Danny's Dinner

The Taste Of Sucess

Presented By SRABANA BAIDYA

INTRODUCTION

Danny, driven by his love for Japanese cuisine, took a bold step in early 2021 by opening a cozy restaurant that features his three favorite dishes: sushi, curry, and ramen. Although Danny's Diner has quickly become a favorite among its patrons, it now faces challenges in sustaining its success. Despite collecting some basic data in its first few months, Danny needs assistance in turning this information into actionable insights. Our goal is to analyze this data and provide Danny with the strategies he needs to make informed decisions and ensure his restaurant thrives.

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

Entity Relationship Diagram

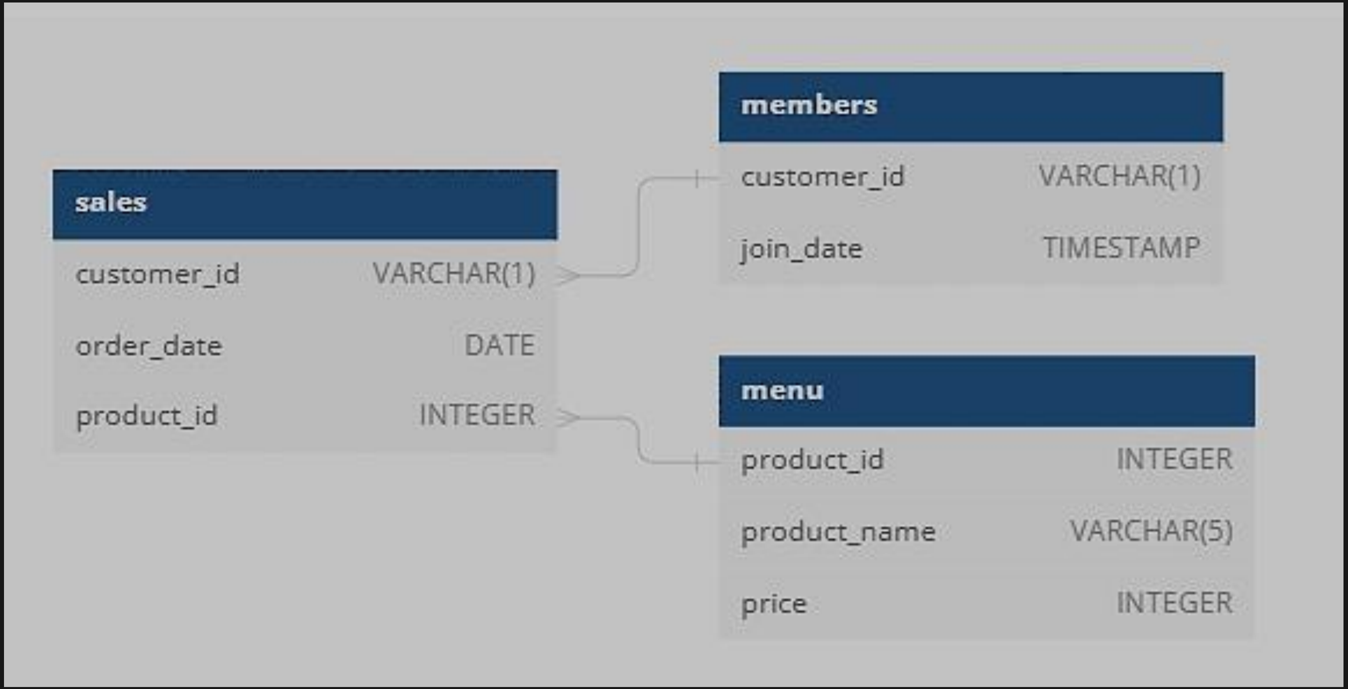
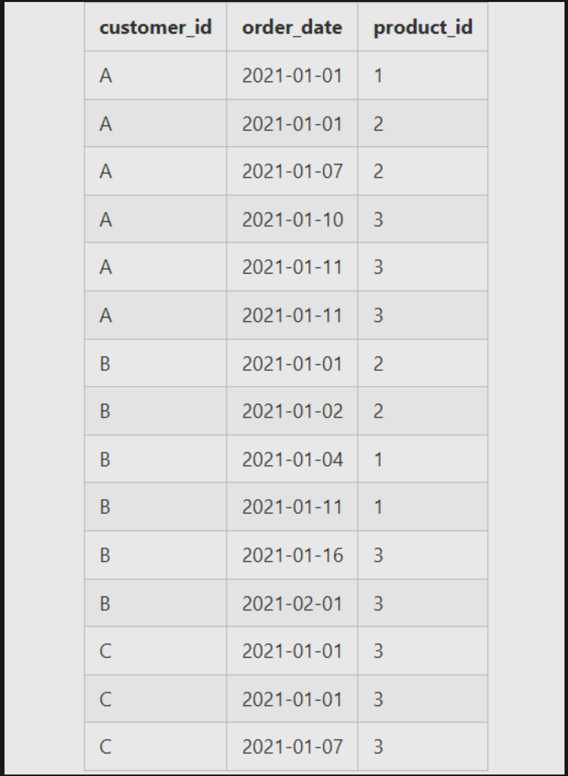


Table 1 : Sales

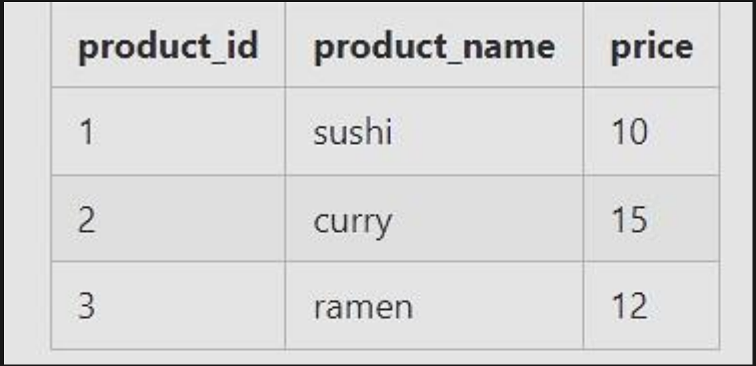
The sales table captures all customer_id level purchases with an corresponding order_date and product_id information for when and what menu items were ordered.



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

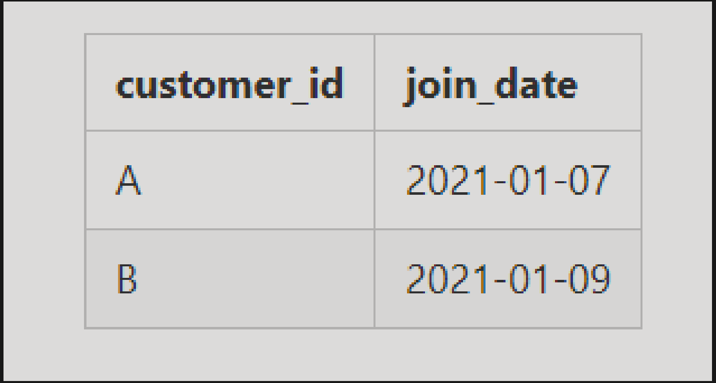
The menu table maps the product_id to the actual product_name and price of each menu item



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.



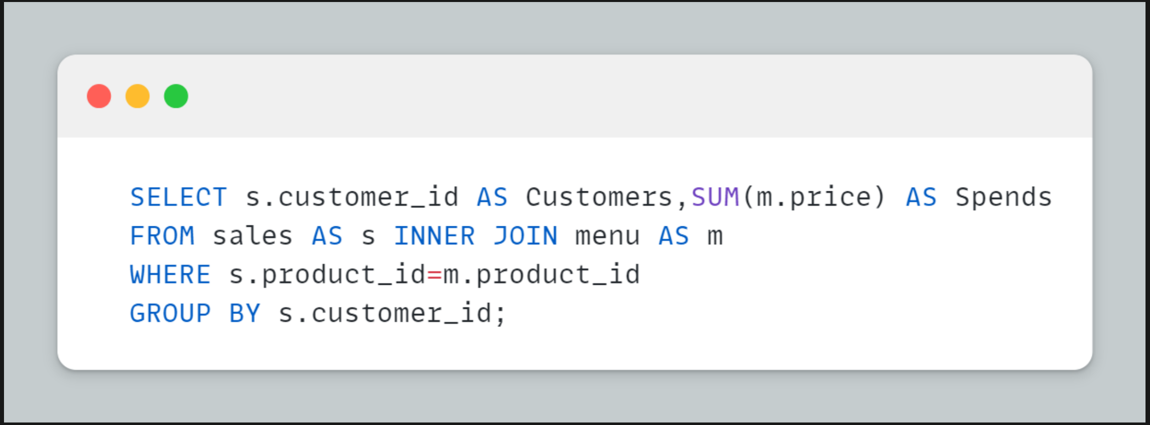
customer_id	join_date
A	2021-01-07
B	2021-01-09

CASE

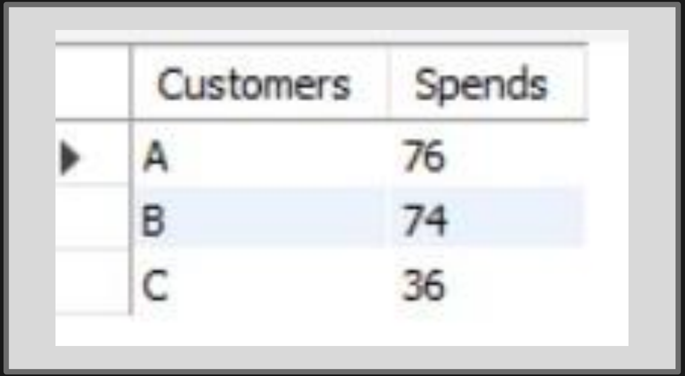
STUDIES

STSSTUDIES

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

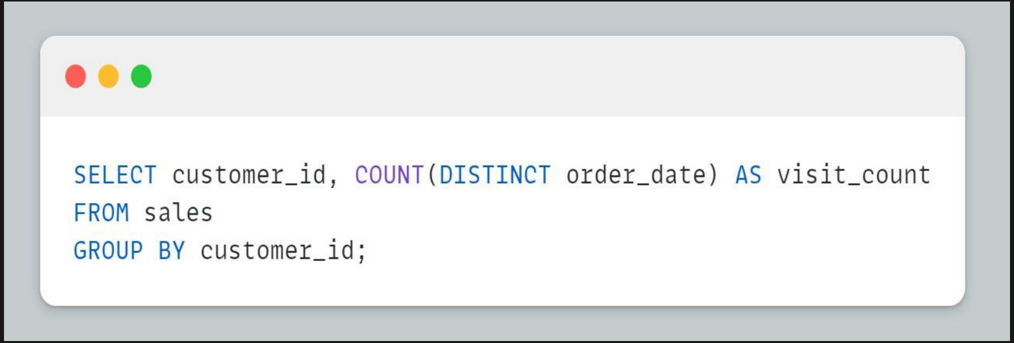


```sql
SELECT s.customer_id AS Customers,SUM(m.price) AS Spends
FROM sales AS s INNER JOIN menu AS m
WHERE s.product_id=m.product_id
GROUP BY s.customer_id;
```

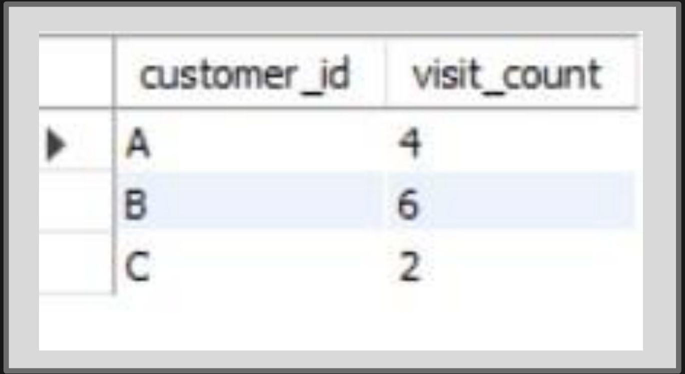


Customers	Spends
A	76
B	74
C	36

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

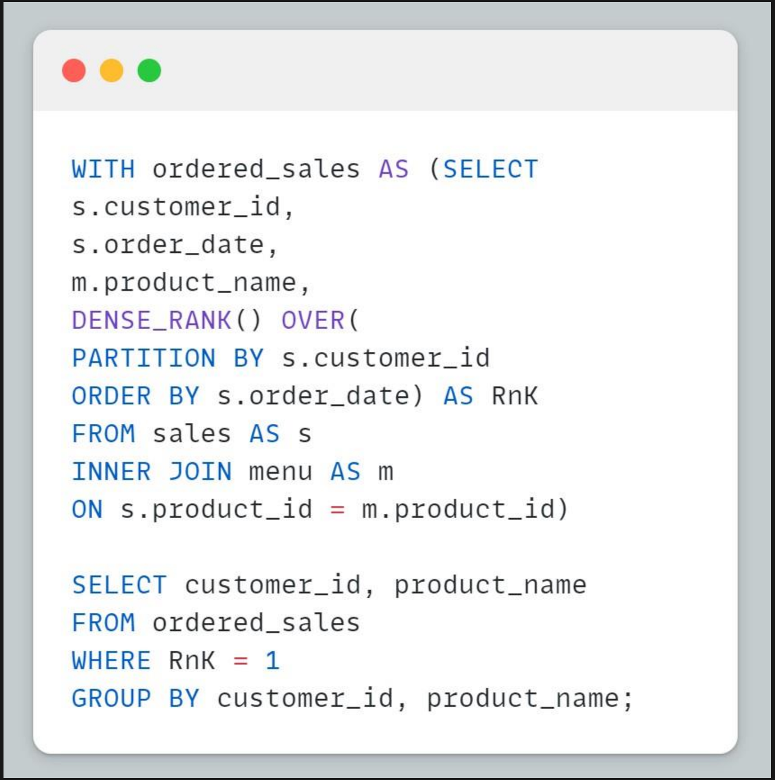


```sql
SELECT customer_id, COUNT(DISTINCT order_date) AS visit_count
FROM sales
GROUP BY customer_id;
```

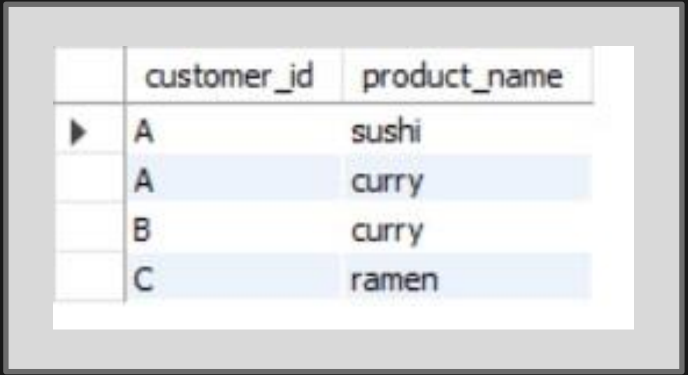


customer_id	visit_count
A	4
B	6
C	2

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

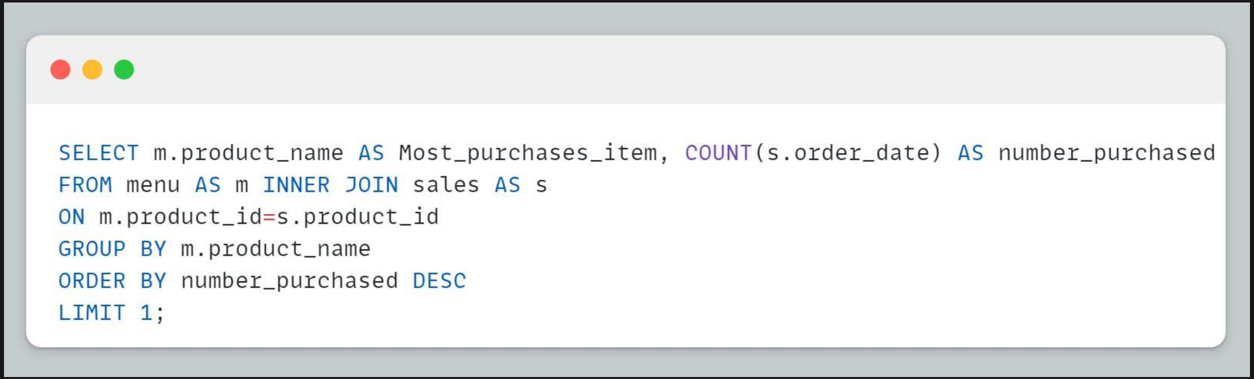


```sql
WITH ordered_sales AS (SELECT
s.customer_id,
s.order_date,
m.product_name,
DENSE_RANK() OVER(
PARTITION BY s.customer_id
ORDER BY s.order_date) AS RnK
FROM sales AS s
INNER JOIN menu AS m
ON s.product_id = m.product_id)

SELECT customer_id, product_name
FROM ordered_sales
WHERE RnK = 1
GROUP BY customer_id, product_name;
```

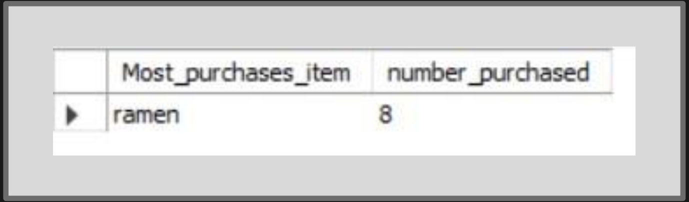


	customer_id	product_name
▶	A	sushi
	A	curry
	B	curry
	C	ramen

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

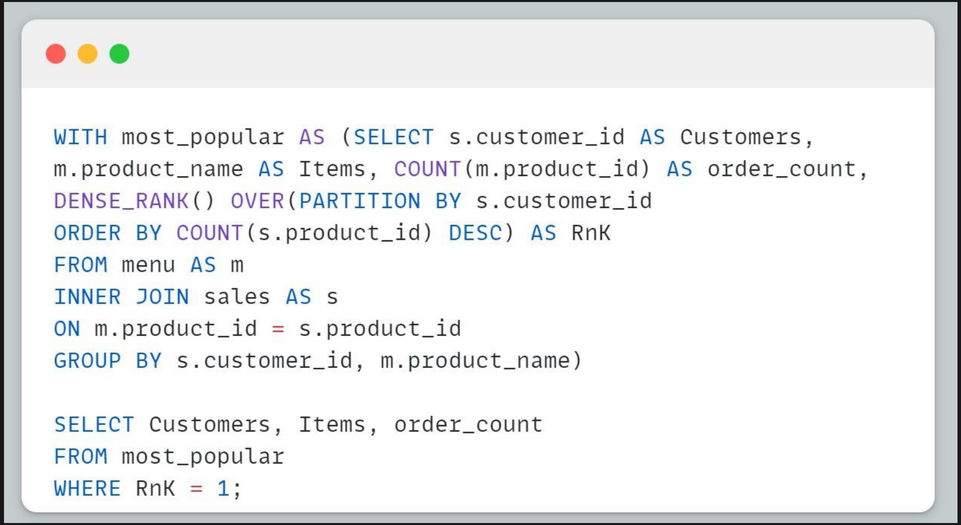


```sql
SELECT m.product_name AS Most_purchases_item, COUNT(s.order_date) AS number_purchased
FROM menu AS m INNER JOIN sales AS s
ON m.product_id=s.product_id
GROUP BY m.product_name
ORDER BY number_purchased DESC
LIMIT 1;
```

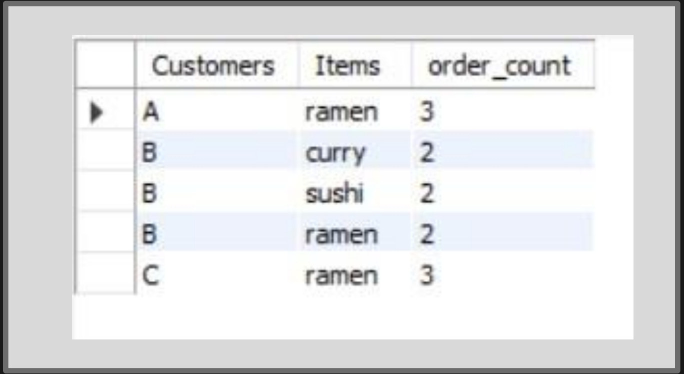


	Most_purchases_item	number_purchased
▶	ramen	8

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

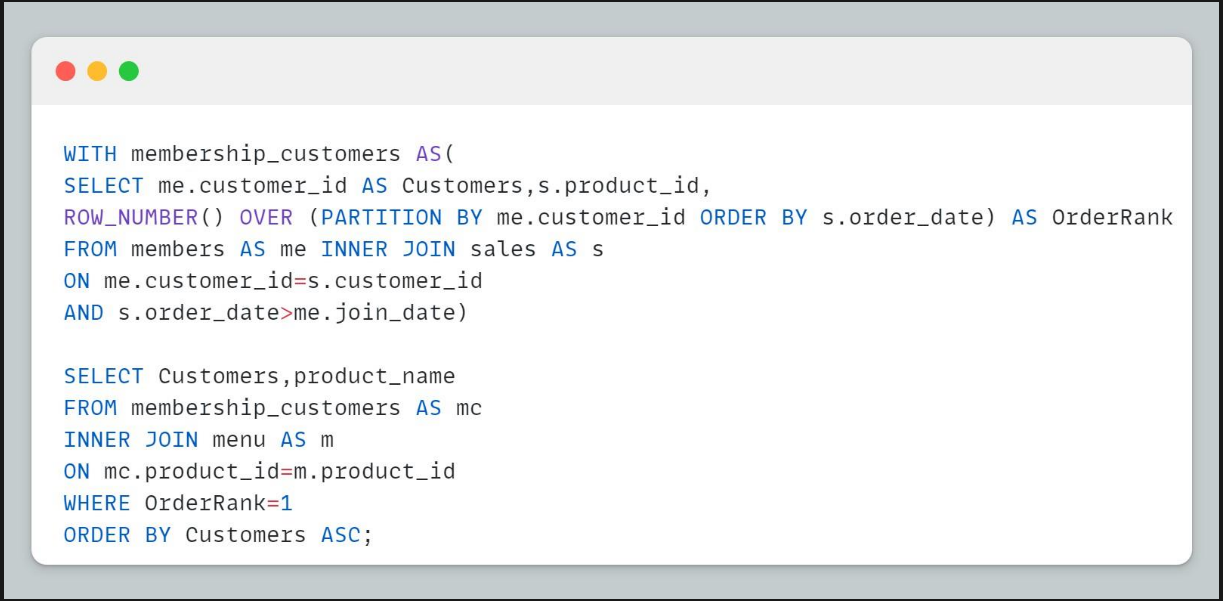


```sql
WITH most_popular AS (SELECT s.customer_id AS Customers,
m.product_name AS Items, COUNT(m.product_id) AS order_count,
DENSE_RANK() OVER(PARTITION BY s.customer_id
ORDER BY COUNT(s.product_id) DESC) AS RnK
FROM menu AS m
INNER JOIN sales AS s
ON m.product_id = s.product_id
GROUP BY s.customer_id, m.product_name)

SELECT Customers, Items, order_count
FROM most_popular
WHERE RnK = 1;
```

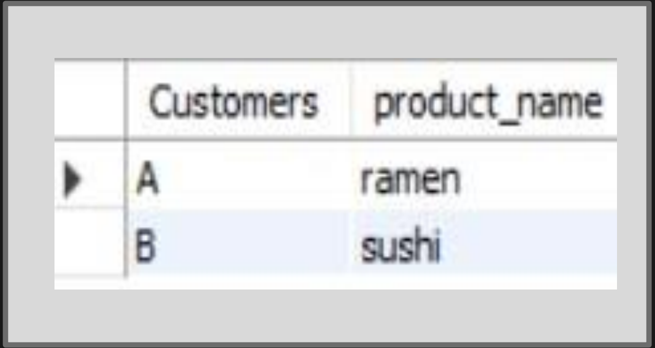


Customers	Items	order_count
A	ramen	3
B	curry	2
B	sushi	2
B	ramen	2
C	ramen	3

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

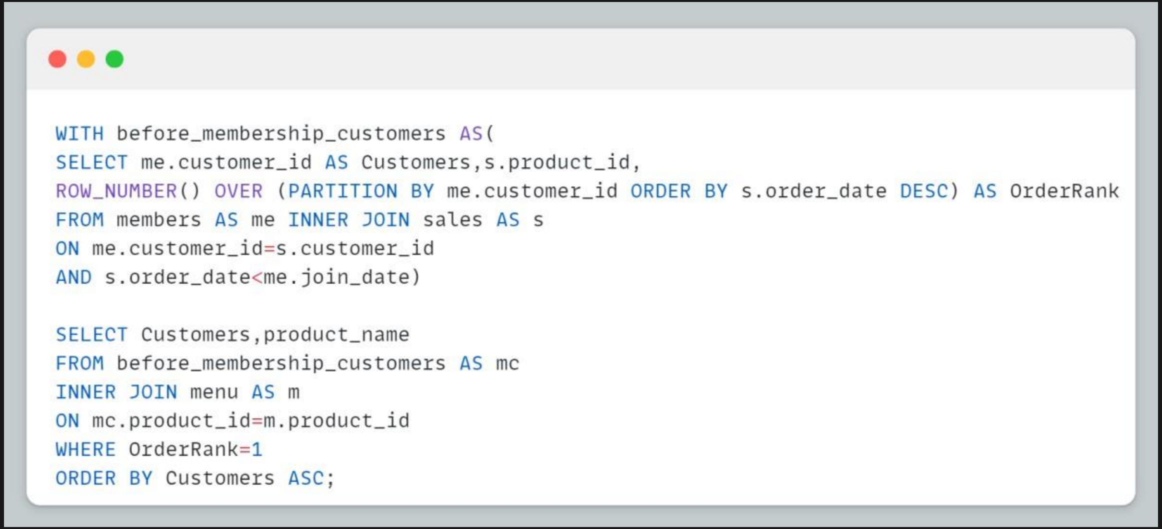


```
WITH membership_customers AS(
SELECT me.customer_id AS Customers,s.product_id,
ROW_NUMBER() OVER (PARTITION BY me.customer_id ORDER BY s.order_date) AS OrderRank
FROM members AS me INNER JOIN sales AS s
ON me.customer_id=s.customer_id
AND s.order_date>me.join_date)

SELECT Customers,product_name
FROM membership_customers AS mc
INNER JOIN menu AS m
ON mc.product_id=m.product_id
WHERE OrderRank=1
ORDER BY Customers ASC;
```

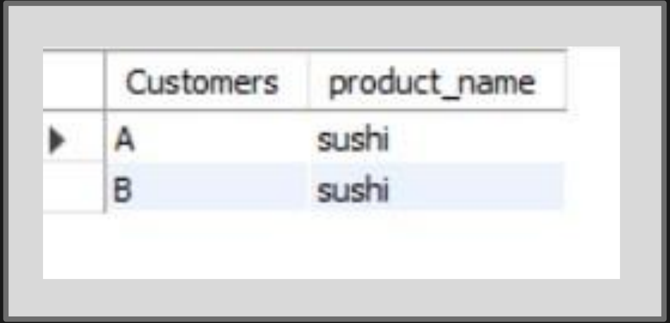


Customers	product_name
A	ramen
B	sushi

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

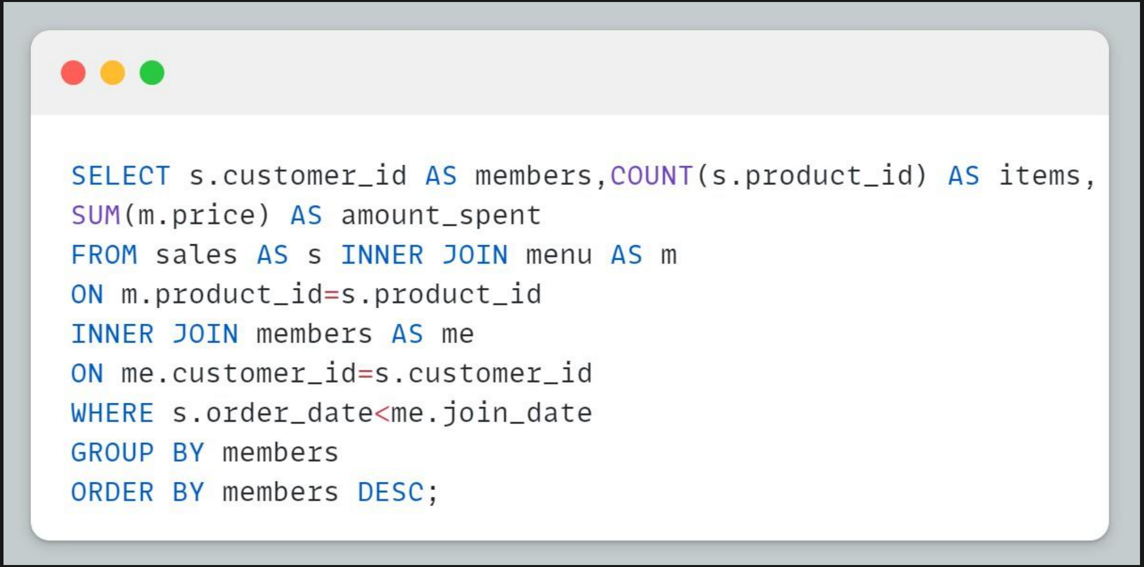


```sql
WITH before_membership_customers AS(
SELECT me.customer_id AS Customers,s.product_id,
ROW_NUMBER() OVER (PARTITION BY me.customer_id ORDER BY s.order_date DESC) AS OrderRank
FROM members AS me INNER JOIN sales AS s
ON me.customer_id=s.customer_id
AND s.order_date<me.join_date)

SELECT Customers,product_name
FROM before_membership_customers AS mc
INNER JOIN menu AS m
ON mc.product_id=m.product_id
WHERE OrderRank=1
ORDER BY Customers ASC;
```

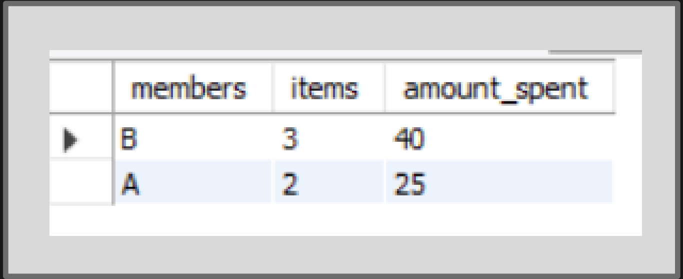


Customers	product_name
A	sushi
B	sushi

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

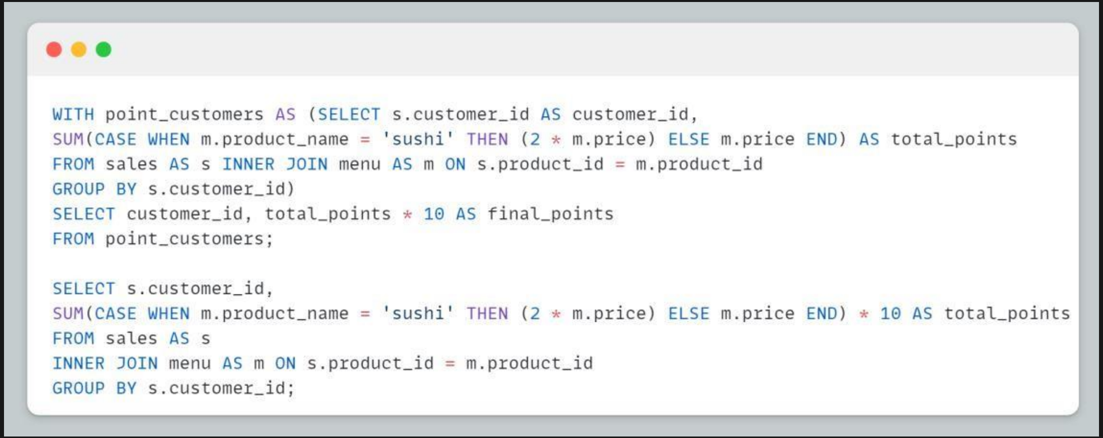


```sql
SELECT s.customer_id AS members,COUNT(s.product_id) AS items,
SUM(m.price) AS amount_spent
FROM sales AS s INNER JOIN menu AS m
ON m.product_id=s.product_id
INNER JOIN members AS me
ON me.customer_id=s.customer_id
WHERE s.order_date<me.join_date
GROUP BY members
ORDER BY members DESC;
```

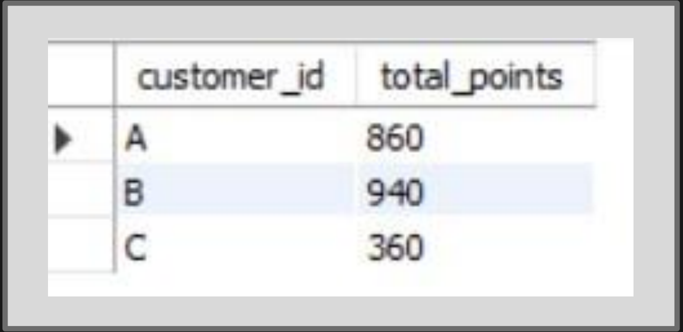


members	items	amount_spent
B	3	40
A	2	25

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

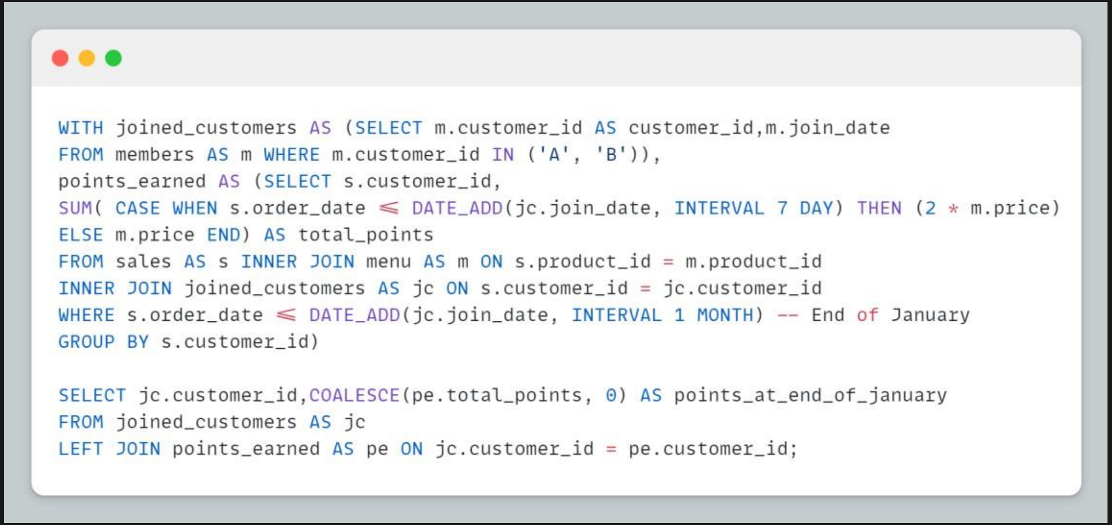


```
WITH point_customers AS (SELECT s.customer_id AS customer_id,
SUM(CASE WHEN m.product_name = 'sushi' THEN (2 * m.price) ELSE m.price END) AS total_points
FROM sales AS s INNER JOIN menu AS m ON s.product_id = m.product_id
GROUP BY s.customer_id)
SELECT customer_id, total_points * 10 AS final_points
FROM point_customers;

SELECT s.customer_id,
SUM(CASE WHEN m.product_name = 'sushi' THEN (2 * m.price) ELSE m.price END) * 10 AS total_points
FROM sales AS s
INNER JOIN menu AS m ON s.product_id = m.product_id
GROUP BY s.customer_id;
```

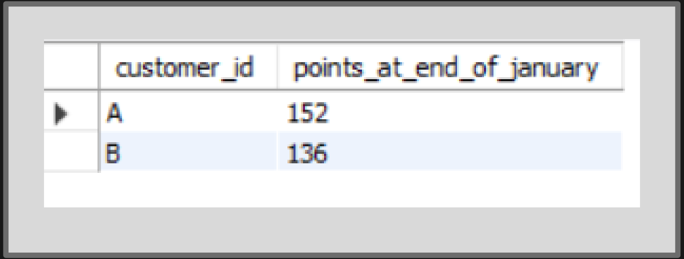


customer_id	total_points
A	860
B	940
C	360

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?



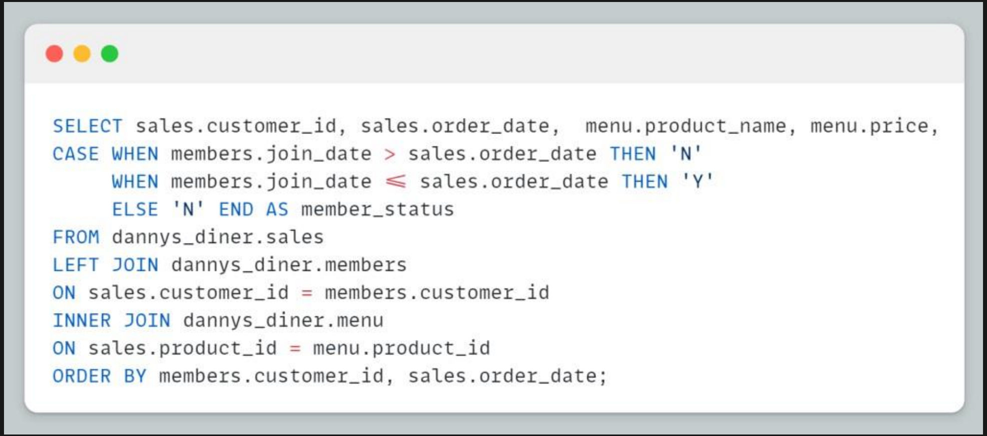
```
WITH joined_customers AS (SELECT m.customer_id AS customer_id,m.join_date
FROM members AS m WHERE m.customer_id IN ('A', 'B')),
points_earned AS (SELECT s.customer_id,
SUM( CASE WHEN s.order_date <= DATE_ADD(jc.join_date, INTERVAL 7 DAY) THEN (2 * m.price)
ELSE m.price END) AS total_points
FROM sales AS s INNER JOIN menu AS m ON s.product_id = m.product_id
INNER JOIN joined_customers AS jc ON s.customer_id = jc.customer_id
WHERE s.order_date <= DATE_ADD(jc.join_date, INTERVAL 1 MONTH) -- End of January
GROUP BY s.customer_id)

SELECT jc.customer_id,COALESCE(pe.total_points, 0) AS points_at_end_of_january
FROM joined_customers AS jc
LEFT JOIN points_earned AS pe ON jc.customer_id = pe.customer_id;
```



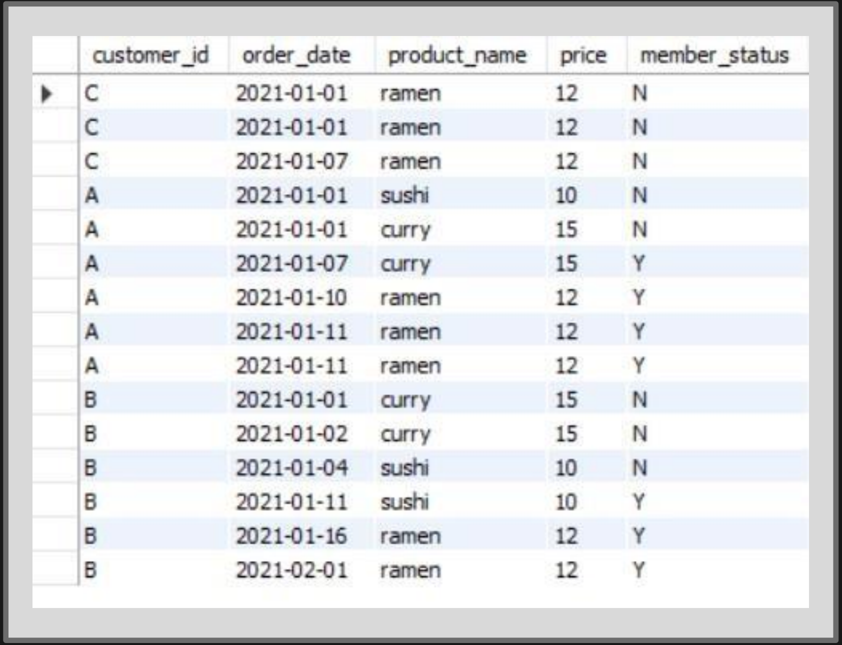
customer_id	points_at_end_of_january
A	152
B	136

Bonus Question

Join All The ThingsRecreate the table with: customer_id, order_date, product_name, price, member (Y/N)



```
SELECT sales.customer_id, sales.order_date,  menu.product_name, menu.price,
CASE WHEN members.join_date > sales.order_date THEN 'N'
     WHEN members.join_date <= sales.order_date THEN 'Y'
     ELSE 'N' END AS member_status
FROM dannys_diner.sales
LEFT JOIN dannys_diner.members
ON sales.customer_id = members.customer_id
INNER JOIN dannys_diner.menu
ON sales.product_id = menu.product_id
ORDER BY members.customer_id, sales.order_date;
```



customer_id	order_date	product_name	price	member_status
C	2021-01-01	ramen	12	N
C	2021-01-01	ramen	12	N
C	2021-01-07	ramen	12	N
A	2021-01-01	sushi	10	N
A	2021-01-01	curry	15	N
A	2021-01-07	curry	15	Y
A	2021-01-10	ramen	12	Y
A	2021-01-11	ramen	12	Y
A	2021-01-11	ramen	12	Y
B	2021-01-01	curry	15	N
B	2021-01-02	curry	15	N
B	2021-01-04	sushi	10	N
B	2021-01-11	sushi	10	Y
B	2021-01-16	ramen	12	Y
B	2021-02-01	ramen	12	Y

THANK YOU