Amazon Sales Analysis

Objective

The purpose of this project is to conduct exploratory data analysis and determine which category of products has the highest revenue, which category offers the most discounts, which category has the highest customer engagement, and other insights.

I. Data Preparation

1. Import Libraries

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.filterwarnings('ignore')
```

2. Load the data set

In [6]:
```
df_amazon = pd.read_csv('amazon.csv')
```

In [7]:
```
df_amazon.head(5)
```

Out[7]:

	product_id	product_name	category	discounted_price	actual_price	discount_percentage	rating	rating_count	about_product	
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	₹399	₹1,099	64%	4.2	24,269	High Compatibility : Compatible With iPhone 12...	AG3D6O4STAQKAY2UVGEUV46KN35Q,AHM
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	₹199	₹349	43%	4	43,994	Compatible with all Type C enabled devices, be...	AECPFYFQVRUWC3KGNLJIOREFP5LQ,AGY
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	Computers&Accessories|Accessories&Peripherals|...	₹199	₹1,899	90%	3.9	7,928	【 Fast Charger& Data Sync】-With built-in safet...	AGU3BBQ2V2DDAMOAKGFAWDDQ6QHA,AE
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	Computers&Accessories|Accessories&Peripherals|...	₹329	₹699	53%	4.2	94,363	The boAt Deuce USB 300 2 in 1 cable is compati...	AEWAZDZZJLQUYVOVGBEUKSLXHQ5A,AG
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	Computers&Accessories|Accessories&Peripherals|...	₹154	₹399	61%	4.2	16,905	[CHARGE & SYNC FUNCTION]- This cable comes wit...	AE3Q6KSUK5P75D5HFYHCRAOLODSA,AFU

3. Checking for null values and data type of each columns

In [9]:
```
df_amazon.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
```

4. Changing Data type of price, percentage & rating related columns to float data type (float data type to provide precision)

4.1. Removing symbols before changing the data types

1. Dealing with discounted_price column

In [13]:
```
df_amazon['discounted_price']
```

Out[13]:
```
0         ₹399
1         ₹199
2         ₹199
3         ₹329
4         ₹154
         ...
1460      ₹379
1461    ₹2,280
1462    ₹2,219
1463    ₹1,399
1464    ₹2,863
Name: discounted_price, Length: 1465, dtype: object
```

In [14]:
```
df_amazon['discounted_price'] = df_amazon['discounted_price'].str.replace('₹','').str.replace(',','')
```

In [15]:
```
df_amazon['discounted_price'] = df_amazon['discounted_price'].astype('float64')
```

In [16]:
```
df_amazon['discounted_price']
```

Out[16]:
```
0        399.0
1        199.0
2        199.0
3        329.0
4        154.0
         ...
1460     379.0
1461    2280.0
1462    2219.0
1463    1399.0
1464    2863.0
Name: discounted_price, Length: 1465, dtype: float64
```

2. Dealing with actual_price column

In [18]:
```
df_amazon['actual_price']
```

Out[18]:
```
0       ₹1,099
1         ₹349
2       ₹1,899
3         ₹699
4         ₹399
         ...
1460      ₹919
1461    ₹3,045
1462    ₹3,080
1463    ₹1,890
1464    ₹3,690
Name: actual_price, Length: 1465, dtype: object
```

In [19]:
```
df_amazon['actual_price'] = df_amazon['actual_price'].str.replace('₹','').str.replace(',','')
```

In [20]:
```
df_amazon['actual_price'] = df_amazon['actual_price'].astype('float64')
```

In [21]:
```
df_amazon['actual_price']
```

Out[21]:
```
0       1099.0
1        349.0
2       1899.0
3        699.0
4        399.0
         ...
1460     919.0
1461    3045.0
1462    3080.0
1463    1890.0
1464    3690.0
Name: actual_price, Length: 1465, dtype: float64
```

3. Dealing with discounted_percentage column

In [23]:
```
df_amazon['discount_percentage']
```

Out[23]:
```
0       64%
1       43%
2       90%
3       53%
4       61%
       ...
1460    59%
1461    25%
1462    28%
1463    26%
1464    22%
Name: discount_percentage, Length: 1465, dtype: object
```

In [24]:
```
df_amazon['discount_percentage'] = df_amazon['discount_percentage'].str.replace('%','')
```

In [25]:
```
df_amazon['discount_percentage'] = df_amazon['discount_percentage'].astype('float64')
```

In [26]:
```
df_amazon['discount_percentage']
```

Out[26]:
```
0       64.0
1       43.0
2       90.0
3       53.0
4       61.0
        ...
1460    59.0
1461    25.0
1462    28.0
1463    26.0
1464    22.0
Name: discount_percentage, Length: 1465, dtype: float64
```

4. Dealing with rating column

In [28]:
```
# df_amazon['rating'] = df_amazon['rating'].astype('float64')
# The above line is giving error -> ValueError: could not convert string to float: '|'
```

In [29]:
```
# Checking for distinct values in ratings column
df_amazon['rating'].unique()
```

Out[29]:
```
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', '|', '2', '2.6', '2.9'], dtype=object)
```

In [30]:
```
error_row = df_amazon.loc[df_amazon['rating'] == '|']
error_row
```

Out[30]:

	product_id	product_name	category	discounted_price	actual_price	discount_percentage	rating	rating_count	about_product	
1279	B08L12N5H1	Eureka Forbes car Vac 100 Watts Powerful Sucti...	Home&Kitchen|Kitchen&HomeAppliances|Vacuum,Cle...	2099.0	2499.0	16.0	|	992	No Installation is provided for this product|1...	AGTDSNT2FKVYEPDPXAA673AIS44A,A

In [31]:
```
error_row['product_name']
# https://www.amazon.in/Eureka-Forbes-Vacuum-Cleaner-Washable/dp/B08L12N5H1
# This product has product rating = 3.9
```

Out[31]:
```
1279    Eureka Forbes car Vac 100 Watts Powerful Sucti...
Name: product_name, dtype: object
```

In [32]:
```
df_amazon['rating'] = df_amazon['rating'].str.replace('|','3.9')
```

In [33]:
```
df_amazon['rating'] = df_amazon['rating'].astype('float64')
```

In [34]:
```
df_amazon['rating']
```

Out[34]:
```
0       4.2
1       4.0
2       3.9
3       4.2
4       4.2
       ...
1460    4.0
1461    4.1
1462    3.6
1463    4.0
1464    4.3
Name: rating, Length: 1465, dtype: float64
```

5. Dealing with rating_count column

In [36]:
```
df_amazon['rating_count']
```

Out[36]:
```
0       24,269
1       43,994
2        7,928
3       94,363
4       16,905
         ...
1460     1,090
1461     4,118
1462       468
1463     8,031
1464     6,987
Name: rating_count, Length: 1465, dtype: object
```

In [37]:
```
df_amazon['rating_count'] = df_amazon['rating_count'].str.replace(',','')
```

In [38]:
```
df_amazon['rating_count'] = df_amazon['rating_count'].astype('float64')
```

In [39]:
```
df_amazon['rating_count']
```

Out[39]:
```
0       24269.0
1       43994.0
2        7928.0
3       94363.0
4       16905.0
         ...
1460     1090.0
1461     4118.0
1462      468.0
1463     8031.0
1464     6987.0
Name: rating_count, Length: 1465, dtype: float64
```

In [40]:
```
# Since rating_count column has 2 missing values
# Handling missing value -> Replacing NaN values with mean
df_amazon['rating_count'].mean()
```

Out[40]:
```
18295.541353383458
```

In [41]:
```
df_amazon['rating_count'] = df_amazon['rating_count'].fillna(df_amazon['rating_count'].mean())
```

In [42]:
```
# checking for missing values
df_amazon['rating_count'].isnull().sum()
```

Out[42]:
```
0
```

4.2. Checking for Duplicate values

In [44]:
```
# checking for duplciates
df_amazon.duplicated().sum()
```

Out[44]:
```
0
```

5. Creating Category and Sub-Category column from Category column

In [46]:
```
# df_amazon[['Category','Subcategory ']] = df_amazon['category'].str.split('|', expand= True)
# Above gives error ValueError: Columns must be same length as key
```

```
# Computers&Accessories|Accessories&Peripherals|Cables&Accessories|Cables|USBCables
df_category_split = df_amazon['category'].str.split('|', expand= True)
```

In [47]:
```
df_category_split
```

Out[47]:

	0	1	2	3	4	5	6
0	Computers&Accessories	Accessories&Peripherals	Cables&Accessories	Cables	USBCables	None	None
1	Computers&Accessories	Accessories&Peripherals	Cables&Accessories	Cables	USBCables	None	None
2	Computers&Accessories	Accessories&Peripherals	Cables&Accessories	Cables	USBCables	None	None
3	Computers&Accessories	Accessories&Peripherals	Cables&Accessories	Cables	USBCables	None	None
4	Computers&Accessories	Accessories&Peripherals	Cables&Accessories	Cables	USBCables	None	None
...	...	...	...	...	...	...	...
1460	Home&Kitchen	Kitchen&HomeAppliances	WaterPurifiers&Accessories	WaterPurifierAccessories	None	None	None
1461	Home&Kitchen	Kitchen&HomeAppliances	SmallKitchenAppliances	Rice&PastaCookers	None	None	None
1462	Home&Kitchen	Heating,Cooling&AirQuality	RoomHeaters	HeatConvectors	None	None	None
1463	Home&Kitchen	Heating,Cooling&AirQuality	Fans	ExhaustFans	None	None	None
1464	Home&Kitchen	Kitchen&HomeAppliances	SmallKitchenAppliances	SandwichMakers	None	None	None

1465 rows × 7 columns

In [48]:
```
df_category_split.isna().sum()
```

Out[48]:
```
0       0
1       0
2       8
3     165
4     943
5    1380
6    1452
dtype: int64
```

In [49]:
```
# Keep only 0th and 1st column
df_category_split.drop(columns=[2,3,4,5,6],inplace= True)
```

In [50]:
```
df_category_split
```

Out[50]:

	0	1
0	Computers&Accessories	Accessories&Peripherals
1	Computers&Accessories	Accessories&Peripherals
2	Computers&Accessories	Accessories&Peripherals
3	Computers&Accessories	Accessories&Peripherals
4	Computers&Accessories	Accessories&Peripherals
...	...	...
1460	Home&Kitchen	Kitchen&HomeAppliances
1461	Home&Kitchen	Kitchen&HomeAppliances
1462	Home&Kitchen	Heating,Cooling&AirQuality
1463	Home&Kitchen	Heating,Cooling&AirQuality
1464	Home&Kitchen	Kitchen&HomeAppliances

1465 rows × 2 columns

In [51]:
```
df_amazon['Category'] = df_category_split[0]
df_amazon['Sub-Category'] = df_category_split[1]
```

In [52]:
```
# Dropping the category column from df_amazon
df_amazon.drop(columns='category',inplace= True)
```

In [53]:
```
df_amazon.head(1)
```

Out[53]:

	product_id	product_name	discounted_price	actual_price	discount_percentage	rating	rating_count	about_product	user_id	user_name	
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	399.0	1099.0	64.0	4.2	24269.0	High Compatibility : Compatible With iPhone 12...	AG3D6O4STAQKAY2UVGEUV46KN35Q,AHMY5CWJMMK5BJRBB...	Manav,Adarsh gupta,Sundeep,S.Sayeed Ahmed,jasp...	R3HX

In [54]:
```
# formating the Category and Sub-Catgory column
```

In [55]:
```
df_amazon['Category'].unique()
```

Out[55]:
```
array(['Computers&Accessories', 'Electronics', 'MusicalInstruments',
       'OfficeProducts', 'Home&Kitchen', 'HomeImprovement', 'Toys&Games',
       'Car&Motorbike', 'Health&PersonalCare'], dtype=object)
```

In [56]:
```
df_amazon['Sub-Category'].unique()
```

Out[56]:
```
array(['Accessories&Peripherals', 'NetworkingDevices',
       'HomeTheater,TV&Video', 'HomeAudio', 'WearableTechnology',
       'Mobiles&Accessories', 'Accessories',
       'Headphones,Earbuds&Accessories', 'ExternalDevices&DataStorage',
       'Microphones', 'GeneralPurposeBatteries&BatteryChargers',
       'OfficePaperProducts', 'CraftMaterials', 'Cameras&Photography',
       'OfficeElectronics', 'Printers,Inks&Accessories', 'Monitors',
       'Components', 'Electrical', 'Arts&Crafts', 'PowerAccessories',
       'Tablets', 'Laptops', 'Kitchen&HomeAppliances',
       'Heating,Cooling&AirQuality', 'Kitchen&Dining',
       'HomeStorage&Organization', 'CarAccessories',
       'HomeMedicalSupplies&Equipment'], dtype=object)
```

In [57]:
```
df_amazon['Category'] = df_amazon['Category'].str.replace('Computers&Accessories','Computers & Accessories')
df_amazon['Category'] = df_amazon['Category'].str.replace('MusicalInstruments','Musical Instruments')
df_amazon['Category'] = df_amazon['Category'].str.replace('OfficeProducts','Office Products')
df_amazon['Category'] = df_amazon['Category'].str.replace('Home&Kitchen','Home & Kitchen')
df_amazon['Category'] = df_amazon['Category'].str.replace('HomeImprovement','Home Improvement')
df_amazon['Category'] = df_amazon['Category'].str.replace('Toys&Games','Toys & Games')
df_amazon['Category'] = df_amazon['Category'].str.replace('Car&Motorbike','Car & Motorbike')
df_amazon['Category'] = df_amazon['Category'].str.replace('Health&PersonalCare','Health & Personal Care')
```

In [58]:
```
df_amazon['Category'].unique()
```

Out[58]:
```
array(['Computers & Accessories', 'Electronics', 'Musical Instruments',
       'Office Products', 'Home & Kitchen', 'Home Improvement',
       'Toys & Games', 'Car & Motorbike', 'Health & Personal Care'],
      dtype=object)
```

In [59]:
```
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('Accessories&Peripherals','Accessories & Peripherals')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('NetworkingDevices','Networking Devices')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('HomeTheater,TV&Video','HomeTheater, TV & Video')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('HomeAudio','Home Audio')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('WearableTechnology','Wearable Technology')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('Mobiles&Accessories','Mobiles & Accessories')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('Headphones,Earbuds&Accessories','Headphones, Earbuds & Accessories')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('ExternalDevices&DataStorage','External Devices & Data Storage')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('GeneralPurposeBatteries&BatteryChargers','General Purpose Batteries & Battery Chargers')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('OfficePaperProducts','Office Paper Products')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('CraftMaterials','Craft Materials')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('Cameras&Photography','Cameras & Photography')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('OfficeElectronics','Office Electronics')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('Printers,Inks&Accessories','Printers, Inks & Accessories')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('Arts&Crafts','Arts & Crafts')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('PowerAccessories','Power Accessories')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('Kitchen&HomeAppliances','Kitchen & Home Appliances')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('Heating,Cooling&AirQuality','Heating, Cooling & Air Quality')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('Kitchen&Dining','Kitchen & Dining')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('HomeStorage&Organization','Home Storage & Organization')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('CarAccessories','Car Accessories')
df_amazon['Sub-Category'] = df_amazon['Sub-Category'].str.replace('HomeMedicalSupplies&Equipment','Home Medical Supplies & Equipment')
```

In [60]:
```
df_amazon['Sub-Category'].unique()
```

Out[60]:
```
array(['Accessories & Peripherals', 'Networking Devices',
       'HomeTheater, TV & Video', 'Home Audio', 'Wearable Technology',
       'Mobiles & Accessories', 'Accessories',
       'Headphones, Earbuds & Accessories',
       'External Devices & Data Storage', 'Microphones',
       'General Purpose Batteries & Battery Chargers',
       'Office Paper Products', 'Craft Materials',
       'Cameras & Photography', 'Office Electronics',
       'Printers, Inks & Accessories', 'Monitors', 'Components',
       'Electrical', 'Arts & Crafts', 'Power Accessories', 'Tablets',
       'Laptops', 'Kitchen & Home Appliances',
       'Heating, Cooling & Air Quality', 'Kitchen & Dining',
       'Home Storage & Organization', 'Car Accessories',
       'Home Medical Supplies & Equipment'], dtype=object)
```

In [61]:
```
df_amazon['rating'].unique()
```

Out[61]:
```
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, 2. , 2.6, 2.9])
```

In [62]:
```
df_amazon.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1465 entries, 0 to 1464
Data columns (total 17 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   product_id           1465 non-null   object 
 1   product_name         1465 non-null   object 
 2   discounted_price     1465 non-null   float64
 3   actual_price         1465 non-null   float64
 4   discount_percentage  1465 non-null   float64
 5   rating               1465 non-null   float64
 6   rating_count         1465 non-null   float64
 7   about_product        1465 non-null   object 
 8   user_id              1465 non-null   object 
 9   user_name            1465 non-null   object 
 10  review_id            1465 non-null   object 
 11  review_title         1465 non-null   object 
 12  review_content       1465 non-null   object 
 13  img_link             1465 non-null   object 
 14  product_link         1465 non-null   object 
 15  Category             1465 non-null   object 
 16  Sub-Category         1465 non-null   object 
dtypes: float64(5), object(12)
memory usage: 194.7+ KB
```

In [63]:
```
df_amazon['rating']
```

Out[63]:
```
0       4.2
1       4.0
2       3.9
3       4.2
4       4.2
       ...
1460    4.0
1461    4.1
1462    3.6
1463    4.0
1464    4.3
Name: rating, Length: 1465, dtype: float64
```

6. Creating Feedback score column

In [65]:
```
# 0 Star
feedback_score = []

for quality in df_amazon['rating'] :
    if quality < 1.0 :
        feedback_score.append('very low quality')
    elif quality < 2.0 :
        feedback_score.append('low quality')
    elif quality < 3.0 :
        feedback_score.append('average quality')
    elif quality < 4.0 :
        feedback_score.append('good quality')
    elif quality < 5.0 :
        feedback_score.append('very good quality')
    elif quality == 5.0 :
        feedback_score.append('excellent quality')

df_amazon['feedback_score'] = feedback_score
df_amazon['feedback_score'] = df_amazon['feedback_score'].astype('category')
```

7. Dealing with user_id and user_name column

In [67]:
```
# The user_id and user_name column has multiple ids and names in same cell.
# Now we need each user_id and user_name in individual rows
# creating a dataframe of it.
```

In [68]:
```
# str.split() will split every element after ',' and then a list is created with thses names with ',' as delimiter
# By default expand = False parameter for split()
split_user_id = df_amazon['user_id'].str.split(',',expand= False )
split_user_name = df_amazon['user_name'].str.split(',', expand= False)
```

In [69]:
```
split_user_name
```

Out[69]:
```
0       [Manav, Adarsh gupta, Sundeep, S.Sayeed Ahmed,...
1       [ArdKn, Nirbhay kumar, Sagar Viswanathan, Asp,...
2       [Kunal, Himanshu, viswanath, sai niharka, saqi...
3       [Omkar dhale, JD, HEMALATHA, Ajwadh a., amar s...
4       [rahuls6099, Swasat Borah, Ajay Wadke, Pranali...
                              ...
1460    [Prabha ds, Raghuram bk, Real Deal, Amazon Cus...
1461    [Manu Bhai, Naveenpittu, Evatira Sangma, JAGAN...
1462    [Nehal Desai, Danish Parwez, Amazon Customer, ...
1463    [Shubham Dubey, E.GURUBARAN, Mayank S., eusuf ...
1464    [Rajib, Ajay B, Vikas Kahol, PARDEEP, Anindya ...
Name: user_name, Length: 1465, dtype: object
```

In [70]:
```
split_user_name
```

Out[70]:
```
0       [Manav, Adarsh gupta, Sundeep, S.Sayeed Ahmed,...
1       [ArdKn, Nirbhay kumar, Sagar Viswanathan, Asp,...
2       [Kunal, Himanshu, viswanath, sai niharka, saqi...
3       [Omkar dhale, JD, HEMALATHA, Ajwadh a., amar s...
4       [rahuls6099, Swasat Borah, Ajay Wadke, Pranali...
                              ...
1460    [Prabha ds, Raghuram bk, Real Deal, Amazon Cus...
1461    [Manu Bhai, Naveenpittu, Evatira Sangma, JAGAN...
1462    [Nehal Desai, Danish Parwez, Amazon Customer, ...
1463    [Shubham Dubey, E.GURUBARAN, Mayank S., eusuf ...
1464    [Rajib, Ajay B, Vikas Kahol, PARDEEP, Anindya ...
Name: user_name, Length: 1465, dtype: object
```

In [71]:
```
exploded_user_id = split_user_id.explode()
exploded_user_name = split_user_name.explode()
```

In [72]:
```
exploded_user_id
```

Out[72]:
```
0       AG3D6O4STAQKAY2UVGEUV46KN35Q
0       AHMY5CWJMMK5BJRBBSNLYT3ONILA
0       AHCTC6ULH4XB6YHDY6PCH2R772LQ
0       AGYHHIERNXKA6P5T7CZLXKVPT7IQ
0       AG4OGOFWXJZTQ2HKYIOCOY3KXF2Q
                  ...
1464    AHXCDNSXAESERITAFELQABFVNLCA
1464    AGRZD6CHLCUNOLMMIMIHUCG7PIFA
1464    AFQZVGSOSOJHKFQQMCEI4725QEKQ
1464    AEALVGXXIP46OZVXKRUXSDWZJMEA
1464    AGEFL3AY7YXEFZA4ZJU3LP7K7OJQ
Name: user_id, Length: 11503, dtype: object
```

In [73]:
```
exploded_user_name
```

Out[73]:
```
0                Manav
0          Adarsh gupta
0              Sundeep
0        S.Sayeed Ahmed
0        jaspreet singh
             ...
1464           PARDEEP
1464    Anindya Pramanik
1464        Vikas Singh
1464     Harshada Pimple
1464             Saw a.
Name: user_name, Length: 11515, dtype: object
```

8. Creating a dataframe with user info

In [75]:
```
df_user_info = pd.DataFrame(exploded_user_name)
df_user_id = pd.DataFrame(exploded_user_id)
```

In [76]:
```
df_user_info['product_name'] = df_amazon['product_name']
df_user_info['Category'] = df_amazon['Category']
df_user_info['Sub-Category'] = df_amazon['Sub-Category']
```

In [77]:
```
# Since multiple rows have same index so use reset.index() & parameter drop = True will not add the old index and a new
# sequential index is assigned
df_user_id = df_user_id.reset_index(drop= True)
df_user_info = df_user_info.reset_index(drop= True)
```

In [78]:
```
# Merging these two dataframes into one customer dataframe
# We set left_index and right_index paramter True so that index column of both the dataframes are used for joining
df_customers_info = pd.merge(df_user_id, df_user_info, left_index= True, right_index= True)
df_customers_info
```

Out[78]:

	user_id	user_name	product_name	Category	Sub-Category
0	AG3D6O4STAQKAY2UVGEUV46KN35Q	Manav	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers & Accessories	Accessories & Peripherals
1	AHMY5CWJMMK5BJRBBSNLYT3ONILA	Adarsh gupta	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers & Accessories	Accessories & Peripherals
2	AHCTC6ULH4XB6YHDY6PCH2R772LQ	Sundeep	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers & Accessories	Accessories & Peripherals
3	AGYHHIERNXKA6P5T7CZLXKVPT7IQ	S.Sayeed Ahmed	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers & Accessories	Accessories & Peripherals
4	AG4OGOFWXJZTQ2HKYIOCOY3KXF2Q	jaspreet singh	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers & Accessories	Accessories & Peripherals
...	...	...	...	...	...
11498	AHXCDNSXAESERITAFELQABFVNLCA	BATTU SURESHKUMAR	Bajaj Majesty RX10 2000 Watts Heat Convector R...	Home & Kitchen	Heating, Cooling & Air Quality
11499	AGRZD6CHLCUNOLMMIMIHUCG7PIFA	Shubham Dubey	Havells Ventil Air DSP 230mm Exhaust Fan (Pist...	Home & Kitchen	Heating, Cooling & Air Quality
11500	AFQZVGSOSOJHKFQQMCEI4725QEKQ	E.GURUBARAN	Havells Ventil Air DSP 230mm Exhaust Fan (Pist...	Home & Kitchen	Heating, Cooling & Air Quality
11501	AEALVGXXIP46OZVXKRUXSDWZJMEA	Mayank S.	Havells Ventil Air DSP 230mm Exhaust Fan (Pist...	Home & Kitchen	Heating, Cooling & Air Quality
11502	AGEFL3AY7YXEFZA4ZJU3LP7K7OJQ	eusuf khan	Havells Ventil Air DSP 230mm Exhaust Fan (Pist...	Home & Kitchen	Heating, Cooling & Air Quality

11503 rows × 5 columns

In [79]:
```
# Check for null values in customer dataframe
df_customers_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 11503 entries, 0 to 11502
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   user_id       11503 non-null  object
 1   user_name     11503 non-null  object
 2   product_name  11503 non-null  object
 3   Category      11503 non-null  object
 4   Sub-Category  11503 non-null  object
dtypes: object(5)
memory usage: 539.2+ KB
```

In [80]:
```
df_amazon.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1465 entries, 0 to 1464
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   product_id           1465 non-null   object 
 1   product_name         1465 non-null   object 
 2   discounted_price     1465 non-null   float64
 3   actual_price         1465 non-null   float64
 4   discount_percentage  1465 non-null   float64
 5   rating               1465 non-null   float64
 6   rating_count         1465 non-null   float64
 7   about_product        1465 non-null   object 
 8   user_id              1465 non-null   object 
 9   user_name            1465 non-null   object 
 10  review_id            1465 non-null   object 
 11  review_title         1465 non-null   object 
 12  review_content       1465 non-null   object 
 13  img_link             1465 non-null   object 
 14  product_link         1465 non-null   object 
 15  Category             1465 non-null   object 
 16  Sub-Category         1465 non-null   object 
 17  feedback_score       1465 non-null   category
dtypes: category(1), float64(5), object(12)
memory usage: 196.3+ KB
```

II. Data Analysis

1. Product V/s Category and Sub-Category

This visual shows which product is more popular among Category and Sub-Category

In [84]:
```
# value_counts() creates a pandas series
# reset_index(name = 'total') this converts the pandas series into a dataframe and name the count column to 'total'
cat_count = df_amazon['Category'].value_counts().reset_index(name = 'total')
cat_count
```

Out[84]:

	Category	total
0	Electronics	526
1	Computers & Accessories	453
2	Home & Kitchen	448
3	Office Products	31
4	Musical Instruments	2
5	Home Improvement	2
6	Toys & Games	1
7	Car & Motorbike	1
8	Health & Personal Care	1

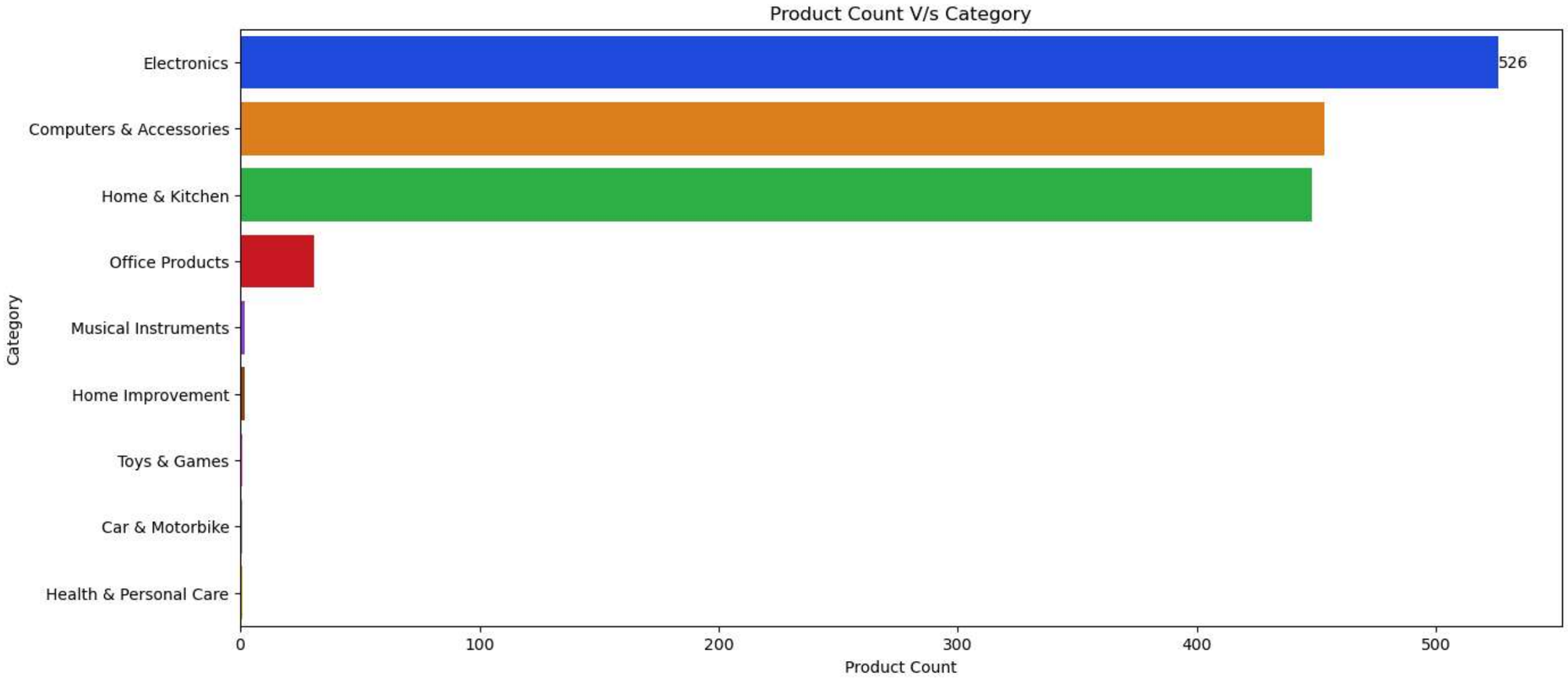
In [85]:
```
# Similarly for Sub-Category
subcat_count = df_amazon['Sub-Category'].value_counts().reset_index(name = 'total')
subcat_count
```

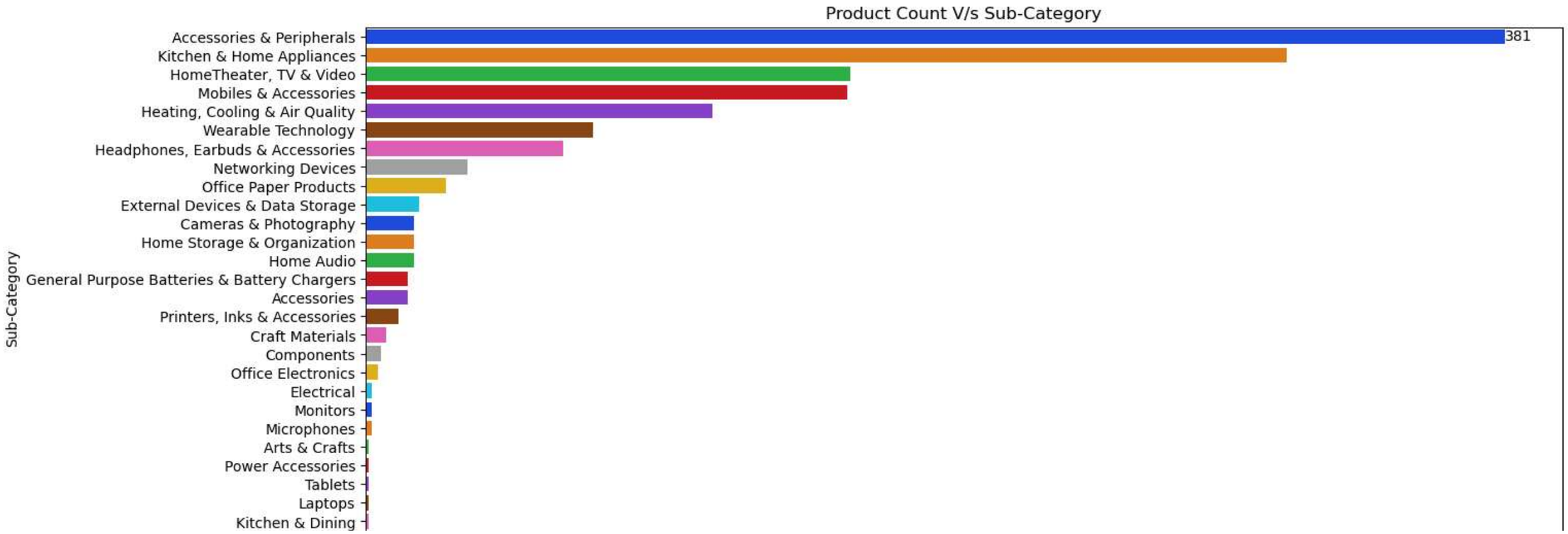
Out[85]:

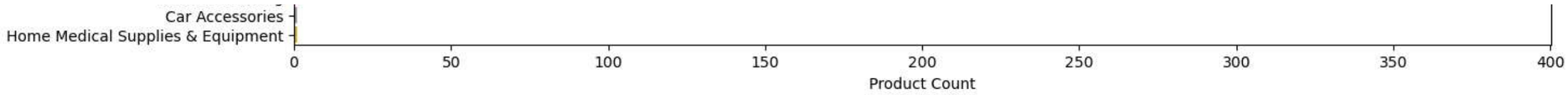
	Sub-Category	total
0	Accessories & Peripherals	381
1	Kitchen & Home Appliances	308
2	HomeTheater, TV & Video	162
3	Mobiles & Accessories	161
4	Heating, Cooling & Air Quality	116
5	Wearable Technology	76
6	Headphones, Earbuds & Accessories	66
7	Networking Devices	34
8	Office Paper Products	27
9	External Devices & Data Storage	18
10	Cameras & Photography	16
11	Home Storage & Organization	16
12	Home Audio	16
13	General Purpose Batteries & Battery Chargers	14
14	Accessories	14
15	Printers, Inks & Accessories	11
16	Craft Materials	7
17	Components	5
18	Office Electronics	4
19	Electrical	2
20	Monitors	2
21	Microphones	2
22	Arts & Crafts	1
23	Power Accessories	1
24	Tablets	1
25	Laptops	1
26	Kitchen & Dining	1
27	Car Accessories	1
28	Home Medical Supplies & Equipment	1

In [86]:
```
# We create a bar chart using seaborn
# First we create a plot (a canvas) using matplotlib.pyplot
# This fig, ax is a tuple fig -> canvas & ax -> individual chart this catches the outuput of subplots() function
# figsize = 15, 15 , rows -> 2 and column -> 1
fig, ax = plt.subplots(2, 1, figsize = (15, 15))
```

```
# sns.barplot to create a bar chart

sns.barplot(ax= ax[0],
            data= cat_count,
            x = 'total',
            y = 'Category',
            palette='bright'
            ).set(title = 'Product Count V/s Category',
                                xlabel = 'Product Count',
                                ylabel = 'Category'
            )

sns.barplot(ax= ax[1],
            data= subcat_count,
            x = 'total',
            y = 'Sub-Category',
            palette='bright'
            ).set(title = 'Product Count V/s Sub-Category',
                                xlabel = 'Product Count',
                                ylabel = 'Sub-Category'

            )

# To add data labels to each bar bar_label()

ax[0].bar_label(ax[0].containers[0])
ax[1].bar_label(ax[1].containers[0])

plt.show()
```


Product Count V/s Category
Electronics
Computers & Accessories
Home & Kitchen
Office Products
Musical Instruments
Home Improvement
Toys & Games
Car & Motorbike
Health & Personal Care
526
Category
0
100
200
300
400
500
Product Count


Product Count V/s Sub-Category
Accessories & Peripherals
Kitchen & Home Appliances
HomeTheater, TV & Video
Mobiles & Accessories
Heating, Cooling & Air Quality
Wearable Technology
Headphones, Earbuds & Accessories
Networking Devices
Office Paper Products
External Devices & Data Storage
Cameras & Photography
Home Storage & Organization
Home Audio
General Purpose Batteries & Battery Chargers
Accessories
Printers, Inks & Accessories
Craft Materials
Components
Office Electronics
Electrical
Monitors
Microphones
Arts & Crafts
Power Accessories
Tablets
Laptops
Kitchen & Dining
381
Sub-Category

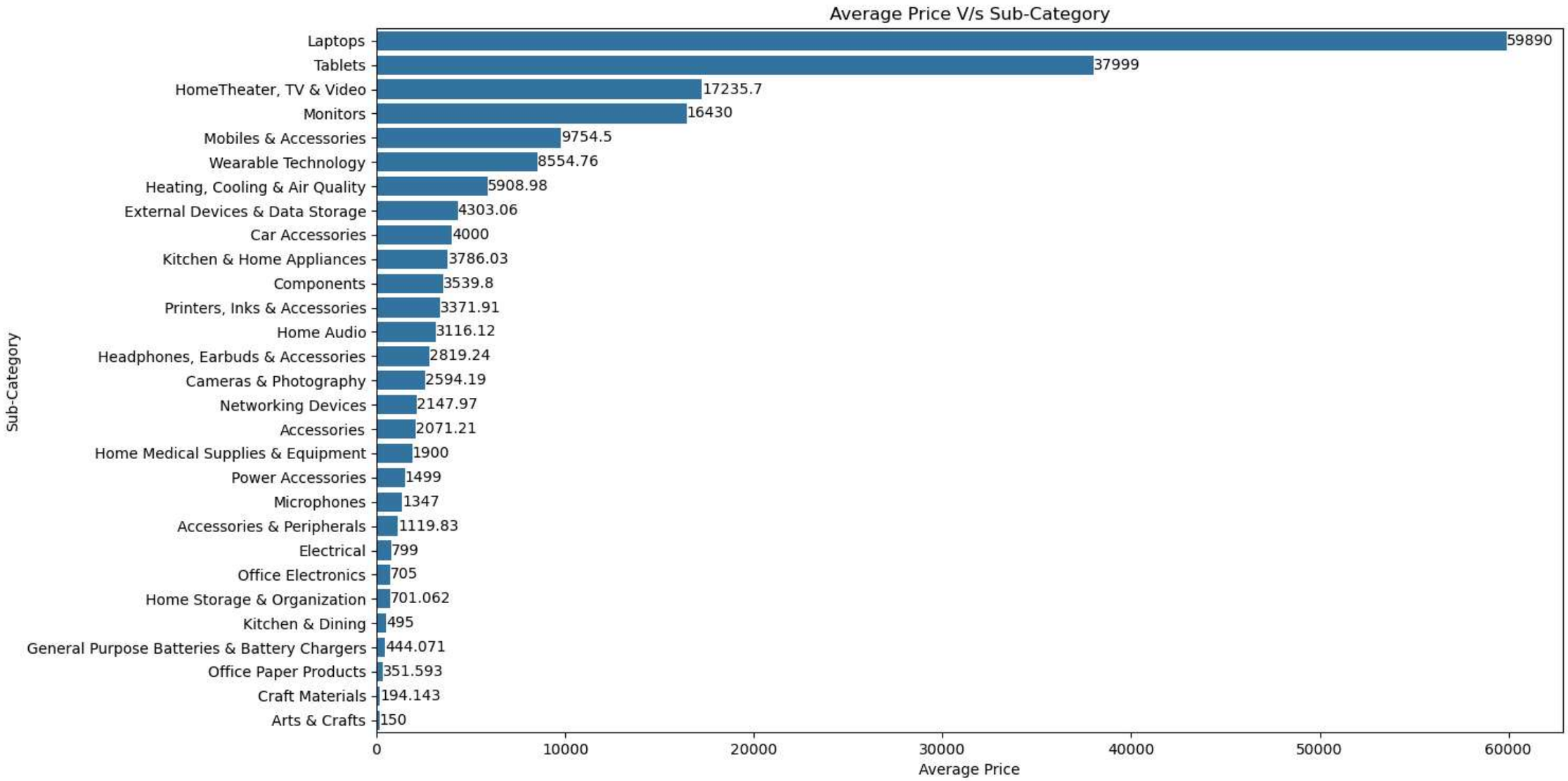


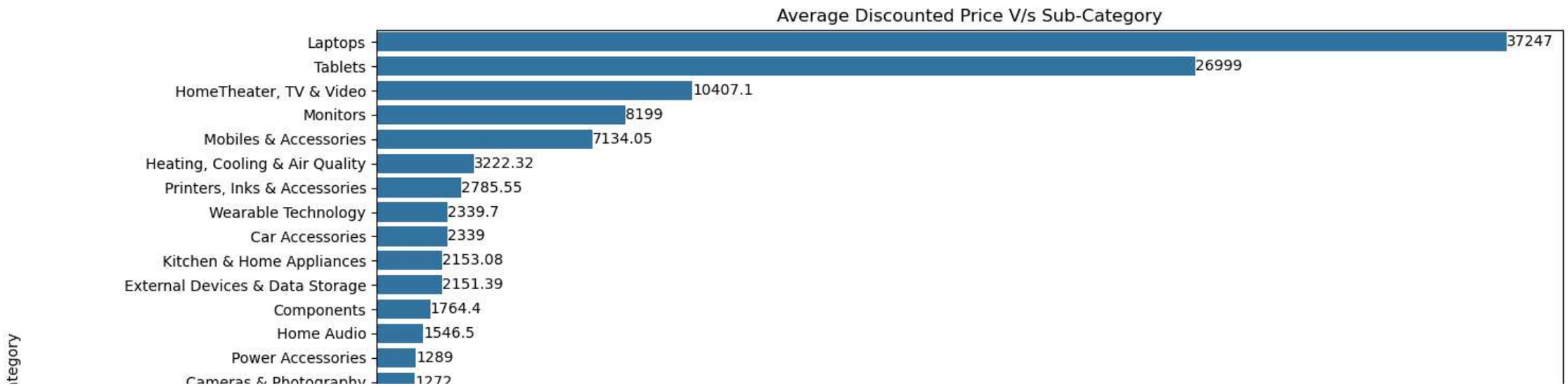
Most Popular Category -> Electronics

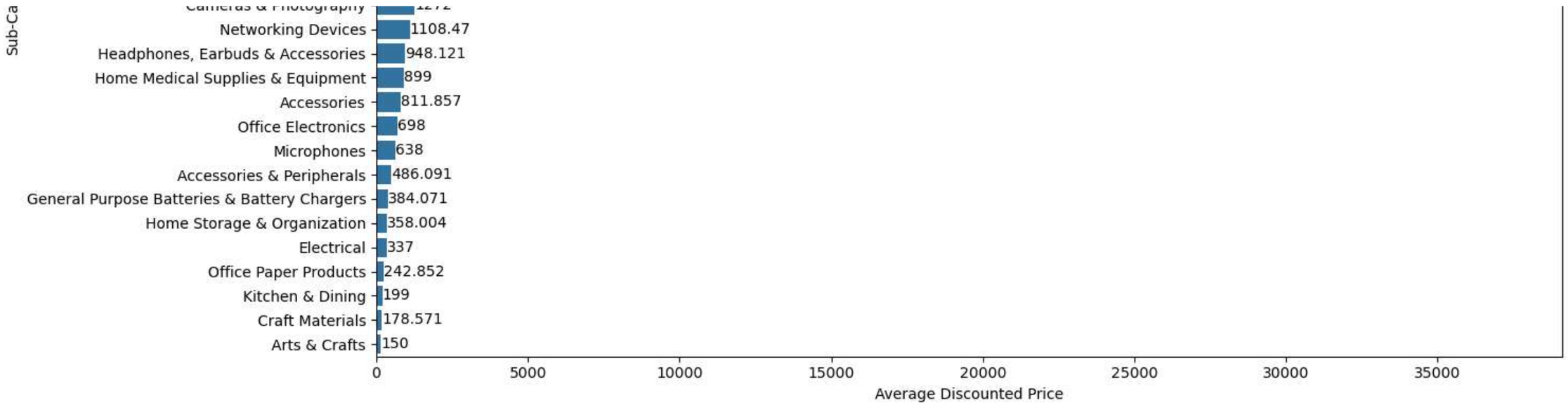
Most Popular Category -> Accessories & Peripheral

2. Average price and Average discounted price V/s Sub-category

In [89]:
```
# create a list to order the sub-category from highest to lowest
# first group up the subcategory by groupby() and
# to calcualte average price we use ['actual_price'].mean()
# then we sort the vaules descending by sort_values(ascending = False)
# we are only interested in order so we take the categories names using .index
# and need the order to be a list so we create a numpy array with .values

subcat_avgprice = df_amazon.groupby('Sub-Category')['actual_price'].mean().sort_values(ascending=False)
subcat_avgdiscountedprice = df_amazon.groupby('Sub-Category')['discounted_price'].mean().sort_values(ascending=False)

subcat_avgprice_order = subcat_avgprice.index.tolist()
subcat_avgdiscountedprice_order = subcat_avgdiscountedprice.index.tolist()

fig, ax = plt.subplots(2, 1, figsize = (15, 15))

# sns.barplot to create a bar chart
sns.barplot(ax= ax[0],
            x = subcat_avgprice.values,
            y = subcat_avgprice.index,
           # palette='bright',
            errorbar= None,
            order= subcat_avgprice_order
            ).set(title = 'Average Price V/s Sub-Category',
                                xlabel = 'Average Price',
                                ylabel = 'Sub-Category'
           )

sns.barplot(ax= ax[1],
            x = subcat_avgdiscountedprice.values,
            y = subcat_avgdiscountedprice.index,
           # palette='bright',
            errorbar= None,
            order= subcat_avgdiscountedprice_order
            ).set(title = 'Average Discounted Price V/s Sub-Category',
                                xlabel = 'Average Discounted Price',
                                ylabel = 'Sub-Category'
           )

# To add data labels to each bar bar_label()

ax[0].bar_label(ax[0].containers[0])
ax[1].bar_label(ax[1].containers[0])
plt.tight_layout()
plt.show()
```


Average Price V/s Sub-Category
Sub-Category
Laptops
Tablets
HomeTheater, TV & Video
Monitors
Mobiles & Accessories
Wearable Technology
Heating, Cooling & Air Quality
External Devices & Data Storage
Car Accessories
Kitchen & Home Appliances
Components
Printers, Inks & Accessories
Home Audio
Headphones, Earbuds & Accessories
Cameras & Photography
Networking Devices
Accessories
Home Medical Supplies & Equipment
Power Accessories
Microphones
Accessories & Peripherals
Electrical
Office Electronics
Home Storage & Organization
Kitchen & Dining
General Purpose Batteries & Battery Chargers
Office Paper Products
Craft Materials
Arts & Crafts
59890
37999
17235.7
16430
9754.5
8554.76
5908.98
4303.06
4000
3786.03
3539.8
3371.91
3116.12
2819.24
2594.19
2147.97
2071.21
1900
1499
1347
1119.83
799
705
701.062
495
444.071
351.593
194.143
150
0
10000
20000
30000
40000
50000
60000
Average Price

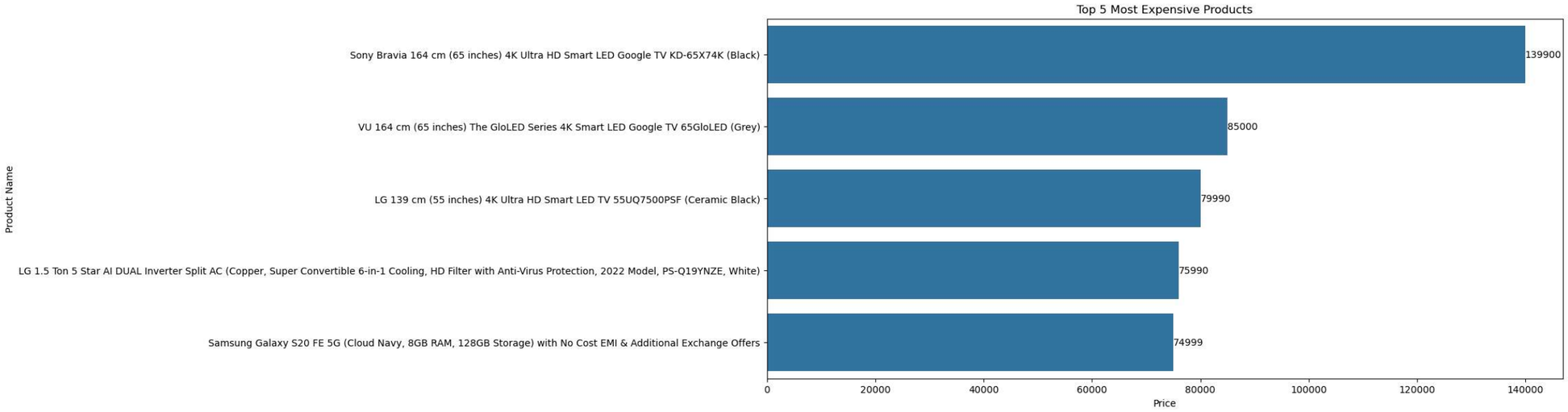

Average Discounted Price V/s Sub-Category
Laptops
Tablets
HomeTheater, TV & Video
Monitors
Mobiles & Accessories
Heating, Cooling & Air Quality
Printers, Inks & Accessories
Wearable Technology
Car Accessories
Kitchen & Home Appliances
External Devices & Data Storage
Components
Home Audio
Power Accessories
37247
26999
10407.1
8199
7134.05
3222.32
2785.55
2339.7
2339
2153.08
2151.39
1764.4
1546.5
1289

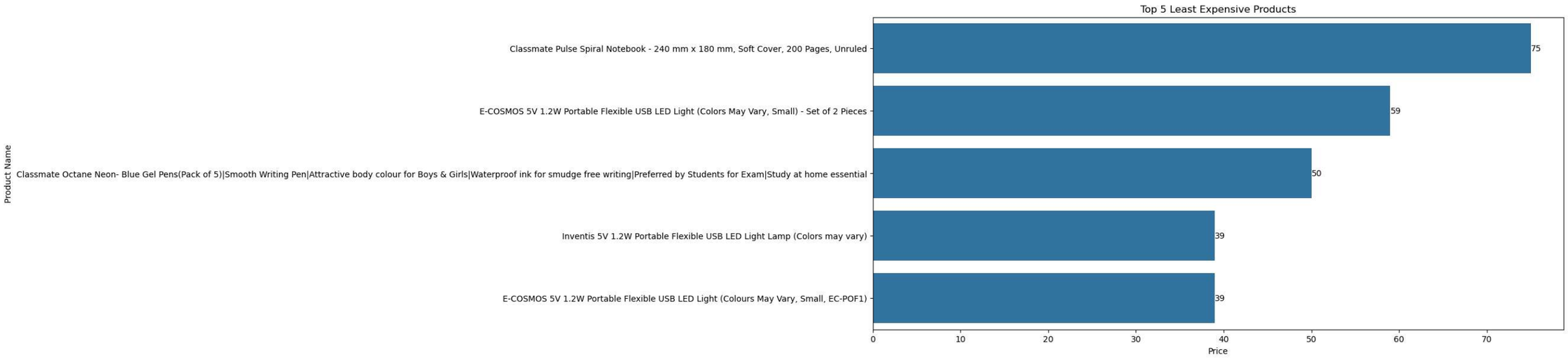


Laptops, Tablets, Home Theater, TV & Video & Monitors categories have the highest disounted price

3. Top 5 Most Expensive products & Top 5 Least Expensive products

In [92]:
```
fig, ax = plt.subplots(2, 1, figsize = (15, 15))

sns.barplot(ax = ax[0],
            data = df_amazon.sort_values('actual_price', ascending=False).head(5),
            x = 'actual_price',
            y = 'product_name').set(title = 'Top 5 Most Expensive Products',
                                    xlabel = 'Price',
                                    ylabel = 'Product Name')


sns.barplot(ax = ax[1],
            data = df_amazon.sort_values('actual_price', ascending=False).tail(5),
            x = 'actual_price',
            y = 'product_name').set(title = 'Top 5 Least Expensive Products',
                                    xlabel = 'Price',
                                    ylabel = 'Product Name')


ax[0].bar_label(ax[0].containers[0])
ax[1].bar_label(ax[1].containers[0])

plt.show()
```





Top 5 Most Expensive products belongs to Electronics Category

Top 5 Least Expensive products belongs to Office products and Electronics Category

4. Top 5 Most & Least purchased products

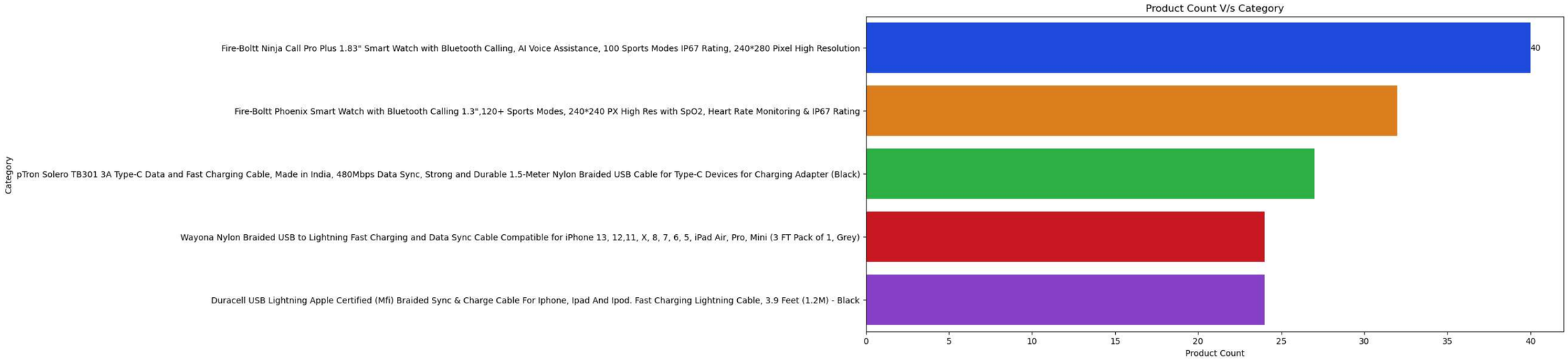
In [95]:
```
product_count = df_customers_info['product_name'].value_counts().reset_index(name = 'total')
product_count
```

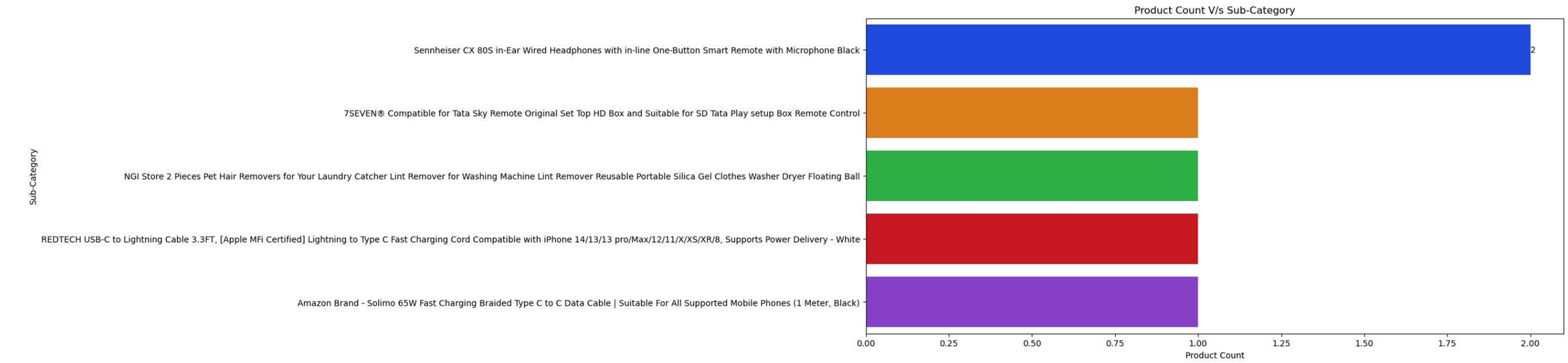
Out[95]:

	product_name	total
0	Fire-Boltt Ninja Call Pro Plus 1.83" Smart Wat...	40
1	Fire-Boltt Phoenix Smart Watch with Bluetooth ...	32
2	pTron Solero TB301 3A Type-C Data and Fast Cha...	27
3	Wayona Nylon Braided USB to Lightning Fast Cha...	24
4	Duracell USB Lightning Apple Certified (Mfi) B...	24
...	...	...
1331	Sennheiser CX 80S in-Ear Wired Headphones with...	2
1332	7SEVEN® Compatible for Tata Sky Remote Origina...	1
1333	NGI Store 2 Pieces Pet Hair Removers for Your ...	1
1334	REDTECH USB-C to Lightning Cable 3.3FT, [Apple...	1
1335	Amazon Brand - Solimo 65W Fast Charging Braide...	1

1336 rows × 2 columns

In [96]:
```
fig, ax = plt.subplots(2, 1, figsize = (15, 15))

# sns.barplot to create a bar chart

sns.barplot(ax= ax[0],
            data= product_count.head(5),
            x = 'total',
            y = 'product_name',
            palette='bright'
            ).set(title = 'Product Count V/s Category',
                                xlabel = 'Product Count',
                                ylabel = 'Category'
            )

sns.barplot(ax= ax[1],
            data= product_count.tail(5),
            x = 'total',
            y = 'product_name',
            palette='bright'
            ).set(title = 'Product Count V/s Sub-Category',
                                xlabel = 'Product Count',
                                ylabel = 'Sub-Category'

            )

# To add data labels to each bar bar_label()

ax[0].bar_label(ax[0].containers[0])
ax[1].bar_label(ax[1].containers[0])

plt.show()
```

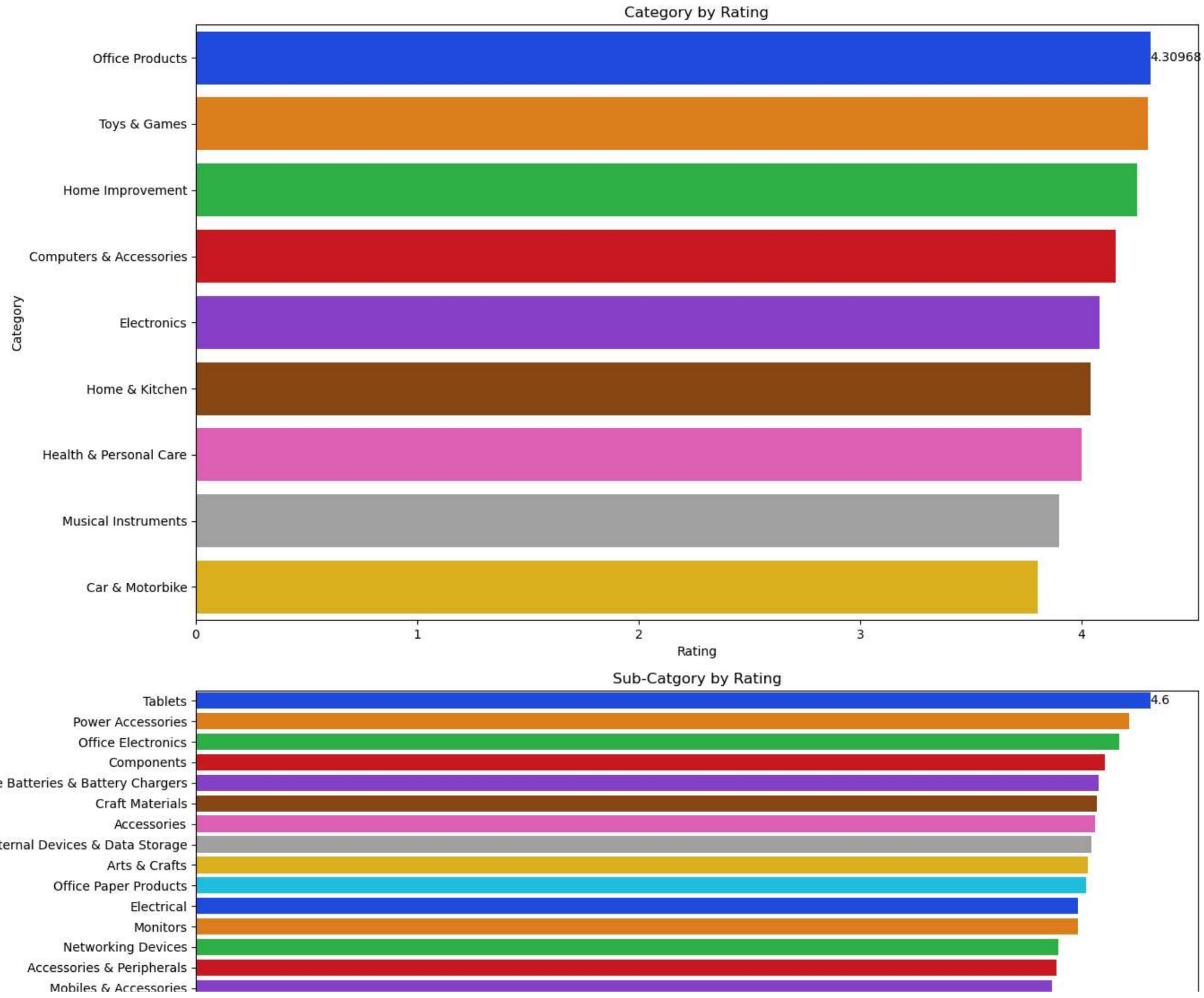


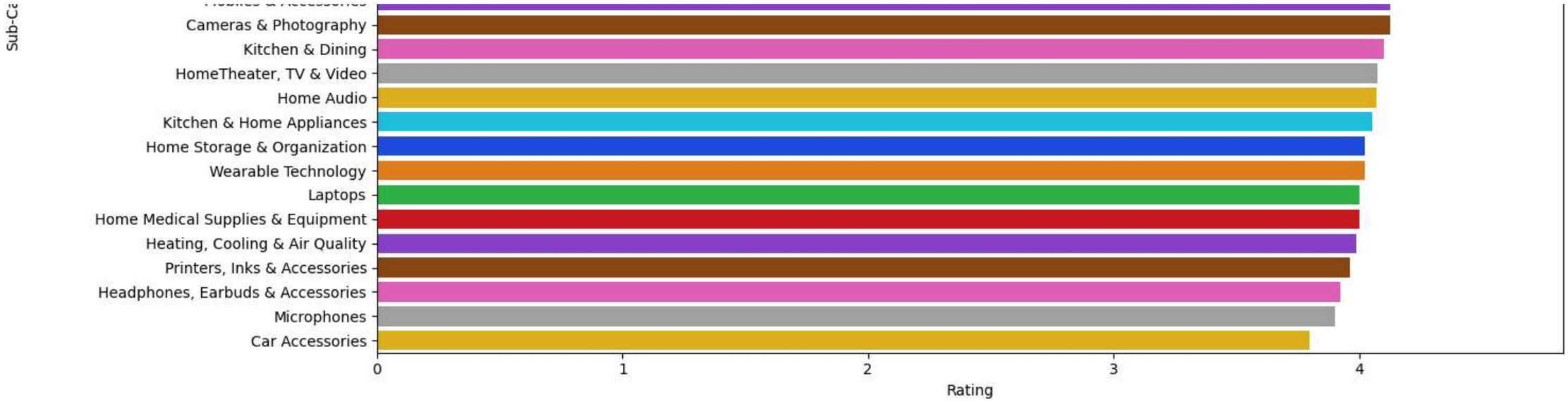


5. Category & Sub-Category by Rating

In [98]:
```
# Here we group up Category and Sub-Category and then take average of the ratings
cat_avgrating = df_amazon.groupby('Category')['rating'].mean().sort_values(ascending=False)
subcat_avgrating = df_amazon.groupby('Sub-Category')['rating'].mean().sort_values(ascending=False)

cat_avgrating_order = cat_avgrating.index.tolist()
subcat_avgrating_order = subcat_avgrating.index.tolist()

fig, ax = plt.subplots(2, 1, figsize = (15, 15))

# sns.barplot to create a bar chart
sns.barplot(ax= ax[0],
            x = cat_avgrating.values,
            y = cat_avgrating.index,
            palette='bright',
            errorbar= None,
            order= cat_avgrating_order
            ).set(title = 'Category by Rating',
```

```
                                xlabel = 'Rating',
                                ylabel = 'Category'
            )

sns.barplot(ax= ax[1],
            x = subcat_avgrating.values,
            y = subcat_avgrating.index,
            palette='bright',
            errorbar= None,
            order= subcat_avgrating_order
            ).set(title = 'Sub-Catgory by Rating',
                                xlabel = 'Rating',
                                ylabel = 'Sub-Category'
            )

# To add data labels to each bar bar_label()

ax[0].bar_label(ax[0].containers[0])
ax[1].bar_label(ax[1].containers[0])
plt.tight_layout()
plt.show()
```

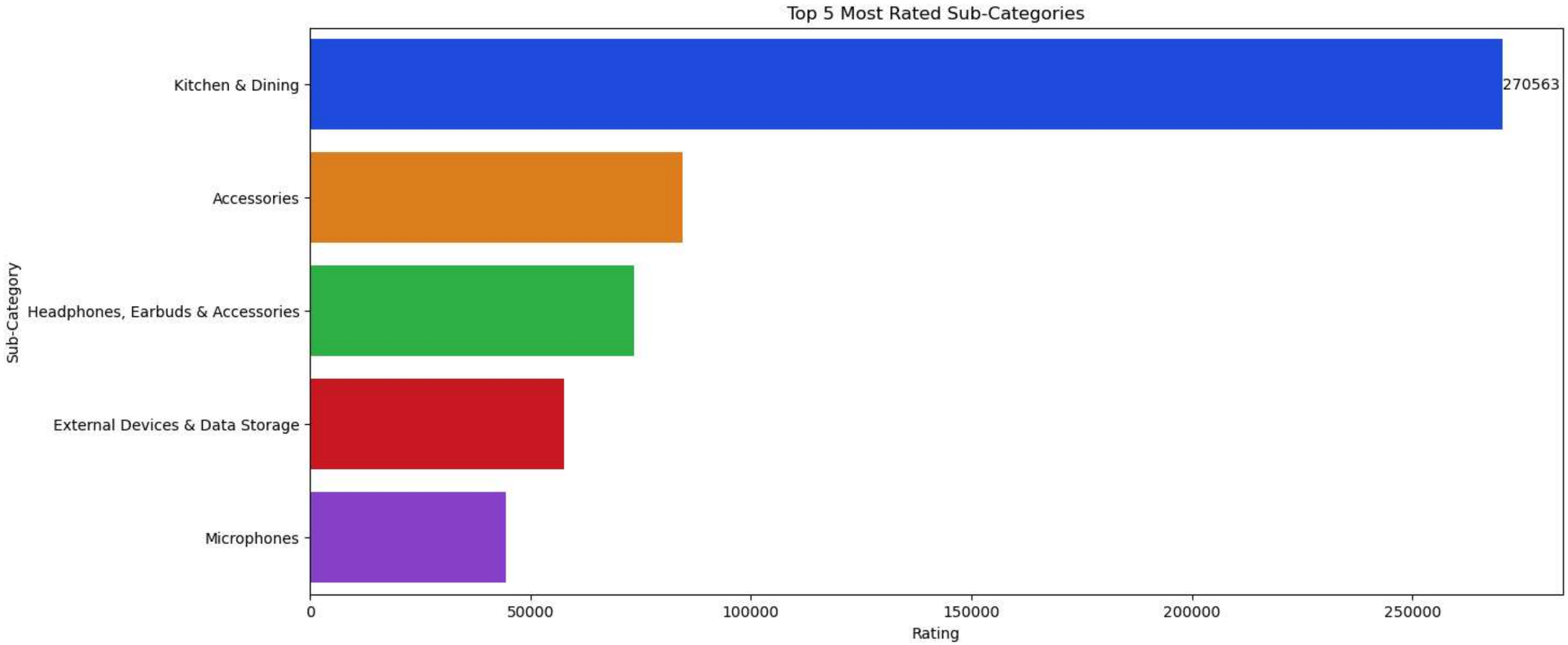

Category by Rating
Office Products
Toys & Games
Home Improvement
Computers & Accessories
Electronics
Home & Kitchen
Health & Personal Care
Musical Instruments
Car & Motorbike
4.30968
Category
0
1
2
3
4
Rating
Sub-Catgory by Rating
Tablets
Power Accessories
Office Electronics
Components
General Purpose Batteries & Battery Chargers
Craft Materials
Accessories
External Devices & Data Storage
Arts & Crafts
Office Paper Products
Electrical
Monitors
Networking Devices
Accessories & Peripherals
Mobiles & Accessories
4.6

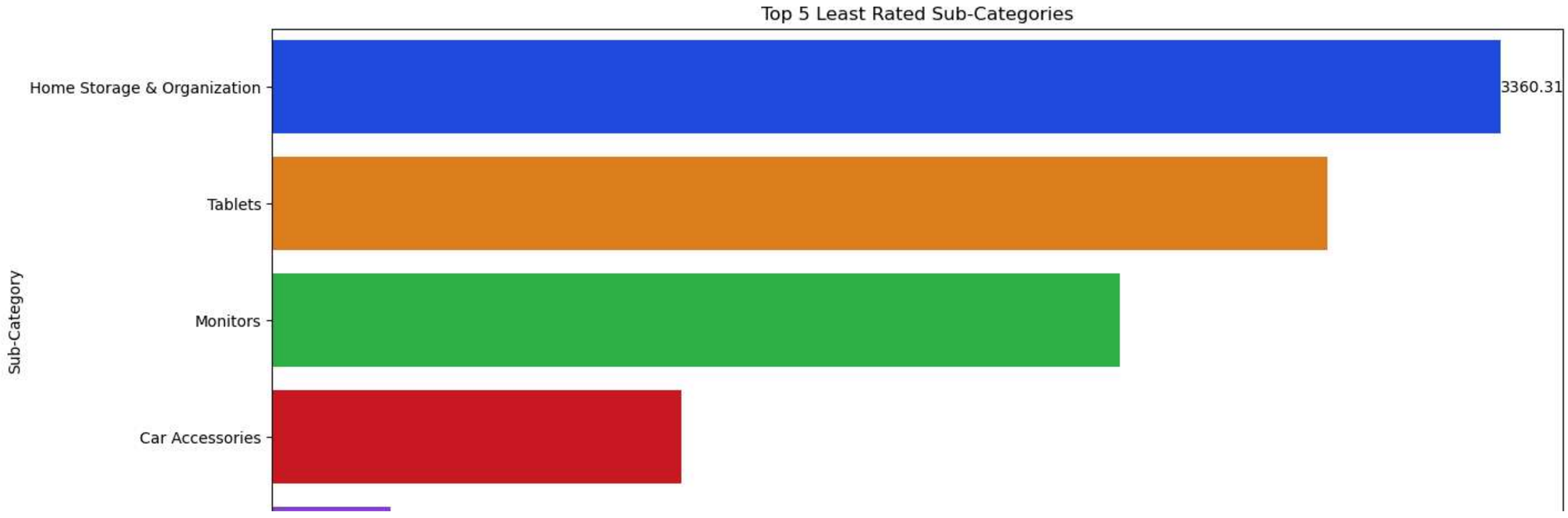


6. Top 5 Most & Least Rated Sub-Categories

In [100…

```
subcat_top5 = df_amazon.groupby('Sub-Category')['rating_count'].mean().sort_values(ascending=False).head(5).reset_index()
subcat_bottom5 = df_amazon.groupby('Sub-Category')['rating_count'].mean().sort_values(ascending=False).tail(5).reset_index()
fig, ax = plt.subplots(2, 1, figsize = (15, 15))

# sns.barplot to create a bar chart
sns.barplot(ax= ax[0],
            data= subcat_top5,
            x = 'rating_count',
            y = 'Sub-Category',
            palette='bright',
            ).set(title = 'Top 5 Most Rated Sub-Categories',
                              xlabel = 'Rating',
                              ylabel = 'Sub-Category'
                )

sns.barplot(ax= ax[1],
            data= subcat_bottom5,
            x = 'rating_count',
            y = 'Sub-Category',
            palette='bright',
            ).set(title = 'Top 5 Least Rated Sub-Categories',
                              xlabel = 'Rating',
                              ylabel = 'Sub-Category'
                )

# To add data labels to each bar bar_label()

ax[0].bar_label(ax[0].containers[0])
ax[1].bar_label(ax[1].containers[0])

plt.show()
```

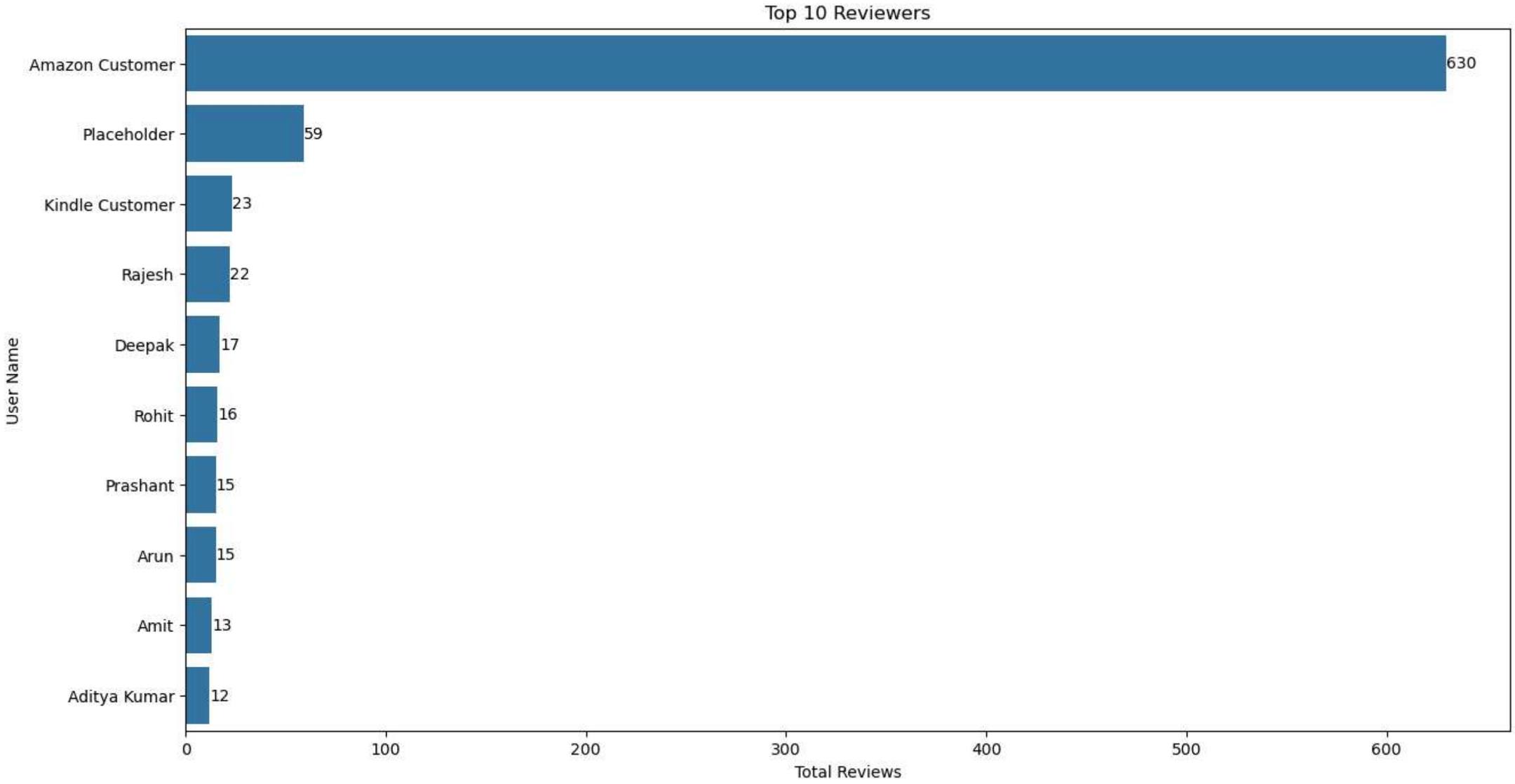

Top 5 Most Rated Sub-Categories
Kitchen & Dining
Accessories
Headphones, Earbuds & Accessories
External Devices & Data Storage
Microphones
270563
Sub-Category
0
50000
100000
150000
200000
250000
Rating


Top 5 Least Rated Sub-Categories
Home Storage & Organization
Tablets
Monitors
Car Accessories
3360.31
Sub-Category

7. Top 10 Customers who reviewed the most products

In [102…

```
reviewers_count = df_customers_info['user_name'].value_counts().reset_index(name = 'counts')
reviewers_count

fig, ax = plt.subplots(1, 1, figsize=(15,8))

sns.barplot(data = reviewers_count.sort_values('counts', ascending=False).head(10),
            x = 'counts',
            y = 'user_name').set(title = 'Top 10 Reviewers',
                                 xlabel = 'Total Reviews',
                                 ylabel = 'User Name')

ax.bar_label(ax.containers[0])
plt.show()
```



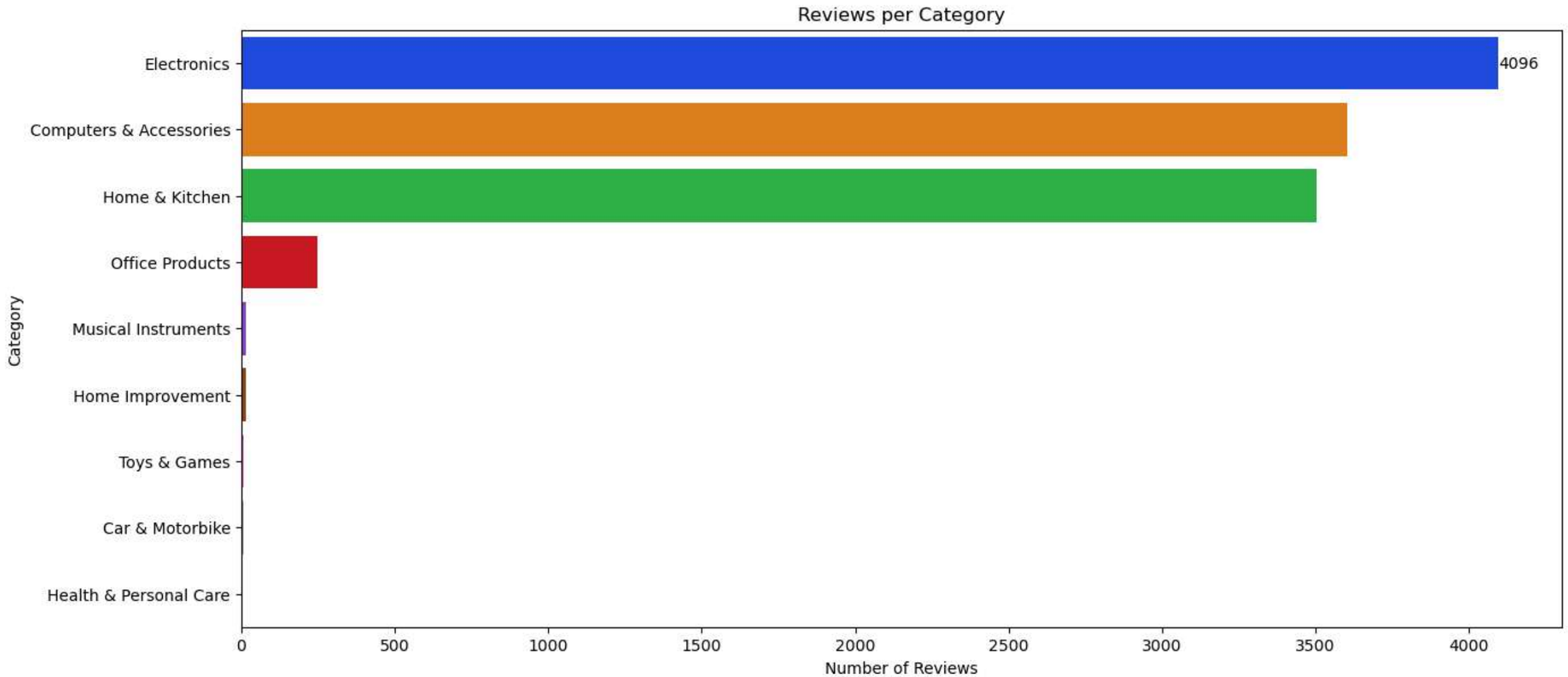
Most of the users are anonymous

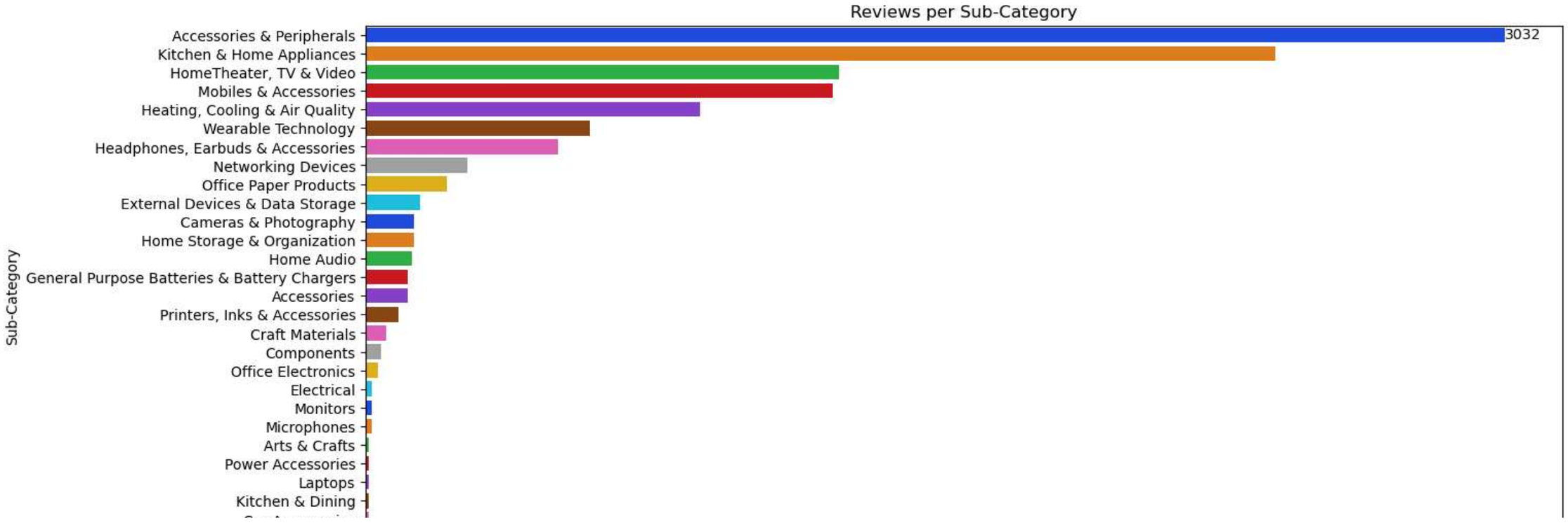
8. Reviews per Category and Sub-Category

In [105…

```
# Total reviews per category and sub-catgory can be calculated by counting the no. of rows in each category
# from the customer_info datafrmae
# We count the number of values per category by value_counts()
# Then we create a dataframe of it using reset_index(name = 'review_count') and name the column as review_count

cat_review_count = df_customers_info['Category'].value_counts().reset_index(name = 'review_count')
subcat_review_count = df_customers_info['Sub-Category'].value_counts().reset_index(name = 'review_count')

fig, ax = plt.subplots(2, 1, figsize = (15, 15))

sns.barplot(ax = ax[0],
            data=  cat_review_count,
```

```
          x = 'review_count',
          y = 'Category',
          palette='bright').set(title = 'Reviews per Category',
                                xlabel = 'Number of Reviews',
                                ylabel = 'Category')

sns.barplot(ax = ax[1],
          data=  subcat_review_count,
          x = 'review_count',
          y = 'Sub-Category',
          palette='bright').set(title = 'Reviews per Sub-Category',
                                xlabel = 'Number of Reviews',
                                ylabel = 'Sub-Category')

ax[0].bar_label(ax[0].containers[0])
ax[1].bar_label(ax[1].containers[0])

plt.show()
```


Reviews per Category
Electronics
Computers & Accessories
Home & Kitchen
Office Products
Musical Instruments
Home Improvement
Toys & Games
Car & Motorbike
Health & Personal Care
Category
4096
0
500
1000
1500
2000
2500
3000
3500
4000
Number of Reviews

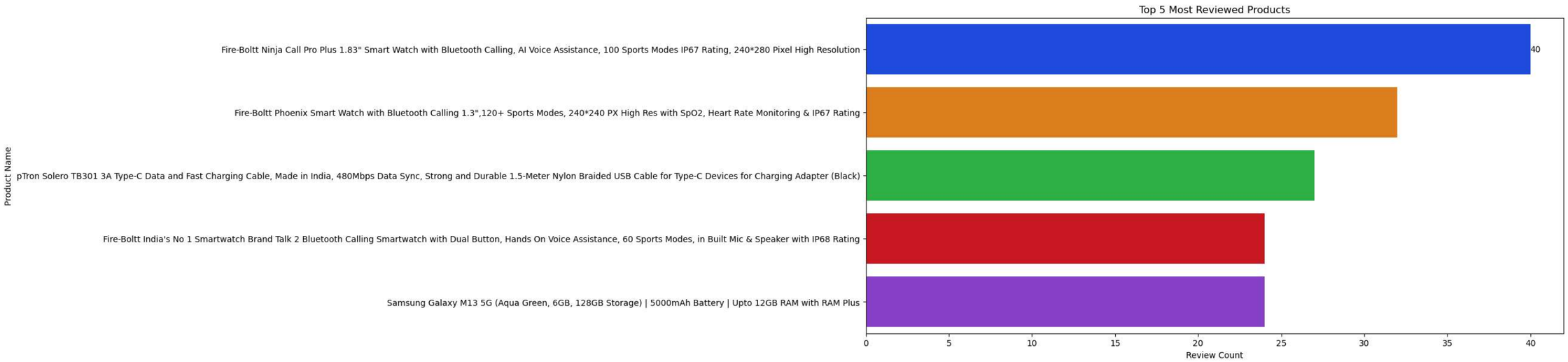

Reviews per Sub-Category
Accessories & Peripherals
Kitchen & Home Appliances
HomeTheater, TV & Video
Mobiles & Accessories
Heating, Cooling & Air Quality
Wearable Technology
Headphones, Earbuds & Accessories
Networking Devices
Office Paper Products
External Devices & Data Storage
Cameras & Photography
Home Storage & Organization
Home Audio
General Purpose Batteries & Battery Chargers
Accessories
Printers, Inks & Accessories
Craft Materials
Components
Office Electronics
Electrical
Monitors
Microphones
Arts & Crafts
Power Accessories
Laptops
Kitchen & Dining
Sub-Category
3032

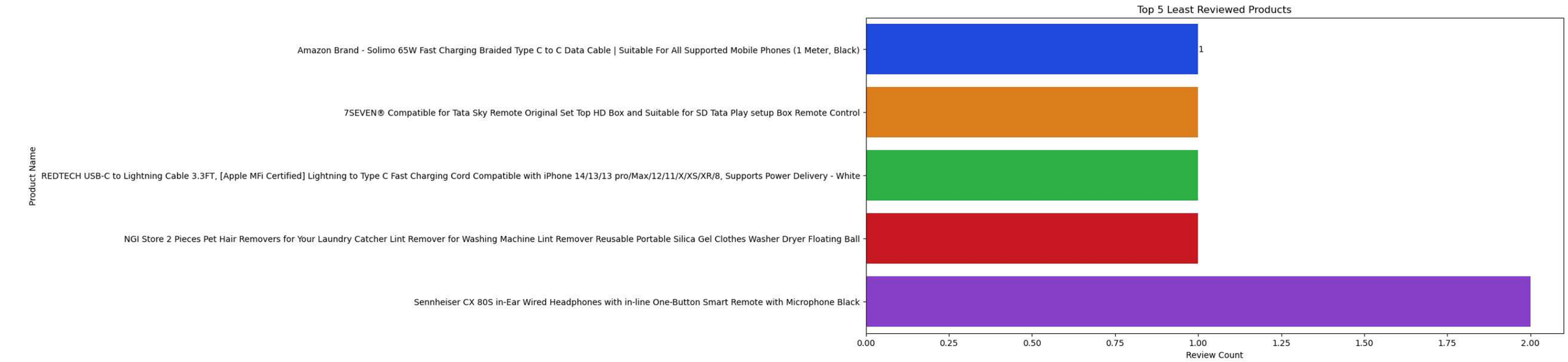
Highly Reviewed Category and Sub-Category are Electronics and Accessories and Pheripherals

9. Top 5 Most and Least Reviewed Products

In [108…

```
product_review_count = df_customers_info['product_name'].value_counts().reset_index(name = 'review_count')

fig, ax = plt.subplots(2, 1, figsize=(15, 15))

sns.barplot(ax = ax[0],
            data= product_review_count.sort_values('review_count', ascending= False).head(5),
            x = 'review_count',
            y = 'product_name',
            palette='bright').set(title = 'Top 5 Most Reviewed Products',
                                  xlabel = 'Review Count',
                                  ylabel = 'Product Name')

sns.barplot(ax = ax[1],
            data= product_review_count.sort_values('review_count', ascending=True).head(5),
            x = 'review_count',
            y = 'product_name',
            palette='bright').set(title = 'Top 5 Least Reviewed Products',
                                  xlabel = 'Review Count',
                                  ylabel = 'Product Name')

ax[0].bar_label(ax[0].containers[0])
ax[1].bar_label(ax[1].containers[0])

plt.show()
```





Top 5 Most and Least Reviewed products are from electronics category

10. Price, discount price and discount percentage distribution

In [111…
```
# for bin range calculation
df_amazon['actual_price'].sort_values(ascending=False).head(5)
```

Out[111…
```
249     139900.0
255      85000.0
283      79990.0
1354     75990.0
568      74999.0
Name: actual_price, dtype: float64
```

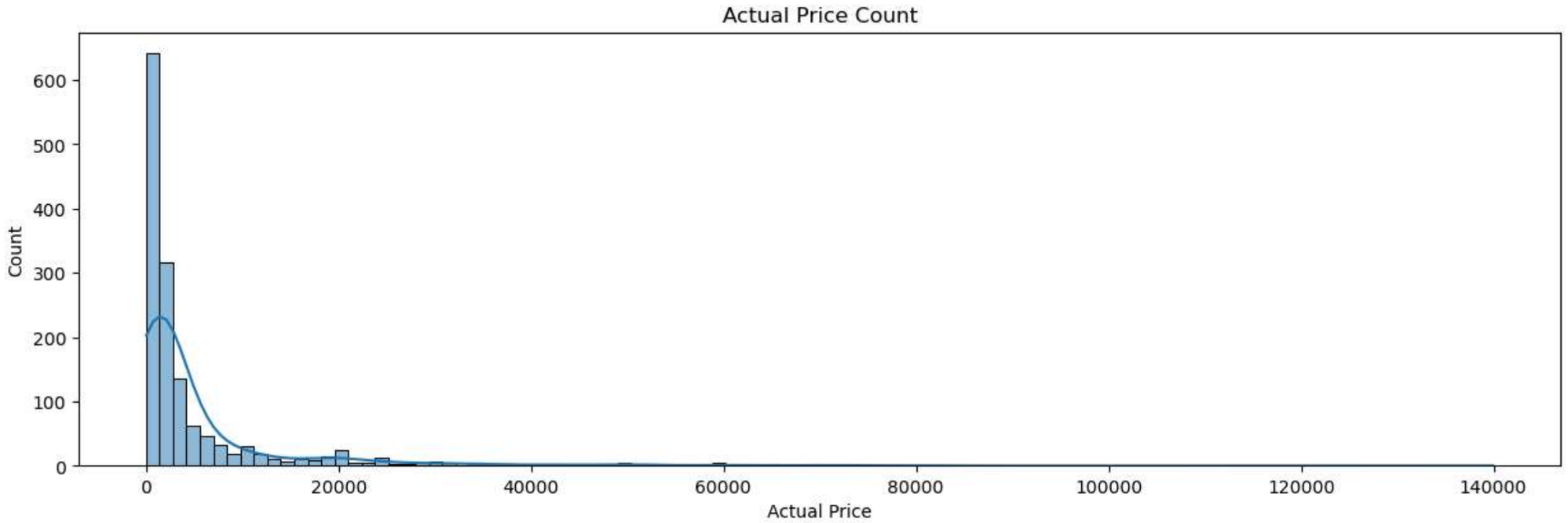
In [112…
```
df_amazon['discounted_price'].sort_values(ascending=False).head(5)
```

Out[112…
```
249    77990.0
325    61999.0
255    54990.0
283    47990.0
192    47990.0
Name: discounted_price, dtype: float64
```

In [113…
```
df_amazon['discount_percentage'].sort_values(ascending=False).head(5)
```

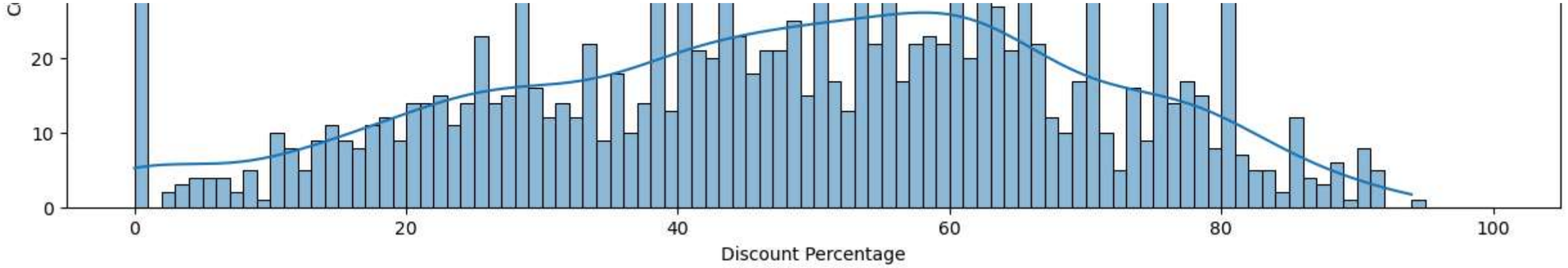
Out[113…
```
695    94.0
368    91.0
380    91.0
334    91.0
372    91.0
Name: discount_percentage, dtype: float64
```

In [114…
```
fig, ax = plt.subplots(3, 1, figsize=(15, 15))

sns.histplot(df_amazon.actual_price,
             kde = True,
             bins = 100,
             binrange = [0,140000],
             ax = ax[0]).set(title = 'Actual Price Count',
                             xlabel = 'Actual Price',
                             ylabel = 'Count')
sns.histplot(df_amazon.discounted_price,
             kde = True,
             bins = 100,
             binrange = [0, 80000],
             ax = ax[1]).set(title = 'Discounted Price Count',
                             xlabel = 'Discounted Price',
                             ylabel = 'Count')
sns.histplot(df_amazon.discount_percentage,
             kde = True,
             bins = 100,
             binrange = [0,100],
             ax = ax[2]).set(title = 'Discount Percentage Count',
                             xlabel = 'Discount Percentage',
                             ylabel = 'Count')

plt.show()
```


Actual Price Count
Count
Actual Price
0
100
200
300
400
500
600
0
20000
40000
60000
80000
100000
120000
140000


Discounted Price Count
Count
Discounted Price
0
100
200
300
400
500
600
700
0
10000
20000
30000
40000
50000
60000
70000
80000

Discount Percentage Count
30
40
50



Price Distribution shows most of the products have price in range of Rs. 100 to Rs. 5000

Discounted Price Distribution shows that higher the price of product higher is the discount

Discounted Percentage shows products are mostly purchased when they have disount 50% to 65 %

In [ ]: