SQL ANALYSIS

DANNIY'S DINER

Presented by Gaurav Thakur

About Danny

A SAVORING HISTORY

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.

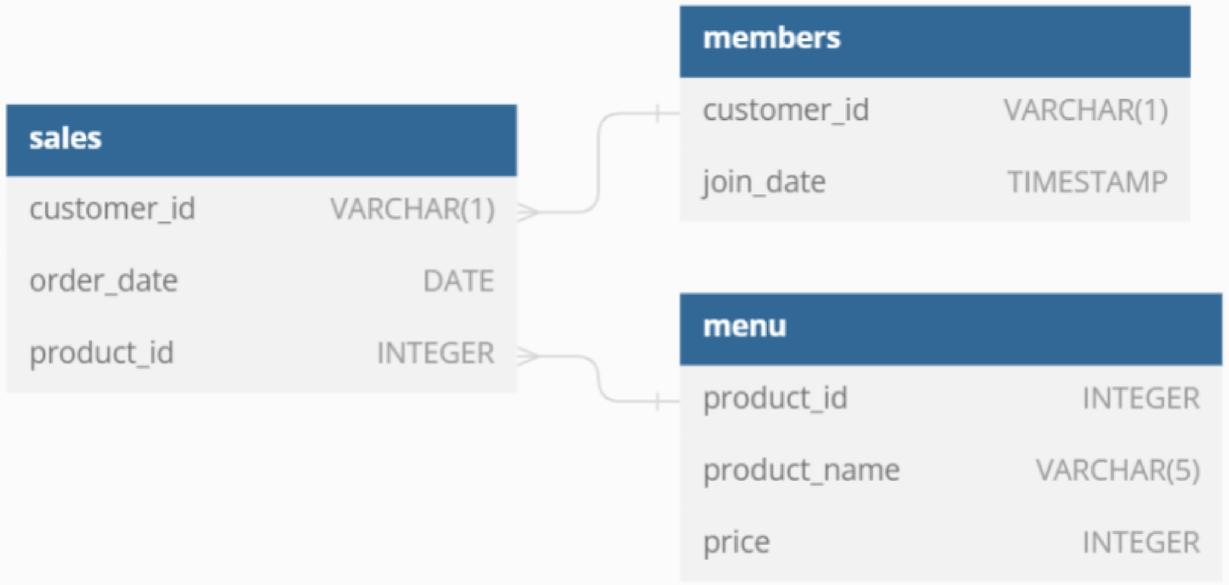
Danny's Diner is in need of your assistance to help the restaurant stay afloat - the restaurant has captured some very basic data from their few months of operation but have no idea how to use their data to help them run the business.

Problem Statement

Danny seeks customer insights for personalized experiences and loyalty program expansion. He plans to decide on program expansion and needs user-friendly datasets for his team. Key datasets: sales, menu, members.

Danny intends to expand the loyalty program with the help of customer insights and user-friendly datasets for his team, avoiding the need for SQL. Given privacy concerns, he has shared a sample of customer data. The three key datasets provided are: sales, menu, and member information.

Entity Relationship Diagram



Query | Query History

```
48  Q2: How many days has each customer visited the restaurant?
49
50  SELECT
51    customer_id, count( distinct order_date ) AS Days_visited
52  FROM dannys_diner.sales
53  GROUP BY customer_id;
54
55
56
```

Data Output | Messages | Notifications

	customer_id character varying (1)	days_visited bigint
1	A	4
2	B	6
3	C	2

Query | Query History

```sql
 Q3: What was the first item from the menu purchased by each customer?

with firstcte 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 first_order
  FROM dannys_diner.sales s
  INNER JOIN dannys_diner.menu m
    ON s.product_id = m.product_id
)

select customer_id,
    order_date,
    product_name from firstcte
where first_order = 1

```

Data Output | Messages | Notifications

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

Query | Query History

```sql
 Q4: What is the most purchased item on the menu and how many times was it purchased by all customers?

with most_ordered as (
select product_id
from dannys_diner.sales
group by product_id
order by count(*) desc
limit 1
),
ordered_count as
(
    select customer_id, product_id , count(*) as no_order
from dannys_diner.sales
where product_id in (select product_id from most_ordered )
group by customer_id, product_id

)
select o.customer_id, o.product_id, m.product_name, o.no_order
from ordered_count o
join dannys_diner.menu m on m.product_id = o.product_id
```

Data Output | Messages | Notifications

	customer_id character varying (1)	product_id integer	product_name character varying (5)	no_order bigint
1	C	3	ramen	3
2	B	3	ramen	2
3	A	3	ramen	3

Query | Query History

```
121
122
123
124   Q5: Which item was the most popular for each customer?
125
126  with rankedorder as (
127  select customer_id, product_id, count(*) as no_order,
128  dense_rank() over (partition by customer_id order by count(*) desc) as rn
129  from dannys_diner.sales
130  group by customer_id, product_id
131
132  )
133  select r.customer_id, m.product_name,r.no_order from rankedorder r
134  join dannys_diner.menu m on m.product_id = r.product_id
135  where r.rn = 1
136  order by r.no_order desc
137
138
139
140
141
```

Data Output | Messages | Notifications

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

Query | Query History

```
143
144  Q6. Which item was purchased first by the customer after they became a member?
145
146 with cte as (
147   select
148     members.customer_id,
149     sales.product_id,
150    row_number() over (
151       partition by members.customer_id
152       order by sales.order_date) rn
153   from dannys_diner.members
154   inner join dannys_diner.sales on members.customer_id = sales.customer_id
155     and sales.order_date > members.join_date )
156
157 select
158   customer_id,
159   product_name
160 from cte
161 inner join dannys_diner.menu
162   on cte.product_id = menu.product_id
163 where rn = 1
164 order by customer_id;
165
```

Data Output | Messages | Notifications

	customer_id character varying (1)	product_name character varying (5)
1	A	ramen
2	B	sushi

Query | Query History

```
161
162
163   Q7: Which item was purchased just before the customer became a member?
164
165
166  with cte as (
167  select m.customer_id , s.product_id,
168  row_number() over( partition by m.customer_id order by s.order_date desc) rn
169  from dannys_diner.members m
170  inner join dannys_diner.sales s on m.customer_id = s.customer_id
171  and s.order_date < m.join_date )
172
173
174  select customer_id, product_name
175  from cte
176  inner join dannys_diner.menu on cte.product_id = menu.product_id
177  where rn = 1
178  order by customer_id;
179
```

Data Output | Messages | Notifications

	customer_id character varying (1)	product_name character varying (5)
1	A	sushi
2	B	sushi

Query | Query History

```
185
186  Q8. What is the total items and amount spent for each member before they became a member?
187
188  select s.customer_id as Customers_Id, count (s.product_id), concat( '$', sum( m.price )) as amt_spend
189  from dannys_diner.sales s
190  join dannys_diner.members me on s.customer_id = me.customer_id and s.order_date < me.join_date
191  join dannys_diner.menu m on s.product_id = m.product_id
192  group by s.customer_id
193  order by sum(m.price) desc
194
195
196
197
```

Data Output | Messages | Notifications

	customers_id character varying (1)	count bigint	amt_spend text
1	B	3	$40
2	A	2	$25

Query | Query History

```
196
197
198
199
200  Q9: If each $1 spent equates to 10 points and sushi has a 2x points multiplier - how many points would each customer have?
201
202 with cte as (
203   select m.product_id,
204     case when product_id = 1 then price * 20 else price * 10
205     end as points
206 from dannys_diner.menu m
207 )
208
209 SELECT s.customer_id, sum(cte.points) as total_points from dannys_diner.sales s
210 join cte on s.product_id = cte.product_id
211 group by s.customer_id
212 order by s.customer_id;
213
```

Data Output | Messages | Notifications

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