```
# Install required packages
!pip install pandas numpy matplotlib seaborn scikit-learn keras tensorflow torch nltk

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV
from keras.models import Sequential
from keras.layers import Dense, LSTM
from keras.callbacks import EarlyStopping
from sklearn.metrics import mean_squared_error
from sklearn.feature_extraction.text import TfidfVectorizer
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import string
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

# Download NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
```

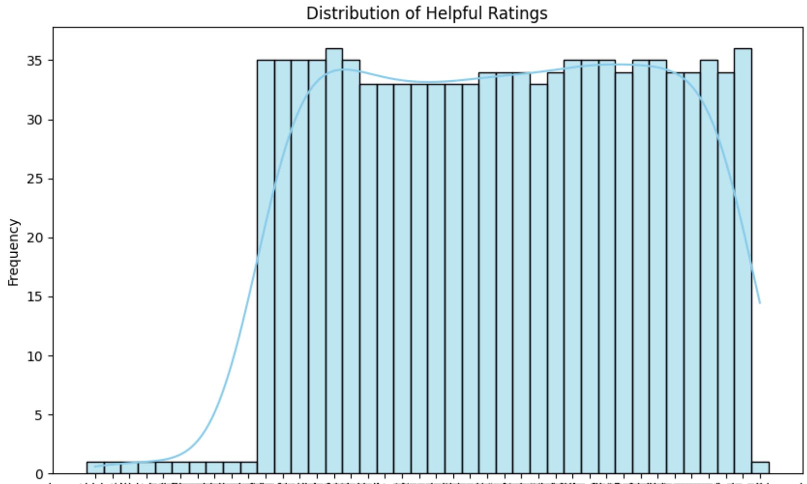
```
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch)
  Downloading nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.1/14.1 MB 67.7 MB/s eta 0:00:00
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch)
  Downloading nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 MB)
```

```
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

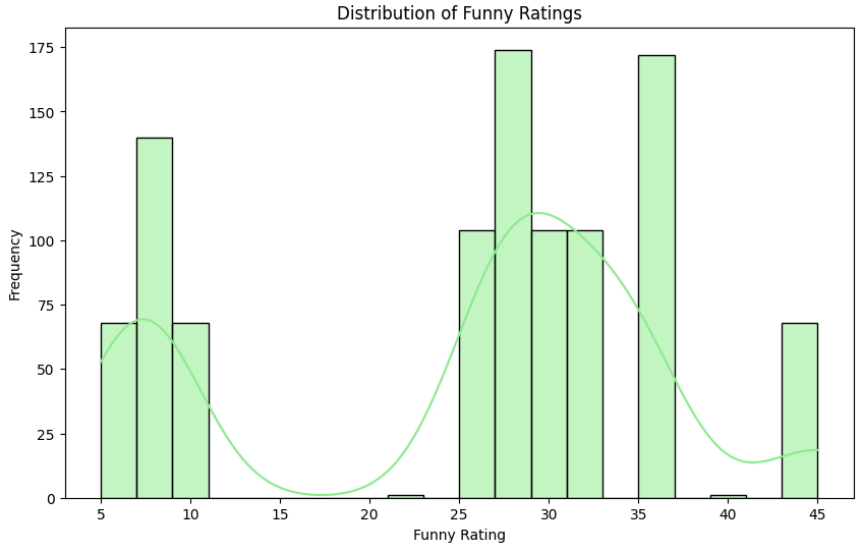
```
# Load the data
data = pd.read_excel('/content/Multilingual PUBG Reviews for Sentiment Analysis.xlsx')

# Preprocess the data
def preprocess_data(data):
    """Preprocess the raw data."""
    try:
        # Convert date columns to datetime
        data['Review_Publication_Date'] = pd.to_datetime(data['Review_Publication_Date'])
        data['Review_Updated_Date'] = pd.to_datetime(data['Review_Updated_Date'])

        # Drop unnecessary columns
        data.drop(['Recommendation_ID', 'Review_Updated_Date'], axis=1, inplace=True)

        # Scale 'Playtime' column
        scaler = MinMaxScaler()
        data['Playtime'] = scaler.fit_transform(data[['Playtime']])

        # Encode categorical variables
        le_recommend_tag = LabelEncoder()
        data['Recommend_Tag'] = le_recommend_tag.fit_transform(data['Recommend_Tag'])
        le_language_tag = LabelEncoder()
        data['Language_Tag'] = le_language_tag.fit_transform(data['Language_Tag'])

        return data
    except Exception as e:
        print(f"Error preprocessing data: {e}")
        return None

data = preprocess_data(data)
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

plt.figure(figsize=(10, 6))
sns.histplot(data['Helpful_Rating'], bins=20, kde=True, color='skyblue')
plt.title('Distribution of Helpful Ratings')
plt.xlabel('Helpful Rating')
plt.ylabel('Frequency')
plt.show()
```

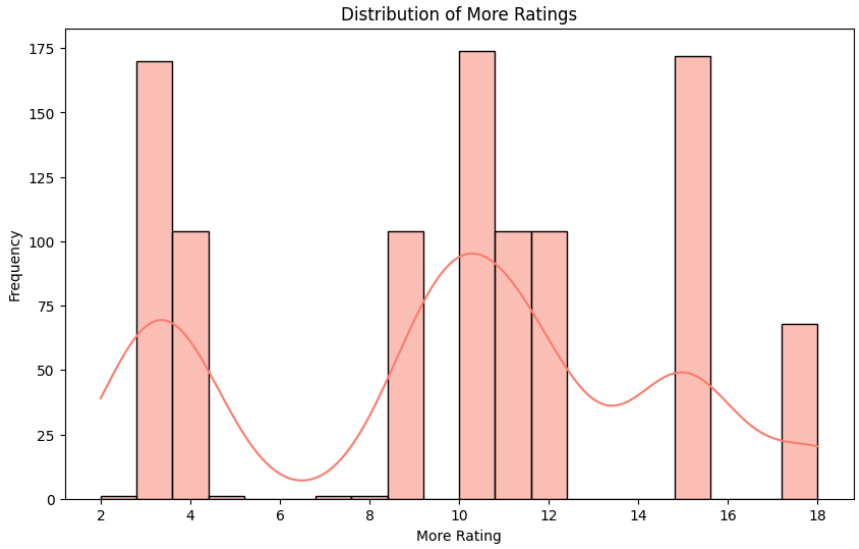
```
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
```



```
plt.figure(figsize=(10, 6))
sns.histplot(data['Funny_Rating'], bins=20, kde=True, color='lightgreen')
plt.title('Distribution of Funny Ratings')
plt.xlabel('Funny Rating')
plt.ylabel('Frequency')
plt.show()
```

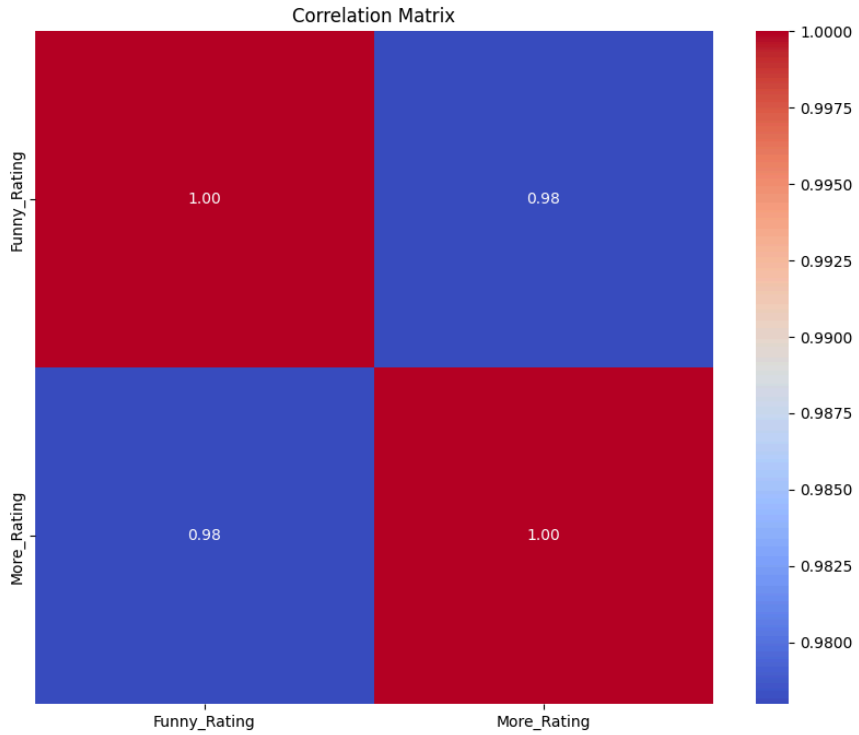



```
plt.figure(figsize=(10, 6))
sns.histplot(data['More_Rating'], bins=20, kde=True, color='salmon')
plt.title('Distribution of More Ratings')
plt.xlabel('More Rating')
plt.ylabel('Frequency')
plt.show()
```

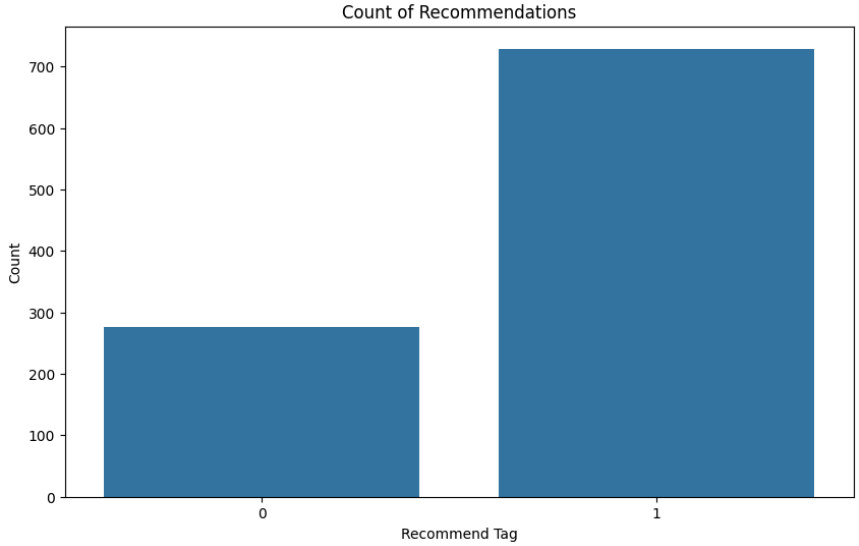



```
plt.figure(figsize=(10, 8))
sns.heatmap(data[['Helpful_Rating', 'Funny_Rating', 'More_Rating']].corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```

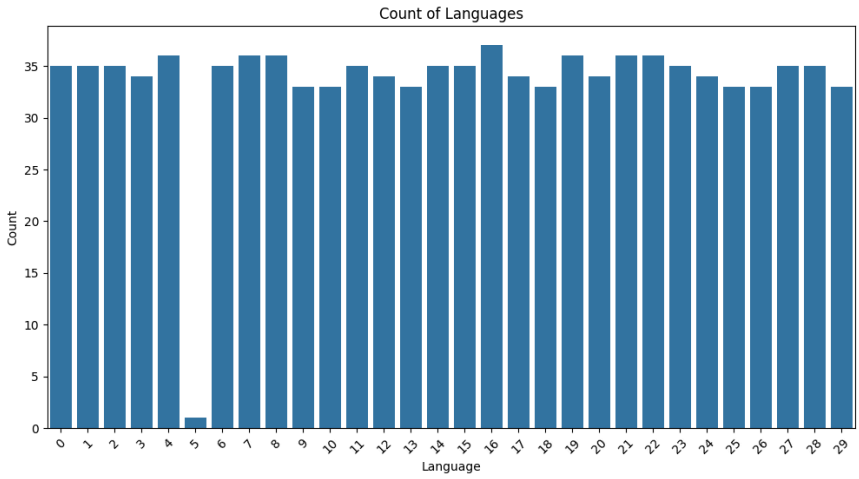
```
<ipython-input-8-8fce9ce07ca8>:2: FutureWarning: The default value of numeric_only i
  sns.heatmap(data[['Helpful_Rating', 'Funny_Rating', 'More_Rating']].corr(), annot=
```



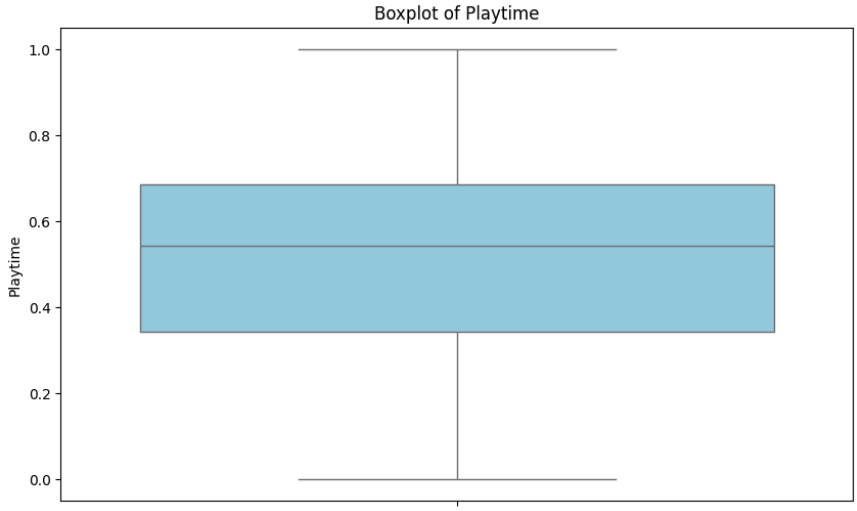
```
plt.figure(figsize=(10, 6))
sns.countplot(data=data, x='Recommend_Tag')
plt.title('Count of Recommendations')
plt.xlabel('Recommend Tag')
plt.ylabel('Count')
plt.show()
```




```
plt.figure(figsize=(12, 6))
sns.countplot(data=data, x='Language_Tag')
plt.title('Count of Languages')
plt.xlabel('Language')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

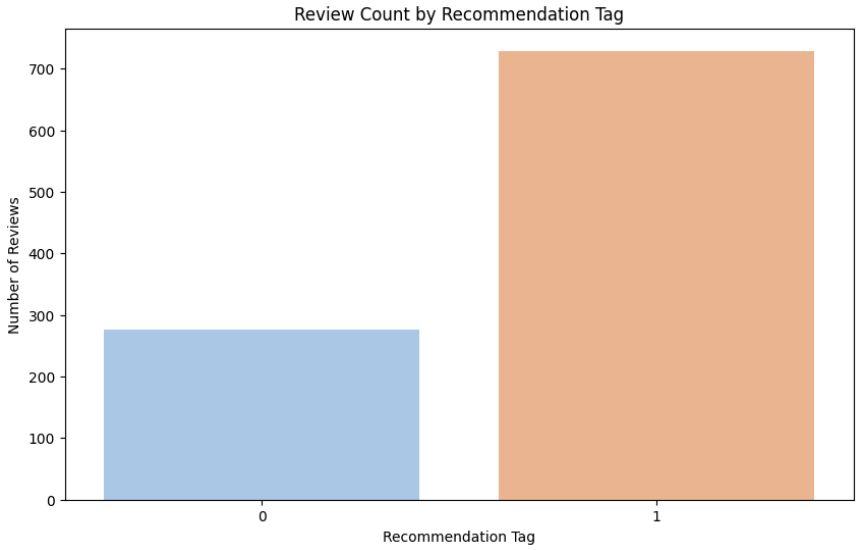


```
plt.figure(figsize=(10, 6))
sns.boxplot(data=data, y='Playtime', color='skyblue')
plt.title('Boxplot of Playtime')
plt.ylabel('Playtime')
plt.show()
```



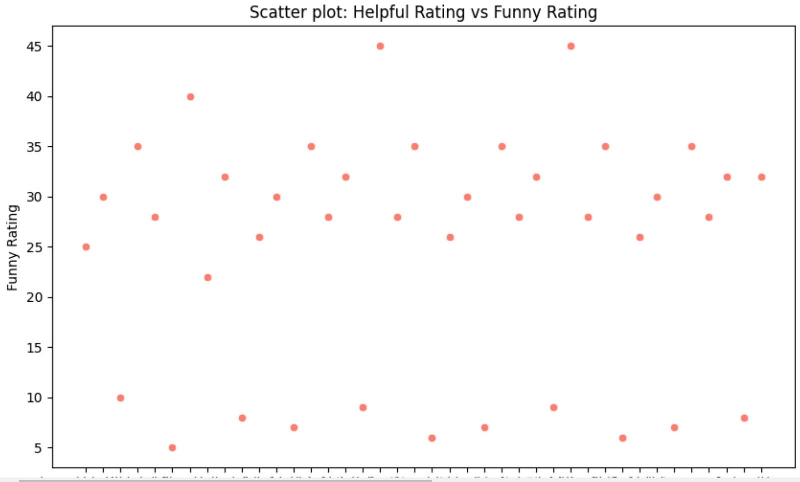

```
plt.figure(figsize=(10, 6))
sns.barplot(data=data, x='Recommend_Tag', y=data.index, estimator=len, palette='pastel')
plt.title('Review Count by Recommendation Tag')
plt.xlabel('Recommendation Tag')
plt.ylabel('Number of Reviews')
plt.show()
```

```
<ipython-input-12-2b467cb54609>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14

  sns.barplot(data=data, x='Recommend_Tag', y=data.index, estimator=len, palette='pas
```

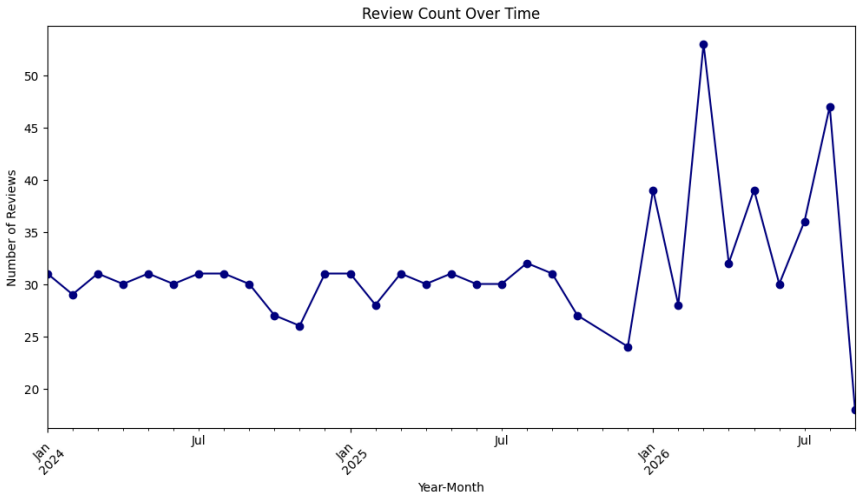


```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='Helpful_Rating', y='Funny_Rating', color='salmon')
plt.title('Scatter plot: Helpful Rating vs Funny Rating')
plt.xlabel('Helpful Rating')
plt.ylabel('Funny Rating')
plt.show()
```

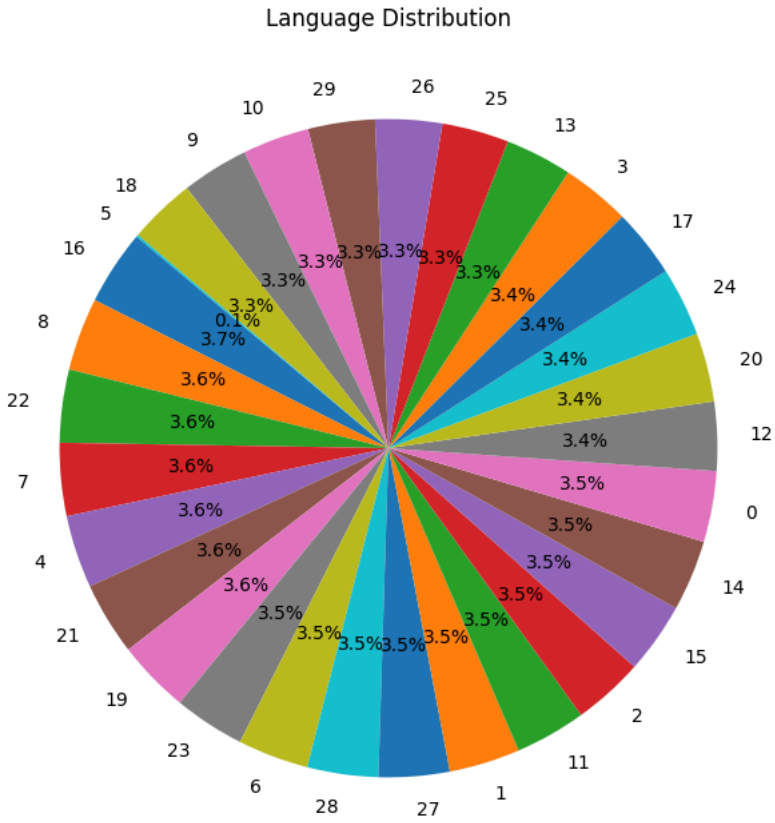
```
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarnir
  fig.canvas.print_figure(bytes_io, **kw)
```



```
plt.figure(figsize=(12, 6))
data['Review_Publication_Date'] = pd.to_datetime(data['Review_Publication_Date'])
data['Year-Month'] = data['Review_Publication_Date'].dt.to_period('M')
review_count_by_month = data.groupby('Year-Month').size()
review_count_by_month.plot(marker='o', color='navy')
plt.title('Review Count Over Time')
plt.xlabel('Year-Month')
plt.ylabel('Number of Reviews')
plt.xticks(rotation=45)
plt.show()
```




```
plt.figure(figsize=(8, 8))
language_counts = data['Language_Tag'].value_counts()
plt.pie(language_counts, labels=language_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Language Distribution')
plt.show()
```

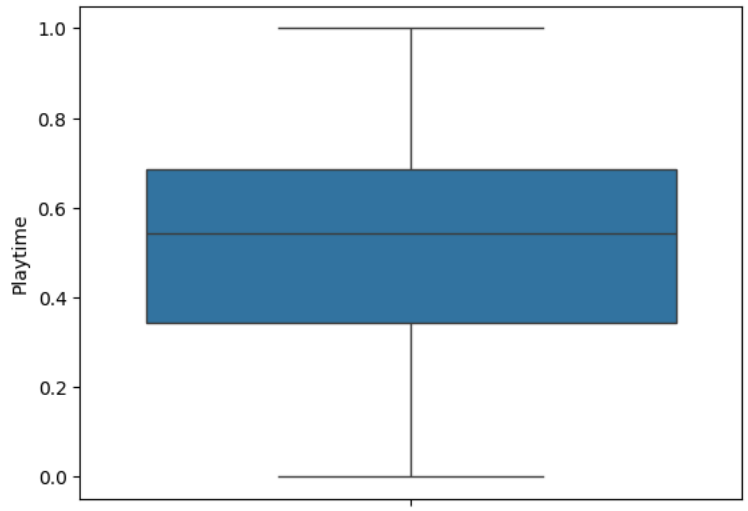



```
# Handle missing values
def handle_missing_values(data):
    """Handle missing values."""
    try:
        print(data.isnull().any())
        print(data.isnull().sum())
        data.dropna(inplace=True)
        print(data.isnull().sum())
        return data
    except Exception as e:
        print(f"Error handling missing values: {e}")
        return None

data = handle_missing_values(data)
```

```
User_ID                  False
Review_Publication_Date  False
Playtime                 False
Recommend_Tag            False
Language_Tag             False
Review_Content           False
Helpful_Rating            True
Funny_Rating              True
More_Rating               True
Unnamed: 11               True
Year-Month               False
dtype: bool
User_ID                  0
Review_Publication_Date  0
Playtime                 0
Recommend_Tag            0
Language_Tag             0
Review_Content           0
Helpful_Rating           1
Funny_Rating             1
More_Rating              1
Unnamed: 11              1
Year-Month               0
dtype: int64
User_ID                  0
Review_Publication_Date  0
Playtime                 0
Recommend_Tag            0
Language_Tag             0
Review_Content           0
Helpful_Rating           0
```

```
Funny_Rating              0
More_Rating               0
Unnamed: 11               0
Year-Month                0
dtype: int64
```

```
# Outlier detection and removal
def remove_outliers(data):
    """Remove outliers using IQR method."""
    try:
        Q1 = data['Playtime'].quantile(0.25)
        Q3 = data['Playtime'].quantile(0.75)
        IQR = Q3 - Q1
        data = data[~((data['Playtime'] < (Q1 - 1.5 * IQR)) | (data['Playtime'] > (Q3 + 1.5 * IQR)))]
        sns.boxplot(data['Playtime'])
        plt.ylabel('Playtime')
        plt.show()
        return data
    except Exception as e:
        print(f"Error removing outliers: {e}")
        return None

data = remove_outliers(data)
```




```
def select_features(data):
    """Select features and target variable."""
    try:
        target = data['Helpful_Rating']
        features = data.drop(['User_ID', 'Review_Publication_Date', 'Playtime','Recommend_Tag','Funny_Rating','More_Rating','Unnamed: 1
        return features, target
    except Exception as e:
        print(f"Error selecting features: {e}")
        return None, None

features, target = select_features(data)
```