

Q1 What are the top 5 brands by receipts scanned among users 21 and over?

```
912 -- Common Table Expression (CTE) to filter out transactions with a final sale value of zero
913 WITH transaction_data_sale_not_null AS (
914     SELECT * FROM TRANSACTION_TAKEHOME
915     WHERE CAST(FINAL_SALE AS FLOAT) != 0.00,
916 -- CTE to concatenate receipt_id and barcode (primary key) and grouping them to remove duplicates
917 transaction_data_sale_no_duplicates AS (
918     SELECT
919         (RECEIPT_ID || COALESCE(BARCODE, '0')) AS combined_pk,
920         RECEIPT_ID,
921         BARCODE,
922         USER_ID,
923         FINAL_SALE
924     FROM
925         transaction_data_sale_not_null
926     GROUP BY combined_pk)
927 -- Main select query
928 SELECT
929     p.BRAND,
930     COUNT(t.RECEIPT_ID) AS receipt_count -- Using count of receipt id to check count of receipts scanned
931 FROM
932     transaction_data_sale_no_duplicates AS t
933 JOIN
934     USER_TAKEHOME AS u ON t.USER_ID = u.ID
935 JOIN
936     PRODUCTS_TAKEHOME AS p ON t.BARCODE = p.BARCODE
937 WHERE
938     strftime('%Y', 'now') - strftime('%Y', u.BIRTH_DATE) >= 21 -- Filtering for users > 21
939     AND p.BRAND <> '' -- Removing cases where brand is blank
940 GROUP BY p.BRAND ORDER BY receipt_count DESC LIMIT 5;
```

! BRAND	receipt_count
NERDS CANDY	3
DOVE	3
TRIDENT	2
SOUR PATCH KIDS	2
MEIJER	2

```

-- Common Table Expression (CTE) to filter out transactions with a final sale value of zero

WITH transaction_data_sale_not_null AS (

    SELECT * FROM TRANSACTION_TAKEHOME

    WHERE CAST(FINAL_SALE AS FLOAT) != 0.00),

-- CTE to concatenate receipt_id and barcode (primary key) and grouping them to remove duplicates
transaction_data_sale_no_duplicates AS (

    SELECT

        (RECEIPT_ID || COALESCE(BARCODE, '0')) AS combined_pk,

        RECEIPT_ID,

        BARCODE,

        USER_ID,

        FINAL_SALE

    FROM

        transaction_data_sale_not_null

    GROUP BY combined_pk)

-- Main select query

SELECT

    p.BRAND,

    COUNT(t.RECEIPT_ID) AS receipt_count -- Using count of receipt id to check count of receipts scanned

FROM

    transaction_data_sale_no_duplicates AS t

JOIN

    USER_TAKEHOME AS u ON t.USER_ID = u.ID

JOIN

    PRODUCTS_TAKEHOME AS p ON t.BARCODE = p.BARCODE

WHERE

    strftime('%Y', 'now') - strftime('%Y', u.BIRTH_DATE) >= 21 -- Filtering for users > 21

    AND p.BRAND <> '' -- Removing cases where brand is blank

GROUP BY p.BRAND ORDER BY receipt_count DESC LIMIT 5;

```

Q2 What are the top 5 brands by sales among users that have had their account for at least six months?

```
943 -- Common Table Expression (CTE) to filter out transactions with a final sale value of zero
944 WITH transaction_data_sale_not_null AS (
945     SELECT * FROM TRANSACTION_TAKEHOME
946     WHERE CAST(FINAL_SALE AS FLOAT) != 0.00 ),
947 -- CTE to concatenate receipt_id and barcode (primary key) and grouping them to remove duplicates
948 transaction_data_sale_no_duplicates AS (
949     SELECT
950         (RECEIPT_ID || COALESCE(BARCODE, '0')) AS combined_pk,
951         RECEIPT_ID,
952         BARCODE,
953         USER_ID,
954         FINAL_SALE
955     FROM
956         transaction_data_sale_not_null
957     GROUP BY combined_pk)
958 -- Main select query
959 SELECT
960     p.BRAND,
961     SUM(CAST(t.FINAL_SALE AS FLOAT)) AS total_sales -- Calculating total sales per brand
962 FROM
963     transaction_data_sale_no_duplicates AS t
964 JOIN
965     USER_TAKEHOME AS u ON t.USER_ID = u.ID -- Join the filtered transactions with the USER_TAKEHOME table on user ID
966 JOIN
967     PRODUCTS_TAKEHOME AS p ON t.BARCODE = p.BARCODE -- Join the resulting table with PRODUCTS_TAKEHOME on barcode to access product details
968 WHERE
969     DATE(u.CREATED_DATE) <= DATE('now', '-6 months') -- Filter to include transactions where the user was created at least 6 months ago
970     AND p.BRAND IS NOT NULL -- Filter out any records where the product brand is null
971 GROUP BY p.BRAND ORDER BY total_sales DESC LIMIT 5;
```

! BRAND	total_sales
CVS	72
DOVE	30.91
TRIDENT	23.36
COORS LIGHT	17.48
TRESEMME	14.58

```

-- Common Table Expression (CTE) to filter out transactions with a final sale value of zero

WITH transaction_data_sale_not_null AS (

    SELECT

        *

    FROM

        TRANSACTION_TAKEHOME

    WHERE

        CAST(FINAL_SALE AS FLOAT) != 0.00

),

-- CTE to concatenate receipt_id and barcode (primary key) and grouping them to remove duplicates

transaction_data_sale_no_duplicates AS (

    SELECT

        (RECEIPT_ID || COALESCE(BARCODE, '0')) AS combined_pk,

        RECEIPT_ID,

        BARCODE,

        USER_ID,

        FINAL_SALE

    FROM

        transaction_data_sale_not_null

    GROUP BY

        combined_pk

)

-- Main select query

SELECT

    p.BRAND,

    SUM(CAST(t.FINAL_SALE AS FLOAT)) AS total_sales -- Calculating total sales per brand

FROM

    transaction_data_sale_no_duplicates AS t

JOIN

    USER_TAKEHOME AS u ON t.USER_ID = u.ID -- Join the filtered transactions with the USER_TAKEHOME table on user ID

JOIN

    PRODUCTS_TAKEHOME AS p ON t.BARCODE = p.BARCODE -- Joining the table with PRODUCTS_TAKEHOME on barcode to access product details

WHERE

    DATE(u.CREATED_DATE) <= DATE('now', '-6 months') -- Filter to include transactions where the user was created at least 6 months ago

    AND p.BRAND IS NOT NULL -- Filter out any records where the product brand is null

GROUP BY

    p.BRAND

ORDER BY    total_sales DESC LIMIT 5;

```

Q3 What is the percentage of sales in the Health & Wellness category by generation? – I have defined generations based on age

```
976 -- Common Table Expression (CTE) to filter out transactions with a final sale value of zero
977 WITH transaction_data_sale_not_null AS (
978     SELECT * FROM TRANSACTION TAKEHOME WHERE CAST(FINAL_SALE AS FLOAT) != 0.00),
979 -- CTE to concatenate receipt_id and barcode (primary key) and grouping them to remove duplicates
980 transaction_data_sale_no_duplicates AS (
981     SELECT
982         (RECEIPT_ID || COALESCE(BARCODE, '0')) AS combined_pk, RECEIPT_ID, BARCODE, USER_ID, FINAL_SALE
983     FROM transaction_data_sale_not_null GROUP BY combined_pk)
984 -- Main select query
985 SELECT CASE -- Creating generations based on birth date
986     WHEN strftime('%Y', 'now') - strftime('%Y', u.birth_date) >= 76 THEN 'Silent Generation'
987     WHEN strftime('%Y', 'now') - strftime('%Y', u.birth_date) BETWEEN 57 AND 75 THEN 'Baby Boomers'
988     WHEN strftime('%Y', 'now') - strftime('%Y', u.birth_date) BETWEEN 42 AND 56 THEN 'Gen X'
989     WHEN strftime('%Y', 'now') - strftime('%Y', u.birth_date) BETWEEN 27 AND 41 THEN 'Millennials'
990     WHEN strftime('%Y', 'now') - strftime('%Y', u.birth_date) <= 26 THEN 'Gen Z'
991 END AS Generation,
992 SUM(CAST(t.FINAL_SALE AS FLOAT)) AS generation_sales,
993 ROUND(SUM(CAST(t.FINAL_SALE AS FLOAT)) * 100.0 / (
994     SELECT
995         SUM(CAST(tr.FINAL_SALE AS FLOAT))
996     FROM
997         transaction_data_sale_no_duplicates AS tr
998 JOIN
999     PRODUCTS TAKEHOME AS pr ON tr.BARCODE = pr.BARCODE -- Join transaction and product tables on barcode.
1000 JOIN
1001     USER TAKEHOME AS ur ON tr.USER_ID = ur.ID -- Join transaction and user tables on user ID
1002 WHERE
1003     pr.CATEGORY_1 = 'Health & Wellness' -- Filter for 'Health & Wellness' category products only.
1004 ), 2) AS percentage_of_sales
1005 FROM
1006     transaction_data_sale_no_duplicates AS t
1007 JOIN
1008     USER TAKEHOME AS u ON t.USER_ID = u.ID
1009 JOIN
1010     PRODUCTS TAKEHOME AS p ON t.BARCODE = p.BARCODE
1011 WHERE p.CATEGORY_1 = 'Health & Wellness' GROUP BY Generation ORDER BY percentage_of_sales DESC;
1012
```

! Generation	generation_sales	percentage_of_sales
Baby Boomers	88.94	55.75

Output:

! Generation	generation_sales	percentage_of_sales
Baby Boomers	88.94	55.75
Gen X	48.42	30.35
Millennials	20.21	12.67
Silent Generation	1.97	1.23

```

-- Common Table Expression (CTE) to filter out transactions with a final sale value of zero

WITH transaction_data_sale_not_null AS (

    SELECT * FROM TRANSACTION_TAKEHOME WHERE CAST(FINAL_SALE AS FLOAT) != 0.00),

-- CTE to concatenate receipt_id and barcode (primary key) and grouping them to remove duplicates
transaction_data_sale_no_duplicates AS (

    SELECT

        (RECEIPT_ID || COALESCE(BARCODE, '0')) AS combined_pk, RECEIPT_ID, BARCODE, USER_ID, FINAL_SALE

    FROM transaction_data_sale_not_null GROUP BY combined_pk)

-- Main select query

SELECT CASE -- Creating generations based on birth date

    WHEN strftime('%Y', 'now') - strftime('%Y', u.birth_date) >= 76 THEN 'Silent Generation'

    WHEN strftime('%Y', 'now') - strftime('%Y', u.birth_date) BETWEEN 57 AND 75 THEN 'Baby Boomers'

    WHEN strftime('%Y', 'now') - strftime('%Y', u.birth_date) BETWEEN 42 AND 56 THEN 'Gen X'

    WHEN strftime('%Y', 'now') - strftime('%Y', u.birth_date) BETWEEN 27 AND 41 THEN 'Millennials'

    WHEN strftime('%Y', 'now') - strftime('%Y', u.birth_date) <= 26 THEN 'Gen Z'

END AS Generation,

SUM(CAST(t.FINAL_SALE AS FLOAT)) AS generation_sales,

ROUND(SUM(CAST(t.FINAL_SALE AS FLOAT)) * 100.0 / (

    SELECT

        SUM(CAST(tr.FINAL_SALE AS FLOAT))

    FROM

        transaction_data_sale_no_duplicates AS tr

    JOIN

        PRODUCTS_TAKEHOME AS pr ON tr.BARCODE = pr.BARCODE -- Join transaction and product tables on barcode.

    JOIN

        USER_TAKEHOME AS ur ON tr.USER_ID = ur.ID -- Join transaction and user tables on user ID

    WHERE

        pr.CATEGORY_1 = 'Health & Wellness' -- Filter for 'Health & Wellness' category products only.

), 2) AS percentage_of_sales

FROM

    transaction_data_sale_no_duplicates AS t

JOIN

    USER_TAKEHOME AS u ON t.USER_ID = u.ID

JOIN

    PRODUCTS_TAKEHOME AS p ON t.BARCODE = p.BARCODE

WHERE p.CATEGORY_1 = 'Health & Wellness' GROUP BY Generation ORDER BY percentage_of_sales DESC;

```

Q4. Who are Fetch's power users?

There are two ways to check power users and I have pasted results for both queries. In the first query I have defined power users as users who have 10 top users who have the highest sales value with fetch. In the second query I have defined power users as users who have 10 top users who have the most number of transactions with fetch

```
758 -- Here I have defined power users as users who have 10 top users who have the highest sales value with fetch
759 -- Common Table Expression (CTE) to filter out transactions with a final sale value of zero
760 WITH transaction_data_sale_not_null AS (
761     SELECT
762         *
763     FROM
764         TRANSACTION_TAKEHOME
765     WHERE CAST(FINAL_SALE AS FLOAT) != 0.00),
766 -- CTE to concatenate receipt_id and barcode (primary key) and grouping them to remove duplicates
767 transaction_data_sale_no_duplicates AS (
768     SELECT
769         (RECEIPT_ID || COALESCE(BARCODE, '0')) AS combined_pk, -- Using a default value for NULL BARCODES
770         RECEIPT_ID,
771         BARCODE,
772         USER_ID,
773         FINAL_SALE,
774         FINAL_QUANTITY
775     FROM
776         transaction_data_sale_not_null
777     GROUP BY
778         combined_pk
779 )
780 -- Main SELECT query to calculate the total sales value per user
781 SELECT
782     t.USER_ID,
783     SUM(CAST(t.FINAL_SALE AS FLOAT)) AS total_sales_value -- Summing up the total sales value
784 FROM
785     transaction_data_sale_no_duplicates AS t
786 JOIN
787     PRODUCTS_TAKEHOME AS p
788 ON
789     t.BARCODE = p.BARCODE
790 GROUP BY
791     t.USER_ID
792 ORDER BY
793     total_sales_value DESC LIMIT 10; -- Only retrieving the top 10 users by total sales value
794
```

! USER_ID	total_sales_value
65e4bc2716cc391732143569	85.79

Output:

! USER_ID	total_sales_value
65e4bc2716cc391732143569	85.79
6183300cf998e47aad2d6f5d	79.74
6475fd16a55bb77a0e279ee0	77.8
643059f0838dd2651fb27f50	72
5d61b8e71ddc4058bd98f776	71.97
642734743d4434e63c191488	69.2
60c0aabdc66e105658856688	64.43
62535ab0fc0da6299a70f5ca	60.46
62474ee9ee2eea7d21cc1a66	59.97
631293bb09b563dae706d55f	59.95

Query:

-- Here I have defined power users as users who have 10 top users who have the highest sales value with fetch

-- Common Table Expression (CTE) to filter out transactions with a final sale value of zero

WITH transaction_data_sale_not_null AS (

SELECT

*

FROM

TRANSACTION_TAKEHOME

WHERE CAST(FINAL_SALE AS FLOAT) != 0.00),

-- CTE to concatenate receipt_id and barcode (primary key) and grouping them to remove duplicates

transaction_data_sale_no_duplicates AS (

SELECT

(RECEIPT_ID || COALESCE(BARCODE, '0')) AS combined_pk, -- Using a default value for NULL BARCODEs

RECEIPT_ID,

BARCODE,

USER_ID,

FINAL_SALE,

FINAL_QUANTITY

FROM

transaction_data_sale_not_null

GROUP BY

combined_pk

)

-- Main SELECT query to calculate the total sales value per user

SELECT

t.USER_ID,

SUM(CAST(t.FINAL_SALE AS FLOAT)) AS total_sales_value -- Summing up the total sales value

FROM

transaction_data_sale_no_duplicates AS t

JOIN

PRODUCTS_TAKEHOME AS p

ON

t.BARCODE = p.BARCODE

GROUP BY

t.USER_ID

ORDER BY

total_sales_value DESC LIMIT 10; -- Only retrieving the top 10 users by total sales value

Here I have defined power users as users who have 10 top users who have the most number of transactions with fetch.

```
758 --Here I have defined power users as users who have 10 top users who have the most number of transactions with fetch.
759 -- Common Table Expression (CTE) to filter out transactions with a final sale value of zero
760 WITH transaction_data_sale_not_null AS (
761     SELECT
762         *
763     FROM
764         TRANSACTION_TAKEHOME
765     WHERE CAST(FINAL_SALE AS FLOAT) != 0.00),
766 -- CTE to concatenate receipt_id and barcode (primary key) and grouping them to remove duplicates
767 transaction_data_sale_no_duplicates AS (
768     SELECT
769         (RECEIPT_ID || COALESCE(BARCODE, '0')) AS combined_pk, -- Using a default value for NULL BARCODES
770         RECEIPT_ID,
771         BARCODE,
772         USER_ID,
773         FINAL_SALE,
774         FINAL_QUANTITY
775     FROM
776         transaction_data_sale_not_null
777     GROUP BY
778         combined_pk
779 )
780 -- Main SELECT query to calculate the total sales value per user
781 SELECT
782     t.USER_ID,
783     COUNT(t.RECEIPT_ID) AS total_transactions -- Counting the number of transactions (receipts) per user
784 FROM
785     transaction_data_sale_no_duplicates AS t
786 JOIN
787     PRODUCTS_TAKEHOME AS p
788 ON
789     t.BARCODE = p.BARCODE
790 GROUP BY
```

! USER_ID	total_transactions
64063c8880552327897186a5	7
62e6f1ce48cc274645652f44	5

Output:

```
758 --Here I have defined power users as users who have 10 top users who have the most number of transactions with fetch.
759 -- Common Table Expression (CTE) to filter out transactions with a final sale value of zero
760 WITH transaction_data_sale_not_null AS (
761     SELECT
762         *
763     FROM
764         TRANSACTION_TAKEHOME
765     WHERE CAST(FINAL_SALE AS FLOAT) != 0.00),
766 -- CTE to concatenate receipt_id and barcode (primary key) and grouping them to remove duplicates
767 transaction_data_sale_no_duplicates AS (
768     SELECT
769         (RECEIPT_ID || COALESCE(BARCODE, '0')) AS combined_pk, -- Using a default value for NULL BARCODES
770         RECEIPT_ID,
771         BARCODE,
772         USER_ID,
773         FINAL_SALE,
774         FINAL_QUANTITY
```

! USER_ID	total_transactions
64063c8880552327897186a5	7
62e6f1ce48cc274645652f44	5
62b6189d37e6e08b0774ce73	5
62ad13c3cc43018bbbf84973	5
62925c1be942f00613f7365e	5
60a42b33f29c34057f5e46a9	5
609c28b122e98d5431152492	5
5e89fe8918bf1a13ef5d874c	5
66651af0e04f743a096e3bf9	4
664129ddb7b24d45d93b1860	4

Query:

```
WITH transaction_data_sale_not_null AS (

    SELECT

        *

    FROM

        TRANSACTION_TAKEHOME

    WHERE CAST(FINAL_SALE AS FLOAT) != 0.00),

-- CTE to concatenate receipt_id and barcode (primary key) and grouping them to remove duplicates
transaction_data_sale_no_duplicates AS (

    SELECT

        (RECEIPT_ID || COALESCE(BARCODE, '0')) AS combined_pk, -- Using a default value for NULL BARCODEs

        RECEIPT_ID,

        BARCODE,

        USER_ID,

        FINAL_SALE,

        FINAL_QUANTITY

    FROM

        transaction_data_sale_not_null

    GROUP BY

        combined_pk

)

-- Main SELECT query to calculate the total sales value per user

SELECT

    t.USER_ID,

    count(t.RECEIPT_ID) as total_transactions -- Counting the number of transactions (receipts) per user

FROM

    transaction_data_sale_no_duplicates AS t

JOIN

    PRODUCTS_TAKEHOME AS p

ON

    t.BARCODE = p.BARCODE

GROUP BY

    t.USER_ID

ORDER BY

    total_transactions DESC LIMIT 10; -- Only retrieving the top 10 users by total sales value
```

Q5 Which is the leading brand in the Dips & Salsa category?

Here I have only considered sales which are > 0 and are unique, removing any double counting and have joined with the products table to get the brand. Category 2 had dips and salsa so I have used category 2 in the where clause.

```
798 -- Initial CTE to filter out transactions with non-null final sales
799 WITH transaction_data_sale_not_null AS (
800     SELECT
801         *
802     FROM
803         TRANSACTION_TAKEHOME
804     WHERE
805         CAST(FINAL_SALE AS FLOAT) != 0.00),
806 -- Second CTE to ensure distinct transactions by concatenating receipt_id and barcode
807 transaction_data_sale_no_duplicates AS (
808     SELECT
809         (RECEIPT_ID || COALESCE(BARCODE, '0')) AS combined_pk, -- Concatenating with COALESCE to handle possible NULL values in BARCODE
810         RECEIPT_ID,
811         BARCODE,
812         USER_ID,
813         FINAL_SALE,
814         FINAL_QUANTITY
815     FROM
816         transaction_data_sale_not_null )
817 -- Main SELECT statement to analyze sales data, summing final sale values by brand
818 SELECT
819     p.BRAND,
820     SUM(t.FINAL_SALE) AS TOTAL_SALES_VALUE,
821     SUM(t.FINAL_QUANTITY) AS TOTAL_QTY
822 FROM
823     transaction_data_sale_no_duplicates AS t
824 JOIN
825     PRODUCTS_TAKEHOME AS p
826 ON
827     t.BARCODE = p.BARCODE
828 WHERE
829     p.CATEGORY_2 LIKE '%Dips & Salsa%'
830 GROUP BY
831     p.BRAND ORDER BY TOTAL_SALES_VALUE DESC LIMIT 1;
832
```

BRAND	TOTAL_SALES_VALUE	TOTAL_QTY
TOSTITOS	260.99	38

-- Initial CTE to filter out transactions with non-null final sales

WITH transaction_data_sale_not_null AS (

SELECT

*

FROM

TRANSACTION_TAKEHOME

WHERE

CAST(FINAL_SALE AS FLOAT) != 0.00),

-- Second CTE to ensure distinct transactions by concatenating receipt_id and barcode

transaction_data_sale_no_duplicates AS (

SELECT

(RECEIPT_ID || COALESCE(BARCODE, '0')) AS combined_pk, -- Concatenating with COALESCE to handle possible NULL values in BARCODE

RECEIPT_ID,

BARCODE,

USER_ID,

FINAL_SALE,

FINAL_QUANTITY

FROM

transaction_data_sale_not_null)

-- Main SELECT statement to analyze sales data, summing final sale values by brand

SELECT

p.BRAND,

SUM(t.FINAL_SALE) AS TOTAL_SALES_VALUE,

SUM(t.FINAL_QUANTITY) AS TOTAL_QTY

FROM

transaction_data_sale_no_duplicates AS t

JOIN

PRODUCTS_TAKEHOME AS p

ON

t.BARCODE = p.BARCODE

WHERE

p.CATEGORY_2 LIKE '%Dips & Salsa%'

GROUP BY

p.BRAND ORDER BY TOTAL_SALES_VALUE DESC LIMIT 1;

Q6. At what percent has Fetch grown year over year? – Answering based on YOY User Signup Growth

Assumption: I have taken the users table to answer this question since the transactions table only had data for only 2024 year. Since we do not have complete data for 2024 yet (only until August), the number for 2024 can be misleading

```
622 -- Extracting year and users registered for each year
623 WITH Yearly_USERS AS (
624     SELECT strftime('%Y', created_date) AS year, COUNT(*) AS users
625     FROM USER_TAKEHOME
626     GROUP BY year),
627 -- using lead function to get the users for next year
628 YOY_Growth_CALC AS (
629     SELECT
630         year,
631         users,
632         LAG(users) OVER (ORDER BY year) AS last_year_users
633     FROM Yearly_USERS)
634 --calculating users and growth
635 SELECT
636     year, users,
637     round(((users - last_year_users) * 100.0 / last_year_users),2) AS YOY_GROWTH
638 FROM YOY_Growth_CALC
639 WHERE last_year_users IS NOT NULL;
640
```

! year	users	YOY_GROWTH
2015	51	70
2016	70	37.25
2017	644	820
2018	2168	236.65
2019	7093	227.17
2020	16883	138.02
2021	19159	13.48
2022	26807	39.92
2023	15464	-42.31
2024	11631	-24.79

-- Extracting year and users registered for each year

WITH Yearly_USERS AS (

SELECT strftime('%Y', created_date) AS year, COUNT(*) AS users

FROM USER_TAKEHOME

GROUP BY year),

-- using lead function to get the users for next year

YOY_Growth_CALC AS (

SELECT

year,

users,

LAG(users) OVER (ORDER BY year) AS last_year_users

FROM Yearly_USERS)

--calculating users and growth

SELECT

year, users,

round(((users - last_year_users) * 100.0 / last_year_users),2) AS YOY_GROWTH

FROM YOY_Growth_CALC

WHERE last_year_users IS NOT NULL;