In [1]:
```
import zipfile
import os
```

In [2]:
```
zip_path='archive.zip'
```

In [3]:
```
with zipfile.ZipFile(zip_path,'r') as zip_ref:
    zip_ref.extractall('extracted_files')
```

In [4]:
```
extracted_files=os.listdir('extracted_files')
print('Extracted Files:',extracted_files)
```

```
Extracted Files: ['amazon.csv']
```

In [5]:
```
import pandas as pd
```

```
  File "C:\Users\HP\anaconda3\Lib\site-packages\IPython\core\interactiveshell.py", line 3505, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)
  File "C:\Users\HP\AppData\Local\Temp\ipykernel_8296\4080736814.py", line 1, in <module>
    import pandas as pd
  File "C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\pandas\__init__.py", line 62, in <module>
    from pandas.core.api import (
  File "C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\pandas\core\api.py", line 28, in <module>
    from pandas.core.arrays import Categorical
  File "C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\pandas\core\arrays\__init__.py", line 1, in <module>
    from pandas.core.arrays.arrow import ArrowExtensionArray
  File "C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\pandas\core\arrays\arrow\__init__.py", line 5, in <module>
    from pandas.core.arrays.arrow.array import ArrowExtensionArray
  File "C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\pandas\core\arrays\arrow\array.py", line 64, in <module>
    from pandas.core.arrays.masked import BaseMaskedArray
  File "C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\pandas\core\arrays\masked.py", line 60, in <module>
    from pandas.core import (
  File "C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\pandas\core\nanops.py", line 52, in <module>
    bn = import_optional_dependency("bottleneck", errors="warn")
  File "C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\pandas\compat\_optional.py", line 135, in import_optional_dependency
    module = importlib.import_module(name)
  File "C:\Users\HP\anaconda3\Lib\importlib\__init__.py", line 126, in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
  File "C:\Users\HP\anaconda3\Lib\site-packages\bottleneck\__init__.py", line 7, in <module>
    from .move import (move_argmax, move_argmin, move_max, move_mean, move_median,

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
AttributeError: _ARRAY_API not found
```

Section A: Data Understanding & Cleaning: Understand the dataset like a data scientist in a product analytics team.

Summarize the dataset: Number of unique users, products, and reviews Top 5 categories by number of products Price range and discount insights Clean and preprocess the data: Convert prices to numeric Parse categories into hierarchy levels Normalize rating scores and count outliers or Create derived features like price_difference, value_for_money_score, weighted ratings. Handle missing values or anomalies Remove duplicates, invalid records Handle missing ratings/reviews with appropriate strategy

In [6]:
```
df=pd.read_csv('extracted_files/amazon.csv')
```

In [7]:
```
df.head()
```

Out[7]:

	product_id	product_name	category	discounted_price	a
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	₹399	
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	₹199	
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	Computers&Accessories|Accessories&Peripherals|...	₹199	
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	Computers&Accessories|Accessories&Peripherals|...	₹329	
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	Computers&Accessories|Accessories&Peripherals|...	₹154	

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1465 entries, 0 to 1464
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   product_id           1465 non-null   object
 1   product_name         1465 non-null   object
 2   category             1465 non-null   object
 3   discounted_price     1465 non-null   object
 4   actual_price         1465 non-null   object
 5   discount_percentage  1465 non-null   object
 6   rating               1465 non-null   object
 7   rating_count         1463 non-null   object
 8   about_product        1465 non-null   object
 9   user_id              1465 non-null   object
 10  user_name            1465 non-null   object
 11  review_id            1465 non-null   object
 12  review_title         1465 non-null   object
 13  review_content       1465 non-null   object
 14  img_link             1465 non-null   object
 15  product_link         1465 non-null   object
dtypes: object(16)
memory usage: 183.3+ KB
```

Number of unique users, products, and reviews

In [9]:
```
unique_users=df['user_id'].str.split(',').explode().nunique()
unique_products=df['product_id'].nunique()
unique_reviews=df['review_id'].str.split(',').explode().nunique()
print(f'Unique Users: {unique_users}')
print(f'Unique Products: {unique_products}')
print(f'Unique Reviews: {unique_reviews}')
```

```
Unique Users: 9050
Unique Products: 1351
Unique Reviews: 9269
```

Top 5 categories by number of products

In [10]:
```
top_5_categories=df['category'].str.split('|').explode().value_counts().head()
print(f'top_5_categories: \n {top_5_categories}')
```

```
top_5_categories:
 category
Electronics               526
Computers&Accessories     453
Home&Kitchen              448
Accessories&Peripherals   381
Kitchen&HomeAppliances    308
Name: count, dtype: int64
```

Price range and discount insights

In [11]:
```
price_features=['discounted_price','actual_price','discount_percentage']
```

In [12]:
```
for col in price_features:
    df[col]=df[col].replace(['₹','%',','],'',regex=True).astype(float)
```

In [13]:
```
df.head()
```

Out[13]:

	product_id	product_name	category	discounted_price	a
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	399.0	
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	199.0	
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	Computers&Accessories|Accessories&Peripherals|...	199.0	
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	Computers&Accessories|Accessories&Peripherals|...	329.0	
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	Computers&Accessories|Accessories&Peripherals|...	154.0	

In [14]:

```
df.isnull().sum()
```

Out[14]:

```
product_id             0
product_name           0
category               0
discounted_price       0
actual_price           0
discount_percentage    0
rating                 0
rating_count           2
about_product          0
user_id                0
user_name              0
review_id              0
review_title           0
review_content         0
img_link               0
product_link           0
dtype: int64
```

In [15]:

```
df=df.dropna()
```

In [16]:

```
df.isnull().sum()
```

Out[16]:
```
product_id             0
product_name           0
category               0
discounted_price       0
actual_price           0
discount_percentage    0
rating                 0
rating_count           0
about_product          0
user_id                0
user_name              0
review_id              0
review_title           0
review_content         0
img_link               0
product_link           0
dtype: int64
```

In [17]:
```
df['rating_count']=df['rating_count'].replace(',','',regex=True).astype(int)
```

In [18]:
```
df.head()
```

Out[18]:

	product_id	product_name	category	discounted_price
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	399.0
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	199.0
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	Computers&Accessories|Accessories&Peripherals|...	199.0
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	Computers&Accessories|Accessories&Peripherals|...	329.0
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	Computers&Accessories|Accessories&Peripherals|...	154.0

In [19]:
```
df['rating'].unique()
```

Out[19]:
```
array(['4.2', '4.0', '3.9', '4.1', '4.3', '4.4', '4.5', '3.7', '3.3',
       '3.6', '3.4', '3.8', '3.5', '4.6', '3.2', '5.0', '4.7', '3.0',
       '2.8', '4', '3.1', '4.8', '2.3', '|', '2', '3', '2.6', '2.9'],
      dtype=object)
```

In [20]:
```
df['rating'] = pd.to_numeric(df['rating'], errors='coerce')
```

In [21]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1463 entries, 0 to 1464
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   product_id           1463 non-null   object 
 1   product_name         1463 non-null   object 
 2   category             1463 non-null   object 
 3   discounted_price     1463 non-null   float64
 4   actual_price         1463 non-null   float64
 5   discount_percentage  1463 non-null   float64
 6   rating               1462 non-null   float64
 7   rating_count         1463 non-null   int64  
 8   about_product        1463 non-null   object 
 9   user_id              1463 non-null   object 
 10  user_name            1463 non-null   object 
 11  review_id            1463 non-null   object 
 12  review_title         1463 non-null   object 
 13  review_content       1463 non-null   object 
 14  img_link             1463 non-null   object 
 15  product_link         1463 non-null   object 
dtypes: float64(4), int64(1), object(11)
memory usage: 194.3+ KB
```

In [22]:
```
df['rating'].unique()
```

Out[22]:
```
array([4.2, 4. , 3.9, 4.1, 4.3, 4.4, 4.5, 3.7, 3.3, 3.6, 3.4, 3.8, 3.5,
       4.6, 3.2, 5. , 4.7, 3. , 2.8, 3.1, 4.8, 2.3, nan, 2. , 2.6, 2.9])
```

In [23]:
```
df=df.dropna(subset=['rating'])
```

In [24]:
```
df['rating']=df['rating'].astype(float)
```

In [25]:
```
# Derived Feature: Price Difference
df['price_difference'] = df['actual_price'] - df['discounted_price']

# Derived Feature: Value for money score (rating/price)
df['value_for_money_score'] = df['rating'] / (df['discounted_price'] + 1)  # +1 to

# Derived Feature: Weighted Rating
df['weighted_rating'] = df['rating'] * df['rating_count']
```

Section B: Exploratory Data Analysis: Think like a product analyst trying to identify buying patterns

Visualize: Most reviewed products Visualize top 10 categories by number of products. Average rating per category Discounts vs actual price correlation User Engagement Insights (5 marks) ○ Which products have high ratings but low review counts? ○ Are highly rated products also heavily reviewed? Create 3 actionable insights for Amazon's product strategy based on EDA.

In [26]:
```
df.head()
```

Out[26]:

	product_id	product_name	category	discounted_price
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	399.0
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	199.0
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	Computers&Accessories|Accessories&Peripherals|...	199.0
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	Computers&Accessories|Accessories&Peripherals|...	329.0
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	Computers&Accessories|Accessories&Peripherals|...	154.0

In [27]:
```
df.isnull().sum()
```

Out[27]:
```
product_id               0
product_name             0
category                 0
discounted_price         0
actual_price             0
discount_percentage      0
rating                   0
rating_count             0
about_product            0
user_id                  0
user_name                0
review_id                0
review_title             0
review_content           0
img_link                 0
product_link             0
price_difference         0
value_for_money_score    0
weighted_rating          0
dtype: int64
```

In [28]:
```
# Remove duplicates
df_cleaned = df.drop_duplicates()
```

In [29]:
```
# Drop rows with missing ratings or prices
df_cleaned = df_cleaned.dropna()
```

In [30]:
```
df_cleaned.head()
```

Out[30]:

	product_id	product_name	category	discounted_price
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	399.0
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	199.0
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	Computers&Accessories|Accessories&Peripherals|...	199.0
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	Computers&Accessories|Accessories&Peripherals|...	329.0
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	Computers&Accessories|Accessories&Peripherals|...	154.0

In [31]:
```
df_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1462 entries, 0 to 1464
Data columns (total 19 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   product_id           1462 non-null   object 
 1   product_name         1462 non-null   object 
 2   category             1462 non-null   object 
 3   discounted_price     1462 non-null   float64
 4   actual_price         1462 non-null   float64
 5   discount_percentage  1462 non-null   float64
 6   rating               1462 non-null   float64
 7   rating_count         1462 non-null   int64  
 8   about_product        1462 non-null   object 
 9   user_id              1462 non-null   object 
 10  user_name            1462 non-null   object 
 11  review_id            1462 non-null   object 
 12  review_title         1462 non-null   object 
 13  review_content       1462 non-null   object 
 14  img_link             1462 non-null   object 
 15  product_link         1462 non-null   object 
 16  price_difference     1462 non-null   float64
 17  value_for_money_score  1462 non-null   float64
 18  weighted_rating      1462 non-null   float64
dtypes: float64(7), int64(1), object(11)
memory usage: 228.4+ KB
```

In [32]:
```
df_cleaned.describe()
```

Out[32]:

	discounted_price	actual_price	discount_percentage	rating	rating_count	price_dif
count	1462.000000	1462.000000	1462.000000	1462.000000	1462.000000	1462
mean	3129.981826	5453.087743	47.672367	4.096717	18307.376881	2323
std	6950.548042	10884.467444	21.613905	0.289497	42766.096572	4608
min	39.000000	39.000000	0.000000	2.000000	2.000000	C
25%	325.000000	800.000000	32.000000	4.000000	1191.500000	37C
50%	799.000000	1670.000000	50.000000	4.100000	5179.000000	80C
75%	1999.000000	4321.250000	63.000000	4.300000	17342.250000	1959
max	77990.000000	139900.000000	94.000000	5.000000	426973.000000	6191C

Most reviewed products Visualize top 10 categories by number of products.

In [33]:
```
# Group by product_id and keep the most common name and max rating count
most_reviewed = (
    df_cleaned.groupby('product_id')
     .agg({
         'product_name': lambda x:x.mode().iloc[0],
         'rating_count': 'max'
     })
     .reset_index()
     .sort_values(by='rating_count', ascending=False)
)

most_reviewed.head(10)
```

Out[33]:

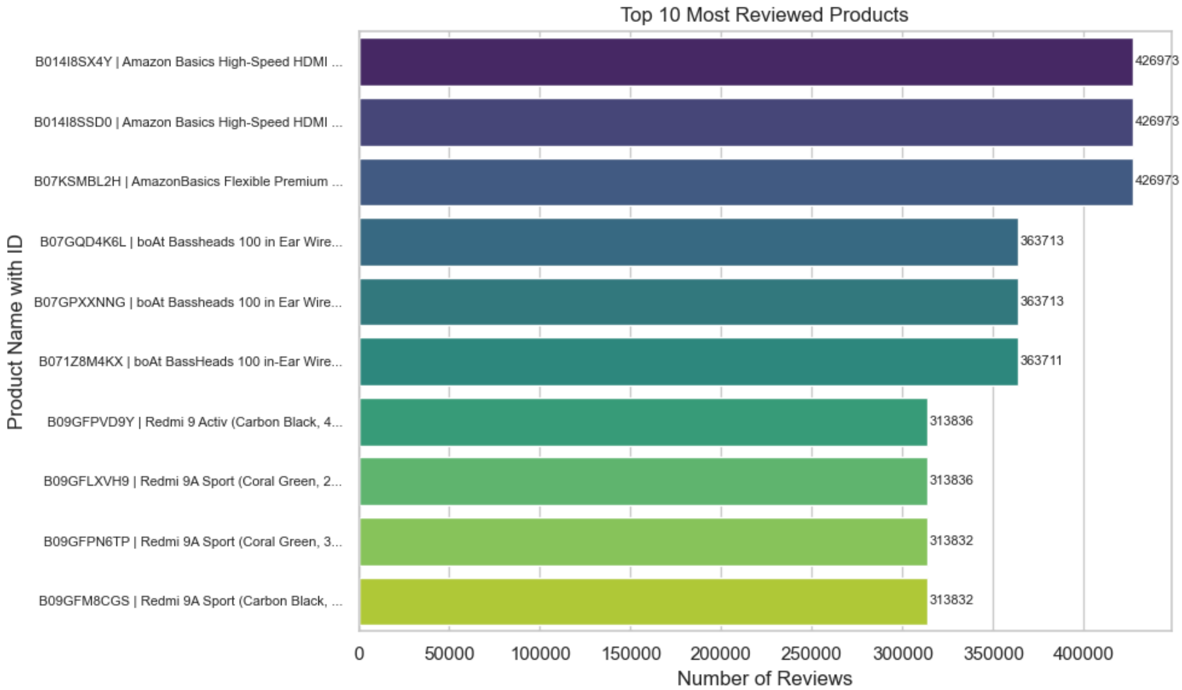
	product_id	product_name	rating_count
138	B014I8SX4Y	Amazon Basics High-Speed HDMI Cable, 6 Feet (2...	426973
137	B014I8SSD0	Amazon Basics High-Speed HDMI Cable, 6 Feet - ...	426973
356	B07KSMBL2H	AmazonBasics Flexible Premium HDMI Cable (Blac...	426973
318	B07GQD4K6L	boAt Bassheads 100 in Ear Wired Earphones with...	363713
317	B07GPXXNNG	boAt Bassheads 100 in Ear Wired Earphones with...	363713
232	B071Z8M4KX	boAt BassHeads 100 in-Ear Wired Headphones wit...	363711
910	B09GFPVD9Y	Redmi 9 Activ (Carbon Black, 4GB RAM, 64GB Sto...	313836
906	B09GFLXVH9	Redmi 9A Sport (Coral Green, 2GB RAM, 32GB Sto...	313836
909	B09GFPN6TP	Redmi 9A Sport (Coral Green, 3GB RAM, 32GB Sto...	313832
907	B09GFM8CGS	Redmi 9A Sport (Carbon Black, 2GB RAM, 32GB St...	313832

In [34]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [35]:
```
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
```

```
sns.barplot(data=most_reviewed.head(10), y=most_reviewed['product_id'].head(10)+' |

for i, row in enumerate(most_reviewed.head(10).itertuples()):
    plt.text(row.rating_count + 1000, i, int(row.rating_count), va='center', fontsi

plt.title("Top 10 Most Reviewed Products")
plt.xlabel("Number of Reviews")
plt.ylabel("Product Name with ID")
plt.yticks(fontsize=8)
plt.tight_layout()
plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_8296\3790122290.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(data=most_reviewed.head(10), y=most_reviewed['product_id'].head(10)
+' | '+most_reviewed['product_name'].head(10).str.slice(0,30)+'...', x='rating_cou
nt', palette='viridis')
```



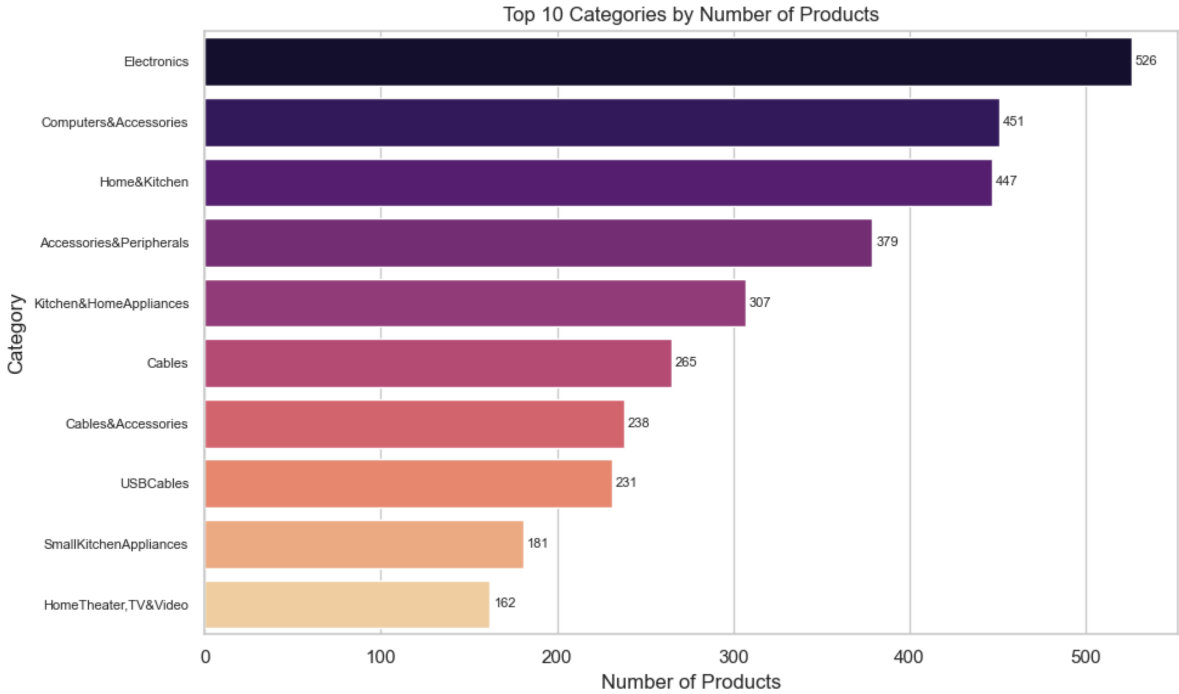
In [36]:
```
top_10_categories=df_cleaned['category'].str.split('|').explode().value_counts().he
print(f'top_10_categories: \n {top_10_categories}')
```

```
top_10_categories:
 category
Electronics                526
Computers&Accessories      451
Home&Kitchen               447
Accessories&Peripherals    379
Kitchen&HomeAppliances     307
Cables                     265
Cables&Accessories         238
USBCables                  231
SmallKitchenAppliances     181
HomeTheater,TV&Video       162
Name: count, dtype: int64
```

In [37]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x=top_10_categories.values, y=top_10_categories.index, palette='magma')

for i,value in enumerate(top_10_categories):
```

```
    plt.text(value+2,i,str(value),va='center',fontsize=8)


plt.title("Top 10 Categories by Number of Products")
plt.xlabel("Number of Products")
plt.ylabel("Category")
plt.yticks(fontsize=8)
plt.tight_layout()
plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_8296\257495889.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=top_10_categories.values, y=top_10_categories.index, palette='magm
a')
```



Average rating per category

In [38]:
```
# Explode category into separate rows
df_exploded = df_cleaned.copy()
df_exploded['category'] = df_exploded['category'].str.split('|')
df_exploded = df_exploded.explode('category')
```

In [39]:
```
df_exploded.shape
```

Out[39]:
```
(6291, 19)
```

In [40]:
```
df_exploded.head()
```

Out[40]:

	product_id	product_name	category	discounted_price	actual_price	discount_per
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories	399.0	1099.0	
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Accessories&Peripherals	399.0	1099.0	
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Cables&Accessories	399.0	1099.0	
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Cables	399.0	1099.0	
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	USBCables	399.0	1099.0	

In [41]:
```
top_10_categories_new = df_exploded['category'].value_counts().head(10).index
top_10_categories_new
```

Out[41]:
```
Index(['Electronics', 'Computers&Accessories', 'Home&Kitchen',
       'Accessories&Peripherals', 'Kitchen&HomeAppliances', 'Cables',
       'Cables&Accessories', 'USBCables', 'SmallKitchenAppliances',
       'HomeTheater,TV&Video'],
      dtype='object', name='category')
```

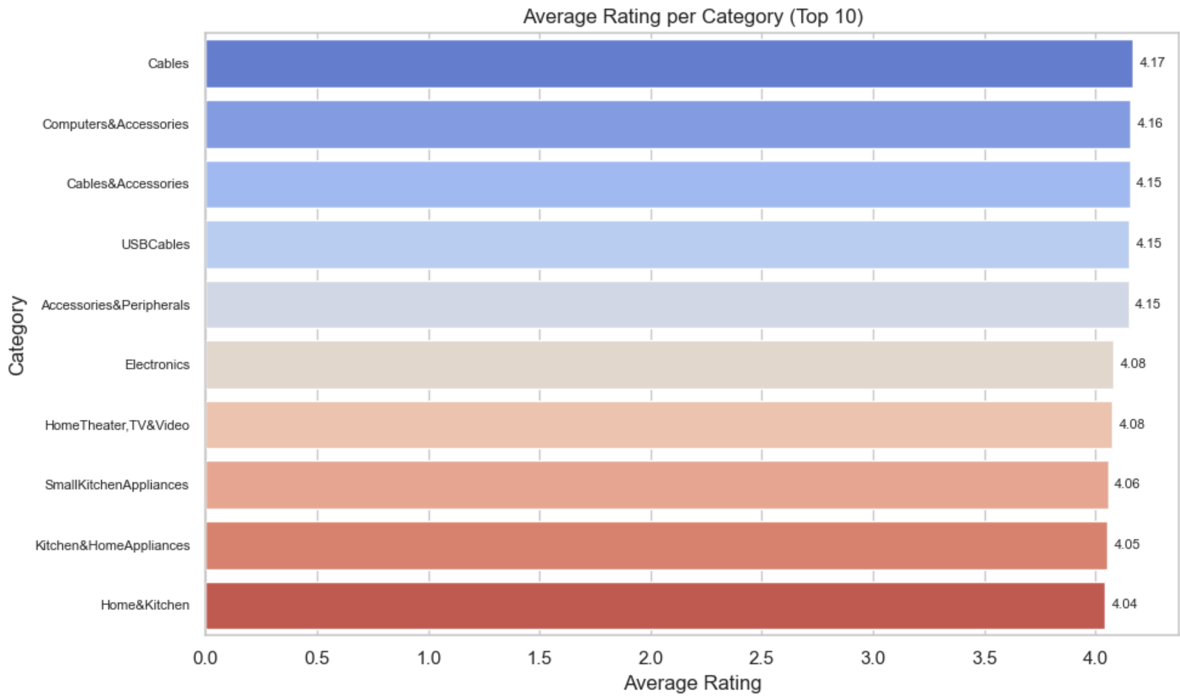
In [42]:
```
# Get average rating for top 10 most frequent categories
avg_rating_per_cat = df_exploded[df_exploded['category'].isin(top_10_categories_nev
avg_rating_per_cat
```

Out[42]:
```
category
Cables                      4.166038
Computers&Accessories       4.155654
Cables&Accessories          4.153782
USBCables                   4.153247
Accessories&Peripherals     4.149340
Electronics                 4.081749
HomeTheater,TV&Video        4.075309
SmallKitchenAppliances      4.056354
Kitchen&HomeAppliances      4.053420
Home&Kitchen                4.040716
Name: rating, dtype: float64
```

In [43]:
```
# Plot
plt.figure(figsize=(10, 6))
sns.barplot(x=avg_rating_per_cat.values, y=avg_rating_per_cat.index, palette='coolw

for i,value in enumerate(avg_rating_per_cat):
```

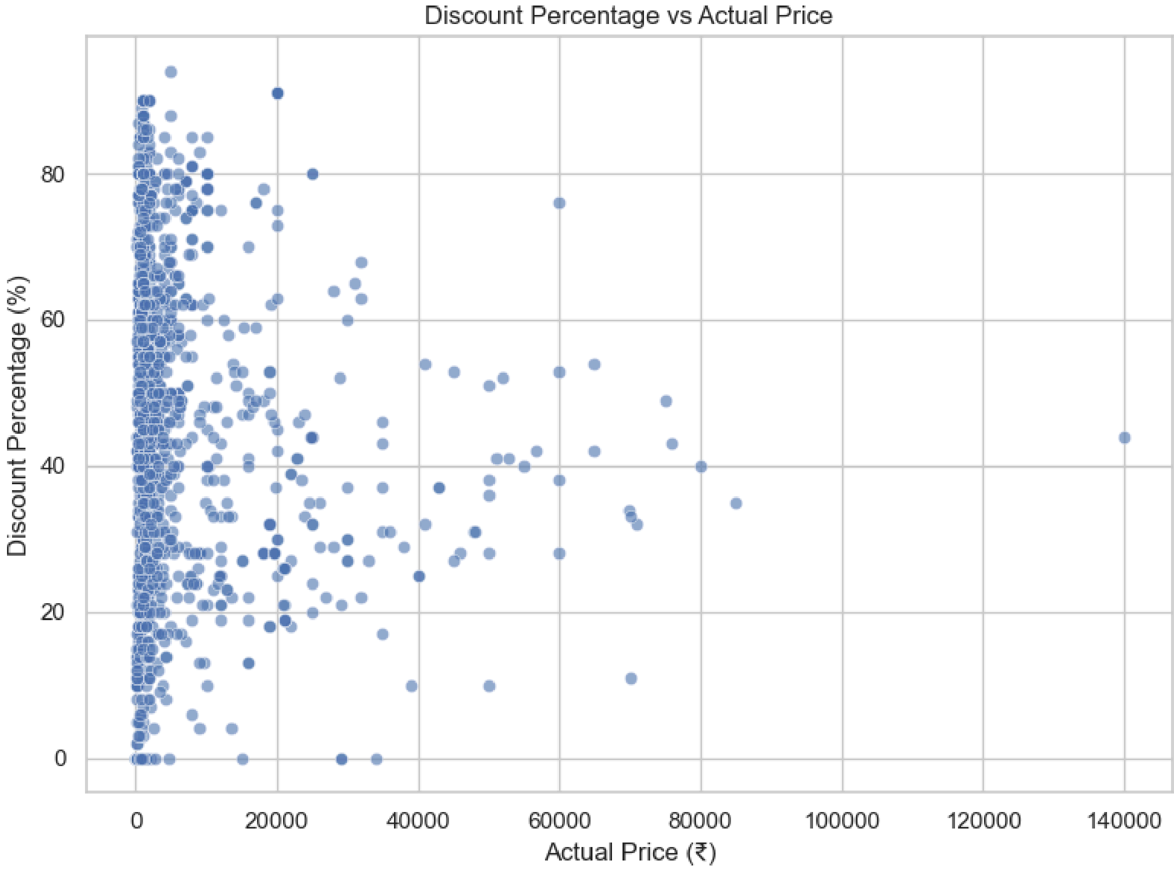
```
    plt.text(value+0.03,i,str(round(value,2)),va='center',fontsize=8)

plt.title("Average Rating per Category (Top 10)")
plt.xlabel("Average Rating")
plt.ylabel("Category")
plt.yticks(fontsize=8)
plt.tight_layout()
plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_8296\3028119629.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=avg_rating_per_cat.values, y=avg_rating_per_cat.index, palette='co
olwarm')
```



Discounts vs actual price correlation

In [44]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df_cleaned, x='actual_price', y='discount_percentage', alpha=0
plt.title("Discount Percentage vs Actual Price")
plt.xlabel("Actual Price (₹)")
plt.ylabel("Discount Percentage (%)")
plt.tight_layout()
plt.show()
```



In [45]:
```
correlation = df_cleaned['actual_price'].corr(df_cleaned['discount_percentage'])
print(f"Correlation between actual price and discount: {correlation:.2f}")
```

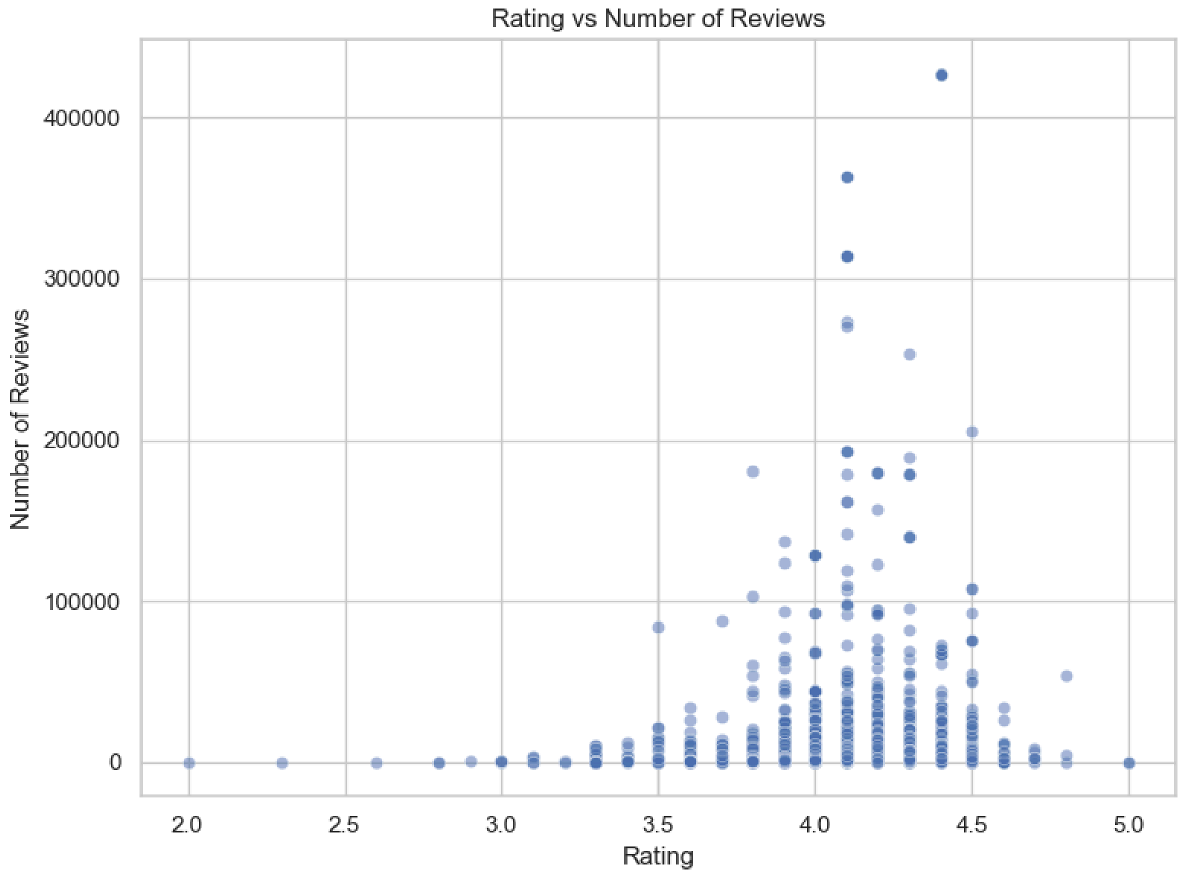
```
Correlation between actual price and discount: -0.12
```

In [46]:
```
high_rating_low_reviews = df_cleaned[(df_cleaned['rating'] >= 4.5) & (df_cleaned['r
high_rating_low_reviews[['product_id', 'product_name', 'rating', 'rating_count']].h
```

Out[46]:

	product_id	product_name	rating	rating_count
174	B0BP7XLX48	Syncwire LTG to USB Cable for Fast Charging Co...	5.0	5
299	B0BNDD9TN6	WANBO X1 Pro (Upgraded) | Native 1080P Full HD...	4.5	7
547	B0BMM7R92G	Noise_Colorfit Smart Watch Charger 2 Pin USB F...	4.5	38
775	B09ZHCJDP1	Amazon Basics Wireless Mouse | 2.4 GHz Connect...	5.0	23
1158	B0BMTZ4T1D	!!1000 Watt/2000-Watt Room Heater!! Fan Heater...	4.5	11
1164	B08QW937WV	Homeistic Applience™ Instant Electric Water He...	4.5	19
1201	B0BQ3K23Y1	Oratech Coffee Frother electric, milk frother ...	4.8	28
1216	B0BN6M3TCM	VRPRIME Lint Roller Lint Remover for Clothes, ...	4.6	79
1226	B0BLC2BYPX	Zuvexa USB Rechargeable Electric Foam Maker - ...	4.7	54
1267	B0B694PXQJ	Gadgetronics Digital Kitchen Weighing Scale & ...	4.5	63

In [47]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df_cleaned, x='rating', y='rating_count', alpha=0.5)
plt.title("Rating vs Number of Reviews")
plt.xlabel("Rating")
plt.ylabel("Number of Reviews")
plt.tight_layout()
plt.show()
```



User Engagement Insights (5 marks)

○ Which products have high ratings but low review counts? ○ Are highly rated products also heavily reviewed? Create 3 actionable insights for Amazon's product strategy based on EDA.

🧠 Observation from the graph: There is a cluster of products around the 4.0–4.5 rating range that have high review counts, some exceeding 300,000+.

However, several 5.0-rated products have very low review counts (mostly under 100).

As rating increases beyond 4.5, number of reviews tends to drop.

The correlation is weak — most high ratings do not guarantee high review counts.

🔹 Actionable Insights for Amazon Based on the above EDA, here are 3 strategic recommendations:

✅ 1. Promote High-Rated but Under-Reviewed Products Several products have excellent ratings (4.5+) but very few reviews. These could be new or niche offerings. Promoting them through Amazon ads, deals, or homepage widgets can increase visibility and boost trust through more user feedback.

✅ 2. Optimize Discount Strategies by Price Segment The scatter plot indicates that mid-range products tend to receive higher discounts, but correlation with actual prices is weak (r = -0.12). Amazon should analyze conversion rates across price-discount bands to refine discounting strategies and maximize ROI.

✅ 3. Diversify Beyond Dominant Categories Categories like Electronics and Computers & Accessories dominate product listings and engagement. Amazon can explore expanding high-rated products in underrepresented categories (e.g., Small Kitchen Appliances, Health gadgets) to tap into unmet demand and niche markets.

✅ Conclusion: Not necessarily. While many heavily reviewed products have decent ratings (around 4.0–4.5), not all highly rated products are heavily reviewed. This is also supported by your correlation analysis and the list of high-rated, low-review products.

Section C: Content-Based Filtering: Act like a content engineer personalizing user feeds based on product metadata.

Vectorize product text (about_product + product_name) using: TF-IDF or embeddings Build a product similarity matrix Recommend top 5 similar products to: A new product with no reviews A product with high user dropout (bad ratings) Add category, price, and discount to enhance content vectors Evaluate recommendations: How diverse and relevant are the content-based results?

In [48]:
```
df_cleaned.head()
```

Out[48]:

	product_id	product_name	category	discounted_price
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	399.0
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	199.0
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	Computers&Accessories|Accessories&Peripherals|...	199.0
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	Computers&Accessories|Accessories&Peripherals|...	329.0
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	Computers&Accessories|Accessories&Peripherals|...	154.0

In [49]:
```
df_cleaned.shape
```

Out[49]: (1462, 19)

In [50]:
```
# After cleaning user_id and exploding (same as hybrid section):
df_cleaned['user_id'] = df_cleaned['user_id'].astype(str).str.split(',')
df_cleaned = df_cleaned.explode('user_id').reset_index(drop=True)

# Build unique product-level DataFrame
product_df = df_cleaned.drop_duplicates(subset='product_id').reset_index(drop=True)
```

Vectorize product text (about_product + product_name) using:

TF-IDF or embeddings

In [51]:
```
product_df['text']=product_df['product_name']+" "+product_df['about_product']
```

In [52]:
```
product_df.shape
```

Out[52]: (1348, 20)

In [53]:
```
product_df.head(2)
```

Out[53]:

	product_id	product_name	category	discounted_price	a
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	399.0	
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	199.0	

In [54]:
```
product_df.iloc[0,-1]
```

Out[54]:
```
"Wayona Nylon Braided USB to Lightning Fast Charging and Data Sync Cable Compatibl
e for iPhone 13, 12,11, X, 8, 7, 6, 5, iPad Air, Pro, Mini (3 FT Pack of 1, Grey)
High Compatibility : Compatible With iPhone 12, 11, X/XsMax/Xr ,iPhone 8/8 Plus,iP
hone 7/7 Plus,iPhone 6s/6s Plus,iPhone 6/6 Plus,iPhone 5/5s/5c/se,iPad Pro,iPad Ai
r 1/2,iPad mini 1/2/3,iPod nano7,iPod touch and more apple devices.|Fast Charge&Da
ta Sync : It can charge and sync simultaneously at a rapid speed, Compatible with
any charging adaptor, multi-port charging station or power bank.|Durability : Dura
ble nylon braided design with premium aluminum housing and toughened nylon fiber w
ound tightly around the cord lending it superior durability and adding a bit to it
s flexibility.|High Security Level : It is designed to fully protect your device f
rom damaging excessive current.Copper core thick+Multilayer shielding, Anti-interf
erence, Protective circuit equipment.|WARRANTY: 12 months warranty and friendly cu
stomer services, ensures the long-time enjoyment of your purchase. If you meet any
question or problem, please don't hesitate to contact us."
```

In [55]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [56]:
```
tfidf=TfidfVectorizer(stop_words='english',max_features=5000)
```

In [57]:
```
tfidf_matrix=tfidf.fit_transform(product_df['text'])
```

In [58]:
```
from sklearn.metrics.pairwise import cosine_similarity
```

Build a product similarity matrix

Recommend top 5 similar products to: A new product with no reviews A product with high user dropout (bad ratings)

In [59]:
```
similarity_matrix=cosine_similarity(tfidf_matrix)
```

In [60]:
```
# Define a new product
new_product_text = "High-quality braided USB Type-C cable for fast charging and dat

# Transform using existing TF-IDF vectorizer
new_vector = tfidf.transform([new_product_text])

# Compute similarity
similarities = cosine_similarity(new_vector, tfidf_matrix).flatten()

# Get top 5 most similar existing products
top_indices = similarities.argsort()[::-1][:5]
product_df.iloc[top_indices][['product_id', 'product_name', 'category', 'rating']]
```

Out[60]:

	product_id	product_name	category	rating
133	B09X79PP8F	MI 2-in-1 USB Type C Cable (Micro USB to Type ...	Computers&Accessories|Accessories&Peripherals|...	3.9
162	B08R69WBN7	Pinnaclz Original Combo of 2 USB Type C Fast C...	Computers&Accessories|Accessories&Peripherals|...	4.0
151	B08QSDKFGQ	Zoul USB Type C Fast Charging 3A Nylon Braided...	Computers&Accessories|Accessories&Peripherals|...	4.3
76	B09YLXYP7Y	Ambrane 60W / 3A Fast Charging Output Cable wi...	Computers&Accessories|Accessories&Peripherals|...	4.0
240	B09YLX91QR	Ambrane 60W / 3A Fast Charging Output Cable wi...	Computers&Accessories|Accessories&Peripherals|...	4.0

In [62]:
```
product_idx = product_df[product_df['rating'] < 2.5].index[0]
```

In [63]:
```
product_idx
```

Out[63]:
```
np.int64(1127)
```

In [64]:
```
similar_indices = similarity_matrix[product_idx].argsort()[::-1][1:6]
```

In [65]:
```
similar_indices
```

Out[65]:
```
array([1268, 1019, 1030, 1169, 1197])
```

In [66]:
```
product_df.iloc[similar_indices][['product_id', 'product_name']]
```

Out[66]:

	product_id	product_name
1268	B0BL3R4RGS	VAPJA® Portable Mini Juicer Cup Blender USB Re...
1019	B09NTHQRW3	InstaCuppa Portable Blender for Smoothie, Milk...
1030	B0B3G5XZN5	InstaCuppa Portable Blender for Smoothie, Milk...
1169	B0BNDGL26T	MR. BRAND Portable USB Juicer Electric USB Jui...
1197	B08TT63N58	ROYAL STEP - AMAZON'S BRAND - Portable Electri...

Add category, price, and discount to enhance content vectors

In [67]:
```
from sklearn.preprocessing import MinMaxScaler
from scipy.sparse import hstack
```

In [68]:
```
product_df.head()
```

Out[68]:

	product_id	product_name	category	discounted_price
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	399.0
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	199.0
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	Computers&Accessories|Accessories&Peripherals|...	199.0
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	Computers&Accessories|Accessories&Peripherals|...	329.0
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	Computers&Accessories|Accessories&Peripherals|...	154.0

In [69]:
```
product_df['cagegory_main']=product_df['category'].str.split('|').str[0]
```

In [70]:
```
product_df['cagegory_main'][:20]
```

Out[70]:
```
0     Computers&Accessories
1     Computers&Accessories
2     Computers&Accessories
3     Computers&Accessories
4     Computers&Accessories
5     Computers&Accessories
6     Computers&Accessories
7     Computers&Accessories
8     Computers&Accessories
9     Computers&Accessories
10    Computers&Accessories
11    Computers&Accessories
12              Electronics
13    Computers&Accessories
14    Computers&Accessories
15    Computers&Accessories
16              Electronics
17    Computers&Accessories
18    Computers&Accessories
19              Electronics
Name: cagegory_main, dtype: object
```

In [71]:
```
product_df['cagegory_main'].nunique()
```

Out[71]:
```
9
```

In [72]:
```
category_dummies=pd.get_dummies(product_df['cagegory_main'])
```

In [73]:
```
category_dummies
```

Out[73]:

	Car&Motorbike	Computers&Accessories	Electronics	Health&PersonalCare	Home&Kitchen
0	False	True	False	False	False
1	False	True	False	False	False
2	False	True	False	False	False
3	False	True	False	False	False
4	False	True	False	False	False
...	...	...	...	...	...
1343	False	False	False	False	True
1344	False	False	False	False	True
1345	False	False	False	False	True
1346	False	False	False	False	True
1347	False	False	False	False	True

1348 rows × 9 columns

In [74]:
```
scaler=MinMaxScaler()
```

In [75]:
```
scaled_numeric=scaler.fit_transform(product_df[['discounted_price', 'actual_price',
scaled_numeric
```

Out[75]:
```
array([[0.00461829, 0.00757895, 0.68085106],
       [0.00205257, 0.00221649, 0.45744681],
       [0.00205257, 0.01329892, 0.95744681],
       ...,
       [0.02796629, 0.02174302, 0.29787234],
       [0.01744686, 0.01323457, 0.27659574],
       [0.03622789, 0.02610449, 0.23404255]], shape=(1348, 3))
```

In [76]:
```
final_content_matrix=hstack([tfidf_matrix,category_dummies.values,scaled_numeric])
print(final_content_matrix)
```

```
<COOrdinate sparse matrix of dtype 'float64'
	with 95478 stored elements and shape (1348, 5012)>
  Coords	Values
  (0, 4847)	0.08473261665908756
  (0, 3122)	0.19460350550018818
  (0, 955)	0.12354553258344658
  (0, 4717)	0.03624642981180733
  (0, 2698)	0.06955698103157318
  (0, 1890)	0.08753359556106963
  (0, 1141)	0.13073254609350804
  (0, 1478)	0.10173460879855381
  (0, 4436)	0.16949557704824741
  (0, 1028)	0.041794932953361334
  (0, 1272)	0.11816213764678432
  (0, 2511)	0.43240936404206304
  (0, 78)	0.06811589589700887
  (0, 54)	0.15751317656947342
  (0, 47)	0.11327194142665345
  (0, 2507)	0.24634249082330717
  (0, 641)	0.11898157664733272
  (0, 3497)	0.11113416343403063
  (0, 2934)	0.1158543443805364
  (0, 2031)	0.07704277272233065
  (0, 3229)	0.06335129645686521
  (0, 2152)	0.05982203998832847
  (0, 2294)	0.07209232194678807
  (0, 1271)	0.054563014843109145
  (0, 4962)	0.10625138189188751
  :	:
  (1339, 5010)	0.008301098948241468
  (1339, 5011)	0.6276595744680851
  (1340, 5009)	0.03407268668779105
  (1340, 5010)	0.01899028320975826
  (1341, 5009)	0.011674000333542867
  (1341, 5010)	0.01615890062276117
  (1341, 5011)	0.6276595744680851
  (1342, 5009)	0.0020525714872163285
  (1342, 5010)	0.006863957786659612
  (1342, 5011)	0.851063829787234
  (1343, 5009)	0.004361714410334698
  (1343, 5010)	0.006291961304437978
  (1343, 5011)	0.6276595744680851
  (1344, 5009)	0.028748829392823697
  (1344, 5010)	0.021492767819477912
  (1344, 5011)	0.26595744680851063
  (1345, 5009)	0.02796628651332247
  (1345, 5010)	0.021743016280449876
  (1345, 5011)	0.2978723404255319
  (1346, 5009)	0.01744685764133879
  (1346, 5010)	0.013234568607403064
  (1346, 5011)	0.2765957446808511
  (1347, 5009)	0.0362278867493682
  (1347, 5010)	0.026104489457389836
  (1347, 5011)	0.23404255319148937
```

In [77]:
```
enhanced_similarity = cosine_similarity(final_content_matrix)
enhanced_similarity
```

Out[77]:
```
array([[1.        , 0.62012474, 0.74066755, ..., 0.10302348, 0.10268728,
        0.07197754],
       [0.62012474, 1.        , 0.63001782, ..., 0.06346171, 0.06598404,
        0.06273694],
       [0.74066755, 0.63001782, 1.        , ..., 0.12486893, 0.12536506,
        0.09165858],
       ...,
       [0.10302348, 0.06346171, 0.12486893, ..., 1.        , 0.59883746,
        0.52392676],
       [0.10268728, 0.06598404, 0.12536506, ..., 0.59883746, 1.        ,
        0.53161204],
       [0.07197754, 0.06273694, 0.09165858, ..., 0.52392676, 0.53161204,
        1.        ]], shape=(1348, 1348))
```

Evaluate recommendations:

How diverse and relevant are the content-based results?

In [78]:
```python
# For product at index i
def get_similar_products(i, sim_matrix):
    indices = sim_matrix[i].argsort()[::-1][1:6]
    return product_df.iloc[indices][['product_id', 'product_name', 'category', 'rat
```

In [79]:
```python
get_similar_products(product_idx, enhanced_similarity)
```

Out[79]:

	product_id	product_name	category	rating
1268	B0BL3R4RGS	VAPJA® Portable Mini Juicer Cup Blender USB Re...	Home&Kitchen|Kitchen&HomeAppliances|SmallKitch...	3.6
1169	B0BNDGL26T	MR. BRAND Portable USB Juicer Electric USB Jui...	Home&Kitchen|Kitchen&HomeAppliances|SmallKitch...	2.8
1197	B08TT63N58	ROYAL STEP - AMAZON'S BRAND - Portable Electri...	Home&Kitchen|Kitchen&HomeAppliances|SmallKitch...	3.1
1095	B0BNQMF152	ROYAL STEP Portable Electric USB Juice Maker J...	Home&Kitchen|Kitchen&HomeAppliances|SmallKitch...	3.7
1279	B0BHNHMR3H	LONAXA Mini Travel Rechargeable Fruit Juicer -...	Home&Kitchen|Kitchen&HomeAppliances|SmallKitch...	3.9

Section D: Collaborative Filtering (User–Item) : Now you're a machine learning engineer building smart recommendations using user behavior.

Create a user-item matrix using user_id, product_id, and rating.

In [80]:
```
user_item_matrix=product_df.pivot_table(index='user_id',columns='product_id',values
user_item_matrix
```

Out[80]:

product_id user_id	B002PD61Y4	B002SZEOLG	B003B00484	B003L62T7W	B(
AE22Y3KIS7SE6LI3HE2VS6WWPU4Q	0.0	0.0	0.0	0.0	
AE23RS3W7GZO7LHYKJU6KSKVM4MQ	0.0	0.0	0.0	0.0	
AE242TR3GQ6TYC6W4SJ5UYYKBTYQ	0.0	0.0	0.0	0.0	
AE27UOZENYSWCQVQRRUQIV2ZM7VA	0.0	0.0	0.0	0.0	
AE2JTMRKTUOIVIZWS2WDGTMNTU4Q	0.0	0.0	0.0	0.0	
...	...	...	...	...	
AHZFKWGDBRQKNMNQ4ZPL52OZBRKA	0.0	0.0	0.0	0.0	
AHZJHJWFZLYD64GVP4PXVI2F4LXA	0.0	0.0	0.0	0.0	
AHZNSNBVKQR4OGJAQHE4DCDA4YHA	0.0	0.0	0.0	0.0	
AHZWJCVEIEI76H2VGMUSN5D735IQ	0.0	0.0	0.0	0.0	
AHZWXUWE3RGLDH4JJUK3HT3VMBJA	0.0	0.0	0.0	0.0	

1164 rows × 1348 columns

In [81]:
```
item_similarity=cosine_similarity(user_item_matrix.T)
item_similarity
```

Out[81]:
```
array([[1., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 0., 1.]], shape=(1348, 1348))
```

In [82]:
```
item_similarity_df=pd.DataFrame(item_similarity,index=user_item_matrix.columns,colu
item_similarity_df
```

Out[82]:

product_id	B002PD61Y4	B002SZEOLG	B003B00484	B003L62T7W	B004IO5BMQ	B005FYNT3G
product_id						
B002PD61Y4	1.0	0.0	0.0	0.0	0.0	0.0
B002SZEOLG	0.0	1.0	0.0	0.0	0.0	0.0
B003B00484	0.0	0.0	1.0	0.0	0.0	0.0
B003L62T7W	0.0	0.0	0.0	1.0	0.0	0.0
B004IO5BMQ	0.0	0.0	0.0	0.0	1.0	0.0
...	...	...	...	...	...	...
B0BPBXNQQT	0.0	0.0	0.0	0.0	0.0	0.0
B0BPCJM7TB	0.0	0.0	0.0	0.0	0.0	0.0
B0BPJBTB3F	0.0	0.0	0.0	0.0	0.0	0.0
B0BQ3K23Y1	0.0	0.0	0.0	0.0	0.0	0.0
B0BR4F878Q	0.0	0.0	0.0	0.0	0.0	0.0

1348 rows × 1348 columns

In [84]:
```
user_item_matrix.loc['AHZFKWGDBRQKNMNQ4ZPL52OZBRKA']
```

Out[84]:
```
product_id
B002PD61Y4    0.0
B002SZEOLG    0.0
B003B00484    0.0
B003L62T7W    0.0
B004IO5BMQ    0.0
             ...
B0BPBXNQQT    0.0
B0BPCJM7TB    0.0
B0BPJBTB3F    0.0
B0BQ3K23Y1    0.0
B0BR4F878Q    0.0
Name: AHZFKWGDBRQKNMNQ4ZPL52OZBRKA, Length: 1348, dtype: float64
```

Apply: User-User Collaborative Filtering (cosine similarity) OR Item-Item Collaborative Filtering (cosine or Pearson)

Recommend top 5 unseen products per user

In [85]:
```
def recommend_items_for_user(user_id):
    if user_id not in user_item_matrix.index:
        print(f"User {user_id} not found.")
        return []
    user_ratings=user_item_matrix.loc[user_id]
    rated_items=user_ratings[user_ratings>0].index

    if len(rated_items)==0:
        print("No rated products for this user (cold-start).")
        return []
```

```
    scores=pd.Series(0,index=user_item_matrix.columns)
    for item in rated_items:
        scores+=item_similarity_df[item]*user_ratings[item]

    scores=scores.drop(labels=rated_items)
    top_recommendations=scores.sort_values(ascending=False).head(5)

    return product_df[product_df['product_id'].isin(top_recommendations.index)][['p
```

In [86]:
```
# Example Use Case ---
example_user = product_df['user_id'].iloc[0]
print(f"\n Top Recommendations for User {example_user} (Collaborative Filtering):")
print(recommend_items_for_user(example_user))
```

```
🔍 Top Recommendations for User AG3D6O4STAQKAY2UVGEUV46KN35Q (Collaborative Filter
ing):
      product_id                                       product_name  rating
46    B002PD61Y4  D-Link DWA-131 300 Mbps Wireless Nano USB Adap...     4.1
143   B002SZEOLG  TP-Link Nano USB WiFi Dongle 150Mbps High Gain...     4.2
654   B003B00484  Duracell Plus AAA Rechargeable Batteries (750 ...     4.3
655   B003L62T7W  Logitech B100 Wired USB Mouse, 3 yr Warranty, ...     4.3
1029  B0BR4F878Q  Swiffer Instant Electric Water Heater Faucet T...     4.8
```

Section E: Hybrid Recommender (Content + Collaborative) : Step into the role of a senior ML engineer combining models for better performance.

In [87]:
```
def get_cf_scores(user_id):
    user_ratings = user_item_matrix.loc[user_id]
    rated_items = user_ratings[user_ratings > 0].index

    scores = pd.Series(0, index=user_item_matrix.columns)
    for item in rated_items:
        scores += item_similarity_df[item] * user_ratings[item]

    scores = scores.drop(labels=rated_items)
    return scores
```

Design a hybrid strategy:

Score fusion: 0.6 *CF_score + 0.4* Content_score

In [94]:
```
# final_content_matrix (TF-IDF + category + scaled numeric features) is already bui
content_similarity_df = pd.DataFrame(enhanced_similarity, index=product_df['product
```

In [89]:
```
def get_content_scores(product_id):
    if product_id not in content_similarity_df.index:
        return pd.Series(dtype=float)
    return content_similarity_df[product_id].drop(product_id)
```

In [90]:
```
def recommend_by_content_only(top_n):
    top_scores=content_similarity_df.mean(axis=1).sort_values(ascending=False).head
    return product_df[product_df['product_id'].isin(top_socres.index)][['product_id
```

In [91]:
```
def get_hybrid_recommendations(user_id,top_n=5):
    if user_id not in user_item_matrix.index:
        print("Cold-start user. Using content-based fallback.")
        return recommend_by_content_only(top_n)

    user_ratings = user_item_matrix.loc[user_id]
    rated_items = user_ratings[user_ratings > 0].index

    if len(rated_items) == 0:
        print("Cold-start user. Using content-based fallback.")
        return recommend_by_content_only(top_n)

    anchor_product=rated_items[-1]
    cbf_scores = get_content_scores(anchor_product)
    cf_scores = get_cf_scores(user_id)

    common_index = cbf_scores.index.intersection(cf_scores.index)
    if len(common_index) == 0:
        return recommend_by_content_only(top_n)

    content_scores = MinMaxScaler().fit_transform(cbf_scores[common_index].values.r
    cf_scores = MinMaxScaler().fit_transform(cf_scores[common_index].values.reshape

    hybrid_score = 0.6 * cf_scores + 0.4 * content_scores

    hybrid_df = pd.DataFrame({'product_id': common_index, 'hybrid_score': hybrid_sc
    top_hybrid = hybrid_df.sort_values(by='hybrid_score', ascending=False).head(top

    return product_df[product_df['product_id'].isin(top_hybrid['product_id'])][['pr
```

In [93]:
```
example_user = product_df['user_id'].iloc[0]
print(f"\n Hybrid Recommendations for User: {example_user}")
print(get_hybrid_recommendations(example_user))
```

```
 Hybrid Recommendations for User: AG3D6O4STAQKAY2UVGEUV46KN35Q
     product_id                                       product_name  rating
104  B07JNVF678  Wayona Nylon Braided USB Data Sync and Fast Ch...     4.2
166  B07JPJJZ2H  Wayona Nylon Braided Lightning USB Data Sync &...     4.2
174  B0BP7XLX48  Syncwire LTG to USB Cable for Fast Charging Co...     5.0
208  B095244Q22  MYVN LTG to USB for Fast Charging & Data Sync ...     3.7
261  B07F1P8KNV  Wayona Nylon Braided Usb Type C 3Ft 1M 3A Fast...     4.2
```

Compare recommendation quality of hybrid vs individual methods.

Evaluate hybrid system on: A cold-start product (new product) A cold-start user (few reviews) Suggest how to improve hybrid performance further using real-world constraints like: Popularity Recent purchases Product availability

🔁 Comparison of Hybrid vs Individual Methods

Content-Based Filtering (CBF):

Recommends products similar to those the user has liked, based on product features (text, category, etc.).

Works well for cold-start products (newly added), since it uses metadata.

Personalization is limited to user preferences inferred from product features.

Collaborative Filtering (CF):

Recommends based on user behavior — “users who bought X also bought Y”.

Provides strong personalization by leveraging peer behavior.

Fails in cold-start scenarios (for new users or products), as it needs sufficient data.

Hybrid Recommender:

Combines the strengths of both approaches: user behavior + product features.

Uses a weighted fusion (e.g., 60% CF, 40% Content).

Handles cold-start cases better than CF alone and improves accuracy over content-only methods.

Produces more diverse and relevant recommendations across different user types.

🧊 Evaluation on Cold-Start Scenarios 🧍 Cold-Start User (Very Few Reviews): A user with no or few prior interactions cannot benefit from collaborative filtering alone.

The hybrid system falls back on content-based filtering, using product similarity and metadata.

This ensures recommendations are still contextually relevant, even for new users.

📦 Cold-Start Product (Newly Added): Since the product has no interactions yet, CF cannot recommend it.

However, the hybrid model uses feature similarity from metadata (e.g., product name, brand, type) to recommend it to users who liked similar products.

This enables visibility of new or niche items in user recommendations.

💡 How to Improve Hybrid Performance Further Popularity Signals:

Integrating product popularity (like number of ratings or purchases) ensures trusted and widely used products get slightly higher visibility.

This helps prevent over-recommendation of very niche items that may not convert well.

Recency / Temporal Weighting:

Giving more importance to recent interactions captures user intent more accurately.

For example, if a user recently searched for mobile accessories, those should be prioritized over older preferences.

Product Availability Filtering:

Ensure only in-stock and purchasable products are recommended.

Recommending unavailable items creates a poor user experience and may reduce trust.

Dynamic Weight Fusion:

Instead of fixed weights (e.g., 0.6 CF + 0.4 Content), we can learn the optimal blend based on performance or context.

For example:

Use more CF when user history is rich.

Use more content when product metadata is strong or for cold-start users.

Diversity Boosting / Exploration:

Add a mechanism to introduce occasional diverse or serendipitous recommendations.

This avoids “echo chamber” effects and helps users discover new products they may not have considered.

In [95]:
```
product_df.head()
```

Out[95]:

	product_id	product_name	category	discounted_price
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	399.0
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	199.0
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	Computers&Accessories|Accessories&Peripherals|...	199.0
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	Computers&Accessories|Accessories&Peripherals|...	329.0
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	Computers&Accessories|Accessories&Peripherals|...	154.0

5 rows × 21 columns

Section F: Bonus: Business Strategy & Deployment

✅ 1. Which model works best for new users? Best Model: Content-Based Filtering (CBF)

Reason: New users have no historical interaction data, so collaborative filtering cannot make any meaningful suggestions.

Content-based filtering uses product metadata (like product name, category, price) to make relevant recommendations based on similarities to popular or trending items.

In your system, this is handled through a fallback to content-based recommendations when user history is missing.

✅ 2. Which model works best for returning users? Best Model: Hybrid (CBF + Collaborative Filtering)

Returning users have past behavior (ratings or purchases), which makes collaborative filtering highly effective.

A hybrid approach (e.g., 60% CF + 40% CBF) works best by:

Leveraging peer-based recommendations (CF).

Enhancing personalization with product feature understanding (CBF).

It provides the most accurate and personalized recommendations for loyal or repeat users.

✅ 3. How can we recommend products with no ratings? Approach: Use Content-Based Filtering + Popularity Boost

CBF handles unrated products well by analyzing product metadata (description, category, brand, etc.) and matching it with users' interests.

Boost visibility by combining with:

Popularity signals (e.g., views, wishlists, clicks).

Cold-launch promotions (like "New Arrivals You May Like").

In production, you might also use product embeddings (e.g., via BERT or product2vec) to understand similarity beyond just TF-IDF.

✅ 4. How would you deploy this system in production? (Tools/Technologies) Deployment Stack (Simplified):

Component Technology Model Training Python (scikit-learn, pandas, numpy), Jupyter Model Storage MLflow or joblib/pickle API Layer FastAPI or Flask (for real-time serving) Backend Server Dockerized service deployed via Kubernetes or AWS ECS Database PostgreSQL / MySQL (for user & product metadata), Redis (for caching top-N recommendations) Cloud Infrastructure AWS/GCP/Azure – EC2 for compute, S3 for data storage Monitoring Prometheus + Grafana, AWS CloudWatch CI/CD GitHub Actions, Jenkins, or AWS CodePipeline AB Testing Optimizely / LaunchDarkly / in-house experimentation framework

✅ 5. What KPIs should Amazon track to measure success? Here are key performance indicators (KPIs) to track recommendation effectiveness and business impact:

📈 Engagement Metrics Click-Through Rate (CTR): % of users who clicked a recommended item.

Conversion Rate: % of users who purchased a recommended product.

Time Spent on Product Pages: Higher time may indicate interest.

🔁 Retention & Personalization Repeat Purchase Rate: Are users returning and buying again?

Diversity & Novelty Scores: Are users getting a mix of items or just popular ones?

💰 Revenue Metrics Average Order Value (AOV): Does recommendation increase order size?

Revenue per User (RPU): Is there a monetary gain from using recommendations?

⚙️ System Health & Model Performance Latency: Time taken to serve recommendations.

Recommendation Coverage: % of users/products receiving recommendations.

Cold Start Accuracy: Relevance for new users/products.

```
In [ ]:
```