FIRST EXERCISE

Two questions were asked in this exercise

- Part – 1: Are there any data quality issues present?
- Part – 2: Are there any fields that are challenging to understand

Part 1 - Data Quality Exploration

The unstructured data that was provided consists of three csv files namely – Products, Users and Transactions. To assess the quality of the data provided, the following process was implemented for every table.

Steps followed:

1) **Datatype verification** – Checking if all the variables in the tables are of appropriate datatype and converting them for proper analysis. The tables are also checked for the presence of records which are in the form of data structures such as lists or arrays.
2) **Duplicates check** – Examining if there are any duplicate records present in each of the tables.
3) **Missing values Check** – Checking for missing records in all the three datasets according to the variables explored. More importantly, checking for the uniqueness of the joining keys – namely, user_id and barcode.
4) **Check for errors** – Examining the data for chronological errors, logical errors etc.
5) **Miscellaneous checks** according to the nature of variables in the data – such as distribution, repetition etc.

Note: Before loading the data into Python, the csv files were opened in Excel and were glanced into for some quick insights about the data. It was noticed that in both “TRANSACTIONS_TAKEHOME” and “PRODUCTS_TAKEHOME” csv files, several rows in the “BARCODE” column were flagged and on further inspection, it was noticed that all the barcodes starting with either “0” or “00” started with an apostrophe (‘) mark. When the barcodes were looked up online to check for legitimacy, the barcodes that started with two zeros ("00") with a preceding apostrophe mark were not present in the database. While not all the barcodes starting with '00' were tested, those that were looked up turned out to be invalid. I would flag this as a **Data Quality Issue**.

Furthermore, it was also observed that numeric-like strings are automatically converted into a float or integer type when being loaded into Python and the leading zeros in the number are removed automatically, as depicted below:

```
[ ] product = pd.read_csv('/content/drive/My Drive/PRODUCTS_TAKEHOME.csv')
    product['BARCODE'].dtype
```

```
dtype('float64')
```

```
[ ] product['BARCODE'].head(20)
```

	BARCODE
0	7.964944e+11
1	2.327801e+10
2	4.618178e+11
3	3.500047e+10
4	8.068109e+11
5	6.626585e+11
6	6.177376e+11
7	7.501839e+12

Barcodes loaded as float datatype with no leading zeros

So, the BARCODE columns in both the csv files were loaded as strings to retain the leading zeros.

Users Table Data Analysis

Datatype Verification

All the variables' datatypes were checked for legitimacy.

```
[6] users.dtypes
```

	0
ID	object
CREATED_DATE	object
BIRTH_DATE	object
STATE	object
LANGUAGE	object
GENDER	object

Datatypes of Users Table

"Object" datatype is a flexible datatype generally refers to strings. The "BIRTH_DATE" and "CREATED_DATE" columns were converted into appropriate date-time objects keeping in mind the time stamp and the time zone information.

```python
users['BIRTH_DATE'] = pd.to_datetime(users['BIRTH_DATE'])
users['CREATED_DATE'] = pd.to_datetime(users['CREATED_DATE'])

# Make 'datetime_no_tz' timezone-aware by localizing it to UTC
users['BIRTH_DATE'] = users['BIRTH_DATE'].dt.tz_convert('UTC')
users['CREATED_DATE'] = users['CREATED_DATE'].dt.tz_convert('UTC')

users.dtypes
```

	0
ID	object
CREATED_DATE	datetime64[ns, UTC]
BIRTH_DATE	datetime64[ns, UTC]
STATE	object
LANGUAGE	object
GENDER	object

dtype: object

Corrected Data Types

The table was then checked for the presence of arrays, lists or dictionaries in any of the columns. They were not present in any of the columns.

```python
def check_column_datastructures(column):
    return any(isinstance(x, (list, dict, np.ndarray)) for x in column)

# Applying the function to each column
columns_with_structures = {col: check_column_datastructures(users[col]) for col in users.columns}

# Displaying which columns contain arrays, lists, or dictionaries
print("Columns containing arrays, lists, or dictionaries:")
for col, contains_structure in columns_with_structures.items():
    if contains_structure:
        print(f"{col} contains complex structures")
```

```
Columns containing arrays, lists, or dictionaries:
```

Checking for the presence of other data structures

The columns were investigated with the help of summary statistics:

```python
users.describe(include = 'all')
```

	ID	CREATED_DATE	BIRTH_DATE	STATE	LANGUAGE	GENDER
count	100000	100000	96325	95188	69492	94108
unique	100000	99942	54721	52	2	11
top	5ef3b4f17053ab141787697d	2023-01-12 18:30:15.000 Z	1970-01-01 00:00:00.000 Z	TX	en	female
freq	1	2	1272	9028	63403	64240

Variable Statistics before date-time conversion

```
[11] users.describe(include = 'all')
```

	ID	CREATED_DATE	BIRTH_DATE	STATE	LANGUAGE	GENDER
count	100000	100000	96325	95188	69492	94108
unique	100000	NaN	NaN	52	2	11
top	5ef3b4f17053ab141787697d	NaN	NaN	TX	en	female
freq	1	NaN	NaN	9028	63403	64240
mean	NaN	2022-01-07 05:31:20.864859648+00:00	1984-09-02 02:39:04.710417920+00:00	NaN	NaN	NaN
min	NaN	2014-04-18 23:14:55+00:00	1900-01-01 00:00:00+00:00	NaN	NaN	NaN
25%	NaN	2020-12-01 21:16:19+00:00	1974-03-04 00:00:00+00:00	NaN	NaN	NaN
50%	NaN	2022-03-07 01:03:37+00:00	1985-10-25 00:00:00+00:00	NaN	NaN	NaN
75%	NaN	2023-01-30 13:47:44.500000+00:00	1998-02-02 05:00:00+00:00	NaN	NaN	NaN
max	NaN	2024-09-11 17:59:15+00:00	2022-04-03 07:00:00+00:00	NaN	NaN	NaN

Variable Statistics after date-time conversion

Observations:

- It was observed that the "ID" column had **no missing values** and all the present records **were unique**. This proves that **there are no duplicate records** present in the users table.
- Some values in the "CREATED_DATE" values seem to be recurring. Further investigation was performed to find potential data quality issues.
- There are some missing "BIRTH_DATE" values too. Secondly, the earliest year in the "BIRTH_DATE" column is 1900 which is very early. This might hinder any analysis concerning the ages of the users.
- There are only 2 unique values in the language column with a lot of missing values.

Missing Values Check

```
users.isna().sum()
```

	0
ID	0
CREATED_DATE	0
BIRTH_DATE	3675
STATE	4812
LANGUAGE	30508
GENDER	5892

dtype: int64

Users Table Missing Values

```
[13] #Lets check for percentages
     users_isna = users.isna().sum()
     users_isna/users.shape[0]
```

	0
ID	0.00000
CREATED_DATE	0.00000
BIRTH_DATE	0.03675
STATE	0.04812
LANGUAGE	0.30508
GENDER	0.05892

dtype: float64

Missing Values Percentage

Data Quality Issues:

- There are 3.67% of missing birth date values.
- More than 30% of the language column is missing. State and Gender columns also have 4.81% and 5.89% missing values respectively.

Thorough analysis was carried out for better understanding.

Check for Errors

Chronological Errors – The presence of records where the created date is earlier than the birth date was checked.

```
[ ] users_cd_b4_bd = users[(users['BD<CD']==False) & (~users['BIRTH_DATE'].isna())]
    users_cd_b4_bd.head()
```

	ID	CREATED_DATE	BIRTH_DATE	STATE	LANGUAGE	GENDER	BD<CD
41974	5f31fc048fa1e914d38d6952	2020-08-11 02:01:41+00:00	2020-10-02 15:27:28+00:00	CA	NaN	NaN	False

Records with created date earlier than the birth date

Minor Data Quality Issue – There is one record where the chronology is illogical. So, it was removed.

Miscellaneous Checks

Focus on the CREATED_DATE and the BIRTH_DATE columns

Observations:

1) When the number of occurrences of each birth date in the "BIRTH_DATE" column was listed, it was observed that 1st January 1970 was recorded 1272 times in the table. This might be because the application's default date is set to 1st January 1970.

```
[ ] #Checking the number of occurences of each date in the "BIRTH_DATE" column
    users['BIRTH_DATE'].value_counts()
```

BIRTH_DATE	count
1970-01-01 00:00:00+00:00	1272
1979-12-11 08:00:00+00:00	63
2000-12-12 00:00:00+00:00	28
2000-12-31 00:00:00+00:00	23
2001-01-01 00:00:00+00:00	16
...	...
2004-02-21 08:00:00+00:00	1

Number of Occurrences of each date in the 'BIRTH_DATE' column

2) While checking the records where the birth dates were missing, the focus shifted to the CREATED_DATE records where the 'BIRTH_DATE' column's records were missing. Then, when the number of created accounts where the birth dates were missing was compared to the total number of accounts created monthly, the following was observed:

```python
#Comparing the counts during the common months in both the cases.
users_crdate_monthwise = users['CREATED_MONTH'].value_counts().sort_index().tail(35)
users_crdate_bdmiss_mwise = users_bdate_miss['CREATED_MONTH'].value_counts().sort_index()
created_date_percent = (users_crdate_bdmiss_mwise/users_crdate_monthwise)*100
created_date_percent
```

CREATED_MONTH	count
2021-11	0.061350
2021-12	NaN
2022-01	0.053505
2022-02	NaN
2022-03	NaN
2022-04	NaN
2022-05	NaN
2022-06	NaN
2022-07	NaN
2022-08	NaN
2022-09	NaN
2022-10	NaN
2022-11	NaN
2022-12	NaN
2023-01	NaN
2023-02	NaN
2023-03	NaN
2023-04	0.538600
2023-05	1.488834
2023-06	3.839733
2023-07	11.363636
2023-08	20.493827
2023-09	27.433628
2023-10	32.082552
2023-11	45.463228
2023-12	55.755756
2024-01	48.850575
2024-02	13.152401
2024-03	7.500000
2024-04	8.583333
2024-05	6.020942
2024-06	8.015873
2024-07	13.352970
2024-08	8.854455

Monthly Ratio of created accounts with missing birthdate to the total number of accounts created monthly

Major Data Quality Issue

The problem of missing birthdates started in November 2021 and the problem worsened over the following months. While the records of 2022 were fine, in 2023, starting from 0.5% of created accounts having a missing birthdate, it reached a staggering 55.75% in 2023 December. This means that for more than half of the accounts created in December of 2023, the birth date of the user is missing! Especially, more than 45% of accounts created in November and December of 2023 and January of 2024, have missing birthdate information. This might be either because of the users having an option to skip to record their birthdate while creating their account or the birthdate is not being recorded properly by the UI. This could affect age-based analysis adversely!

Observation - out of all the records where birthdate is missing, 97.98% of them have missing gender information and one reason might be that both the gender and the birthdate details were asked on the same page of the application.

3) Furthermore, the records where the difference between account creation date and the birthdate was greater than 100 were filtered out:

```
[ ] users_age100 = users_filtered[users_filtered['year difference']>100]
    users_age100.head()
```

	ID	CREATED_DATE	BIRTH_DATE	STATE	LANGUAGE	GENDER	year difference
3656	62be5974baa38d1a1f6b6725	2022-07-01 02:18:28+00:00	1903-01-01 05:00:00+00:00	PA	en	male	119.493498
5878	60ac6acc79ed9200a6ebc24a	2021-05-25 03:11:08+00:00	1901-05-25 00:00:00+00:00	UT	NaN	female	120.000000
6499	60a6e4af3369535cb6c4c89d	2021-05-20 22:37:35+00:00	1901-10-13 00:00:00+00:00	CA	NaN	male	119.600274
11146	608601a7b14c7f309d219fc6	2021-04-25 23:56:23+00:00	1904-10-28 19:56:38+00:00	FL	es-419	female	116.490075
14270	59bc4fd3e4b03f96c479b7e6	2017-09-15 22:10:27+00:00	1905-07-28 00:00:00+00:00	CA	en	female	112.134155

```
[ ] users_age100.describe(include= 'all')
```

	ID	CREATED_DATE	BIRTH_DATE	STATE	LANGUAGE	GENDER	year difference
count	56	56	56	50	22	53	56.000000
unique	56	NaN	NaN	25	2	5	NaN
top	62be5974baa38d1a1f6b6725	NaN	NaN	CA	en	female	NaN
freq	1	NaN	NaN	6	20	28	NaN
mean	NaN	2021-02-15 14:13:51.464285696+00:00	1907-12-05 10:18:15.928571392+00:00	NaN	NaN	NaN	113.198103

Filtering out accounts with more than 100-year difference between the created date and birth date

Data Quality Issue – There are 56 accounts with a difference of more than 100-year difference between the created date and birth date. On checking the presence of these accounts in the Transactions dataset, it was confirmed that none of the records existed in the transactions.

```
[ ] #Lets rename the 'user_id' column name to match the one in users table
    transact.rename(columns={'USER_ID' : 'ID'},inplace=True)
    #
```

```
[ ] common_ids = pd.merge(users_age100,transact, on='ID', how='inner')
    common_ids.shape
```

```
(0, 14)
```

Checking for matches in the transaction dataset

4) The records of both state and language column were checked where the birthdates were missing.

```
[ ] #Lets check the breakup of missing values in the 'LANGUAGE' column where the birth_date values are null.
    users_bdate_miss['LANGUAGE'].isna().value_counts()
```

LANGUAGE	count
False	3643
True	32

dtype: int64

```
[ ] #Lets check the breakup of missing values in the 'STATE' column where the birth_date values are null.
    users_bdate_miss['STATE'].isna().value_counts()
```

STATE	count
True	2116
False	1559

dtype: int64

Missing value breakdown for both STATE and LANGUAGE columns where the birthdates are missing

Observations:

- Out of 5892 missing STATE values, 2116 missing values are present where there are missing birthdates values. That is 36% of the total missing values.
- Though there are almost 30,500 missing values in the "LANGUAGE" column, only 32 values are missing where the birth dates are missing. That is roughly 1.04% of the total missing values.

These can be flagged as **Data Quality Issues**.

Focus on the Gender Column

Minor Data Quality Issue – There are 2 categories in the gender column which mean the same. These are "prefer_not_to_say" and "Prefer not to say".

Products Table Data Analysis

Datatype Verification

All the variables' datatypes were checked for legitimacy. All the variables were of object datatype.

```
[ ] product.columns

Index(['CATEGORY_1', 'CATEGORY_2', 'CATEGORY_3', 'CATEGORY_4', 'MANUFACTURER',
       'BRAND', 'BARCODE'],
      dtype='object')
```

Products table's variables' datatypes

The table was then checked for the presence of arrays, lists or dictionaries in any of the columns. They were not present in any of the columns.

```python
def check_column_types(column):
    return any(isinstance(x, (list, dict, np.ndarray)) for x in column)

# Apply the function to each column
columns_with_structures = {col: check_column_types(product_real[col]) for col in product_real.columns}

# Display which columns contain arrays, lists, or dictionaries
print("Columns containing arrays, lists, or dictionaries:")
for col, contains_structure in columns_with_structures.items():
    if contains_structure:
        print(f"{col} contains complex structures")
```

```
Columns containing arrays, lists, or dictionaries:
```

Checking for the presence of other data structures

Duplicates Check

Data Quality Issue – When checked for duplicate records, the product table contained 57 duplicate records. So, they were removed.

```python
product_dupe = product[product.duplicated()]
product_dupe.shape
```

```
(57, 7)
```

```python
#Removing the duplicate records
product_real = product.drop_duplicates()
```

Number of Duplicate records in Products and their removal

The columns were investigated with the help of summary statistics:

```python
product_real.describe(include='all')
```

	CATEGORY_1	CATEGORY_2	CATEGORY_3	CATEGORY_4	MANUFACTURER	BRAND	BARCODE
count	845384	844073	784932	67455	619021	619023	841527
unique	27	121	344	127	4354	8122	841525
top	Health & Wellness	Candy	Confection Candy	Lip Balms	PLACEHOLDER MANUFACTURER	REM BRAND	017000329260
freq	512687	121034	56964	9737	86902	20813	2

Variable Statistics

Observations

- There were missing values in all the columns. The most concerning being the barcode, brand and Manufacturer. The issue of missing values was investigated further in the later sections of this document.
- **Minor Data Quality Issue** – As this table consists of different products information, it is expected for the barcodes which are recorded to be unique. But there are 2 barcodes which are repeated. More has been discussed in Part 2 of this document.

Missing Values Check

```
[ ] product_real.isna().sum()
```

	0
CATEGORY_1	111
CATEGORY_2	1422
CATEGORY_3	60563
CATEGORY_4	778040
MANUFACTURER	226474
BRAND	226472
BARCODE	3968

dtype: int64

```
[ ] #Lets check for percentages
    product_isna = product_real.isna().sum()
    product_isna/product_real.shape[0]
```

	0
CATEGORY_1	0.000131
CATEGORY_2	0.001682
CATEGORY_3	0.071630
CATEGORY_4	0.920218
MANUFACTURER	0.267860
BRAND	0.267857
BARCODE	0.004693

dtype: float64

Missing Values in Products

Data Quality Issues:

- 92.02% percent of the category 4 were missing values.
- There were almost equal number of missing values in both Manufacturer column and the brand column which was 26.78%.
- The most concerning observation was that there were several products whose barcodes were missing. Further analysis was done to understand the implications.

Miscellaneous Checks

Focus on CATEGORY Columns

Observations:

1) When all the unique categories were looked at in CATEGORY_1 column, one category that looked peculiar was "Needs Review". So that category was analyzed further. It was found that all there were 547 products whose category_1 was "Needs Review" and most of them belonged to Beverages and Snacks. Secondly, all the other categories were missing

for the ‘Needs Review’ category while the brand, barcode and manufacturer information was available.

```
[64] #Checking the number of rows where 'CATEGORY_1' is 'Needs Review'
     product_real[product_real['CATEGORY_1']=='Needs Review'].shape
```

```
(547, 7)
```

```
[65] product_real[product_real['CATEGORY_1']=='Needs Review'].isna().sum()
```

	0
CATEGORY_1	0
CATEGORY_2	547
CATEGORY_3	547
CATEGORY_4	547
MANUFACTURER	0
BRAND	0
BARCODE	0

```
[68] products_cat1_NR['MANUFACTURER'].value_counts()
```

MANUFACTURER	count
PEPSICO	186
KEURIG DR PEPPER	145
THE COCA-COLA COMPANY	139
SARGENTO	27
MARS WRIGLEY	23
MOLSONCOORS	22
GALLO	3
GENERAL MILLS	2

A quick glance at the ‘Needs Review’ Category

2) Another interesting observation here is that most of the brands whose CATEGORY_1 is not filled belong to either beverages or snacks.

```
[71] product_cat1_na['BRAND'].value_counts()
```

BRAND	count
COCA-COLA	31
CHEETOS	27
COORS LIGHT	20
POLAR	15
SNICKERS	5
PEPSI	5
FRITO-LAY	3
BUBLY SPARKLING WATER	2
CHEERIOS	1
SARGENTO	1

```
[70] product_cat1_na.isna().sum()
```

	0
CATEGORY_1	111
CATEGORY_2	111
CATEGORY_3	111
CATEGORY_4	111
MANUFACTURER	0
BRAND	0
BARCODE	0

A brief glance at the brands and missing values where 'Category_1' is missing

3) Another observation is that all the other categories are missing where 'CATEGORY_1' is missing. Furthermore, all CATEGORY_3 and CATEGORY_4 values are missing where there are missing CATEGORY_2 values. It was observed that category columns were branched. For example, CATEGROY_2 items are a subset of CATEGORY 1 and CATEGORY_3 is a subset of CATEGORY_2. That may be the reason why there are a lot of missing CATEGORY_4 values. The branching can be observed below:

```
product_real.groupby(['CATEGORY_1','CATEGORY_2'])['CATEGORY_3'].nunique().sort_values(ascending=False)
```

CATEGORY_1	CATEGORY_2	CATEGORY_3
Health & Wellness	Skin Care	16
	Medicines & Treatments	14
	Bath & Body	12
	Hair Care	11
Snacks	Dips & Salsa	10
	...	...
	Cookies	0
Media	Books	0

Branching of Categories

```
product_cat3_na.isna().sum()
```

	0
CATEGORY_1	111
CATEGORY_2	1422
CATEGORY_3	60563
CATEGORY_4	60563
MANUFACTURER	20314
BRAND	20314
BARCODE	1849

dtype: int64

Missing values of Categories 3 and 4

4) 59.14% of the missing "CATEGORY_4" values are from the health and Awareness category. Secondly, 94.48% of the missing barcode values have missing category_4 values.

```
[84] heal_well = cat1_break.iloc[0]/cat1_break.sum()
     heal_well
```

```
0.5914164403178183
```

```
[85] mis_bar_rat = cat4_all_na.iloc[6]
     mis_bar_rat/product_real['BARCODE'].isna().sum()
```

```
0.9448084677419355
```

Percentage of missing barcode values in the records with missing category_4 values

Focus on Brand and Manufacturer Columns

Observations: All the missing brand and manufacturer details belong to two categories - Health & Wellness and Snacks.

```
[91] product_brand_na['CATEGORY_1'].value_counts()
```

CATEGORY_1	count
Health & Wellness	128401
Snacks	98071

dtype: int64

```
[92] product_manu_na['CATEGORY_1'].value_counts()
```

CATEGORY_1	count
Health & Wellness	128403
Snacks	98071

Category_1 breakdown of brands and manufacturers

Focus on the BARCODE column

Data Quality Issues:

- There were 73307 non-null records (8.71% of all the non-null values) where the barcodes did not follow the 12-digit format.

```
[334] product_real['barcode_is_12_digit'] = product_real['BARCODE'].map(lambda x: len(str(x)) == 12)
      print(product_real['barcode_is_12_digit'].value_counts())
```

```
barcode_is_12_digit
True     768220
False     77275
Name: count, dtype: int64
<ipython-input-334-da4ed5253986>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
  product_real['barcode_is_12_digit'] = product_real['BARCODE'].map(lambda x: len(str(x)) == 12)
```

```
[398] false_barcodes = product_real[(product_real['barcode_is_12_digit'] == False) & (product_real['BARCODE'].notna())]
      false_barcodes.shape
```

```
(73307, 8)
```

Presence of invalid barcodes in Products data

- It was noticed that almost 4000 rows were present where the barcode values were missing. On further analysis, it was found that most of the missing barcodes were in the 'Health and Wellness' category with the highest in skincare followed by the ‘Snacks' category with the highest in candy.

```
[96] #number of records where the barcode is missing.
     product_real['BARCODE'].isna().sum()
```

```
3968
```

Number of products with missing barcodes

```
#Grouping the records firstly according to category_1 followed by according to category_2 where the barcode is null.
brand_count = product_bar_na.groupby(['CATEGORY_1','CATEGORY_2'])['BRAND'].count().sort_values(ascending=False)
brand_count
```

		BRAND
CATEGORY_1	CATEGORY_2	
Health & Wellness	Skin Care	378
	Medicines & Treatments	350
	Bath & Body	325
	Hair Care	298
Snacks	Candy	257
	Snack Bars	149
	Chips	130
	Crackers	105

Category-wise breakdown of columns with missing barcodes

Transactions Table Data Analysis

Datatype Verification

All the variables' datatypes were checked for legitimacy. All the variables were of object datatype.

```
transact.dtypes
```

	0
RECEIPT_ID	object
PURCHASE_DATE	object
SCAN_DATE	object
STORE_NAME	object
USER_ID	object
BARCODE	object
FINAL_QUANTITY	object
FINAL_SALE	object

```
[346] transact['PURCHASE_DATE'] = pd.to_datetime(transact['PURCHASE_DATE'])
      transact['SCAN_DATE'] = pd.to_datetime(transact['SCAN_DATE'])


      transact['SCAN_DATE'] = transact['SCAN_DATE'].dt.tz_convert('UTC')

      transact.dtypes
```

	0
RECEIPT_ID	object
PURCHASE_DATE	datetime64[ns]
SCAN_DATE	datetime64[ns, UTC]
STORE_NAME	object
USER_ID	object
BARCODE	object
FINAL_QUANTITY	object
FINAL_SALE	object

Datatypes before and after date-time conversion

The "PURCHASE_DATE" and "SCAN_DATE" columns were converted into appropriate date-time objects keeping in mind the time stamp and the time zone information.

The table was then checked for the presence of arrays, lists or dictionaries in any of the columns. They were not present in any of the columns.

```
def check_column_datastructures(column):
    return any(isinstance(x, (list, dict, np.ndarray)) for x in column)

# Applying the function to each column
columns_with_structures = {col: check_column_datastructures(transact[col]) for col in transact.columns}

# Displaying which columns contain arrays, lists, or dictionaries
print("Columns containing arrays, lists, or dictionaries:")
for col, contains_structure in columns_with_structures.items():
    if contains_structure:
        print(f"{col} contains complex structures")
```

```
Columns containing arrays, lists, or dictionaries:
```

Checking for the presence of other data structures

Duplicates Check

Data Quality Issue – When checked for duplicate records, the transactions table contained 171 duplicate records. So, they were removed.

```
[349] transact_dupe = transact.duplicated().sum()
      print("Total number of duplicate values in the transaction dataset:", transact_dupe)
```

```
Total number of duplicate values in the transaction dataset: 171
```

```
[350] #Removing the duplicate values
      transact.drop_duplicates(inplace=True)
```

Checking for duplicate records and their removal

The columns were investigated further with the help of summary statistics:

```
transact.describe(include='all')
```

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE
count	49829	49829	49829	49829	49829	44094	49829	49829
unique	24440	89	24440	954	17694	11028	87	1435
top	0fb89572-c817-47e2-bd11-6f467baacbb2	2024-07-03	2024-06-25 17:51:10.487 Z	WALMART	64e62de5ca929250373e6cf5	078742223759	1.00	
freq	6	772	6	21249	22	181	35536	12486

```
[ ] transact.describe(include='all')
```

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE
count	49829	49829	49829	49829	49829	44094	49829	49829
unique	24440	NaN	NaN	954	17694	11028	87	1435
top	0fb89572-c817-47e2-bd11-6f467baacbb2	NaN	NaN	WALMART	64e62de5ca929250373e6cf5	078742223759	1.00	
freq	6	NaN	NaN	21249	22	181	35536	12486
mean	NaN	2024-07-24 08:05:52.565775104	2024-07-26 22:53:49.655994368+00:00	NaN	NaN	NaN	NaN	NaN
min	NaN	2024-06-12 00:00:00	2024-06-12 06:36:34.910000+00:00	NaN	NaN	NaN	NaN	NaN
25%	NaN	2024-07-03 00:00:00	2024-07-05 10:44:40.377999872+00:00	NaN	NaN	NaN	NaN	NaN
50%	NaN	2024-07-23 00:00:00	2024-07-26 10:57:39.848999936+00:00	NaN	NaN	NaN	NaN	NaN
75%	NaN	2024-08-15 00:00:00	2024-08-17 15:39:25.552999936+00:00	NaN	NaN	NaN	NaN	NaN

Summary Statistics before and after date-time conversion

Observations – There were many issues that were observed from the above tables

- Though there were 49,829 recorded observations, very few values were unique in each of the columns. A deeper dive into the matter later exposed some fallacies. Especially the ‘SCAN_DATE’ column, because it contained even the timestamp and if were 49,829 recorded observations and only 24440 were unique values, there was a high chance that a lot of scanned items had been recorded more than once!
- Just the barcode column had missing values with a lot of recurring values. This could be an issue when merging the tables for analyses.

- The data in the ‘FINAL_QUANTITY' column and the 'FINAL_SALE' column seemed ambiguous as seen in the image below. The ‘FINAL_SALE’ column had a lot of rows with one space character (“ ”) and the corresponding ‘FINAL_QUANTITY’ rows had a value. Then where there was a ‘FINAL_SALE’ value, there the ‘FINAL_QUANTITY’ was zero. Further analysis was performed to understand the columns better. Additionally, both the columns should be in integer format. So, they were converted into a numeric datatype after the analysis.

```
transact.head()
```

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE
0	0000d256-4041-4a3e-adc4-5623fb6e0c99	2024-08-21	2024-08-21 14:19:06.539 Z	WALMART	63b73a7f3d310dceeabd4758	015300014978	1.00	
1	0001455d-7a92-4a7b-a1d2-c747af1c8fd3	2024-07-20	2024-07-20 09:50:24.206 Z	ALDI	62c08877baa38d1a1f6c211a	NaN	zero	1.49
2	00017e0a-7851-42fb-bfab-0baa96e23586	2024-08-18	2024-08-19 15:38:56.813 Z	WALMART	60842f207ac8b7729e472020	078742229751	1.00	
3	000239aa-3478-453d-801e-66a82e39c8af	2024-06-18	2024-06-19 11:03:37.468 Z	FOOD LION	63fcd7cea4f8442c3386b589	783399746536	zero	3.49
4	00026b4c-dfe8-49dd-b026-4c2f0fd5c6a1	2024-07-04	2024-07-05 15:56:43.549 Z	RANDALLS	6193231ae9b3d75037b0f928	047900501183	1.00	

Glimpse of the transactions data

Missing Values Check

Data Quality Issue – Only the ‘BARCODE’ column had missing values. That is 13.06% of the total ‘BARCODE’ values present. It was also a very concerning observation because ‘BARCODE’ column acted as a key to connect both the products and the transactions table.

```
[ ] transact.isna().sum()
```

	0
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

Missing Values in the Transactions Table

Check for Errors

Chronological Errors – The presence of records where the scan date is earlier than the purchase date was checked.

```
[ ] # Comparing Purchasing dates and Scanning dates and storing it in a new variable
    transact['PD<SD'] = transact['PURCHASE_DATE'] <= transact['SCAN_DATE'].dt.date

    transact['PD<SD'].value_counts()
```

PD<SD	count
True	49735
False	94

Check for chronological Errors

Data Quality Issue

There were 94 records where the scanned date is before the purchased date. This meant that the users scanned the receipt before they made a purchase.

Miscellaneous Checks

Firstly, the focus was on three columns – ‘BARCODE’, ’FINAL_QUANTITY’ and ‘FINAL_SALE’

Focus on 'FINAL_SALE' and 'FINAL_QUANTITY' columns

To understand the presence of missing values, firstly, the unique values in both the columns were looked at for a better understanding.

```
transact['FINAL_QUANTITY'].value_counts()
```

FINAL_QUANTITY	count
1.00	35536
zero	12491
2.00	1285
3.00	184
4.00	139
...	...
6.22	1
1.22	1
1.23	1

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

FINAL_SALE	count
	12486
1.25	1313
1.00	732
2.99	587
1.99	581
...	...
16.47	1
10.92	1
61.30	1

Glimpse at the unique values in both the columns

Then the whole dataset was split into two data frames, one where the ‘FINAL_SALE’ columns were blank spaces and one where they had values. The two data frames were compared over the ‘BARCODE’ columns, and they had 6805 barcodes in common. After that, when the number of observations with the blank final sales with unique barcodes was checked even that turned out to be 6805!

```python
#Finding common barcodes in two dataframes, one with the blanks and one without the blanks.
common_barcodes = set(transact_sale_NB['BARCODE']).intersection(set(transact_sale_blank['BARCODE']))
combar_num = len(common_barcodes)
combar_num
```

```
6805
```

```python
#Creating an array with unique barcode values where the FINAL_SALE column is blank
x = transact_sale_blank['BARCODE'].unique()
len(x)
```

```
6805
```

```python
blank_bar_unique = set(transact_sale_blank['BARCODE'].unique())
# Finding additional unique barcodes that may be present in the dataset with blank 'FINAL_SALE' values.
missing_barcode =  blank_bar_unique - common_barcodes
missing_barcode
```

```
set()
```

‘FINAL_SALE’ column analysis workflow

Observations:

❖ From the above results it was determined that there were no additional unique barcodes in the data frame with a blank "FINAL_SALE" column that are not present in the data frame with the rest of the values.

So, after looking at the above results it was thought initially that all the blank values could be replaced with corresponding final sale values where the barcodes matched.

But the main issue came to light after trying to find out the reason for the blank spaces.

A few random barcodes were chosen from the blank ‘FINAL_SALE’ data frame and all the instances of those barcodes were looked at in the original unsplit data. They were grouped by the other columns and below were the results:

```python
filter_bycode1.groupby(['RECEIPT_ID','USER_ID','STORE_NAME','SCAN_DATE','PURCHASE_DATE','FINAL_SALE','FINAL_QUANTITY'])['BARCODE'].apply(list)
```

RECEIPT_ID	USER_ID	STORE_NAME	SCAN_DATE	PURCHASE_DATE	FINAL_SALE	FINAL_QUANTITY	BARCODE
1542749e-101f-430d-8b33-fd8a200fe1de	5ddefc35ce77bc78062d42fb	PUBLIX	2024-07-21 15:58:37.476000+00:00	2024-07-20		1.00	[511111045496]
					0.99	1.00	[511111045496]
17f0820c-c480-4482-80e1-796670646674	5cad608ae03e4b18c625d2da	WINN-DIXIE	2024-06-23 12:29:25.252000+00:00	2024-06-20	4.99	1.00	[511111045496]
						zero	[511111045496]
294a2785-22be-4a4f-b8a9-f387991ad0fa	5ecb23aea0134313d2d8a3dd	KROGER	2024-06-23 08:43:56.378000+00:00	2024-06-17		1.00	[511111045496]
					2.50	1.00	[511111045496]
486c09c9-c5ce-4d4f-b004-024344662d13	666f1d84465f309038ab3a6c	PRICE CHOPPER	2024-09-02 13:54:37.016000+00:00	2024-09-02	1.00	1.00	[511111045496]
						zero	[511111045496]
67e06a61-0eb4-4138-bdf0-c6e46c9a4526	63dc6c96dcb50fbd3083edb1	SHOP RITE	2024-06-27 12:31:18.008000+00:00	2024-06-20	1.27	1.00	[511111045496]
						zero	[511111045496]
82fb7d5d-3bcd-4aab-a079-45447bf4b4a5	6621f37cc41e9f27acd82170	STOP & SHOP	2024-06-16 18:20:19.585000+00:00	2024-06-15		1.00	[511111045496]
					0.57	1.00	[511111045496]

```python
filter_bycode3.groupby(['RECEIPT_ID','USER_ID','STORE_NAME','SCAN_DATE','PURCHASE_DATE',
                        'FINAL_SALE','FINAL_QUANTITY'])['BARCODE'].apply(list).head(20)
```

RECEIPT_ID	USER_ID	STORE_NAME	SCAN_DATE	PURCHASE_DATE	FINAL_SALE	FINAL_QUANTITY	BARCODE
006490a1-65b3-4ec7-8130-08767846432a	617cb0e1e5e5187841b701dc	PUBLIX	2024-09-02 12:18:56.915000+00:00	2024-09-02		1.00	[311111224057]
					7.99	1.00	[311111224057]
01cf54de-0f0b-4a89-aa35-227ba103ed59	62ec127f4e73e2db30eaadae	PLAVERS CAFE LIDA	2024-08-15 07:58:59.635000+00:00	2024-08-10	6.00	2.00	[311111224057]
						zero	[311111224057]
08e5e4a4-f72e-44ed-984e-28d66d565410	60a31588671b6805df74cd1e	ALDI	2024-08-11 18:18:03.811000+00:00	2024-08-09		1.00	[311111224057]
					2.68	1.00	[311111224057]
097724fe-0b26-48b9-acd2-706f566cb5a7	637e59e9d6f2a49c49941f38	RALPH'S CAQUAS	2024-07-14 08:37:49.898000+00:00	2024-07-14	1.75	1.00	[311111224057]
						zero	[311111224057]
0a105723-7aab-46a1-9016-97177a5a7314	635d382e68d1ed9fd3b30e09	SUPERMAX	2024-09-05 22:13:41.844000+00:00	2024-09-05	1.42	1.00	[311111224057]
						zero	[311111224057]

```python
#Checking the FINAL_QUANTITY and FINAL_COLUMNS where the barcode is identical in 3 instances.
filter_bycode3[filter_bycode3['STORE_NAME']=='PUEBLO']
```

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE	PD<SD
11814	7a09e2bc-53e0-411e-a72f-9ff3b636bf46	2024-08-25	2024-08-25 14:52:08.512000+00:00	PUEBLO	665b575ce04f743a096ca7ea	311111224057	1.00		True
11815	7a09e2bc-53e0-411e-a72f-9ff3b636bf46	2024-08-25	2024-08-25 14:52:08.512000+00:00	PUEBLO	665b575ce04f743a096ca7ea	311111224057	zero	0.79	True
42493	7a09e2bc-53e0-411e-a72f-9ff3b636bf46	2024-08-25	2024-08-25 14:52:08.512000+00:00	PUEBLO	665b575ce04f743a096ca7ea	311111224057	1.00	0.79	True

Random Barcodes grouped by other columns

It was then verified if every barcode got repeated more than once.

```python
#Checking if each unique value in the "BARCODE" column has appeared only once
unique_barcode = transact['BARCODE'].value_counts()
single_barcode_occurrence = unique_barcode[unique_barcode == 1].index.tolist()
print("Values that appear only once in BARCODE column:", single_barcode_occurrence)
```

```
Values that appear only once in BARCODE column: []
```

Verification for single occurrence of a barcode in the whole data

Major Data Quality Issue

The earlier decision of replacing the blank 'FINAL_SALE' values with their corresponding counterpart would have proved highly detrimental for the analysis. On further inspection, it was found that there was a presence of duplicate entries! It seems that every item got scanned more than once at the same time and date for every user! An interesting observation here was, wherever the sale value and the quantity were recorded correctly for an item at a particular time, then the rest of the duplicate records that got recorded at the same scanned time had the following changes: Either the sale value was left blank, or the sale quantity was mentioned as "zero".

Solution to correct this data discrepancy to go forward with the asked business insights:

The rows where "zero" was written in the "FINAL_QUANTITY" and the rows where there was a blank "FINAL_SALE" column were removed.

```
transact_filtered = transact[~((transact['FINAL_QUANTITY'] == 'zero') | (transact['FINAL_SALE'] == ' '))]
transact_filtered.head()
```

Filtering out the extra rows

After that, the 'FINAL_SALE' and 'FINAL_QUANTITY' were converted into numeric datatype.

```
[186] transact_filtered.loc[:, ['FINAL_SALE', 'FINAL_QUANTITY']] = transact_filtered[['FINAL_SALE', 'FINAL_QUANTITY']].apply(pd.to_numeric,
                                                                                                                              errors='coerce')
```

Converting the columns in numeric datatype

Focus on the BARCODE column

Data Quality Issues

There are 2856 null barcode values and an additional 63 invalid barcodes in the transactions database that did not follow the 12-digit format. This might be because of the application not scanning the barcodes properly or because of an improper use by the customer. On checking for an overlap between these records and the product records, it was found that 56 invalid barcodes were present in both the tables.

```
transact_filtered.loc[:,'barcode_is_12_digit'] = transact_filtered['BARCODE'].map(lambda x: len(str(x)) == 12)
print(transact_filtered.loc[:,'barcode_is_12_digit'].value_counts())
```

```
barcode_is_12_digit
True     21933
False     2919
Name: count, dtype: int64
<ipython-input-188-bb2598c95050>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  transact_filtered.loc[:,'barcode_is_12_digit'] = transact_filtered['BARCODE'].map(lambda x: len(str(x)) == 12)
```

```
invalid_transact_barcodes = transact_filtered[(transact_filtered['BARCODE'].notna()) & (transact_filtered['barcode_is_12_digit']==False)]
invalid_transact_barcodes.shape
```

```
(63, 10)
```

```
common_invalid_barcodes = pd.merge(invalid_transact_barcodes[invalid_transact_barcodes['BARCODE'].notna()],
                                   false_barcodes[false_barcodes['BARCODE'].notna()], on='BARCODE', how='inner')
common_invalid_barcodes.shape
```

```
(56, 17)
```

Invalid Barcodes and the number of common invalid barcodes

Issues found after merging the finalized data sets

Users and Transactions

It was found that there are only 130 common records (might contain recurring user ids) when Users and Transactions were joined using an inner join.

```
394] users_transact_inner = pd.merge(transact_final, users_final, left_on='USER_ID', right_on='ID', how='inner')
     users_transact_inner.shape

(130, 14)
```

Number of common records between Users and Transactions

Data Quality Issue

It was observed that seen that there were plenty of users who made purchases, but their records were missing from the users table. That would adversely affect the business analyses which could be drawn from the users data based on the transactions made because, a lot of valid transactions would go unaccounted for because of the missing users data!

```
users_transact_left = pd.merge(transact_final, users_final, left_on='USER_ID', right_on='ID', how='left')
users_transact_left.shape

(24852, 14)
```

Left Join of Users and Transactions

Products and Transactions

Data Quality Issue

After performing an inner join it was found that more than half of the transactions have been missed after the join. This means that there were a lot of products that were bought that didn't have records in the products table.

```
product_transact_inner = pd.merge(transact_final[transact_final['BARCODE'].notna()],
                                  product_final2[product_final2['BARCODE'].notna()], left_on='BARCODE', right_on='BARCODE', how='inner')
product_transact_inner.shape

(12363, 14)
```

Inner Join of Products and Transactions

Part 2 – Difficulties in Interpretation

Duplication of CREATED_DATE in the Users Table

While analyzing the 'BIRTH_DATE' and the 'CREATED_DATE' columns in the Users table, it was observed that several of the values in the 'CREATED_DATE' were recurring.

```
users['CREATED_DATE'].value_counts()
```

CREATED_DATE	count
2021-08-06 14:51:13+00:00	2
2019-08-28 02:21:44+00:00	2
2024-04-11 02:56:41+00:00	2
2024-03-11 17:03:02+00:00	2
2024-02-25 20:43:59+00:00	2
...	...
2024-08-25 03:30:41+00:00	1
2021-03-12 13:08:04+00:00	1

```
users_crdt_twice = users[users['CREATED_DATE'].map(users['CREATED_DATE'].value_counts()) == 2]
users_crdt_twice.shape
```

```
(116, 6)
```

Recurring values in the 'CREATED_DATE' column of the Users table

The rows where the repetition happened were analyzed further to figure out a reason for this anomaly.

```
users_crdt_twice = users[users['CREATED_DATE'].map(users['CREATED_DATE'].value_counts()) == 2]
users_crdt_twice.shape
```

```
(116, 6)
```

```
users_crdt_twice.groupby(['CREATED_DATE','STATE','BIRTH_DATE'])['ID'].apply(list).head(15)
```

CREATED_DATE	STATE	BIRTH_DATE	ID
2019-08-25 02:02:11+00:00	AR	1975-08-12 05:00:00+00:00	[5d61ec22fe79a7584c9b573c]
	VA	1972-06-07 00:00:00+00:00	[5d61ec231ddc4058bd9a6233]
2019-08-28 02:21:44+00:00	MS	1990-10-15 05:00:00+00:00	[5d65e537d09cf73c7b6a1585]
	NH	1975-05-15 00:00:00+00:00	[5d65e5381ddc403b76f4dc72]
2020-01-08 01:42:14+00:00	CT	1979-02-07 05:00:00+00:00	[5e153376128c2c120e86e57f]
2020-02-16 17:04:11+00:00	AR	1990-12-03 06:00:00+00:00	[5e49760aacedab1335b03b89]
	GA	1972-04-15 05:00:00+00:00	[5e49760b164813133fc63ac2]
2020-04-29 02:24:48+00:00	CA	1992-10-12 00:00:00+00:00	[5ea8e56f2244e629eacf9b09]
	UT	1978-06-07 06:00:00+00:00	[5ea8e56f2244e629eacf9b07]
2020-05-04 01:01:40+00:00	CA	1979-03-08 08:00:00+00:00	[5eaf6974cefff2142582eab6]
	MI	1999-03-04 05:00:00+00:00	[5eaf6973787646145f3a1ccf]

The values were grouped according to the other columns to find a pattern, such as same state, or a matching birthdate or a pattern in the user id. But nothing substantial was discovered. An effort was made to find a pattern in the time periods when this occurred, but even this didn't result in anything fruitful.

```
users_crdt_twice_og = users_crdt_twice.drop_duplicates(subset=['CREATED_DATE'], keep='first')
users_crdt_twice_og.loc[:,'CREATED_MONTH'] = users_crdt_twice_og['CREATED_DATE'].dt.to_period('M')
users_crdt_twice_og['CREATED_MONTH'].value_counts().sort_index()
```

CREATED_MONTH	count
2019-08	2
2020-01	1
2020-02	1
2020-04	1
2020-05	1
2020-08	1
2020-10	2
2020-12	2
2021-01	2
2021-06	1
2021-08	1
2021-11	1
2021-12	1
2022-02	5
2022-03	1
2022-06	3
2022-07	4
2022-09	3
2022-10	2
2022-11	2
2022-12	1
2023-01	2
2023-02	1
2023-04	1
2023-05	1
2023-06	1
2023-10	1
2023-11	1
2024-02	1
2024-03	1
2024-04	2
2024-06	1
2024-07	3
2024-08	1
2024-09	3

Months when the repetitions happened

This is a **Data Quality Issue** unless there is an underlying reason for this repetition. But the reason remains inconclusive.

Values in the 'GENDER' column of the Users Table

There were 2 categories listed in the 'GENDER' column – 'Prefer not to say' and 'prefer_not_to_say' – which mean one and the same. This might be a bug, but it was flagged as a **minor data quality issue**.

Repeated Barcodes in the Products Table

Though the products table consists of different products, there were 2 barcodes which appeared twice in the records.

```
[ ] product_real['BARCODE'].value_counts()
```

BARCODE	count
017000329260	2
052336919068	2
796494407820	1

```
[ ] product_real[(product_real['BARCODE']=='017000329260') | (product_real['BARCODE']=='052336919068')]
```

	CATEGORY_1	CATEGORY_2	CATEGORY_3	CATEGORY_4	MANUFACTURER	BRAND	BARCODE
28421	Health & Wellness	Hair Care	Hair Color	NaN	HENKEL	SCHWARZKOPF	052336919068
213340	Health & Wellness	Hair Care	Hair Color	NaN	HENKEL	SCHWARZKOPF	017000329260
304021	Health & Wellness	Hair Care	Hair Color	NaN	HENKEL	GÖT2B	017000329260
709607	Health & Wellness	Hair Care	Hair Color	NaN	HENKEL	GÖT2B	052336919068

Recurring barcode details in products table

While the brand is different, the barcode was the same for both the products and all the four products belong to the same manufacturer - Henkel. The reason seemed difficult to comprehend. When checked for the products' presence in the transaction table, it turned out to be negative.

```
[ ] transact[(transact['BARCODE']=='017000329260') | (transact['BARCODE']=='052336919068')]
```

RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	ID	BARCODE	FINAL_QUANTITY	FINAL_SALE

Checking for the presence of the products with recurring barcodes in Transactions

Two or more identical barcodes scanned on the same day but with different final_sale values

While reviewing the transactions table after the repeated transactions were filtered out, it was noticed that there were still repeated receipt ids. On further digging, it was observed that there were records with the same barcode, receipt id, purchase date, scan date and quantity but with a different price.

```
#Viewing some records where the recipt_id and the barcodes are identitical
receipt_view = transact_filtered[(transact_filtered['RECEIPT_ID']=='19f725b1-e708-4060-b159-acc7674e5405')|
                    (transact_filtered['RECEIPT_ID']=='431fe612-ed55-470e-939c-043ad31f33f3')]
receipt_view
```

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE	PD<SD
31668	19f725b1-e708-4060-b159-acc7674e5405	2024-08-23	2024-09-03 09:47:23.950000+00:00	SHOP RITE	619c29ca06570b5913a151fb	311111921628	1.0	2.99	True
31675	19f725b1-e708-4060-b159-acc7674e5405	2024-08-23	2024-09-03 09:47:23.950000+00:00	SHOP RITE	619c29ca06570b5913a151fb	311111921628	1.0	3.96	True
43005	431fe612-ed55-470e-939c-043ad31f33f3	2024-09-07	2024-09-07 16:39:01.409000+00:00	DOLLAR GENERAL STORE	5e038cebcb322c11de193bb7	012000504051	1.0	6.25	True
43006	431fe612-ed55-470e-939c-043ad31f33f3	2024-09-07	2024-09-07 16:39:01.409000+00:00	DOLLAR GENERAL STORE	5e038cebcb322c11de193bb7	012000504051	1.0	5.25	True

Records of the transactions table with identical records

First Theory - It might be because a user bought two or more identical products but somehow the quantity got entered wrong.
Second Theory - Another hypothesis is that the product might have been on an offer - something like, buy 1 for 5 dollars and get the second pack for 2.5 dollars.
The second one made more sense.
But it was flagged as a minor data quality issue. More context is needed for better understanding.

Other miscellaneous difficult interpretations

1) Needs Review category of User's table needs more emphasis and explanation.
2) A lot of CATEGORY_3 and CATEGORY_4 values are missing, and a lot of categories seemed redundant.