In [14]:
```python
# Task 1: Sentiment Labeling Code

# Import necessary libraries
from textblob import TextBlob
import pandas as pd
import re

# Load the dataset
file_path = 'test(in).csv'
data = pd.read_csv(file_path)

# Text Preprocessing: Clean the text
def simple_preprocess_text(text):
    text = re.sub(r'[^A-Za-z\s]', '', text)  # Remove unwanted characters, r
    text = text.lower()  # Convert to lowercase
    text = text.strip()  # Remove leading/trailing spaces
    return text

data['cleaned_body'] = data['body'].apply(simple_preprocess_text)

# Sentiment Labeling using TextBlob
def get_sentiment(text):
    blob = TextBlob(text)
    sentiment_score = blob.sentiment.polarity
    if sentiment_score > 0:
        return 'Positive'
    elif sentiment_score < 0:
        return 'Negative'
    else:
        return 'Neutral'

data['sentiment'] = data['cleaned_body'].apply(get_sentiment)

# Check the first few rows after sentiment labeling
print(data[['body', 'sentiment']].head())
```

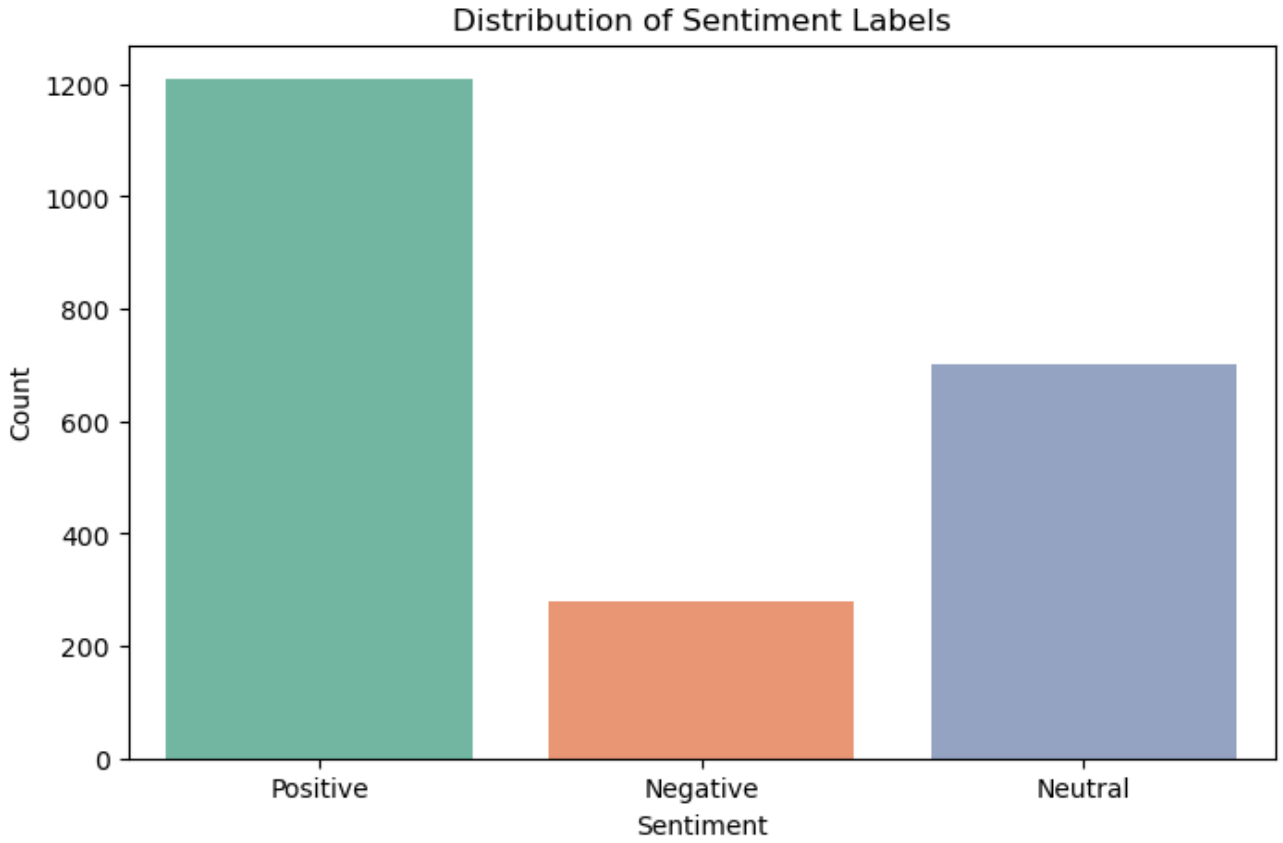
```
                                                body sentiment
0  EnronOptions Announcement\n\n\nWe have updated...  Positive
1  Marc,\n\nUnfortunately, today is not going to ...  Negative
2  When: Wednesday, June 06, 2001 10:00 AM-11:00 ...   Neutral
3  we were thinking papasitos (we can meet somewh...  Negative
4  Since you never gave me the $20 for the last t...  Negative
```

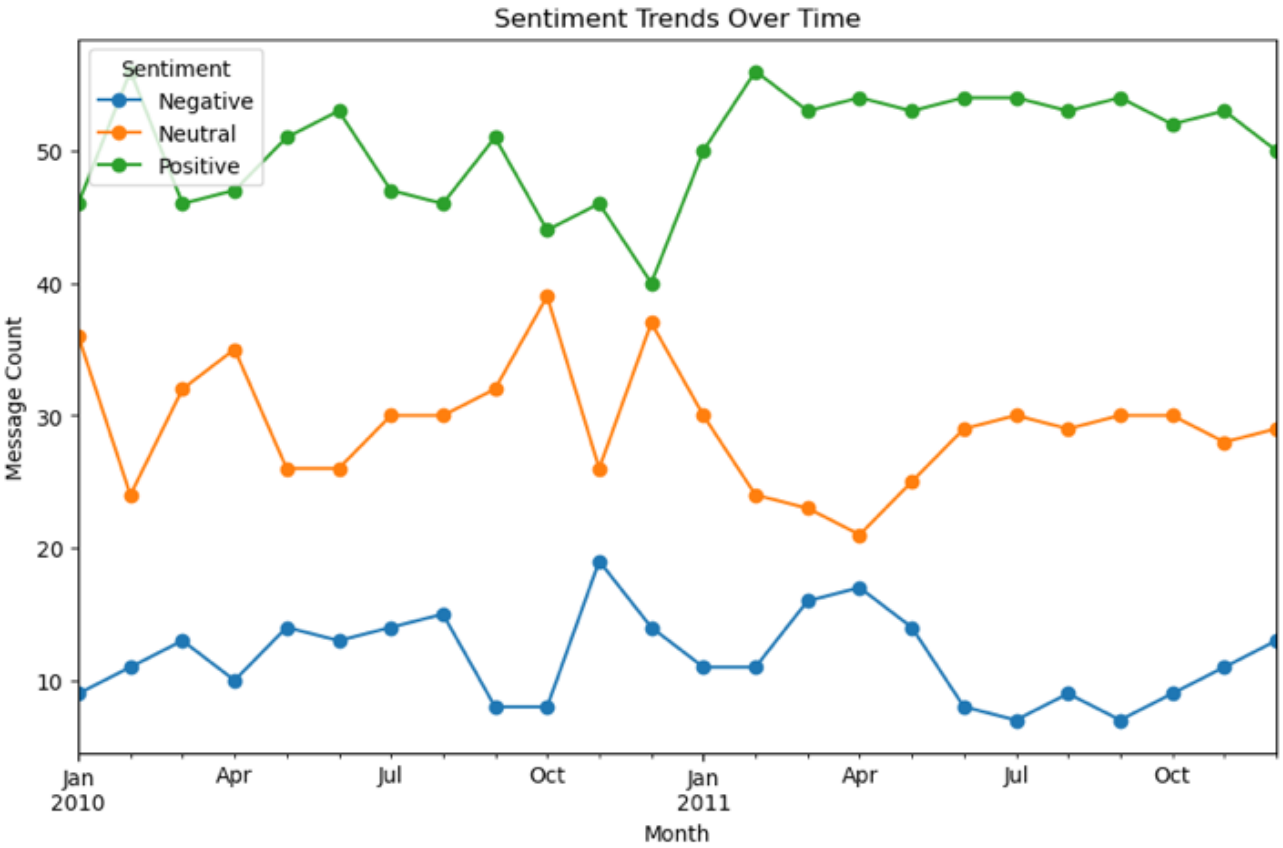
In [15]:
```python
# Task 2: Exploratory Data Analysis (EDA)

import matplotlib.pyplot as plt
import seaborn as sns

# Check data structure
```

```
print(data.info())

# Check for missing values
print(data.isnull().sum())

# Distribution of sentiment labels
sentiment_counts = data['sentiment'].value_counts()

# Plot the distribution of sentiment labels
plt.figure(figsize=(8, 5))
sns.countplot(x='sentiment', data=data, palette='Set2')
plt.title('Distribution of Sentiment Labels')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()

# Sentiment distribution over time
# Convert date column to datetime format
data['date'] = pd.to_datetime(data['date'], errors='coerce')
data['month'] = data['date'].dt.to_period('M')

# Plot sentiment trends over months
monthly_sentiment = data.groupby(['month', 'sentiment']).size().unstack().fi

monthly_sentiment.plot(kind='line', figsize=(10, 6), marker='o')
plt.title('Sentiment Trends Over Time')
plt.xlabel('Month')
plt.ylabel('Message Count')
plt.legend(title='Sentiment', loc='upper left')
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2191 entries, 0 to 2190
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Subject       2191 non-null   object
 1   body          2191 non-null   object
 2   date          2191 non-null   object
 3   from          2191 non-null   object
 4   cleaned_body  2191 non-null   object
 5   sentiment     2191 non-null   object
dtypes: object(6)
memory usage: 102.8+ KB
None
Subject         0
body            0
date            0
from            0
cleaned_body    0
sentiment       0
dtype: int64
```

```
/var/folders/ks/jy925sb56tsg1fkcfkb47kkr0000gn/T/ipykernel_81655/2666615492.
py:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.countplot(x='sentiment', data=data, palette='Set2')
```





In [16]:
```
# Task 3: Employee Score Calculation
```

```
# Convert date column to datetime format (if not already done)
data['date'] = pd.to_datetime(data['date'], errors='coerce')

# Add a 'month' column to group by month
data['month'] = data['date'].dt.to_period('M')

# Map sentiment to score: Positive = 1, Negative = -1, Neutral = 0
sentiment_score_mapping = {'Positive': 1, 'Negative': -1, 'Neutral': 0}
data['score'] = data['sentiment'].map(sentiment_score_mapping)

# Group by employee and month, and sum the scores for each group
monthly_scores = data.groupby(['from', 'month'])['score'].sum().reset_index(

# Display the first few rows of the monthly scores
print(monthly_scores.head())
```

```
                         from    month  score
0  bobette.riner@ipgdirect.com  2010-01      2
1  bobette.riner@ipgdirect.com  2010-02      8
2  bobette.riner@ipgdirect.com  2010-03      4
3  bobette.riner@ipgdirect.com  2010-04      4
4  bobette.riner@ipgdirect.com  2010-05      2
```

In [17]:
```
# Task 4: Employee Ranking

# For positive employees, take top 3 per month
top_positive = monthly_scores.sort_values(by=['month', 'score'], ascending=[

# Filter out only negative sentiment scores for top negative employees
negative_scores = monthly_scores[monthly_scores['score'] < 0]

# Sort by score in ascending order to get the most negative employees
top_negative = negative_scores.sort_values(by=['month', 'score'], ascending=

# Display the top positive and negative employees
print("Top Positive Employees:")
print(top_positive[['from', 'month', 'score']])

print("\nTop Negative Employees:")
print(top_negative[['from', 'month', 'score']])
```

```
Top Positive Employees:
                            from    month  score
120      kayne.coulter@enron.com  2010-01      9
24        don.baughman@enron.com  2010-01      5
48           eric.bass@enron.com  2010-01      5
73         john.arnold@enron.com  2010-02     10
1    bobette.riner@ipgdirect.com  2010-02      8
..                           ...      ...    ...
142      kayne.coulter@enron.com  2011-11      7
190     patti.thompson@enron.com  2011-11      7
167      lydia.delgado@enron.com  2011-12      6
191     patti.thompson@enron.com  2011-12      6
143      kayne.coulter@enron.com  2011-12      5

[72 rows x 3 columns]

Top Negative Employees:
                          from    month  score
222      sally.beck@enron.com  2010-07     -2
225      sally.beck@enron.com  2010-10     -1
179  patti.thompson@enron.com  2010-12     -1
203   rhonda.denton@enron.com  2010-12     -1
132   kayne.coulter@enron.com  2011-01     -1
230      sally.beck@enron.com  2011-03     -1
184  patti.thompson@enron.com  2011-05     -1
```

In [18]:
```
# Task 5: Flight Risk Identification

# Calculate the rolling count of negative messages for each employee
data['negative_flag'] = data['sentiment'] == 'Negative'

# Create a 30-day rolling window to count the number of negative messages
data['rolling_negative_count'] = data.groupby('from')['negative_flag'].rolli

# Identify flight risk employees (those with 4 or more negative messages in
flight_risk_employees = data[data['rolling_negative_count'] >= 4]['from'].ur

print("Flight Risk Employees:")
print(flight_risk_employees)
```

```
Flight Risk Employees:
['johnny.palmer@enron.com' 'john.arnold@enron.com'
 'lydia.delgado@enron.com' 'bobette.riner@ipgdirect.com'
 'eric.bass@enron.com' 'sally.beck@enron.com' 'patti.thompson@enron.com'
 'kayne.coulter@enron.com' 'rhonda.denton@enron.com'
 'don.baughman@enron.com']
```

In [19]:
```
# Task 6: Predictive Modeling

from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Add 'month_num' to monthly_scores dataframe
monthly_scores['month_num'] = monthly_scores['month'].dt.month

# Feature engineering: create message frequency (message count)
monthly_scores['message_count'] = data.groupby(['from', 'month'])['score'].t

# Create feature and target variables
X = monthly_scores[['month_num', 'message_count']]
y = monthly_scores['score']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar

# Train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 9.995379830973505
R-squared: -0.11662893379378181
```

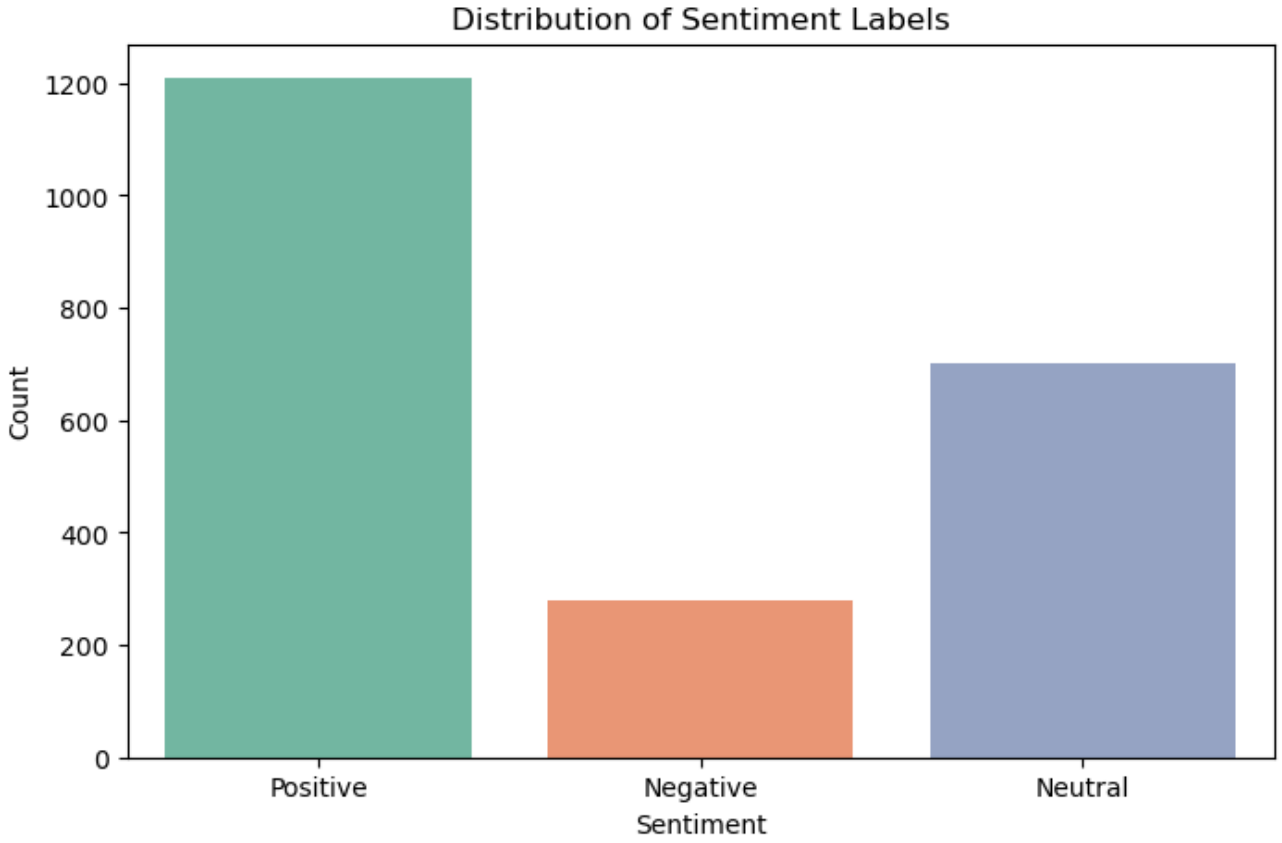
In [20]:

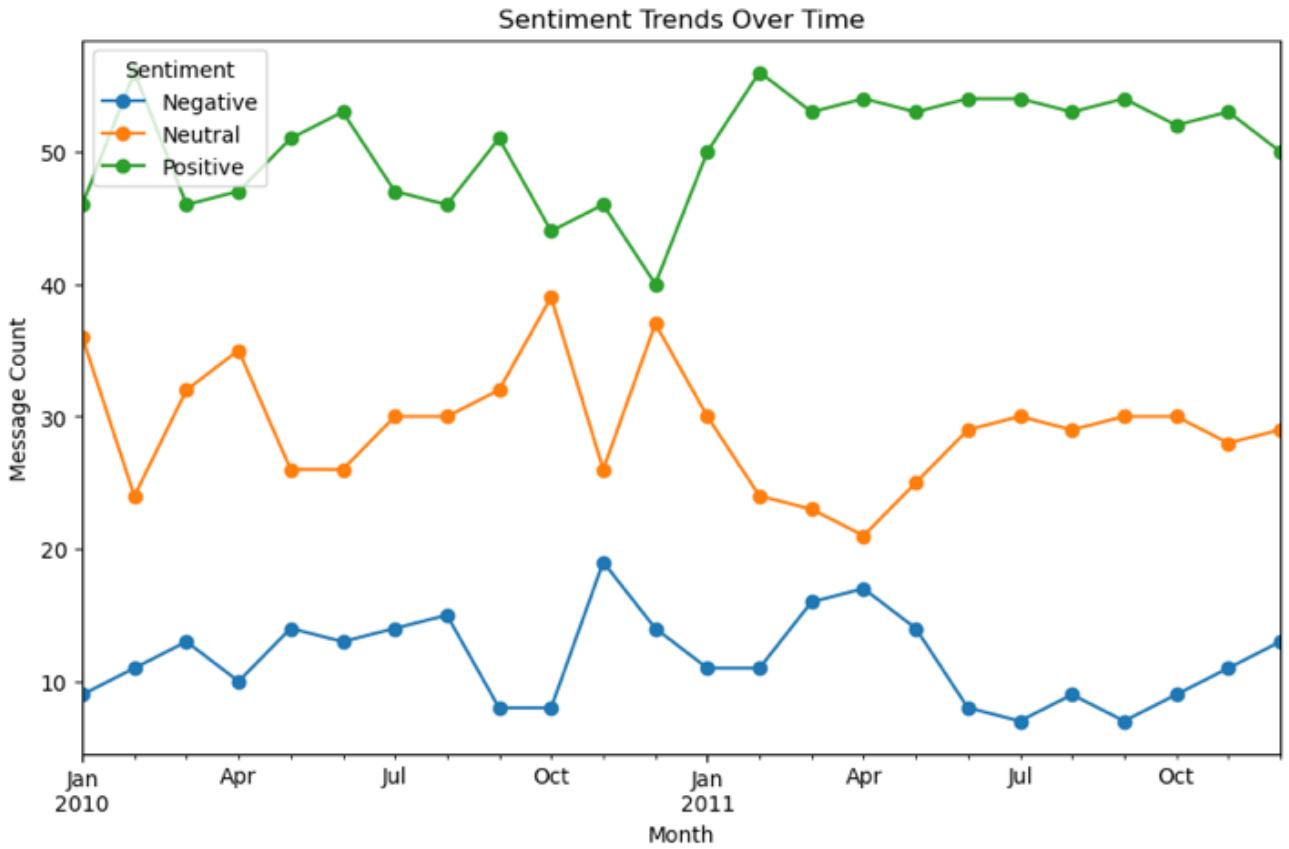
```
import matplotlib.pyplot as plt
import seaborn as sns
import os

# 1. Create directory for saving visualizations
output_dir = '/Users/seankwon/Documents/GitHub/employee_analysis/Visualizati
os.makedirs(output_dir, exist_ok=True)

# 2. Sentiment Distribution Plot
plt.figure(figsize=(8, 5))
sns.countplot(x='sentiment', data=data, palette='Set2')
plt.title('Distribution of Sentiment Labels')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.savefig(f'{output_dir}/sentiment_distribution.png')
plt.show()
```

```
# 3. Sentiment Trends Over Time
monthly_sentiment = data.groupby(['month', 'sentiment']).size().unstack().fi
monthly_sentiment.plot(kind='line', figsize=(10, 6), marker='o')
plt.title('Sentiment Trends Over Time')
plt.xlabel('Month')
plt.ylabel('Message Count')
plt.legend(title='Sentiment', loc='upper left')
plt.savefig(f'{output_dir}/sentiment_trends.png')
plt.show()

# 4. Top Positive and Negative Employees Visualization
# For positive employees, take top 3 per month
top_positive = monthly_scores.sort_values(by=['month', 'score'], ascending=[
top_negative = monthly_scores[monthly_scores['score'] < 0].sort_values(by=['

fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Plot for top positive employees
sns.countplot(x='from', data=top_positive, palette='Set1', ax=axes[0])
axes[0].set_title('Top Positive Employees')
axes[0].set_xlabel('Employee')
axes[0].set_ylabel('Count')

# Plot for top negative employees
sns.countplot(x='from', data=top_negative, palette='coolwarm', ax=axes[1])
axes[1].set_title('Top Negative Employees')
axes[1].set_xlabel('Employee')
axes[1].set_ylabel('Count')

plt.tight_layout()
plt.savefig(f'{output_dir}/employee_rankings.png')
plt.show()

# 5. Flight Risk Identification Visualization
flight_risk_employees = data[data['rolling_negative_count'] >= 4]['from'].ur
flight_risk_counts = pd.Series(flight_risk_employees).value_counts()

plt.figure(figsize=(10, 6))
sns.barplot(x=flight_risk_counts.index, y=flight_risk_counts.values, palette
plt.title('Flight Risk Employees')
plt.xlabel('Employee')
plt.ylabel('Count of Flight Risk Messages')
plt.xticks(rotation=90)
plt.savefig(f'{output_dir}/flight_risk_identification.png')
plt.show()

# 6. Model Performance Visualization (Actual vs Predicted Sentiment Scores)
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red'
```

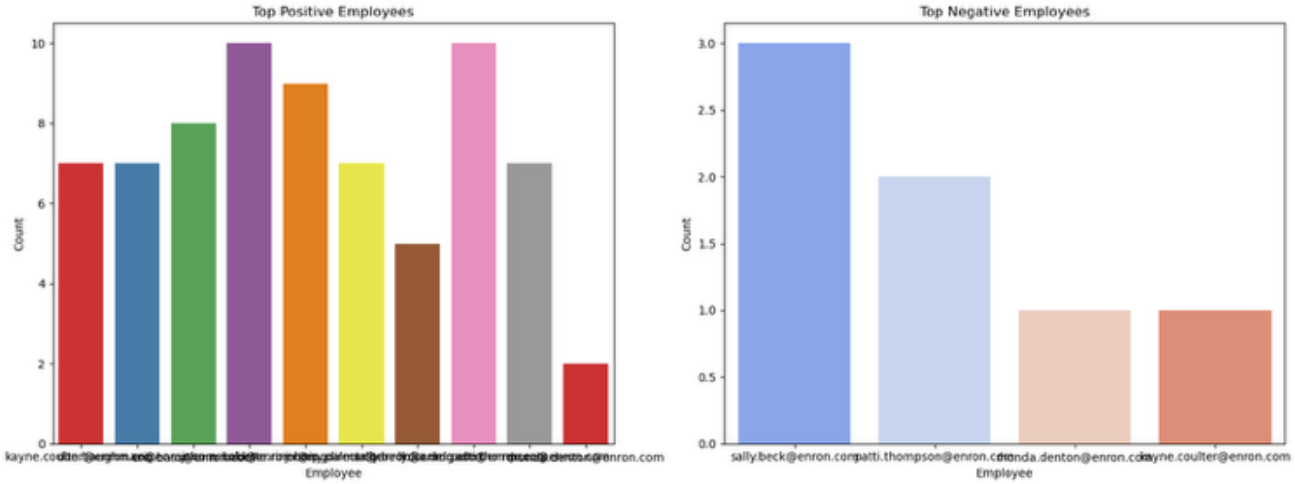
```
plt.title('Actual vs Predicted Sentiment Scores')
plt.xlabel('Actual Scores')
plt.ylabel('Predicted Scores')
plt.savefig(f'{output_dir}/model_performance.png')
plt.show()
```

```
/var/folders/ks/jy925sb56tsg1fkcfkb47kkr0000gn/T/ipykernel_81655/2120357634.
py:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.countplot(x='sentiment', data=data, palette='Set2')
```

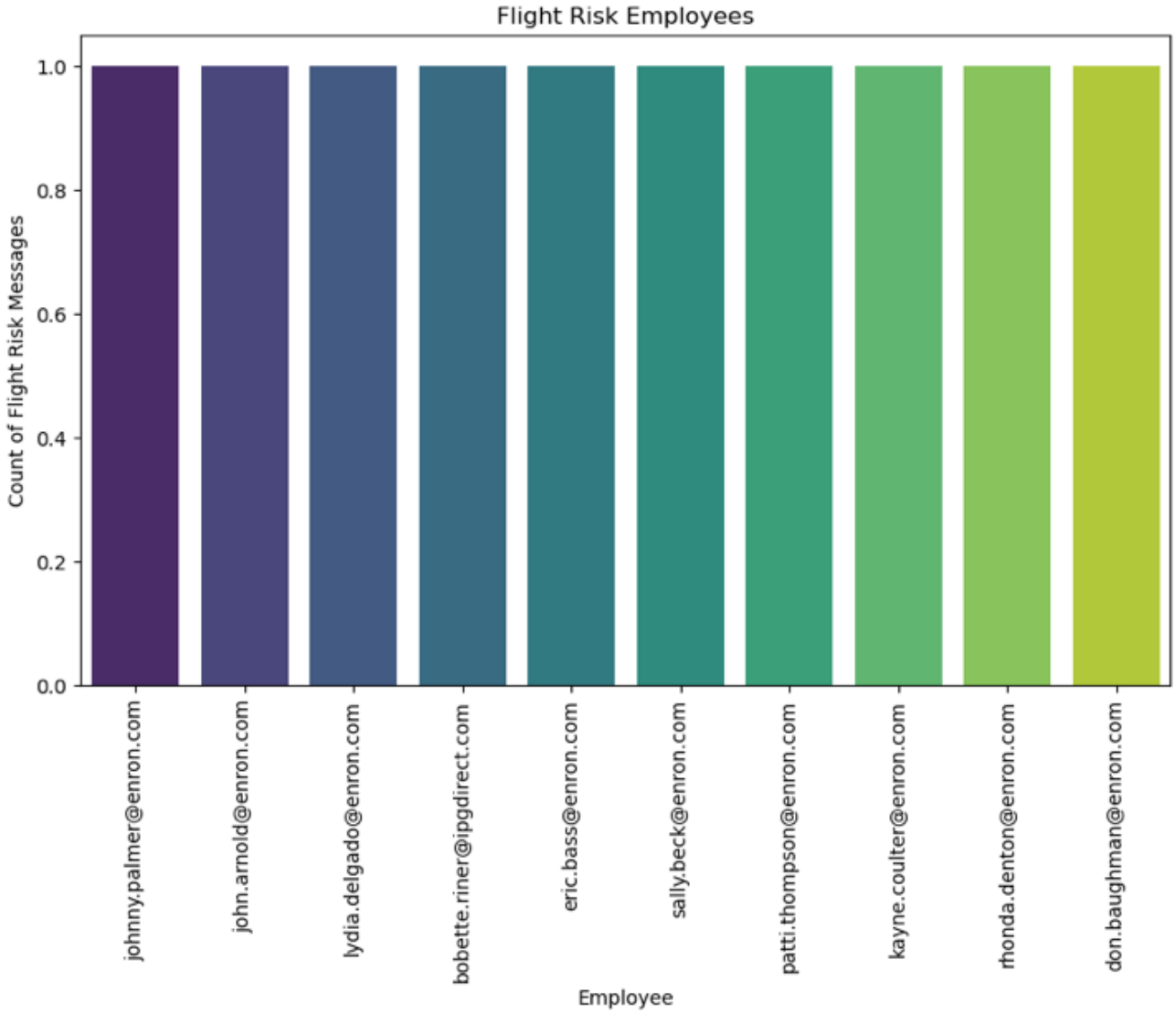


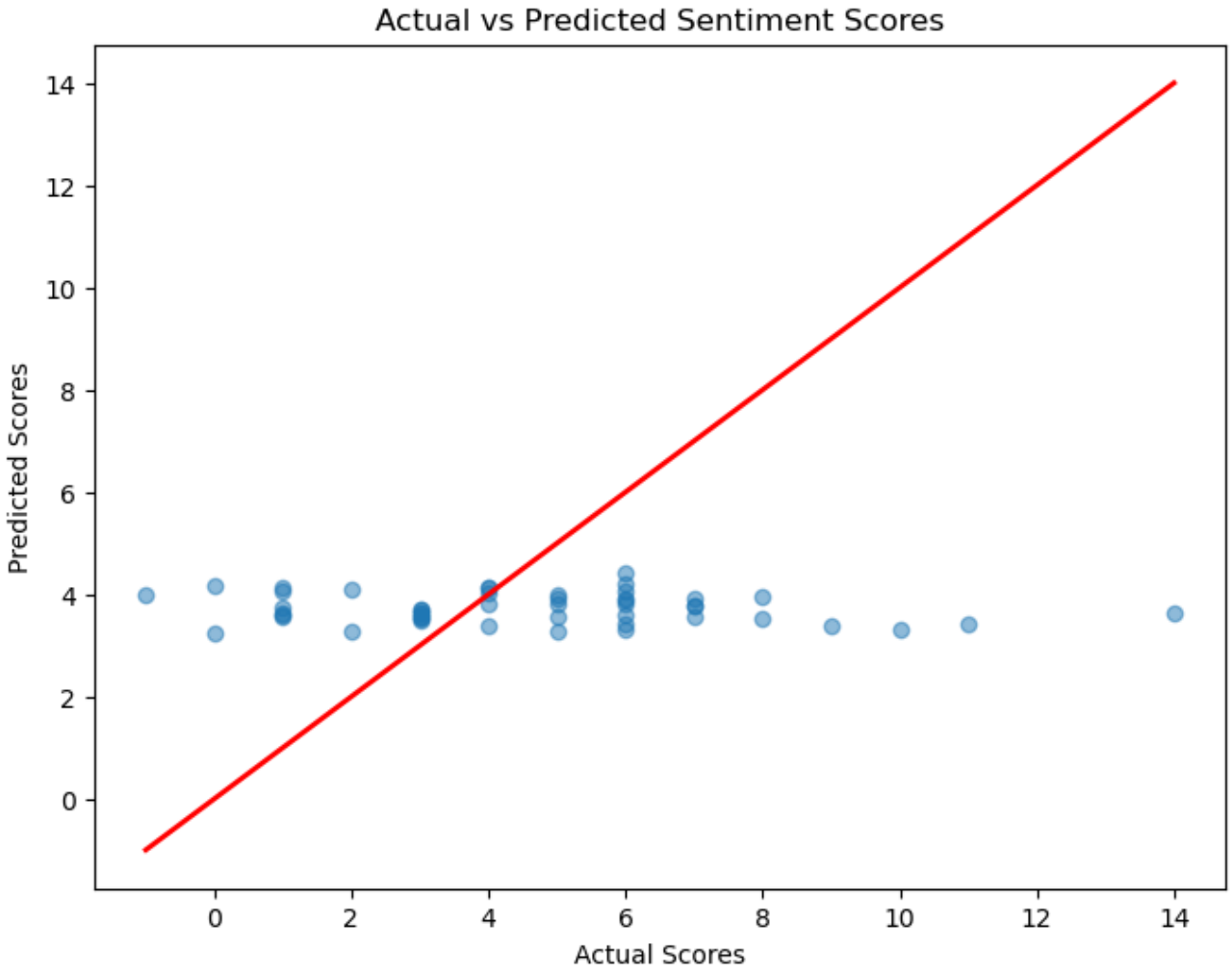



```
/var/folders/ks/jy925sb56tsg1fkcfkb47kkr0000gn/T/ipykernel_81655/2120357634.
py:36: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.countplot(x='from', data=top_positive, palette='Set1', ax=axes[0])
/var/folders/ks/jy925sb56tsg1fkcfkb47kkr0000gn/T/ipykernel_81655/2120357634.
py:42: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.countplot(x='from', data=top_negative, palette='coolwarm', ax=axes[1])
```




```
/var/folders/ks/jy925sb56tsg1fkcfkb47kkr0000gn/T/ipykernel_81655/2120357634.
py:56: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.barplot(x=flight_risk_counts.index, y=flight_risk_counts.values, palet
te='viridis')
```


Flight Risk Employees
Count of Flight Risk Messages
1.0
0.8
0.6
0.4
0.2
0.0
johnny.palmer@enron.com
john.arnold@enron.com
lydia.delgado@enron.com
bobette.riner@ipgdirect.com
eric.bass@enron.com
sally.beck@enron.com
patti.thompson@enron.com
kayne.coulter@enron.com
rhonda.denton@enron.com
don.baughman@enron.com
Employee


Actual vs Predicted Sentiment Scores
Predicted Scores
Actual Scores
0
2
4
6
8
10
12
14

In [ ]: