8WEEKSQLCHALLENGE.COM
CASE STUDY #1
DANNY'S DINER
THE TASTE OF SUCCESS
DATAWITHDANNY.COM

Case Study #1 of 8 Week SQL Challenge

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.

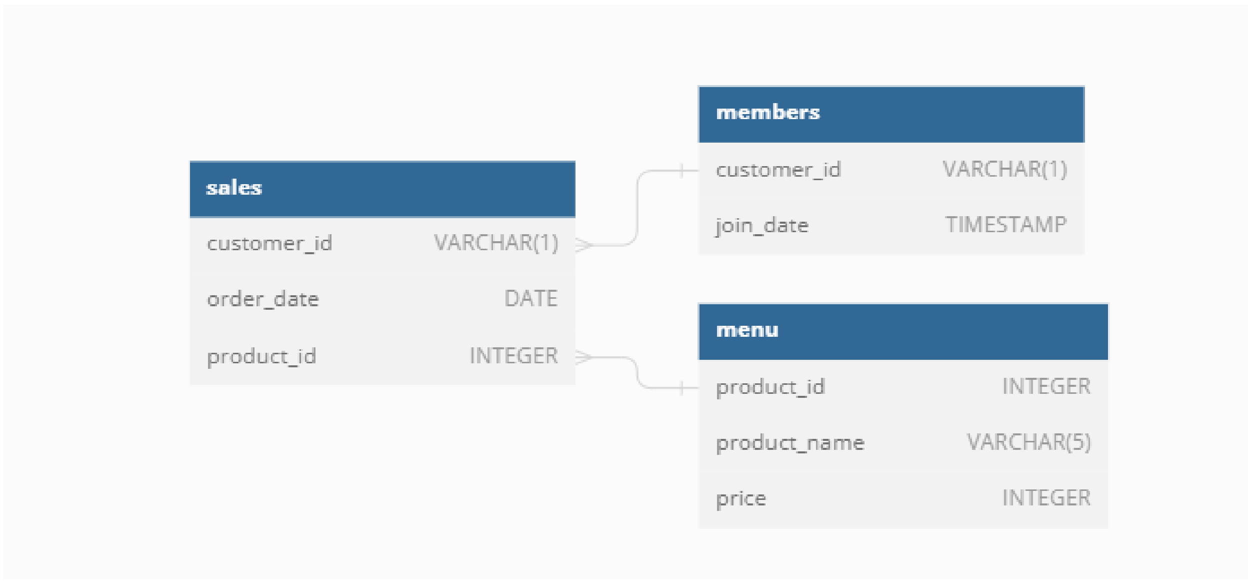
Danny has provided you with a sample of his overall customer data due to privacy issues - but he hopes that these examples are enough for you to write fully functioning SQL queries to help him answer his questions!

Danny has shared with you 3 key datasets for this case study:

- `sales`
- `menu`
- `members`

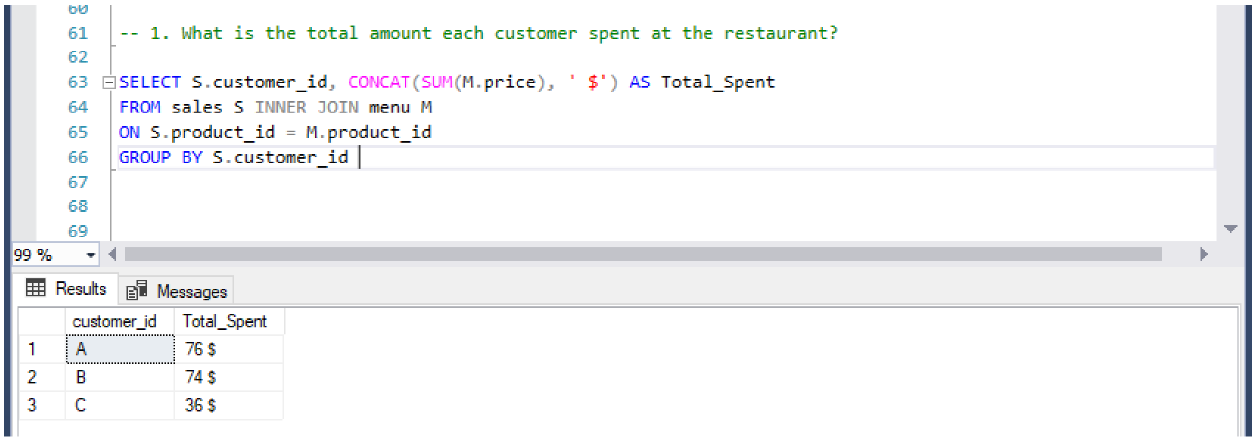
You can inspect the entity relationship diagram and example data below.

Entity Relationship Diagram



Case Study Questions.

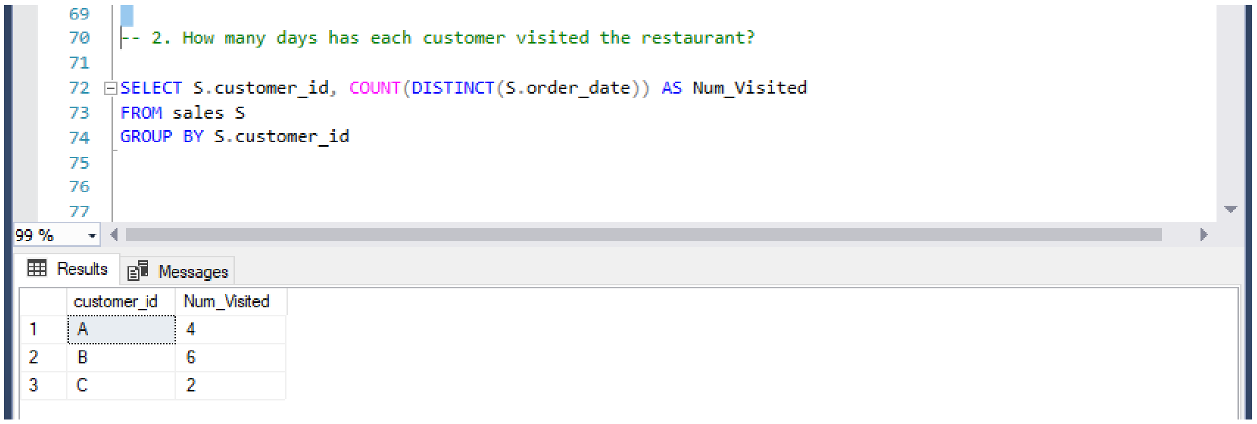
1. What is the total amount each customer spent at the restaurant?



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

SELECT S.customer_id, CONCAT(SUM(M.price), ' $') AS Total_Spent
FROM sales S INNER JOIN menu M
ON S.product_id = M.product_id
GROUP BY S.customer_id
```

	customer_id	Total_Spent
1	A	76 $
2	B	74 $
3	C	36 $

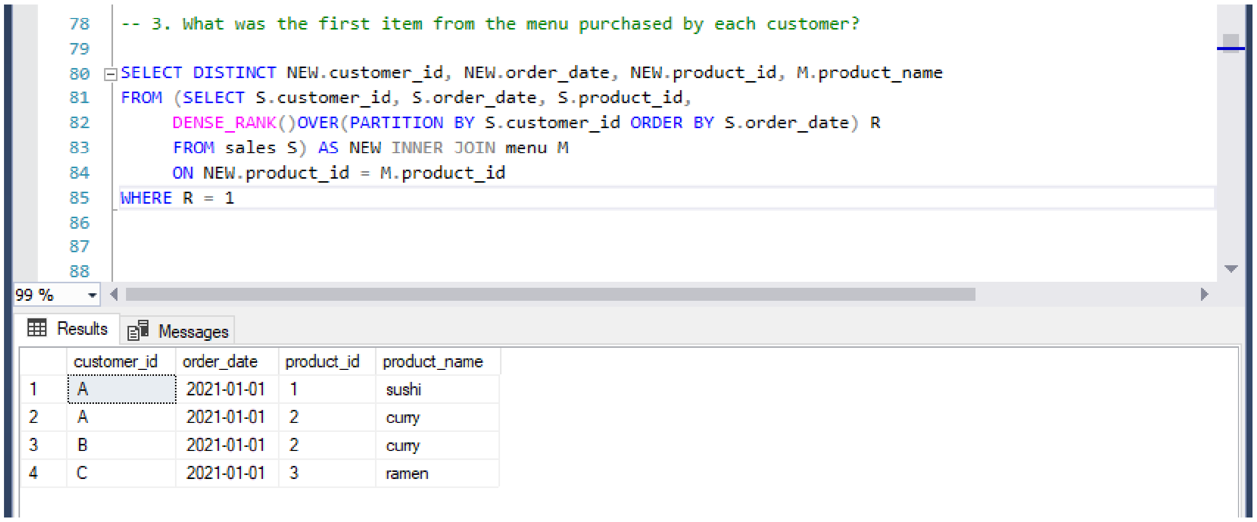
2. How many days has each customer visited the restaurant?



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

SELECT S.customer_id, COUNT(DISTINCT(S.order_date)) AS Num_Visited
FROM sales S
GROUP BY S.customer_id
```

	customer_id	Num_Visited
1	A	4
2	B	6
3	C	2

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



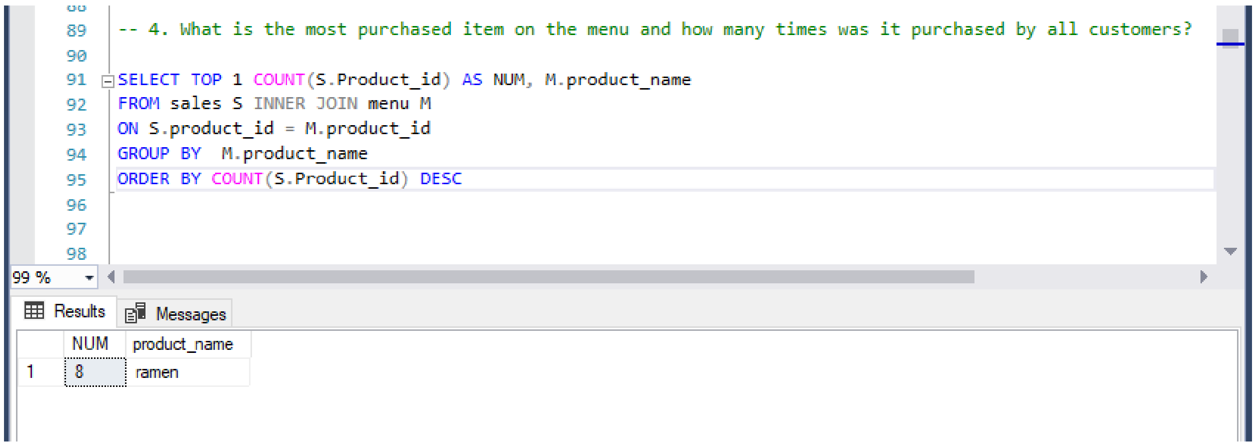
```sql
-- 3. What was the first item from the menu purchased by each customer?

SELECT DISTINCT NEW.customer_id, NEW.order_date, NEW.product_id, M.product_name
FROM (SELECT S.customer_id, S.order_date, S.product_id,
     DENSE_RANK()OVER(PARTITION BY S.customer_id ORDER BY S.order_date) R
     FROM sales S) AS NEW INNER JOIN menu M
     ON NEW.product_id = M.product_id
WHERE R = 1
```

99 %

Results Messages

	customer_id	order_date	product_id	product_name
1	A	2021-01-01	1	sushi
2	A	2021-01-01	2	curry
3	B	2021-01-01	2	curry
4	C	2021-01-01	3	ramen

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



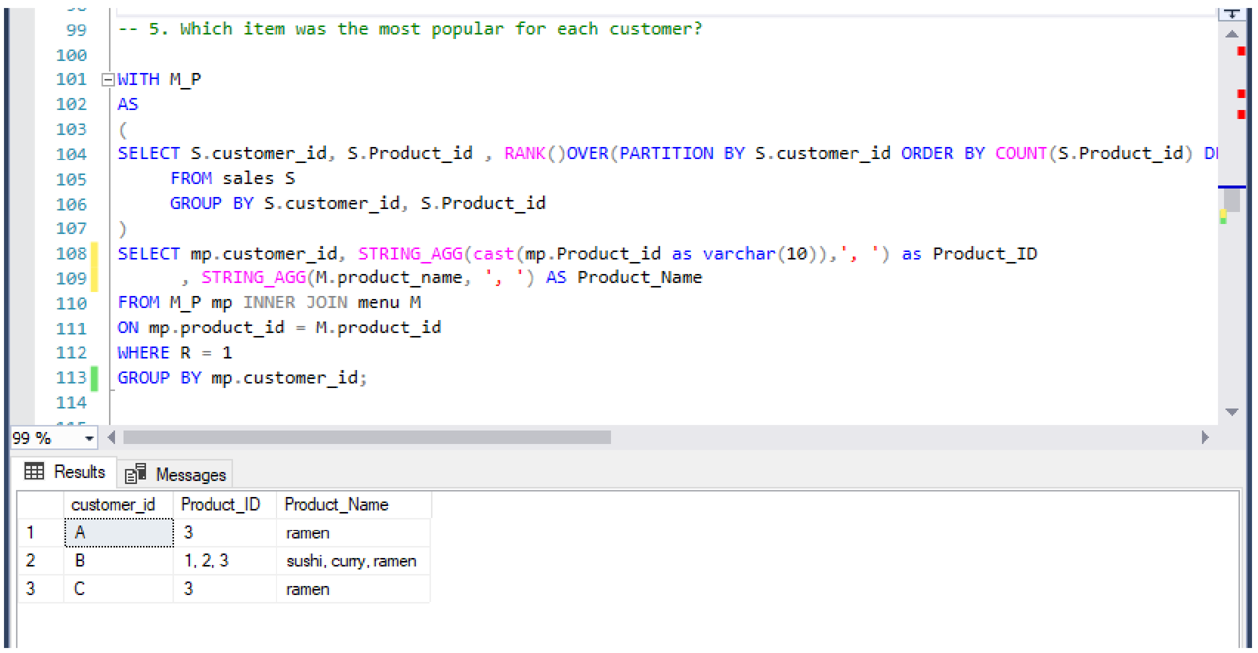
```sql
-- 4. What is the most purchased item on the menu and how many times was it purchased by all customers?

SELECT TOP 1 COUNT(S.Product_id) AS NUM, M.product_name
FROM sales S INNER JOIN menu M
ON S.product_id = M.product_id
GROUP BY  M.product_name
ORDER BY COUNT(S.Product_id) DESC
```

99 %

Results Messages

	NUM	product_name
1	8	ramen

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



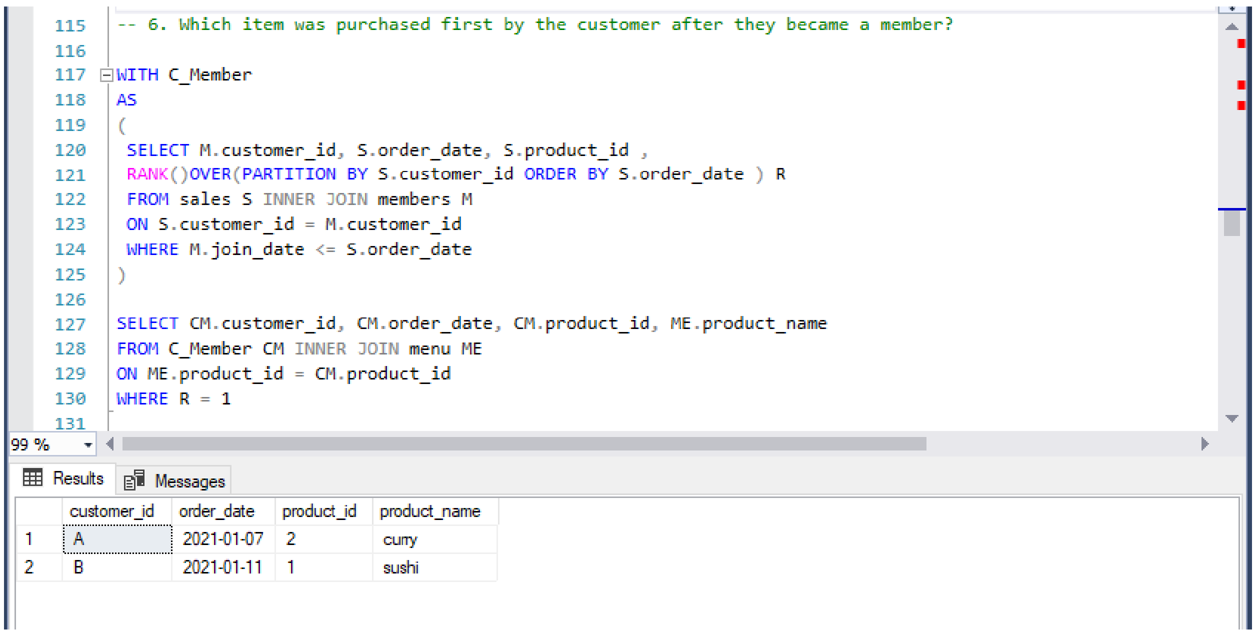
```
-- 5. Which item was the most popular for each customer?

WITH M_P
AS
(
SELECT S.customer_id, S.Product_id , RANK()OVER(PARTITION BY S.customer_id ORDER BY COUNT(S.Product_id) DE
      FROM sales S
      GROUP BY S.customer_id, S.Product_id
)
SELECT mp.customer_id, STRING_AGG(cast(mp.Product_id as varchar(10)),', ') as Product_ID
      , STRING_AGG(M.product_name, ', ') AS Product_Name
FROM M_P mp INNER JOIN menu M
ON mp.product_id = M.product_id
WHERE R = 1
GROUP BY mp.customer_id;
```

99 %

Results Messages

	customer_id	Product_ID	Product_Name
1	A	3	ramen
2	B	1, 2, 3	sushi, curry, ramen
3	C	3	ramen

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



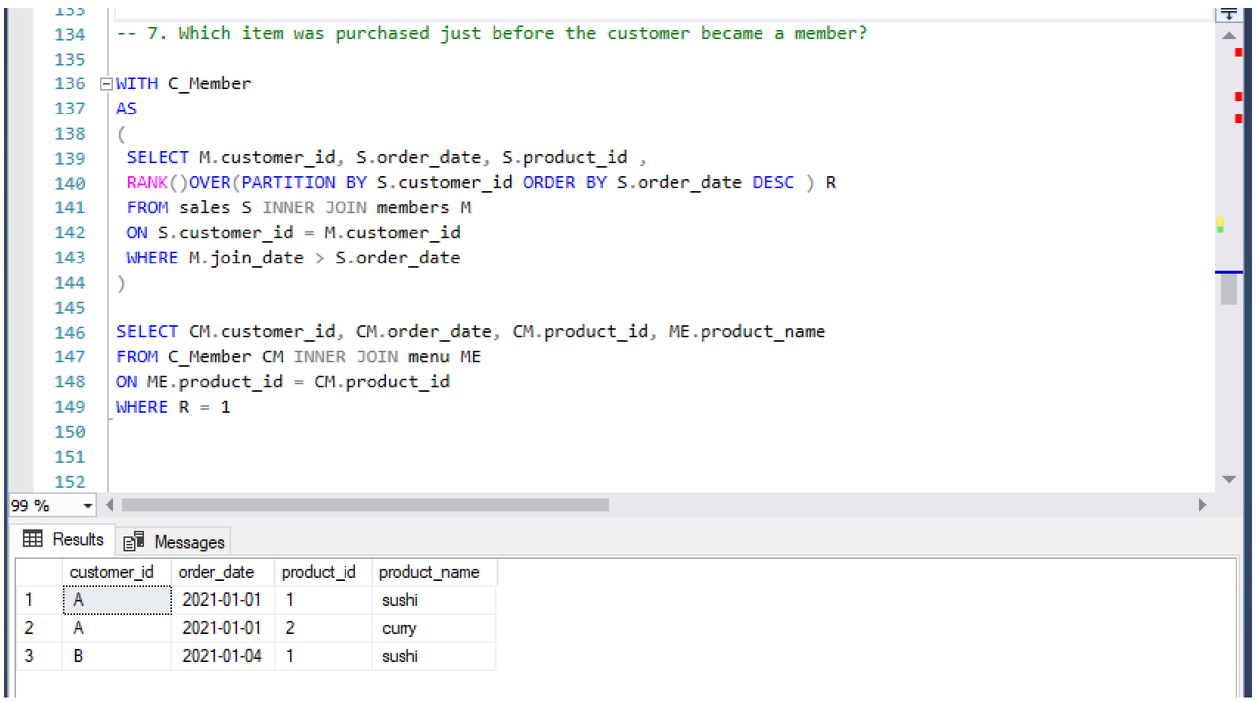
```
-- 6. Which item was purchased first by the customer after they became a member?

WITH C_Member
AS
(
 SELECT M.customer_id, S.order_date, S.product_id ,
 RANK()OVER(PARTITION BY S.customer_id ORDER BY S.order_date ) R
 FROM sales S INNER JOIN members M
 ON S.customer_id = M.customer_id
 WHERE M.join_date <= S.order_date
)

SELECT CM.customer_id, CM.order_date, CM.product_id, ME.product_name
FROM C_Member CM INNER JOIN menu ME
ON ME.product_id = CM.product_id
WHERE R = 1
```

99 %

Results Messages

	customer_id	order_date	product_id	product_name
1	A	2021-01-07	2	curry
2	B	2021-01-11	1	sushi

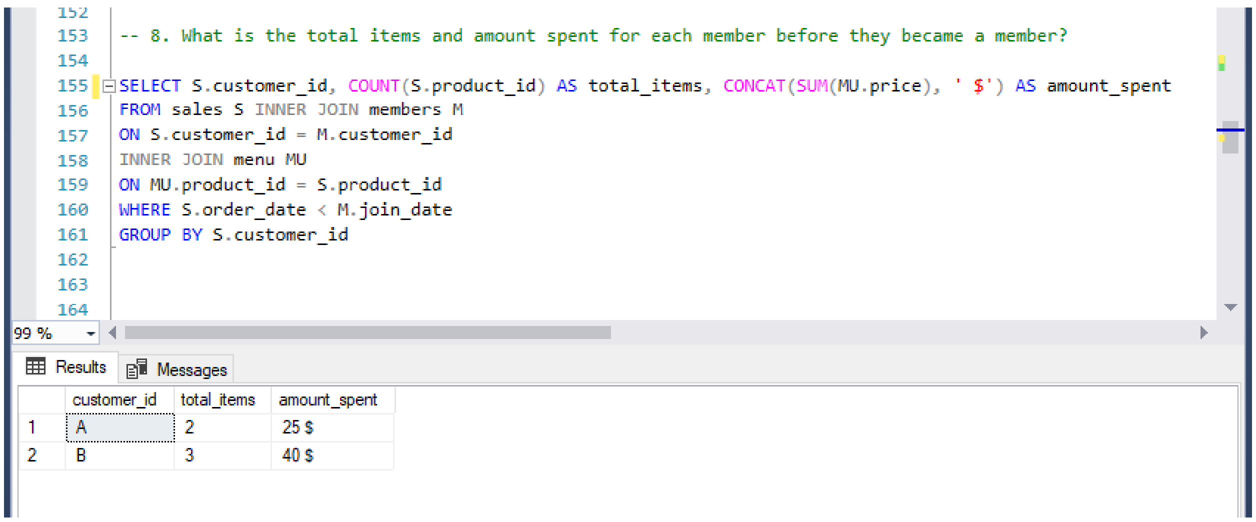
7. Which item was purchased just before the customer became a member?



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

WITH C_Member
AS
(
 SELECT M.customer_id, S.order_date, S.product_id ,
 RANK()OVER(PARTITION BY S.customer_id ORDER BY S.order_date DESC ) R
 FROM sales S INNER JOIN members M
 ON S.customer_id = M.customer_id
 WHERE M.join_date > S.order_date
)

SELECT CM.customer_id, CM.order_date, CM.product_id, ME.product_name
FROM C_Member CM INNER JOIN menu ME
ON ME.product_id = CM.product_id
WHERE R = 1
```

	customer_id	order_date	product_id	product_name
1	A	2021-01-01	1	sushi
2	A	2021-01-01	2	curry
3	B	2021-01-04	1	sushi

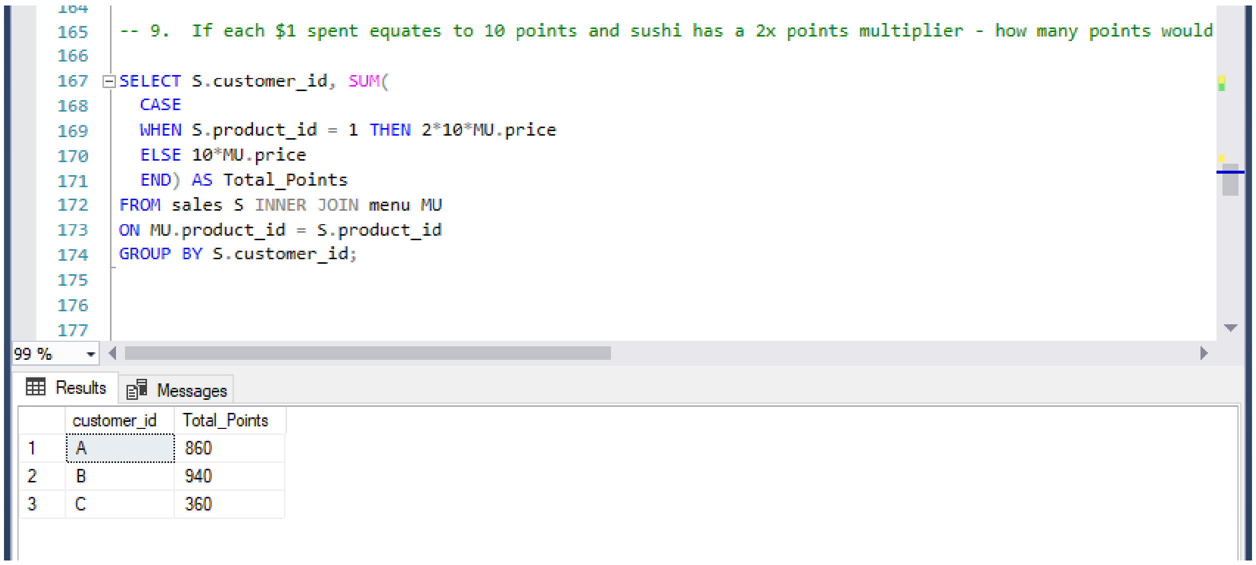
8. What is the total items and amount spent for each member before they became a member?



```
-- 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, CONCAT(SUM(MU.price), ' $') AS amount_spent
FROM sales S INNER JOIN members M
ON S.customer_id = M.customer_id
INNER JOIN menu MU
ON MU.product_id = S.product_id
WHERE S.order_date < M.join_date
GROUP BY S.customer_id
```

	customer_id	total_items	amount_spent
1	A	2	25 $
2	B	3	40 $

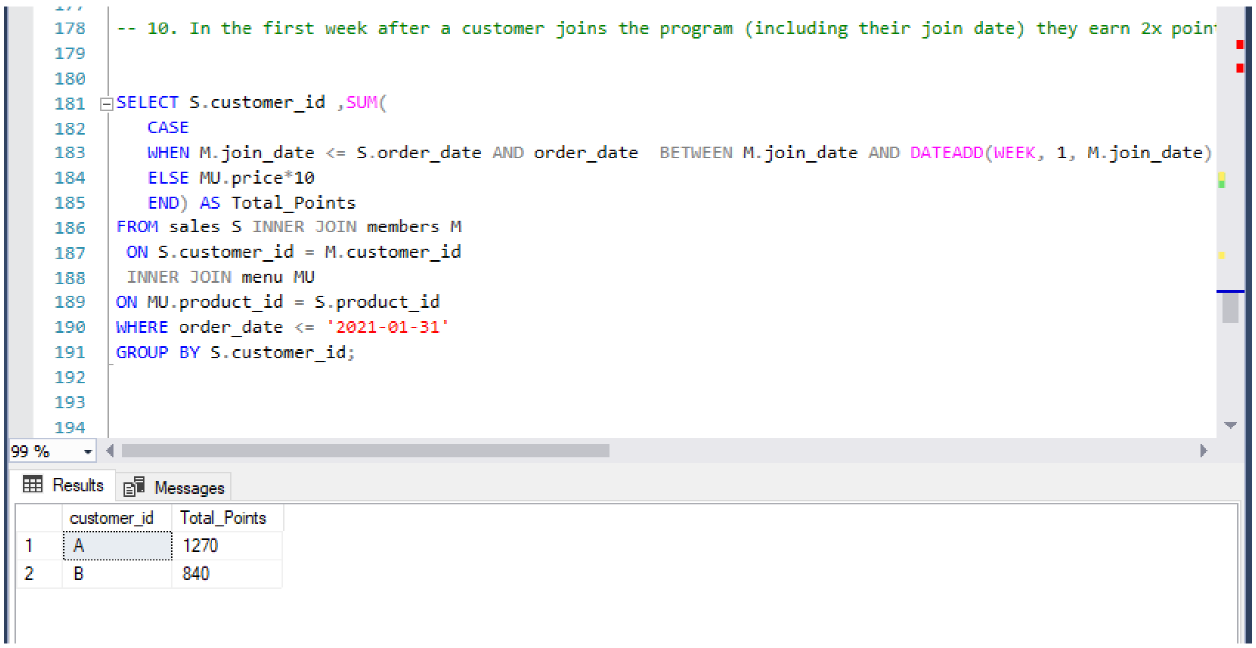
9. If each $1 spent equates to 10 points and sushi has a 2x points multiplier - how many points would each customer have?



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

SELECT S.customer_id, SUM(
   CASE
   WHEN S.product_id = 1 THEN 2*10*MU.price
   ELSE 10*MU.price
   END) AS Total_Points
FROM sales S INNER JOIN menu MU
ON MU.product_id = S.product_id
GROUP BY S.customer_id;
```

	customer_id	Total_Points
1	A	860
2	B	940
3	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?



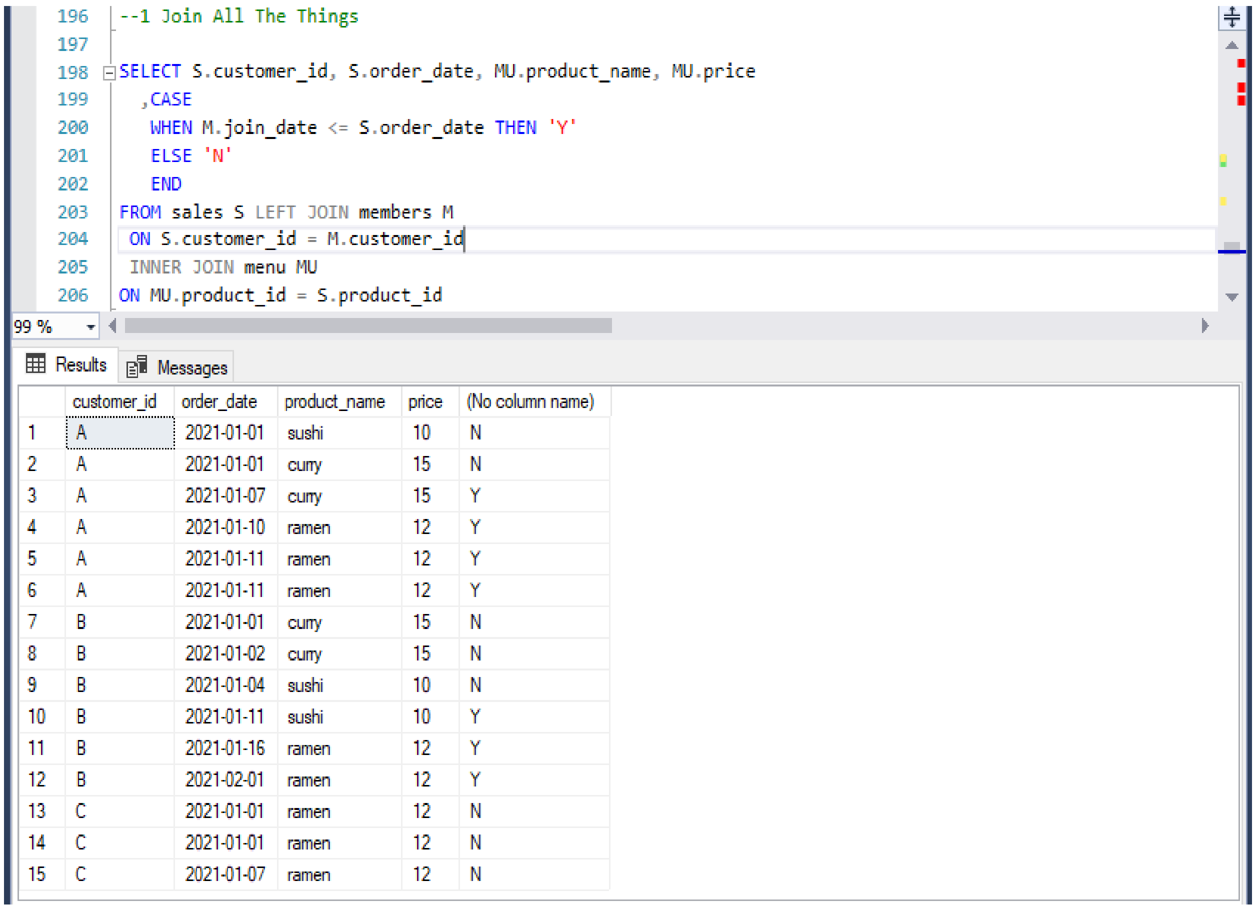
```
-- 10. In the first week after a customer joins the program (including their join date) they earn 2x poin

SELECT S.customer_id ,SUM(
    CASE
    WHEN M.join_date <= S.order_date AND order_date  BETWEEN M.join_date AND DATEADD(WEEK, 1, M.join_date)
    ELSE MU.price*10
    END) AS Total_Points
FROM sales S INNER JOIN members M
 ON S.customer_id = M.customer_id
 INNER JOIN menu MU
ON MU.product_id = S.product_id
WHERE order_date <= '2021-01-31'
GROUP BY S.customer_id;
```

	customer_id	Total_Points
1	A	1270
2	B	840

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.



```sql
--1 Join All The Things

SELECT S.customer_id, S.order_date, MU.product_name, MU.price
  ,CASE
   WHEN M.join_date <= S.order_date THEN 'Y'
   ELSE 'N'
   END
FROM sales S LEFT JOIN members M
 ON S.customer_id = M.customer_id
 INNER JOIN menu MU
ON MU.product_id = S.product_id
```

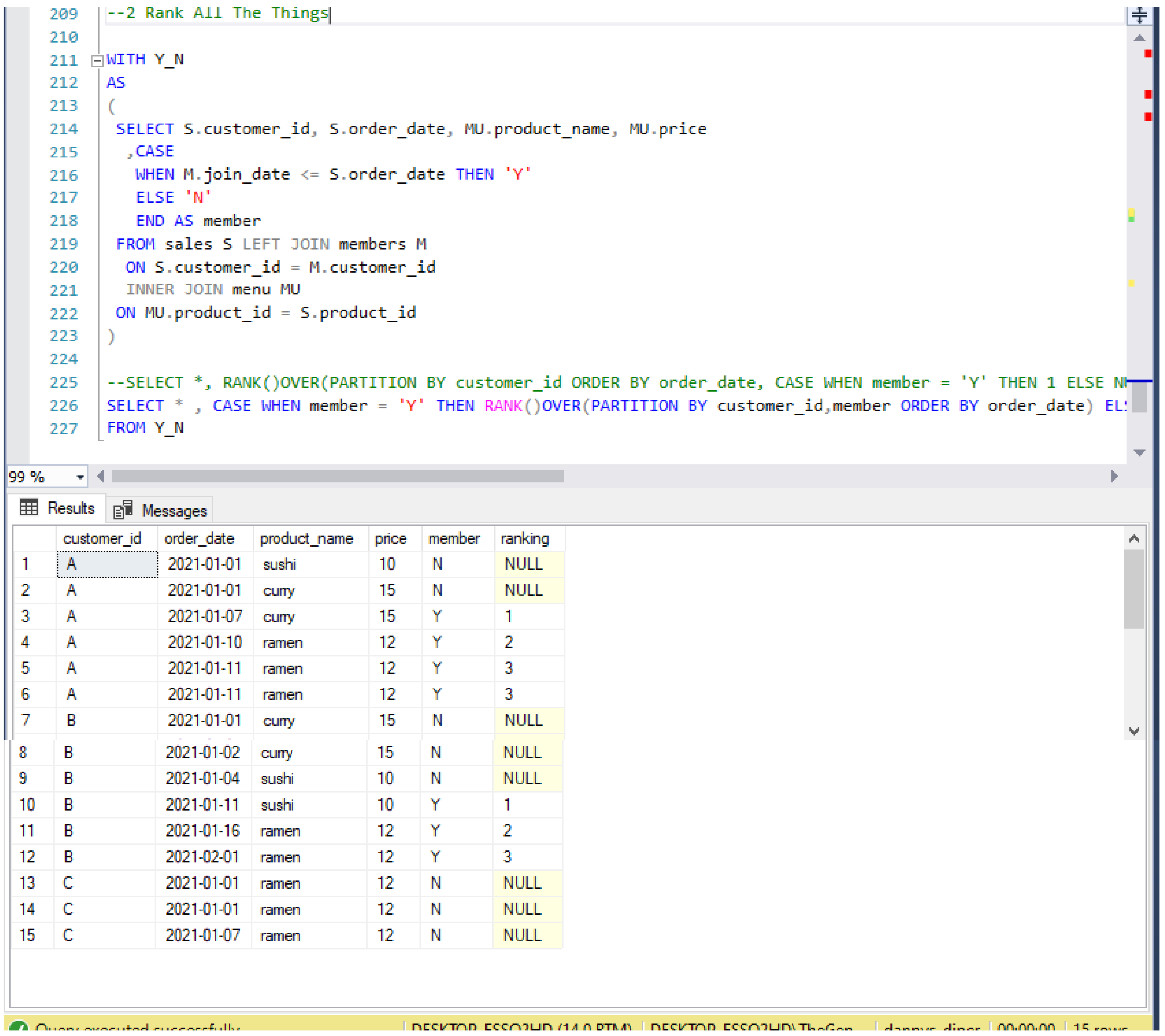
99 %

Results Messages

	customer_id	order_date	product_name	price	(No column name)
1	A	2021-01-01	sushi	10	N
2	A	2021-01-01	curry	15	N
3	A	2021-01-07	curry	15	Y
4	A	2021-01-10	ramen	12	Y
5	A	2021-01-11	ramen	12	Y
6	A	2021-01-11	ramen	12	Y
7	B	2021-01-01	curry	15	N
8	B	2021-01-02	curry	15	N
9	B	2021-01-04	sushi	10	N
10	B	2021-01-11	sushi	10	Y
11	B	2021-01-16	ramen	12	Y
12	B	2021-02-01	ramen	12	Y
13	C	2021-01-01	ramen	12	N
14	C	2021-01-01	ramen	12	N
15	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.



```sql
--2 Rank All The Things

WITH Y_N
AS
(
 SELECT S.customer_id, S.order_date, MU.product_name, MU.price
  ,CASE
   WHEN M.join_date <= S.order_date THEN 'Y'
   ELSE 'N'
   END AS member
 FROM sales S LEFT JOIN members M
  ON S.customer_id = M.customer_id
  INNER JOIN menu MU
 ON MU.product_id = S.product_id
)

--SELECT *, RANK()OVER(PARTITION BY customer_id ORDER BY order_date, CASE WHEN member = 'Y' THEN 1 ELSE N
SELECT * , CASE WHEN member = 'Y' THEN RANK()OVER(PARTITION BY customer_id,member ORDER BY order_date) EL
FROM Y_N
```

99 %

Results Messages

	customer_id	order_date	product_name	price	member	ranking
1	A	2021-01-01	sushi	10	N	NULL
2	A	2021-01-01	curry	15	N	NULL
3	A	2021-01-07	curry	15	Y	1
4	A	2021-01-10	ramen	12	Y	2
5	A	2021-01-11	ramen	12	Y	3
6	A	2021-01-11	ramen	12	Y	3
7	B	2021-01-01	curry	15	N	NULL
8	B	2021-01-02	curry	15	N	NULL
9	B	2021-01-04	sushi	10	N	NULL
10	B	2021-01-11	sushi	10	Y	1
11	B	2021-01-16	ramen	12	Y	2
12	B	2021-02-01	ramen	12	Y	3
13	C	2021-01-01	ramen	12	N	NULL
14	C	2021-01-01	ramen	12	N	NULL
15	C	2021-01-07	ramen	12	N	NULL

Query executed successfully. | DESKTOP-ESSO2HD (14.0 RTM) | DESKTOP-ESSO2HD\TheGen... | dannys_diner | 00:00:00 | 15 rows