Amazon Best Sellers Store Analysis

by Joshua Nguyen

Abstract Amazon is one of the world's most well known e-commerce website with a user subscription surpassing 200 million people as of early 2022. Ergo, without a doubt, it is a major achievement when an Amazon store has a product that makes Amazon's Best Sellers List which showcases the most popular products sold on Amazon.

Project Overview

1. This project explores different statistical metrics to analyze and assay *Amazon's* Best Sellers. Additionally, this project will display the results that is easily read and interpreted by the non-technical population who may not easily understand statistical vernacular.
2. This project will explore the relationship between *Amazons Best Sellers*'s rating, rating count, price and order ranking for most popular items sold on Amazon's site.
3. This project aims to provide conclusions that will help *Amazon's* stores maintain high popularity and increase sales.
4. This project strictly utilizes **Python** for data extraction, transformation and data analysis.

Objectives

The overall goal of this project is to build a business report that showcases sale metrics along with graphical representation information about **Amazon Best Sellers** so that stores on Amazon can improve their marketing campaign and improve profits.

Questions of Interest

1. **Group 1 - Individual Logistics**
   (a) What are the top ten highest rated products sold on Amazon's Best Sellers?
   (b) What are the top ten products with the most ratings count?
   (c) What are the top ten most expensive items that are sold on the Amazon's Best Sellers page?
2. **Group 2 - Collective Logistics**
   (a) Which category type has the highest average price?
   (b) Which category type has the highest average ratings? highest ratings count?
3. **Group 3 - Descriptive Statistics**
   (a) How does the amount of ratings affect the price of the item being sold?
   (b) Does Amazon intentionally rank higher priced products in the top ten orderings?
   (c) Describe the relationship between ratings and ratings count for items in the Electronics category.

Extraction, Transformation and Preparation

Extracting The Data

This project uses web-scraping to extract the data from the Amazon's Best Sellers Page.

Import Libraries

```
# Import libraries
import re
from bs4 import BeautifulSoup
import requests
import time
from datetime import datetime
from urllib.request import urlopen
```

Data of Interest

This *getData* function retrieves the information we are seeking. The function takes the input *url* of the Amazon Best Sellers and the page number of the page. Here, the function extracts valuable information like category, product and pricing, ect.

```
headers = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
                (KHTML,like Gecko) Chrome/98.0.4758.102 Safari/537.36'}

product_list = []

def getData(url, pageNo):

    url = eval("f'" + url +"'")

    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.text, 'html.parser')
    products = soup.find_all('div', {'id':'gridItemRoot'})

    for product in products:
        items = {
        'category' : soup.find('h1', attrs={'class':'a-size-large a-spacing-medium a-text
            -bold'}).text,
        'orders' : product.find('span', attrs={'class':'zg-bdg-text'}).text,
        'name' : [x.get_text() for x in product.find_all('div', attrs={'class':'_p13n-zg
            -list-grid-desktop_truncationStyles_p13n-sc-css-line-clamp-3__g3dy1'})]
            if len([x.get_text() for x in product.find_all('div', attrs={'class':'_p13n-zg
            -list-grid-desktop_truncationStyles_p13n-sc-css-line-clamp-3__g3dy1'})])
            > 0 else [x.get_text() for x in product.find_all('div', attrs={'class':'_p13n-zg
            -list-grid-desktop_truncationStyles_p13n-sc-css-line-clamp-1__1Fn1y'})],
        'prices' : [x.get_text() for x in product.find_all('span', attrs={'class':"_p13n-zg
            -list-grid-desktop_price_p13n-sc-price__3mJ9Z"})],
        'ratings' : [x.get_text() for x in product.find_all('span', attrs={'class':
            'a-icon-alt'})],
        'rating_count' : [x.get_text() for x in product.find_all('span', attrs={'class':
            'a-size-small'})],
        }
        product_list.append(items)

    return
```

Retrieving Amazon Best Seller Links

The Amazon Best Sellers Page has a list of categories displayed in the left hand sidebar. The next function will paginate through that sidebar to retrieve the *href* of each category.


```python
def getLinks(url):
    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.text, 'html.parser')

    links = soup.find_all('div', {'role': 'treeitem', 'class': '_p13n-zg-nav-tree-all_style
        _zg-browse-item__1rdKf _p13n-zg-nav-tree-all_style_zg-browse-height-large__1z5B8'})

    mylink = []
    for link in links:
        a = link.findAll('a')
        for x in a:
            href_link = x.get('href')
            newlink = 'https://www.amazon.com'+href_link
            mylink.append(newlink)

    return(mylink)
```

Retrieving URL

The *getData* function takes the input of a url not href. The next function outputs a list of url's that will be passed to the *getData* function for data extraction.

```
def getURL():

    URL = 'https://www.amazon.com/Best-Sellers-Electronics/zgbs/electronics/ref=zg_bs_nav_0'

    response = requests.get(URL, headers=headers)
    soup = BeautifulSoup(response.text, 'html.parser')

    links = soup.find_all('div', {'role': 'treeitem', 'class': '_p13n-zg-nav-tree-all_style
        _zg-browse-item__1rdKf _p13n-zg-nav-tree-all_style_zg-browse-height-large__1z5B8'})

    mylinks = []
    mylinks.append('https://www.amazon.com/Best-Sellers-Electronics/zgbs/electronics/ref
        =zg_bs_pg_{pageNo}?_encoding=UTF8&pg={pageNo}')

    for link in links:
        a = link.findAll('a')
        for x in a:
            href_link = x.get('href')
            newlink = 'https://www.amazon.com/'+href_link
            newlink2 = 'https://www.amazon.com/'+href_link

            newlink = str(newlink).replace('_nav_electronics_1',
                "_pg_{pageNo}?_encoding=UTF8&pg={pageNo}")
            mylinks.append(newlink)

            newlink3 = getLinks(newlink2) # list of sub-links
            for link2 in newlink3:
                newlink_prime = str(link2).replace('_nav_electronics_2',
                    "_pg_{pageNo}?_encoding=UTF8&pg={pageNo}")
                mylinks.append(newlink_prime)

    return(mylinks)
```

Producing the DataFrame

```
import pandas as pd

allLinks = getURL()

pageNo = 2
for URL in allLinks:
    for x in range (0,pageNo+1):
        getData(URL, x)

df = pd.DataFrame(product_list,
    columns=['category', 'orders', 'name', 'prices', 'ratings', 'rating_count'])

df.to_csv('amazon_best_sellers.csv', index=None, header=True) # optional
```

Data Transformation And Cleaning

```
import numpy as np

df.head()
```

	category	orders	name	prices	ratings	rating_count
0	Best Sellers in Electronics	#1	['Fire TV Stick Lite with Alexa Voice Remote L...	['$29.99']	['4.7 out of 5 stars']	['221,078']
1	Best Sellers in Electronics	#2	['Fire TV Stick 4K streaming device with lates...	['$39.99']	['4.8 out of 5 stars']	['61,896']
2	Best Sellers in Electronics	#3	['Fire TV Stick with Alexa Voice Remote (inclu...	['$39.99']	['4.7 out of 5 stars']	['168,919']
3	Best Sellers in Electronics	#4	['Echo Dot (4th Gen, 2020 release) | Smart spe...	['$49.99']	['4.7 out of 5 stars']	['413,401']
4	Best Sellers in Electronics	#5	['Apple AirTag']	['$29.00']	['4.7 out of 5 stars']	['42,541']

```
# Deleting brackets
df['prices'] = df['prices'].str.replace("[", "").str.replace("]", "")
df['name'] = df['name'].str.replace("[", "").str.replace("]", "")
df['ratings'] = df['ratings'].str.replace("[", "").str.replace("]", "")
df['rating_count'] = df['rating_count'].str.replace("[", "").str.replace("]", "")

# Deleting quotation marks, commas and converting object columns to correct variable type

# orders column
df['orders'] = df['orders'].str.replace("#", "")
df['orders'] = pd.to_numeric(df['orders'], errors='coerce').fillna(0,downcast='infer')

# name column
df['name'] = df['name'].str.replace("'", "")

# Prices column
df['prices'] = df['prices'].str.replace("'", "").str.replace("$", "")
df['prices'] = pd.to_numeric(df['prices'], errors='coerce').fillna(0,downcast='infer')

# rating_count column
df['rating_count'] = df['rating_count'].str.replace("'", "").str.replace(",", "")
df['rating_count'] = pd.to_numeric(df['rating_count'], errors='coerce').fillna(0,
    downcast='infer')

# ratings column
df['ratings'] = df['ratings'].str.replace("'", "")
df['ratings'] = pd.to_numeric(df['ratings'].str.split(' out of 5 stars').str[0])

df.head()
```

	category	orders	name	prices	ratings	rating_count
0	Best Sellers in Electronics	1	Fire TV Stick Lite with Alexa Voice Remote Lit...	29.99	4.7	221078
1	Best Sellers in Electronics	2	Fire TV Stick 4K streaming device with latest ...	39.99	4.8	61896
2	Best Sellers in Electronics	3	Fire TV Stick with Alexa Voice Remote (include...	39.99	4.7	168919
3	Best Sellers in Electronics	4	Echo Dot (4th Gen, 2020 release) | Smart speak...	49.99	4.7	413401
4	Best Sellers in Electronics	5	Apple AirTag	29.00	4.7	42541

```
# Dropping rows with no values

df.replace(str(0), np.nan, inplace=True)
df.replace(0, np.nan, inplace=True)
df = df.dropna()
```

Exploratory Data Analysis (EDA)

```
# Import libraries
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

df.describe()
```

	orders	prices	ratings	rating_count
count	12697.000000	12697.000000	12697.000000	12697.000000
mean	16.857683	76.962561	4.453446	15074.706939
std	12.368590	113.168886	0.328111	41033.447771
min	1.000000	0.990000	1.000000	1.000000
25%	8.000000	15.990000	4.300000	441.000000
50%	16.000000	31.960000	4.500000	2541.000000
75%	24.000000	89.990000	4.700000	10540.000000
max	80.000000	999.990000	5.000000	875687.000000

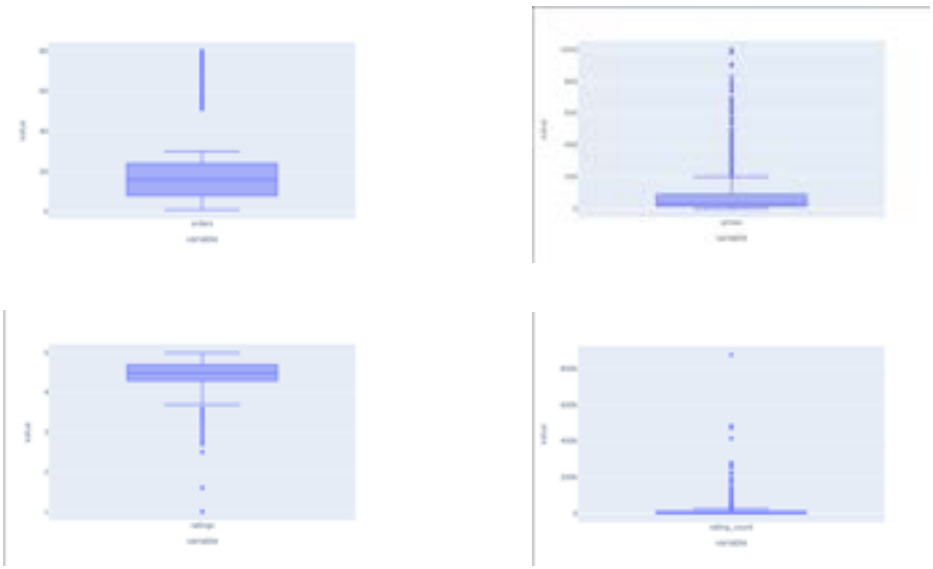
Figure: Summary Statistics of Data

Interestingly, the highest sold product on Amazon's Best Sellers is $999, and the cheapest item costs only $1 USD. The average pricing is $76 USD. Furthermore, the average rating for Amazon's best sellers is 4.45 stars. Unfortunately, even lowly rated items can make Amazon's Best Sellers list as items with one star ranking have made the accolades. However, it is important to see that some items on the list have as low as only one rating count with other items achieving over 875,000 ratings. The average rating count for the said items is 15,000 ratings. The standard deviation for the rating counts is 41,033 ratings. Therefore, there exists large variance in the data.

```
# Correlation Map

plt.figure(figsize=(12,10))
corr_plot = sns.heatmap(df.corr(), annot= True, cmap='PuBu')
corr_plot
```

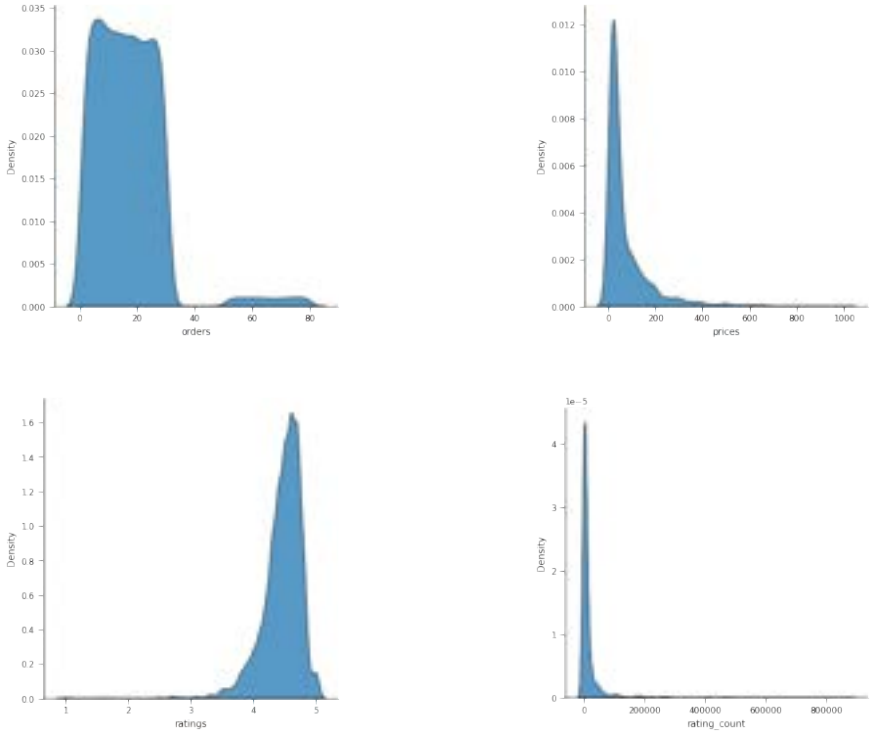
Unfortunately, there doesn't exist many strong correlation within the data. The strongest correlation is between ratings and rating counts with a positive correlation score of 0.17. But let's continue with the analysis.


```
# Viewing Boxplot Distribution of Numerical Variables
for i in df.select_dtypes(exclude=['object']).columns:
    px.box(df, y = [i], width=700).show()
```

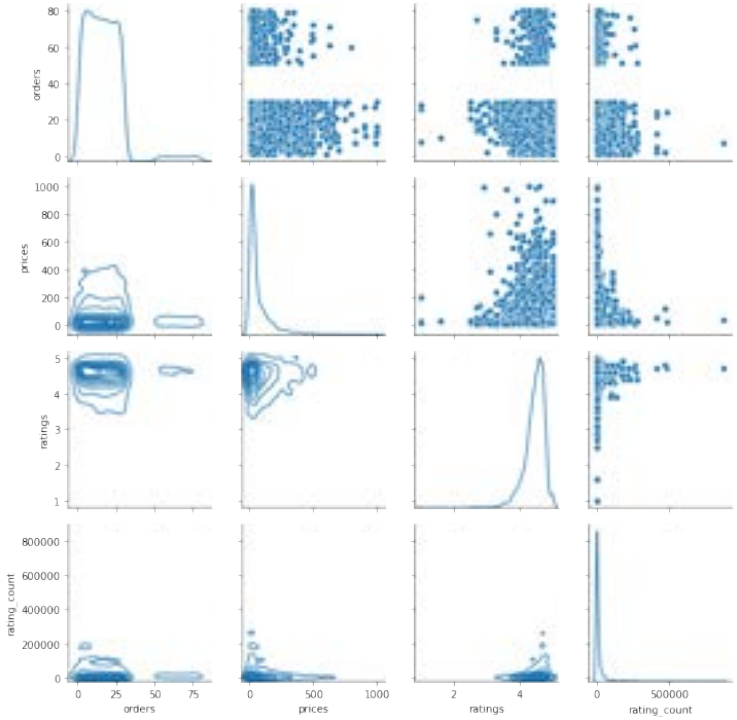


```
# Histogram/Kernel Density Estimation Distribution of Numerical Variables
for i in df.select_dtypes(exclude=['object']).columns:
    sns.displot(data=df, x= df[i], kind = "kde", multiple = "stack")
```

When viewing the distributions of the numerical variables, we see that ratings count and prices are right skewed. However, ratings is left skewed. The orders grouping is almost normally distributed when looking at the boxplot graph; however, there exists obvious outliers from both the boxplot and the kernel density estimation.

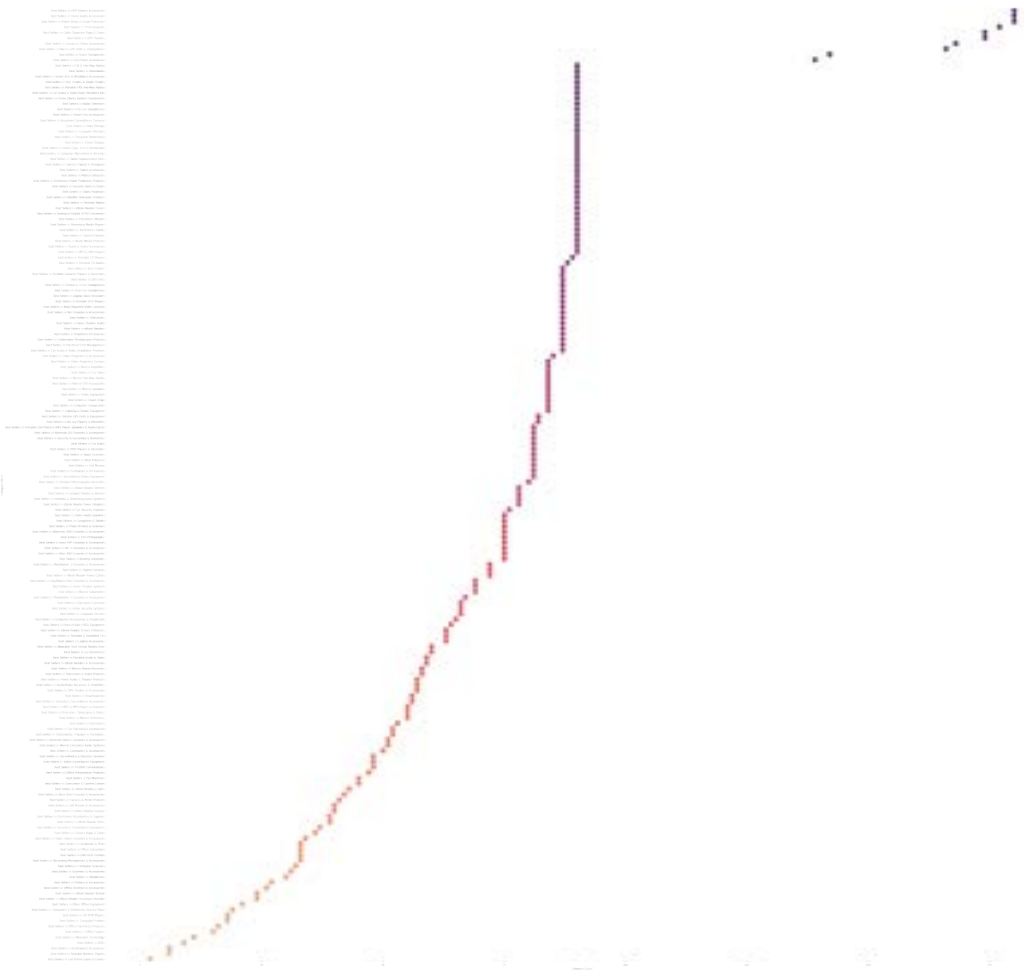



```
# Viewing Nurmerical Distributions
g = sns.PairGrid(df, diag_sharey=False)
g.map_upper(sns.scatterplot)
g.map_diag(sns.kdeplot)
g.map_lower(sns.kdeplot)
```



To view the categorical distribution, we will need to group the data by each category. Afterwards, we see can see that the category with the most items is **Best Sellers in GPS System Accessories** which has 180 items.

```
# Viewing Categorical Distrubution
cat_count = df['category'].value_counts()
cat_name = df['category'].value_counts().index

# Creating temporary dataframe to store new table values
temp_df = {'categories': cat_name, 'category_count': cat_count}
df2 = pd.DataFrame(data=temp_df)

# Making Grid
ax = sns.PairGrid(df2, height = 50, x_vars=["category_count"], y_vars=["categories"],
    aspect=1)

# Ploting values
ax.map(sns.stripplot, size = 20, orient = "h", jitter = False, palette = "flare_r",
    linewidth = 1, edgecolor = 'w')
ax.set(xlabel = "Category Count", ylabel= "Category Name")
sns.despine(left=True, bottom=True)
```



Group 1 Questions

Finding The Most Highly Rated Item on Amazon's Best Sellers

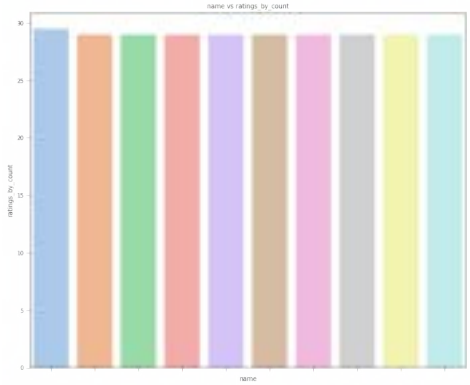
```
# Top Ten Rated Products

#top_ten_rated = df.nlargest(10, 'ratings', )
top_ten_rated = df.sort_values(by='ratings', ascending=False)
top_ten_rated.drop_duplicates(subset='name', keep='first').head(10)

# Top Ratings Does Not Provide Much Insight
# Will Need Pandas Lambda Function to Calculate Weighted Count

M = df['rating_count'].mean()
Q = -M/np.log(1/2)

df_prime = df.copy()
df_prime = df_prime.assign(ratings_by_count  = lambda x: (5*x['ratings']+
    5*(1-np.exp(-x['rating_count']/Q))))

top_ten_rated_count = df_prime.sort_values(by='ratings_by_count', ascending=False)
temp_data = top_ten_rated_count.drop_duplicates(subset='ratings_by_count',
    keep='first').head(10)
temp_data
```

	category	orders	name	prices	ratings	rating_count	ratings_by_count
13450	Best Sellers in Nintendo Switch Consoles & Acc...	11.0	SanDisk 256GB microSDXC-Card, Licensed for Nin...	41.79	4.9	177512.0	29.498574
6181	Best Sellers in Earbud & In-Ear Headphones	2.0	Apple AirPods (2nd Generation)	119.98	4.8	472670.0	29.000000
185	Best Sellers in Camera & Photo Accessories	6.0	SanDisk 256GB Extreme microSDXC UHS-I Memory C...	32.98	4.8	266496.0	28.999976
93	Best Sellers in Electronics	14.0	Roku Express	HD Streaming Media Player with ...	29.00	4.8	145212.0	28.993702
79	Best Sellers in Electronics Accessories & Supp...	20.0	SanDisk 128GB Ultra MicroSDXC UHS-I Memory Car...	16.99	4.8	126653.0	28.985215
156	Best Sellers in Camera & Photo Accessories	7.0	SanDisk 128GB Extreme PRO SDXC UHS-I Card - C1...	29.79	4.8	121833.0	28.981547
3902	Best Sellers in Cell Phone Accessories	3.0	Apple 20W USB-C Power Adapter	17.98	4.8	107985.0	28.965119
13489	Best Sellers in Nintendo Switch Consoles & Acc...	20.0	Nintendo Switch with Neon Blue and Neon Red Jo...	318.98	4.8	105828.0	28.961482
6911	Best Sellers in Home Audio Speakers	12.0	JBL Flip 4, Black - Waterproof, Portable & Dur...	93.95	4.8	99981.0	28.949600
13477	Best Sellers in Nintendo Switch Consoles & Acc...	8.0	amFilm Tempered Glass Screen Protector for Nin...	7.99	4.8	98753.0	28.946673

```
# Creating Barplot Function for Data Analysis

def v_barplot(df, col1, col2):

    plt.figure(figsize=[12,10])
    sns.set_context('paper')
    temp_plot = sns.barplot(data = df.sort_values(by=col2, ascending=False), x = col1,
        y = col2, palette='pastel')
    temp_plot.set(title= col1 + ' vs ' + col2)
    temp_plot.set(xticklabels = [])

    # legend
    plt.legend(title = col1, loc = (1.10,0.25), labels = [i for i in df[col1]])
    plt.show(temp_plot)
```

```
v_barplot(temp_data, 'name', 'ratings_by_count')
```

Accounting for the different rating counts, the product sold on Amazon's Best Seller lists with the highest ratings is the **SanDisk 256GB microSDXC-Card** with a weighted ratings of 29.498574.



Of note, we used the following formula to calculate the weighted ratings count. This allowed to account for items that possess significantly more or less counts of ratings.

$$ratings\_by\_count = \frac{5p}{10} + 5(1 - e^{\frac{-q}{Q}}) \tag{1}$$

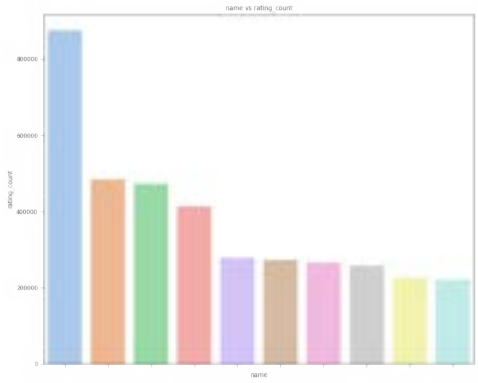
where p =ratings, q = rating counts, Q = approximate quantity or average of the group. Therefore,

$$Q = -\frac{M}{\ln \frac{1}{2}} \tag{2}$$

where M = mean rating counts.

Finding The Amazon's Best Seller Item With The Most Ratings

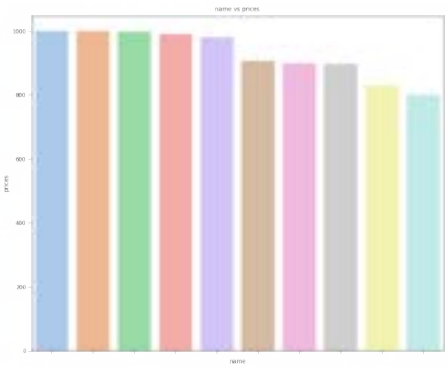
```
# Products With The Top Ten Ratings Count

top_ten_rated_count = df.sort_values(by='rating_count', ascending=False)
top_ten_rated_count = top_ten_rated_count.drop_duplicates(subset='rating_count',
    keep='first').head(10)

v_barplot(top_ten_rated_count, 'name', 'rating_count')
```



The **Echo Dot Smart Speaker** is the top item on Amazon's Best Sellers list with the most rating count.

Finding the Most Expensive Item on Amazon's Best Sellers

```
# Top Ten Most Expensive Amazon Best Sellers Products

top_ten_expensive = df.sort_values(by='prices', ascending=False)
top_ten_expensive = top_ten_expensive.drop_duplicates(subset='prices', keep='first').head(10)

v_barplot(top_ten_expensive, 'name', 'prices')
```

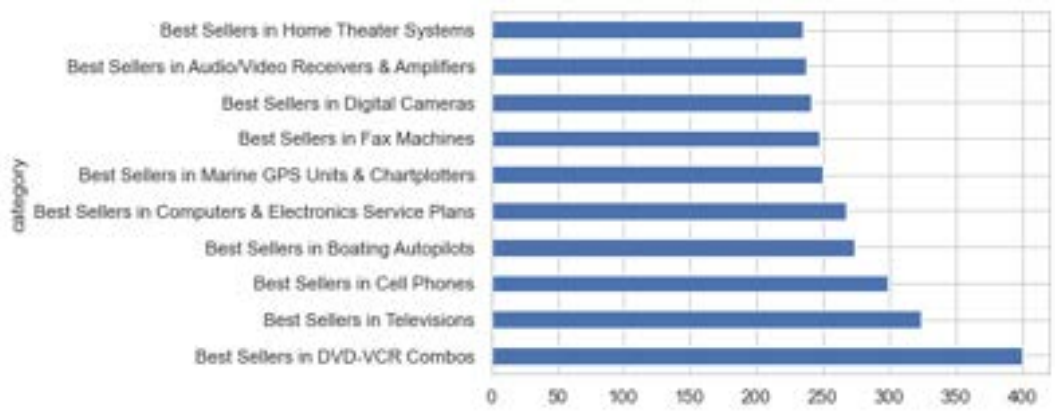


The **REOLINK 4k Security Camera System** is the most expensive item on Amazon's Best Sellers list.

Group 2 Questions

Finding the Category With the Highest Average Price

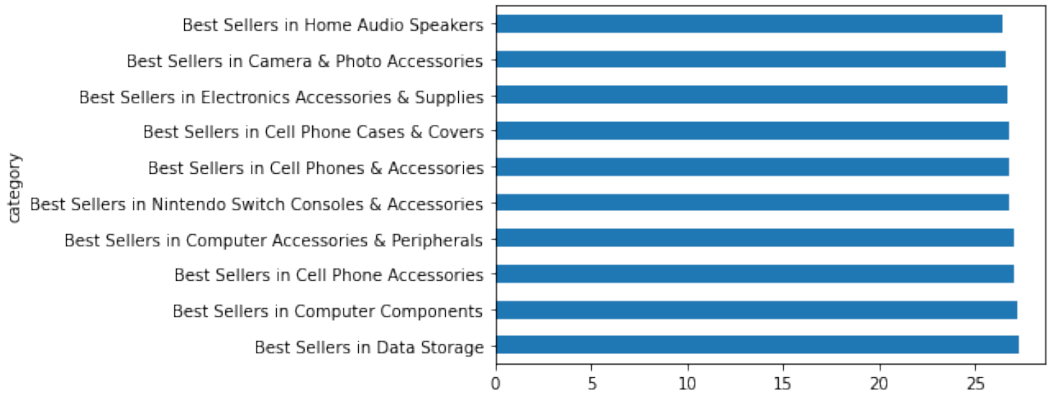
```
# Most Expensive Category

df.groupby('category')['prices'].mean().sort_values(ascending=False)
    .head(10).T.plot(kind='barh')
```



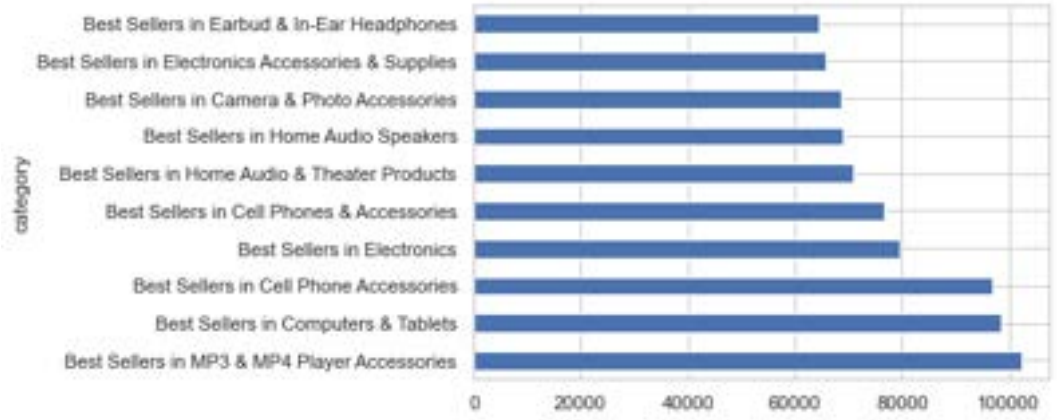
The most expensive category on Amazon's Best Sellers is the **Best Sellers in DVD-VCR Combos** with an average pricing of about $400 USD.

Finding the Category With the Highest Average Ratings and Ratings Count

```
# Most Highly Rated Category. Notice our top ten all have the same ratings.
# Need to account for product ratings with minimal rating counts.
# Using Weighted Ratings By Count to compare average values

df_prime.groupby('category')['ratings_by_count'].mean().sort_values(ascending=False).
    head(10).T.plot(kind='barh')
```




```
# Categories With The Most Rating Counts

df.groupby('category')['rating_count'].mean().sort_values(ascending=False)
    .head(10).T.plot(kind='barh')
```

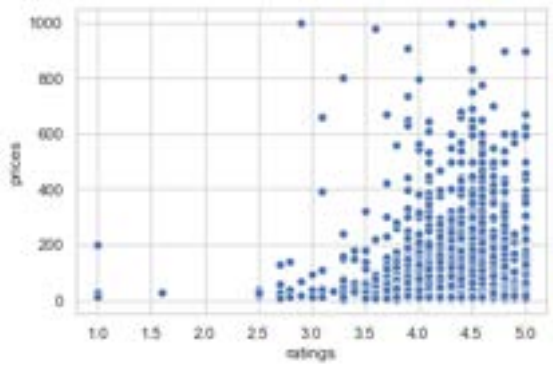


The category **Best Sellers in MP3 & MP4 Player Accessories** has the most ratings on Amazon's Best Sellers with an average rating count of ratings.

Group 3 Questions

Finding the Relationship Between Ratings and Item Price

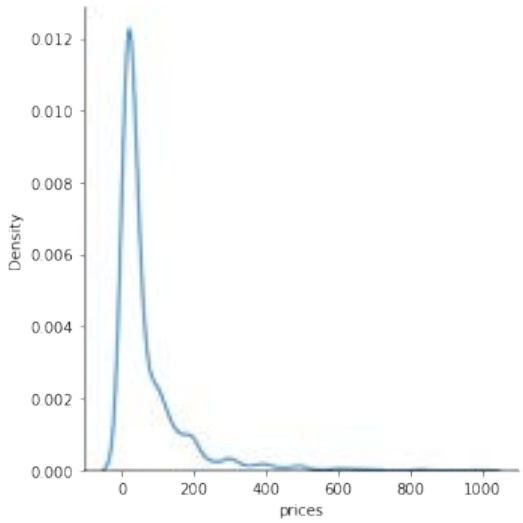
```
# Ratings Vs Price

sns.set_theme(style="whitegrid")
sns.scatterplot(data=df, x = 'ratings', y = 'prices')
```



From the plot above, there does seem to be a slight positive correlation between ratings and price, i.e., higher rank items tend to be more expensive. However, recall from the correlation plot in the EDA section, there does not exist a strong statistical positive correlation between these two variables.

Finding the Relationship Between Item Price and Item Ordering

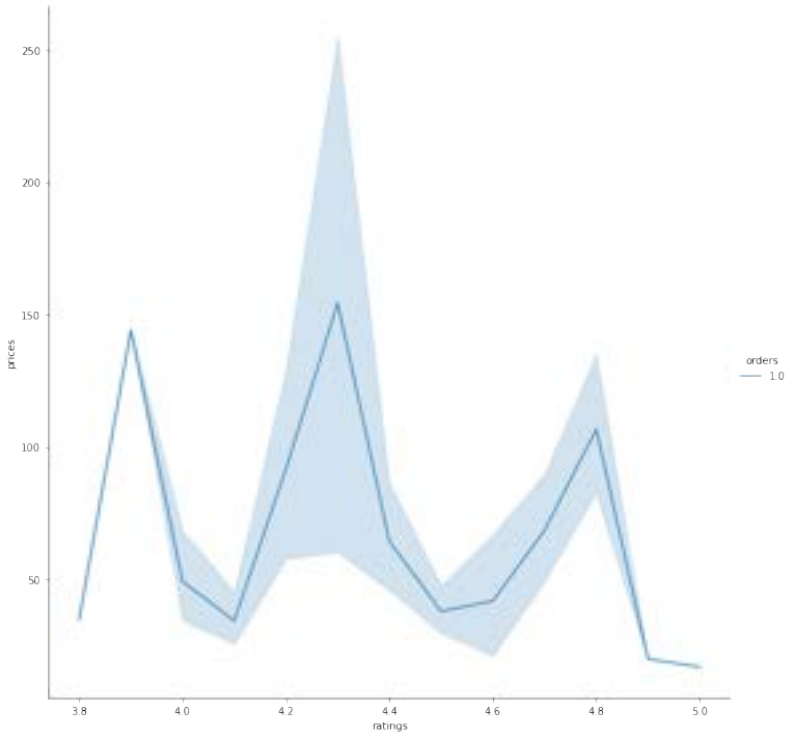
```
# Top Ten orderings vs Price

top_ten_orders = df[df['orders'] <= 10]
sns.displot(top_ten_orders, x = 'prices', kind = 'kde')
```



First, we want to look at the distribution for all items with ordering less than 10. We see that the pricing for these filters is right skewed, but the majority of the pricing ranges between the moderate range of $50-100 USD.

```
# First Order Ranking vs Price

sns.relplot(data = df.query("orders == 1"), x = 'ratings', y = 'prices', kind = 'line',
    hue = 'orders', height=10)
```

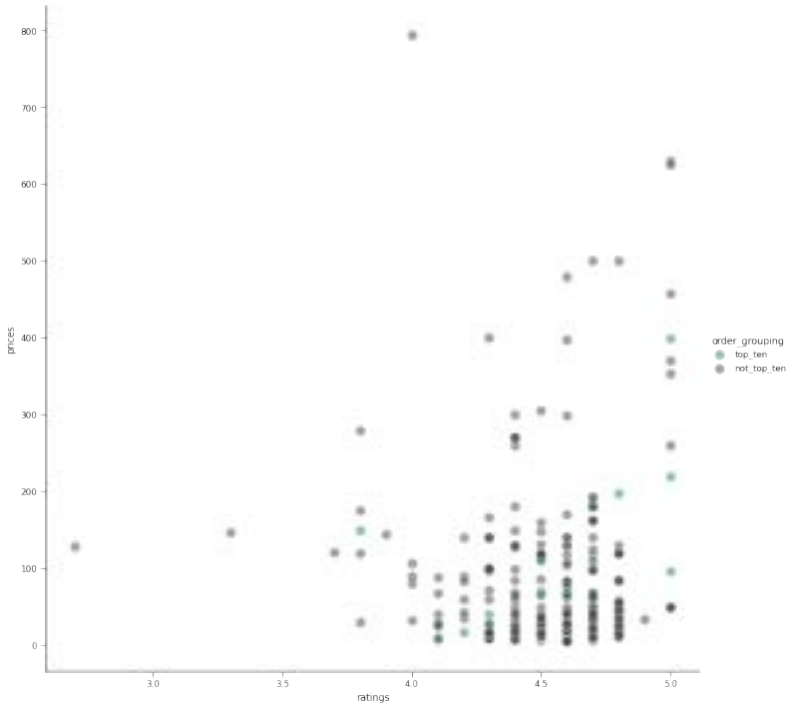
Next, we look at the pricing distribution between the different ratings for first ordered items. For items ranked first overall, the most expensive items have average ratings of 4.4 stars. On both extremes, the average pricing of the item will decrease to near $150 USD.



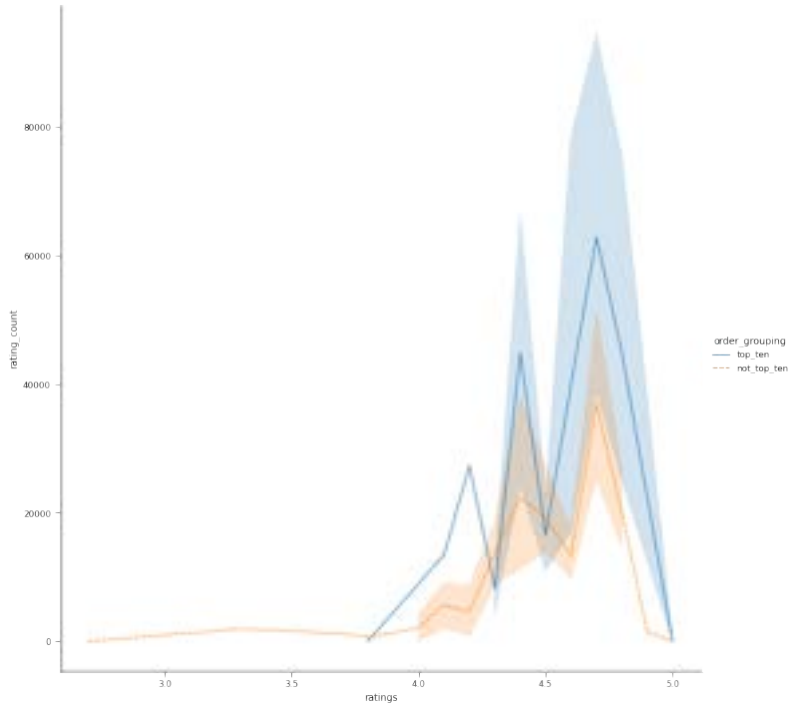
Finding the Relationship Between Ratings and Ratings Count For Items in the Electronics Category

```
# Creating Electronics Category
Electronics = df[df['category'].str.contains("Electronics") == True]

# Grouping Orders Variable For Visualization
Electronics['order_grouping'] = pd.cut(Electronics['orders'], bins = [0, 10, 81],
    labels = ['top_ten', 'not_top_ten'])

# Creating Scatterplot for Visualization
palette = dict(top_ten = "seagreen", not_top_ten = ".3")
g = sns.FacetGrid(Electronics, hue = 'order_grouping', palette= palette, height=10)
g.map(sns.scatterplot, 'ratings', 'prices', s = 100, alpha = .5)
g.add_legend()
```

Here, we see some positive correlation as predicted in our correlation plot above. That is, the higher ranked items with more stars tend to be higher in price. Moreover, we can also see that items in the top ten ordering tend to be cheaper than their non-top ten counterparts. However, it is important to note that this can be due to their different subset sizes.



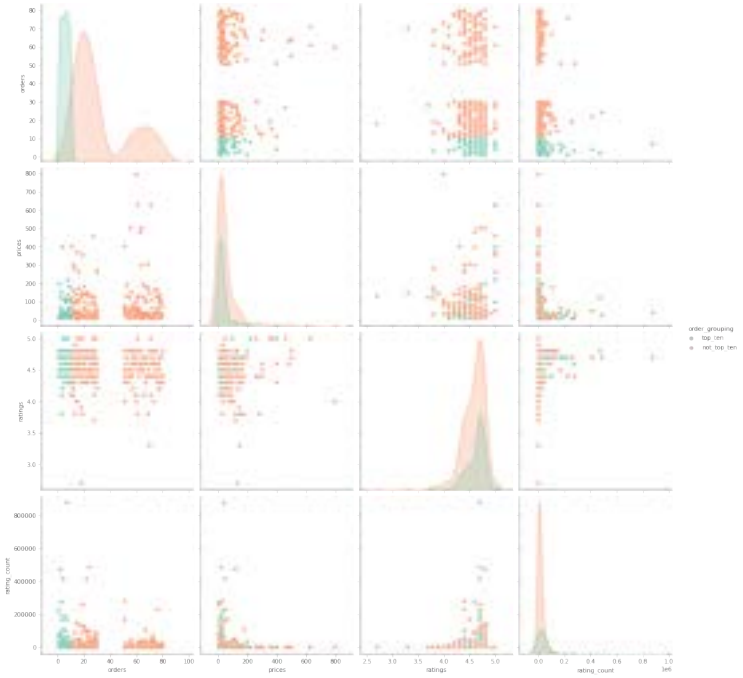

```
# Creating Line Plot
sns.relplot(data = Electronics, x = 'ratings', y = 'rating_count', kind = 'line',
    height = 10, hue = "order_grouping", style="order_grouping")
```



On average, we can also see that items that make the top ten order rankings will generally have more rating counts. However, the rating counts will decrease significantly for both items in the top ten ranking and those not in the top ten.

```
# Viewing Distribution for Electronics Category
sns.pairplot(data=Electronics, hue = 'order_grouping', palette= "Set2", diag_kind= "kde",
    height = 4)
```

Interestingly, from the Kernel Density Estimation, products in the top ten ranking statistically on average will cost less. However, they will also on average how a lower rating count and average rating.



Statistical Testing

```
# Mann-Whitney U Test

import scipy.stats as scs

def order_impact(col1):
    array1 = np.array(Electronics[Electronics['order_grouping']== "top_ten"][col1])
    array2= np.array(Electronics[Electronics['order_grouping']== "not_top_ten"][col1])
    t,p_value = scs.mannwhitneyu(array1, array2, method="asymptotic")
    if p_value < 0.05:
        print(col1," has a significant impact on the ordering rank of items on
            Amazons Best Sellers. The p-value is: ",p_value)
    else:
        print(col1," does not have a significant impact on the ordering rank of items on
            Amazons Best Sellers. The p-value is: ",p_value)
```

```
for i in Electronics.select_dtypes([np.number]).columns:
    order_impact(i)
```

```
orders has a significant impact on the ordering rank of items on Amazons Best Sellers.
    The p-value is:  5.603683794439682e-78
prices does not have a significant impact on the ordering rank of items on Amazons Best Sellers.
    The p-value is:  0.05278452935040999
```

```
ratings has a significant impact on the ordering rank of items on Amazons Best Sellers.
    The p-value is:  0.017610442092790467
rating_count has a significant impact on the ordering rank of items on Amazons Best Sellers.
    The p-value is:  3.111401951319074e-10
```

Conclusion

Using the Mann-Whitney Test, we see that the ratings and the amount of ratings a product receives on Amazon is significantly different between groups that make the top ten orderings on Amazon's Best Sellers compared to those products that don't. However, the pricing of an item is not statistically significant between items in the top ten Amazon's Best Sellers list compared to items not on the list. Overall,

1. Having higher **Ratings** and **Ratings Count** will statistically improve the order ranking of a product on Amazon's Best Sellers.
2. An Amazon store should not be selective about the pricing of an item. The **Price** of an item, *expensive* or *cheap*, will not statistically affect the order ranking of a Amazon's Best Sellers product. Therefore, the most optimal price is one that maximizes earned profits and minimizes production costs.
3. An Amazon store should sell an **Electronic** item because products in the **Electronics Category** tend to have the highest average ratings and rating counts, specifically the top three categories a store should focus on includes **Home Audio Speakers, Camera and Photo Accessories and Electronic Accessories and Supplies**