In [2]:
```
import pandas as pd
```

In [38]:
```
transaction = pd.read_csv('TRANSACTION_TAKEHOME.csv')
products = pd.read_csv('PRODUCTS_TAKEHOME.csv')
users = pd.read_csv('USER_TAKEHOME.csv')
```

1. Data Exploration and Data Cleanup

1.1 Transaction Dataset

In [4]:
```
transaction.head()
```

Out[4]:

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID
0	0000d256-4041-4a3e-adc4-5623fb6e0c99	2024-08-21	2024-08-21 14:19:06.539 Z	WALMART	63b73a7f3d310dceeabd4758
1	0001455d-7a92-4a7b-a1d2-c747af1c8fd3	2024-07-20	2024-07-20 09:50:24.206 Z	ALDI	62c08877baa38d1a1f6c211a
2	00017e0a-7851-42fb-bfab-0baa96e23586	2024-08-18	2024-08-19 15:38:56.813 Z	WALMART	60842f207ac8b7729e472020
3	000239aa-3478-453d-801e-66a82e39c8af	2024-06-18	2024-06-19 11:03:37.468 Z	FOOD LION	63fcd7cea4f8442c3386b589
4	00026b4c-dfe8-49dd-b026-4c2f0fd5c6a1	2024-07-04	2024-07-05 15:56:43.549 Z	RANDALLS	6193231ae9b3d75037b0f928

In [5]:
```
# Count missing values in each column
print(transaction.isnull().sum())
```

```
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
```

In [6]:
```
# Check data types of each column
print(transaction.dtypes)
```

```
RECEIPT_ID         object
PURCHASE_DATE      object
SCAN_DATE          object
STORE_NAME         object
USER_ID            object
BARCODE           float64
FINAL_QUANTITY     object
FINAL_SALE         object
dtype: object
```

In [7]:
```
# Convert columns to appropriate data types
transaction['PURCHASE_DATE'] = pd.to_datetime(transaction['PURCHASE_DATE'])
transaction['SCAN_DATE'] = pd.to_datetime(transaction['SCAN_DATE'])
```

In [8]:
```
transaction['FINAL_QUANTITY'] = pd.to_numeric(transaction['FINAL_QUANTITY'], errors
```

In [9]:
```
transaction['FINAL_QUANTITY'] = transaction['FINAL_QUANTITY'].replace('zero', 0).fi
```

In [10]:
```
transaction['FINAL_QUANTITY']
```

Out[10]:
```
0        1
1        0
2        1
3        0
4        1
        ..
49995    2
49996    1
49997    1
49998    1
49999    1
Name: FINAL_QUANTITY, Length: 50000, dtype: int32
```

In [11]:
```
# Convert FINAL_SALE to numeric, coercing errors (e.g., None to NaN)
transaction['FINAL_SALE'] = pd.to_numeric(transaction['FINAL_SALE'], errors='coerce
```

In [12]:
```
#data types after cleanup
transaction.dtypes
```

Out[12]:
```
RECEIPT_ID                    object
PURCHASE_DATE          datetime64[ns]
SCAN_DATE         datetime64[ns, UTC]
STORE_NAME                    object
USER_ID                       object
BARCODE                      float64
FINAL_QUANTITY                 int32
FINAL_SALE                   float64
dtype: object
```

In [13]:
```
# Convert PURCHASE_DATE to timezone-aware
transaction['PURCHASE_DATE'] = pd.to_datetime(transaction['PURCHASE_DATE']).dt.tz_l
```

```
# Add a column to flag rows where SCAN_DATE < PURCHASE_DATE
transaction['DATE_DISCREPANCY'] = transaction['SCAN_DATE'] < transaction['PURCHASE_
```

In [14]:
```
#filter rows where scan date is before purchase date
transaction[transaction['DATE_DISCREPANCY'] == True]
```

Out[14]:

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	
51	008c1dcc-0f96-4b04-98c8-2a2bb63ef89d	2024-07-21 00:00:00+00:00	2024-07-20 19:54:23.133000+00:00	WALMART	5dc24cdb682
455	04a320ed-2903-45e5-8fd7-6eaf08daef32	2024-06-29 00:00:00+00:00	2024-06-28 11:03:31.783000+00:00	DOLLAR GENERAL STORE	62855f677086
494	05023b3d-5f83-47a7-a17c-8e8521d0bc94	2024-09-08 00:00:00+00:00	2024-09-07 22:22:29.903000+00:00	SHOP RITE	666a43c77c0
675	06ce3da3-a588-4c37-93b4-0b6d11e42704	2024-06-22 00:00:00+00:00	2024-06-21 12:34:15.665000+00:00	BIG LOTS	646f6ffb7a34
870	08d0e78f-3e63-40a3-8eb0-73fdf76da52c	2024-06-22 00:00:00+00:00	2024-06-21 20:50:01.298000+00:00	DOLLAR GENERAL STORE	664cafb6e04
...	...	...	...	...	
46034	08d0e78f-3e63-40a3-8eb0-73fdf76da52c	2024-06-22 00:00:00+00:00	2024-06-21 20:50:01.298000+00:00	DOLLAR GENERAL STORE	664cafb6e04
46539	718aa730-b62f-4e18-8dba-1d7105dac341	2024-09-05 00:00:00+00:00	2024-09-04 20:14:00.374000+00:00	WALMART	5e0f561efa89
46941	af2b818f-4a92-4e98-958c-65f2ce0b271d	2024-06-15 00:00:00+00:00	2024-06-14 10:57:23.892000+00:00	DOLLAR GENERAL STORE	64de6465516
47653	72bb7b71-d958-4a46-ae62-43abdeb0e693	2024-06-15 00:00:00+00:00	2024-06-14 19:55:56.672000+00:00	WALMART	649726ea127c
47837	99c2e8dc-9dc7-4267-9342-0b19c3fb35a0	2024-06-15 00:00:00+00:00	2024-06-14 22:07:18.702000+00:00	WALMART	5e48ddd01a90

94 rows × 9 columns

The Code shows the rows with date discrepancies. There are 9 rows where scan date is before purchase date, meaning there is something wrong with these records. We should take a look into the record and see where it went wrong to improve our data recordings later on.

1.2 Users Dataset

In [15]:
```
users.head()
```

Out[15]:

	ID	CREATED_DATE	BIRTH_DATE	STATE	LANGUAGE	GENDER
0	5ef3b4f17053ab141787697d	2020-06-24 20:17:54.000 Z	2000-08-11 00:00:00.000 Z	CA	es-419	female
1	5ff220d383fcfc12622b96bc	2021-01-03 19:53:55.000 Z	2001-09-24 04:00:00.000 Z	PA	en	female
2	6477950aa55bb77a0e27ee10	2023-05-31 18:42:18.000 Z	1994-10-28 00:00:00.000 Z	FL	es-419	female
3	658a306e99b40f103b63ccf8	2023-12-26 01:46:22.000 Z	NaN	NC	en	NaN
4	653cf5d6a225ea102b7ecdc2	2023-10-28 11:51:50.000 Z	1972-03-19 00:00:00.000 Z	PA	en	female

In [16]:
```
# show data type for each column before any clean up
users.dtypes
```

Out[16]:
```
ID              object
CREATED_DATE    object
BIRTH_DATE      object
STATE           object
LANGUAGE        object
GENDER          object
dtype: object
```

In [17]:
```
# Convert CREATED_DATE and BIRTH_DATE to datetime
users['CREATED_DATE'] = pd.to_datetime(users['CREATED_DATE'], errors='coerce')
users['BIRTH_DATE'] = pd.to_datetime(users['BIRTH_DATE'], errors='coerce')
```

In [18]:
```
users.head()
```

Out[18]:

	ID	CREATED_DATE	BIRTH_DATE	STATE	LANGUAGE	GENDER
0	5ef3b4f17053ab141787697d	2020-06-24 20:17:54+00:00	2000-08-11 00:00:00+00:00	CA	es-419	female
1	5ff220d383fcfc12622b96bc	2021-01-03 19:53:55+00:00	2001-09-24 04:00:00+00:00	PA	en	female
2	6477950aa55bb77a0e27ee10	2023-05-31 18:42:18+00:00	1994-10-28 00:00:00+00:00	FL	es-419	female
3	658a306e99b40f103b63ccf8	2023-12-26 01:46:22+00:00	NaT	NC	en	NaN
4	653cf5d6a225ea102b7ecdc2	2023-10-28 11:51:50+00:00	1972-03-19 00:00:00+00:00	PA	en	female

In [19]:
```
# Check missing values
users.isnull().sum()
```

Out[19]:
```
ID                  0
CREATED_DATE        0
BIRTH_DATE       3675
STATE            4812
LANGUAGE        30508
GENDER           5892
dtype: int64
```

In [20]:
```
# Display unique states
unique_states = users['STATE'].unique()
print("Unique states in the dataset:", unique_states)
```

```
Unique states in the dataset: ['CA' 'PA' 'FL' 'NC' 'NY' 'IN' nan 'OH' 'TX' 'NM' 'PR'
'CO' 'AZ' 'RI' 'MO'
 'NJ' 'MA' 'TN' 'LA' 'NH' 'WI' 'IA' 'GA' 'VA' 'DC' 'KY' 'SC' 'MN' 'WV'
 'DE' 'MI' 'IL' 'MS' 'WA' 'KS' 'CT' 'OR' 'UT' 'MD' 'OK' 'NE' 'NV' 'AL'
 'AK' 'AR' 'HI' 'ME' 'ND' 'ID' 'WY' 'MT' 'SD' 'VT']
```

In [21]:
```
# Display unique languages
unique_languages = user['LANGUAGE'].unique()
print("Unique languages in the dataset:", unique_languages)
```

```
Unique languages in the dataset: ['es-419' 'en' nan]
```

In [22]:
```
duplicates = users[users.duplicated(subset=['ID'])]
print(f"Number of duplicate rows: {len(duplicates)}")
```

```
Number of duplicate rows: 0
```

There are no duplicate users in the dataset, so it it good to go

In [23]:
```
users = user.copy()
users['GENDER'] = users['GENDER'].fillna('Unknown') #fill in the null values with u
```

In [24]:
```
users['LANGUAGE'] = users['LANGUAGE'].fillna('Unknown') #fill in the null values wi
```

In [25]:
```
users['BIRTH_DATE'] = users['BIRTH_DATE'].fillna(pd.Timestamp("1900-01-01")) #Repla
```

1.3 Products Dataset

In [26]:
```
products.head()
```

Out[26]:

	CATEGORY_1	CATEGORY_2	CATEGORY_3	CATEGORY_4	MANUFACTURER	BRAND	
0	Health & Wellness	Sexual Health	Conductivity Gels & Lotions	NaN	NaN	NaN	7.
1	Snacks	Puffed Snacks	Cheese Curls & Puffs	NaN	NaN	NaN	2.
2	Health & Wellness	Hair Care	Hair Care Accessories	NaN	PLACEHOLDER MANUFACTURER	ELECSOP	4.
3	Health & Wellness	Oral Care	Toothpaste	NaN	COLGATE-PALMOLIVE	COLGATE	3.
4	Health & Wellness	Medicines & Treatments	Essential Oils	NaN	MAPLE HOLISTICS AND HONEYDEW PRODUCTS INTERCHA...	MAPLE HOLISTICS	8.

In [27]:
```
products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 845552 entries, 0 to 845551
Data columns (total 7 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   CATEGORY_1    845441 non-null  object 
 1   CATEGORY_2    844128 non-null  object 
 2   CATEGORY_3    784986 non-null  object 
 3   CATEGORY_4    67459 non-null   object 
 4   MANUFACTURER  619078 non-null  object 
 5   BRAND         619080 non-null  object 
 6   BARCODE       841527 non-null  float64
dtypes: float64(1), object(6)
memory usage: 45.2+ MB
```

In [28]:
```
# Check missing values
products.isnull().sum()
```

Out[28]:
```
CATEGORY_1          111
CATEGORY_2         1424
CATEGORY_3        60566
CATEGORY_4       778093
MANUFACTURER     226474
BRAND            226472
BARCODE            4025
dtype: int64
```

Something to notice here is number of missing values for barcode. This column is essential to recognize which product it is referring to. Without this data, I think it will be hard to make insights from those records. But I will still keep those records for now in case it will influence our analysis later on.

In [29]:
```
products['MANUFACTURER'] = products['MANUFACTURER'].replace("PLACEHOLDER MANUFACTUR
```

In [30]:
```
products['MANUFACTURER'] = products['MANUFACTURER'].fillna('Unknown') #fill in the
```

In [31]:
```
products['BRAND'] = products['BRAND'].fillna('Unknown') #fill in the null values wi
```

In [32]:
```
# Drop duplicates based on all columns
products_cleaned = products.drop_duplicates()
```

In [33]:
```
products_cleaned
```

Out[33]:

	CATEGORY_1	CATEGORY_2	CATEGORY_3	CATEGORY_4	MANUFACTURER	BRA
0	Health & Wellness	Sexual Health	Conductivity Gels & Lotions	NaN	Unknown	Unkn
1	Snacks	Puffed Snacks	Cheese Curls & Puffs	NaN	Unknown	Unkn
2	Health & Wellness	Hair Care	Hair Care Accessories	NaN	Unknown	ELEC
3	Health & Wellness	Oral Care	Toothpaste	NaN	COLGATE-PALMOLIVE	COLG
4	Health & Wellness	Medicines & Treatments	Essential Oils	NaN	MAPLE HOLISTICS AND HONEYDEW PRODUCTS INTERCHA...	MA HOLIS
...	...	...	...	...	...	
845547	Health & Wellness	Topical Muscle & Joint Relief Treatments	Braces & Wraps	NaN	Unknown	Unkn
845548	Snacks	Cookies	NaN	NaN	TREEHOUSE FOODS, INC.	LOFTHO
845549	Snacks	Candy	Confection Candy	NaN	HARIBO GMBH & CO KG	HAR
845550	Snacks	Nuts & Seeds	Hazelnuts	NaN	DOUBLE-COLA CO	JUN
845551	Health & Wellness	First Aid	First Aid Kits	NaN	3M	NEXC

845337 rows × 7 columns

In [34]:

```
# Exclude null BARCODE values
non_null_barcodes = products_cleaned [products_cleaned ['BARCODE'].notnull()]

# Find duplicate BARCODE entries
duplicate_barcodes = non_null_barcodes[non_null_barcodes.duplicated(subset=['BARCOD

# Display duplicates
print("Duplicate BARCODE entries (excluding null values):\n", duplicate_barcodes)
```

```
Duplicate BARCODE entries (excluding null values):
              CATEGORY_1    CATEGORY_2                  CATEGORY_3  \
162     Health & Wellness  Hair Removal          Shaving Gel & Cream
28421   Health & Wellness     Hair Care                   Hair Color
36017              Snacks         Candy           Candy Variety Pack
37152              Snacks         Candy             Confection Candy
56987              Snacks  Nuts & Seeds                     Almonds
96438              Snacks      Crackers               Other Crackers
108325             Snacks         Candy                       Mints
114131             Snacks         Candy               Chocolate Candy
123194  Health & Wellness     Skin Care       Lip Balms & Treatments
127335             Snacks  Nuts & Seeds                   Pistachios
132547             Snacks       Cookies                         NaN
137250             Snacks  Nuts & Seeds                     Peanuts
139121             Snacks         Candy               Chocolate Candy
144679             Snacks         Candy               Chocolate Candy
171015             Snacks  Nuts & Seeds                  Covered Nuts
181902             Snacks         Candy             Confection Candy
184572             Snacks         Candy               Chocolate Candy
193347             Snacks         Candy               Chocolate Candy
206172             Snacks      Crackers               Graham Crackers
213340  Health & Wellness     Hair Care                   Hair Color
216314             Snacks         Candy               Chocolate Candy
260690             Snacks         Candy               Chocolate Candy
274698             Snacks         Candy               Chocolate Candy
287404             Snacks         Candy               Chocolate Candy
300327             Snacks         Candy             Confection Candy
304021  Health & Wellness     Hair Care                   Hair Color
333771  Health & Wellness     Skin Care                   Eye Creams
379746             Snacks         Candy                          Gum
402382             Snacks         Candy             Confection Candy
422809             Snacks         Candy               Chocolate Candy
428256  Health & Wellness     Skin Care  Facial Lotion & Moisturizer
454537             Snacks         Candy               Chocolate Candy
468720             Snacks         Candy                          Gum
528709             Snacks         Candy                          Gum
539918             Snacks         Candy                          Gum
585634             Snacks         Candy               Chocolate Candy
594684             Snacks         Candy                          Gum
596778  Health & Wellness     First Aid          Ointments & Liquids
610681             Snacks  Nuts & Seeds                  Snack Seeds
645266             Snacks         Chips                       Crisps
668717  Health & Wellness     Skin Care                   Eye Creams
681268             Snacks  Nuts & Seeds                     Almonds
708466             Snacks         Candy                       Mints
709607  Health & Wellness     Hair Care                   Hair Color
717446             Snacks         Candy                          Gum
720019             Snacks      Crackers               Graham Crackers
766255             Snacks         Candy               Chocolate Candy
768682  Health & Wellness     Skin Care    Skin Toners & Astringents
776953             Snacks  Nuts & Seeds                  Covered Nuts
777589             Snacks         Candy                       Mints
782036             Snacks         Chips                      Crisps
783021             Snacks       Cookies                         NaN
810359  Health & Wellness  Hair Removal          Shaving Gel & Cream
841230             Snacks         Candy               Chocolate Candy
```

	CATEGORY_4	MANUFACTURER
162	Women's Shaving Gel & Cream	Unknown
28421	NaN	HENKEL
36017	NaN	THE HERSHEY COMPANY
37152	NaN	PERFETTI VAN MELLE
56987	NaN	Unknown
96438	NaN	Unknown
108325	NaN	PERFETTI VAN MELLE
114131	NaN	GRUPO NACIONAL DE CHOCOLATES SA
123194	Medicated Lip Treatments	E.T. BROWNE DRUG CO., INC.
127335	NaN	LIDL US, LLC
132547	NaN	Unknown
137250	NaN	MARS WRIGLEY
139121	NaN	Unknown
144679	NaN	THE HERSHEY COMPANY
171015	NaN	TRADER JOE'S
181902	NaN	GRUPO NACIONAL DE CHOCOLATES SA
184572	NaN	Unknown
193347	NaN	MARS WRIGLEY
206172	NaN	LIDL US, LLC
213340	NaN	HENKEL
216314	NaN	FERRERO GROUP
260690	NaN	LIDL US, LLC
274698	NaN	PROCTER & GAMBLE
287404	NaN	NESTLE
300327	NaN	PERFETTI VAN MELLE
304021	NaN	HENKEL
333771	NaN	Unknown
379746	NaN	Unknown
402382	NaN	THE HERSHEY COMPANY
422809	NaN	THE HERSHEY COMPANY
428256	NaN	R.M. PALMER COMPANY, LLC
454537	NaN	MARS WRIGLEY
468720	NaN	GENERAL LICENSED IP MANUFACTURER
528709	NaN	THE HERSHEY COMPANY
539918	NaN	THE HERSHEY COMPANY
585634	NaN	KINDER'S
594684	NaN	THE HERSHEY COMPANY
596778	NaN	THE JM SMUCKER COMPANY
610681	NaN	SUNRIDGE FARMS
645266	NaN	TRADER JOE'S
668717	NaN	WALA HEILMITTEL GMBH
681268	NaN	TRADER JOE'S
708466	NaN	LOTUS BRANDS INC
709607	NaN	HENKEL
717446	NaN	GENERAL LICENSED IP MANUFACTURER
720019	NaN	Unknown
766255	NaN	THE HERSHEY COMPANY
768682	Astringents	THE JM SMUCKER COMPANY
776953	NaN	Unknown
777589	NaN	PERFETTI VAN MELLE
782036	NaN	Unknown
783021	NaN	Unknown
810359	Men's Shaving Gel & Cream	Unknown
841230	NaN	MARS WRIGLEY

	BRAND	BARCODE
162	PRORASO	8.019914e+07
28421	SCHWARZKOPF	5.233692e+10
36017	HERSHEY'S	3.422007e+06
37152	MENTOS	8.730629e+07
56987	BRAND NOT KNOWN	2.015908e+07
96438	Unknown	2.073325e+07
108325	MENTOS	8.710854e+07
114131	NUTRESA	7.505306e+07
123194	PALMER'S SKIN & HAIR CARE	1.018158e+06
127335	LIDL	2.052244e+07
132547	Unknown	2.003108e+07
137250	M&M'S	4.003207e+06
139121	BRAND NOT KNOWN	4.043100e+05
144679	REESE'S	3.431207e+06
171015	TRADER JOE'S	9.693070e+05
181902	NUTRESA	7.505306e+07
184572	Unknown	5.042617e+07
193347	M&M'S	4.003207e+06
206172	LIDL	2.073306e+07
213340	SCHWARZKOPF	1.700033e+10
216314	KINDER	8.031017e+07
260690	LIDL	2.052244e+07
274698	BOUNTY	4.011122e+07
287404	NESTLE	5.042617e+07
300327	MENTOS	8.710854e+07
304021	GÖT2B	1.700033e+10
333771	Unknown	4.220849e+07
379746	Unknown	3.600540e+08
402382	REESE'S	3.431207e+06
422809	HERSHEY'S	3.422007e+06
428256	PALMER	1.018158e+06
454537	MARS	4.011122e+07
468720	DISNEY	3.454503e+06
528709	ICE BREAKERS	3.454503e+06
539918	BUBBLE YUM	3.473009e+06
585634	KINDER'S	8.031017e+07
594684	ICE BREAKERS	3.484708e+06
596778	DICKINSON'S	5.265169e+06
610681	SUNRIDGE FARMS	7.019830e+05
645266	TRADER JOE'S	7.019830e+05
668717	DR HAUSCHKA	4.220849e+07
681268	TRADER JOE'S	9.693070e+05
708466	ECO-DENT	3.600540e+08
709607	GÖT2B	5.233692e+10
717446	DISNEY	3.484708e+06
720019	PRIVATE LABEL	2.073306e+07
766255	REESE'S	3.473009e+06
768682	DICKINSON'S	5.265169e+06
776953	BRAND NOT KNOWN	2.015908e+07
777589	MENTOS	8.730629e+07
782036	PRIVATE LABEL	2.073325e+07
783021	PRIVATE LABEL	2.003108e+07
810359	PRORASO	8.019914e+07
841230	M&M'S	4.043100e+05

From this result, we see that there are still some duplicate barcodes in the dataset. These duplicate barcodes have at least one category name that is different. To resolve this, I group the data by barcode, and join the category if it is not the same. This will ensure we only have one record for each barcode. The code is performed below.

In [35]:
```
# Group by BARCODE and aggregate the category columns
merged_products = (
    products.groupby('BARCODE', dropna=False)
    .agg({
        'CATEGORY_1': lambda x: ', '.join(x.dropna().unique()),
        'CATEGORY_2': lambda x: ', '.join(x.dropna().unique()),
        'CATEGORY_3': lambda x: ', '.join(x.dropna().unique()),
        'CATEGORY_4': lambda x: ', '.join(x.dropna().unique()),
        'MANUFACTURER': lambda x: ', '.join(x.dropna().unique()),
        'BRAND': lambda x: ', '.join(x.dropna().unique()),
    })
    .reset_index()
)

# Display the consolidated dataset
print(merged_products)
```

```
KeyboardInterrupt
```

In [ ]:
```
# Exclude null BARCODE values
non_null_barcodes = merged_products [merged_products ['BARCODE'].notnull()]

# Find duplicate BARCODE entries
duplicate_barcodes = non_null_barcodes[non_null_barcodes.duplicated(subset=['BARCOD

# Display duplicates
print("Duplicate BARCODE entries (excluding null values):\n", duplicate_barcodes)
```

As a result, we can see there are no duplicate barcodes excluding the null barcodes.

2. Summarize the Questions

1. Are there any data quality issues present?

To summarize the data issues, there are missing values and duplicates across the datasets, with variations in their nature and impact.

For the transaction dataset, there are missing values in FINAL_SALE and BARCODE. This could lead to incomplete information about the total receipt amounts and product details. A more significant issue is the date discrepancies observed in some cases, where the SCAN_DATE occurs before the PURCHASE_DATE, which is logically inconsistent.

For the user dataset, missing values are present in BIRTH_DATE, GENDER, and STATE. While these gaps may not critically affect the analysis, having more complete information would

enhance our understanding of the customers. However, addressing these gaps is likely not the top priority at this stage.

Finally, for the products dataset, the primary concern lies with the BARCODE. There are some missing values (approximately 0.4%), and there are duplicate records for the same BARCODE. From my understanding, each product should correspond to a unique BARCODE, making it the primary key of the table. These duplicates and missing values could cause inconsistencies in identifying products accurately.

2. Are there any fields that are challenging to understand?

I was a bit confused about final quantity and final sale in the transaction dataset because there are zeros in final quantity where final sale has a number and some final quantities with number but final sale has no number. Also, I think the purpose and hierarchy of CATEGORY_1 through CATEGORY_4 might need some more explanations.

In [ ]:

Additional work

Top 5 brands by receipts scanned among users 21 and over

In [42]:
```
# Merge transaction with products to get the BRAND column
transaction_products = transaction.merge(products, on='BARCODE', how='left')

# Merge with users to get users aged 21 and over
users['BIRTH_DATE'] = pd.to_datetime(users['BIRTH_DATE'], errors='coerce').dt.tz_lo
current_date = pd.Timestamp.now().tz_localize(None)
users['AGE'] = users['BIRTH_DATE'].apply(lambda x: (current_date - x).days // 365 i
users_21_and_over = users[users['AGE'] >= 21]

transaction_users = transaction_products.merge(users_21_and_over, left_on='USER_ID'

# Group by BRAND and count receipts
brand_receipts = transaction_users.groupby('BRAND').size().reset_index(name='RECEIP

# Sort by receipt count in descending order and select the top 5 brands
top_5_brands = brand_receipts.sort_values(by='RECEIPT_COUNT', ascending=False).head

print("Top 5 Brands by Receipts Scanned (21+):")
print(top_5_brands)
```

```
Top 5 Brands by Receipts Scanned (21+):
                         BRAND  RECEIPT_COUNT
164                  COCA-COLA            628
37   ANNIE'S HOMEGROWN GROCERY            576
204                       DOVE            558
69                    BAREFOOT            552
535                     ORIBE            504
```

In [ ]:

In [ ]:

In [ ]: