CASE STUDY 1 - DANNYS 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

Link to Dataset: https://8weeksqlchallenge.com/case-study-1/

Case Study Questions
Each of the following case study questions can be answered using a single SQL statement:

1. **What is the total amount each customer spent at the restaurant?**
2. **How many days has each customer visited the restaurant?**
3. **What was the first item from the menu purchased by each customer?**
4. **What is the most purchased item on the menu and how many times was it purchased by all customers?**
5. **Which item was the most popular for each customer?**
6. **Which item was purchased first by the customer after they became a member?**
7. **Which item was purchased just before the customer became a member?**
8. **What is the total items and amount spent for each member before they became a member?**
9. **If each $1 spent equates to 10 points and sushi has a 2x points multiplier - how many points would each customer have?**
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?**
11. **Join All The Things**
12. **Rank All The Things**

SOLUTIONS

```
CREATE OR REPLACE DATABASE DEMO_DATABASE;
CREATE OR REPLACE SCHEMA DANNYS_DINER;

CREATE OR REPLACE 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 OR REPLACE 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 OR REPLACE 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
  CUSTOMER_ID,
  SUM(PRICE) AS TOTAL_SPENT
FROM
  SALES AS S
  INNER JOIN MENU AS M ON M.PRODUCT_ID = S.PRODUCT_ID
GROUP BY
  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?

```
SELECT
  CUSTOMER_ID,
  COUNT(DISTINCT ORDER_DATE) AS DAYS_VISITED
FROM
  SALES
GROUP BY
  CUSTOMER_ID;
```

	CUSTOMER_ID	...	DAYS_VISITED
1	A		4
2	B		6
3	C		2

-- 3. WHAT WAS THE FIRST ITEM FROM THE MENU PURCHASED BY EACH CUSTOMER?

```
WITH CTE AS (
  SELECT
    CUSTOMER_ID,
    ORDER_DATE,
    PRODUCT_NAME,
    RANK() OVER(PARTITION BY CUSTOMER_ID ORDER BY ORDER_DATE) AS RNK,
    ROW_NUMBER() OVER(PARTITION BY CUSTOMER_ID ORDER BY ORDER_DATE ASC) AS RN
  FROM
    SALES AS S
    INNER JOIN MENU AS M ON S.PRODUCT_ID = M.PRODUCT_ID
)
SELECT
  CUSTOMER_ID, PRODUCT_NAME
FROM CTE
WHERE
RNK = 1;
```

	CUSTOMER_ID	...	PRODUCT_NAME
1	A		sushi
2	A		curry
3	B		curry
4	C		ramen
5	C		ramen

-- 4. WHAT IS THE MOST PURCHASED ITEM ON THE MENU AND HOW MANY TIMES WAS IT PURCHASED BY ALL CUSTOMERS?

```
SELECT
  PRODUCT_NAME,
  COUNT(ORDER_DATE) AS ORDERS
FROM
  SALES AS S
  INNER JOIN MENU AS M ON S.PRODUCT_ID = M.PRODUCT_ID
GROUP BY
  PRODUCT_NAME
ORDER BY
  COUNT(ORDER_DATE) DESC
LIMIT 1;
```

	PRODUCT_NAME	...	ORDERS
1	ramen		8

-- 5. WHICH ITEM WAS THE MOST POPULAR FOR EACH CUSTOMER?

```
WITH CTE AS (
  SELECT
    PRODUCT_NAME,
    CUSTOMER_ID,
    COUNT(ORDER_DATE) AS ORDERS,
    RANK() OVER(PARTITION BY CUSTOMER_ID ORDER BY COUNT(ORDER_DATE) DESC) AS RNK,
    ROW_NUMBER() OVER(PARTITION BY CUSTOMER_ID ORDER BY COUNT(ORDER_DATE) DESC) AS
RN
  FROM
    SALES AS S
    INNER JOIN MENU AS M ON S.PRODUCT_ID = M.PRODUCT_ID
  GROUP BY
    PRODUCT_NAME,
    CUSTOMER_ID
)
SELECT
  CUSTOMER_ID,
  PRODUCT_NAME
FROM CTE
WHERE RNK = 1;
```

	CUSTOMER_ID	PRODUCT_NAME
1	A	ramen
2	B	curry
3	B	sushi
4	B	ramen
5	C	ramen

-- 6. WHICH ITEM WAS PURCHASED FIRST BY THE CUSTOMER AFTER THEY BECAME A MEMBER?

```
WITH CTE AS (
  SELECT
    S.CUSTOMER_ID,
    ORDER_DATE,
    JOIN_DATE,
    PRODUCT_NAME,
    RANK() OVER(PARTITION BY S.CUSTOMER_ID ORDER BY ORDER_DATE ASC) AS RNK,
    ROW_NUMBER() OVER(PARTITION BY S.CUSTOMER_ID ORDER BY ORDER_DATE) AS RN

FROM
    SALES AS S
    INNER JOIN MENU AS M ON S.PRODUCT_ID = M.PRODUCT_ID
    INNER JOIN MEMBERS AS MEM ON MEM.CUSTOMER_ID = S.CUSTOMER_ID
  WHERE
    ORDER_DATE >= JOIN_DATE
  ORDER BY
    ORDER_DATE ASC
)
SELECT
  CUSTOMER_ID,
  PRODUCT_NAME
FROM
  CTE
WHERE
  RNK = 1;
```

	CUSTOMER_ID	PRODUCT_NAME
1	A	curry
2	B	sushi

-- 7. WHICH ITEM WAS PURCHASED JUST BEFORE THE CUSTOMER BECAME A MEMBER?

```
WITH CTE AS (
  SELECT
    S.CUSTOMER_ID,
    ORDER_DATE,
    JOIN_DATE,
    PRODUCT_NAME,
    RANK() OVER(PARTITION BY S.CUSTOMER_ID ORDER BY ORDER_DATE ASC) AS RNK,
    ROW_NUMBER() OVER(PARTITION BY S.CUSTOMER_ID ORDER BY ORDER_DATE) AS RN
  FROM
    SALES AS S
    INNER JOIN MENU AS M ON S.PRODUCT_ID = M.PRODUCT_ID
    INNER JOIN MEMBERS AS MEM ON MEM.CUSTOMER_ID = S.CUSTOMER_ID
  WHERE
    ORDER_DATE < JOIN_DATE
  ORDER BY
    ORDER_DATE ASC
)
SELECT CUSTOMER_ID, PRODUCT_NAME
FROM CTE
WHERE
  RNK = 1;
```

	CUSTOMER_ID	PRODUCT_NAME
1	A	sushi
2	B	curry
3	A	curry

-- 8. WHAT IS THE TOTAL ITEMS AND AMOUNT SPENT FOR EACH MEMBER BEFORE THEY BECAME A MEMBER?

```
SELECT
  S.CUSTOMER_ID,
  COUNT(M.PRODUCT_ID) AS TOTAL_ITEMS,
  SUM(M.PRICE) AS AMOUNT_SPENT
FROM
  SALES AS S
  INNER JOIN MENU AS M ON S.PRODUCT_ID = M.PRODUCT_ID
  INNER JOIN MEMBERS AS MEM ON MEM.CUSTOMER_ID = S.CUSTOMER_ID
WHERE
  ORDER_DATE < 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?

```
SELECT
  CUSTOMER_ID,
  SUM(
    CASE PRODUCT_NAME
      WHEN 'SUSHI' THEN PRICE * 10 * 2
      ELSE PRICE * 10
    END
  ) AS POINTS
FROM
  MENU AS M
  INNER JOIN SALES AS S ON S.PRODUCT_ID = M.PRODUCT_ID
GROUP BY
  CUSTOMER_ID;
```

	CUSTOMER_ID	POINTS
1	A	760
2	B	740
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?

```
SELECT
  S.CUSTOMER_ID,
  SUM(
    CASE
      WHEN S.ORDER_DATE BETWEEN MEM.JOIN_DATE AND DATEADD('DAY', 6, MEM.JOIN_DATE)
THEN PRICE * 10 * 2
      WHEN PRODUCT_NAME = 'SUSHI' THEN PRICE * 10 * 2
      ELSE PRICE * 10
    END
  ) AS POINTS
FROM
  MENU AS M
  INNER JOIN SALES AS S ON S.PRODUCT_ID = M.PRODUCT_ID
  INNER JOIN MEMBERS AS MEM ON MEM.CUSTOMER_ID = S.CUSTOMER_ID
WHERE
  DATE_TRUNC('MONTH', S.ORDER_DATE) = '2021-01-01'
GROUP BY
  S.CUSTOMER_ID;
```

	CUSTOMER_ID	POINTS
1	A	1270
2	B	720

--11. JOIN ALL THE THINGS

```
SELECT S.CUSTOMER_ID, ORDER_DATE, PRODUCT_NAME, PRICE,
  CASE
    WHEN JOIN_DATE IS NULL THEN 'N'
    WHEN ORDER_DATE < JOIN_DATE THEN 'N'
    ELSE 'Y'
  END AS MEMBER
FROM
  SALES AS S
  INNER JOIN MENU AS M ON S.PRODUCT_ID = M.PRODUCT_ID
  LEFT JOIN MEMBERS AS MEM ON MEM.CUSTOMER_ID = S.CUSTOMER_ID
ORDER BY
  S.CUSTOMER_ID,
  ORDER_DATE,
  PRICE DESC;
```

	CUSTOMER_ID	ORDER_DATE	PRODUCT_NAME	PRICE	MEMBER
1	A	2021-01-01	curry	15	N
2	A	2021-01-01	sushi	10	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

--12. RANK ALL THE THINGS

```
WITH CTE AS (
  SELECT
    S.CUSTOMER_ID, S.ORDER_DATE, PRODUCT_NAME, PRICE,
    CASE
      WHEN JOIN_DATE IS NULL THEN 'N'
      WHEN ORDER_DATE < JOIN_DATE THEN 'N'
      ELSE 'Y'
    END AS MEMBER
  FROM
    SALES AS S
    INNER JOIN MENU AS M ON S.PRODUCT_ID = M.PRODUCT_ID
    LEFT JOIN MEMBERS AS MEM ON MEM.CUSTOMER_ID = S.CUSTOMER_ID
  ORDER BY
    CUSTOMER_ID, ORDER_DATE, PRICE DESC)
SELECT *, CASE
    WHEN MEMBER = 'N'  THEN NULL
    ELSE RANK() OVER(PARTITION BY CUSTOMER_ID, MEMBER ORDER BY ORDER_DATE)
  END AS RNK
FROM CTE;
```

	CUSTOMER_ID	ORDER_DATE	PRODUCT_NAME	PRICE	MEMBER	RNK
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