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

Query:

```
SELECT
    SL.CUSTOMER_ID,
    SUM(ME.PRICE) AS TOTAL_AMOUNT_SPENT
FROM
    SALES SL
    LEFT JOIN MENU ME ON SL.PRODUCT_ID = ME.PRODUCT_ID
GROUP BY
    SL.CUSTOMER_ID
ORDER BY
    TOTAL_AMOUNT_SPENT DESC;
```

Result:

	customer_id character varying (5)	total_amount_spent bigint
1	A	76
2	B	74
3	C	36

Customer A spent the most amount ($76) at the restaurant.

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

Query:

```
SELECT
    CUSTOMER_ID,
    COUNT(DISTINCT ORDER_DATE) AS VISITED
FROM
    SALES
GROUP BY
    CUSTOMER_ID
ORDER BY
    VISITED DESC;
```

Result:

	customer_id character varying (5)	visited bigint
1	B	6
2	A	4
3	C	2

Customer A Visited the restaurant most frequently.

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

Query:

```
WITH
    RANK_TABLE AS (
        SELECT
            *,
            DENSE_RANK() OVER (
                PARTITION BY
                    CUSTOMER_ID
                ORDER BY
                    ORDER_DATE ASC
            ) AS RN
        FROM
            MENU ME
            JOIN SALES SL ON ME.PRODUCT_ID = SL.PRODUCT_ID
    )
SELECT DISTINCT
    CUSTOMER_ID,
    PRODUCT_NAME
FROM
    RANK_TABLE
WHERE
    RN = 1
ORDER BY
    CUSTOMER_ID ASC;
```

Result:

	customer_id character varying (5)	product_name character varying (10)
1	A	curry
2	A	sushi
3	B	curry
4	C	ramen

Customer A purchase Curry and Sushi for the first time.

Customer B purchase Curry for the first time.

Customer C purchase Ramen for the first time.

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

Query:

```
SELECT
    ME.PRODUCT_NAME,
    COUNT(SL.PRODUCT_ID) AS MOST_PURCHASED_ITEM
FROM
    MENU ME
    JOIN SALES SL ON ME.PRODUCT_ID = SL.PRODUCT_ID
GROUP BY
    ME.PRODUCT_NAME LIMIT
    1;
```

Result:

	product_name character varying (10)	most_purchased_item bigint
1	ramen	8

The most purchased item on the menu was Ramen.
It was purchased 8 items by all customers.

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

Query:

```
WITH
    MOST_POPULAR AS (
        SELECT
            SL.CUSTOMER_ID,
            ME.PRODUCT_NAME,
            COUNT(ME.PRODUCT_ID) AS ORDER_COUNT,
            DENSE_RANK() OVER (
                PARTITION BY
                    SL.CUSTOMER_ID
                ORDER BY
                    COUNT(SL.CUSTOMER_ID) DESC
            ) AS RANK
        FROM
            MENU ME
            JOIN SALES SL ON ME.PRODUCT_ID = SL.PRODUCT_ID
        GROUP BY
            SL.CUSTOMER_ID,
            ME.PRODUCT_NAME
    )
SELECT
    CUSTOMER_ID,
    PRODUCT_NAME,
    ORDER_COUNT
FROM
    MOST_POPULAR
WHERE
    RANK = 1;
```

Result:

	customer_id character varying (5)	product_name character varying (10)	order_count bigint
1	A	ramen	3
2	B	sushi	2
3	B	curry	2
4	B	ramen	2
5	C	ramen	3

Ramen is most popular among all customers but for Customer B Sushi and Curry also most popular.

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

Query:

```
WITH
    FIRST_PURCHASE AS (
        SELECT
            SL.CUSTOMER_ID,
            SL.ORDER_DATE,
            SL.PRODUCT_ID,
            MM.JOIN_DATE,
            ME.PRODUCT_NAME,
            ROW_NUMBER() OVER (
                PARTITION BY
                    SL.CUSTOMER_ID
                ORDER BY
                    ORDER_DATE ASC
            ) AS RN
        FROM
            SALES SL
            LEFT JOIN MEMBERS MM ON SL.CUSTOMER_ID = MM.CUSTOMER_ID
            JOIN MENU ME ON SL.PRODUCT_ID = ME.PRODUCT_ID
        WHERE
            SL.ORDER_DATE >= MM.JOIN_DATE
    )
SELECT
    CUSTOMER_ID,
    PRODUCT_ID,
    PRODUCT_NAME
FROM
    FIRST_PURCHASE
WHERE
    RN = 1;
```

Result:

	customer_id character varying (5)	product_id integer	product_name character varying (10)
1	A	2	curry
2	B	1	sushi

There are three customers but Customer C did not become a member.
Customer A and B became a member and purchased Curry and Sushi first.

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

Query:

```
WITH
    PURCHASED_PRIOR_MEMBER AS (
        SELECT DISTINCT
            SL.CUSTOMER_ID,
            ME.PRODUCT_NAME,
            ROW_NUMBER() OVER (
                PARTITION BY
                    SL.CUSTOMER_ID
                ORDER BY
                    SL.ORDER_DATE DESC
            ) AS RN
        FROM
            SALES SL
            LEFT JOIN MEMBERS MM ON SL.CUSTOMER_ID = MM.CUSTOMER_ID
            JOIN MENU ME ON SL.PRODUCT_ID = ME.PRODUCT_ID
        WHERE
            SL.ORDER_DATE < MM.JOIN_DATE
    )
SELECT
    CUSTOMER_ID,
    PRODUCT_NAME
FROM
    PURCHASED_PRIOR_MEMBER
WHERE
    RN = 1
ORDER BY
    CUSTOMER_ID ASC;
```

Result:

	customer_id character varying (5)	product_name character varying (10)
1	A	sushi
2	B	sushi

Both customers purchased Sushi just before they became a member.

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

Query:

```
SELECT
    SL.CUSTOMER_ID,
    COUNT(SL.PRODUCT_ID) AS TOTAL_ITEMS,
    SUM(ME.PRICE) AS TOTAL_AMOUNT_SPENT
FROM
    SALES SL
    LEFT JOIN MEMBERS MM ON SL.CUSTOMER_ID = MM.CUSTOMER_ID
    JOIN MENU ME ON SL.PRODUCT_ID = ME.PRODUCT_ID
WHERE
    SL.ORDER_DATE < MM.JOIN_DATE
GROUP BY
    SL.CUSTOMER_ID
ORDER BY
    SL.CUSTOMER_ID;
```

Result:

	customer_id character varying (5)	total_items bigint	total_amount_spent bigint
1	A	2	25
2	B	3	40

For customer A the total items they purchased is 2 and the amount they spent was $25.

For customer B the total items they purchased is 3 and the amount they spent was $40.

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

Query:

```
WITH
    POINTS_TABLE AS (
        SELECT
            SL.CUSTOMER_ID,
            ME.PRODUCT_NAME,
            CASE
                WHEN ME.PRODUCT_NAME IN ('sushi') THEN ME.PRICE * 20
                ELSE ME.PRICE * 10
            END AS POINTS
        FROM
            SALES SL
            JOIN MENU ME ON SL.PRODUCT_ID = ME.PRODUCT_ID
    )
SELECT
    CUSTOMER_ID,
    SUM(POINTS) AS TOTAL_POINTS
FROM
    POINTS_TABLE
GROUP BY
    CUSTOMER_ID
ORDER BY
    CUSTOMER_ID ASC;
```

Result:

	customer_id character varying (5)	total_points bigint
1	A	860
2	B	940
3	C	360

Customer B has the greatest number of points and Customer C has least number of points.

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?

Query:

```sql
WITH
    CTE AS (
        SELECT
            *,
            CASE
                WHEN PRODUCT_NAME = 'sushi' THEN PRICE * 10 * 2
                ELSE PRICE * 10
            END POINTS,
            CASE
                WHEN ORDER_DATE - JOIN_DATE <= 7 THEN 2
                ELSE 1
            END MULTIPLIER
        FROM
            MEMBERS
            NATURAL JOIN SALES
            NATURAL JOIN MENU
    ),
    CTE_2 AS (
        SELECT
            *,
            POINTS * MULTIPLIER AS TOTAL_POINTS
        FROM
            CTE
    )
SELECT
    CUSTOMER_ID,
    SUM(TOTAL_POINTS) AS TOTAL_POINTS
FROM
    CTE_2
GROUP BY
    CUSTOMER_ID;
```

Result:

	customer_id character varying	total_points bigint
1	A	1720
2	B	1760

Bonus Question

Recreate the following table output using the available data:

customer_id	order_date	product_name	price	member
A	2021-01-01	curry	15	N
A	2021-01-01	sushi	10	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

Query:

```
SELECT
    SL.CUSTOMER_ID,
    SL.ORDER_DATE,
    ME.PRODUCT_NAME,
    ME.PRICE,
    CASE
        WHEN SL.ORDER_DATE >= MM.JOIN_DATE THEN 'Y'
        ELSE 'N'
    END AS MEMBER
FROM
    SALES SL
    LEFT JOIN MEMBERS MM ON SL.CUSTOMER_ID = MM.CUSTOMER_ID
    JOIN MENU ME ON SL.PRODUCT_ID = ME.PRODUCT_ID;
```

Result:

	customer_id character varying (5)	order_date date	product_name character varying (10)	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

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.

Query:

```sql
WITH
    MEMBER_TABLE AS (
        SELECT
            SL.CUSTOMER_ID,
            SL.ORDER_DATE,
            ME.PRODUCT_NAME,
            ME.PRICE,
            CASE
                WHEN SL.ORDER_DATE >= MM.JOIN_DATE THEN 'Y'
                ELSE 'N'
            END AS MEMBER
        FROM
            SALES SL
            LEFT JOIN MEMBERS MM ON SL.CUSTOMER_ID = MM.CUSTOMER_ID
            JOIN MENU ME ON SL.PRODUCT_ID = ME.PRODUCT_ID
    )
SELECT
    *,
    CASE
        WHEN MEMBER = 'N' THEN NULL
        ELSE RANK() OVER (
            PARTITION BY
                CUSTOMER_ID,
                MEMBER
            ORDER BY
                ORDER_DATE ASC
        )
    END AS RN
FROM
    MEMBER_TABLE;
```

Result:

	customer_id character varying (5)	order_date date	product_name character varying (10)	price integer	member text	rn 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]