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

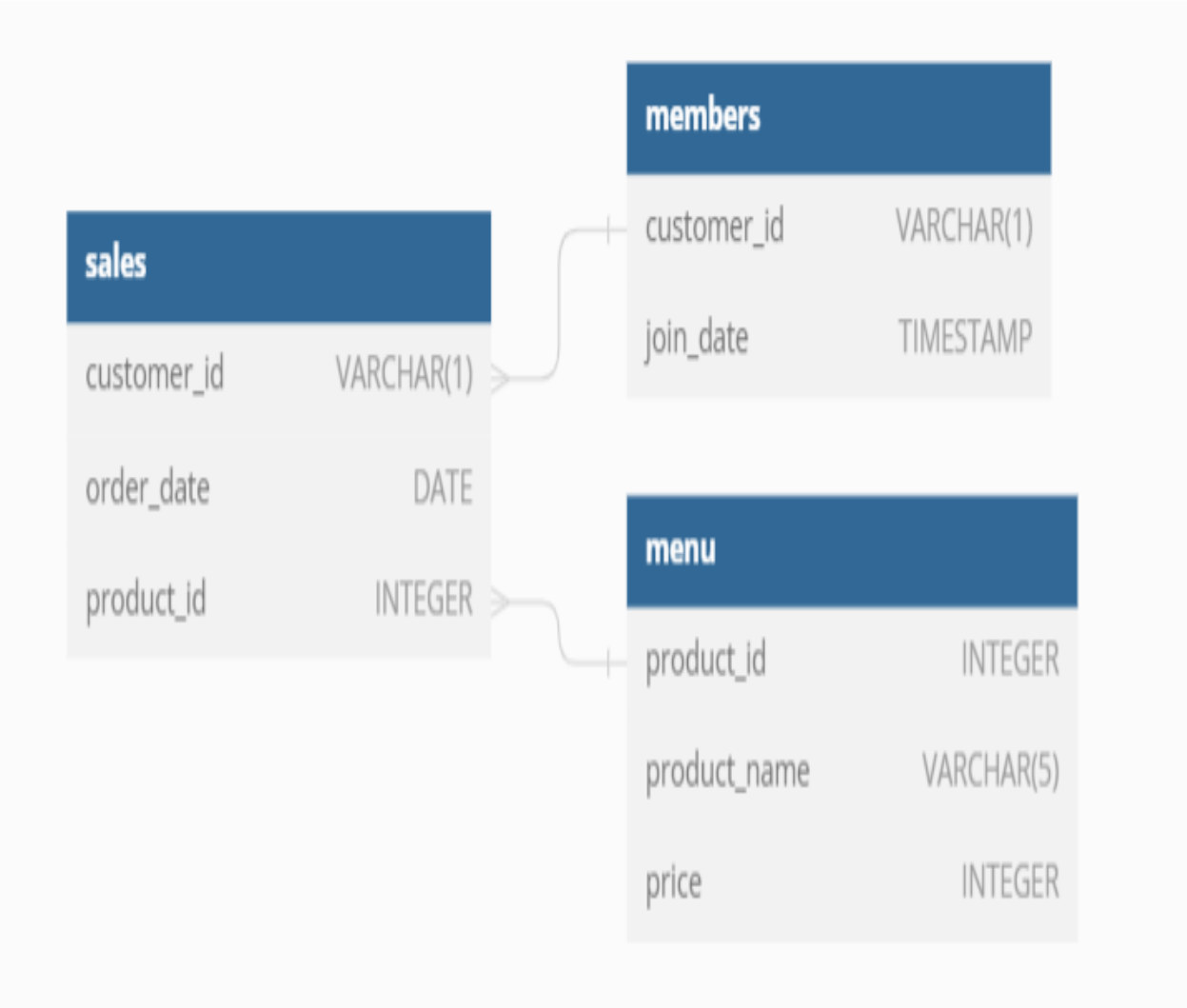

Table 1: Sales

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

SQL SCHEMA

```sql
CREATE SCHEMA dannys_diner;
SET search_path = dannys_diner;

CREATE TABLE sales (
  "customer_id" VARCHAR(1),
  "order_date" DATE,
  "product_id" INTEGER
);

INSERT INTO sales
  ("customer_id", "order_date", "product_id")
VALUES
  ('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');
```

```sql
CREATE TABLE menu (
  "product_id" INTEGER,
  "product_name" VARCHAR(5),
  "price" INTEGER
);

INSERT INTO menu
  ("product_id", "product_name", "price")
VALUES
  ('1', 'sushi', '10'),
  ('2', 'curry', '15'),
  ('3', 'ramen', '12');


CREATE TABLE members (
  "customer_id" VARCHAR(1),
  "join_date" DATE
);

INSERT INTO members
  ("customer_id", "join_date")
VALUES
  ('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?
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 times 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?

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

```
SELECT customer_id, SUM(PRICE) AS total_amount_spent FROM menu
JOIN sales ON sales.product_id = menu.product_id
GROUP BY customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	total_amount_spent
A	76
B	74
C	36

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

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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	number_of_days_visited
A	4
B	6
C	2

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

```
SELECT DISTINCT(customer_id), product_name FROM sales
JOIN menu ON sales.product_id = menu.product_id
WHERE order_date = ANY(SELECT MIN(order_date) FROM sales GROUP BY customer_id);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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?

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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_name	number_of_times_purchased
ramen	8

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

```
WITH order_info AS
 (SELECT product_name, customer_id,
      count(product_name) AS order_count,
     rank() over(PARTITION BY customer_id
                ORDER BY count(product_name) DESC) AS rank_num
  FROM dannys_diner.menu
  INNER JOIN dannys_diner.sales ON menu.product_id = sales.product_id
  GROUP BY customer_id, product_name)
SELECT customer_id, product_name,      order_count
FROM order_info
WHERE rank_num = 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

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

```
WITH member_sales_cte AS
(
 SELECT s.customer_id, m.join_date, s.order_date,   s.product_id,
        DENSE_RANK() OVER(PARTITION BY s.customer_id
 ORDER BY s.order_date) AS first_item
    FROM sales AS s
 JOIN members AS m
  ON s.customer_id = m.customer_id
 WHERE s.order_date >= m.join_date
)
SELECT s.customer_id, s.order_date, m2.product_name
FROM member_sales_cte AS s
JOIN menu AS m2
 ON s.product_id = m2.product_id
WHERE first_item = 1
ORDER BY s.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	order_date	product_name
A	2021-01-07	curry
B	2021-01-11	sushi

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

```
WITH prior_member_purchased_cte AS
(
 SELECT s.customer_id, m.join_date, s.order_date, s.product_id,
        DENSE_RANK() OVER(PARTITION BY s.customer_id
        ORDER BY s.order_date DESC) AS first_item
 FROM sales AS s
 JOIN members AS m
  ON s.customer_id = m.customer_id
 WHERE s.order_date < m.join_date
)
SELECT s.customer_id, s.order_date, m2.product_name
FROM prior_member_purchased_cte AS s
JOIN menu AS m2
 ON s.product_id = m2.product_id
WHERE first_item = 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	order_date	product_name
A	2021-01-01	sushi
B	2021-01-04	sushi
A	2021-01-01	curry

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

```
SELECT s.customer_id, COUNT(DISTINCT(s.product_id)) AS total_items, SUM(mm.price)
FROM sales AS s
JOIN members AS m ON s.customer_id = m.customer_id
JOIN menu AS mm ON s.product_id = mm.product_id
WHERE s.order_date < m.join_date
GROUP BY s.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	total_items	SUM(mm.price)
A	2	25
B	2	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 price_points AS
(
SELECT *,
CASE
 WHEN product_id = 1 THEN price * 20
 ELSE price * 10
 END AS points
FROM menu
)
SELECT s.customer_id, SUM(p.points) AS total_points
FROM price_points AS p
JOIN sales AS s
 ON p.product_id = s.product_id
GROUP BY s.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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?

```
 WITH program_last_day_cte AS
 (SELECT join_date,
       DATE_ADD(join_date, INTERVAL 7 DAY) AS program_last_date,
       customer_id
  FROM dannys_diner.members)
SELECT s.customer_id,
     SUM(CASE
          WHEN order_date BETWEEN join_date AND program_last_date THEN price*10*2
          WHEN order_date NOT BETWEEN join_date AND program_last_date
             AND product_name = 'sushi' THEN price*10*2
          WHEN order_date NOT BETWEEN join_date AND program_last_date
             AND product_name != 'sushi' THEN price*10
        END) AS customer_points
FROM dannys_diner.menu AS m
INNER JOIN dannys_diner.sales AS s ON m.product_id = s.product_id
INNER JOIN program_last_day_cte AS mem ON mem.customer_id = s.customer_id
AND order_date <='2021-01-31'
AND order_date >=join_date
GROUP BY s.customer_id
ORDER BY s.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	customer_points
A	1020
B	440