Case Study #1 : Danny's Dinner Solution Using SQL

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

```
SELECT
     sales.customer_id,
    SUM(menu.price) AS 'Total Spending in $'
FROM sales
INNER JOIN menu ON
sales.product_id=menu.product_id
GROUP BY sales.customer_id
ORDER BY sales.customer_id;
```

customer_id	total_sales
A	76
B	74
C	36

Q 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
A	4
B	6
C	2

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

```sql
SELECT
    customer_id,
    product_name AS first_product_ordered
FROM (
    SELECT
        sales.customer_id,
        menu.product_name,
        DENSE_RANK() OVER (
               PARTITION BY sales.customer_id
               ORDER BY sales.order_date) AS rnk
    FROM sales
    INNER JOIN menu
          ON sales.product_id = menu.product_id
) AS sales_rank
WHERE rnk = 1
GROUP BY customer_id, product_name;
```

customer_id	first_product_ordered
A	sushi
A	curry
B	curry
C	ramen

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

This Question needs to be solved in two parts, where in part 1, we will find the most purchased item and in Part 2, we will fetch the results for how many times customers purchase this item.

```sql
-- Part 1
SELECT
     menu.product_name AS 'Most purchased Product',
    COUNT(sales.product_id) AS 'purchase count'
FROM sales
INNER JOIN menu
     ON sales.product_id = menu.product_id
GROUP BY menu.product_name
ORDER BY COUNT(sales.product_id) DESC
LIMIT 1;  -- this gives the most purchased item on menu
```

most_purchased_item	purchase_count
ramen	8

```sql
-- Part 2
SELECT
    sales.customer_id,
    COUNT(sales.product_id) AS purchase_count
FROM sales
INNER JOIN menu
     ON sales.product_id = menu.product_id
WHERE sales.product_id = (SELECT product_id
                          FROM sales
                          GROUP BY product_id
                          ORDER BY COUNT(product_id)
                               DESC LIMIT 1)
GROUP BY sales.customer_id ,
         sales.product_id,
         menu.product_name
ORDER BY purchase_count DESC;

-- this gives the list of customers who have purchased
highest purchased item
```

customer_id	purchase_count
A	3
B	2
C	3

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

```sql
WITH cte_popular_products AS (
    SELECT
        sales.customer_id,
        menu.product_name,
        COUNT(*) AS purchase_count,
        DENSE_RANK() OVER (
                    PARTITION BY sales.customer_id
                     ORDER BY COUNT(*) DESC) AS rnk
    FROM sales
    JOIN menu ON sales.product_id = menu.product_id
    GROUP BY sales.customer_id, menu.product_name
)
SELECT
    customer_id,
    product_name,
    purchase_count
FROM cte_popular_products
WHERE rnk = 1;
```

customer_id	product_name	purchase_count
A	ramen	3
B	curry	2
B	sushi	2
B	ramen	2
C	ramen	3

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

```sql
WITH cte_after_membership AS (
  SELECT
    members.customer_id,
    sales.product_id,
    DENSE_RANK() OVER (
      PARTITION BY members.customer_id
      ORDER BY sales.order_date) AS densrank
  FROM members
  INNER JOIN sales
    ON members.customer_id = sales.customer_id
    AND sales.order_date > members.join_date
)
SELECT
  cte_after_membership.customer_id,
  menu.product_name
FROM cte_after_membership
INNER JOIN menu
  ON cte_after_membership.product_id = menu.product_id
WHERE densrank = 1
ORDER BY cte_after_membership.customer_id ASC;
```

customer_id	product_name
A	ramen
B	sushi

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

```sql
WITH cte_before_membership AS (
  SELECT
    members.customer_id,
    sales.product_id,
    DENSE_RANK() OVER (
      PARTITION BY members.customer_id
      ORDER BY sales.order_date) AS densrank
  FROM members
  INNER JOIN sales
    ON members.customer_id = sales.customer_id
    AND sales.order_date < members.join_date
)
SELECT
  cte_before_membership.customer_id,
  menu.product_name
FROM cte_before_membership
INNER JOIN menu
  ON cte_before_membership.product_id = menu.product_id
WHERE densrank = 1
ORDER BY cte_before_membership.customer_id ASC;
```

customer_id	product_name
A	sushi
A	curry
B	curry

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

```sql
SELECT
    sales.customer_id,
    COUNT(sales.product_id) AS total_items,
    SUM(menu.price) AS total_amount_spent
FROM sales
JOIN members ON sales.customer_id = members.customer_id
JOIN menu ON sales.product_id = menu.product_id
WHERE sales.order_date < members.join_date
GROUP BY sales.customer_id
ORDER BY sales.customer_id;
```

customer_id	total_items	total_amount_spent
A	2	25
B	3	40

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

```sql
SELECT
    sales.customer_id,
    SUM(menu.price),
    SUM(CASE
        WHEN menu.product_name = 'sushi' THEN menu.price * 20
        ELSE menu.price * 10
    END) AS total_points
FROM sales
     JOIN menu ON sales.product_id = menu.product_id
GROUP BY customer_id;
```

customer_id	total_price	total_points
A	76	860
B	74	940
C	36	360

Q 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 customers A and B have at the end of January?

```
SELECT
     sales.customer_id,
    SUM(menu.price) AS total_cost,
    SUM(CASE
          WHEN sales.order_date BETWEEN members.join_date AND
DATE_ADD(members.join_date, INTERVAL 6 DAY) THEN menu.price*2*10
            ELSE CASE
                    WHEN product_name = 'sushi' THEN price * 20
                    ELSE price * 10
                    END
                END) AS points
FROM sales
INNER JOIN menu ON sales.product_id = menu.product_id
INNER JOIN members ON sales.customer_id = members.customer_id
WHERE sales.order_date <='2021-01-31'
AND sales.customer_id IN ('A', 'B')
GROUP BY sales.customer_id
ORDER BY sales.customer_id;
```

customer_id	total_price	points
A	76	1370
B	62	820

Bonus Questions:

Q1. Join All The Things

Recreate the table with columns:
customer_id, order_date, product_name, price, member_status (Y/N)

```sql
SELECT
  sales.customer_id,
  sales.order_date,
  menu.product_name,
  menu.price,
  CASE
    WHEN sales.order_date >= members.join_date THEN 'Y'
    ELSE 'N' END AS member_status
FROM sales
LEFT JOIN members
  ON sales.customer_id = members.customer_id
INNER JOIN menu
  ON sales.product_id = menu.product_id
ORDER BY members.customer_id, sales.order_date;
```

customer_id	order_date	product_name	price	member_status
A	2021-01-01	sushi	10	N
A	2021-01-01	curry	15	N
A	2021-01-07	curry	15	Y
A	2021-01-10	ramen	12	Y
A	2021-01-11	ramen	12	Y
A	2021-01-11	ramen	12	Y
B	2021-01-01	curry	15	N
B	2021-01-02	curry	15	N
B	2021-01-04	sushi	10	N
B	2021-01-11	sushi	10	Y
B	2021-01-16	ramen	12	Y
B	2021-02-01	ramen	12	Y
C	2021-01-01	ramen	12	N
C	2021-01-01	ramen	12	N
C	2021-01-07	ramen	12	N

Q2. Rank All The Things

Rahul 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.

```
WITH cte_all_joined AS (
    SELECT
        s.customer_id,
        s.order_date AS order_date,
        m.product_name,
        m.price,
        CASE
            WHEN s.customer_id IS NOT NULL
                 AND s.order_date >= mm.join_date THEN 'Y'
            ELSE 'N'
        END AS member_status
    FROM sales s
    INNER JOIN menu m ON s.product_id = m.product_id
    LEFT JOIN members mm ON s.customer_id = mm.customer_id
)
SELECT
    *,
    CASE
        WHEN member_status = 'Y' THEN
            DENSE_RANK() OVER (
                    PARTITION BY customer_id
                ORDER BY order_date)
        ELSE
            NULL
    END AS ranking
FROM
    cte_all_joined
ORDER BY
    customer_id,
    order_date,
    product_name;
```

customer_id	order_date	product_name	price	member_status	ranking
A	2021-01-01	curry	15	N	
A	2021-01-01	sushi	10	N	
A	2021-01-07	curry	15	Y	2
A	2021-01-10	ramen	12	Y	3
A	2021-01-11	ramen	12	Y	4
A	2021-01-11	ramen	12	Y	4
B	2021-01-01	curry	15	N	
B	2021-01-02	curry	15	N	
B	2021-01-04	sushi	10	N	
B	2021-01-11	sushi	10	Y	4
B	2021-01-16	ramen	12	Y	5
B	2021-02-01	ramen	12	Y	6
C	2021-01-01	ramen	12	N	
C	2021-01-01	ramen	12	N	
C	2021-01-07	ramen	12	N	

That was a fun experience! I deepened my understanding of using CTEs, ranking functions, CASE statements, and joins to analyze Danny's customer data effectively.

For more such SQL Challenges and Data Analysis related stuff, Subscribe my youtube channel www.youtube.com/@iThinkData

Github: www.github.com/vaibhavchavan20

LinkedIn: www.linkedin.com/in/vaibhav-chavan

Youtube: http://www.youtube.com/@iThinkData

WhatsApp: www.bit.ly/WhatsAppiThinkData