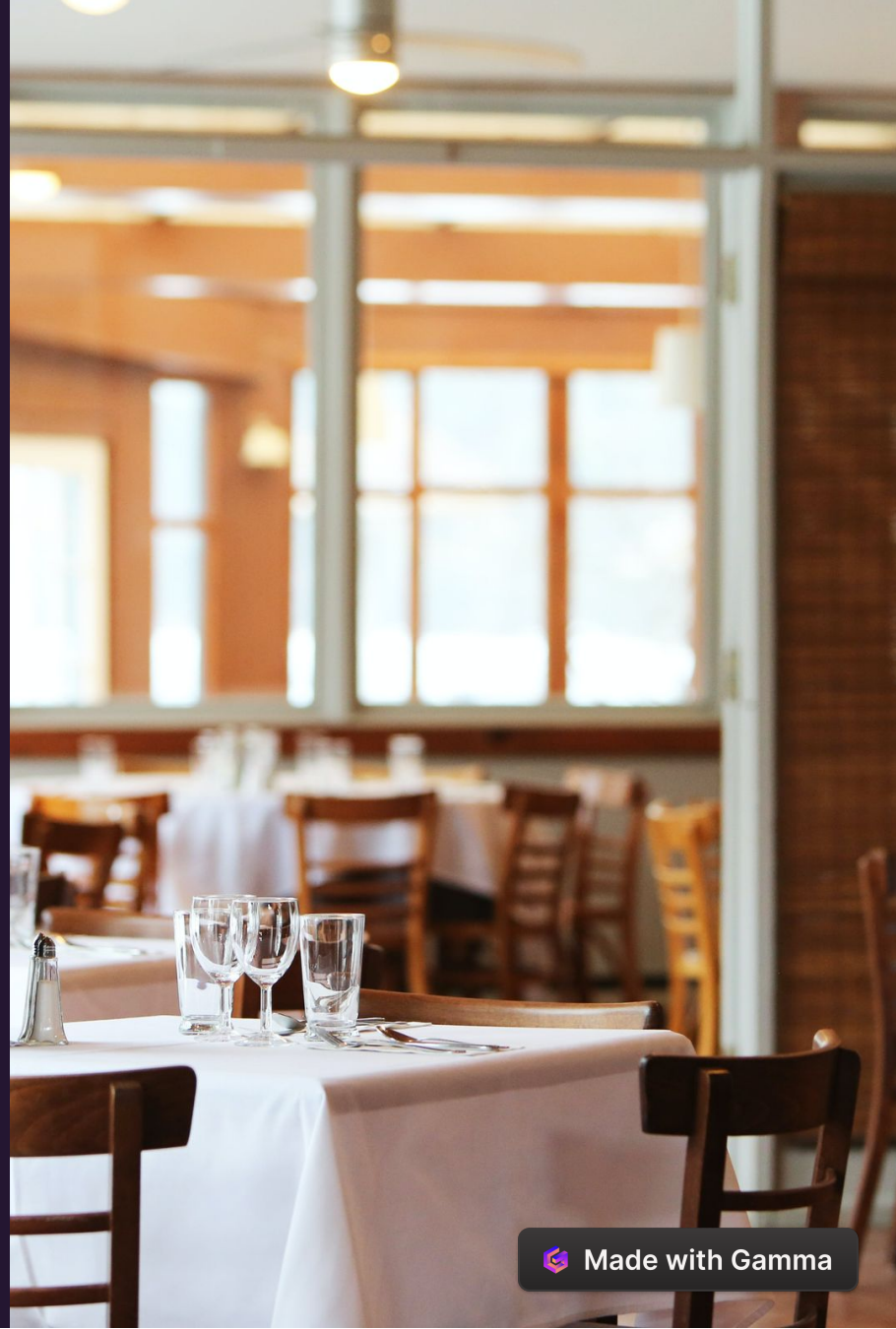
Case Study #1

- Danny's Diner

Welcome to our comprehensive solution overview for Danny's Diner case study. Join us as we dive into the queries and unveil the valuable insights that can be derived from the provided tables: `sales`, `menu`, and `members`.

By Vijaykumar Chauhan



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

Input Tables

Sales

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

Menu

product_id	product_name	price
1	sushi	10
2	curry	15
3	ramen	12

Members

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

Schema

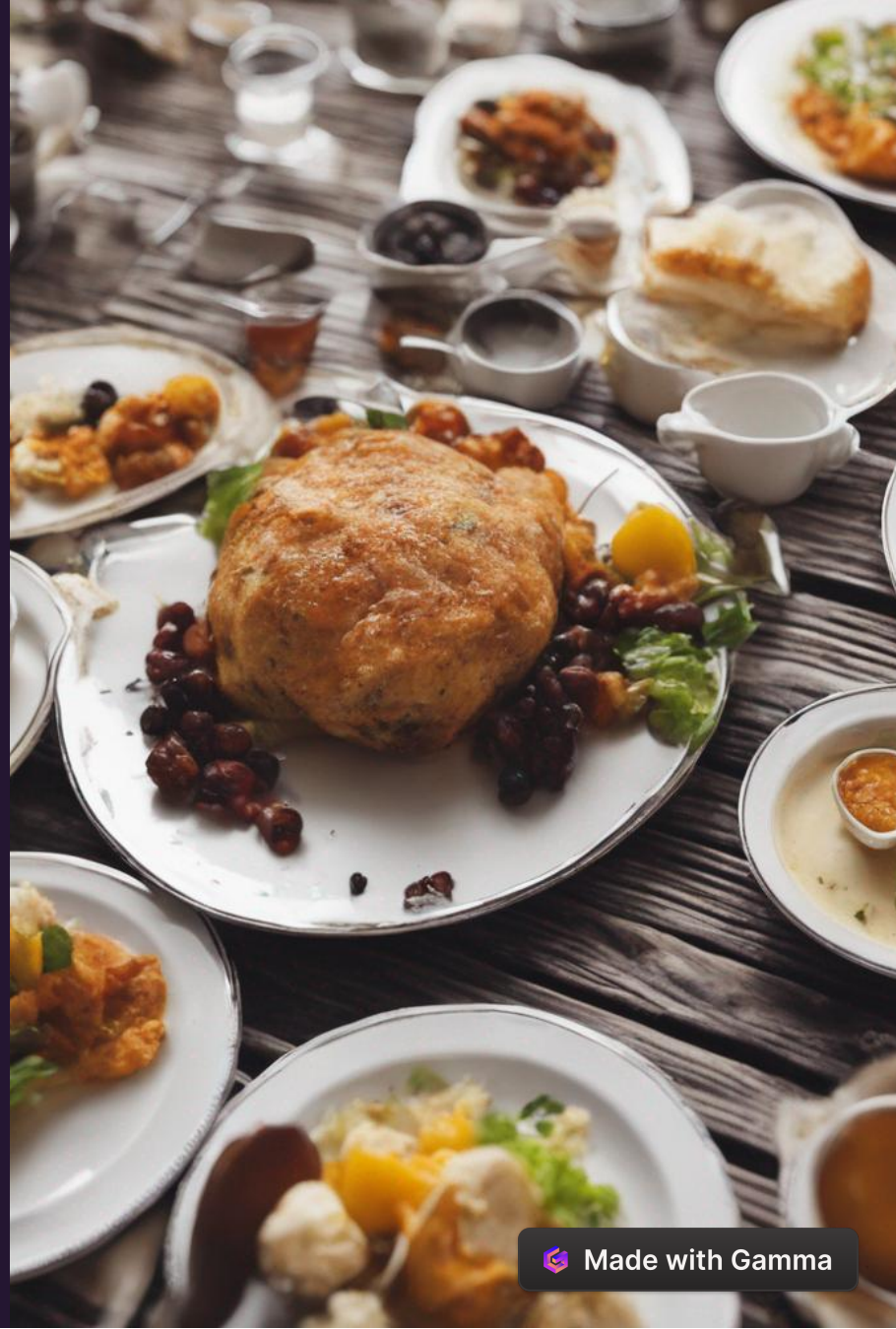
```
CREATE SCHEMA dannys_diner;

SET search_path = dannys_diner;

CREATE TABLE sales (

"customer_id" VARCHAR(1),

"order_date" DATE,

"product_id" INTEGER);

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');
```

```
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');
```

```
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');
```

Solution Overview

In order to tackle the first case study, we have devised various queries that will allow us to extract meaningful information from the available data. Let's take a closer look at each query and its purpose in our analysis.



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

```
WITH CustomerSpending AS (

SELECT s.customer_id, SUM(m.price) AS amount_spent

FROM sales s

INNER JOIN menu m ON s.product_id = m.product_id

GROUP BY s.customer_id )

SELECT * FROM CustomerSpending

ORDER BY amount_spent DESC;
```

Output

customer_id	amount_spent
A	76
B	74
C	36

Query 2: How many days has each customer visited the restaurant?

```
WITH CTE AS (
SELECT customer_id, EXTRACT(DAY FROM order_date) AS day_visited
FROM sales )
SELECT customer_id, COUNT(DISTINCT day_visited) AS total_days_visited
FROM CTE
GROUP BY customer_id
ORDER BY total_days_visited DESC;
```

Output

customer_id	total_days_visited
B	5
A	4
C	2

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

```
WITH FirstPurchase AS (

SELECT customer_id,

MIN(order_date) AS first_order_date FROM sales

GROUP BY customer_id )

SELECT F.customer_id, M.product_name

FROM FirstPurchase F

JOIN sales S ON F.customer_id = S.customer_id AND F.first_order_date =
S.order_date

JOIN menu M ON S.product_id = M.product_id;
```

Output

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

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

```
WITH CTE AS (

SELECT customer_id, product_id, COUNT(*) AS total_purchases

FROM sales

GROUP BY customer_id, product_id )

SELECT M.product_name, CTE.total_purchases

FROM CTE

INNER JOIN menu M ON M.product_id = CTE.product_id

ORDER BY CTE.total_purchases DESC

LIMIT 1;
```

Output

product_name	total_purchases
ramen	3

Query 5: Which item was the most popular for each customer?

```
WITH CTE AS (

SELECT customer_id, product_id, COUNT(*) AS total

FROM sales

GROUP BY customer_id, product_id )

SELECT C.customer_id, M.product_name

FROM (  SELECT customer_id, product_id, total,

RANK( ) OVER (PARTITION BY customer_id ORDER BY total DESC ) AS rnk

FROM CTE ) AS C

JOIN menu M ON M.product_id = C.product_id

WHERE rnk = 1;
```

Output

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

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

```
WITH RANK AS (

SELECT S.CUSTOMER_ID, M.PRODUCT_NAME,

DENSE_RANK() OVER (PARTITION BY S.CUSTOMER_ID

ORDER BY S.ORDER_DATE) AS RNK

FROM SALES S

JOIN MENU M ON M.PRODUCT_ID = S.PRODUCT_ID

JOIN MEMBERS MEM ON MEM.CUSTOMER_ID = S.CUSTOMER_ID

WHERE S.ORDER_DATE >= MEM.JOIN_DATE )

SELECT R.CUSTOMER_ID, R.PRODUCT_NAME

FROM RANK R

WHERE RNK = 1;
```

Output

customer_id	product_name
A	curry
B	sushi

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

```
WITH RANK AS (

SELECT S.CUSTOMER_ID, M.PRODUCT_NAME,

DENSE_RANK( ) OVER (PARTITION BY S.CUSTOMER_ID

ORDER BY S.ORDER_DATE) AS RNK

FROM SALES S

JOIN MENU M ON M.PRODUCT_ID = S.PRODUCT_ID

JOIN MEMBERS MEM ON MEM.CUSTOMER_ID = S.CUSTOMER_ID

WHERE S.ORDER_DATE < MEM.JOIN_DATE )

SELECT R.CUSTOMER_ID,R.PRODUCT_NAME

FROM RANK R

WHERE RNK = 1;
```

Output

customer_id	product_name
A	sushi
A	curry
B	curry

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

```
SELECT S.CUSTOMER_ID,
COUNT(*) AS TOTAL_ITEMS,
SUM(M.PRICE) AS AMOUNT_SPENT
FROM SALES S
JOIN MEMBERS MS ON
MS.CUSTOMER_ID = S.CUSTOMER_ID
AND S.ORDER_DATE < MS.JOIN_DATE
JOIN MENU M ON M.PRODUCT_ID = S.PRODUCT_ID
GROUP BY 1;
```

Output

customer_id	total_items	amount_spent
B	3	40
A	2	25

Query 9: If each $1 spent equates to 10 points and sushi has a 2x points multiplier - how many points would each customer have?

```
WITH POINTSCTE AS (

SELECT *, CASE WHEN PRODUCT_ID = 1

THEN 20 * PRICE ELSE 10 * PRICE END

AS POINTS FROM MENU )

SELECT S.CUSTOMER_ID, SUM(PC.POINTS) AS TOTAL_POINTS

FROM POINTSCTE PC RIGHT JOIN SALES S ON

S.PRODUCT_ID = PC.PRODUCT_ID GROUP BY S.CUSTOMER_ID

ORDER BY TOTAL_POINTS DESC;
```

Output

customer_id	total_items	amount_spent
B	3	40
A	2	25

Query 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 s.customer_id, SUM( CASE WHEN

(DATE_TRUNC('day', s.order_date) - DATE_TRUNC('day', me.join_date) BETWEEN '0 days'
AND '7 days') OR (m.product_id = 1) THEN m.price * 20

ELSE m.price * 10 END) AS Points

FROM members AS me

INNER JOIN sales AS s ON s.customer_id = me.customer_id

INNER JOIN menu AS m ON m.product_id = s.product_id

WHERE s.order_date >= me.join_date

AND s.order_date <= '2021-01-31'

GROUP BY s.customer_id

order by 1;
```

Output

customer_id	points
A	1020
B	440

Note : You can use date_trunc, extract,date_diff and date_part functions also

Bonus Questions

Join All The Things

The following questions are related 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:

```
CREATE TABLE customer_orders AS

SELECT s.customer_id, s.order_date, m.product_name, m.price,

CASE WHEN mem.customer_id IS NOT NULL THEN 'Y'

ELSE 'N' END AS member

FROM sales s

JOIN menu m

 ON s.product_id = m.product_id

LEFT JOIN  members mem

ON s.customer_id = mem.customer_id

AND s.order_date >= mem.join_date;
```

Output

```
SELECT  * FROM   customer_orders;
```

customer_id	order_date	product_name	price	member
B	2021-01-04	sushi	10	N
A	2021-01-01	sushi	10	N
B	2021-01-11	sushi	10	Y
B	2021-01-01	curry	15	N
B	2021-01-02	curry	15	N
A	2021-01-01	curry	15	N
A	2021-01-07	curry	15	Y
A	2021-01-11	ramen	12	Y
A	2021-01-11	ramen	12	Y
A	2021-01-10	ramen	12	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

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.

```
CREATE TABLE customer_product_ranking AS

WITH RankedProducts AS (

SELECT s.customer_id, s.order_date, m.product_name,

m.price, CASE WHEN mem.customer_id IS NOT NULL THEN 'Y' ELSE 'N'

END AS member,

ROW_NUMBER() OVER (PARTITION BY s.customer_id,

mem.customer_id ORDER BY s.order_date) AS ranking

FROM sales s JOIN menu m

 ON s.product_id = m.product_id

LEFT JOIN members mem

ON s.customer_id = mem.customer_id

AND s.order_date >= mem.join_date )

SELECT customer_id, order_date, product_name, price, member,

CASE WHEN member = 'Y' THEN ranking ELSE NULL

END AS ranking FROM RankedProducts;
```

Output

```
SELECT * FROM customer_product_ranking;
```

customer_id	order_date	product_name	price	member	ranking
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	4
A	2021-01-01	curry	15	N	null
A	2021-01-01	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
B	2021-01-01	curry	15	N	null
B	2021-01-02	curry	15	N	null
B	2021-01-04	sushi	10	N	null
C	2021-01-01	ramen	12	N	null
C	2021-01-01	ramen	12	N	null
C	2021-01-07	ramen	12	N	null

Thankyou