

SQL Capstone 1 by John Castro

Introduction

A local employee seriously loves Japanese food so, at the beginning of 2021, he decides to embark upon a risky venture. He opens up a cute little restaurant that sells his 3 favorite foods: sushi, curry, and ramen.

Tukuyomi's Diner needs your assistance to help the restaurant stay afloat — it has captured some fundamental data from its few months of operation. Still, it has no idea how to use its 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*
- *which menu items are their favorite.*

This more profound 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 essential datasets so his team can quickly inspect the data without needing to use SQL.

The data set contains the following 3 tables, which you may refer to the relationship diagram below to understand the connection.

sales

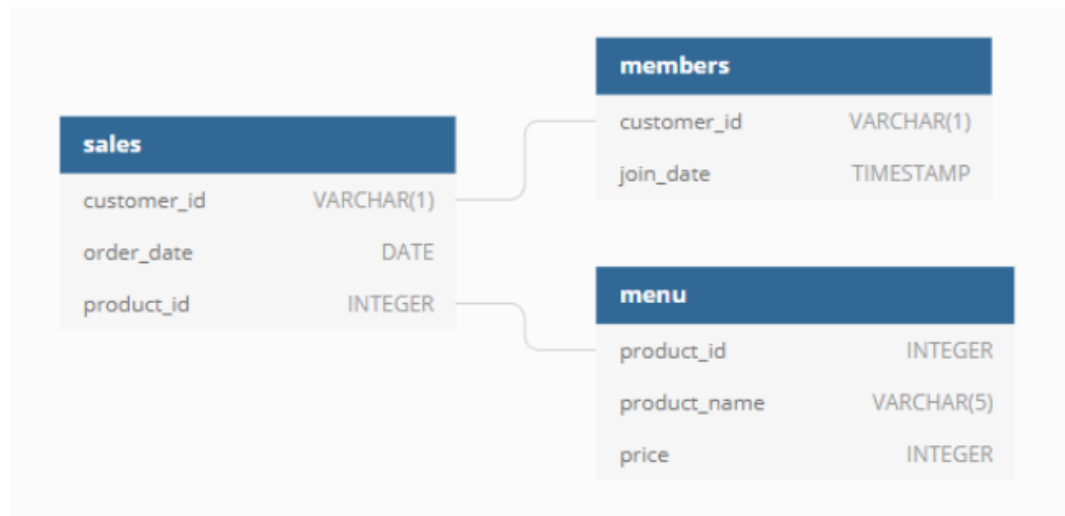
members

menu

Table Relationship

Table Relationship

Entity Relationship Diagram



Case Study Questions

1. How much did each diner spend overall at the establishment?
2. How many times has each person dine-in at the restaurant?
3. Which dish did each customer order first from the menu?
4. What item on the menu is the most popular, and how frequently do all customers order it?
5. Which product was the most well-liked by each customer?
6. After becoming a member, which product was initially bought by the customer?
7. What was the last thing the client bought before signing up for membership?
8. Before becoming a member, how much money and what things did each member spend in total?
9. How many points would each client have if every dollar spent was worth ten points and sushi had a two-fold points multiplier?

Submission Guidelines

1. Per each question, provide the SQL Code and the sample screenshot
2. Provide a 1-3 sentence discussion of each answer to the question

1. How much did each diner spend overall at the establishment?

Query

Query History

1

2

3

4

5

6

7

8

9

10

SELECT

sales.customer_id

AS

diner,

SUM(menu.price)

AS

overall_spent

FROM

tukuyomi_diner.menu

JOIN

tukuyomi_diner.sales

ON

menu.product_id=sales.product_id

GROUP BY

sales.customer_id

ORDER BY

2

DESC;

Data Output

Messages

Notifications

diner

character varying (1)

overall_spent

bigint

1

A

76

2

B

74

3

C

36

Total rows: 3 of 3

Query complete 00:00:00.049

The query above shows the total amount of money spent by each customer irrespective of membership for the specific period. Customer A spent the most with 76 units, followed by customer B with 74, and lastly by customer C with 36. We can say that both customers A and B are virtually equal in being the most frequent diners among the three customers. This data can hint about their enjoyment of the dishes and future dining behavior.

2. How many times has each person dine-in at the restaurant?

Query

Query History

1

2

3

4

5

6

7

SELECT

sales.customer_id

AS

customer,

COUNT(sales.order_date)

AS

dine_count

FROM

tukuyomi_diner.sales

GROUP BY

customer_id

ORDER BY

COUNT(sales.order_date)

DESC;

Data Output

Messages

Notifications

≡+

📄

▼

📋

🗑️

🗄️

⬇️

📈

	customer character varying (1) 🔒	dine_count bigint 🔒
1	B	6
2	A	6
3	C	3

Total rows: 3 of 3

Query complete 00:00:00.055

The query above shows dine-count by customer. Both customers A and B both have the highest number of dine-counts of 6 times. Customer C only had 3. This again can be helpful data in projecting future dining behavior of these customers.

3. Which dish did each customer order first from the menu?

Query

Query History

1

2

3

4

5

6

7

8

9

10

SELECT

sales.customer_id,

menu.product_name,

MIN(sales.order_date) AS order_date

FROM tukuyomi_diner.menu

JOIN tukuyomi_diner.sales

ON menu.product_id=sales.product_id

GROUP BY 1, 2

ORDER BY 3 ASC

LIMIT 4;

Data Output

Messages

Notifications

≡+

	customer_id <div>character varying (1)</div> <div></div>	product_name <div>character varying (5)</div> <div></div>	order_date <div>date</div> <div></div>
1	A	sushi	2021-01-01
2	A	curry	2021-01-01
3	C	ramen	2021-01-01
4	B	curry	2021-01-01

Total rows: 4 of 4

Query complete 00:00:00.107

The query selects the first time a customer buys a dish. We can see that customer A ordered both sushi and curry, customer B ordered curry, and customer C ordered ramen as their first dishes. This shows that more customers want to test out curry first as their introductory dish in the restaurant. But we might need to have more data to test correlation.

4. What item on the menu is the most popular, and how frequently do all customers order it?

Query

Query History

1

2

3

4

5

6

7

8

9

10

11









12

```
SELECT menu.product_name,  
       COUNT(sales.order_date) AS times_ordered  
FROM   tukuyomi_diner.menu  
JOIN   tukuyomi_diner.sales  
       ON menu.product_id=sales.product_id  
GROUP BY 1  
ORDER BY 2 DESC;
```

Data Output

Messages

Notifications



	product_name character varying (5)	times_ordered bigint
1	ramen	8
2	curry	4
3	sushi	3

Total rows: 3 of 3

Query complete 00:00:00.054

The query above shows the most famous dishes based on order frequency. Ramen is the most famous with a count of 8 orders, followed by curry with 4, and lastly by sushi with 3. The owner might want to create more and various ways to cater ramen to potentially encourage customer following and profit.

5. Which product was the most well-liked by each customer?

Query Query History

```
1 SELECT DISTINCT sales.customer_id,
2 menu.product_name,
3 COUNT(sales.product_id) AS order_count
4 FROM tukuyomi_diner.menu
5 JOIN tukuyomi_diner.sales
6 ON menu.product_id=sales.product_id
7 GROUP BY 1, 2
8 ORDER BY 3 DESC
9 LIMIT 5;
10
```

Data Output Messages Notifications

	customer_id character varying (1)	product_name character varying (5)	order_count bigint
1	C	ramen	3
2	A	ramen	3
3	B	sushi	2
4	B	ramen	2
5	B	curry	2

Total rows: 5 of 5 Query complete 00:00:00.108

The query above shows the most frequently ordered menu item by each customer to signify preference. All customers ordered ramen the most, where both A and C bought it three times each. Customer B had ramen tied with sushi and curry. It seems that if a customer has a great preference for a dish, it would most probably be ramen; else, they might have an eclectic taste for all dishes like customer B. The management can curate their ads and promos to this apparent trend.

6. After becoming a member, which product was initially bought by the customer?

Query

Query History

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

```
SELECT DISTINCT members.customer_id,
                 menu.product_name,
                 members.join_date,
                 MIN(sales.order_date) AS order_date,
                 MIN(sales.order_date-members.join_date) AS MinDateDiff
FROM   tukuyomi_diner.menu
JOIN   tukuyomi_diner.sales
      ON menu.product_id=sales.product_id
JOIN   tukuyomi_diner.members
      ON sales.customer_id = members.customer_id
WHERE  order_date >= join_date
GROUP BY 1, 2, 3
ORDER BY 5 ASC
LIMIT 2;
```

Data Output

Messages

Notifications

≡+

📄

▼

📋

🗑️

🗄️

⬇️

📈

	customer_id character varying (1) 🔒	product_name character varying (5) 🔒	join_date date 🔒	order_date date 🔒	mindatediff integer 🔒
1	A	curry	2021-01-07	2021-01-07	0
2	B	sushi	2021-01-09	2021-01-11	2

Total rows: 2 of 2 Query complete 00:00:00.053

The query above shows the orders after a customer becomes a member (including the same day of becoming a member). As the data are grouped by customer, we see that for customer A's first order since becoming a member was curry. For the other member, customer B, ordered sushi as the first dish since becoming a member. It is interesting that their most preferred dish (ramen) was not ordered first by either member.

7. What was the last thing the client bought before signing up for membership?

Fields required: members.customer_id, members.join_date, sales.order_date, menu.product_name

Query

Query History

```
1 SELECT DISTINCT members.customer_id,
2                 menu.product_name,
3                 MIN(sales.order_date) AS order_date,
4                 members.join_date,
5                 MIN(members.join_date-sales.order_date) AS MinDateDiff
6 FROM tukuyomi_diner.menu
7 JOIN tukuyomi_diner.sales
8     ON menu.product_id=sales.product_id
9 JOIN tukuyomi_diner.members
10    ON sales.customer_id = members.customer_id
11 WHERE order_date <= join_date
12 GROUP BY 1, 2, 4
13 ORDER BY 5 ASC
14 LIMIT 2;
```

Data Output

Messages

Notifications

	customer_id character varying (1)	product_name character varying (5)	order_date date	join_date date	mindatediff integer
1	A	curry	2021-01-01	2021-01-07	0
2	B	sushi	2021-01-04	2021-01-09	5

Total rows: 2 of 2 Query complete 00:00:00.333

The query above shows customer_id, product_name, order_date, join_date, and the minimum date difference as categorizer. All order_dates come before their join_date to be members. Customer A purchased curry before signing up for membership while Customer B ordered sushi. They must have enjoyed the meal and wanted to enjoy member perks as they plan to enjoy the restaurant dishes more in the future. While both customers favored ramen, it may be that they have seen that there are other items they also enjoy that they thought it was worth signing up for membership to enjoy future deals or promos in their return in the future.

8. Before becoming a member, how much money and what things did each member spend in total?

Fields required: members.customer_id, members.join_date, sales.order_date, menu.product_name, SUM(menu.price)

Conditions: order_date < join_date

Query		Query History	
1	SELECT	members.customer_id,	
2		menu.product_name,	
3		SUM(menu.price)	
4	FROM	tukuyomi_diner.menu	
5	JOIN	tukuyomi_diner.sales	
6		ON menu.product_id=sales.product_id	
7	JOIN	tukuyomi_diner.members	
8		ON sales.customer_id = members.customer_id	
9	WHERE	sales.order_date <= members.join_date	
10	GROUP BY	1, 2	
11	ORDER BY	SUM(menu.price) DESC;	

Data Output		Messages	Notifications
<div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div></div>			
	customer_id character varying (1)	product_name character varying (5)	sum bigint
1	B	curry	30
2	A	curry	30
3	B	sushi	10
4	A	sushi	10
Total rows: 4 of 4		Query complete 00:00:00.166	

The query above shows what each member ordered and how much they have spent in each product in total before becoming a member. Both customers A and B have equally spent 30 units of money on curry and 10 units of money on sushi before becoming a member. We can think that this is quite a peculiar pattern for both. If we have more data, this may prove to be key in transforming normal customers into members. Therefore, we suggest that the management feature both curry and sushi that can make the customers order and taste them; and most likely they realize that besides the ramen there are other good foods hence worth the membership sign-up.

Query	Query History
1	SELECT members.customer_id,SUM(menu.price)
2	FROM tukuyomi_diner.menu
3	JOIN tukuyomi_diner.sales
4	ON menu.product_id=sales.product_id
5	JOIN tukuyomi_diner.members
6	ON sales.customer_id = members.customer_id
7	WHERE sales.order_date <= members.join_date
8	GROUP BY 1
9	ORDER BY 1, SUM(menu.price) DESC ;
10	
11	

Data Output	Messages	Notifications
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>		
	customer_id character varying (1) 🔒	sum bigint 🔒
1	A	40
2	B	40

The query above shows the total amount of money each member spent before becoming a member. Both customers A and B paid 40 units of money. We can say that this can be a benchmark price that the management should look into making normal customers spend at par with to make them try a decent amount and variety of dishes while also making them feel invested in the restaurant having paid that much money so that they sign-up for membership soon thereafter.

9. How many points would each client have if every dollar spent was worth ten points and sushi had a two-fold points multiplier?

Fields required: sales.customer_id,, menu.price,

Conditions: case when menu.product_name = 'sushi' then menu.price*2*10

Else menu.price *10

END AS Points

Query

Query History

1

SELECT sales.customer_id,

2

SUM(CASE

3

WHEN menu.product_name = 'sushi' THEN menu.price*2*10

4

ELSE menu.price*10

5

END) AS Points

6

FROM tukuyomi_diner.menu

7

JOIN tukuyomi_diner.sales

8

ON menu.product_id=sales.product_id

9

FULL OUTER JOIN tukuyomi_diner.members

10

ON sales.customer_id = members.customer_id

11

GROUP BY sales.customer_id

12

ORDER BY Points DESC;

Data Output

Messages

Notifications

	customer_id character varying (1)	points bigint
1	B	940
2	A	860
3	C	360

Total rows: 3 of 3

Query complete 00:00:00.662

The query above shows customers' points wherein every dollar they spent is allotted ten points, but ordering sushi will give them twenty points per dollar spent on the dish. In descending order, customer B is at the top with 940 points, followed by customer A with 860 points, and lastly by customer C with 360 points. This pointing system can be used to encourage customers, member or not, to not only patronize the dishes but particularly of sushi. Higher total points in a given period (e.g., a month or quarter) will be given special discounts or other forms of prize to increase customer fidelity/loyalty.