FinalDisneyDataset

The unit of observation is a movie review, therefore each row pertains to a movie review. Each subsection below pertains to a variable describing the observations, with the heading being the variable name. The reported numbers in each variable subsection in the format n(m) represent n total observations for the variable and m number of missing values for the variable.

In [ ]:

```
! git clone https://github.com/jillianhaig/Project1_DS4002 # so we can access data loaded from shared github
```

```
Cloning into 'Project1_DS4002'...
remote: Enumerating objects: 476, done.
remote: Counting objects: 100% (87/87), done.
remote: Compressing objects: 100% (75/75), done.
remote: Total 476 (delta 63), reused 12 (delta 12), pack-reused 389 (from 1)
Receiving objects: 100% (476/476), 13.39 MiB | 9.07 MiB/s, done.
Resolving deltas: 100% (205/205), done.
```

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
disney_df = pd.read_csv("/content/Project1_DS4002/Data/Final Datasets/FinalDisneyDataset.csv")
```

rating

This variable takes on values 1-10 and represents the rating the movie reviewer gave the particular movie indicated by the UniqueID variable.

In [ ]:

```
print(disney_df["rating"].count(), "(", sum(disney_df["rating"].str.contains("Null")),")")
```

```
15531 ( 1234 )
```

In [ ]:

```
temp_series = disney_df["rating"][~disney_df["rating"].str.contains("Null")]
temp_series = pd.to_numeric(temp_series)
temp_series.describe()
```

Out[ ]:

	rating
count	14297.000000
mean	7.968525
std	2.331565
min	1.000000
25%	7.000000
50%	9.000000
75%	10.000000
max	10.000000

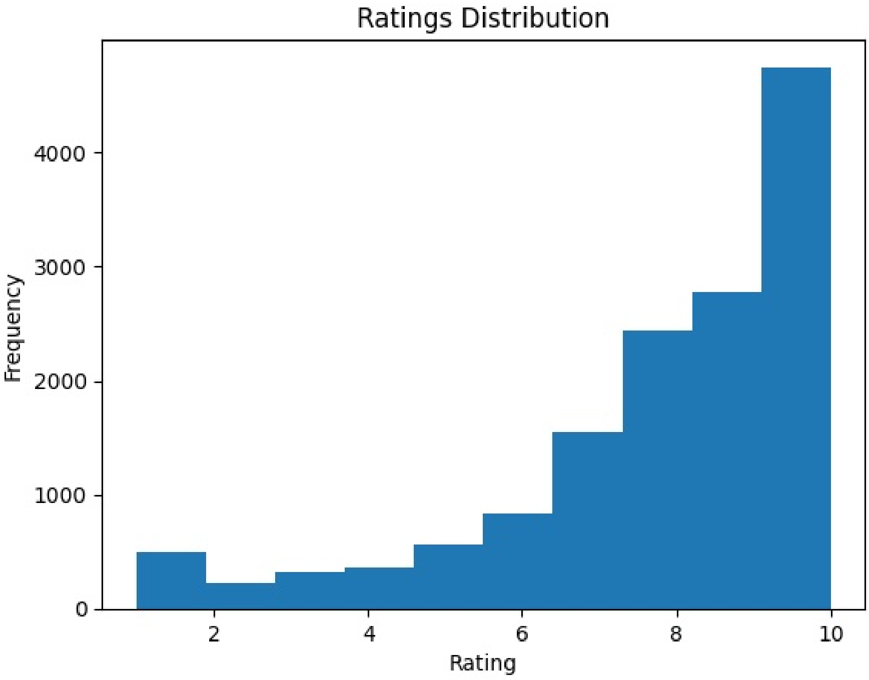
dtype: float64

In [ ]:

```
plt.hist(temp_series, bins=10)
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.title('Ratings Distribution')
```

Out[ ]:

```
Text(0.5, 1.0, 'Ratings Distribution')
```



helpful

This variable takes on values greater than or equal to 0 and represents the number of votes by other users who thought the review was helpful.

In [ ]:
```
print(disney_df["helpful"].count(), "(", sum(disney_df["helpful"].isnull()),")")
```

```
15531 ( 0 )
```

In [ ]:
```
disney_df["helpful"].describe()
```

Out[ ]:

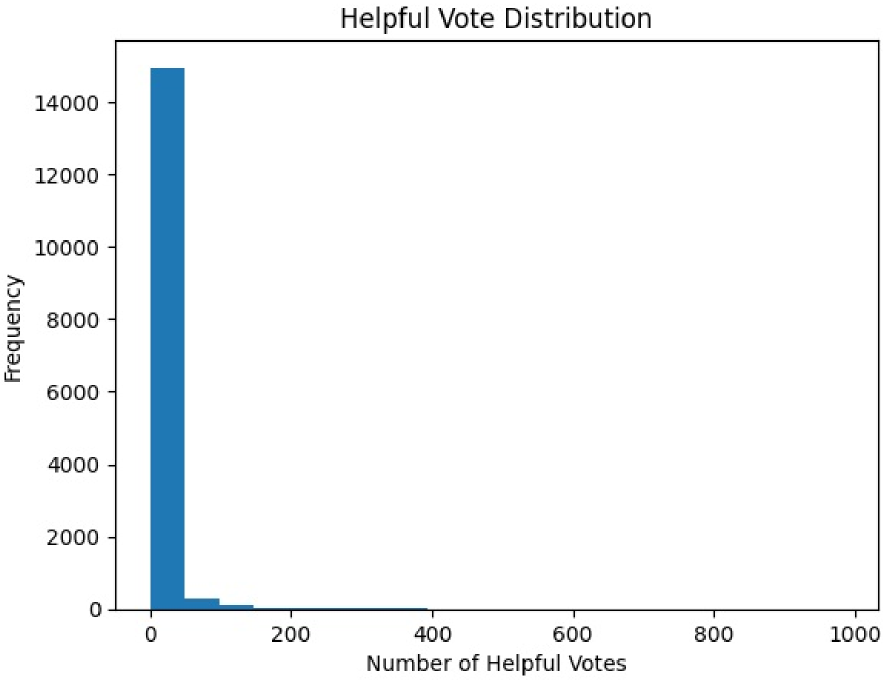
	helpful
count	15531.000000
mean	8.871097
std	37.212960
min	0.000000
25%	0.000000
50%	1.000000
75%	4.000000
max	984.000000

dtype: float64

In [ ]:
```
plt.hist(disney_df["helpful"], bins=20)
plt.xlabel('Number of Helpful Votes')
plt.ylabel('Review Frequency')
plt.title('Helpful Vote Distribution')
```

Out[ ]:

```
Text(0.5, 1.0, 'Helpful Vote Distribution')
```



total

This variable takes on values greater than or equal to 0 and represents the total number of votes on the review by other users, a sum of both the helpful and unhelpful votes.

In [ ]:

```
print(disney_df["total"].count(), "(", sum(disney_df["total"].isnull()),")")
```

```
15531 ( 0 )
```

In [ ]:

```
disney_df["total"].describe()
```

Out[ ]:

	total
count	15531.000000
mean	18.073595
std	62.640130
min	0.000000
25%	1.000000
50%	3.000000
75%	11.000000
max	1154.000000

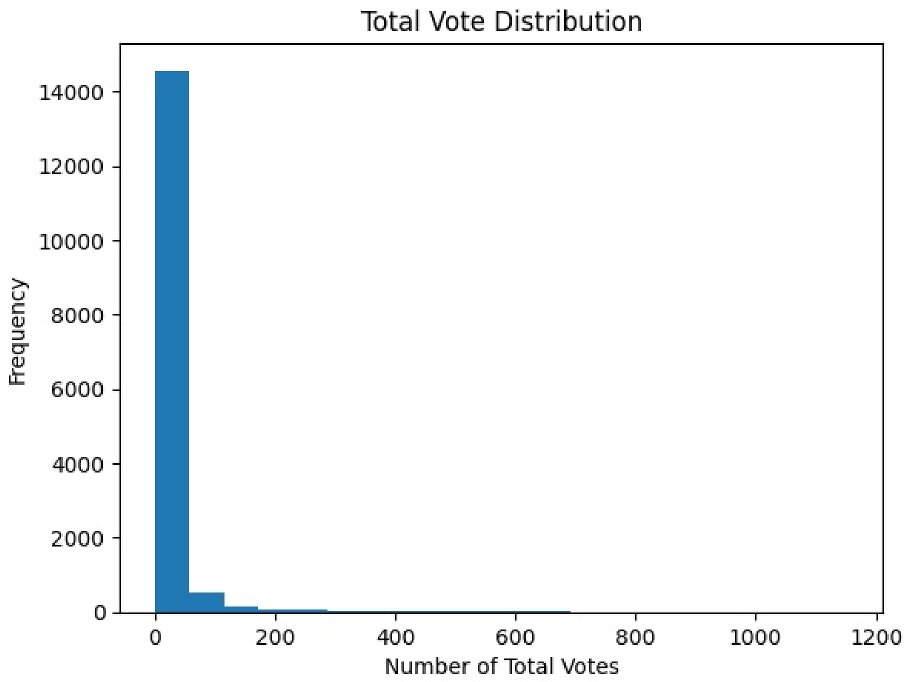
dtype: float64

In [ ]:

```
plt.hist(disney_df["total"], bins=20)
plt.xlabel('Number of Total Votes')
plt.ylabel('Review Frequency')
plt.title('Total Vote Distribution')
```

Out[ ]:

```
Text(0.5, 1.0, 'Total Vote Distribution')
```



date

This variable takes on date values and represents the time of the review.

```
In [ ]: print(disney_df["date"].count(), "(", sum(disney_df["date"].isnull()),")")
```

```
15531 ( 0 )
```

```
In [ ]: disney_df["date"].value_counts()
```

Out[ ]:

date	count
2019-06-22	102
2019-06-23	77
2019-06-24	69
2019-06-21	69
2008-06-27	52
...	...
2015-05-29	1
2017-07-14	1
2013-08-11	1
2013-10-13	1
2012-11-20	1

4952 rows × 1 columns

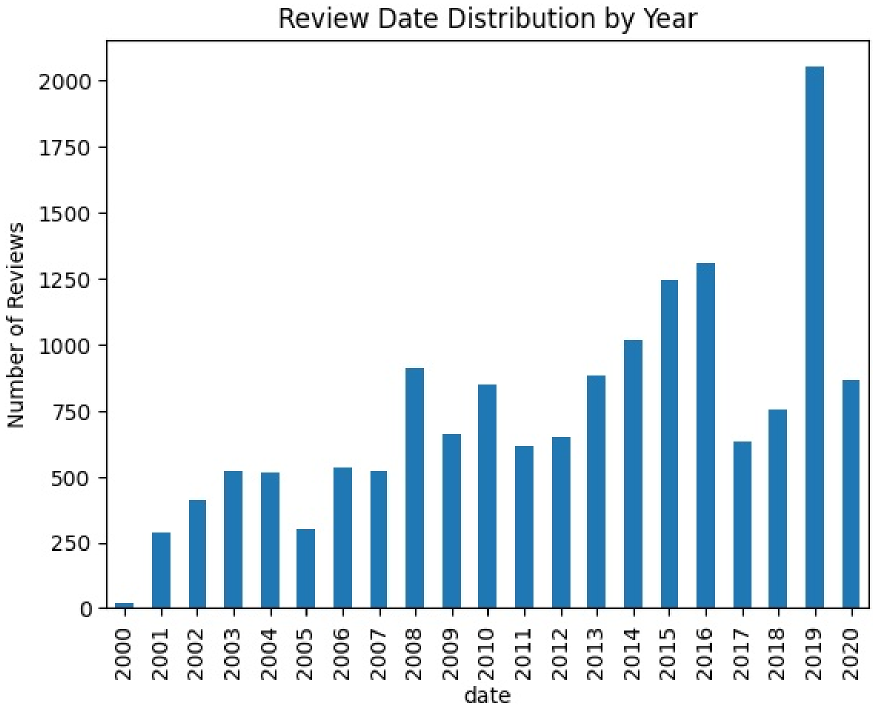
dtype: int64

```
In [ ]: disney_df["date"] = disney_df["date"].astype("datetime64[ns]")
```

```
disney_df.groupby([disney_df["date"].dt.year]).count()["date"].plot(kind="bar",title="Review Date Distribution
```

Out[ ]:

```
<Axes: title={'center': 'Review Date Distribution by Year'}, xlabel='date', ylabel='Number of Reviews'>
```



title

This variable takes on text values and represents the title of the review.

In [ ]:

```
print(disney_df["title"].count(), "(", sum(disney_df["title"].isnull()),")")
```

```
15531 ( 0 )
```

In [ ]:

```
disney_df["title"].value_counts()
```

Out[ ]:

	count
title	
Amazing\n	20
Great movie\n	18
Great Movie\n	18
Beautiful\n	16
Brilliant\n	15
...	...
Big blockbuster success\n	1
Perfect happy ending!\n	1
Dreary and boring\n	1
So good!\n	1
Tremendous world-building\n	1

14424 rows × 1 columns

dtype: int64

review

This variable takes on text values and represents the actual review content of the review.

In [ ]:

```
print(disney_df["review"].count(), "(", sum(disney_df["review"].isnull()),")")
```

```
15531 ( 0 )
```

In [ ]:

```
disney_df["review"].describe()
```

Out[ ]:

	review
count	15531
unique	15491
top	Finding Nemo is a good movie and the fact that...
freq	5

dtype: object

UniqueID

This variable takes on text values and represents a unique identifier for the movie the review is based on.

In [ ]:

```
print(disney_df["UniqueID"].count(), "(", sum(disney_df["UniqueID"].isnull()),")")
```

In [ ]:

```
disney_df["UniqueID"].value_counts()
```

Out[ ]:

	count
UniqueID	
Toy Story 4 2019	1253
WALL·E 2008	1243
Frozen 2013	1109
Inside Out 2015	1012
Finding Nemo 2003	951
The Incredibles 2004	936
Up 2009	870
Toy Story 3 2010	864
Ratatouille 2007	731
Zootopia 2016	701
Cars 2006	663
Monsters, Inc. 2001	660
Big Hero 6 2014	547
Moana 2016	520
Brave 2012	515
Tangled 2010	489
Wreck-It Ralph 2012	442
Finding Dory 2016	427
Lilo & Stitch 2002	393
Cars 2 2011	336
Monsters University 2013	331
The Emperor's New Groove 2000	324
Bolt 2008	214

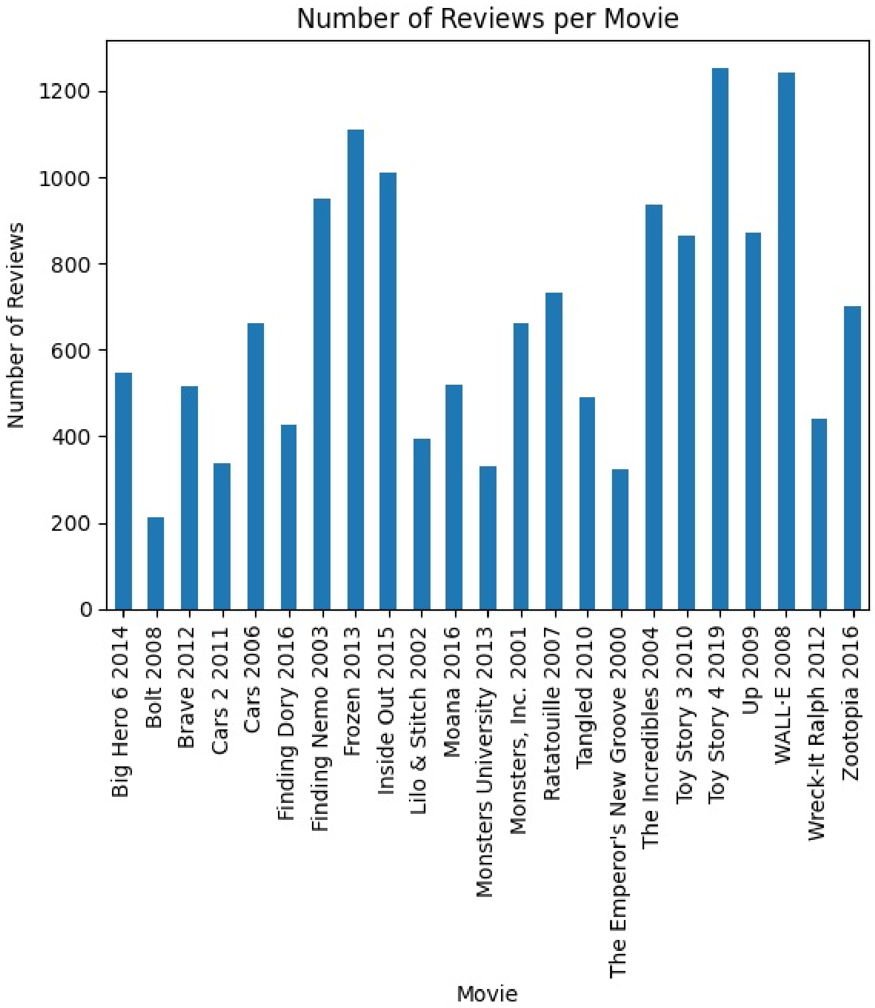
dtype: int64

In [ ]:

```
disney_df.groupby(disney_df["UniqueID"]).count()["review"].plot(kind="bar",title="Number of Reviews per Movie",
```

Out[ ]:

```
<Axes: title={'center': 'Number of Reviews per Movie'}, xlabel='Movie', ylabel='Number of Reviews'>
```



release_date

This variable takes on date values and represents the date of the movie release.

In [ ]:
```
print(disney_df["release_date"].count(), "(", sum(disney_df["release_date"].isnull()),")")
```

```
15531 ( 0 )
```

In [ ]:
```
disney_df["release_date"].value_counts()
```

Out[ ]:

release_date	count
6/21/2019	1253
6/27/2008	1243
11/27/2013	1109
6/19/2015	1012
5/30/2003	951
11/5/2004	936
5/29/2009	870
6/18/2010	864
6/29/2007	731
3/17/2016	701
6/9/2006	663
11/2/2001	660
11/7/2014	547
11/23/2016	520
6/22/2012	515
11/24/2010	489
11/2/2012	442
6/17/2016	427
6/21/2002	393
6/24/2011	336
6/21/2013	331
12/15/2000	324
11/21/2008	214

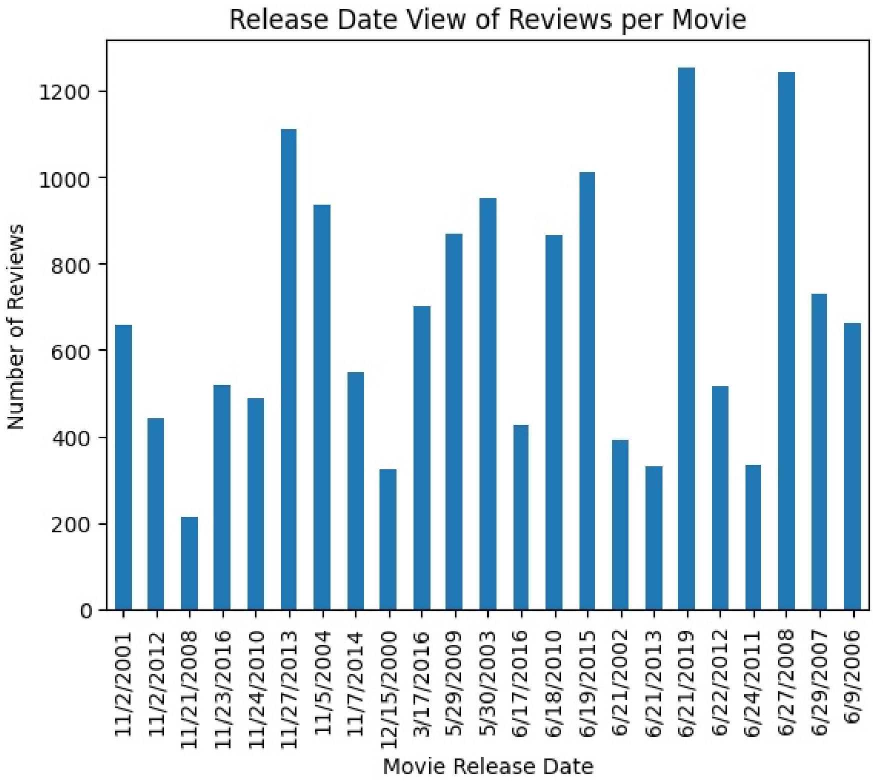
dtype: int64

In [ ]:

```
disney_df.groupby(disney_df["release_date"]).count()["review"].plot(kind="bar",title="Release Date View of Revi
```

Out[ ]:

```
<Axes: title={'center': 'Release Date View of Reviews per Movie'}, xlabel='Movie Release Date', ylabel='Number
of Reviews'>
```



recent?

This variable takes on values 0 or 1 and represents the recency status of the review, which is calculated based on the movie release date (release_date) and review time (date) difference. After comparing the difference, if it was within a year, a 1 would be assigned as the column value and if it was not within a year, a 0 would be assigned.

In [ ]:

```
print(disney_df["recent?"].count(), "(", sum(disney_df["recent?"].isnull()),")")
```

```
15531 ( 0 )
```

In [ ]:

```
temp_df = disney_df["recent?"].astype("category")
temp_df.value_counts()
```

Out[ ]:

recent?	count
1	9996
0	5535

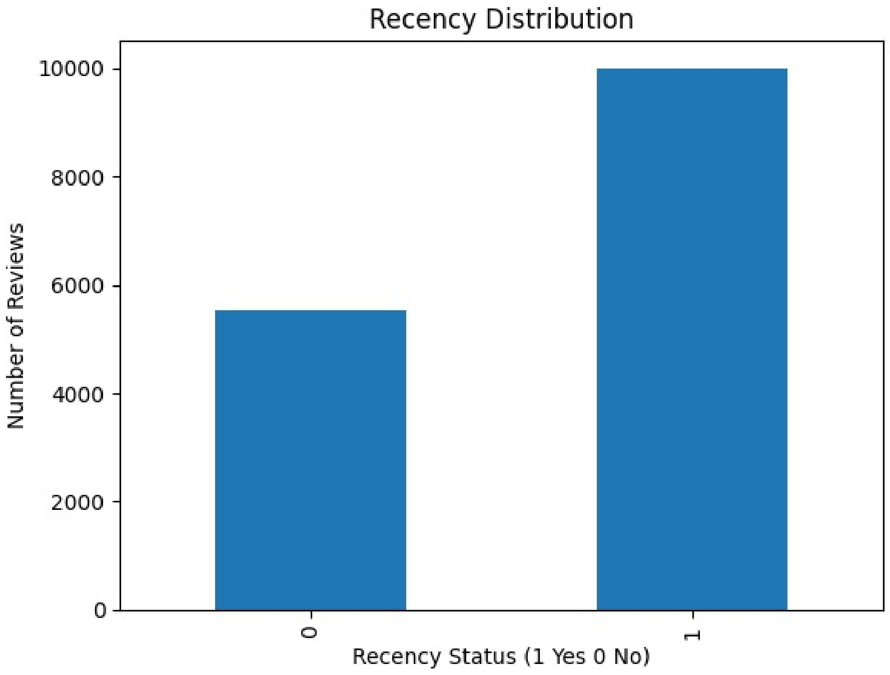
dtype: int64

In [ ]:

```
disney_df.groupby(disney_df["recent?"]).count()["rating"].plot(kind="bar",title="Recency Distribution", ylabel=
```

Out[ ]:

```
<Axes: title={'center': 'Recency Distribution'}, xlabel='Recency Status (1 Yes 0 No)', ylabel='Number of Review
s'>
```



negative

This variable takes on values between 0 and 1 and represents the proportion of text that fall in the negative category based on the Vader sentiment analysis.

In [ ]:

```
print(disney_df["negative"].count(), "(", sum(disney_df["negative"].isnull()),")")
```

```
15531 ( 0 )
```

In [ ]:

```
disney_df["negative"].describe()
```

Out[ ]:

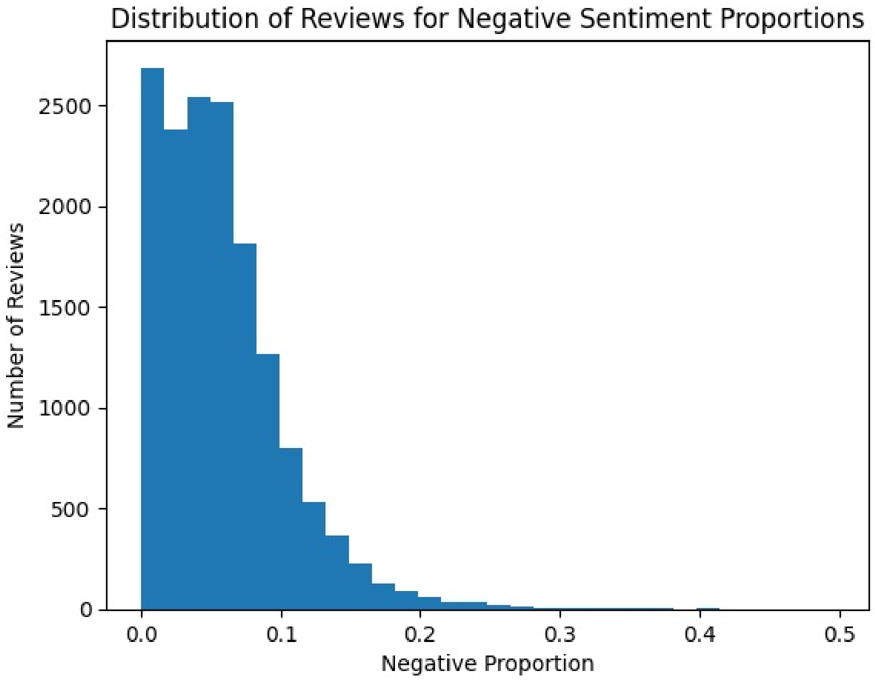
	negative
count	15531.000000
mean	0.057636
std	0.045903
min	0.000000
25%	0.026000
50%	0.050000
75%	0.079000
max	0.497000

dtype: float64

In [ ]:

```
plt.hist(disney_df["negative"], bins=30)
plt.xlabel('Negative Proportion')
plt.ylabel('Number of Reviews')
plt.title('Distribution of Negative Sentiment Proportions')
```

Out[ ]:
```
Text(0.5, 1.0, 'Distribution of Reviews for Negative Sentiment Proportions')
```



positive

This variable takes on values between 0 and 1 and represents the proportion of text that fall in the positive category based on the Vader sentiment analysis.

In [ ]:
```
print(disney_df["positive"].count(), "(", sum(disney_df["positive"].isnull()),")")
```

```
15531 ( 0 )
```

In [ ]:
```
disney_df["positive"].describe()
```

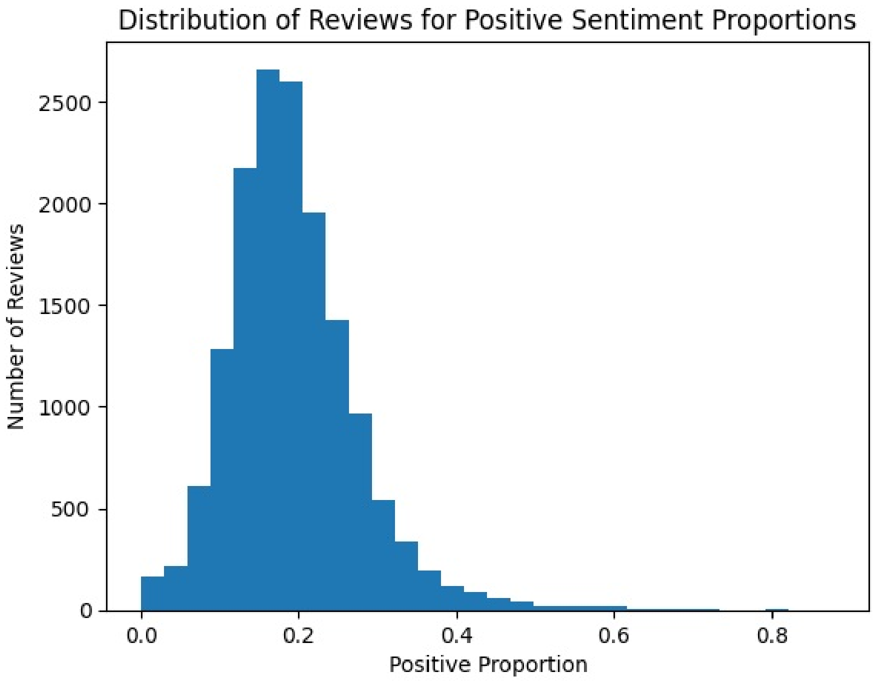
Out[ ]:

	positive
count	15531.000000
mean	0.192899
std	0.083139
min	0.000000
25%	0.140000
50%	0.183000
75%	0.234000
max	0.880000

dtype: float64

In [ ]:
```
plt.hist(disney_df["positive"], bins=30)
plt.xlabel('Positive Proportion')
plt.ylabel('Number of Reviews')
plt.title('Distribution of Positive Sentiment Proportions')
```

Out[ ]:
```
Text(0.5, 1.0, 'Distribution of Reviews for Positive Sentiment Proportions')
```



compound

This variable takes on values between -1 and 1 and represents the normalized sentiment measure for the given review. Generally, neutral sentiment values are classified as being around 0, more positive as values get closer to 1 and more negative as values get closer to -1.

In [ ]:

```
print(disney_df["compound"].count(), "(", sum(disney_df["compound"].isnull()),")")
```

```
15531 ( 0 )
```

In [ ]:

```
disney_df["compound"].describe()
```

Out[ ]:

	compound
count	15531.000000
mean	0.794716
std	0.462184
min	-0.999100
25%	0.894950
50%	0.977900
75%	0.993450
max	0.999900

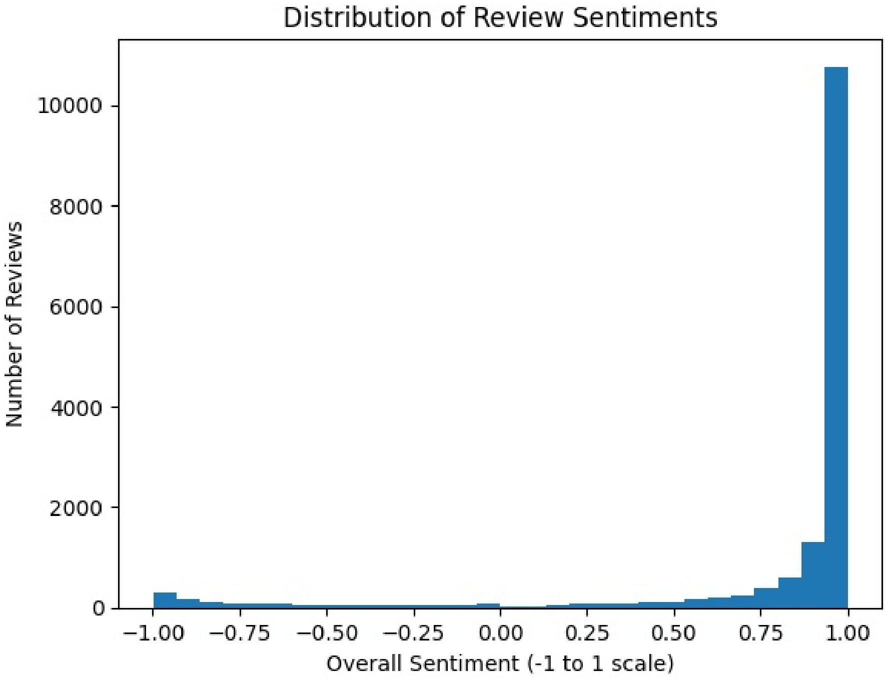
dtype: float64

In [ ]:

```
plt.hist(disney_df["compound"], bins=30)
plt.xlabel('Overall Sentiment (-1 to 1 scale)')
plt.ylabel('Number of Reviews')
plt.title('Distribution of Review Sentiments')
```

Out[ ]:

```
Text(0.5, 1.0, 'Distribution of Review Sentiments')
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js