In [80]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

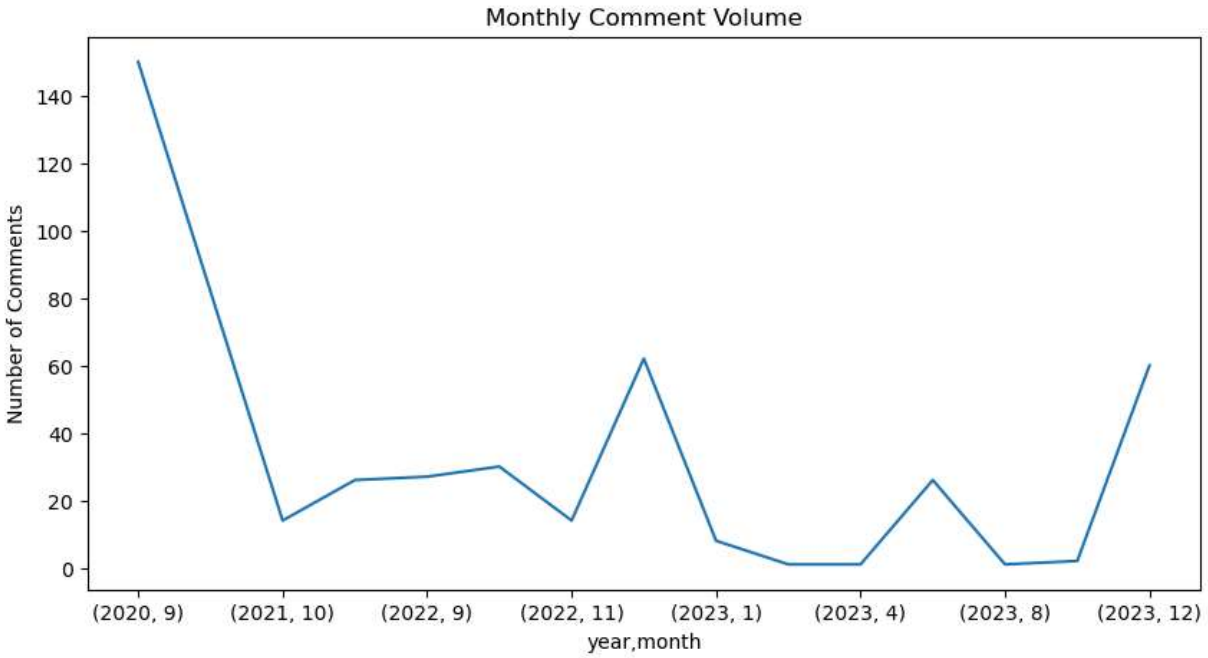
In [57]:
```
# Load the dataset
data = pd.read_csv("climate_nasa.csv")

# Preview first few rows
print(data.head())
print(data.columns)
```

```
                       date  likesCount  \
0  2022-09-07T17:12:32.000Z           2
1  2022-09-08T14:51:13.000Z           0
2  2022-09-07T17:19:41.000Z           1
3  2022-09-08T00:51:30.000Z           4
4  2022-09-07T19:06:20.000Z          16

                                        profileName  commentsCount  \
0  4dca617d86b3fdce80ba7e81fb16e048c9cd9798cdfd6d...            NaN
1  518ab97f2d115ba5b6f03b2fba2ef2b120540c9681288b...            NaN
2  d82e8e24eb633fd625b0aef9b3cb625cfb044ceb8483e1...            3.0
3  37a509fa0b5177a2233c7e2d0e2b2d6916695fa9fba3f2...            NaN
4  e54fbbd42a729af9d04d9a5cc1f9bbfe8081a31c219ecb...           26.0

                                                text
0  Neat comparison I have not heard it before.\n ...
1  An excellent way to visualise the invisible! T...
2  Does the CO2/ghg in the troposphere affect the...
3  excellent post! I defo feel the difference - o...
4  Yes, and carbon dioxide does not harm the Eart...
Index(['date', 'likesCount', 'profileName', 'commentsCount', 'text'], dtype='objec
t')
```

In [58]:
```
# Check for missing values
print(data.isnull().sum())

# Drop rows with missing text comments
data = data.dropna(subset=['text'])

# Convert 'Date' to datetime format
data['Date'] = pd.to_datetime(data['date'])
```

```
date              0
likesCount        0
profileName       0
commentsCount   278
text             18
dtype: int64
```

In [59]:
```
#Visualize likes and comment trends overtime

data['year'] = data['Date'].dt.year
data['month'] = data['Date'].dt.month

# Comments over time
monthly_counts = data.groupby(['year', 'month']).size()

monthly_counts.plot(kind='line', figsize=(10, 5), title='Monthly Comment Volume')
plt.ylabel('Number of Comments')
plt.show()
```

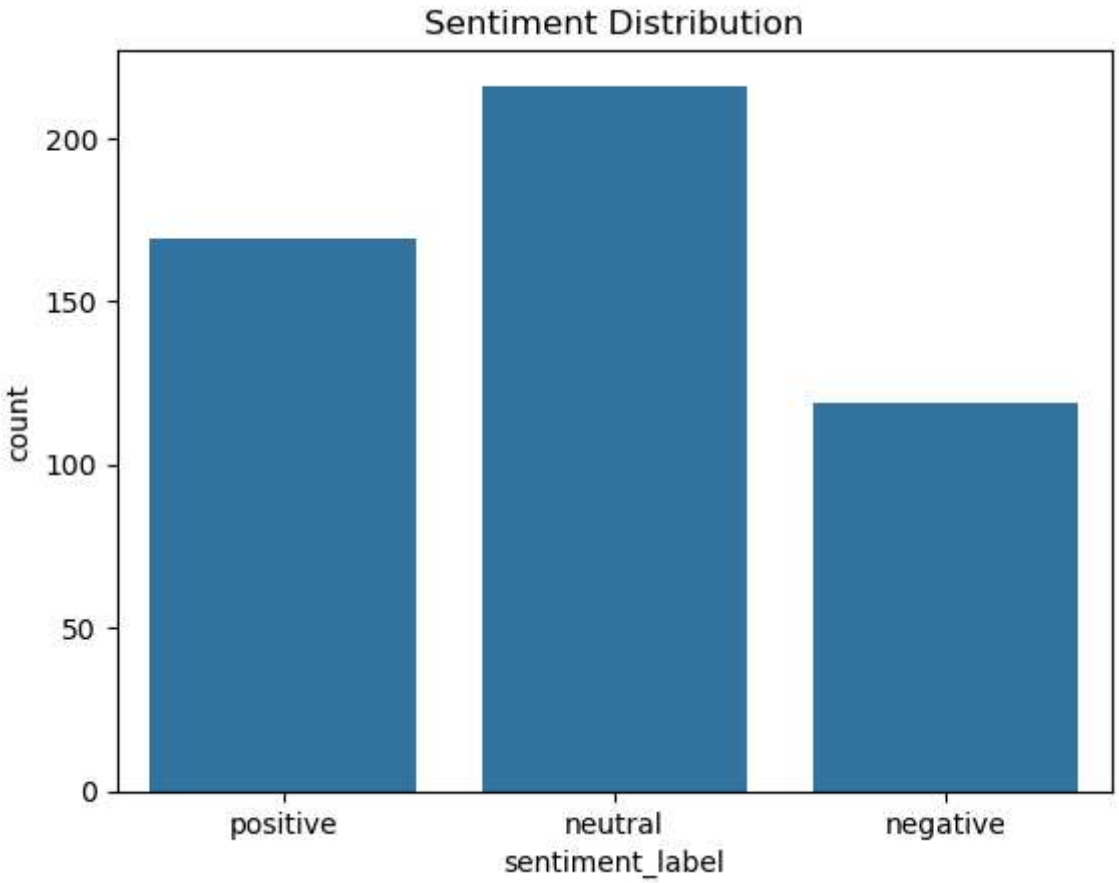


In [60]:
```
#Top liked comments

top_likes = data.sort_values('likesCount', ascending=False).head(5)
print(top_likes[['text', 'likesCount']])
```

```
                                                  text  likesCount
433  I can only hope to someday attain the level of...         126
397  You are being gas-lighted. Have any of you hea...         120
126  I love watch NASA dunk on people who think the...         110
427  Too bad we don’t see Oct 1821 or 1721, just ke...          62
450  To those who cite a very cold day as evidence ...          55
```

In [61]:
```
# Add sentiment scores
data['sentiment'] = data['text'].apply(lambda x: sia.polarity_scores(x)['compound']

# Categorize sentiment
data['sentiment_label'] = data['sentiment'].apply(lambda x: 'positive' if x > 0.2 e
```

In [62]:
```
#Visualize sentiment distribution

sns.countplot(x='sentiment_label', data=data)
plt.title('Sentiment Distribution')
plt.show()
```



In [63]:
```
#Feature extraction

vectorizer = TfidfVectorizer(max_features=500)
X = vectorizer.fit_transform(data['text'])

# Use LikesCount as target
y = data['likesCount']
```

In [64]:
```
#Model training

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

model = RandomForestRegressor()
model.fit(X_train, y_train)
```

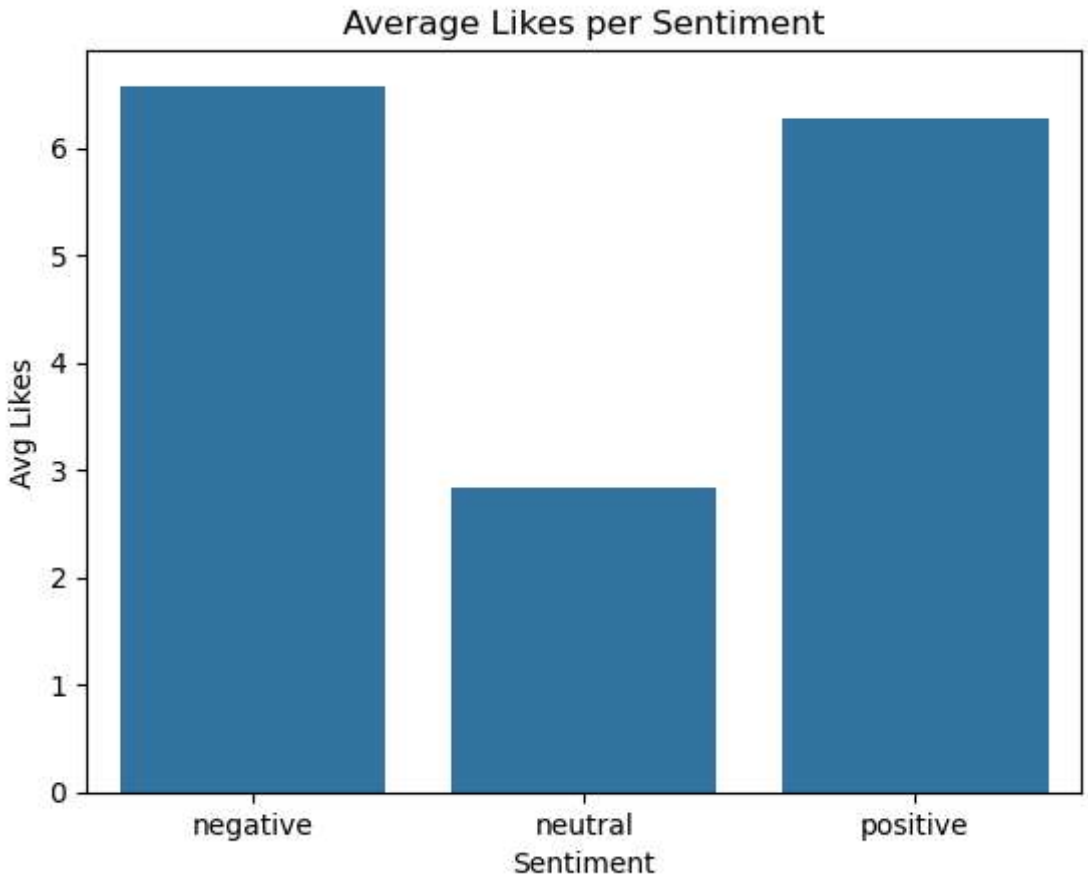
```
y_pred = model.predict(X_test)
print("MSE:", mean_squared_error(y_test, y_pred))
```

```
MSE: 327.0826241651351
```

In [66]:
```
#Sentiment grouping

# Comment length
data['comment_length'] = data['text'].apply(lambda x: len(x.split()))
```

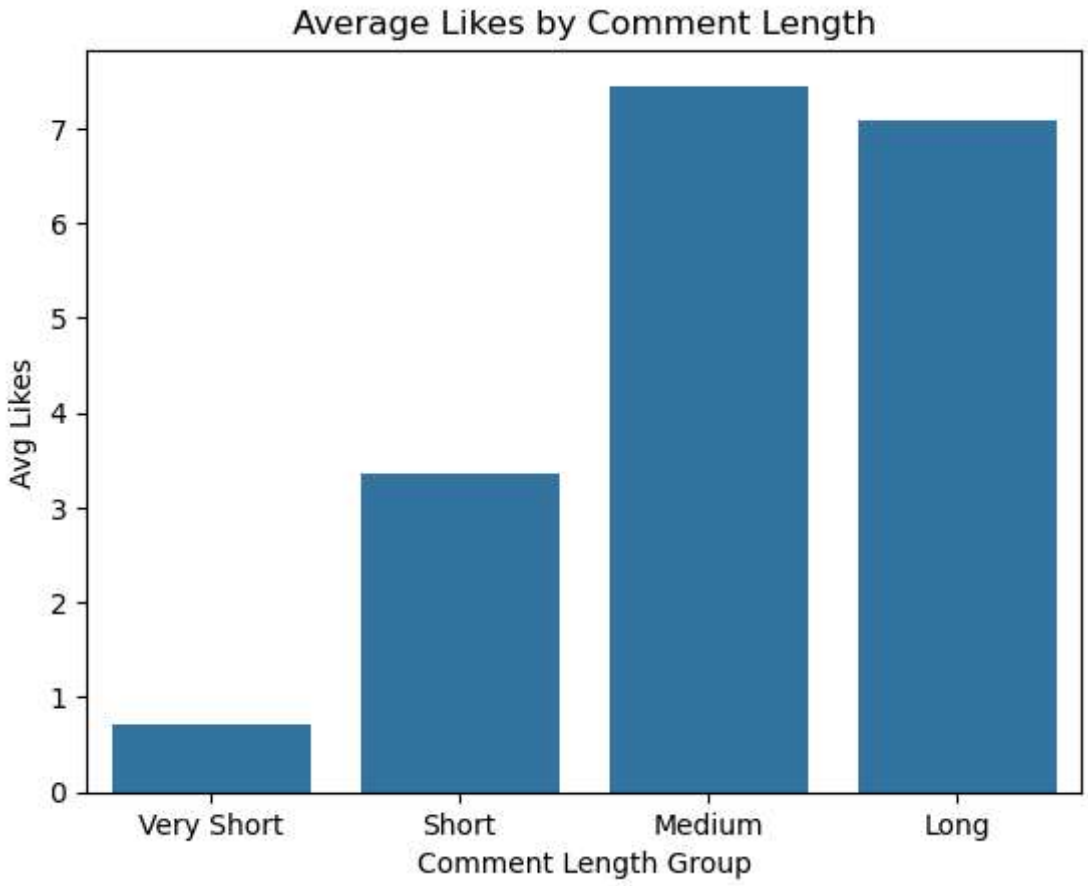
In [ ]:
```
#Sentiment vs Likes

avg_likes_sentiment = data.groupby('sentiment_label')['likesCount'].mean()

print("Average Likes by Sentiment:")
print(avg_likes_sentiment)

# Plot
sns.barplot(x=avg_likes_sentiment.index, y=avg_likes_sentiment.values)
plt.title("Average Likes per Sentiment")
plt.ylabel("Avg Likes")
plt.xlabel("Sentiment")
plt.show()
```

```
Average Likes by Sentiment:
sentiment_label
negative    6.579832
neutral     2.833333
positive    6.266272
Name: likesCount, dtype: float64
```



In [ ]:
```
#Comment Length vs Likes

# Categorize by comment length
data['length_group'] = pd.cut(data['comment_length'],
                              bins=[0, 5, 15, 30, 1000],
                              labels=['Very Short', 'Short', 'Medium', 'Long'])

avg_likes_length = data.groupby('length_group')['likesCount'].mean()

print("Average Likes by Comment Length:")
print(avg_likes_length)

# Plot
sns.barplot(x=avg_likes_length.index, y=avg_likes_length.values)
plt.title("Average Likes by Comment Length")
plt.ylabel("Avg Likes")
plt.xlabel("Comment Length Group")
plt.show()
```

```
Average Likes by Comment Length:
length_group
Very Short    0.710000
Short         3.357664
Medium        7.446154
Long          7.074074
Name: likesCount, dtype: float64
```



In [82]:
```
# Convert PeriodIndex to ordinal for regression
X = np.arange(len(sentiment_proportions)).reshape(-1, 1)
future_X = np.array([[len(sentiment_proportions)]])

predictions = {}
for sentiment in sentiment_proportions.columns:
    y = sentiment_proportions[sentiment].values
    model = LinearRegression()
    model.fit(X, y)
    pred = model.predict(future_X)[0]
    predictions[sentiment] = round(pred, 3)

# Show projection
print("Projected Sentiment Distribution for Next Month:")
for k, v in predictions.items():
    print(f"{k.capitalize()}: {max(0, min(v, 1)):.2f}")
```

```
Projected Sentiment Distribution for Next Month:
Negative: 0.04
Neutral: 0.62
Positive: 0.33
```

In [83]:
```
# Prepare monthly sentiment data
monthly_sentiment = data.copy()
monthly_sentiment['year_month'] = monthly_sentiment['Date'].dt.to_period('M')
# Drop rows where conversion failed
monthly_sentiment = monthly_sentiment.dropna(subset=['date'])
sentiment_counts = monthly_sentiment.groupby(['year_month', 'sentiment_label']).siz
```

```
# Normalize to proportions
sentiment_proportions = sentiment_counts.div(sentiment_counts.sum(axis=1), axis=0)

# Plot trend
sentiment_proportions.plot(figsize=(12,6), title="Sentiment Trend Over Time")
plt.ylabel("Proportion")
plt.show()
```