In [ ]:
```
#This notebook centers on analyzing brand data to detect any possible
# We start by importing the data and standardizing the nested JSON str
```

In [16]:
```
import pandas as pd
import json

# Open the file and read lines
with open('/Users/project/brands.json', 'r') as file:
    data = file.readlines()

# Parse JSON and convert to DataFrame
brands_df = pd.json_normalize([json.loads(line) for line in data])
```

In [17]:
```
brands_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1167 entries, 0 to 1166
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   barcode        1167 non-null   object
 1   category       1012 non-null   object
 2   categoryCode   517 non-null    object
 3   name           1167 non-null   object
 4   topBrand       555 non-null    object
 5   _id.$oid       1167 non-null   object
 6   cpg.$id.$oid   1167 non-null   object
 7   cpg.$ref       1167 non-null   object
 8   brandCode      933 non-null    object
dtypes: object(9)
memory usage: 82.2+ KB
```

In [18]:
```
# Renaming the column value
brands_df.rename(columns={'_id.$oid': 'brandId'}, inplace=True)
```

In [4]:
```
# splitting the data that might be useful to establish if there is any

split_data = brands_df['name'].str.split('@', expand=True)

# Extract the ID (second part after splitting) and strip any leading c
brands_df['named_id'] = split_data[1].str.strip()
```

In [9]:
```
brands_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1167 entries, 0 to 1166
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   barcode        1167 non-null   object
 1   category       1012 non-null   object
 2   categoryCode   517 non-null    object
 3   name           1167 non-null   object
 4   topBrand       555 non-null    object
 5   brandId        1167 non-null   object
 6   cpg.$id.$oid   1167 non-null   object
 7   cpg.$ref       1167 non-null   object
 8   brandCode      933 non-null    object
 9   named_id       428 non-null    object
dtypes: object(10)
memory usage: 91.3+ KB
```

In [11]:
```
# checking for duplicates

duplicate_rows = brands_df[brands_df.duplicated()]
duplicate_rows.count()
```

Out[11]:
```
barcode         0
category        0
categoryCode    0
name            0
topBrand        0
brandId         0
cpg.$id.$oid    0
cpg.$ref        0
brandCode       0
named_id        0
dtype: int64
```

In [ ]:
```
# Name column has name contains name and code of the product.
# We can seperate the id[numeric value] and made a seperate one.
# we can try filling the brandcode values that are [NaN] with name col
# This is one of the question we can raise for clarification.
```

In [17]:
```
brands_df.isnull().sum()
```

Out[17]:
```
barcode           0
category        155
categoryCode    650
name              0
topBrand        612
_id.$oid          0
cpg.$id.$oid      0
cpg.$ref          0
brandCode       234
named_id        739
dtype: int64
```

In [12]:
```
value_counts = brands_df['category'].value_counts()

# Print the result
print(value_counts)
```

```
category
Baking                          369
Beer Wine Spirits                90
Snacks                           75
Candy & Sweets                   71
Beverages                        63
Magazines                        44
Health & Wellness                44
Breakfast & Cereal               40
Grocery                          39
Dairy                            33
Condiments & Sauces              27
Frozen                           24
Personal Care                    20
Baby                             18
Canned Goods & Soups             12
Beauty                            9
Cleaning & Home Improvement       6
Deli                              6
Beauty & Personal Care            6
Household                         5
Bread & Bakery                    5
Dairy & Refrigerated              5
Outdoor                           1
Name: count, dtype: int64
```

In [15]:
```
brands_df.head(5)
```

Out[15]:

category	categoryCode	name	topBrand	brandId	
Baking	BAKING	test brand @1612366101024	False	601ac115be37ce2ead437551	601ac1
Beverages	BEVERAGES	Starbucks	False	601c5460be37ce2ead43755f	5332f5
Baking	BAKING	test brand @1612366146176	False	601ac142be37ce2ead43755d	601ac1
Baking	BAKING	test brand @1612366146051	False	601ac142be37ce2ead43755a	601ac1
Candy & Sweets	CANDY_AND_SWEETS	test brand @1612366146827	False	601ac142be37ce2ead43755e	5332fa

In [19]:
```
brands_df.isnull().sum()
```

Out[19]:
```
barcode           0
category        155
categoryCode    650
name              0
topBrand        612
brandId           0
cpg.$id.$oid      0
cpg.$ref          0
brandCode       234
dtype: int64
```

In [ ]:
```
# Issues or Anamolies found in the data

# There are no duplicates that are found in the data
# There are however emply values that are present in the dataset ['top
# categoryCode and brandCode are posing similar values.
```