--QUESTIONS

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

```
SELECT
  s.customer_id,
  SUM(m.price) AS total_spent
FROM sales s
JOIN menu m
ON s.product_id = m.product_id
GROUP BY s.customer_id
ORDER BY s.customer_id;
```

RESULT:

	customer_id character varying (1)	total_spent bigint
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 visit_days
 FROM sales
 GROUP BY customer_id;
```

RESULT:

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

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

WITH orders_rank As

(

SELECT customer_id,

        order_date,

        product_id,

        DENSE_RANK()OVER(PARTITION BY customer_id ORDER BY order_date ASC ) AS orders_rank

        FROM sales

)

SELECT

o.customer_id,

m.product_name

FROM orders_rank o JOIN menu m

ON o.product_id=m.product_id

WHERE o.orders_rank=1

ORDER BY o.customer_id;
```

RESULT:

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

```
--CAN USE DENSE_RANK/ROW_NUMBER SINCE THERE IS NO TIMESTAMP IS GIVEN

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

SELECT m.product_name,

COUNT(m.product_name) AS most_count

FROM sales s JOIN menu m

ON s.product_id=m.product_id

GROUP BY m.product_name

ORDER BY most_count DESC

LIMIT 1;
```

RESULT:

	product_name character varying (5)	most_count bigint
1	ramen	8

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

```
WITH Fev_item AS(
SELECT
        s.customer_id,
        m.product_name,
        COUNT(m.product_id) AS order_count,
        DENSE_RANK() OVER(PARTITION BY s.customer_id ORDER BY COUNT(s.customer_id)DESC)
AS  rnk
        FROM menu AS m
        JOIN sales As s
        ON m.product_id=s.product_id
        GROUP BY 1,2
)
 SELECT customer_id,product_name,order_count
 FROM fev_item
 WHERE rnk=1;
```

RESULT:

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

--Cust A likes ramen most,

B likes all the items,

C likes ramen most.

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

```
WITH member_cte AS
(
SELECT
        s.customer_id,mb.join_date,s.order_date,s.product_id,
        DENSE_RANK() OVER(PARTITION BY s.customer_id
                                        ORDER BY s.order_date) AS rnk
        FROM sales s
        JOIN members mb
        ON s.customer_id=mb.customer_id
        WHERE s.order_date>=mb.join_date
)
SELECT c.customer_id,c.order_date,m.product_name
FROM member_cte c
JOIN menu m
ON c.product_id=m.product_id
WHERE rnk=1
ORDER BY c.customer_id;
```

RESULT:

	customer_id character varying (1)	order_date date	product_name character varying (5)
1	A	2021-01-07	curry
2	B	2021-01-11	sushi

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

```
WITH before_member_cte AS
(
SELECT
        s.customer_id,mb.join_date,s.order_date,s.product_id,
        DENSE_RANK() OVER(PARTITION BY s.customer_id
                                        ORDER BY s.order_date DESC) AS rnk
        FROM sales s
        JOIN members mb
        ON s.customer_id=mb.customer_id
        WHERE s.order_date<mb.join_date

)

SELECT bc.customer_id,bc.order_date,m.product_name
FROM before_member_cte bc
JOIN menu m
ON bc.product_id=m.product_id
WHERE rnk=1
ORDER BY bc.customer_id;
```

RESULT:

	customer_id character varying (1)	unique_menu bigint	total_sales bigint
1	A	2	25
2	B	2	40

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

```
SELECT s.customer_id,
COUNT(DISTINCT s.product_id) AS unique_menu,
SUM(m.price) AS total_sales
FROM sales AS s
JOIN members AS mb
 ON s.customer_id = mb.customer_id
JOIN menu AS m
 ON s.product_id = m.product_id
WHERE s.order_date < mb.join_date
GROUP BY s.customer_id;
```

RESULT:

	customer_id character varying (1)	unique_menu bigint	total_sales bigint
1	A	2	25
2	B	2	40

--Before becoming member

A ordered 2 items and spent total 25

And B ordered 2 items and spent 40.

```
-- 9.  If each $1 spent equates to 10 points and sushi has a 2x points multiplier -

-- how many points would each customer have?


WITH cte

AS

(

SELECT

product_id,product_name,price,

CASE

WHEN product_name='sushi' THEN price*20

ELSE price*10

END AS points

FROM menu

        )


        SELECT

        s.customer_id,

        SUM(c.points) AS totalPoints

        FROM cte c

        JOIN sales s

        ON c.product_id=s.product_id

        GROUP BY s.customer_id

        ORDER BY s.customer_id;
```

RESULT:

	customer_id character varying (1)	totalpoints bigint
1	A	860
2	B	940
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?

WITH date_cte

AS

(

SELECT

        customer_id,

        join_date,

        join_date + INTERVAL '6 DAYS' AS first_week,

        (DATE_TRUNC('MONTH','2021-01-31'::DATE)+ INTERVAL '1 MONTH'-INTERVAL '1 DAY') AS last_day

        FROM members

)

SELECT s.customer_id,

SUM(

        CASE WHEN m.product_name='sushi' THEN 2*10*m.price

  WHEN s.order_date BETWEEN dc.join_date AND dc.first_week THEN 2*10*m.price

  ELSE 10*m.price

        END) AS points

FROM sales s

JOIN date_cte dc

ON s.customer_id=dc.customer_id

JOIN menu m

ON s.product_id=m.product_id

 WHERE

 dc.join_date<=s.order_date

AND s.order_date<=dc.last_day

GROUP BY s.customer_id

ORDER BY s.customer_id;
```

RESULT:

	customer_id character varying (1)	points bigint
1	A	1020
2	B	320

```
--BOnus questions

--JOIN ALL THINGS

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


SELECT

s.customer_id,

s.order_date,

m.product_name,

m.price,

CASE

WHEN mb.join_date>s.order_date THEN 'N'

        WHEN mb.join_date<=s.order_date THEN 'Y'

        ELSE 'N'

        END

AS member

FROM sales s

LEFT JOIN menu m

ON s.product_id=m.product_id

LEFT JOIN members mb

ON s.customer_id=mb.customer_id;
```

RESULT:

	customer_id character varying (1)	order_date date	product_name character varying (5)	price integer	member text
1	A	2021-01-07	curry	15	Y
2	A	2021-01-11	ramen	12	Y
3	A	2021-01-11	ramen	12	Y
4	A	2021-01-10	ramen	12	Y
5	A	2021-01-01	sushi	10	N
6	A	2021-01-01	curry	15	N
7	B	2021-01-04	sushi	10	N
8	B	2021-01-11	sushi	10	Y
9	B	2021-01-01	curry	15	N
10	B	2021-01-02	curry	15	N
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

```
--Rank All The Things

WITH cust_cte AS

(

 SELECT

s.customer_id,

s.order_date,

m.product_name,

m.price,

CASE

WHEN mb.join_date>s.order_date THEN 'N'

        WHEN mb.join_date<=s.order_date THEN 'Y'

        ELSE 'N'

        END

AS member

FROM sales s

LEFT JOIN menu m

ON s.product_id=m.product_id

LEFT JOIN members mb

ON s.customer_id=mb.customer_id


)

SELECT customer_id,order_date,product_name,price,member,

CASE

WHEN member='N' THEN NULL

ELSE RANK() OVER(PARTITION BY customer_id,member ORDER BY order_date)

END AS ranking

FROM cust_cte;
```

RESULT:

	customer_id character varying (1)	order_date date	product_name character varying (5)	price integer	member text	ranking bigint
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]