CASE STUDY #1 DANNY'S DINER

8-Week-SQL-Challenge

8WEEKSQLCHALLENGE.COM

CASE STUDY #1

THE TASTE OF SUCCESS

DATAWITHDANNY.COM

CREATE TABLE sales, menu, members

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

customer_id	order_date	product_id

```sql
CREATE TABLE menu (
    "product_id" INTEGER,
    "product_name" VARCHAR(5),
    "price" INTEGER
);
```

product_id	product_name	price

```sql
CREATE TABLE members(
    "customer_id" varchar(1),
    "join_date" date
);
```

customer_id	join_date

INSERT INTO VALUE sales

```sql
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');
```

	customer_id	order_date	product_id
1	A	2021-01-01	1
2	A	2021-01-01	2
3	A	2021-01-07	2
4	A	2021-01-10	3
5	A	2021-01-11	3
6	A	2021-01-11	3
7	B	2021-01-01	2
8	B	2021-01-02	2
9	B	2021-01-04	1
10	B	2021-01-11	1
11	B	2021-01-16	3
12	B	2021-02-01	3
13	C	2021-01-01	3
14	C	2021-01-01	3
15	C	2021-01-07	3

INSERT INTO VALUE menu

```
INSERT INTO menu
("product_id","product_name","price")
VALUES
('1','sushi','10'),
('2','curry','15'),
('3','ramen','12');
```

	product_id	product_name	price
1	1	sushi	10
2	2	curry	15
3	3	ramen	12

INSERT INTO VALUE members

```
INSERT INTO members
  ("customer_id", "join_date")
VALUES
  ('A', '2021-01-07'),
  ('B', '2021-01-09');
```

	customer_id	join_date
1	A	2021-01-07
2	B	2021-01-09

Case Study Question

```
/* --------------------
   Case Study Questions
   --------------------*/

-- 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?

```

Case Study Question 1

```sql
-- 1. What is the total amount each customer spent at the restaurant?
SELECT * FROM sales
SELECT * FROM menu;

SELECT
  sales.customer_id,
  SUM(menu.price) AS total_sales
FROM sales
INNER JOIN menu
  ON sales.product_id = menu.product_id
GROUP BY sales.customer_id
ORDER BY sales.customer_id ASC;
```

	customer_id	total_sales
1	A	76
2	B	74
3	C	36

Case Study Question 2

```sql
-- 2. How many days has each customer visited the restaurant?
select sales.customer_id, COUNT( DISTINCT sales.order_date) AS visit_count
from sales
group by customer_id;
```

	customer_id	visit_count
1	A	4
2	B	6
3	C	2

Case Study Question 3

```sql
WITH ordered_sales AS(
SELECT
    sales.customer_id,
    sales.order_date,
    menu.product_name,
    DENSE_RANK() OVER (
      PARTITION BY sales.customer_id
      ORDER BY sales.order_date) AS rank
  FROM sales
  INNER JOIN menu
    ON sales.product_id = menu.product_id
)

SELECT customer_id,
        product_name
FROM ordered_sales
WHERE rank = 1
GROUP BY customer_id, product_name;
```

- **WITH using create Temporary table**
- **PARTITION using separate Categories**
- **DENSE_RANK using rank arrangement**
- **OVER () using window function**

	customer_id	product_name
1	A	curry
2	A	sushi
3	B	curry
4	C	ramen

Case Study Question 4

```sql
-- 4. What is the most purchased item on the menu
-- and how many times was it purchased by all customers?
SELECT TOP 1 sales.product_id,
        menu.product_name ,
        COUNT ( sales.order_date ) AS total
FROM sales
INNER JOIN menu
 ON sales.product_id = menu.product_id
GROUP BY sales.product_id, menu.product_name
ORDER BY total DESC
```

	product_id	product_name	total
1	3	ramen	8

Case Study Question 5

```
WITH most_popular AS (SELECT sales.customer_id, menu.product_name,
    COUNT (menu.product_id) AS order_count,
    DENSE_RANK () OVER (
        PARTITION BY sales.customer_id
        ORDER BY COUNT(sales.customer_id) DESC) AS rank
FROM sales
INNER JOIN menu
 ON sales.product_id = menu.product_id
GROUP BY sales.customer_id, menu.product_name
)

SELECT
  customer_id,
  product_name,
  order_count
FROM most_popular
WHERE rank = 1;
```

	customer_id	product_name	order_count
1	A	ramen	3
2	B	sushi	2
3	B	curry	2
4	B	ramen	2
5	C	ramen	3

Case Study Question 6

```sql
-- 6. Which item was purchased first by the customer after they became a
WITH joined_as_number AS (SELECT members.customer_id, members.join_date,
        sales.order_date, sales.product_id,
        ROW_NUMBER() OVER(
            PARTITION BY  members.customer_id
            ORDER BY sales.order_date) AS row_num
FROM members
INNER JOIN sales
 ON members.customer_id = sales.customer_id
 AND sales.order_date > members.join_date
 )

SELECT joined_as_number.customer_id, joined_as_number.join_date,
        joined_as_number.order_date, menu.product_name
FROM joined_as_number
INNER JOIN menu
    ON joined_as_number.product_id = menu.product_id
WHERE row_num = 1
```

	customer_id	join_date	order_date	product_name
1	A	2021-01-07	2021-01-10	ramen
2	B	2021-01-09	2021-01-11	sushi

Case Study Question 7

```sql
-- 7. Which item was purchased just before the customer became a member?
WITH purchased_prior_member AS (
SELECT members.customer_id, members.join_date,
        sales.order_date, sales.product_id,
        ROW_NUMBER() OVER(
            PARTITION BY  members.customer_id
            ORDER BY sales.order_date DESC) AS row_num
FROM members
INNER JOIN sales
 ON members.customer_id = sales.customer_id
 AND sales.order_date < members.join_date
 )

SELECT purchased_prior_member.customer_id, purchased_prior_member.join_date,
        purchased_prior_member.order_date, menu.product_name
FROM purchased_prior_member
INNER JOIN menu
    ON purchased_prior_member.product_id = menu.product_id
WHERE row_num = 1
```

	customer_id	join_date	order_date	product_name
1	A	2021-01-07	2021-01-01	sushi
2	B	2021-01-09	2021-01-04	sushi

Case Study Question 8

```sql
-- 8. What is the total items and amount spent for each member
--before they became a member?
SELECT
  sales.customer_id,
  COUNT(sales.product_id) AS total_items,
  SUM(menu.price) AS total_sales
FROM sales
INNER JOIN members
  ON sales.customer_id = members.customer_id
  AND sales.order_date < members.join_date
INNER JOIN menu
  ON sales.product_id = menu.product_id
GROUP BY sales.customer_id
ORDER BY sales.customer_id;
```

	customer_id	total_items	total_sales
1	A	2	25
2	B	3	40

Case Study Question 9

```sql
-- 9.  If each $1 spent equates to 10 points and sushi has a
--2x points multiplier - how many points would each customer have?
WITH points_cte AS (
  SELECT
    menu.product_id,
    CASE
      WHEN product_id = 1 THEN price * 20
      ELSE price * 10 END AS points
  FROM menu
)

SELECT
  sales.customer_id,
  SUM(points_cte.points) AS total_points
FROM sales
INNER JOIN points_cte
  ON sales.product_id = points_cte.product_id
GROUP BY sales.customer_id
ORDER BY sales.customer_id;
```

	customer_id	total_points
1	A	860
2	B	940
3	C	360

Case Study Question 10

```sql
-- 10. In the first week after a customer joins
WITH dates_cte AS (
  SELECT
    customer_id,
      join_date,
      DATEADD(day, 6, members.join_date) AS valid_date,
      DATEADD(month, 1, DATEADD(day, -1, DATETRUNC(month, CAST('2021-01-31' AS datetime)))) AS last_date
  FROM members
)

SELECT
  sales.customer_id,
  SUM(CASE
    WHEN menu.product_name = 'sushi' THEN 2 * 10 * menu.price
    WHEN sales.order_date BETWEEN dates.join_date AND dates.valid_date THEN 2 * 10 * menu.price
    ELSE 10 * menu.price END) AS points
FROM sales
INNER JOIN dates_cte AS dates
  ON sales.customer_id = dates.customer_id
  AND dates.join_date <= sales.order_date
  AND sales.order_date <= dates.last_date
INNER JOIN menu
  ON sales.product_id = menu.product_id
GROUP BY sales.customer_id;
```

	customer_id	points
1	A	1020
2	B	320

BONUS QUESTIONS

Join All The Things

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

```sql
---BONUS QUESTIONS
-- Join All The Things
-- Recreate the table with: customer_id, order_date, product_name, price, member (Y/N)
SELECT
    sales.customer_id,
    sales.order_date,
    menu.product_name,
    menu.price,
    CASE
        WHEN members.join_date > sales.order_date THEN 'N'
        WHEN members.join_date <= sales.order_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
1	C	2021-01-01	ramen	12	N
2	C	2021-01-01	ramen	12	N
3	C	2021-01-07	ramen	12	N
4	A	2021-01-01	sushi	10	N
5	A	2021-01-01	curry	15	N
6	A	2021-01-07	curry	15	Y
7	A	2021-01-10	ramen	12	Y
8	A	2021-01-11	ramen	12	Y
9	A	2021-01-11	ramen	12	Y
10	B	2021-01-01	curry	15	N
11	B	2021-01-02	curry	15	N
12	B	2021-01-04	sushi	10	N
13	B	2021-01-11	sushi	10	Y
14	B	2021-01-16	ramen	12	Y
15	B	2021-02-01	ramen	12	Y

Rank All The Things

Danny 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.

```sql
--Rank All The Things
--Danny 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 customers_data AS (
    SELECT
        sales.customer_id,
        sales.order_date,
        menu.product_name,
        menu.price,
        CASE
            WHEN members.join_date > sales.order_date THEN 'N'
            WHEN members.join_date <= sales.order_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
    )

SELECT
  *,
  CASE
    WHEN member_status = 'N' then NULL
    ELSE RANK () OVER (
      PARTITION BY customer_id, member_status
      ORDER BY order_date
  ) END AS ranking
FROM customers_data;
```

	customer_id	order_date	product_name	price	member_status	ranking
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

Learned new knowledge SQL

DISTINCT - get unique record

DENSE_RANK -

ROW_NUMBER - Rank the records in order

RANK - The chart rankings recorded and did not increase consistently

OVER() - windows function

WITH() - create temporary table

DATEADD - Add and subtract date and time in MS sql

DATETRUNC - truncate the date and time as required, The truncated information will be returned to the original

CAST - value type coercion in MS SQL

CASE_WHEN - Check and consider each different condition

interval - records a specific time, INTERVAL records and calculates the time interval in using postgresql

PROGRAMMER MENTORSHIP K2

Thank You

Contact Us

0931754728

trantrongphuqn123@gmail.com

Gerente General