Brands_Quality

May 29, 2024

1 Import Libraries

[1]:
```
import pandas as pd
import numpy as np
import json
```

2 Convert JSON into DataFrame Object

[2]:
```
brands = pd.read_json('brands.json', lines=True)
```

3 Get overview of data

[3]:
```
brands.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1167 entries, 0 to 1166
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   _id           1167 non-null   object
 1   barcode       1167 non-null   int64
 2   category      1012 non-null   object
 3   categoryCode  517 non-null    object
 4   cpg           1167 non-null   object
 5   name          1167 non-null   object
 6   topBrand      555 non-null    float64
 7   brandCode     933 non-null    object
dtypes: float64(1), int64(1), object(6)
memory usage: 73.1+ KB
```

[4]:
```
brands.head()
```

[4]:

	_id	barcode	category	\
0	{'$oid': '601ac115be37ce2ead437551'}	511111019862	Baking	
1	{'$oid': '601c5460be37ce2ead43755f'}	511111519928	Beverages	
2	{'$oid': '601ac142be37ce2ead43755d'}	511111819905	Baking	

```
3  {'$oid': '601ac142be37ce2ead43755a'}  511111519874          Baking
4  {'$oid': '601ac142be37ce2ead43755e'}  511111319917  Candy & Sweets

       categoryCode                                                cpg  \
0            BAKING  {'$id': {'$oid': '601ac114be37ce2ead437550'}, …
1         BEVERAGES  {'$id': {'$oid': '5332f5fbe4b03c9a25efd0ba'}, …
2            BAKING  {'$id': {'$oid': '601ac142be37ce2ead437559'}, …
3            BAKING  {'$id': {'$oid': '601ac142be37ce2ead437559'}, …
4  CANDY_AND_SWEETS  {'$id': {'$oid': '5332fa12e4b03c9a25efd1e7'}, …

                         name  topBrand                      brandCode
0  test brand @1612366101024       0.0                            NaN
1                  Starbucks       0.0                      STARBUCKS
2  test brand @1612366146176       0.0  TEST BRANDCODE @1612366146176
3  test brand @1612366146051       0.0  TEST BRANDCODE @1612366146051
4  test brand @1612366146827       0.0  TEST BRANDCODE @1612366146827
```

4 Evaluation

4.1 Null/NaN/Missing Values

[5]:
```
# Get the total number of records with Null values
brands.isnull().sum()
```

[5]:
```
_id              0
barcode          0
category       155
categoryCode   650
cpg              0
name             0
topBrand       612
brandCode      234
dtype: int64
```

Odd that more than half of the records in the Brands file are missing a categoryCode. Let's take a quick look at a null categoryCode row.

4.1.1 Missing categoryCodes v Categories

[6]:
```
nulls = brands.loc[brands['categoryCode'].isnull()]
nulls
```

[6]:
```
                                    _id       barcode  \
7    {'$oid': '5cdad0f5166eb33eb7ce0faa'}  511111104810
8    {'$oid': '5ab15636e4b0be0a89bb0b07'}  511111504412
9    {'$oid': '5c408e8bcd244a1fdb47aee7'}  511111504788
11   {'$oid': '57c08106e4b0718ff5fcb02c'}  511111102540
```

```
12    {'$oid': '588ba07be4b02187f85cdadd'}  511111201076
…                                      …           …
1159  {'$oid': '585a96cbe4b03e62d1ce0e88'}  511111501619
1160  {'$oid': '5887a216e4b02187f85cdad5'}  511111401155
1161  {'$oid': '5332f709e4b03c9a25efd0f2'}  511111403845
1163  {'$oid': '5dc1fca91dda2c0ad7da64ae'}  511111706328
1165  {'$oid': '5a021611e4b00efe02b02a57'}  511111400608

                 category categoryCode  \
7      Condiments & Sauces          NaN
8     Canned Goods & Soups          NaN
9                  Baking          NaN
11                    NaN          NaN
12                 Baking          NaN
…                    …          …
1159           Beverages          NaN
1160                Deli          NaN
1161    Beer Wine Spirits          NaN
1163   Breakfast & Cereal          NaN
1165             Grocery          NaN

                                                    cpg  \
7     {'$ref': 'Cogs', '$id': {'$oid': '559c2234e4b0…
8     {'$ref': 'Cogs', '$id': {'$oid': '5a734034e4b0…
9     {'$ref': 'Cogs', '$id': {'$oid': '59ba6f1ce4b0…
11    {'$ref': 'Cpgs', '$id': {'$oid': '5332f5f2e4b0…
12    {'$ref': 'Cogs', '$id': {'$oid': '559c2234e4b0…
…                                                  …
1159  {'$ref': 'Cogs', '$id': {'$oid': '5332f5fbe4b0…
1160  {'$ref': 'Cogs', '$id': {'$oid': '559c2234e4b0…
1161  {'$ref': 'Cogs', '$id': {'$oid': '5332f709e4b0…
1163  {'$ref': 'Cogs', '$id': {'$oid': '53e10d6368ab…
1165  {'$ref': 'Cogs', '$id': {'$oid': '5332f5f6e4b0…

                      name  topBrand             brandCode
7                J.L. Kraft       NaN            J.L. KRAFT
8     Campbell's Home Style       0.0  CAMPBELLS HOME STYLE
9                      test       NaN                  TEST
11               MorningStar       NaN                   NaN
12                   Calumet       0.0                CALUMET
…                      …       …                     …
1159              Pepsi Max       0.0
1160               Claussen       0.0              CLAUSSEN
1161              Blue Moon       0.0             BLUE MOON
1163   Dippin Dots® Cereal       NaN    DIPPIN DOTS CEREAL
1165      LIPTON TEA Leaves       0.0     LIPTON TEA Leaves
```

```
[650 rows x 8 columns]
```

[7]:
```
# In row 7 the category 'Condiments & Sauces' follows the same style as Candy &␣
 ↪Sweets,
# let's check out if there's a categoryCode associated with Condiments & Sauces
brands.loc[(brands['category'] == 'Condiments & Sauces') &␣
 ↪brands['categoryCode'].notnull() ]
```

[7]:
```
Empty DataFrame
Columns: [_id, barcode, category, categoryCode, cpg, name, topBrand, brandCode]
Index: []
```

[8]:
```
# Was that code written poorly, or is there no categoryCode associated with␣
 ↪Condiments & Sauces?
# Let's compare against a category we know has a categoryCode, Baking:
brands.loc[(brands['category'] == 'Baking') & brands['categoryCode'].notnull() ]
```

[8]:
```
                                        _id       barcode category  \
0     {'$oid': '601ac115be37ce2ead437551'}  511111019862   Baking
2     {'$oid': '601ac142be37ce2ead43755d'}  511111819905   Baking
3     {'$oid': '601ac142be37ce2ead43755a'}  511111519874   Baking
5     {'$oid': '601ac142be37ce2ead43755b'}  511111719885   Baking
6     {'$oid': '601ac142be37ce2ead43755c'}  511111219897   Baking
…                                      …           …      …
1145  {'$oid': '5f5bc4f1be37ce17125ac0e9'}  511111716594   Baking
1152  {'$oid': '5f3e9172be37ce5a0e01b955'}  511111715559   Baking
1158  {'$oid': '5f628215be37ce78e6e2efab'}  511111716648   Baking
1162  {'$oid': '5f77274dbe37ce6b592e90c0'}  511111116752   Baking
1166  {'$oid': '6026d757be37ce6369301468'}  511111019930   Baking

     categoryCode                                                cpg  \
0          BAKING  {'$id': {'$oid': '601ac114be37ce2ead437550'}, …
2          BAKING  {'$id': {'$oid': '601ac142be37ce2ead437559'}, …
3          BAKING  {'$id': {'$oid': '601ac142be37ce2ead437559'}, …
5          BAKING  {'$id': {'$oid': '601ac142be37ce2ead437559'}, …
6          BAKING  {'$id': {'$oid': '601ac142be37ce2ead437559'}, …
…           …                                                  …
1145       BAKING  {'$ref': 'Cogs', '$id': {'$oid': '5f5bc4f1be37…
1152       BAKING  {'$ref': 'Cogs', '$id': {'$oid': '5f3e9172be37…
1158       BAKING  {'$ref': 'Cogs', '$id': {'$oid': '5f628214be37…
1162       BAKING  {'$ref': 'Cogs', '$id': {'$oid': '5f77274dbe37…
1166       BAKING  {'$id': {'$oid': '6026d757be37ce6369301467'}, …

                           name  topBrand                      brandCode
0     test brand @1612366101024       0.0                            NaN
2     test brand @1612366146176       0.0  TEST BRANDCODE @1612366146176
3     test brand @1612366146051       0.0  TEST BRANDCODE @1612366146051
```

```
5     test brand @1612366146091       0.0  TEST BRANDCODE @1612366146091
6     test brand @1612366146133       0.0  TEST BRANDCODE @1612366146133
…                           …       …                              …
1145  test brand @1599849713740       NaN  TEST BRANDCODE @1599849713740
1152  test brand @1597935986434       NaN  TEST BRANDCODE @1597935986434
1158  test brand @1600291349042       NaN  TEST BRANDCODE @1600291349043
1162  test brand @1601644365844       NaN                            NaN
1166  test brand @1613158231643       0.0  TEST BRANDCODE @1613158231644

[359 rows x 8 columns]
```

Why are some categoryCodes missing when the category itself clearly exists? I assume either: 1. This was designed this way so that when one value is missing, the other gets filled in. 2. Data sourcing issues. Whatever is sending the Brand info is faulty.

4.2 Unique Values

[9]:
```
brands_json = brands.copy()
```

[10]:
```
# Convert dictionary columns to JSON strings to allow nunique() to run
brands_json._id = brands_json._id.apply(json.dumps)
brands_json.cpg = brands_json.cpg.apply(json.dumps)
```

[11]:
```
brands_json.nunique()
```

[11]:
```
_id             1167
barcode         1160
category          23
categoryCode      14
cpg              206
name            1156
topBrand           2
brandCode        897
dtype: int64
```

[12]:
```
# compared to the whole:
brands.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1167 entries, 0 to 1166
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   _id           1167 non-null   object
 1   barcode       1167 non-null   int64
 2   category      1012 non-null   object
 3   categoryCode  517 non-null    object
 4   cpg           1167 non-null   object
```

```
 5   name          1167 non-null   object
 6   topBrand      555 non-null    float64
 7   brandCode     933 non-null    object
dtypes: float64(1), int64(1), object(6)
memory usage: 73.1+ KB
```

Findings: 1. Each primary key, _id, is unique. Good. 2. Most of the barcodes are unique, interesting. 3. Only 23 categories? Makes sense for goods.

4.3 Rectify data types shown between .info() & .head()¶

[13]:
```
# When looking at the JSON files, topBrand is showing as a bool, instead of 0.0/
 ↪1.0, changing the data type to bool
brands['topBrand'] = brands['topBrand'].astype('bool')
```

[14]:
```
brands
```

[14]:

	_id	barcode	category	categoryCode	cpg	name	topBrand	brandCode
0	{'$oid': '601ac115be37ce2ead437551'}	511111019862	Baking	BAKING	{'$id': {'$oid': '601ac114be37ce2ead437550'}, …	test brand @1612366101024	False	NaN
1	{'$oid': '601c5460be37ce2ead43755f'}	511111519928	Beverages	BEVERAGES	{'$id': {'$oid': '5332f5fbe4b03c9a25efd0ba'}, …	Starbucks	False	STARBUCKS
2	{'$oid': '601ac142be37ce2ead43755d'}	511111819905	Baking	BAKING	{'$id': {'$oid': '601ac142be37ce2ead437559'}, …	test brand @1612366146176	False	TEST BRANDCODE @1612366146176
3	{'$oid': '601ac142be37ce2ead43755a'}	511111519874	Baking	BAKING	{'$id': {'$oid': '601ac142be37ce2ead437559'}, …	test brand @1612366146051	False	TEST BRANDCODE @1612366146051
4	{'$oid': '601ac142be37ce2ead43755e'}	511111319917	Candy & Sweets	CANDY_AND_SWEETS	{'$id': {'$oid': '5332fa12e4b03c9a25efd1e7'}, …			
…	…	…	…	…	…			
1162	{'$oid': '5f77274dbe37ce6b592e90c0'}	511111116752	Baking	BAKING	{'$ref': 'Cogs', '$id': {'$oid': '5f77274dbe37…			
1163	{'$oid': '5dc1fca91dda2c0ad7da64ae'}	511111706328	Breakfast & Cereal	NaN	{'$ref': 'Cogs', '$id': {'$oid': '53e10d6368ab…			
1164	{'$oid': '5f494c6e04db711dd8fe87e7'}	511111416173	Candy & Sweets	CANDY_AND_SWEETS	{'$ref': 'Cogs', '$id': {'$oid': '5332fa12e4b0…			
1165	{'$oid': '5a021611e4b00efe02b02a57'}	511111400608	Grocery	NaN	{'$ref': 'Cogs', '$id': {'$oid': '5332f5f6e4b0…			
1166	{'$oid': '6026d757be37ce6369301468'}	511111019930	Baking	BAKING	{'$id': {'$oid': '6026d757be37ce6369301467'}, …			

```
4     test brand @1612366146827     False  TEST BRANDCODE @1612366146827
…                            …       …                              …
1162  test brand @1601644365844      True                            NaN
1163        Dippin Dots® Cereal      True             DIPPIN DOTS CEREAL
1164  test brand @1598639215217      True  TEST BRANDCODE @1598639215217
1165          LIPTON TEA Leaves     False              LIPTON TEA Leaves
1166  test brand @1613158231643     False  TEST BRANDCODE @1613158231644

[1167 rows x 8 columns]
```

4.4 Category Inspection

[15]:
```
# Check out the set of unique categories in brands:
category_set = brands['category'].unique()
```

[16]:
```
category_set
```

[16]:
```
array(['Baking', 'Beverages', 'Candy & Sweets', 'Condiments & Sauces',
       'Canned Goods & Soups', nan, 'Magazines', 'Breakfast & Cereal',
       'Beer Wine Spirits', 'Health & Wellness', 'Beauty', 'Baby',
       'Frozen', 'Grocery', 'Snacks', 'Household', 'Personal Care',
       'Dairy', 'Cleaning & Home Improvement', 'Deli',
       'Beauty & Personal Care', 'Bread & Bakery', 'Outdoor',
       'Dairy & Refrigerated'], dtype=object)
```

[17]:
```
# Show the raw numbers
brands['category'].value_counts()
```

[17]:
```
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
```

[18]:
```
# Show as percentages:
total = brands['category'].value_counts().sum()
print(total)
```

```
1012
```

[19]:
```
print((brands['category'].value_counts() / total) * 100.0)
```

```
category
Baking                         36.462451
Beer Wine Spirits               8.893281
Snacks                          7.411067
Candy & Sweets                  7.015810
Beverages                       6.225296
Magazines                       4.347826
Health & Wellness               4.347826
Breakfast & Cereal              3.952569
Grocery                         3.853755
Dairy                           3.260870
Condiments & Sauces             2.667984
Frozen                          2.371542
Personal Care                   1.976285
Baby                            1.778656
Canned Goods & Soups            1.185771
Beauty                          0.889328
Cleaning & Home Improvement     0.592885
Deli                            0.592885
Beauty & Personal Care          0.592885
Household                       0.494071
Bread & Bakery                  0.494071
Dairy & Refrigerated            0.494071
Outdoor                         0.098814
Name: count, dtype: float64
```

Of COURSE baking is the top category, with spirits being a close 2nd.

5 Conclusions

The source that sends the Brands unstructured dataset seems to have some quality issues. This is most profound when looking at the category & categoryCode columns. The quality issues are found in 2 ways:

1. The current usage of both columns is redundant - categoryCode entries are just category entries written in all caps and using underscores instead of spaces. I would recommend creating another another data entity where categoryCode is a few characters, and category is written-out description. EX: BAK -> Baking, etc
2. Many missing categoryCode entries. How are there more than 50% missing categoryCode entries?