In [ ]:
```
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
import pandas as pd
from bs4 import BeautifulSoup
import re

# Download the VADER lexicon for sentiment analysis
nltk.download('vader_lexicon')

def preprocess_text(text):
    # Remove HTML tags
    text = BeautifulSoup(text, "html.parser").get_text()
    # Remove non-alphabetic characters
    text = re.sub("[^a-zA-Z]", " ", text)
    # Convert to lowercase
    text = text.lower()
    # Remove extra whitespace
    text = " ".join(text.split())
    return text

def analyze_sentiment(email_text):
    sia = SentimentIntensityAnalyzer()
    sentiment = sia.polarity_scores(email_text)
    return sentiment

def categorize_sentiment(sentiment_score):
    if sentiment_score['compound'] >= 0.05:
        return 'Positive'
    elif sentiment_score['compound'] <= -0.05:
        return 'Negative'
    else:
        return 'Neutral'

def generate_feedback(sentiment, subject, body):
    feedback_templates = {
        'Negative': "**Action Required:** {body_summary} We need yo
        'Neutral': "**Information:** {body_summary}. Please let us
        'Positive': """**Thank you!** We appreciate your {subject_k
    }

    # Summarize body (optional, adjust based on your needs)
    body_summary = " ".join(body.split()[:10])  # Take the first 10

    subject_keywords = " ".join(word for word in subject.lower().sp


    # Define body_summary within the function
    if body:  # Check if body is not empty
        body_summary = " ".join(body.split()[:10])  # Summarize if
    else:
        body_summary = "No body content available."
```

```
  # Choose appropriate template and format feedback
    template = feedback_templates[sentiment]
    feedback = template.format(body_summary=body_summary, date="{20
    return feedback

# Read the email dataset
emails_df = pd.read_csv('email_dataset.csv')

# Preprocess the email text
emails_df['cleaned_email_text'] = emails_df['Body Description'].app

# Analyze sentiment
emails_df['sentiment_scores'] = emails_df['cleaned_email_text'].app

# Categorize sentiment
emails_df['sentiment'] = emails_df['sentiment_scores'].apply(catego

# Generate feedback
emails_df['feedback'] = emails_df.apply(lambda row: generate_feedba

# Expand sentiment scores into separate columns
sentiment_df = emails_df['sentiment_scores'].apply(pd.Series)

# Merge sentiment scores with the original dataframe
result_df = pd.concat([emails_df, sentiment_df], axis=1)
# Display the results
print(result_df['feedback'].head(5))# Save the result to a new CSV
result_df.to_csv('analyzed_emails_with_priority.csv', index=False)
```