Amazon Sales

November 25, 2024

0.1 Introduction

This dataset consists more than 1000 of real products with their identification number listed in the Amazon marketplace specifically from the region India. I noticed the region due to the currency used in the dataset is Rupee India. My objective is to clean and prepare the data due to the raw data being very unorganized. I will then move on to finding insights about the data and try to elaborate in the form of visualization.

[2]:
```
#import packeges
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
pwd
```

[2]:
```
'C:\\Users\\Hesham\\Amazon Sales'
```

[3]:
```
#import Dataset
Amazon = pd.read_csv('C:\\Users\\Hesham\\Amazon Sales\\amazon.csv')
```

0.2 Data cleaning & Preparation

Before making analyzing the data, it is important to clean and prepare data. The methods used to clean and prepare the data are as listed below:

- Changing Data Types of Columns from object to Floats
- Filling in Missing Information
- Checking For Duplicate Rows
- Splitting Long Strings
- Creating Various New Columns

[4]:
```
#checking first 5 rows
Amazon.head()
```

[4]:
```
   product_id                                       product_name  \
0  B07JW9H4J1  Wayona Nylon Braided USB to Lightning Fast Cha…
1  B098NS6PVG  Ambrane Unbreakable 60W / 3A Fast Charging 1.5…
2  B096MSW6CT  Sounce Fast Phone Charging Cable & Data Sync U…
3  B08HDJ86NZ  boAt Deuce USB 300 2 in 1 Type-C & Micro USB S…
```

```
4  B08CF3B7N1  Portronics Konnect L 1.2M Fast Charging 3A 8 P…

                                                category discounted_price  \
0  Computers&Accessories|Accessories&Peripherals|…              399
1  Computers&Accessories|Accessories&Peripherals|…              199
2  Computers&Accessories|Accessories&Peripherals|…              199
3  Computers&Accessories|Accessories&Peripherals|…              329
4  Computers&Accessories|Accessories&Peripherals|…              154

  actual_price discount_percentage rating rating_count  \
0        1,099                 64%    4.2       24,269
1          349                 43%    4.0       43,994
2        1,899                 90%    3.9        7,928
3          699                 53%    4.2       94,363
4          399                 61%    4.2       16,905

                                        about_product  \
0  High Compatibility : Compatible With iPhone 12…
1  Compatible with all Type C enabled devices, be…
2   Fast Charger& Data Sync -With built-in safet…
3  The boAt Deuce USB 300 2 in 1 cable is compati…
4  [CHARGE & SYNC FUNCTION]- This cable comes wit…

                                          user_id  \
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…
```

Note that the currency being used in **Indian Rupee.**

[5]:
```
Amazon.columns
```

[5]:
```
Index(['product_id', 'product_name', 'category', 'discounted_price',
       'actual_price', 'discount_percentage', 'rating', 'rating_count',
       'about_product', 'user_id', 'user_name', 'review_id', 'review_title',
       'review_content', 'img_link', 'product_link'],
      dtype='object')
```

Features

- product_id - Product ID
- product_name - Name of the Product
- category - Category of the Product
- discounted_price - Discounted Price of the Product
- actual_price - Actual Price of the Product
- discount_percentage - Percentage of Discount for the Product
- rating - Rating of the Product
- rating_count - Number of people who voted for the Amazon rating
- about_product - Description about the Product
- user_id - ID of the user who wrote review for the Product
- user_name - Name of the user who wrote review for the Product

- review_id - ID of the user review
- review_title - Short review
- review_content - Long review
- img_link - Image Link of the Product
- product_link - Official Website Link of the Producto nk of the Product

[6]:
```
#checking num of rows and columns
Amazon.shape
```

[6]:
```
(1465, 16)
```

[7]:
```
#checking Data types
Amazon.dtypes
```

[7]:
```
product_id              object
product_name            object
category                object
discounted_price        object
actual_price            object
discount_percentage     object
rating                  object
rating_count            object
about_product           object
user_id                 object
user_name               object
review_id               object
review_title            object
review_content          object
img_link                object
product_link            object
dtype: object
```

[8]:
```
#Changing the data type of discounted price and actual price
Amazon['discounted_price'] = Amazon['discounted_price'].str.replace(" ",'')
Amazon['discounted_price'] = Amazon['discounted_price'].str.replace(",",'')
Amazon['discounted_price'] = Amazon['discounted_price'].astype('float64')

Amazon['actual_price'] = Amazon['actual_price'].str.replace(" ",'')
Amazon['actual_price'] = Amazon['actual_price'].str.replace(",",'')
Amazon['actual_price'] = Amazon['actual_price'].astype('float64')
```

[9]:
```
#Changing Datatype and values in Discount Percentage
Amazon['discount_percentage'] = Amazon['discount_percentage'].str.
 ↪replace("%",'').astype('float64')
Amazon['discount_percentage'] = Amazon['discount_percentage'] / 100

Amazon['discount_percentage']
```

[9]:
```
0       0.64
1       0.43
2       0.90
3       0.53
4       0.61
        …
1460    0.59
1461    0.25
1462    0.28
1463    0.26
1464    0.22
Name: discount_percentage, Length: 1465, dtype: float64
```

[10]:
```
#Finding unusual string in the rating column
Amazon['rating'].value_counts()
```

[10]:
```
rating
4.1    244
4.3    230
4.2    228
4.0    129
3.9    123
4.4    123
3.8     86
4.5     75
4       52
3.7     42
3.6     35
3.5     26
4.6     17
3.3     16
3.4     10
4.7      6
3.1      4
5.0      3
3.0      3
4.8      3
3.2      2
2.8      2
2.3      1
|        1
2        1
3        1
2.6      1
2.9      1
Name: count, dtype: int64
```

[11]:
```
#Inspecting the strange row
Amazon.query('rating == "|"')
```

[11]:

```
      product_id                                          product_name  \
1279  B08L12N5H1  Eureka Forbes car Vac 100 Watts Powerful Sucti…

                                                 category  discounted_price  \
1279  Home&Kitchen|Kitchen&HomeAppliances|Vacuum,Cle…            2099.0

      actual_price  discount_percentage rating rating_count  \
1279        2499.0                 0.16      |          992

                                          about_product  \
1279  No Installation is provided for this product|1…

                                                user_id  \
1279  AGTDSNT2FKVYEPDPXAA673AIS44A,AER2XFSWNN4LAUCJ5…

                                               user_name  \
1279  Divya,Dr Nefario,Deekshith,Preeti,Prasanth R,P…

                                               review_id  \
1279  R2KKTKM4M9RDVJ,R1O692MZOBTE79,R2WRSEWL56SOS4,R…

                                            review_title  \
1279  Decent product,doesn't pick up sand,Ok ok,Must…

                                          review_content  \
1279  Does the job well,doesn't work on sand. though…

                                                img_link  \
1279  https://m.media-amazon.com/images/W/WEBP_40237…

                                            product_link
1279  https://www.amazon.in/Eureka-Forbes-Vacuum-Cle…
```

I went to the amazon page to get the rating and found that the product id of **B08L12N5H1 has a rating of 4.** So I am going to give the item rating a 4.0 as well.

Source: https://www.amazon.in/Eureka-Forbes-Vacuum-Cleaner-Washable/dp/B08L12N5H1

[12]:
```
##Changing Rating Columns Data Type
Amazon['rating'] = Amazon['rating'].str.replace("|", '4.0').astype('float64')
```

[13]:
```
#Changing Rating Column Data Type
Amazon['rating_count'] = Amazon['rating_count'].str.replace(',', '').
 ↪astype('float64')
```

[14]:
```
#checking for duplicate
duplicates = Amazon.duplicated()
Amazon[duplicates]
```

[14]:
```
Empty DataFrame
Columns: [product_id, product_name, category, discounted_price, actual_price,
discount_percentage, rating, rating_count, about_product, user_id, user_name,
review_id, review_title, review_content, img_link, product_link]
Index: []
```

[15]:
```
#checking missing values
Amazon.isna().sum()
```

[15]:
```
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
```

[16]:
```
#Creating a new DataFrame with Selected Column
df = Amazon[['product_id', 'product_name', 'category', 'discounted_price',␣
↪'actual_price', 'discount_percentage',
            'rating', 'rating_count']].copy()
```

[17]:
```
#Splitting the Strings in the category column
catsplit = Amazon['category'].str.split('|', expand=True)
catsplit
```

[17]:
```
                         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

                              2                       3          4     5  \
0             Cables&Accessories                  Cables  USBCables  None
1             Cables&Accessories                  Cables  USBCables  None
2             Cables&Accessories                  Cables  USBCables  None
3             Cables&Accessories                  Cables  USBCables  None
4             Cables&Accessories                  Cables  USBCables  None
…                            …                       …          …    …
1460  WaterPurifiers&Accessories  WaterPurifierAccessories       None  None
1461      SmallKitchenAppliances         Rice&PastaCookers       None  None
1462                 RoomHeaters            HeatConvectors       None  None
1463                        Fans               ExhaustFans       None  None
1464      SmallKitchenAppliances           SandwichMakers       None  None

         6
0     None
1     None
2     None
3     None
4     None
…     …
1460  None
1461  None
1462  None
1463  None
1464  None

[1465 rows x 7 columns]
```

[18]:
```
#Renaming category column
catsplit = catsplit.rename(columns={0:'category_1', 1:'category_2', 3:
 ↪'category_3'})
```

[19]:
```
#Adding categories to the new dataframe
df['category_1'] = catsplit['category_1']
df['category_2'] = catsplit['category_2']

df.drop(columns = 'category', inplace = True)

df
```

[19]:
```
      product_id                                       product_name  \
0     B07JW9H4J1  Wayona Nylon Braided USB to Lightning Fast Cha…
1     B098NS6PVG  Ambrane Unbreakable 60W / 3A Fast Charging 1.5…
2     B096MSW6CT  Sounce Fast Phone Charging Cable & Data Sync U…
3     B08HDJ86NZ  boAt Deuce USB 300 2 in 1 Type-C & Micro USB S…
4     B08CF3B7N1  Portronics Konnect L 1.2M Fast Charging 3A 8 P…
…            …                                                …
1460  B08L7J3T31  Noir Aqua - 5pcs PP Spun Filter + 1 Spanner | …
1461  B01M6453MB  Prestige Delight PRWO Electric Rice Cooker (1 …
1462  B009P2LIL4  Bajaj Majesty RX10 2000 Watts Heat Convector R…
1463  B00J5DYCCA  Havells Ventil Air DSP 230mm Exhaust Fan (Pist…
1464  B01486F4G6  Borosil Jumbo 1000-Watt Grill Sandwich Maker (…

      discounted_price  actual_price  discount_percentage  rating  \
0                399.0        1099.0                 0.64     4.2
1                199.0         349.0                 0.43     4.0
2                199.0        1899.0                 0.90     3.9
3                329.0         699.0                 0.53     4.2
4                154.0         399.0                 0.61     4.2
…                    …             …                    …       …
1460             379.0         919.0                 0.59     4.0
1461            2280.0        3045.0                 0.25     4.1
1462            2219.0        3080.0                 0.28     3.6
1463            1399.0        1890.0                 0.26     4.0
1464            2863.0        3690.0                 0.22     4.3

      rating_count             category_1                  category_2
0          24269.0  Computers&Accessories     Accessories&Peripherals
1          43994.0  Computers&Accessories     Accessories&Peripherals
2           7928.0  Computers&Accessories     Accessories&Peripherals
3          94363.0  Computers&Accessories     Accessories&Peripherals
4          16905.0  Computers&Accessories     Accessories&Peripherals
…              …                      …                          …
1460        1090.0           Home&Kitchen      Kitchen&HomeAppliances
1461        4118.0           Home&Kitchen      Kitchen&HomeAppliances
1462         468.0           Home&Kitchen  Heating,Cooling&AirQuality
1463        8031.0           Home&Kitchen  Heating,Cooling&AirQuality
1464        6987.0           Home&Kitchen      Kitchen&HomeAppliances

[1465 rows x 9 columns]
```

[20]:
```
#checking a category 1 unique values
df['category_1'].value_counts()
```

[20]:
```
category_1
Electronics              526
Computers&Accessories    453
```

```
Home&Kitchen              448
OfficeProducts             31
MusicalInstruments          2
HomeImprovement             2
Toys&Games                  1
Car&Motorbike               1
Health&PersonalCare         1
Name: count, dtype: int64
```

[21]:
```
#Fixing Strings in the Category_1 Column
df['category_1'] = df['category_1'].str.replace('&',' & ')
df['category_1'] = df['category_1'].str.replace('OfficeProducts', 'Office␣
 ↪Products')
df['category_1'] = df['category_1'].str.replace('MusicalInstruments', 'Musical␣
 ↪Instruments')
df['category_1'] = df['category_1'].str.replace('HomeImprovement', 'Home␣
 ↪Improvement')
```

[22]:
```
#Checking category_2 unique values
df['category_2'].value_counts()
```

[22]:
```
category_2
Accessories&Peripherals                    381
Kitchen&HomeAppliances                     308
HomeTheater,TV&Video                       162
Mobiles&Accessories                        161
Heating,Cooling&AirQuality                 116
WearableTechnology                          76
Headphones,Earbuds&Accessories              66
NetworkingDevices                           34
OfficePaperProducts                         27
ExternalDevices&DataStorage                 18
Cameras&Photography                         16
HomeStorage&Organization                    16
HomeAudio                                   16
GeneralPurposeBatteries&BatteryChargers     14
Accessories                                 14
Printers,Inks&Accessories                   11
CraftMaterials                               7
Components                                   5
OfficeElectronics                            4
Electrical                                   2
Monitors                                     2
Microphones                                  2
Arts&Crafts                                  1
PowerAccessories                             1
Tablets                                      1
```

```
Laptops                                    1
Kitchen&Dining                             1
CarAccessories                             1
HomeMedicalSupplies&Equipment              1
Name: count, dtype: int64
```

[23]:
```
##Fixing Strings in Category_2 column
df['category_2'] = df['category_2'].str.replace('&', ' & ')
df['category_2'] = df['category_2'].str.replace(',', ', ')

df['category_2'] = df['category_2'].str.replace('HomeAppliances', 'Home␣
 ↪Appliances')
df['category_2'] = df['category_2'].str.replace('AirQuality', 'Air Quality')
df['category_2'] = df['category_2'].str.replace('WearableTechnology', 'Wearable␣
 ↪Technology')
df['category_2'] = df['category_2'].str.replace('NetworkingDevices',␣
 ↪'Networking Devices')
df['category_2'] = df['category_2'].str.replace('OfficePaperProducts', 'Office␣
 ↪Paper Products')

df['category_2'] = df['category_2'].str.replace('ExternalDevices', 'External␣
 ↪Devices')
df['category_2'] = df['category_2'].str.replace('DataStorage', 'Data Storage')
df['category_2'] = df['category_2'].str.replace('HomeStorage', 'Home Storage')
df['category_2'] = df['category_2'].str.replace('HomeAudio', 'Home Audio')

df['category_2'] = df['category_2'].str.replace('GeneralPurposeBatteries',␣
 ↪'General Purpose Batteries')
df['category_2'] = df['category_2'].str.replace('BatteryChargers', 'Battery␣
 ↪Chargers')
df['category_2'] = df['category_2'].str.replace('CraftMaterials', 'Craft␣
 ↪Materials')
df['category_2'] = df['category_2'].str.replace('OfficeElectronics', 'Office␣
 ↪Electronics')

df['category_2'] = df['category_2'].str.replace('PowerAccessories', 'Power␣
 ↪Accessories')
df['category_2'] = df['category_2'].str.replace('CarAccessories', 'Car␣
 ↪Accessories')
df['category_2'] = df['category_2'].str.replace('HomeMedicalSupplies', 'Home␣
 ↪Medical Supplies')
df['category_2'] = df['category_2'].str.replace('HomeTheater', 'Home Theater')
```

[24]:
```
# Removing Whitespace from product_id
df['product_id'].str.strip()
```

[24]:
```
0       B07JW9H4J1
1       B098NS6PVG
2       B096MSW6CT
3       B08HDJ86NZ
4       B08CF3B7N1
           ...
1460    B08L7J3T31
1461    B01M6453MB
1462    B009P2LIL4
1463    B00J5DYCCA
1464    B01486F4G6
Name: product_id, Length: 1465, dtype: object
```

[25]:
```
#creating categories for rankings

rating_score = []
for score in df['rating']:
    if score < 2.0: rating_score.append('Poor')
    elif score < 3.0: rating_score.append('Below Average')
    elif score < 4.0: rating_score.append('Average')
    elif score < 5.0: rating_score.append('Above Average')
    elif score == 5.0: rating_score.append('Excellent')
```

Created a Rating Category that consists of:

1. Score below 2.0 = Poor
2. Score range of 2.0 - 2.9 = Below Average
3. Score range of 3.0 - 3.9 = Average
4. Score Range of 4.0 - 4.9 = Above Average
5. Score of 5.0 = Excellent

[26]:
```
#creating a new column and changing the data type

df['rating_score'] = rating_score

df['rating_score'] = df['rating_score'].astype('category')
```

[27]:
```
#Reordered Categories
df['rating_score'] = df['rating_score'].cat.reorder_categories(['Below␣
 ↪Average', 'Average', 'Above Average', 'Excellent'],
                                                               ordered=True)
```

[28]:
```
#Creating a Difference of Price Column between the Actual Price and Discounted␣
 ↪Price

df['difference_price'] = df['actual_price'] - df['discounted_price']
```

[29]:
```
#Result After Cleaning and Preperation after first cleaned dataframe
df.head()
```

[29]:

```
   product_id                                          product_name  \
0  B07JW9H4J1  Wayona Nylon Braided USB to Lightning Fast Cha…
1  B098NS6PVG  Ambrane Unbreakable 60W / 3A Fast Charging 1.5…
2  B096MSW6CT  Sounce Fast Phone Charging Cable & Data Sync U…
3  B08HDJ86NZ  boAt Deuce USB 300 2 in 1 Type-C & Micro USB S…
4  B08CF3B7N1  Portronics Konnect L 1.2M Fast Charging 3A 8 P…

   discounted_price  actual_price  discount_percentage  rating  rating_count  \
0             399.0        1099.0                 0.64     4.2       24269.0
1             199.0         349.0                 0.43     4.0       43994.0
2             199.0        1899.0                 0.90     3.9        7928.0
3             329.0         699.0                 0.53     4.2       94363.0
4             154.0         399.0                 0.61     4.2       16905.0

                category_1                 category_2   rating_score  \
0  Computers & Accessories  Accessories & Peripherals  Above Average
1  Computers & Accessories  Accessories & Peripherals  Above Average
2  Computers & Accessories  Accessories & Peripherals        Average
3  Computers & Accessories  Accessories & Peripherals  Above Average
4  Computers & Accessories  Accessories & Peripherals  Above Average

   difference_price
0             700.0
1             150.0
2            1700.0
3             370.0
4             245.0
```

[30]:
```
#Subsetting Reviewers Identifications
reviewers = Amazon[['user_id','user_name']]
reviewers
```

[30]:

```
                                                user_id  \
0     AG3D6O4STAQKAY2UVGEUV46KN35Q,AHMY5CWJMMK5BJRBB…
1     AECPFYFQVRUWC3KGNLJIOREFP5LQ,AGYYVPDD7YG7FYNBX…
2     AGU3BBQ2V2DDAMOAKGFAWDDQ6QHA,AESFLDV2PT363T2AQ…
3     AEWAZDZZJLQUYVOVGBEUKSLXHQ5A,AG5HTSFRRE6NL3M5S…
4     AE3Q6KSUK5P75D5HFYHCRAOLODSA,AFUGIFH5ZAFXRDSZH…
…                                                   …
1460  AHITFY6AHALOFOHOZEOC6XBP4FEA,AFRABBODZJZQB6Z4U…
1461  AFG5FM3NEMOL6BNFRV2NK5FNJCHQ,AGEINTRN6Z563RMLH…
1462  AGVPWCMAHYQWJOQKMUJN4DW3KM5Q,AF4Q3E66MY4SR7YQZ…
1463  AF2JQCLSCY3QJATWUNNHUSVUPNQQ,AFDMLUXC5LS5RXDJS…
1464  AFGW5PT3R6ZAVQR4Y5MWVAKBZAYA,AG7QNJ2SCS5VS5VYY…
```

```
                                                   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…
…                                                    …
1460   Prabha ds,Raghuram bk,Real Deal,Amazon Custome…
1461   Manu Bhai,Naveenpittu,Evatira Sangma,JAGANNADH…
1462   Nehal Desai,Danish Parwez,Amazon Customer,Amaz…
1463   Shubham Dubey,E.GURUBARAN,Mayank S.,eusuf khan…
1464   Rajib,Ajay B,Vikas Kahol,PARDEEP,Anindya Prama…

[1465 rows x 2 columns]
```

[31]:
```
#Splitting the strings in the user_id column
reviewer_id_split = reviewers['user_id'].str.split(',', expand=False)
reviewer_id_split
```

[31]:
```
0        [AG3D6O4STAQKAY2UVGEUV46KN35Q, AHMY5CWJMMK5BJR…
1        [AECPFYFQVRUWC3KGNLJIOREFP5LQ, AGYYVPDD7YG7FYN…
2        [AGU3BBQ2V2DDAMOAKGFAWDDQ6QHA, AESFLDV2PT363T2…
3        [AEWAZDZZJLQUYVOVGBEUKSLXHQ5A, AG5HTSFRRE6NL3M…
4        [AE3Q6KSUK5P75D5HFYHCRAOLODSA, AFUGIFH5ZAFXRDS…
                                ...
1460     [AHITFY6AHALOFOHOZEOC6XBP4FEA, AFRABBODZJZQB6Z…
1461     [AFG5FM3NEMOL6BNFRV2NK5FNJCHQ, AGEINTRN6Z563RM…
1462     [AGVPWCMAHYQWJOQKMUJN4DW3KM5Q, AF4Q3E66MY4SR7Y…
1463     [AF2JQCLSCY3QJATWUNNHUSVUPNQQ, AFDMLUXC5LS5RXD…
1464     [AFGW5PT3R6ZAVQR4Y5MWVAKBZAYA, AG7QNJ2SCS5VS5V…
Name: user_id, Length: 1465, dtype: object
```

[32]:
```
#Making user id display 1 id per row
reviewer_id_exp = reviewer_id_split.explode() #The explode() method converts␣
 ↪each element of the specified column(s) into a row.
reviewer_id_clean  = reviewer_id_exp.reset_index(drop=True) #method allows you␣
 ↪reset the index back to the default 0, 1, 2 etc indexes.

reviewer_id_clean
```

[32]:
```
0        AG3D6O4STAQKAY2UVGEUV46KN35Q
1        AHMY5CWJMMK5BJRBBSNLYT3ONILA
2        AHCTC6ULH4XB6YHDY6PCH2R772LQ
3        AGYHHIERNXKA6P5T7CZLXKVPT7IQ
4        AG4OGOFWXJZTQ2HKYIOCOY3KXF2Q
                    ...
```

```
11498    AHXCDNSXAESERITAFELQABFVNLCA
11499    AGRZD6CHLCUNOLMMIMIHUCG7PIFA
11500    AFQZVGSOSOJHKFQQMCEI4725QEKQ
11501    AEALVGXXIP46OZVXKRUXSDWZJMEA
11502    AGEFL3AY7YXEFZA4ZJU3LP7K7OJQ
Name: user_id, Length: 11503, dtype: object
```

[33]:
```
#Splitting the strings in the user_name column
reviewer_name_split = reviewers['user_name'].str.split(',', expand=False)

reviewer_name_split
```

[33]:
```
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
```

[34]:
```
#Making user name display 1 id per row

review_name_exp = reviewer_name_split.explode()

reviewer_name_clean = review_name_exp.reset_index(drop=True)

reviewer_name_clean
```

[34]:
```
0                  Manav
1            Adarsh gupta
2                 Sundeep
3           S.Sayeed Ahmed
4           jaspreet singh
                …
11510             PARDEEP
11511    Anindya Pramanik
11512         Vikas Singh
11513     Harshada Pimple
11514              Saw a.
Name: user_name, Length: 11515, dtype: object
```

[35]:
```
#Creating 2 Data Frames to be merged
df21 = pd.DataFrame(data=reviewer_id_clean)

df22 = pd.DataFrame(data=reviewer_name_clean)
```

[36]:
```
#Merging the 2 dataframe containing user_id and user_name
df2 = pd.merge(df21, df22, left_index=True, right_index=True)
df2.head()
```

[36]:

	user_id	user_name
0	AG3D6O4STAQKAY2UVGEUV46KN35Q	Manav
1	AHMY5CWJMMK5BJRBBSNLYT3ONILA	Adarsh gupta
2	AHCTC6ULH4XB6YHDY6PCH2R772LQ	Sundeep
3	AGYHHIERNXKA6P5T7CZLXKVPT7IQ	S.Sayeed Ahmed
4	AG4OGOFWXJZTQ2HKYIOCOY3KXF2Q	jaspreet singh

0.3 Data Exploration

insights through Visualizations

[57]:
```
#Setting Visualization Style

#sns.set_style(style='darkgrid')

#sns.set_palette(palette="icefire")
```

[37]:
```
#Main Category and Sub-Category

main_sub = df[['category_1', 'category_2', 'product_id']]

main_sub = main_sub.rename(columns={'category_1' :'Main Category', 'category_2'␣
↪: 'Sub-Category', 'product_id':'Product ID'})

main_sub_piv = pd.pivot_table(main_sub, index=['Main Category',␣
↪'Sub-Category'], aggfunc='count')

main_sub_piv
```

[37]:

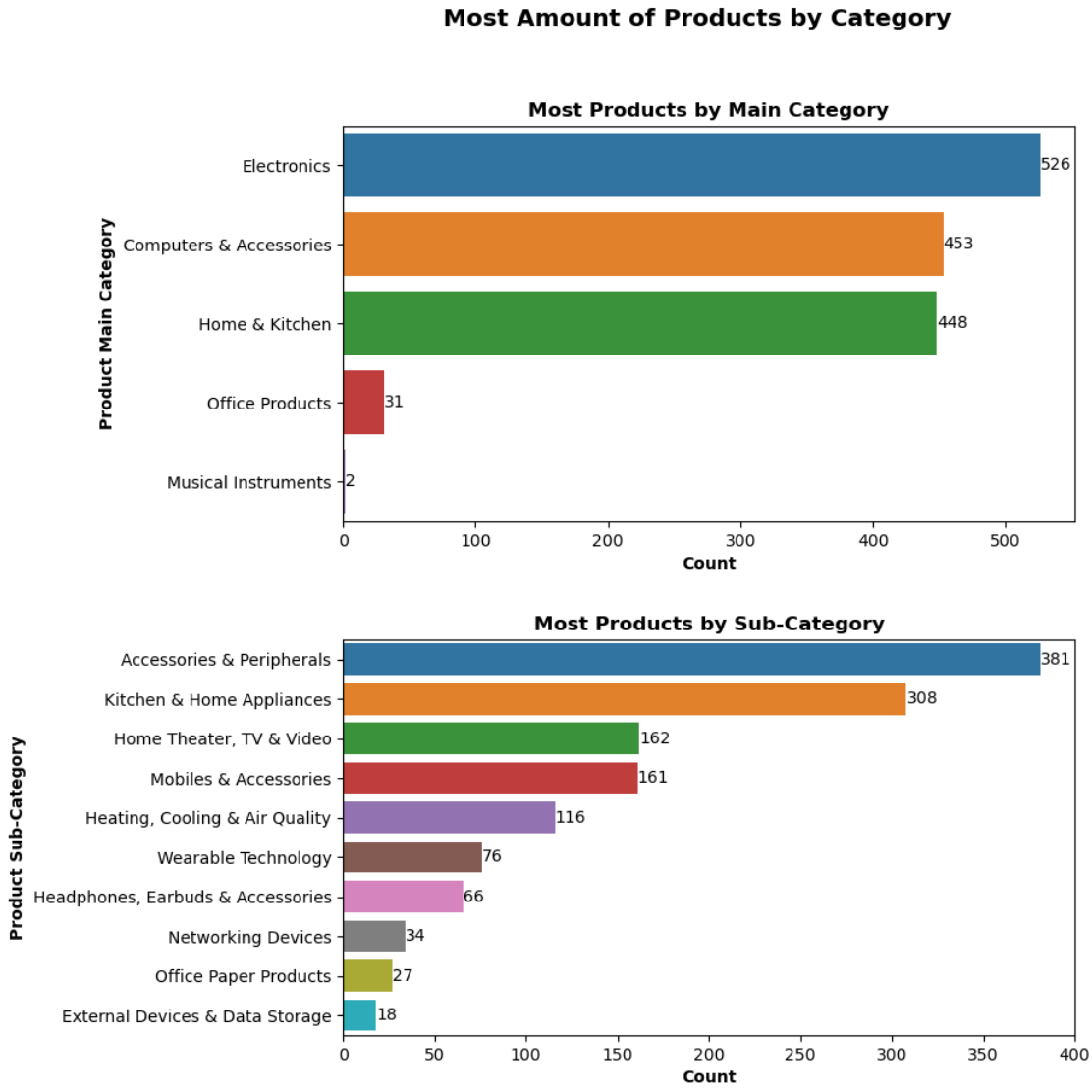
		Product ID
Main Category	Sub-Category	
Car & Motorbike	Car Accessories	1
Computers & Accessories	Accessories & Peripherals	381
	Components	5
	External Devices & Data Storage	18
	Laptops	1
	Monitors	2
	Networking Devices	34
	Printers, Inks & Accessories	11

	Tablets	1
Electronics	Accessories	14
	Cameras & Photography	16
	General Purpose Batteries & Battery Chargers	14
	Headphones, Earbuds & Accessories	66
	Home Audio	16
	Home Theater, TV & Video	162
	Mobiles & Accessories	161
	Power Accessories	1
	Wearable Technology	76
Health & PersonalCare	Home Medical Supplies & Equipment	1
Home & Kitchen	Craft Materials	7
	Heating, Cooling & Air Quality	116
	Home Storage & Organization	16
	Kitchen & Dining	1
	Kitchen & Home Appliances	308
Home Improvement	Electrical	2
Musical Instruments	Microphones	2
Office Products	Office Electronics	4
	Office Paper Products	27
Toys & Games	Arts & Crafts	1

0.3.1 Most amount of products by category

[38]:
```
most_main_items = df['category_1'].value_counts().head(5).
 ↪rename_axis('category_1').reset_index(name='counts')

most_sub_items = df['category_2'].value_counts().head(10).
 ↪rename_axis('category_2').reset_index(name='counts')

fig, ax = plt.subplots(2, 1, figsize=(8, 10))
fig.suptitle('Most Amount of Products by Category', fontweight='heavy',␣
 ↪size='x-large')

sns.barplot(ax=ax[0], data=most_main_items, x='counts', y='category_1')
sns.barplot(ax=ax[1], data=most_sub_items, x='counts', y='category_2')

plt.subplots_adjust(hspace = 0.3)

ax[0].set_xlabel('Count', fontweight='bold')
ax[0].set_ylabel('Product Main Category', fontweight='bold')

ax[1].set_xlabel('Count', fontweight='bold')
ax[1].set_ylabel('Product Sub-Category', fontweight='bold')

ax[0].set_title('Most Products by Main Category', fontweight='bold')
ax[1].set_title('Most Products by Sub-Category', fontweight='bold')
```

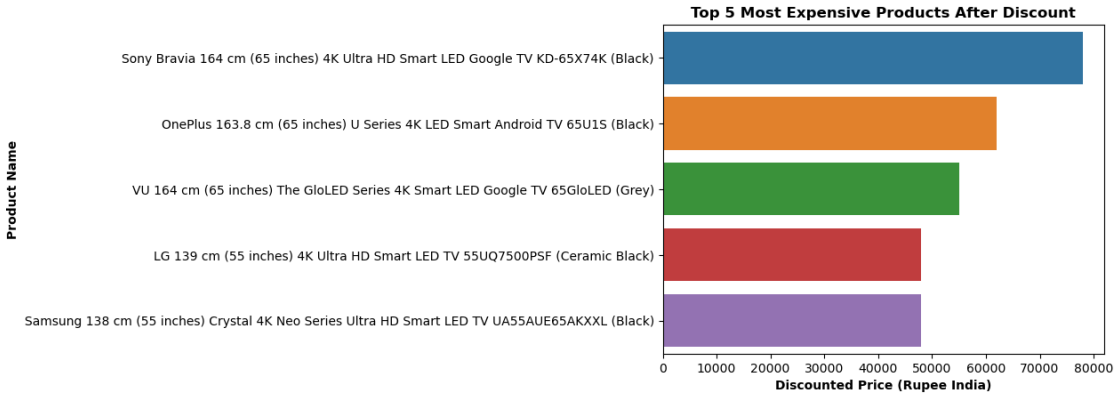
```
ax[0].bar_label(ax[0].containers[0])
ax[1].bar_label(ax[1].containers[0])

plt.show()
```



Electronics especially accessories & peripherals and kitchen & home appliances covers most of the products in this dataset. In general

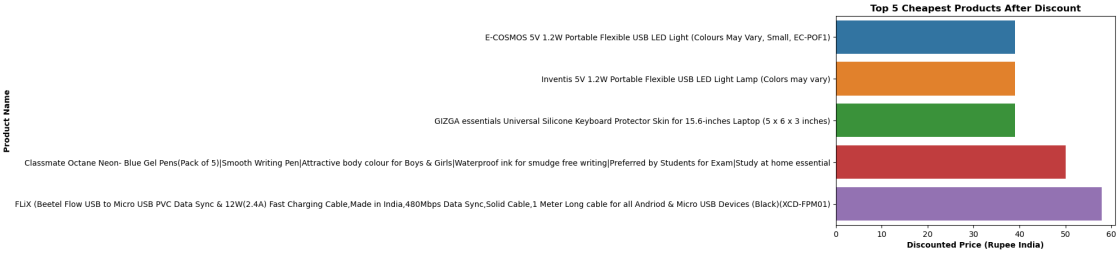
0.3.2 Top 5 Most Expensive Products After Discount

[39]:
```
disc_exp = sns.barplot(data=df.sort_values('discounted_price', ascending=False).
 ↪head(5), x='discounted_price', y='product_name')

disc_exp.set_title('Top 5 Most Expensive Products After Discount',␣
 ↪fontweight='bold')
disc_exp.set_xlabel('Discounted Price (Rupee India)', fontweight='bold')
disc_exp.set_ylabel('Product Name', fontweight='bold')

plt.show()
```



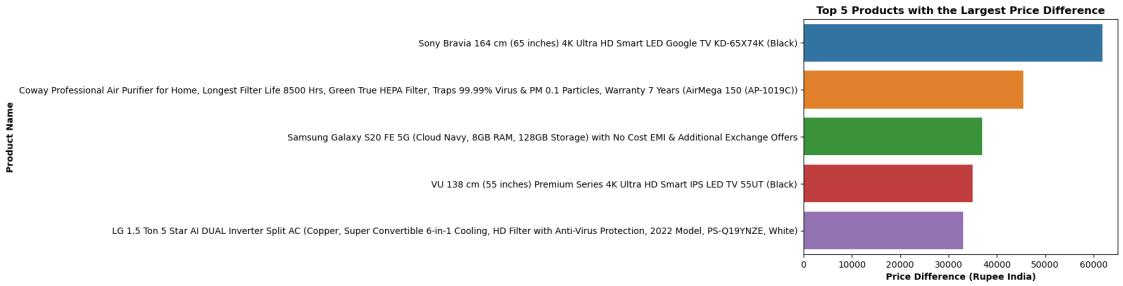
0.3.3 Top 5 Cheapest Products After Discount

[40]:
```
disc_cheap = sns.barplot(data=df.sort_values('discounted_price').head(5),␣
 ↪x='discounted_price', y='product_name')

disc_cheap.set_title('Top 5 Cheapest Products After Discount',␣
 ↪fontweight='bold')
disc_cheap.set_xlabel('Discounted Price (Rupee India)', fontweight='bold')
disc_cheap.set_ylabel('Product Name', fontweight='bold')

plt.show()
```



0.3.4 Top 5 Products with the largest difference in price due to discount

[41]:
```
dif_price_large = sns.barplot(data= df.sort_values('difference_price',␣
 ↪ascending=False).head(5), x='difference_price', y='product_name')

dif_price_large.set_title('Top 5 Products with the Largest Price Difference',␣
 ↪fontweight='bold')
dif_price_large.set_xlabel('Price Difference (Rupee India)', fontweight='bold')
dif_price_large.set_ylabel('Product Name', fontweight='bold')

plt.show()
```



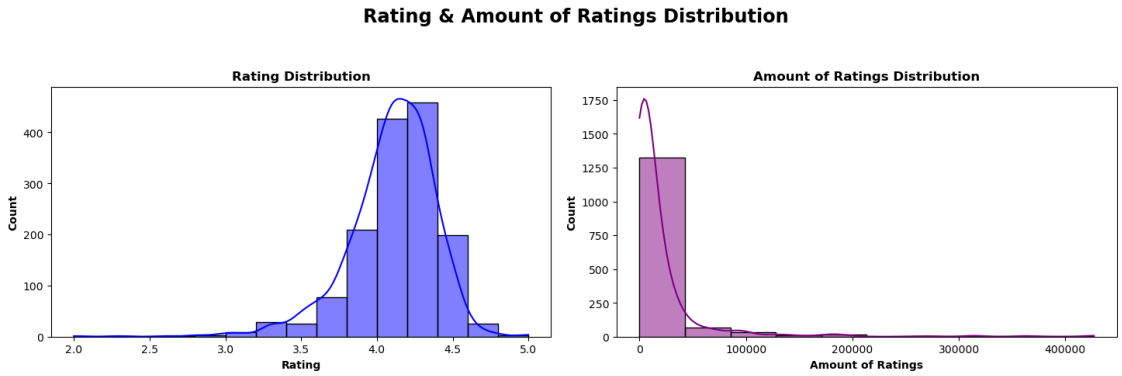
Product Ratings

0.3.5 Rating & Amount of Rating Distribution

[42]:
```
fig, ax = plt.subplots(1, 2, figsize=(15, 5))

fig.suptitle('Rating & Amount of Ratings Distribution', fontweight='heavy',␣
 ↪size='xx-large')

fig.tight_layout(pad=3.0) #adjust the padding between and around subplots

sns.histplot(ax=ax[0], data=df, x='rating', bins=15, kde=True, color='blue')
sns.histplot(ax=ax[1], data=df, x='rating_count', bins=10, kde=True,␣
 ↪color='purple')

ax[0].set_xlabel('Rating', fontweight='bold')
ax[1].set_xlabel('Amount of Ratings', fontweight='bold')

ax[0].set_ylabel('Count', fontweight='bold')
ax[1].set_ylabel('Count', fontweight='bold')

ax[0].set_title('Rating Distribution', fontweight='bold')
```

```
ax[1].set_title('Amount of Ratings Distribution', fontweight='bold')

plt.show()
```

```
C:\Users\Hesham\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Hesham\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

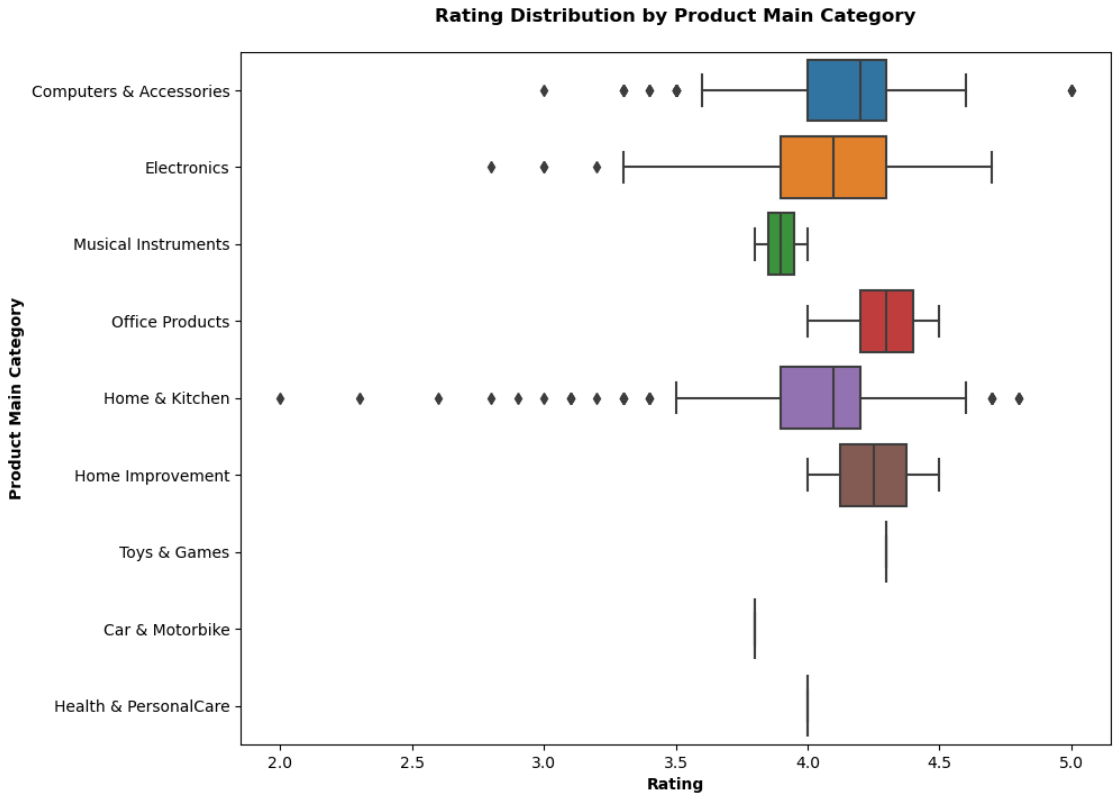


Most of the product rating range around **4.0 - 4.375** with **no products under the score of 2.0.** The rating distribution in slightly left-skewed.

The amount of ratings given to a product is very widespread. Most of the products that have been rated, have around **0 - 5000 amount of rating** for each product. Interestingly there are products that have more than 40,000 ratings. The amount of **ratings distribution is highly right skewed.**

0.3.6 Rating Distribution by Product Main Category

[43]:
```
fig, ax = plt.subplots(figsize = (10, 8))

sns.boxplot(ax = ax, data = df, x = 'rating', y = 'category_1')

ax.set_xlabel('Rating', fontweight='bold')
ax.set_ylabel('Product Main Category', fontweight='bold')
ax.set_title('Rating Distribution by Product Main Category',␣
 ↪fontweight='heavy', size='large', y=1.03)

plt.show()
```



Toys & Games, Car & Motorbike, and Health & Personal Care product ratings's are around **3.75 - 4.375. All Home Improvement, and Office Products** have a **minimal rating of 4.0.**

Many of the **Computer & Accessories, and Electronics** products have ratings in the range of **3.6 - 4.6.** Though these categories do have products that have a high rating such as 5.0 and low rating, going down to 2.75.

Noticeably, the **Home & Kitchen products** have a really widespread rating going to as **high as 4.75** and going as **low as 2.0** rating, which is the lowest rating out of all the products in this dataset. However, most of the products in this category fall in the range of around **3.8 - 4.6.**

[44]:
```
#Rating of Products based on Rating Category

rate_main_cat = df.groupby(['category_1','rating_score']).agg('count').iloc[:
↪,1].rename_axis().reset_index(name='Amount')

rate_main_cat = rate_main_cat.rename(columns = {'category_1' : 'Main Category',␣
↪'rating_score' : 'Rating Category'})

rate_main_cat
```

```
C:\Users\Hesham\AppData\Local\Temp\ipykernel_6440\1469982976.py:3:
```

```
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  rate_main_cat = df.groupby(['category_1','rating_score']).agg('count').iloc[:,
1].rename_axis().reset_index(name='Amount')
```

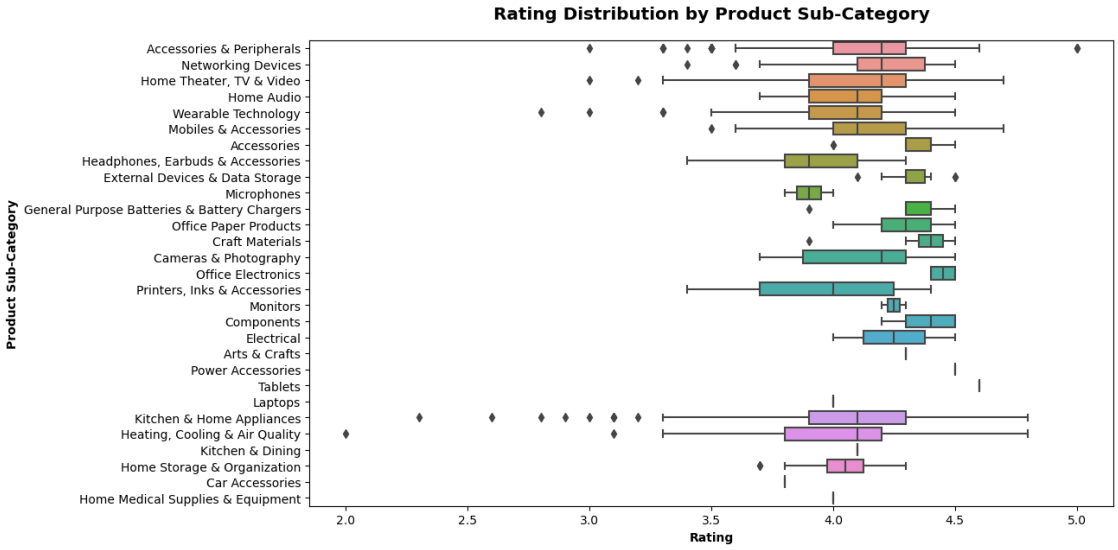
[44]:

	Main Category	Rating Category	Amount
0	Car & Motorbike	Below Average	0
1	Car & Motorbike	Average	1
2	Car & Motorbike	Above Average	0
3	Car & Motorbike	Excellent	0
4	Computers & Accessories	Below Average	0
5	Computers & Accessories	Average	75
6	Computers & Accessories	Above Average	375
7	Computers & Accessories	Excellent	3
8	Electronics	Below Average	1
9	Electronics	Average	132
10	Electronics	Above Average	393
11	Electronics	Excellent	0
12	Health & PersonalCare	Below Average	0
13	Health & PersonalCare	Average	0
14	Health & PersonalCare	Above Average	1
15	Health & PersonalCare	Excellent	0
16	Home & Kitchen	Below Average	5
17	Home & Kitchen	Average	139
18	Home & Kitchen	Above Average	304
19	Home & Kitchen	Excellent	0
20	Home Improvement	Below Average	0
21	Home Improvement	Average	0
22	Home Improvement	Above Average	2
23	Home Improvement	Excellent	0
24	Musical Instruments	Below Average	0
25	Musical Instruments	Average	1
26	Musical Instruments	Above Average	1
27	Musical Instruments	Excellent	0
28	Office Products	Below Average	0
29	Office Products	Average	0
30	Office Products	Above Average	31
31	Office Products	Excellent	0
32	Toys & Games	Below Average	0
33	Toys & Games	Average	0
34	Toys & Games	Above Average	1
35	Toys & Games	Excellent	0

Above is the list of the amount of products under specific ratings for each main category.

0.3.7 Rating Distribution by Product Sub-Category

[45]:
```
fig, ax = plt.subplots(figsize=(12, 7))

sns.boxplot(ax=ax, data=df, x='rating', y='category_2')

ax.set_xlabel('Rating', fontweight='bold')
ax.set_ylabel('Product Sub-Category', fontweight='bold')
ax.set_title('Rating Distribution by Product Sub-Category', fontweight='heavy',␣
 ↪size='x-large', y=1.03)

plt.show()
```



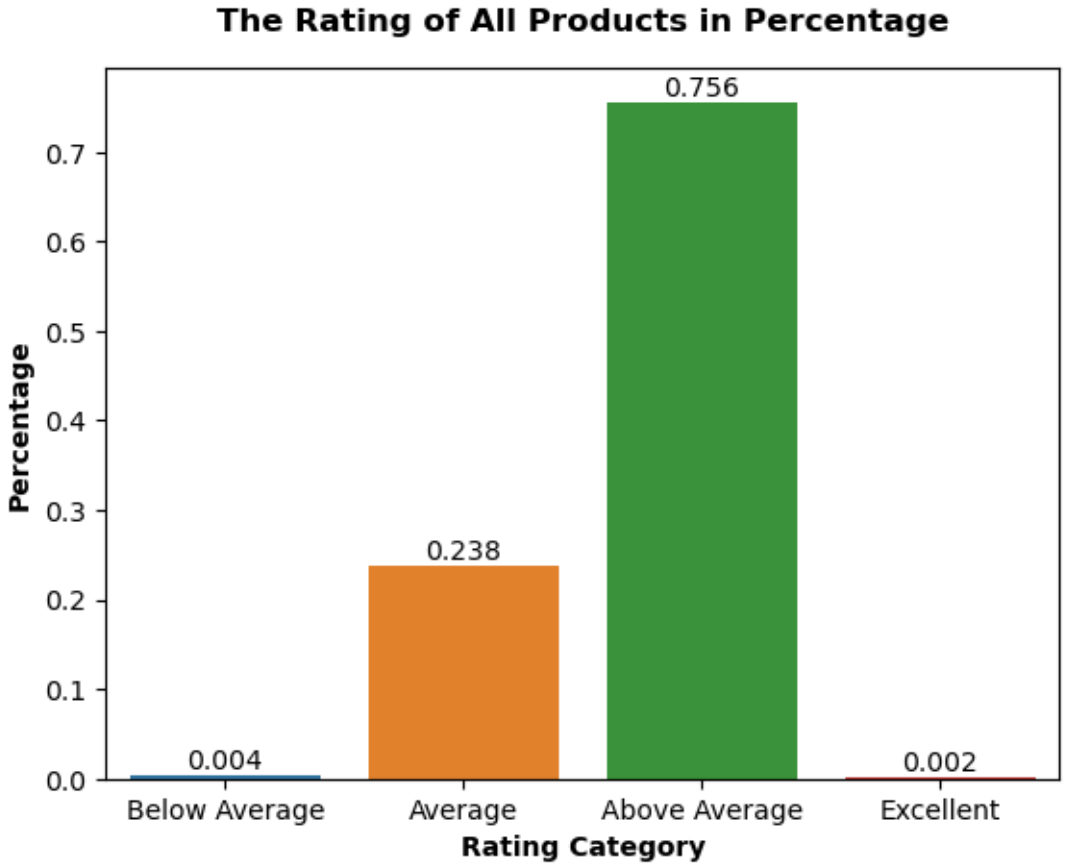
In the Rating Distribution by Product Sub-Category graph, I have noticed that the **highest rated product** comes from the sub-category of **Accessories & Peripherals.** The lowest rated product comes from the subcategory of **Heating, Cooling & Air Quality**

0.3.8 The Rating of All Products in Percentage

[46]:
```
rating_ordered = ['Below Average', 'Average', 'Above Average', 'Excellent']

rating_count = df['rating_score'].value_counts(normalize=True).
 ↪rename_axis('rating').reset_index(name='counts')

rating_count['counts'] = rating_count['counts'].round(3)

rating_count_plot = sns.barplot(data=rating_count, x='rating', y='counts',␣
 ↪order=rating_ordered)
```

```
rating_count_plot.set_xlabel('Rating Category', fontweight='bold')
rating_count_plot.set_ylabel('Percentage', fontweight='bold')
rating_count_plot.set_title('The Rating of All Products in Percentage',␣
 ↪fontweight='heavy', size='large', y=1.03)


rating_count_plot.bar_label(rating_count_plot.containers[0])

plt.show()
```

```
C:\Users\Hesham\anaconda3\Lib\site-packages\seaborn\categorical.py:641:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
```



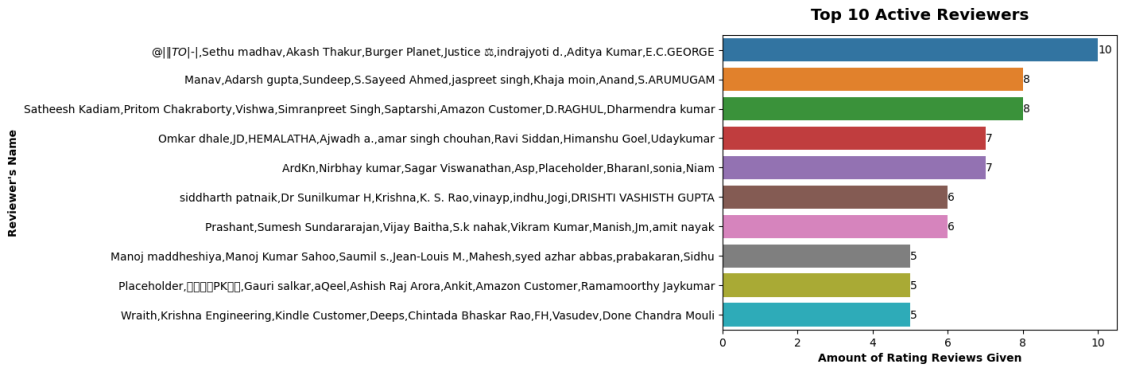
Most of the products in this dataset have been rated **Above Average.** There are **extremely few products** that are rated **Below Average and Excellent. No products are rated as Poor** in this dataset.

0.3.9 Reviewers

0.3.10 Reviewers who gave ratings and reviews for more than one product

[47]:
```
top_reviewer = data=Amazon['user_name'].value_counts().head(10).
 ↪rename_axis('username').reset_index(name='counts')

top_review_plot = sns.barplot(data=top_reviewer, x='counts', y='username')

top_review_plot.bar_label(top_review_plot.containers[0])

top_review_plot.set_xlabel('Amount of Rating Reviews Given', fontweight='bold')
top_review_plot.set_ylabel("Reviewer's Name", fontweight='bold')
top_review_plot.set_title('Top 10 Active Reviewers', fontweight='heavy',␣
 ↪size='x-large', y=1.03)

plt.show()
```

```
C:\Users\Hesham\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152:
UserWarning: Glyph 2358 (\N{DEVANAGARI LETTER SHA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Hesham\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152:
UserWarning: Matplotlib currently does not support Devanagari natively.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Hesham\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152:
UserWarning: Glyph 2381 (\N{DEVANAGARI SIGN VIRAMA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Hesham\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152:
UserWarning: Glyph 2352 (\N{DEVANAGARI LETTER RA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Hesham\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152:
UserWarning: Glyph 2368 (\N{DEVANAGARI VOWEL SIGN II}) missing from current
font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Hesham\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152:
UserWarning: Glyph 2332 (\N{DEVANAGARI LETTER JA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```

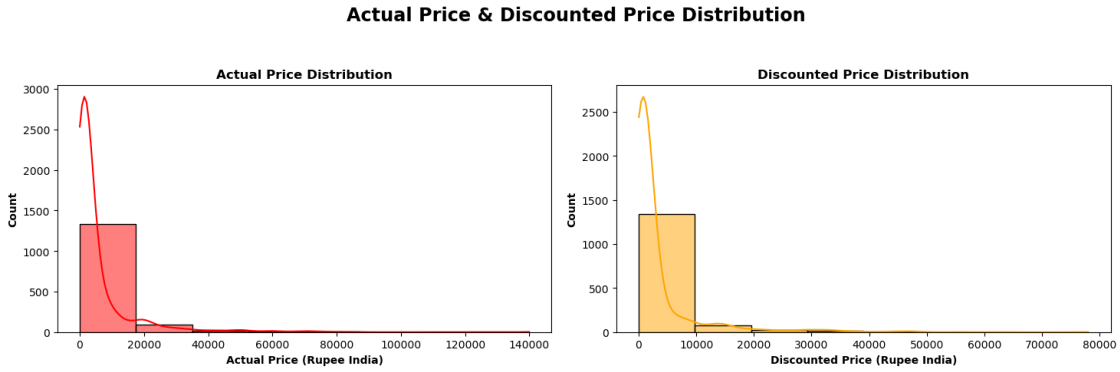


0.3.11 Product Pricing

0.3.12 Actual Price & Discounted Price Distribution

[48]:
```
fig, ax = plt.subplots(1, 2, figsize=(15, 5))

fig.suptitle('Actual Price & Discounted Price Distribution',␣
 ↪fontweight='heavy', size='xx-large')

fig.tight_layout(pad=3.0)

sns.histplot(ax=ax[0], data=df, x='actual_price', bins=8, kde=True, color='red')
sns.histplot(ax=ax[1], data=df, x='discounted_price', bins=8, kde=True,␣
 ↪color='orange')

ax[0].set_xlabel('Actual Price (Rupee India)', fontweight='bold')
ax[1].set_xlabel('Discounted Price (Rupee India)', fontweight='bold')

ax[0].set_ylabel('Count', fontweight='bold')
ax[1].set_ylabel('Count', fontweight='bold')

ax[0].set_title('Actual Price Distribution', fontweight='bold')
ax[1].set_title('Discounted Price Distribution', fontweight='bold')

plt.show()
```

```
C:\Users\Hesham\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Hesham\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

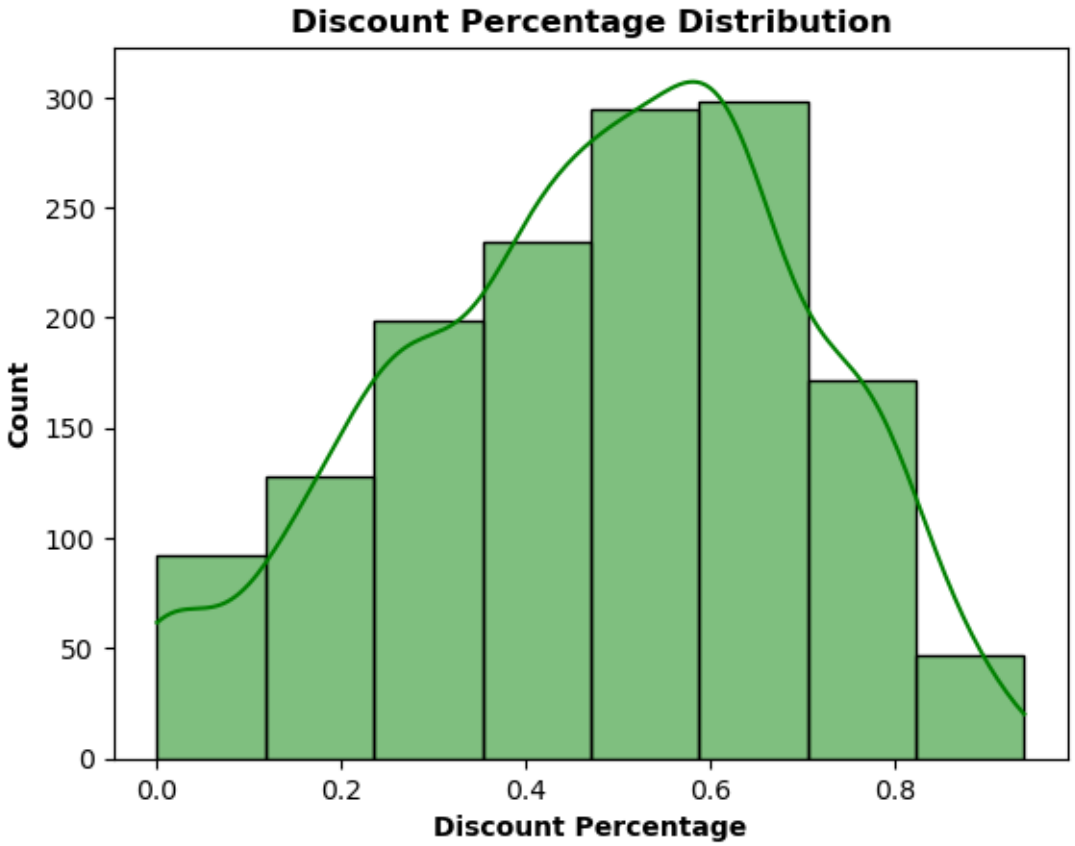


Both graphs show the same distribution which is **Right or Positvely Skewed.**

0.3.13 Discount Percentage Distribution

[49]:
```
disc_hist = sns.histplot(data=df, x='discount_percentage', bins=8, kde=True,␣
 ↪color='green')

disc_hist.set_xlabel('Discount Percentage', fontweight='bold')
disc_hist.set_ylabel('Count', fontweight='bold')
disc_hist.set_title('Discount Percentage Distribution', fontweight='heavy',␣
 ↪size='large')

plt.show()
```

```
C:\Users\Hesham\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



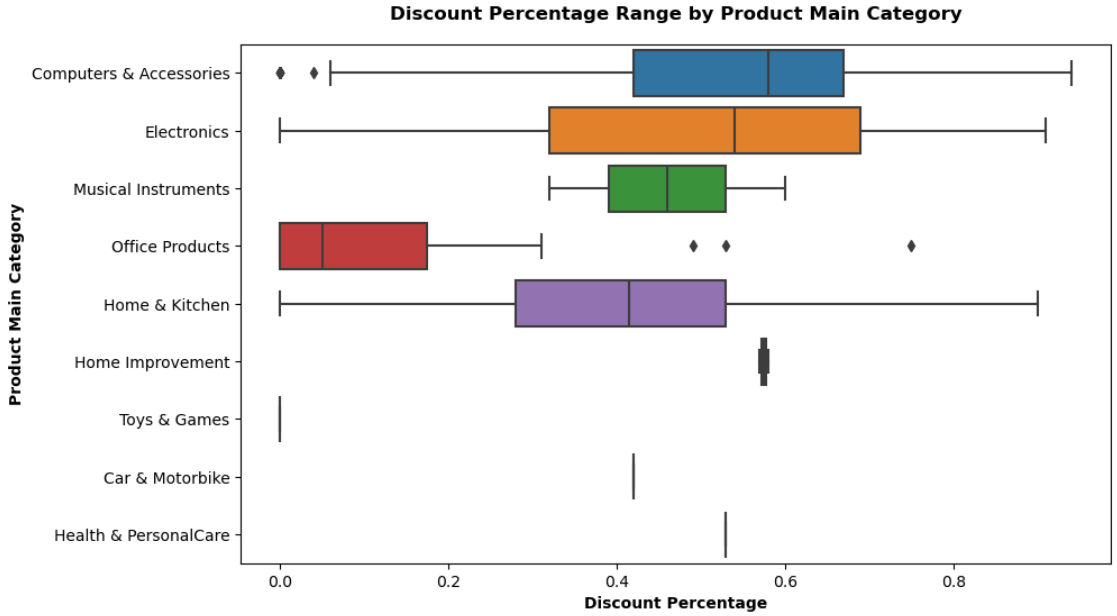
Most products on the dataset have **discounts at around 50% - 70%.**

[50]:
```
#Specific Details about the Discount Percentage
df['discount_percentage'].describe()
```

[50]:
```
count    1465.000000
mean        0.476915
std         0.216359
min         0.000000
25%         0.320000
50%         0.500000
75%         0.630000
max         0.940000
Name: discount_percentage, dtype: float64
```

0.3.14 The Discount Range by Product Main Category

[51]:
```
fig, ax = plt.subplots(figsize = (10, 6))

sns.boxplot(data = df, x = 'discount_percentage', y = 'category_1')

ax.set_xlabel('Discount Percentage', fontweight = 'bold')
ax.set_ylabel('Product Main Category', fontweight = 'bold')
ax.set_title('Discount Percentage Range by Product Main Category', fontweight =␣
 ↪'heavy', size = 'large', y = 1.03)

plt.show()
```



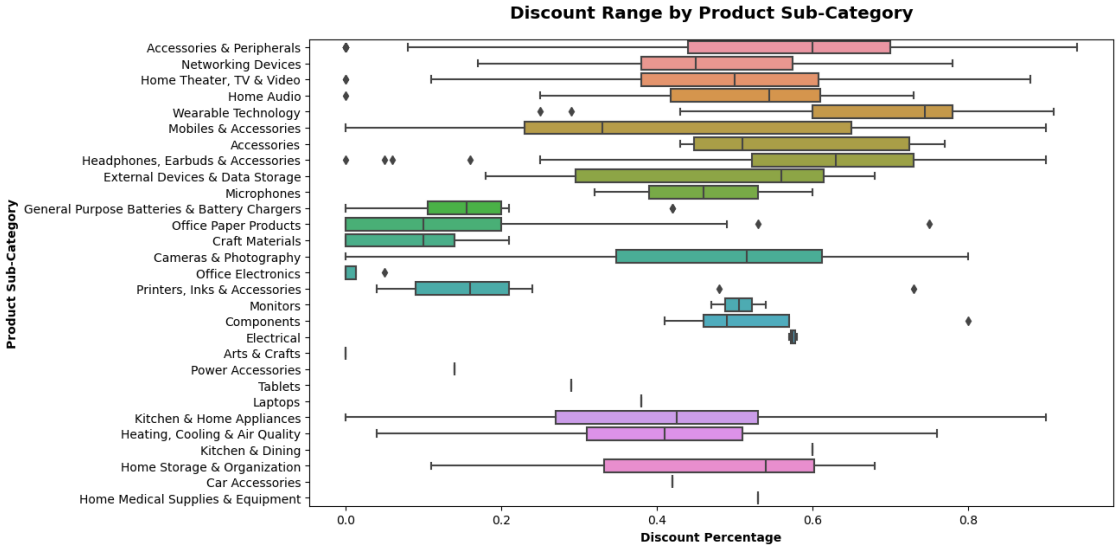
Computers & Accessories, Electronics and Home & Kitchen products have a **large spread of discount variation** ranging a minimal of 0% to more than 90% discount.

Toys & Games, Cars & Motorbikes, Health & Personal Care, and Home Improvement have the **least spread** of discount variation.

Office Products does not give a large amount of discount compared to other products in the Main Category.

0.3.15 The Discount Range by Product Sub-Category

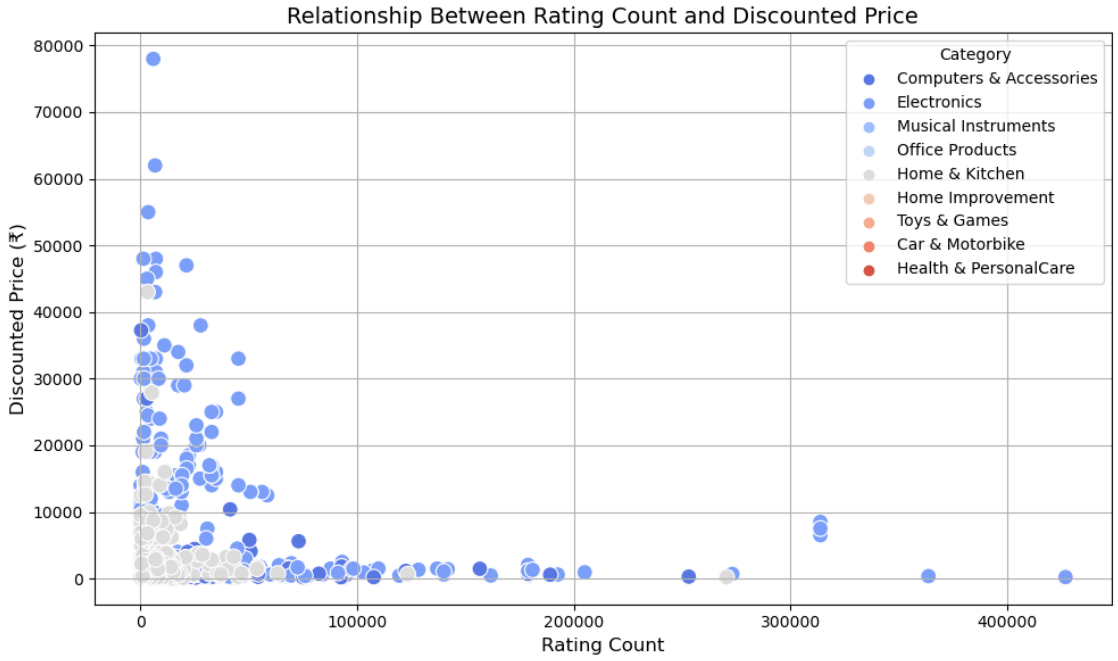
[52]:
```
fig, ax = plt.subplots(figsize=(12, 7))

sns.boxplot(data = df, x = 'discount_percentage', y = 'category_2')

ax.set_xlabel('Discount Percentage', fontweight = 'bold')
ax.set_ylabel('Product Sub-Category', fontweight = 'bold')
ax.set_title('Discount Range by Product Sub-Category', fontweight = 'heavy',␣
 ↪size = 'x-large', y = 1.03)

plt.show()
```



0.3.16 the number of reviews vs discounted prices

[53]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='rating_count', y='discounted_price',␣
 ↪hue='category_1', palette='coolwarm', s=100)
plt.title('Relationship Between Rating Count and Discounted Price', fontsize=14)
plt.xlabel('Rating Count', fontsize=12)
```

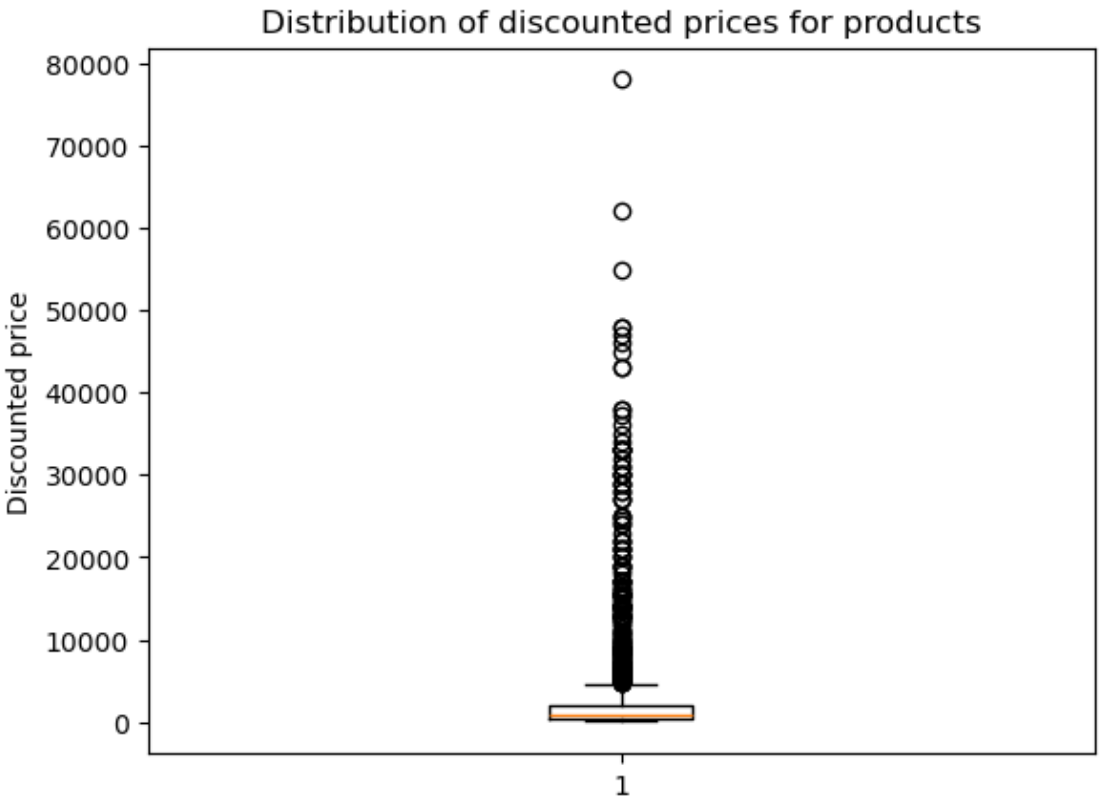
```
plt.ylabel('Discounted Price ( )', fontsize=12)
plt.legend(title='Category', loc='upper right')
plt.grid(True)
plt.tight_layout()
plt.show()
```



0.3.17 distribute the discounted prices

[54]:
```
average_discounted_price = Amazon['discounted_price'].mean()
print("Average discounted price:", average_discounted_price)

plt.boxplot(Amazon['discounted_price'])
plt.title("Distribution of discounted prices for products")
plt.ylabel("Discounted price")
plt.show()
```

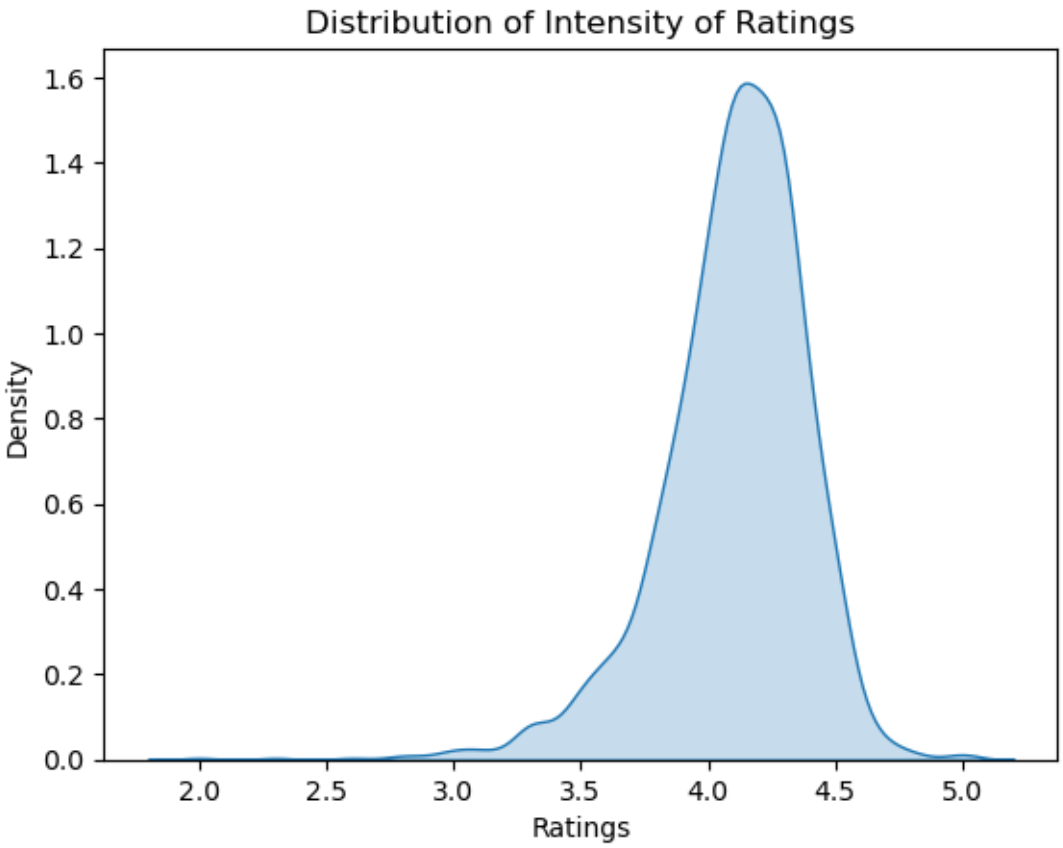
```
Average discounted price: 3125.3108737201364
```



0.3.18 Distribute ratings more widely

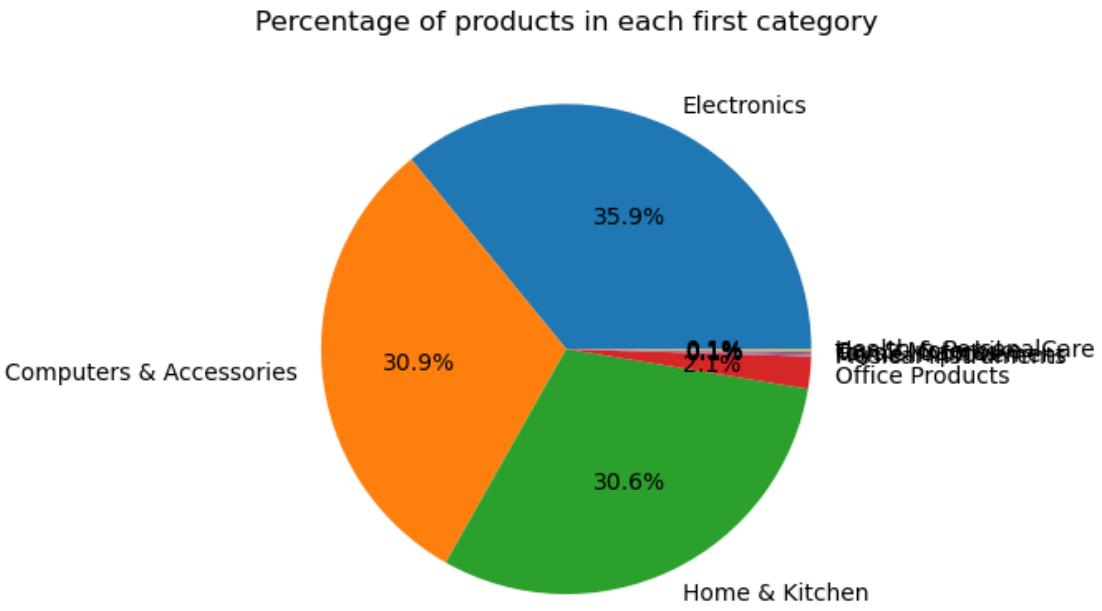
[56]:
```
sns.kdeplot(Amazon['rating'], shade=True)
plt.title('Distribution of Intensity of Ratings')
plt.xlabel('Ratings')
plt.show()
```

```
C:\Users\Hesham\AppData\Local\Temp\ipykernel_6440\3382035131.py:3:
FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(Amazon['rating'], shade=True)
C:\Users\Hesham\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



0.3.19 The percentage of products that belong to each first category

[57]:
```
category_counts = df['category_1'].value_counts()

plt.pie(category_counts, labels=category_counts.index, autopct='%1.1f%%')
plt.title('Percentage of products in each first category')
plt.show()
```


Percentage of products in each first category
Electronics
35.9%
Computers & Accessories
30.9%
30.6%
Home & Kitchen
2.1%
Office Products

[ ]: