```
import pandas as pd
df = pd.read_csv('/content/YoutubeCommentsDataSet.csv')
df.head()
```

	Comment	Sentiment
0	lets not forget that apple pay in 2014 require...	neutral
1	here in nz 50 of retailers don't even have con...	negative
2	i will forever acknowledge this channel with t...	positive
3	whenever i go to a place that doesn't take app...	negative
4	apple pay is so convenient secure and easy to ...	positive

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

```
print("Shape of dataset:", df.shape)
print("\nColumns and Data Types:")
print(df.dtypes)
print("\nMissing Values:")
print(df.isnull().sum())
```

```
Shape of dataset: (18408, 2)

Columns and Data Types:
Comment      object
Sentiment    object
dtype: object

Missing Values:
Comment      44
Sentiment     0
dtype: int64
```

```
df = df.dropna(subset=['Comment'])
print("New shape after removing missing comments:", df.shape)
```

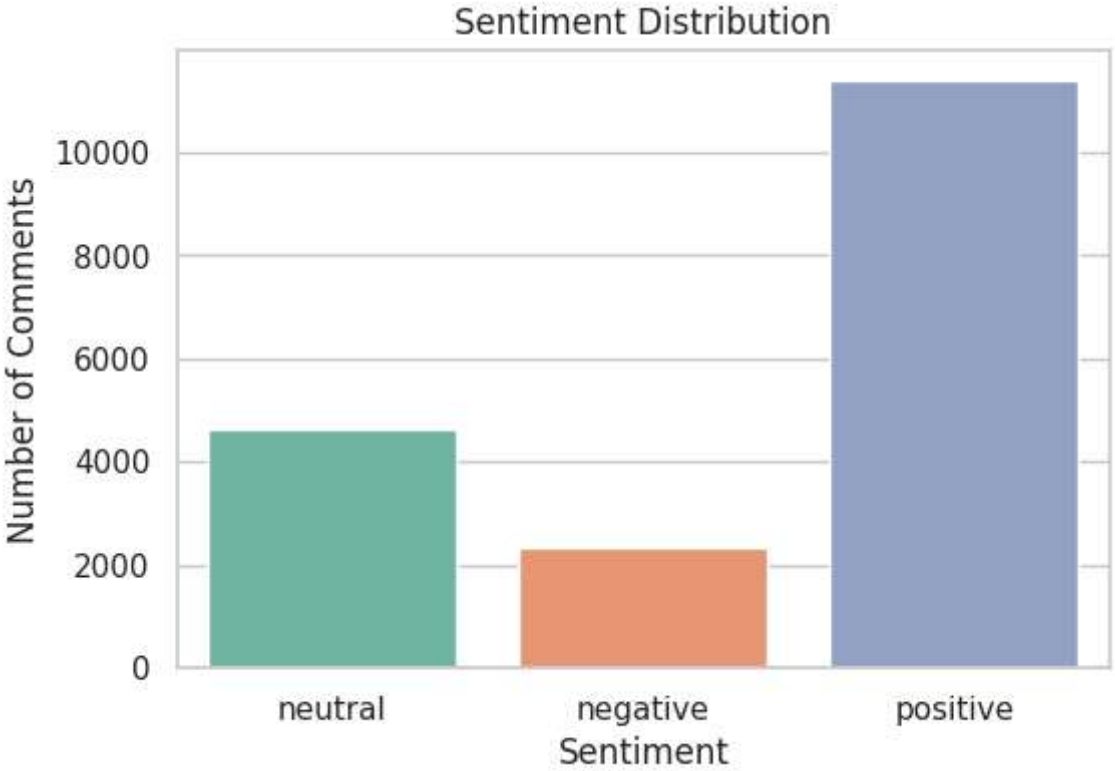
```
New shape after removing missing comments: (18364, 2)
```

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="whitegrid")
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='Sentiment', palette='Set2')
plt.title("Sentiment Distribution")
plt.xlabel("Sentiment")
plt.ylabel("Number of Comments")
plt.show()
```

```
/tmp/ipython-input-4-3820995207.py:5: FutureWarning:

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

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



```
import re
import string
def clean_text(text):
    text = text.lower()                                # lowercase
    text = re.sub(r'http\S+|www\S+|https\S+', '', text) # remove links
    text = re.sub(r'@\w+|\#', '', text)                 # remove mentions and hashtags
    text = re.sub(r'[^\w\s]', '', text)                 # remove punctuation
    text = re.sub(r'\d+', '', text)                     # remove numbers
    text = text.strip()                                 # remove leading/trailing spaces
    return text
df['Cleaned_Comment'] = df['Comment'].apply(clean_text)

df[['Comment', 'Cleaned_Comment']].head()
```

	Comment	Cleaned_Comment
0	lets not forget that apple pay in 2014 require...	lets not forget that apple pay in required a ...
1	here in nz 50 of retailers don’t even have con...	here in nz of retailers dont even have contac...
2	i will forever acknowledge this channel with t...	i will forever acknowledge this channel with t...
3	whenever i go to a place that doesn’t take app...	whenever i go to a place that doesnt take appl...

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(max_features=5000)  # limit to top 5000 words
X = tfidf.fit_transform(df['Cleaned_Comment']).toarray()
X.shape
```

```
(18364, 5000)
```

```
import numpy as np
from sklearn.preprocessing import LabelEncoder

# Create encoder
le = LabelEncoder()

# Transform Sentiment column into numbers
y = le.fit_transform(df['Sentiment'])

# Check unique values after encoding
np.unique(y, return_counts=True)
```

```
(array([0, 1, 2]), array([ 2337,  4625, 11402]))
```

```
from sklearn.model_selection import train_test_split

# Use X from TF-IDF step (already created)
```

```
# Use y from label encoding step (already created)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("Train size:", len(X_train))
print("Test size:", len(X_test))
```

```
Train size: 14691
Test size: 3673
```

```
from sklearn.model_selection import train_test_split

# Split the TF-IDF numeric data (X) and encoded labels (y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("Train size:", len(X_train))
print("Test size:", len(X_test))
```

```
Train size: 14691
Test size: 3673
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(max_features=5000)
X = tfidf.fit_transform(df['Cleaned_Comment']).toarray()
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matri

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

# Accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy: {:.2f}%".format(accuracy * 100))
```

```
# Classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=le.classes_))

# Confusion Matrix
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Model Accuracy: 75.82%

Classification Report:
              precision    recall  f1-score   support

    negative       0.60      0.32      0.42       441
     neutral       0.63      0.59      0.61       912
    positive       0.81      0.91      0.86      2320

    accuracy                           0.76      3673
   macro avg       0.68      0.61      0.63      3673
weighted avg       0.74      0.76      0.74      3673


Confusion Matrix:
[[ 141  137  163]
 [  58  535  319]
 [  36  175 2109]]
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Create confusion matrix again
cm = confusion_matrix(y_test, y_pred)

# Plot the heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=le.classes_, yticklabels=le.classes_)

plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Sentiment Prediction')
plt.show()
```