Case Study #1 - Danny's Diner

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

Example Datasets

All datasets exist within the `dannys_diner` database schema - be sure to include this reference within your SQL scripts as you start exploring the data and answering the case study questions.

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

```
CREATE TABLE sales (
  customer_id VARCHAR(1),
  order_date DATE,
  product_id INTEGER
);

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');


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');
```

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

```
select s.customer_id, sum(m.price) as Total_Amount from sales s join menu m
on s.product_id = m.product_id
group by s.customer_id;
```

Output:

```
customer_id Total_Amount
----------- ------------
A           76
B           74
C           36
```

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

```
select customer_id, count(distinct order_date) as visited from sales group by customer_id;
```

Output:

```
customer_id visited
----------- -----------
A           4
B           6
C           2
```

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

```
with first_item as(
select s.customer_id, m.product_id, m.product_name,
```

```
row_number() over(partition by s.customer_id order by s.customer_id) as
first_item_purchased
from sales s join menu m on s.product_id = m.product_id
)
select customer_id, product_id, product_name from first_item where first_item_purchased
= 1;
```

Output:

```
customer_id product_id  product_name
----------- ----------- ------------
A                   1 sushi
B                   2 curry
C                   3 ramen
```

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

```
select top 1 m.product_name, count(s.product_id) AS total_purchases from sales s join
menu m
on s.product_id = m.product_id
group by product_name
ORDER BY total_purchases DESC;
```

Output:

```
product_name total_purchases
------------ ---------------
ramen                   8
```

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

```
with most_popular as (
select s.customer_id, m.product_name, count(s.product_id) as "popular",
dense_rank() over(partition by s.customer_id order by count(s.product_id) desc) as
dense_rank
from sales s join menu m on s.product_id=m.product_id
group by s.customer_id, m.product_name
)
select customer_id, product_name, popular from most_popular where dense_rank=1;
```

Output:

```
customer_id product_name popular
----------- ------------ -----------
A           ramen                 3
```

```
B       sushi           2
B       curry           2
B       ramen            2
C       ramen            3
```

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

```
with first as (
select s.customer_id, s.product_id, me.product_name,
row_number() over(partition by s.customer_id order by s.customer_id) as rnk
from sales s
join members m on s.customer_id=m.customer_id
join menu me on s.product_id=me.product_id
where s.order_date >= m.join_date
)

select customer_id, product_id, product_name from first where rnk = 1
```

Output:

```
customer_id product_id  product_name
----------- ----------- ------------
A                2 curry
B                1 sushi
```

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

```
with first as (
select s.customer_id, s.product_id, me.product_name,
DENSE_RANK() over(partition by s.customer_id order by s.order_date) as rnk
from sales s
join members m on s.customer_id=m.customer_id
join menu me on s.product_id=me.product_id
where s.order_date < m.join_date
)

select customer_id, product_id, product_name from first where rnk = 1;
```

Output:

```
customer_id product_id  product_name
----------- ----------- ------------
A                1 sushi
A                2 curry
B                2 curry
```

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

```
with first as (
select s.customer_id, sum(me.price) as total_amount, count(s.product_id) as "total_items",
DENSE_RANK() over(partition by s.customer_id order by s.customer_id) as rnk
from sales s
join members m on s.customer_id=m.customer_id
join menu me on s.product_id=me.product_id
where s.order_date < m.join_date
group by s.customer_id
)

select customer_id, total_items, total_amount from first where rnk = 1;
```

Output:

```
customer_id total_items total_amount
----------- ----------- ------------
A                   2          25
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?

```
Select S.customer_id, sum(
Case
When m.product_id = 1 THEN m.price*20
Else m.price*10
End) as Points
From Sales S
Join menu m
On m.product_id = S.product_id
Group by S.customer_id;
```

Output:

```
customer_id 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?

```
SELECT m.Customer_id, SUM(n.price * 20) AS Total_points
FROM
members m
JOIN
sales s ON m.customer_id = s.customer_id
JOIN
menu n ON n.product_id = s.product_id
WHERE
s.order_date >= m.join_date
AND MONTH(s.order_date) = 1
GROUP BY m.customer_id;
```

Output:

```
Customer_id Total_points
----------- ------------
A                  1020
B                   440
```