By: Jose Diaz, No teammates

Goal

To begin, the goal of this project is to find out neat and intersting facts about movie data. In the real world this could be used for anything ranging from simple social media posts for a film company or film product company to production companies looking for trends in how their films or how their directors are performing. My hope is to be able to provide interesting insights that without this data would be near impossible to find.

```
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

```
origDF = pd.read_csv('/content/rotten_tomatoes_movies.csv')
modelDF = origDF.copy()
```

```
modelDF.head()
```

	rotten_tomatoes_link	movie_title	movie_info	critics_consensus	content_
0	m/0814255	Percy Jackson & the Olympians: The Lightning T...	Always trouble-prone, the life of teenager Per...	Though it may seem like just another Harry Pot...	
1	m/0878835	Please Give	Kate (Catherine Keener) and her husband Alex (...	Nicole Holofcener's newest might seem slight i...	

```
modelDF.columns
```

```
Index(['rotten_tomatoes_link', 'movie_title', 'movie_info',
       'critics_consensus', 'content_rating', 'genres', 'directors', 'authors',
       'actors', 'original_release_date', 'streaming_release_date', 'runtime',
       'production_company', 'tomatometer_status', 'tomatometer_rating',
       'tomatometer_count', 'audience_status', 'audience_rating',
       'audience_count', 'tomatometer_top_critics_count',
       'tomatometer_fresh_critics_count', 'tomatometer_rotten_critics_count'],
      dtype='object')
```

Dropping columns that we do not need

- Most of the columns that are unrelated to the films must be dropped

```
modelDF.drop(['rotten_tomatoes_link', 'critics_consensus', 'authors', 'tomatometer_fre
```

	movie_title	movie_info	content_rating	genres	directors	actors
0	Percy Jackson & the Olympians: The Lightning T...	Always trouble-prone, the life of teenager Per...	PG	Action & Adventure, Comedy, Drama, Science Fic...	Chris Columbus	Logan Lerman, Brandon T. Jackson, Alexandra Da...
1	Please Give	Kate (Catherine Keener) and her husband Alex (...	R	Comedy	Nicole Holofcener	Catherine Keener, Amanda Peet, Oliver Platt, R...
2	10	A successful, middle-aged Hollywood songwriter...	R	Comedy, Romance	Blake Edwards	Dudley Moore, Bo Derek, Julie Andrews, Robert ...
3	12 Angry Men (Twelve Angry Men)	Following the closing arguments in a murder tr...	NR	Classics, Drama	Sidney Lumet	Martin Balsam, John Fiedler, Lee J. Cobb, E.G....
4	20,000 Leagues Under The Sea	In 1866, Professor Pierre M. Aronnax (Paul Luk...	G	Action & Adventure, Drama, Kids & Family	Richard Fleischer	James Mason, Kirk Douglas, Paul Lukas, Peter L...
...	...	...	...	...	...	...
17707	Zoot Suit	Mexican-American gangster Henry Reyna (Daniel ...	R	Drama, Musical & Performing Arts	Luis Valdez	Daniel Valdez, Edward James Olmos, Charles Aid...
17708	Zootopia	From the largest elephant to the smallest shre...	PG	Action & Adventure, Animation, Comedy	Byron Howard, Rich Moore, Jared Bush	J.K. Simmons, Kristen Bell, Octavia Spencer, A...
						Anthony Quinn

17709	Zorba the Greek	Traveling to inspect an abandoned mine his fat	NR	Action & Adventure, Art House & International	NaN	Quinn, Alan Bates, Irene

Checking to see what columns are missing data

- Some columns we can have blanks and they don't matter, like movie info
- In others we will get rid of the rows where we have blanks

colonial Drama Endfield Ulla

```
modelDF.isna().sum()
```

```
rotten_tomatoes_link                 0
movie_title                          0
movie_info                         321
critics_consensus                 8578
content_rating                       0
genres                              19
directors                          194
authors                           1542
actors                             352
original_release_date             1166
streaming_release_date             384
runtime                            314
production_company                 499
tomatometer_status                  44
tomatometer_rating                  44
tomatometer_count                   44
audience_status                    448
audience_rating                    296
audience_count                     297
tomatometer_top_critics_count        0
tomatometer_fresh_critics_count      0
tomatometer_rotten_critics_count     0
dtype: int64
```

```
modelDF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17712 entries, 0 to 17711
Data columns (total 22 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   rotten_tomatoes_link              17712 non-null  object
 1   movie_title                       17712 non-null  object
 2   movie_info                        17391 non-null  object
 3   critics_consensus                 9134 non-null   object
 4   content_rating                    17712 non-null  object
 5   genres                            17693 non-null  object
 6   directors                         17518 non-null  object
 7   authors                           16170 non-null  object
 8   actors                            17360 non-null  object
 9   original_release_date             16546 non-null  object
 10  streaming_release_date            17328 non-null  object
```

```
 11  runtime                           17398 non-null  float64
 12  production_company                17213 non-null  object
 13  tomatometer_status                17668 non-null  object
 14  tomatometer_rating                17668 non-null  float64
 15  tomatometer_count                 17668 non-null  float64
 16  audience_status                   17264 non-null  object
 17  audience_rating                   17416 non-null  float64
 18  audience_count                    17415 non-null  float64
 19  tomatometer_top_critics_count     17712 non-null  int64
 20  tomatometer_fresh_critics_count   17712 non-null  int64
 21  tomatometer_rotten_critics_count  17712 non-null  int64
dtypes: float64(5), int64(3), object(14)
memory usage: 3.0+ MB
```

```
modelDF.describe()
```

	runtime	tomatometer_rating	tomatometer_count	audience_rating	audie
count	17398.000000	17668.000000	17668.000000	17416.000000	1.7
mean	102.214048	60.884763	57.139801	60.554260	1.4
std	18.702511	28.443348	68.370047	20.543369	1.7
min	5.000000	0.000000	5.000000	0.000000	5.0
25%	90.000000	38.000000	12.000000	45.000000	7.0
50%	99.000000	67.000000	28.000000	63.000000	4.2
75%	111.000000	86.000000	75.000000	78.000000	2.4
max	266.000000	100.000000	574.000000	100.000000	3.5

```
modelDF[modelDF['runtime']<60]
```

	rotten_tomatoes_link	movie_title	movie_info	critics_
327	m/1002611-blood_feast	Blood Feast	In the sleepy suburbs of Miami, seemingly norm...	
452	m/1007949-frosty_the_snowman	Frosty the Snowman	A discarded magic top hat brings to life the s...	Frosty the a jolly
704	m/1017962-rudolph_the_rednosed_reindeer	Rudolph the Red-Nosed Reindeer	A reindeer with a glowing nose saves Christmas...	Rudo Nosed
1455	m/1136634-journey	Journey Into Amazing Caves	Scientists Nancy Aulenbach and Dr. Hazel Barto...	
1467	m/1141548-sacred_planet	Sacred Planet	Some of the wildest, most beautifully stunning...	
...	...	...	...	
16424	m/traffic_stop	Traffic Stop	A 26-year-old teacher from Austin, Texas, is v...	
16677	m/uncovered_the_whole_truth_about_the_iraq_war	Uncovered: The Whole Truth About the Iraq War	The Bush administration's decision to go to wa...	
[illegible]	[illegible]	Vikings:	Filmmaker Marc Fafard	

```
modelDF[modelDF['tomatometer_count']<20]
```

	rotten_tomatoes_link	movie_title	movie_info	critics_consensus	cont
8	m/10002008-charly	Charly (A Heartbeat Away)	Cultural differences, past loves and personal ...	NaN	
9	m/1000204-abraham_lincoln	Abraham Lincoln	The 16th U.S. president (Walter Huston) is por...	NaN	
10	m/10002114-dark_water	Dark Water	In this moody Japanese horror film, newly-sing...	NaN	
13	m/10002519-breaking_point	The Breaking Point	A charter-boat captain winds up in the middle ...	NaN	
16	m/10002673-prowler	The Prowler (Cost of Living)	After being frightened by a peeping Tom at her...	NaN	
...	...	...	...	...	
17701	m/zombies_of_mass_destruction	ZMD: Zombies of Mass Destruction	An Iranian college student (Janette Armand) an...	NaN	
17702	m/zoo_2018	Zoo	A 12-year old boy and his misfit friends enlis...	NaN	
17707	m/zoot_suit	Zoot Suit	Mexican-American gangster Henry Reyna (Daniel ...	NaN	

17709	m/zorba_the_greek	Zorba the Greek	Traveling to inspect an abandoned mine his fat...	NaN
17711	m/zulu_dawn	Zulu Dawn	Sir Henry Bartle Frere's (John Mills) vastly o...	NaN

6862 rows × 22 columns

```
modelDF[['tomatometer_count', 'audience_count']][modelDF['audience_count']<50]
```

	tomatometer_count	audience_count
217	26.0	24.0
316	10.0	7.0
563	12.0	45.0
886	9.0	6.0
1848	9.0	37.0
...	...	...
17531	NaN	6.0
17615	45.0	37.0
17619	76.0	20.0
17655	5.0	35.0
17697	28.0	27.0

565 rows × 2 columns

What to do with movies that have low user scores but high critic scores

- These would be films that are unknown to average users like us but favored by critics
- 'Unknown goodies'

```
modelDF.head()
```

	rotten_tomatoes_link	movie_title	movie_info	critics_consensus	content_r
0	m/0814255	Percy Jackson & the Olympians: The Lightning T...	Always trouble-prone, the life of teenager Per...	Though it may seem like just another Harry Pot...	
1	m/0878835	Please Give	Kate (Catherine Keener) and her husband Alex (...	Nicole Holofcener's newest might seem slight i...	
2	m/10	10	A successful, middle-aged Hollywood songwriter...	Blake Edwards' bawdy comedy may not score a pe...	
3	m/1000013-12_angry_men	12 Angry Men (Twelve Angry Men)	Following the closing arguments in a murder tr...	Sidney Lumet's feature debut is a superbly wri...	
4	m/1000079-20000_leagues_under_the_sea	20,000 Leagues Under The Sea	In 1866, Professor Pierre M. Aronnax	One of Disney's finest live-action adventures,...	

```
type(modelDF['genres'][0])
```

```
str
```

```
modelDF['genres'].size
```

```
17712
```

```
modelDF['genres'].isna().sum()
```

```
19
```

After exploring the data, the cleaning begins

```
modelDF.columns
```

```
Index(['rotten_tomatoes_link', 'movie_title', 'movie_info',
       'critics_consensus', 'content_rating', 'genres', 'directors', 'authors',
       'actors', 'original_release_date', 'streaming_release_date', 'runtime',
       'production_company', 'tomatometer_status', 'tomatometer_rating',
       'tomatometer_count', 'audience_status', 'audience_rating',
       'audience_count', 'tomatometer_top_critics_count',
       'tomatometer_fresh_critics_count', 'tomatometer_rotten_critics_count'],
      dtype='object')
```

```
rows_removed = modelDF.dropna()
```

```
rows_removed.isna().sum()
```

```
rotten_tomatoes_link                0
movie_title                         0
movie_info                          0
critics_consensus                   0
content_rating                      0
genres                              0
directors                           0
authors                             0
actors                              0
original_release_date               0
streaming_release_date              0
runtime                             0
production_company                  0
tomatometer_status                  0
tomatometer_rating                  0
tomatometer_count                   0
audience_status                     0
audience_rating                     0
audience_count                      0
tomatometer_top_critics_count       0
tomatometer_fresh_critics_count     0
tomatometer_rotten_critics_count    0
dtype: int64
```

```
rows_removed[['audience_count','tomatometer_count']]
```

	audience_count	tomatometer_count
0	254421.0	149.0
1	11574.0	142.0
2	14684.0	24.0
3	105386.0	54.0
4	68918.0	27.0

Here I have decided that only films who have atleast 50 audience count and 20 critic counts should be included in the dataset

- these numbers are arbitrary, but they are chosen to avoid scenarios where a film only has 1 rating of each and is rated incredibly high or incredibly low skewing the data

17708	101511.0	291.0

```
rows_removed[['audience_count','tomatometer_count']][rows_removed['audience_count']<5(
```

```
(56, 2)
```

```
rows_removed[['audience_count','tomatometer_count']][rows_removed['tomatometer_count']
```

```
(47, 2)
```

need these values to be removed ^^^^^^

```
rows_removed[['audience_count','tomatometer_count']].loc[rows_removed['audience_count'
```

	audience_count	tomatometer_count
2253	29.0	67.0
2330	39.0	23.0
2890	29.0	92.0
2968	40.0	52.0
4130	37.0	86.0
4146	34.0	36.0
4414	22.0	43.0
4526	34.0	28.0
4657	40.0	31.0
5534	25.0	44.0
6012	9.0	24.0
6103	34.0	26.0
6178	6.0	26.0
6664	16.0	40.0
6799	32.0	86.0
7043	34.0	50.0
7072	42.0	26.0
7356	6.0	64.0
7939	45.0	43.0
8074	37.0	72.0
8285	42.0	39.0
9077	38.0	73.0
9752	22.0	28.0
10083	47.0	52.0
10284	48.0	41.0
10370	44.0	71.0
10698	33.0	31.0
10699	33.0	36.0
11308	31.0	47.0
11318	13.0	31.0

11491	42.0	66.0
11498	5.0	33.0
11858	20.0	33.0
11965	47.0	43.0
12658	44.0	41.0
13507	14.0	28.0
13967	12.0	24.0
14166	30.0	28.0
14468	12.0	27.0
14563	36.0	28.0
14625	26.0	36.0
14688	47.0	27.0
15225	6.0	31.0
15285	40.0	28.0
15580	41.0	17.0

```
rows_removed[['audience_count','tomatometer_count']].loc[rows_removed['tomatometer_cou
```

	audience_count	tomatometer_count
2	14684.0	24.0
4	68918.0	27.0
14	10563.0	28.0
15	1935.0	24.0
20	33946.0	48.0
...	...	...
17688	1628.0	20.0
17693	466.0	30.0
17695	3657.0	36.0
17696	20323.0	26.0
17710	30193.0	23.0

2602 rows × 2 columns

```
rows_removed.drop(
rows_removed.loc[rows_removed['tomatometer_count']<50].index, inplace=True)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4174: SettingWithCop

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
```

```
rows_removed.shape
```

```
(5475, 22)
```

```
rows_removed.drop(rows_removed.loc[rows_removed['audience_count']<50].index, inplace=T
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4174: SettingWithCop

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
```

```
rows_removed.shape
```

```
(5458, 22)
```

```
rows_removed.head()
```

	rotten_tomatoes_link	movie_title	movie_info	critics_consensus	content_rati
0	m/0814255	Percy Jackson & the Olympians: The Lightning T...	Always trouble-prone, the life of teenager Per...	Though it may seem like just another Harry Pot...	
1	m/0878835	Please Give	Kate (Catherine Keener) and her husband Alex (...	Nicole Holofcener's newest might seem slight i...	

```
rows_removed.columns
```

```
Index(['rotten_tomatoes_link', 'movie_title', 'movie_info',
       'critics_consensus', 'content_rating', 'genres', 'directors', 'authors',
       'actors', 'original_release_date', 'streaming_release_date', 'runtime',
       'production_company', 'tomatometer_status', 'tomatometer_rating',
       'tomatometer_count', 'audience_status', 'audience_rating',
       'audience_count', 'tomatometer_top_critics_count',
       'tomatometer_fresh_critics_count', 'tomatometer_rotten_critics_count'],
      dtype='object')
```

```
rows_removed['runtime'].loc[rows_removed['runtime']<60]
```

```
Series([], Name: runtime, dtype: float64)
```

Looks like nothing is less than 60 minutes runtime

- Here I wanted to make sure that we are only including films that were actually films
- Rotten tomatoes also has data on things such as TV shows and short films which for this discussion will be left out of the equation

```
data = rows_removed.drop(columns=['rotten_tomatoes_link', 'movie_info', 'critics_conse
```

```
data
```

	movie_title	content_rating	genres	directors	actors	original_releas
0	Percy Jackson & the Olympians: The Lightning T...	PG	Action & Adventure, Comedy, Drama, Science Fic...	Chris Columbus	Logan Lerman, Brandon T. Jackson, Alexandra Da...	20
1	Please Give	R	Comedy	Nicole Holofcener	Catherine Keener, Amanda Peet, Oliver Platt, R...	20
3	12 Angry Men (Twelve Angry Men)	NR	Classics, Drama	Sidney Lumet	Martin Balsam, John Fiedler, Lee J. Cobb, E.G....	19
5	10,000 B.C.	PG-13	Action & Adventure, Classics, Drama	Roland Emmerich	Steven Strait, Camilla Belle, Cliff Curtis,	20

This is my final clean dataset and is what I will use as my starting data for all of my visualizations

Mystery &

```
data.to_csv('clean_data.csv')
```

```
new = pd.read_csv('clean_data.csv')
```

```
new = new.drop(columns=['Unnamed: 0'])
```

```
new
```

	movie_title	content_rating	genres	directors	actors	original_release
0	Percy Jackson & the Olympians: The Lightning T...	PG	Action & Adventure, Comedy, Drama, Science Fic...	Chris Columbus	Logan Lerman, Brandon T. Jackson, Alexandra Da...	201
1	Please Give	R	Comedy	Nicole Holofcener	Catherine Keener, Amanda Peet, Oliver Platt, R...	201
2	12 Angry Men (Twelve Angry Men)	NR	Classics, Drama	Sidney Lumet	Martin Balsam, John Fiedler, Lee J. Cobb, E.G....	195
3	10,000 B.C.	PG-13	Action & Adventure,	Roland	Steven Strait, Camilla	200

```
new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5458 entries, 0 to 5457
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   movie_title             5458 non-null   object 
 1   content_rating          5458 non-null   object 
 2   genres                  5458 non-null   object 
 3   directors               5458 non-null   object 
 4   actors                  5458 non-null   object 
 5   original_release_date   5458 non-null   object 
 6   streaming_release_date  5458 non-null   object 
 7   runtime                 5458 non-null   float64
 8   production_company      5458 non-null   object 
 9   tomatometer_status      5458 non-null   object 
 10  tomatometer_rating      5458 non-null   float64
 11  tomatometer_count       5458 non-null   float64
 12  audience_status         5458 non-null   object 
 13  audience_rating         5458 non-null   float64
 14  audience_count          5458 non-null   float64
dtypes: float64(5), object(10)
memory usage: 639.7+ KB
```

At this point in the project I came back and decided that it would be better to fix some things

- Specifically I decided to split up the 'genres' and 'actors' column since they contained strings with several genres/actors.
- I planned to possibly use these values for my visualizations but had trouble figuring out the best way to use these columns that held lists.
- Currently working with these columns proved to be very difficult, but in the future these columns could give lots of insight into who are the most popular actors and other related insights.

```
new['genres'][0].split(', ')
```

```
['Action & Adventure', 'Comedy', 'Drama', 'Science Fiction & Fantasy']
```

```
new.genres = new.genres.apply(lambda s: s.split(', '))
```

```
new['genres']
```

```
0       [Action & Adventure, Comedy, Drama, Science Fi...
1                                                [Comedy]
2                                      [Classics, Drama]
3                  [Action & Adventure, Classics, Drama]
4       [Action & Adventure, Classics, Mystery & Suspe...
                              ...
5453                                   [Comedy, Romance]
5454                          [Comedy, Special Interest]
5455                                            [Comedy]
5456          [Action & Adventure, Comedy, Kids & Family]
5457              [Action & Adventure, Animation, Comedy]
Name: genres, Length: 5458, dtype: object
```

```
new.head()
```

```
new.actors = new.actors.apply(lambda s: s.split(', '))
```

[Action & Logan

Visualizations Begin Here

Science Alexandra

```
new[['runtime', 'original_release_date']]
```

	runtime	original_release_date
0	119.0	2010-02-12
1	90.0	2010-04-30
2	95.0	1957-04-13
3	109.0	2008-03-07
4	80.0	1935-08-01
...	...	...
5453	101.0	2011-07-08
5454	89.0	2001-09-28
5455	102.0	2016-02-12
5456	88.0	2006-08-11
5457	108.0	2016-03-04

5458 rows × 2 columns

```
type(new['original_release_date'][0])
```

```
str
```

```
new['original_release_date'] = pd.to_datetime(new['original_release_date'])
```

```
new.head()
```

	movie_title	content_rating	genres	directors	actors	original_release_da
0	Percy Jackson & the Olympians:	PG	[Action & Adventure, Comedy, Fl...	Chris	[Logan Lerman, Brandon T... D	2010-02

```
new['year'] = pd.DatetimeIndex(new['original_release_date']).year
```

- In this dataset we have two ratings.
- 'tomatometer_critics' are ratings that come from actual movie critics. Individuals who rate movies for publications/for a living.
- 'audience_rating' are ratings from everyday users that do not rate movies for a living.
- To be able to better visualize I will make a new 'rating' column which will just be the average of the two ratings.
- From here on forward, any visualization comparing against 'rating' will be comparing against the average of the two rating systems unless otherwise stated.

```
new['rating'] = (new["tomatometer_rating"] + new["audience_rating"]) / 2
```

```
runtimeYear = new[['runtime', 'year']]
```

```
runtimeYear
```

	runtime	year
0	119.0	2010
1	90.0	2010
2	95.0	1957
3	109.0	2008
4	80.0	1935
...	...	...
5453	101.0	2011
5454	89.0	2001
5455	102.0	2016
5456	88.0	2006
5457	108.0	2016

5458 rows × 2 columns

```
runtimeYearFinal = runtimeYear.groupby('year')['runtime'].mean().to_frame().reset_inde
```

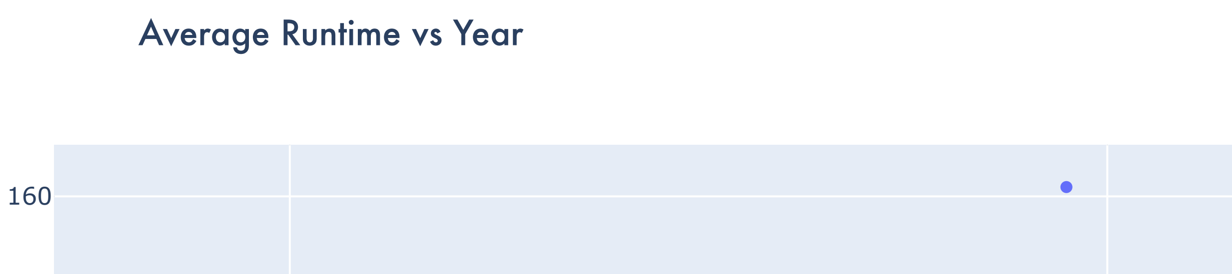
```
runtimeYearFinal = runtimeYear.groupby('year')['runtime'].mean().to_frame().reset_inde

runtimeYearFinal
```

	year	runtime
0	1920	69.000000
1	1922	65.000000
2	1925	83.000000
3	1927	105.333333
4	1928	77.000000
...	...	...
87	2016	107.573222
88	2017	109.273810
89	2018	110.557447
90	2019	108.403509
91	2020	102.230769

92 rows × 2 columns

```
runYearFig = px.scatter(runtimeYearFinal, x='year', y='runtime', title='Average Runtin
runYearFig.update_layout(title_font_family="Futura")
runYearFig.show()
# runYearFig.write_image('finalFigs/Average Runtime vs Year.png')
```



Average Runtime vs Year

- For this first figure the first thing we see was that for many years the lengths of films had been increasing
- This started to level out in 1980 where the average runtime of films begins to drop.
- From https://www.statisticshowto.com/lowess-smoothing/, "LOWESS (Locally Weighted Scatterplot Smoothing), sometimes called LOESS (locally weighted smoothing), is a popular tool used in regression analysis that creates a smooth line through a timeplot or scatter plot to help you to see relationship between variables and foresee trends."
- The trend is clear to see even without the LOWESS trendline and in this case matches our data perfectly

Notable Film:

- In 1939, the epic film 'Gone With The Wind' was released and had a runtime of 222.0 minutes making it one of the longest films in history
- This length also influenced greatly the average for 1939 runtime of films as you can see above
- In this dataset there are only 4 other films that have a longer runtime than Gone With the Wind

```
new[['movie_title', 'runtime', 'year']].loc[new['runtime'] >= 222].sort_values(by='ru
```

	movie_title	runtime	year
2121	Gone With the Wind	222.0	1939
2100	Gods and Generals	223.0	2003
2735	Lagaan: Once Upon a Time in India	223.0	2001
188	Hamlet	242.0	1996
3266	Mysteries of Lisbon	266.0	2011

```
ratingRuntime = new[['rating', 'runtime']]
```

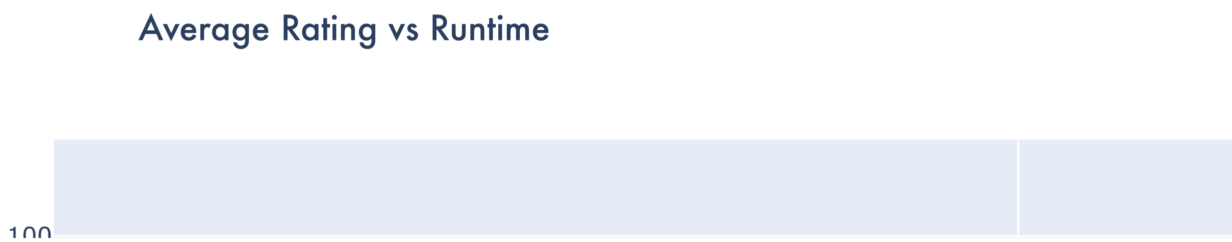
```
ratingRuntime
```

	rating	runtime
0	51.0	119.0
1	75.5	90.0
2	98.5	95.0
3	22.5	109.0
4	91.0	80.0
...	...	...
5453	27.5	101.0
5454	72.0	89.0
5455	21.0	102.0
5456	18.5	88.0
5457	95.0	108.0

5458 rows × 2 columns

```
ratingRunFinal = ratingRuntime.groupby('runtime')['rating'].mean().to_frame().reset_in
```

```
ratingRunFig = px.scatter(ratingRunFinal, x='runtime', y='rating', trendline='lowess',
ratingRunFig.update_layout(title_font_family='Futura')
ratingRunFig.show()
# ratingRunFig.write_image('finalFigs/Average Rating vs Runtime.png')
```



Average Rating vs Runtime

- From this graph we can see that as runtime is increased, generally films tend to have better ratings.
- This could be due to longer films having more time to better form, develop, and go through a whole plot.
- At the beginning of this figure we have films that are "short", just barely over an hour, generally these films are older.
- Since these films are older, their high ratings are most likely attributed to their age and not their runtime

```
new.columns
```

```
Index(['movie_title', 'content_rating', 'genres', 'directors', 'actors',
       'original_release_date', 'streaming_release_date', 'runtime',
       'production_company', 'tomatometer_status', 'tomatometer_rating',
       'tomatometer_count', 'audience_status', 'audience_rating',
       'audience_count', 'year', 'rating'],
      dtype='object')
```

- Here I was trying to grab the top companies and have their average ratings to use for my next visualization
- It is not the cleanest way to achieve my goal, but it got the job done

```
top50 = new[['movie_title', 'directors', 'year', 'runtime', 'production_company', 'ton
```

```
top50
```

	movie_title	directors	year	runtime	production_company	tomatometer_rati
0	Percy Jackson & the Olympians: The Lightning T...	Chris Columbus	2010	119.0	20th Century Fox	49
1	Please Give	Nicole Holofcener	2010	90.0	Sony Pictures Classics	87
2	12 Angry Men (Twelve Angry Men)	Sidney Lumet	1957	95.0	Criterion Collection	100

```
topProducers_counts_ratings = top50['production_company'].value_counts().to_frame().re
```

```
topProducers_counts_ratings
```

	index	production_company
0	Warner Bros. Pictures	348
1	20th Century Fox	311
2	Universal Pictures	307
3	Paramount Pictures	253
4	Sony Pictures Classics	207
...	...	...
717	Lionsgate/Summit	1
718	Magnolia Picutres	1
719	Piki Films	1
720	Cult Epics	1
721	Rogue Pictures/Universal Studios	1

722 rows × 2 columns

```
topProducers_counts_ratings = topProducers_counts_ratings.sort_values(by='index')
```

```
topProducers_counts_ratings
```

	index	production_company
386	1091	1
1	20th Century Fox	311
199	20th Century Fox Distribution	3
471	20th Century Fox Film	1
427	20th Century Fox/Regency Films	1
...	...	...
64	Zeitgeist Films	14
383	Zenith International Films	1
520	Zodiac Pictures	1

```
avg_ratings_prod = top50.groupby('production_company')['rating'].mean().to_frame().res
```

```
avg_ratings_prod
```

	production_company	rating
0	1091	52.000000
1	20th Century Fox	54.419614
2	20th Century Fox Distribution	59.666667
3	20th Century Fox Film	31.000000
4	20th Century Fox/Regency Films	47.000000
...	...	...
717	Zeitgeist Films	84.321429
718	Zenith International Films	97.000000
719	Zodiac Pictures	73.000000
720	levelFILM	82.000000
721	s	85.000000

722 rows × 2 columns

```
topProducers_counts_ratings = topProducers_counts_ratings.reset_index().drop(columns=[
```

```
topProducers_counts_ratings
```

	index	production_company
0	1091	1
1	20th Century Fox	311
2	20th Century Fox Distribution	3
3	20th Century Fox Film	1
4	20th Century Fox/Regency Films	1
...	...	...
717	Zeitgeist Films	14
718	Zenith International Films	1
719	Zodiac Pictures	1
720	levelFILM	1
721	s	1

```
avg_ratings_prod
```

	production_company	rating
0	1091	52.000000
1	20th Century Fox	54.419614
2	20th Century Fox Distribution	59.666667
3	20th Century Fox Film	31.000000
4	20th Century Fox/Regency Films	47.000000
...	...	...
717	Zeitgeist Films	84.321429
718	Zenith International Films	97.000000
719	Zodiac Pictures	73.000000
720	levelFILM	82.000000
721	s	85.000000

722 rows × 2 columns

```
topProducers_counts_ratings = topProducers_counts_ratings.join(avg_ratings_prod['ratin
```

```
topProducers_counts_ratings
```

	index	production_company	rating
0	1091	1	52.000000
1	20th Century Fox	311	54.419614
2	20th Century Fox Distribution	3	59.666667
3	20th Century Fox Film	1	31.000000
4	20th Century Fox/Regency Films	1	47.000000
...	...	...	...
717	Zeitgeist Films	14	84.321429
718	Zenith International Films	1	97.000000
719	Zodiac Pictures	1	73.000000
720	levelFILM	1	82.000000
721	s	1	85.000000

722 rows × 3 columns

- Now I had what I wanted

```
topProducers_counts_ratings = topProducers_counts_ratings.rename(columns={'index':'pro
```

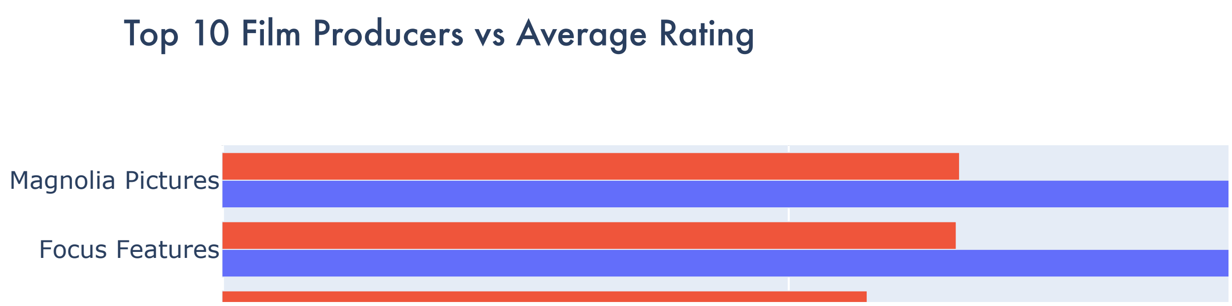
```
topProducers_counts_ratings.sort_values(by='films_produced', ascending=False).head(25)
```

	production_company	films_produced	average_rating
682	Warner Bros. Pictures	348	56.952586
1	20th Century Fox	311	54.419614
651	Universal Pictures	307	58.573290
442	Paramount Pictures	253	59.913043
556	Sony Pictures Classics	207	72.857488
265	IFC Films	169	68.035503
553	Sony Pictures	140	54.264286
117	Columbia Pictures	122	56.922131
210	Focus Features	118	64.779661
365	Magnolia Pictures	118	65.072034
384	Miramax Films	116	67.599138
401	New Line Cinema	110	56.700000

```
y = topProducers_counts_ratings.sort_values(by='films_produced', ascending=False).heac

tenProd_avgRating = go.Figure(data=[
    go.Bar(name='Films Produced', x=topProducers_counts_ratings.sort_values(by='films_
    go.Bar(name='Average Rating', x=topProducers_counts_ratings.sort_values(by='films_
])

tenProd_avgRating.update_layout(barmode='group', title='Top 10 Film Producers vs Avera
tenProd_avgRating.show()
# tenProd_avgRating.write_image('finalFigs/Top 10 Film Producers vs Average Rating.pnç
```



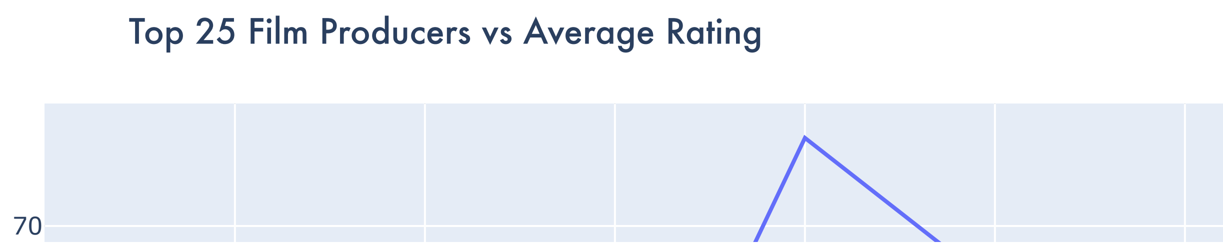
Top 10 Film Producers vs Average Rating

- From this figure we see that the biggest producers of films on average produce films that are decent at best.
- These production companies produce an IMMENSE amount of films but neither of the top 10 produce films that are substantially better than the next.
- The best rated movies on average come from Sony Pictures Classics, as the name suggests these films are older and thus are more highly rated

---

- To build on this, lets look at the top 25 and their ratings

```
top25_rating_line = px.line(topProducers_counts_ratings.sort_values(by='films_producec
top25_rating_line.update_layout(title='Top 25 Film Producers vs Average Rating', title
top25_rating_line.show()
# top25_rating_line.write_image('finalFigs/Top 25 Film Producers vs Average Rating.pnc
```



Top 25 Film Producers vs Average Rating

- From this visualization of the top 25 film producers, again we see that Sony Pictures Classics is the production company with the highest ratings
- The second highest rated production company is Fox Searchlight. According to https://en.wikipedia.org/wiki/Searchlight_Pictures, "The studio has grossed over $5.3 billion worldwide and amassed 26 Golden Globe Awards, 47 BAFTA awards, and 43 Academy Awards."
- With all of these award winning films, it makes sense that Searchlight would have a high Average rating of all of their films

```
topProducers_counts_ratings['production_company']
```

```
0                               1091
1                   20th Century Fox
2       20th Century Fox Distribution
3              20th Century Fox Film
4      20th Century Fox/Regency Films
                        ...
717                  Zeitgeist Films
718       Zenith International Films
719                  Zodiac Pictures
720                        levelFILM
721                                s
Name: production_company, Length: 722, dtype: object
```

- With this code, again not the best, I wanted to get the total number of films made per 'top x' company

```
topProducers_counts_ratings['films_produced'].sum()
```

```
5458
```

```
numFilms = []
```

```
numFilms.append(topProducers_counts_ratings.sort_values(by='films_produced', ascending
```

```
numFilms.append(topProducers_counts_ratings.sort_values(by='films_produced', ascending

numFilms.append(topProducers_counts_ratings.sort_values(by='films_produced', ascending

numFilms.append(topProducers_counts_ratings.sort_values(by='films_produced', ascending

numFilms.append(topProducers_counts_ratings.sort_values(by='films_produced', ascending

numFilms.append(topProducers_counts_ratings.sort_values(by='films_produced', ascending
```

```
numFilms
```

```
[1426, 667, 1026, 678, 597, 1064]
```

```
len(new['production_company'].unique())
```

```
722
```

```
films = ['','','','','','']
text = ['Top 5', 'Top 10', 'Top 25', 'Top 50', 'Top 100', 'All 722']
```

```
values = [1426, 2093, 3119, 3797, 4394, 5458]
```

```
values
```

```
[1426, 2093, 3119, 3797, 4394, 5458]
```

```
total = 5458
```

```
values2 = [x / total*100 for x in values]
```

```
values3 = [round(num, 3) for num in values2]
```

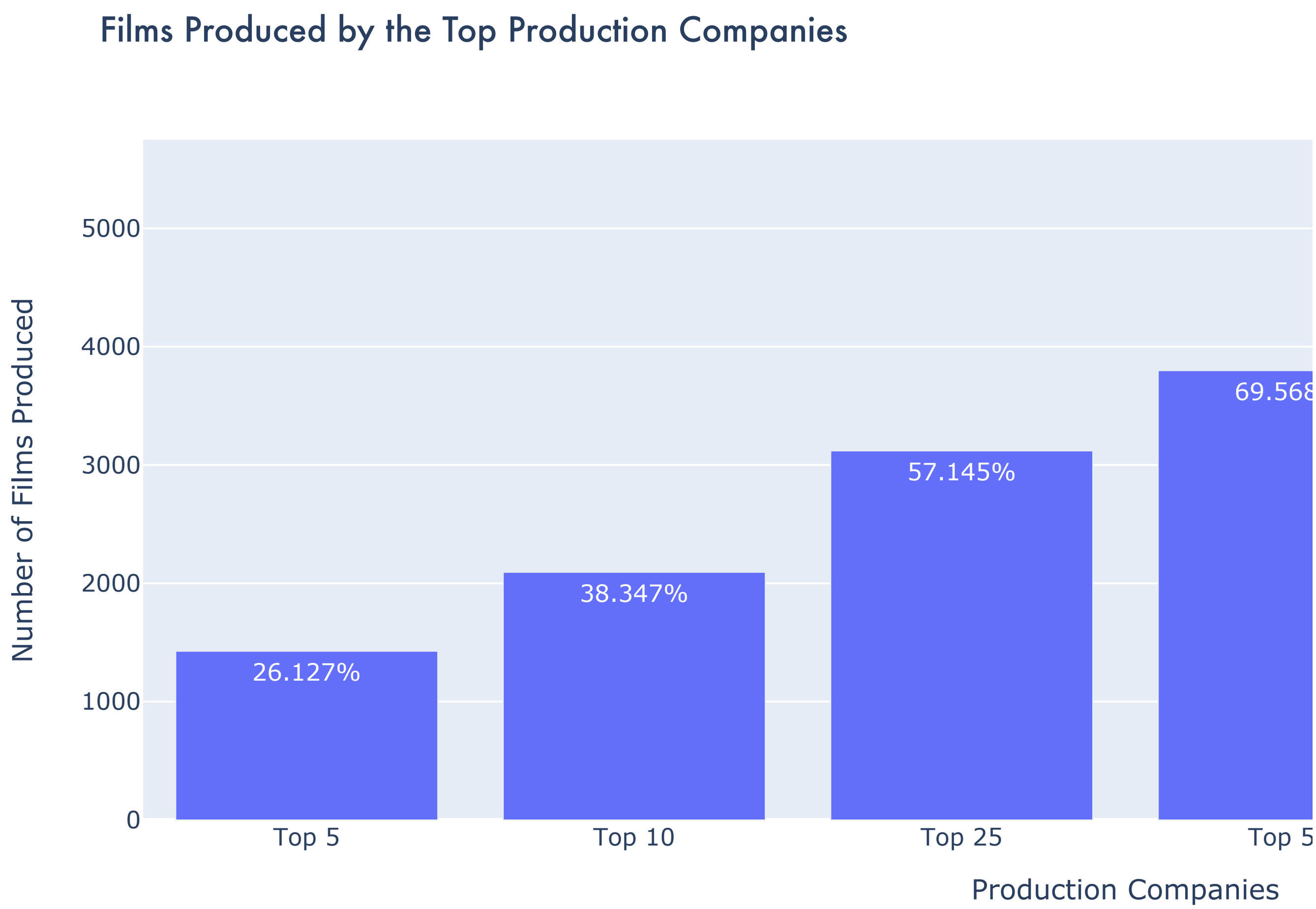
```
values3
```

```
[26.127, 38.347, 57.145, 69.568, 80.506, 100.0]
```

```
values4 = [str(x) + '%' for x in values3]
```

```
marketShare = px.bar(x=text, y=values, title='Films Produced by the Top Production Com
marketShare.update_layout(xaxis_title='Production Companies', yaxis_title='Number of F
marketShare.show()
```

```
# marketShare.write_image('finalFigs/Films Produced by Top Production Companies.png')
```



Films Produced by the Top Production Companies

- For this figure I wanted to look at how many films came from the largest film production companies.
- The most surprising fact is that the top 5 production companies have produced a whopping 26.127% of the films that make up this dataset.
- One fourth of all the films in this dataset come from these 5 companies.
- Building from that we see that the top 25 companies produce more than 50% of all the films in this dataset.
- In this dataset there are a total of 722 production companies, of all of these the top 100 made 80.506% of all the films in this dataset. The reach and the scale of these production companies is incredible and with this visualization we are better able to see how and why these companies are so massive

- Looking to get ratings per year

```
avgRating_year = new.groupby('year')[['tomatometer_rating', 'audience_rating','rating'
```

```
avgRating_year
```

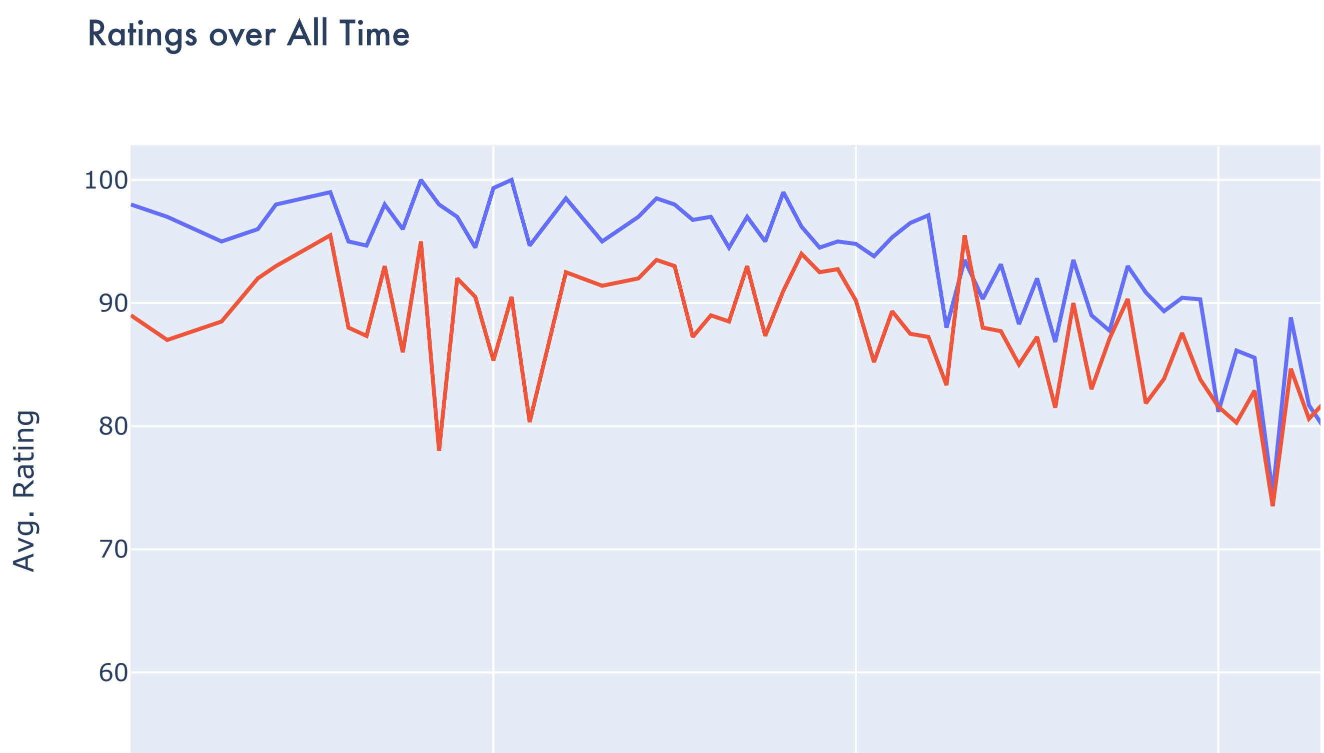
	year	tomatometer_rating	audience_rating	rating
0	1920	98.000000	89.000000	93.500000
1	1922	97.000000	87.000000	92.000000
2	1925	95.000000	88.500000	91.750000
3	1927	96.000000	92.000000	94.000000
4	1928	98.000000	93.000000	95.500000
...	...	...	...	...
87	2016	65.326360	60.121339	62.723849
88	2017	64.686508	61.297619	62.992063
89	2018	67.382979	61.306383	64.344681
90	2019	66.017544	59.578947	62.798246
91	2020	75.480769	58.307692	66.894231

92 rows × 4 columns

```
x = avgRating_year['year']
y = avgRating_year['rating']
y1 = avgRating_year['tomatometer_rating']
y2 = avgRating_year['audience_rating']

ratings_year = go.Figure()
# ratings_year.add_trace(go.Scatter(x=x, y=y, name='Average Rating'))
ratings_year.add_trace(go.Scatter(x=x, y=y1, name='Critic Rating', mode='lines'))
ratings_year.add_trace(go.Scatter(x=x, y=y2, name='Audience Rating', mode='lines'))

ratings_year.update_layout(xaxis_title='Year', yaxis_title='Avg. Rating', title='Ratin
ratings_year.show()
# ratings_year.write_image("finalFigs/Ratings over All Time.png")
```



Ratings over All Time

- Looking at this figure it looks like for the most part, audience scores tend to be lower than critic scores except for a period in the 2000s where audience scores seem to be much higher than critic scores.
- This is strange given that it seems to come out of nowhere, this gap between critics and audience scores could be contributed to an influx of internet users. This includes movie fanatics finding Rotten Tomatoes and leaving favorable reviews on their favorite films.
- This visualization is flawed given that very few films were made before 1980 and even less have reviews from critics or audience members.
- To combat this we will make another visualization with more significant years in film history.

```
avgRating_year.loc[avgRating_year['year'] > 1979]
```

	year	tomatometer_rating	audience_rating	rating
51	1980	81.166667	81.583333	81.375000
52	1981	86.142857	80.285714	83.214286
53	1982	85.555556	82.888889	84.222222
54	1983	74.500000	73.500000	74.000000
55	1984	88.833333	84.666667	86.750000
56	1985	81.700000	80.600000	81.150000
57	1986	79.500000	82.166667	80.833333
58	1987	83.450000	82.300000	82.875000
59	1988	88.562500	85.312500	86.937500
60	1989	85.200000	83.600000	84.400000
61	1990	72.636364	74.409091	73.522727
62	1991	75.117647	80.176471	77.647059
63	1992	75.920000	76.680000	76.300000
64	1993	80.560000	79.880000	80.220000
65	1994	78.500000	78.916667	78.708333
66	1995	74.114286	73.314286	73.714286
67	1996	73.355556	73.488889	73.422222
68	1997	72.084746	70.220339	71.152542
69	1998	59.662921	64.595506	62.129213
70	1999	54.167939	63.511450	58.839695
71	2000	49.522013	57.943396	53.732704
72	2001	53.220000	64.240000	58.730000
73	2002	54.904977	61.380090	58.142534
74	2003	52.158140	61.232558	56.695349
75	2004	52.779817	63.573394	58.176606
76	2005	53.084034	63.554622	58.319328
77	2006	53.352727	64.105455	58.729091
78	2007	56.519164	64.658537	60.588850
79	2008	53.913420	58.376623	56.145022
80	2009	56.217021	57.646809	56.931915

81	2010	59.025000	58.604167	58.814583
82	2011	59.632479	59.769231	59.700855
83	2012	61.162393	60.448718	60.805556
84	2013	61.262357	58.954373	60.108365
85	2014	63.140684	60.452471	61.796578

```
moviesPerYear = new['year'].value_counts().to_frame().reset_index().sort_values(by='ir
```

```
moviesPerYear
```

	level_0	index	year
0	91	1920	1
1	90	1922	1
2	79	1925	2
3	63	1927	3
4	83	1928	1
...	...	...	...
87	7	2016	239
88	5	2017	252
89	9	2018	235
90	20	2019	114
91	23	2020	52

92 rows × 3 columns

```
moviesPerYear = moviesPerYear.drop(columns=['level_0'])
```

```
moviesPerYear = moviesPerYear.rename(columns={'index':'year','year':'films_produced'})
```

```
avgRating_year = avgRating_year.join(moviesPerYear['films_produced'])
```

```
avgRating_year
```

	year	tomatometer_rating	audience_rating	rating	films_produced
0	1920	98.000000	89.000000	93.500000	1
1	1922	97.000000	87.000000	92.000000	1
2	1925	95.000000	88.500000	91.750000	2
3	1927	96.000000	92.000000	94.000000	3
4	1928	98.000000	93.000000	95.500000	1
...	...	...	...	...	...
87	2016	65.326360	60.121339	62.723849	239
88	2017	64.686508	61.297619	62.992063	252
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

- lets look at films in years that ATLEAST 10 films were made

```
x = avgRating_year['year'].loc[avgRating_year['films_produced'] >= 10]
y = avgRating_year['rating'].loc[avgRating_year['films_produced'] >= 10]
y1 = avgRating_year['tomatometer_rating'].loc[avgRating_year['films_produced'] >= 10]
y2 = avgRating_year['audience_rating'].loc[avgRating_year['films_produced'] >= 10]

ratings_someTime = go.Figure()
# ratings_year.add_trace(go.Scatter(x=x, y=y, name='Average Rating'))
ratings_someTime.add_trace(go.Scatter(x=x, y=y1, name='Critic Rating', mode='lines'))
ratings_someTime.add_trace(go.Scatter(x=x, y=y2, name='Audience Rating', mode='lines')

ratings_someTime.update_layout(xaxis_title='Year', yaxis_title='Rating', title='Rating
ratings_someTime.show()
# ratings_someTime.write_image("finalFigs/Ratings over Significant Times.png")
```

Ratings over Some Time

- In this graph again we see that in the mid 2000s many movies were either GOOD or BAD depending on who you asked.
- Professional movie critics did not rate the films produced in this period from 1998 - 2010 as highly as typical audiences like you and I did.
- For this visualization, only years where atleast 10 films were produced were included. This to me is a much fairer comparison since the previous graph had years that as little as only 1 film were produced.
- Also from this figure we can see that older films generally tend to be rated higher by critics, this could be because older films have higher fandom , more time for analysis and critical thinking to occur, and more nostalgia over films produced in the last decade.
- The highest rated movies of this era according to audience score are below

```
new.columns
```

```
Index(['movie_title', 'content_rating', 'genres', 'directors', 'actors',
       'original_release_date', 'streaming_release_date', 'runtime',
       'production_company', 'tomatometer_status', 'tomatometer_rating',
       'tomatometer_count', 'audience_status', 'audience_rating',
       'audience_count', 'year', 'rating'],
      dtype='object')
```

```
topAudienceMovies = new.loc[(new['year'] >= 1997) & (new['year'] <= 2010)].sort_values
```

```
topAudienceMovies.head(10)
```

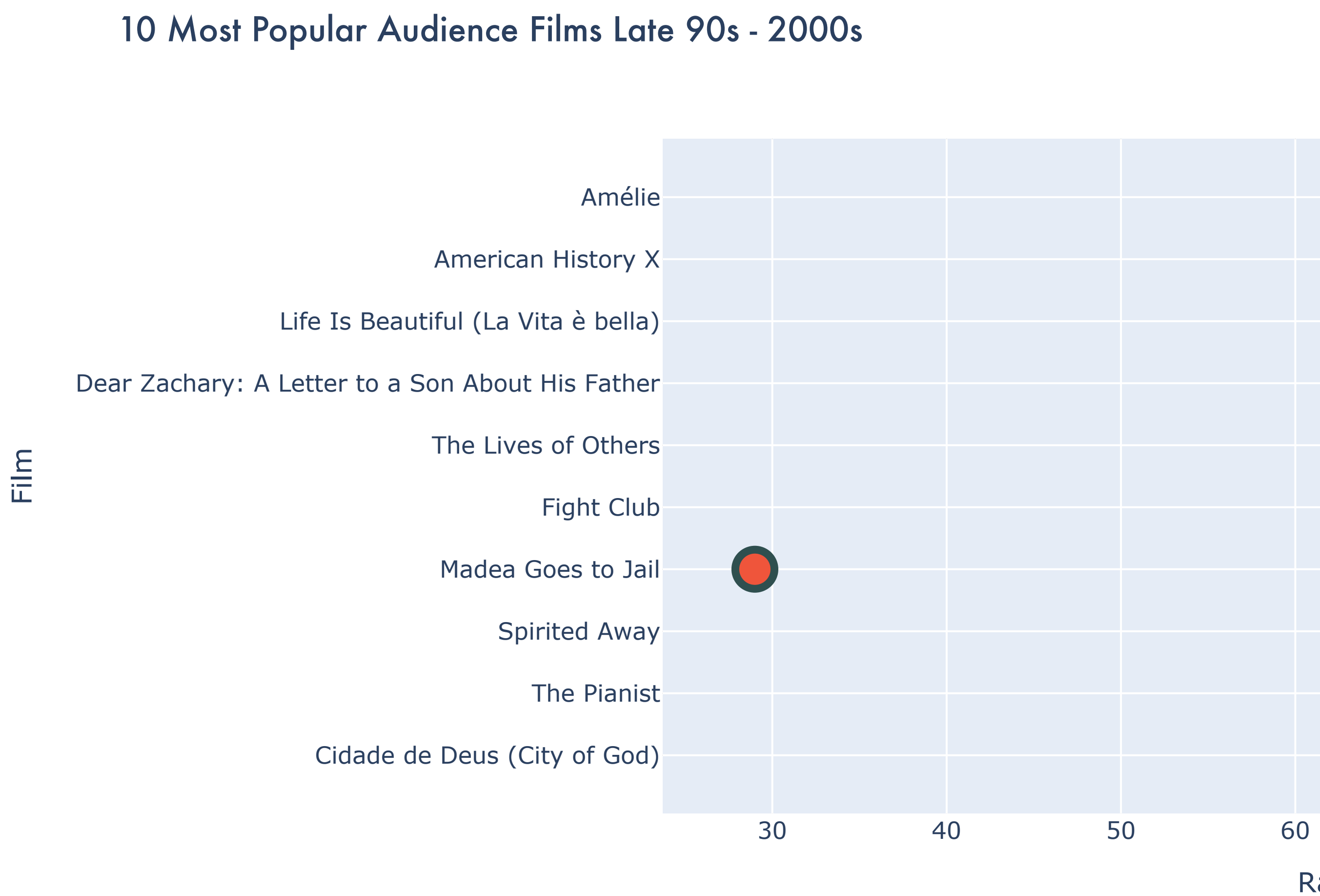
	movie_title	content_rating	genres	directors	actors	origina
1363	Cidade de Deus (City of God)	R	[Action & Adventure, Art House & International...	Fernando Meirelles, Kátia Lund	[Alexandre Rodrigues, Leandro Firmino da Hora,...	
3546	The Pianist	R	[Drama]	Roman Polanski	[Adrien Brody, Emilia Fox, Thomas Kretschmann,...	
4136	Spirited Away	PG	[Animation, Drama, Kids & Family, Science Fict...	Hayao Miyazaki	[Rumi Hiiragi, Miyu Irino, Mari Natsuki, Yumi ...	
2957	Madea Goes to Jail	PG-13	[Comedy, Drama]	Tyler Perry	[Tyler Perry, Derek Luke, Keshia Knight Pullia...	
1893	Fight Club	R	[Comedy, Drama]	David Fincher	[Brad Pitt, Edward Norton, Helena Bonham Carte...	
4696	The Lives of Others	R	[Art House & International, Drama]	Florian Henckel von Donnersmarck	[Martina Gedeck, Sebastian Koch, Hans Bauer, U...	
452	Dear Zachary: A Letter to a Son About His Father	NR	[Documentary, Drama, Special Interest]	Kurt Kuenne	[David Bagby, Kathleen Bagby, Heather Arnold, ...	
208	Life Is Beautiful (La Vita è bella)	PG-13	[Art House & International, Comedy, Drama]	Roberto Benigni	[Roberto Benigni, Nicoletta Braschi, Giorgio C...	
739	American	R	[Drama]	Tony Kaye	[Edward Norton, Avery	
			[Art House &		[Audrey Tautou	

```
y = topAudienceMovies['movie_title'].head(10)
x = topAudienceMovies['audience_rating'].head(10)
x2 = topAudienceMovies['tomatometer_rating'].head(10)

topAudience = go.Figure()
topAudience.add_trace(go.Scatter(x=x, y=y, mode='markers', name='Audience', marker=dic
```

```
topAudience.add_trace(go.Scatter(x=x2, y=y, mode='markers', name='Critics', marker=dic
topAudience.add_trace(go.Scatter(x=[91, 29, 79, 80, 83, 89], y=['Cidade de Deus (City
            color='#ef553b',
            size=20,
            line=dict(
                color='DarkSlateGrey',
                width=4
            )
)))

topAudience.update_layout(xaxis_title='Rating', yaxis_title='Film', title='10 Most Pop
# topAudience.write_image('finalFigs/10 Most Popular Audience Films Late 90s-2000s.pn
topAudience.show()
```



10 Most Popular Audience Films Late 90s - 2000s

- With this figure we can see that many of the most popular films were actually not rated well by critics.
- This is especially true with films such as Madea Goes To Jail, Fight Club, and Life is Beautiful.

- These films have high audience ratings but significantly lower critic ratings.
- During this period is when the divide between the audience and the critics is the highest
- Now looking to make use of the directors column

```
directors_ratings = new.groupby('directors')['rating'].mean().to_frame().reset_index()
```

```
directors_ratings
```

	directors	rating
0	A.T. White, A..T. White	67.00
1	Aaron Blaise, Bob Walker	51.00
2	Aaron Horvath, Peter Rida Michail	81.50
3	Aaron Katz (II)	60.75
4	Aaron Moorhead, Justin Benson	77.00
...	...	...
2588	Ziad Doueiri	85.25
2589	Zoe Cassavetes	63.00
2590	Zoe Lister-Jones	80.00
2591	Émile Gaudreault	49.00
2592	Éva Gárdos	61.00

2593 rows × 2 columns

```
directors_counts = new['directors'].value_counts().to_frame().reset_index().sort_value
```

```
directors_counts
```

	level_0	index	directors
0	1321	A.T. White, A..T. White	1
1	1166	Aaron Blaise, Bob Walker	1
2	1433	Aaron Horvath, Peter Rida Michail	1
3	711	Aaron Katz (II)	2
...	...	...	...

```
directors_counts = directors_counts.drop(columns='level_0')
```

```
directors_counts = directors_counts.rename(columns={'index':'directors','directors':'f
```

```
directors_counts
```

	directors	films_directed
0	A.T. White, A..T. White	1
1	Aaron Blaise, Bob Walker	1
2	Aaron Horvath, Peter Rida Michail	1
3	Aaron Katz (II)	2
4	Aaron Moorhead, Justin Benson	1
...	...	...
2588	Ziad Doueiri	2
2589	Zoe Cassavetes	1
2590	Zoe Lister-Jones	1
2591	Émile Gaudreault	1
2592	Éva Gárdos	1

2593 rows × 2 columns

Here i am going to sort both of the two arrays by the director names and then join my column from one into the other

I am going to do it this way because earlier i had issues that when joining my data would get shuffled

```
directors_counts = directors_counts.sort_values(by='directors')
```

```
directors_ratings = directors_ratings.sort_values(by='directors')
```

```
directors_counts
```

	directors	films_directed
0	A.T. White, A..T. White	1
1	Aaron Blaise, Bob Walker	1
2	Aaron Horvath, Peter Rida Michail	1
3	Aaron Katz (II)	2
4	Aaron Moorhead, Justin Benson	1
...	...	...
2588	Ziad Doueiri	2
2589	Zoe Cassavetes	1
2590	Zoe Lister-Jones	1
2591	Émile Gaudreault	1
2592	Éva Gárdos	1

2593 rows × 2 columns

```
directors_ratings
```

	directors	rating
0	A.T. White, A..T. White	67.00
1	Aaron Blaise, Bob Walker	51.00
2	Aaron Horvath, Peter Rida Michail	81.50
3	Aaron Katz (II)	60.75
4	Aaron Moorhead, Justin Benson	77.00
...	...	...
2588	Ziad Doueiri	85.25
2589	Zoe Cassavetes	63.00
2590	Zoe Lister-Jones	80.00
2591	Émile Gaudreault	49.00
2592	Éva Gárdos	61.00

2593 rows × 2 columns

```
directors_ratings = directors_ratings.join(directors_counts['films_directed'])
```

```
directors_ratings
```

	directors	rating	films_directed
0	A.T. White, A..T. White	67.00	1
1	Aaron Blaise, Bob Walker	51.00	1
2	Aaron Horvath, Peter Rida Michail	81.50	1
3	Aaron Katz (II)	60.75	2
4	Aaron Moorhead, Justin Benson	77.00	1
...	...	...	...
2588	Ziad Doueiri	85.25	2
2589	Zoe Cassavetes	63.00	1
2590	Zoe Lister-Jones	80.00	1
2591	Émile Gaudreault	49.00	1
2592	Éva Gárdos	61.00	1

2593 rows × 3 columns

```
directors_ratings.sort_index()
```

	directors	rating	films_directed
0	A.T. White, A..T. White	67.00	1
1	Aaron Blaise, Bob Walker	51.00	1
2	Aaron Horvath, Peter Rida Michail	81.50	1
3	Aaron Katz (II)	60.75	2
4	Aaron Moorhead, Justin Benson	77.00	1
...	...	...	...
2588	Ziad Doueiri	85.25	2
2589	Zoe Cassavetes	63.00	1
2590	Zoe Lister-Jones	80.00	1
2591	Émile Gaudreault	49.00	1
2592	Éva Gárdos	61.00	1

2593 rows × 3 columns

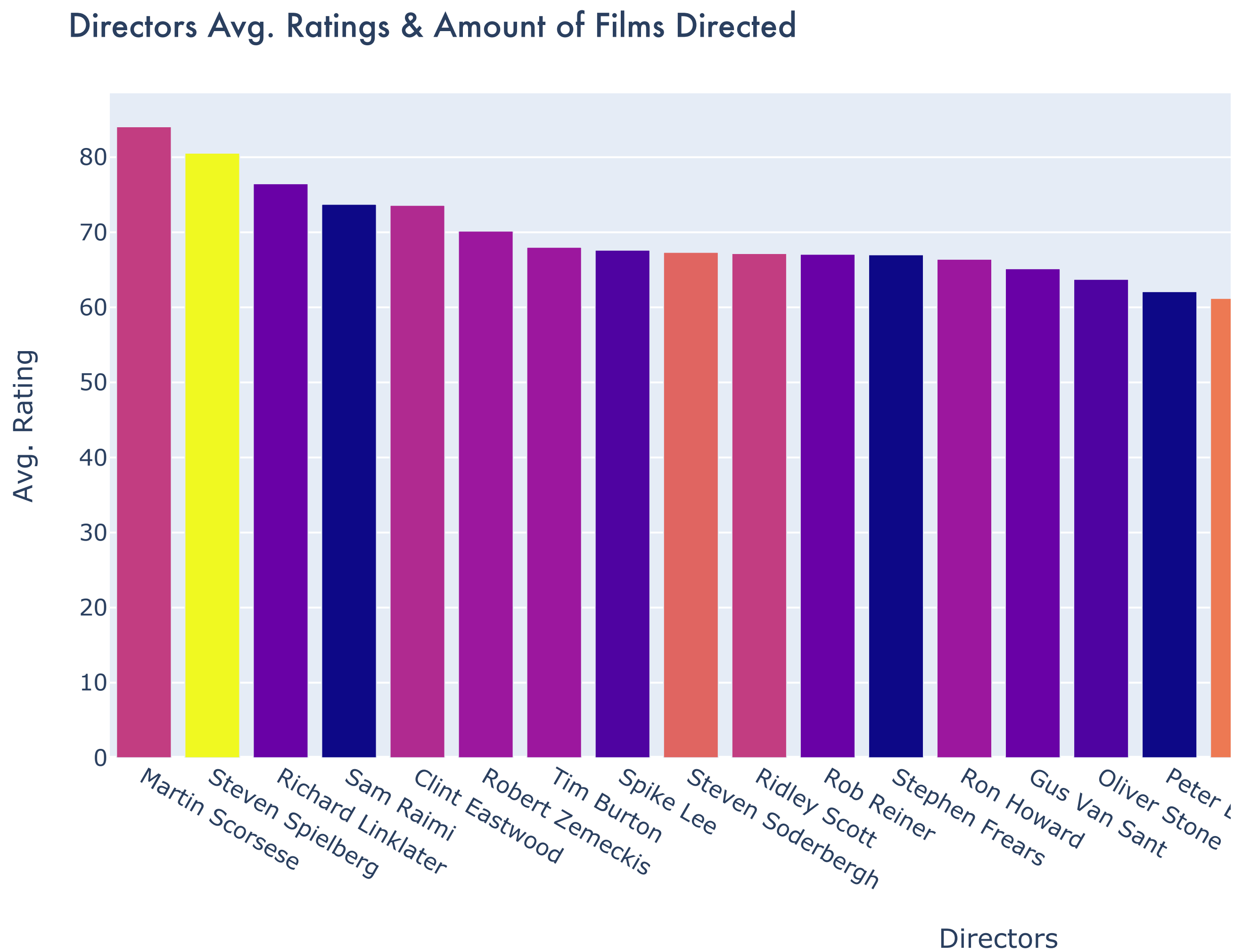
```
directors_ratings = directors_ratings.loc[directors_ratings['films_directed']>10].sort
```

```
directors = directors_ratings.loc[directors_ratings['films_directed']>10]['directors']

ratings = directors_ratings.loc[directors_ratings['films_directed']>10]['rating']

filmCount = directors_ratings['films_directed']


popularDirectors = px.bar(directors_ratings, x=directors, y=ratings, hover_data=['dire
popularDirectors.update_layout(title = 'Directors Avg. Ratings & Amount of Films Direc
popularDirectors.show()
# popularDirectors.write_image('finalFigs/Directors Avg Ratings & Amount of Films Dire
```



Directors Avg. Ratings & Amount of Films Directed

- From this bar graph we can see that the most highly rated director on average is Martin Scorsese. He has an average rating of 80.53 but has directed 8 less films than the second highest rated director Steven Spielberg

- One contributing factor of this could be that films by Steven produce more profits for their respective production company than those by Martin. Unfortunately this dataset does not have economical data and as such this can only be a possibilty of one of the many reasons why Steven has directed more films than Martin
- Other directors of interest are Steven Soderbergh and Woody Allen, when looking at their ratings their close peers have directed significantly less films than they have.
- This is especially true with Woody Allen who is a bright orange and his two peers are dark blue and dark purple meaning that they have made several fewer films than Woody

```
directors_ratings
```

	directors	rating	films_directed
1630	Martin Scorsese	84.055556	18
2345	Steven Spielberg	80.538462	26
2060	Richard Linklater	76.464286	14

```
new[['content_rating', 'movie_title', 'directors', 'year']]
```

	content_rating	movie_title	directors	year
0	PG	Percy Jackson & the Olympians: The Lightning T...	Chris Columbus	2010
1	R	Please Give	Nicole Holofcener	2010
2	NR	12 Angry Men (Twelve Angry Men)	Sidney Lumet	1957
3	PG-13	10,000 B.C.	Roland Emmerich	2008
4	NR	The 39 Steps	Alfred Hitchcock	1935
...	...	...	...	...
5453	PG	Zookeeper	Frank Coraci, Walt Becker	2011
5454	PG-13	Zoolander	Ben Stiller	2001
5455	PG-13	Zoolander 2	Ben Stiller	2016
5456	PG	Zoom	Peter Hewitt	2006
5457	PG	Zootopia	Byron Howard, Rich Moore, Jared Bush	2016

```
new['month'] = pd.DatetimeIndex(new['original_release_date']).month
```

```
new
```

	movie_title	content_rating	genres	directors	actors	original_release
0	Percy Jackson & the Olympians: The Lightning T...	PG	[Action & Adventure, Comedy, Drama, Science Fi...	Chris Columbus	[Logan Lerman, Brandon T. Jackson, Alexandra D...	201
1	Please Give	R	[Comedy]	Nicole Holofcener	[Catherine Keener, Amanda Peet, Oliver Platt, ...	201
2	12 Angry Men (Twelve Angry Men)	NR	[Classics, Drama]	Sidney Lumet	[Martin Balsam, John Fiedler, Lee J. Cobb, E.G...	195
3	10,000 B.C.	PG-13	[Action & Adventure, Classics, Drama]	Roland Emmerich	[Steven Strait, Camilla Belle, Cliff Curtis, J...	200
4	The 39 Steps	NR	[Action & Adventure, Classics, Mystery & Suspe...	Alfred Hitchcock	[Robert Donat, Madeleine Carroll, Godfrey Tear...	193
...	...	...	...	...	...	
5453	Zookeeper	PG	[Comedy, Romance]	Frank Coraci, Walt Becker	[Kevin James, Rosario Dawson, Ken Jeong, Lesli...	201
5454	Zoolander	PG-13	[Comedy, Special Interest]	Ben Stiller	[Ben Stiller, Owen Wilson, Will Ferrell, Chris...	200
					[Ben Stiller,	

5455	Zoolander 2	PG-13	[Comedy]	Ben Stiller	Owen Wilson, Will Ferrell, Penel...	201
			[Action & ... Kids &		[Tim Allen, ... Chevy	

```
content_rating = new[['movie_title', 'content_rating', 'rating', 'year']]
```

```
content_rating
```

	movie_title	content_rating	rating	year
0	Percy Jackson & the Olympians: The Lightning T...	PG	51.0	2010
1	Please Give	R	75.5	2010
2	12 Angry Men (Twelve Angry Men)	NR	98.5	1957
3	10,000 B.C.	PG-13	22.5	2008
4	The 39 Steps	NR	91.0	1935
...	...	...	...	...
5453	Zookeeper	PG	27.5	2011
5454	Zoolander	PG-13	72.0	2001
5455	Zoolander 2	PG-13	21.0	2016
5456	Zoom	PG	18.5	2006
5457	Zootopia	PG	95.0	2016

5458 rows × 4 columns

```
content_rating_80s = content_rating.loc[(content_rating['year'] >= 1980) & (content_ra
```

```
content_rating_80s
```

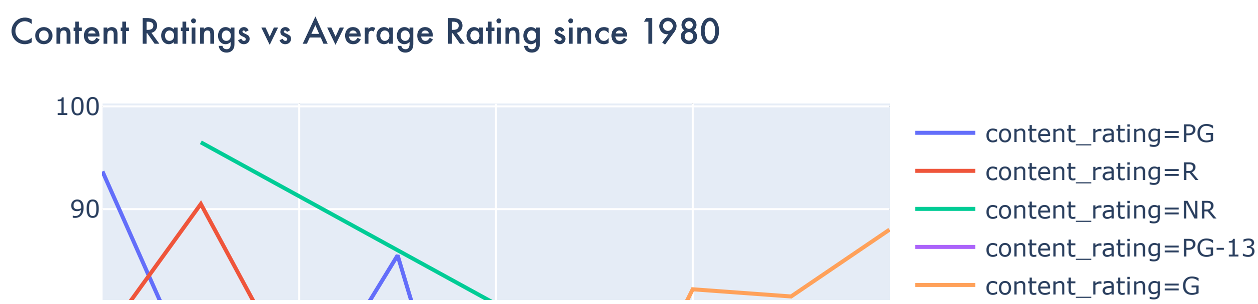
	movie_title	content_rating	rating	year
0	Percy Jackson & the Olympians: The Lightning T...	PG	51.0	2010
1	Please Give	R	75.5	2010
5	The Lost City	R	44.5	2005
7	Deep Blue	G	73.5	2005
13	Saint Ralph	PG-13	74.5	2005

```
content_ratings_80s = content_rating_80s.groupby(['year', 'content_rating'])['rating']
```

```
content_ratings_80s.head()
```

	year	content_rating	rating
0	1980	PG	93.666667
1	1980	R	77.277778
2	1985	NR	96.500000
3	1985	PG	72.125000
4	1985	PG-13	64.500000

```
content_year = px.line(content_ratings_80s, x='year', y='rating', color='content_ratir
content_year.update_layout(title='Content Ratings vs Average Rating since 1980', xaxis
content_year.show()
# content_year.write_image('finalFigs/Content Ratings vs Avg Rating since 1980.png')
```



Content Ratings vs Average Rating since 1980

- For this visualization I decided to go from 1980 until now because the amount of films from this era is much higher in the dataset and the films by this time all began to have content ratings that are still in use today.
- From this graph we see that films for children, rated G, are very popular with critics and audience members alike.
- Rated G films are ranked the highest according to the average rating of these films for the year 2020 and for good reason. They are perfect for children and provide simple storylines with ever increasing animation quality that makes them look as if they were shot in real life.
- What this leads me to believe is that if a movie producer wanted to produce films to please critics and audiences the best type of movie to make would be rated G movies.

Conclusion

At the beginning of this project I hoped to find interesting information that would not be easy to find or discern loooking at a csv file. I believe that I have achieved that goal. Of all of the visualizations that I have created here the one that most shocks me is the one where I compared the critic scores and audience scores during the mid 2000s when the gap between the two was the highest. Critics absolutely disliked the Madea movie, but the audience loved it and that to me was interesting because I am a big fan of Tyler Perry and all of his work. The other visualization that really blew my mind was the one about the biggest production companies. I always knew that the movie industry was very lucrative but I never would have thought that 5 firms created one fourth of the total films in this dataset. That is an insight that really shocks me given that there are so many production companies yet 5 are producing films day in and day out. These insights and visualizations are only the tip of the iceburg. If I had more time and just a little more knowledge I would like to create several visualizations using the 'genres' and 'actors' columns. With this project they proved very difficult to work with due to them being lists of strings, but I know that one way or another they can be worked with and must. I tried to create a chart with genres but had great difficulty trying to figure out how to group films based on genres especially when films were classified as more than one