```python
import pandas as pd

# Re-loading the Excel file and its sheets due to kernel interruption
file_path = '/content/Combined_Data.xlsx'
xls = pd.ExcelFile(file_path)

# Load the sheets into DataFrames
item_review_df = pd.read_excel(xls, 'Item_review')
user_review_df = pd.read_excel(xls, 'User_review')
fashion_retail_df = pd.read_excel(xls, 'Fashion_retail')

# Filtering reviews related to sustainability
sustainable_keywords = ['sustainable', 'eco', 'organic', 'recycled',
'environment']
pattern = '|'.join(sustainable_keywords)

# Extracting relevant reviews from User_review sheet
sustainable_reviews_df =
user_review_df[user_review_df['review_text'].str.contains(pattern,
case=False, na=False) |

user_review_df['review_summary'].str.contains(pattern, case=False,
na=False)]
```

```
----------------------------------------------------------------------
-----
ModuleNotFoundError                       Traceback (most recent call
last)
<ipython-input-2-ae1e33be544f> in <cell line: 18>()
     16
user_review_df['review_summary'].str.contains(pattern, case=False,
na=False)]
     17
---> 18 import ace_tools as tools;
tools.display_dataframe_to_user(name="Sustainable Reviews Data",
dataframe=sustainable_reviews_df)
     19
     20 sustainable_reviews_df.head()

ModuleNotFoundError: No module named 'ace_tools'

----------------------------------------------------------------------
-----
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
----------------------------------------------------------------------
```

```
-----



sustainable_reviews_df.head()
```

{"repr_error":"0","type":"dataframe","variable_name":"sustainable_reviews_df"}

```
import pandas as pd

# Load the Excel file


# Load necessary columns from User_review sheet
user_review_df = pd.read_excel(xls, 'User_review',
usecols=['review_text', 'review_summary'])

# Define keywords related to sustainability
sustainable_keywords = ['sustainable', 'eco', 'organic', 'recycled',
'environment']
pattern = '|'.join(sustainable_keywords)

# Extract relevant reviews
sustainable_reviews_df =
user_review_df[user_review_df['review_text'].str.contains(pattern,
case=False, na=False) |

user_review_df['review_summary'].str.contains(pattern, case=False,
na=False)]

# Display the filtered data
print(sustainable_reviews_df.head())

# Identify popular materials and styles (example for further analysis)
# This part is highly customizable based on specific materials and
styles you want to track
materials_keywords = ['cotton', 'linen', 'hemp', 'bamboo', 'recycled
polyester']
materials_pattern = '|'.join(materials_keywords)

styles_keywords = ['bohemian', 'minimalist', 'vintage', 'modern',
'classic']
styles_pattern = '|'.join(styles_keywords)

popular_materials_df =
sustainable_reviews_df[sustainable_reviews_df['review_text'].str.conta
ins(materials_pattern, case=False, na=False) |

sustainable_reviews_df['review_summary'].str.contains(materials_patter
n, case=False, na=False)]
```

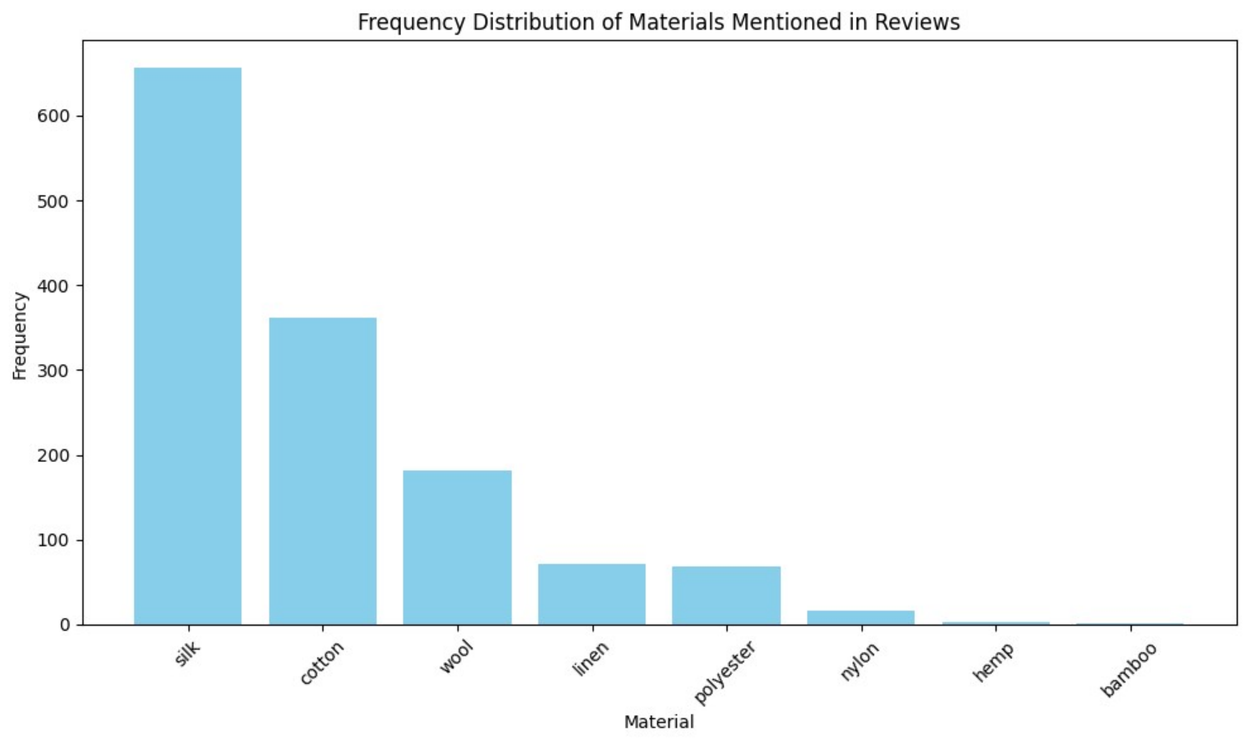
```
popular_styles_df =
sustainable_reviews_df[sustainable_reviews_df['review_text'].str.conta
ins(styles_pattern, case=False, na=False) |

sustainable_reviews_df['review_summary'].str.contains(styles_pattern,
case=False, na=False)]

# Display popular materials and styles
print("Popular Materials:\n", popular_materials_df.head())
print("Popular Styles:\n", popular_styles_df.head())

# Analyze common sustainable practices
sustainable_practices_keywords = ['reuse', 'reduce', 'recycle',
'upcycle', 'ethical', 'fair trade']
practices_pattern = '|'.join(sustainable_practices_keywords)

sustainable_practices_df =
sustainable_reviews_df[sustainable_reviews_df['review_text'].str.conta
ins(practices_pattern, case=False, na=False) |

sustainable_reviews_df['review_summary'].str.contains(practices_patter
n, case=False, na=False)]

# Display sustainable practices
print("Common Sustainable Practices:\n",
sustainable_practices_df.head())

                                          review_text  \
3   I rented this for my company's black tie award...
4   I have always been petite in my upper body and...
13  I ordered this dress as a replacement because ...
20  I rented this dress for a spring wedding. I ha...
24  It fit perfectly - I have a small waist, norma...

                                       review_summary
3    Dress arrived on time and in perfect condition.
4                        Was in love with this dress !!!
13                   Ordered this as a replacement dress
20                          Perfect for a spring wedding!
24  Great dress for a wedding with long ceremonies...
Popular Materials:
                                            review_text  \
1055  I wore this to the office and to a daytime bru...
7111  typically btwn a 2 and a 4, got the 2 based on...
7388  I rented this to wear to a concert and loved i...
7659  This was a cool item but didn't work for me.  ...
9953  I love the print of this shirt! I wore this to...

                review_summary
```

```
1055       So many compliments
7111      cute but SIZE DOWN!!
7388  A great piece! Loved it!
7659     Lots of stiff fabric.
9953           TEAM RTR REVIEW
Popular Styles:
                                                review_text  \
193   I'm not sure if I kept the dresses in the bag ...
201       Comfy, cute, and classic. I would recommend.
873   The dress is extremely comfortable, and it fit...
912   I enjoyed wearing this royal blue dress out to...
1137  I felt like a billion bucks! Loved this dress....

                                          review_summary
193   The cut of the back is beautiful and makes the...
201                                   Such a fun print. \n
873   This dress is hands down my favorite outfit I ...
912                                    Royal Blue Classic!
1137                   Fantastic, classic yet edgy dress!
Common Sustainable Practices:
                                                review_text  \
19300   Wore this dress for my husbands office Christm...
44265   The fit was wonderful - fit like a glove and h...
47527   I wore this dress for a wedding.  The wedding ...
134698  I wore this to a wedding in Philadelphia and p...
163970  Love the halter neckline on this dress! The de...

                               review_summary
19300                Got compliments all night!
44265                                 Love it!
47527   A lovely dress for a more formal affair
134698    Beautiful color, tons of compliments!
163970                              Stylist Review

import pandas as pd
from collections import Counter


# Load necessary columns from User_review sheet
user_review_df = pd.read_excel(xls, 'User_review',
usecols=['review_text', 'review_summary'])

# Define keywords related to materials
materials_keywords = ['cotton', 'linen', 'hemp', 'bamboo', 'recycled
polyester', 'silk', 'wool', 'nylon', 'polyester']
pattern = '|'.join(materials_keywords)

# Extract relevant reviews
sustainable_reviews_df =
user_review_df[user_review_df['review_text'].str.contains(pattern,
case=False, na=False) |

user_review_df['review_summary'].str.contains(pattern, case=False,
na=False)]

# Function to count keyword mentions
def count_mentions(text_series, keywords):
    counter = Counter()
    for text in text_series.dropna():
        words = text.lower().split()
        for keyword in keywords:
            if keyword in words:
                counter[keyword] += 1
    return counter

# Count mentions in review_text and review_summary
mentions_review_text =
count_mentions(sustainable_reviews_df['review_text'],
materials_keywords)
mentions_review_summary =
count_mentions(sustainable_reviews_df['review_summary'],
materials_keywords)

# Combine counts from both columns
total_mentions = mentions_review_text + mentions_review_summary

# Convert to DataFrame for better visualization
mentions_df = pd.DataFrame.from_dict(total_mentions, orient='index',
columns=['count']).reset_index()
mentions_df = mentions_df.rename(columns={'index':
'material'}).sort_values(by='count', ascending=False)

# Display the results
print(mentions_df)

# Save the result to a CSV file (optional)
mentions_df.to_csv('material_mentions.csv', index=False)
```

```
    material  count
1       silk    656
0     cotton    361
2       wool    181
4      linen     71
3  polyester     69
6      nylon     16
7       hemp      3
5     bamboo      2
```

```
mentions_df.head()
```

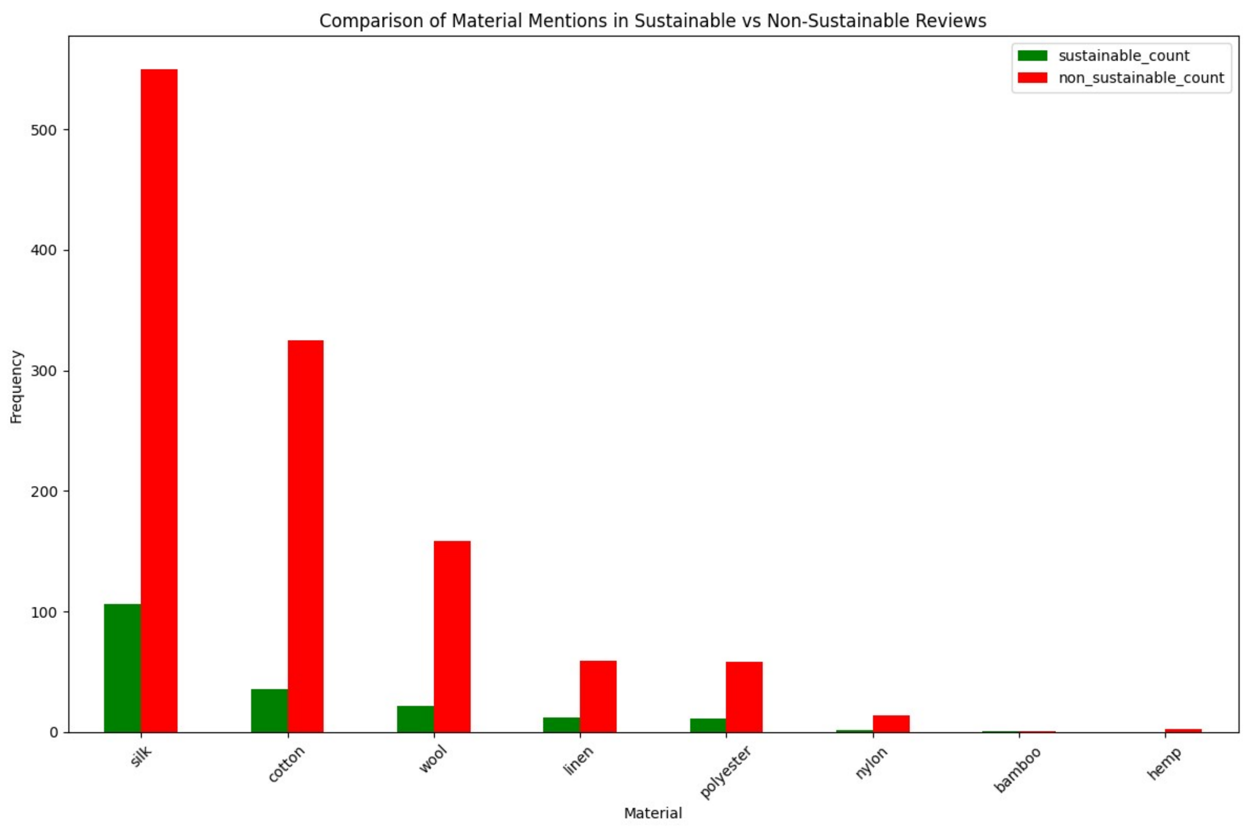
```python
# Plotting the frequency distribution
plt.figure(figsize=(10, 6))
plt.bar(mentions_df['material'], mentions_df['count'],
color='skyblue')
plt.xlabel('Material')
plt.ylabel('Frequency')
plt.title('Frequency Distribution of Materials Mentioned in Reviews')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Display the DataFrame
```



```python
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter



# Load necessary columns from User_review sheet
user_review_df = pd.read_excel(xls, 'User_review',
usecols=['review_text', 'review_summary'])

# Define keywords related to sustainability and materials
sustainable_keywords = ['sustainable', 'eco', 'organic', 'recycled',
'environment']
```

```
materials_keywords = ['cotton', 'linen', 'hemp', 'bamboo', 'recycled
polyester', 'silk', 'wool', 'nylon', 'polyester']

sustainable_pattern = '|'.join(sustainable_keywords)
materials_pattern = '|'.join(materials_keywords)

# Extract relevant reviews for sustainability
sustainable_reviews_df =
user_review_df[user_review_df['review_text'].str.contains(sustainable_
pattern, case=False, na=False) |

user_review_df['review_summary'].str.contains(sustainable_pattern,
case=False, na=False)]

# Extract relevant reviews for materials
material_reviews_df =
user_review_df[user_review_df['review_text'].str.contains(materials_pa
ttern, case=False, na=False) |

user_review_df['review_summary'].str.contains(materials_pattern,
case=False, na=False)]

# Function to count keyword mentions
def count_mentions(text_series, keywords):
    counter = Counter()
    for text in text_series.dropna():
        text = str(text)  # Ensure text is a string
        words = text.lower().split()
        for keyword in keywords:
            if keyword in words:
                counter[keyword] += 1
    return counter

# Count mentions in sustainable reviews
mentions_sustainable_reviews_text =
count_mentions(sustainable_reviews_df['review_text'],
materials_keywords)
mentions_sustainable_reviews_summary =
count_mentions(sustainable_reviews_df['review_summary'],
materials_keywords)

# Count mentions in non-sustainable reviews
non_sustainable_reviews_df =
material_reviews_df[~material_reviews_df.index.isin(sustainable_review
s_df.index)]
mentions_non_sustainable_reviews_text =
count_mentions(non_sustainable_reviews_df['review_text'],
materials_keywords)
mentions_non_sustainable_reviews_summary =
count_mentions(non_sustainable_reviews_df['review_summary'],
```

```python
materials_keywords)

# Combine counts
total_mentions_sustainable = mentions_sustainable_reviews_text +
mentions_sustainable_reviews_summary
total_mentions_non_sustainable = mentions_non_sustainable_reviews_text
+ mentions_non_sustainable_reviews_summary

# Convert to DataFrame for better visualization
sustainable_mentions_df =
pd.DataFrame.from_dict(total_mentions_sustainable, orient='index',
columns=['sustainable_count']).reset_index()
sustainable_mentions_df =
sustainable_mentions_df.rename(columns={'index':
'material'}).sort_values(by='sustainable_count', ascending=False)

non_sustainable_mentions_df =
pd.DataFrame.from_dict(total_mentions_non_sustainable, orient='index',
columns=['non_sustainable_count']).reset_index()
non_sustainable_mentions_df =
non_sustainable_mentions_df.rename(columns={'index':
'material'}).sort_values(by='non_sustainable_count', ascending=False)

# Merge the DataFrames for comparison
comparison_df = pd.merge(sustainable_mentions_df,
non_sustainable_mentions_df, on='material', how='outer').fillna(0)

# Plotting the frequency distribution
comparison_df.plot(kind='bar', x='material', figsize=(12, 8),
color=['green', 'red'])
plt.xlabel('Material')
plt.ylabel('Frequency')
plt.title('Comparison of Material Mentions in Sustainable vs Non-
Sustainable Reviews')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Display the comparison DataFrame
print(comparison_df)

# Save the result to a CSV file (optional)
comparison_df.to_csv('sustainable_vs_non_sustainable_mentions.csv',
index=False)
```



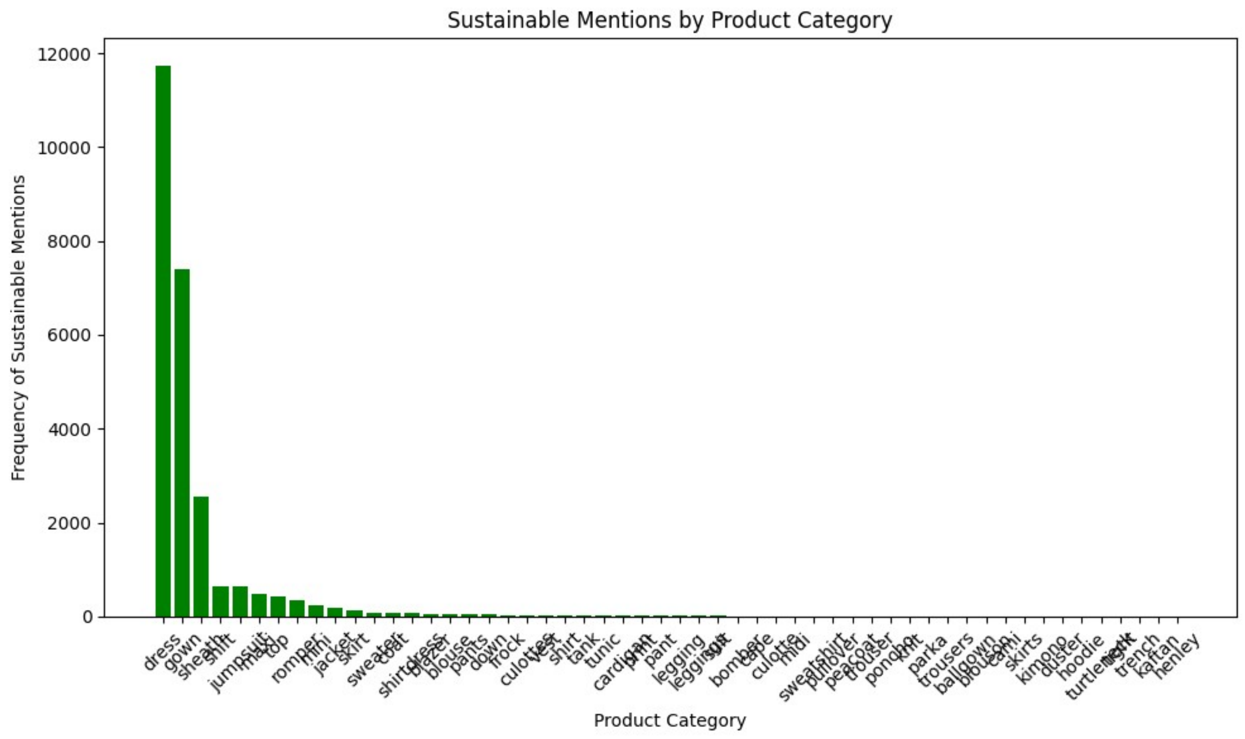
```
    material  sustainable_count  non_sustainable_count
0       silk              106.0                    550
1     cotton               36.0                    325
2       wool               22.0                    159
3      linen               12.0                     59
4  polyester               11.0                     58
5      nylon                2.0                     14
6     bamboo                1.0                      1
7       hemp                0.0                      3
```

```
comparison_df.head()
```

{"summary":"{\n \"name\": \"comparison_df\",\n \"rows\": 8,\n \"fields\": [\n   {\n   \"column\": \"material\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 8,\n   \"samples\": [\n   \"cotton\",\n   \"nylon\",\n   \"silk\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"sustainable_count\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 35.402784394128425,\n   \"min\": 0.0,\n   \"max\": 106.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   36.0,\n   2.0,\n   106.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"non_sustainable_count\",\n   \"properties\": {\n

\"dtype\": \"number\",\n   \"std\": 196,\n   \"min\": 1,\n   \"max\": 550,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   325,\n   14,\n   550\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"comparison_df"}

```
import pandas as pd
import matplotlib.pyplot as plt



# Load necessary columns from User_review sheet
user_review_df = pd.read_excel(xls, 'User_review',
usecols=['review_text', 'review_summary', 'category'])

# Define keywords related to sustainability
sustainable_keywords = ['sustainable', 'eco', 'organic', 'recycled',
'environment']
sustainable_pattern = '|'.join(sustainable_keywords)

# Extract relevant reviews for sustainability
sustainable_reviews_df =
user_review_df[user_review_df['review_text'].str.contains(sustainable_
pattern, case=False, na=False) |

user_review_df['review_summary'].str.contains(sustainable_pattern,
case=False, na=False)]

# Count mentions by product category
category_counts =
sustainable_reviews_df['category'].value_counts().reset_index()
category_counts.columns = ['category', 'count']

# Plotting the frequency distribution by category
plt.figure(figsize=(10, 6))
plt.bar(category_counts['category'], category_counts['count'],
color='green')
plt.xlabel('Product Category')
plt.ylabel('Frequency of Sustainable Mentions')
plt.title('Sustainable Mentions by Product Category')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Display the category counts DataFrame
print(category_counts)

# Save the result to a CSV file (optional)
category_counts.to_csv('sustainable_mentions_by_category.csv',
index=False)
```



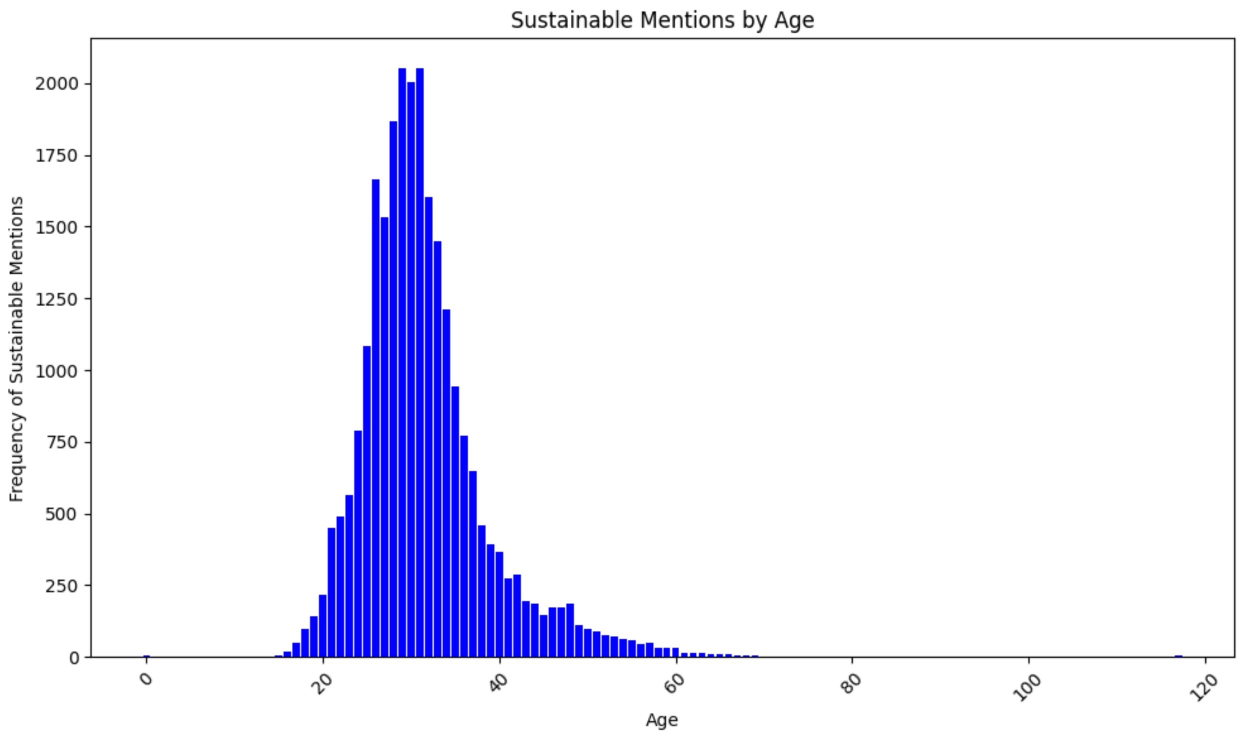
```
category_counts.head()
```

```
{"summary":"{\n  \"name\": \"category_counts\",\n  \"rows\": 54,\n
\"fields\": [\n    {\n      \"column\": \"category\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 54,\n        \"samples\": [\n
\"culottes\",\n          \"turtleneck\",\n          \"hoodie\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"count\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 1885,\n
\"min\": 1,\n        \"max\": 11730,\n        \"num_unique_values\":
34,\n        \"samples\": [\n          45,\n          24,\n
8\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"category_counts"}
```

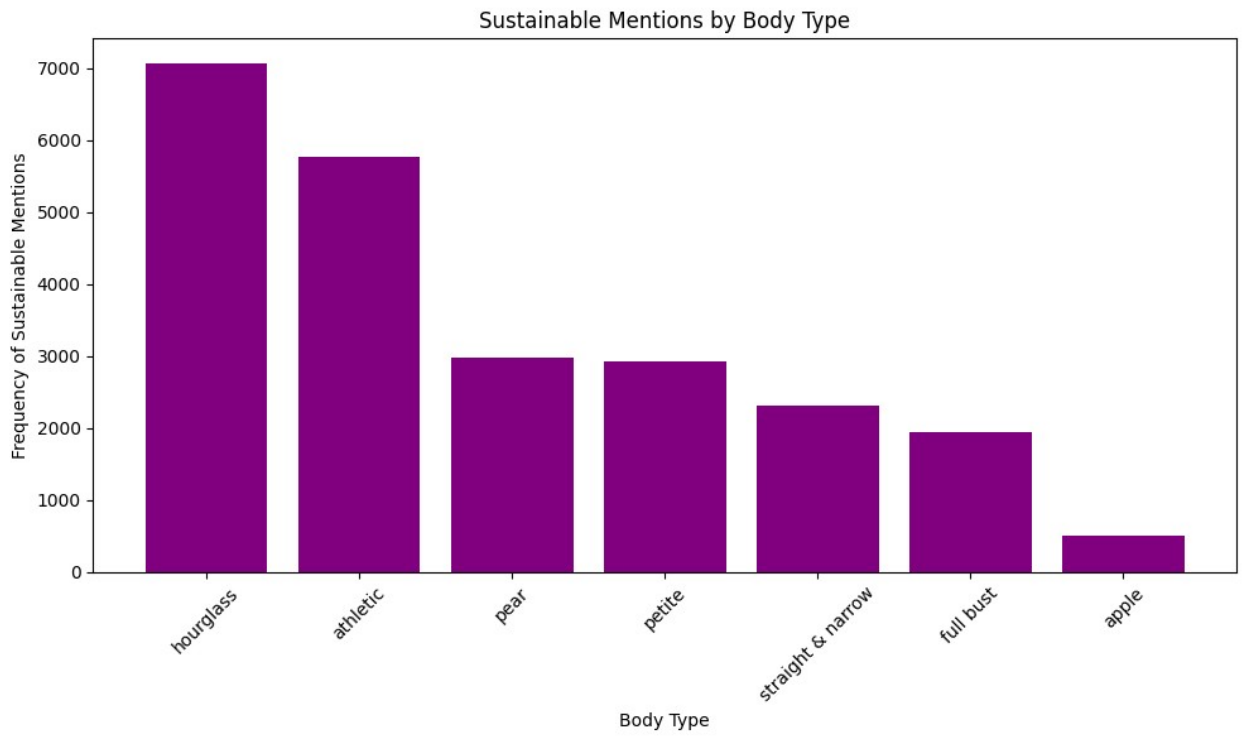
```
import pandas as pd
import matplotlib.pyplot as plt


# Load necessary columns from User_review sheet
user_review_df = pd.read_excel(xls, 'User_review',
usecols=['review_text', 'review_summary', 'age', 'body type'])

# Define keywords related to sustainability
sustainable_keywords = ['sustainable', 'eco', 'organic', 'recycled',
'environment']
```

```
sustainable_pattern = '|'.join(sustainable_keywords)

# Extract relevant reviews for sustainability
sustainable_reviews_df =
user_review_df[user_review_df['review_text'].str.contains(sustainable_
pattern, case=False, na=False) |

user_review_df['review_summary'].str.contains(sustainable_pattern,
case=False, na=False)]

# Analyze by age
age_distribution =
sustainable_reviews_df['age'].value_counts().reset_index()
age_distribution.columns = ['age', 'count']

# Plotting the frequency distribution by age
plt.figure(figsize=(10, 6))
plt.bar(age_distribution['age'], age_distribution['count'],
color='blue')
plt.xlabel('Age')
plt.ylabel('Frequency of Sustainable Mentions')
plt.title('Sustainable Mentions by Age')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Display the age distribution DataFrame
print("Sustainable Mentions by Age")
print(age_distribution)

# Save the result to a CSV file (optional)
age_distribution.to_csv('sustainable_mentions_by_age.csv',
index=False)

# Analyze by body type
body_type_distribution = sustainable_reviews_df['body
type'].value_counts().reset_index()
body_type_distribution.columns = ['body type', 'count']

# Plotting the frequency distribution by body type
plt.figure(figsize=(10, 6))
plt.bar(body_type_distribution['body type'],
body_type_distribution['count'], color='purple')
plt.xlabel('Body Type')
plt.ylabel('Frequency of Sustainable Mentions')
plt.title('Sustainable Mentions by Body Type')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
# Display the body type distribution DataFrame
print("Sustainable Mentions by Body Type")
print(body_type_distribution)

# Save the result to a CSV file (optional)
body_type_distribution.to_csv('sustainable_mentions_by_body_type.csv',
index=False)
```



```
Sustainable Mentions by Age
     age  count
0   31.0   2054
1   29.0   2051
2   30.0   2004
3   28.0   1869
4   26.0   1663
..   ...    ...
57  70.0      2
58  14.0      2
59   9.0      1
60  72.0      1
61   3.0      1

[62 rows x 2 columns]
```



```
Sustainable Mentions by Body Type
           body type  count
0          hourglass   7054
1           athletic   5760
2               pear   2978
3             petite   2917
4  straight & narrow   2303
5          full bust   1943
6              apple    512

import pandas as pd
from textblob import TextBlob
import matplotlib.pyplot as plt

# Load the Excel file

# Load necessary columns from the sheets
item_review_df = pd.read_excel(xls, 'Item_review',
usecols=['review_summary', 'review_text', 'user_id'])
user_review_df = pd.read_excel(xls, 'User_review',
usecols=['review_text', 'review_summary', 'user_id'])
fashion_retail_df = pd.read_excel(xls, 'Fashion_retail',
usecols=['Customer Reference ID', 'Item Purchased', 'Purchase Amount
(USD)', 'Review Rating', 'Payment Method'])

# Define keywords related to sustainability
sustainable_keywords = ['sustainable', 'eco', 'organic', 'recycled',
'environment']
sustainable_pattern = '|'.join(sustainable_keywords)

# Filter sustainable reviews from both sheets
sustainable_reviews_item =
item_review_df[item_review_df['review_text'].str.contains(sustainable_
pattern, case=False, na=False) |

item_review_df['review_summary'].str.contains(sustainable_pattern,
case=False, na=False)]
sustainable_reviews_user =
user_review_df[user_review_df['review_text'].str.contains(sustainable_
pattern, case=False, na=False) |

user_review_df['review_summary'].str.contains(sustainable_pattern,
case=False, na=False)]

# Combine sustainable reviews
sustainable_reviews = pd.concat([sustainable_reviews_item,
sustainable_reviews_user])

# Display the combined sustainable reviews
sustainable_reviews.head()
```

```
{"summary":"{\n  \"name\": \"sustainable_reviews\",\n  \"rows\":
29393,\n  \"fields\": [\n    {\n      \"column\": \"user_id\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
289881,\n        \"min\": 9,\n        \"max\": 999987,\n
\"num_unique_values\": 23440,\n        \"samples\": [\n
150945,\n          736319,\n          978667\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"review_summary\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 24925,\n        \"samples\": [\n          \"The
fit and class of the dress was a knock out.\",\n          \"this is an
amazing produc\",\n          \"This dress was so perfect for a
Halloween wedding, masquerade ball theme!\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"review_text\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 29307,\n        \"samples\": [\n
\"Flirty and fun! This is such a unique dress by AMUR; the entire
design is seriously one of a kind. The length of the dress mixed with
the exaggerated sleeves and laced neckline with open shoulders isn\\
u00e2\\u20ac\\u2122t something you see everyday. It runs true to size
and doesn\\u00e2\\u20ac\\u2122t stretch, so I definitely recommend
going a size up. I\\u00e2\\u20ac\\u2122m a true 6 and that fit me
perfectly. The bust area is super comfortable, so it\\u00e2\\u20ac\\
u2122s perfect for any cup size, however I do recommend wearing a
strapless bra with it. Paired with nude or black heels, this is your
```

ideal dress to a dinner, date, or party! \",\n   \"Fits like a glove. Both sizes I ordered 2R and 4R fit perfect but I liked it to hug the curves a bit more so I went with the 2. Length was great with 4\\\" heels. The sequin do rub your skin some if you're sitting back in a chair for long. You can't go wrong with this, but I do recommend spanx, as I have a pretty small stomach but you could still see a little pooch! Lol otherwise gorgeous dress and I was sad to take it off. Nude strap heels look best with it for sure!\",\n   \"I love this cardi. It's more dark mustard coloured though than in the picture. It is really soft and quiet warm. The small fits perfect, the medium would have been baggy on me. I can do up the buttons but they pull just a tiny bit, nothing out of the ordinary though. The sleeves come to about mid forearm but can be pulled down a tiny bit of pushed up. It is very light weight. However it did come with a bit of a pulled thread in one of the sleeves but I can poke that back in with a needle no worries. Overall I am very happy with my purchase and would recommend it.\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"sustainable_reviews"}

```python
# Function to analyze sentiment
def get_sentiment(text):
    blob = TextBlob(str(text))
    return blob.sentiment.polarity

# Apply sentiment analysis
sustainable_reviews['sentiment'] =
sustainable_reviews['review_text'].apply(get_sentiment)

# Display sentiment analysis results
sustainable_reviews[['review_text', 'sentiment']].head()
```

{"summary":"{\n \"name\": \"sustainable_reviews[['review_text', 'sentiment']]\",\n \"rows\": 5,\n \"fields\": [\n   {\n   \"column\": \"review_text\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 5,\n   \"samples\": [\n   \"This has become one of my favorite dresses. So flattering and the color is beautiful. I'm 5'1 and it falls barely below the knee. Machine washable. Love that.\",\n   \"I feel absolutely gorgeous in this dress. I bought it for work, and I love it. The color is a lovely, rich emerald green color. I can move around in this dress as it has a bit of stretch. The pleats sort of camouflage any food babies I might have going on while still flattering the rest of my figure. The fabric is soft, though I would recommend wearing a slip or something similar to hide underwear lines. I like wearing it with silver jewelry and black Tstrap heels.There are a couple reasons why I wouldn't give this 5/5 stars: There needs to be a zipper. It's a bit of a struggle to get this dress on and off, especially if I don't want deodorant or makeup smudging on it. I think this needs to be hand wash only. There are a few pills on it now sooner than I thought there would be. I would actually buy this dress again if it came with a hidden side zipper.\",\n   \"According to the size chart I should have ordered a small, but I read other reviews and sized up to a medium and I'm glad I did. The dress fits snuggly but not too tight. The color is true to the picture, deep and rich.   I've received multiple compliments on the dress. Definitely recommend this purchase.\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"sentiment\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.1995310123509061,\n   \"min\": 0.07460317460317460,\n   \"max\": 0.47500000000000003,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   0.47500000000000003,\n   0.11089743589743592,\n   0.07460317460317460\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

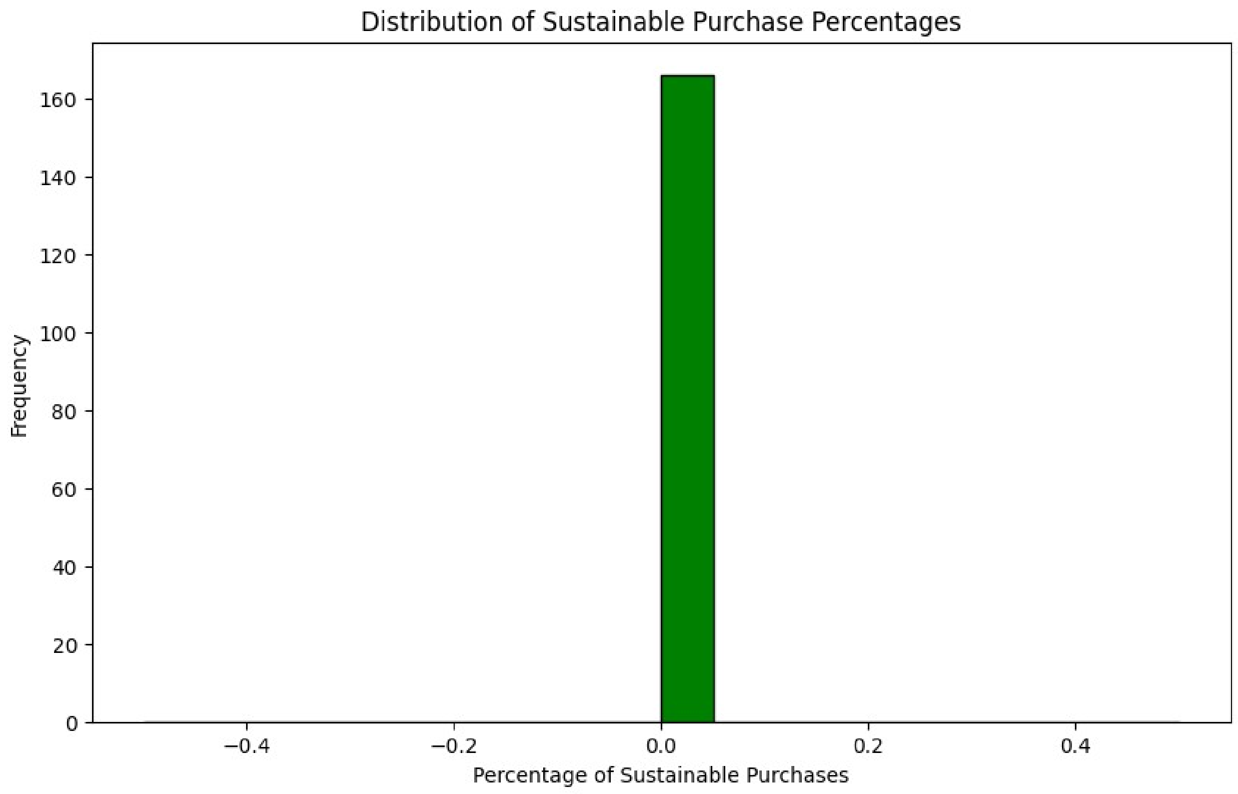
```python
# Rename columns for merging
sustainable_reviews = sustainable_reviews.rename(columns={'user_id': 'Customer Reference ID'})

# Merge sustainable reviews with fashion retail data
merged_df = pd.merge(sustainable_reviews, fashion_retail_df, on='Customer Reference ID', how='inner')

# Display the merged DataFrame
merged_df.head()
```

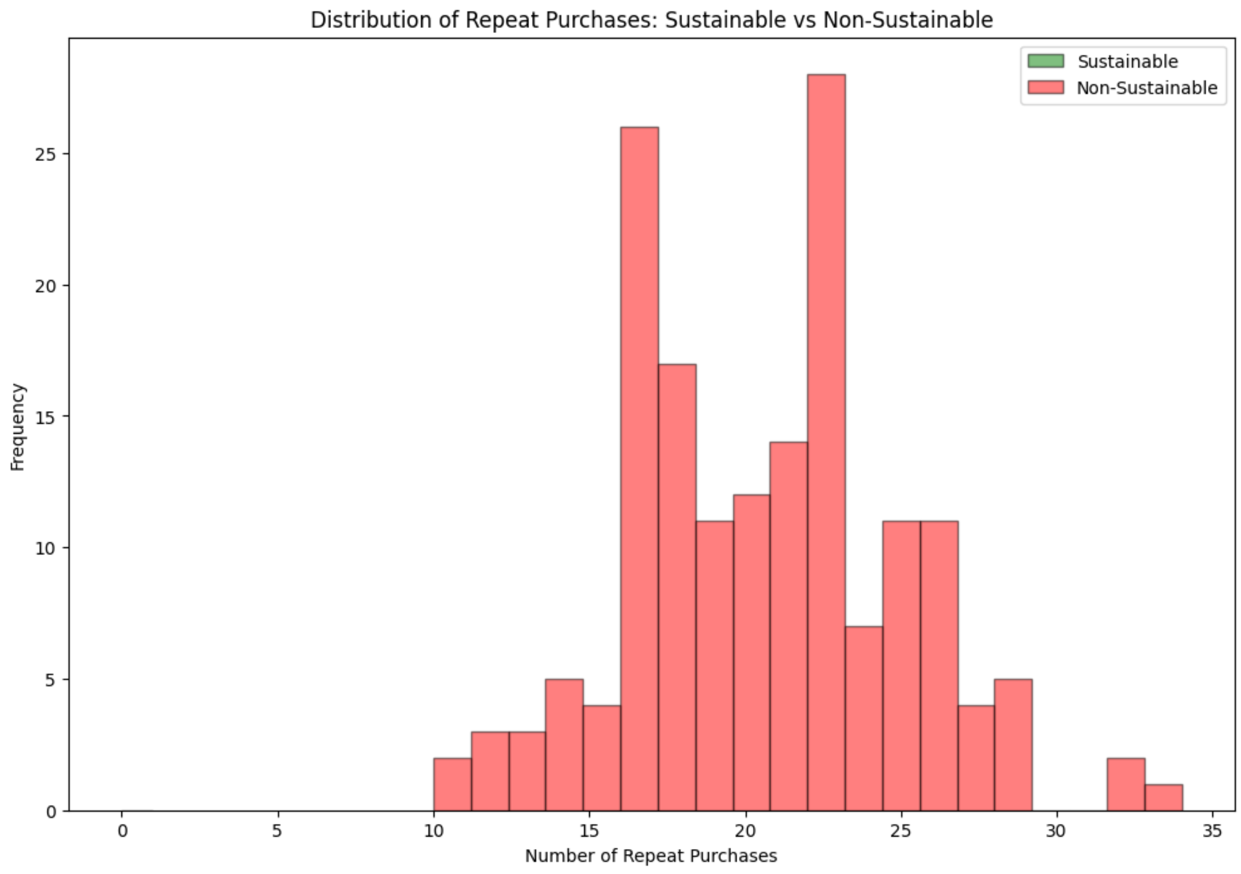
{"summary":"{\n \"name\": \"merged_df\",\n \"rows\": 17,\n \"fields\": [\n   {\n   \"column\": \"Customer Reference ID\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 3997,\n   \"max\": 3997,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   3997\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"review_summary\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n   \"Super fun and SO many compliments!\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"review_text\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n   \"I wore this dress for a rehearsal and rehearsal dinner. This is one of my favorite dresses ever! The size 12 fit much better than the backup size 14. My bra showed the tiniest amount in the arm holes, but it wasn't noticeable enough to be an issue. I definitely needed a bra in this dress. The colors are beautiful, and the pattern is so fun and unique. I got compliments all night long! The cut really accentuated my hourglass figure. The dress was very comfortable all night long (and it was a late night). Definitely recommend this dress!!\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\

```
n    },\n    {\n      \"column\": \"sentiment\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.0,\n        \"min\": 0.2398214285714286,\n        \"max\":
0.2398214285714286,\n        \"num_unique_values\": 1,\n
\"samples\": [\n          0.2398214285714286\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Item Purchased\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 16,\n        \"samples\": [\n
\"Wallet\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Purchase Amount (USD)\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 785.6607389346224,\n
\"min\": 10.0,\n        \"max\": 3003.0,\n
\"num_unique_values\": 14,\n        \"samples\": [\n          108.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Review Rating\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.1305206424815617,\n        \"min\": 1.6,\n        \"max\": 4.9,\n
\"num_unique_values\": 11,\n        \"samples\": [\n          4.2\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Payment Method\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"Credit
Card\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"merged_df"}
```

```python
import pandas as pd
import matplotlib.pyplot as plt



# Load necessary columns from the sheets
user_review_df = pd.read_excel(xls, 'User_review',
usecols=['review_text', 'review_summary', 'user_id', 'item_id'])
fashion_retail_df = pd.read_excel(xls, 'Fashion_retail',
usecols=['Customer Reference ID', 'Item Purchased', 'Purchase Amount
(USD)', 'Date Purchase', 'Review Rating'])

# Define keywords related to sustainability
sustainable_keywords = ['sustainable', 'eco', 'organic', 'recycled',
'environment']
sustainable_pattern = '|'.join(sustainable_keywords)

# Filter sustainable reviews
sustainable_reviews =
user_review_df[user_review_df['review_text'].str.contains(sustainable_
pattern, case=False, na=False) |
user_review_df['review_summary'].str.contains(sustainable_pattern,
case=False, na=False)]

# Get item IDs of sustainable items
sustainable_item_ids = sustainable_reviews['item_id'].unique()

# Mark items as sustainable or not
fashion_retail_df['is_sustainable'] = fashion_retail_df['Item
Purchased'].isin(sustainable_item_ids)

# Count repeat purchases for sustainable and non-sustainable items
repeat_purchases = fashion_retail_df.groupby(['Customer Reference ID',
'is_sustainable']).size().reset_index(name='purchase_count')

# Calculate loyalty metrics
loyalty_metrics = repeat_purchases.groupby('Customer Reference
ID').agg(
    total_purchases=pd.NamedAgg(column='purchase_count',
aggfunc='sum'),
    sustainable_purchases=pd.NamedAgg(column='purchase_count',
aggfunc=lambda x: x[repeat_purchases['is_sustainable']].sum())
).reset_index()

# Add percentage of sustainable purchases
loyalty_metrics['sustainable_percentage'] =
(loyalty_metrics['sustainable_purchases'] /
loyalty_metrics['total_purchases']) * 100

# Display loyalty metrics
print(loyalty_metrics.head())

# Visualize the distribution of sustainable purchase percentages
plt.figure(figsize=(10, 6))
plt.hist(loyalty_metrics['sustainable_percentage'], bins=20,
color='green', edgecolor='black')
plt.xlabel('Percentage of Sustainable Purchases')
plt.ylabel('Frequency')
plt.title('Distribution of Sustainable Purchase Percentages')
plt.show()

# Visualize repeat purchase behavior
sustainable_repeat =
repeat_purchases[repeat_purchases['is_sustainable']]
non_sustainable_repeat =
repeat_purchases[~repeat_purchases['is_sustainable']]

plt.figure(figsize=(12, 8))
plt.hist(non_sustainable_repeat['purchase_count'], bins=20, alpha=0.5,
label='Non-Sustainable', color='red', edgecolor='black')
plt.xlabel('Number of Repeat Purchases')
plt.ylabel('Frequency')
plt.title('Distribution of Repeat Purchases:  Non-Sustainable')
plt.legend()
plt.show()
```

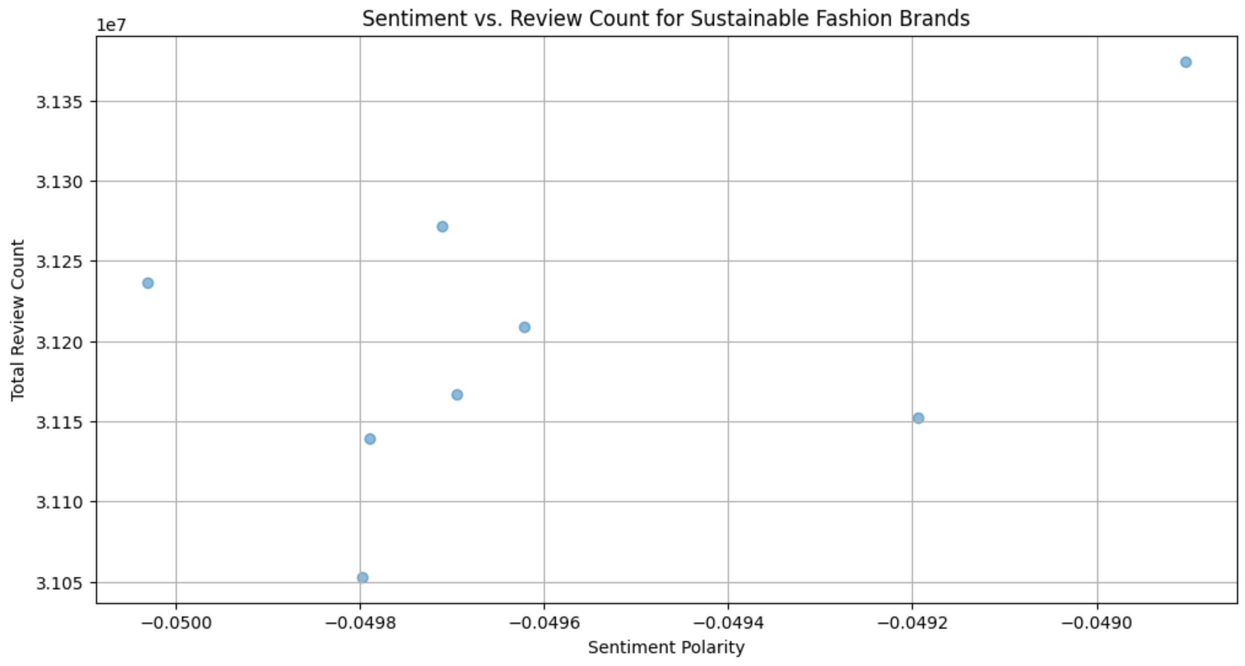
```
   Customer Reference ID  total_purchases  sustainable_purchases  \
0                   3957               14                      0
1                   3958               20                      0
2                   3959               22                      0
3                   3960               18                      0
4                   3961               22                      0

   sustainable_percentage
0                     0.0
1                     0.0
2                     0.0
3                     0.0
4                     0.0
```

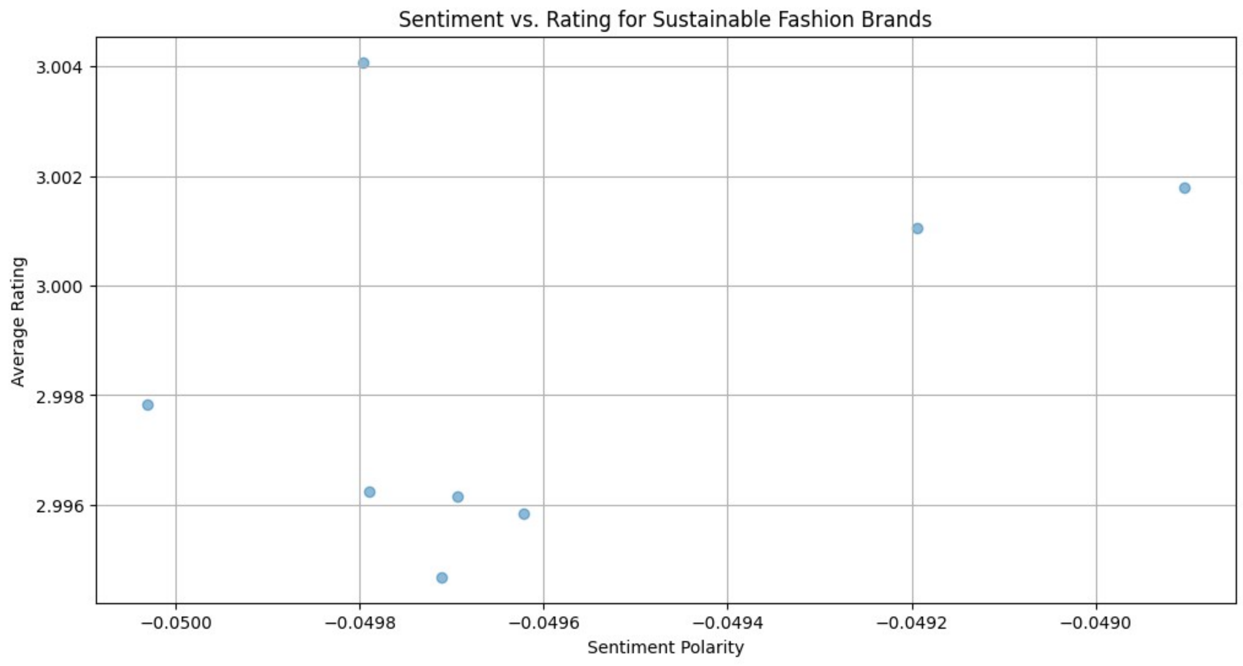




```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud

# Load the Excel file
file_path = '/content/drive/MyDrive/Combined_Data.xlsx'
xls = pd.ExcelFile(file_path)

# Load necessary columns from the sheets
data_df = pd.read_excel(xls, 'UK_brand_data')

# Define keywords related to sustainable fashion
sustainable_keywords = ['sustainable', 'eco', 'organic', 'recycled',
'environment']
sustainable_pattern = '|'.join(sustainable_keywords)

# Filter sustainable reviews
sustainable_reviews =
data_df[data_df['Description'].str.contains(sustainable_pattern,
case=False, na=False) |
                              data_df['Social Media
Comments'].str.contains(sustainable_pattern, case=False, na=False)]

# Identify popular materials, styles, and practices
materials = sustainable_reviews['Style Attributes'].value_counts()
styles = sustainable_reviews['Category'].value_counts()
practices = sustainable_reviews['feedback'].value_counts()

# Display the results
print("Popular Materials:\n", materials)
print("\nPopular Styles:\n", styles)
print("\nPopular Practices:\n", practices)

# Word cloud for popular materials and styles
text = ' '.join(sustainable_reviews['Style
Attributes'].dropna().astype(str).tolist())
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(text)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Popular Materials and Styles in Sustainable Fashion')
plt.show()

Popular Materials:
 Series([], Name: count, dtype: int64)

Popular Styles:
 Series([], Name: count, dtype: int64)

Popular Practices:
 Series([], Name: count, dtype: int64)

----------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-34-eaaabcaf544c> in <cell line: 32>()
     30 # Word cloud for popular materials and styles
     31 text = ' '.join(sustainable_reviews['Style
Attributes'].dropna().astype(str).tolist())
---> 32 wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(text)
     33
     34 plt.figure(figsize=(10, 5))

/usr/local/lib/python3.10/dist-packages/wordcloud/wordcloud.py in
generate(self, text)
```

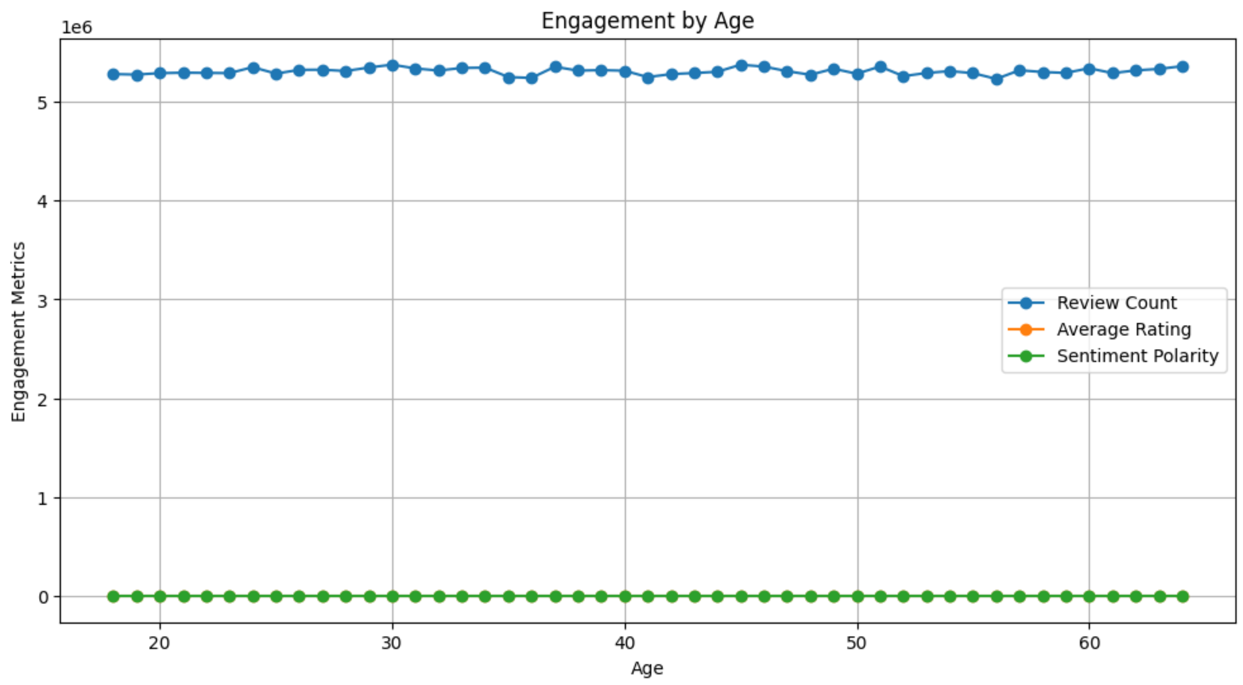
```
    640         self
    641         """
--> 642         return self.generate_from_text(text)
    643
    644     def _check_generated(self):

/usr/local/lib/python3.10/dist-packages/wordcloud/wordcloud.py in
generate_from_text(self, text)
    622         """
    623         words = self.process_text(text)
--> 624         self.generate_from_frequencies(words)
    625         return self
    626

/usr/local/lib/python3.10/dist-packages/wordcloud/wordcloud.py in
generate_from_frequencies(self, frequencies, max_font_size)
    408         frequencies = sorted(frequencies.items(),
key=itemgetter(1), reverse=True)
    409         if len(frequencies) <= 0:
--> 410             raise ValueError("We need at least 1 word to plot
a word cloud, "
    411                              "got %d." % len(frequencies))
    412         frequencies = frequencies[:self.max_words]

ValueError: We need at least 1 word to plot a word cloud, got 0.

# Sentiment Analysis on Social Media Comments
from textblob import TextBlob

# Function to analyze sentiment
def get_sentiment(text):
    blob = TextBlob(str(text))
    return blob.sentiment.polarity

# Apply sentiment analysis
data_df['sentiment'] = data_df['Social Media
Comments'].apply(get_sentiment)

# Correlation between sentiment and popularity
popularity = data_df.groupby('Brand').agg({
    'sentiment': 'mean',
    'Review Count': 'sum',
    'Rating': 'mean'
}).reset_index()

# Visualize the relationship between sentiment and review count
plt.figure(figsize=(12, 6))
plt.scatter(popularity['sentiment'], popularity['Review Count'],
alpha=0.5)
plt.xlabel('Sentiment Polarity')
plt.ylabel('Total Review Count')
plt.title('Sentiment vs. Review Count for Sustainable Fashion Brands')
plt.grid(True)
plt.show()

# Visualize the relationship between sentiment and rating
plt.figure(figsize=(12, 6))
plt.scatter(popularity['sentiment'], popularity['Rating'], alpha=0.5)
plt.xlabel('Sentiment Polarity')
plt.ylabel('Average Rating')
plt.title('Sentiment vs. Rating for Sustainable Fashion Brands')
plt.grid(True)
plt.show()
```

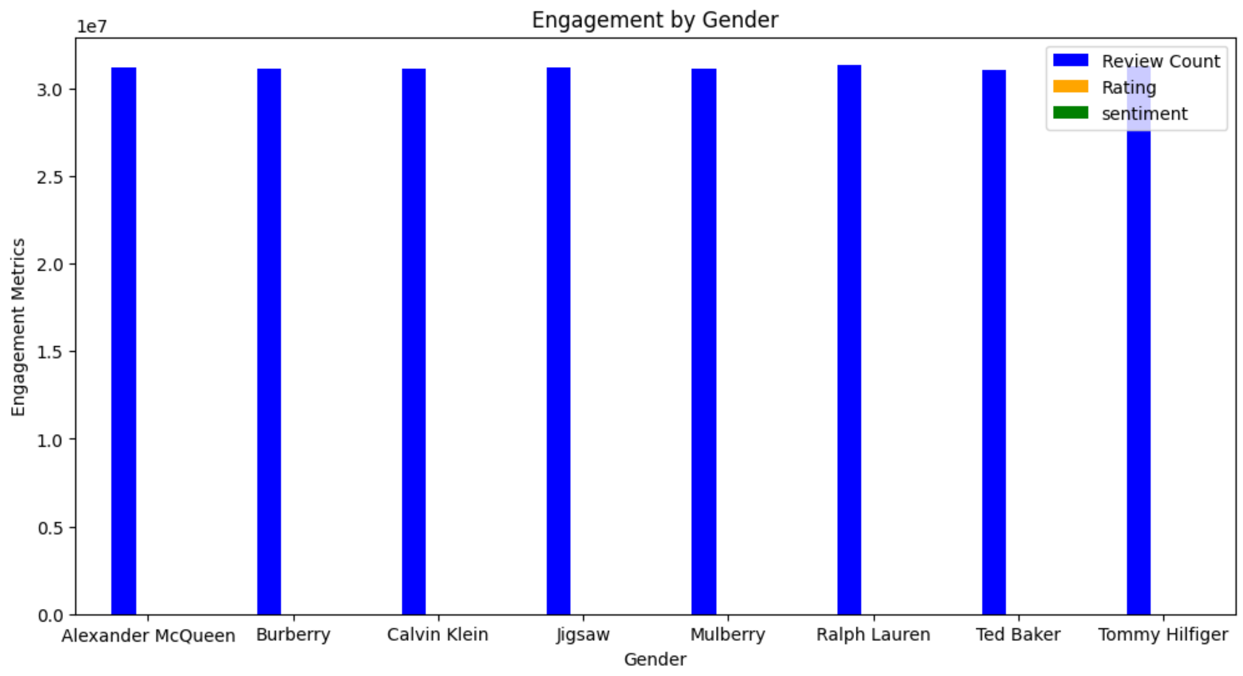




```python
# Analyze engagement rates by demographic factors
def calculate_engagement_rate(df, group_by_col):
    grouped_df = df.groupby(group_by_col).agg({
        'Review Count': 'sum',
        'Rating': 'mean',
        'sentiment': 'mean'
    }).reset_index()
    return grouped_df

# Engagement rates by age
age_engagement = calculate_engagement_rate(data_df, 'Age')
print("\nEngagement Rates by Age:\n", age_engagement)

# Engagement rates by gender
gender_engagement = calculate_engagement_rate(data_df, 'Brand')  #
Assuming Brand represents gender in this context
print("\nEngagement Rates by Gender:\n", gender_engagement)

# Visualize engagement by age
plt.figure(figsize=(12, 6))
plt.plot(age_engagement['Age'], age_engagement['Review Count'],
marker='o', label='Review Count')
plt.plot(age_engagement['Age'], age_engagement['Rating'], marker='o',
label='Average Rating')
plt.plot(age_engagement['Age'], age_engagement['sentiment'],
marker='o', label='Sentiment Polarity')
plt.xlabel('Age')
plt.ylabel('Engagement Metrics')
plt.title('Engagement by Age')
plt.legend()
plt.grid(True)
plt.show()

# Visualize engagement by gender
gender_engagement.plot(kind='bar', x='Brand', figsize=(12, 6),
color=['blue', 'orange', 'green'])
plt.xlabel('Gender')
plt.ylabel('Engagement Metrics')
plt.title('Engagement by Gender')
plt.xticks(rotation=0)
plt.show()


Engagement Rates by Age:
     Age  Review Count    Rating  sentiment
0    18       5281867  2.996537  -0.049763
1    19       5276799  2.991866  -0.050216
2    20       5291104  2.993276  -0.048389
3    21       5296189  2.998248  -0.050350
4    22       5294075  2.994613  -0.051544
5    23       5291504  3.005887  -0.050545
6    24       5352198  2.992099  -0.052159
7    25       5286554  3.007877  -0.048730
8    26       5324792  3.010940  -0.051122
9    27       5326242  2.995362  -0.049728
10   28       5313148  2.996880  -0.049128
11   29       5348259  2.991913  -0.047838
12   30       5377443  3.005378  -0.050138
13   31       5338876  2.992550  -0.048741
14   32       5317771  3.004866  -0.051964
15   33       5344067  3.001090  -0.048447
16   34       5347371  2.991565  -0.050482
17   35       5251979  2.995258  -0.049292
18   36       5243043  3.002400  -0.049247
19   37       5356216  3.007895  -0.050478
20   38       5317073  3.007599  -0.050959
21   39       5322127  3.005345  -0.048023
22   40       5316167  3.002292  -0.049793
23   41       5251470  3.004623  -0.049910
24   42       5281688  3.001130  -0.047151
25   43       5292341  2.991177  -0.049291
26   44       5305353  3.007769  -0.049902
27   45       5376819  3.004407  -0.048800
28   46       5357751  2.980036  -0.048412
29   47       5311612  2.999561  -0.049910
30   48       5274706  2.995909  -0.049895
31   49       5336154  2.993096  -0.049238
32   50       5285293  2.997978  -0.048936
33   51       5358785  3.007525  -0.049339
```

```
34   52       5260954  3.002966  -0.049070
35   53       5293127  2.994268  -0.049272
36   54       5311975  3.001688  -0.048390
37   55       5291745  3.001268  -0.050746
38   56       5233028  2.989707  -0.049587
39   57       5320457  3.003708  -0.048209
40   58       5302824  2.993223  -0.049637
41   59       5293773  2.984328  -0.050489
42   60       5341582  2.984163  -0.050233
43   61       5291526  3.002131  -0.049768
44   62       5319072  3.000439  -0.048947
45   63       5333980  2.990247  -0.049637
46   64       5361552  3.004002  -0.048997

Engagement Rates by Gender:
                Brand  Review Count    Rating  sentiment
0  Alexander McQueen      31236512  2.997839  -0.050030
1           Burberry      31166571  2.996150  -0.049694
2        Calvin Klein      31152396  3.001044  -0.049194
3             Jigsaw      31208649  2.995833  -0.049621
4           Mulberry      31139142  2.996242  -0.049790
5        Ralph Lauren      31374326  3.001783  -0.048904
6          Ted Baker      31052965  3.004080  -0.049797
7     Tommy Hilfiger      31271870  2.994681  -0.049711
```





```
# Function to analyze sentiment
def get_sentiment(text):
    blob = TextBlob(str(text))
    return blob.sentiment.polarity

# Apply sentiment analysis to social media comments
data_df['sentiment'] = data_df['Social Media
Comments'].apply(get_sentiment)

# Display the sentiment analysis results
data_df[['Social Media Comments', 'sentiment']].head()
```

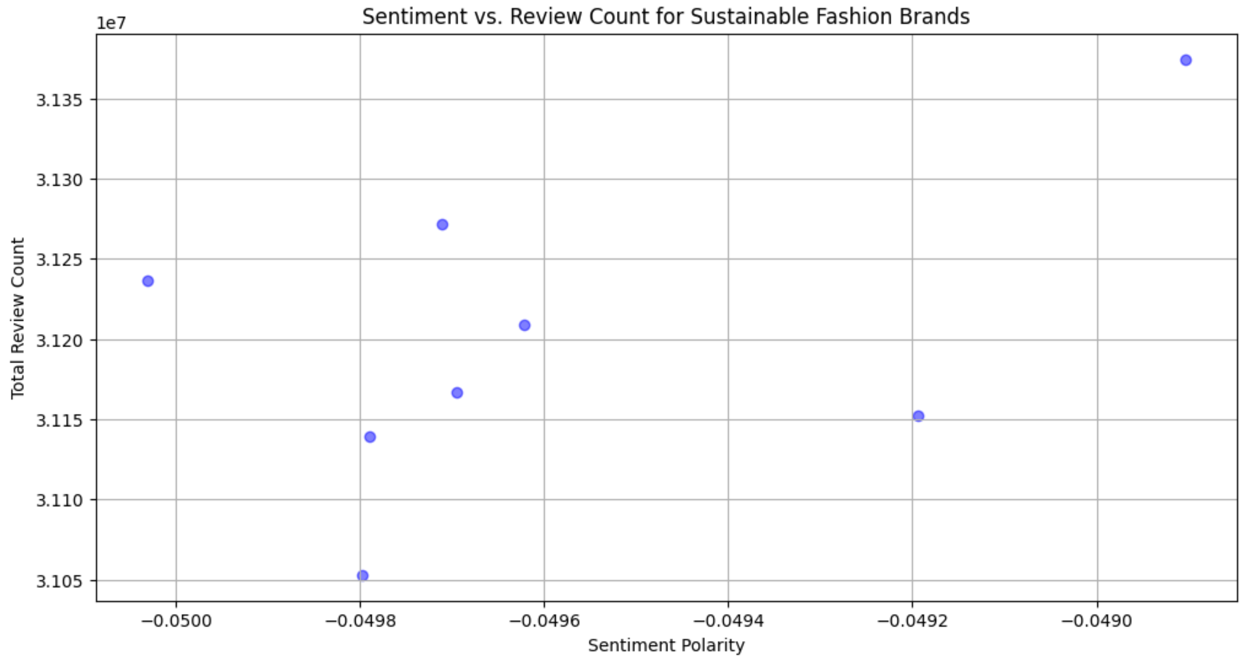
{"summary":"{\n \"name\": \"data_df[['Social Media Comments',
'sentiment']]\",\n \"rows\": 5,\n \"fields\": [\n   {\n
\"column\": \"Social Media Comments\",\n   \"properties\": {\n
\"dtype\": \"string\",\n   \"num_unique_values\": 4,\n
\"samples\": [\n   \"Neutral\",\n   \"Other\",\n
\"Mixed\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"sentiment\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.13181426326464066,\n   \"min\": -
0.3,\n   \"max\": 0.0,\n   \"num_unique_values\": 3,\n
\"samples\": [\n   0.0,\n   -0.3,\n   -0.125\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe"}

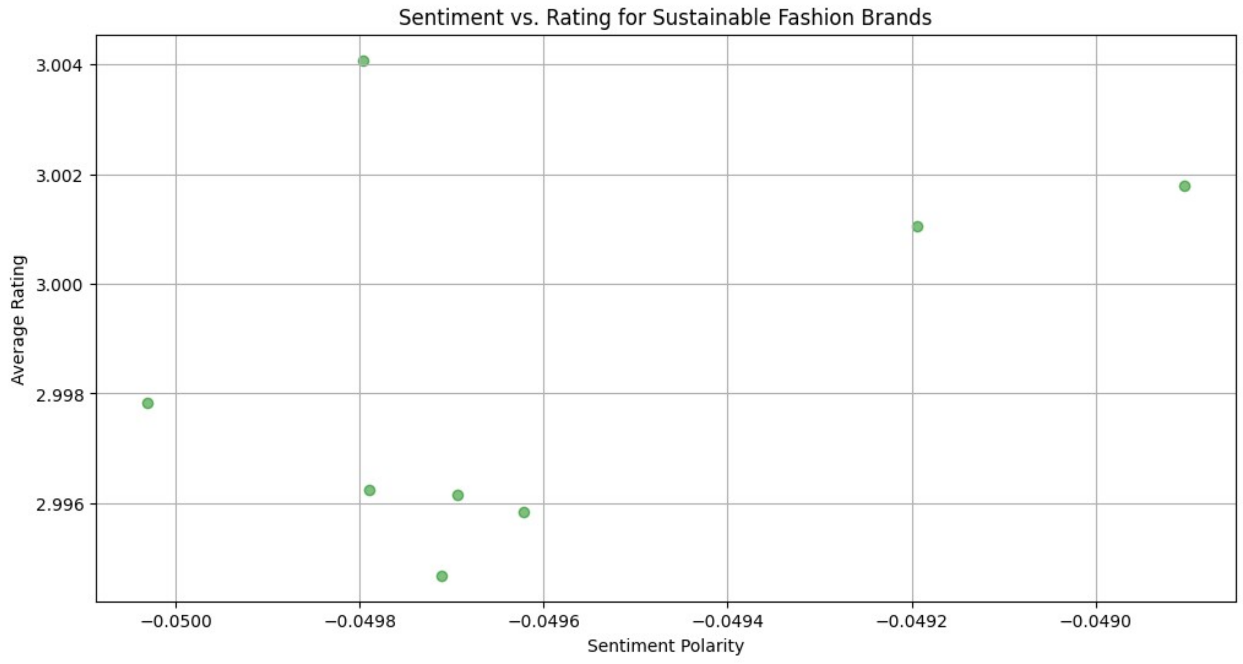
```
# Group data by brand and calculate average sentiment, total review
count, and average rating
```

```
brand_sentiment = data_df.groupby('Brand').agg({
    'sentiment': 'mean',
    'Review Count': 'sum',
    'Rating': 'mean'
}).reset_index()

# Display the aggregated data
print(brand_sentiment)

# Correlation between sentiment and review count
correlation_review_count =
brand_sentiment['sentiment'].corr(brand_sentiment['Review Count'])
print(f'Correlation between sentiment and review count:
{correlation_review_count}')

# Correlation between sentiment and rating
correlation_rating =
brand_sentiment['sentiment'].corr(brand_sentiment['Rating'])
print(f'Correlation between sentiment and rating:
{correlation_rating}')
```

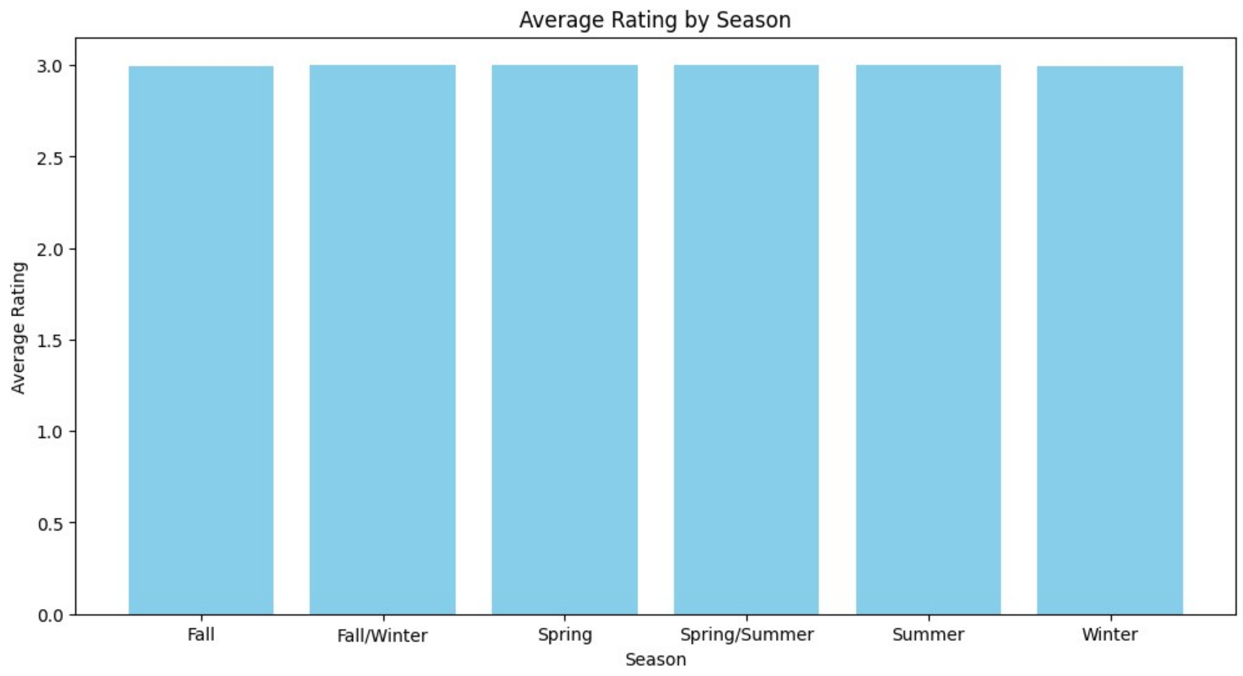
```
                Brand  sentiment  Review Count    Rating
0  Alexander McQueen  -0.050030      31236512  2.997839
1           Burberry  -0.049694      31166571  2.996150
2       Calvin Klein  -0.049194      31152396  3.001044
3             Jigsaw  -0.049621      31208649  2.995833
4           Mulberry  -0.049790      31139142  2.996242
5       Ralph Lauren  -0.048904      31374326  3.001783
6          Ted Baker  -0.049797      31052965  3.004080
7     Tommy Hilfiger  -0.049711      31271870  2.994681
Correlation between sentiment and review count: 0.49269357939219693
Correlation between sentiment and rating: 0.41840576362947296
```

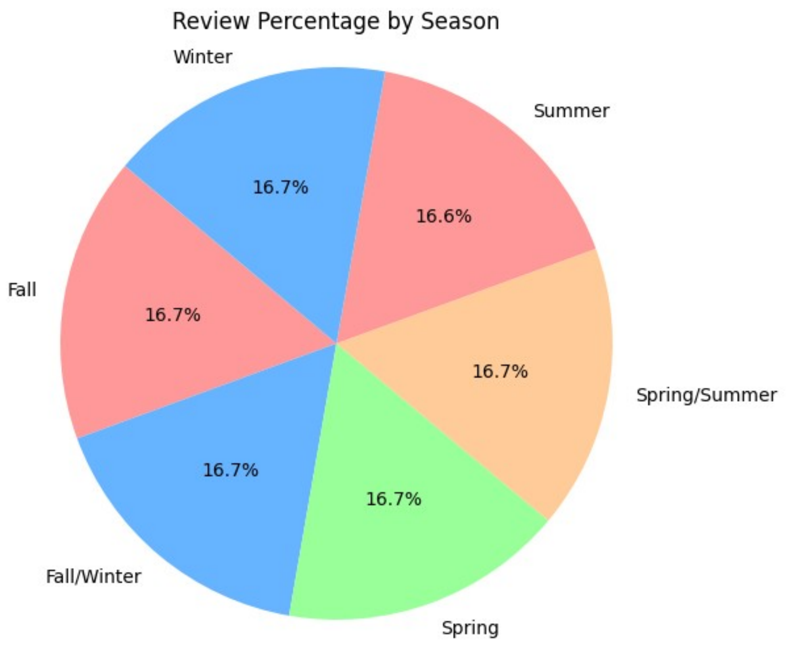
```
# Visualize the relationship between sentiment and review count
plt.figure(figsize=(12, 6))
plt.scatter(brand_sentiment['sentiment'], brand_sentiment['Review
Count'], alpha=0.5, color='blue')
plt.xlabel('Sentiment Polarity')
plt.ylabel('Total Review Count')
plt.title('Sentiment vs. Review Count for Sustainable Fashion Brands')
plt.grid(True)
plt.show()

# Visualize the relationship between sentiment and rating
plt.figure(figsize=(12, 6))
plt.scatter(brand_sentiment['sentiment'], brand_sentiment['Rating'],
alpha=0.5, color='green')
plt.xlabel('Sentiment Polarity')
plt.ylabel('Average Rating')
plt.title('Sentiment vs. Rating for Sustainable Fashion Brands')
plt.grid(True)
plt.show()
```





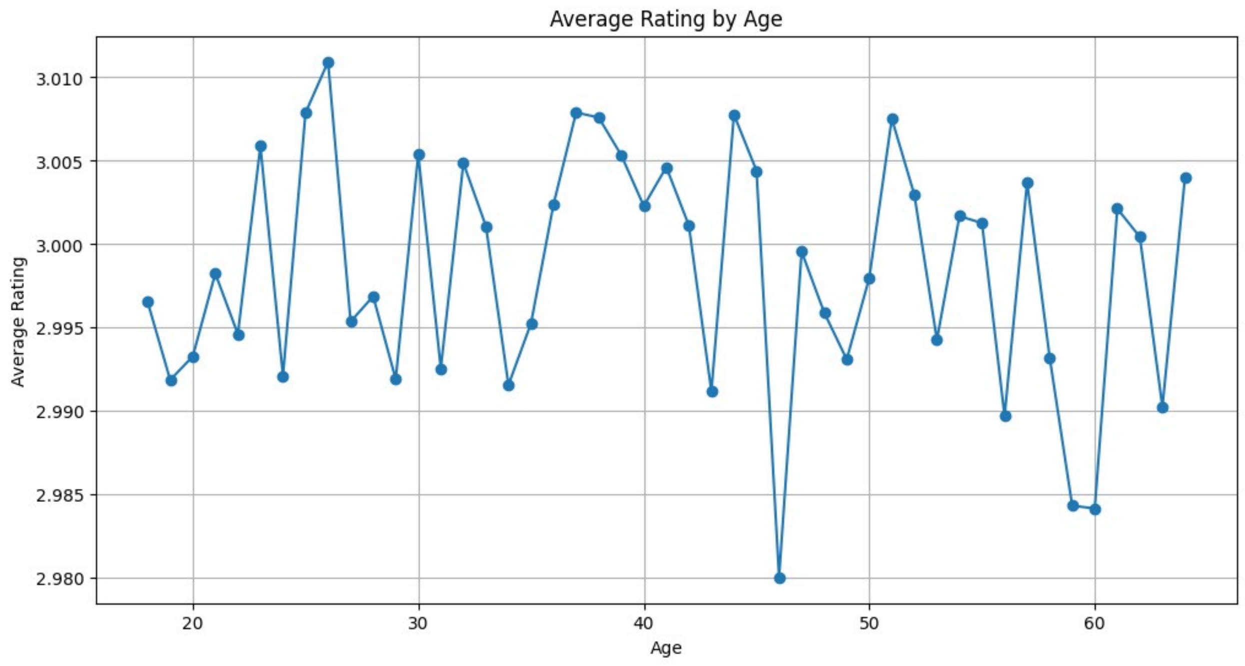
```
# Filter out necessary columns
season_reviews_df = data_df[['Season', 'Rating', 'Review Count']]

# Group by season and calculate average rating and total review count
seasonal_metrics = season_reviews_df.groupby('Season').agg({
    'Rating': 'mean',
    'Review Count': 'sum'
}).reset_index()

# Display the aggregated data
print(seasonal_metrics)

          Season    Rating  Review Count
0           Fall  2.992157      41686744
1    Fall/Winter  3.001432      41592779
2         Spring  2.998440      41597477
3  Spring/Summer  2.998514      41559884
4         Summer  3.003637      41461895
5         Winter  2.996592      41703652

# Calculate the percentage of reviews for each season
total_reviews = season_reviews_df['Review Count'].sum()
seasonal_metrics['Review Percentage'] = (seasonal_metrics['Review
Count'] / total_reviews) * 100

# Display the calculated metrics
print(seasonal_metrics)

          Season    Rating  Review Count  Review Percentage
0           Fall  2.992157      41686744          16.701257
1    Fall/Winter  3.001432      41592779          16.663611
2         Spring  2.998440      41597477          16.665494
3  Spring/Summer  2.998514      41559884          16.650432
4         Summer  3.003637      41461895          16.611174
5         Winter  2.996592      41703652          16.708031

import matplotlib.pyplot as plt

# Plot average rating by season
plt.figure(figsize=(12, 6))
plt.bar(seasonal_metrics['Season'], seasonal_metrics['Rating'],
color='skyblue')
plt.xlabel('Season')
plt.ylabel('Average Rating')
plt.title('Average Rating by Season')
plt.show()


# Plot review percentage by season
plt.figure(figsize=(12, 6))
plt.pie(seasonal_metrics['Review Percentage'],
labels=seasonal_metrics['Season'], autopct='%1.1f%%', startangle=140,
colors=['#ff9999','#66b3ff','#99ff99','#ffcc99'])
plt.axis('equal')
plt.title('Review Percentage by Season')
plt.show()
```

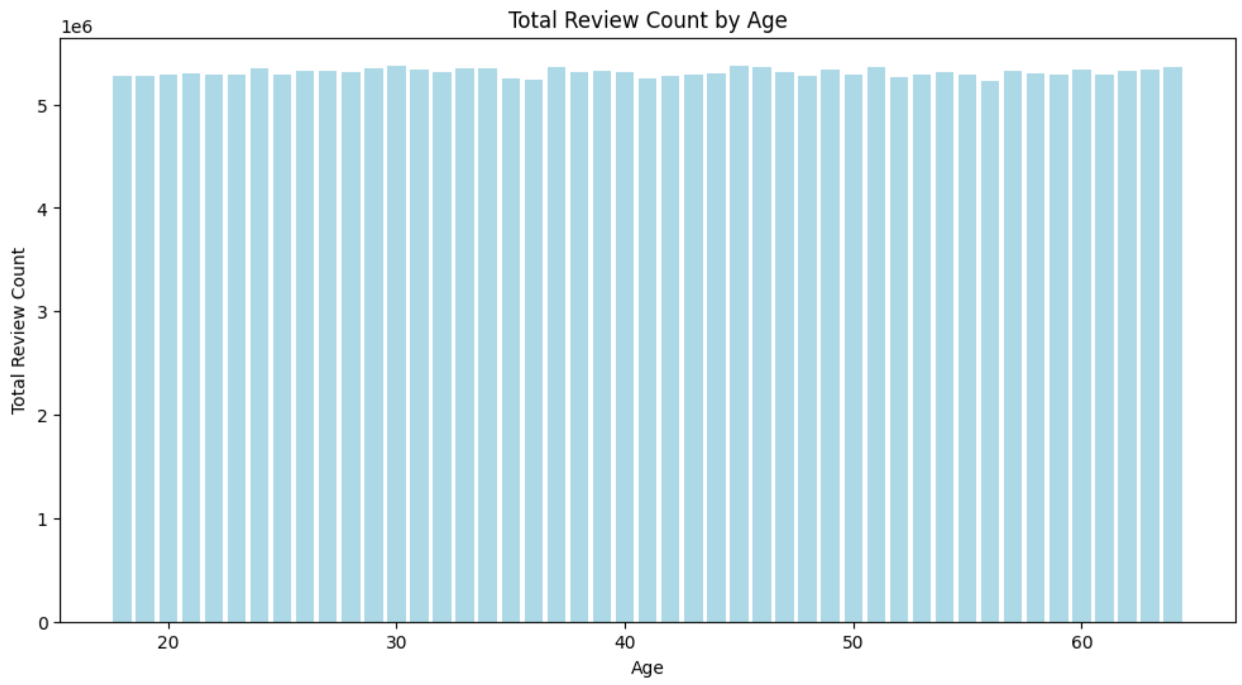


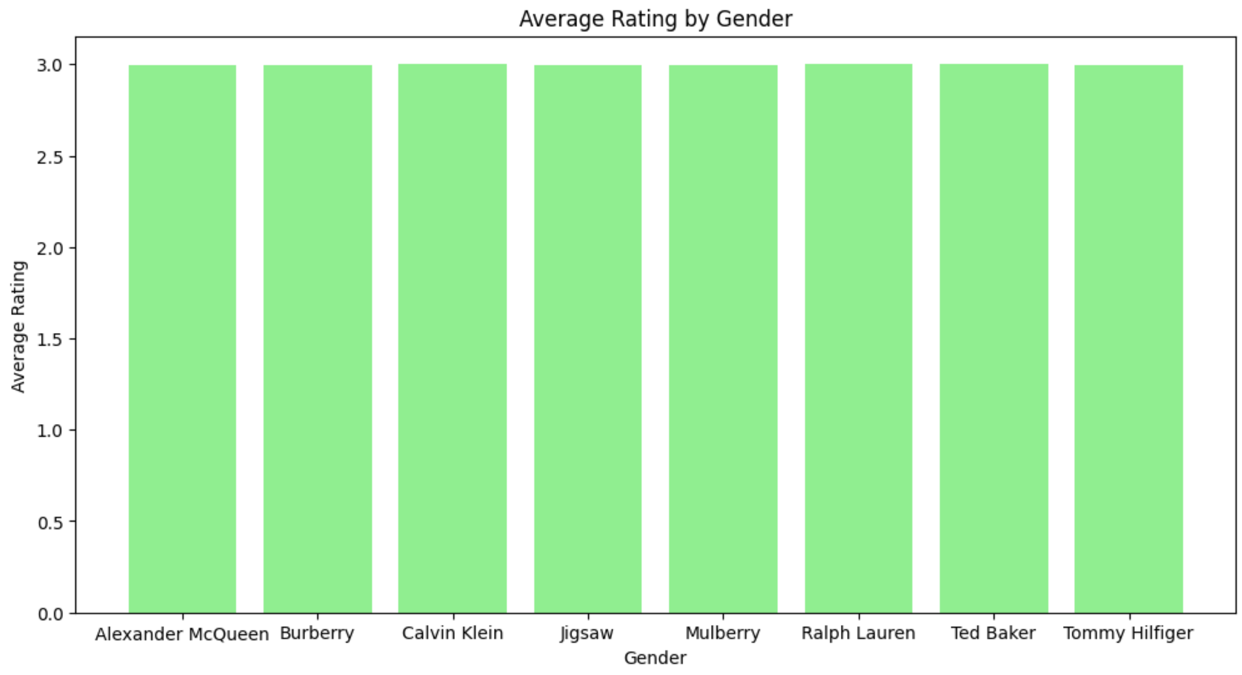


```
import matplotlib.pyplot as plt

# Plot average rating by age
plt.figure(figsize=(12, 6))
plt.plot(age_metrics['Age'], age_metrics['Rating'], marker='o',
label='Average Rating')
plt.xlabel('Age')
```

```python
plt.ylabel('Average Rating')
plt.title('Average Rating by Age')
plt.grid(True)
plt.show()

# Plot review count by age
plt.figure(figsize=(12, 6))
plt.bar(age_metrics['Age'], age_metrics['Review Count'],
color='lightblue')
plt.xlabel('Age')
plt.ylabel('Total Review Count')
plt.title('Total Review Count by Age')
plt.show()

# Plot average rating by gender
plt.figure(figsize=(12, 6))
plt.bar(gender_metrics['Brand'], gender_metrics['Rating'],
color='lightgreen')
plt.xlabel('Gender')
plt.ylabel('Average Rating')
plt.title('Average Rating by Gender')
plt.show()
```







```
# Analyze review metrics by purchase history
purchase_history_metrics = data_df.groupby('Purchase History').agg({
    'Rating': 'mean',
    'Review Count': 'sum'
}).reset_index()

# Display the aggregated data
```

```
print("Review Metrics by Purchase History:\n",
purchase_history_metrics)
```

```
Review Metrics by Purchase History:
   Purchase History    Rating  Review Count
0    Above Average  2.999457      25089716
1          Average  3.002609      24892336
2    Below Average  3.003931      24878064
3             High  2.998025      24959646
4              Low  3.001847      25037740
5           Medium  2.994813      24908790
6       Negligible  2.994973      25000645
7      Significant  2.993407      24831735
8        Very High  2.998455      24892949
9         Very Low  2.997009      25110810
```