mba-amazon-product

November 27, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
df = pd.read_csv('aamazon.csv')
df.head()
```

[2]:

	product_id	product_name	category	discounted_price	actual_price	discount_percentage	rating	rating_count	about_product	user_id
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…	
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5…	Computers&Accessories|Accessories&Peripherals|…	199	349	43%	4.0	43,994	Compatible with all Type C enabled devices, be…	
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…	
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…	
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…	

```
0  AG3D6O4STAQKAY2UVGEUV46KN35Q,AHMY5CWJMMK5BJRBB…
1  AECPFYFQVRUWC3KGNLJIOREFP5LQ,AGYYVPDD7YG7FYNBX…
2  AGU3BBQ2V2DDAMOAKGFAWDDQ6QHA,AESFLDV2PT363T2AQ…
3  AEWAZDZZJLQUYVOVGBEUKSLXHQ5A,AG5HTSFRRE6NL3M5S…
4  AE3Q6KSUK5P75D5HFYHCRAOLODSA,AFUGIFH5ZAFXRDSZH…

                                              user_name  \
0  Manav,Adarsh gupta,Sundeep,S.Sayeed Ahmed,jasp…
1  ArdKn,Nirbhay kumar,Sagar Viswanathan,Asp,Plac…
2  Kunal,Himanshu,viswanath,sai niharka,saqib mal…
3  Omkar dhale,JD,HEMALATHA,Ajwadh a.,amar singh …
4  rahuls6099,Swasat Borah,Ajay Wadke,Pranali,RVK…

                                              review_id  \
0  R3HXWT0LRP0NMF,R2AJM3LFTLZHFO,R6AQJGUP6P86,R1K…
1  RGIQEG07R9HS2,R1SMWZQ86XIN8U,R2J3Y1WL29GWDE,RY…
2  R3J3EQQ9TZI5ZJ,R3E7WBGK7ID0KV,RWU79XKQ6I1QF,R2…
3  R3EEUZKKK9J36I,R3HJVYCLYOY554,REDECAZ7AMPQC,R1…
4  R1BP4L2HH9TFUP,R16PVJEXKV6QZS,R2UPDB81N66T4P,R…

                                         review_title  \
0  Satisfied,Charging is really fast,Value for mo…
1  A Good Braided Cable for Your Type C Device,Go…
2  Good speed for earlier versions,Good Product,W…
3  Good product,Good one,Nice,Really nice product…
4  As good as original,Decent,Good one for second…

                                       review_content  \
0  Looks durable Charging is fine tooNo complains…
1  I ordered this cable to connect my phone to An…
2  Not quite durable and sturdy,https://m.media-a…
3  Good product,long wire,Charges good,Nice,I bou…
4  Bought this instead of original apple, does th…

                                              img_link  \
0  https://m.media-amazon.com/images/W/WEBP_40237…
1  https://m.media-amazon.com/images/W/WEBP_40237…
2  https://m.media-amazon.com/images/W/WEBP_40237…
3  https://m.media-amazon.com/images/I/41V5FtEWPk…
4  https://m.media-amazon.com/images/W/WEBP_40237…

                                         product_link
0  https://www.amazon.in/Wayona-Braided-WN3LG1-Sy…
1  https://www.amazon.in/Ambrane-Unbreakable-Char…
2  https://www.amazon.in/Sounce-iPhone-Charging-C…
3  https://www.amazon.in/Deuce-300-Resistant-Tang…
4  https://www.amazon.in/Portronics-Konnect-POR-1…
```

[3]:
```
df = df[['product_id', 'product_name', 'discounted_price',
       'actual_price', 'discount_percentage', 'rating', 'rating_count',␣
 ↪'user_id', 'category']]
df.columns
```

[3]:
```
Index(['product_id', 'product_name', 'discounted_price', 'actual_price',
       'discount_percentage', 'rating', 'rating_count', 'user_id', 'category'],
      dtype='object')
```

1 Data Cleaning & Preparation

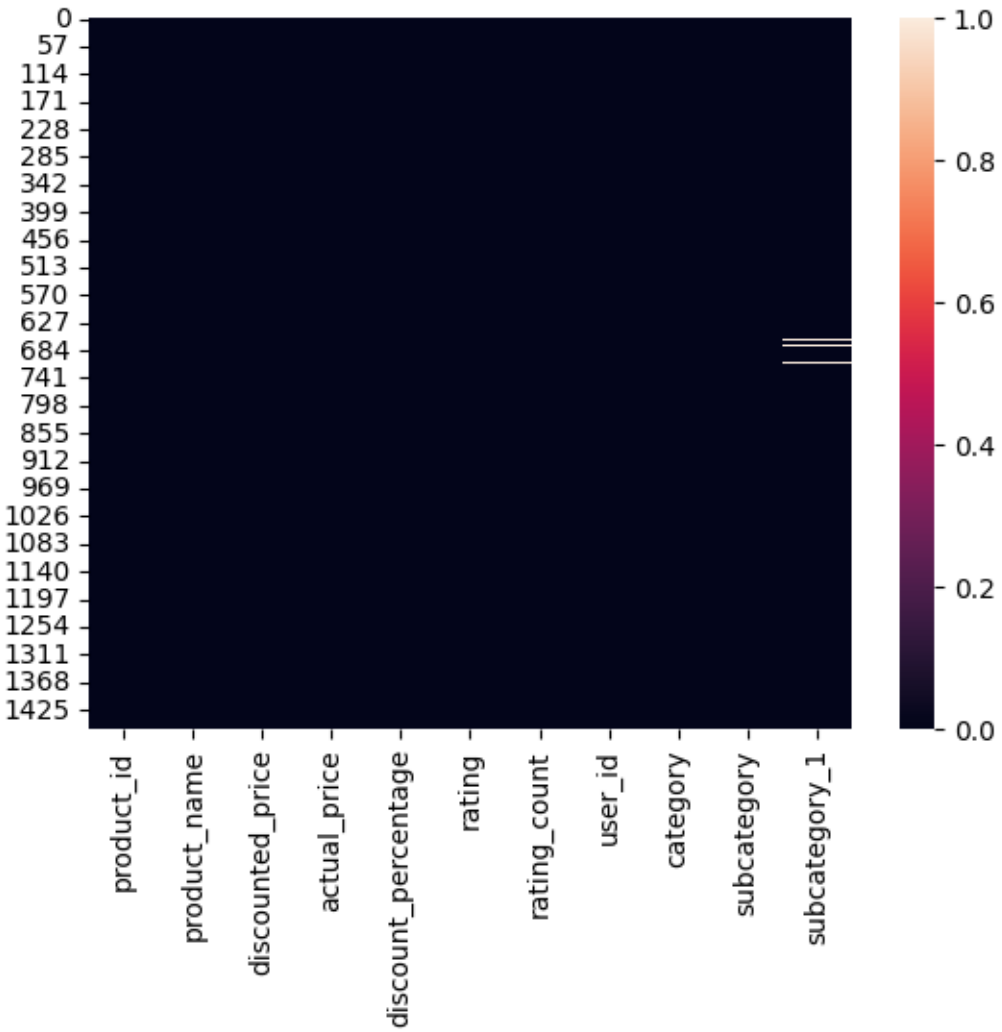
[4]:
```
# Split the 'category' into separate columns
for i in range(1, 4):
    column_name = f'subcategory_{i}'
    df[column_name] = df['category'].str.split("|").str[-i]

df.columns
```

[4]:
```
Index(['product_id', 'product_name', 'discounted_price', 'actual_price',
       'discount_percentage', 'rating', 'rating_count', 'user_id', 'category',
       'subcategory_1', 'subcategory_2', 'subcategory_3'],
      dtype='object')
```

[5]:
```
df['discounted_price'] = df['discounted_price'].apply(lambda x: x.
 ↪replace(' ',''))
df['actual_price'] = df['actual_price'].apply(lambda x: x.replace(' ',''))
df['discount_percentage'] = df['discount_percentage'].apply(lambda x: x.
 ↪replace('%',''))
df = df.drop('category', axis=1)

# renaming specific columns
df = df.rename(columns={'subcategory_1': 'category', 'subcategory_2':
 ↪'subcategory','subcategory_3' :'subcategory_1'})



df.columns
```

[5]:
```
Index(['product_id', 'product_name', 'discounted_price', 'actual_price',
       'discount_percentage', 'rating', 'rating_count', 'user_id', 'category',
       'subcategory', 'subcategory_1'],
      dtype='object')
```

[6]:
```
sns.heatmap(df.isnull())
plt.show()
```



[7]:
```
df.isna().sum()
```

[7]:
```
product_id             0
product_name           0
discounted_price       0
actual_price           0
discount_percentage    0
rating                 0
rating_count           2
user_id                0
category               0
subcategory            0
subcategory_1          8
dtype: int64
```

[8]:
```
df.isna().sum()
```

[8]:
```
product_id             0
product_name           0
discounted_price       0
actual_price           0
discount_percentage    0
rating                 0
rating_count           2
user_id                0
category               0
subcategory            0
subcategory_1          8
dtype: int64
```

[9]:
```
df =df.dropna()
df.isna().sum()
```

[9]:
```
product_id             0
product_name           0
discounted_price       0
actual_price           0
discount_percentage    0
rating                 0
rating_count           0
user_id                0
category               0
subcategory            0
subcategory_1          0
dtype: int64
```

[10]:
```
df.dtypes
```

[10]:
```
product_id             object
product_name           object
discounted_price       object
actual_price           object
discount_percentage    object
rating                 object
rating_count           object
user_id                object
category               object
subcategory            object
subcategory_1          object
dtype: object
```

[11]:
```
def replace_comma(df,columns):
    for col in columns:
        df[col] = df[col].str.replace(',','')

columns = ['actual_price', 'rating','rating_count', 'discounted_price']
replace_comma(df,columns)
```

[12]:
```
def change_type(df,columns):
    for col in columns:
        df[col] = pd.to_numeric(df[col], errors='coerce')
        df[col] = np.round(df[col].fillna(0)).astype('Int64')
columns  = ['actual_price','rating_count',␣
 ↪'discounted_price','discount_percentage']
change_type(df, columns)
```

[13]:
```
df['discount_percentage'] = df['discount_percentage'] / 100

df['rating'] = df['rating'].str.replace('|','0')
df['rating'] = df['rating'].astype(float)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1455 entries, 0 to 1464
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   product_id           1455 non-null   object
 1   product_name         1455 non-null   object
 2   discounted_price     1455 non-null   Int64
 3   actual_price         1455 non-null   Int64
 4   discount_percentage  1455 non-null   Float64
 5   rating               1455 non-null   float64
 6   rating_count         1455 non-null   Int64
 7   user_id              1455 non-null   object
 8   category             1455 non-null   object
 9   subcategory          1455 non-null   object
 10  subcategory_1        1455 non-null   object
dtypes: Float64(1), Int64(3), float64(1), object(6)
memory usage: 142.1+ KB
```

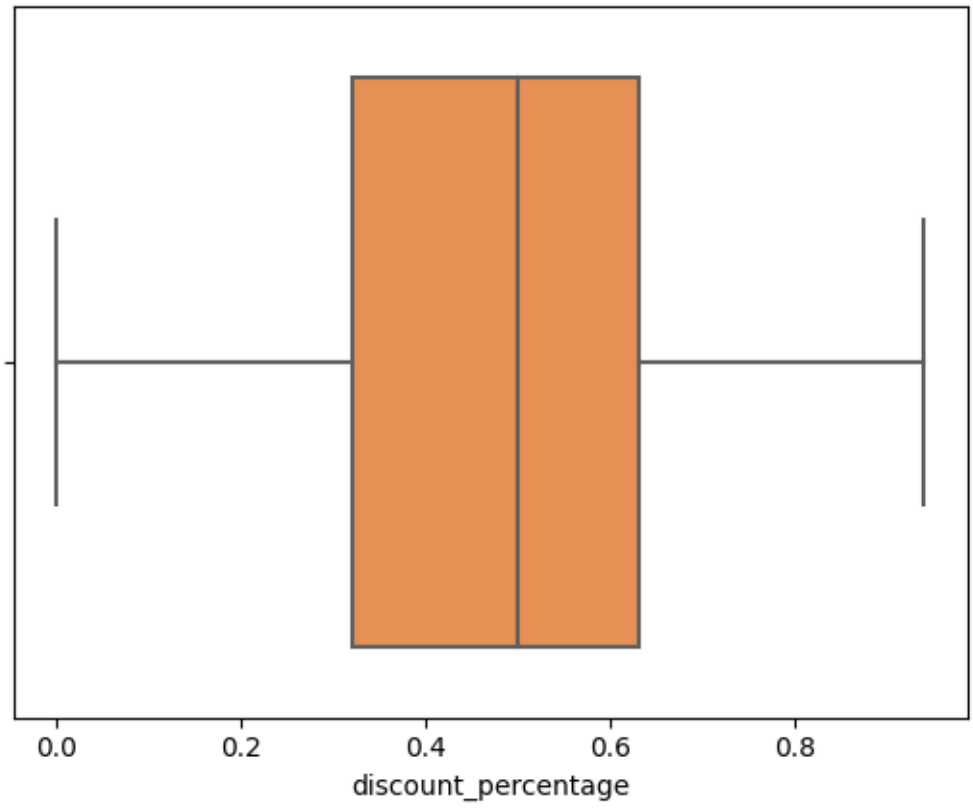
2 Data Exploration & Visualization

[14]:
```
df.head()
```

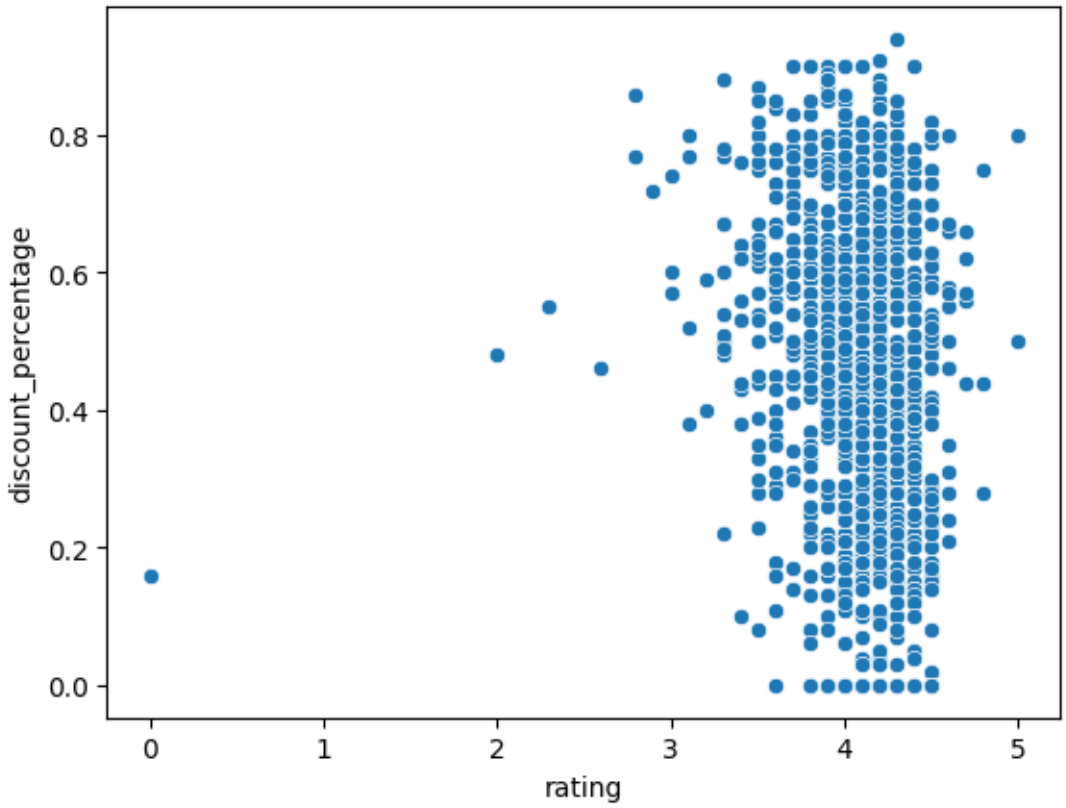
[14]:

	product_id	product_name	discounted_price	actual_price	discount_percentage	rating	rating_count	user_id	category	subcategory	subcategory_1
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha…	399	1099	0.64	4.2	24269	AG3D6O4STAQKAY2UVGEUV46KN35Q,AHMY5CWJMMK5BJRBB…	USBCables	Cables	Cables&Accessories
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5…	199	349	0.43	4.0	43994	AECPFYFQVRUWC3KGNLJIOREFP5LQ,AGYYVPDD7YG7FYNBX…	USBCables	Cables	Cables&Accessories
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U…	199	1899	0.9	3.9	7928	AGU3BBQ2V2DDAMOAKGFAWDDQ6QHA,AESFLDV2PT363T2AQ…	USBCables	Cables	Cables&Accessories
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S…	329	699	0.53	4.2	94363	AEWAZDZZJLQUYVOVGBEUKSLXHQ5A,AG5HTSFRRE6NL3M5S…	USBCables	Cables	Cables&Accessories
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P…	154	399	0.61	4.2	16905	AE3Q6KSUK5P75D5HFYHCRAOLODSA,AFUGIFH5ZAFXRDSZH…	USBCables	Cables	Cables&Accessories

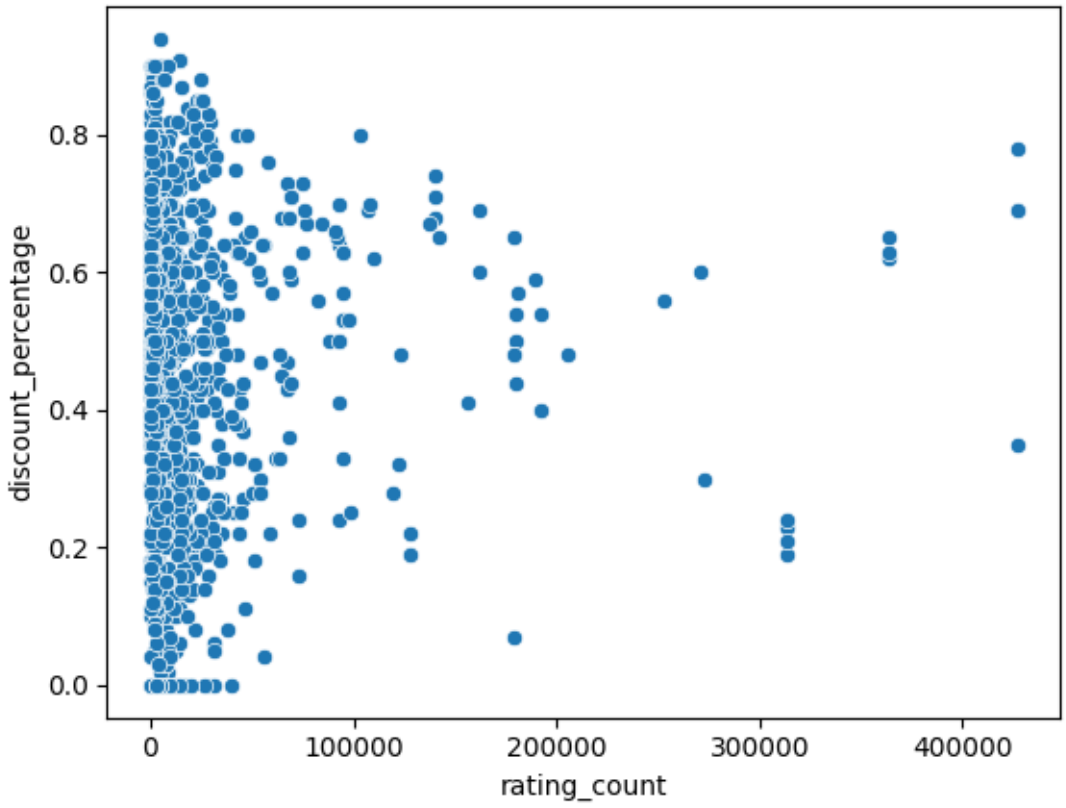
[15]:
```
sns.boxplot(df, x='discount_percentage',palette = 'Oranges');
```



[16]:
```
sns.scatterplot(df,x='rating', y='discount_percentage')
plt.show()
```



[17]:
```
sns.scatterplot(df,x='rating_count', y='discount_percentage')
plt.show()
```



[18]:
```
df.groupby(['category','subcategory'])['rating_count'].sum().
  ↪sort_values(ascending=False).head(5)
```

[18]:
```
category      subcategory
In-Ear        Headphones                  4204939
USBCables     Cables                      3547816
Smartphones   Smartphones&BasicMobiles    2493269
HDMICables    Cables                      1906054
SmartWatches  WearableTechnology          1644476
Name: rating_count, dtype: Int64
```

[19]:
```
df.columns
```

[19]:
```
Index(['product_id', 'product_name', 'discounted_price', 'actual_price',
       'discount_percentage', 'rating', 'rating_count', 'user_id', 'category',
       'subcategory', 'subcategory_1'],
      dtype='object')
```

[20]:
```
df.groupby('category')['discount_percentage'].mean().nlargest(10)
```

[20]:
```
category
CableConnectionProtectors         0.9
Earpads                           0.9
PhoneCharms                       0.9
DustCovers                      0.875
Shower&WallMounts                0.82
Adapters                     0.803333
InternalHardDrives                0.8
USBtoUSBAdapters                0.785
Stands                       0.758182
NotebookComputerStands       0.756667
Name: discount_percentage, dtype: Float64
```

[21]:
```
df.groupby('subcategory')['discounted_price'].sum().nlargest(10)
```

[21]:
```
subcategory
Televisions                   1608017
Smartphones&BasicMobiles      1086799
SmallKitchenAppliances         248273
WearableTechnology             177817
WaterHeaters&Geysers           137634
Vacuums                        124616
Cables                          98043
RoomHeaters                     97371
WaterPurifiers&Accessories      91067
Headphones                      62118
Name: discounted_price, dtype: Int64
```

[22]:
```
df.columns
```

[22]:
```
Index(['product_id', 'product_name', 'discounted_price', 'actual_price',
       'discount_percentage', 'rating', 'rating_count', 'user_id', 'category',
       'subcategory', 'subcategory_1'],
      dtype='object')
```

[23]:
```
df.groupby('category')['discount_percentage'].mean().round(1).nlargest(10)
```

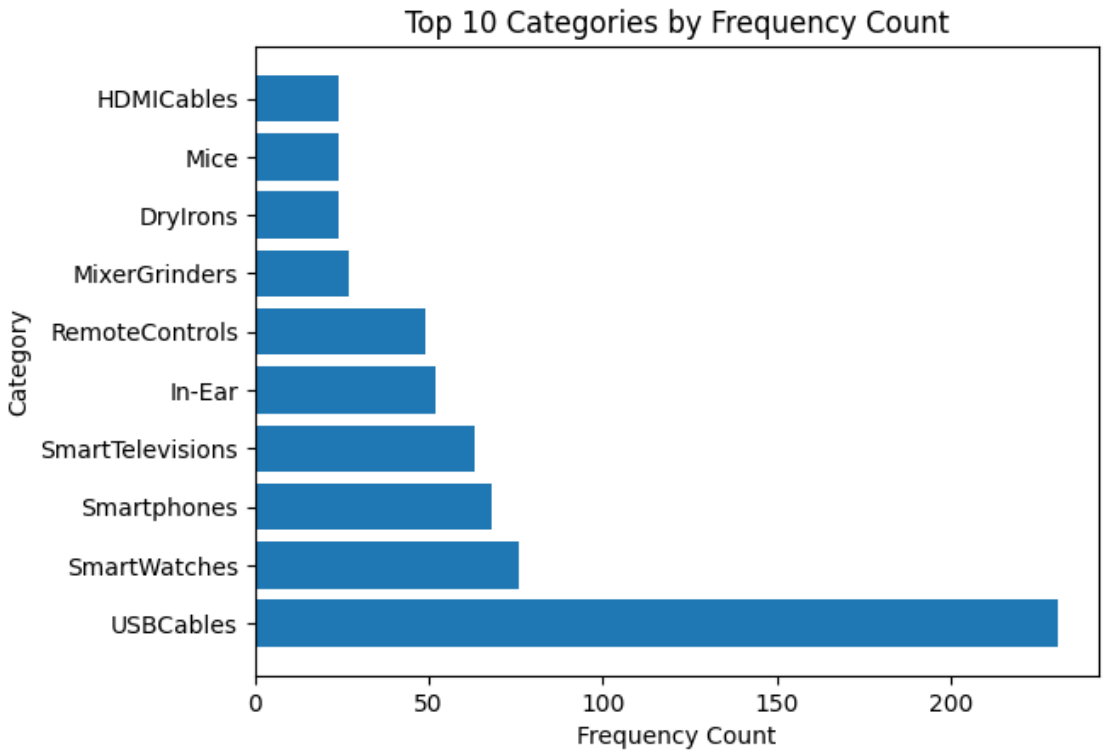
[23]:
```
category
CableConnectionProtectors    0.9
DustCovers                   0.9
Earpads                      0.9
PhoneCharms                  0.9
Adapters                     0.8
Caddies                      0.8
InternalHardDrives           0.8
NotebookComputerStands       0.8
Shower&WallMounts            0.8
```

```
Stands                          0.8
Name: discount_percentage, dtype: Float64
```

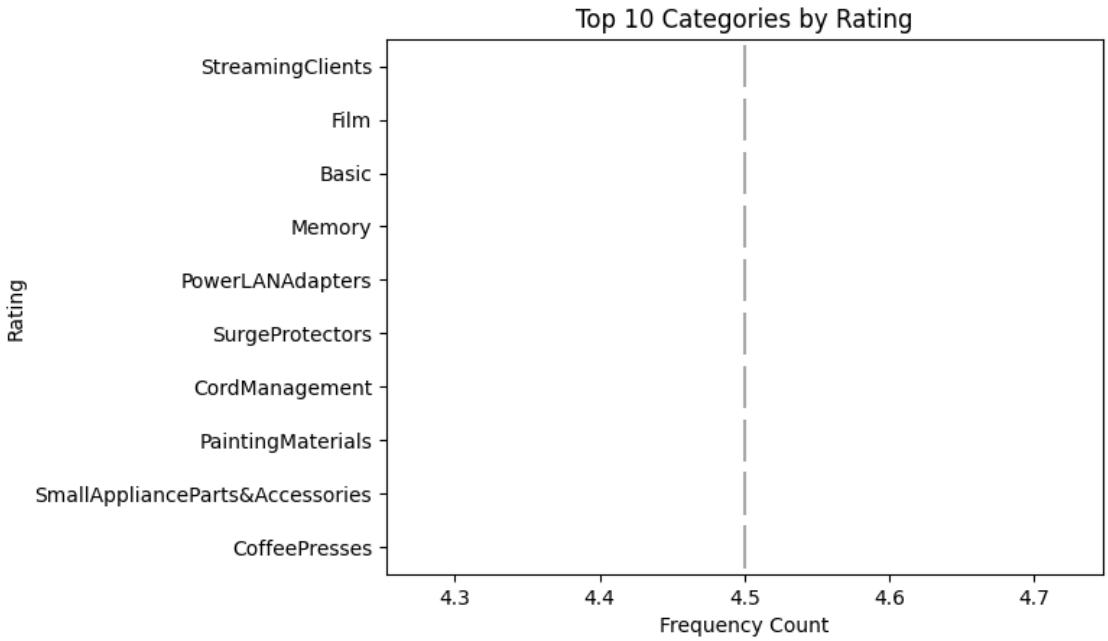
[24]:
```
df.groupby('subcategory')['discount_percentage'].mean().round(1).nlargest(10)
```

[24]:
```
subcategory
Cables&Accessories                0.9
Décor                             0.9
Headphones,Earbuds&Accessories    0.9
Adapters                          0.8
Cameras&Photography               0.8
HardDriveAccessories              0.8
Bags,Cases&Sleeves                0.7
Cables&Adapters                   0.7
Cases&Covers                      0.7
Keyboard&MiceAccessories          0.7
Name: discount_percentage, dtype: Float64
```

[25]:
```
category_counts = df['category'].value_counts().sort_values(ascending = False)
top_10_categories = category_counts.head(10)

plt.barh(top_10_categories.index, top_10_categories.values)
plt.xlabel('Frequency Count')
plt.ylabel('Category')
plt.title('Top 10 Categories by Frequency Count')
plt.show()
```



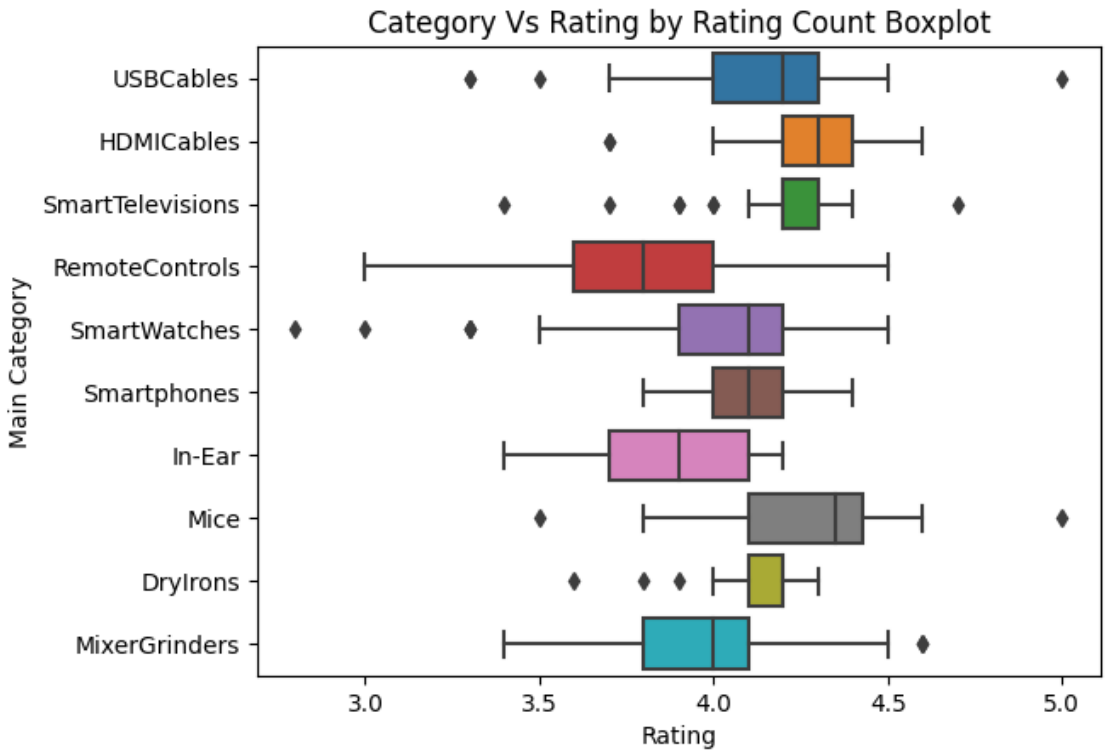
[26]:
```
rating = df.groupby('category')['rating'].mean().sort_values(ascending = False).
  ↪head(10).index
subset_dff = df[df['category'].isin(rating)]
sns.boxenplot(data = subset_dff,x='rating' ,y = 'category')
plt.xlabel('Frequency Count')
plt.ylabel('Rating')
plt.title('Top 10 Categories by Rating')
plt.show()
```



[27]:
```
df.columns
```

[27]:
```
Index(['product_id', 'product_name', 'discounted_price', 'actual_price',
       'discount_percentage', 'rating', 'rating_count', 'user_id', 'category',
       'subcategory', 'subcategory_1'],
      dtype='object')
```

[28]:
```
top_categories = df['category'].value_counts().head(10).index
subset_df = df[df['category'].isin(top_categories)]

sns.boxplot(data=subset_df, x='rating', y='category', orient = 'h')
plt.title("Category Vs Rating by Rating Count Boxplot")
plt.xlabel("Rating")
plt.ylabel("Main Category")
plt.show()
```



[29]:
```
df.category.value_counts().head(10)
```

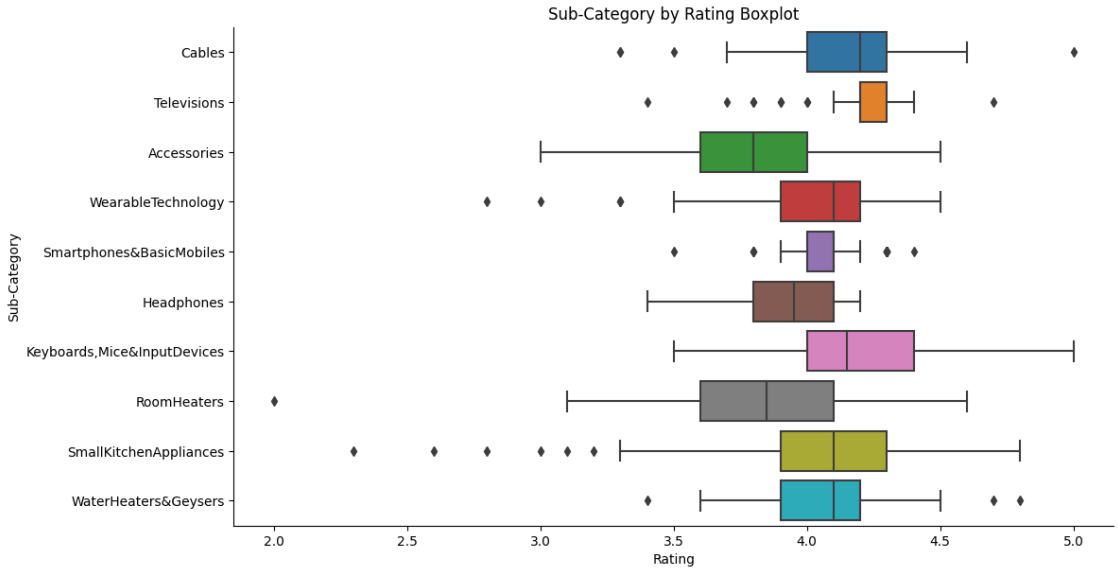
[29]:
```
category
USBCables           231
SmartWatches         76
Smartphones          68
SmartTelevisions     63
In-Ear               52
RemoteControls       49
MixerGrinders        27
DryIrons             24
Mice                 24
HDMICables           24
Name: count, dtype: int64
```

[30]:
```
top_subcategory = df['subcategory'].value_counts().head(10).index
sub_cat_filtered = df[df['subcategory'].isin(top_subcategory)]

# Set up the figure and axis
plt.figure(figsize=(5, 3))

# Create a box plot using Seaborn's catplot
```

```
sns.catplot(data=sub_cat_filtered, x='rating', y='subcategory', kind='box',␣
 ↪height=6, aspect=2)

# Set the title and labels
plt.title("Sub-Category by Rating Boxplot")
plt.xlabel("Rating")
plt.ylabel("Sub-Category")

# Show the plot
plt.show()
```

```
C:\Users\athar\AppData\Local\Programs\Python\Python311\Lib\site-
packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to
tight
  self._figure.tight_layout(*args, **kwargs)
```

```
<Figure size 500x300 with 0 Axes>
```



2.1 Recommendation

[31]:
```
from prettytable import PrettyTable

# Load data from CSV
df = pd.read_csv('aamazon.csv')

# Convert 'DiscountRate' from percentage to float, handling non-numeric␣
 ↪characters
numeric_cols = ['discount_percentage', 'rating']
```

```
df[numeric_cols] = df[numeric_cols].apply(lambda x: x.str.rstrip('%')).apply(pd.
 ↪to_numeric, errors='coerce')

# Convert 'rating_count' from comma-separated to numeric
df['rating_count'] = df['rating_count'].replace({',': ''}, regex=True).
 ↪astype(float)

df.fillna(0, inplace=True)  # Replace NaN values with 0 or choose another␣
 ↪default value

# Define weights for each feature
weight_rating = 0.4
weight_rating_count = 0.4
weight_discount = 0.2

for i in range(1, 4):
    column_name = f'subcategory_{i}'
    df[column_name] = df['category'].str.split("|").str[-i]

# Create a weighted sum column
df['WeightedSum'] = (
    weight_rating * df['rating'] +
    weight_rating_count * df['rating_count'] +
    weight_discount * df['discount_percentage']
)

# Round the 'WeightedSum' column to 2 decimal places
df['WeightedSum'] = df['WeightedSum'].round(2)

# Sort the DataFrame by the weighted sum in descending order
df = df.sort_values(by='WeightedSum', ascending=False)

# Display only the top 50 recommended products
top_50_df = df.head(50)

# Assuming top_50_df is a pandas DataFrame
table_data = [['Product ID', 'Subcategory', 'Discount %', 'Rating', 'Rating␣
 ↪Count', 'Weighted Sum']]

# Populate the table data
for _, row in top_50_df.iterrows():
    table_data.append([row['product_id'], row['subcategory_1'],␣
 ↪row['discount_percentage'], row['rating'], row['rating_count'],␣
 ↪row['WeightedSum']])

fig, ax = plt.subplots(figsize=(10, 4))
ax.axis('off')  # Turn off the axis
```

```
# Create the table
table = ax.table(cellText=table_data, loc='center', cellLoc='center',␣
 ↪colLabels=None)

# Style the table
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.2)

plt.show()
```

Product ID	Subcategory	Discount %	Rating	Rating Count	Weighted Sum
B014I8SX4Y	HDMICables	78	4.4	426973.0	170806.56
B07KSMBL2H	HDMICables	69	4.4	426973.0	170804.76
B07KSMBL2H	HDMICables	69	4.4	426972.0	170804.36
B014I8SSD0	HDMICables	35	4.4	426973.0	170797.96
B07GPXXNNG	In-Ear	65	4.1	363713.0	145499.84
B07GQD4K6L	In-Ear	62	4.1	363713.0	145499.24
B071Z8M4KX	In-Ear	63	4.1	363711.0	145498.64
B09GFLXVH9	Smartphones	24	4.1	313836.0	125540.84
B09GFPVD9Y	Smartphones	23	4.1	313836.0	125540.64
B09GFPN6TP	Smartphones	21	4.1	313832.0	125538.64
B09GFM8CGS	Smartphones	19	4.1	313832.0	125538.24
B01MF8MB65	In-Ear	30	4.1	273189.0	109283.24
B01LWYDEQ7	Choppers	60	4.1	270563.0	108238.84
B005FYNT3G	PenDrives	56	4.3	253105.0	101254.92
B09X7DY7Q4	MicroSD	48	4.5	205052.0	82032.2
B01DF26V7A	In-Ear	54	4.1	192589.0	77048.04
B01DEWVZ2C	In-Ear	40	4.1	192590.0	77045.64
B01DEWVZ2C	In-Ear	40	4.1	192587.0	77044.44
B01N6LU1VF	PenDrives	59	4.3	189104.0	75655.12
B08JQN8DGZ	In-Ear	57	3.8	180998.0	72412.12
B0088TKTY2	WirelessUSBAdapters	54	4.2	179691.0	71888.88
B008IFXQFU	WirelessUSBAdapters	50	4.2	179691.0	71888.08
B002SZEOLG	WirelessUSBAdapters	44	4.2	179692.0	71887.28
B08HVL8QN3	PowerBanks	48	4.3	178912.0	71576.12
B08HVJCW95	PowerBanks	48	4.3	178912.0	71576.12
B08HV83HL3	PowerBanks	7	4.3	178912.0	71567.92
B07DC4RZPY	USBCables	65	4.1	178817.0	71541.44
B08H9Z3XQW	In-Ear	69	4.1	161677.0	64686.24
B07S9S86BF	In-Ear	60	4.1	161679.0	64685.24
B00A0VCJPI	Repeaters&Extenders	41	4.2	156638.0	62665.08
B08TV2P1N8	In-Ear	65	4.1	141841.0	56751.04
B09MT84WV5	MicroSD	74	4.3	140035.0	56030.52
B09MT84WV5	MicroSD	71	4.3	140036.0	56030.32
B09MT6XSFW	MicroSD	68	4.3	140036.0	56029.72
B09N3ZNHTY	In-Ear	67	3.9	136954.0	54796.56
B09YDFDVNS	BasicMobiles	22	4.0	128311.0	51330.4
B09YDFKJF8	BasicMobiles	22	4.0	128311.0	51330.4
B09V2PZDX8	BasicMobiles	19	4.0	128311.0	51329.8
B09V2Q4QVQ	BasicMobiles	19	4.0	128311.0	51329.8
B07WMS7TWB	ElectricKettles	48	3.9	123365.0	49357.16
B01HGCLUH6	Routers	32	4.2	122478.0	48999.28
B07CD2BN46	In-Ear	28	4.1	119466.0	47793.64
B092X94QNQ	In-Ear	62	4.1	109864.0	43959.64
B00NH11KIK	USBCables	70	4.5	107687.0	43090.6
B00NH11KIK	USBCables	70	4.5	107686.0	43090.2
B07PR1CL3S	On-Ear	69	4.1	107151.0	42875.84
B07LG59NPV	In-Ear	80	3.8	103052.0	41238.32
B083T5G5PM	In-Ear	25	4.1	98250.0	39306.64
B01FSYQ2A4	On-Ear	53	4.1	97175.0	38882.24
B01FSYQ2A4	On-Ear	53	4.1	97174.0	38881.84

[32]:
```
import matplotlib.pyplot as plt
from matplotlib.table import Table

# Group by 'Subcategory' and iterate over groups
for subcategory, group in top_50_df.groupby('subcategory_1'):
    # Drop duplicates based on 'ProductID'
    group = group.drop_duplicates(subset='product_id')

    # Sort the DataFrame by 'Weighted_Average' in descending order
    sorted_group = group.sort_values(by='WeightedSum', ascending=False)

    # Create a list to store table data
    table_data = [['ProductID', 'Actual Price', 'Discounted Price', 'Rating']]

    # Populate the table data
    for _, row in sorted_group.iterrows():
        table_data.append([row['product_id'], row['actual_price'],␣
↪row['discounted_price'], row['rating']])

    # Create a Matplotlib figure and axis
    fig, ax = plt.subplots(figsize=(8, 3))
    ax.axis('off')  # Turn off the axis

    # Create the table using Matplotlib's table function
    table = ax.table(cellText=table_data, loc='center', colLabels=None,␣
↪cellLoc='center', bbox=[0, 0, 1, 1])

    # Style the table
    table.auto_set_font_size(False)
    table.set_fontsize(8)

    # Add a title for the current subcategory
    ax.set_title(f"Top Products in {subcategory}", fontsize=12, color='blue', ␣
↪pad=20)

    # Show the table
    plt.show()
```

Top Products in BasicMobiles

ProductID	Actual Price	Discounted Price	Rating
B09YDFDVNS	₹1,699	₹1,324	4.0
B09YDFKJF8	₹1,699	₹1,324	4.0
B09V2PZDX8	₹1,599	₹1,299	4.0
B09V2Q4QVQ	₹1,599	₹1,299	4.0

Top Products in Choppers

ProductID	Actual Price	Discounted Price	Rating
B01LWYDEQ7	₹495	₹199	4.1

Top Products in ElectricKettles

ProductID	Actual Price	Discounted Price	Rating
B07WMS7TWB	₹1,245	₹649	3.9

Top Products in HDMICables

ProductID	Actual Price	Discounted Price	Rating
B014I8SX4Y	₹1,400	₹309	4.4
B07KSMBL2H	₹700	₹219	4.4
B014I8SSD0	₹475	₹309	4.4

Top Products in In-Ear

ProductID	Actual Price	Discounted Price	Rating
B07GPXXNNG	₹999	₹349	4.1
B07GQD4K6L	₹999	₹379	4.1
B071Z8M4KX	₹999	₹365	4.1
B01MF8MB65	₹999	₹699	4.1
B01DF26V7A	₹1,299	₹599	4.1
B01DEWVZ2C	₹999	₹599	4.1
B08JQN8DGZ	₹2,990	₹1,299	3.8
B08H9Z3XQW	₹1,490	₹455	4.1
B07S9S86BF	₹1,490	₹599	4.1
B08TV2P1N8	₹3,990	₹1,399	4.1
B09N3ZNHTY	₹4,490	₹1,499	3.9
B07CD2BN46	₹599	₹429	4.1
B092X94QNQ	₹3,990	₹1,499	4.1
B07LG59NPV	₹4,499	₹899	3.8
B083T5G5PM	₹1,990	₹1,490	4.1

Top Products in MicroSD

ProductID	Actual Price	Discounted Price	Rating
B09X7DY7Q4	₹1,800	₹939	4.5
B09MT84WV5	₹3,999	₹1,059	4.3
B09MT6XSFW	₹1,899	₹599	4.3

Top Products in On-Ear

ProductID	Actual Price	Discounted Price	Rating
B07PR1CL3S	₹3,990	₹1,220	4.1
B01FSYQ2A4	₹2,990	₹1,399	4.1

Top Products in PenDrives

ProductID	Actual Price	Discounted Price	Rating
B005FYNT3G	₹650	₹289	4.3
B01N6LU1VF	₹1,400	₹579	4.3

Top Products in PowerBanks

ProductID	Actual Price	Discounted Price	Rating
B08HVL8QN3	₹2,199	₹1,149	4.3
B08HVJCW95	₹2,199	₹1,149	4.3
B08HV83HL3	₹2,199	₹2,049	4.3

Top Products in Repeaters&Extenders

ProductID	Actual Price	Discounted Price	Rating
B00A0VCJPI	₹2,499	₹1,469	4.2

Top Products in Routers

ProductID	Actual Price	Discounted Price	Rating
B01HGCLUH6	₹1,699	₹1,149	4.2

Top Products in Smartphones

ProductID	Actual Price	Discounted Price	Rating
B09GFLXVH9	₹8,499	₹6,499	4.1
B09GFPVD9Y	₹10,999	₹8,499	4.1
B09GFPN6TP	₹9,499	₹7,499	4.1
B09GFM8CGS	₹7,999	₹6,499	4.1

Top Products in USBCables

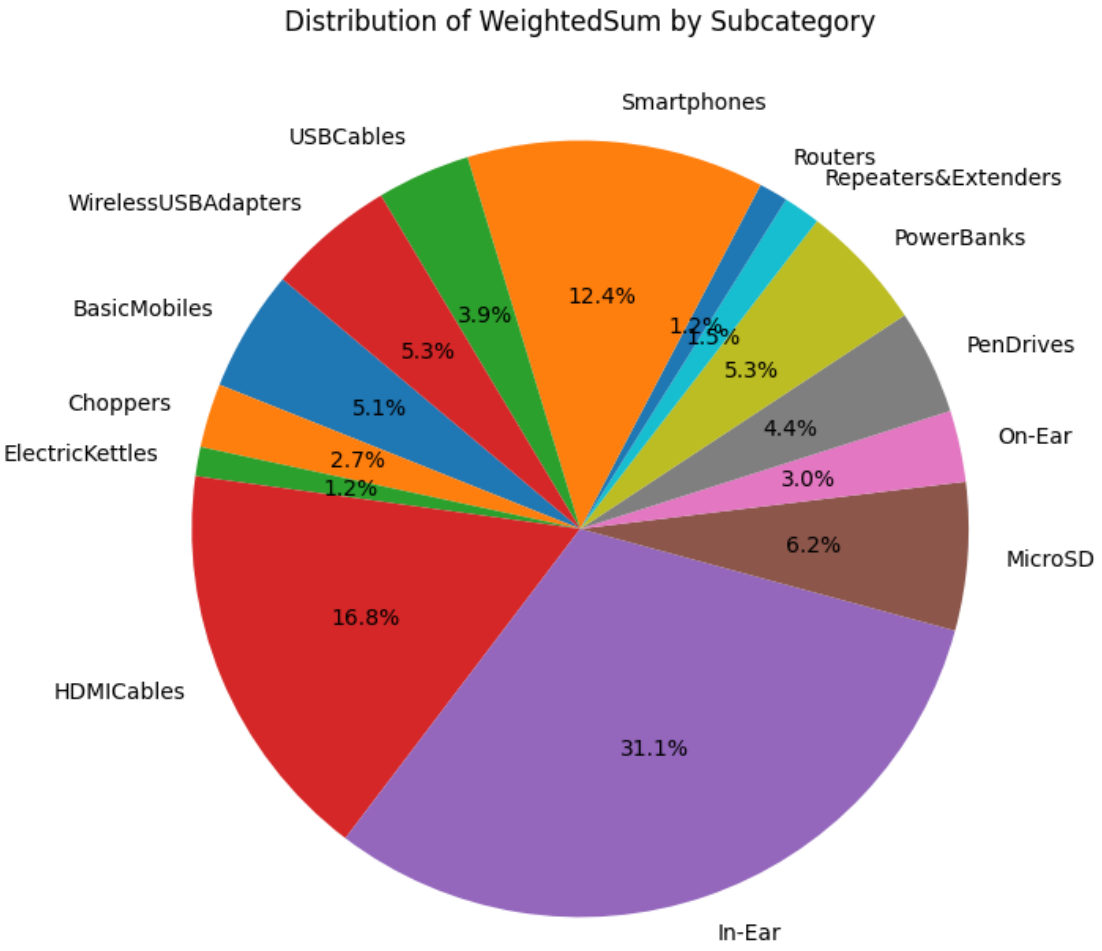
ProductID	Actual Price	Discounted Price	Rating
B07DC4RZPY	₹1,999	₹709	4.1
B00NH11KIK	₹695	₹209	4.5

Top Products in WirelessUSBAdapters

ProductID	Actual Price	Discounted Price	Rating
B0088TKTY2	₹1,399	₹649	4.2
B008IFXQFU	₹999	₹499	4.2
B002SZEOLG	₹1,339	₹749	4.2

[33]:
```
# Assuming df is your DataFrame
plt.figure(figsize=(8, 8))

# Group by subcategory and calculate the sum of WeightedSum
subcategory_sum = top_50_df.groupby('subcategory_1')['WeightedSum'].sum()

# Plot the pie chart
plt.pie(subcategory_sum, labels=subcategory_sum.index, autopct='%1.1f%%',␣
↪startangle=140)

# Set the title
plt.title("Distribution of WeightedSum by Subcategory")
```

```
# Show the plot
plt.show()
```



[34]:
```
# Group by 'Subcategory' and select the top 1 product based on the weighted␣
 ↪average
top_products = top_50_df.sort_values(by='WeightedSum', ascending=False).
 ↪groupby('subcategory_1').head(1)

# Plotting using Matplotlib with styling
fig, ax = plt.subplots(figsize=(12, 5))
ax.axis('off')  # Turn off axis for better appearance

# Create a table and add it to the plot with custom colors and styling
table_data = []
```

```
for _, row in top_products.iterrows():
    table_data.append([row['subcategory_1'], row['product_id'],␣
 ↪row['actual_price'], row['discounted_price'], row['rating']])

table = ax.table(cellText=table_data,
                 colLabels=['Subcategory', 'ProductID', 'Actual Price',␣
 ↪'Discounted Price', 'Rating'],
                 loc='center',
                 colColours=['#dc91eb', '#459fed', '#f24e7e', '#eb8154',␣
 ↪'#6ef580'],  # Use a gradient of coral for column headers
                 cellColours=[['#cbaee6', '#97bdf0', '#f296b1', '#f2a483',␣
 ↪'#a0ebbc']]*len(top_products),  # Use the same gradient for cells
                 cellLoc='center',
                 bbox=[0, 0, 1, 1])

# Styling adjustments
table.auto_set_font_size(True)
table.set_fontsize(10)

plt.title('Top Product from Each Subcategory', fontsize=14, color='#40466e')
plt.show()
```

Top Product from Each Subcategory

Subcategory	ProductID	Actual Price	Discounted Price	Rating
HDMICables	B014I8SX4Y	₹1,400	₹309	4.4
In-Ear	B07GPXXNNG	₹999	₹349	4.1
Smartphones	B09GFLXVH9	₹8,499	₹6,499	4.1
Choppers	B01LWYDEQ7	₹495	₹199	4.1
PenDrives	B005FYNT3G	₹650	₹289	4.3
MicroSD	B09X7DY7Q4	₹1,800	₹939	4.5
WirelessUSBAdapters	B0088TKTY2	₹1,399	₹649	4.2
PowerBanks	B08HVL8QN3	₹2,199	₹1,149	4.3
USBCables	B07DC4RZPY	₹1,999	₹709	4.1
Repeaters&Extenders	B00A0VCJPI	₹2,499	₹1,469	4.2
BasicMobiles	B09YDFDVNS	₹1,699	₹1,324	4.0
ElectricKettles	B07WMS7TWB	₹1,245	₹649	3.9
Routers	B01HGCLUH6	₹1,699	₹1,149	4.2
On-Ear	B07PR1CL3S	₹3,990	₹1,220	4.1

[ ]: