FIRST EXERCISE

Two questions were asked in this exercise

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

Part 1 - Data Quality Exploration

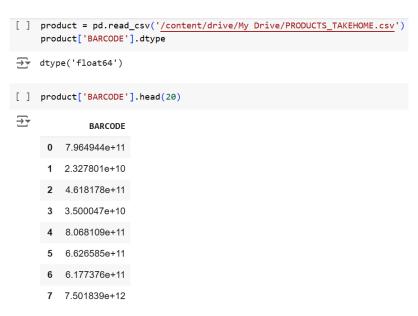
The unstructured data that was provided consists of three csv files namely – Products, Users and Transactions. To assess the quality of the data provided, the following process was implemented for every table.

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:



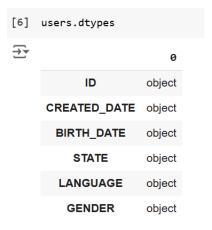
Barcodes loaded as float datatype with no leading zeros

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

Users Table Data Analysis

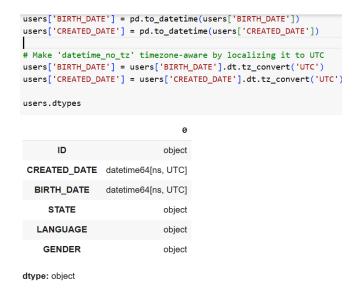
Datatype Verification

All the variables' datatypes were checked for legitimacy.



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.



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.

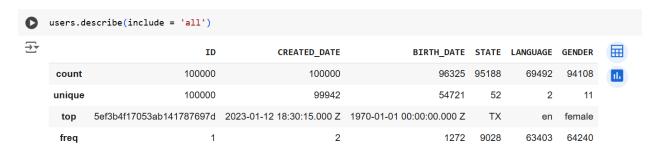
```
def check_column_datastructures(column):
    return any(isinstance(x, (list, dict, np.ndarray)) for x in column)

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

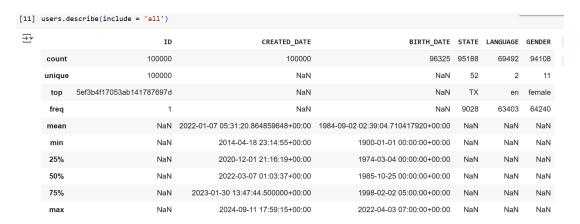
# Displaying which columns contain arrays, lists, or dictionaries
print("Columns containing arrays, lists, or dictionaries:")
for col, contains_structure in columns_with_structures.items():
    if contains_structure:
        print(f"{col} contains complex structures")
Columns containing arrays, lists,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

Checking for the presence of other data structures

The columns were investigated with the help of summary statistics:



Variable Statistics before date-time conversion

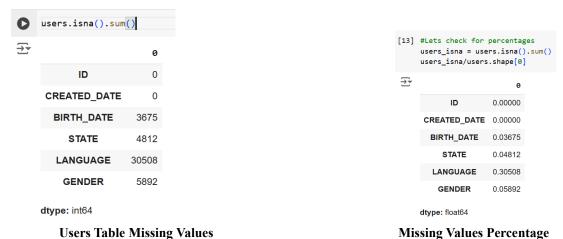


Variable Statistics after date-time conversion

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



Oscis Table Missing Var

wissing values i ciccitage

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.



Records with created date earlier than the birth date

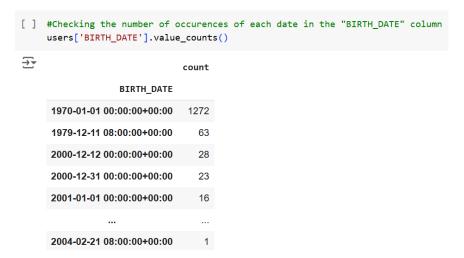
Minor Data Quality Issue – There is one record where the chronology is illogical. So, it was removed.

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.



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:

u: u: c:	sers_crda sers_crda reated_da	te_monthwi: te_bdmiss_r	se = mwise	ring the common users['CREATED e = users_bdate_ users_crdate_bdm	_MONTH'].val _miss['CREA	lue_counts().so TED_MONTH'].val	ue_counts().sc
}		c	ount				
(CREATED_M	ONTH					
	2021-11	0.06	1350				
	2021-12	2	NaN				
	2022-01	0.05	3505				
	2022-02	2	NaN				
	2022-03		NaN				
	2022-04		NaN NaN				
	2022-06		NaN				
2022		NaN					
2022	2_08	NaN				2023-08	20.493827
						2023-09	27.433628
2022		NaN				2023-10	32.082552
2022	2-10	NaN				2023-11	45.463228
2022	2-11	NaN				2023-12	55.755756
2022	2-12	NaN				2024-01	48.850575
2023	3-01	NaN				2024-02	13.152401
2023	3-02	NaN				2024-03	7.500000
2023	3-03	NaN				2024-04	8.583333
2023	3-04	0.538600				2024-05	6.020942
2023	3-05	1.488834				2024-06	8.015873
2023	3-06	3.839733				2024-07	13.352970
2023		11.363636				2024-08	8.854455
2023	J-01	11.505050					2.2300

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:



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.

[]			e breakup of missing values in the 'LANGUAGE' column where the birth_date values are nullss['LANGUAGE'].isna().value_counts()
₹		cour	nt
	LANGUA	GE	
	False	364	43
	True	3	32
	dtype: in	t64	
[]			ne breakup of missing values in the 'STATE' column where the birth_date values are null. ss['STATE'].isna().value_counts()
_ →		count	
	STATE		
	True	2116	
	False	1559	
	dtype: in	t64	

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.

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.

```
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:
```

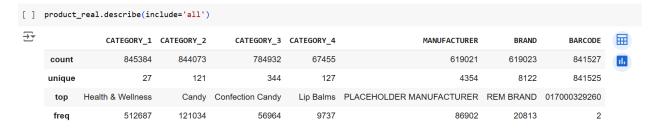
Checking for the presence of other data structures

Duplicates Check

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

Number of Duplicate records in Products and their removal

The columns were investigated with the help of summary statistics:

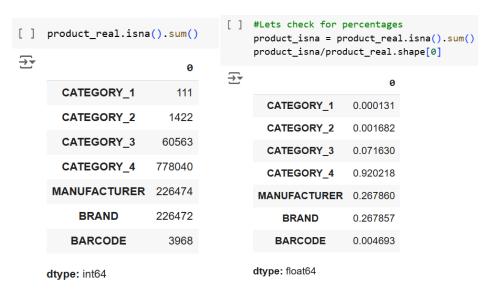


Variable Statistics

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



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.

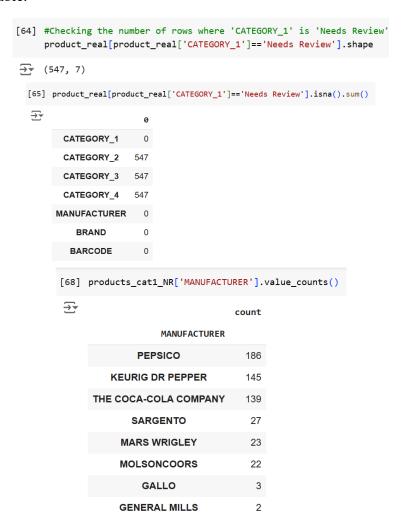
Miscellaneous Checks

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.



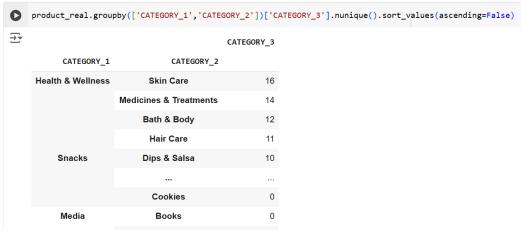
A quick glance at the 'Needs Review' Category

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

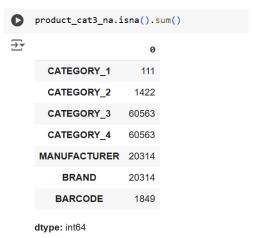


A brief glance at the brands and missing values where 'Category_1' is missing

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:



Branching of Categories



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.

```
(s | S4| heal_well = cat1_break.iloc[0]/cat1_break.sum()
    heal_well

    0.5914164403178183

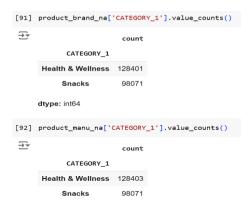
(s | S5| mis_bar_rat = cat4_all_na.iloc[6]
    mis_bar_rat/product_real['BARCODE'].isna().sum()

    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.



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: <a href="https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni">https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni</a>
        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

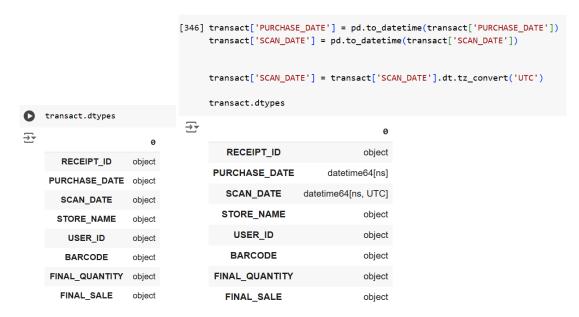


Category-wise breakdown of columns with missing barcodes

Transactions Table Data Analysis

Datatype Verification

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



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:
```

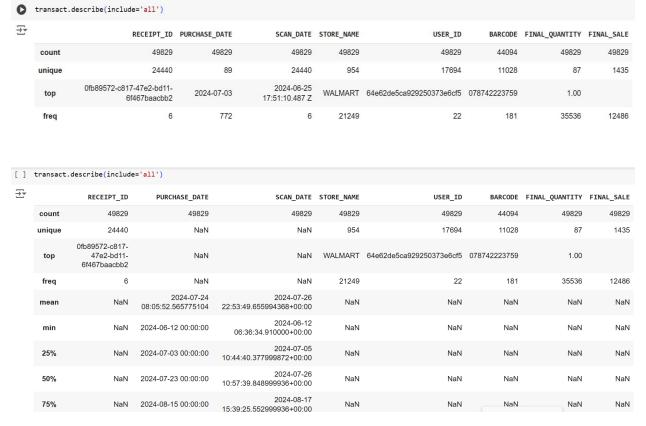
Checking for the presence of other data structures

Duplicates Check

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

Checking for duplicate records and their removal

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

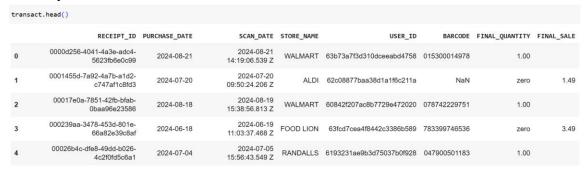


Summary Statistics before and after date-time conversion

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.

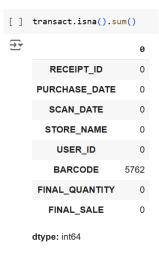
The data in the 'FINAL_QUANTITY' column and the 'FINAL_SALE' column seemed ambiguous as seen in the image below. The 'FINAL_SALE' column had a lot of rows with one space character (" ") and the corresponding 'FINAL_QUANTITY' rows had a value. Then where there was a 'FINAL_SALE' value, there the 'FINAL_QUANTITY' 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.



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.



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.

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.



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!

```
#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

#Creating an array with unique barcode values where the FINAL_SALE column is blank

x = transact_sale_blank['BARCODE'].unique()

len(x)

6805

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:

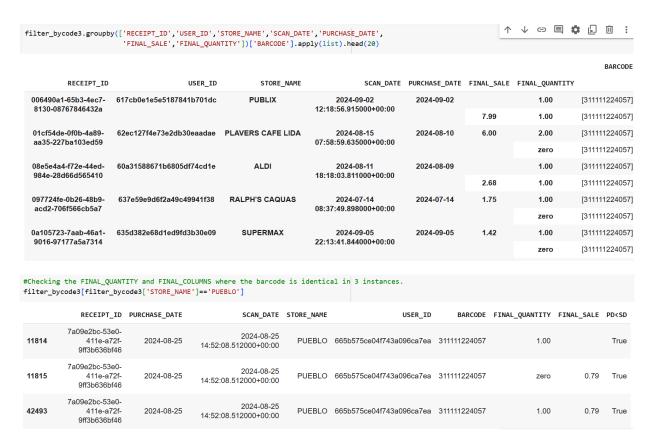
❖ From the above results it was determined that there were no additional unique barcodes in the data frame with a blank "FINAL_SALE" column that are not present in the data frame with the rest of the values.

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:

							BARCODE
RECEIPT_ID	USER_ID	STORE_NAME	SCAN_DATE	PURCHASE_DATE	FINAL_SALE	FINAL_QUANTITY	
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-	5ecb23aea0134313d2d8a3dd	KROGER	2024-06-23 08:43:56.378000+00:00	2024-06-17		1.00	[511111045496
o8a9-f387991ad0fa					2.50	1.00	[511111045496
86c09c9-c5ce-4d4f-	666f1d84465f309038ab3a6c	PRICE CHOPPER	2024-09-02 13:54:37.016000+00:00	2024-09-02	1.00	1.00	[511111045496
b004-024344662d13						zero	[511111045496
7e06a61-0eb4-4138-	63dc6c96dcb50fbd3083edb1	SHOP RITE	2024-06-27 12:31:18.008000+00:00	2024-06-20	1.27	1.00	[511111045496
bdf0-c6e46c9a4526						zero	[511111045496
2fb7d5d-3bcd-4aab- a079-45447bf4b4a5	6621f37cc41e9f27acd82170	STOP & SHOP	2024-06-16	2024-06-15		1.00	[511111045496
			18:20:19.585000+00:00		0.57	1.00	[511111045496



Random Barcodes grouped by other columns

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

```
#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: <a href="https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy">https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy</a>
 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
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.

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.



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)
                                                                                 ID
           CREATED_DATE STATE
                                             BIRTH_DATE
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.



Months when the repetitions happened

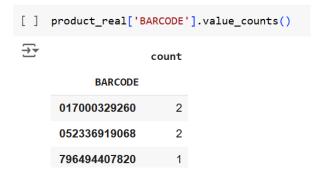
This is a **Data Quality Issue** unless there is an underlying reason for this repetition. But the reason remains inconclusive.

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.





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.



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.



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.