on the condition that they are members.

Conclusions

From this analysis, I was able to determine and answer questions posed by Danny's Diner. Insights from this analysis include:

- Customer A spent the most money ($76) during this period, followed by Customer B ($74) and Customer C ($36). However, Customer B had the most visits to the Diner, followed by Customer A.

- The most popular item on the menu is ramen, which was purchased 8 times.