Fetch Data Analyst Assessment

- Arjun Kumaran

Part 1: Explore the data

Review the unstructured csv files and answer the following questions with code that supports your conclusions:

- Are there any data quality issues present?
- Are there any fields that are challenging to understand?

We recommend using SQL or python and data visualization to examine the data.

Data Sources:

1. PRODUCTS_TAKEHOME.csv – Products Table
2. USER_TAKEHOME.csv – Users Table
3. TRANSACTION_TAKEHOME.csv – Transactions Table

Language used to analyse the data – **Python**

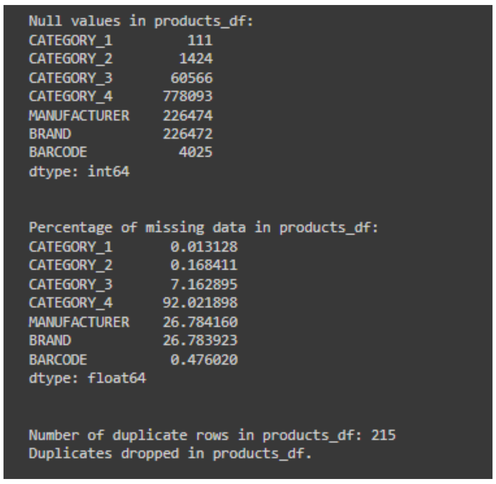
Firstly, a **basic data quality check** function was written commonly for all tables that does the following:

1. Counts null values and calculates percentage of null values in a column if null values are present in it.
2. Counts and drops duplicates if present.
3. Standardizes the text formatting by removing whitespaces and converting to lower-case.
4. Checks data-types.

<u>Products Table:</u>

Data Redundancy Issue: There were 215 duplicates in the dataset and were removed.

Data Completeness Issue:



```
Null values in products_df:
CATEGORY_1         111
CATEGORY_2        1424
CATEGORY_3       60566
CATEGORY_4      778093
MANUFACTURER    226474
BRAND           226472
BARCODE           4025
dtype: int64


Percentage of missing data in products_df:
CATEGORY_1       0.013128
CATEGORY_2       0.168411
CATEGORY_3       7.162895
CATEGORY_4      92.021898
MANUFACTURER    26.784160
BRAND           26.783923
BARCODE          0.476020
dtype: float64


Number of duplicate rows in products_df: 215
Duplicates dropped in products_df.
```

There is a lot of missing data in the dataset. For eg: 92% of data in the Category_4 column is missing. After reviewing the data, I found that category_1 is the main category. category_2 is the subcategory of category 1. category_3 is the subcategory of Category 2 and category 4 is the subcategory of category 3.

Missing data can break the links between categories and subcategories, making it difficult to understand the complete structure and relationships within the data.

Apart from having 226474 null values in the Manufacturer column, it has a placeholder string "placeholder manufacturer" 86895 times.

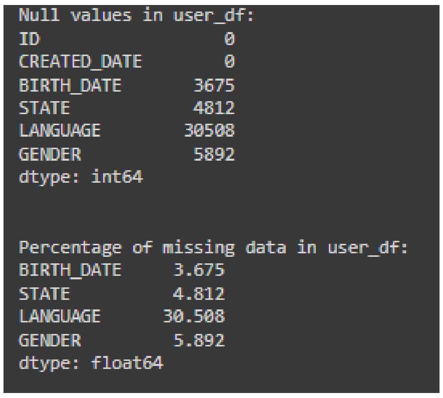
How should we handle these placeholder strings? Should they be considered as null values?

226472 values in the Brand column are also null values.

Data Type Mismatches: Numeric column "Barcode" is in object datatype. It had to be changed to INT.

2) Users Table:

Data Completeness Issue:

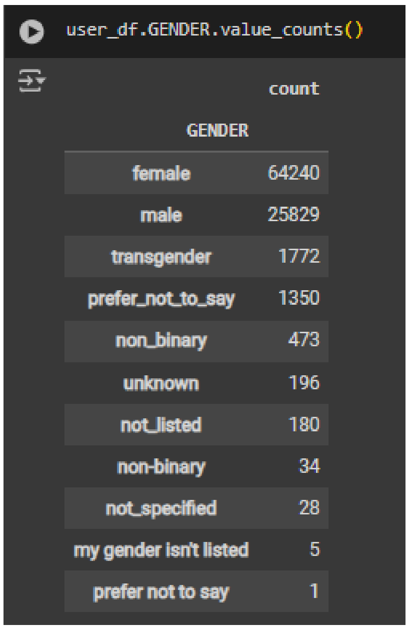


```
Null values in user_df:
ID                  0
CREATED_DATE        0
BIRTH_DATE       3675
STATE            4812
LANGUAGE        30508
GENDER           5892
dtype: int64


Percentage of missing data in user_df:
BIRTH_DATE     3.675
STATE          4.812
LANGUAGE      30.508
GENDER         5.892
dtype: float64
```

There is a lot of null values in this dataset. For eg: 30% of rows in Language are missing. This would impact the reliablity of any user demographic analysis (Insights into their behavior, preferences, and trends based on language) done using this data.

Data Type Mismatches: Datetime columns "CREATED_DATE" and "BIRTH_DATE" are in object datatype. It had to be changed to datetime.

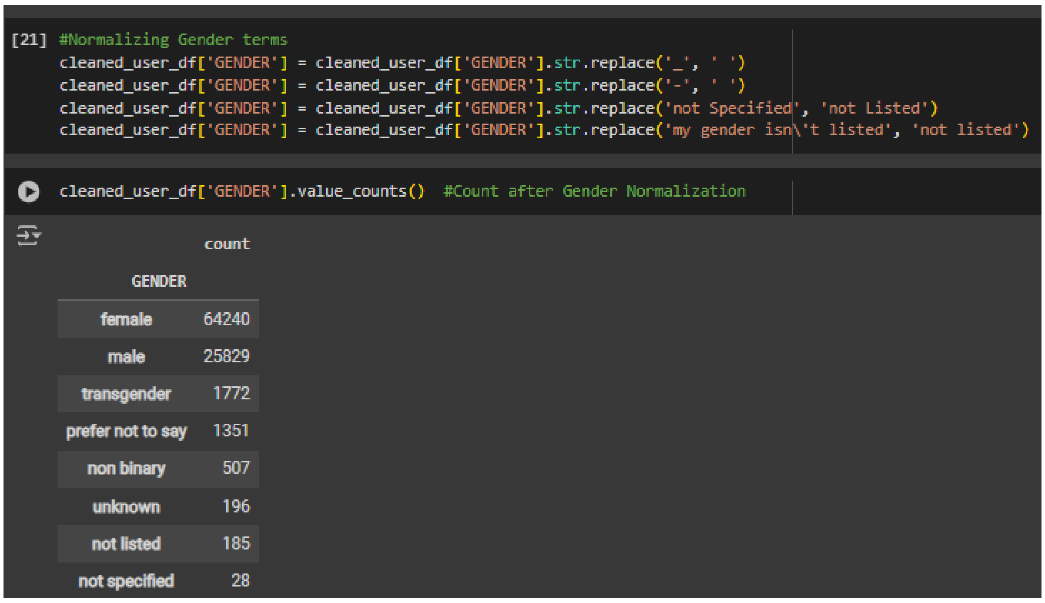
Inconsistent Data:



```
user_df.GENDER.value_counts()
```

GENDER	count
female	64240
male	25829
transgender	1772
prefer_not_to_say	1350
non_binary	473
unknown	196
not_listed	180
non-binary	34
not_specified	28
my gender isn't listed	5
prefer not to say	1

There is inconsistency in the Gender terms and it had to be normalized.

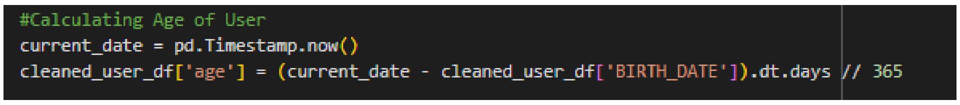


```
[21] #Normalizing Gender terms
     cleaned_user_df['GENDER'] = cleaned_user_df['GENDER'].str.replace('_', ' ')
     cleaned_user_df['GENDER'] = cleaned_user_df['GENDER'].str.replace('-', ' ')
     cleaned_user_df['GENDER'] = cleaned_user_df['GENDER'].str.replace('not Specified', 'not Listed')
     cleaned_user_df['GENDER'] = cleaned_user_df['GENDER'].str.replace('my gender isn\'t listed', 'not listed')
```

```
cleaned_user_df['GENDER'].value_counts()  #Count after Gender Normalization
```

GENDER	count
female	64240
male	25829
transgender	1772
prefer not to say	1351
non binary	507
unknown	196
not listed	185
not specified	28

Invalid Data (Range Violation): To check invalid entries in the birth_date column, age column was calculated using the following formula:



```
#Calculating Age of User
current_date = pd.Timestamp.now()
cleaned_user_df['age'] = (current_date - cleaned_user_df['BIRTH_DATE']).dt.days // 365
```

I assumed that if the age of the user was below 5 or above 100, it could be considered as invalid entries.

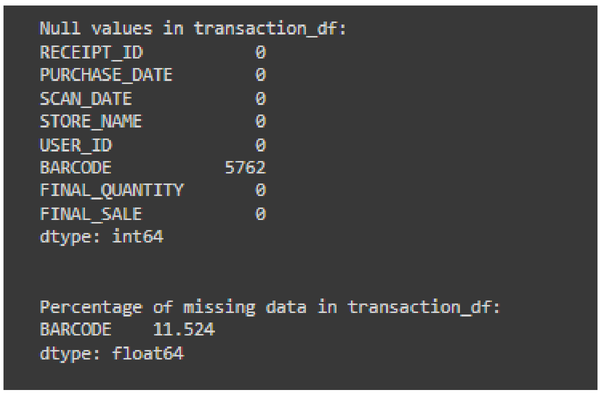
The Dataset had 75 invalid ages and they were removed from the dataset.

Following this check, the Age column was dropped from the dataset.

3) Transaction Table:

Data Redundancy Issue: There are 171 duplicate rows in this dataset and they have been removed.

Data Completeness Issue:



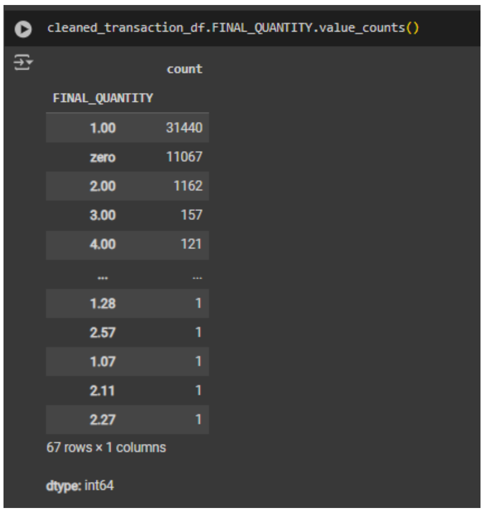
```
Null values in transaction_df:
RECEIPT_ID           0
PURCHASE_DATE        0
SCAN_DATE            0
STORE_NAME           0
USER_ID              0
BARCODE           5762
FINAL_QUANTITY       0
FINAL_SALE           0
dtype: int64


Percentage of missing data in transaction_df:
BARCODE    11.524
dtype: float64
```

There are null values only in the barcode column and it has been removed. Compared to the User and Products table, this table has very less null values. Upon Further Analysis, I found null values in the final quantity column.

Data Type Mismatches:

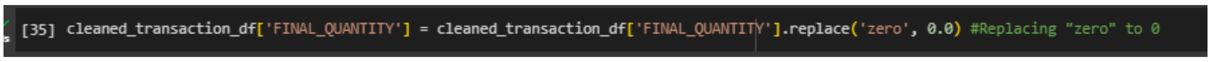
1. Barcode column had to be changed to INT Datatype.
2. Datetime columns "PURCHASE_DATE" and "SCAN_DATE" are in object datatype. It had to be changed to datetime.
3. Numeric columns "FINAL_QUANTITY" and "FINAL_SALE" are in object datatype. It had to be changed to Numeric.

Inconsistent Data:

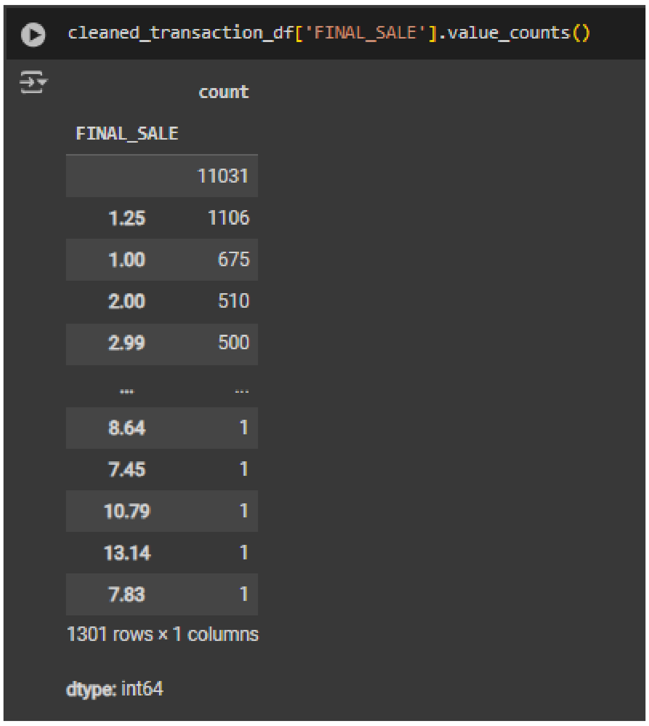


```
cleaned_transaction_df.FINAL_QUANTITY.value_counts()

                   count
FINAL_QUANTITY
      1.00         31440
      zero         11067
      2.00          1162
      3.00           157
      4.00           121
      ...            ...
      1.28             1
      2.57             1
      1.07             1
      2.11             1
      2.27             1
67 rows × 1 columns

dtype: int64
```

In the FINAL_QUANTITY column, It can be seen that 0 is represented as a string and it had to be changed as this is a numeric column.



```
[35] cleaned_transaction_df['FINAL_QUANTITY'] = cleaned_transaction_df['FINAL_QUANTITY'].replace('zero', 0.0) #Replacing "zero" to 0
```



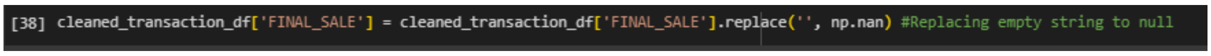
```
cleaned_transaction_df['FINAL_SALE'].value_counts()
```

FINAL_SALE	count
	11031
1.25	1106
1.00	675
2.00	510
2.99	500
...	...
8.64	1
7.45	1
10.79	1
13.14	1
7.83	1

1301 rows × 1 columns

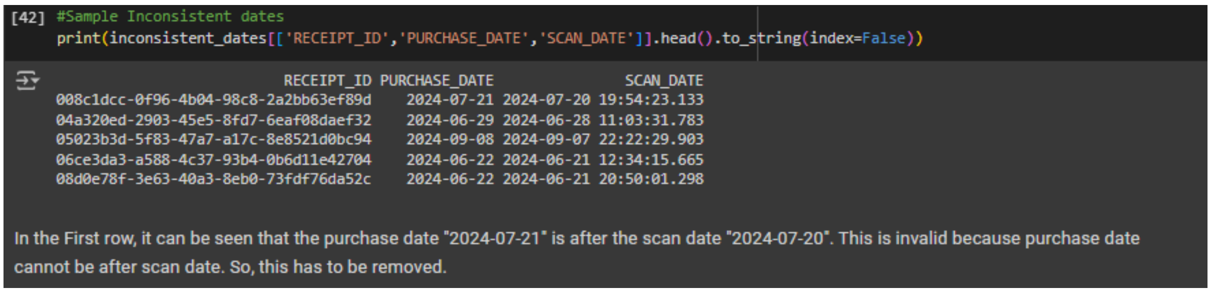
dtype: int64

In the FINAL_SALE column, It can be seen that null values are represented as a empty strings and it had to be changed as this is a numeric column.



```
[38] cleaned_transaction_df['FINAL_SALE'] = cleaned_transaction_df['FINAL_SALE'].replace('', np.nan) #Replacing empty string to null
```

Date Constraint Violation: Purchase Dates cannot be after Scan Dates as the user can scan receipts only after purchasing.



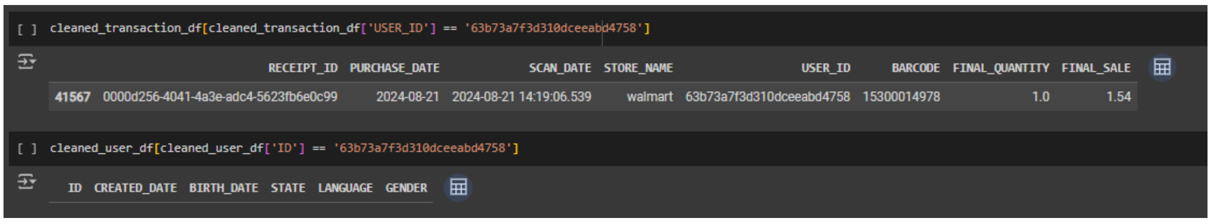
```
[42] #Sample Inconsistent dates
     print(inconsistent_dates[['RECEIPT_ID','PURCHASE_DATE','SCAN_DATE']].head().to_string(index=False))

                              RECEIPT_ID PURCHASE_DATE                SCAN_DATE
     008c1dcc-0f96-4b04-98c8-2a2bb63ef89d    2024-07-21 2024-07-20 19:54:23.133
     04a320ed-2903-45e5-8fd7-6eaf08daef32    2024-06-29 2024-06-28 11:03:31.783
     05023b3d-5f83-47a7-a17c-8e8521d0bc94    2024-09-08 2024-09-07 22:22:29.903
     06ce3da3-a588-4c37-93b4-0b6d11e42704    2024-06-22 2024-06-21 12:34:15.665
     08d0e78f-3e63-40a3-8eb0-73fdf76da52c    2024-06-22 2024-06-21 20:50:01.298
```

In the First row, it can be seen that the purchase date "2024-07-21" is after the scan date "2024-07-20". This is invalid because purchase date cannot be after scan date. So, this has to be removed.

The dataset had 47 inconsistent dates and had to be removed.

Data Integrity Issues:

The Transaction Table has User IDs that are not present in the User Table. This is logically not possible as a user has to create an account before scanning receipts through the app.



```
[ ] cleaned_transaction_df[cleaned_transaction_df['USER_ID'] == '63b73a7f3d310dceeabd4758']
```

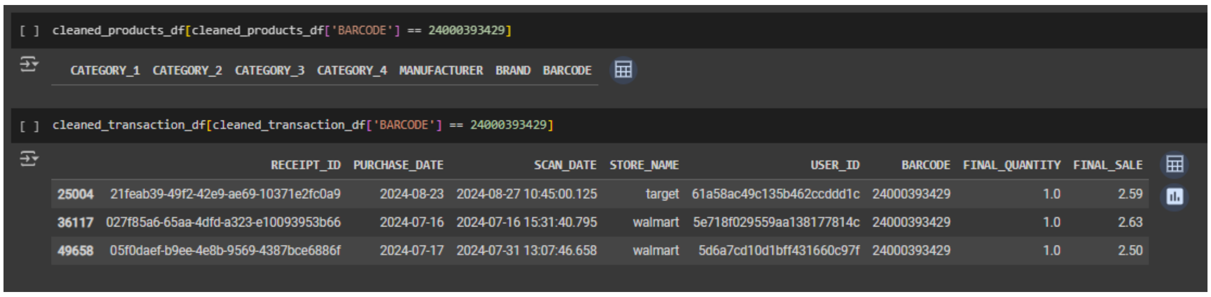
	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE
41567	0000d256-4041-4a3e-adc4-5623fb6e0c99	2024-08-21	2024-08-21 14:19:06.539	walmart	63b73a7f3d310dceeabd4758	15300014978	1.0	1.54

```
[ ] cleaned_user_df[cleaned_user_df['ID'] == '63b73a7f3d310dceeabd4758']
```

ID	CREATED_DATE	BIRTH_DATE	STATE	LANGUAGE	GENDER

Upon reviewing the data, I found out that the transaction table has only only 85 unique user IDs in the transaction table that correspond to data in the User table. This is a big data integrity issue as the transaction table contains 15,879 Unique IDs that are not present in the User table. The dataset must be validated as this will impact any analysis made.

The Transaction Table also has Barcodes that are not present in the Products Table. While this is logically possible as some products might not be in Fetch's product database, they should be added in the future.



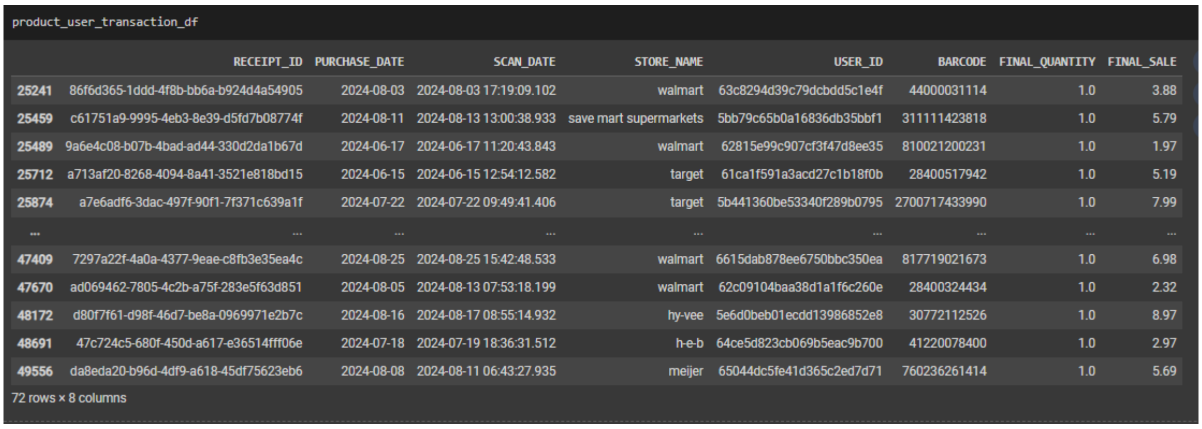
```
[ ] cleaned_products_df[cleaned_products_df['BARCODE'] == 24000393429]
```

CATEGORY_1	CATEGORY_2	CATEGORY_3	CATEGORY_4	MANUFACTURER	BRAND	BARCODE

```
[ ] cleaned_transaction_df[cleaned_transaction_df['BARCODE'] == 24000393429]
```

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE
25004	21feab39-49f2-42e9-ae69-10371e2fc0a9	2024-08-23	2024-08-27 10:45:00.125	target	61a58ac49c135b462ccddd1c	24000393429	1.0	2.59
36117	027f85a6-65aa-4dfd-a323-e10093953b66	2024-07-16	2024-07-16 15:31:40.795	walmart	5e718f029559aa138177814c	24000393429	1.0	2.63
49658	05f0daef-b9ee-4e8b-9569-4387bce6886f	2024-07-17	2024-07-31 13:07:46.658	walmart	5d6a7cd10d1bff431660c97f	24000393429	1.0	2.50

This dataframe has only 72 rows because it only has rows from the transaction table that have corresponding data in Users table(ID) and Products Table(Barcode).

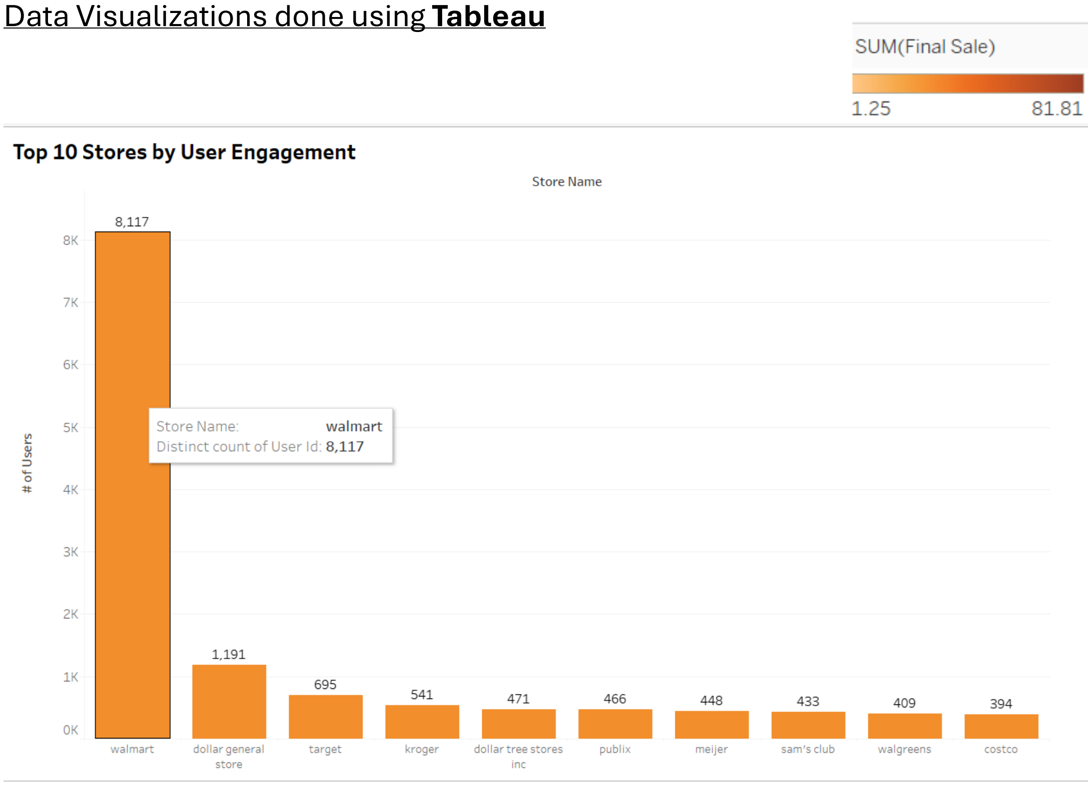


```
product_user_transaction_df
```

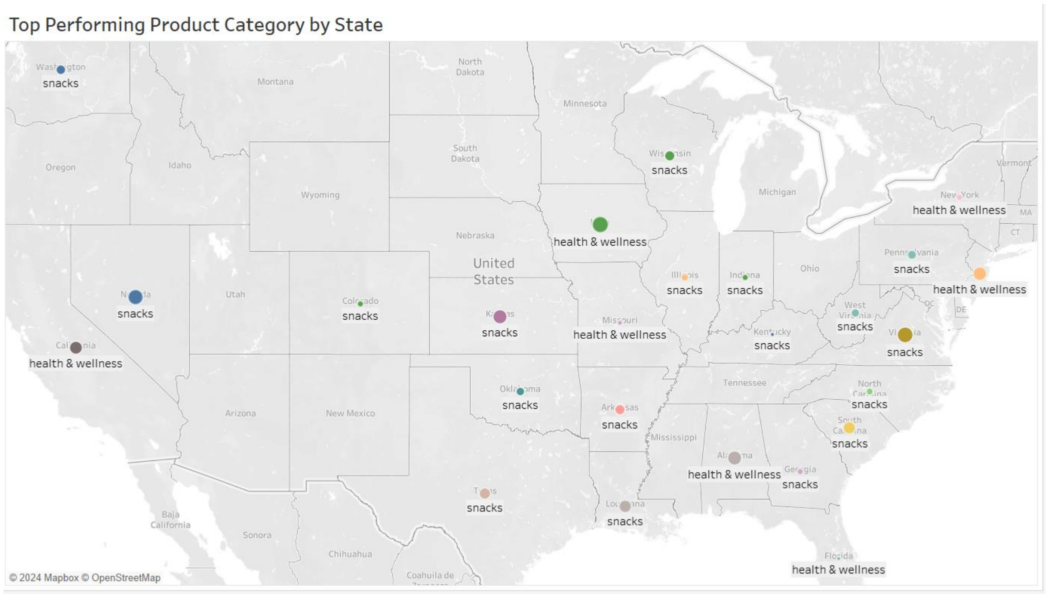
	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE
25241	86f6d365-1ddd-4f8b-bb6a-b924d4a54905	2024-08-03	2024-08-03 17:19:09.102	walmart	63c8294d39c79dcbdd5c1e4f	44000031114	1.0	3.88
25459	c61751a9-9995-4eb3-8e39-d5fd7b08774f	2024-08-11	2024-08-13 13:00:38.933	save mart supermarkets	5bb79c65b0a16836db35bbf1	311111423818	1.0	5.79
25489	9a6e4c08-b07b-4bad-ad44-330d2da1b67d	2024-06-17	2024-06-17 11:20:43.843	walmart	62815e99c907cf3f47d8ee35	810021200231	1.0	1.97
25712	a713af20-8268-4094-8a41-3521e818bd15	2024-06-15	2024-06-15 12:54:12.582	target	61ca1f591a3acd27c1b18f0b	28400517942	1.0	5.19
25874	a7e6adf6-3dac-497f-90f1-7f371c639a1f	2024-07-22	2024-07-22 09:49:41.406	target	5b441360be53340f289b0795	2700717433990	1.0	7.99
...	...	...	...	...	...	...	...	...
47409	7297a22f-4a0a-4377-9eae-c8fb3e35ea4c	2024-08-25	2024-08-25 15:42:48.533	walmart	6615dab878ee6750bbc350ea	817719021673	1.0	6.98
47670	ad069462-7805-4c2b-a75f-283e5f63d851	2024-08-05	2024-08-13 07:53:18.199	walmart	62c09104baa38d1a1f6c260e	28400324434	1.0	2.32
48172	d80f7f61-d98f-46d7-be8a-0969971e2b7c	2024-08-16	2024-08-17 08:55:14.932	hy-vee	5e6d0beb01ecdd13986852e8	30772112526	1.0	8.97
48691	47c724c5-680f-450d-a617-e36514fff06e	2024-07-18	2024-07-19 18:36:31.512	h-e-b	64ce5d823cb069b5eac9b700	41220078400	1.0	2.97
49556	da8eda20-b96d-4df9-a618-45df75623eb6	2024-08-08	2024-08-11 06:43:27.935	meijer	65044dc5fe41d365c2ed7d71	760236261414	1.0	5.69

72 rows × 8 columns

Data Visualizations done using **Tableau**



From the above plot, it can be seen that Walmart is the store where most of Fetch's active user base shops followed by Dollar General store. Increasing reward incentives at these top-performing stores will encourage continued loyalty and drive even more transactions. Exclusive promotions or bonus points at these stores could further boost user engagement.



From the above plot, it can be inferred that the top performing category (using sales) by state is Snacks followed by Health & Wellness. Targeted rewards campaigns for these categories could boost engagement.