Exploration of Products Table

Displaying Missing Values from the Products table:

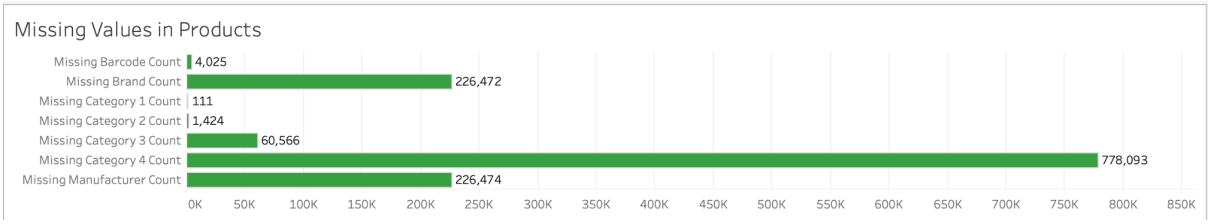
Query:

```
print("\nMissing Values before dropping duplicate records:")
print(products_df.isnull().sum())
```

Output:

```
Missing Values before dropping duplicate records:
>>> print(products_df.isnull().sum())
CATEGORY_1          111
CATEGORY_2         1424
CATEGORY_3        60566
CATEGORY_4       778093
MANUFACTURER     226474
BRAND            226472
BARCODE            4025
```

Graphical view:



Displaying count of Duplicate records:

```
Number of Duplicate records:
>>> print(products_df.duplicated().sum())
215
```

Displaying the count of Missing Values:

```
Missing Values before dropping duplicate records:
>>> print(products_df.isnull().sum())
CATEGORY_1          111
CATEGORY_2         1422
CATEGORY_3        60563
CATEGORY_4       777884
MANUFACTURER     226464
BRAND            226462
BARCODE            3968
```

Displaying the number of Duplicate barcode rows:

Query:

```
# Number of duplicate barcodes
print("\nNumber of duplicate barcode rows is : ")
print(duplicate_products_df.shape[0])
```

Output:

```
Number of duplicate barcode rows is :
>>> print(duplicate_products_df.shape[0])
4022
```

These are the records in the products table with duplicate barcode values:

```
>>> print(duplicate_products_df.sort_values(by="BARCODE").head(100))
```

	CATEGORY_1	CATEGORY_2	CATEGORY_3	CATEGORY_4	MANUFACTURER	BRAND	BARCODE
841230	Snacks	Candy	Chocolate Candy	NaN	MARS WRIGLEY	M&M'S	404310
139121	Snacks	Candy	Chocolate Candy	NaN	PLACEHOLDER MANUFACTURER	BRAND NOT KNOWN	404310
610681	Snacks	Nuts & Seeds	Snack Seeds	NaN	SUNRIDGE FARMS	SUNRIDGE FARMS	701983
645266	Snacks	Chips	Crisps	NaN	TRADER JOE'S	TRADER JOE'S	701983
681268	Snacks	Nuts & Seeds	Almonds	NaN	TRADER JOE'S	TRADER JOE'S	969307
171015	Snacks	Nuts & Seeds	Covered Nuts	NaN	TRADER JOE'S	TRADER JOE'S	969307
428256	Health & Wellness	Skin Care	Facial Lotion & Moisturizer	NaN	R.M. PALMER COMPANY, LLC	PALMER	1018158
123194	Health & Wellness	Skin Care	Lip Balms & Treatments	Medicated Lip Treatments	E.T. BROWNE DRUG CO., INC.	PALMER'S SKIN & HAIR CARE	1018158
36017	Snacks	Candy	Candy Variety Pack	NaN	THE HERSHEY COMPANY	HERSHEY'S	3422007
422809	Snacks	Candy	Chocolate Candy	NaN	THE HERSHEY COMPANY	HERSHEY'S	3422007
402382	Snacks	Candy	Confection Candy	NaN	THE HERSHEY COMPANY	REESE'S	3431207
144679	Snacks	Candy	Chocolate Candy	NaN	THE HERSHEY COMPANY	REESE'S	3431207
468720	Snacks	Candy	Gum	NaN	GENERAL LICENSED IP MANUFACTURER	DISNEY	3454503
528709	Snacks	Candy	Gum	NaN	THE HERSHEY COMPANY	ICE BREAKERS	3454503
766255	Snacks	Candy	Chocolate Candy	NaN	THE HERSHEY COMPANY	REESE'S	3473009
539918	Snacks	Candy	Gum	NaN	THE HERSHEY COMPANY	BUBBLE YUM	3473009
717446	Snacks	Candy	Gum	NaN	GENERAL LICENSED IP MANUFACTURER	DISNEY	3484708
594684	Snacks	Candy	Gum	NaN	THE HERSHEY COMPANY	ICE BREAKERS	3484708
193347	Snacks	Candy	Chocolate Candy	NaN	MARS WRIGLEY	M&M'S	4003207
137250	Snacks	Nuts & Seeds	Peanuts	NaN	MARS WRIGLEY	M&M'S	4003207
596778	Health & Wellness	First Aid	Ointments & Liquids	NaN	THE JM SMUCKER COMPANY	DICKINSON'S	5265169
768682	Health & Wellness	Skin Care	Skin Toners & Astringents	Astringents	THE JM SMUCKER COMPANY	DICKINSON'S	5265169
132547	Snacks	Cookies	NaN	NaN	NaN	NaN	20031077
783021	Snacks	Cookies	NaN	NaN	PLACEHOLDER MANUFACTURER	PRIVATE LABEL	20031077
56987	Snacks	Nuts & Seeds	Almonds	NaN	PLACEHOLDER MANUFACTURER	BRAND NOT KNOWN	20159078
776953	Snacks	Nuts & Seeds	Covered Nuts	NaN	PLACEHOLDER MANUFACTURER	BRAND NOT KNOWN	20159078
260690	Snacks	Candy	Chocolate Candy	NaN	LIDL US, LLC	LIDL	20522445
127335	Snacks	Nuts & Seeds	Pistachios	NaN	LIDL US, LLC	LIDL	20522445
206172	Snacks	Crackers	Graham Crackers	NaN	LIDL US, LLC	LIDL	20733056
720019	Snacks	Crackers	Graham Crackers	NaN	PLACEHOLDER MANUFACTURER	PRIVATE LABEL	20733056
96438	Snacks	Crackers	Other Crackers	NaN	NaN	NaN	20733254
782036	Snacks	Chips	Crisps	NaN	PLACEHOLDER MANUFACTURER	PRIVATE LABEL	20733254
274690	Snacks	Candy	Chocolate Candy	NaN	PROCTER & GAMBLE	BOUNTY	40111216
454537	Snacks	Candy	Chocolate Candy	NaN	MARS WRIGLEY	MARS	40111216
668717	Health & Wellness	Skin Care	Eye Creams	NaN	WALA HEILMITTEL GMBH	DR HAUSCHKA	42208488
333771	Health & Wellness	Skin Care	Eye Creams	NaN	NaN	NaN	42208488
287404	Snacks	Candy	Chocolate Candy	NaN	NESTLE	NESTLE	50426171
184572	Snacks	Candy	Chocolate Candy	NaN	NaN	NaN	50426171
181902	Snacks	Candy	Confection Candy	NaN	GRUPO NACIONAL DE CHOCOLATES SA	NUTRESA	75053055
114131	Snacks	Candy	Chocolate Candy	NaN	GRUPO NACIONAL DE CHOCOLATES SA	NUTRESA	75053055
162	Health & Wellness	Hair Removal	Shaving Gel & Cream	Women's Shaving Gel & Cream	PLACEHOLDER MANUFACTURER	PRORASO	80199137
810359	Health & Wellness	Hair Removal	Shaving Gel & Cream	Men's Shaving Gel & Cream	PLACEHOLDER MANUFACTURER	PRORASO	80199137
216314	Snacks	Candy	Chocolate Candy	NaN	FERRERO GROUP	KINDER	80310167
585634	Snacks	Candy	Chocolate Candy	NaN	KINDER'S	KINDER'S	80310167
300327	Snacks	Candy	Confection Candy	NaN	PERFETTI VAN MELLE	MENTOS	87108538
108325	Snacks	Candy	Mints	NaN	PERFETTI VAN MELLE	MENTOS	87108538
37152	Snacks	Candy	Confection Candy	NaN	PERFETTI VAN MELLE	MENTOS	87306286
777589	Snacks	Candy	Mints	NaN	PERFETTI VAN MELLE	MENTOS	87306286
379746	Snacks	Candy	Gum	NaN	NaN	NaN	360054002
708466	Snacks	Candy	Mints	NaN	LOTUS BRANDS INC	ECO-DENT	360054002
304021	Health & Wellness	Hair Care	Hair Color	NaN	HENKEL	GÖT2B	17000329260
213340	Health & Wellness	Hair Care	Hair Color	NaN	HENKEL	SCHWARZKOPF	17000329260
28421	Health & Wellness	Hair Care	Hair Color	NaN	HENKEL	SCHWARZKOPF	52336919068
709607	Health & Wellness	Hair Care	Hair Color	NaN	HENKEL	GÖT2B	52336919068

Calculating the percentage of duplicate barcode rows out of total rows:

Query:

```
# Filter the dataframe to include only duplicate barcodes
duplicate_products_df = products_df[products_df['BARCODE'].duplicated(keep=False)]

# The percentage of duplicate barcode rows out of the total rows
print("\nPercentage of duplicate barcode rows out of total rows is: ")
print((duplicate_products_df.shape[0] / products_df.shape[0]) * 100)
```

Output:

```
Percentage of duplicate barcode rows out of total rows is:
>>> print((duplicate_products_df.shape[0] / products_df.shape[0]) * 100)
0.4757865797900719
```

The percentage of duplicate barcode rows out of the total rows is 0.48%

Removing all duplicate barcode values rows:

Query:

```
# Drop all duplicate barcode values rows from the products table
products_df = products_df.drop_duplicates(subset=['BARCODE'], keep=False)


print(products_df.isnull().sum())
```

Output:

```
>>> print(products_df.isnull().sum())
CATEGORY_1          111
CATEGORY_2          661
CATEGORY_3        58712
CATEGORY_4       774085
MANUFACTURER     226212
BRAND            226210
BARCODE               0
```

Reasons to remove rows where BARCODE is having NULL or DUPLICATE values:

- Since we are assuming that barcode is uniquely identifying each product in the PRODUCTS table, it should be unique and should not contain NULL values.
- Also the duplicate and missing barcode values amount to only 0.48% of the total records in the PRODUCTS table, hence removing them will not cause a drastic change in our analysis process.

Converting all other columns to varchar apart from ‘BARCODE’:

Query:

```
# Convert all columns except 'BARCODE' to string (VARCHAR equivalent)
for col in products_df.columns:
   if col != 'BARCODE':
       products_df[col] = products_df[col].astype(str)

# Verify data types
print(products_df.dtypes)
```

Output:

```
>>> print(products_df.dtypes)
CATEGORY_1       object
CATEGORY_2       object
CATEGORY_3       object
CATEGORY_4       object
MANUFACTURER     object
BRAND            object
BARCODE           Int64
dtype: object
```

Showing that if CATEGORY_1 is ‘nan’ or empty then all other categories are blank as well:

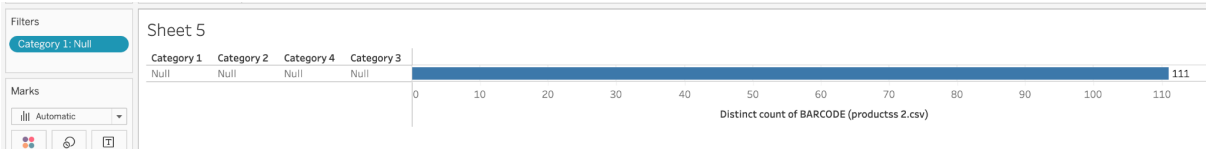
Query:

```
nan_cat1 = products_df[products_df['CATEGORY_1'] == 'nan']
print(nan_cat1)
```

Output:

	CATEGORY_1	CATEGORY_2	CATEGORY_3	CATEGORY_4	MANUFACTURER	BRAND	BARCODE
5184	nan	nan	nan	nan	KEURIG DR PEPPER	POLAR	715371108216
15048	nan	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	196365503574
25313	nan	nan	nan	nan	MOLSONCOORS	COORS LIGHT	198181051598
35604	nan	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	196983108397
43214	nan	nan	nan	nan	KEURIG DR PEPPER	POLAR	644376098768
49899	nan	nan	nan	nan	GENERAL MILLS	CHEERIOS	511111403630
53253	nan	nan	nan	nan	MOLSONCOORS	COORS LIGHT	198181051789
64977	nan	nan	nan	nan	MOLSONCOORS	COORS LIGHT	198181051826
67458	nan	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	49000075465
70290	nan	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	197315067825
77607	nan	nan	nan	nan	MOLSONCOORS	COORS LIGHT	198181051635
81528	nan	nan	nan	nan	MARS WRIGLEY	SNICKERS	511111914983
98822	nan	nan	nan	nan	PEPSICO	CHEETOS	888783778695
101201	nan	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	196983109042
111941	nan	nan	nan	nan	PEPSICO	CHEETOS	28400745048
114216	nan	nan	nan	nan	MOLSONCOORS	COORS LIGHT	198199018743
119798	nan	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	511111703983
120062	nan	nan	nan	nan	PEPSICO	CHEETOS	195566252762
126932	nan	nan	nan	nan	PEPSICO	CHEETOS	198040233967
141069	nan	nan	nan	nan	PEPSICO	CHEETOS	198040234469
142100	nan	nan	nan	nan	PEPSICO	CHEETOS	311111872449
143369	nan	nan	nan	nan	PEPSICO	CHEETOS	197419194960
147019	nan	nan	nan	nan	PEPSICO	FRITO-LAY	28400752015
153370	nan	nan	nan	nan	PEPSICO	BUBLY SPARKLING WATER	511111305354
156938	nan	nan	nan	nan	KEURIG DR PEPPER	POLAR	74027500331
169244	nan	nan	nan	nan	PEPSICO	CHEETOS	198181685182
173804	nan	nan	nan	nan	KEURIG DR PEPPER	POLAR	71537411614
187453	nan	nan	nan	nan	PEPSICO	CHEETOS	198181685519
195535	nan	nan	nan	nan	KEURIG DR PEPPER	POLAR	644376098935
198139	nan	nan	nan	nan	PEPSICO	PEPSI	511111503972
198711	nan	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	686817056531
213948	nan	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	727875074263
218306	nan	nan	nan	nan	PEPSICO	CHEETOS	28400721011

Graphical view:



Similarly showing if CATEGORY_2 is ‘nan’ or empty then CATEGORY_3 and CATEGORY_4 are blank as well:

Query:

```
print(products_df[products_df['CATEGORY_2'] == 'nan'])
```

Output:

	CATEGORY_1	CATEGORY_2	CATEGORY_3	CATEGORY_4	MANUFACTURER	BRAND	BARCODE
1992	Needs Review	nan	nan	nan	PEPSICO	FRITO-LAY	28400020480
2597	Needs Review	nan	nan	nan	MOLSONCOORS	COORS LIGHT	71990300814
3997	Needs Review	nan	nan	nan	PEPSICO	PEPSI	120005105644
5184	nan	nan	nan	nan	KEURIG DR PEPPER	POLAR	715371108216
5528	Needs Review	nan	nan	nan	PEPSICO	CHEETOS	100412608327
6150	Needs Review	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	49000062908
9575	Needs Review	nan	nan	nan	KEURIG DR PEPPER	POLAR	39153800205
11029	Needs Review	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	49000060676
11194	Needs Review	nan	nan	nan	PEPSICO	FRITO-LAY	28400003087
11377	Needs Review	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	49000553901
13947	Needs Review	nan	nan	nan	MARS WRIGLEY	SNICKERS	655956023155
15048	nan	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	196365503574
17502	Needs Review	nan	nan	nan	PEPSICO	CHEETOS	28400705226
18980	Needs Review	nan	nan	nan	PEPSICO	FRITO-LAY	720996059690
19607	Needs Review	nan	nan	nan	PEPSICO	BUBLY SPARKLING WATER	100378692941
20539	Needs Review	nan	nan	nan	PEPSICO	FRITO-LAY	720995017844
22163	Needs Review	nan	nan	nan	PEPSICO	CHEETOS	15300200920
23271	Needs Review	nan	nan	nan	PEPSICO	FRITO-LAY	16590267
25313	nan	nan	nan	nan	MOLSONCOORS	COORS LIGHT	198181051598
27293	Needs Review	nan	nan	nan	KEURIG DR PEPPER	POLAR	71537201208
27689	Snacks	nan	nan	nan	THE HERSHEY COMPANY	HERSHEY'S	34000479269
32808	Needs Review	nan	nan	nan	KEURIG DR PEPPER	POLAR	311111772909
33298	Needs Review	nan	nan	nan	PEPSICO	FRITO-LAY	28400041409
35604	nan	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	196983108397
40660	Needs Review	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	49000248258
41472	Needs Review	nan	nan	nan	PEPSICO	CHEETOS	311111372895
42430	Needs Review	nan	nan	nan	PEPSICO	FRITO-LAY	639277527868
43214	nan	nan	nan	nan	KEURIG DR PEPPER	POLAR	644376098768
43362	Needs Review	nan	nan	nan	KEURIG DR PEPPER	POLAR	71537021547
45468	Needs Review	nan	nan	nan	PEPSICO	BUBLY SPARKLING WATER	372426252432
49167	Needs Review	nan	nan	nan	PEPSICO	PEPSI	700716630439
49899	nan	nan	nan	nan	GENERAL MILLS	CHEERIOS	511111403630
50309	Needs Review	nan	nan	nan	KEURIG DR PEPPER	POLAR	71537411454
50511	Needs Review	nan	nan	nan	KEURIG DR PEPPER	POLAR	311111672933
51805	Needs Review	nan	nan	nan	PEPSICO	FRITO-LAY	700707151615
53253	nan	nan	nan	nan	MOLSONCOORS	COORS LIGHT	198181051789
53335	Needs Review	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	49000995169
53412	Needs Review	nan	nan	nan	PEPSICO	FRITO-LAY	720995824503
54248	Needs Review	nan	nan	nan	THE COCA-COLA COMPANY	COCA-COLA	195566684747
57801	Needs Review	nan	nan	nan	PEPSICO	FRITO-LAY	720995403951

To confirm that each category is dependent on its previous category:

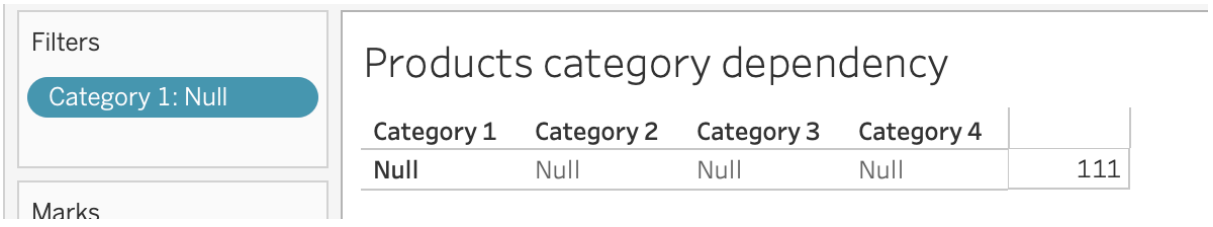
Query:

```
category1_nan = products_df[products_df['CATEGORY_1'] == 'nan']
print(f"CATEGORY_1 -> {category1_nan['CATEGORY_1'].count()} , CATEGORY_2 ->
{category1_nan['CATEGORY_2'].count()}\n")
```

Output:

```
CATEGORY_1 -> 111 , CATEGORY_2 -> 111
```

Graphical view:



For every category1 that is ‘nan’ or blank its corresponding category_2 is also ‘nan’:

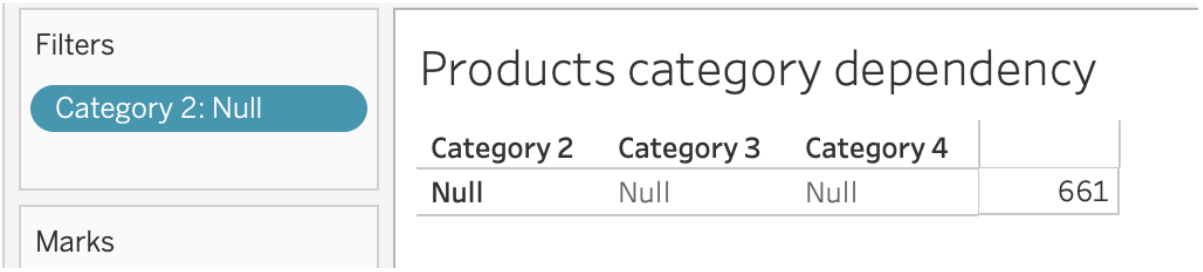
Query:

```
category2_nan = products_df[products_df['CATEGORY_2'] == 'nan']
print(f"CATEGORY_2 -> {category2_nan['CATEGORY_2'].count()} , CATEGORY_3 ->
{category2_nan['CATEGORY_3'].count()}\n")
```

Output:

```
CATEGORY_2 -> 661 , CATEGORY_3 -> 661
```

Graphical view:



Finally for every category_3 that is 'nan' or blank its corresponding category_4 is also 'nan':

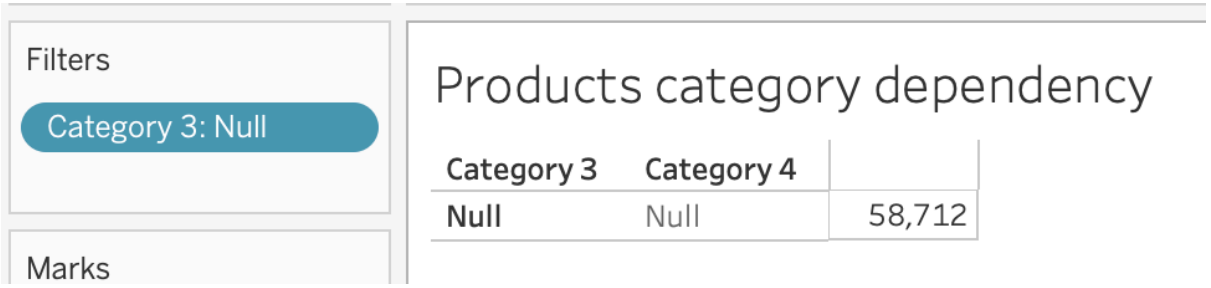
Query:

```
category3_nan = products_df[products_df['CATEGORY_3'] == 'nan']
print(f"CATEGORY_3 -> {category3_nan['CATEGORY_3'].count()} , CATEGORY_4 ->
{category3_nan['CATEGORY_4'].count()}\n")
```

Output:

```
CATEGORY_3 -> 58712 , CATEGORY_4 -> 58712
```

Graphical view:



Getting the total number of unique items in CATEGORY_1 along with their counts:

Query:

```
# Get the total number of unique items in CATEGORY_1 along with their counts
category1_counts = products_df["CATEGORY_1"].value_counts().reset_index()
category1_counts.columns = ["CATEGORY_1", "COUNT"]
```

Output:

```
>>> print(category1_counts)
                CATEGORY_1   COUNT
0        Health & Wellness  510376
1                   Snacks  322997
2                Beverages    3977
3                   Pantry     867
4    Apparel & Accessories     840
5                    Dairy     592
6             Needs Review     547
7                  Alcohol     475
8            Home & Garden     115
9                      nan     111
10           Deli & Bakery      66
11                  Frozen      62
12          Meat & Seafood      49
13          Sporting Goods      47
14         Office & School      45
15              Restaurant      35
16           Toys & Games      28
17     Household Supplies      24
18                 Produce      20
19  Animals & Pet Supplies      16
20    Arts & Entertainment       7
21              Electronics       5
22                  Mature       3
23        Vehicles & Parts       3
24          Baby & Toddler       3
25                  Beauty       2
26                   Media       2
27          Luggage & Bags       1
>>>
```

Foundings:
Health & Wellness has most number of products, followed by Snack and Beverages:

Getting the total number of unique items in 'MANUFACTURER' column with count:

Query:

```
# Get the total number of unique items in MANUFACTURER along with their counts
manufacturers_count = products_df["MANUFACTURER"].value_counts().reset_index()
manufacturers_count.columns = ["MANUFACTURER", "COUNT"]
print(manufacturers_count.head(10))
```

Output:

```
>>> print(manufacturers_count.head(10))
                MANUFACTURER   COUNT
0                        nan  226212
1  PLACEHOLDER MANUFACTURER   86892
2          REM MANUFACTURER   20813
3          PROCTER & GAMBLE   20796
4                   L'OREAL   16673
5                 UNILEVER    16655
6                   PEPSICO   14258
7         JOHNSON & JOHNSON   10287
8       THE HERSHEY COMPANY    9960
9               MARS WRIGLEY   9645
```

Getting the total number of unique items in 'BRAND' column with count:

Query:

```
# Get the total number of unique items in BRAND along with their counts
brand_count = products_df["BRAND"].value_counts().reset_index()
brand_count.columns = ["BRAND", "COUNT"]
print(brand_count.head(10))
```

Output:

```
>>> print(brand_count.head(10))
              BRAND   COUNT
0               nan  226210
1         REM BRAND   20813
2  BRAND NOT KNOWN    17020
3     PRIVATE LABEL   13464
4               CVS    6400
5              SEGO    4831
6            MEIJER    4050
7              DOVE    3834
8         RITE AID     3238
9            MATRIX    2958
```

Showing Number of unique brands associated with each manufacturer:

Query:

```
# Number of unique brands associated with each manufacturer
```

```
manufacturer_brand_count =
products_df.groupby("MANUFACTURER")["BRAND"].nunique().reset_index()
manufacturer_brand_count.columns = ["MANUFACTURER", "BRAND_COUNT"]

# Sort by BRAND_COUNT in descending order
manufacturer_brand_count = manufacturer_brand_count.sort_values(by="BRAND_COUNT",
ascending=False)

print(manufacturer_brand_count.head(10))
```

Output:

```
>>> print(manufacturer_brand_count.head(10))
                              MANUFACTURER  BRAND_COUNT
3005            PLACEHOLDER MANUFACTURER         1224
1445   GENERAL LICENSED IP MANUFACTURER           87
3078                    PROCTER & GAMBLE           67
2938                             PEPSICO           61
4020                             UNILEVER          55
1446                       GENERAL MILLS           49
3057  PRESTIGE CONSUMER HEALTHCARE INC.            48
2547              MONDELĒZ INTERNATIONAL           46
799                              CONAGRA           41
2665                              NESTLE           39
```

Graphical view:

Assessment Answers: Data Quality & Challenges in the Products Table

Q. Are there any data quality issues present?

Yes there are some data quality issues present in the product dataset.
Missing and invalid values:

a. There were 215 duplicate records which were removed.
b. According to my assumption the BARCODE in PRODUCTS table is supposed to uniquely identify the products.
c. But during my analysis I found that around 0.48% of the total data contains Duplicate or NULL BARCODES.

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

There are **no** fields in the Products Table that are challenging to understand. The dataset is well-structured, with clear field names such as BARCODE, CATEGORY_1, CATEGORY_2, CATEGORY_3, CATEGORY_4, MANUFACTURER, and BRAND, which are intuitive and follow a logical hierarchy. The analysis confirmed dependencies between categories and validated data consistency, making interpretation straightforward.

Overall, while there were **data quality issues**, no fields were difficult to understand based on the analysis.