Fetch Data Analyst Assessment

- Arjun Kumaran

Part – 2 Provide SQL queries

Closed-ended questions:

- What are the top 5 brands by receipts scanned among users 21 and over?
- What are the top 5 brands by sales among users that have had their account for at least six months?
- What is the percentage of sales in the Health & Wellness category by generation?

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

- Who are Fetch's power users?
- Which is the leading brand in the Dips & Salsa category?
- At what percent has Fetch grown year over year?

DB used: **PostgreSQL**

Tool used: **DBeaver**

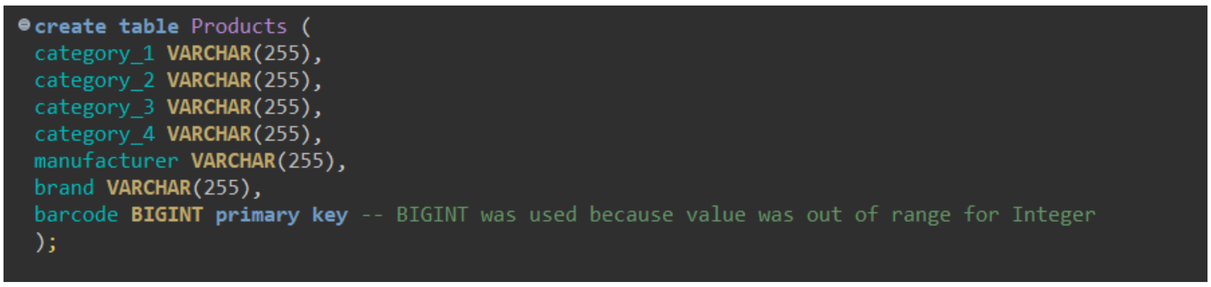
<u>Table Creation:</u>

Products Table:

Out of 845337 values, 841342 values in the Barcode column are unique, 4025 values are null and there are no empty strings. This has been used as the primary key for the products table after removing the duplicates and null values.

The reason why Barcode has been used as a primary key here is because one barcode will link only to one product. It is a unique Identifier.

Code:

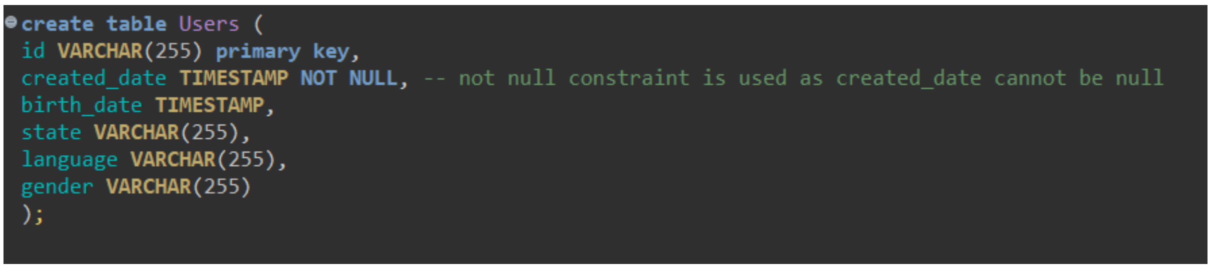


```sql
create table Products (
category_1 VARCHAR(255),
category_2 VARCHAR(255),
category_3 VARCHAR(255),
category_4 VARCHAR(255),
manufacturer VARCHAR(255),
brand VARCHAR(255),
barcode BIGINT primary key -- BIGINT was used because value was out of range for Integer
);
```

Users Table:

All the values in the ID column are unique, no null values and there are no empty strings. This can be used as the primary key in SQL for the Users table.

Code:



```sql
create table Users (
id VARCHAR(255) primary key,
created_date TIMESTAMP NOT NULL, -- not null constraint is used as created_date cannot be null
birth_date TIMESTAMP,
state VARCHAR(255),
language VARCHAR(255),
gender VARCHAR(255)
);
```

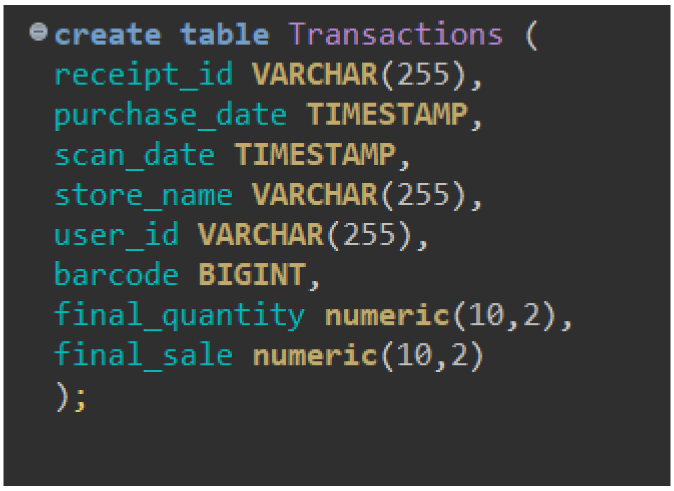
Transaction Table:

Out of 49829 values, there are only 21639 unique values in the receipt ID column and 11027 unique barcodes. I believe receipt Ids are not unique because multiple products would be listed in a single receipt and each product would be in a separate row. In the case of barcode, different receipts can have the same products.

Hence, Receipt_ID cannot be used as the primary key for this table. A possible solution in the future would be to keep another column with sequential product numbering for each receipt and use that column with receipt id to create a composite key.

In this case, I am not creating a primary key for this table.

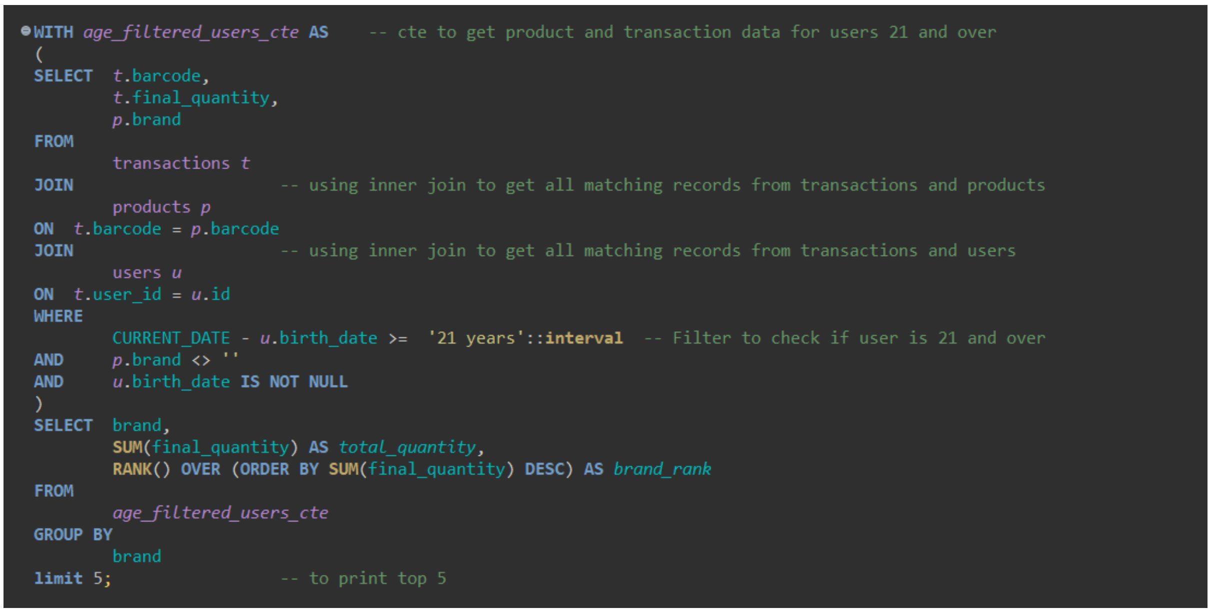
Code:



```sql
create table Transactions (
receipt_id VARCHAR(255),
purchase_date TIMESTAMP,
scan_date TIMESTAMP,
store_name VARCHAR(255),
user_id VARCHAR(255),
barcode BIGINT,
final_quantity numeric(10,2),
final_sale numeric(10,2)
);
```

Queries:

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

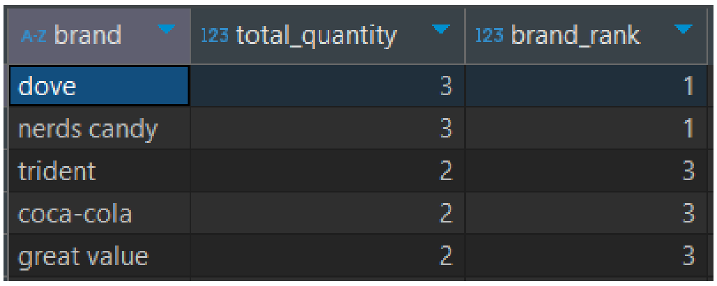


```sql
WITH age_filtered_users_cte AS    -- cte to get product and transaction data for users 21 and over
(
SELECT  t.barcode,
        t.final_quantity,
        p.brand
FROM
        transactions t
JOIN                     -- using inner join to get all matching records from transactions and products
        products p
ON  t.barcode = p.barcode
JOIN                     -- using inner join to get all matching records from transactions and users
        users u
ON  t.user_id = u.id
WHERE
        CURRENT_DATE - u.birth_date >=  '21 years'::interval  -- Filter to check if user is 21 and over
AND     p.brand <> ''
AND     u.birth_date IS NOT NULL
)
SELECT  brand,
        SUM(final_quantity) AS total_quantity,
        RANK() OVER (ORDER BY SUM(final_quantity) DESC) AS brand_rank
FROM
        age_filtered_users_cte
GROUP BY
        brand
limit 5;                 -- to print top 5
```

I’ve created a CTE (Common Table Expression) to get the product and transaction data for users 21 and over. Firstly, I’ve selected barcode and final_quantity from transaction table and brand from product table. I’ve joined the tables using INNER JOIN to get all the matching records from the transactions, products and users table. The data is filtered using the WHERE Statement. In the WHERE condition, the age (difference between the current date and the user’s birth date) should be greater than or equal to 21.The brand and birth_date values should not be null.

From the CTE, I’ve selected the brand, total_quantity and brand_rank. Brand_rank is calculated by using the RANK function where the data is ordered by total_quantity in descending order. The data is then grouped by brand and the result are limited to display the top 5 brands.

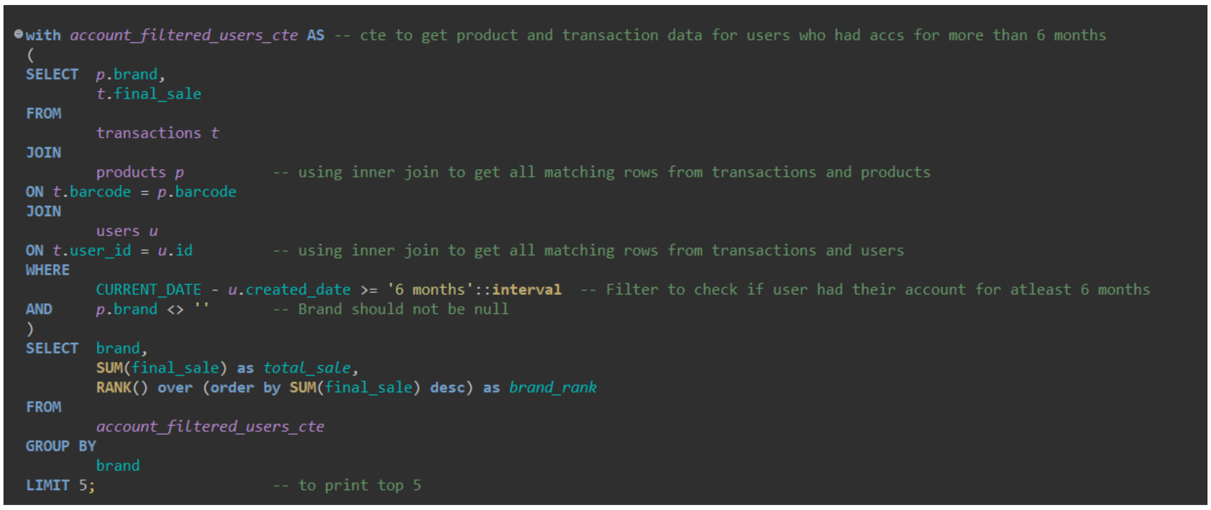
Output:



brand	total_quantity	brand_rank
dove	3	1
nerds candy	3	1
trident	2	3
coca-cola	2	3
great value	2	3

The Top 5 Brands are Dove, Nerds Candy, Trident, Coca-Cola and Great Value.

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

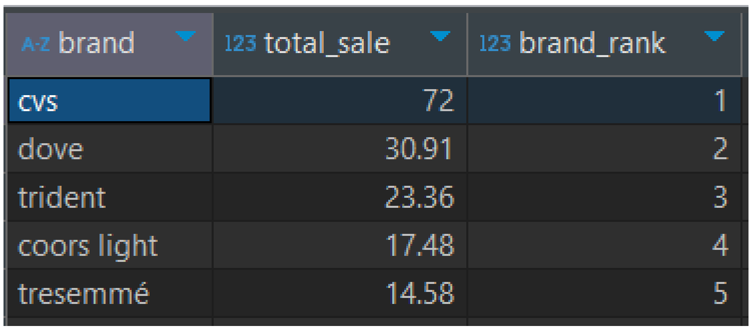


```
with account_filtered_users_cte AS -- cte to get product and transaction data for users who had accs for more than 6 months
(
SELECT  p.brand,
        t.final_sale
FROM
        transactions t
JOIN
        products p          -- using inner join to get all matching rows from transactions and products
ON t.barcode = p.barcode
JOIN
        users u
ON t.user_id = u.id         -- using inner join to get all matching rows from transactions and users
WHERE
        CURRENT_DATE - u.created_date >= '6 months'::interval  -- Filter to check if user had their account for atleast 6 months
AND     p.brand <> ''       -- Brand should not be null
)
SELECT  brand,
        SUM(final_sale) as total_sale,
        RANK() over (order by SUM(final_sale) desc) as brand_rank
FROM
        account_filtered_users_cte
GROUP BY
        brand
LIMIT 5;                    -- to print top 5
```

I’ve created a CTE to get the product and transaction data for users who’ve had their account for more than six months. I’ve used INNER JOIN to get the matching rows from all the tables. In the Where condition, the difference between created date and current date should be greater than or equal to 6 months and the brand value cannot be null.

From this CTE, I’ve select the brand, total_sale and brand_rank. The brand rank value is calculated by using the RANK function where the table is ordered by total_sale in descending order and then the data is grouped by brand and the results are limited to display the top 5 brands.

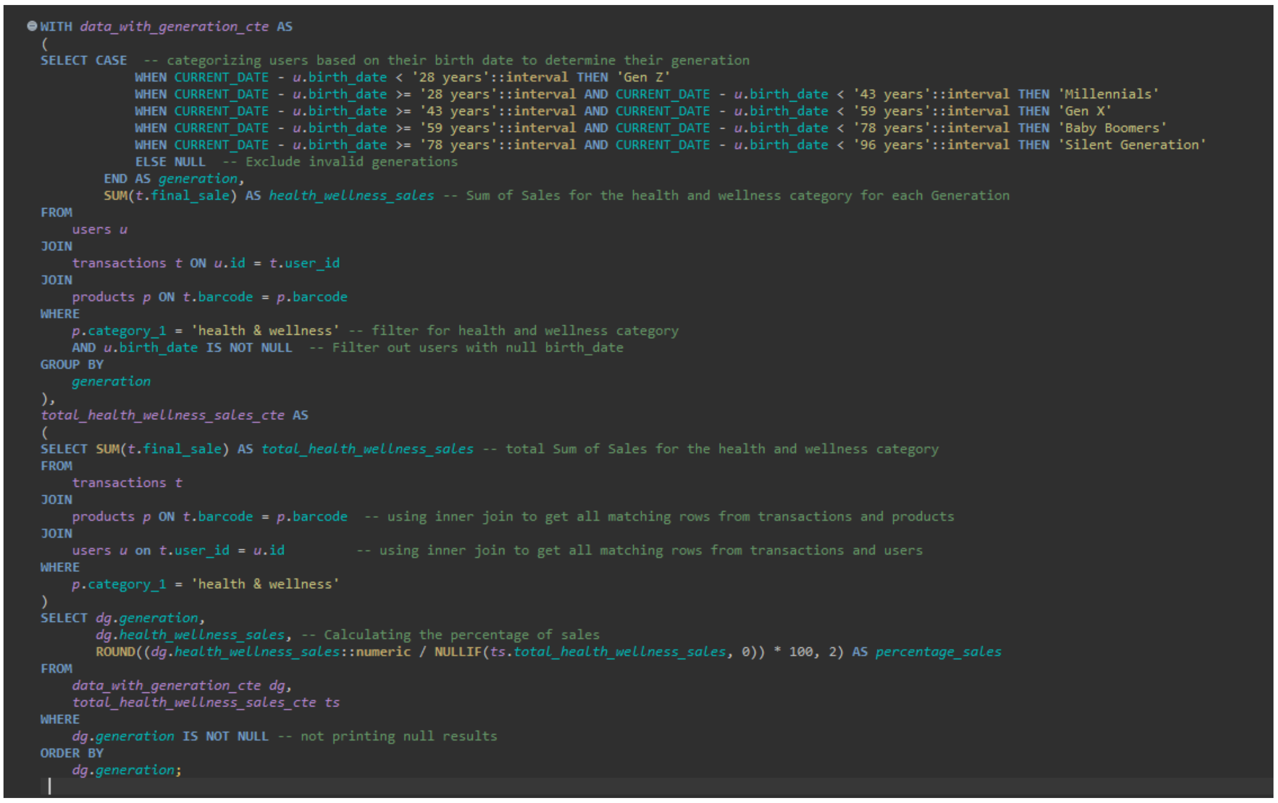
Output:



brand	total_sale	brand_rank
cvs	72	1
dove	30.91	2
trident	23.36	3
coors light	17.48	4
tresemmé	14.58	5

The top 5 brands are CVS, Dove, Trident, Coors light and Tresemme.

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

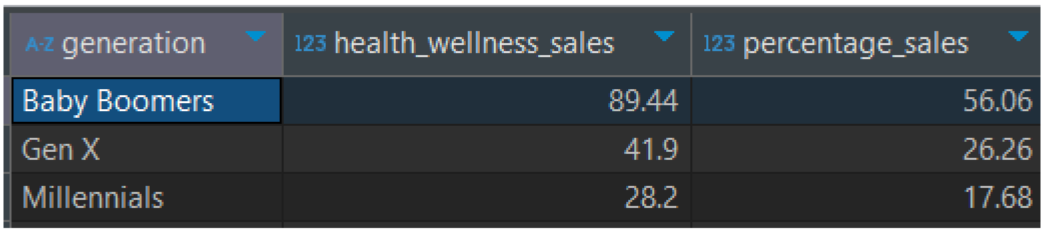


```sql
WITH data_with_generation_cte AS
(
SELECT CASE  -- categorizing users based on their birth date to determine their generation
            WHEN CURRENT_DATE - u.birth_date < '28 years'::interval THEN 'Gen Z'
            WHEN CURRENT_DATE - u.birth_date >= '28 years'::interval AND CURRENT_DATE - u.birth_date < '43 years'::interval THEN 'Millennials'
            WHEN CURRENT_DATE - u.birth_date >= '43 years'::interval AND CURRENT_DATE - u.birth_date < '59 years'::interval THEN 'Gen X'
            WHEN CURRENT_DATE - u.birth_date >= '59 years'::interval AND CURRENT_DATE - u.birth_date < '78 years'::interval THEN 'Baby Boomers'
            WHEN CURRENT_DATE - u.birth_date >= '78 years'::interval AND CURRENT_DATE - u.birth_date < '96 years'::interval THEN 'Silent Generation'
            ELSE NULL  -- Exclude invalid generations
        END AS generation,
        SUM(t.final_sale) AS health_wellness_sales -- Sum of Sales for the health and wellness category for each Generation
FROM
    users u
JOIN
    transactions t ON u.id = t.user_id
JOIN
    products p ON t.barcode = p.barcode
WHERE
    p.category_1 = 'health & wellness' -- filter for health and wellness category
    AND u.birth_date IS NOT NULL  -- Filter out users with null birth_date
GROUP BY
    generation
),
total_health_wellness_sales_cte AS
(
SELECT SUM(t.final_sale) AS total_health_wellness_sales -- total Sum of Sales for the health and wellness category
FROM
    transactions t
JOIN
    products p ON t.barcode = p.barcode  -- using inner join to get all matching rows from transactions and products
JOIN
    users u on t.user_id = u.id         -- using inner join to get all matching rows from transactions and users
WHERE
    p.category_1 = 'health & wellness'
)
SELECT dg.generation,
       dg.health_wellness_sales, -- Calculating the percentage of sales
       ROUND((dg.health_wellness_sales::numeric / NULLIF(ts.total_health_wellness_sales, 0)) * 100, 2) AS percentage_sales
FROM
    data_with_generation_cte dg,
    total_health_wellness_sales_cte ts
WHERE
    dg.generation IS NOT NULL -- not printing null results
ORDER BY
    dg.generation;
```

In the First CTE, I've calculated generations based on birth_date using CASE WHEN statements and calculated the sum of sales for the health and wellness category for each generation by filtering using the WHERE statement and I also filtered out rows that had null values in the birth_date column.

In the Second CTE, I've calculated the total health and wellness sales. Using Both CTEs, I've selected the generation,sale per generation and calculated the percentage of sales per generation relative to total sales. The Final result was filtered to not display null values and was ordered by generation.

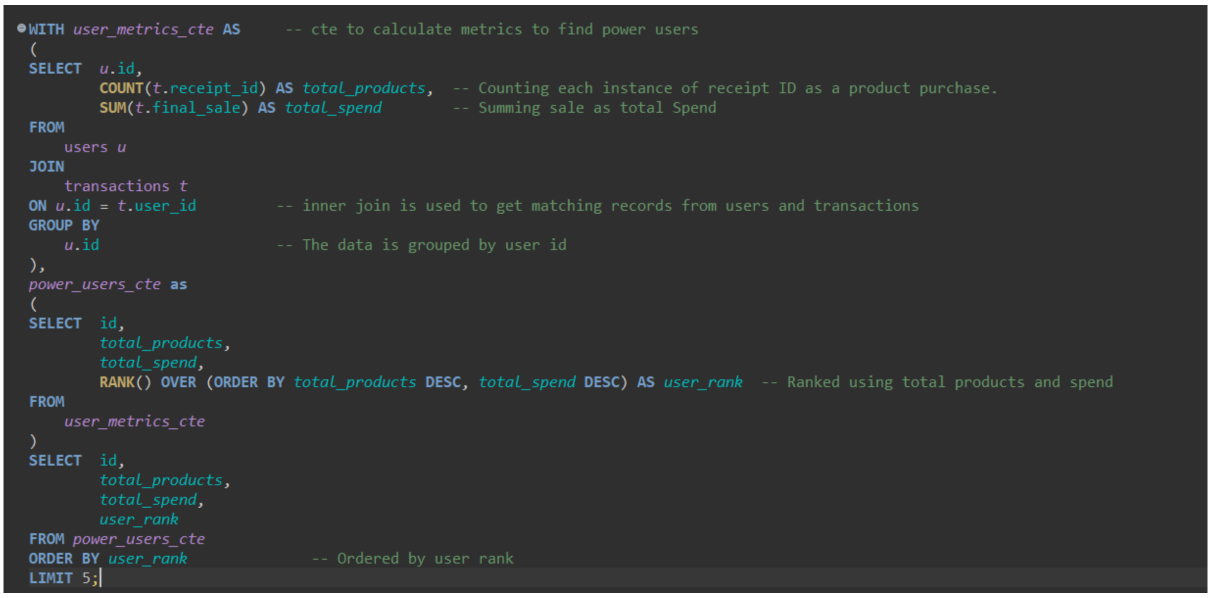
Output:



generation	health_wellness_sales	percentage_sales
Baby Boomers	89.44	56.06
Gen X	41.9	26.26
Millennials	28.2	17.68

Baby Boomers contributed to 56% sales in the Health and Wellness Category.

Who are Fetch’s power users?



```
WITH user_metrics_cte AS     -- cte to calculate metrics to find power users
(
SELECT  u.id,
        COUNT(t.receipt_id) AS total_products,  -- Counting each instance of receipt ID as a product purchase.
        SUM(t.final_sale) AS total_spend        -- Summing sale as total Spend
FROM
    users u
JOIN
    transactions t
ON u.id = t.user_id          -- inner join is used to get matching records from users and transactions
GROUP BY
    u.id                     -- The data is grouped by user id
),
power_users_cte as
(
SELECT  id,
        total_products,
        total_spend,
        RANK() OVER (ORDER BY total_products DESC, total_spend DESC) AS user_rank  -- Ranked using total products and spend
FROM
    user_metrics_cte
)
SELECT  id,
        total_products,
        total_spend,
        user_rank
FROM power_users_cte
ORDER BY user_rank               -- Ordered by user rank
LIMIT 5;
```

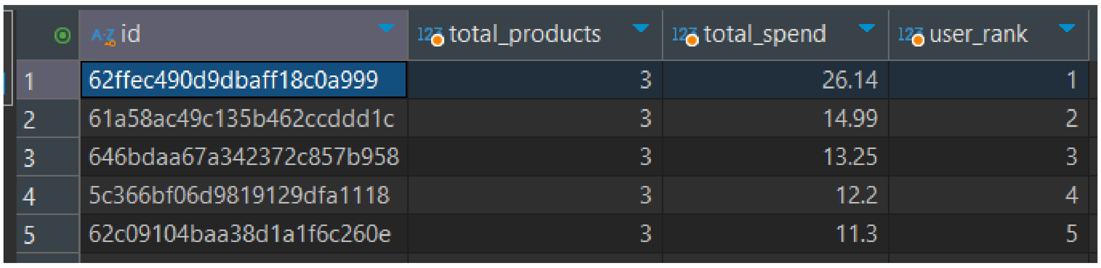
I have assumed users who have **bought the most items** and have **spent the highest amount of money** to be power users.

In the First CTE, I’ve selected the id, total_products (counts receipt_id to determine total products purchased) and total_spend (sum of final sale). I’ve used INNER JOIN between the tables to get matching rows and I’ve grouped by user id.

In the second CTE, I’ve selected user id, total_products and total_spend and user_ rank. User_Rank was calculated using the RANK FUNCTION where the table is ordered bu total products and total spend in descending order to get highest value on top.

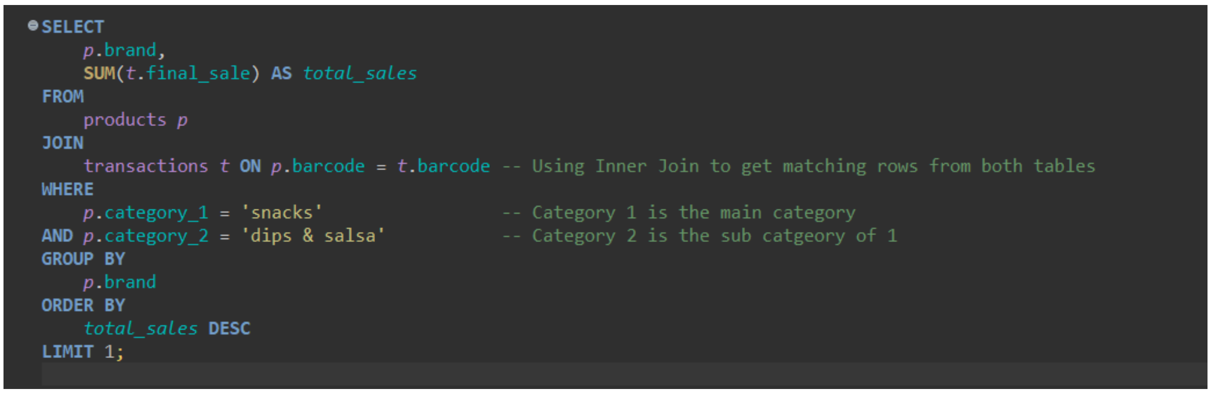
From the Second CTE, I’ve selected all the metrics and limited the results by 5 to display the top 5 power users.

OUTPUT:



	id	total_products	total_spend	user_rank
1	62ffec490d9dbaff18c0a999	3	26.14	1
2	61a58ac49c135b462ccddd1c	3	14.99	2
3	646bdaa67a342372c857b958	3	13.25	3
4	5c366bf06d9819129dfa1118	3	12.2	4
5	62c09104baa38d1a1f6c260e	3	11.3	5

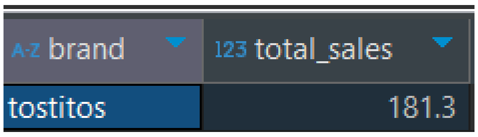
Which is the leading brand in the Dips & Salsa category?



```sql
SELECT
    p.brand,
    SUM(t.final_sale) AS total_sales
FROM
    products p
JOIN
    transactions t ON p.barcode = t.barcode -- Using Inner Join to get matching rows from both tables
WHERE
    p.category_1 = 'snacks'               -- Category 1 is the main category
AND p.category_2 = 'dips & salsa'         -- Category 2 is the sub catgeory of 1
GROUP BY
    p.brand
ORDER BY
    total_sales DESC
LIMIT 1;
```

I've selected the brand and summed the final_sale column as total_sales from products and transactions table. I've joined those tables using INNER JOIN and then grouped the data by BRAND. The table is then ordered byy total_sales in descending order and the result is limited to 1 to show the leading brand.

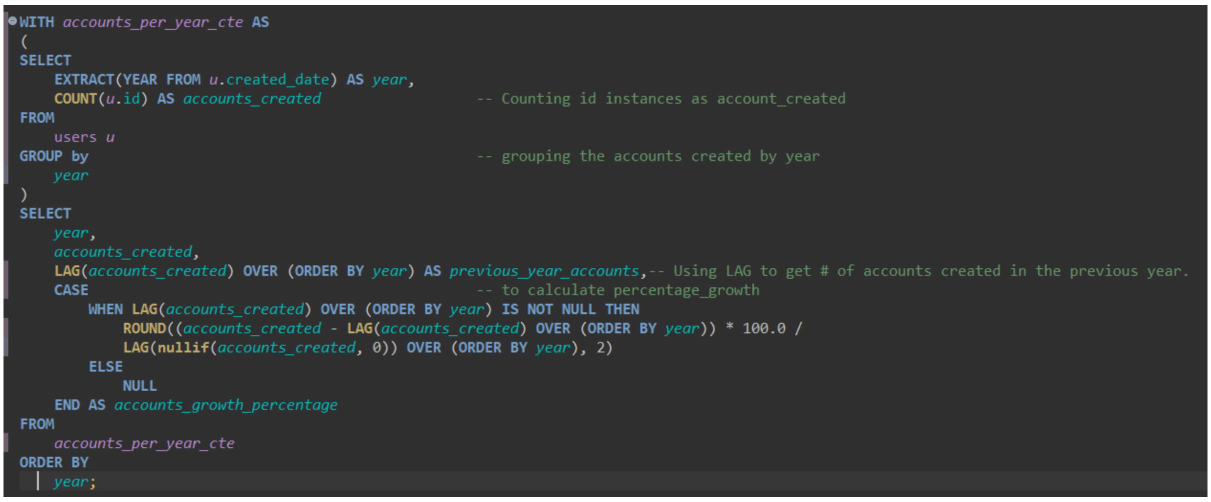
Output:



brand	total_sales
tostitos	181.3

The leading brand in the Dips and Salsa Category is Tostitos.

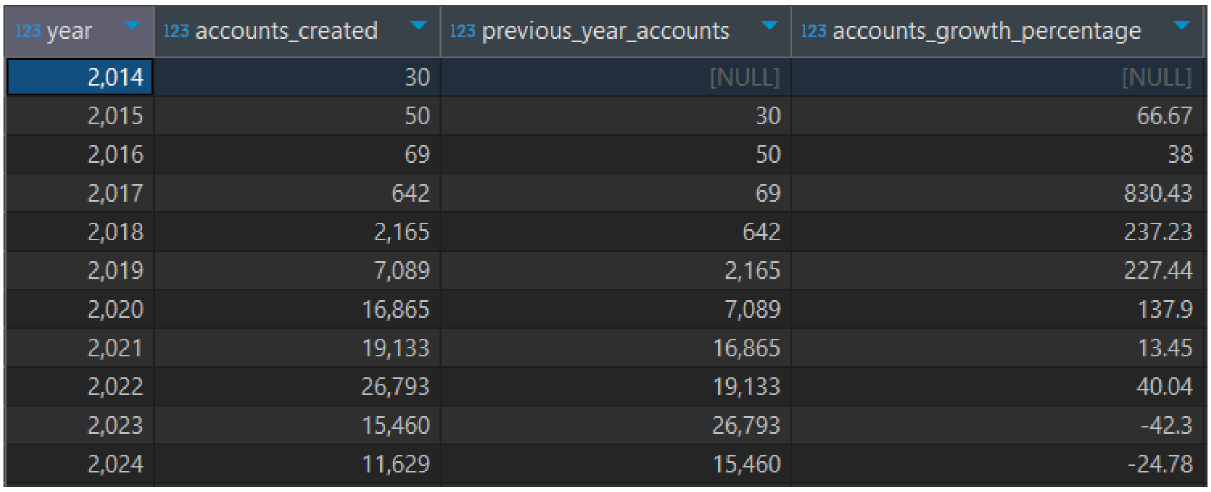
At what percent has Fetch grown year over year?



```sql
WITH accounts_per_year_cte AS
(
SELECT
    EXTRACT(YEAR FROM u.created_date) AS year,
    COUNT(u.id) AS accounts_created                 -- Counting id instances as account_created
FROM
    users u
GROUP by                                            -- grouping the accounts created by year
    year
)
SELECT
    year,
    accounts_created,
    LAG(accounts_created) OVER (ORDER BY year) AS previous_year_accounts,-- Using LAG to get # of accounts created in the previous year.
    CASE                                            -- to calculate percentage_growth
        WHEN LAG(accounts_created) OVER (ORDER BY year) IS NOT NULL THEN
            ROUND((accounts_created - LAG(accounts_created) OVER (ORDER BY year)) * 100.0 /
            LAG(nullif(accounts_created, 0)) OVER (ORDER BY year), 2)
        ELSE
            NULL
    END AS accounts_growth_percentage
FROM
    accounts_per_year_cte
ORDER BY
    year;
```

I've created a CTE to count the number of accounts created and it is grouped by year. From that CTE, I've selected year, accounts created and used LAG function to fetch the number of accounts created in the previous year. I've then used CASE WHEN statement to calculate percentage growth year over year (YoY). The Result is then order by year.

Output:



year	accounts_created	previous_year_accounts	accounts_growth_percentage
2,014	30	[NULL]	[NULL]
2,015	50	30	66.67
2,016	69	50	38
2,017	642	69	830.43
2,018	2,165	642	237.23
2,019	7,089	2,165	227.44
2,020	16,865	7,089	137.9
2,021	19,133	16,865	13.45
2,022	26,793	19,133	40.04
2,023	15,460	26,793	-42.3
2,024	11,629	15,460	-24.78

From this table, it can be seen that from 2014 to 2022, the accounts created were increasing every year but from 2023, it has been on a decline.