```
# Import libs
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


# Import file
df = pd.read_csv('/content/amazon.csv')


# Take a look at the data
df.head()
```

	product_id	product_name	category	discounted_
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|...	

Next steps: View recommended plots

```
# Check for data type
df.dtypes
```

```
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
```

```
# Check for num of rows and columns
df.shape
```

```
(1465, 16)
```

```
# Replace string value and change data type
df['actual_price'] = df['actual_price'].str.replace('₹','')
df['actual_price'] = df['actual_price'].str.replace(',','').astype('float64')

df['discounted_price'] = df['discounted_price'].str.replace('₹','')
df['discounted_price'] = df['discounted_price'].str.replace(',','').astype('float64')


df['rating_count'] = df['rating_count'].str.replace(',','').astype('float64')


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

```
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
```

```
# Look at the strange row
df[df['rating'] == '|']
```

	product_id	product_name	category	discou
1279	B08L12N5H1	Eureka Forbes car Vac 100 Watts Powerful Sucti...	Home&Kitchen|Kitchen&HomeAppliances|Vacuum,Cle...	

```
df['rating'] = df['rating'].str.replace('|','4.0').astype('float64')


# Create new data frame with selected columns
df1 = df[['product_id', 'product_name', 'category', 'discounted_price', 'actual_price', 'discount_percentage', 'rating', 'rating_count']].c


# Split `category` column
cat_split = df1['category'].str.split('|', expand=True)
cat_split.isnull().sum()
```

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

```
# Rename column
cat_split = cat_split.rename(columns={0:'Main category', 1:'Sub category'})
```

```
# Add new cols to data frame and drop the old ones
df1['Main category'] = cat_split['Main category']
df1['Sub category'] = cat_split['Sub category']
df1.drop(columns ='category', inplace=True)
df1
```

	product_id	product_name	discounted_price	actual_price	discount_percentag
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	399.0	1099.0	64%
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	199.0	349.0	43%
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	199.0	1899.0	90%
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	329.0	699.0	53%
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	154.0	399.0	61%
...	...	...	...	...	.
1460	B08L7J3T31	Noir Aqua - 5pcs PP Spun Filter + 1 Spanner | ...	379.0	919.0	59%
1461	B01M6453MB	Prestige Delight PRWO Electric Rice Cooker (1 ...	2280.0	3045.0	25%
1462	B009P2LIL4	Bajaj Majesty RX10 2000 Watts Heat Convector R...	2219.0	3080.0	28%
1463	B00J5DYCCA	Havells Ventil Air DSP 230mm Exhaust Fan (Pist...	1399.0	1890.0	26%
1464	B01486F4G6	Borosil Jumbo 1000-Watt Grill Sandwich Maker (...	2863.0	3690.0	22%

1465 rows × 9 columns

Next steps: View recommended plots

```
df1['Main category'].value_counts()
```

```
Main category
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
```

```
# Fix the strings in `Main category`
df1['Main category'] = df1['Main category'].str.replace('&', ' & ')
df1['Main category'] = df1['Main category'].str.replace('OfficeProducts', 'Office Products')
df1['Main category'] = df1['Main category'].str.replace('MusicalInstruments', 'Musical Instruments')
df1['Main category'] = df1['Main category'].str.replace('HomeImprovement', 'Home Improvement')


df1['Sub category'].value_counts()
```

```
Sub category
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
```

```
# I will do the same with `Sub category`
df1['Sub category'] = df1['Sub category'].str.replace('&', ' & ')
df1['Sub category'] = df1['Sub category'].str.replace(',', ', ')
df1['Sub category'] = df1['Sub category'].str.replace('HomeAppliances', 'Home Appliances')
df1['Sub category'] = df1['Sub category'].str.replace('AirQuality', 'Air Quality')
df1['Sub category'] = df1['Sub category'].str.replace('WearableTechnology', 'Wearable Technology')
df1['Sub category'] = df1['Sub category'].str.replace('NetworkingDevices', 'Networking Devices')
df1['Sub category'] = df1['Sub category'].str.replace('OfficePaperProducts', 'Office Paper Products')
df1['Sub category'] = df1['Sub category'].str.replace('ExternalDevices', 'External Devices')
df1['Sub category'] = df1['Sub category'].str.replace('DataStorage', 'Data Storage')
df1['Sub category'] = df1['Sub category'].str.replace('HomeStorage', 'Home Storage')
df1['Sub category'] = df1['Sub category'].str.replace('HomeAudio', 'Home Audio')
df1['Sub category'] = df1['Sub category'].str.replace('GeneralPurposeBatteries', 'General Purpose Batteries')
df1['Sub category'] = df1['Sub category'].str.replace('BatteryChargers', 'Battery Chargers')
df1['Sub category'] = df1['Sub category'].str.replace('CraftMaterials', 'Craft Materials')
df1['Sub category'] = df1['Sub category'].str.replace('OfficeElectronics', 'Office Electronics')
df1['Sub category'] = df1['Sub category'].str.replace('PowerAccessories', 'Power Accessories')
df1['Sub category'] = df1['Sub category'].str.replace('CarAccessories', 'Car Accessories')
df1['Sub category'] = df1['Sub category'].str.replace('HomeMedicalSupplies', 'Home Medical Supplies')
df1['Sub category'] = df1['Sub category'].str.replace('HomeTheater', 'Home Theater')


df1.head()
```

	product_id	product_name	discounted_price	actual_price	discount_percentage	rat
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	399.0	1099.0	64%	
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	199.0	349.0	43%	
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	199.0	1899.0	90%	
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C &	329.0	699.0	53%	

Next steps: View recommended plots

```
df1.dtypes
```

```
product_id              object
product_name            object
discounted_price       float64
actual_price           float64
discount_percentage     object
rating                 float64
rating_count           float64
Main category           object
Sub category            object
dtype: object
```

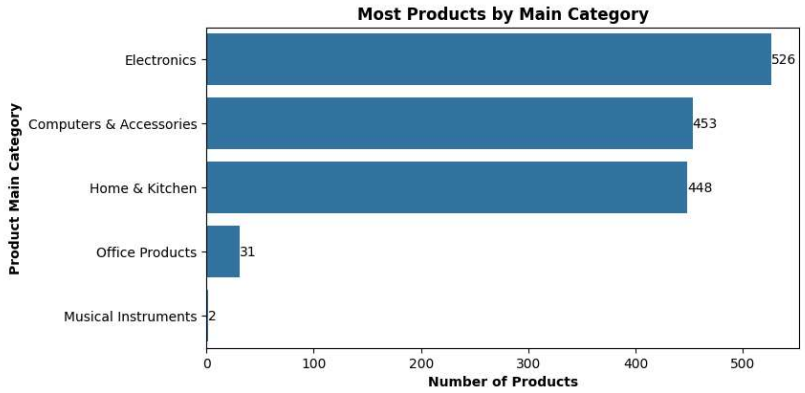
Exploratory Data Analysis

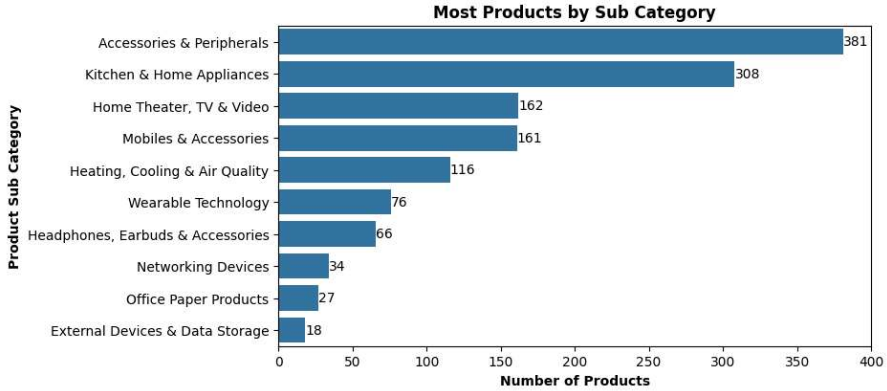
```
# Number of products by Main-Sub category
cat = df1[['Main category', 'Sub category', 'product_id']]
cat_piv = pd.pivot_table(cat, index=['Main category', 'Sub category'], aggfunc='count')
cat_piv
```

		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

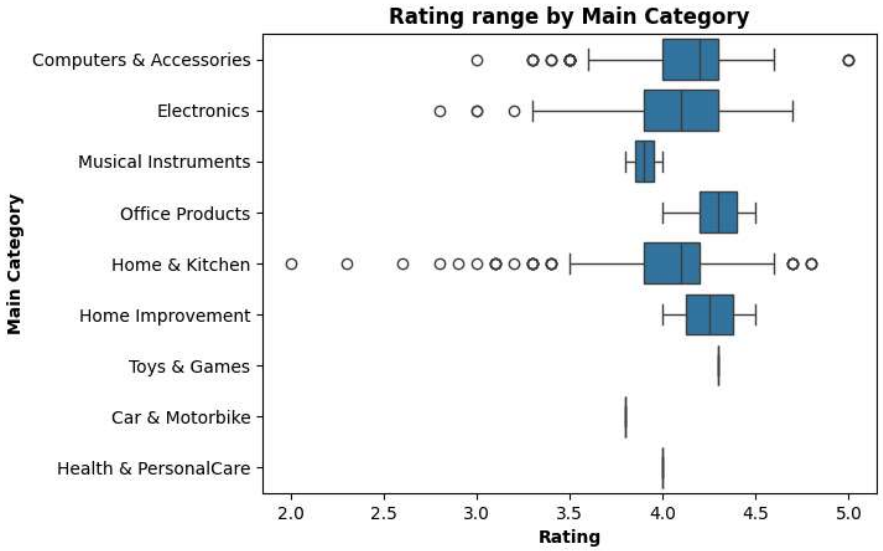
Next steps: View recommended plots

```
# Most products by Main & Sub category
main_cat_pro = df1['Main category'].value_counts().head(5).rename_axis('Main category').reset_index(name = 'count')
sub_cat_pro = df1['Sub category'].value_counts().head(10).rename_axis('Sub category').reset_index(name = 'count')

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

sns.barplot(ax=ax[0], data = main_cat_pro, x='count', y='Main category')
sns.barplot(ax=ax[1], data = sub_cat_pro, x='count', y='Sub category')

plt.subplots_adjust(hspace = 0.3)

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

ax[1].set_xlabel('Number of Products', 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()
```

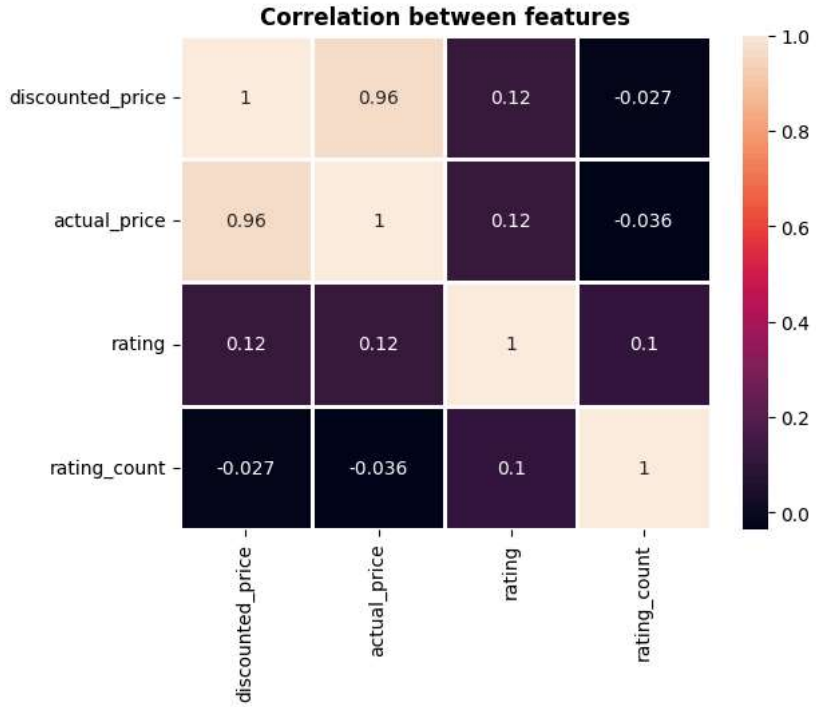




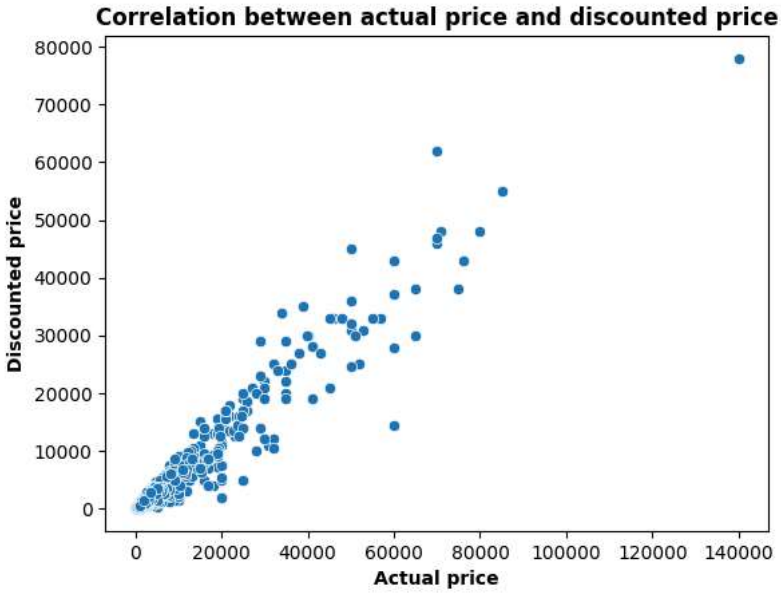
```
sns.boxplot(data = df1, x='rating', y='Main category')
plt.title('Rating range by Main Category', fontweight = 'bold')
plt.xlabel('Rating', fontweight = 'bold')
plt.ylabel('Main Category', fontweight = 'bold')
plt.show()
```



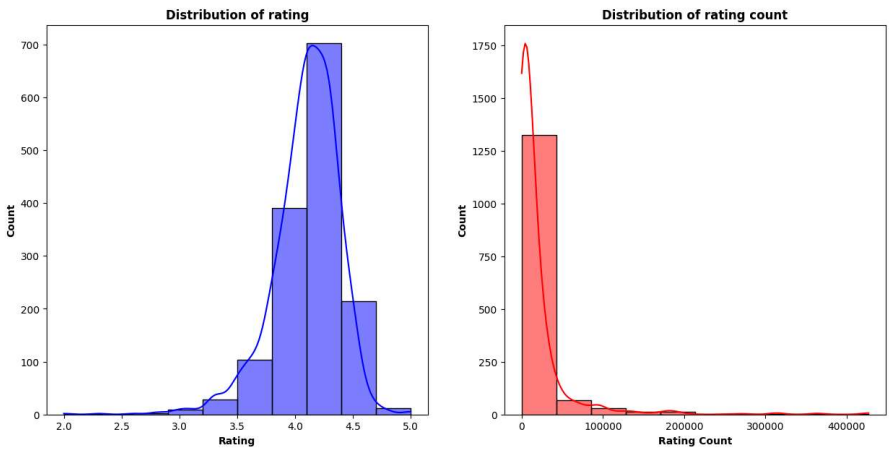

```
#Correlation between Numerical variables
corr = df1.select_dtypes('number').corr()
sns.heatmap(data = corr, annot = True, linewidths = 1)
plt.title('Correlation between features', fontweight = 'bold')
plt.show()
```




```
sns.scatterplot(data = df1, x='actual_price', y='discounted_price')
plt.title('Correlation between actual price and discounted price', fontweight = 'bold')
plt.xlabel('Actual price', fontweight = 'bold')
plt.ylabel('Discounted price', fontweight = 'bold')
plt.show()
```




```
fig, ax = plt.subplots(1,2, figsize = (15,7))


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

ax[0].set_title('Distribution of rating', fontweight='bold')
ax[1].set_title('Distribution of rating count', fontweight='bold')

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

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


plt.show()
```



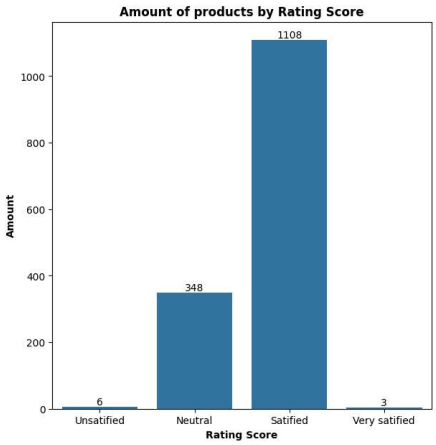
The rating range of most products is around 3.75 and 4.38. The distribution of rating is left_skewed with no products rated lower than 2.0. The range of rating_count is really widespread fall from 0 to over 40000. Most products have the amount of rating between 0-5000. The rating_count distribution is strictly right_skewed.

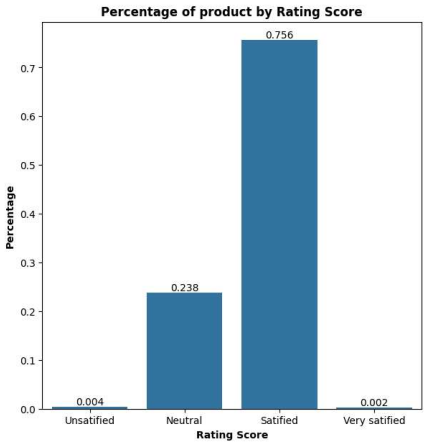
```
# Create new category column `rating_score`
rating_score = []
for i in df1['rating']:
    if i < 2.0: rating_score.append('Very unsatified')
    elif i < 3.0: rating_score.append('Unsatified')
    elif i < 4.0: rating_score.append('Neutral')
    elif i < 5.0: rating_score.append('Satified')
    elif i == 5.0: rating_score.append('Very satified')


df1['rating_score'] = rating_score
df1['rating_score'] = df1['rating_score'].astype('category')
# Reorder cateories
df1['rating_score'] = df1['rating_score'].cat.reorder_categories(['Unsatified', 'Neutral', 'Satified','Very satified'], ordered=True)
df1.head()
```

	product_id	product_name	discounted_price	actual_price	discount_percentage	rat
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	399.0	1099.0	64%	
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	199.0	349.0	43%	
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	199.0	1899.0	90%	
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	329.0	699.0	53%	
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	154.0	399.0	61%	

Next steps: View recommended plots

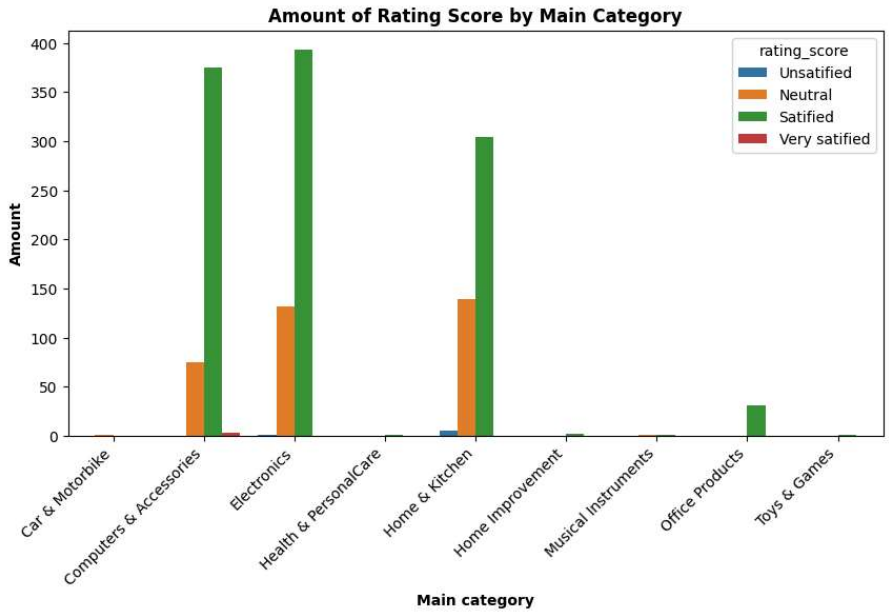
```
rating_score_pct = df1['rating_score'].value_counts(normalize=True).rename_axis('score').reset_index(name='score_pct')
rating_score_pct['score_pct'] = rating_score_pct['score_pct'].round(3)
fig, ax = plt.subplots(1,2, figsize = (15,7))
sns.countplot(ax=ax[0], data=df1, x='rating_score')
sns.barplot(ax=ax[1],data = rating_score_pct, x='score', y='score_pct')

ax[0].set_title('Amount of products by Rating Score', fontweight = 'bold')
ax[1].set_title('Percentage of product by Rating Score', fontweight = 'bold')

ax[0].set_xlabel('Rating Score', fontweight = 'bold')
ax[1].set_xlabel('Rating Score', fontweight = 'bold')

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

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



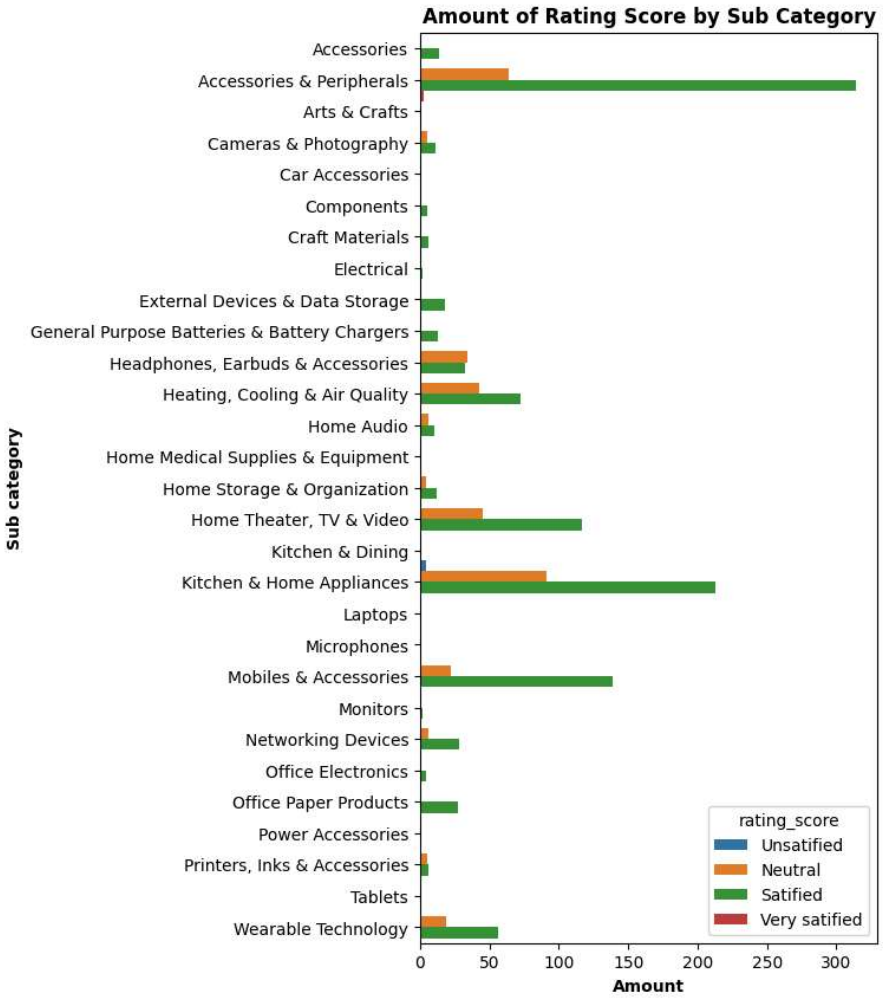



```
#Based on the above, over 75% of products listed in the marketplace are rated as 'Satified' by customers. Whereas, only 6 products as well
```

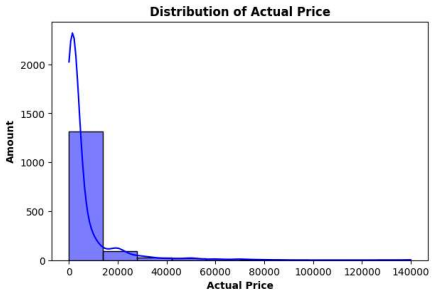
```
main_rating_score = df1.groupby(['Main category', 'rating_score']).agg('count').iloc[:, -1].rename_axis().reset_index(name='count')
fig, ax = plt.subplots(figsize=(10,5))
ax = sns.barplot(data = main_rating_score, x='Main category', y='count', hue = 'rating_score')
ax.set_title('Amount of Rating Score by Main Category', fontweight = 'bold')
ax.set_xlabel('Main category', fontweight = 'bold')
ax.set_ylabel('Amount', fontweight = 'bold')
plt.xticks(rotation=45, ha='right')
plt.show()
```

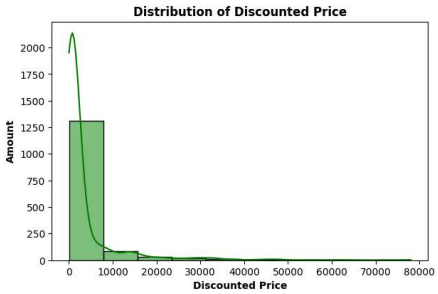


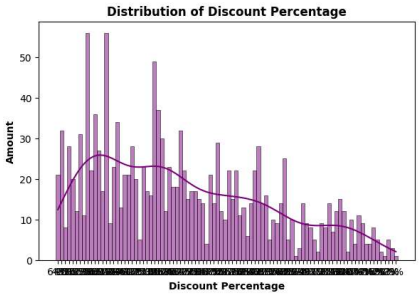
```
sub_rating_score = df1.groupby(['Sub category', 'rating_score']).agg('count').iloc[:, -1].rename_axis().reset_index(name='count')
fig, ax = plt.subplots(figsize=(5,10))
ax = sns.barplot(data = sub_rating_score, x='count', y='Sub category', hue='rating_score', width = 1.4)
ax.set_title('Amount of Rating Score by Sub Category', fontweight = 'bold')
ax.set_xlabel('Amount', fontweight = 'bold')
ax.set_ylabel('Sub category', fontweight = 'bold')
plt.show()
```



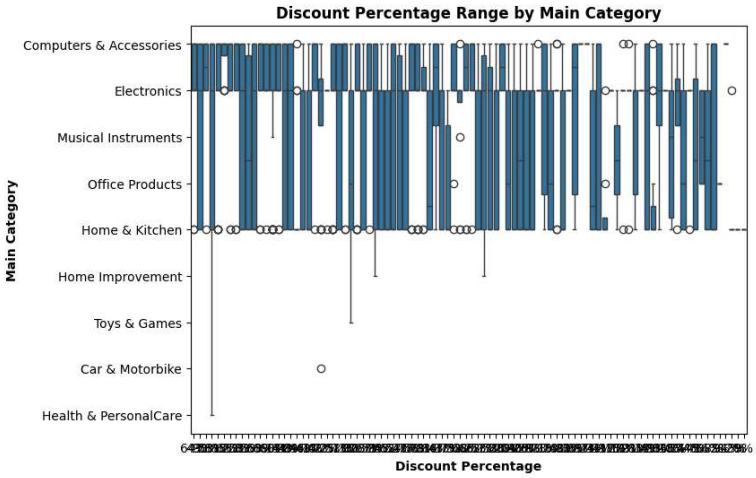

```
fig, ax = plt.subplots(2,2, figsize = (15,10))
plt.subplots_adjust(hspace = 0.3)

sns.histplot(ax=ax[0,0], data = df1, x='actual_price', bins = 10, kde = True, color = 'blue')
sns.histplot(ax=ax[0,1], data = df1, x='discounted_price', bins = 10, kde = True, color = 'green')
sns.histplot(ax=ax[1,0], data = df1, x='discount_percentage', bins = 10, kde = True, color = 'purple')

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

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

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

ax.flat[-1].set_visible(False)
plt.show()
```

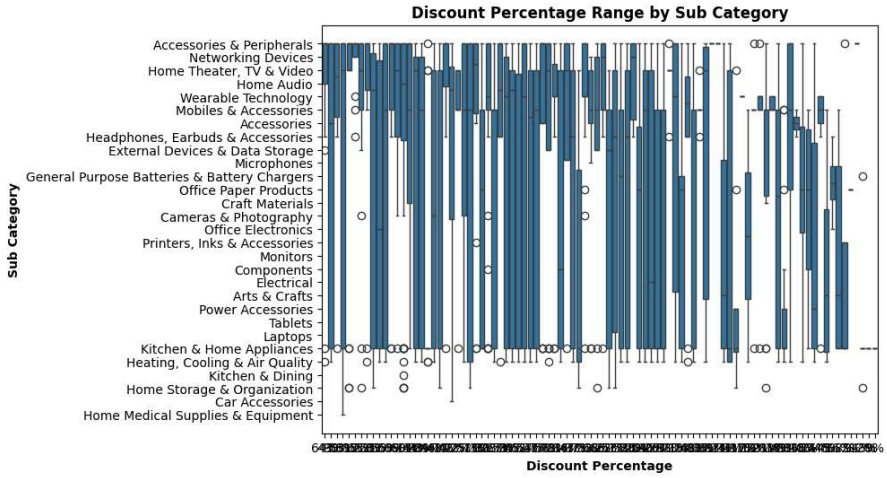







```
# In the above graph both Actual price and Discounted price have a widespread range of distribution, specifically 0-140000 and 0-78000 resp
# The scale of discount percentage is much more balanced with most of the products having discounts at around 50%-60%.


fig, ax = plt.subplots(2,1, figsize = (8,13))
sns.boxplot(ax=ax[0], data = df1, x='discount_percentage', y='Main category')
sns.boxplot(ax=ax[1], data = df1, x='discount_percentage', y='Sub category')

ax[0].set_title('Discount Percentage Range by Main Category', fontweight = 'bold')
ax[1].set_title('Discount Percentage Range by Sub Category', fontweight = 'bold')

ax[0].set_xlabel('Discount Percentage', fontweight = 'bold')
ax[1].set_xlabel('Discount Percentage', fontweight = 'bold')

ax[0].set_ylabel('Main Category', fontweight = 'bold')
ax[1].set_ylabel('Sub Category', fontweight = 'bold')


plt.show()
```






```
fig, ax = plt.subplots(2,1, figsize = (10,13))
sns.scatterplot(ax = ax[0], data = df1, x = 'actual_price', y = 'rating_score', color = 'red')
sns.scatterplot(ax = ax[1], data = df1, x = 'discounted_price', y = 'rating_score', color ='green')

ax[0].set_title('Actual Price Range by Rating Score', fontweight = 'bold')
ax[1].set_title('Discounted Price Range by Rating Score', fontweight = 'bold')

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

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

plt.show()
```


```
# # Based on the above graph, There is a remarkable change between actual_price and discounted_price in satified-score products.
# Most products' actual_price range around 0 - 30000. About discounted_price, most products' price falls under 20000 after discounts.


#PRICE PREDICTION-Feature Selection
# Check for null value
df1.isnull().sum()
```

```
product_id            0
product_name          0
discounted_price      0
actual_price          0
discount_percentage   0
rating                0
rating_count          2
Main category         0
Sub category          0
rating_score          0
dtype: int64
```

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


X1 = df1[['rating', 'rating_count', 'actual_price']]
y1 = df1['discounted_price']


# Apply model
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size = 0.2)
lr_model = LinearRegression()
lr_model.fit(X1_train, y1_train)

y1_pred = lr_model.predict(X1_test)
```

Mean of Residuals

```
import numpy as np

residuals = y1_test - y1_pred
mean_residuals = np.mean(residuals)
print("Mean of Residuals:", mean_residuals)
```

```
Mean of Residuals: 33.46588216356606
```

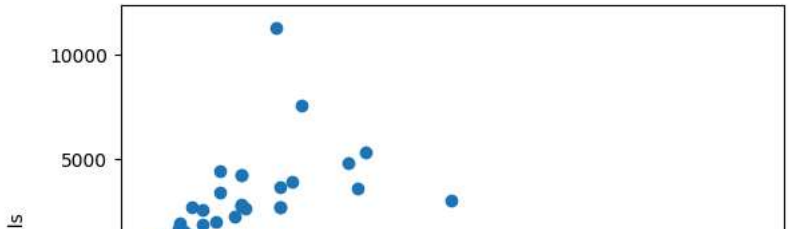
Check of heteroscedicity

```
import statsmodels.api as sm

# Add constant to the predictor dataset
X1_test_with_constant = sm.add_constant(X1_test)

# Perform the Breusch-Pagan test
_, pvalue, _, _ = het_breuschpagan(residuals, X1_test_with_constant)
print("P-value of Breusch-Pagan test:", pvalue)
```

```
P-value of Breusch-Pagan test: 1.0861936888025216e-20
```

Linearity of variables

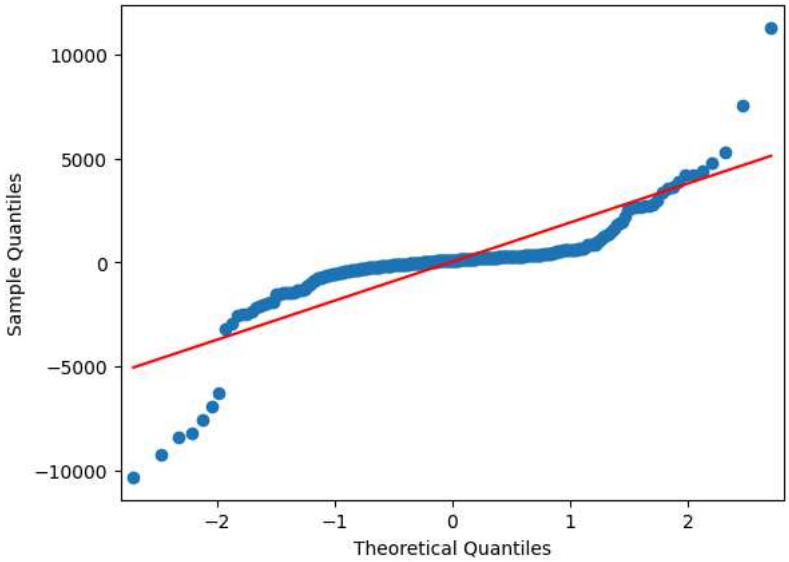
```
import matplotlib.pyplot as plt

plt.scatter(y1_pred, residuals)
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.axhline(y=0, color='red', linestyle='--')
plt.show()
```



Normality of Error Terms

```
from scipy.stats import shapiro
from statsmodels.graphics.gofplots import qqplot

# Statistical test
stat, p = shapiro(residuals)
print("Shapiro-Wilk Test p-value:", p)

# Q-Q plot
qqplot(residuals, line='s')
plt.show()
```

```
Shapiro-Wilk Test p-value: 7.530841699030664e-23
```




```
# Calculating Intercept, Coefficient, R Squared Value
from sklearn.metrics import r2_score

print('Linear Regression Intercept: ',lr_model.intercept_)
print('Linear Regression Coefficient: ',lr_model.coef_)
print('R2 Score: ', r2_score(y1_test, y1_pred))
```

```
Linear Regression Intercept:  -625.4501382508738
Linear Regression Coefficient:  [8.50630802e+01 1.38811876e-03 6.18958979e-01]
R2 Score:  0.947907686772606
```