Insights on consumer behavior, top brands, and growth trends

1. **Top 5 brands by sales among users that have had their account for at least six months?**

   a. Create a CTE `eligible_users` to calculate months since each account creation.

   b. Use CTE `total_sales_amount` to join `transactions`, `eligible_users`, and `products` on `user_id` and `barcode`. Calculate total sales per brand, rank by sales, filter for accounts ≥6 months, and exclude null brands.

   c. Select the top 5 brands from `total_sales_amount`.

```sql
%%sql
WITH eligible_users AS (
    SELECT
        id,
        created_date,
        -- Calculate the months since the account was created
        (strftime('%Y', 'now') - strftime('%Y', created_date)) * 12 +
        (strftime('%m', 'now') - strftime('%m', created_date)) AS months_since_created
    FROM users
),
total_sales_amount AS (
    SELECT p.brand,months_since_created,
    SUM(t.sale) AS total_sales,
    ROW_NUMBER() OVER (ORDER BY SUM(t.sale) DESC) AS rank
    FROM transactions t
    JOIN eligible_users u ON t.user_id = u.id
    JOIN products p on t.barcode = p.barcode
    WHERE months_since_created >= 6
    AND p.brand IS NOT NULL
    GROUP BY p.brand
)
SELECT brand, total_sales
FROM total_sales_amount
WHERE rank <=5
ORDER BY total_sales DESC
```

brand	total_sales
CVS	72.0
TRIDENT	46.72
DOVE	42.88
COORS LIGHT	34.96
AXE	15.98

2. **Top 5 leading brand in the Dips & Salsa category**

   This SQL identifies the **top 5 brands** in "Dips & Salsa" by **market share and receipt count**:

   1. `dips_and_salsa` **CTE**: Filters products in "Dips & Salsa."

   2. `brand_sales` **CTE**: Calculates each brand's total sales and unique receipt count.

   3. `total_category_sales` **CTE**: Computes overall sales in the category.

   4. **Final Query**: Calculates each brand's market share, ranks by market share and receipt count, and selects the top 5.

   Result: "TOSTITOS" leads in total sales, receipt count, and market share, followed by **GOOD FOODS, PACE, MARKETSIDE,** and **FRITOS**.

```sql
%%sql
WITH dips_and_salsa AS (
    SELECT p.brand, p.barcode
    FROM products p
    WHERE LOWER(p.category_2) = 'dips & salsa'
    AND p.brand IS NOT NULL
),
brand_sales AS (
    SELECT
        ds.brand,
        COUNT(DISTINCT t.receipt_id) AS receipt_count,
        SUM(t.sale) AS brand_total_sales
    FROM transactions t
    JOIN dips_and_salsa ds ON t.barcode = ds.barcode
    GROUP BY ds.brand
),
total_category_sales AS (
    SELECT SUM(brand_total_sales) AS total_sales
    FROM brand_sales
)
SELECT
    bs.brand,
    bs.brand_total_sales,
    bs.receipt_count,
    ROUND((bs.brand_total_sales * 100.0 / tcs.total_sales), 2) AS market_share_percent
FROM brand_sales bs
JOIN total_category_sales tcs ON 1=1
ORDER BY market_share_percent DESC, receipt_count DESC
LIMIT 5;
```

brand	brand_total_sales	receipt_count	market_share_percent
TOSTITOS	260.99	36	14.1
GOOD FOODS	118.89	9	6.42
PACE	118.58	24	6.41
MARKETSIDE	103.29	16	5.58
FRITOS	91.73	19	4.96

3. **Year-over-Year User Growth and Trends**

   - **Growth Metric**: Growth is measured by annual user account creation.

- **Date Reliability**: `created_date` accurately reflects account creation year.

- **Exclusion of NULL Dates**: Users without `created_date` are excluded.

- **YOY Calculation**: YOY growth is the percentage change in user count between consecutive years; the first year has no growth calculation.

```sql
%%sql
WITH yearly_user_counts AS (
    -- Step 1: Calculate user count per year
    SELECT
        STRFTIME('%Y', created_date) AS year,
        COUNT(*) AS user_count
    FROM users
    WHERE created_date IS NOT NULL
    GROUP BY year
    ORDER BY year
)
SELECT
    year,
    user_count,
    previous_year_count,
    CASE
        WHEN previous_year_count IS NULL THEN NULL
        ELSE ROUND(((user_count - previous_year_count) * 100.0) / previous_year_count, 2)
    END AS yoy_growth_percentage
FROM (
    -- Step 2: Add previous year count only once
    SELECT
        year,
        user_count,
        LAG(user_count) OVER (ORDER BY year) AS previous_year_count
    FROM yearly_user_counts
) AS growth_data
ORDER BY year;
```

year	user_count	previous_year_count	yoy_growth_percentage
2014	30	None	None
2015	51	30	70.0
2016	70	51	37.25
2017	644	70	820.0
2018	2168	644	236.65
2019	7093	2168	227.17
2020	16883	7093	138.02
2021	19159	16883	13.48
2022	26807	19159	39.92
2023	15464	26807	-42.31
2024	11631	15464	-24.79

Year over Year Growth