In [53]:

```
import pandas as pd

# Load the product and category files
products_df = pd.read_csv('archive/amazon_products.csv')
categories_df = pd.read_csv('archive/amazon_categories.csv')
```

In [54]:

```
products_df.head()
```

Out[54]:

	asin	title	imgUrl	productURL
0	B014TMV5YE	Sion Softside Expandable Roller Luggage, Black...	https://m.media-amazon.com/images/I/815dLQKYIY...	https://www.amazon.com/dp/B014TMV5YE
1	B07GDLCQXV	Luggage Sets Expandable PC+ABS Durable Suitcas...	https://m.media-amazon.com/images/I/81bQlm7vf6...	https://www.amazon.com/dp/B07GDLCQXV
2	B07XSCCZYG	Platinum Elite Softside Expandable Checked Lug...	https://m.media-amazon.com/images/I/71EA35zvJB...	https://www.amazon.com/dp/B07XSCCZYG
3	B08MVFKGJM	Freeform Hardside Expandable with Double Spinn...	https://m.media-amazon.com/images/I/91k6NYLQyI...	https://www.amazon.com/dp/B08MVFKGJM
4	B01DJLKZBA	Winfield 2 Hardside Expandable Luggage with Sp...	https://m.media-amazon.com/images/I/61NJoaZcP9...	https://www.amazon.com/dp/B01DJLKZBA

In [55]:

```
categories_df.head()
```

Out[55]:

	id	category_name
0	1	Beading & Jewelry Making
1	2	Fabric Decorating
2	3	Knitting & Crochet Supplies
3	4	Printmaking Supplies

	id	category_name
4	5	Scrapbooking & Stamping Supplies

In [56]:
```
merged_df = products_df.merge(categories_df, how='left', left_on='category_id', right_o
merged_df.head()
```

Out[56]:

	asin	title	imgUrl	productURL
0	B014TMV5YE	Sion Softside Expandable Roller Luggage, Black...	https://m.media-amazon.com/images/I/815dLQKYIY...	https://www.amazon.com/dp/B014TMV5YE
1	B07GDLCQXV	Luggage Sets Expandable PC+ABS Durable Suitcas...	https://m.media-amazon.com/images/I/81bQlm7vf6...	https://www.amazon.com/dp/B07GDLCQXV
2	B07XSCCZYG	Platinum Elite Softside Expandable Checked Lug...	https://m.media-amazon.com/images/I/71EA35zvJB...	https://www.amazon.com/dp/B07XSCCZYG
3	B08MVFKGJM	Freeform Hardside Expandable with Double Spinn...	https://m.media-amazon.com/images/I/91k6NYLQyl...	https://www.amazon.com/dp/B08MVFKGJM
4	B01DJLKZBA	Winfield 2 Hardside Expandable Luggage with Sp...	https://m.media-amazon.com/images/I/61NJoaZcP9...	https://www.amazon.com/dp/B01DJLKZBA

In [57]:
```
merged_df.drop(columns=['id'], inplace=True)
merged_df.head()
```

Out[57]:

	asin	title	imgUrl	productURL
0	B014TMV5YE	Sion Softside Expandable Roller Luggage, Black...	https://m.media-amazon.com/images/I/815dLQKYIY...	https://www.amazon.com/dp/B014TMV5YE
1	B07GDLCQXV	Luggage Sets	https://m.media-amazon.com/images/I/81bQlm7vf6...	https://www.amazon.com/dp/B07GDLCQXV

	asin	title	imgUrl	productURL
		Expandable PC+ABS Durable Suitcas...		
2	B07XSCCZYG	Platinum Elite Softside Expandable Checked Lug...	https://m.media-amazon.com/images/I/71EA35zvJB...	https://www.amazon.com/dp/B07XSCCZYG
3	B08MVFKGJM	Freeform Hardside Expandable with Double Spinn...	https://m.media-amazon.com/images/I/91k6NYLQyI...	https://www.amazon.com/dp/B08MVFKGJM
4	B01DJLKZBA	Winfield 2 Hardside Expandable Luggage with Sp...	https://m.media-amazon.com/images/I/61NJoaZcP9...	https://www.amazon.com/dp/B01DJLKZBA

In [58]:
```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1426337 entries, 0 to 1426336
Data columns (total 12 columns):
 #   Column             Non-Null Count    Dtype  
---  ------             --------------    -----  
 0   asin               1426337 non-null  object 
 1   title              1426336 non-null  object 
 2   imgUrl             1426337 non-null  object 
 3   productURL         1426337 non-null  object 
 4   stars              1426337 non-null  float64
 5   reviews            1426337 non-null  int64  
 6   price              1426337 non-null  float64
 7   listPrice          1426337 non-null  float64
 8   category_id        1426337 non-null  int64  
 9   isBestSeller       1426337 non-null  bool   
 10  boughtInLastMonth  1426337 non-null  int64  
 11  category_name      1426337 non-null  object 
dtypes: bool(1), float64(3), int64(3), object(5)
memory usage: 131.9+ MB
```

In [59]:
```
merged_df.isnull().sum()
```

Out[59]:
```
asin               0
title              1
imgUrl             0
productURL         0
stars              0
reviews            0
price              0
listPrice          0
category_id        0
```

```
isBestSeller         0
boughtInLastMonth    0
category_name        0
dtype: int64
```

In [60]:
```
merged_df.dropna(subset=['title'], inplace=True)
```

In [61]:
```
# Calculate discount percentage
merged_df['discount_percentage'] = ((merged_df['listPrice'] - merged_df['price']) / mer
merged_df['discount_percentage'].fillna(0, inplace=True)
merged_df['discount_percentage'] = merged_df['discount_percentage'].replace([float('-in
merged_df.head()
```

Out[61]:

	asin	title	imgUrl	productURL
0	B014TMV5YE	Sion Softside Expandable Roller Luggage, Black...	https://m.media-amazon.com/images/I/815dLQKYIY...	https://www.amazon.com/dp/B014TMV5YE
1	B07GDLCQXV	Luggage Sets Expandable PC+ABS Durable Suitcas...	https://m.media-amazon.com/images/I/81bQlm7vf6...	https://www.amazon.com/dp/B07GDLCQXV
2	B07XSCCZYG	Platinum Elite Softside Expandable Checked Lug...	https://m.media-amazon.com/images/I/71EA35zvJB...	https://www.amazon.com/dp/B07XSCCZYG
3	B08MVFKGJM	Freeform Hardside Expandable with Double Spinn...	https://m.media-amazon.com/images/I/91k6NYLQyI...	https://www.amazon.com/dp/B08MVFKGJM
4	B01DJLKZBA	Winfield 2 Hardside Expandable Luggage with Sp...	https://m.media-amazon.com/images/I/61NJoaZcP9...	https://www.amazon.com/dp/B01DJLKZBA

In [62]:
```
merged_df['popularity_score'] = merged_df['stars'] * merged_df['reviews']
```

In [63]:
```
category_freq = merged_df['category_name'].value_counts(normalize=True)
merged_df['category_freq'] = merged_df['category_name'].map(category_freq)
```

In [64]:
```
merged_df['price_range'] = pd.qcut(merged_df['price'], q=5, labels=['Very Low', 'Low',
```

In [65]:
```
# Calculate average stars and reviews for bestsellers and non-bestsellers
bestseller_stats = merged_df.groupby('isBestSeller').agg(
    avg_stars=('stars', 'mean'),
    avg_reviews=('reviews', 'mean')
).reset_index()

print(bestseller_stats)
```

```
   isBestSeller  avg_stars  avg_reviews
0         False   3.996539   167.903922
1          True   4.494038  2318.628521
```

In [66]:
```
merged_df['has_discount'] = merged_df['discount_percentage'] > 0
```

In [67]:
```
merged_df.head()
```

Out[67]:

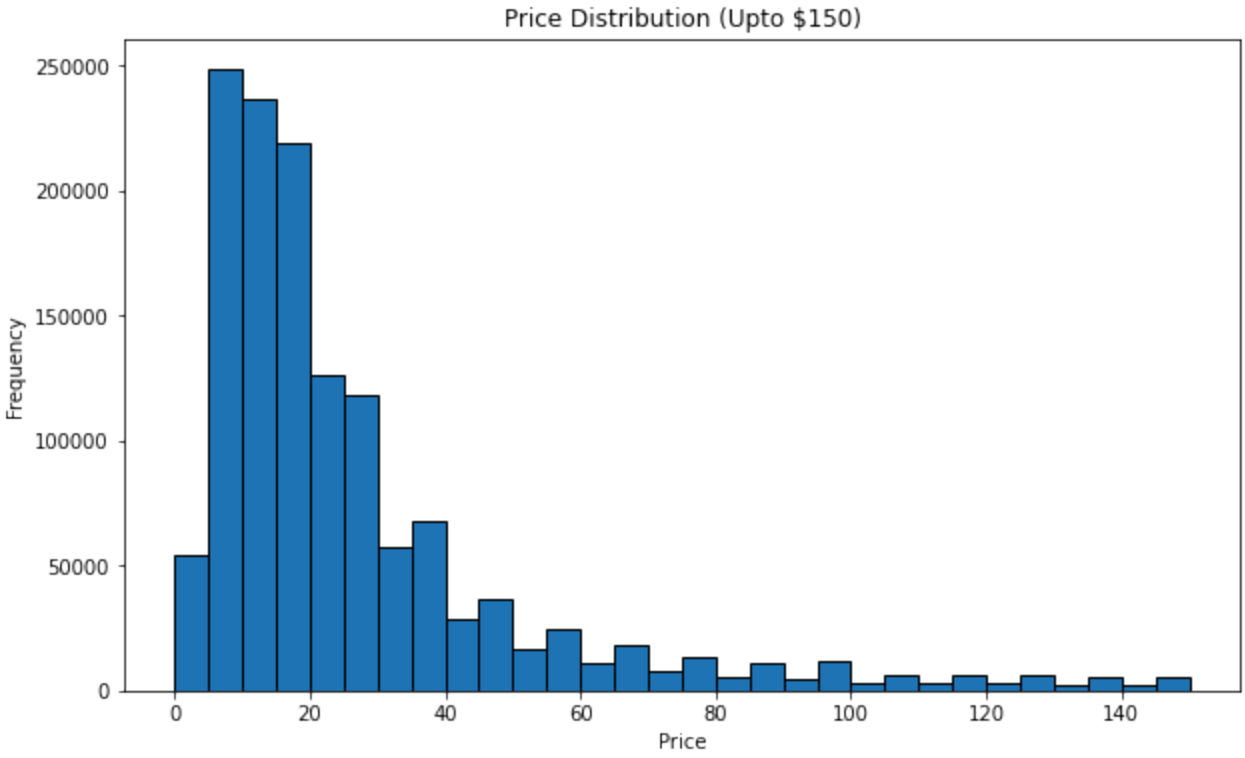
	asin	title	imgUrl	productURL
0	B014TMV5YE	Sion Softside Expandable Roller Luggage, Black...	https://m.media-amazon.com/images/I/815dLQKYIY...	https://www.amazon.com/dp/B014TMV5YE
1	B07GDLCQXV	Luggage Sets Expandable PC+ABS Durable Suitcas...	https://m.media-amazon.com/images/I/81bQlm7vf6...	https://www.amazon.com/dp/B07GDLCQXV
2	B07XSCCZYG	Platinum Elite Softside Expandable Checked Lug...	https://m.media-amazon.com/images/I/71EA35zvJB...	https://www.amazon.com/dp/B07XSCCZYG
3	B08MVFKGJM	Freeform Hardside Expandable with Double Spinn...	https://m.media-amazon.com/images/I/91k6NYLQyI...	https://www.amazon.com/dp/B08MVFKGJM
4	B01DJLKZBA	Winfield 2 Hardside Expandable Luggage with Sp...	https://m.media-amazon.com/images/I/61NJoaZcP9...	https://www.amazon.com/dp/B01DJLKZBA

In [68]:
```
merged_df.price.describe()
```

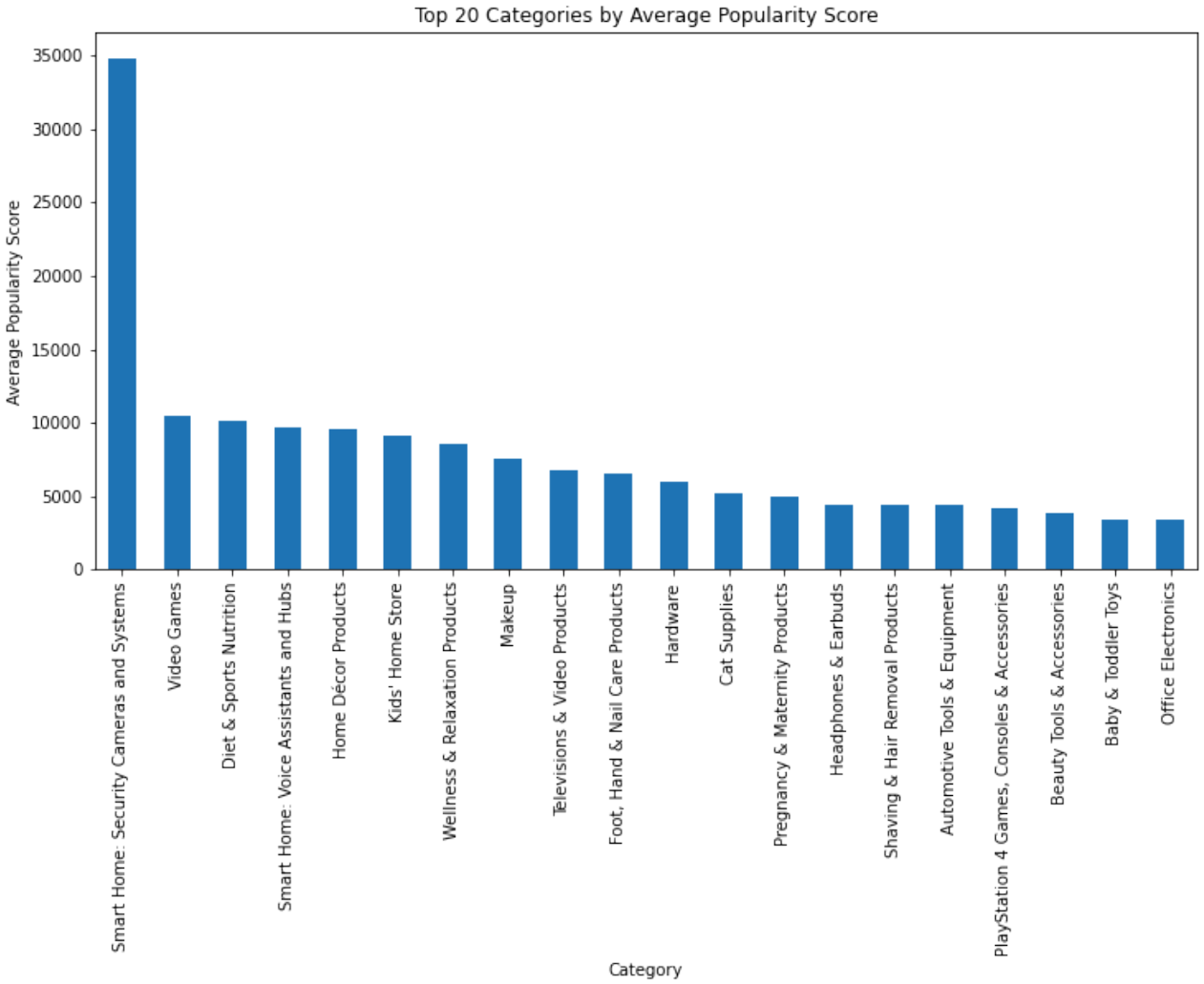
Out[68]:
```
count    1.426336e+06
mean     4.337541e+01
std      1.302893e+02
min      0.000000e+00
25%      1.199000e+01
50%      1.995000e+01
75%      3.599000e+01
max      1.973181e+04
Name: price, dtype: float64
```

In [69]:
```
import matplotlib.pyplot as plt
filtered_df = merged_df[merged_df['price'] <= 150]

plt.figure(figsize=(10, 6))
filtered_df['price'].plot(kind='hist', bins=30, edgecolor='black')
plt.title('Price Distribution (Upto $150)')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```

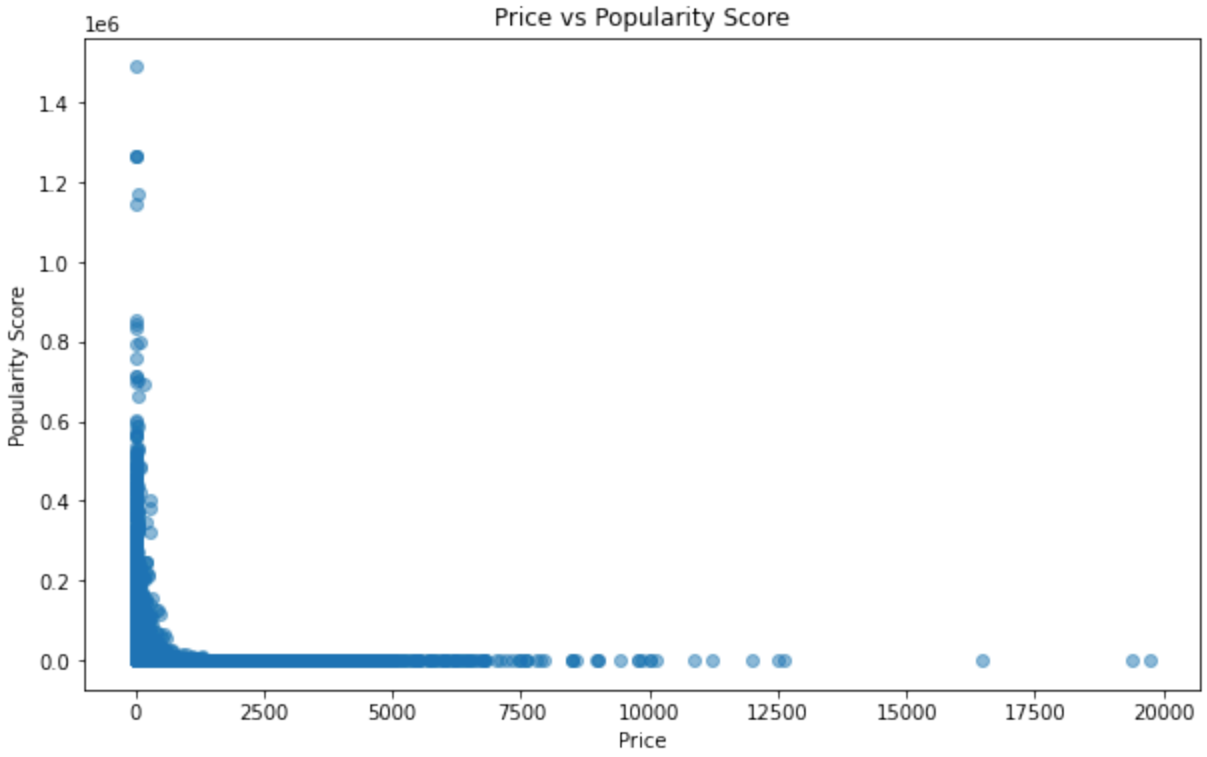


In [70]:
```
category_popularity = merged_df.groupby('category_name')['popularity_score'].mean().sor
top_categories = category_popularity.head(20)
top_categories.plot(kind='bar', figsize=(12, 6))
plt.title('Top 20 Categories by Average Popularity Score')
plt.xlabel('Category')
plt.ylabel('Average Popularity Score')
plt.show()
```



In [71]:

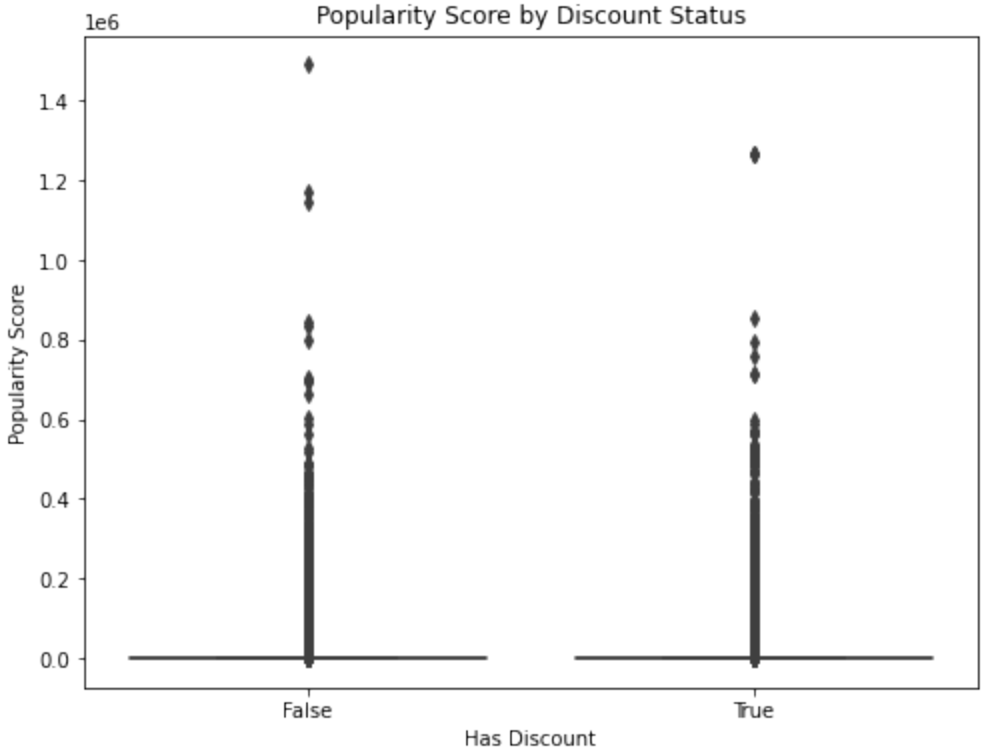
```
plt.figure(figsize=(10, 6))
plt.scatter(merged_df['price'], merged_df['popularity_score'], alpha=0.5)
plt.title('Price vs Popularity Score')
plt.xlabel('Price')
plt.ylabel('Popularity Score')
plt.show()
```



In [72]:

```
import seaborn as sns

plt.figure(figsize=(8, 6))
sns.boxplot(x='has_discount', y='popularity_score', data=merged_df)
plt.title('Popularity Score by Discount Status')
plt.xlabel('Has Discount')
plt.ylabel('Popularity Score')
plt.show()
```



In [73]:

```
import seaborn as sns

# Calculate average price and discount percentage for best-sellers vs. non-best-sellers
bestseller_analysis = merged_df.groupby('isBestSeller').agg(
    avg_price=('price', 'mean'),
    avg_discount_percentage=('discount_percentage', 'mean')
).reset_index()
```

In [74]:

```
# Plotting the average price and discount percentage for best-sellers vs. non-best-sell
fig, ax1 = plt.subplots(figsize=(12, 6))

# Plotting average price
sns.barplot(x='isBestSeller', y='avg_price', data=bestseller_analysis, ax=ax1, palette=
ax1.set_ylabel('Average Price', color='b')
ax1.set_title('Average Price and Discount Percentage for Best-Sellers vs. Non-Best-Sell

# Creating a second y-axis for the discount percentage
ax2 = ax1.twinx()
sns.lineplot(x='isBestSeller', y='avg_discount_percentage', data=bestseller_analysis, a
ax2.set_ylabel('Average Discount Percentage', color='r')

plt.show()
```

In [75]:

```
# Convert 'price_range' to numerical values
price_range_mapping = {'Very Low': 1, 'Low': 2, 'Medium': 3, 'High': 4, 'Very High': 5}
merged_df['price_range_num'] = merged_df['price_range'].map(price_range_mapping).astype

# Convert 'has_discount' to numerical values (True -> 1, False -> 0)
merged_df['has_discount_num'] = merged_df['has_discount'].astype(int)

#Convert 'isBestSeller' to numerical values (True -> 1, False -> 0)
merged_df['isBestSeller_num'] = merged_df['isBestSeller'].astype(int)
```

```
# Display the updated dataframe
merged_df.head()
```

```
c:\Users\harsh\AppData\Local\Programs\Python\Python38\lib\site-packages\pandas\core\arra
ys\categorical.py:528: RuntimeWarning: invalid value encountered in cast
  fill_value = lib.item_from_zerodim(np.array(np.nan).astype(dtype))
```

Out[75]:

	asin	title	imgUrl	productURL
0	B014TMV5YE	Sion Softside Expandable Roller Luggage, Black...	https://m.media-amazon.com/images/I/815dLQKYIY...	https://www.amazon.com/dp/B014TMV5YE
1	B07GDLCQXV	Luggage Sets Expandable PC+ABS Durable Suitcas...	https://m.media-amazon.com/images/I/81bQlm7vf6...	https://www.amazon.com/dp/B07GDLCQXV
2	B07XSCCZYG	Platinum Elite Softside Expandable Checked Lug...	https://m.media-amazon.com/images/I/71EA35zvJB...	https://www.amazon.com/dp/B07XSCCZYG
3	B08MVFKGJM	Freeform Hardside Expandable with Double Spinn...	https://m.media-amazon.com/images/I/91k6NYLQyI...	https://www.amazon.com/dp/B08MVFKGJM
4	B01DJLKZBA	Winfield 2 Hardside Expandable Luggage with Sp...	https://m.media-amazon.com/images/I/61NJoaZcP9...	https://www.amazon.com/dp/B01DJLKZBA

In [76]:

```
from sklearn.preprocessing import StandardScaler

# Select the features for clustering
features = ['price', 'stars', 'category_freq', 'discount_percentage', 'has_discount_num
X = merged_df[features]

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [77]:

```
X_scaled
```

Out[77]:

```
array([[ 0.74153896,  0.37230665, -1.4824595 , ..., -0.56501227,
         1.41410129, -0.0775193 ],
       [ 0.97179579,  0.37230665, -1.4824595 , ...,  1.76987306,
```

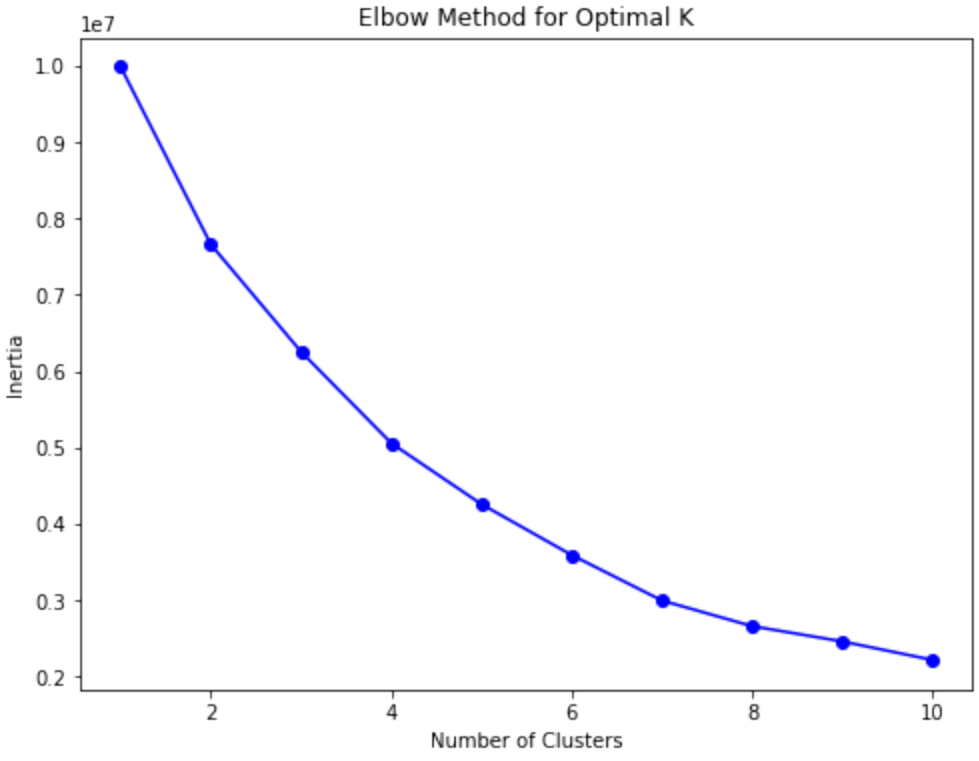
```
        1.41410129, -0.0775193 ],
       [ 2.47230279,  0.44669525, -1.4824595 , ...,  1.76987306,
        1.41410129, -0.0775193 ],
       ...,
       [-0.26736971, -0.29719072,  1.50580172, ..., -0.56501227,
        -1.38214707, -0.0775193 ],
       [ 0.08430922,  0.37230665,  1.50580172, ...,  1.76987306,
        1.41410129, -0.0775193 ],
       [-0.18869862,  0.66986104,  1.50580172, ..., -0.56501227,
        0.01597711, -0.0775193 ]])
```

In [78]:

```python
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

inertia = []
K = range(1, 11)
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

# Plot the elbow curve
plt.figure(figsize=(8, 6))
plt.plot(K, inertia, 'bo-')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal K')
plt.show()
```



In [79]:

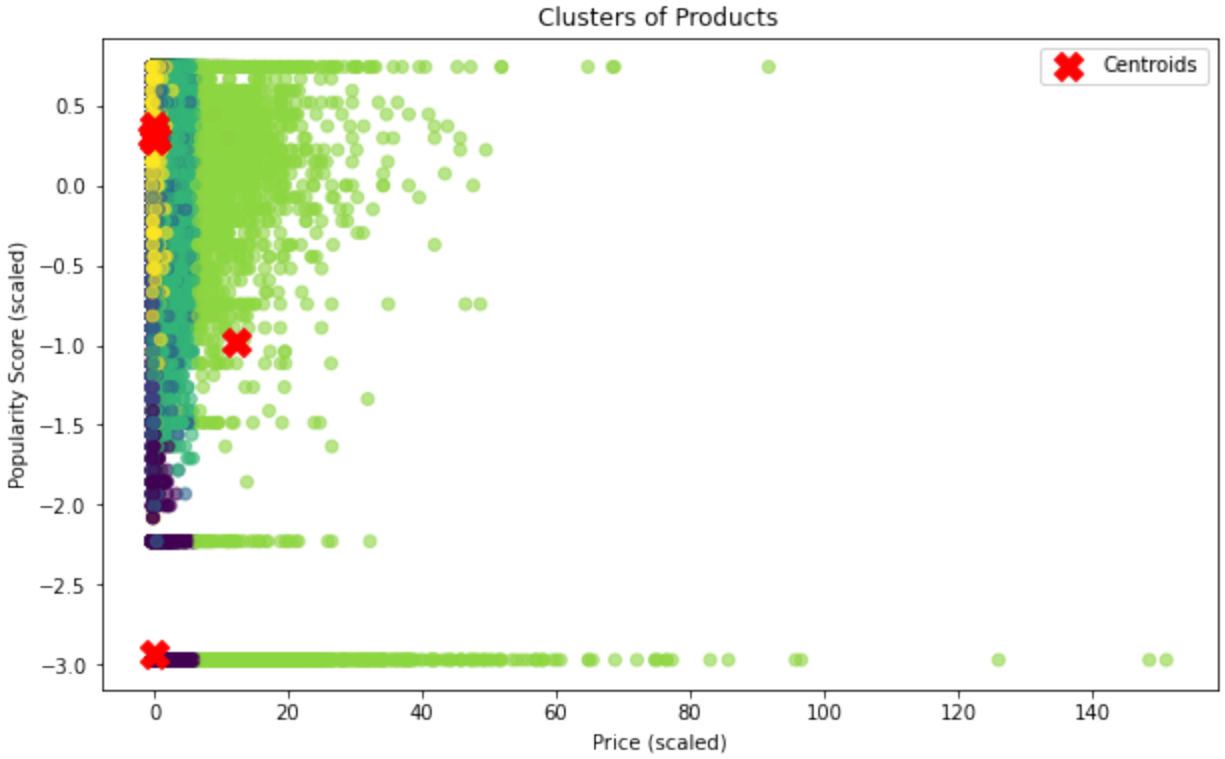
```python
# Apply K-Means with k = 7
k_optimal = 7
kmeans = KMeans(n_clusters=k_optimal, random_state=42, n_init=10)
merged_df['cluster'] = kmeans.fit_predict(X_scaled)
```

```
# Display cluster centers for analysis
centroids = kmeans.cluster_centers_
print("Cluster Centers:\n", scaler.inverse_transform(centroids))
```

```
Cluster Centers:
 [[ 4.96351231e+01  4.73489828e-02  6.20673311e-03  1.37358689e+00
   9.79398287e-02  3.09759786e+00 -4.59701721e-17]
 [ 1.28271205e+01  4.41852873e+00  5.01455559e-03 -6.89602829e-03
   1.98130031e-06  1.90349878e+00 -4.59701721e-17]
 [ 3.85696773e+01  4.38104565e+00  6.28516405e-03  2.14570157e+01
   1.00000000e+00  3.03374833e+00 -4.59701721e-17]
 [ 2.97217981e+01  4.49403756e+00  5.66862471e-03  1.21670637e+01
   4.70892019e-01  2.80422535e+00  1.00000000e+00]
 [ 7.75461940e+01  4.40122948e+00  4.87287419e-03 -7.45862393e-02
   1.19128957e-03  4.48976017e+00 -4.68375339e-17]
 [ 1.63592641e+03  2.68141593e+00  5.06622130e-03  9.38981416e-02
   1.29505720e-02  5.00000000e+00 -3.72965547e-17]
 [ 3.82309387e+01  4.42956615e+00  1.40894240e-02  4.82022530e-01
   4.48909541e-02  3.37591066e+00 -4.33680869e-17]]
```

In [80]:

```
# Scatter plot for visualizing clusters based on price and popularity_score
plt.figure(figsize=(10, 6))
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=merged_df['cluster'], cmap='viridis', alp
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='X', s=200, label='Centro
plt.xlabel('Price (scaled)')
plt.ylabel('Popularity Score (scaled)')
plt.title('Clusters of Products')
plt.legend()
plt.show()
```



In [81]:

```
import numpy as np
from sklearn.metrics import silhouette_score

# Sample 10% of the data for silhouette score calculation because it will take too long
sampled_data = merged_df.sample(frac=0.10, random_state=42)
```

```
sampled_X_scaled = scaler.transform(sampled_data[features])

# Calculate silhouette score on the sampled data
silhouette_avg = silhouette_score(sampled_X_scaled, sampled_data['cluster'])
print("Silhouette Score for K-Means Clustering (Sampled):", silhouette_avg)
```

```
Silhouette Score for K-Means Clustering (Sampled): 0.39818409678916317
```

In [82]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

In [83]:

```
# Define features and targets
features = ['stars', 'category_freq', 'discount_percentage', 'has_discount_num', 'price
X = merged_df[features]

# Targets for prediction
y_price = merged_df['price']

# Split data into training and test sets (80% train, 20% test)
X_train_price, X_test_price, y_train_price, y_test_price = train_test_split(X, y_price,
```

In [84]:

```
# Initialize and train the model
lin_reg_price = LinearRegression()
lin_reg_price.fit(X_train_price, y_train_price)

# Make predictions
y_pred_price = lin_reg_price.predict(X_test_price)

# Evaluate the model
mae_price = mean_absolute_error(y_test_price, y_pred_price)
mse_price = mean_squared_error(y_test_price, y_pred_price)
r2_price = r2_score(y_test_price, y_pred_price)

print("Linear Regression - Price Prediction")
print("Mean Absolute Error:", mae_price)
print("Mean Squared Error:", mse_price)
print("R^2 Score:", r2_price)
```

```
Linear Regression - Price Prediction
Mean Absolute Error: 36.69571328505955
Mean Squared Error: 15723.934608584797
R^2 Score: 0.10695645822602695
```

In [41]:

```
# Initialize and train the model
rf_price = RandomForestRegressor(random_state=42, n_estimators=100)
rf_price.fit(X_train_price, y_train_price)

# Make predictions
y_pred_price_rf = rf_price.predict(X_test_price)

# Evaluate the model
mae_price_rf = mean_absolute_error(y_test_price, y_pred_price_rf)
mse_price_rf = mean_squared_error(y_test_price, y_pred_price_rf)
r2_price_rf = r2_score(y_test_price, y_pred_price_rf)
```

```
print("\nRandom Forest - Price Prediction")
print("Mean Absolute Error:", mae_price_rf)
print("Mean Squared Error:", mse_price_rf)
print("R^2 Score:", r2_price_rf)
```

```
Random Forest - Price Prediction
Mean Absolute Error: 20.322096428268498
Mean Squared Error: 12299.433681166156
R^2 Score: 0.30145157113246823
```

In [42]:

```
# Feature importance for price prediction model
feature_importances_price = rf_price.feature_importances_
print("\nFeature Importances for Price Prediction:", dict(zip(features, feature_importa
```

```
Feature Importances for Price Prediction: {'stars': 0.08943249526563392, 'category_fre
q': 0.42347377261763913, 'discount_percentage': 0.07561485023918989, 'has_discount_num':
0.0052462278109501066, 'price_range_num': 0.40568142810871227, 'isBestSeller_num': 0.000
5512259578745926}
```

In [85]:

```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

In [86]:

```
# Ridge Regression for Price
ridge_price = Ridge(alpha=1.0)
ridge_price.fit(X_train_price, y_train_price)
y_pred_price_ridge = ridge_price.predict(X_test_price)

# Evaluate the model
mae_price_ridge = mean_absolute_error(y_test_price, y_pred_price_ridge)
mse_price_ridge = mean_squared_error(y_test_price, y_pred_price_ridge)
r2_price_ridge = r2_score(y_test_price, y_pred_price_ridge)

print("\nRidge Regression - Price Prediction")
print("Mean Absolute Error:", mae_price_ridge)
print("Mean Squared Error:", mse_price_ridge)
print("R^2 Score:", r2_price_ridge)
```

```
Ridge Regression - Price Prediction
Mean Absolute Error: 36.67289622015926
Mean Squared Error: 15724.10871637196
R^2 Score: 0.1069465697446279
```

In [87]:

```
from xgboost import XGBRegressor
# Initialize XGBoost regressor
xgb_price = XGBRegressor(objective='reg:squarederror', random_state=42, n_estimators=10

# Train the model
xgb_price.fit(X_train_price, y_train_price)

# Make predictions
y_pred_price_xgb = xgb_price.predict(X_test_price)

# Evaluate the model
mae_price_xgb = mean_absolute_error(y_test_price, y_pred_price_xgb)
mse_price_xgb = mean_squared_error(y_test_price, y_pred_price_xgb)
r2_price_xgb = r2_score(y_test_price, y_pred_price_xgb)

print("\nXGBoost - Price Prediction")
```

```
print("Mean Absolute Error:", mae_price_xgb)
print("Mean Squared Error:", mse_price_xgb)
print("R^2 Score:", r2_price_xgb)
```

```
XGBoost - Price Prediction
Mean Absolute Error: 20.160883539758085
Mean Squared Error: 12152.67097287302
R^2 Score: 0.3097869841240032
```

In [88]:

```
# Feature importance for price prediction
print("Feature Importances for Price Prediction:", xgb_price.feature_importances_)
```

```
Feature Importances for Price Prediction: [0.07070964 0.25723383 0.02671749 0.
0.6442789  0.00106011]
```