Fetch Data Analyst Take Home

Table of Contents

Section 1, Data Exploration.

This section explores the datasets and performs the minimum necessary data cleaning for the datasets. This section explores raw datasets, duplicates, null values, and data types.

Data cleaning may be necessary for the datasets. There are several guidelines for the cleaning process.

- Preserve as many data entries as possible. Entries will not be removed unless there are solid reasons.
- Create necessary value mapping to ensure value consistency.

The section will also explore the entity relationship.

Section 1-1. Data Exploration: Users table

Raw data: The users table contains 100,000 rows and 6 columns. The 6 columns include: id, created_date, birth_date, state, language, and gender. The id should be the primary key for this table, and it is also a foreign key in the transactions table.

Duplicates in raw data: 0 duplicates and 0 duplicated by id.

Null values by columns in raw data.

Column Name	Null Count	Null Percentage
id	0	0.00%
Created_date	0	0.00%
Birth_date	3,675	3.68%
State	4,812	4.81%
Language	30,508	30.51%
Gender	5,892	5.89%

Table 1-1-1. Null Values by Columns in User Raw Data

Data cleaning steps.

- Step 1.1.1 Value consistency in gender.
  Briefing: some values in gender are similar

Condition	Description
Situation	Value consistency needed
Task	Replace values with similar meaning by standardized values
Action	Create a replacement mapping to replace some values. Refer to Table 1.1.1.1 for details in genders.
Result	Reduced 11 unique values to 6 unique values.

Gender	Interpretation
"female"	Female.
"male"	Male.
NaN	Null value. Null values could be due to (1) intentionally left blank; (2) user has not filled any information; (3) system errors. Since the source of null values are unknown, null values are preserved.
"transgender"	Transgender.

"not_specified"	Users intentionally left blank, including scenarios when gender is not specified, user prefers not to say, and unknown.
"non_binary"	Non-binary.
"not_listed"	Users' gender not listed.

Table 1-1-2. Gender Mapping

- Step 1.1.2 Invalid birth date data.
  Briefing: some birth dates are invalid. Calculate the age at user creation, and the age ranges from -1 to 121.

Condition	Description
Situation	Invalid birth dates
Task	Handle invalid birth dates, set them to null values and exclude from potential analysis
Action	For birth dates later than creation date, set the birth date to null.
Result	Handled one birth date and set to null value.

  Reasoning for this step: For users data table, only the birth dates with obvious issues, i.e., negative age at creation, are set to null. No entries are dropped from the user table since the users without birth dates should also be valid (with created dates).
  Notes for further analysis: age should be calculated against transaction date values. Users of Fetch rewards should be able to use and understand an application (and potentially make purchases). Although the birth date data is preserved to the best extent, further analysis involving ages should exclude users with a certain age range.

Section 1-2. Data Exploration: Products table

Raw data: The products table contains 845,552 rows and 7 columns. The 7 columns include: category_1, category_2, category_3, category_4, manufacturer, brand, barcode. The barcode should be the primary key for this table, and it is also a foreign key in the transactions table.

Duplicates: 215 duplicates and 4209 duplicates by barcode.

Null values by columns.

Column Name	Null Count	Null Percentage
Category_1	111	0.01%
Category_2	1,424	0.17%
Category_3	60,566	7.16%
Category_4	778,093	92.02%
Manufacturer	226,474	26.78%
Brand	225,472	26.78%
Barcode	4,025	0.48%

Table 1-2-1. Null Values by Columns in Product Raw Table

Data cleaning steps.

- Step 1.2.1 Duplicated data entries (rows).

Briefing: duplicated rows with the same values in all columns.

Condition	Description
Situation	Duplicated data entries in products table.
Task	Drop duplicates (de-dup)
Action	Removed rows with the same value across all columns.
Result	Dropped 215 rows.

- Step 1.2.2 Null values in barcode.
  Briefing: Null values in barcode

Condition	Description
Situation	Null values in barcode.
Task	Null value removal
Action	Removed rows with null values in barcode column.
Result	Dropped 3968 rows

  Reasoning for this step: Based on the ER diagram, barcode serves as a foreign key in the transactions table, creating a link between the products and transactions table. Given that many columns in the products table may be null valued, missing barcode will make the product indifferentiable.

- Step 1.2.3 Single barcodes correspond to multiple products.
  Briefing: after step 1.1 and 1.2, there are multiple products with the same barcode. Those products may have different manufacturers and category.

Condition	Description
Situation	Multiple products have same barcode.
Task	Preserve one product per barcode.
Action	Preserved the first row for rows with the same barcodes.
Result	Dropped 27 rows

  Reasoning for this step: Based on common sense, one barcode should correspond to one product in inventory and sales in grocery stores. Having the same barcode for different products will cause confusion and management issues. Also, barcode is a foreign key with one-to-many relation to transaction table.

- Step 1.2.3 Same product details but different barcodes.
  Briefing: there are some rows with same product details but different barcodes.

Condition	Description
Situation	Same product details for different barcodes.
Task	Create a mapping of barcodes.
Action	Created a mapping of barcodes.
Result	Created a mapping for 825842 barcodes.

  Reasoning for this step: Although the barcodes are different, the product details are the same. This could be due to one of the following reasons:
  (1) The product details are missing, so similar products have same details based on the few information available.

(2) Two products are variants of the same product (varied by size, colors, etc.)
(3) One product has been entered the database multiple time (multiple entries).
(4) Product may be sold in different stores.
This mapping is used for connecting the dots. Further analysis in Section 1-4.

Section 1-3. Data Quality: Transactions table

Raw data: The transactions table contains 50,000 rows and 8 columns. The 8 columns include: receipt_id, purchase_date, scan_date, store_name, user_id, barcode, quantity (named "final_quantity" in the dataset), and sale (named "final_sale" in the dataset). The primary key(s) are not certain after the investigation.

Duplicates: 171 duplicates with the same values across all columns. More to be processed in the data cleaning.

Null values by columns.

Column Name	Null Count	Null Percentage
Receipt_id	0	0.00%
Purchase_date	0	0.00%
Scan_date	0	0.00%
Store_name	0	0.00%
User_id	0	0.00%
Barcode	5,762	11.52%
Final_quantity	0	0.00%
Final_sale	0	0.00%

Table 1-3-1. Null Values by Columns in Transaction Raw Table

Data cleaning steps.

- Step 1.3.1 Duplicated data entries (rows)
  Briefing: duplicated rows with the same values in all columns.

Condition	Description
Situation	Duplicated data entries in products table.
Task	Drop duplicates (de-dup)
Action	Removed rows with the same value across all columns.
Result	Dropped 171 rows.

- Step 1.3.2 Invalid value type in quantity.
  Briefing: some values in quantity column are not numeric and appears as "zero".

Condition	Description
Situation	Invalid value type (string) in quantity
Task	Replace "zero" with correct numerical expression.
Action	Replaced "zero" with 0.
Result	Replaced "zero" in 12491 rows.

- Step 1.3.3 Invalid value type in sales
  Briefing: invalid value type in sales. Some sales entries appear as white space (" ").

Condition	Description
Situation	Invalid value type (string) in sales
Task	Replace space with correct numerical expression.
Action	Replaced space with 0.
Result	Replaced space in 12486 rows. Removed 164 duplicates.

  Notes for further analysis: unlike step 1.3.2, "zero" may be directly translated to 0. This step replaced white space with 0. Before the replacement, 473 entries (rows) have sales as "0.00". In summary, sales should be a positive number, and 0 values (either originally valued 0 or valued white space) should be an exception. All further analysis should pay attention to these transaction entries.

- Step 1.3.4 Resolving data issues on "tcols"* (see Note 1-3-1).
  Briefing: The "tcols"* refers to the all the attribute variables except quantity and sale. For the one unique value in all tcols* attributes, there should be one quantity and sales**. But there are multiple entries per unique values across tcols with similar values or zero values.

Condition	Description
Situation	For the same tcols* values, there are multiple quantity and sales entry.
Task	For the same tcols* values, keep as few rows/entries as possible.
Action	Step A. Preserved all rows with one entry of quantity/sale Step B. Preserved all rows with null values in barcodes Step C. Dropped rows with zeros in both quantity and sale. Step D. Preserved all rows with one entry after steps B and C. Step E. Dropped rows with zero in either quantity or sale. Step F. Dropped rows with the same tcols, the same positive quantity/sales, but different in values of quantity/sales.
Result	Step A. Dropped 0 rows. Preserved 154 rows. Step B. Dropped 0 rows. Preserved 5532 rows. Step C. Dropped 57 rows. Step D. Dropped 0 rows. Preserved 56 rows. Step E. Dropped 21894 rows. Step F. Dropped 16 rows.

  [* Note 1-3-1. tcols = ('receipt_id', 'purchase_date', 'scan_date', 'store_name', 'user_id', 'barcode')]
  [** Note 1-3-2. One tcols combination should have exactly one entry (row). Each unique tcols combination refers to a receipt entry by a user scanned at a specific scan datetime (in milliseconds) on a specific product (by barcode) purchased at a store. One scanned receipt may have different products, but the barcode should be different. (Exception: Null values in barcodes.) One user may scan different receipts, but the scan datetime (in milliseconds) should be different.]

Reasoning for this step:
One tcols combination should have exactly one entry (row). Each unique tcols combination refers to a receipt entry by a user scanned at a specific scan datetime (in milliseconds) on a specific product (by barcode) purchased at a store. One scanned receipt may have different products, but the barcode should be different. One user may scan different receipts, but the scan datetime (in milliseconds) should be different.

Transaction Table Issues

- Major Issue.
  The transaction table does not seem to have a column (or list of columns) that serve as primary key(s). It is obvious that receipt ID cannot be served as a primary key. One receipt ID may be correlated with different products (barcodes).
  As discussed previously, tcols should serve as the primary keys. If that is the case, then all tcols should be required when entering the data.
- Data Cleaning Limitations.
  Keep in mind that the data cleaning step does not remove certain entries that cannot help in any analytics
  1. Transactions with quantity zero or sales zero but cannot find another valid entry (positive quantity and sales) with the same tcols information.
  2. Transactions that have barcodes missing.

  Are there entries invalid entries? There are a few possibilities that could have caused these issues.
  1. Possibility 1. System read error: The receipt cannot be read (due to blurry image or missing information). The system filled as much information as possible. When the same receipt was scanned again by the same user, the system generated a different receipt ID since the scanning happened at a different timestamp. Then the first record is a duplicate and should be dropped.
  2. Possibility 2. Same system read error. But this time the user did not re-scan and gave up. Then this record is a failed transaction that led to bad customer experience.
  3. Possibility 3. Data loss. The system may have read and processed the dataset, but some information may be lost before data reached to the transaction table. Then the entry is a valid entry, and immediate actions are needed.

  It is ideal to keep these entries before more information is obtained.
- Three tables:
  As an intermediate approach, transaction table should keep three copies and derivations from the raw data.
  1. Raw data: "transaction"
  2. Intermediate data: "tranasction2"
  3. View data (or production data): "transaction3".

  Why three table?
  As mentioned, the entries with zero quantity, zero sales, or no barcodes do not cannot be interpreted. Therefore, they should be ignored in most analytical

scenarios. The view data is the data that only have the interpretable and meaningful entries. Other teams may refer to the view data for analysis. However, as discussed previously, those entries may be valid data (or can be valid data after intervention). The intermediate data is for internal use. By comparing the intermediate and view data, one may quickly locate the entries that needed to be fixed. Additionally, those entries may be used in some occasions.

Section 1-4. Data Quality: Entity Relationship

Section 1-4-1. Understanding Relationship between Users and Transactions Table

Based on the ER diagram, the id in users table is a foreign key in the transactions table (one-to-many mapping).

- Users table: 100,000 unique ids
- Transactions table: 17,603 unique user ids.
- The user id is a 24-character long string variable consisting of numbers and letters.
- Less than 1% of the unique user ids in the transaction table can be found in the user table.

Since so few transactions (receipts scanned) can be allocated to the proper user, the analysis in section 2 does not provide many insights in understanding the user demographics.

Section 1-4-2. Understanding Relationship between Products and Transactions Table

Based on the ER diagram, the barcode in the products table is a foreign key in the transactions table (one-to-many mapping).

- Products table: 841,342 unique barcodes. It is an integer variable ranging from 185 to 62,911,081,607,944.
- Transactions table: 11,027 unique barcodes. It is an integer variable ranging from -1 to 9,347,108,002,132.
- Around 60% of the unique barcodes in the transactions table can be found in the user table.

Barcodes Confusion Issue: same product details correspond to different barcodes. A mapping was created in step 1.2.3. As stated in that analysis, the product details could be the same across different barcodes for any of the following reasons.

(1) The product details are missing, so similar products have same details based on the few information available.
(2) Two products are variants of the same product (varied by size, colors, etc.)
(3) One product has been entered the database multiple time (multiple entries).
(4) Product may be sold in different stores.

By linking the products and transactions table, we may confirm that based on data available in the transaction table, around less than 40% of the products with the same product details (in the products table) that could be found the store of purchase

in the transaction table follows (4). Although they have the same product details and different barcodes, they are sold in different stores.

Why do we care about the potential barcode confusion? Although it would be recommended that products table should retain as many entries as possible, barcodes should be populated with standardized valued during receipt scanning. If there are any product entries that follow (3), then a mapping table is necessary to standardize the data and provide a reference for analysis. Depending on the system, there may be two entries in the transactions table with different barcode for the same product from the same receipt. Potential duplicates in products table may cause duplicates in transactions table.

Relationship to Transactions.

The key issues with the products table and its relationship with the transactions table are listed below.

(1) low information granularity on products

Compared to the other two tables, the products table has null values across columns. In columns like brand, and manufacturer that are essential for analysis, over 25% of the entries have missing information.

(2) product table list incomplete.

Only 60% of the barcodes in the transactions table can be found in the products table.

(3) barcode information missing in transactions.

The connection between products table and transactions table also has great room for improvement. Compared to the user id that links users and transactions, there are too many null values in barcodes in the transactions table.

Section 1-5. Understanding Fields

Users table:

- Language: "en" or "en-149"

Products tables:

- Understanding the category columns (category column 1 to 4)

Section 2. Analytics and Queries.

Section 2-1. Top 5 Brands by Receipt among Adult Users.

[Assumptions, Query, Result, Interpretation]

Understanding the question:

Among receipts scanned by users with age 21 or above (age 21 included), count the number of receipts scanned for each brand. Select the 5 brands with the top receipt number counts.

Assumptions.

Assumption 1. Since brand name and age are required in this analysis, any transaction(s) that cannot find the product brand name or the user age information are discarded (in SQL syntax, inner joins are used).

Assumption 2. All birth dates in the Users table are accurate. Ages are not fixed but calculated from birth dates to scanned dates (as transactions table covers multiple months).

Assumption 3. Transactions with zero quantity or sales are discarded. These transactions do not contribute to an actual purchase/product sale, so they don't count when finding the top brands.

Assumption 4. If there is a tie in receipt number counts, then order by (1) total sales and (2) total quantity.

Assumption 5. If the brand name is not available, then it is not eligible to be a top brand.

Query.

```
SELECT P.BRAND
FROM (SELECT * FROM transaction2
WHERE BARCODE IS NOT NULL AND USER_ID IS NOT NULL
AND FINAL_QUANTITY>0 AND FINAL_SALE>0) T
JOIN (SELECT * FROM products1
WHERE BARCODE IS NOT NULL) P
ON T.BARCODE=P.BARCODE
JOIN (SELECT * FROM user1
WHERE BIRTH_DATE IS NOT NULL) u
ON T.USER_ID=u.ID
WHERE FLOOR(JULIANDAY(T.SCAN_DATE)-JULIANDAY(U.BIRTH_DATE)/365.25)>=21
GROUP BY 1
ORDER BY COUNT(DISTINCT T.RECEIPT_ID) DESC,
SUM(FINAL_SALE) DESC, SUM(FINAL_QUANTITY) DESC
LIMIT 5
```

Query 2.1.1 SQL Query for Top 5 Brands by Receipt for Users Aged 21 and Above.

[Note: julianday is the SQLite syntax used by pandasql. In My SQL syntax, date difference could be calculated using DATEDIFF(date1, date2)].

Result.

Index	Brand

0	NERDS CANDY
1	DOVE
2	TRIDENT
3	GREAT VALUE
4	MEIJER

Table 2-1-1. Top 5 Brands by Receipt for Users Aged 21 and Above

Interpretation of the result:

As specified in Section 1-4, few used ids from the transaction table can be found in the user table. Query 2.1.2 used dynamic age calculation as specified in assumption 2 and used total sales and quantity to avoid ties as specified in assumption 4. [

If any party is interested in the top brands among all users, table 2-1-2 has provided a summary.

Index	Brand
0	COCA-COLA
1	GREAT VALUE
2	PEPSI
3	EQUATE
4	LAY'S

Table 2-1-2. Top 5 Brands by Receipt for All Users

Section 2-2. Top 5 Brands by Sales among Tenured Users

[Assumptions, Query, Result, Interpretation].

Understanding the question:

Among receipts scanned by users with account creation date more than six months, add the sales for each brand. Select the 5 brands with the top sales volume.

Assumptions.

Assumption 1. The scanned receipts in the transactions table that cannot find the brand name in the products table and the created date in the user table are discarded.

Assumption 2. The use tenure (use with account for at least x period) is calculated from the receipt scan date and the user (account) created date.

Assumption 3. Transactions with zero quantity or sales are discarded. These transactions do not contribute to an actual purchase/product sale, so they don't count when finding the top brands.

Assumption 4. If there is a tie in total sales, then order by total quantity.

Assumption 5. If the brand name is not available, then it is not eligible to be a top brand.

Query.

```
SELECT P.BRAND
FROM (SELECT * FROM transaction2
WHERE BARCODE IS NOT NULL AND USER_ID IS NOT NULL
AND FINAL_QUANTITY>0 AND FINAL_SALE>0) T
JOIN (SELECT * FROM products1
WHERE BARCODE IS NOT NULL AND BRAND IS NOT NULL) P
ON T.BARCODE=P.BARCODE
JOIN (SELECT * FROM user1
WHERE BIRTH_DATE IS NOT NULL) u
ON T.USER_ID=u.ID
WHERE
(12*STRFTIME('%Y', T.SCAN_DATE)-STRFTIME('%Y', U.CREATED_DATE)) +
(STRFTIME('%m', T.SCAN_DATE)-STRFTIME('%m', U.CREATED_DATE)) >= 6
GROUP BY 1 ORDER BY SUM(FINAL_SALE) DESC, SUM(FINAL_QUANTITY) DESC LIMIT 5
```

Query 2.2.1 SQL Query for Top 5 Brands by Sales among Users that have had their accounts for at least six months.

[Note: strftime is the SQLite syntax used by pandasql. In MySQL syntax, monthly difference may be calculated by TIMESTAMPDIFF(unit, dt1, dt2). For this case, one may use TIMESTAMPDIFF(MONTH, U.CREATED_DATE, T.SCAN_DATE).]

Result.

Index	BRAND
0	CVS
1	DOVE
2	TRIDENT
3	COORS LIGHT
4	TRESEMMÉ

Table 2-2-1. Top 5 Brands by Sales among Users that have had their accounts for at least six months.

Interpretation.

As specified in Section 1-4 (like Section 2-1), few user ids in the transactions table can be found in the user table. Query 2.2.1 summarizes the top 5 brands by sales

Section 2-3. Fetch Power Users

[Pre-analysis, Assumptions, Analysis part 1, Query, Result & Implementation 1, Analysis Part 2, Result & Implementation 2]

Understanding the question:

Power users: the users who actively use the Fetch Rewards app based on the given data. The power users could be quantitatively measured by the following metrics.

Area of Focus	Quantitative Metrics	Interpretation
Frequency	Total Number of Receipts Scanned	High frequency per week
Consistency	Number of Receipts scanned over weeks	Consistent volume over weeks
Recency	Receipts scanned in the last X weeks	Actively using Fetch in the recent week
Quantity	Total quantity over time	Contribute high volume purchase
Sales	Total sales over time	Contribute high monetary value of purchase
Loyalty	Users created for half year(s)	Long-term retention. Users created over 1 year
Reliance	Number repeated purchases.	Repeated purchases of products.
Exploration	Number of different brands and categories.	Explored different brands and categories in Fetch.

Table 2-3-1. Power User Metrics.

Pre-analysis.

Transaction table contains transactions from week 24 to week 36 (13 weeks, approximately one quarter) in 2024. None of the users have receipts scanned at all 13 weeks. The top users have receipts scanned in 6 of the 13 weeks.

Assumptions.

Assumption 1. The transaction activities (receipt scanning) are the same as all the other activities on the app, hence receipt scanning activity is a good measure of user engagement level.

Assumption 2. All users recently joined were documented. Users with missing creation date are assumed satisfy the loyalty condition. Given the low number of users that have profile in user table and the high number of conditions filtering for power users, this assumption helps to retain a healthy number of users.

Assumption 3. Products mapping table provides an accurate mapping of brands and categories

Assumption 4. If a receipt scanned in the transaction table does not have a valid barcode, quantity, or sale, it is also a result of customer activity.

Analysis Part 1.

The analysis evaluates the metrics listed in table 2-3-1. Metrics listed in the table are not final. Each single criterion should not disqualify more than 20% of the users. Some metrics

Select the subset of users that satisfies the following conditions.

- Frequency: Users with 2 unique receipts from week 24 to 46 in 2024.
- Consistency: Users with one unique receipt for at least 2 weeks from week 24 to 46 in 2024.
- Recency: Users with at least 1 receipt scanned in weeks 35 and 36.
- Quantity Users with total quantity>=2 from week 24 to 46
- Sales: Users with total sales>=8 from week 24 to 46
- Loyalty metric not used. [See Assumption 2].
- Reliance metric not used.
- Exploration metrics not used.

Users that satisfied all conditions are Fetch's power users. The threshold of each condition is usually set at 75th percentile and does not reach beyond 80th percentile.

[Note 2-3-1. Based on the analysis, power users that have satisfied frequency, quantity, and sales conditions will also satisfy consistency and recency conditions.]

Query.

```
WITH CTE1 AS (
  SELECT USER_ID, COUNT(DISTINCT RECEIPT_ID) AS RECEIPT_UNIQUE
  FROM transaction2
  GROUP BY USER_ID
), CTE2 AS (
  SELECT USER_ID,
  SUM(FINAL_QUANTITY) AS QUANTITY_SUM, SUM(FINAL_SALE) AS SALE_SUM
  FROM transaction2
  WHERE FINAL_QUANTITY>0 AND FINAL_SALE>0 AND BARCODE IS NOT NULL
  GROUP BY USER_ID
)
SELECT COUNT(C1.USER_ID) AS POWER_USERS
FROM (SELECT * FROM CTE1 WHERE RECEIPT_UNIQUE>=2) C1
JOIN (SELECT * FROM CTE2 WHERE QUANTITY_SUM>=2 AND SALE_SUM>=8) C2
ON C1.USER_ID=C2.USER_ID
```

Query 2.3.1. SQL Query for Power Users

[Note 2-3-2. As stated in Note 2-3-1, the power users are selected by frequency, quantity, and sales conditions. Adding consistency and recency conditions will not change the power users based on research.]

Result and Implementation 1

Selected 2,180 users as power users based on the criteria listed.

Notice that the frequency condition does not require the transaction (receipt scanned) to have a valid barcode, quantity and sale. Both activities and purchases can be counted towards frequency. As stated in assumption 4, an invalid receipt scanned (invalid transaction) may be due to customer taking blurry pictures of the receipts. However, it also reflects the activity of customers.

Analysis Part 2.

This analysis leverages unsupervised machine learning for customer segmentation. The follow metrics are used.

- Frequency: Number of unique receipts.
- Consistency: Number of weeks with at least one valid receipt scanned.
- Recency: Number of weeks with at least one valid receipt scanned in weeks 33, 34, 35, and 36.
- Quantity: Total valid quantity
- Sales: Total valid sales
- Reliance: Number of brands purchased more than once.
- Exploration: Number of brands purchased

Some metrics, such as use creation date and user demographics, are not used. There are too many missing data and will not be helpful in the segmentation process.

Result and Implementation 2.

K-Means Clustering.

Using Elbow Method and Silhouette Score, k=6 is optimal in K-Means clustering.

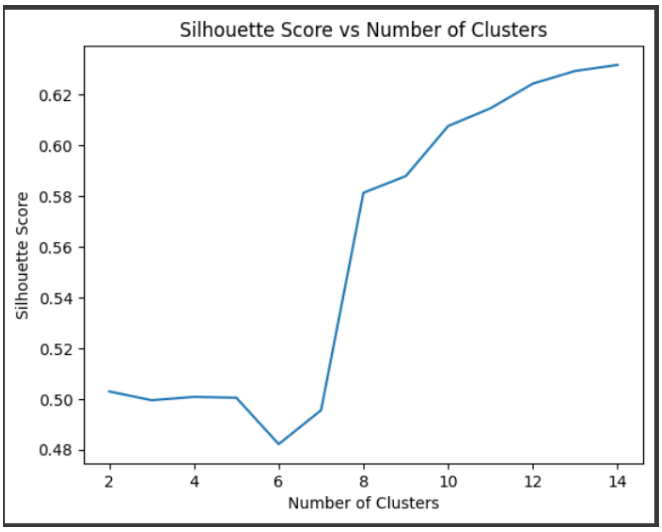

Figure 2-3-1. Silhouette Score & Number of Clusters

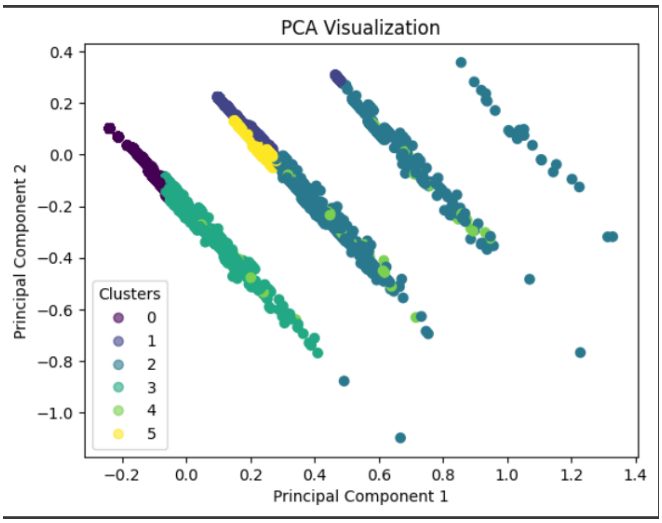

Figure 2-3-2. PCA Visualization of Cluster

Figure 2-3-2 provides a visualization of the cluster groups after dimensionality reduction (PCA). Table 2-3-2 provides metrics statistics for different user groups.

Metrics	Group 0	Group 1	Group 2	Group 3	Group 4	Group 5
Frequency	Low	Low	High	Med	High	Low
Consistency	Low	Med	High	High	High	Med
Recency	Low	Med	High	Low	Med	Med
Quantity	Low	Med	High	High	High	Med
Sales	Low	Low	High	Med	High	Med
Reliance	Low	Low	Low	Low	Med	Low
Exploration	Low	Low	High	Med	Med	Med
Count	9,772	2,080	1,380	1,660	119	2,683

Table 2-3-2. User Group Statistics

User Group Summary

- Group 0: Inactive users with very few activities.
- Group 1: Light users who have started using Fetch recently.
- Group 2: Power users.
- Group 3: Loyal users with a few recent activities. Prefer average products.
- Group 4: Loyal users with some recent activities. Prefer high-value products and have brand loyalty.
- Group 5: Light users with consistent average purchasing behavior.

Based on the summary above, the K-Means clustering algorithm has selected 1,380 users as power users. Compared to analysis 1 of this section, the number of power users are fewer. But the users are selected by a more systematic and more reasonable manner.

Potential for other clustering algorithms. Figure 2-3-2, the customer groups are clearly separated.

Section 2-4. Dips and Salsa Leading Brand

[Assumptions, Query, Result].

Understanding the question:

Among all brands that have products in the dips and salsa category, find one brand that is leading the dips and salsa market. The leading brand will be a frontrunner in a variety of quantitative metrics.

Area of Focus	Quantitative Metrics	Interpretation
Quantity Dominance	Overall quantity sold (or share of quantity).	Share of total quantity purchase.
Sales Dominance	Overall sales volume (or sales share).	Share of total sales generated.
Brand Customer Loyalty	Number of users with repeated purchase	Repeated purchase
Consistent Transactions	Median and Interquartile range of quantity per week.	Transaction volume consistency over time.
Product availability	Number of stores where the receipt took place	Product available in different stores

Table 2.4.1 Brand Metrics

Assumptions.

Assumption 1. The "leading" brand is the brand that has strong business values. The business value is usually associated with the purchasing volume and metrics that can be converted to

Assumption 2. The transactions table contains all the receipt scanned between 2024-06-12 and 2024-09-08.

Assumption 3. The transactions table is a good representation of the overall market (including the dips and sales market). The metrics derived from the transactions table (such as market share) correctly reflect the position of the brands in the real market.

Assumption 4. The transactions table and the receipt scanned are good representation of the overall market condition. The dips and salsa product industry does not have seasonal changes or fluctuations.

Assumption 5. The transactions table entries with zero quantity, zero sales, or null barcodes are not considered in any metrics calculation.

Query.

All brands are ranked by the brand metrics listed in table 2.4.1. The overall ranking of a brand is calculated by the average of the brand rankings in all the metrics. The leading brand is the brand that has the highest overall ranking.

```
/* select the proper subset to join */
WITH CTE1 AS (
  SELECT P.*, T.*
  FROM (SELECT * FROM products1
  WHERE CATEGORY_2 IS NOT NULL AND CATEGORY_2 LIKE '%dips%') P
  JOIN (SELECT * FROM transaction2
  WHERE BARCODE IS NOT NULL AND FINAL_QUANTITY>0
  AND FINAL_SALE>0) T
  ON P.BARCODE=T.BARCODE
),
/* calculate the total quantity and sales for each brand */
CTE2 AS (
  SELECT BRAND, SUM(FINAL_QUANTITY) AS TOTAL_QUANTITY,
  SUM(FINAL_SALE) AS TOTAL_SALE, COUNT(DISTINCT USER_ID) AS UNIQUE_USER,
  COUNT(DISTINCT STORE_NAME) AS UNIQUE_STORE
  FROM CTE1
  GROUP BY BRAND
),
/* calculate the weekly quantity and assign rn to identify
median and quantile range */
CTE3 AS (
  SELECT C11.BRAND, C11.WEEK_NUM,
    COALESCE(C10.TOTAL_QUANTITY, 0) AS TOTAL_QUANTITY,
    ROW_NUMBER() OVER(PARTITION BY C11.BRAND
    ORDER BY C10.TOTAL_QUANTITY) AS RN
  FROM (
    SELECT B1.BRAND, W1.WEEK_NUM
    FROM (SELECT DISTINCT BRAND FROM CTE1) AS B1 CROSS JOIN
    (SELECT DISTINCT STRFTIME('%W', SCAN_DATE) AS WEEK_NUM
    FROM transaction2) W1 ) C11
  LEFT JOIN(
    SELECT BRAND, STRFTIME('%W', SCAN_DATE) AS WEEK_NUM,
    SUM(FINAL_QUANTITY) AS TOTAL_QUANTITY
    FROM CTE1 GROUP BY BRAND, WEEK_NUM) C10
  ON C10.BRAND=C11.BRAND AND C10.WEEK_NUM=C11.WEEK_NUM
  WHERE C11.BRAND IS NOT NULL
  ORDER BY C11.BRAND, C11.WEEK_NUM
), CTE_MEDIAN AS (
  SELECT BRAND, TOTAL_QUANTITY AS Q_MEDIAN
  FROM CTE3 WHERE RN=7
), CTE_Q25 AS (
  SELECT BRAND, AVG(TOTAL_QUANTITY) AS Q_PERC25
  FROM CTE3 WHERE RN=3 OR RN=4 GROUP BY BRAND
), CTE_Q75 AS (
  SELECT BRAND, AVG(TOTAL_QUANTITY) AS Q_PERC75
  FROM CTE3 WHERE RN=10 OR RN=11 GROUP BY BRAND
), CTE_USERPURCHASE AS (
  SELECT BRAND, USER_ID, COUNT(*) AS USER_PURCHASE
  FROM CTE1 GROUP BY BRAND, USER_ID
)
SELECT BRAND,
```

```
ROUND((TOTAL_SALE_RANK + TOTAL_QUANTITY_RANK + UNIQUE_USER_RANK +
UNIQUE_STORE_RANK + Q_MEDIAN_RANK + Q_RANGE_RANK +
REPEATED_USERS_RANK)/7,0) AS AVG_RANK
FROM (
SELECT C1.BRAND, C1.TOTAL_SALE, C1.TOTAL_QUANTITY, C1.UNIQUE_USER,
C1.UNIQUE_STORE, C1.Q_MEDIAN, C1.Q_PERC75-C1.Q_PERC25 AS Q_RANGE,
C1.REPEATED_USERS,
/* calculate the rank for each metric */
RANK() OVER (ORDER BY TOTAL_SALE DESC) AS TOTAL_SALE_RANK,
RANK() OVER (ORDER BY TOTAL_QUANTITY DESC) AS TOTAL_QUANTITY_RANK,
RANK() OVER (ORDER BY UNIQUE_USER DESC) AS UNIQUE_USER_RANK,
RANK() OVER (ORDER BY UNIQUE_STORE DESC) AS UNIQUE_STORE_RANK,
RANK() OVER (ORDER BY Q_MEDIAN DESC) AS Q_MEDIAN_RANK,
RANK() OVER (ORDER BY Q_PERC75-C1.Q_PERC25 DESC) AS Q_RANGE_RANK,
RANK() OVER (ORDER BY REPEATED_USERS DESC) AS REPEATED_USERS_RANK
FROM (
SELECT CTE2.*, CTE_MEDIAN.Q_MEDIAN,
CTE_Q25.Q_PERC25, CTE_Q75.Q_PERC75,
COALESCE(C4.REPEATED_USERS, 0) AS REPEATED_USERS
FROM CTE2 JOIN CTE_MEDIAN ON CTE2.BRAND=CTE_MEDIAN.BRAND
JOIN CTE_Q25 ON CTE2.BRAND=CTE_Q25.BRAND
JOIN CTE_Q75 ON CTE2.BRAND=CTE_Q75.BRAND
LEFT JOIN (SELECT BRAND, COUNT(USER_ID) AS REPEATED_USERS
FROM CTE_USERPURCHASE WHERE USER_PURCHASE>1 GROUP BY BRAND) C4
ON CTE2.BRAND=C4.BRAND ) C1
)
ORDER BY AVG_RANK ASC
LIMIT 5
```

Query 2.4.1 SQL Query to Identify Dips and Salsa Leading Brand.

[Note 2-4-1. SQLite does not support direct calculation of median and quantile using functions like median() or percentile(). The calculation is performed using row number. There are 13 weeks in total for each brand, and the median/quantile values can be directly derived.]

Result

Index	BRAND	AVG_RANK
0	TOSTITOS	1

Table 2-4-1. Dips and Salsa Leading Brand.

The leading dips and salsa brand is Tostitos. Top 5 brands are Tostitos, Pace, Fritos, Dean's Dairy Dip, and Heluva Good!

Section 2-5. Fetch YOY Growth.

[Assumptions, Analysis, Query, Result]

Understanding the question.

Note that only the user data can provide YOY information. The transactions table only have 2024 data available. The year-over-year growth discussed in this question will focus on the growth of users. The YOY growth may be captured by a list of metrics listed below.

Area of Focus	Quantitative Metrics	Interpretation
User Base Cumulative	User base volume	The cumulative number of users over years.
User Base Growth Quarterly	YOY same quarter growth rate = 100*[Curr Q User-Prev Q user]/[Prev Q user]	The growth of current quarter compared to the same quarter last year, avoid seasonality.

Table 2-5-1. YOY Growth Metrics

In addition to the metrics discussed in table 2-5-1, if the transaction data over years is also available, additional metrics can be investigated. (1) The growth of revenue per user. (2) The growth of partnered brands and their customer loyalty. (3) The growth of power users over years. (4) The number of active users across years (user retention).

Assumptions

Assumption 1. For the sake of this analysis, assume that the user table contains all the users registered from 2014Q2 to 2024Q3.

Analysis.

Based on the user table, Fetch had the highest user growth in year 2022. Since then, the user growth has slowed down (see figure 2-5-1).

The user growth in each quarter and each month are consistent with the user growth trend found in figure 2-5-1. As observed in figure 2-5-2, the user growth in Q1 is usually smoother than the other 3 quarters. Generally, user growths (absolute numbers) in Q3 and Q4 are slightly higher than those in Q1 and Q2.

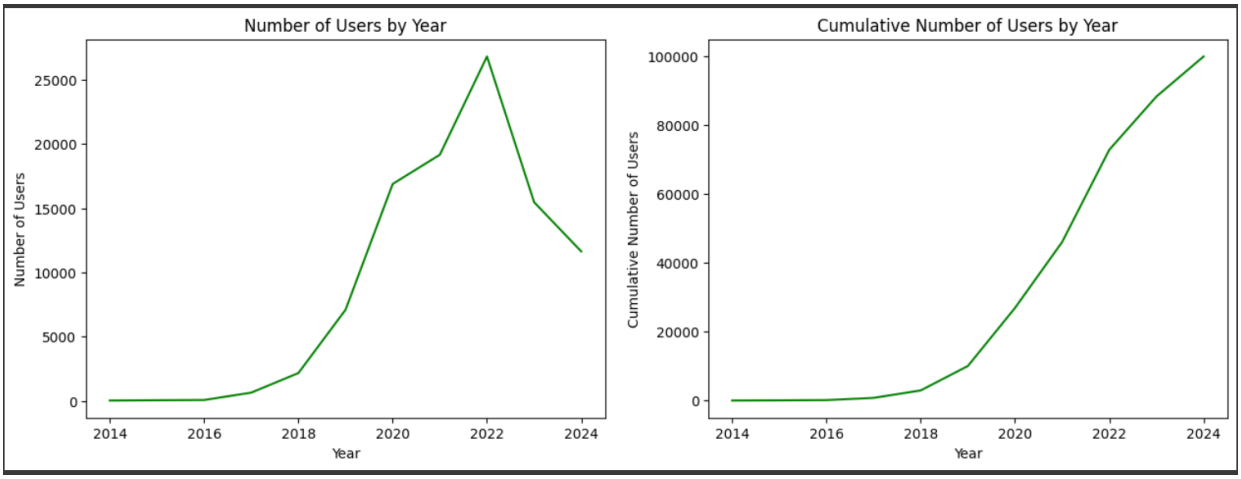

Figure 2-5-1. Cumulative and User Growth by Year.

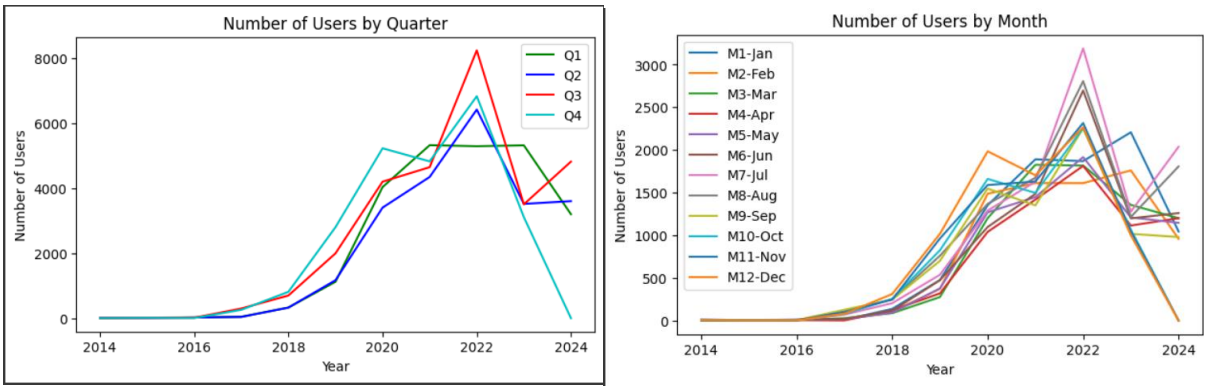

Figure 2-5-2. User Growth by Quarter and by Month.

Query.

```
WITH CTE_YEAR1 AS (
  /* create annual user count and cumulative user count */
  SELECT STRFTIME('%Y', CREATED_DATE) AS YEAR,
  COUNT(ID) AS USER_CNT,
  SUM(COUNT(ID)) OVER (ORDER BY STRFTIME('%Y', CREATED_DATE)) AS CUM_USER_CNT
  FROM user1 GROUP BY YEAR
), CTE_QUARTER1 AS (
  /* create summary on year and quarter level */
  SELECT STRFTIME('%Y', CREATED_DATE) AS YEAR,
  CASE WHEN STRFTIME('%m', CREATED_DATE) BETWEEN '01' AND '03' THEN 'Q1'
  WHEN STRFTIME('%m', CREATED_DATE) BETWEEN '04' AND '06' THEN 'Q2'
  WHEN STRFTIME('%m', CREATED_DATE) BETWEEN '07' AND '09' THEN 'Q3'
  WHEN STRFTIME('%m', CREATED_DATE) BETWEEN '10' AND '12' THEN 'Q4'
  END AS QUARTER,
  COUNT(ID) AS USER_CNT
  FROM user1 GROUP BY YEAR, QUARTER
)
/* calculate the yoy percentage growth for each quarter */
SELECT Y.YEAR, USER_CNT, CUM_USER_CNT, Q1, Q2, Q3, Q4,
ROUND(100*(Q1-LAG(Q1) OVER(ORDER BY Q3.YEAR))/(
  LAG(Q1) OVER(ORDER BY Q3.YEAR)),2) AS YOY_Q1_INC_PCT,
ROUND(100*(Q2-LAG(Q2) OVER(ORDER BY Q3.YEAR))/(
  LAG(Q2) OVER(ORDER BY Q3.YEAR)),2) AS YOY_Q2_INC_PCT,
ROUND(100*(Q3-LAG(Q3) OVER(ORDER BY Q3.YEAR))/(
  LAG(Q3) OVER(ORDER BY Q3.YEAR)),2) AS YOY_Q3_INC_PCT,
ROUND(100*(Q4-LAG(Q4) OVER(ORDER BY Q3.YEAR))/(
  LAG(Q4) OVER(ORDER BY Q3.YEAR)),2) AS YOY_Q4_INC_PCT
FROM CTE_YEAR1 AS Y
JOIN (
  /* put each year's quarter data into one row */
  SELECT YEAR,
  SUM(CASE WHEN QUARTER='Q1' THEN CUM_USER_CNT ELSE 0 END) AS Q1,
  SUM(CASE WHEN QUARTER='Q2' THEN CUM_USER_CNT ELSE 0 END) AS Q2,
  SUM(CASE WHEN QUARTER='Q3' THEN CUM_USER_CNT ELSE 0 END) AS Q3,
```

```
  SUM(CASE WHEN QUARTER='Q4' THEN CUM_USER_CNT ELSE 0 END) AS Q4
  FROM (SELECT *,
  SUM(USER_CNT) OVER(ORDER BY YEAR, QUARTER) AS CUM_USER_CNT
  FROM CTE_QUARTER1) Q2 GROUP BY YEAR
) Q3
ON Y.YEAR=Q3.YEAR
```

Query 2.5.1 SQL Query for Fetch User YOY Growth.

[Note 2-5-1. SQLite syntax (used by Pandasql) does not support direct extract of quarter from datetime variables. In MySQL syntax, one may use QUARTER() to directly extract the quarter information.]

Result.

	YEAR	USER_CNT	CUM_USER_CNT	Q1	Q2	Q3	Q4	YOY_Q1_INC_PCT	YOY_Q2_INC_PCT	YOY_Q3_INC_PCT	YOY_Q4_INC_PCT
0	2014	30	30	-	15	24	30				
1	2015	51	81	48	58	69	81		286	187	170
2	2016	70	151	106	130	145	151	120	124	110	86
3	2017	644	795	195	236	536	795	83	81	269	426
4	2018	2,168	2,963	1,124	1,451	2,150	2,963	476	514	301	272
5	2019	7,093	10,056	4,086	5,256	7,253	10,056	263	262	237	239
6	2020	16,883	26,939	14,095	17,501	21,706	26,939	244	232	199	167
7	2021	19,159	46,098	32,267	36,617	41,268	46,098	128	109	90	71
8	2022	26,807	72,905	51,395	57,820	66,069	72,905	59	57	60	58
9	2023	15,464	88,369	78,228	81,749	85,257	88,369	52	41	29	21
10	2024	11,631	100,000	91,571	95,177	100,000	-	17	16	17	-100

Table 2-5-1. Fetch User YOY Growth.

Section 3, Stakeholder communication.

Section 3-1. Data Quality Issue Summary (technical).

Quality Issue Users Table.

- User table birth date. As discussed in [Section 1-1, step 1.1.2], the age at creation calculated from the birth dates and created date can range from negative value to 121.
- User table does not have the entire user population. Many user ids in the transaction table cannot be found in the user table [discussed in Section 1-4].

Quality Issue Products Table.

- Not all entries are populated with barcodes.
- Too many entries have null values across multiple columns.
- Duplicates by barcodes. Barcodes should be unique.
- Many barcodes have the same product details. Created a mapping table for intermediate references. Need to clarify and standardize the barcodes and avoid confusion.
- Products table does not have the full list of products. Many barcodes in the transactions table cannot be found in the products table.

Quality Issue Transactions Table.

- No primary key value(s). One receipt ID may correspond to different barcodes, resulting in two valid rows.
- Too many potential duplicates identified by the combination of [receipt_id, purchase_date, scan_date, store_name, user_id, and barcode] (*tcols). Combination of columns cannot be key values either.
- Too many entries (rows) have missing barcodes (foreign key), thus the products cannot be determined for those transactions.
- Value issues with quantity and sales. Quantity and sales should always be numeric and cannot be zeros.
- There are entries with the same tcols, positive quantity and positive sales, but the quantity and sales are different.

Section 3-2. Data Trends.

The YOY user growth is an interesting trend. Table 2-5-1 provides the annual user incremental values and the total number of users at each year and quarter respectively. The table also provides the YOY quarterly percentage increase. Figure 2-5-2 provides quarterly user increment values. All tables and graphs may be shared upon request.

Section 3-3. Request for Action.

Immediate Actions

- Fix the missing users and products in the users table and the products table.
- Identify the key value(s) in the transaction table. If tcols variables are key values, then all attributes/columns in tcols are required attributes and cannot be missing during data entry.
- Mark barcode as required in transactions table.
- Require the quantity and sales to be numeric in transactions table.
- Check for duplicates and stop duplicates at data entry.
- Enforce that each table should have a primary key, and the primary key must be unique for each row.

Phase 2 Actions.

- Inquire accurate customer information (such as birth dates). Ask for less precise birth year ranges to reduce safety and data privacy concerns.
- Inquire the receipt scanning process and system. Understand the zeros in sales/quantity and improvements. Understand receipt ID generation. Understand the procedures for receipt reading failures.
- Understand the barcodes and product details.
- Improve the products table to reduce the null values in the four categories, brand, and manufacturer columns.

Section 3-4. Email Message.

Email Title: Update on Investigation

Hello!

I am Hanwei, and I am investigating on the internal datasets on scanned receipts (transactions), users, and products. I would like to provide an update on my investigation.

There are several major data issues regarding the datasets.

- Missing critical information.
  For example, some scanned receipts do not provide information on the product purchased (missing barcode information).
- Inaccurate and invalid information in datasets.
  For example, some scanned receipts have invalid values (zero, string, etc.) in quantity and/or sales.
- Many users/products not documented.
  Many scanned receipts correspond to users and products not found in the dataset.
- Duplicate issues.
  Some data entries look the same. Clarifications needed to determine if they are duplicates.

I have some outstanding questions on the datasets.

- How do we gather user information such as birth dates, state, and gender?
- How do we gather product information?
- How do we identify product barcodes?
- How information from receipts do we process from scanning?

I would love to share an interesting trend observed in the dataset.

- Fetch rewards usually observe higher number of user creation (user acquisition) in the second half of the year (Q3 and Q4)

Request for Action.

- During data entry, mark some information (data columns) as required.
- During data entry, restrict the format and range of some information.
- Automate a data cleaning processing for all datasets.
- Deal with missing data immediately.
- Establish standard procedure and documentation for data entry and communicate with the teams responsible.

Help needed.

- Need help from data engineers and data owners.
- Access to important documentations.

- Start a cross-functional project with clear goals and milestones. Notify all associated parties and allocate resources and work hours accordingly.

It's been a pleasure to work on the datasets. I see great potential in a project to improve the existing data issues. Thank you for taking your time to review the update, and I look forward to hearing from you.

Best regards,

Hanwei