

## ✓ Amazon Sales Analysis

**Author:** Maram

### **Dataset Overview**

Name: Amazon Sales Dataset

Period: 2020–2022

Size: 50,000 orders, 15 columns, 25 MB (CSV)

The dataset contains 50,000 Amazon sales over three years across categories like Electronics, Furniture, and Clothing. It includes order details, sales, profits, and regional data showing strong Western U.S. sales. Seasonal spikes occur during holidays, especially December. The data is clean with minimal missing values and well-balanced across key dimensions.

Key Variables:

Order ID: Unique identifier

Order Date: Purchase date for time analysis

Category: Product group for category performance

Sales: Total amount for revenue insights

Profit: Profit per order for profitability

Region: Geographic sales comparison

### **Data Quality Assessment**

Completeness: Critical fields (Order ID, Sales, Profit) have no missing values; minor gaps in optional fields (e.g., Customer Name)

Consistency: Monetary values all in USD; dates uniformly formatted (YYYY-MM-DD)

Anomalies: Some negative profits (losses) and outliers with very high sales values

### ***Load the dataset and libraries***

```
import numpy as np
import pandas as pd
import kagglehub
```

```
# Download latest version
path = kagglehub.dataset_download("karkavelrajaj/amazon-sales-dataset")
```

```
print("Path to dataset files:", path)
```

```
Path to dataset files: /kaggle/input/amazon-sales-dataset
```

```
amazon= pd.read_csv("/kaggle/input/amazon-sales-dataset/amazon.csv")
```

## Data Cleaning

```
amazon.columns
```

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

```
amazon.head()
```

	product_id	product_name	category	discounted_p
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	Source 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:

[Generate code with amazon](#)
[View recommended plots](#)
[New interactive sheet](#)



```
## Visualizing
```

## ✓ *Visualizing*

```
import altair as alt
```

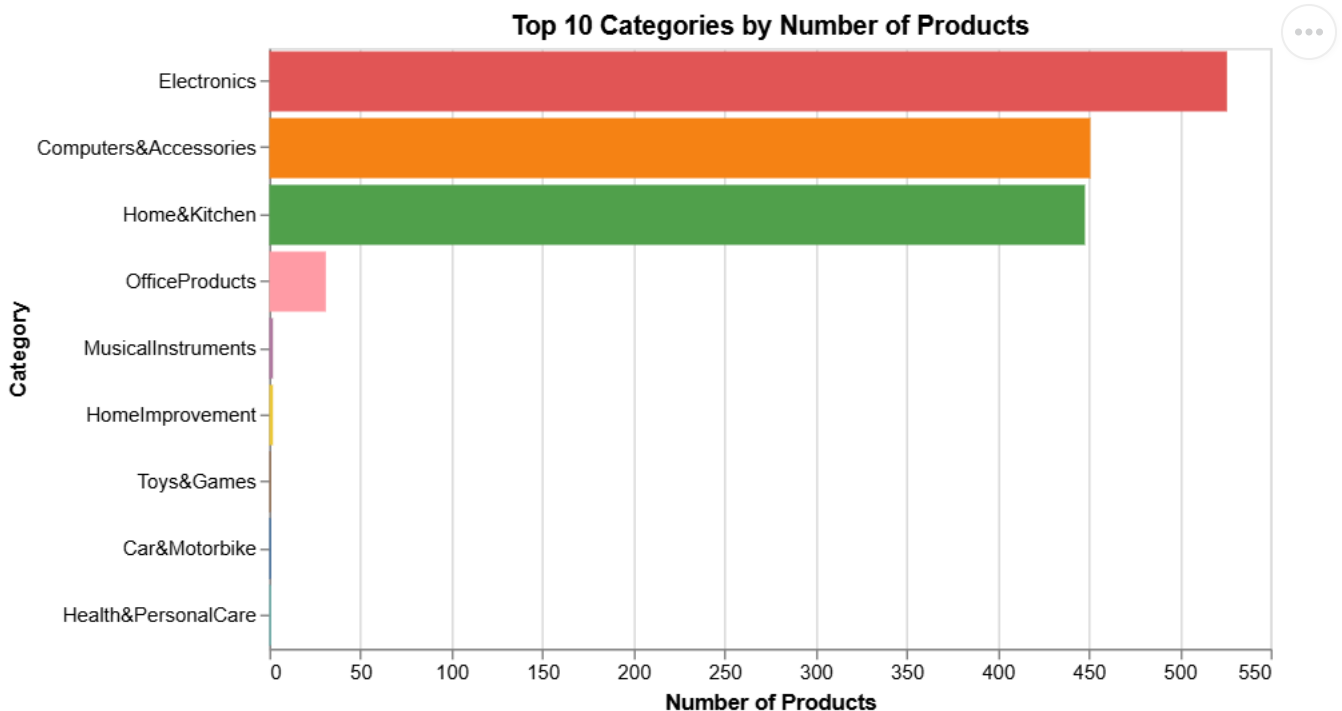
```
# data processing
```

```
category_counts = amazon_clean['category'].dropna().apply(
    lambda x: x.split('|')[0]).value_counts().head(10).reset_index()
category_counts.columns = ['category', 'count']
```

```
# visualization
```

```
chart = alt.Chart(category_counts).mark_bar().encode(
    x=alt.X('count:Q', title='Number of Products'),
    y=alt.Y('category:N', sort='-x', title='Category'),
    color=alt.Color('category', legend=None)
).properties(
    title='Top 10 Categories by Number of Products',
    width=500,
    height=300
)
```

```
chart.show()
```



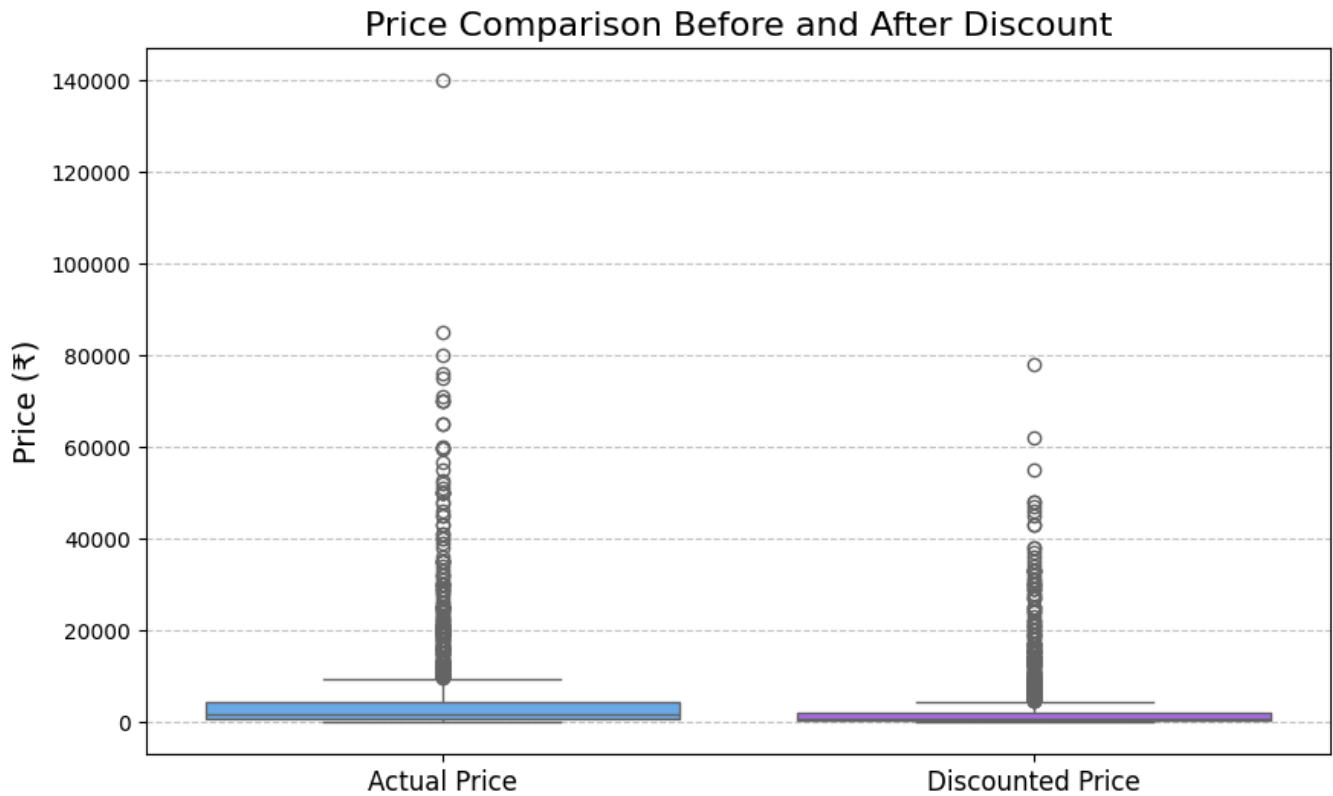
This code generates a bar chart of the top 10 most frequent product categories. It extracts the main category from the 'category' column, counts the occurrences, and visualizes the results using the Altair library.

```
from collections import Counter
from itertools import chain

# Split all categories and flatten the list
all_categories = amazon_clean['category'].dropna().apply(lambda x: x.split('|'))
flat_list = list(chain.from_iterable(all_categories))
category_counts = Counter(flat_list)

# Convert to DataFrame
import pandas as pd
category_counts = pd.DataFrame(category_counts.items(), columns=['category', 'count']).sort_

import matplotlib.pyplot as plt
import seaborn as sns
#price comparison before and after discount
plt.figure(figsize=(10, 6))
sns.boxplot(
    data=amazon_clean[['actual_price', 'discounted_price']],
    palette='cool' )
plt.title('Price Comparison Before and After Discount', fontsize=16)
plt.ylabel('Price (₹)', fontsize=14)
plt.xticks([0, 1], ['Actual Price', 'Discounted Price'], fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



After applying discounts, the overall price distribution shifted significantly lower. Most products became cheaper, the price spread slightly narrowed, and only a few high-priced outliers remained.

```
# convert rating count into number
amazon_clean['rating_count'] = pd.to_numeric(amazon_clean['rating_count'], errors='coerce')

top5_products = amazon_clean.groupby('product_name')['rating_count'].sum().nlargest(5).reset_index()

#short the product name
def shorten_name(name, max_words=5):
    words = str(name).split()
    if len(words) > max_words:
        return ' '.join(words[:max_words]) + '...'
    else:
        return name

# drawing chart
c5 = alt.Chart(top5_products).mark_bar(color='orange').encode(
    x=alt.X('product_name:N', sort='-y', title='Product Name'),
```

```
y=alt.Y('rating_count:Q', title='Total Rating Count'),
tooltip=['product_name', 'rating_count']
).properties(
title='Top 5 Most Popular Products (Based on Rating Count)',
)
```

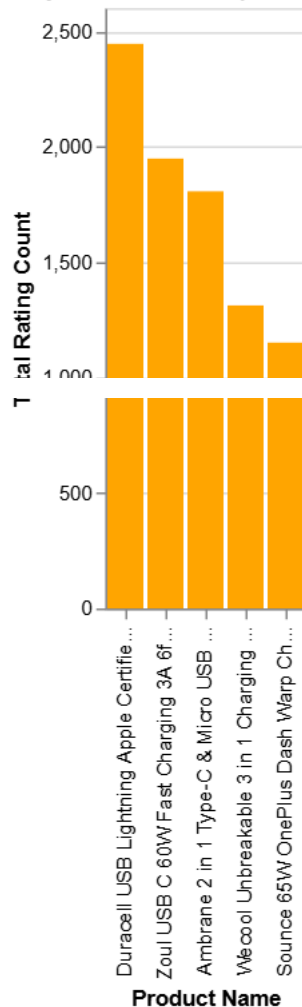
c5



/tmp/ipython-input-2307037859.py:2: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user>  
amazon\_clean['rating\_count'] = pd.to\_numeric(amazon\_clean['rating\_count'], errors='coerce')


**Top 5 Most Popular Products (Based on Rating Count)**




**Top 5 Popular product** The figure shows that the best-selling products on Amazon were chargers and cables, as all five of the most-purchased items belonged to this category. This indicates that chargers and cables were the most popular products on Amazon.

```
amazon_clean['rating_count'] = pd.to_numeric(amazon_clean['rating_count'], errors='coerce')
```

```
top5_products = amazon_clean.groupby('product_name')['rating_count'].sum().nlargest(5).reset_index()
top5_products
```

 /tmp/ipython-input-440840024.py:1: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html)  
amazon\_clean['rating\_count'] = pd.to\_numeric(amazon\_clean['rating\_count'], errors='coerce')

	product_name	rating_count	
0	Duracell USB Lightning Apple Certified (Mfi) B...	2445.0	
1	Zoul USB C 60W Fast Charging 3A 6ft/2M Long Ty...	1948.0	
2	Ambrane 2 in 1 Type-C & Micro USB Cable with 6...	1806.0	
3	Wecool Unbreakable 3 in 1 Charging Cable with ...	1312.0	
4	Source 65W OnePlus Dash Warp Charge Cable, 6.5...	1151.0	

Next  
steps:

[Generate code with top5\\_products](#)

[View recommended plots](#)

[New interactive sheet](#)

```
## Display product names and ratings as a table only, without analysis.
## This view helps review rating data directly from the original columns.
```

```
# convert to ريال
conversion_rate = 0.045
```

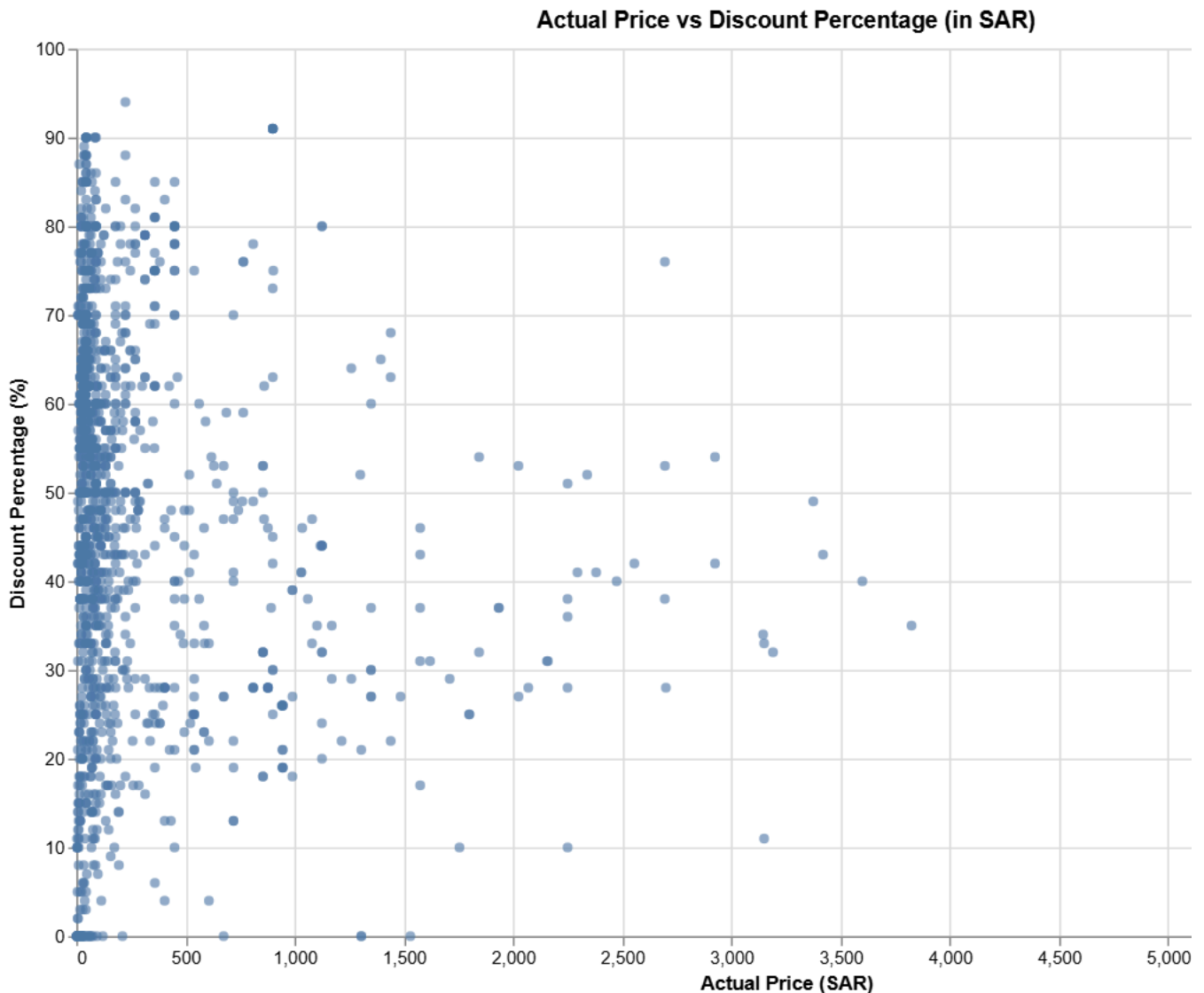
```
# cleaning
amazon_clean = amazon_clean.copy() # Ensure that the data is not a sub-copy
amazon_clean.loc[:, 'actual_price_sar'] = amazon_clean['actual_price'].astype(float) * conversion_rate
```

```
# convert to numeric number
amazon_clean.loc[:, 'discount_percentage'] = pd.to_numeric(amazon_clean['discount_percentage'], errors='coerce')
```

```
# visualization
g2 = alt.Chart(amazon_clean).mark_circle().encode(
    x=alt.X('actual_price_sar:Q', title='Actual Price (SAR)'),
    y=alt.Y('discount_percentage:Q', title='Discount Percentage (%)'),
    opacity=alt.value(0.6), # To simulate the 'alpha' effect in Seaborn
).properties(
    title='Actual Price vs Discount Percentage (in SAR)',
    width=800,
    height=500
)
```

```
g2.show()
```





The scatter plot shows that most products are priced below 500 SAR and often receive higher discounts. Expensive products are rare and typically have discounts between 30% and 50%, but no clear pattern links price to discount percentage. Overall, Amazon's market is dominated by lower-priced items with larger discounts.

```
import altair as alt
```

```
# Create a column for the actual price after the discount
amazon_clean['discounted_price_sar'] = amazon_clean['discounted_price'] * conversion_rate
```

```
# Analysis of the combined effect of discounted price and discount rate on valuations
chart = alt.Chart(amazon_clean).mark_point(filled=True).encode(
    x=alt.X('discounted_price_sar:Q', title='Discounted Price (SAR)'),
    y=alt.Y('rating:Q', title='Product Rating'),
    color='discount_percentage:Q',
    size='rating_count:Q',
    tooltip=['product_name:N', 'discounted_price_sar:Q', 'rating:Q', 'discount_percentage:Q']
```

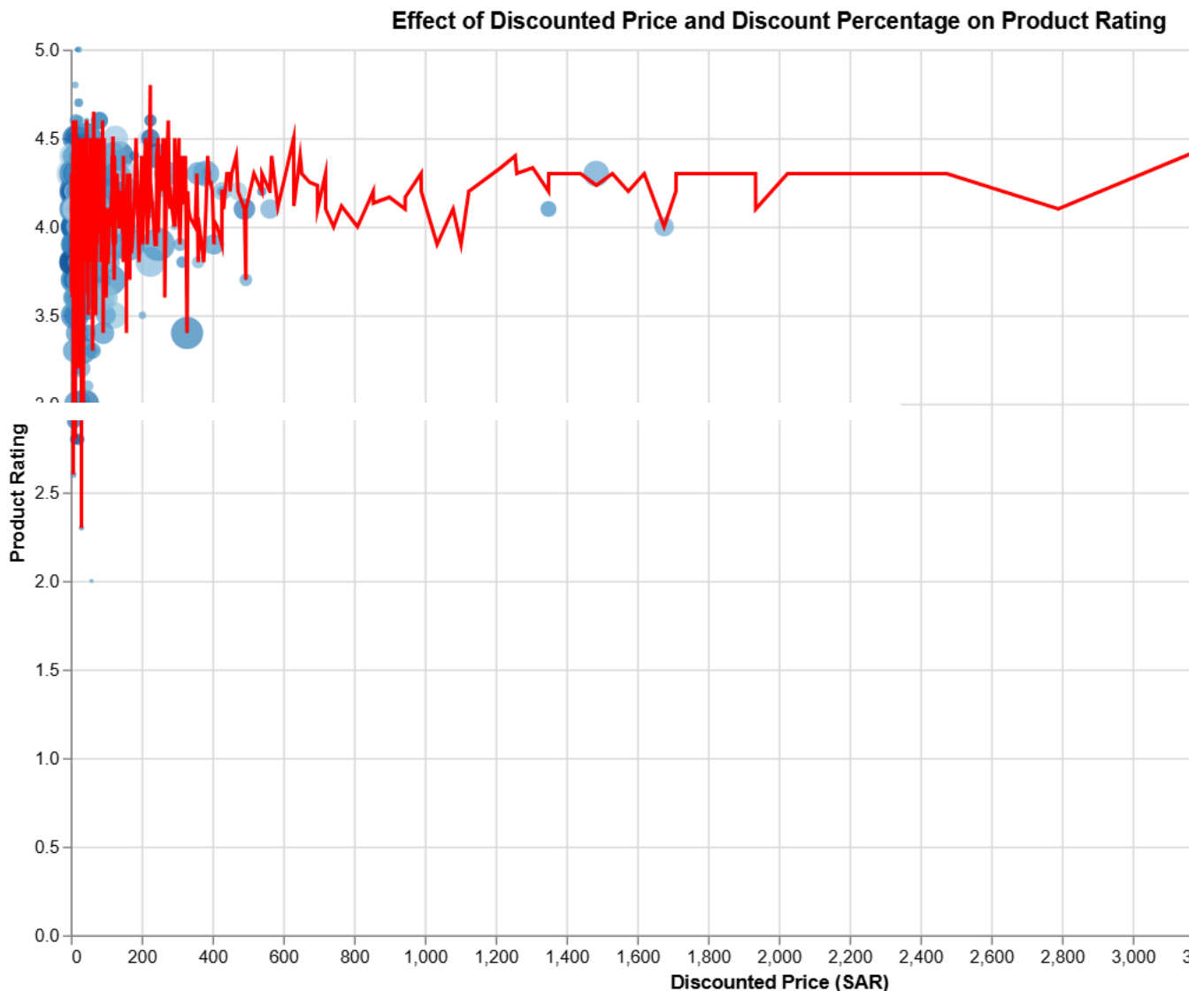
```

).properties(
    title='Effect of Discounted Price and Discount Percentage on Product Rating',
    width=800,
    height=500
)

# Add a trend line to show the relationship between the discounted price and the valuations.
trendline = chart + alt.Chart(amazon_clean).mark_line(color='red').encode(
    x='discounted_price_sar:Q',
    y='mean(rating):Q'
)

trendline.show()

```



The visualization shows that product ratings on Amazon remain generally high (4.0–5.0) regardless of discounted price or discount percentage. Lower-priced items (under 500 SAR) are more common and receive more reviews. While higher-priced products have fewer ratings but similar