```python
import pandas as pd

# Load the uploaded df
df = pd.read_csv('IMDB dataset.csv')

# Display the first few rows and column info
df.head(), df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB
```

```
(                                                review sentiment
 0  One of the other reviewers has mentioned that ...  positive
 1  A wonderful little production. <br /><br />The...  positive
 2  I thought this was a wonderful way to spend ti...  positive
 3  Basically there's a family where a little boy ...  negative
 4  Petter Mattei's "Love in the Time of Money" is...  positive,
 None)
```

```python
import re
def clean_text(text):
    text = text.lower()  # set lower case
    text = re.sub(r"<.*?>", "", text)  # remove HTML
    text = re.sub(r"[^\w\s]", "", text)  # remove punctuation marks
    text = re.sub(r"\d+", "", text)  # remove numbers
    text = text.strip()  # remove extra spaces
    return text

df['review'] = df['review'].apply(clean_text)

df['review'][1]
```

```
'a wonderful little production the filming technique is very
unassuming very oldtimebbc fashion and gives a comforting and
sometimes discomforting sense of realism to the entire piece the
actors are extremely well chosen michael sheen not only has got all
the polari but he has all the voices down pat too you can truly see
the seamless editing guided by the references to williams diary
entries not only is it well worth the watching but it is a terrificly
written and performed piece a masterful production about one of the
great masters of comedy and his life the realism really comes home
with the little things the fantasy of the guard which rather than use
the traditional dream techniques remains solid then disappears it
plays on our knowledge and our senses particularly with the scenes
```

```
concerning orton and halliwell and the sets particularly of their flat
with halliwells murals decorating every surface are terribly well
done'

X = df['review']
y = df['sentiment']

print(f"df length: {len(df)}: \n {df.columns} \n")
print(f"X 'review' length: {len(X)}: \n {X} \n")
print(f"y 'sentiment' length: {len(y)}: \n {y}")

df length: 50000:
 Index(['review', 'sentiment'], dtype='object')

X 'review' length: 50000:
 0        one of the other reviewers has mentioned that ...
1        a wonderful little production the filming tech...
2        i thought this was a wonderful way to spend ti...
3        basically theres a family where a little boy j...
4        petter matteis love in the time of money is a ...
                              ...
49995    i thought this movie did a down right good job...
49996    bad plot bad dialogue bad acting idiotic direc...
49997    i am a catholic taught in parochial elementary...
49998    im going to have to disagree with the previous...
49999    no one expects the star trek movies to be high...
Name: review, Length: 50000, dtype: object

y 'sentiment' length: 50000:
 0        positive
1        positive
2        positive
3        negative
4        positive
          ...
49995    positive
49996    negative
49997    negative
49998    negative
49999    negative
Name: sentiment, Length: 50000, dtype: object

import matplotlib.pyplot as plt

# Count the number of samples in each label
label_counts = df['sentiment'].value_counts()

# Plot a column diagram (bar chart)
label_counts.plot(kind='bar', color='skyblue', edgecolor='black')

plt.title('Number of Samples per Label')
```

```
plt.xlabel('Label')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()

import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Download VADER lexicon if not already present
nltk.download("vader_lexicon")

# Initialize VADER
sid = SentimentIntensityAnalyzer()

# Apply VADER compound score
df['vader_compound'] = df['review'].apply(lambda x:
sid.polarity_scores(x)['compound'])

# Classify compound score
def classify_sentiment(score):
    if score >= 0.05:
        return 'Positive'
    elif score <= -0.05:
        return 'Negative'
    else:
        return 'Neutral'

df['vader_sentiment'] = df['vader_compound'].apply(classify_sentiment)

# Show a sample
df[['review', 'sentiment', 'vader_compound',
'vader_sentiment']].head(10)

import seaborn as sns
import matplotlib.pyplot as plt

sns.histplot(df['vader_compound'], bins=30, kde=True, color='skyblue')
plt.title("Distribution of VADER Sentiment Scores")
plt.xlabel("Compound Score")
plt.ylabel("Review Count")
plt.show()

pd.crosstab(df['sentiment'], df['vader_sentiment'])

import numpy as np

# Simulate a box office success score (0–100) based on sentiment
```

```
df['success_score'] = df['vader_compound'].apply(lambda x: np.clip((x
+ 1) * 50, 0, 100))

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X = df[['vader_compound']]
y = df['success_score']

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

model = LinearRegression()
model.fit(X_train, y_train)

preds = model.predict(X_test)
rmse = mean_squared_error(y_test, preds, squared=False)
print("RMSE:", rmse)
```