Second: provide SQL queries

Closed-ended questions:

- What are the top 5 brands by receipts scanned among users 21 and over?

```
-- Calculate age of users and filter for 21 and over
WITH Users21AndOver AS (
    SELECT
        USER_ID,
        BIRTH_DATE,
        DATEDIFF(YEAR, BIRTH_DATE, GETDATE()) AS AGE
    FROM user
    WHERE DATEDIFF(YEAR, BIRTH_DATE, GETDATE()) >= 21
)
-- Join datasets and count receipts
SELECT
    p.BRAND,
    COUNT(t.RECEIPT_ID) AS ReceiptCount
FROM transaction t
JOIN Users21AndOver u ON t.USER_ID = u.USER_ID
JOIN product p ON t.BARCODE = p.BARCODE
GROUP BY p.BRAND
ORDER BY ReceiptCount DESC
LIMIT 5;
```

- What are the top 5 brands by sales among users that have had their account for at least six months?

```
-- Calculate account age and filter for accounts at least 6 months old
WITH Users6Months AS (
    SELECT
        USER_ID,
        CREATED_DATE,
        DATEDIFF(MONTH, CREATED_DATE, GETDATE()) AS AccountAgeMonths
    FROM user
    WHERE DATEDIFF(MONTH, CREATED_DATE, GETDATE()) >= 6
)
-- Join datasets and sum sales
SELECT
    p.BRAND,
    SUM(t.FINAL_SALE) AS TotalSales
FROM transaction t
JOIN Users6Months u ON t.USER_ID = u.USER_ID
JOIN product p ON t.BARCODE = p.BARCODE
WHERE t.FINAL_SALE IS NOT NULL
GROUP BY p.BRAND
ORDER BY TotalSales DESC
LIMIT 5;
```

- What is the percentage of sales in the Health & Wellness category by generation?

```
-- Define generations based on birth year
WITH Generations AS (
    SELECT
        USER_ID,
        BIRTH_DATE,
        CASE
            WHEN YEAR(BIRTH_DATE) BETWEEN 1946 AND 1964 THEN 'Baby Boomers'
            WHEN YEAR(BIRTH_DATE) BETWEEN 1965 AND 1980 THEN 'Generation X'
            WHEN YEAR(BIRTH_DATE) BETWEEN 1981 AND 1996 THEN 'Millennials'
            WHEN YEAR(BIRTH_DATE) >= 1997 THEN 'Generation Z'
            ELSE 'Unknown'
        END AS Generation
    FROM user
)

-- Filter for Health & Wellness category and calculate sales by generation
WITH HealthWellnessSales AS (
    SELECT
        g.Generation,
        SUM(t.FINAL_SALE) AS GenerationSales
    FROM transaction t
    JOIN Generations g ON t.USER_ID = g.USER_ID
    JOIN product p ON t.BARCODE = p.BARCODE
```

```
    WHERE p.CATEGORY_1 = 'Health & Wellness' AND t.FINAL_SALE IS NOT NULL
    GROUP BY g.Generation
)
-- Calculate total sales and percentage by generation
SELECT
    Generation,
    GenerationSales,
    ROUND((GenerationSales * 100.0 / SUM(GenerationSales) OVER()), 2) AS PercentageOfSales
FROM HealthWellnessSales
ORDER BY PercentageOfSales DESC;
```

Open-ended questions: for these, make assumptions and clearly state them when answering the question.

- Who are Fetch's power users?

```
WITH UserMetrics AS (
    SELECT
        t.USER_ID,
        COUNT(DISTINCT t.RECEIPT_ID) AS TotalReceipts, - - total number of receipts scanned
        COUNT(DISTINCT t.STORE_NAME) AS UniqueStores,  - - total number of unique stores
        COUNT(DISTINCT p.BRAND) AS UniqueBrands - - total number of unique brands
    FROM transaction t
    LEFT JOIN product p ON t.BARCODE = p.BARCODE
    GROUP BY t.USER_ID
),
- - Thresholds for filtering users with higher than average engagement to fetch
Thresholds AS (
```

```
    SELECT
        AVG(TotalReceipts) AS AvgReceipts,
        AVG(UniqueStores) AS AvgStores,
        AVG(UniqueBrands) AS AvgBrands
    FROM UserMetrics
)
SELECT
    u.USER_ID,
    u.TotalReceipts,
    u.UniqueStores,
    u.UniqueBrands
FROM UserMetrics u
CROSS JOIN Thresholds t
WHERE
    u.TotalReceipts > t.AvgReceipts * 1.5 AND -- 50% above average
    u.UniqueStores > t.AvgStores AND
    u.UniqueBrands > t.AvgBrands
ORDER BY u.TotalReceipts DESC;
```

I have filtered out users who scanned more than 1.5* average receipts and has more than average stores and brands' receipts scanned. This approach identifies users who significantly contribute to Fetch's success and demonstrate strong engagement. These power users could be targeted for personalized rewards, retention efforts, and insights into user behavior.

- Which is the leading brand in the Dips & Salsa category?

```
SELECT
    p.BRAND,
    COUNT(t.RECEIPT_ID) AS TotalReceipts,
```

```
    SUM(t.FINAL_SALE) AS TotalSales
FROM
    product p
JOIN
    transaction t
ON
    p.BARCODE = t.BARCODE
WHERE
    p.CATEGORY_2 = 'Dips & Salsa' -- Filtering for the Dips & Salsa category
GROUP BY
    p.BRAND
ORDER BY
    TotalSales DESC -- Sort by total sales to identify the leading brand
LIMIT 3; -- Return the top 3 results
```

- At what percent has Fetch grown year over year?

```
-- Step 1: Calculate total receipts for each year
WITH YearlyReceipts AS (
    SELECT
        YEAR(t.PURCHASE_DATE) AS Year, -- Extract the year from the purchase date
        COUNT(DISTINCT t.RECEIPT_ID) AS TotalReceipts -- Count unique receipts for each year
    FROM
        transaction t -- Source table containing transaction data
    GROUP BY
        YEAR(t.PURCHASE_DATE) -- Group data by year
),

-- Step 2: Calculate Year-over-Year growth percentage
GrowthRate AS (
    SELECT
        Year, -- Year of the data
```

```
        TotalReceipts, -- Total receipts for the year
        LAG(TotalReceipts) OVER (ORDER BY Year) AS PreviousYearReceipts, -- Fetch receipts from the previous year
        ROUND(
            (TotalReceipts - LAG(TotalReceipts) OVER (ORDER BY Year)) -- Difference between current and previous year receipts
            / NULLIF(LAG(TotalReceipts) OVER (ORDER BY Year), 0) * 100, -- Divide by previous year receipts and multiply by 100 for percentage
            2 -- Round the result to 2 decimal places
        ) AS YoYGrowthPercent -- Year-over-Year growth percentage
    FROM
        YearlyReceipts -- Use the yearly receipts calculated in the previous step
)

-- Step 3: Retrieve and display the results
SELECT
    Year, -- Year of the data
    TotalReceipts, -- Total number of receipts for the year
    PreviousYearReceipts, -- Receipts from the previous year
    YoYGrowthPercent -- Calculated Year-over-Year growth percentage
FROM
    GrowthRate -- Use the growth rate data calculated in the previous step
ORDER BY Year; -- Sort results by year for better readability
```