```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("/content/SocialMediaUsersDataset.csv")
df.head()
```

	UserID	Name	Gender	DOB	Interests	City	Country
0	1	Jesse Lawhorn	Female	1958-10-15	'Movies', 'Fashion', 'Fashion', 'Books'	Sibolga	Indonesia
1	2	Stacy Payne	Female	2004-07-21	'Gaming', 'Finance and investments', 'Outdoor ...	Al Abyār	Libya
2	3	Katrina Nicewander	Female	2000-02-07	'DIY and crafts', 'Music', 'Science', 'Fashion'	Wādī as Sīr	Jordan
3	4	Eric Yarbrough	Male	1985-04-14	'Outdoor activities', 'Cars and automobiles'	Matera	Italy
4	5	Daniel Adkins	Female	1955-09-18	'Politics', 'History'	Biruaca	Venezuela

Next steps: Generate code with df | View recommended plots | New interactive sheet

Basic Data Inspection

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 7 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   UserID     100000 non-null  int64
 1   Name       100000 non-null  object
 2   Gender     100000 non-null  object
 3   DOB        100000 non-null  object
 4   Interests  100000 non-null  object
 5   City       100000 non-null  object
 6   Country    100000 non-null  object
dtypes: int64(1), object(6)
memory usage: 5.3+ MB
```

Text Cleaning Function

```
import re
def clean_text(text):
    text = str(text)
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'@\S+', '', text)
    text = re.sub(r'#\S+', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    text = text.lower()
    return text
```

Handling Missing Data

```
import numpy as np
print(df.isnull().sum())
numerical_cols = df.select_dtypes(include=np.number).columns
for col in numerical_cols:
```

```
    df[col].fillna(df[col].mean(), inplace=True)
categorical_cols = df.select_dtypes(exclude=np.number).columns
for col in categorical_cols:
    df[col].fillna(df[col].mode()[0], inplace=True)
print(df.isnull().sum())
```

```
UserID       0
Name         0
Gender       0
DOB          0
Interests    0
City         0
Country      0
dtype: int64
UserID       0
Name         0
Gender       0
DOB          0
Interests    0
City         0
Country      0
dtype: int64
<ipython-input-17-f780349a7fe4>:5: FutureWarning: A value is trying to be set on a copy of a D
The behavior will change in pandas 3.0. This inplace method will never work because the interm

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: valu


  df[col].fillna(df[col].mean(), inplace=True)
<ipython-input-17-f780349a7fe4>:8: FutureWarning: A value is trying to be set on a copy of a D
The behavior will change in pandas 3.0. This inplace method will never work because the interm

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: valu


  df[col].fillna(df[col].mode()[0], inplace=True)
```

Sentiment Analysis

```
from textblob import TextBlob
def get_sentiment(text):
    analysis = TextBlob(text)
    return analysis.sentiment.polarity
df.head()
```

	UserID	Name	Gender	DOB	Interests	City	Country
0	1	Jesse Lawhorn	Female	1958-10-15	'Movies', 'Fashion', 'Fashion', 'Books'	Sibolga	Indonesia
1	2	Stacy Payne	Female	2004-07-21	'Gaming', 'Finance and investments', 'Outdoor ...	Al Abyār	Libya
2	3	Katrina Nicewander	Female	2000-02-07	'DIY and crafts', 'Music', 'Science', 'Fashion'	Wādī as Sīr	Jordan
3	4	Eric Yarbrough	Male	1985-04-14	'Outdoor activities', 'Cars and automobiles'	Matera	Italy
4	5	Daniel Adkins	Female	1955-09-18	'Politics', 'History'	Biruaca	Venezuela

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df_encoded = pd.get_dummies(df[['Gender', 'City', 'Country']], drop_first=True)
```

Clean & Standardize ‘Gender’

```
df['Gender'] = df['Gender'].str.strip().str.lower()
df['Gender'].value_counts()
```

Gender	count
male	50069
female	49931

dtype: int64

Extract Age from DOB

```
from datetime import datetime
df['DOB'] = pd.to_datetime(df['DOB'], errors='coerce')
df['Age'] = df['DOB'].apply(lambda x: datetime.now().year - x.year if pd.notnull(x) else None)
```

Create Age Groups

```
def age_group(age):
    if age < 18:
        return 'Teen'
    elif age < 30:
        return 'Young Adult'
    elif age < 50:
        return 'Adult'
    else:
        return 'Senior'
df['AgeGroup'] = df['Age'].apply(age_group)
```

Clean Interests (Text Preprocessing)

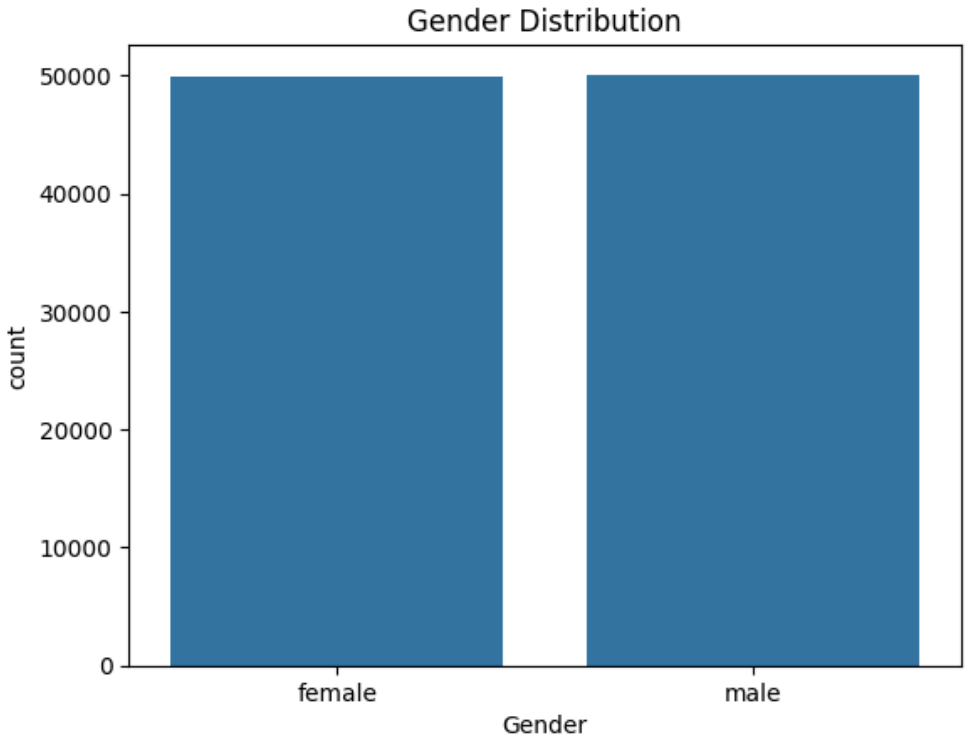
```
import re
df['Cleaned_Interests'] = df['Interests'].str.lower().apply(lambda x: re.sub(r'[^a-zA-Z\s]', '', x
```

Word Count in Interests

```
df['Interest_Word_Count'] = df['Cleaned_Interests'].apply(lambda x: len(x.split()))
```

Gender Distribution Plot

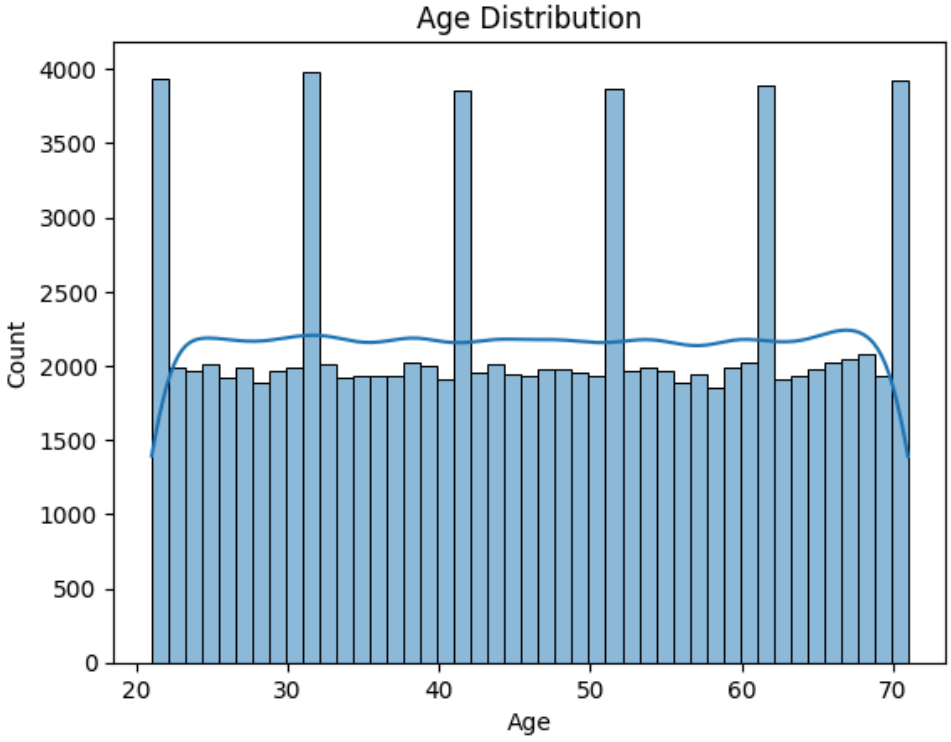
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x='Gender', data=df)
plt.title("Gender Distribution")
plt.show()
```



Age Distribution Plot

```
sns.histplot(df['Age'], kde=True)
plt.title("Age Distribution")
```

```
Text(0.5, 1.0, 'Age Distribution')
```



Word Cloud of Interests

```
from wordcloud import WordCloud
text = ' '.join(df['Cleaned_Interests'].dropna())
wordcloud = WordCloud(width=800, height=400).generate(text)
plt.figure(figsize=(10,5))
```

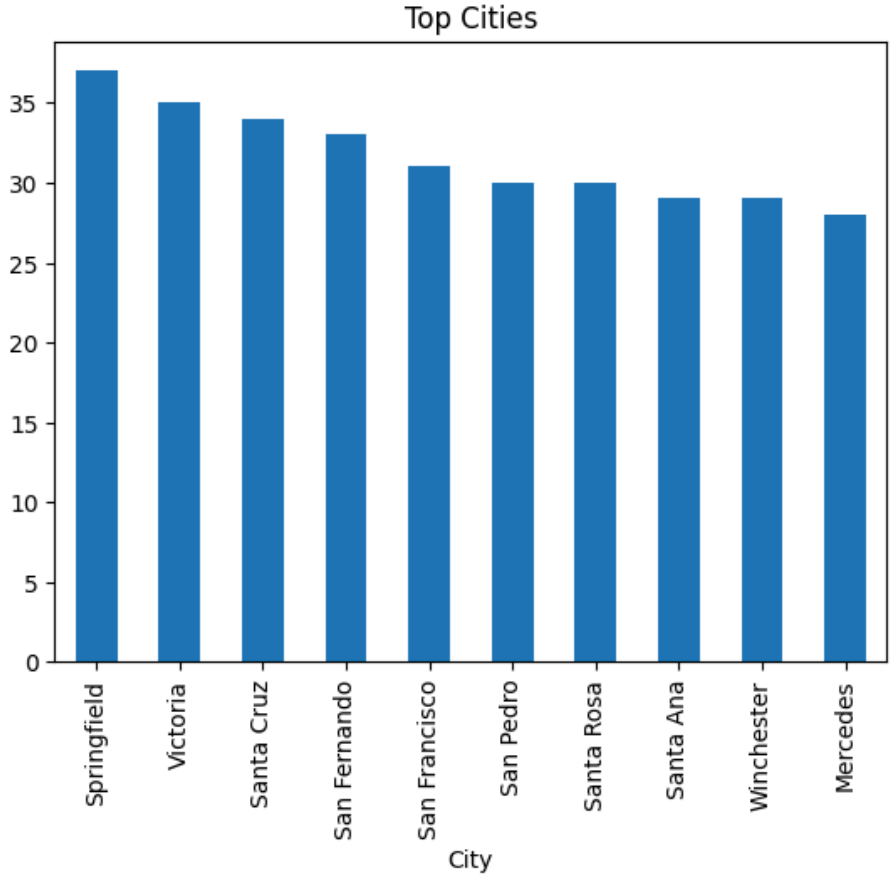
```
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
```

```
(np.float64(-0.5), np.float64(799.5), np.float64(399.5), np.float64(-0.5))
```

Top Cities/Countries

```
df['City'].value_counts().head(10).plot(kind='bar', title="Top Cities")
```

```
<Axes: title={'center': 'Top Cities'}, xlabel='City'>
```



Encode Categorical Data

```
df_encoded = pd.get_dummies(df[['Gender', 'City', 'AgeGroup']], drop_first=True)
```

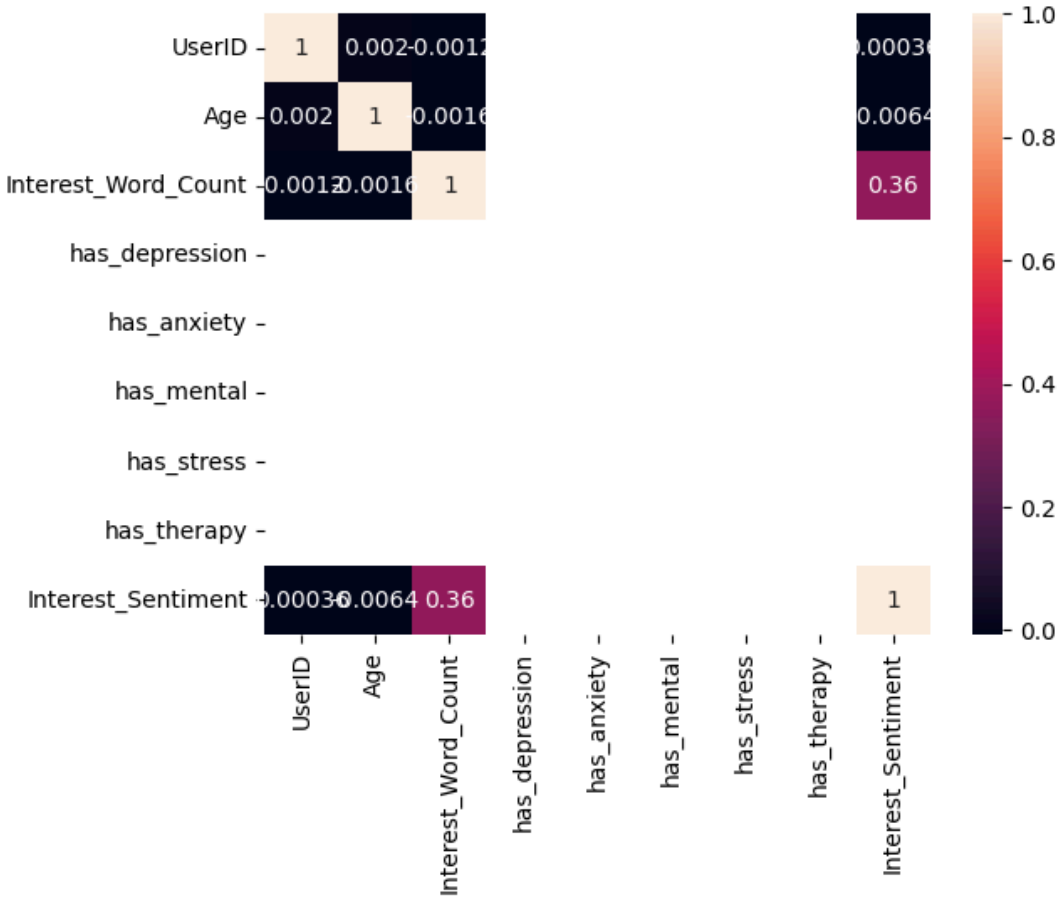
Interest Keyword Flags

```
keywords = ['depression', 'anxiety', 'mental', 'stress', 'therapy']
for kw in keywords:
    df[f'has_{kw}'] = df['Cleaned_Interests'].apply(lambda x: 1 if kw in x else 0)
```

sns.heatmap(df.corr(numeric_only=True), annot=True)

```
sns.heatmap(df.corr(numeric_only=True), annot=True)
```

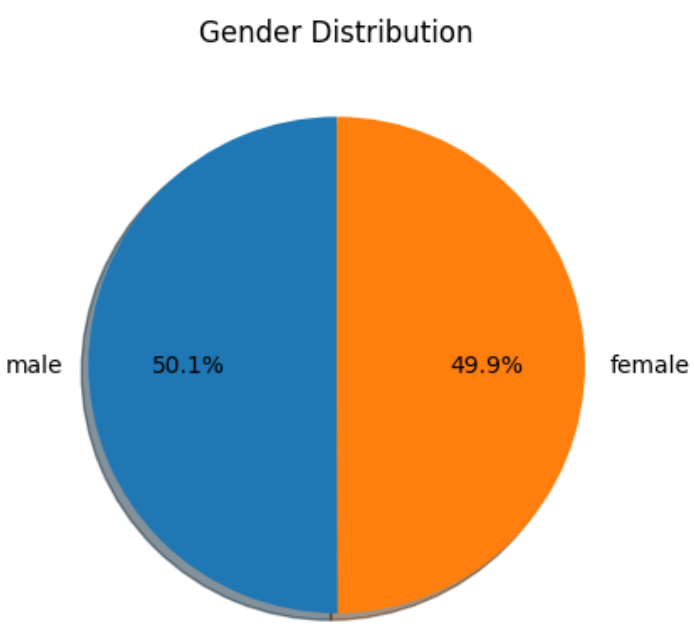
```
<Axes: >
```



Pie Chart of Gender Distribution

Train-Test Split

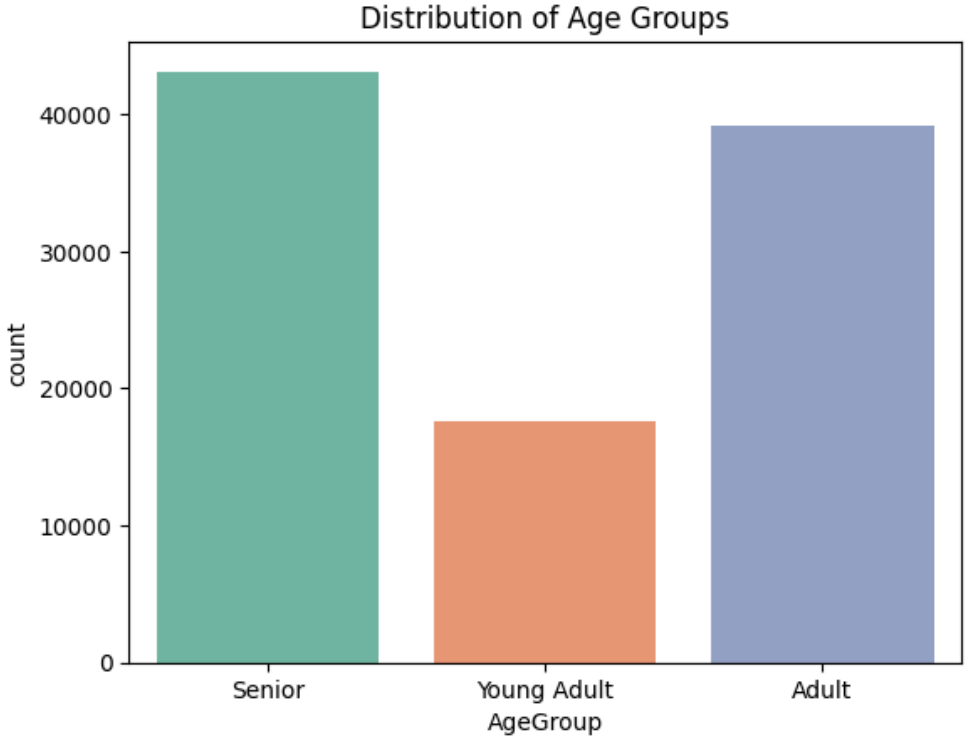
```
df['Gender'].value_counts().plot(kind='pie', autopct='%1.1f%%', startangle=90, shadow=True)
plt.title("Gender Distribution")
plt.ylabel('')
plt.show()
```



Count Plot of Age Groups

```
sns.countplot(data=df, x='AgeGroup', palette='Set2')
plt.title("Distribution of Age Groups")
```

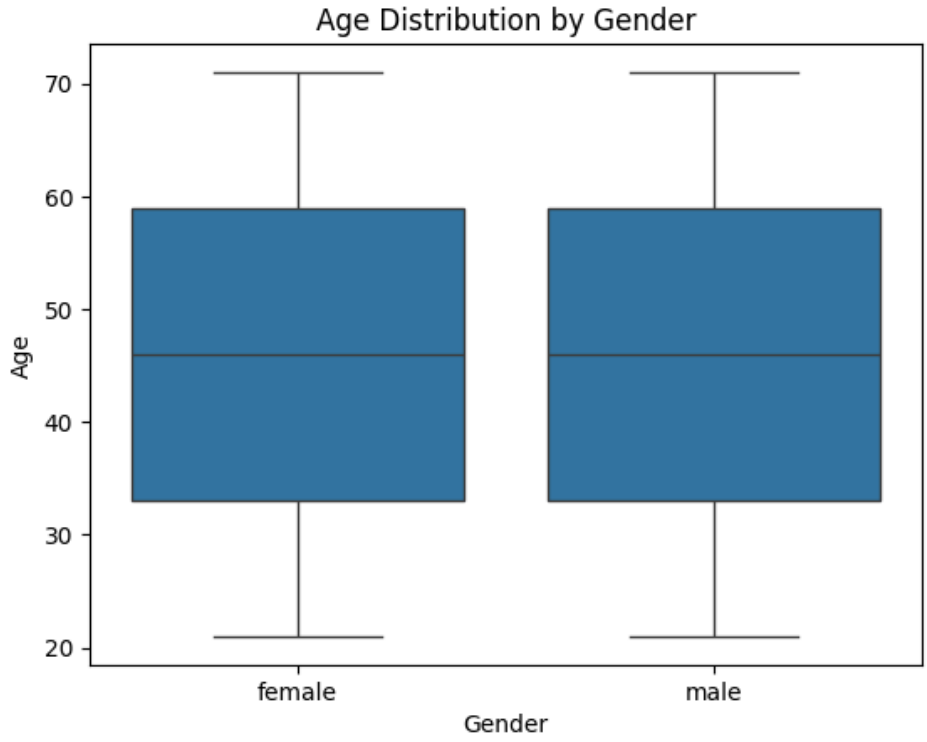
```
<ipython-input-74-9c234f7a5489>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign

  sns.countplot(data=df, x='AgeGroup', palette='Set2')
Text(0.5, 1.0, 'Distribution of Age Groups')
```



Boxplot of Age by Gender

```
sns.boxplot(x='Gender', y='Age', data=df)
plt.title("Age Distribution by Gender")
```

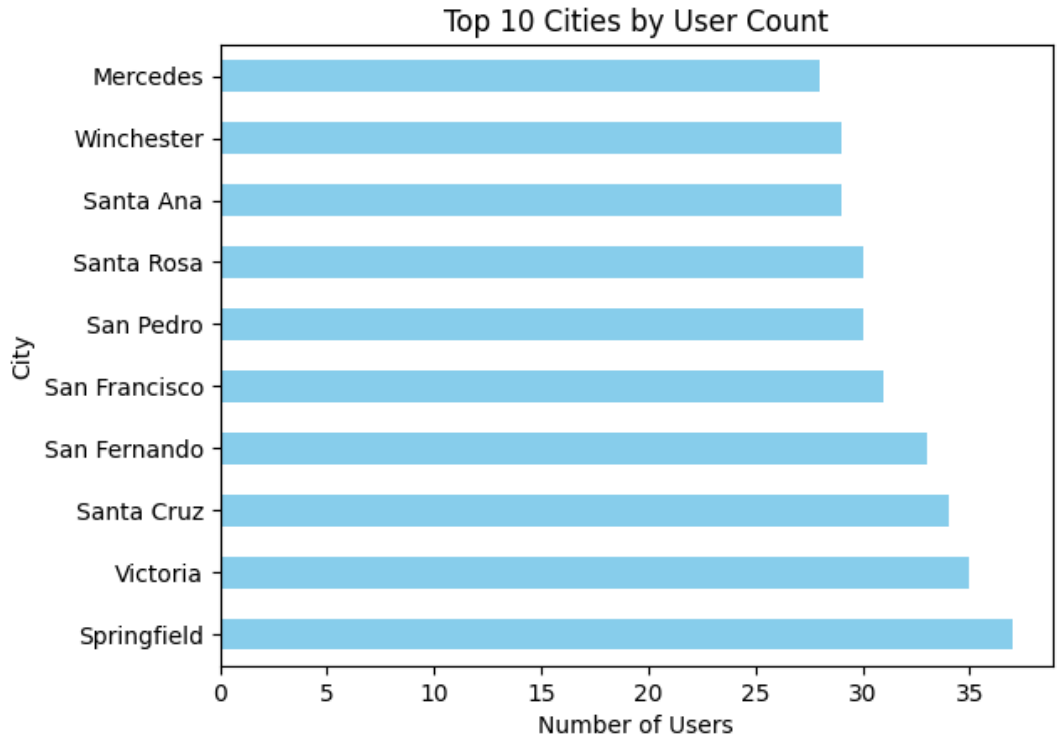
```
Text(0.5, 1.0, 'Age Distribution by Gender')
```



Most Common Cities (Top 10)

```
df['City'].value_counts().head(10).plot(kind='barh', color='skyblue')
plt.title("Top 10 Cities by User Count")
plt.xlabel("Number of Users")
```

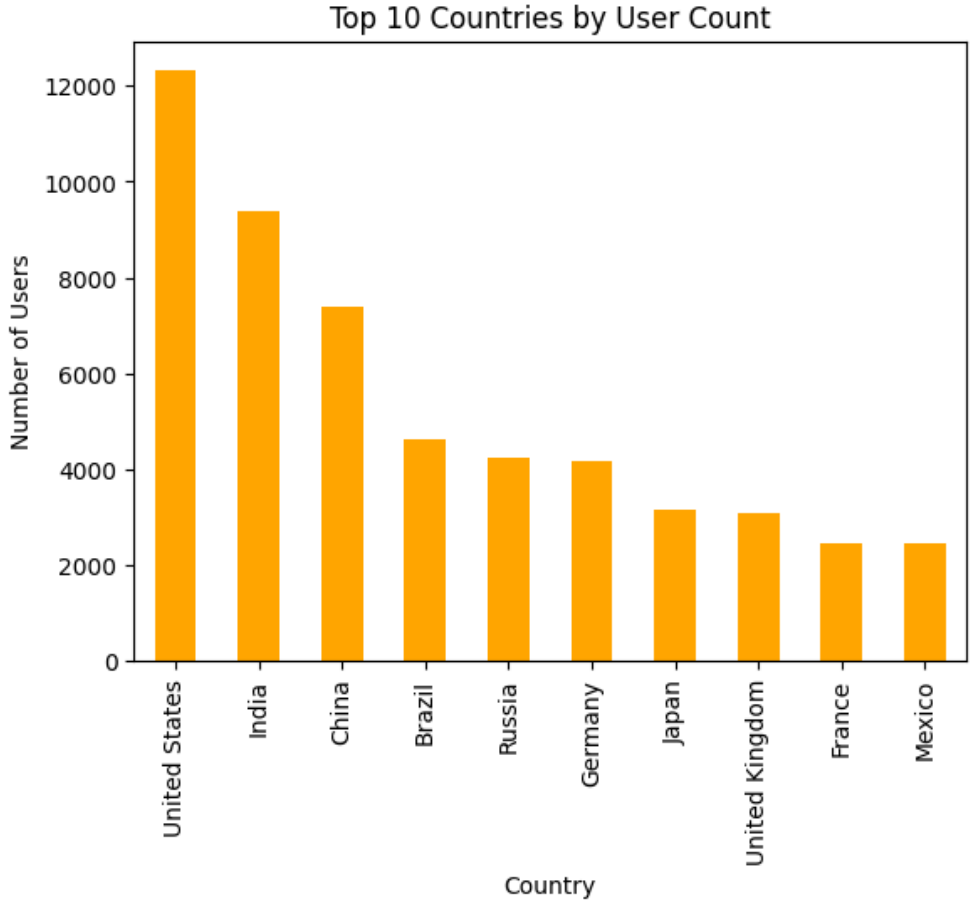
```
Text(0.5, 0, 'Number of Users')
```



Most Common Countries (Top 10)

```
df['Country'].value_counts().head(10).plot(kind='bar', color='orange')
plt.title("Top 10 Countries by User Count")
plt.ylabel("Number of Users")
```

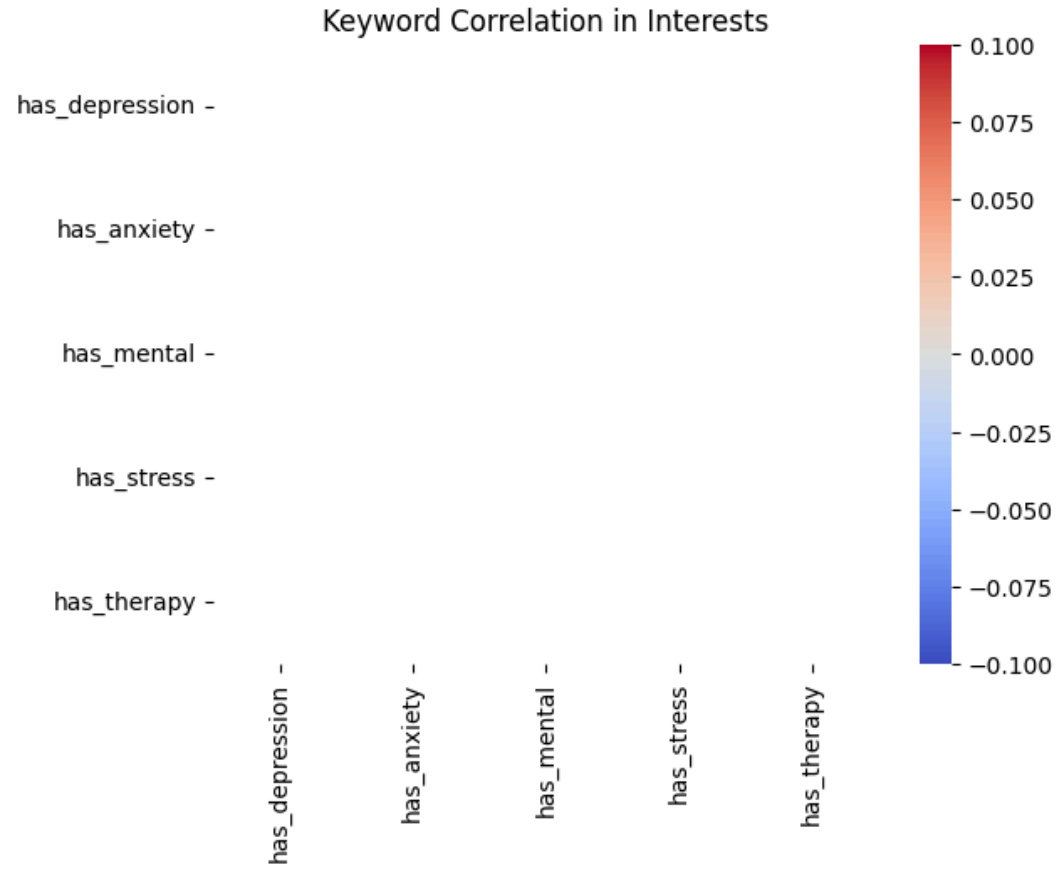
```
Text(0, 0.5, 'Number of Users')
```



Heatmap of Interests Keyword Flags

```
import seaborn as sns

keyword_cols = [col for col in df.columns if col.startswith('has_')]
sns.heatmap(df[keyword_cols].corr(), annot=True, cmap='coolwarm')
plt.title("Keyword Correlation in Interests")
```

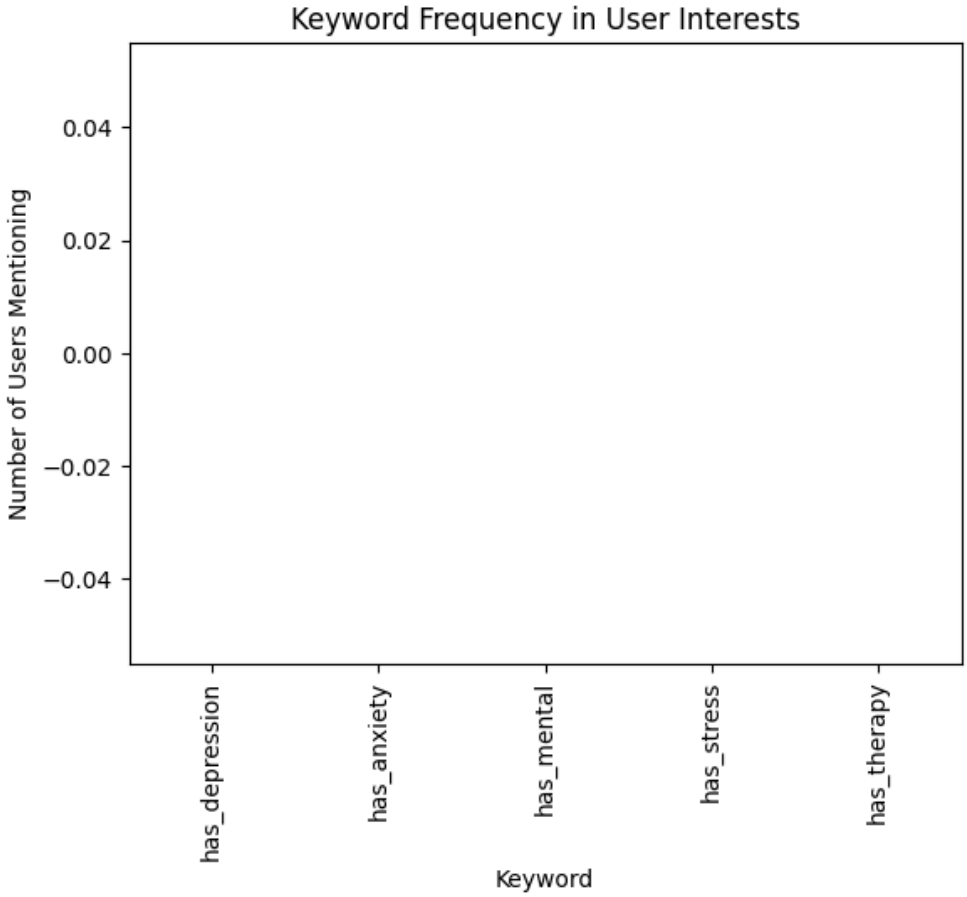
```
/usr/local/lib/python3.11/dist-packages/seaborn/matrix.py:202: RuntimeWarning: All-NaN slice e
  vmin = np.nanmin(calc_data)
/usr/local/lib/python3.11/dist-packages/seaborn/matrix.py:207: RuntimeWarning: All-NaN slice e
  vmax = np.nanmax(calc_data)
Text(0.5, 1.0, 'Keyword Correlation in Interests')
```



Barplot of Keyword Count in Interests

```
keyword_counts = df[[col for col in df.columns if col.startswith('has_')]].sum()
keyword_counts.plot(kind='bar', color='teal')
plt.title("Keyword Frequency in User Interests")
plt.xlabel("Keyword")
plt.ylabel("Number of Users Mentioning")
```

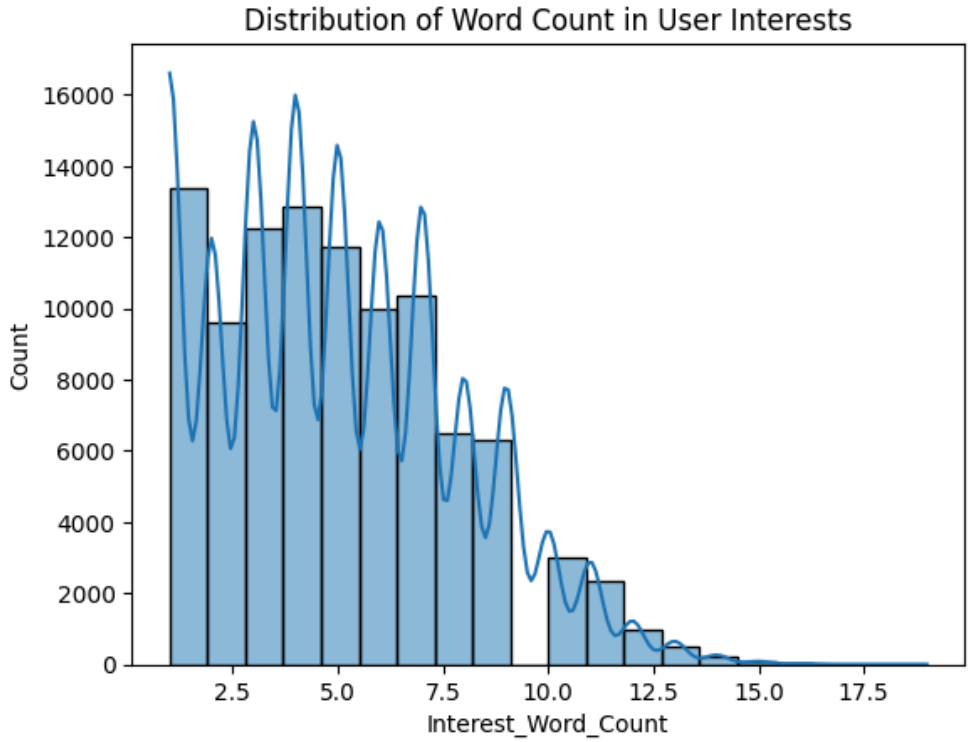
```
Text(0, 0.5, 'Number of Users Mentioning')
```



Distribution of Word Count in Interests

```
sns.histplot(df['Interest_Word_Count'], kde=True, bins=20)
plt.title("Distribution of Word Count in User Interests")
```

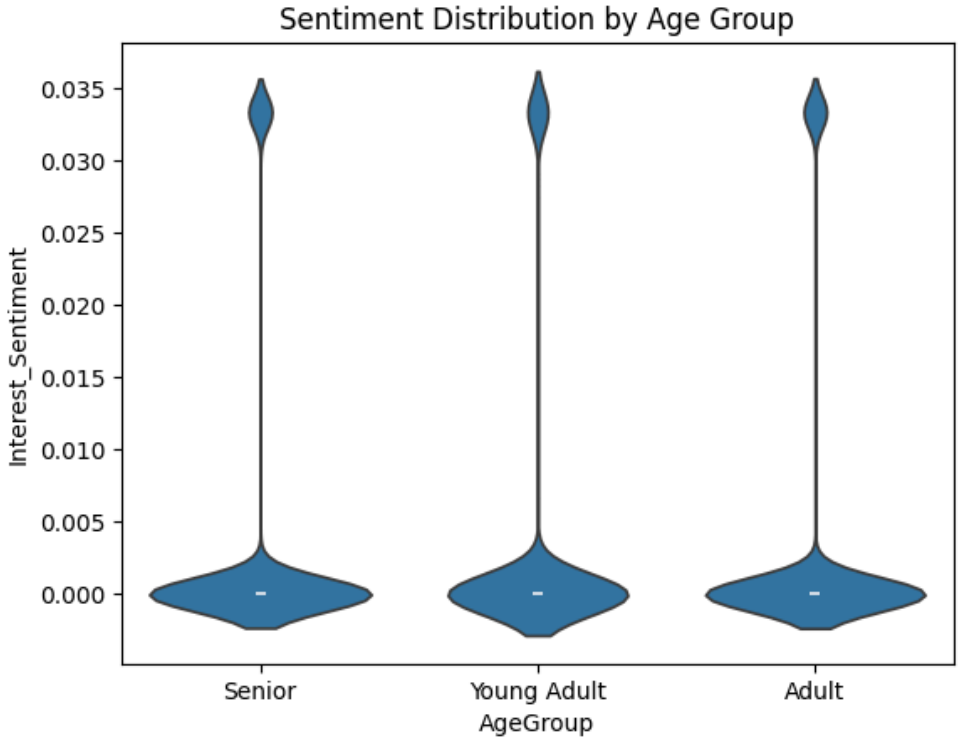
```
Text(0.5, 1.0, 'Distribution of Word Count in User Interests')
```



Violin Plot: Sentiment by Age Group

```
sns.violinplot(x='AgeGroup', y='Interest_Sentiment', data=df)
plt.title("Sentiment Distribution by Age Group")
```

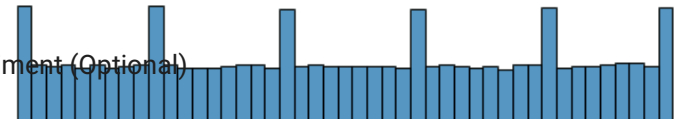
```
Text(0.5, 1.0, 'Sentiment Distribution by Age Group')
```



Jointplot: Age vs Interest Word Count

```
sns.jointplot(data=df, x='Age', y='Interest_Word_Count', kind='hex')
```

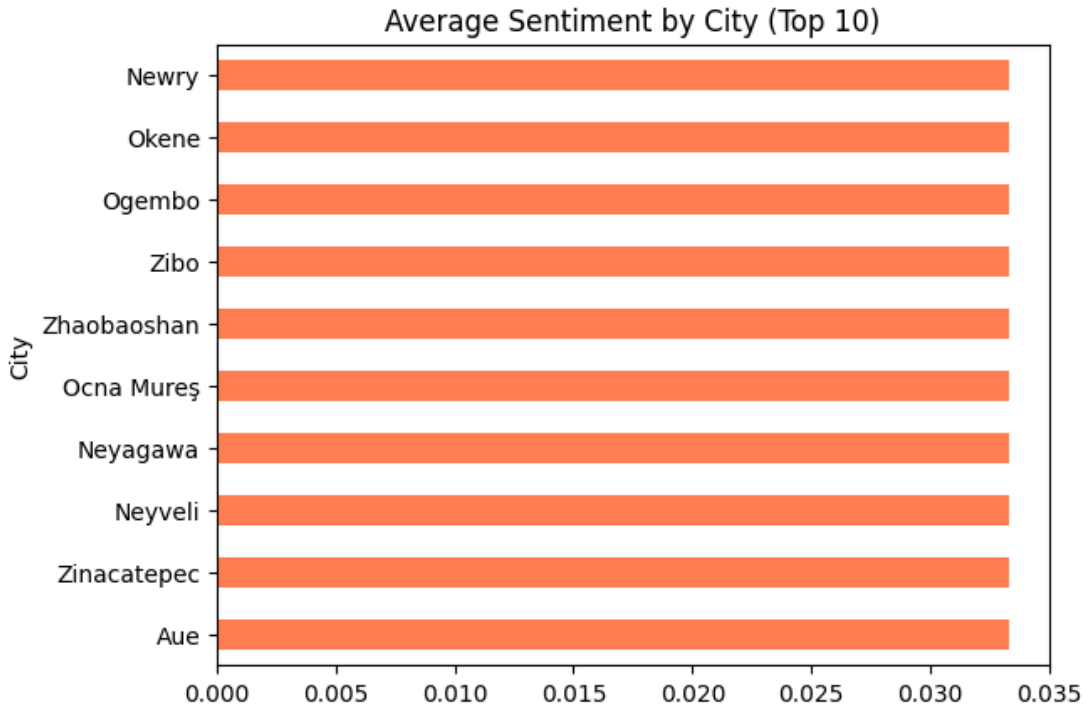
```
<seaborn.axisgrid.JointGrid at 0x796ddae82d50>
```



City vs Avg Sentiment (Optional)

```
city_sentiment = df.groupby('City')['Interest_Sentiment'].mean().sort_values(ascending=False).heac
city_sentiment.plot(kind='barh', color='coral')
plt.title("Average Sentiment by City (Top 10)")
```

```
Text(0.5, 1.0, 'Average Sentiment by City (Top 10)')
```



Age vs Keyword Flag Heatmap

```
age_keywords = df.groupby('AgeGroup')[[col for col in df.columns if col.startswith('has_')]].mean(
sns.heatmap(age_keywords, annot=True, cmap="YlGnBu")
plt.title("Avg Keyword Mentions by Age Group")
```

```
Text(0.5, 1.0, 'Avg Keyword Mentions by Age Group')
```