Appendix

In [3]:
```
import pandas as pd
from datetime import datetime
import json
```

In [4]:
```
# Upload users table
users=pd.read_csv('users.csv', index_col=0)
display(users.head())
```

DOB	State	Age
1983-07-31	Oregon	42
1998-07-27	Massachusetts	27
1950-08-08	Idaho	75
1969-08-03	Florida	56
2001-07-26	Georgia	24

In [5]:
```
# Upload products table
products=pd.read_csv('products.csv', index_col=0)
display(products[:3])
```

Uniq_id	SKU	Name	Description
b6c0b6bea69c722939585baeac73c13d	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	Youll return to our Alfred Dunner pull-on capr...
93e5272c51d8cce02597e3ce67b7ad0a	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	Youll return to our Alfred Dunner pull-on capr...
013e320f2f2ec0cf5b3ff5418d688528	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	Youll return to our Alfred Dunner pull-on capr...

In [6]:
```
# Upload reviews table
```

```
reviews = pd.read_csv('reviews.csv', index_col=1)
display(reviews[:3])
```

Username	Uniq_id	Score	Review
fsdv4141	b6c0b6bea69c722939585baeac73c13d	2	You never have to worry about the fit...Alfred...
krpz1113	b6c0b6bea69c722939585baeac73c13d	1	Good quality fabric. Perfect fit. Washed very ...
mbmg3241	b6c0b6bea69c722939585baeac73c13d	2	I do not normally wear pants or capris that ha...

In [7]:
```
# Upload jcpenney_products.json table
records = []
with open('jcpenney_products.json', 'r') as f:
    for line in f:
        line = line.strip()
        if line:  # skip empty lines
            try:
                record = json.loads(line)  # parse each JSON object
                records.append(record)
            except json.JSONDecodeError:
                # If the line is partial or malformed, try to fix/r
                pass
jcpenney_products = pd.DataFrame(records)
if 'uniq_id' in jcpenney_products.columns:
    jcpenney_products.rename(columns={'uniq_id': 'Uniq_id'}, inplac
    jcpenney_products.set_index('Uniq_id', inplace=True)
display(jcpenney_products[:3])
```

	sku	name_title	description
Uniq_id			
b6c0b6bea69c722939585baeac73c13d	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...
93e5272c51d8cce02597e3ce67b7ad0a	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...
013e320f2f2ec0cf5b3ff5418d688528	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...

In [8]:
```
# Upload jcpenney_reviewers.json table
jcpenney_reviewers = pd.read_json('jcpenney_reviewers.json', lines='
jcpenney_reviewers.set_index('Username', inplace=True)
display(jcpenney_reviewers.head())
```

	DOB	State	Reviewed
Username			
bkpn1412	31.07.1983	Oregon	[cea76118f6a9110a893de2b7654319c0]
gqjs4414	27.07.1998	Massachusetts	[fa04fe6c0dd5189f54fe600838da43d3]
eehe1434	08.08.1950	Idaho	[]
hkxj1334	03.08.1969	Florida	[f129b1803f447c2b1ce43508fb822810, 3b0c9bc0be6...
jjbd1412	26.07.2001	Georgia	[]

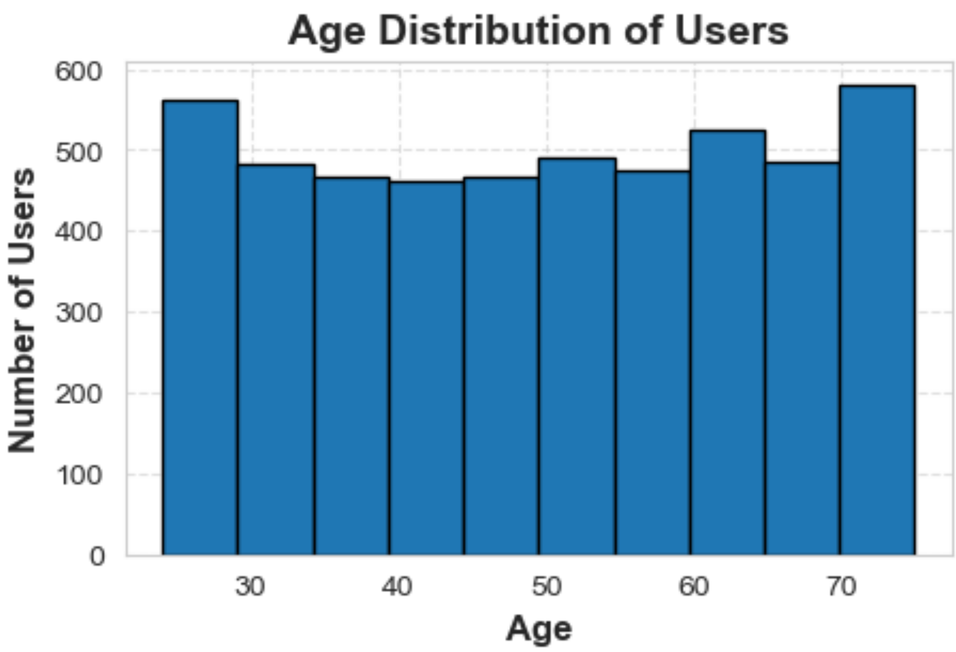
In [9]:
```
# Add Ages of customers
from datetime import date
users = pd.read_csv('users.csv')
users['DOB'] = pd.to_datetime(users['DOB'], errors='coerce')
def calculate_age(born):
    if pd.isnull(born):
        return None
    today = date.today()
    return today.year - born.year - ((today.month, today.day) < (bo
users['Age'] = users['DOB'].apply(calculate_age)
users = users[['DOB', 'State', 'Age']]
users.to_csv('users.csv', index=False)
```

```
display(users[:3])
```

	DOB	State	Age
0	1983-07-31	Oregon	42
1	1998-07-27	Massachusetts	27
2	1950-08-08	Idaho	75

In [27]:

```
# Visualisation of Ages
import matplotlib.pyplot as plt
plt.figure(figsize=(5, 3))
plt.hist(users['Age'], bins=10, edgecolor='black')
plt.title('Age Distribution of Users', fontsize=14, fontweight='bol
plt.xlabel('Age', fontsize=12, fontweight='bold')
plt.ylabel('Number of Users', fontsize=12, fontweight='bold')
plt.grid(True, linestyle='--', alpha=0.6)
plt.savefig('AgeDistribution.png', dpi = 300)
plt.show()
```



Observation

The age distribution of customers is relatively even, with no significant peaks. Customers represent a wide range of age groups, from 24 to 72 years old. The difference between the most and least common ages is relatively small, indicating that customers of all age groups are well represented and that the store appeals to a broad, diverse audience without a pronounced age bias.

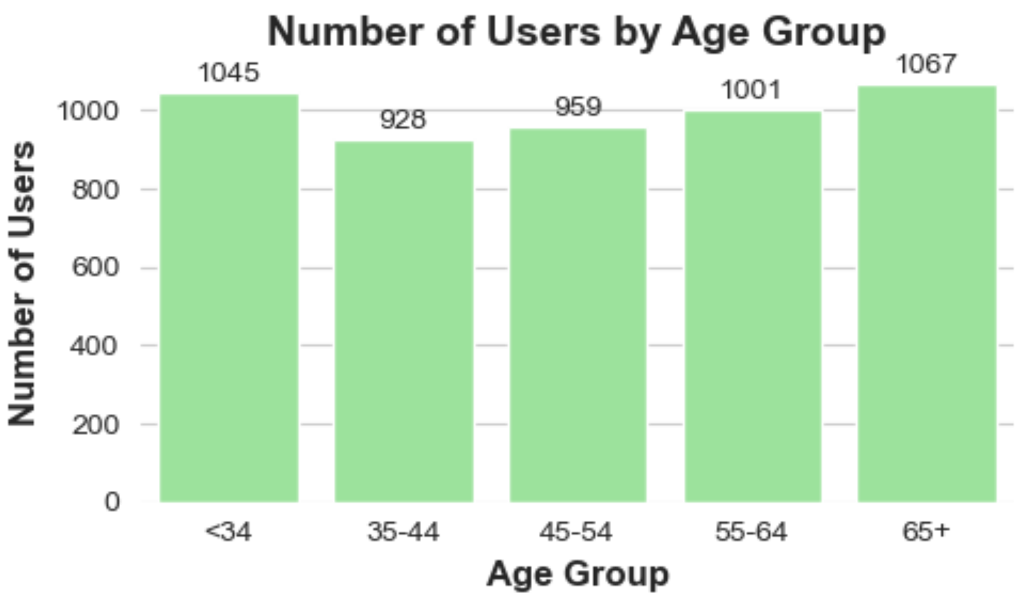
In [11]:

```
# Ages groups with numbers of customers
bins = [0, 34, 44, 54, 64, 100]
labels = ['<34', '35-44', '45-54', '55-64', '65+']
users['Age_Group'] = pd.cut(users['Age'], bins=bins, labels=labels,
users.to_csv('users_with_age_group.csv', index=False)
```

```
age_group_counts = users['Age_Group'].value_counts().sort_index()
print("Number of users per age group:")
display(age_group_counts)
```

```
Number of users per age group:
Age_Group
<34      1045
35-44     928
45-54     959
55-64    1001
65+      1067
Name: count, dtype: int64
```

In [26]:
```
# Visualisation of Ages Groups
import seaborn as sns
plt.figure(figsize=(5, 3))
sns.set_style('whitegrid')
sns.barplot(x=age_group_counts.index, y=age_group_counts.values, co
for i, value in enumerate(age_group_counts.values):
    plt.text(i, value + 10, str(value), ha='center', va='bottom', f
plt.title('Number of Users by Age Group', fontsize=14, fontweight='
plt.xlabel('Age Group', fontsize=12, fontweight='bold')
plt.ylabel('Number of Users', fontsize=12, fontweight='bold')
sns.despine(left=True, bottom=True)
plt.tight_layout()
plt.savefig('Ages groups.png', dpi = 300)
plt.show()
```



Observation

We divide customers into the following age groups:

- under 34 years old Young Adults
- 35–44 Adults 1
- 45–54 Adults 2
- 55–64 Older Adults

- Over 65 Elderly People

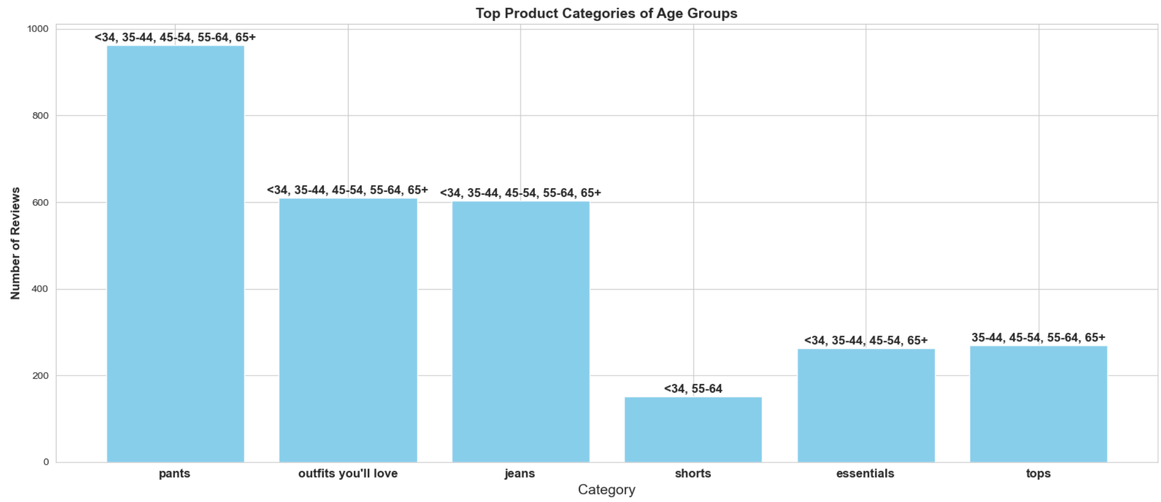
And the graph confirmed previous observation that the needs of each age group are being met by the company, as none of the groups appears to be prioritised.

In [13]:
```
# Top 5 categories for each Age group
jcpenney_products = pd.read_json('jcpenney_products.json', lines=Tr
jcpenney_reviewers = pd.read_json('jcpenney_reviewers.json', lines=
reviews = pd.read_csv('reviews.csv', index_col=1)
products = pd.read_csv('products.csv', index_col=0)
users = pd.read_csv('users.csv', index_col=0)
from datetime import date
jcpenney_reviewers = jcpenney_reviewers.copy()
jcpenney_reviewers['DOB'] = pd.to_datetime(jcpenney_reviewers['DOB'
def calculate_age(born):
    if pd.isnull(born):
        return None
    today = date.today()
    return today.year - born.year - ((today.month, today.day) < (bo
jcpenney_reviewers['Age'] = jcpenney_reviewers['DOB'].apply(calcula
bins = [0, 34, 44, 54, 64, 100]
labels = ['<34', '35-44', '45-54', '55-64', '65+']
jcpenney_reviewers['Age_Group'] = pd.cut(jcpenney_reviewers['Age'],
# Products
if 'uniq_id' in jcpenney_products.columns:
    jcpenney_products.rename(columns={'uniq_id': 'Uniq_id'}, inplac
# Join
reviews = reviews.reset_index().rename(columns={'index': 'Username'
merged = (reviews.merge(jcpenney_reviewers[['Username', 'Age_Group'
# Ignore useless categories
ignore_words = ['view all', 'view all brands', 'sale', 'clearance',
merged = merged[merged['category'].notna() &(merged['category'].str
# Top 5
category_counts = (merged.groupby(['Age_Group', 'category'], observ
top5_dict = {}
for age_group, group in category_counts.groupby('Age_Group', observ
    top5 = group.sort_values('Review_Count', ascending=False).head(
    top5_list = [f"{row['category']} ({row['Review_Count']})" for _
    while len(top5_list) < 5:
        top5_list.append("")
    top5_dict[age_group] = top5_list
# Table
top5_df = pd.DataFrame(top5_dict)
display(top5_df)
top5_df.to_csv('Top5_Categories_by_Age.csv', index=False)
```

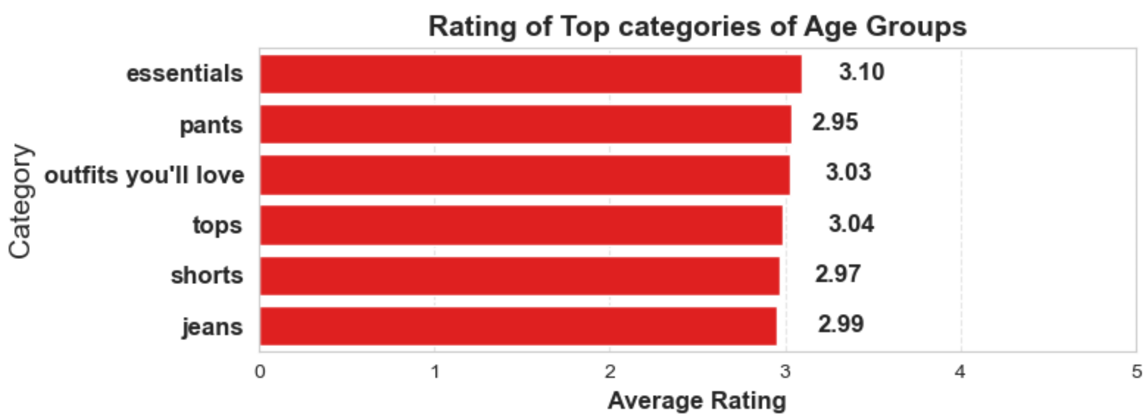
	<34	35-44	45-54	55-64	65+
0	pants (195)	pants (179)	pants (175)	pants (204)	pants (210)
1	outfits you'll love (139)	jeans (106)	outfits you'll love (115)	jeans (135)	jeans (130)
2	jeans (136)	outfits you'll love (104)	jeans (96)	outfits you'll love (125)	outfits you'll love (127)
3	shorts (76)	tops (67)	essentials (62)	shorts (75)	tops (80)
4	essentials (69)	essentials (62)	tops (58)	tops (64)	essentials (69)

In [32]:

```
# Visualisation of Top categories of Age Groups
ignore_categories = ['view all', 'view all brands', 'sale', '', Non
data = category_counts[~category_counts['category'].isin(ignore_cat
top5_per_age_list = []
for age_group, group in data.groupby('Age_Group', observed=False):
    top5 = group.sort_values('Review_Count', ascending=False).head(
    top5_per_age_list.append(top5)
top5_per_age = pd.concat(top5_per_age_list, ignore_index=True)
categories = top5_per_age['category'].unique()
category_totals = []
age_labels = []
for cat in categories:
    subset = top5_per_age[top5_per_age['category'] == cat]
    total_reviews = subset['Review_Count'].sum()
    ages = ', '.join(subset['Age_Group'].astype(str).tolist())
    category_totals.append(total_reviews)
    age_labels.append(ages)
plt.figure(figsize=(16,7))
bars = plt.bar(categories, category_totals, color='skyblue')
for bar, label in zip(bars, age_labels):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height + 5, label, ha
plt.xlabel('Category', fontsize=14)
plt.ylabel('Number of Reviews', fontsize=12, fontweight='bold')
plt.title('Top Product Categories of Age Groups', fontsize=14, font
plt.xticks(rotation=0, ha='center', fontsize=12, fontweight='bold')
plt.tight_layout()
plt.savefig('Top categories of Age Groups', dpi = 300)
plt.show()
```



In [35]:
```
# Rating of the Top categories of Age groups
target_categories = ['pants','jeans','tops','essentials',"outfits yo
# Average ratings for these categories
if 'category' in jcpenney_products.columns and 'average_product_rat
    avg_scores = (jcpenney_products[jcpenney_products['category'].i
        .groupby('category', observed=False)['average_product_ratin
        .mean()
        .reset_index()
        .rename(columns={'average_product_rating': 'Average Rating'}
        .sort_values('Average Rating', ascending=False))
else:
    print("Missing columns: check if 'category' or 'average_product_
    avg_scores = pd.DataFrame()
plt.figure(figsize=(8, 3))
sns.barplot(data=avg_scores, x='Average Rating', y='category', colo
plt.title('Rating of Top categories of Age Groups', fontsize=14, fon
plt.xlabel('Average Rating', fontsize=12, fontweight='bold')
plt.ylabel('Category', fontsize=14)
plt.yticks(fontsize=12, fontweight='bold')
plt.xlim(0, 5)
plt.grid(axis='x', linestyle='--', alpha=0.5)
for index, row in avg_scores.iterrows():
    plt.text(row['Average Rating'] + 0.2, index, f"{row['Average Ra
plt.tight_layout()
plt.savefig('Rating of Top categories of Age Groups', dpi = 300)
plt.show()
```



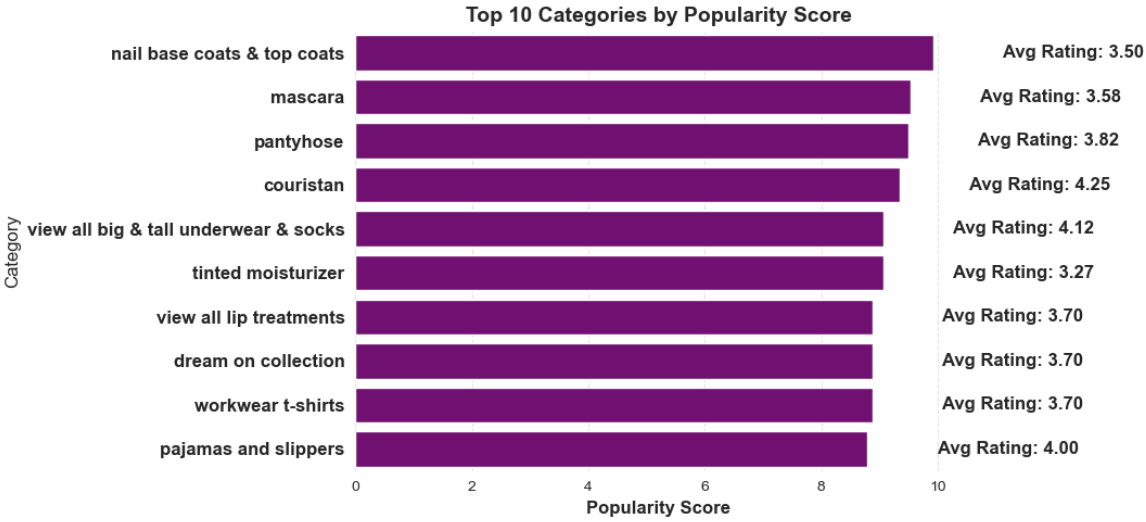
Observation

The most popular product categories, which indicate high demand, have very low ratings. This suggests a problem with product quality that does not meet customers' expectations.

In [16]:
```
# Which categories are statistically high-performing, even with few
import numpy as np
df = jcpenney_products[jcpenney_products['total_number_reviews'] > (
df['popularity_score'] = df['average_product_rating'] * np.log1p(df
category_scores = (df.groupby('category', observed=False).agg({'popu
        'total_number_reviews': 'sum',
        'average_product_rating': 'mean'}).sort_values(by='populari
top_categories = category_scores.head(10).reset_index()
top_categories.rename(columns={'popularity_score': 'Popularity Score
display(top_categories)
```

	category	Popularity Score	Total Reviews	Average Rating
0	nail base coats & top coats	9.916247	16	3.500000
1	mascara	9.522681	27	3.575000
2	pantyhose	9.487825	11	3.818182
3	couristan	9.338204	8	4.250000
4	view all big & tall underwear & socks	9.063551	8	4.125000
5	tinted moisturizer	9.057123	15	3.266667
6	view all lip treatments	8.872213	10	3.700000
7	dream on collection	8.872213	10	3.700000
8	workwear t-shirts	8.872213	10	3.700000
9	pajamas and slippers	8.788898	8	4.000000

In [38]:
```
# Visualisation of Top 10 Categories by Popularity Score
import seaborn as sns
plt.figure(figsize=(11, 5))
sns.barplot(data=top_categories,x='Popularity Score',y='category',co
sns.despine(left=True, bottom=True)
plt.title('Top 10 Categories by Popularity Score', fontsize=14, font
plt.xlabel('Popularity Score', fontsize=12, fontweight='bold')
plt.ylabel('Category', fontsize=12)
plt.yticks(fontweight='bold', fontsize=12)
for i, (score, rating) in enumerate(zip(top_categories['Popularity S
    plt.text(score + 1.2, i, f'Avg Rating: {rating:.2f}', va='cente
plt.grid(axis='x', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.savefig('Top_10_Categories_by_Popularity_Score.png', dpi=300)
plt.show()
```



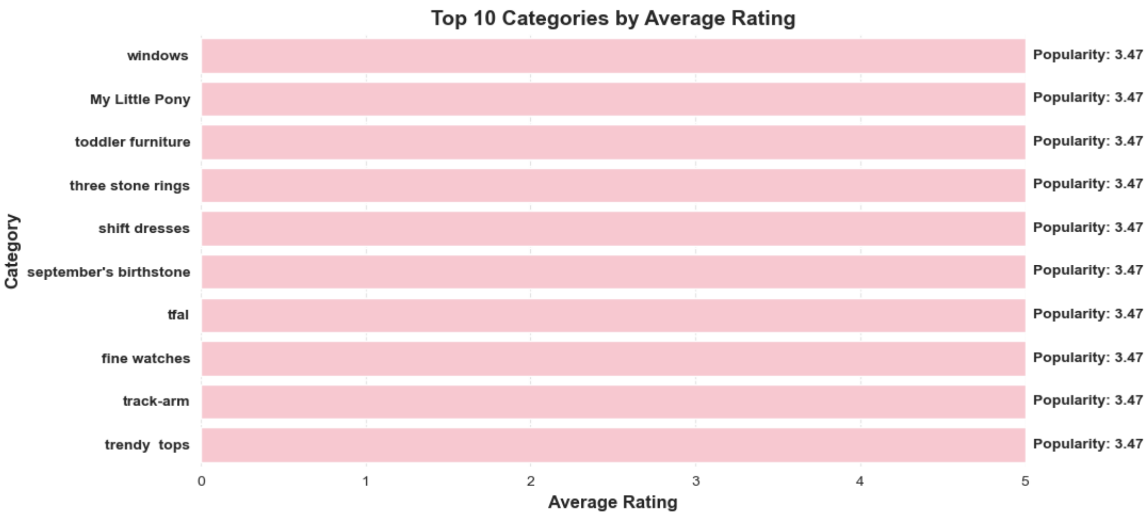
Observation

Although the resulting categories appear statistically high-performing, their relatively low ratings confirm earlier findings that popular products often fail to meet customer quality expectations. This suggests a broader issue with product quality across the range.

In [18]:
```
# Top 10 Categories by Average Rating
df = jcpenney_products.copy()
df = df.dropna(subset=['average_product_rating', 'category'])
ignore_words = ['view all', 'view all brands', 'sale', 'clearance',
df = df[~df['category'].str.lower().isin(ignore_words)]
df['popularity_score'] = df['average_product_rating'] * np.log1p(df
top_categories_by_rating = (
    df.groupby('category', observed=False).agg({
        'average_product_rating': 'mean',
        'total_number_reviews': 'sum',
        'popularity_score': 'mean'}).reset_index())
top_categories_by_rating = top_categories_by_rating.sort_values('av
top_categories_table = top_categories_by_rating.copy()
top_categories_table.rename(columns={'category': 'Category','averag
top_categories_table['Average Rating'] = top_categories_table['Avera
top_categories_table['Popularity Score'] = top_categories_table['Po
top_categories_table = top_categories_table[['Category', 'Average Ra
top_categories_table = top_categories_table.reset_index(drop=True)
top_categories_by_rating = top_categories_by_rating.sort_values(by=
display(top_categories_table)
top_categories_table.to_csv('Top 10 Categories by Average Rating.cs
```

	Category	Average Rating	Popularity Score	Total Reviews
0	windows	5.0	3.47	1
1	My Little Pony	5.0	3.47	1
2	toddler furniture	5.0	3.47	1
3	three stone rings	5.0	3.47	1
4	shift dresses	5.0	3.47	1
5	september's birthstone	5.0	3.47	1
6	tfal	5.0	3.47	1
7	fine watches	5.0	3.47	1
8	track-arm	5.0	3.47	1
9	trendy tops	5.0	3.47	1

In [39]:
```
# Visualisation of the Top 10 Categories by Average Rating
df_plot = top_categories_table.copy()
plt.figure(figsize=(11, 5))
sns.barplot(data=df_plot,x='Average Rating',y='Category',color='pin
sns.despine(left=True, bottom=True)
plt.title('Top 10 Categories by Average Rating', fontsize=14, fontwe
plt.xlabel('Average Rating', fontsize=12, fontweight='bold')
plt.ylabel('Category', fontsize=12, fontweight='bold')
plt.yticks(fontweight='bold')
for i, (rating, score) in enumerate(zip(df_plot['Average Rating'], 
    plt.text(rating + 0.05, i, f'Popularity: {score:.2f}', va='cent
plt.xlim(0, 5)
plt.grid(axis='x', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.savefig('Top_10_Categories_by_Average_Rating.png', dpi=300)
plt.show()
```

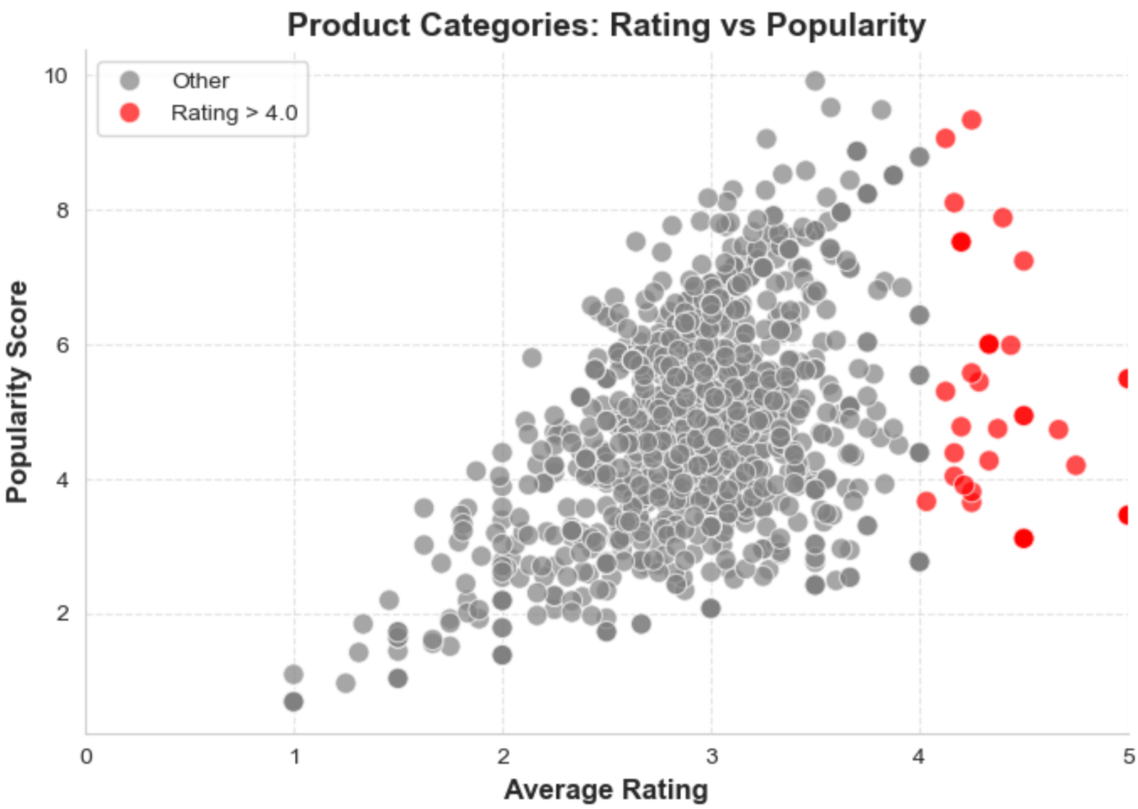


Observation

Although several categories show high average ratings but low popularity scores, this likely reflects limited exposure or niche demand rather than product excellence alone. These items may represent untapped marketing opportunities, suggesting that greater visibility could increase overall sales performance.

In [41]:

```
# Visualisation of High rating categories
df = jcpenney_products[jcpenney_products['total_number_reviews'] > (
df['popularity_score'] = df['average_product_rating'] * np.log1p(df
category_stats = (df.groupby('category', observed=False).agg({
        'average_product_rating': 'mean',
        'total_number_reviews': 'sum',
        'popularity_score': 'mean'}).reset_index())
category_stats['Category Type'] = np.where(category_stats['average_p
plt.figure(figsize=(7, 5))
sns.scatterplot(data=category_stats,x='average_product_rating',y='po
plt.title('Product Categories: Rating vs Popularity', fontsize=14, '
plt.xlabel('Average Rating', fontsize=12, fontweight='bold')
plt.ylabel('Popularity Score', fontsize=12, fontweight='bold')
plt.xlim(0, 5)
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend(title='', loc='upper left')
sns.despine()
plt.tight_layout()
plt.savefig('Rating_vs_Popularity_All_Categories.png', dpi=300)
plt.show()
```



In [23]:

```
# Calculate percentage of categories and products with rating > 4.0
high_rating_count = (category_stats['average_product_rating'] > 4.0
total_categories = len(category_stats)
percentage_high_rating = (high_rating_count / total_categories) * 1(
```

```
print(f"Categories with average rating > 4.0: {high_rating_count} ou
high_rating_products = (df['average_product_rating'] > 4.0).sum()
total_products = len(df)
percentage_high_products = (high_rating_products / total_products) :
print(f"Products with rating > 4.0: {high_rating_products} out of {
```

```
Categories with average rating > 4.0: 70 out of 1169 (5.99%)
Products with rating > 4.0: 628 out of 7964 (7.89%)
```

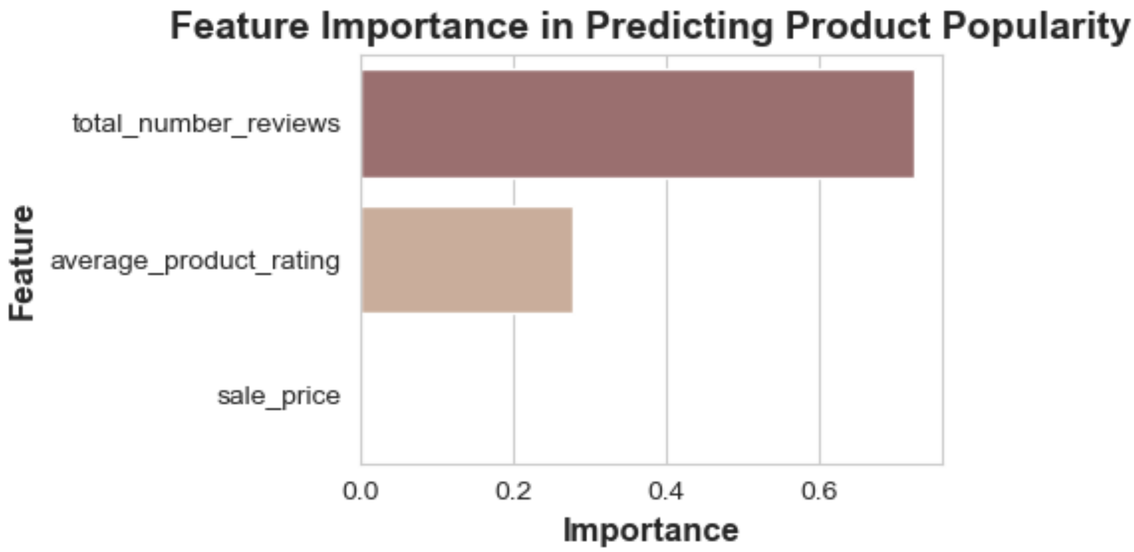
Observation

This scatter plot presents all product categories.
Categories highlighted in red represent those achieving exceptionally high customer satisfaction (average rating above 4.0).
The fact that only a few categories meet this standard underscores a broader challenge in maintaining consistent product quality.
These high-rated categories should be viewed as strategic benchmarks for improving the quality of other products and enhancing overall customer satisfaction.

In [42]:

```
# Forecast of popularity factors
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
df = jcpenney_products.copy()
df = df[df['total_number_reviews'] > 0]
df = df.dropna(subset=['average_product_rating', 'sale_price'])
import re
def clean_price(price):
    if pd.isna(price):
        return np.nan
    price = str(price).replace('$', '').strip()
    if '-' in price:
        parts = re.split(r'[—–]', price)
        parts = [float(p) for p in parts if p.replace('.', '', 1).i
        if len(parts) == 2:
            return np.mean(parts)
    try:
        return float(price)
    except:
        return np.nan
df['sale_price'] = df['sale_price'].apply(clean_price)
df = df.dropna(subset=['sale_price'])
df['popularity_score'] = df['average_product_rating'] * np.log1p(df
X = df[['sale_price', 'average_product_rating', 'total_number_review
y = df['popularity_score']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size:
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)
importances = pd.DataFrame({'Feature': X.columns,'Importance': mode
plt.figure(figsize=(5, 3))
sns.barplot(x='Importance',y='Feature',hue='Feature',          data=
plt.title('Feature Importance in Predicting Product Popularity', fo
plt.xlabel('Importance', fontsize=12, fontweight='bold')
```

```
plt.ylabel('Feature', fontsize=12, fontweight='bold')
plt.tight_layout()
plt.savefig('Forecast of popularity factors.png', dpi=300)
plt.show()
display(importances)
```



	Feature	Importance
2	total_number_reviews	0.724028
1	average_product_rating	0.275901
0	sale_price	0.000070

Observation

The feature importance analysis shows that the number of reviews is the strongest predictor of product popularity, followed by average customer rating, while price has almost no impact.
This indicates that encouraging customers to leave more reviews and maintaining high satisfaction levels will be far more effective for increasing product popularity than changing prices.