Creating & Visualizing a Movie Database

Corbin Couger

4/07/2023

Milestone 1

For this project I'm going to look at movies and how their reviews stand amongst public opinion vs. critical opinion. I am really intrigued by this because I am an advent movie goer. I enjoy cinema and am always curious how a critics review lines up with mine or the public's view. I know there have been many times throughout the past where a movie tanks in the rotten tomatoes reviews, but ends up being highly praised and enjoyed by the public. As this first milestone advances, I will explain more of my plans, but to start I know to accomplish this task, I will need some data. Looking over many sources to try and build a final dataset, I have come across 3 separate datasets all from different sources and formats.

Datasets & Their Description

1. Rotten Tomatoes Dataset (csv file):

This dataset is from kaggle and is a compliation of many different variables on many different films. A few of the 26 different variables include:

- title (movie title)
- year (year movie was released)
- critic_score (rotten tomatoes score)
- total_reviews (total number of reviews for the movie)
- genre (movie's genre)
- etc.

This dataset gives me everything I need and more to be able to capture the critical opinon side of my analysis.

URL: [https://www.kaggle.com/datasets/thedevastator/rotten-tomatoes-top-movies-ratings-and-technical (https://www.kaggle.com/datasets/thedevastator/rotten-tomatoes-top-movies-ratings-and-technical)](https://www.kaggle.com/datasets/thedevastator/rotten-tomatoes-top-movies-ratings-and-technical)

2. TMDB (api):

This api offers a lot for the public opinion side of movie ratings, it is an open api that captures many different audience scores. If you have ever used Letterboxd, it is a similar interface as that. It uses public opinion, reivews, recommendations, etc. I first wanted to use LetterBoxd which is where I personally go to get public reviews and movie recommendations, but after some exploration this api is free, and seems to be a great alternative. Some of the variables I would want to pull in from this api include:

- title (movie title)

- title (movie title)
- year (year movie was released)
- user score (public's score)
- genre (movies' genre)
- etc.

I know this source will be reliable and also provide more than enough info for my analysis.

URL: https://www.themoviedb.org/ (https://www.themoviedb.org/)

3. List of Academy Award-winning films (wikipedia table)

To sweeten things up in this little competition between public and critical opinion, I am also going to add in if the movie has won an academy award. This isn't just if the movie itself won 'best picture', it is if it won any academy award in any category. This is something I want to add as, though the academy are technically 'critics', quite a bit of public opinion play into it. This is something I will find to be interesting and will want to pull into this analysis. The variables of this table include:

- film (movie title)
- year (year movie was released)
- awards (number of awards movie won)
- nominations (number of nominations the movie recieved)

This will be a good determination if public or critical opinion matters when it comes to Oscar awards.

URL: https://en.wikipedia.org/wiki/List_of_Academy_Award-winning_films (https://en.wikipedia.org/wiki/List_of_Academy_Award-winning_films)

The Data Relationship

For this data to come together I will be joining the data using the movie's title and, because there could be multiple different movies with the same title, the year the movie was released. I do feel that this will help connect the movies correctly, and will get me the dataset I'm imagining. I plan to first bring in each dataset and as I'm bringing them in, I will want to strip each movie title of whitespace and speical characters, I will also want to lowercase each movie title as I want to make sure, no matter how the data comes in, that it all looks the same. This is how I plan to join the three datasets together and overall I will gather around 30 columns and 1000+ rows of data.

The one note I want to add is for my final data source (List of Academy Award-winning films) is that not all movies will have fields populate here. This is only going to populate for those movies that recieved nominations and won.

How I will accomplish this

My plan for this project is going to be simple, I want to follow each milestone the best I can; so here is how I want to tackle each milestone ahead and what I believe I have to do to accomplish them. For milestone 2 I will focus on cleaning and formatting the first data source (Rotten Tomatoes csv). I'm going to come up with a universal set of headers for each data source I bring in; this will mostly be for just the 'movie title' and 'year' variable so

that each dataset will merge easier. It being a csv will make it easier than the others to bring into python. It will be a pandas dataframe and should simply get imported and then manipulated to remain consistent with the others and will get rid of any outliers, bad data or duplicates. In this step I will also go into the ethical implications of my project (if any).

Like Milestone 2, Milestone 3 will be importing some data. This time it will be the website data (List of Academy Award-winning films from Wikipedia) and will be reading it in using the URL. I will do some transformations to implement the data into the main dataset and keep it consistent with the others. Like before, I will remove any outliers (which there shouldn't be any), bad data, and duplicates. I will do the same for Milestone 4 which is connecting and pulling in the other data source (TMDB api) and cleaning and transforming that like the others. After this I will start to merge the data sets together on the consistent variables I've mentioned. With the clean data, it should merge together smoothly and then I will begin visualizing the database. I will visualize with various methods (different packages and software) and will create meaningful plots with the data.

Project Description

In this analysis, I want to visualize the difference between public vs. critical opinions on various films. I will even validate some of these opinions with awards a film has won.

Meaning, Ethics, and Challenges

This project has meaning for me because I really enjoy film and all entertainment in general. Though I'm not a writer and don't write movie reviews often, I still enjoy seeing genuine authentic opinions on film. In most cases, genuine reactions and thoughts on films tend to be more accurate, digestible, and funny. I find that some critic reviews, mostly from rotten tomatoes, are taken too seriously and I will most likely not take a critic review into account as a recommendation over a public review. This could be because I have come to trust reviews/recommendations from apps like LetterBoxd, instead of looking up a movie online. The community on that app alone is huge and, in my opinion, there is much more meaning in that app and what the community there has built, then what is done by the critics. Ethically, I don't see a lot wrong with this project nor the data I'm pulling in. It is all ethically obtained, and there isn't any problems with connecting these data sources. There have been times in the past however, where a film gets poor critic reviews and then that movie flops. I do think that is a concern, but nothing I'm dealing with in this project. As for challenges, I could have a hard time pulling in the api data. I'm still a beginner with that and will be heavily reliant on the api documentation to get that data in. Beyond that, my challenges will more likely come spontaneously and I do not plan on facing any difficulties completing this project if I stick to the plan.

Milestone 2

Now that I've described my vision, plan, and meaning (as well as finding the data), I'm going to start pulling in some data and will be starting with my flat file (Rotten Tomatoes Dataset csv file) and do some transformations on it to get it ready for connecting it to my other sources. So, let me get started!

In [204]:

```
import pandas as pd
import unicodedata
import sys
import matplotlib.pyplot as plt
#milestone 3
from bs4 import BeautifulSoup
import requests

#milestone 4
import urllib.request
import urllib.parse
import urllib.error
import json

#milestone5
import sqlite3
import numpy as np
```

In [2]:
```
rotten_raw_df = pd.read_csv('rotten_tomatoes_top_movies.csv')
rotten_raw_df.head()
```

Out[2]:

	Unnamed: 0	title	year	synopsis	critic_score	people_score	consensus	total_
0	0	Black Panther	2018	After the death of his father, T'Challa return...	96	79.0	Black Panther elevates superhero cinema to thr...	
1	1	Avengers: Endgame	2019	Adrift in space with no food or water, Tony St...	94	90.0	Exciting, entertaining, and emotionally impact...	
2	2	Mission: Impossible -- Fallout	2018	Ethan Hunt and the IMF team join forces with C...	97	88.0	Fast, sleek, and fun, Mission: Impossible - Fa...	
3	3	Mad Max: Fury Road	2015	Years after the collapse of civilization, the ...	97	86.0	With exhilarating action and a surprising amou...	
4	4	Spider-Man: Into the Spider-Verse	2018	Bitten by a radioactive spider in the subway, ...	97	93.0	Spider-Man: Into the Spider-Verse matches bold...	

5 rows × 26 columns

I'll now look at the datatypes for each column to see if I'm working with what I am needing:

In [3]:
```
rotten_raw_df.dtypes
```

Out[3]:
```
Unnamed: 0                    int64
title                        object
year                          int64
synopsis                     object
critic_score                  int64
people_score                float64
consensus                    object
total_reviews                 int64
total_ratings                object
type                         object
rating                       object
genre                        object
original_language            object
director                     object
producer                     object
writer                       object
release_date_(theaters)      object
release_date_(streaming)     object
box_office_(gross_usa)       object
runtime                      object
production_co                object
sound_mix                    object
aspect_ratio                 object
view_the_collection          object
crew                         object
link                         object
dtype: object
```

Ok most of the variables I'm interested in seem to be the type I need them to be so I will start with some transformations.

1. Cleaning Up 'title' Variable

Since I will eventually be joining all my data together using the title and release year columns, I need a clean universal 'title' column. I am planning on making them all lower case, removing any extra characters (punctuation, dashes, etc.). This way even if my sources have inconsistent data entry (ex. source 1: 'The truman Show', source 2: 'the Truman Show.') I will be able to still match the data up.

In [4]:
```
rotten_raw_df.title = [title.lower() for title in rotten_raw_df.title]
```

In [5]:
```
rotten_raw_df.head()
```

Out[5]:

	Unnamed: 0	title	year	synopsis	critic_score	people_score	consensus	total_
0	0	black panther	2018	After the death of his father, T'Challa return...	96	79.0	Black Panther elevates superhero cinema to thr...	
1	1	avengers: endgame	2019	Adrift in space with no food or water, Tony St...	94	90.0	Exciting, entertaining, and emotionally impact...	
2	2	mission: impossible -- fallout	2018	Ethan Hunt and the IMF team join forces with C...	97	88.0	Fast, sleek, and fun, Mission: Impossible - Fa...	
3	3	mad max: fury road	2015	Years after the collapse of civilization, the ...	97	86.0	With exhilarating action and a surprising amou...	
4	4	spider-man: into the spider-verse	2018	Bitten by a radioactive spider in the subway, ...	97	93.0	Spider-Man: Into the Spider-Verse matches bold...	

5 rows × 26 columns

In [6]:
```
rotten_raw_df.title = [title.strip() for title in rotten_raw_df.title] 

punctuation = dict.fromkeys(i for i in range(sys.maxunicode)
    if unicodedata.category(chr(i)).startswith('P'))

rotten_raw_df.title = [title.translate(punctuation) for title in rotten_
```

In [7]:
```
rotten_raw_df
```

Out[7]:

	Unnamed: 0	title	year	synopsis	critic_score	people_score	cons
0	0	black panther	2018	After the death of his father, T'Challa return...	96	79.0	P el sup cine
1	1	avengers endgame	2019	Adrift in space with no food or water, Tony St...	94	90.0	E enter emot im
2	2	mission impossible	2018	Ethan Hunt and the IMF team join	97	88.0	Fast, ar M

Ok cool! That is taken care of, and onto the next!

2. Checking for Blanks (NaN)

I want to check the two columns I will be joining my datasets on for blanks/NULL values. This is going to be important as I dont want there to be any data that is missing in this analysis, because I could throw off my model(s). So, I'll get to it!

In [8]:
```
print('NaN for title:', sum(rotten_raw_df.title.isnull()))
```

```
NaN for title: 0
```

In [9]:
```
print('NaN for year:', sum(rotten_raw_df.year.isnull()))
```

```
NaN for year: 0
```

Perfect! One thing I do not need to worry about for this source!

3. Cleaning Up 'type' variable

Since I wam going to comparing reviews between the public and critics, I might want to broaden my analysis to get another angle. So, I will be using a cleaned up version of the 'type' or 'genre' of the films to see which reviews were more impactful to awards and money made.

In [10]:
```
rotten_raw_df.type = [t.lower() for t in rotten_raw_df.type] # lower ca
rotten_raw_df.type = [t.strip() for t in rotten_raw_df.type] # removes
rotten_raw_df.type = [t.translate(punctuation) for t in rotten_raw_df.t
```

In [11]:
```
rotten_raw_df.head()
```

Out[11]:

	Unnamed: 0	title	year	synopsis	critic_score	people_score	consensus	total
0	0	black panther	2018	After the death of his father, T'Challa return...	96	79.0	Black Panther elevates superhero cinema to thr...	
1	1	avengers endgame	2019	Adrift in space with no food or water, Tony St...	94	90.0	Exciting, entertaining, and emotionally impact...	
2	2	mission impossible fallout	2018	Ethan Hunt and the IMF team join forces with C...	97	88.0	Fast, sleek, and fun, Mission: Impossible - Fa...	
3	3	mad max fury road	2015	Years after the collapse of civilization, the ...	97	86.0	With exhilarating action and a surprising amou...	
4	4	spiderman into the spiderverse	2018	Bitten by a radioactive spider in the subway, ...	97	93.0	Spider-Man: Into the Spider-Verse matches bold...	

5 rows × 26 columns

4. Changing 'box_office_(gross_usa)' to an integer variable

This is one of the variables that I need to change its type. I don't want this varibale to be in the format it is in, I need it to just be a solid number. Example being, $858.4M -> 858,400,000 or 858.4, but note the column is in millions. This might be interesting and could be a little challenge in getting it to the state I want it to be.

When initially testing, there are a few movies that are measured in thousands (k), So I wrote some code to account for that.

In [12]:
```
rotten_raw_df['box_office_(gross_usa)'] = rotten_raw_df["box_office_(gr
```

In [13]:
```
def value_to_float(x):
    if type(x) == float or type(x) == int:
        return x
    if 'K' in x:
        if len(x) > 1:
            return float(x.replace('K', '')) * 1000
        return 1000.0
    if 'M' in x:
        if len(x) > 1:
            return float(x.replace('M', '')) * 1000000
        return 1000000.0
    if 'B' in x:
        return float(x.replace('B', '')) * 1000000000
    return 0.0

rotten_raw_df['box_office_(gross_usa)'] = rotten_raw_df['box_office_(gr
```

In [14]:
```
rotten_raw_df.head()
```

Out[14]:

	Unnamed: 0	title	year	synopsis	critic_score	people_score	consensus	total
0	0	black panther	2018	After the death of his father, T'Challa return...	96	79.0	Black Panther elevates superhero cinema to thr...	
1	1	avengers endgame	2019	Adrift in space with no food or water, Tony St...	94	90.0	Exciting, entertaining, and emotionally impact...	
2	2	mission impossible fallout	2018	Ethan Hunt and the IMF team join forces with C...	97	88.0	Fast, sleek, and fun, Mission: Impossible - Fa...	
3	3	mad max fury road	2015	Years after the collapse of civilization, the ...	97	86.0	With exhilarating action and a surprising amou...	
4	4	spiderman into the spiderverse	2018	Bitten by a radioactive spider in the subway, ...	97	93.0	Spider-Man: Into the Spider-Verse matches bold...	

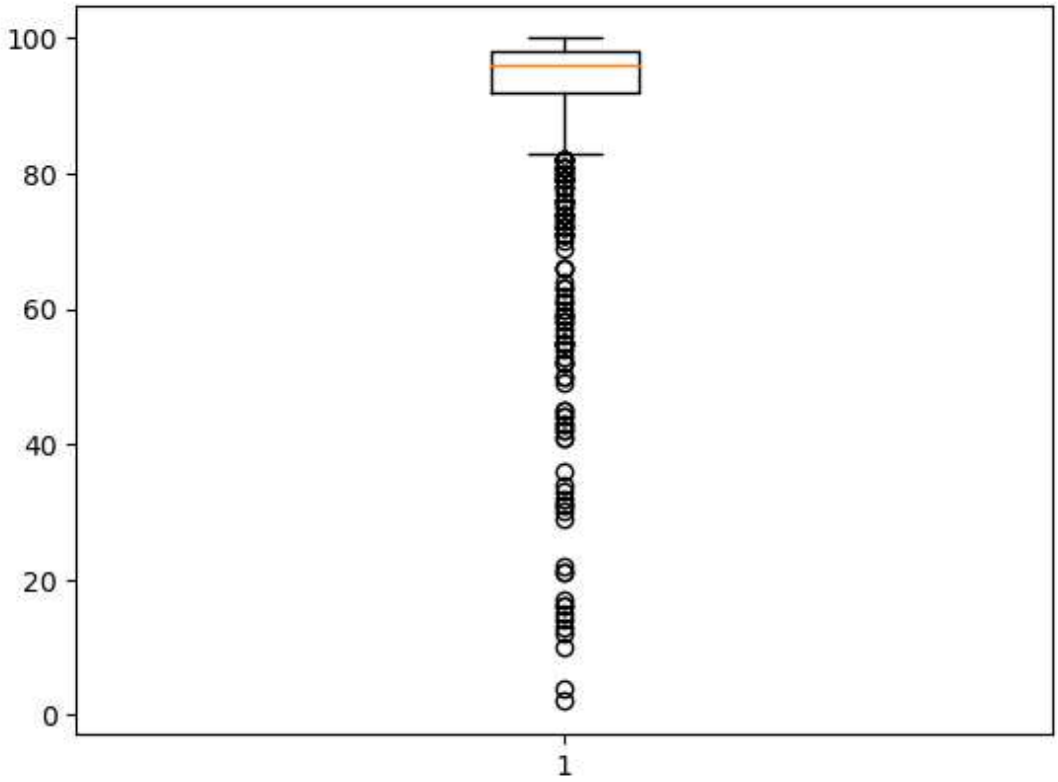
5 rows × 26 columns

5. Outliers in Critic Score

For the last transformation on this source, I'm going to be checking if there are outliers in the 'critic_score' column. At this point, without seeing the other data sources yet, I'm not going to be removing outliers, I just want to check and see if they are there and then brainstorm why.

In [15]:
```
plt.boxplot(rotten_raw_df.critic_score)
```

Out[15]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1957db5dfd0>,
  <matplotlib.lines.Line2D at 0x1957db7c2e0>],
 'caps': [<matplotlib.lines.Line2D at 0x1957db7c5b0>,
  <matplotlib.lines.Line2D at 0x1957db7c880>],
 'boxes': [<matplotlib.lines.Line2D at 0x1957db5dd00>],
 'medians': [<matplotlib.lines.Line2D at 0x1957db7cb50>],
 'fliers': [<matplotlib.lines.Line2D at 0x1957db7ce20>],
 'means': []}
```



Wow, there must be some really bad movies out there according to these critics. This is interesting becasue I wonder where these movies that are outliers here, will be in the public reviews data source. I only (might) remove outliers once I join the datasets together. This way I keep some 'rawness' to it.

Milestone 3

Moving on to my next milestone in this project, importing my website data (List of Academy Award Winning Films)! This one is the one I am anticipating having the biggest challenge with. But, once I get that data imported I will do some similar transformations on it as I have done in the previous milestone to make sure the data is all in the same format.

In [151]:
```
url = 'https://en.wikipedia.org/wiki/List_of_Academy_Award-winning_films
r = requests.get(url)
```

In [152]:
```
soup = BeautifulSoup(r.text, 'html.parser')
wikitables=soup.find('table',{'class':"wikitable"})
wikitables
```

Out[152]:
```
<table class="wikitable sortable">
<tbody><tr>
<th>Film
</th>
<th>Year
</th>
<th data-sort-type="number">Awards
</th>
<th data-sort-type="number">Nominations
</th></tr>
<tr style="background:#EEDD82">
<td><i><b><a href="/wiki/Everything_Everywhere_All_at_Once" title
="Everything Everywhere All at Once">Everything Everywhere All at
Once</a></b></i></td>
<td><a href="/wiki/2022_in_film" title="2022 in film">2022</a></t
d>
<td>7</td>
<td>11
</td></tr>
```

In [153]:
```
df=pd.read_html(str(wikitables))
# convert list to dataframe
academydf=pd.DataFrame(df[0])
```

In [154]:
```
academydf.head()
```

Out[154]:

	Film	Year	Awards	Nominations
0	Everything Everywhere All at Once	2022	7	11
1	All Quiet on the Western Front	2022	4	9
2	The Whale	2022	2	3
3	Top Gun: Maverick	2022	1	6
4	Black Panther: Wakanda Forever	2022	1	5

Phew ok sweet! I got that table imported with the thanks to Pandas and BeautifulSoup, now I will get to some transformations and cleaning.

1. Cleaning Up 'Film' Variable

Like before I want to make sure that these film titles are going to be able to join up nicely when I put all 3 tables together, so I must get them all in the same format like I mentioned before.

In [155]:
```
academydf.Film = [Film.strip() for Film in academydf.Film] # removes an

academydf.Film = [Film.translate(punctuation) for Film in academydf.Fil

academydf.Film = [Film.lower() for Film in academydf.Film] # lowercases
```

In [156]:
```
academydf.head()
```

Out[156]:

	Film	Year	Awards	Nominations
0	everything everywhere all at once	2022	7	11
1	all quiet on the western front	2022	4	9
2	the whale	2022	2	3
3	top gun maverick	2022	1	6
4	black panther wakanda forever	2022	1	5

2. Checking for Blanks (NaN)

Like before, I want to check the two joining variables (in this case: 'Film' and 'Year') and see if they have any missing values.

In [157]:
```
print('NaN for Film variable:', academydf.Film.isnull().sum())
```

```
NaN for Film variable: 0
```

In [158]:
```
print('NaN for Year variable:', academydf.Year.isnull().sum())
```

```
NaN for Year variable: 0
```

Perfect! 0 in each. The one challenge will be when joining these frames; there will be some data sets that dont have some of the same movies as others.. this is something I will discuss later.

3. Changing Data Types

I next want to convert the 'Awards' and 'Nominations' variables to integers as they are both object variables and should be integer. To do this, I must first clean up each column. For a preface, this data set also includes if the film was nominated/won a non-competitive award. This is signified by a parenthesis, ex. 3(2) which would be 3 competitive and 2 non-competitive

In [159]:
```
# example

print(academydf[academydf['Film'].str.contains("raiders of the lost ark
```

```
                         Film  Year Awards Nominations
592  raiders of the lost ark  1981  4 (1)           8
```

As you can see, the 'Awards' column for the 'Raiders of the Lost Ark' film shows 4 (1) which is not going to easily convert to a integer. So, I will have to split that out of there.

In [160]:
```
academydf.dtypes
```

Out[160]:
```
Film           object
Year           object
Awards         object
Nominations    object
dtype: object
```

In [161]:
```
academydf[['Awards_comp', 'Awards_noncomp']] = academydf.Awards.astype(
```

In [162]:
```
academydf.head()
```

Out[162]:

	Film	Year	Awards	Nominations	Awards_comp	Awards_noncomp
0	everything everywhere all at once	2022	7	11	7	None
1	all quiet on the western front	2022	4	9	4	None
2	the whale	2022	2	3	2	None
3	top gun maverick	2022	1	6	1	None
4	black panther wakanda forever	2022	1	5	1	None

In [163]:
```
academydf[['Nominations_comp', 'Nominations_noncomp']] = academydf.Nomi
```

In [164]:
```
academydf.head()
```

Out[164]:

	Film	Year	Awards	Nominations	Awards_comp	Awards_noncomp	Nominatior
0	everything everywhere all at once	2022	7	11	7	None	
1	all quiet on the western front	2022	4	9	4	None	
2	the whale	2022	2	3	2	None	
3	top gun maverick	2022	1	6	1	None	
4	black panther wakanda forever	2022	1	5	1	None	

In [165]:
```
academydf.dtypes
```

Out[165]:
```
Film                 object
Year                 object
Awards               object
Nominations          object
Awards_comp          object
Awards_noncomp       object
Nominations_comp     object
Nominations_noncomp  object
dtype: object
```

In [166]:
```
academydf.Nominations_comp = academydf.Nominations_comp.astype('int')
```

In [167]:
```
academydf.Awards_comp = academydf.Awards_comp.astype('int')
```

In [168]:
```
academydf.dtypes
```

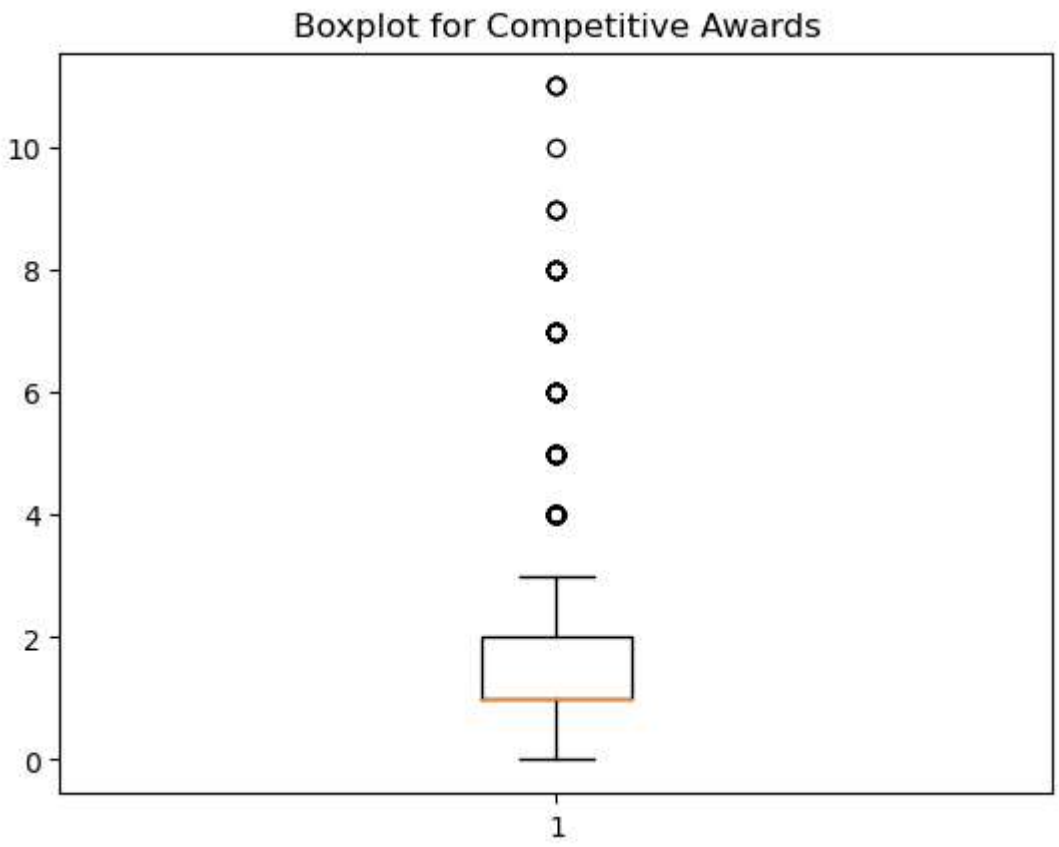
Out[168]:
```
Film                 object
Year                 object
Awards               object
Nominations          object
Awards_comp           int32
Awards_noncomp       object
Nominations_comp      int32
Nominations_noncomp  object
dtype: object
```

4. Checking for Outliers in 'Awards_comp' & 'Nominations_comp'

I want to check for oultiers in each of these variables, though I wont remove them, it will be cool to see which ones are outliers and see how those pair up with the reviews.

In [169]:
```
plt.boxplot(academydf.Awards_comp)
plt.title('Boxplot for Competitive Awards')
```

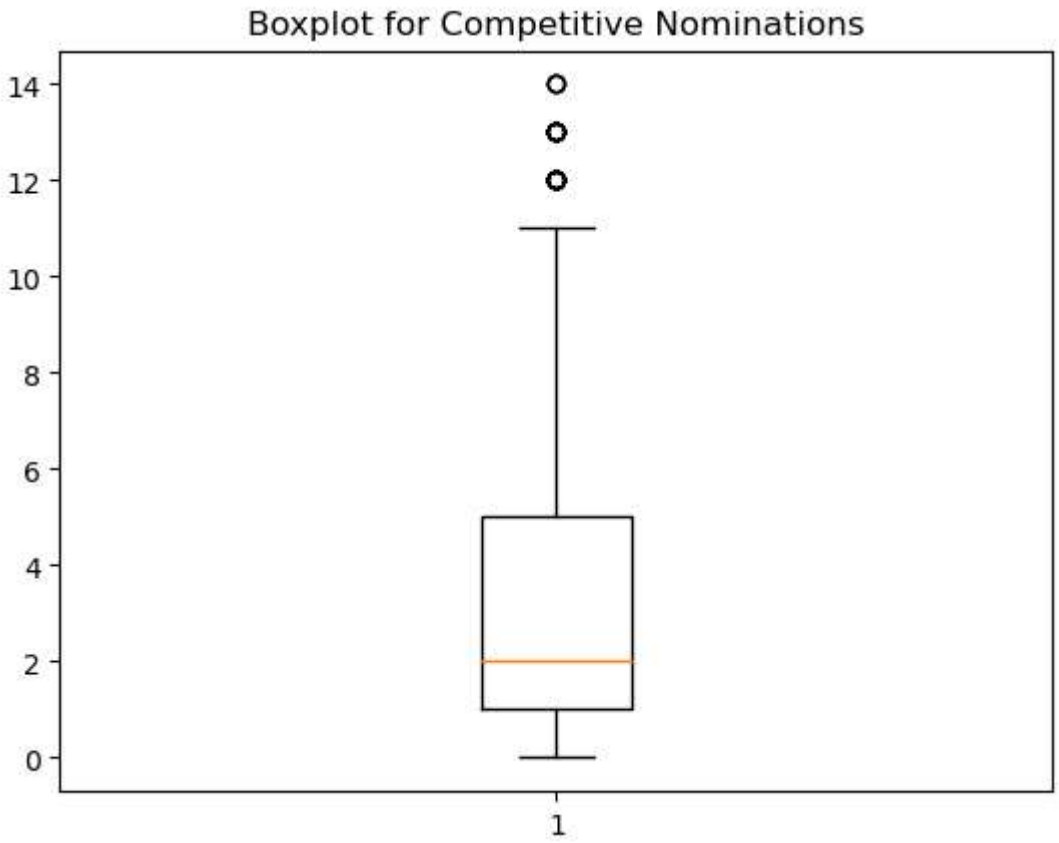
Out[169]:
```
Text(0.5, 1.0, 'Boxplot for Competitive Awards')
```



Looks like any film with 4 or more competitive awards is an outlier in this dataframe.

In [170]:
```
plt.boxplot(academydf.Nominations_comp)
plt.title('Boxplot for Competitive Nominations')
```

Out[170]: Text(0.5, 1.0, 'Boxplot for Competitive Nominations')



Looks like any films with 12 or more nominations are outliers. I will investigate all of these outliers later on in the analysis.

5. Getting Rid of Not-Needed Columns

Now that I have cleaned this dataframe, I want to get rid of the original 'Awards' and 'Nominations' columns as well as the noncomp columns as they will not be needed in my analysis.

In [171]:
```
del academydf['Awards']
del academydf['Nominations']
del academydf['Awards_noncomp']
del academydf['Nominations_noncomp']
```

In [172]:
```
academydf.head()
```

Out[172]:

	Film	Year	Awards_comp	Nominations_comp
0	everything everywhere all at once	2022	7	11
1	all quiet on the western front	2022	4	9
2	the whale	2022	2	3
3	top gun maverick	2022	1	6
4	black panther wakanda forever	2022	1	5

Milestone 4

Now, I have one last dataset to pull in and clean up! In this milestone I will be pulling in my 3rd and final source: the TMDB (API) data. To do this I will use a few useful packages to get this data in as some JSON. Then I will create my data frame out of this and use that to do some transformations.

In [38]:
```
api_key = '813dc569bee81028a382a4035e2f566c'

TMDBurl = f'https://api.themoviedb.org/3/search/movie?api_key={api_key}&
```

1-4. Multiple Transformations

In the following function I do a lot of my transformations on this dataset, I have to do this to get the data from my API. Since this API does not send back a full data set meaning I have to request data by each movie, I use the movies in the Rotten Tomatoes data set.

1. My first transformation is creating a new variable for the movie title that replaces the spaces with +'s. This is to add as an extension on my API request, since my API requires the movie title be on the end of the URL.
2. Secondly, once I have gotten the request, I need to grab the title, year, and score variable for each movie that was sent back with the request. There are some movies that have the same title, and then there are some requests that send back a general search of that movie. But I get rid of uppercase letters, leading and trailing whitespace, and punctuation. This is so I can match it up and check against the Rotten Tomatoes movie title.
3. Thirdly, when pulling in the year variable from the request, I was given a date seperated by -'s so I split the date from the request to just pull the year.
4. Forthly, I needed to match up the variables (title and year) to get the right movie. Within a try/except I have an if statment, the year I pulled in needed to be converted to an integer for it to match up. So, I converted that variable to an integer in that if statement.

In [108]:

```
def get_movie_data(movie, year_match, title_list, year_list, rate_list)
    movie = movie.strip()
    movie_plus = movie.replace(' ', '+')
    new_TMDBurl = TMDBurl+movie_plus
    print(new_TMDBurl)
    try:
        response = requests.request('GET', new_TMDBurl)
    except:
        return print('Failed to connect to API.')


    parsed = json.loads(response.text)
    for r in parsed['results']:
        title = r['original_title']
        year = r['release_date'].split('-')[0]
        score = r['vote_average']
        title = title.lower()
        title = title.strip()
        title = title.translate(punctuation)
        try:
            if title == movie and int(year) == int(year_match):
                list_data = [title, year, score]
                #print(list_data)
                return list_data
            else:
                continue
        except:
                continue

    #return (title_list, year_list, rate_list)
```

In [109]:

```
title_list = []
year_list = []
rate_list = []
for m, y in zip(rotten_raw_df.title, rotten_raw_df.year):
    list_data = get_movie_data(m, y, title_list, year_list, rate_list)
    print(list_data)
    try:
        title_list.append(list_data[0])
        year_list.append(list_data[1])
        rate_list.append(list_data[2])
    except:
        print(m, 'does not exist in API database')
        continue
    #break


'''
for m in rotten_raw_df.title:
    print(m)
'''
```

```
https://api.themoviedb.org/3/search/movie?api_key=813dc569bee8102
8a382a4035e2f566c&query=black+panther (https://api.themoviedb.or
g/3/search/movie?api_key=813dc569bee81028a382a4035e2f566c&query=b
lack+panther)
['black panther', '2018', 7.393]
https://api.themoviedb.org/3/search/movie?api_key=813dc569bee8102
8a382a4035e2f566c&query=avengers+endgame (https://api.themoviedb.
org/3/search/movie?api_key=813dc569bee81028a382a4035e2f566c&query
=avengers+endgame)
['avengers endgame', '2019', 8.266]
https://api.themoviedb.org/3/search/movie?api_key=813dc569bee8102
8a382a4035e2f566c&query=mission+impossible++fallout (https://api.
themoviedb.org/3/search/movie?api_key=813dc569bee81028a382a4035e2
f566c&query=mission+impossible++fallout)
['mission impossible  fallout', '2018', 7.402]
https://api.themoviedb.org/3/search/movie?api_key=813dc569bee8102
8a382a4035e2f566c&query=mad+max+fury+road (https://api.themovied
b.org/3/search/movie?api_key=813dc569bee81028a382a4035e2f566c&que
ry=mad+max+fury+road)
```

In [110]:
```
title_list
```

Out[110]:
```
['black panther',
 'avengers endgame',
 'mission impossible  fallout',
 'mad max fury road',
 'spiderman into the spiderverse',
 'wonder woman',
 'dunkirk',
 'coco',
 'thor ragnarok',
 'logan',
 'star wars the last jedi',
 'star wars the force awakens',
 'the adventures of robin hood',
 'king kong',
 'spiderman far from home',
 'incredibles 2',
 'zootopia',
 'war for the planet of the apes',
 'spiderman homecoming',
```

In [111]:
```
year_list
```

Out[111]:
```
['2018',
 '2019',
 '2018',
 '2015',
 '2018',
 '2017',
 '2017',
 '2017',
 '2017',
 '2017',
 '2017',
 '2015',
 '1938',
 '1933',
 '2019',
 '2018',
 '2016',
 '2017',
 '2017',
```

In [112]:
```
rate_list
```

Out[112]:
```
[7.393,
 8.266,
 7.402,
 7.575,
 8.406,
 7.24,
 7.459,
 8.226,
 7.599,
 7.815,
 6.837,
 7.3,
 7.464,
 7.629,
 7.459,
 7.485,
 7.747,
 7.156,
 7.349,
```

In [115]:
```
TMDB_df = pd.DataFrame(list(zip(title_list, year_list, rate_list)),
               columns =['title', 'year_released', 'public_rating'])
TMDB_df.to_csv('TMDB_df.csv')
```

In [123]:
```
TMDB_df = pd.read_csv('TMDB_df.csv')
del TMDB_df['Unnamed: 0']
TMDB_df.head()
```

Out[123]:

	title	year_released	public_rating
0	black panther	2018	7.393
1	avengers endgame	2019	8.266
2	mission impossible fallout	2018	7.402
3	mad max fury road	2015	7.575
4	spiderman into the spiderverse	2018	8.406

5. Changing 'public_rating' Variable

Something that I want to keep consistent is units in how the movie ratings are measured. The Rotten Tomatoes reviews are taken out of 100 and have no decimal places. The TMDB ratings are taken out of 10 and have 3 decimal places. So, I am going to convert and round the 'public_rating' variable to match up with the same units as the Rotten Tomatoe rating.

In [124]:
```
TMDB_df.public_rating = [round(r*10, 0) for r in TMDB_df.public_rating]
TMDB_df.head()
```

Out[124]:

	title	year_released	public_rating
0	black panther	2018	74.0
1	avengers endgame	2019	83.0
2	mission impossible fallout	2018	74.0
3	mad max fury road	2015	76.0
4	spiderman into the spiderverse	2018	84.0

Milestone 5.

Wrapping up this wrangling of my data and project, I can now finally get my data to interact with eachother. I will be creating a database where these 3 tables will live! Once done, I can join them together and then create one big database from that, then I can create some visualizations with this big dataframe. I'm excited to get this going so let me get to it!

Rotten Tomatoes

I'll start creating tables in the order I pulled my data sources in, so first the Rotten Tomatoes cleaned dataframe.

In [131]:
```
#bringing down all column names and types for each table to make it eas
rotten_raw_df.dtypes
#del rotten_raw_df['Unnamed: 0']
```

Out[131]:
```
title                          object
year                            int64
synopsis                       object
critic_score                    int64
people_score                  float64
consensus                      object
total_reviews                   int64
total_ratings                  object
type                           object
rating                         object
genre                          object
original_language              object
director                       object
producer                       object
writer                         object
release_date_(theaters)        object
release_date_(streaming)       object
box_office_(gross_usa)        float64
runtime                        object
production_co                  object
sound_mix                      object
aspect_ratio                   object
view_the_collection            object
crew                           object
link                           object
dtype: object
```

In [138]:

```
rotten_name = 'RottenTomato'

conn = sqlite3.connect('movie.db')
query = '''Create table if not Exists RottenTomato (title text,
year integer,
synopsis text,
critic_score integer,
people_score real,
consensus text,
total_reviews integer,
total_ratings text,
type text,
rating text,
genre text,
original_language text,
director text,
producer text,
writer text,
release_date_theaters text,
release_date_streaming text,
box_office_gross_usa real,
runtime text,
production_co text,
sound_mix text,
aspect_ratio text,
view_the_collection text,
crew text,
link text)'''
cursor = conn.cursor()
cursor.execute(query)
rotten_raw_df.to_sql(rotten_name,conn,if_exists='replace',index=False)
conn.commit()
conn.close()
```

In [140]:
```
conn = sqlite3.connect('movie.db')
cursor = conn.cursor()

rottentest = cursor.execute('SELECT * FROM RottenTomato LIMIT 25')
for row in rottentest:
    print(row)

conn.close()
```

```
('black panther', 2018, "After the death of his father, T'Challa
returns home to the African nation of Wakanda to take his rightfu
l place as king. When a powerful enemy suddenly reappears, T'Chal
la's mettle as king -- and as Black Panther -- gets tested when h
e's drawn into a conflict that puts the fate of Wakanda and the e
ntire world at risk. Faced with treachery and danger, the young k
ing must rally his allies and release the full power of Black Pan
ther to defeat his foes and secure the safety of his people.", 9
6, 79.0, "Black Panther elevates superhero cinema to thrilling ne
w heights while telling one of the MCU's most absorbing stories -
- and introducing some of its most fully realized characters.", 5
19, '50,000+', 'action  adventure', 'PG-13 (Sequences of Action V
iolence|A Brief Rude Gesture)', 'adventure, action, fantasy', 'En
glish', 'Ryan Coogler', 'Kevin Feige', 'Ryan Coogler, Joe Robert
Cole', 'Feb 16, 2018 wide', 'May 2, 2018', 700200000.0, '2h 14m',
'Walt Disney Pictures', 'DTS, Dolby Atmos', 'Scope (2.35:1)', 'Ma
rvel Cinematic Universe', "Chadwick Boseman, Michael B. Jordan, L
upita Nyong'o, Danai Gurira, Martin Freeman, Daniel Kaluuya, Leti
tia Wright, Winston Duke, Angela Bassett, Forest Whitaker, Ryan C
```

Academy Awards

Now I have the RottenTomato table in my movies database, I can now create the Academy table using the cleaned academy dataframe. There still is some cleaning to do with the year column like I had mentioned above, I will deal with this now

In [176]:
```
for y in academydf.Year:
    print(y)
```

```
2022
2022
2022
2022
2022
2022
2022
2022
2022
2022
2022
2022
2022
2021
2021
2021
2021
2021
2021
```

Some dates have extensions (ex. 2020/21), so basically when the movie was in theaters/being released. So I will just chop off that last half of those years with that extension.

In [177]:
```
academydf['Year'] = academydf['Year'].str.split('/').str[0]
```

In [178]:
```
for y in academydf.Year:
    print(y)
```

```
2022
2022
2022
2022
2022
2022
2022
2022
2022
2022
2022
2022
2022
2021
2021
2021
2021
2021
2021
```

In [179]:
```
academydf.dtypes
academydf['Year'] = academydf['Year'].astype('int')
```

In [180]:
```
academydf.dtypes
```

Out[180]:
```
Film               object
Year                int32
Awards_comp         int32
Nominations_comp    int32
dtype: object
```

In [181]:
```
academy_name = 'academy'

conn = sqlite3.connect('movie.db')
query = '''Create table if not Exists academy (film text, year integer,
cursor = conn.cursor()
cursor.execute(query)
academydf.to_sql(academy_name,conn,if_exists='replace',index=False)
conn.commit()
conn.close()
```

In [182]:
```
conn = sqlite3.connect('movie.db')
cursor = conn.cursor()
academytest = cursor.execute('SELECT * FROM academy LIMIT 25')
for row in academytest:
    print(row)
```

```
('everything everywhere all at once', 2022, 7, 11)
('all quiet on the western front', 2022, 4, 9)
('the whale', 2022, 2, 3)
('top gun maverick', 2022, 1, 6)
('black panther wakanda forever', 2022, 1, 5)
('avatar the way of water', 2022, 1, 4)
('women talking', 2022, 1, 2)
('guillermo del toros pinocchio', 2022, 1, 1)
('navalny', 2022, 1, 1)
('the elephant whisperers', 2022, 1, 1)
('an irish goodbye', 2022, 1, 1)
('the boy the mole the fox and the horse', 2022, 1, 1)
('rrr', 2022, 1, 1)
('coda', 2021, 3, 3)
('dune', 2021, 6, 10)
('the eyes of tammy faye', 2021, 2, 2)
('no time to die', 2021, 1, 3)
('the windshield wiper', 2021, 1, 1)
('the long goodbye', 2021, 1, 1)
```

TMDB

Finally I can use the database I created from my API to make the final table I will need in my database.

In [183]:
```
TMDB_df.dtypes
```

Out[183]:
```
title             object
year_released      int64
public_rating    float64
dtype: object
```

In [184]:
```
tmdb_name = 'tmdb'

conn = sqlite3.connect('movie.db')
query = '''Create table if not Exists tmdb (title text, year_released i
cursor = conn.cursor()
cursor.execute(query)
TMDB_df.to_sql(tmdb_name,conn,if_exists='replace',index=False)
conn.commit()
conn.close()
```

In [185]:
```
conn = sqlite3.connect('movie.db')
cursor = conn.cursor()
tmdbtest = cursor.execute('SELECT * FROM tmdb LIMIT 25')
for row in tmdbtest:
    print(row)
conn.close()
```

```
('black panther', 2018, 74.0)
('avengers endgame', 2019, 83.0)
('mission impossible  fallout', 2018, 74.0)
('mad max fury road', 2015, 76.0)
('spiderman into the spiderverse', 2018, 84.0)
('wonder woman', 2017, 72.0)
('dunkirk', 2017, 75.0)
('coco', 2017, 82.0)
('thor ragnarok', 2017, 76.0)
('logan', 2017, 78.0)
('star wars the last jedi', 2017, 68.0)
('star wars the force awakens', 2015, 73.0)
('the adventures of robin hood', 1938, 75.0)
('king kong', 1933, 76.0)
('spiderman far from home', 2019, 75.0)
('incredibles 2', 2018, 75.0)
('zootopia', 2016, 77.0)
('war for the planet of the apes', 2017, 72.0)
('spiderman homecoming', 2017, 73.0)
('baby driver', 2017, 75.0)
('metropolis', 1927, 82.0)
('jaws', 1975, 77.0)
('up', 2009, 79.0)
('shazam', 2019, 70.0)
('the dark knight', 2008, 85.0)
```

Now it is TIME!

Now that I have all my dataframes into a database as tables, I can take those tables, join them up and the create a new dataframe out of that!

In [198]:
```
conn = sqlite3.connect('movie.db')
query = '''SELECT DISTINCT academy.film, academy.year, RottenTomato.cri
FROM academy
JOIN tmdb ON academy.film = tmdb.title AND academy.year = tmdb.year_rel
JOIN RottenTomato ON academy.film = RottenTomato.title AND academy.year
cursor = conn.cursor()
movietest = cursor.execute(query)
#for row in movietest:
    #print(row)

moviedf = pd.read_sql(query,conn)
moviedf.head(10)
```

Out[198]:

	Film	Year	critic_score	public_rating	Awards_comp	Nominations_comp
0	toy story 4	2019	97	75.0	1	2
1	once upon a time in hollywood	2019	85	74.0	2	10
2	black panther	2018	96	74.0	3	7
3	blackkklansman	2018	96	75.0	1	6
4	spiderman into the spiderverse	2018	97	84.0	1	1
5	dunkirk	2017	92	75.0	3	8
6	three billboards outside ebbing missouri	2017	90	81.0	2	7
7	blade runner 2049	2017	88	75.0	2	5
8	coco	2017	97	82.0	2	2
9	la la land	2016	90	79.0	6	14

In [200]:
```
moviedf.shape
```

Out[200]: (98, 6)

After combining all the data tables together, it seems that these 3 sources mostly the academy dataframe only had 98 movies within it that were also on the other two dataframes. This is still ok because I can see still get my visualizations I want to do done with this amount of data.

Next I want to check and see the average critic score and public score from the dataset

In [201]:
```
moviedf.to_csv('moviedf.csv')
```

In [215]:
```
moviedf = pd.read_csv('moviedf.csv')
moviedf.head(10)
```

Out[215]:

	Unnamed: 0	Film	Year	critic_score	public_rating	Awards_comp	Nominatic
0	0	toy story 4	2019	97	75.0	1	
1	1	once upon a time in hollywood	2019	85	74.0	2	
2	2	black panther	2018	96	74.0	3	
3	3	blackkklansman	2018	96	75.0	1	
4	4	spiderman into the spiderverse	2018	97	84.0	1	
5	5	dunkirk	2017	92	75.0	3	
6	6	three billboards outside ebbing missouri	2017	90	81.0	2	
7	7	blade runner 2049	2017	88	75.0	2	
8	8	coco	2017	97	82.0	2	
9	9	la la land	2016	90	79.0	6	

In [217]:
```
#del moviedf['Unnamed: 0']
```

In [218]:
```
print('Average Rating by Critics:', round(moviedf.critic_score.mean(),2
```

```
Average Rating by Critics: 95.84
Average Rating by Public: 76.85
```

Seems that for these awarded/nominated movies in my dataframe, the public on average rated them lower than the critics did. This could start the answer to what I'm questioning with this data: Do public or critic opinions determine if a movie is nominated/awarded?

Visualization 1:

In [220]:
```
top5movies = moviedf.sort_values(by=['Awards_comp'], ascending = False)
top5movies.head()
```

Out[220]:

	Film	Year	critic_score	public_rating	Awards_comp	Nominations_comp
85	on the waterfront	1954	99	80.0	8	12
75	gone with the wind	1939	91	80.0	8	13
79	lawrence of arabia	1962	97	80.0	7	10
70	the bridge on the river kwai	1957	95	78.0	7	8
69	the best years of our lives	1946	97	78.0	7	8

In [228]:

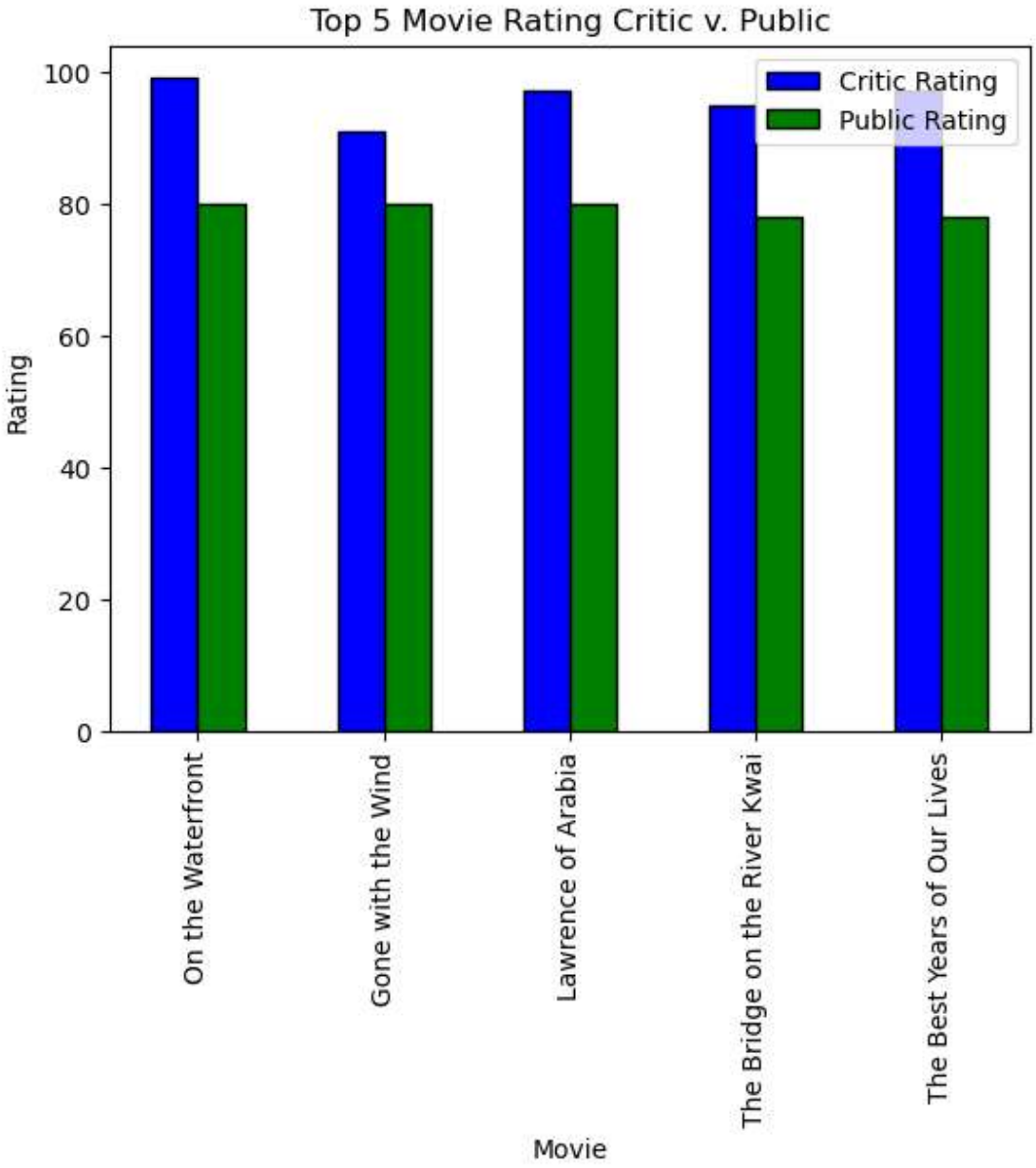
```
r = np.arange(len(top5movies.Film))
width = 0.25


plt.bar(r, top5movies.critic_score, color = 'b',
        width = width, edgecolor = 'black',
        label='Critic Rating')
plt.bar(r + width, top5movies.public_rating, color = 'g',
        width = width, edgecolor = 'black',
        label='Public Rating')

plt.xlabel("Movie")
plt.ylabel("Rating")
plt.title("Top 5 Movie Rating Critic v. Public")

# plt.grid(linestyle='--')
plt.xticks(r + width/2,['On the Waterfront','Gone with the Wind','Lawre
plt.legend()

plt.show()
```



This first visualization further proves the averages I previously calculated. Based on this visualization, the critic's score seems to have more of an impact on determining if the movie wins awards or not.

Visualization 2:

For this next visual, I want to take a step back from awards and take a closer look at the public ratings. I want to see the top 5 public rated films to see how the rest of the data in my dataset look.

In [229]:
```
top5public = moviedf.sort_values(by=['public_rating'], ascending = Fals
top5public.head()
```

Out[229]:

	Film	Year	critic_score	public_rating	Awards_comp	Nominations_comp
29	the dark knight	2008	94	85.0	2	8
52	the empire strikes back	1980	94	84.0	1	3
4	spiderman into the spiderverse	2018	97	84.0	1	1
38	the lord of the rings the two towers	2002	95	84.0	2	6
56	one flew over the cuckoos nest	1975	94	84.0	5	9

In [230]:
```
top5critic = moviedf.sort_values(by=['critic_score'], ascending = False
top5critic.head()
```

Out[230]:

	Film	Year	critic_score	public_rating	Awards_comp	Nominations_comp
48	henry v	1989	100	72.0	1	3
65	the adventures of robin hood	1938	100	75.0	3	4
42	toy story	1995	100	80.0	0	3
76	the grapes of wrath	1940	100	78.0	2	7
86	the philadelphia story	1940	100	77.0	2	6

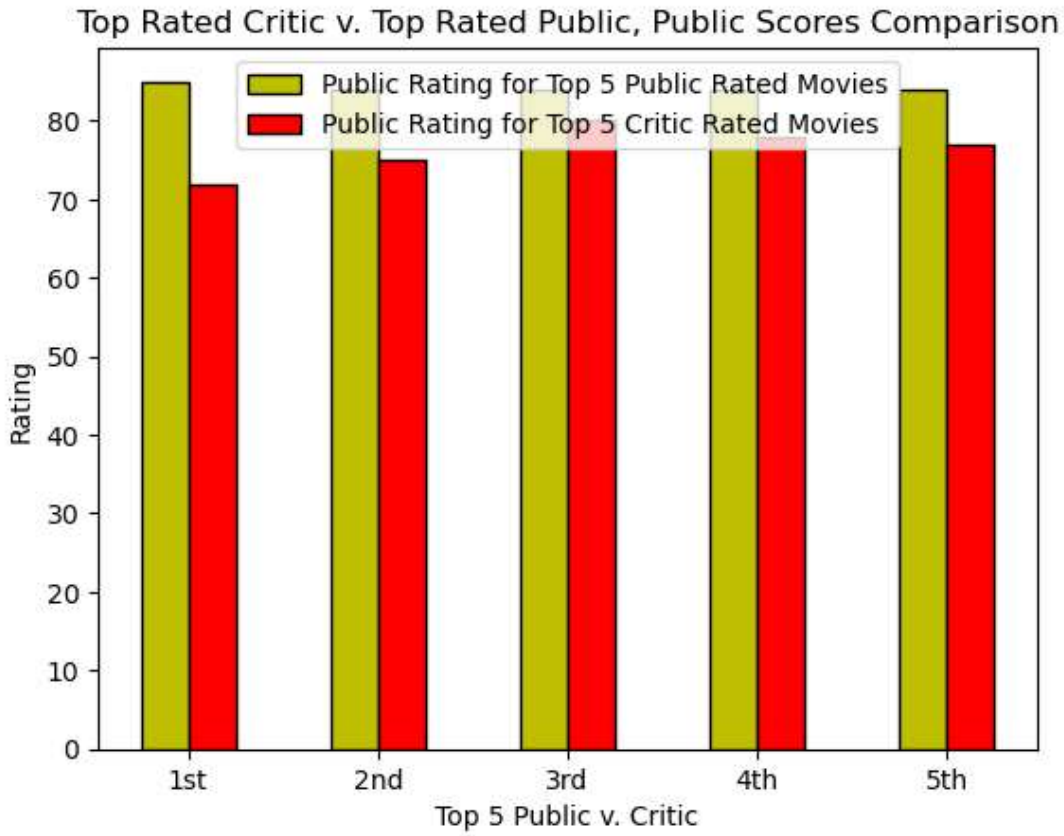
In [233]:

```
plt.bar(r, top5public.public_rating, color = 'y',
        width = width, edgecolor = 'black',
        label='Public Rating for Top 5 Public Rated Movies')
plt.bar(r + width, top5critic.public_rating, color = 'r',
        width = width, edgecolor = 'black',
        label='Public Rating for Top 5 Critic Rated Movies')

plt.xlabel("Top 5 Public v. Critic")
plt.ylabel("Rating")
plt.title("Top Rated Critic v. Top Rated Public, Public Scores Comparis

# plt.grid(linestyle='--')
plt.xticks(r + width/2,['1st','2nd','3rd','4th', '5th'])
plt.legend()

plt.show()
```

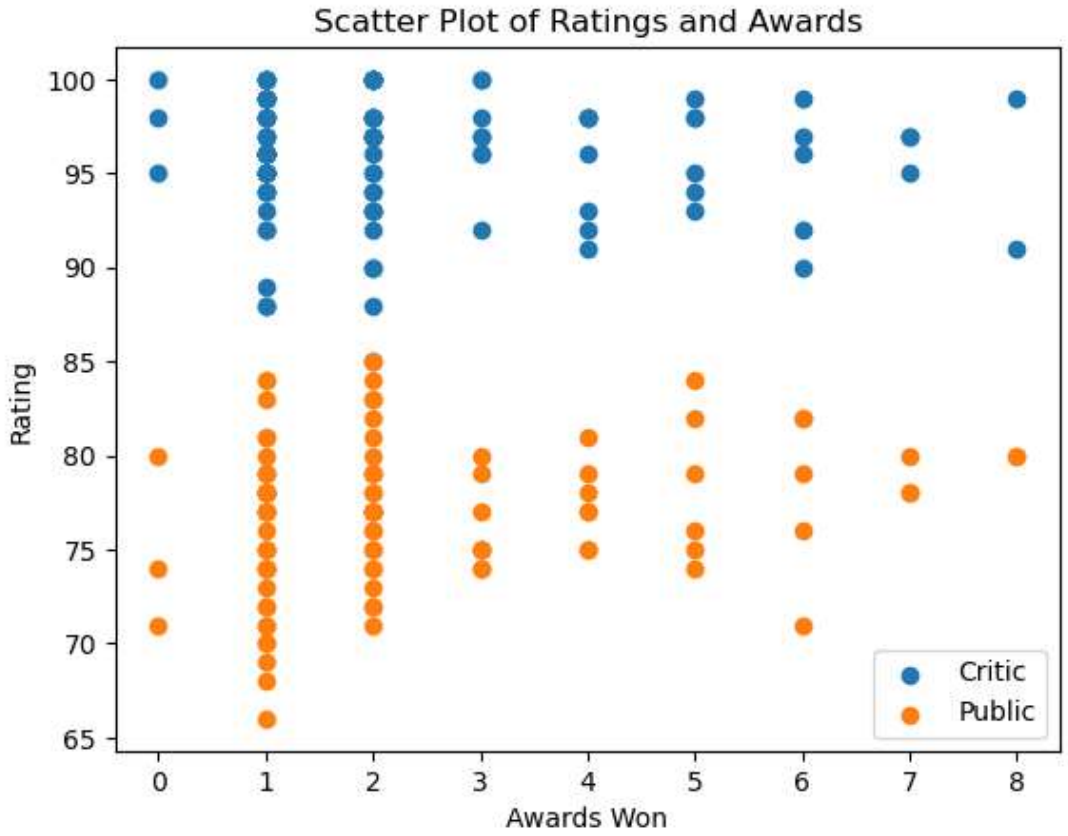


As I can see, the public rating has a higher score for different movies than what are the highest score on the critic rating's top list. Based on the two datasets I made with the top 5 rated from each, the public rating's top 5 brougt in more Oscar's than the top 5 on the critic scores! This is insightful as it could mean there is a possible correlation between the public rating and Oscar's won.

Visualization 3:

Next I want to visualize the correlation between crtic and public rating, and the number of awards won.

In [239]:
```
plt.scatter(moviedf.Awards_comp, moviedf.critic_score)
plt.scatter(moviedf.Awards_comp, moviedf.public_rating)
plt.legend(['Critic', 'Public'])
plt.xlabel("Awards Won")
plt.ylabel("Rating")
plt.title("Scatter Plot of Ratings and Awards")
plt.show()
```



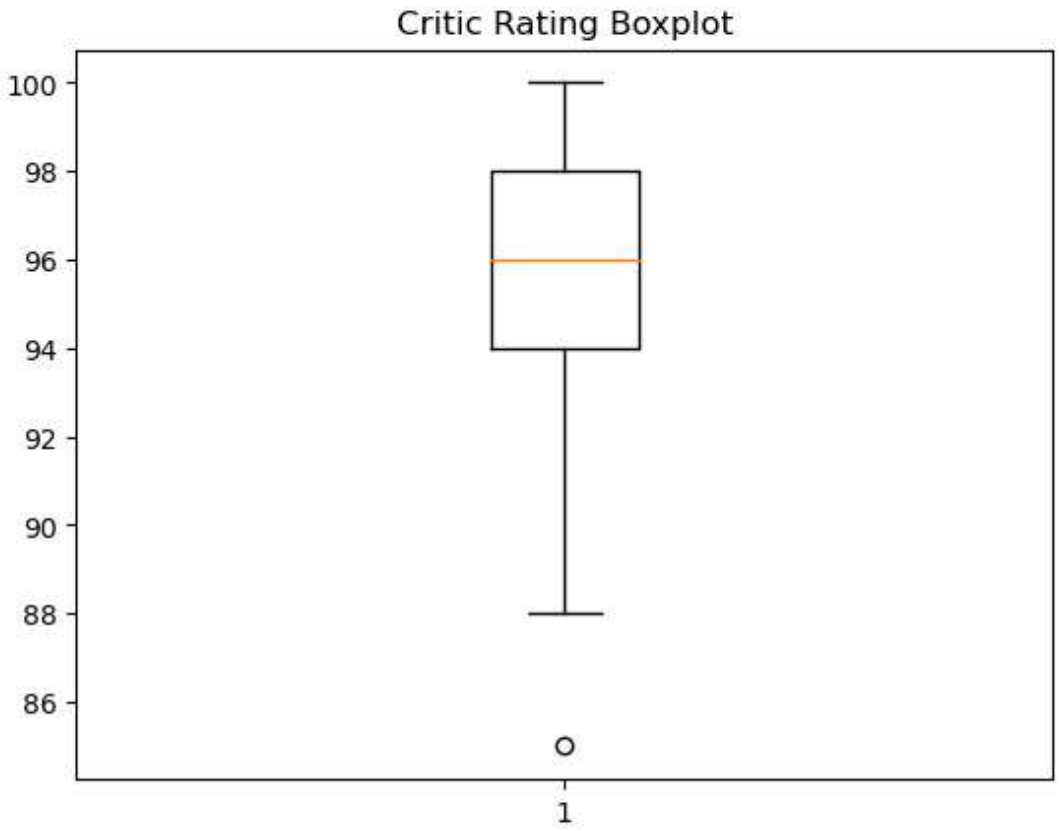
Looking at this visual, I can see that both scores 'seem' to be a 'good' determinate of if a movie wins 1 or 2 awards. Now, 1 or 2 awards has been what I've really wanted to focus on for this project. A movie getting 3, 4, or even 8 awards doesn't mean much to mean in this sense. Just winning 1 or 2 is meaningful enough. This visualization helps me see what I was aiming for, but doesn't mean that the variables are correlated with winning an award.

Visualization 4 and 5:

In the beginning of this project, I looked at some outliers that I decided to keep in the dataset and would address it when I joined the dataframes. I want to now visualize both the boxplot for the critic and public rating. These will be my last two visuals that will hope me see how much of the data is an outlier.

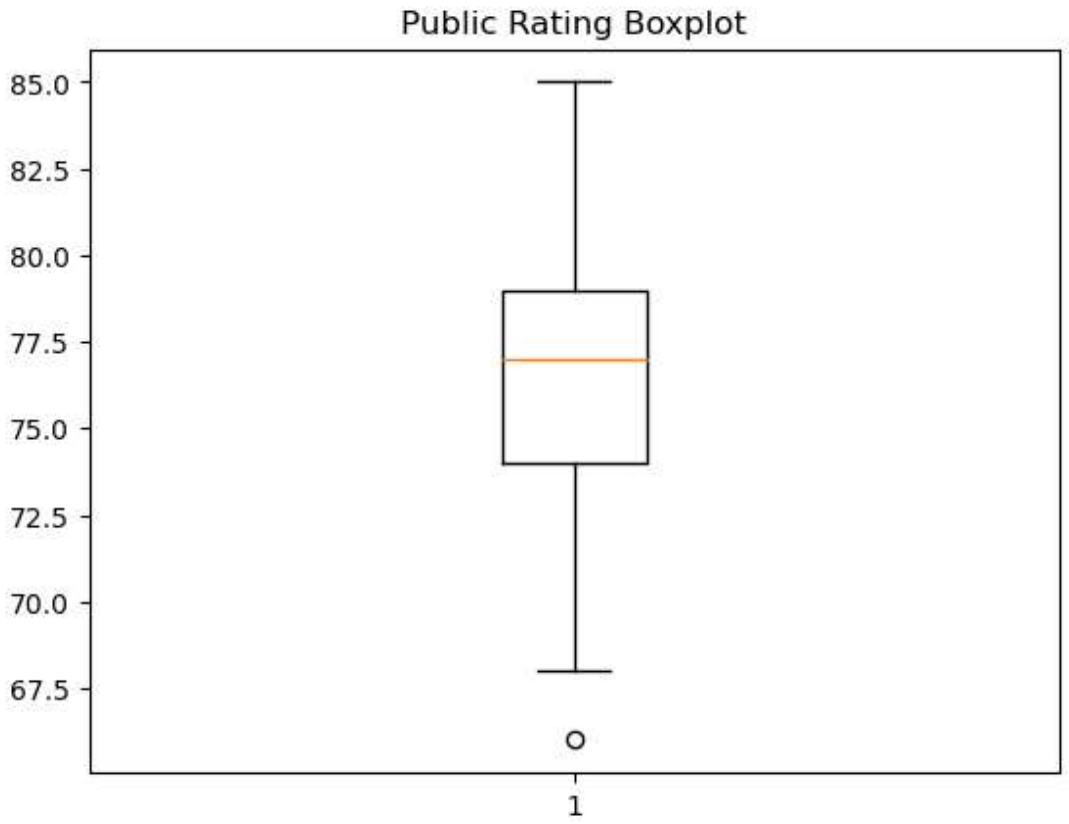
In [240]:
```
plt.boxplot(moviedf.critic_score)
plt.title('Critic Rating Boxplot')
```

Out[240]: Text(0.5, 1.0, 'Critic Rating Boxplot')



In [241]:
```
plt.boxplot(moviedf.public_rating)
plt.title('Public Rating Boxplot')
```

Out[241]: Text(0.5, 1.0, 'Public Rating Boxplot')



Both of these boxplots show 1 outlier each. This is interesting but also good because that means just by filtering/joining the data took care of a lot of my outliers. I'm sure the Awards column has some outliers with those movies that won 0 or 8 awards, but if I were to continue this analysis, I would most likely create a dummy variable that is binary and says if a movie won an award or not. So, the number of awards doesn't matter.

Conclusion.

To sum things up, this project has been extensive and spanned over a couple of weeks. I've learned a lot doing this project not only about the data I worked on, but mostly the Python (data wrangling, cleaning, and visualizing) that I practiced. My goal for this project was to visualize the possible relationship(s) between critic and public ratings with awarded movies. I wanted to see if the data could, visually, tell me which is better at determining if a movie won an Oscar; Public or Critic rating. I accomplished this, but also cannot officially determine if there is any answer yet. I would need to further analyze the data through some sort of classification model. This would be fun to try but wasn't my goal for the project.

This project taught me the whole process of data wrangling and capturing data from different source types. I used a .csv, website (URL), and an API to bring in all of this data. There were some challenges when doing so, the API took FOREVER and there were

probably some more efficient ways to code this, but it was a learning experience. My main takeaway from this is that I want to get more into building dataframes or even databases from APIs and do different types of analysis. I liked that I got to work on an interesting topic that I like, movies, and discover more on questions I have surrounding the rating of films and how it effects the awarding success of films. Another cool analysis would be to determine if ratings effect the box office revenue a movie gets. That is for another day haha, but something that I know based off these datasets, is it is doable.

In [ ]: