Load Data

In [1]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [2]:

```python
# Load the data
data = pd.read_csv('googleplaystore.csv')
```

In [3]:

```python
# Check for null values and drop rows with null values
data.dropna(inplace=True)
```

In [4]:

```python
# Convert Size column to numeric
def convert_size(size):
    if 'M' in size:
        return float(size.replace('M', '')) * 1000
    elif 'k' in size:
        return float(size.replace('k', ''))
    else:
        return None

data['Size'] = data['Size'].apply(lambda x: convert_size(x) if x != 'Varies with device' else Nor
data.dropna(subset=['Size'], inplace=True)
```

In [5]:

```python
# Convert Reviews to numeric
data['Reviews'] = pd.to_numeric(data['Reviews'], errors='coerce')
data.dropna(subset=['Reviews'], inplace=True)
```

In [6]:

```python
# Convert Installs to numeric
data['Installs'] = data['Installs'].str.replace('+', '').str.replace(',', '').astype(int)
```

In [7]:

```python
# Convert Price to numeric
data['Price'] = data['Price'].str.replace('$', '').astype(float)
```

In [8]:

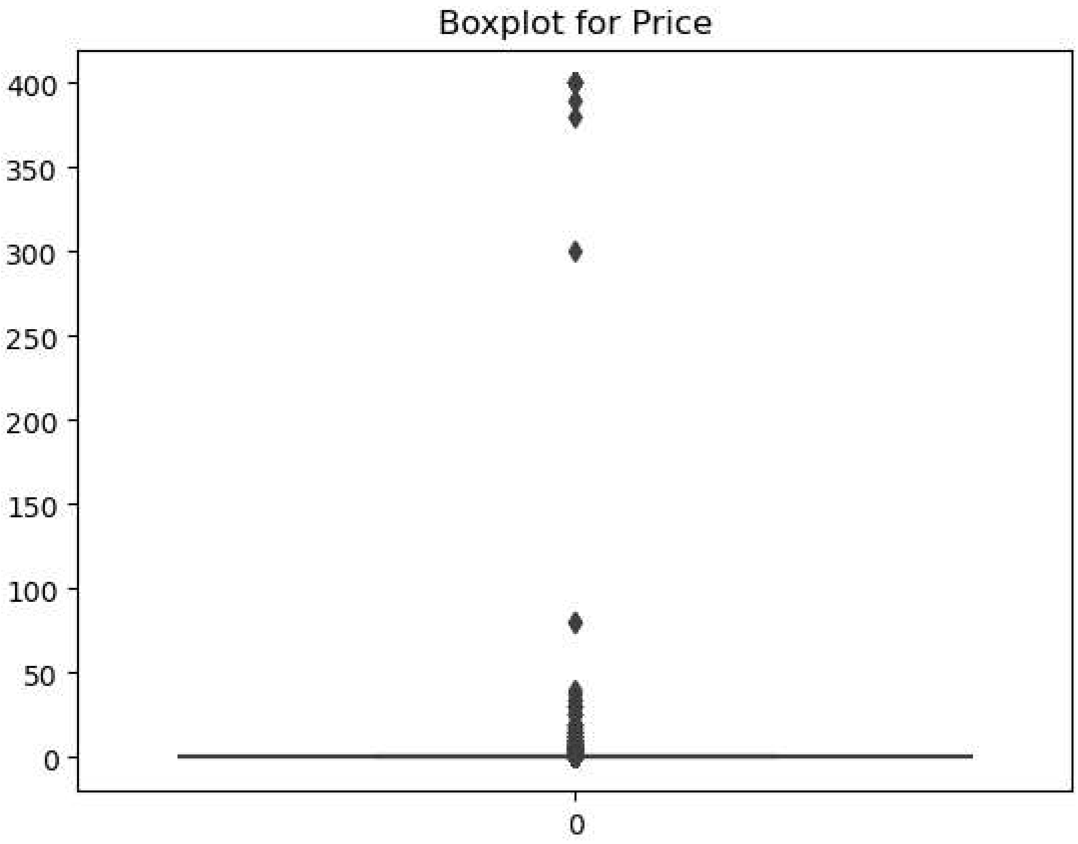
```python
# Convert Type to numeric
data['Type'] = data['Type'].apply(lambda x: 0 if x == 'Free' else 1)
```

In [9]:

```python
# Sanity checks
data = data[(data['Rating'] >= 1) & (data['Rating'] <= 5)]
data = data[data['Reviews'] <= data['Installs']]
data = data[~((data['Type'] == 0) & (data['Price'] > 0))]
```

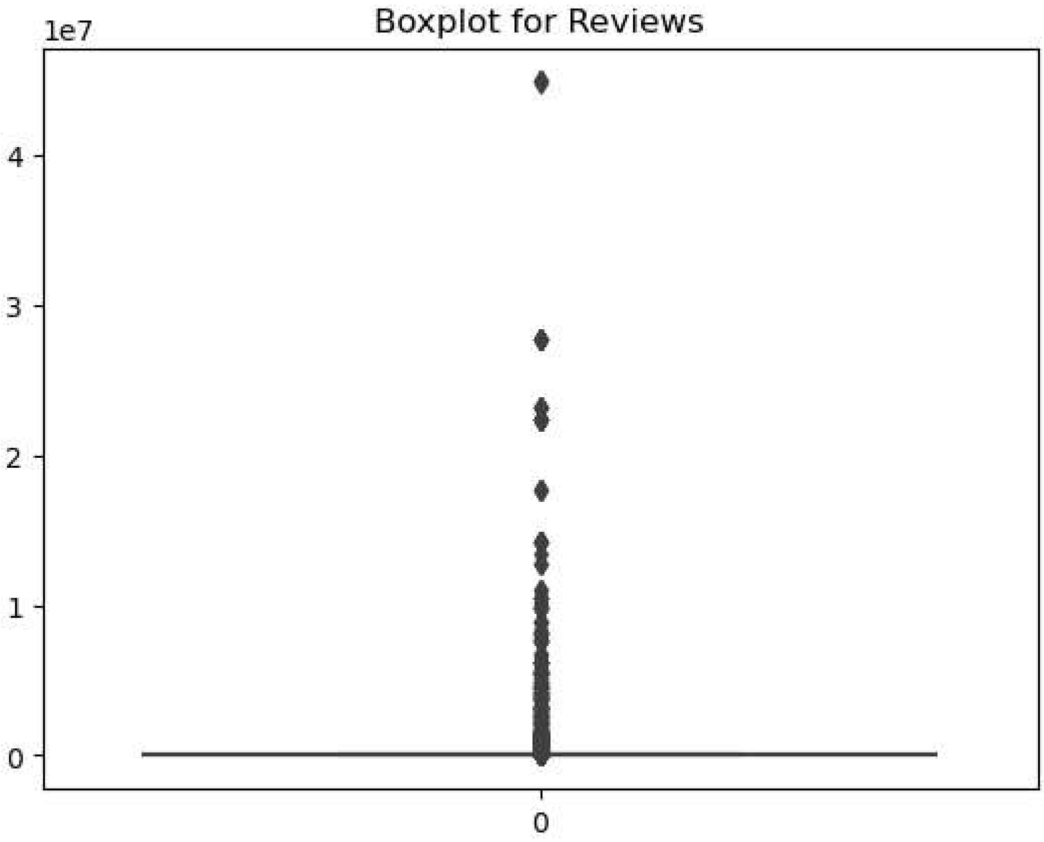
In [10]:

```python
# Univariate Analysis
sns.boxplot(data['Price'])
plt.title('Boxplot for Price')
plt.show()
```



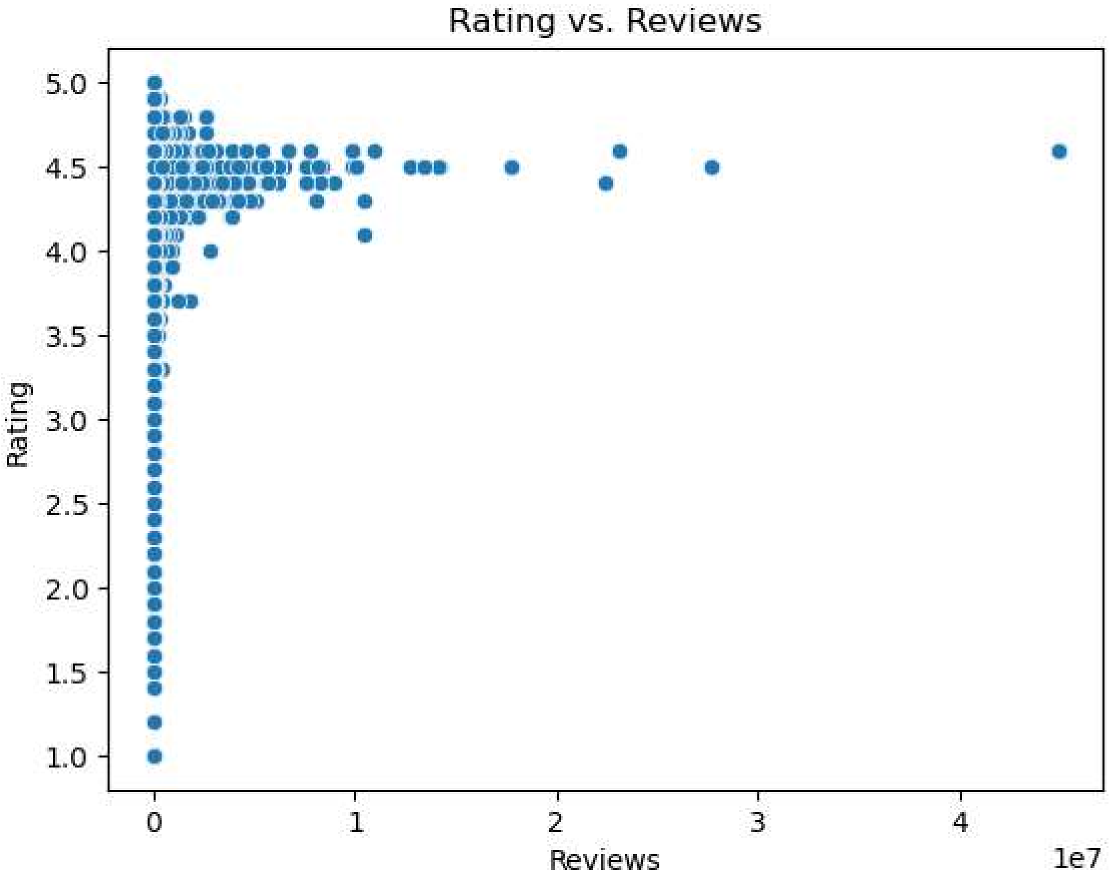
In [11]:

```
sns.boxplot(data['Reviews'])
plt.title('Boxplot for Reviews')
plt.show()
```



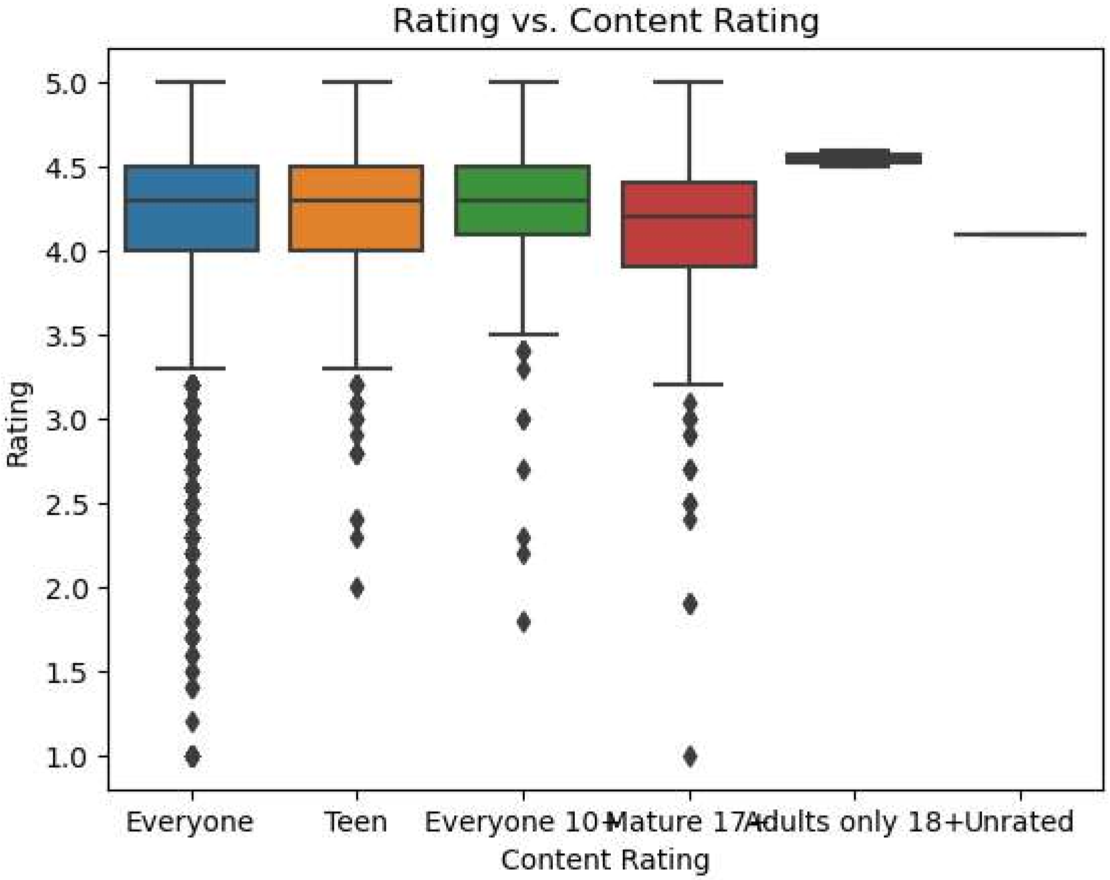
In [12]:

```
sns.scatterplot(x='Reviews', y='Rating', data=data)
plt.title('Rating vs. Reviews')
plt.show()
```



In [13]:

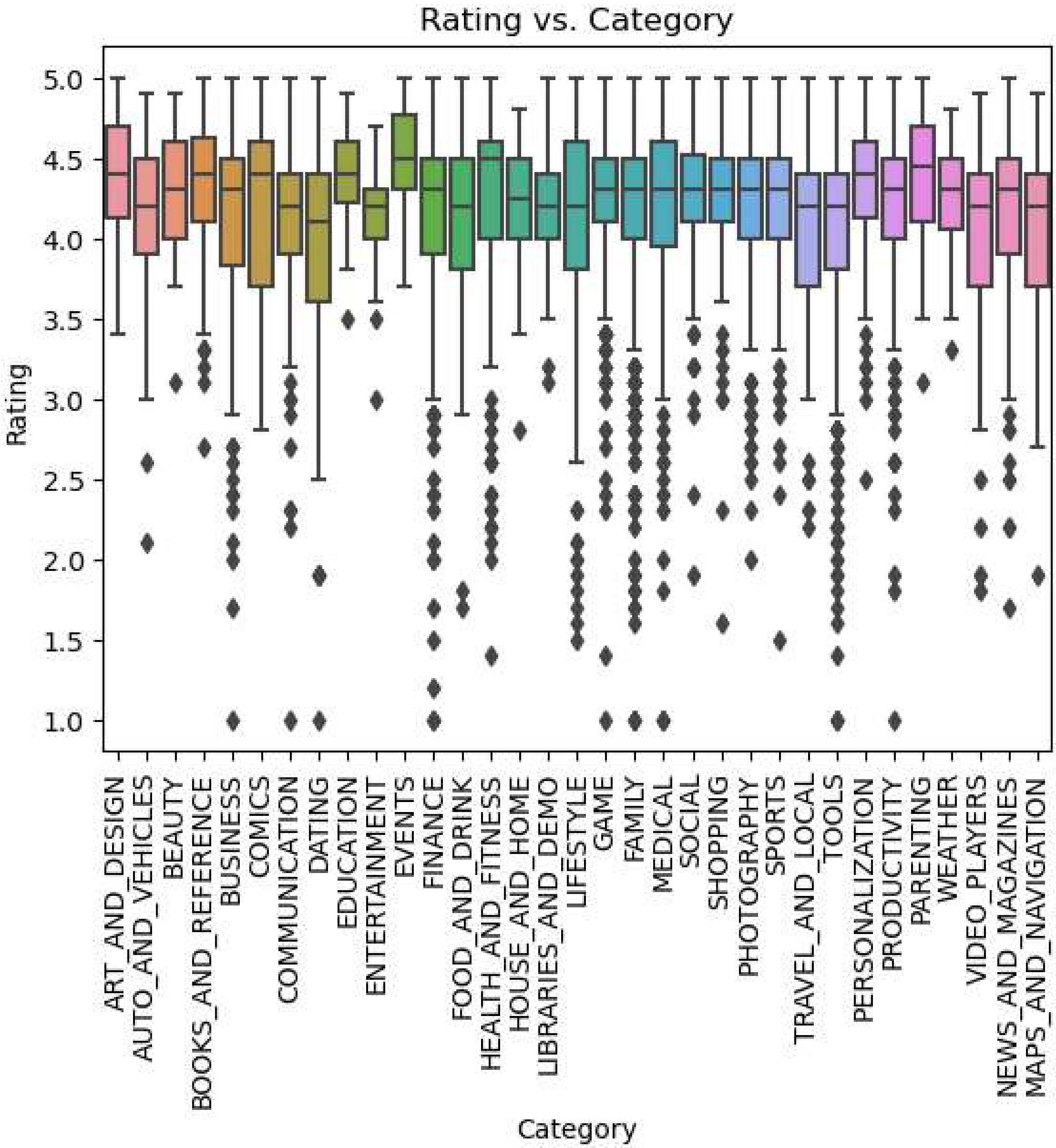
```
sns.boxplot(x='Content Rating', y='Rating', data=data)
plt.title('Rating vs. Content Rating')
plt.show()
```



In [14]:

```
sns.boxplot(x='Category', y='Rating', data=data)
plt.xticks(rotation=90)
```

```python
plt.title('Rating vs. Category')
plt.show()
```



In [15]:

```python
inp1 = data.copy()
inp1['Reviews'] = np.log1p(inp1['Reviews'])
inp1['Installs'] = np.log1p(inp1['Installs'])
inp1.drop(columns=['App', 'Last Updated', 'Current Ver', 'Android Ver'], inplace=True)
inp2 = pd.get_dummies(inp1, columns=['Category', 'Genres', 'Content Rating'], drop_first=True)
```

In [16]:

```python
# Train-Test Split
df_train, df_test = train_test_split(inp2, test_size=0.3, random_state=52)
```

In [17]:

```python
# Separate Features and Target
X_train = df_train.drop(columns=['Rating'])
y_train = df_train['Rating']
X_test = df_test.drop(columns=['Rating'])
y_test = df_test['Rating']
```

In [18]:

```python
# Model Building
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[18]:

```
▾ LinearRegression

LinearRegression()
```

In [19]:

```python
# R2 on Train Set
train_predictions = model.predict(X_train)
train_r2 = r2_score(y_train, train_predictions)
print(f'R2 on train set: {train_r2}')
```

```
R2 on train set: 0.15657567007505557
```

In [20]:

```python
# Predictions on Test Set
test_predictions = model.predict(X_test)
test_r2 = r2_score(y_test, test_predictions)
print(f'R2 on test set: {test_r2}')
```

```
R2 on test set: 0.14293382054219905
```

In [ ]: