Schema (PostgreSQL v13)

```
CREATE SCHEMA dannys_diner;
SET search_path = dannys_diner;

CREATE 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 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 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');
```

Query #1

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

customer_id	total_amount
B	74
C	36
A	76

This query calculates the total amount spent by each customer by summing the prices of their orders in the 'sales' table.

Query #2

```
SELECT
  customer_id,
  COUNT(DISTINCT order_date) AS days_visited
FROM
  dannys_diner.sales
GROUP BY
  customer_id;
```

customer_id	days_visited
A	4
B	6
C	2

This query counts the number of distinct days each customer visited the restaurant based on their orders in the 'sales' table.

Query #3

```
SELECT
  s.customer_id,
  MIN(s.order_date) AS first_purchase_date,
  MIN(m.product_name) AS first_purchased_item
FROM
  dannys_diner.sales s
INNER JOIN
  dannys_diner.menu m
ON
  s.product_id = m.product_id
GROUP BY
  s.customer_id;
```

customer_id	first_purchase_date	first_purchased_item
B	2021-01-01T00:00:00.000Z	curry
C	2021-01-01T00:00:00.000Z	ramen
A	2021-01-01T00:00:00.000Z	curry

The query finds the earliest purchase date and the corresponding item for each customer by joining the 'sales' and 'menu' tables.

Query #4

```
SELECT
  m.product_name AS purchased_item,
  COUNT(*) AS total_purchases
FROM
  dannys_diner.sales s
INNER JOIN
  dannys_diner.menu m
ON
  s.product_id = m.product_id
GROUP BY
  m.product_name
ORDER BY
  total_purchases DESC
LIMIT 5;
```

purchased_item	total_purchases
ramen	8
curry	4
sushi	3

This query identifies the most purchased item on the menu by counting the occurrences of each item in the 'sales' table and ordering the result accordingly.

Query #5

```
SELECT
  customer_id,
  product_name AS most_popular_item,
  total_purchases
FROM (
  SELECT
    s.customer_id,
    m.product_name,
    COUNT(*) AS total_purchases,
    RANK() OVER (PARTITION BY s.customer_id ORDER BY COUNT(*) DESC) AS rnk
  FROM
    dannys_diner.sales s
  INNER JOIN
    dannys_diner.menu m
  ON
    s.product_id = m.product_id
  GROUP BY
    s.customer_id, m.product_name
) ranked_items
WHERE
  rnk = 1;
```

customer_id	most_popular_item	total_purchases
A	ramen	3
B	ramen	2
B	curry	2
B	sushi	2
C	ramen	3

This query determines the most popular item for each customer by grouping orders based on customer and product, and then ordering the result by customer and total purchases.

Query #6

```
WITH member_first_purchase AS (
  SELECT
    s.customer_id,
```

```
    m.product_name AS first_purchased_item,
    s.order_date AS first_purchase_date,
    ROW_NUMBER() OVER (PARTITION BY s.customer_id ORDER BY s.order_date) AS rnk
  FROM
    dannys_diner.sales s
  INNER JOIN
    dannys_diner.menu m
  ON
    s.product_id = m.product_id
  INNER JOIN
    dannys_diner.members mem
  ON
    s.customer_id = mem.customer_id
    AND s.order_date > mem.join_date
)

SELECT
  customer_id,
  first_purchased_item,
  first_purchase_date
FROM
  member_first_purchase
WHERE
  rnk = 1;
```

customer_id	first_purchased_item	first_purchase_date
A	ramen	2021-01-10T00:00:00.000Z
B	sushi	2021-01-11T00:00:00.000Z

This query uses a Common Table Expression (CTE) with `ROW_NUMBER()` to find the first item purchased by each customer after joining the program.

Query #7

```
WITH member_last_purchase AS (
  SELECT
    s.customer_id,
    m.product_name AS last_purchased_item,
```

```
    s.order_date AS last_purchase_date,
    ROW_NUMBER() OVER (PARTITION BY s.customer_id ORDER BY s.order_date DESC) AS
rnk
  FROM
    dannys_diner.sales s
  INNER JOIN
    dannys_diner.menu m
  ON
    s.product_id = m.product_id
  INNER JOIN
    dannys_diner.members mem
  ON
    s.customer_id = mem.customer_id
    AND s.order_date < mem.join_date
)

SELECT
  customer_id,
  last_purchased_item,
  last_purchase_date
FROM
  member_last_purchase
WHERE
  rnk = 1;
```

customer_id	last_purchased_item	last_purchase_date
A	sushi	2021-01-01T00:00:00.000Z
B	sushi	2021-01-04T00:00:00.000Z

This query uses `ROW_NUMBER()` to find the item purchased just before each customer became a member.

Query #8

```
WITH member_purchase_summary AS (
  SELECT
    mem.customer_id,
    COUNT(*) AS total_items,
```

```
    SUM(m.price) AS total_amount_spent
  FROM
    dannys_diner.sales s
  INNER JOIN
    dannys_diner.menu m
  ON
    s.product_id = m.product_id
  INNER JOIN
    dannys_diner.members mem
  ON
    s.customer_id = mem.customer_id
    AND s.order_date < mem.join_date
  GROUP BY
    mem.customer_id
)

SELECT
  customer_id,
  total_items,
  total_amount_spent
FROM
  member_purchase_summary;
```

customer_id	total_items	total_amount_spent
B	3	40
A	2	25

This query calculates the total number of items and the total amount spent for each member before they joined the program.

Query #9

```
WITH points_summary AS (
  SELECT
    s.customer_id,
    SUM(CASE WHEN m.product_name = 'sushi' THEN 2 * m.price ELSE m.price END) AS
total_amount_spent
  FROM
```

```
    dannys_diner.sales s
  INNER JOIN
    dannys_diner.menu m
  ON
    s.product_id = m.product_id
  GROUP BY
    s.customer_id
)

SELECT
  customer_id,
  total_amount_spent * 10 AS total_points
FROM
  points_summary;
```

customer_id	total_points
B	940
C	360
A	860

This query calculates the points for each customer based on their total amount spent, applying a 2x multiplier for sushi purchases.

Query #10

```
WITH points_summary AS (
  SELECT
    s.customer_id,
    SUM(
      CASE
        WHEN s.order_date >= mem.join_date AND s.order_date < mem.join_date +
INTERVAL '7 days' THEN 2 * m.price
        ELSE m.price
      END
    ) AS total_amount_spent
  FROM
    dannys_diner.sales s
  INNER JOIN
```

```
    dannys_diner.menu m
  ON
    s.product_id = m.product_id
  INNER JOIN
    dannys_diner.members mem
  ON
    s.customer_id = mem.customer_id
  GROUP BY
    s.customer_id
)

SELECT
  customer_id,
  total_amount_spent * 10 AS total_points
FROM
  points_summary
WHERE
  customer_id IN ('A', 'B');
```

customer_id	total_points
B	840
A	1270

This query calculates the points for customers A and B during the first week after joining, applying a 2x multiplier for all items.
