In [ ]:

```
#loading the requried libraries
# Imports and configuration
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
print("Libraries imported. Working dir:", os.getcwd())
```

```
Libraries imported. Working dir: /content
```

In [ ]:

```
df = pd.read_csv('/content/amazon.csv')
df
```

Out[ ]:

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

1465 rows × 16 columns

In [ ]:

```
#Checking out First Few Rows

df.head()
```

Out[ ]:

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

In [ ]:

```
#Checking out First Few Rows

df.head()
```

Out[ ]:

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

In [ ]:

```
#Checking Number of Rows and Columns

df.shape
```

Out[ ]:

```
(1465, 16)
```

In [ ]:

```
#Checking Data Types for each Column

df.dtypes
```

```
Out[ ]:                           0
                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
```

In [ ]:

```
#Changing the data type of discounted price and actual price

if df['discounted_price'].dtype == 'object':
  df['discounted_price'] = df['discounted_price'].str.replace("₹",'')
  df['discounted_price'] = df['discounted_price'].str.replace(",",'')
  df['discounted_price'] = df['discounted_price'].astype('float64')

if df['actual_price'].dtype == 'object':
  df['actual_price'] = df['actual_price'].str.replace("₹",'')
  df['actual_price'] = df['actual_price'].str.replace(",",'')
  df['actual_price'] = df['actual_price'].astype('float64')
df["discounted_price"]
```

Out[ ]:

	discounted_price
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

1465 rows × 1 columns

dtype: float64

In [ ]:

```
df['discount_percentage'] = df['discount_percentage'].str.replace('%','').asty

df['discount_percentage'] = df['discount_percentage'] / 100

df['discount_percentage']
```

Out[ ]:

	discount_percentage
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

1465 rows × 1 columns

dtype: float64

In [ ]:

```
df['rating'].value_counts()
```

```
Out[ ]:          count
        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
           3.0       3
           4.8       3
           5.0       3
           2.8       2
           3.2       2
           2.3       1
             |       1
             2       1
             3       1
           2.6       1
           2.9       1

        dtype: int64
```

In [ ]:

```
duplicates = df.duplicated()
df[duplicates]
```

Out[ ]:

product_id	product_name	category	discounted_price	actual_price	discount_

In [ ]:

```
df.isna().sum()
```

Out[ ]:

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

dtype: int64

In [ ]:

```
df1 = df[['product_id', 'product_name', 'category', 'discounted_price', 'actua
```

In [ ]:

```
#Splitting the Strings in the category column

catsplit = df['category'].str.split('|', expand=True)
catsplit
```

Out[ ]:

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

1465 rows × 7 columns

In [ ]:

```
#Renaming category column

catsplit = catsplit.rename(columns={0:'category_1', 1:'category_2', 2:'categor
```

In [ ]:

```
#Adding categories to the new dataframe

df1['category_1'] = catsplit['category_1']
df1['category_2'] = catsplit['category_2']

df1.drop(columns='category', inplace=True)

df1
```

Out[ ]:

	product_id	product_name	discounted_price	actual_price	discount_perc
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	399.0	1099.0	
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	199.0	349.0	
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	199.0	1899.0	
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	329.0	699.0	
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	154.0	399.0	
...	...	...	...	...	
1460	B08L7J3T31	Noir Aqua - 5pcs PP Spun Filter + 1 Spanner | ...	379.0	919.0	
1461	B01M6453MB	Prestige Delight PRWO Electric Rice Cooker (1 ...	2280.0	3045.0	
1462	B009P2LIL4	Bajaj Majesty RX10 2000 Watts Heat Convector R...	2219.0	3080.0	
1463	B00J5DYCCA	Havells Ventil Air DSP 230mm Exhaust Fan (Pist...	1399.0	1890.0	
1464	B01486F4G6	Borosil Jumbo 1000-Watt Grill Sandwich Maker (...	2863.0	3690.0	

1465 rows × 9 columns

In [ ]:

```
#Checking category_1 unique values

df1['category_1'].value_counts()
```

```
Out[ ]:                            count
                     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

dtype: int64
```

```
In [ ]: #Fixing Strings in Category_2 column

        df1['category_2'] = df1['category_2'].str.replace('&', ' & ')
        df1['category_2'] = df1['category_2'].str.replace(',', ', ')
        df1['category_2'] = df1['category_2'].str.replace('HomeAppliances', 'Home Appl
        df1['category_2'] = df1['category_2'].str.replace('AirQuality', 'Air Quality')
        df1['category_2'] = df1['category_2'].str.replace('WearableTechnology', 'Weara
        df1['category_2'] = df1['category_2'].str.replace('NetworkingDevices', 'Networ
        df1['category_2'] = df1['category_2'].str.replace('OfficePaperProducts', 'Offi
        df1['category_2'] = df1['category_2'].str.replace('ExternalDevices', 'External
        df1['category_2'] = df1['category_2'].str.replace('DataStorage', 'Data Storage
        df1['category_2'] = df1['category_2'].str.replace('HomeStorage', 'Home Storage
        df1['category_2'] = df1['category_2'].str.replace('HomeAudio', 'Home Audio')
        df1['category_2'] = df1['category_2'].str.replace('GeneralPurposeBatteries', '
        df1['category_2'] = df1['category_2'].str.replace('BatteryChargers', 'Battery 
        df1['category_2'] = df1['category_2'].str.replace('CraftMaterials', 'Craft Mat
        df1['category_2'] = df1['category_2'].str.replace('OfficeElectronics', 'Office
        df1['category_2'] = df1['category_2'].str.replace('PowerAccessories', 'Power A
        df1['category_2'] = df1['category_2'].str.replace('CarAccessories', 'Car Acces
        df1['category_2'] = df1['category_2'].str.replace('HomeMedicalSupplies', 'Home
        df1['category_2'] = df1['category_2'].str.replace('HomeTheater', 'Home Theater
```

```
In [ ]: # Removing Whitespace from product_id

        df1['product_id'].str.strip()
```

Out[ ]:

	product_id
0	B07JW9H4J1
1	B098NS6PVG
2	B096MSW6CT
3	B08HDJ86NZ
4	B08CF3B7N1
...	...
1460	B08L7J3T31
1461	B01M6453MB
1462	B009P2LIL4
1463	B00J5DYCCA
1464	B01486F4G6

1465 rows × 1 columns

dtype: object

In [ ]:

```python
# Convert 'rating' to numeric, coercing errors
df1['rating'] = pd.to_numeric(df1['rating'], errors='coerce')

rating_score = []

for score in df1['rating']:
    if pd.isna(score): # Handle NaN values
        rating_score.append('Unknown')
    elif 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')

df1['rating_score'] = rating_score
print(df1['rating_score'])
```

```
0       Above Average
1       Above Average
2             Average
3       Above Average
4       Above Average
            ...
1460    Above Average
1461    Above Average
1462          Average
1463    Above Average
1464    Above Average
Name: rating_score, Length: 1465, dtype: object
```

In [ ]:

```
#Creating Difference of Price Column between Actual Price and Discounted Price

df1['difference_price'] = df1['actual_price'] - df1['discounted_price']
```

In [ ]:

```
#Result After Cleaning and Preperation after first cleaned dataframe

df1.head()
```

Out[ ]:

	product_id	product_name	discounted_price	actual_price	discount_percent
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	399.0	1099.0	
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	199.0	349.0	
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	199.0	1899.0	
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	329.0	699.0	
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	154.0	399.0	

In [ ]:

```
reviewers = df[['user_id','user_name']]
reviewers
```

Out[ ]:

	user_id	
0	AG3D6O4STAQKAY2UVGEUV46KN35Q,AHMY5CWJMMK5BJRBB...	M gupta,Sunde A
1	AECPFYFQVRUWC3KGNLJIOREFP5LQ,AGYYVPDD7YG7FYNBX...	ArdKn,Nirbhay Viswanatha
2	AGU3BBQ2V2DDAMOAKGFAWDDQ6QHA,AESFLDV2PT363T2AQ...	Kunal,Himanshu,v niharka
3	AEWAZDZZJLQUYVOVGBEUKSLXHQ5A,AG5HTSFRRE6NL3M5S...	dhale,JD,HEMALA a.,a
4	AE3Q6KSUK5P75D5HFYHCRAOLODSA,AFUGIFH5ZAFXRDSZH...	rahuls Wadke,P
...	...	
1460	AHITFY6AHALOFOHOZEOC6XBP4FEA,AFRABBODZJZQB6Z4U...	Prabha ds,Raghu Deal,Amazo
1461	AFG5FM3NEMOL6BNFRV2NK5FNJCHQ,AGEINTRN6Z563RMLH...	Bhai,Naveen Sangma,JA
1462	AGVPWCMAHYQWJOQKMUJN4DW3KM5Q,AF4Q3E66MY4SR7YQZ...	Nehal Par Custo
1463	AF2JQCLSCY3QJATWUNNHUSVUPNQQ,AFDMLUXC5LS5RXDJS...	Dubey,E.GURUBA S.,
1464	AFGW5PT3R6ZAVQR4Y5MWVAKBZAYA,AG7QNJ2SCS5VS5VYY...	Rajib, Kahol,PARD

1465 rows × 2 columns

In [ ]:

```
#Splitting the strings in user_id column

reviewer_id_split = reviewers['user_id'].str.split(',', expand=False)

reviewer_id_split
```

Out[ ]:

	user_id
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...

1465 rows × 1 columns

dtype: object

In [ ]:

```
reviewer_id_exp = reviewer_id_split.explode()

reviewer_id_clean = reviewer_id_exp.reset_index(drop=True)

reviewer_id_clean
```

Out[ ]:

	user_id
0	AG3D6O4STAQKAY2UVGEUV46KN35Q
1	AHMY5CWJMMK5BJRBBSNLYT3ONILA
2	AHCTC6ULH4XB6YHDY6PCH2R772LQ
3	AGYHHIERNXKA6P5T7CZLXKVPT7IQ
4	AG4OGOFWXJZTQ2HKYIOCOY3KXF2Q
...	...
11498	AHXCDNSXAESERITAFELQABFVNLCA
11499	AGRZD6CHLCUNOLMMIMIHUCG7PIFA
11500	AFQZVGSOSOJHKFQQMCEI4725QEKQ
11501	AEALVGXXIP46OZVXKRUXSDWZJMEA
11502	AGEFL3AY7YXEFZA4ZJU3LP7K7OJQ

11503 rows × 1 columns

dtype: object

In [ ]:

```
#Splitting the strings in user_name column

reviewer_name_split = reviewers['user_name'].str.split(',', expand=False)

reviewer_name_split
```

Out[ ]:

	user_name
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 ...

1465 rows × 1 columns

dtype: object

In [ ]:

```
#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
```

Out[ ]:

	user_name
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.

11515 rows × 1 columns

dtype: object

In [ ]:

```
#Creating 2 Data Frames to be merged

df21 = pd.DataFrame(data=reviewer_id_clean)
df22 = pd.DataFrame(data=reviewer_name_clean)
```

In [ ]:

```
#Merging the 2 dataframe containing user_id and user_name

df2 = pd.merge(df21, df22, left_index=True, right_index=True)
df2.head()
```

Out[ ]:

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

DATA EXPLORATION

In [ ]:

```
#Setting Visualization Style
```

```

sns.set_style(style='darkgrid')

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

In [ ]:

```
#Main Category and Sub-Category

main_sub = df1[['category_1', 'category_2', 'product_id']]

main_sub = main_sub.rename(columns={'category_1' :'Main Category', 'category_2

main_sub_piv = pd.pivot_table(main_sub, index=['Main Category', 'Sub-Category'

main_sub_piv
```

Out[ ]:

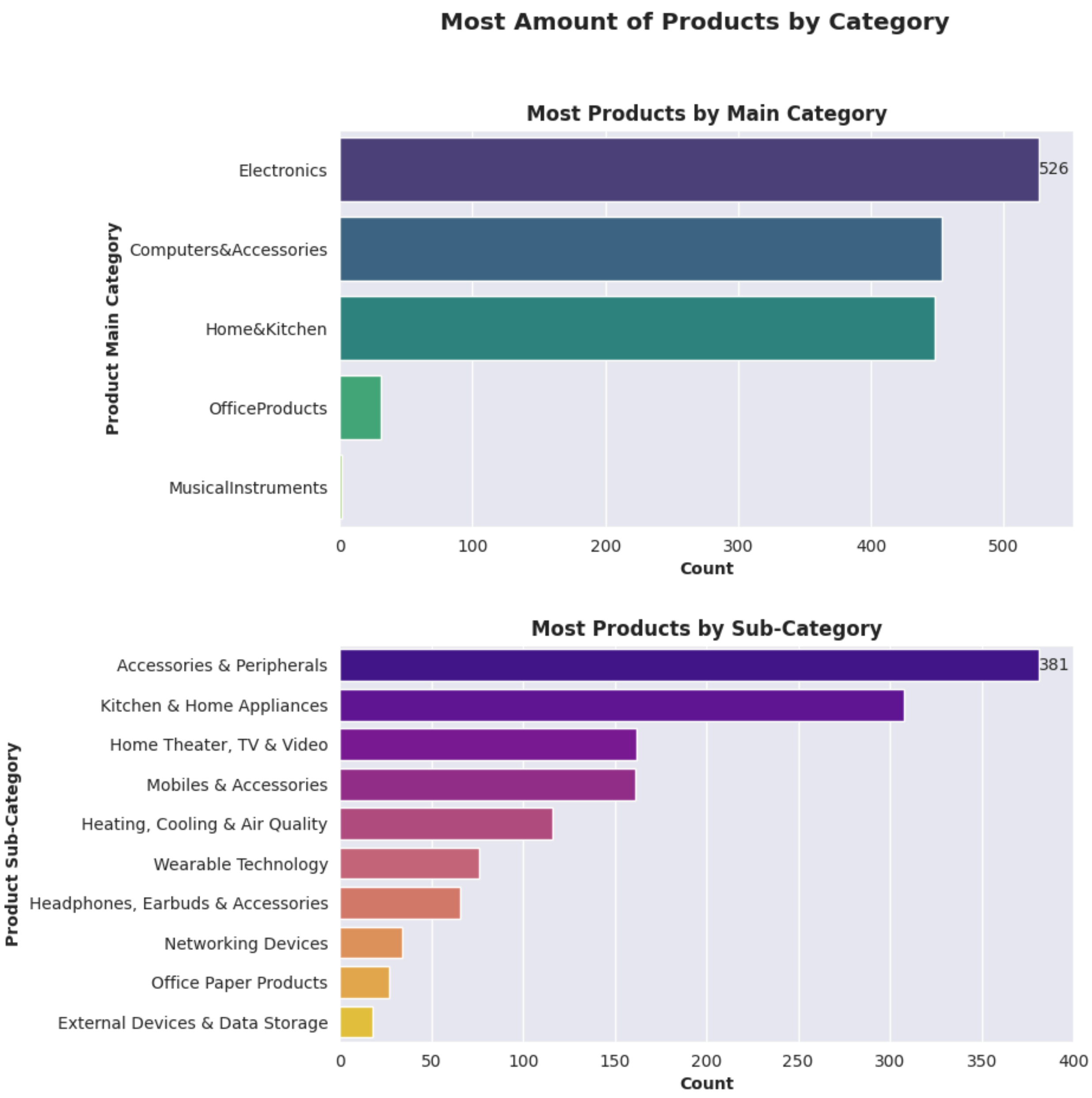
		Product ID
Main Category	Sub-Category	
Car&Motorbike	Car Accessories	1
Computers&Accessories	Accessories & Peripherals	381
	Components	5
	External Devices & Data Storage	18
	Laptops	1
	Monitors	2
	Networking Devices	34
	Printers, Inks & Accessories	11
	Tablets	1
Electronics	Accessories	14
	Cameras & 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
HomeImprovement	Electrical	2
MusicalInstruments	Microphones	2
OfficeProducts	Office Electronics	4
	Office Paper Products	27
Toys&Games	Arts & Crafts	1

Data Visualization

In [ ]:

```
#Most amount of products by category

most_main_items = df1['category_1'].value_counts().head(5).rename_axis('catego

most_sub_items = df1['category_2'].value_counts().head(10).rename_axis('catego

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

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

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()
```

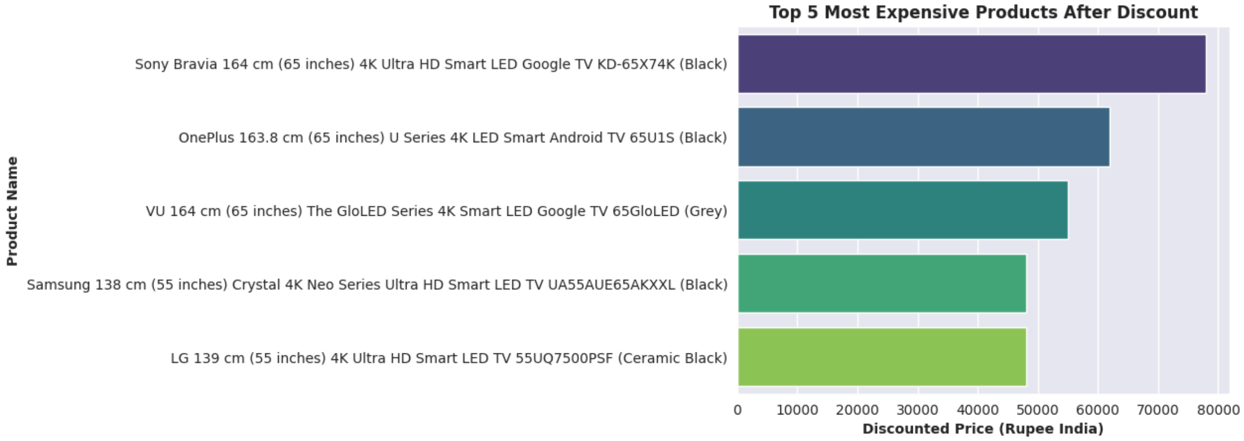
```
/tmp/ipython-input-897027567.py:10: FutureWarning:

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

  sns.barplot(ax=ax[0], data=most_main_items, x='counts', y='category_1', palet
te='viridis')
/tmp/ipython-input-897027567.py:11: FutureWarning:

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

  sns.barplot(ax=ax[1], data=most_sub_items, x='counts', y='category_2', palett
e='plasma')
```



In [ ]:

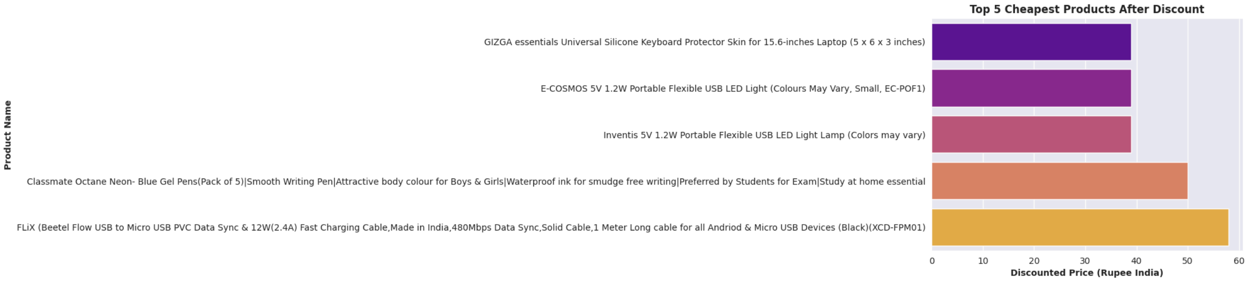
```
#Top 5 Most Expensive Products After Discount

disc_exp = sns.barplot(data=df1.sort_values('discounted_price', ascending=Fals

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

plt.show()
```

```
/tmp/ipython-input-1218021803.py:3: FutureWarning:

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

  disc_exp = sns.barplot(data=df1.sort_values('discounted_price', ascending=Fal
se).head(5), x='discounted_price', y='product_name', palette='viridis')
```



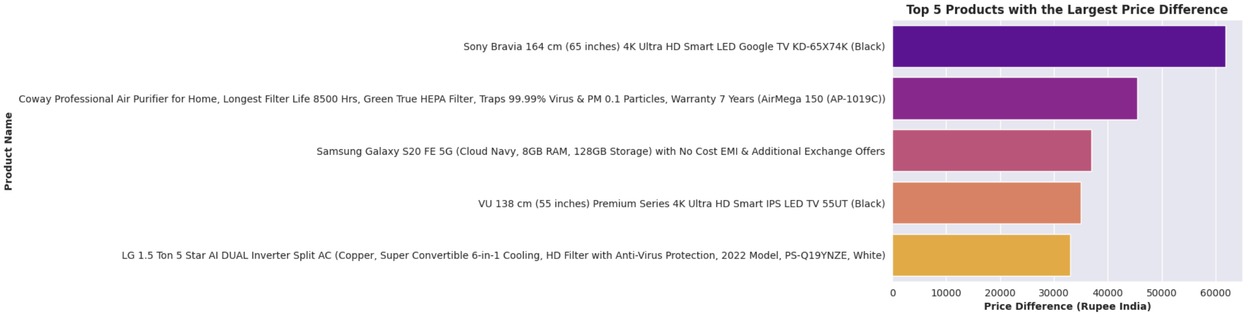
```
In [ ]: #Top 5 Cheapest Products After Discount

        disc_cheap = sns.barplot(data=df1.sort_values('discounted_price').head(5), x='

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

        plt.show()
```

```
/tmp/ipython-input-1665944627.py:3: FutureWarning:

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

  disc_cheap = sns.barplot(data=df1.sort_values('discounted_price').head(5),
x='discounted_price', y='product_name', palette='plasma')
```



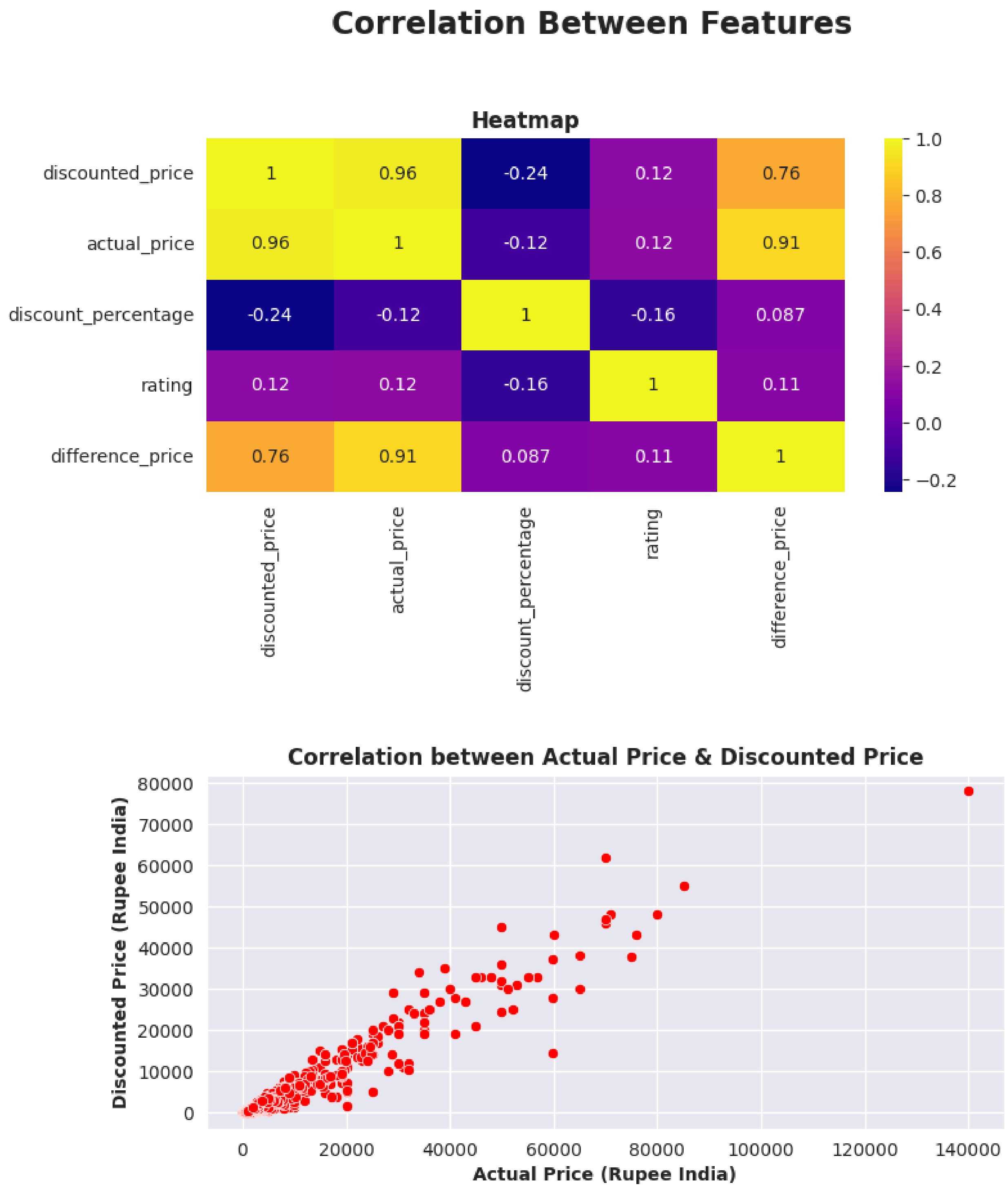
```
In [ ]: #Top 5 Products with the largest difference in price due to discount

        dif_price_large = sns.barplot(data= df1.sort_values('difference_price', ascend
        dif_price_large.set_title('Top 5 Products with the Largest Price Difference',
        dif_price_large.set_xlabel('Price Difference (Rupee India)', fontweight='bold'
        dif_price_large.set_ylabel('Product Name', fontweight='bold')

        plt.show()
```

```
/tmp/ipython-input-3422872111.py:3: FutureWarning:

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

  dif_price_large = sns.barplot(data= df1.sort_values('difference_price', ascen
ding=False).head(5), x='difference_price', y='product_name', palette='plasma' )
```



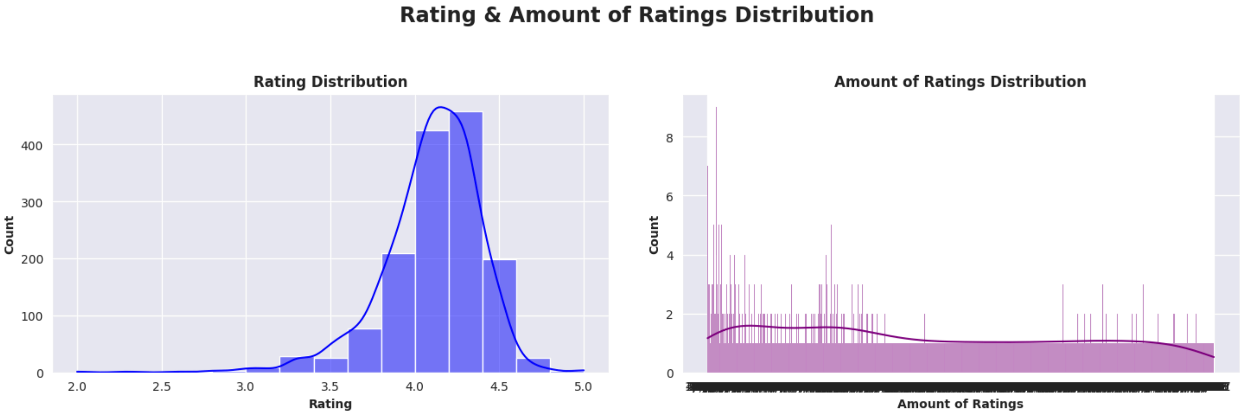
In [ ]:

```
#Heatmap & Correlation between Actual Price & Discounted Price

fig, ax = plt.subplots(2, 1, figsize=(8, 10))

fig.suptitle('Correlation Between Features', fontweight='heavy', size='xx-larg

# Select only numeric columns for correlation and heatmap
numeric_df1 = df1.select_dtypes(include=np.number)

sns.heatmap(ax=ax[0], data=numeric_df1.corr(), annot=True, cmap='plasma')
sns.scatterplot(ax=ax[1], data=df1, y='discounted_price', x='actual_price', co

plt.subplots_adjust(hspace = 0.8)

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

ax[0].set_title('Heatmap', fontweight='bold')
ax[1].set_title('Correlation between Actual Price & Discounted Price', fontwei

plt.show()
```



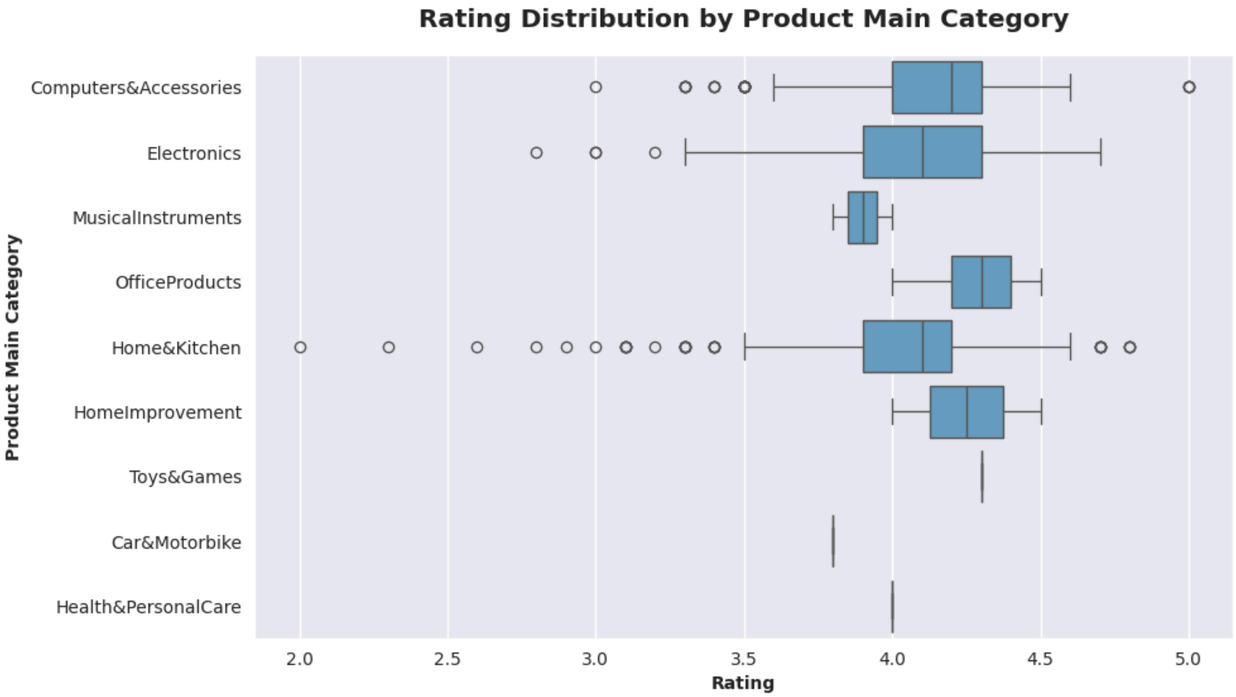
In [ ]:

```
fig, ax = plt.subplots(1, 2, figsize=(15, 5))

fig.suptitle('Rating & Amount of Ratings Distribution', fontweight='heavy', si

fig.tight_layout(pad=3.0)

sns.histplot(ax=ax[0], data=df1, x='rating', bins=15, kde=True, color='blue')
sns.histplot(ax=ax[1], data=df1, x='rating_count', bins=10, kde=True, color='p
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()
```



In [ ]:

```
#Rating Distribution by Product Main Category

fig, ax = plt.subplots(figsize=(10, 6))

sns.boxplot(ax=ax, data=df1, 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

plt.show()
```



In [ ]:

```
#Rating of Products based on Rating Category

rate_main_cat = df1.groupby(['category_1','rating_score']).agg('count').iloc[:

rate_main_cat = rate_main_cat.rename(columns = {'category_1' : 'Main Category'

rate_main_cat
```

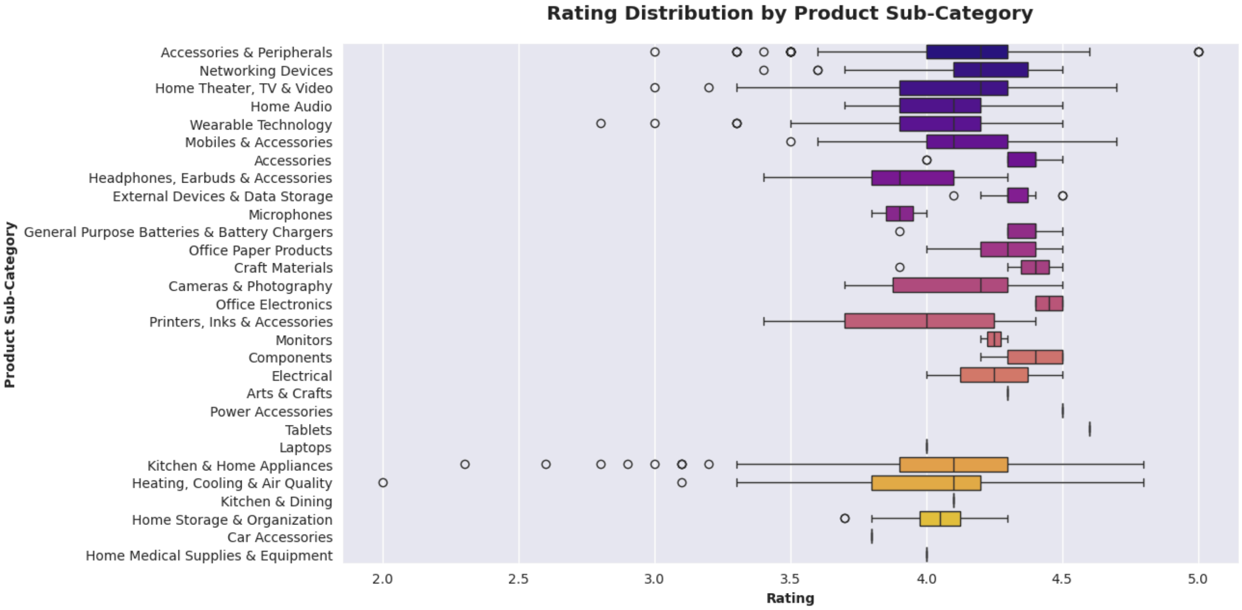
Out[ ]:

	Main Category	Rating Category	Amount
0	Car&Motorbike	Average	1
1	Computers&Accessories	Above Average	375
2	Computers&Accessories	Average	75
3	Computers&Accessories	Excellent	3
4	Electronics	Above Average	393
5	Electronics	Average	132
6	Electronics	Below Average	1
7	Health&PersonalCare	Above Average	1
8	Home&Kitchen	Above Average	303
9	Home&Kitchen	Average	139
10	Home&Kitchen	Below Average	5
11	Home&Kitchen	Unknown	1
12	HomeImprovement	Above Average	2
13	MusicalInstruments	Above Average	1
14	MusicalInstruments	Average	1
15	OfficeProducts	Above Average	31
16	Toys&Games	Above Average	1

In [ ]:

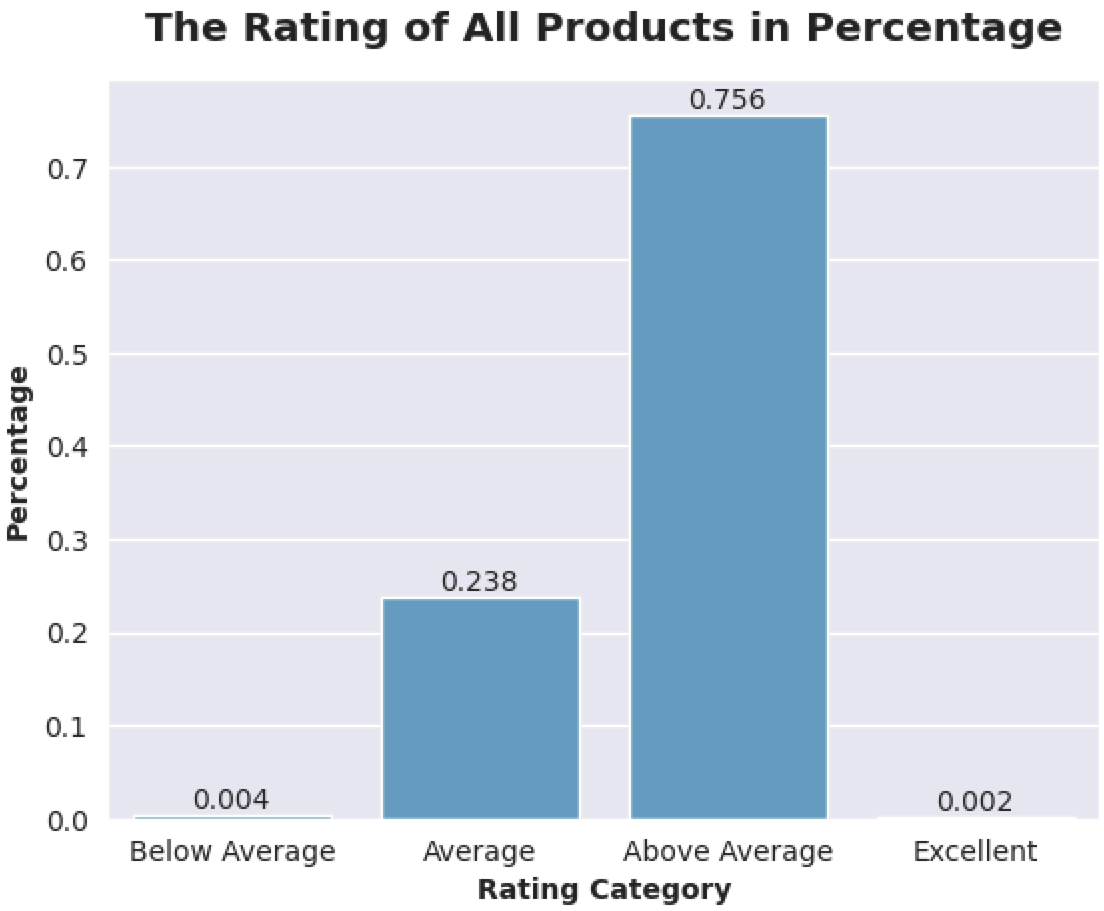
```python
#Rating Distribution by Product Sub-Category

fig, ax = plt.subplots(figsize=(12, 7))

sns.boxplot(ax=ax, data=df1, x='rating', y='category_2', palette='plasma')

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'

plt.show()
```

```
/tmp/ipython-input-1482046637.py:5: FutureWarning:

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

  sns.boxplot(ax=ax, data=df1, x='rating', y='category_2', palette='plasma')
```



In [ ]:

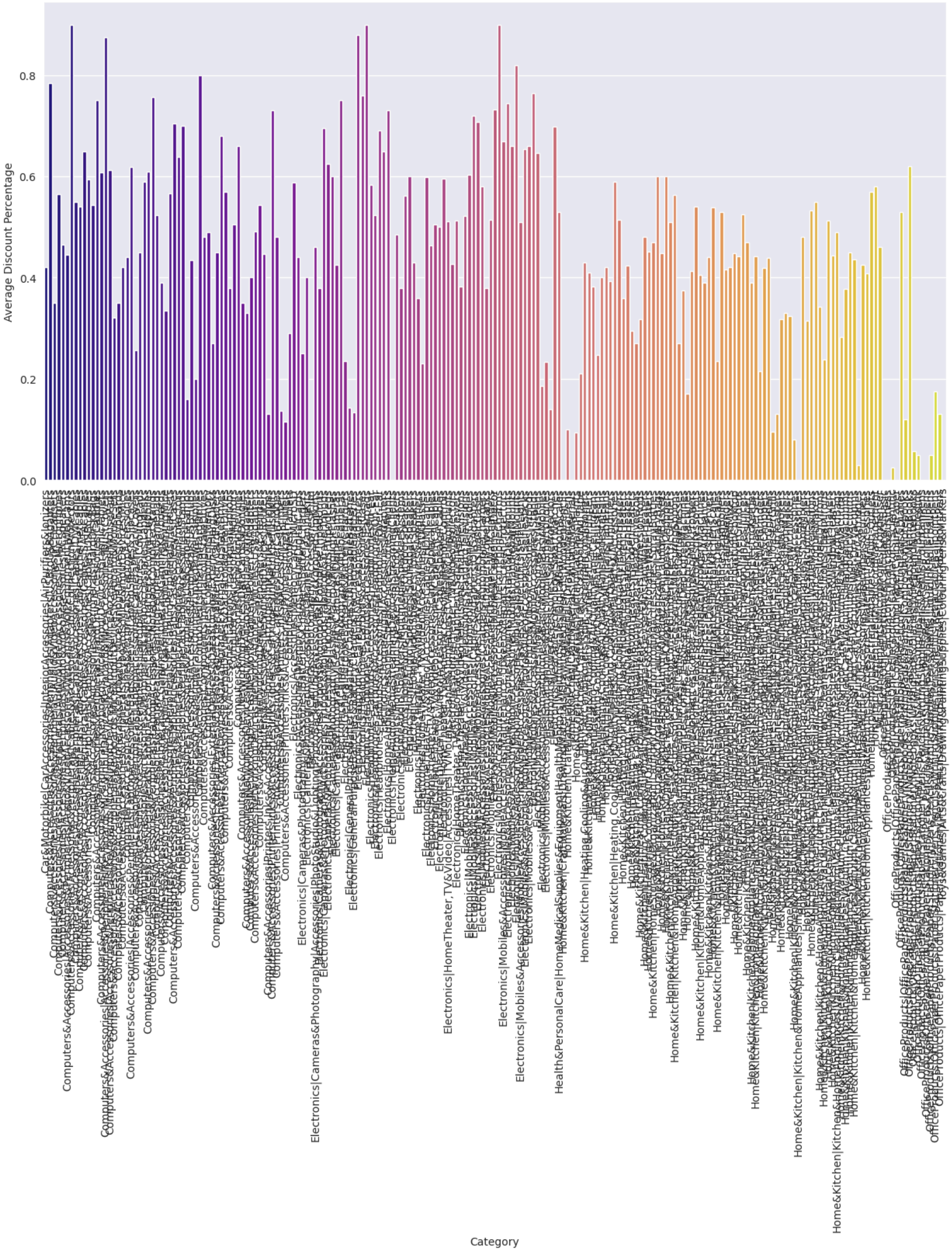
```
#The Rating of All Products in Percentage

rating_ordered = ['Below Average', 'Average', 'Above Average', 'Excellent']

rating_count = df1['rating_score'].value_counts(normalize=True).rename_axis('r

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

rating_count_plot = sns.barplot(data=rating_count, x='rating', y='counts', orc

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', fontwe


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



```
In [ ]: # Calculate average discount percentage per category
        avg_discount_per_category = df.groupby('category')['discount_percentage'].mear

        # Display results
        print(avg_discount_per_category)

        # Optional: Visualization
        sns.barplot(x=avg_discount_per_category.index, y=avg_discount_per_category.val
        plt.xlabel("Category")
        plt.ylabel("Average Discount Percentage")
        plt.xticks(rotation=90) # Rotate x-axis labels for better readability
        plt.tight_layout() # Adjust layout to prevent labels from overlapping
        plt.show()
```

```
category
Car&Motorbike|CarAccessories|InteriorAccessories|AirPurifiers&Ionizers
0.420
Computers&Accessories|Accessories&Peripherals|Adapters|USBtoUSBAdapters
0.785
Computers&Accessories|Accessories&Peripherals|Audio&VideoAccessories|PCHeadsets
0.350
Computers&Accessories|Accessories&Peripherals|Audio&VideoAccessories|PCMicropho
nes                                        0.565
Computers&Accessories|Accessories&Peripherals|Audio&VideoAccessories|PCSpeakers
0.465

...
OfficeProducts|OfficePaperProducts|Paper|Stationery|Pens,Pencils&WritingSupplie
s|Pens&Refills|GelInkRollerballPens        0.000
OfficeProducts|OfficePaperProducts|Paper|Stationery|Pens,Pencils&WritingSupplie
s|Pens&Refills|LiquidInkRollerballPens     0.050
OfficeProducts|OfficePaperProducts|Paper|Stationery|Pens,Pencils&WritingSupplie
s|Pens&Refills|RetractableBallpointPens    0.175
OfficeProducts|OfficePaperProducts|Paper|Stationery|Pens,Pencils&WritingSupplie
s|Pens&Refills|StickBallpointPens          0.130
Toys&Games|Arts&Crafts|Drawing&PaintingSupplies|ColouringPens&Markers
0.000
Name: discount_percentage, Length: 211, dtype: float64
```

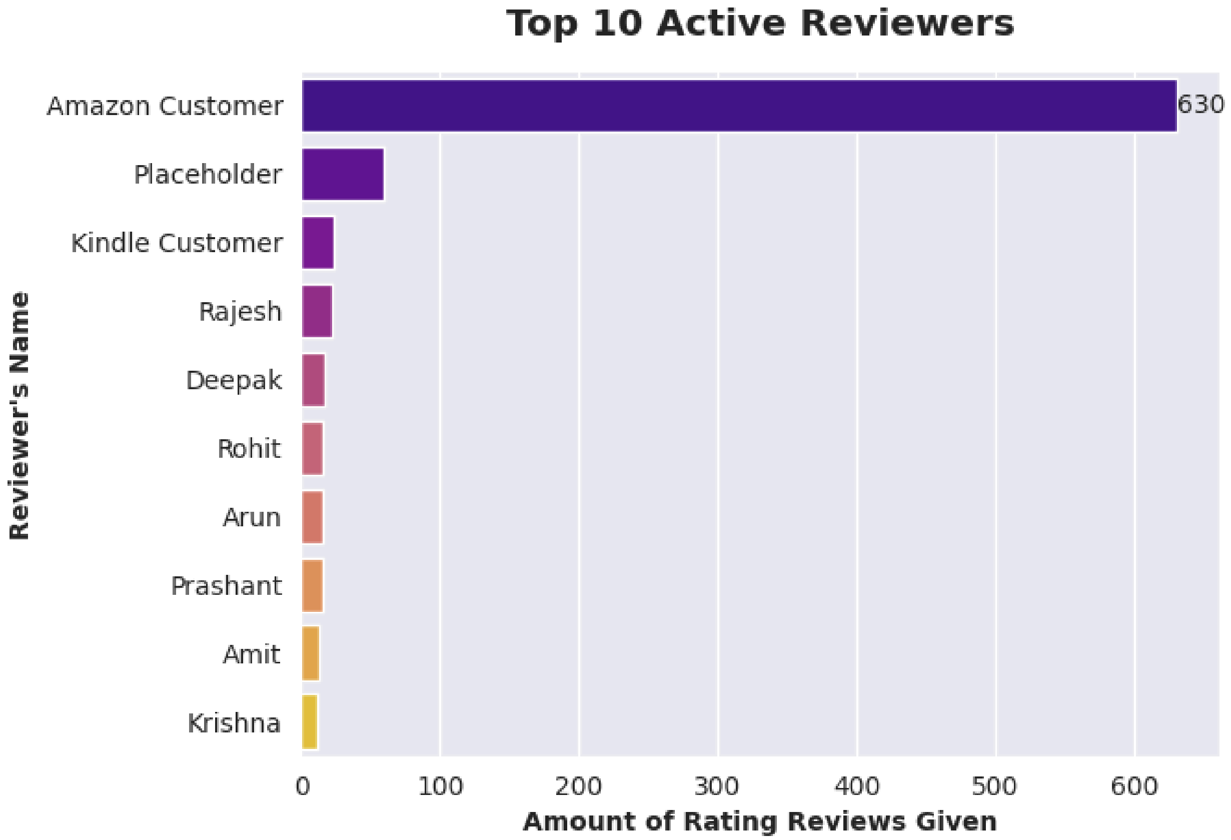
```
/tmp/ipython-input-849646895.py:9: FutureWarning:

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

  sns.barplot(x=avg_discount_per_category.index, y=avg_discount_per_category.va
lues , palette='plasma')
/tmp/ipython-input-849646895.py:13: UserWarning: Tight layout not applied. The
bottom and top margins cannot be made large enough to accommodate all Axes deco
rations.
  plt.tight_layout() # Adjust layout to prevent labels from overlapping
```



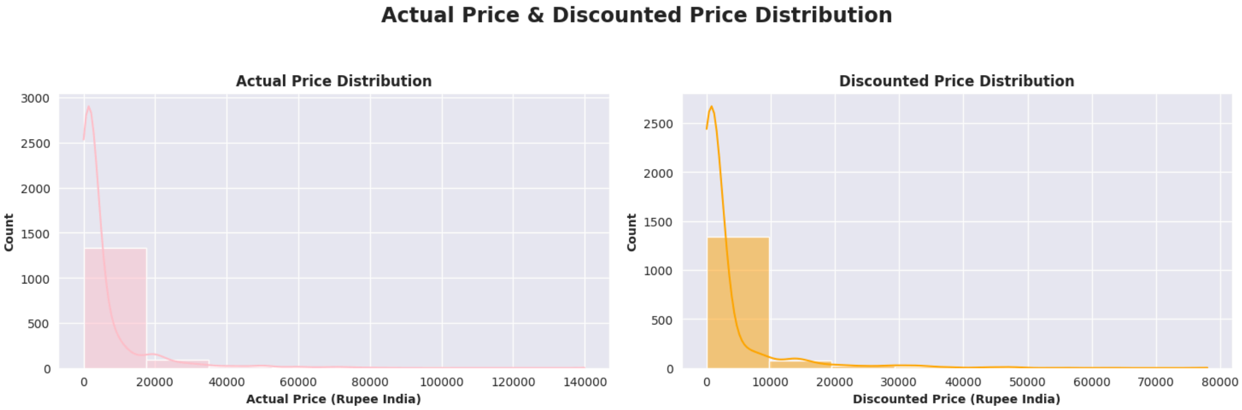
In [ ]:

```
#Reviewers who gave ratings and reviews for more than one product

top_reviewer = data=df2['user_name'].value_counts().head(10).rename_axis('user
top_review_plot = sns.barplot(data=top_reviewer, x='counts', y='username',pale

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=

plt.show()
```

```
/tmp/ipython-input-3043782225.py:5: FutureWarning:

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

  top_review_plot = sns.barplot(data=top_reviewer, x='counts', y='username',pal
ette='plasma' )
```

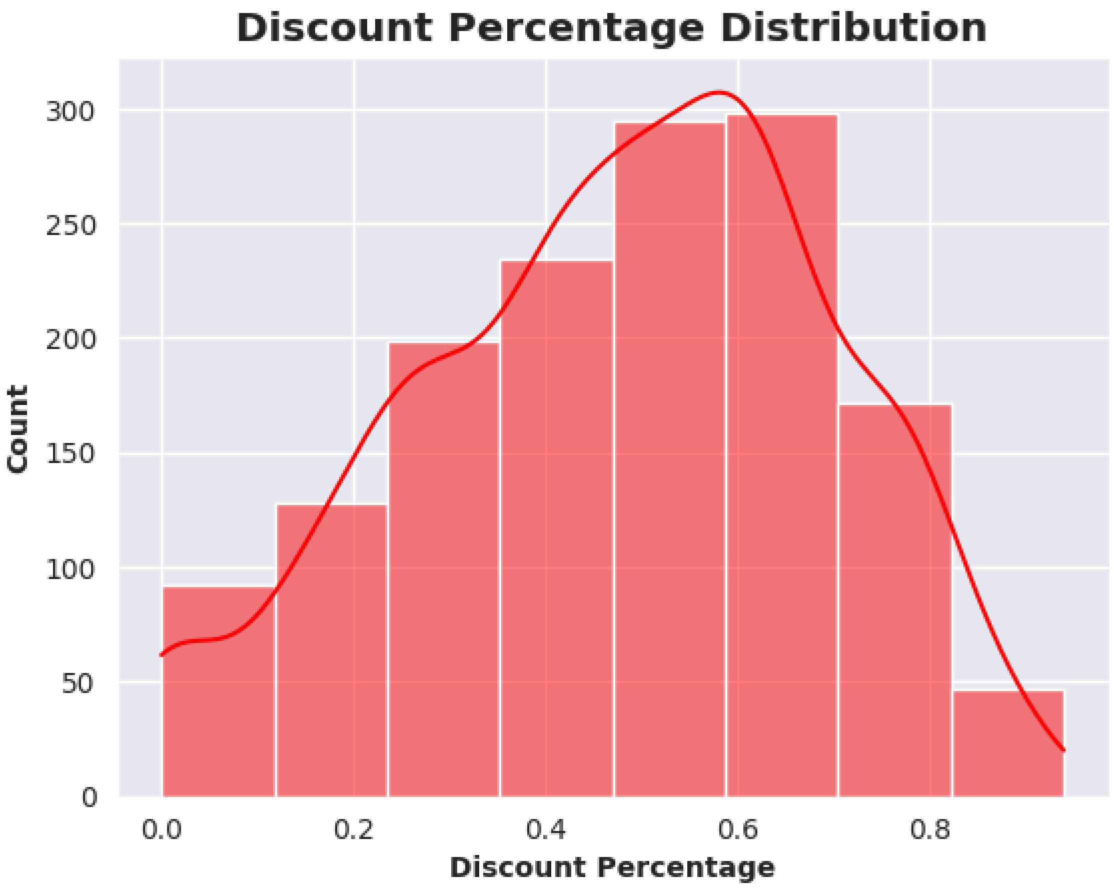


```
In [ ]: #Actual Price & Discounted Price Distribution
        fig, ax = plt.subplots(1, 2, figsize=(15, 5))

        fig.suptitle('Actual Price & Discounted Price Distribution', fontweight='heavy

        fig.tight_layout(pad=3.0)

        sns.histplot(ax=ax[0], data=df1, x='actual_price', bins=8, kde=True, color='pi
        sns.histplot(ax=ax[1], data=df1, x='discounted_price', bins=8, kde=True, color
        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()
```



In [ ]:

```
#Discount Percentage Distribution

disc_hist = sns.histplot(data=df1, x='discount_percentage', bins=8, kde=True,


disc_hist.set_xlabel('Discount Percentage', fontweight='bold')
disc_hist.set_ylabel('Count', fontweight='bold')
disc_hist.set_title('Discount Percentage Distribution', fontweight='heavy', si

plt.show()
```



```
In [ ]: df1['discount_percentage'].describe()
```

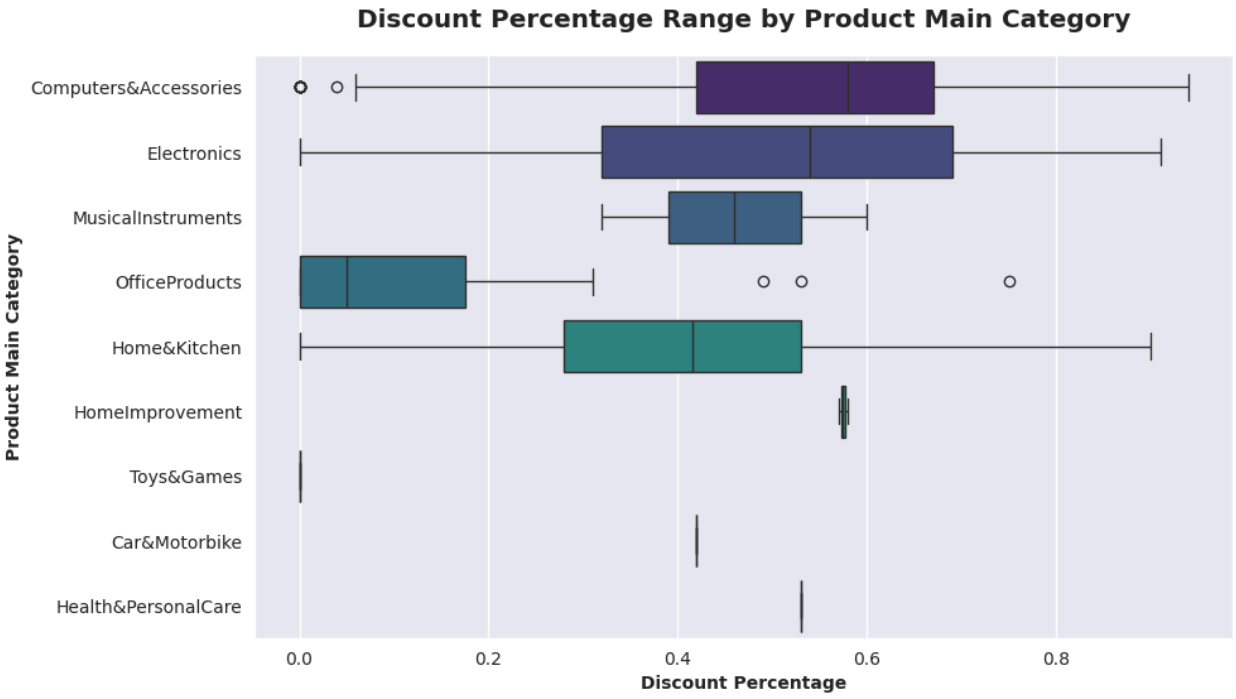
Out[ ]:

	discount_percentage
count	1465.000000
mean	0.476915
std	0.216359
min	0.000000
25%	0.320000
50%	0.500000
75%	0.630000
max	0.940000

dtype: float64

```
In [ ]: # The Discount Range by Product Main Category

        fig, ax = plt.subplots(figsize=(10, 6))

        sns.boxplot(data=df1, x='discount_percentage', y='category_1', palette='viridi
        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=


        plt.show()
```

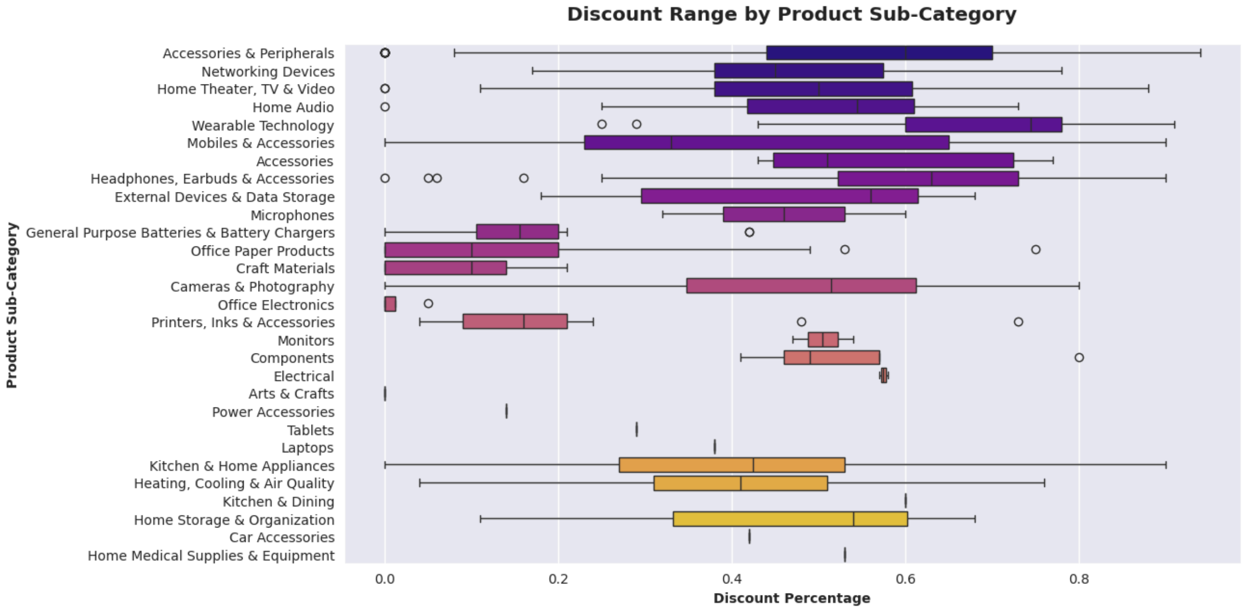
```
/tmp/ipython-input-1620759910.py:5: FutureWarning:

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

  sns.boxplot(data=df1, x='discount_percentage', y='category_1', palette='virid
is')
```



In [ ]:

```
# The Discount Range by Product Sub-Category

fig, ax = plt.subplots(figsize=(12, 7))

sns.boxplot(data=df1, x='discount_percentage', y='category_2', palette='plasma

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', siz

plt.show()
```

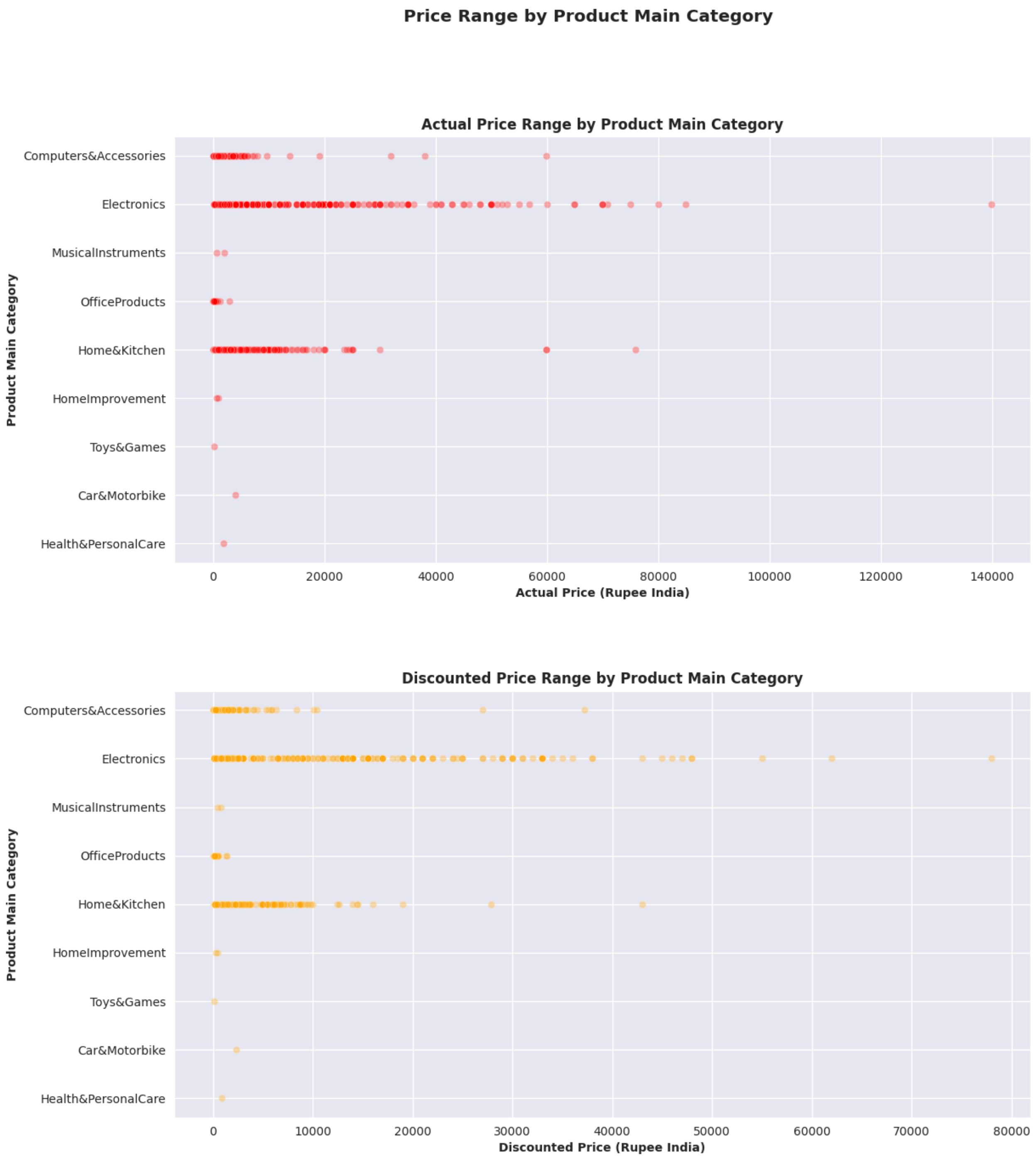
```
/tmp/ipython-input-2132455771.py:5: FutureWarning:

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

  sns.boxplot(data=df1, x='discount_percentage', y='category_2', palette='plasm
a')
```



Discount Range by Product Sub-Category

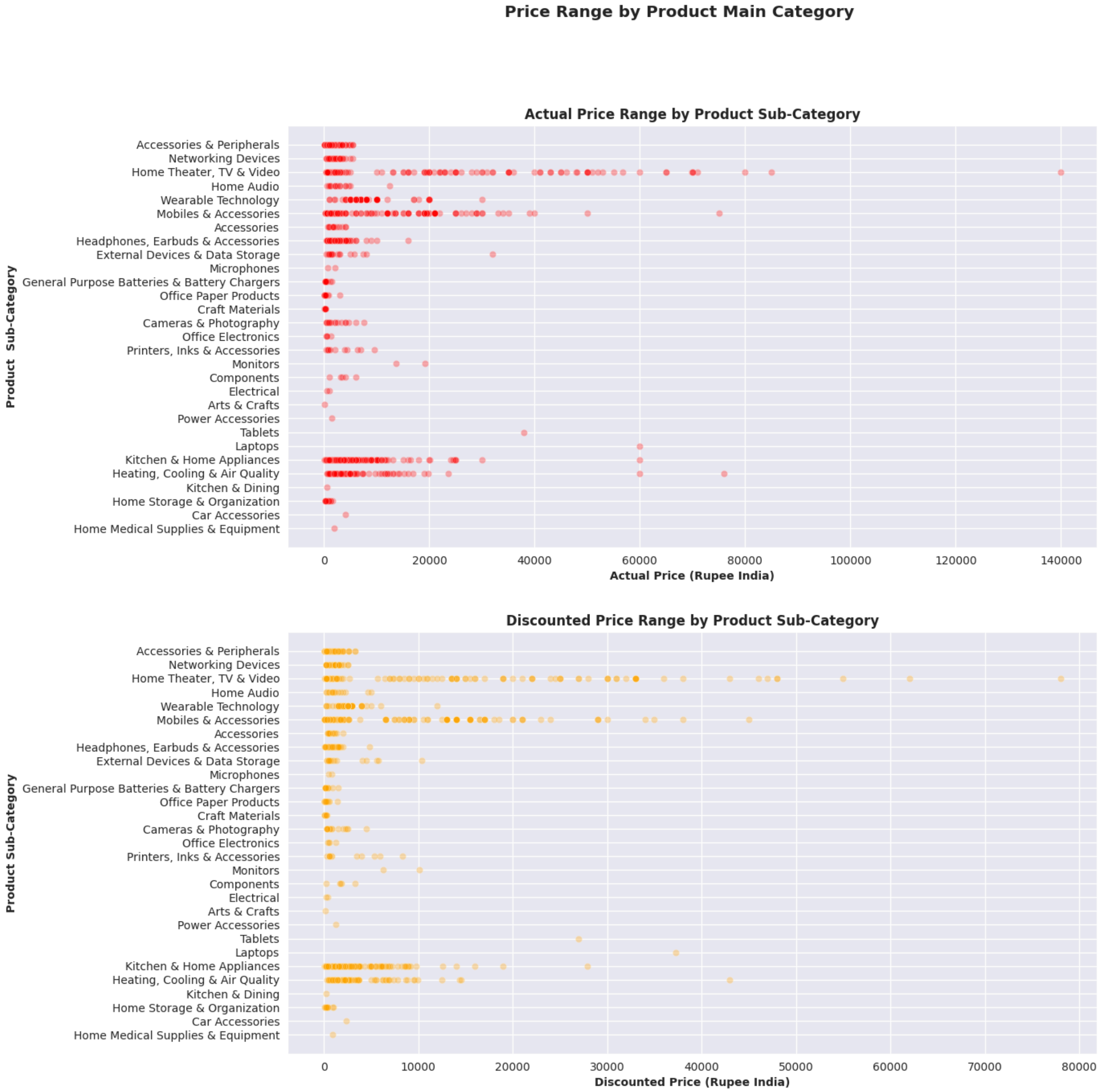
In [ ]:

```python
#Actual Price Range and Discounted Price Range by Product Main Category

fig, ax = plt.subplots(2, 1, figsize=(13,15))

fig.suptitle('Price Range by Product Main Category', fontweight='heavy', size=

sns.scatterplot(ax=ax[0], data=df1, x='actual_price', y='category_1', alpha=0.
sns.scatterplot(ax=ax[1], data=df1, x='discounted_price', y='category_1', alph

ax[0].set_xlabel('Actual Price (Rupee India)', fontweight='bold')
ax[0].set_ylabel('Product Main Category', fontweight='bold')
ax[0].set_title('Actual Price Range by Product Main Category', fontweight='bol

ax[1].set_xlabel('Discounted Price (Rupee India)', fontweight='bold')
ax[1].set_ylabel('Product Main Category', fontweight='bold')
ax[1].set_title('Discounted Price Range by Product Main Category', fontweight=

plt.subplots_adjust(hspace = 0.3)
plt.show()
```



In [ ]:

```
fig, ax = plt.subplots(2, 1, figsize=(13, 15))

fig.suptitle('Price Range by Product Main Category', fontweight='heavy', size=

sns.scatterplot(ax=ax[0], data=df1, x='actual_price', y='category_2', alpha=0.
sns.scatterplot(ax=ax[1], data=df1, x='discounted_price', y='category_2', alph

ax[0].set_xlabel('Actual Price (Rupee India)', fontweight='bold')
ax[0].set_ylabel('Product  Sub-Category', fontweight='bold')
ax[0].set_title('Actual Price Range by Product Sub-Category', fontweight='bold
ax[1].set_xlabel('Discounted Price (Rupee India)', fontweight='bold')
ax[1].set_ylabel('Product Sub-Category', fontweight='bold')
ax[1].set_title('Discounted Price Range by Product Sub-Category', fontweight='

plt.subplots_adjust(hspace = 0.2)

plt.show()
```



In [ ]:

```
#Pivot table of Prices

def p25(g):
    return np.percentile(g, 25)

def p75(g):
    return np.percentile(g, 75)
```

```python
actual_price_pivot = df1.pivot_table(values=['actual_price', 'discounted_price

actual_price_pivot
```

```
/tmp/ipython-input-3882879156.py:9: FutureWarning: The provided callable <funct
ion median at 0x7fcaa21d4540> is currently using DataFrameGroupBy.median. In a
future version of pandas, the provided callable will be used directly. To keep
current behavior pass the string "median" instead.
  actual_price_pivot = df1.pivot_table(values=['actual_price', 'discounted_pric
e'], index=['category_1', 'category_2'], aggfunc=([p25, np.median, np.mean, p7
5]))
/tmp/ipython-input-3882879156.py:9: FutureWarning: The provided callable <funct
ion mean at 0x7fcaa277c220> is currently using DataFrameGroupBy.mean. In a futu
re version of pandas, the provided callable will be used directly. To keep curr
ent behavior pass the string "mean" instead.
  actual_price_pivot = df1.pivot_table(values=['actual_price', 'discounted_pric
e'], index=['category_1', 'category_2'], aggfunc=([p25, np.median, np.mean, p7
5]))
```

Out[ ]:

		p25		
		actual_price	discounted_price	actual_pri
category_1	category_2			
Car&Motorbike	Car Accessories	4000.00	2339.00	4000
Computers&Accessories	Accessories & Peripherals	499.00	199.00	999
	Components	3100.00	1709.00	3500
	External Devices & Data Storage	1074.25	504.00	1575
	Laptops	59890.00	37247.00	59890
	Monitors	15090.00	7249.00	16430
	Networking Devices	1208.00	530.00	1949
	Printers, Inks & Accessories	811.00	597.00	1999
	Tablets	37999.00	26999.00	37999
Electronics	Accessories	1150.00	479.00	1800
	Cameras & Photography	946.00	386.50	1999
	General Purpose Batteries & Battery Chargers	205.00	166.75	282
	Headphones, Earbuds & Accessories	999.00	450.50	1994
	Home Audio	1274.00	736.50	2394
	Home Theater, TV & Video	824.00	349.00	2749
	Mobiles & Accessories	1299.00	399.00	2999
	Power Accessories	1499.00	1289.00	1499
	Wearable	5999.00	1599.00	7990

			p25	
		actual_price	**discounted_price**	**actual_pri**
category_1	**category_2**			
	Technology			
Health&PersonalCare	**Home Medical Supplies & Equipment**	1900.00	899.00	190
Home&Kitchen	**Craft Materials**	132.50	114.50	22
	Heating, Cooling & Air Quality	1990.00	1049.00	306
	Home Storage & Organization	374.00	199.00	64
	Kitchen & Dining	495.00	199.00	49
	Kitchen & Home Appliances	1000.00	596.00	196
HomeImprovement	**Electrical**	699.00	293.00	79
MusicalInstruments	**Microphones**	1023.00	558.00	134
OfficeProducts	**Office Electronics**	511.25	501.50	54
	Office Paper Products	120.00	107.00	17
Toys&Games	**Arts & Crafts**	150.00	150.00	15

Predicting Discounted Price of Products

I will attempt to make a Simple Linear Regression Model. The Independent Varaible will be the Actual Price and the Dependent Variable will be Discounted Price.

In [ ]:

```
#Extracting Independent and Dependent Variables

X = df1.iloc[:, 3].values.reshape(-1, 1)
y = df1.iloc[:, 2].values.reshape(-1, 1)

#Splitting the dataset into the Training Set and Test Set
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

In [ ]:

```
#Fitting Simple Linear Regression to the Training Set

from sklearn.linear_model import LinearRegression

reg = LinearRegression()
reg.fit(X_train, y_train)
```

Out[ ]:

▾ LinearRegression ⓘ ?

LinearRegression()

In [ ]:

```
#Calculating the Coefficients

reg.coef_

#Calculating the Intercept

reg.intercept_
```

Out[ ]:

```
array([-167.20433789])
```

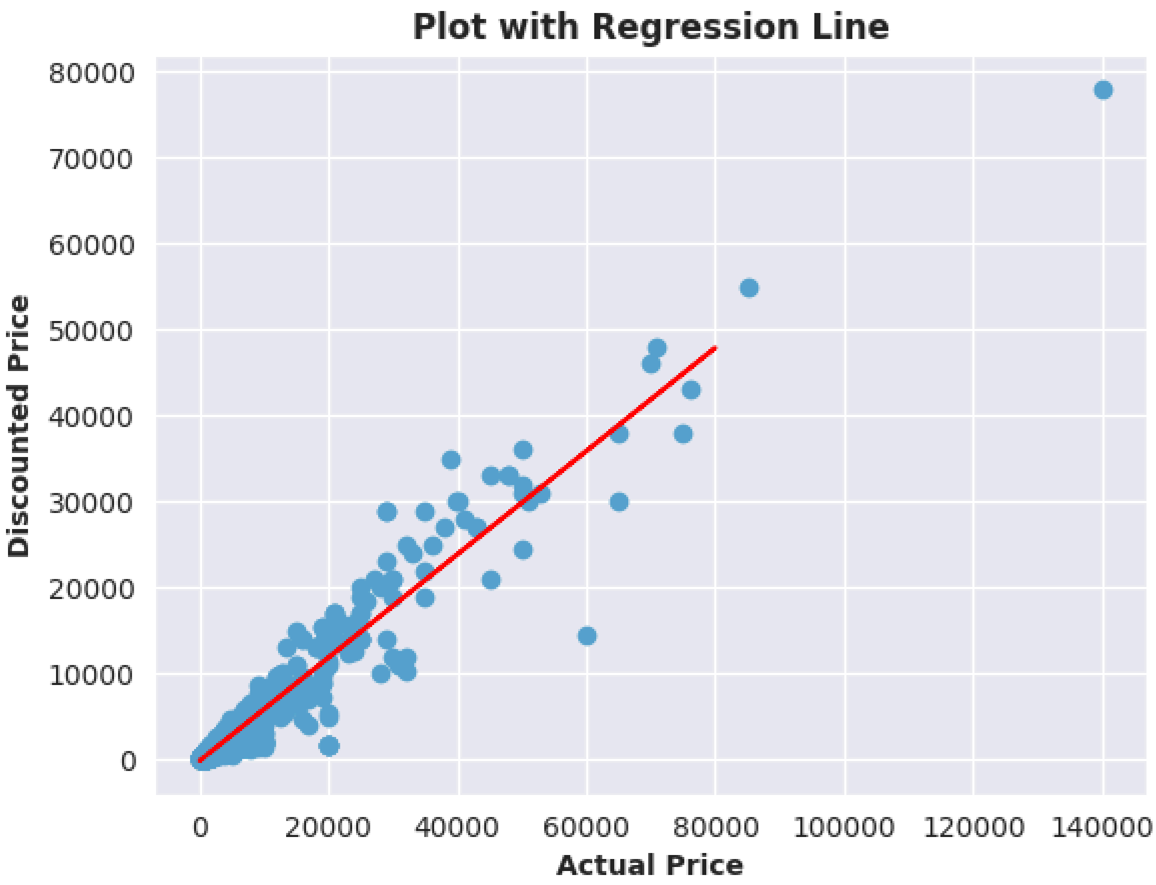
In [ ]:

```
#Calculating the R Squared Value

from sklearn.metrics import r2_score
y_pred = reg.predict(X_test)
print('R2 Score: ', r2_score(y_test, y_pred))
```

```
R2 Score:  0.9224573166916071
```

In [ ]:

```
#Scatter Plot with Regression Line

plt.scatter(X_train, y_train)
plt.plot(X_test, y_pred, color='red')
plt.xlabel('Actual Price', fontweight='bold')
plt.ylabel('Discounted Price', fontweight='bold')
plt.title('Plot with Regression Line', fontweight='bold')
plt.show()
```



In [ ]:

```
#Cross Validation Result

from sklearn.model_selection import cross_val_score, KFold

kf = KFold(n_splits=10, shuffle=True, random_state=21)

cv_results = cross_val_score(reg, X, y, cv=kf)

print('Cross Validation Results Mean: ', cv_results.mean())
```

```
Cross Validation Results Mean:  0.9159405293663456
```

In [ ]:

```
#Filling in some missing values from Rating Count Column

df1['rating_count'].fillna(df1['rating_count'].mode()[0], inplace=True)
```

In [ ]:

```
#Ridge Regression

from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split

# Assuming Xl and yl are already defined and cleaned from the previous cell
Xl = df1[['actual_price', 'rating', 'rating_count']]
yl = df1['discounted_price']

Xl_train, Xl_test, yl_train, yl_test = train_test_split(Xl, yl, random_state =
ridge = Ridge(alpha = 0.1)
ridge.fit(Xl_train, yl_train)
ridge_predict = ridge.predict(Xl_test)
print('Ridge score: ',ridge.score(Xl_test, yl_test))
```

```
Ridge score:  0.9239316602873826
```

In [ ]:

```
#Linear Regression with 3 Predictors

reg2 = LinearRegression()
reg2.fit(Xl_train, yl_train)

yl_pred = reg2.predict(Xl_test)
print('R2 Score: ', r2_score(yl_test, yl_pred))
```

```
R2 Score:  0.9239314780666216
```

In [ ]:

```
#Applying Preprocessing using Standard Scaler

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X2 = df1[['actual_price', 'rating', 'rating_count']]
y2 = df1['discounted_price']

X2 = scaler.fit_transform(X2)

X2_train, X2_test, y2_train, y2_test = train_test_split(X2,y2,random_state = 2

regss = LinearRegression()
regss.fit(X2_train, y2_train)

y2_pred = regss.predict(X2_test)
print('R2 Score: ', r2_score(y2_test, y2_pred))
```

```
R2 Score:  0.9239314780665037
```

Evaluating Simple Linear Regression Model

In [ ]:

```
import statsmodels.formula.api as smf

ols_data = df1[['discounted_price', 'actual_price']]
ols_formula = 'discounted_price ~ actual_price'
ols_model = smf.ols(formula=ols_formula, data=ols_data)
ols_results = ols_model
```

In [ ]:

```
#Importing ols function
```

```
from statsmodels.formula.api import ols

OLS = ols(formula = ols_formula, data=ols_data)
model = OLS.fit()
```

In [ ]:

```
model.summary()
```

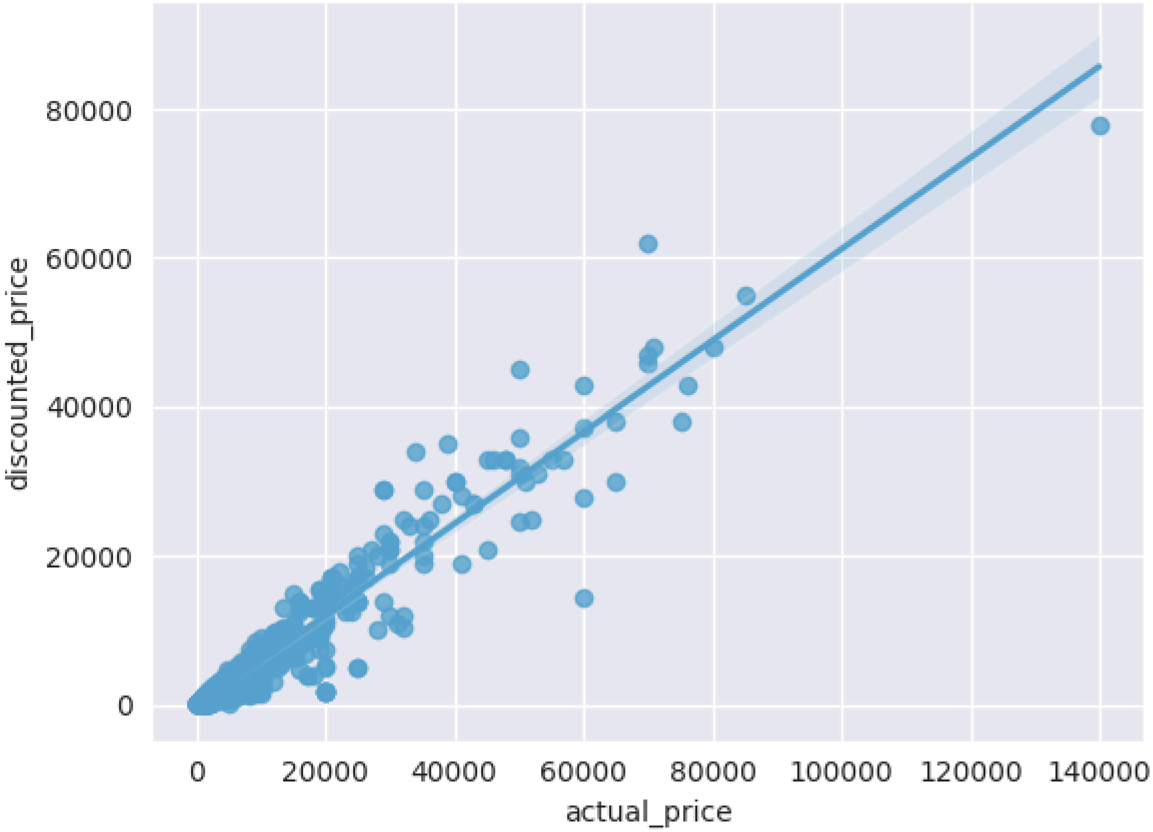
Out[ ]:

```
                              OLS Regression Results
       Dep. Variable:   discounted_price           R-squared:      0.925
              Model:               OLS       Adj. R-squared:      0.925
             Method:     Least Squares          F-statistic:  1.812e+04
               Date:  Mon, 10 Nov 2025  Prob (F-statistic):       0.00
               Time:          05:29:49       Log-Likelihood:    -13137.
   No. Observations:              1465                  AIC:  2.628e+04
       Df Residuals:              1463                  BIC:  2.629e+04
           Df Model:                 1
   Covariance Type:          nonrobust

                    coef   std err         t   P>|t|    [0.025    0.975]
   Intercept   -219.2597    55.486    -3.952   0.000  -328.099  -110.420
 actual_price     0.6142     0.005   134.600   0.000     0.605     0.623

         Omnibus:  521.405     Durbin-Watson:      1.947
 Prob(Omnibus):     0.000  Jarque-Bera (JB):  62749.178
            Skew:  -0.600          Prob(JB):       0.00
        Kurtosis:  35.040          Cond. No.   1.36e+04


Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.36e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [ ]:

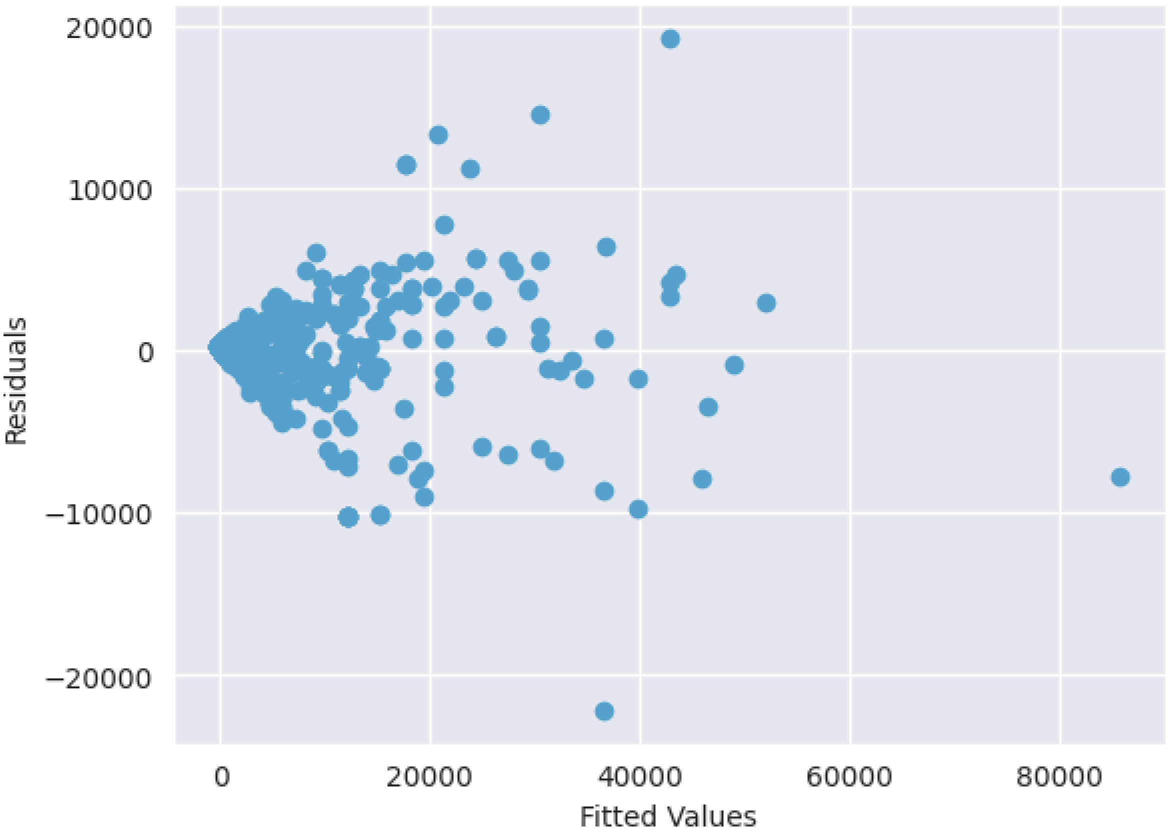
```
#Subset X Variable
```

```
X_ols = ols_data['actual_price']

#Get Prediction From Models
fitted_values = model.predict(X_ols)

#Calculate residuals
residuals = model.resid

sns.regplot(data=ols_data, x='actual_price', y='discounted_price')

plt.show()
```



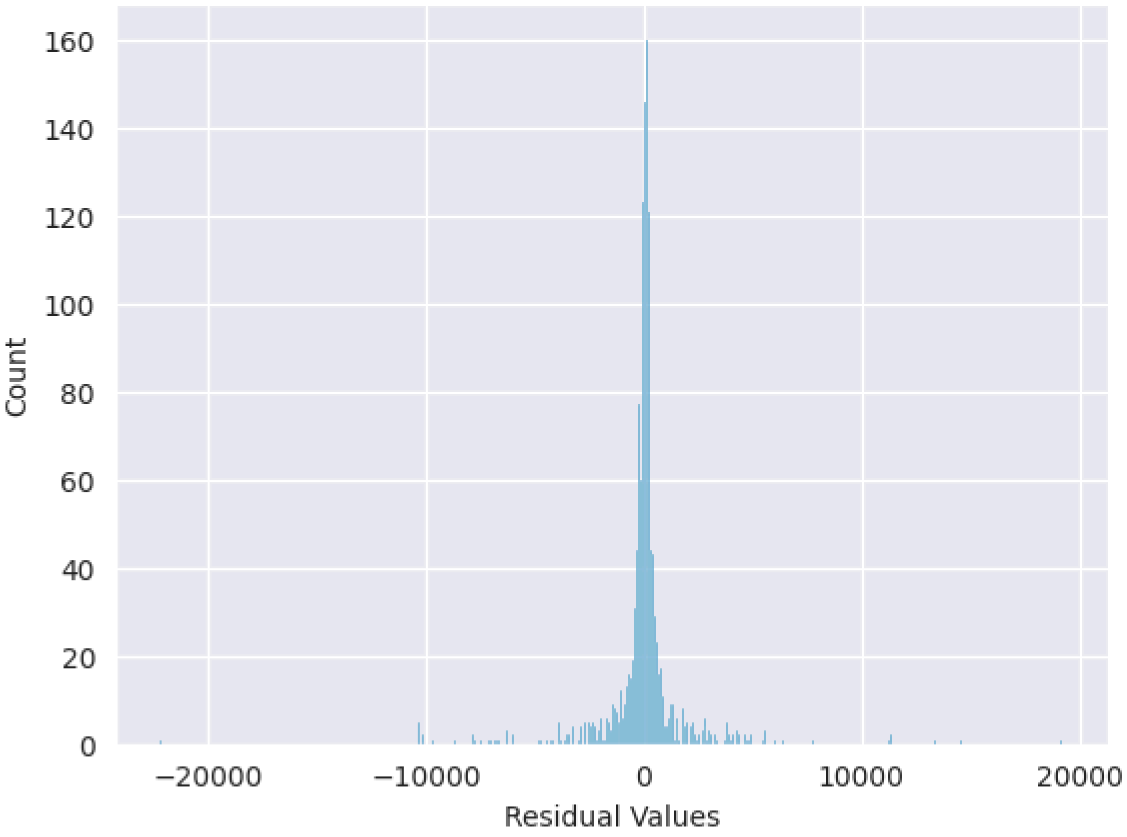
In [ ]:

```
#Checking for Homoscedasticity

plt.scatter(fitted_values, residuals)

plt.xlabel('Fitted Values')
plt.ylabel('Residuals')

plt.show()
```



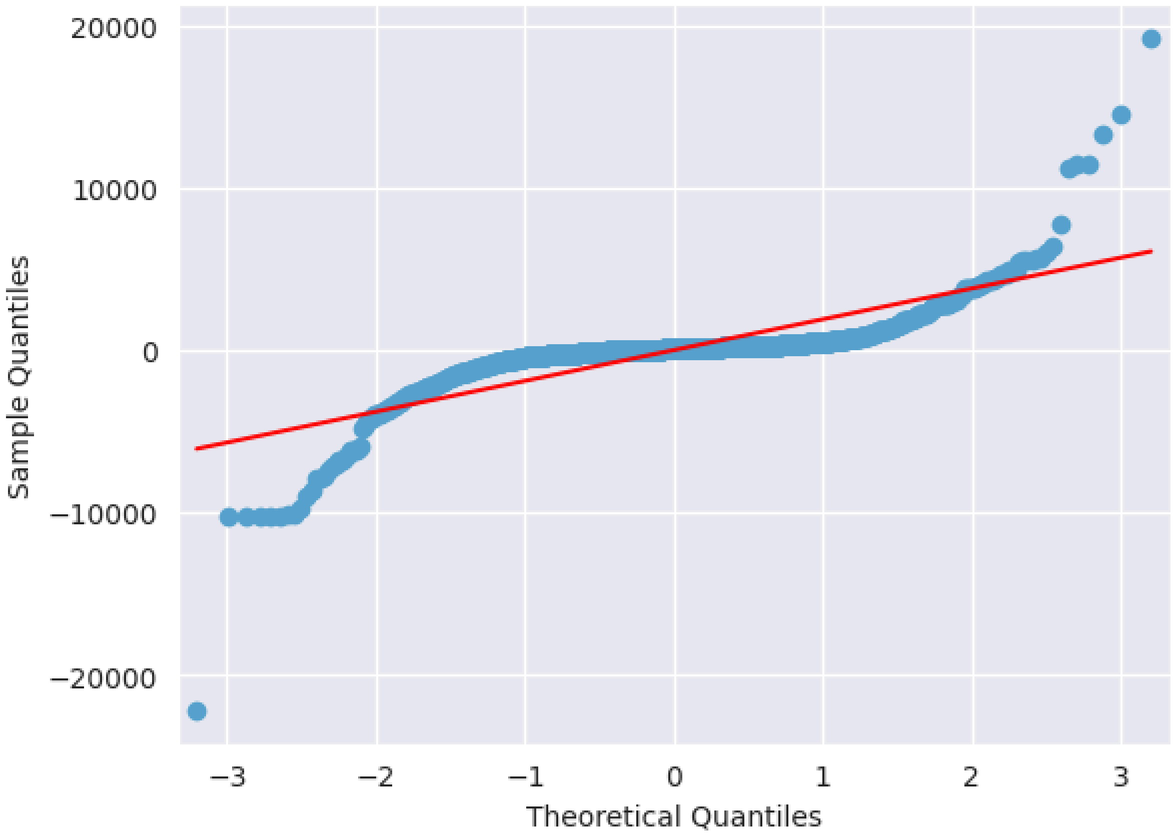
In [ ]:

```
#Checking for Normal Distribution of the Error

ax = sns.histplot(residuals)
ax.set_xlabel('Residual Values')
plt.show()
```



In [ ]:

```
#Quantile-Quantile Plot

import statsmodels.api as sm

ax = sm.qqplot(model.resid, line='s')

plt.show()
```



Conclusion:

The analysis of Amazon’s sales data provides meaningful insights into customer purchasing behavior, product performance, and overall business trends. By exploring historical patterns, we identified key factors influencing sales, such as seasonal demand peaks, category-wise performance differences, and the impact of pricing on order volume. Applying forecasting techniques helps predict future sales more accurately, enabling better inventory planning, marketing decisions, and resource allocation. The results show that data-driven forecasting can significantly improve operational efficiency and profitability. Overall, the study highlights the importance of using analytical methods and machine learning models to support strategic decision-making in e-commerce environments.

Observation

Here is a clear and well-structured **Observation** section for your **Amazon Sales Analysis / Forecasting** project:

Observations

1. **Sales show noticeable fluctuations** across months, indicating strong seasonal patterns. Certain periods—such as festival seasons or year-end—record significantly higher order volumes.

2. **Product categories perform differently**, with some categories consistently dominating sales while others show irregular demand. This highlights category-specific customer preferences.

3. **Pricing has a direct correlation with demand.** Products with competitive pricing and discounts tend to attract more customers and generate higher order volumes.

4. **High-return or low-rated products negatively affect overall sales**, suggesting that product quality and customer satisfaction strongly influence purchasing decisions.

5. **Geographical sales distribution reveals concentrated demand** in certain regions, indicating stronger market presence or better logistics in those areas.

6. **Historical data shows upward or downward trends**, helping identify growth opportunities and potential risks in future sales.

7. **Forecasting models demonstrate clear patterns**, indicating that Amazon’s sales can be predicted with reasonable accuracy using time-series or regression-based techniques.