Introduction

(a) Project Description

This project aims to identify key trends and insights in **Amazon's fashion hoodies and sweatshirts category for both men's and women's clothing** . By comparing the best-selling products in Amazon's fashion hoodies and sweatshirts, there are some analytic aims we pursue:

- **Brand Dominance**
- **Price Sensitivity**
- **Customer Satisfaction and Review Analysis**
- **Sales Performance**
- **Diversity in Product Options**
- **Origin Analysis**

(b) Summarize findings

1. Compared to women's brands, best sellers have a higher concentration of men's brands. Interestingly, there is no overlap between the top 10 best-selling brands for men and women. It may indicate the specialization trend for brand in Amazon's fashion hoodies and sweatshirts.
2. Concise, informative titles are more necessary. However, titles and descriptions are generally longer for women’s items, clustering near 100 characters, while men’s titles are more varied but shorter.
3. In terms of price, Amazon's best sellers fashion hoodies and sweatshirts are average in about 30 dollars. While men’s products generally have a broader price range, women’s products tend to be more clustered.
4. Compared to men, women's products have more sale volumns. The average monthly sales for women’s products is 639 units, significantly higher than men’s at 170.5 units.
5. Men’s products hold a slight edge in ratings (4.46 vs. 4.33 for women’s). Feedback on men’s items highlights comfort and value, while women’s reviews emphasize color and material quality. Common positive themes include “quality” and “comfort,” with concerns focused on “size” and “material quality.”
6. China is the primary origin for products in both categories, though men’s items exhibit a broader diversity of origins.

Data Collection Process

Address feedback from Project 1 deliverable

Feedback: (a) In your code advise you to separate the code defining functions and executing them, into separate code chunks. This can make things easier for debugging. I think that you could code some of the information in the loop extracting the data as functions so that your code is more concise.

(b) When you display tables of summary statistics it may be more helpful for you to only present numeric variables, so that your results don't display as many NAs.

(a) Revised Code with Modular Functions

Imports, Setup, and Initialization

In [ ]:
```
import logging
from datetime import datetime
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import csv
import time

# Configure logging
logging.basicConfig(
    filename='scraper_log.log',
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
)

# Start logging and note start time
logging.info("Starting the web scraping process.")
start_time = datetime.now()

# Setup Selenium and URLs
options = webdriver.ChromeOptions()
options.headless = False
driver = webdriver.Chrome(options=options)

# URLs for Best Sellers in Men’s and Women’s Hoodies & Sweatshirts
urls = {
    "Men": "https://www.amazon.com/Best-Sellers-Clothing-Shoes-Jewelry-Mens-Fashion-Hoodies-Sweatshirts/zgbs/fashion/12
    "Women": "https://www.amazon.com/Best-Sellers-Clothing-Shoes-Jewelry-Womens-Fashion-Hoodies-Sweatshirts/zgbs/fashio
}

# Define CSV files
```

```
csv_files = {
    "Women": "women_best_sellers_hoodies1104.csv",
    "Men": "men_best_sellers_hoodies1104.csv"
}
header = [
    "rank", "brand_name", "product_title", "product_url", "avg_rating", "num_ratings",
    "sold_last_month", "actual_price", "saving_percent", "sizes", "colors",
    "product_details", "about_this_item", "review_keywords"
]
```

Utility Functions

In [ ]:
```
# Function to write CSV headers
def initialize_csv_files():
    for category, csv_file in csv_files.items():
        with open(csv_file, mode='w', newline='', encoding='utf-8') as file:
            writer = csv.writer(file)
            writer.writerow(header)

# Function to scroll and load items on the page
def load_all_products():
    scroll_pause_time = 4
    incremental_scroll_height = 1000
    items_loaded = 0

    while items_loaded < 50:
        driver.execute_script(f"window.scrollBy(0, {incremental_scroll_height});")
        time.sleep(scroll_pause_time)
        items_loaded = len(driver.find_elements(
            By.XPATH, '//div[@class="a-column a-span12 a-text-center _cDEzb_grid-column_2hIsc"]'
        ))
        logging.info(f"Items loaded: {items_loaded}/50")
        if items_loaded >= 50:
            break

# Function to write product data to CSV
def write_product_data(category, data):
    with open(csv_files[category], mode='a', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        writer.writerow(data)
```

Data Extraction Functions

In [ ]:
```
# Function to extract product details
def extract_product_details(product):
    try:
        rank = product.find_element(By.CSS_SELECTOR, ".zg-bdg-text").text
        product_url = product.find_element(
            By.CSS_SELECTOR, 'a.a-link-normal').get_attribute("href")
        product_title = product.find_element(
            By.CSS_SELECTOR, '._cDEzb_p13n-sc-css-line-clamp-3_g3dy1').text
        avg_rating = product.find_element(
            By.CSS_SELECTOR, 'a[title*="out of 5 stars"]').get_attribute("title").split()[0]
        num_ratings = product.find_element(
            By.CSS_SELECTOR, '.a-size-small').text
        actual_price = product.find_element(
            By.CSS_SELECTOR, "._cDEzb_p13n-sc-price_3mJ9Z").text
        return rank, product_url, product_title, avg_rating, num_ratings, actual_price
    except Exception as e:
        logging.error(f"Error extracting product brief details: {e}")
        return None, None, None, None, None, None

# Function to navigate and get detailed product info
def get_detailed_product_info(product_url):
    try:
        driver.get(product_url)
        WebDriverWait(driver, 5).until(EC.presence_of_element_located((By.ID, "bylineInfo")))
        brand_name = driver.find_element(By.ID, "bylineInfo").text if \
            driver.find_elements(By.ID, "bylineInfo") else None
        sold_last_month = driver.find_element(By.ID, "social-proofing-faceout-title-tk_bought").text.split(' ')[0] if \
            driver.find_elements(By.ID, "social-proofing-faceout-title-tk_bought") else None
        typical_price = driver.find_element(By.CSS_SELECTOR, '.aok-relative .basisPrice .a-offscreen').text if \
            driver.find_elements(By.CSS_SELECTOR, '.aok-relative .basisPrice .a-offscreen') else None
        saving_percent = driver.find_element(By.CSS_SELECTOR, '#corePriceDisplay_desktop_feature_div .a-size-large.a-co
            driver.find_elements(By.CSS_SELECTOR, '#corePriceDisplay_desktop_feature_div .a-size-large.a-color-price.sa

        # Retrieve options for sizes and colors
        sizes = [option.text for option in driver.find_elements(By.XPATH, '//select[@id="native_dropdown_selected_size_
        colors = [img.get_attribute("alt") for img in driver.find_elements(By.XPATH, '//ul[@class="a-unordered-list a-n

        # Extract product details and additional information
        product_details = [{"description": element.find_element(By.XPATH, './/div[@class="a-fixed-left-grid-col a-col-l
                            "value": element.find_element(By.XPATH, './/div[@class="a-fixed-left-grid-col a-col-right"]
                            for element in driver.find_elements(By.XPATH, '//div[@class="a-fixed-left-grid product-fact
        about_this_item = [about.text for about in driver.find_elements(By.XPATH, '//ul[@class="a-unordered-list a-vert
        review_keywords = [kw.get_attribute("data-csa-c-item-id")
                           for kw in driver.find_elements(By.XPATH, '//div[@class="a-section a-spacing-small a-spacing-

        return brand_name, sold_last_month, typical_price, saving_percent, sizes, colors, product_details, about_this_i

    except Exception as e:
        logging.error(f"Error extracting product brief details: {e}")
```

```
        return None, None, None, None, None, None, None, None, None
```

Main Scraping Function

In [ ]:
```
# Main scraping function
def scrape_category(category, url):
    try:
        driver.get(url)
        time.sleep(3)
        logging.info(f"Accessing {category} category at {url}")
    except Exception as e:
        logging.error(f"Failed to load {url}: {e}")
        return

    # Loop through pages within the current category
    for page_num in range(2):
        logging.info(f"Scraping {category} - Page {page_num + 1}/2")

        # Ensure all items on page are loaded
        load_all_products()

        # Find product containers on the loaded page
        list_results = driver.find_elements(
            By.XPATH, '//div[@class="a-column a-span12 a-text-center _cDEzb_grid-column_2hIsc"]'
        )
        logging.info(f"Found {len(list_results)} products on {category} - Page {page_num + 1}/2")

        for product in tqdm(list_results, desc=f"{category} - Page {page_num + 1}/2"):
            rank, product_url, product_title, avg_rating, num_ratings, actual_price = extract_product_details(product)
```

```
            if product_url:
                brand_name, sold_last_month, typical_price, saving_percent, sizes, colors, product_details, about_this_

                # Return to the previous page for the next product
                driver.back()

                product_data = [rank, brand_name, product_title, product_url, avg_rating, num_ratings,
                                sold_last_month, actual_price, saving_percent, sizes, colors,
                                product_details, about_this_item, review_keywords]
                write_product_data(category, product_data)

        if page_num < 1:
            try:
                next_page_button = driver.find_element(By.XPATH, '//li[@class="a-last"]/a')
                next_page_button.click()
                time.sleep(5)
            except NoSuchElementException:
                logging.warning("Next page button not found. Ending pagination.")
                break
```

Execute the Scraping

In [ ]:
```
# Execution
initialize_csv_files()
for category, url in urls.items():
    scrape_category(category, url)

# Log end time and calculate duration
end_time = datetime.now()
duration = end_time - start_time
logging.info(f"Scraping completed. Total duration: {duration}")
print(f"Scraping started at: {start_time.strftime('%Y-%m-%d %H:%M:%S')}")
print(f"Scraping completed at: {end_time.strftime('%Y-%m-%d %H:%M:%S')}")
print(f"Total duration: {duration}")

# Close the browser
driver.quit()
```

```
Men - Page 1/2: 100%|██████████| 50/50 [04:00<00:00,  4.82s/it]
Men - Page 2/2: 100%|██████████| 50/50 [03:59<00:00,  4.80s/it]
Women - Page 1/2: 100%|██████████| 50/50 [03:36<00:00,  4.33s/it]
Women - Page 2/2: 100%|██████████| 50/50 [05:00<00:00,  6.01s/it]
Scraping started at: 2024-11-04 18:13:48
Scraping completed at: 2024-11-04 19:46:41
Total duration: 1:32:52.893738
```

(b) Generate a statistical summary table for numeric columns only

In [ ]:
```
# Import the project1 dataset
import pandas as pd
pre_cleaned_df = pd.read_csv("hoodies_gendered_bestsellers_.csv")

# Generate the statistical summary table for numeric columns only
numeric_cols = pre_cleaned_df.select_dtypes(include=['float64', 'int64'])

summary_table = pd.DataFrame({
    'Observations': [len(pre_cleaned_df)] * numeric_cols.shape[1],
    'Missing Values': numeric_cols.isnull().sum(),
    'Data Type': numeric_cols.dtypes,
    'Mean': numeric_cols.mean(),
    'Variance': numeric_cols.var()
```

```
})

# Display the summary table
print("Summary of Observations, Missing Data, Mean, and Variance for Numeric Variables:")
display(summary_table)
```

```
Summary of Observations, Missing Data, Mean, and Variance for Numeric Variables:
```

	Observations	Missing Values	Data Type	Mean	Variance
rank	200	0	int64	50.500000	8.374372e+02
actual_price	200	1	float64	31.025477	1.716855e+02
avg_rating	200	0	float64	4.392000	3.832764e-02
num_ratings	200	0	int64	4242.555000	6.883901e+07
sold_last_month	200	24	float64	459.943182	8.420577e+05

Data Analysis

1. Data Cleaning

1. **Create new columns**:
   - `title_length` : The length of the `product_title` text.
   - `description_length` : The character length of `about_this_item` .
   - `country_of_origin` : Extracted from `product_details` when "Country of Origin" is available.
2. **`saving_percent`** : Converted entries to numeric values, replacing non-numeric and missing values with 0.
3. **`review_keywords`** : Parsed and categorized the keywords by sentiment, creating three new columns:
   - `positive_keywords` : strings of positive_keywords
   - `negative_keywords` : strings of negative_keywords
   - `neutral_keywords` : strings of neutral_keywords

In [ ]:
```
import pandas as pd
import ast

# Load the dataset
df = pd.read_csv("hoodies_gendered_bestsellers_.csv")

# 1. Add new columns
df['title_length'] = df['product_title'].fillna('').apply(len)  # Length of `product_title`
df['description_length'] = df['about_this_item'].fillna('').apply(len)  # Length of `about_this_item`

# Extract `country_of_origin` from `product_details` if available
def extract_country(details):
    try:
        details = ast.literal_eval(details)
    except (ValueError, SyntaxError):
        return None
    for item in details:
        if isinstance(item, dict) and item.get('description') == 'Country of Origin':
            return item.get('value')
    return None

df['country_of_origin'] = df['product_details'].apply(extract_country)

# 2. Convert `saving_percent` to numeric, replace missing values with 0
df['saving_percent'] = df['saving_percent'].str.replace('%', '').astype(float).fillna(0)

# 3. Parse `review_keywords` by sentiment
def parse_keywords(keywords):
    try:
        keywords_list = ast.literal_eval(keywords)
    except (ValueError, SyntaxError):
        return '', '', ''
    pos = ', '.join([kw.replace('_POSITIVE', '') for kw in keywords_list if '_POSITIVE' in kw])
    neg = ', '.join([kw.replace('_NEGATIVE', '') for kw in keywords_list if '_NEGATIVE' in kw])
    neu = ', '.join([kw.replace('_NEUTRAL', '') for kw in keywords_list if '_NEUTRAL' in kw])
    return pos, neg, neu

df['positive_keywords'], df['negative_keywords'], df['neutral_keywords'] = zip(*df['review_keywords'].apply(parse_keywo


# Save the cleaned data with target columns
cleaned_df = df[['category', 'rank', 'brand_name', 'title_length', 'actual_price', 'avg_rating', 'num_ratings', 'sold_l
    'saving_percent', 'sizes', 'colors',  'description_length',
    'country_of_origin', 'positive_keywords', 'negative_keywords', 'neutral_keywords']]
display(cleaned_df)
```

	category	rank	brand_name	title_length	actual_price	avg_rating	num_ratings	sold_last_month	saving_percent	sizes	
0	Men	1	Carhartt	73	64.99	4.8	21409	1000.0	0.0	['Select', 'Small', 'Medium', 'Large', 'Large ...	['Bl H H
1	Men	2	Carhartt	65	41.99	4.8	45265	200.0	-24.0	['Select', 'X-Small', 'Small', 'Medium', 'Larg...	Heat
2	Men	3	Amazon Essentials	72	29.40	4.5	52390	300.0	0.0	['Select', 'X-Small', 'Small', 'Medium', 'Larg...	He
3	Men	4	JMIERR	110	34.99	4.3	407	1000.0	0.0	['Select', 'Small', 'Medium', 'Large', 'X-Larg...	[' 'P
4	Men	5	Carhartt	61	59.99	4.8	24184	1000.0	0.0	['Select', 'Small', 'Medium', 'Large', 'Large ...	['Bl He G
...	...	...	...	...	...	...	...	...	...	...	
195	Women	96	AURUZA	121	14.99	4.4	73	100.0	-25.0	['Select', 'Small', 'Medium', 'Large', 'X-Larg...	['Bl blacl
196	Women	97	Eutten	123	26.99	4.2	289	700.0	-25.0	['Select', 'Small', 'Medium', 'Large', 'X-Large']	['Ap Gr '
197	Women	98	Fisoew	120	19.99	3.9	334	100.0	0.0	['Select', 'Small', 'Medium', 'Large', 'X-Larg...	['I 'Bro
198	Women	99	Dokotoo	114	29.99	4.6	22	NaN	0.0	['Select', 'Small', 'Medium', 'Large', 'X-Large']	['E 'Cof
199	Women	100	StunShow	115	31.91	4.3	446	100.0	-20.0	['Select', 'Small', 'Medium', 'Large', 'X-Larg...	['A Brov

200 rows × 16 columns

2. Descriptive Statistics

Results (a) Table to present information. This should include tables of summary statistics, number of observations.

In [ ]:
```
# Number of observations
num_observations = cleaned_df.shape[0]
print(f"Number of observations: {num_observations}")

# Summary statistics for numerical columns
print("Summary statistics table for numerical columns:")
display(cleaned_df.describe())
```

```
Number of observations: 200
Summary statistics table for numerical columns:
```

	rank	title_length	actual_price	avg_rating	num_ratings	sold_last_month	saving_percent	description_length
count	200.000000	200.000000	199.000000	200.000000	200.000000	176.000000	200.000000	200.000000
mean	50.500000	95.320000	31.025477	4.392000	4242.555000	459.943182	-14.710000	528.235000
std	28.938507	25.257023	13.102881	0.195774	8296.927614	917.637026	16.640325	165.502766
min	1.000000	0.000000	12.490000	3.600000	15.000000	50.000000	-61.000000	2.000000
25%	25.750000	80.000000	23.990000	4.300000	403.500000	100.000000	-27.000000	439.500000
50%	50.500000	102.000000	28.980000	4.400000	1295.000000	200.000000	-9.000000	544.000000
75%	75.250000	116.000000	32.990000	4.500000	3813.000000	400.000000	0.000000	639.000000
max	100.000000	125.000000	109.990000	4.800000	53999.000000	9000.000000	0.000000	933.000000

3. Further Analyses

Results (b) Figures to present information, which include description of any categorical or numeric data. There should be a high focus on interpreting insights from the data.

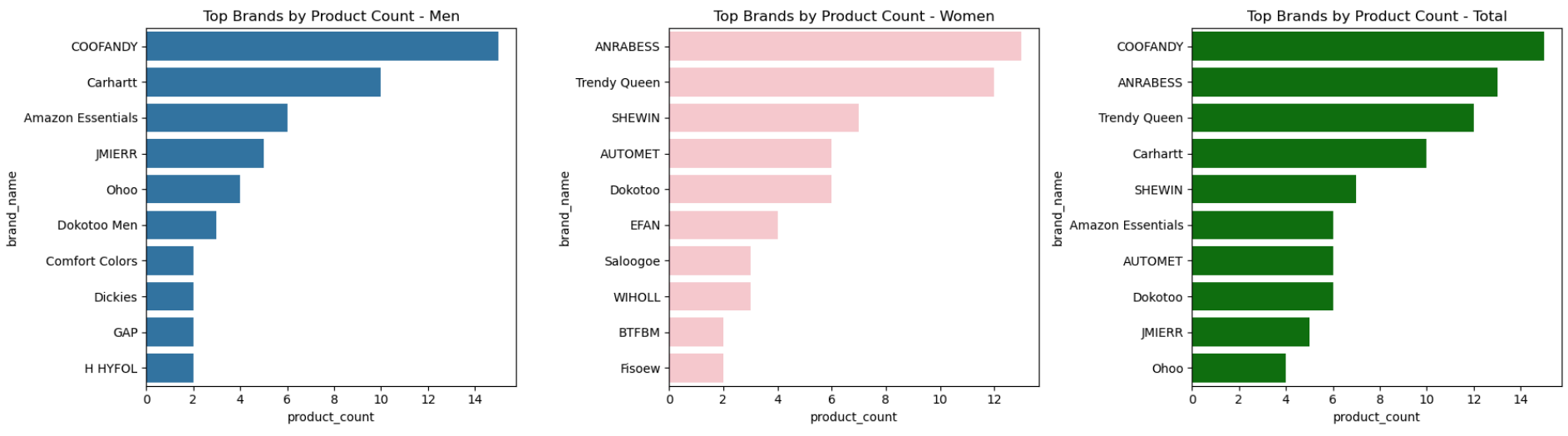
1. Brand Dominance:

- Count brand occurrences and plot top brands by product count for each category(men / women / total rank).

Insight:

1. Compared to women's brands, best sellers have a higher concentration of men's brands.
2. Interestingly, there is no overlap between the top 10 best-selling brands for men and women. It may indicate the specialization trend for brand in Amazon's fashion hoodies and sweatshirts.

In [ ]:
```python
import matplotlib.pyplot as plt
import seaborn as sns

def brand_dominance_analysis(df):
    # Count brand occurrences
    brand_counts = df.groupby(['category', 'brand_name']).size().reset_index(name='product_count')
    # Filter top brands by product count
    top_brands_men = brand_counts[brand_counts['category'] == 'Men'].nlargest(10, 'product_count')
    top_brands_women = brand_counts[brand_counts['category'] == 'Women'].nlargest(10, 'product_count')
    top_brands_total = brand_counts.nlargest(10, 'product_count')

    # Plotting top brands by category
    fig, axes = plt.subplots(1, 3, figsize=(18, 5))
    sns.barplot(x='product_count', y='brand_name', data=top_brands_men, ax=axes[0])
    axes[0].set_title('Top Brands by Product Count - Men')
    sns.barplot(x='product_count', y='brand_name', data=top_brands_women, ax=axes[1], color="pink")
    axes[1].set_title('Top Brands by Product Count - Women')
    sns.barplot(x='product_count', y='brand_name', data=top_brands_total, ax=axes[2], color='green')
    axes[2].set_title('Top Brands by Product Count - Total')
    plt.tight_layout()
    plt.show()

brand_dominance_analysis(cleaned_df)
```



2. Title and Description Analysis

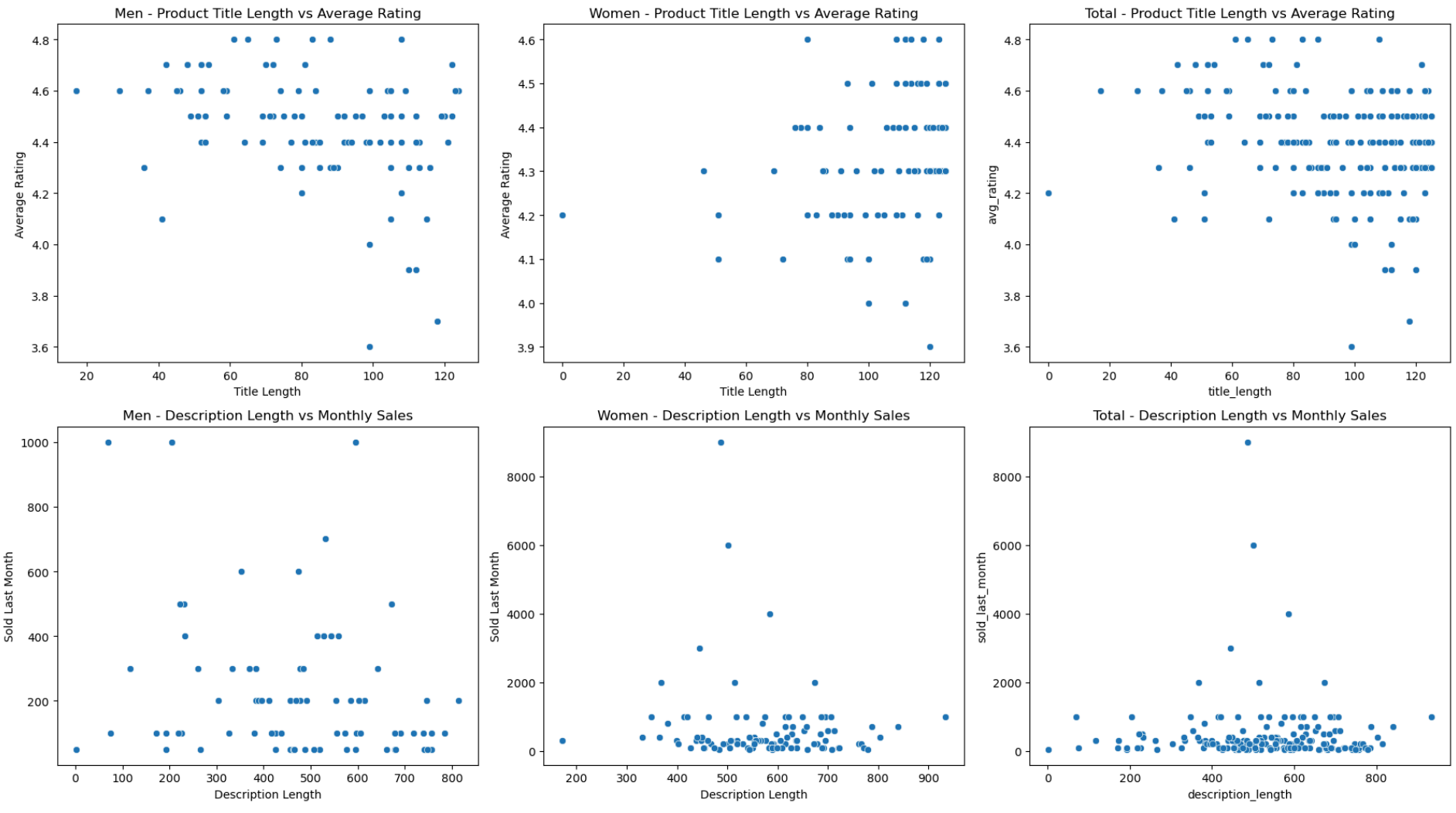
- Based on character length for `product_title` and `about_this_item`, analyze its relationship with ratings and sales.

Insights:

1. On average, titles are 95 characters long(not exclude the brands name), and descriptions are around 528 characters across both categories.
2. Titles and descriptions are generally longer for women’s items, clustering near 100 characters, while men’s titles are more varied but shorter.
3. Concise, informative titles are more necessary. We can infer this from the graph showing Title length might not have significant correlation with customer ratings across categories.

In [ ]:
```python
def title_description_analysis_by_category(df):
    # Scatter plot for title length vs average rating by category
    fig, axes = plt.subplots(2, 3, figsize=(18, 10))
    categories = ['Men', 'Women']
```

```
    for i, category in enumerate(categories):
        # Title Length vs Average Rating by Category
        sns.scatterplot(x='title_length', y='avg_rating', data=df[df['category'] == category], ax=axes[0, i])
        axes[0, i].set_title(f'{category} - Product Title Length vs Average Rating')
        axes[0, i].set_xlabel('Title Length')
        axes[0, i].set_ylabel('Average Rating')

        # Description Length vs Monthly Sales by Category
        sns.scatterplot(x='description_length', y='sold_last_month', data=df[df['category'] == category], ax=axes[1, i])
        axes[1, i].set_title(f'{category} - Description Length vs Monthly Sales')
        axes[1, i].set_xlabel('Description Length')
        axes[1, i].set_ylabel('Sold Last Month')

    # Combined Plot for all data (total)
    sns.scatterplot(x='title_length', y='avg_rating', data=df, ax=axes[0, 2])
    axes[0, 2].set_title('Total - Product Title Length vs Average Rating')
    sns.scatterplot(x='description_length', y='sold_last_month', data=df, ax=axes[1, 2])
    axes[1, 2].set_title('Total - Description Length vs Monthly Sales')

    plt.tight_layout()
    plt.show()

# Run title and description analysis
title_description_analysis_by_category(cleaned_df)
```



3. Price Sensitivity

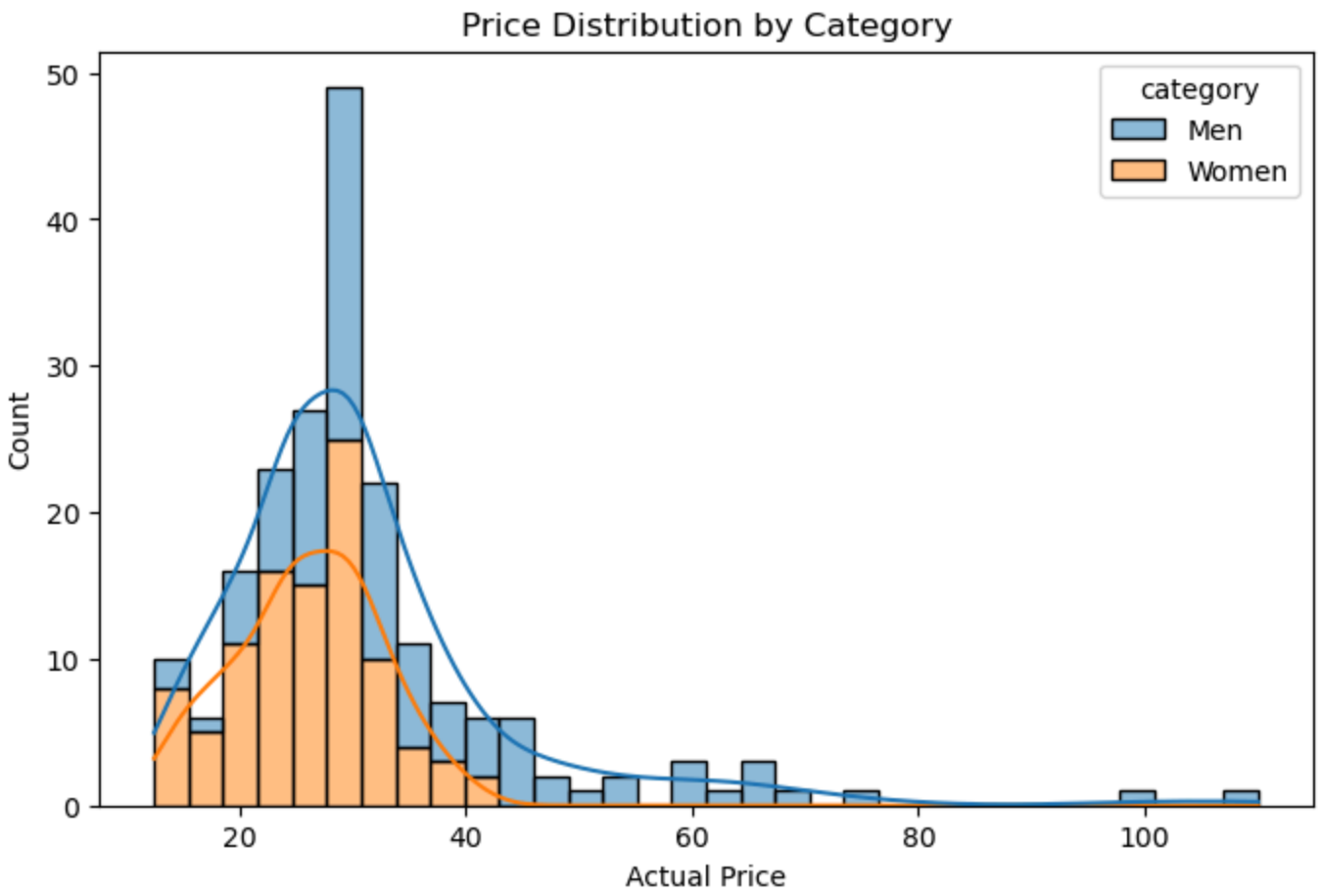
- **Price Distribution**: Create histograms of `actual_price` for each category to visualize price ranges.

Insights:

1. Men's products generally have a broader price range, while women's products tend to be clustered at specific price points.
2. The average discount (saving percent) for top-ranked products is -17.13%, compared to -12.29% for lower-ranked products. Top-ranked items tend to have slightly higher discounts, potentially contributing to their popularity.

In [ ]:

```
def price_sensitivity_analysis(df):
    # Price Distribution Histogram by Category
    plt.figure(figsize=(8, 5))
    sns.histplot(data=df, x='actual_price', hue='category', multiple='stack', kde=True)
    plt.title('Price Distribution by Category')
    plt.xlabel('Actual Price')
    plt.show()


# Run price sensitivity analysis with separate plots
price_sensitivity_analysis(cleaned_df)
```



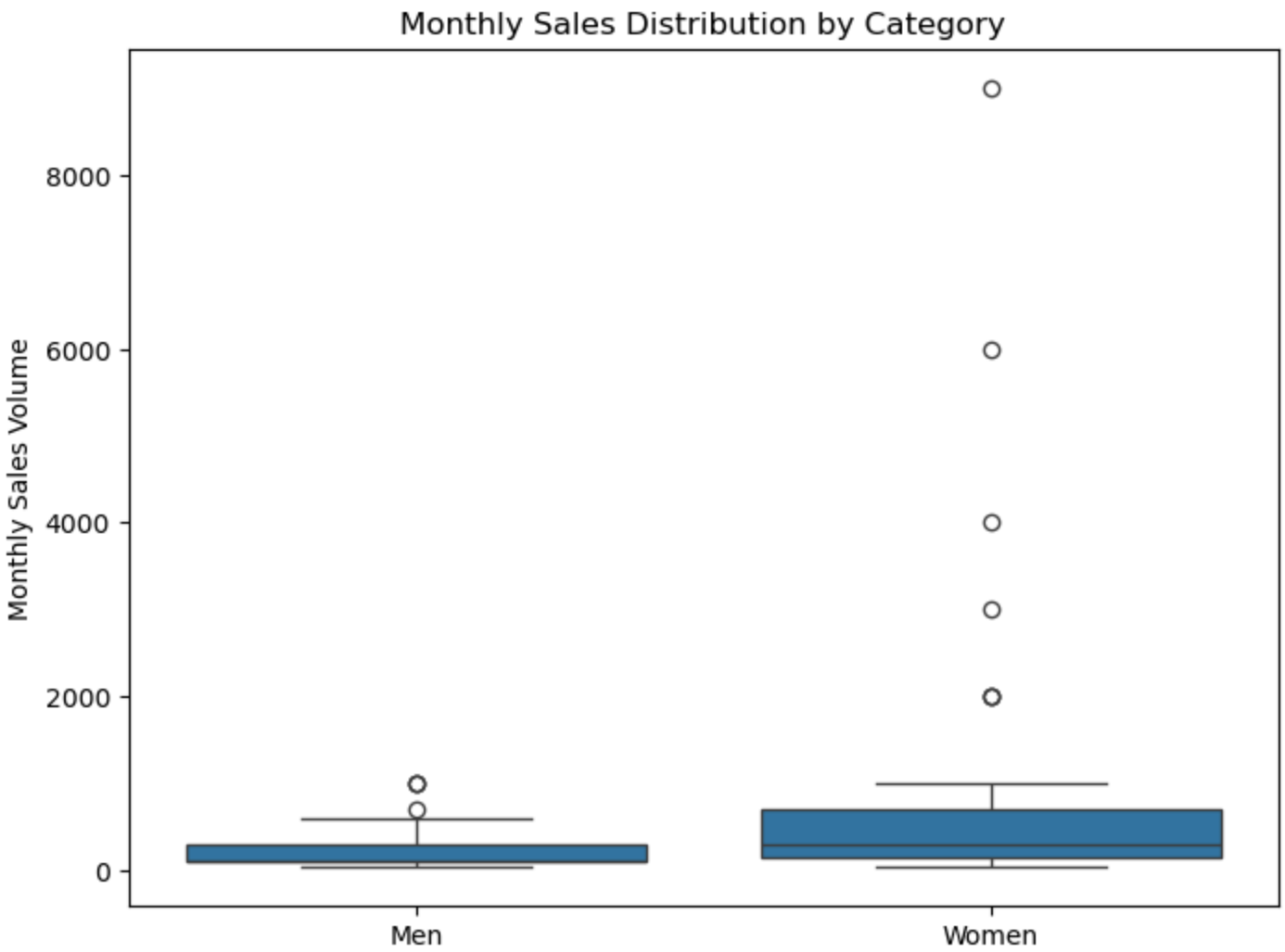
4. Sales Performance

- **Sales Volume Analysis**: Plot against `rank` to understand if higher rankings correlate with monthly sales volume.
- **Category Comparison**: Use box plots to compare average monthly sales across categories, brands, and price tiers.

Insights:

1. The scatter plot reveals that higher-ranked products tend to have higher sales volumes, with more consistency in the men's category. In contrast, women's products show greater variation in sales volume across ranks.
2. The average monthly sales for women's products is 639 units, significantly higher than men's at 170.5 units. Overall, the total average monthly sales across categories is 404.75 units.

In [ ]:
```python
def sales_performance_analysis(df):
    # Scatter plot for Sales Volume vs Rank to visualize correlation by category
    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=df, x='rank', y='sold_last_month', hue='category')
    plt.title('Sales Volume vs Rank by Category')
    plt.xlabel('Rank')
    plt.ylabel('Monthly Sales Volume')
    plt.show()

    # Box Plot for Monthly Sales across Categories
    plt.figure(figsize=(8, 6))
    sns.boxplot(data=df, x='category', y='sold_last_month')
    plt.title('Monthly Sales Distribution by Category')
    plt.xlabel('Category')
    plt.ylabel('Monthly Sales Volume')
    plt.show()

    # Calculate and compare average monthly sales by category and total
    avg_sales_men = df[df['category'] == 'Men']['sold_last_month'].mean()
    avg_sales_women = df[df['category'] == 'Women']['sold_last_month'].mean()
    avg_sales_total = df['sold_last_month'].mean()

    return avg_sales_men, avg_sales_women, avg_sales_total

# Run sales performance analysis
avg_sales_men, avg_sales_women, avg_sales_total = sales_performance_analysis(cleaned_df)
avg_sales_men, avg_sales_women, avg_sales_total
```



Out[ ]: (221.42857142857142, 645.4545454545455, 459.9431818181818)

5. Customer Satisfaction and Customer Review Analysis

- **Rating Analysis**: Plot the distribution of `avg_rating` to identify products with high customer satisfaction. Analyze keywords from `review_keywords` to categorize feedback.
- **Word Cloud Visualization**: Create word clouds for frequently mentioned review aspects for each category.

Insights:

1. Average Rating: Men's products score an average of 4.46, slightly above women's at 4.33, with an overall average of 4.39.
2. Functional vs. Aesthetic Focus: Men's feedback emphasizes comfort and value, while women's feedback shows a preference for color and material quality.
3. Common Themes: Across both categories, positive feedback often mentions "quality," "fit," and "comfort," while negative feedback frequently includes "size issues" and "material quality."

   A. Top Positive Keywords:

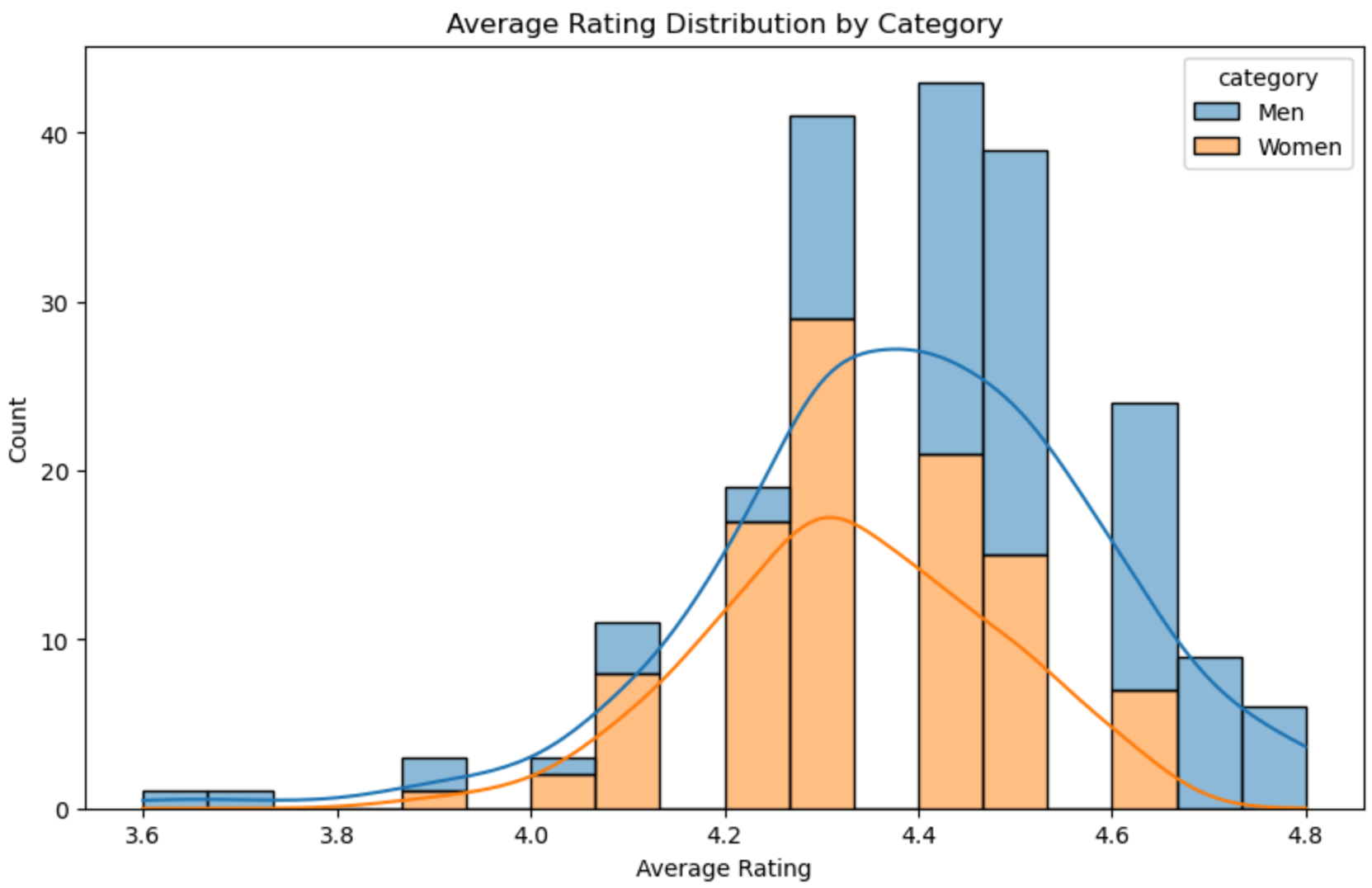
   - Men: "Comfort," "Value," "Warmth," and "Fit" emphasize practical attributes.
   - Women: "Warmth," "Color," "Material," and "Softness" highlight aesthetic and sensory appeal.

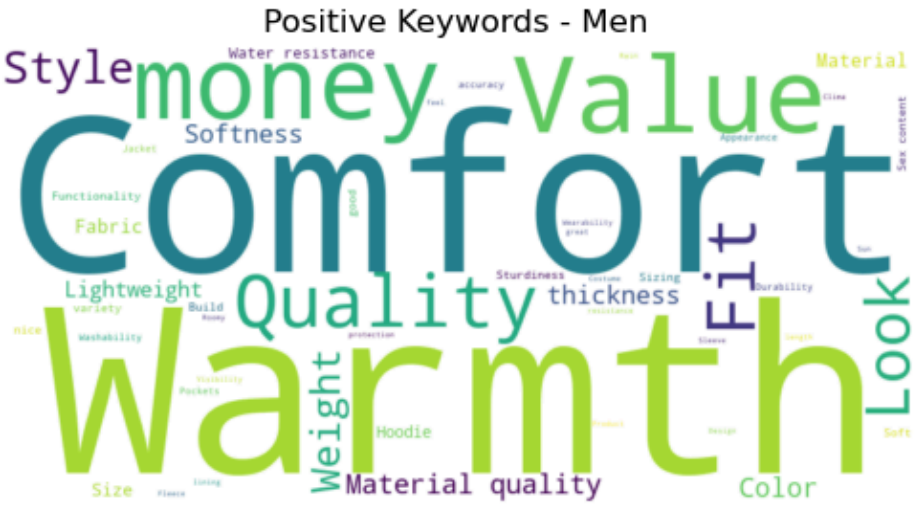
   B. Top Negative Keywords:

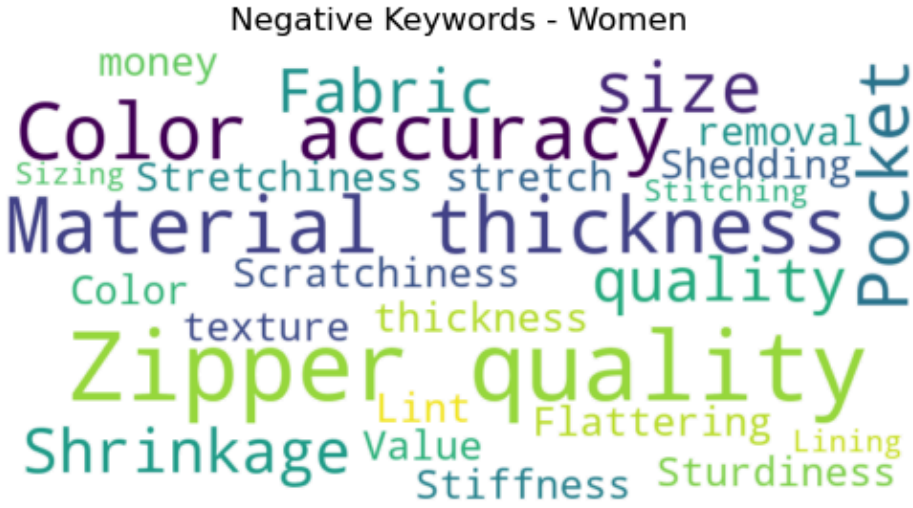
   - Men: "Zipper," "quality," and "Fabric" reflect concerns over specific product features.

- Women: “Zipper,” “quality,” “Color,” and “Material accuracy” point to fabric and color issues.

In [ ]:
```
from wordcloud import WordCloud

def customer_satisfaction_analysis(df):
    # Rating Analysis: Distribution of `avg_rating` by category
    plt.figure(figsize=(10, 6))
    sns.histplot(data=df, x='avg_rating', hue='category', multiple='stack', kde=True)
    plt.title('Average Rating Distribution by Category')
    plt.xlabel('Average Rating')
    plt.show()


def gendered_sentimental_wordclouds(df):
    # Prepare positive, negative, and neutral keywords separately for men and women
    pos_keywords_men = " ".join(df[(df['category'] == 'Men') & df['positive_keywords'].notna()]['positive_keywords'])
    pos_keywords_women = " ".join(df[(df['category'] == 'Women') & df['positive_keywords'].notna()]['positive_keywords'
    neg_keywords_men = " ".join(df[(df['category'] == 'Men') & df['negative_keywords'].notna()]['negative_keywords'])
    neg_keywords_women = " ".join(df[(df['category'] == 'Women') & df['negative_keywords'].notna()]['negative_keywords'

    # Generate and display word clouds for gendered and sentimental categories
    fig, axes = plt.subplots(2, 2, figsize=(10, 10))

    # Positive Word Cloud - Men
    axes[0, 0].imshow(WordCloud(width=800, height=400, background_color='white').generate(pos_keywords_men), interpolat
    axes[0, 0].axis('off')
    axes[0, 0].set_title("Positive Keywords - Men")

    # Positive Word Cloud - Women
    axes[0, 1].imshow(WordCloud(width=800, height=400, background_color='white').generate(pos_keywords_women), interpol
    axes[0, 1].axis('off')
    axes[0, 1].set_title("Positive Keywords - Women")

    # Negative Word Cloud - Men
    axes[1, 0].imshow(WordCloud(width=800, height=400, background_color='white').generate(neg_keywords_men), interpolat
    axes[1, 0].axis('off')
    axes[1, 0].set_title("Negative Keywords - Men")

    # Negative Word Cloud - Women
    axes[1, 1].imshow(WordCloud(width=800, height=400, background_color='white').generate(neg_keywords_women), interpol
    axes[1, 1].axis('off')
    axes[1, 1].set_title("Negative Keywords - Women")

    plt.tight_layout()
    plt.show()

# Run the gendered and sentimental word cloud visualization
customer_satisfaction_analysis(cleaned_df)
gendered_sentimental_wordclouds(cleaned_df)
```







6. Origin Analysis

- Compare origin distributions across categories

Insights:

1. China leads as the most common origin for products in both categories.
2. Men's products show greater diversity in country of origin, while women's items are more concentrated in a few key origin.

In [ ]:
```python
import plotly.express as px

def visualize_origin_distribution_by_category(df):
    # Count occurrences of each country of origin within each category
    origin_category_counts = df.groupby(['country_of_origin', 'category']).size().reset_index(name='count')

    # Create a bar plot using Plotly to visualize the distribution by category and origin
    fig = px.bar(
        origin_category_counts,
        x='country_of_origin',
        y='count',
        color='category',
        title="Distribution of Product Origins by Category",
        labels={'country_of_origin': 'Country of Origin', 'count': 'Product Count'},
        barmode='group'
    )

    fig.update_layout(xaxis_title="Country of Origin", yaxis_title="Product Count")
    fig.show()

# Run the visualization for origin distribution by category
visualize_origin_distribution_by_category(cleaned_df)
```

Discussion

1. We did not drop missing values in the title_name and actual_price columns because the number of missing entries was minimal, and these rows contained other relevant descriptive and outcome information, which did not impact the main analyses.
2. The absence of origin information for some products may have limited the depth of origin-based insights in our comparisons.
3. Since we collected the data from Amazon's best seller rank, future analyses could explore differences in product descriptions and performance metrics (e.g., sales, reviews) across ranking clustered levels.