Week 22 Challenge - NLP

Vader Sentiment Analyzer from the Natural Language Toolkit (NLTK) library to perform sentiment analysis on the Twitter data.

The Vader Sentiment Analyzer is a lexicon and rule-based sentiment analysis tool that is specifically designed for analyzing sentiment in social media text, such as tweets. It assigns a sentiment score to each piece of text, indicating the overall sentiment polarity (positive, negative, or neutral) and intensity.

Before we begin with the mongoDB task, we need to make sure we have installed mongoDB - PyMongo

In [4]:
```
import pymongo
```

Connect the jupyter with the MongoDB server by using MongoDB details

In [5]:
```
client = pymongo.MongoClient("mongodb://localhost:27017/")
db = client["Trial"]
```

checking the list of database

In [6]:
```
print(client.list_database_names())
```

```
['Trial', 'admin', 'config', 'local']
```

In [7]:
```
import pandas as pd

# Load the Twitter dataset from the .json files
tweets_df = pd.read_csv(r"C:\Users\junai\OneDrive - Middlesex University\Applied Data Analytics\week 22\twitter_cleaned\tweets.cs
users_df = pd.read_csv(r"C:\Users\junai\OneDrive - Middlesex University\Applied Data Analytics\week 22\twitter_cleaned\users.csv"

# Display the first few rows of the datasets
tweets_df.head(), users_df.head()
```

Out[7]:
```
(      id             timestamp  \
 0   8653  2009-04-06T21:21:56Z
 1  12020  2009-04-06T21:22:47Z
 2  23858  2009-04-06T21:25:53Z
 3  51844  2009-04-06T21:33:11Z
 4  52341  2009-04-06T21:33:20Z

                                                 text
 0  falling asleep. just heard about that tracy gi...
 1  i have a sad feeling that dallas is not going ...
 2  @statravelau just got ur newsletter, those far...
 3  @djalizay i really don't think people choose t...
 4  my mind and body are severely protesting this ...  ,
       id           user    age
 0   8653    hpfangirl94    old
 1  12020     HybridMink  young
 2  23858  driveaway2008    old
 3  51844     lennytoups    old
 4  52341        Jemimus  young)
```

this code segment fetches data from the MongoDB collection named 'Trial', retrieves specific fields specified in the fields_to_retrieve dictionary, and stores the retrieved data in a list of dictionaries named data_list, making it easier to work with the data in Python.

In [8]:
```python
collection_name = 'Trial'

# Access the collection
collection = db[collection_name]

# Define fields to retrieve
fields_to_retrieve = {
    'user_id': 1,
    'tweet_content': 1,
    'timestamp': 1,
    'user_age': 1,
    # Add other relevant fields here
}

# Retrieve data from the collection
data = collection.find({}, fields_to_retrieve)

# Convert retrieved data to a list of dictionaries
data_list = list(data)
```

In [10]:
```
print(data_list)
```

```
[]
```

In [11]:
```
import pymongo

# Connection details
host = 'localhost:27017'
port = 27017
database_name = 'ADA_trial'

# Establish connection to MongoDB without authentication
client = pymongo.MongoClient(host, port)

# Access the database
db = client[database_name]
```

In [12]:
```
print(client.list_database_names())
```

```
['ADA_trial', 'Trial', 'admin', 'config', 'local']
```

In [43]:
```
import pandas as pd

# Load the Twitter dataset from the .json files
tweets_df = pd.read_csv(r"C:\Users\junai\OneDrive - Middlesex University\Applied Data Analytics\week 22\twitter_cleaned\tweets.cs
users_df = pd.read_csv(r"C:\Users\junai\OneDrive - Middlesex University\Applied Data Analytics\week 22\twitter_cleaned\users.csv"

# Display the first few rows of the datasets
tweets_df.head(), users_df.head()
```

Out[43]:
```
(      id             timestamp  \
 0   8653  2009-04-06T21:21:56Z
 1  12020  2009-04-06T21:22:47Z
 2  23858  2009-04-06T21:25:53Z
 3  51844  2009-04-06T21:33:11Z
 4  52341  2009-04-06T21:33:20Z

                                                 text
 0  falling asleep. just heard about that tracy gi...
 1  i have a sad feeling that dallas is not going ...
 2  @statravelau just got ur newsletter, those far...
 3  @djalizay i really don't think people choose t...
 4  my mind and body are severely protesting this ...  ,
       id           user    age
 0   8653    hpfangirl94    old
 1  12020     HybridMink  young
 2  23858  driveaway2008    old
 3  51844     lennytoups    old
 4  52341        Jemimus  young)
```

In [44]:
```python
# Convert DataFrame to dictionary (each row becomes a dictionary)
tweets_data_dict = tweets_df.to_dict(orient='records')

# Access the MongoDB collection for tweets
tweets_collection = db['tweets_collection']

# Insert tweets data into MongoDB collection
tweets_collection.insert_many(tweets_data_dict)

print("Tweets data inserted into MongoDB successfully!")
```

```
Tweets data inserted into MongoDB successfully!
```

In [45]:
```python
# Convert DataFrame to dictionary (each row becomes a dictionary)
users_data_dict = users_df.to_dict(orient='records')

# Access the MongoDB collection for users
users_collection = db['users_collection']

# Insert users data into MongoDB collection
users_collection.insert_many(users_data_dict)

print("Users data inserted into MongoDB successfully!")
```

```
Users data inserted into MongoDB successfully!
```

In [47]:
```python
# Assuming you have two DataFrames: tweets_df and users_df

# Merge the datasets based on the 'user_id' column
merged_df = pd.merge(tweets_df, users_df, on='id', how='left')
```

In [48]:
```python
# Access the MongoDB collection for tweets
tweets_collection = db['tweets_collection']

# Define fields to retrieve
tweet_fields = {
    'id': 1,
    'text': 1,
    'timestamp': 1,
    # Add more fields as needed
}

# Retrieve tweets data from MongoDB collection
tweets_data = tweets_collection.find({}, tweet_fields)

# Convert retrieved tweets data to a list of dictionaries
tweets_data_list = list(tweets_data)
```

In [49]:
```python
# Access the MongoDB collection for users
users_collection = db['users_collection']

# Define fields to retrieve
user_fields = {
    'id': 1,
    'user': 1,
    'age': 1,
    # Add more fields as needed
}

# Retrieve user data from MongoDB collection
users_data = users_collection.find({}, user_fields)

# Convert retrieved user data to a list of dictionaries
users_data_list = list(users_data)
```

In [50]:
```python
import pandas as pd
```

```
# Convert retrieved tweets data to a pandas DataFrame
tweets_df = pd.DataFrame(tweets_data_list)

# Convert retrieved users data to a pandas DataFrame
users_df = pd.DataFrame(users_data_list)
```

NLTK stands for Natural Language Toolkit. It is a leading platform for building Python programs to work with human language data. NLTK provides easy-to-use interfaces and libraries for tasks such as tokenization, stemming, tagging, parsing, and semantic reasoning.

Stopwords are common words that are often filtered out during text preprocessing because they typically do not carry significant meaning or information for analysis.

Stemming is the process of reducing words to their base or root form. The Porter stemming algorithm is a widely used stemming algorithm that applies a set of rules to reduce words to their stems. Stemming helps in reducing the dimensionality of the text data and can improve the performance of text analysis tasks such as information retrieval or classification.

In [51]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import string

# Download NLTK resources (run this only once)
nltk.download('punkt')
nltk.download('stopwords')

# Initialize NLTK components
stop_words = set(stopwords.words('english'))
ps = PorterStemmer()

# Function to preprocess text
def preprocess_text(text):
    # Tokenization
    tokens = word_tokenize(text)

    # Lowercasing
    tokens = [token.lower() for token in tokens]

    # Removing punctuation
    tokens = [token for token in tokens if token not in string.punctuation]
```

```
    # Removing stopwords
    tokens = [token for token in tokens if token not in stop_words]

    # Stemming (optional)
    # tokens = [ps.stem(token) for token in tokens]

    # Join tokens back into text
    preprocessed_text = ' '.join(tokens)

    return preprocessed_text

# Apply preprocessing to tweet content
tweets_df['clean_tweet'] = tweets_df['text'].apply(preprocess_text)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\junai\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\junai\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [52]:
```
print(tweets_df.head())
```

```
                        _id     id             timestamp  \
0  65f8dbecaad7e1701be77c0e   8653  2009-04-06T21:21:56Z
1  65f8dbecaad7e1701be77c0f  12020  2009-04-06T21:22:47Z
2  65f8dbecaad7e1701be77c10  23858  2009-04-06T21:25:53Z
3  65f8dbecaad7e1701be77c11  51844  2009-04-06T21:33:11Z
4  65f8dbecaad7e1701be77c12  52341  2009-04-06T21:33:20Z

                                             text  \
0  falling asleep. just heard about that tracy gi...
1  i have a sad feeling that dallas is not going ...
2  @statravelau just got ur newsletter, those far...
3  @djalizay i really don't think people choose t...
4  my mind and body are severely protesting this ...

                                        clean_tweet
0  falling asleep heard tracy girl 's body found ...
1  sad feeling dallas going show got ta say thoug...
2  statravelau got ur newsletter fares really unb...
3  djalizay really n't think people choose way th...
4  mind body severely protesting quot getting quo...
```

In [53]:
```
# Select a sample tweet and print original and preprocessed text
sample_tweet_index = 0  # Change this to the index of the tweet you want to inspect
original_tweet = tweets_df.loc[sample_tweet_index, 'text']
preprocessed_tweet = tweets_df.loc[sample_tweet_index, 'clean_tweet']

print("Original Tweet:")
print(original_tweet)
print("\nPreprocessed Tweet:")
print(preprocessed_tweet)
```

```
Original Tweet:
falling asleep. just heard about that tracy girl's body being found. how sad  my heart breaks for that family.

Preprocessed Tweet:
falling asleep heard tracy girl 's body found sad heart breaks family
```

In [54]:
```
# Check length of preprocessed text for all tweets
tweets_df['clean_tweet_length'] = tweets_df['clean_tweet'].apply(len)
print(tweets_df['clean_tweet_length'].describe())
```

```
count    62434.000000
mean        64.468046
std         25.388203
min          5.000000
25%         45.000000
50%         65.000000
75%         84.000000
max        154.000000
Name: clean_tweet_length, dtype: float64
```

this code snippet demonstrates how to use NLTK's Vader Sentiment Analyzer to analyze the sentiment of tweets and add sentiment scores to a DataFrame for further analysis or visualization.

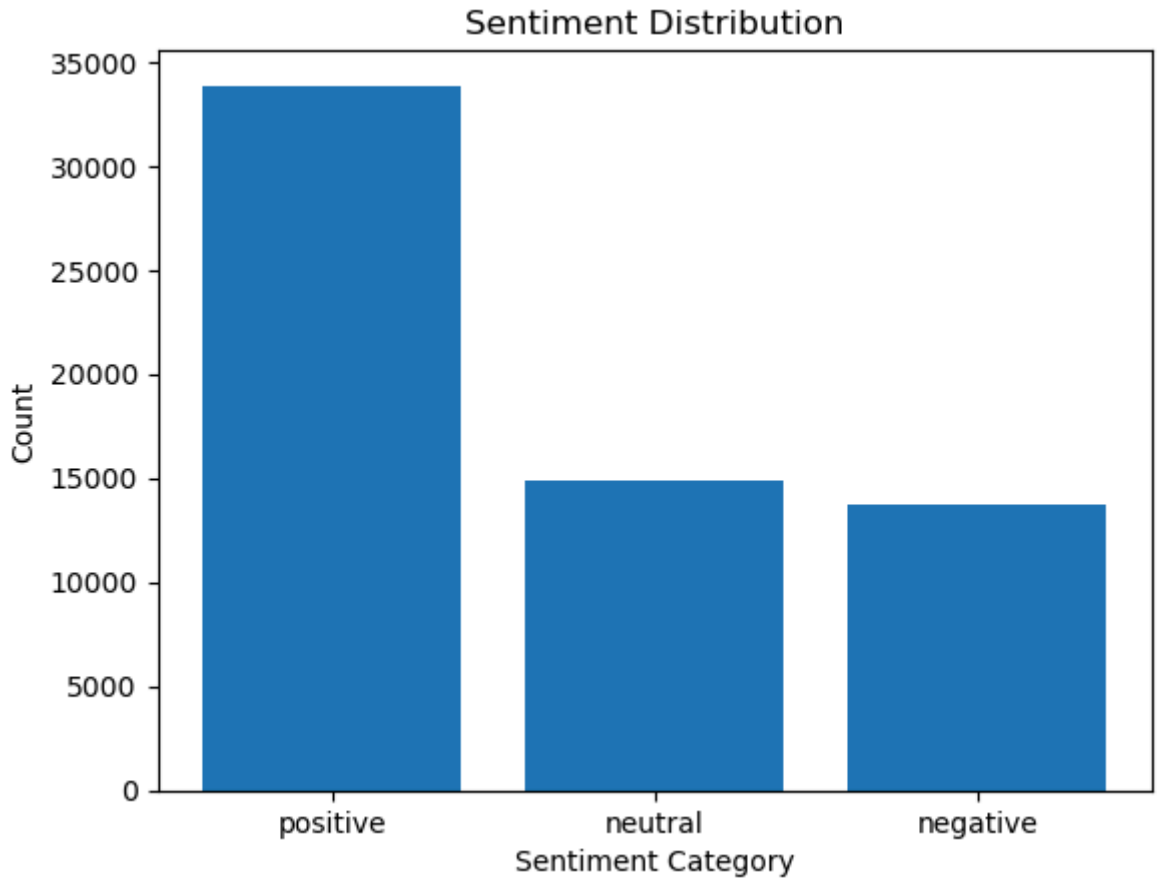
This analyzer is pre-trained on sentiment analysis tasks and is specifically tailored for social media text like tweets.This code defines a function get_sentiment_score(text) that takes a piece of text as input and returns the compound sentiment score. The compound score is a normalized score ranging from -1 (extremely negative) to 1 (extremely positive), with values closer to 0 indicating neutral sentiment.

In [55]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Initialize the Vader Sentiment Analyzer
analyzer = SentimentIntensityAnalyzer()
```

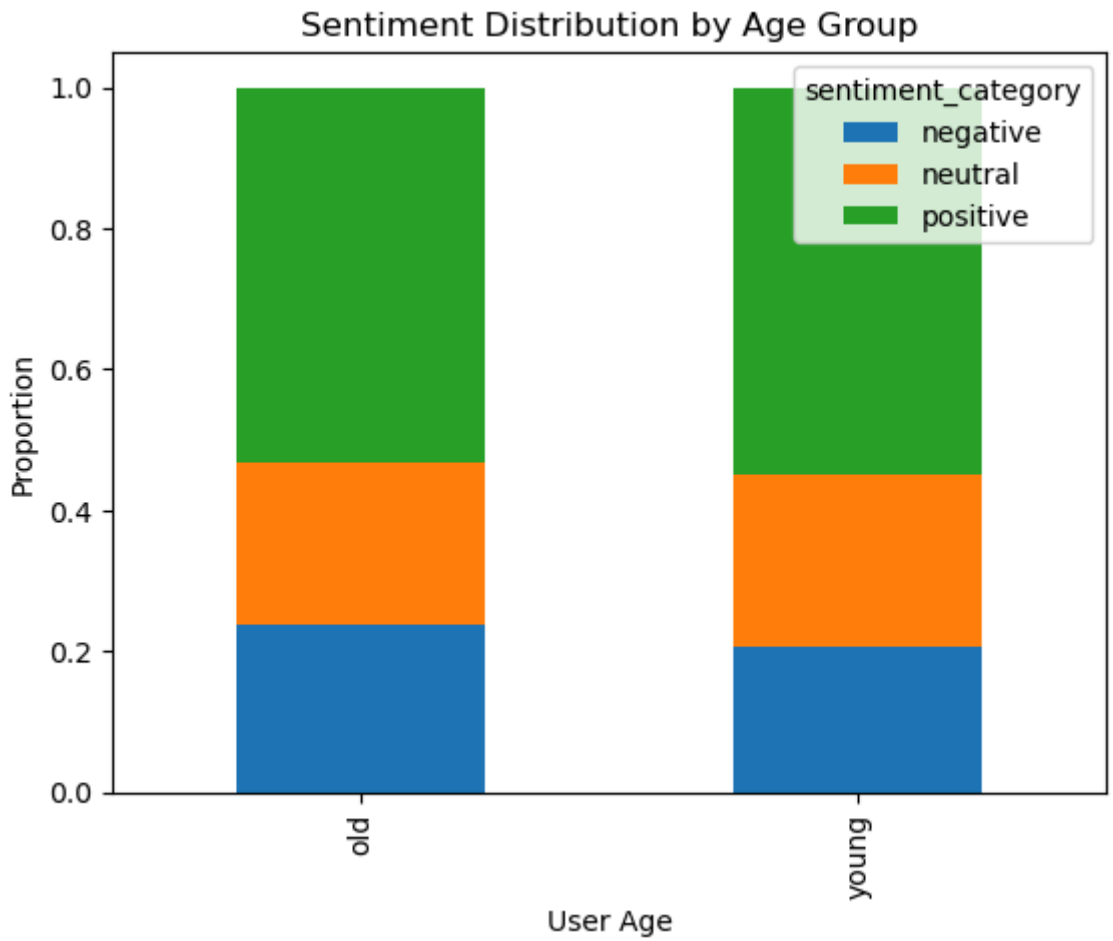
```
# Function to get sentiment score for a text
def get_sentiment_score(text):
    return analyzer.polarity_scores(text)['compound']

# Apply sentiment analysis to the tweet content
tweets_df['sentiment_score'] = tweets_df['text'].apply(get_sentiment_score)
```

In [56]:
```
# Function to categorize sentiment
def categorize_sentiment(score):
    if score > 0.05:
        return 'positive'
    elif score < -0.05:
        return 'negative'
    else:
        return 'neutral'

# Apply sentiment categorization
tweets_df['sentiment_category'] = tweets_df['sentiment_score'].apply(categorize_sentiment)
```

In [57]:
```
import matplotlib.pyplot as plt

# Plot sentiment distribution
sentiment_distribution = tweets_df['sentiment_category'].value_counts()
plt.bar(sentiment_distribution.index, sentiment_distribution.values)
plt.xlabel('Sentiment Category')
plt.ylabel('Count')
plt.title('Sentiment Distribution')
plt.show()
```



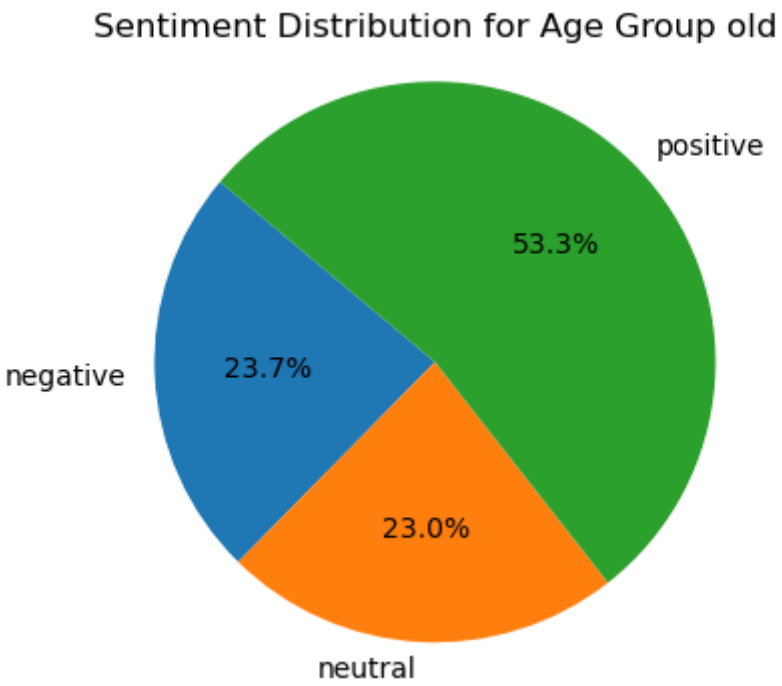
In [58]:
```
# Assuming 'users_df' contains user demographics including age

# Merge 'tweets_df' with 'users_df' based on 'user_id' column
merged_df = pd.merge(tweets_df, users_df, on='id', how='left')

# Analyze sentiment distribution across different age groups
sentiment_by_age = merged_df.groupby('age')['sentiment_category'].value_counts(normalize=True).unstack()
sentiment_by_age.plot(kind='bar', stacked=True)
plt.xlabel('User Age')
plt.ylabel('Proportion')
plt.title('Sentiment Distribution by Age Group')
plt.show()
```

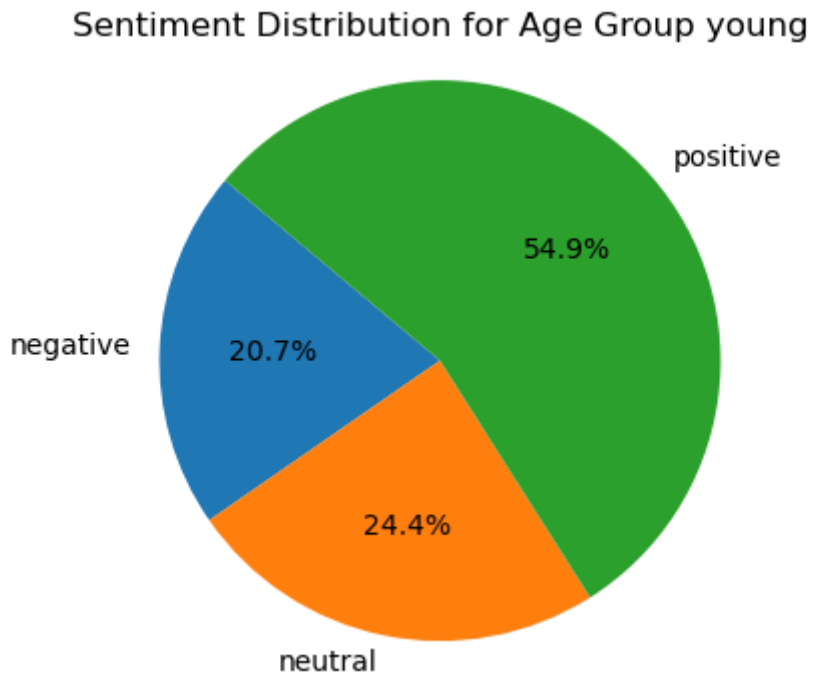


In [59]:

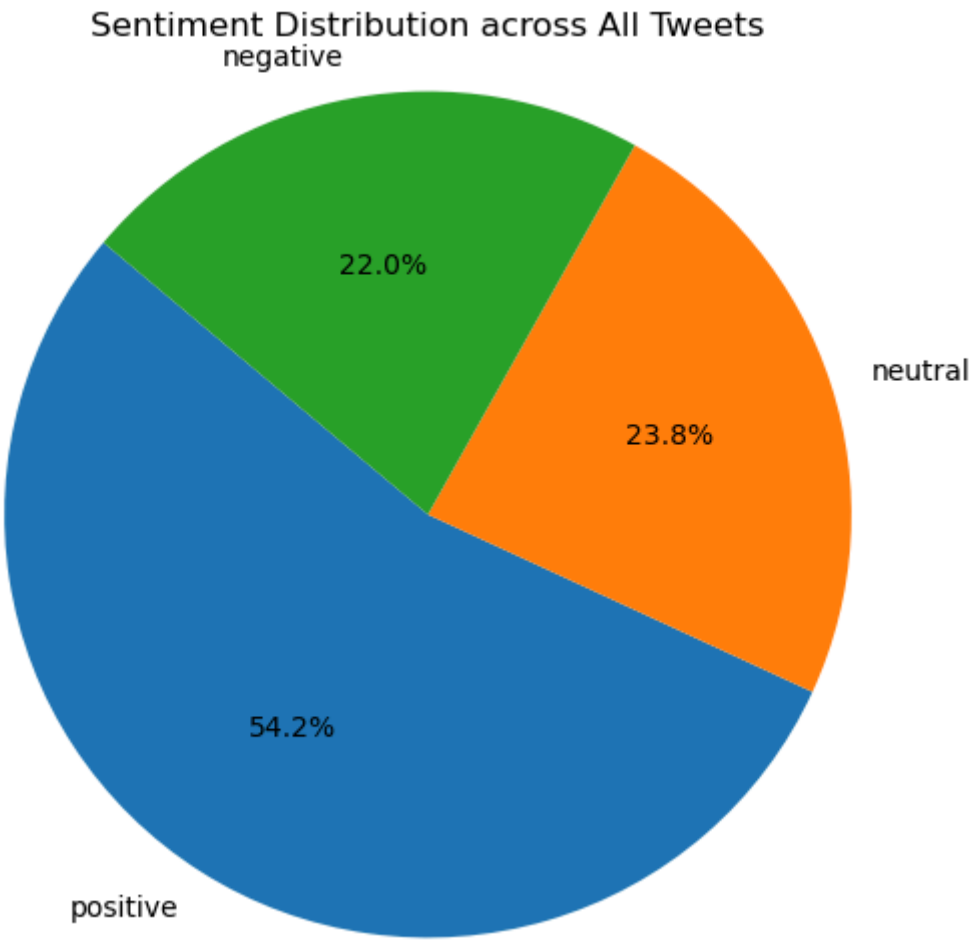
```python
import matplotlib.pyplot as plt

# Analyze sentiment distribution across different age groups
sentiment_by_age = merged_df.groupby('age')['sentiment_category'].value_counts(normalize=True).unstack()

# Plot pie charts for each age group
for age_group in sentiment_by_age.index:
    sentiment_distribution = sentiment_by_age.loc[age_group]
    plt.figure(figsize=(6, 4))  # Adjust figure size as needed
    plt.pie(sentiment_distribution, labels=sentiment_distribution.index, autopct='%1.1f%%', startangle=140)
    plt.title(f'Sentiment Distribution for Age Group {age_group}')
```

```
    plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
    plt.show()
```





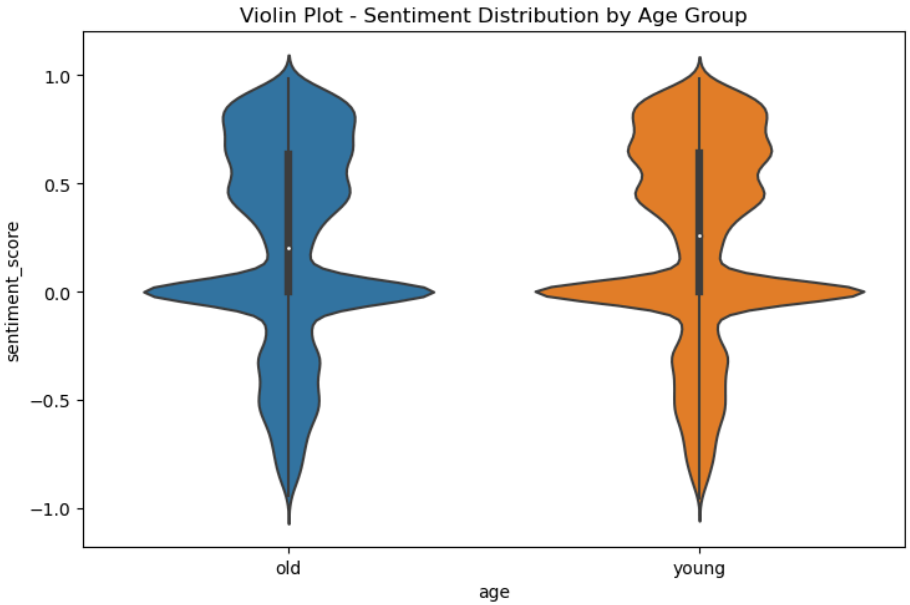
In [60]:
```python
import matplotlib.pyplot as plt

# Plotting sentiment distribution using a pie chart
sentiment_counts = tweets_df['sentiment_category'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Sentiment Distribution across All Tweets')
plt.axis('equal')
plt.show()
```

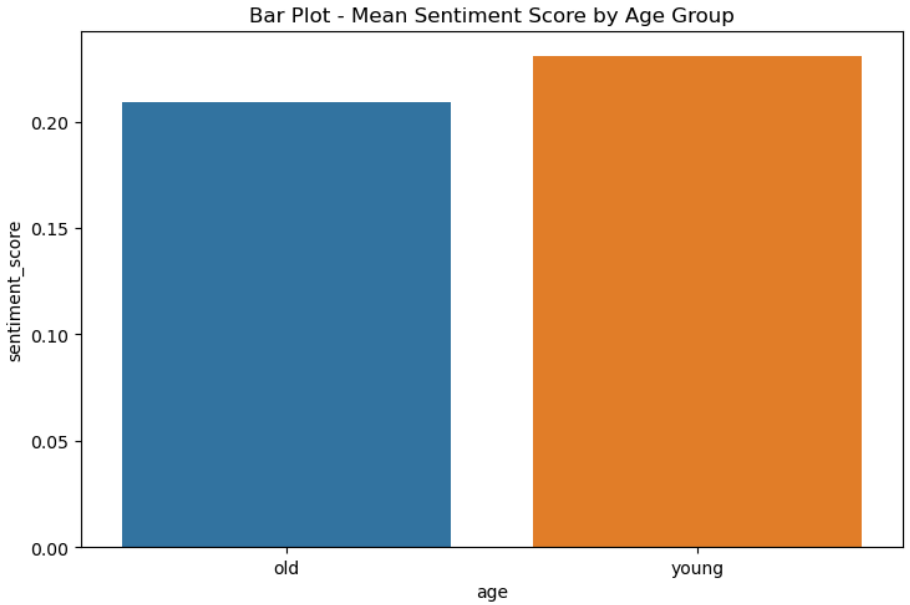


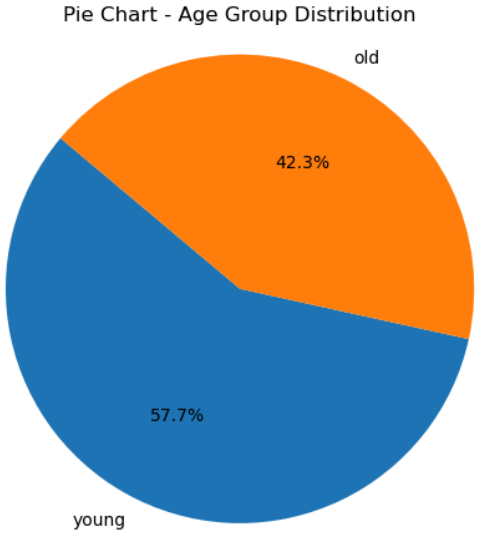
In [67]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Create subplots
fig, axs = plt.subplots(2, 2, figsize=(15, 10))

# Plot Violin Plot
sns.violinplot(x='age', y='sentiment_score', data=merged_df, ax=axs[0, 0])
axs[0, 0].set_title('Violin Plot - Sentiment Distribution by Age Group')

# Plot Bar Plot
sns.barplot(x='age', y='sentiment_score', data=merged_df, ci=None, ax=axs[0, 1])
```

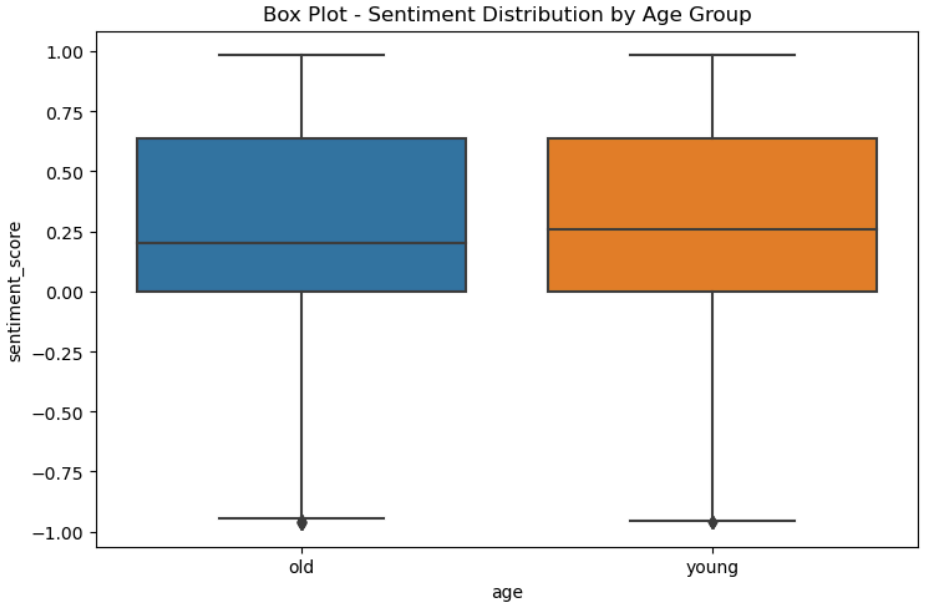
```
axs[0, 1].set_title('Bar Plot - Mean Sentiment Score by Age Group')

# Plot Pie Chart
age_counts = merged_df['age'].value_counts()
axs[1, 0].pie(age_counts, labels=age_counts.index, autopct='%1.1f%%', startangle=140)
axs[1, 0].set_title('Pie Chart - Age Group Distribution')
axs[1, 0].axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

# Plot Box Plot
sns.boxplot(x='age', y='sentiment_score', data=merged_df, ax=axs[1, 1])
axs[1, 1].set_title('Box Plot - Sentiment Distribution by Age Group')

plt.tight_layout()
plt.show()
```

```
C:\Users\junai\AppData\Local\Temp\ipykernel_11148\3937623405.py:12: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='age', y='sentiment_score', data=merged_df, ci=None, ax=axs[0, 1])
```









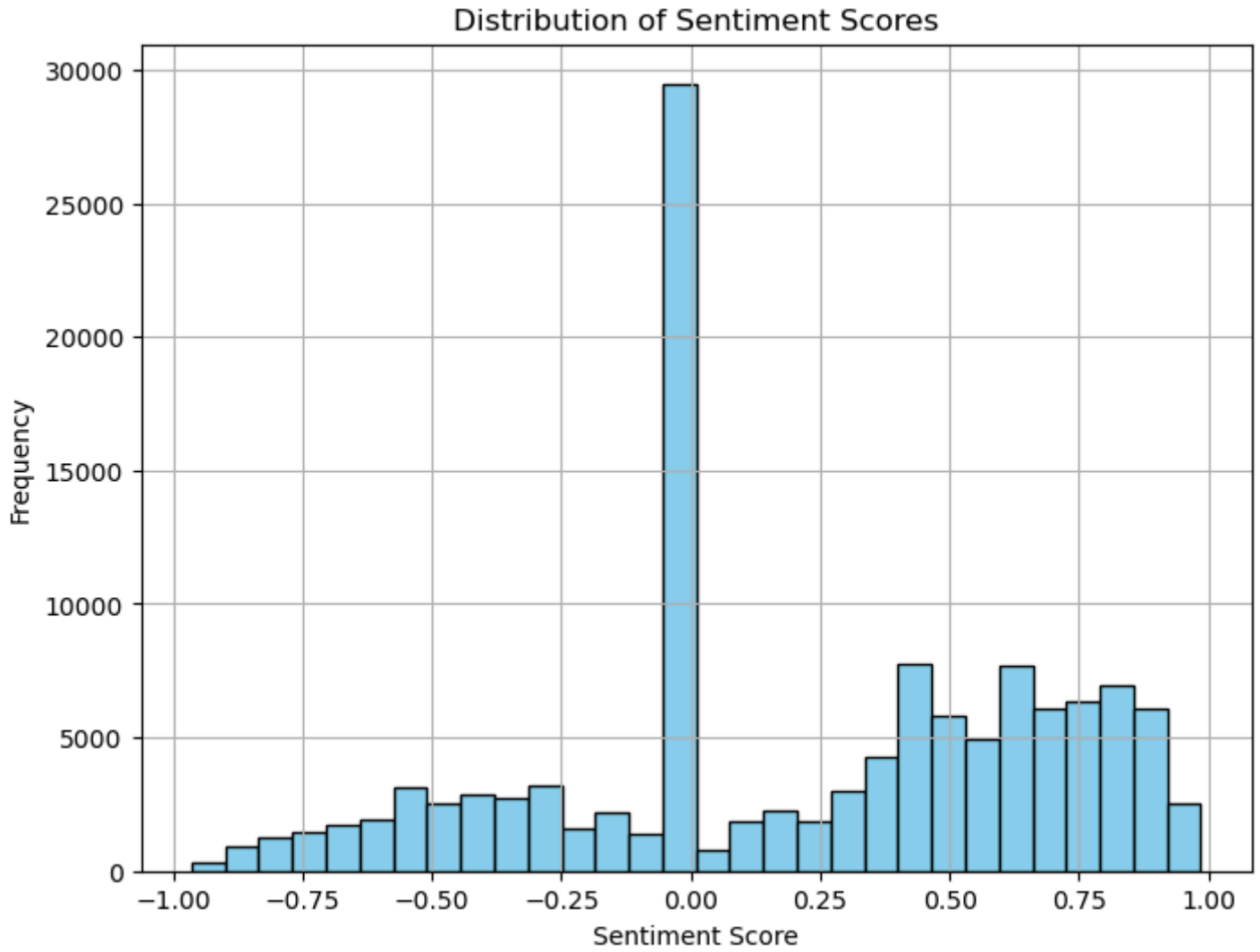
In [68]:
```
# Assuming 'merged_df' is your DataFrame containing tweet data and sentiment scores
print(merged_df['sentiment_score'])
```

```
0        -0.5719
1        -0.5719
2        -0.3818
3        -0.3818
4        -0.3150
           ...
124863    0.6395
124864    0.8625
124865    0.8625
124866   -0.1531
124867   -0.1531
Name: sentiment_score, Length: 124868, dtype: float64
```

In [69]:
```python
# View the first few rows of the DataFrame along with sentiment scores
print(merged_df[['text', 'sentiment_score']].head())
```

```
                                                text  sentiment_score
0  falling asleep. just heard about that tracy gi...          -0.5719
1  falling asleep. just heard about that tracy gi...          -0.5719
2  i have a sad feeling that dallas is not going ...          -0.3818
3  i have a sad feeling that dallas is not going ...          -0.3818
4  @statravelau just got ur newsletter, those far...          -0.3150
```

In [70]:
```python
import matplotlib.pyplot as plt

# Plot histogram of sentiment scores
plt.figure(figsize=(8, 6))
plt.hist(merged_df['sentiment_score'], bins=30, color='skyblue', edgecolor='black')
plt.xlabel('Sentiment Score')
plt.ylabel('Frequency')
plt.title('Distribution of Sentiment Scores')
plt.grid(True)
plt.show()
```


Distribution of Sentiment Scores
Frequency
Sentiment Score
30000
25000
20000
15000
10000
5000
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

In [ ]: