In [ ]:
```python
import sqlite3
import pandas
import seaborn
import matplotlib.pyplot as plt
```

In [ ]:
```python
my_ratings = pandas.read_csv('ratings.csv')
```

In [ ]:
```python
# Dropping less clean columns that have cleaner counterparts in downloadable datasets
my_ratings.drop(['URL', 'Genres', 'Release Date', 'Directors'], axis=1, inplace=True, errors='ignore')

# Renaming columns to match downloadable datasets
my_ratings.rename(columns={
  'Const': 'tconst',
  'Title': 'primaryTitle',
  'Your Rating': 'yourRating',
  'Date Rated': 'dateRated',
  'Title Type': 'titleType',
  'IMDb Rating': 'averageRating',
  'Num Votes': 'numVotes',
  'Runtime (mins)': 'runtimeMinutes'
}, inplace=True)
```

In [ ]:
```python
# From now on, we work with these columns
print(my_ratings.columns)

# Dropping everything but movies
my_ratings = my_ratings[my_ratings['titleType'] == 'movie']
print("Number of ratings:", len(my_ratings))
```

```
Index(['tconst', 'yourRating', 'dateRated', 'primaryTitle', 'titleType',
       'averageRating', 'runtimeMinutes', 'Year', 'numVotes'],
      dtype='object')
Number of ratings: 4327
```

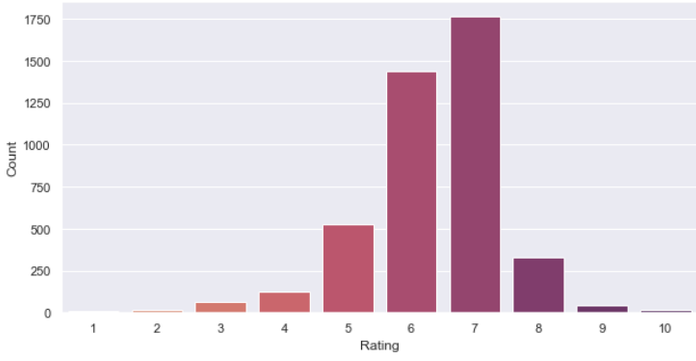
Analysis 0 - Ratings histogram

Computing a baseline of sorts.

This may highlight biases towards specific numbers on the 1 to 10 scale.

In [ ]:
```python
seaborn.set(rc = {'figure.figsize': (10,5)})
plot = seaborn.countplot(my_ratings['yourRating'], palette='flare')
_ = plot.set_xlabel('Rating')
_ = plot.set_ylabel('Count')
```

```
/Users/antriksh/Programming/PersonalMovieAnalysis/venv/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```



Analysis 1 - Most and least popular movies

In [ ]:
```python
rfdb = sqlite3.connect('rackfocus_out.db')
titles = pandas.read_sql_query('select tconst, startYear as releaseYear from title_basics', rfdb)
rfdb.close()
```

Most popular

In [ ]:
```python
my_ratings.merge(titles, on='tconst').sort_values(by='numVotes', ascending=False).head(15)[['primaryTitle', 'releaseYear', 'numVotes']]
```

Out[ ]:

	primaryTitle	releaseYear	numVotes
420	The Shawshank Redemption	1994.0	2741048
2992	The Dark Knight	2008.0	2713919
1075	Inception	2010.0	2409105
1074	Fight Club	1999.0	2182660
374	Forrest Gump	1994.0	2132773
412	Pulp Fiction	1994.0	2106302
1043	The Matrix	1999.0	1954860
858	The Lord of the Rings: The Fellowship of the Ring	2001.0	1913262
3444	The Godfather	1972.0	1906146
3708	Interstellar	2014.0	1905400
1375	The Lord of the Rings: The Return of the King	2003.0	1884506
1061	The Dark Knight Rises	2012.0	1739180
1376	The Lord of the Rings: The Two Towers	2002.0	1701085
530	Se7en	1995.0	1694784
1513	Django Unchained	2012.0	1596674

Least popular

In [ ]:
```python
my_ratings.merge(titles, on='tconst').sort_values(by='numVotes', ascending=True).head(15)[['primaryTitle', 'releaseYear', 'numVotes']]
```

Out[ ]:

	primaryTitle	releaseYear	numVotes
3178	For Lack of Better	2015.0	20
1575	Pedal-Driven: A Bikeumentary	2011.0	34
2737	Vertical Frontier	2002.0	43
2541	Warriors	2015.0	48
1689	Trash and Progress	2012.0	53
2330	PAY 2 PLAY: Democracy's High Stakes	2014.0	64
731	Death by Design: Where Parallel Worlds Meet	1997.0	68
1526	Boys of Bonneville: Racing on a Ribbon of Salt	2011.0	70
755	The Listening Project	2008.0	79
3276	Reset	2016.0	80
326	Smoke	1993.0	84
1431	Cash Crop	2010.0	89
1301	The Edge of Dreaming	2009.0	89
3642	Follow My Voice: With the Music of Hedwig	2006.0	95
1278	Absinthe	2010.0	98

Clearing dataframes

In [ ]:
```
titles.iloc[0:0]
```

Out[ ]:

tconst	releaseYear

Analysis 2 - Directors

In [ ]:
```
rfdb = sqlite3.connect('rackfocus_out.db')
names = pandas.read_sql_query('select nconst, primaryName from name_basics', rfdb)
title_director = pandas.read_sql_query('select tconst, nconst from title_principals where category="director"', rfdb)
rfdb.close()

print(names.columns)
print(title_director.columns)
```

```
Index(['nconst', 'primaryName'], dtype='object')
Index(['tconst', 'nconst'], dtype='object')
```

"Favorite" directors

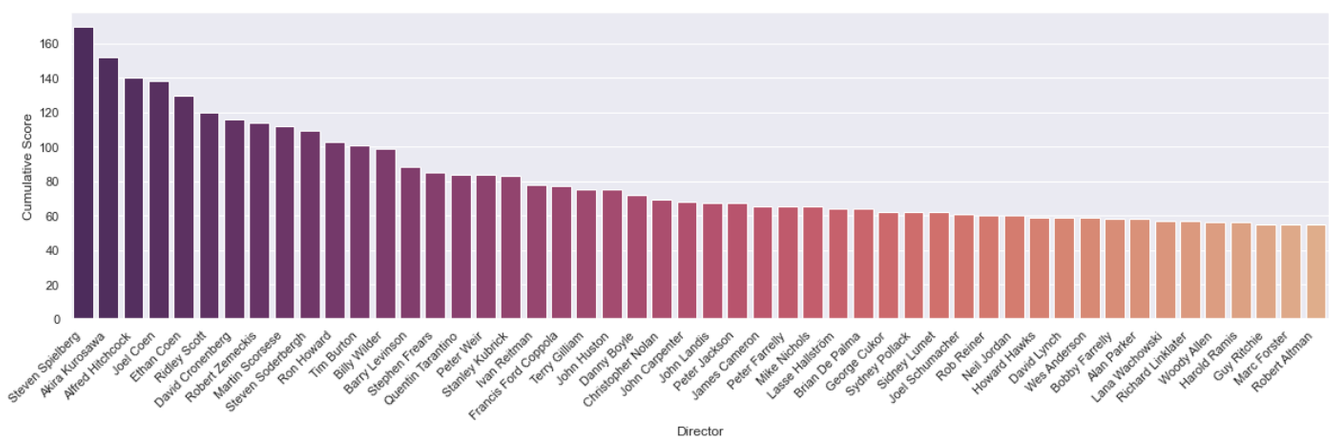
In [ ]:
```
my_r_director = my_ratings.merge(title_director, on='tconst')
```

Cumulative ratings (raw)

A naïve way of finding favorite directors - just tally up your raw rating numbers per director.

In [ ]:
```
my_r_d_cumscore = my_r_director.groupby('nconst')['yourRating'].sum().reset_index()
my_r_d_cumscore_results = my_r_d_cumscore.sort_values(by='yourRating', ascending=False).head(50).merge(names, on='nconst')
my_r_d_cumscore_results.rename(columns={
  'primaryName': 'Director',
  'yourRating': 'Cumulative Score'
}, inplace=True)
```

In [ ]:
```
seaborn.set(rc = {'figure.figsize': (20,5)})
plot = seaborn.barplot(data=my_r_d_cumscore_results, x='Director', y='Cumulative Score', palette='flare_r')
_ = plot.set_xticklabels(plot.get_xticklabels(), rotation=45, horizontalalignment='right')
```



Mean ratings (refined)

Possibly better way of finding favorite directors - find the average rating per director.

In [ ]:
```
my_r_d_avgscore = my_r_director.groupby('nconst')['yourRating'].mean().reset_index()
my_r_d_avgscore_results = my_r_d_avgscore.sort_values(by='yourRating', ascending=False).head(50).merge(names, on='nconst')
my_r_d_avgscore_results.rename(columns={
  'primaryName': 'Director',
  'yourRating': 'Cumulative Score'
}, inplace=True)
my_r_d_avgscore_results[['Director', 'Cumulative Score']]
```

Out[ ]:

	Director	Cumulative Score
0	Richard Thorpe	10.0
1	Norman Taurog	10.0
2	Dean Parisot	9.0
3	Kimberly Peirce	8.5
4	Lee Unkrich	8.2
5	Joseph Sargent	8.0

6	Laurence Dunmore	8.0
7	Henry Levin	8.0
8	Mark Levinson	8.0
9	Harold Crooks	8.0
10	Matthew Robinson	8.0
11	Charles Crichton	8.0
12	Richard Loncraine	8.0
13	Peter Lord	8.0
14	Robert Lepage	8.0
15	Sebastián Cordero	8.0
16	Jim Handley	8.0
17	Angelina Maccarone	8.0
18	Jillian Schlesinger	8.0
19	Lixin Fan	8.0
20	Tom Stoppard	8.0
21	Claude Berri	8.0
22	Marc Caro	8.0
23	Clark Gregg	8.0
24	Yann Samuell	8.0
25	Mervyn LeRoy	8.0
26	Eric Leighton	8.0
27	Alejandro Agresti	8.0
28	Bradley King	8.0
29	Isti Madarász	8.0
30	Vicky Jenson	8.0
31	Dito Montiel	8.0
32	Ford Beebe Jr.	8.0
33	Thymaya Payne	8.0
34	Mathieu Roy	8.0
35	Jeff Orlowski-Yang	8.0
36	Michael Cimino	8.0
37	Daniel Dencik	8.0
38	Céline Sciamma	8.0
39	Veit Helmer	8.0
40	Ulrike Koch	8.0
41	Jeff Prosserman	8.0
42	David Slade	8.0
43	Franny Armstrong	8.0
44	Geoffrey Smith	8.0
45	Carlos Brooks	8.0
46	Cy Endfield	8.0
47	Kate Davis	8.0
48	Terence Davies	8.0
49	Curt Morgan	8.0

Clearing dataframes

In [ ]:
```
names.iloc[0:0]
title_director.iloc[0:0]
my_r_director.iloc[0:0]
my_r_d_cumscore.iloc[0:0]
my_r_d_avgscore.iloc[0:0]
my_r_d_cumscore_results.iloc[0:0]
my_r_d_avgscore_results.iloc[0:0]
```

Out[ ]:

nconst	Cumulative Score	Director

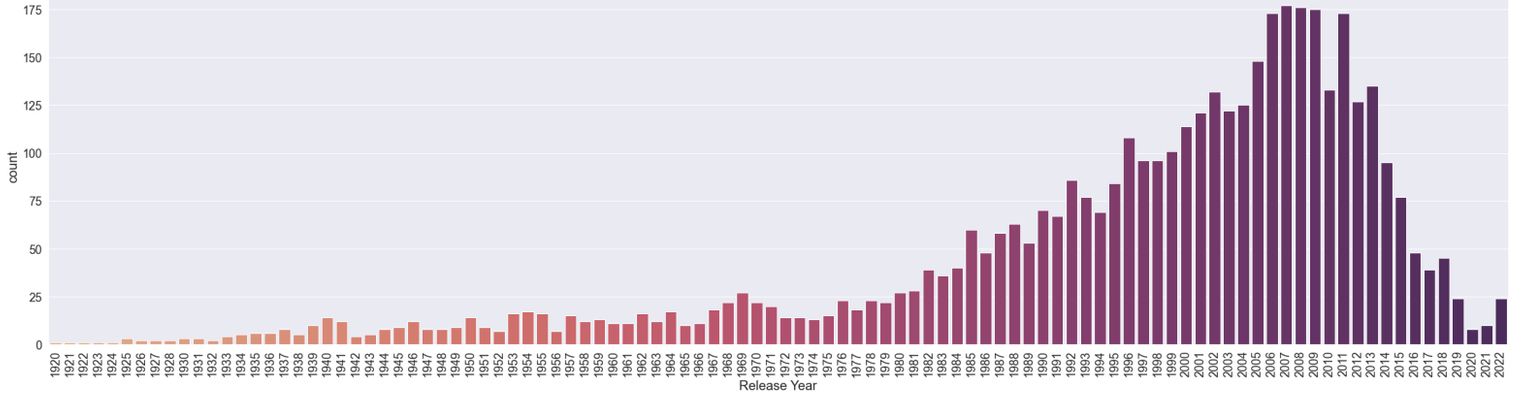
Analysis 3 - Through the years

In [ ]:
```
rfdb = sqlite3.connect('rackfocus_out.db')
titles = pandas.read_sql_query('select tconst, startYear as releaseYear from title_basics where titleType="movie"', rfdb)
rfdb.close()
```

Movie count by release year

In [ ]:
```
my_r_titles_release_year = my_ratings.merge(titles, on='tconst')
my_r_titles_release_year.rename(columns={
  'count': 'Count',
  'releaseYear': 'Release Year'
}, inplace=True)
seaborn.set(rc = {'figure.figsize': (40,10), 'axes.labelsize': 20, 'xtick.labelsize': 18, 'ytick.labelsize': 18})
plot = seaborn.countplot(my_r_titles_release_year['Release Year'], palette='flare')
_ = plot.set_xticklabels([int(float(label.get_text())) for label in plot.get_xticklabels()], rotation=90)
```

```
/Users/antriksh/Programming/PersonalMovieAnalysis/venv/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```



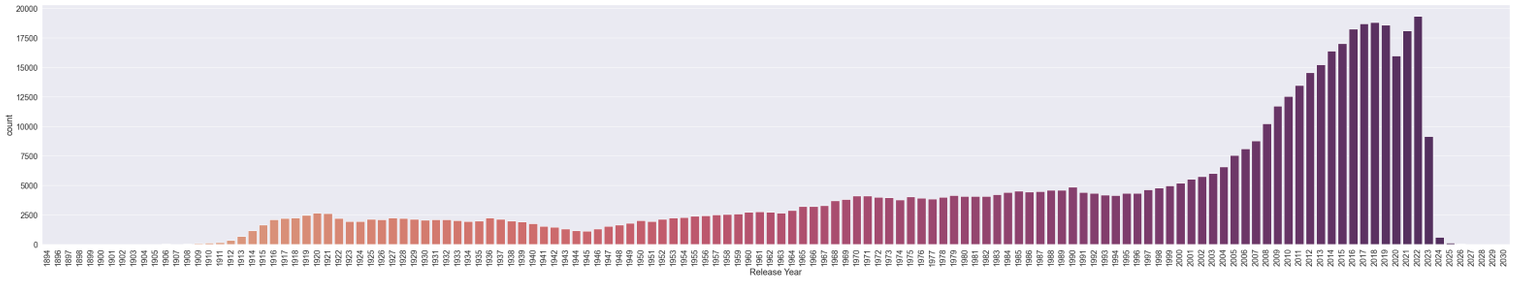
Plotting all movie release years on IMDb

Data could be unclean.

In [ ]:
```
seaborn.set(rc = {'figure.figsize': (60,10), 'axes.labelsize': 20, 'xtick.labelsize': 18, 'ytick.labelsize': 18})
plot = seaborn.countplot(titles['releaseYear'], palette='flare')
plot.set_xticklabels([int(float(label.get_text())) for label in plot.get_xticklabels()], rotation=90)
plot.set_xlabel('Release Year')
```

```
/Users/antriksh/Programming/PersonalMovieAnalysis/venv/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

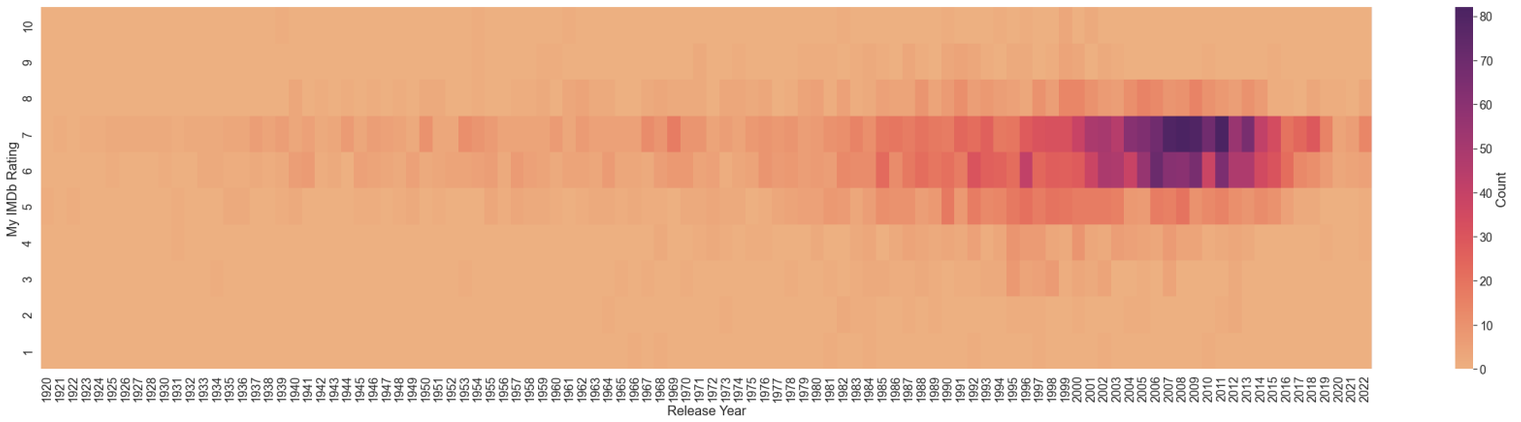
Out[ ]:
```
Text(0.5, 0, 'Release Year')
```



Release year / rating heatmap

In [ ]:
```
my_r_titles = my_ratings.merge(titles, on='tconst')
my_r_titles.rename(columns={
  'yourRating': 'Your Rating',
  'releaseYear': 'Release Year'
}, inplace=True)
my_r_release_year = pandas.crosstab(my_r_titles['Your Rating'], my_r_titles['Release Year'])
seaborn.set(rc = {'figure.figsize': (45,10), 'axes.labelsize': 20, 'xtick.labelsize': 18, 'ytick.labelsize': 18})
plot = seaborn.heatmap(my_r_release_year, cbar_kws={'label': 'Count'}, cmap='flare')
plot.set_xticklabels([int(float(label.get_text())) for label in plot.get_xticklabels()], rotation=90)
plot.invert_yaxis()
plot.set_ylabel('My IMDb Rating')
```

Out[ ]:
```
Text(373.5, 0.5, 'My IMDb Rating')
```



Clearing dataframes

In [ ]:
```
my_r_titles_release_year.iloc[0:0]
my_r_titles.iloc[0:0]
my_r_release_year.iloc[0:0]
```

Out[ ]:

Release Year	1920.0	1921.0	1922.0	1923.0	1924.0	1925.0	1926.0	1927.0	1928.0	1930.0	...	2013.0	2014.0	2015.0	2016.0	2017.0	2018.0	2019.0	2020.0	2021.0	2022.0
Your Rating																					

0 rows × 102 columns

Analysis 4 - Ratings by runtime

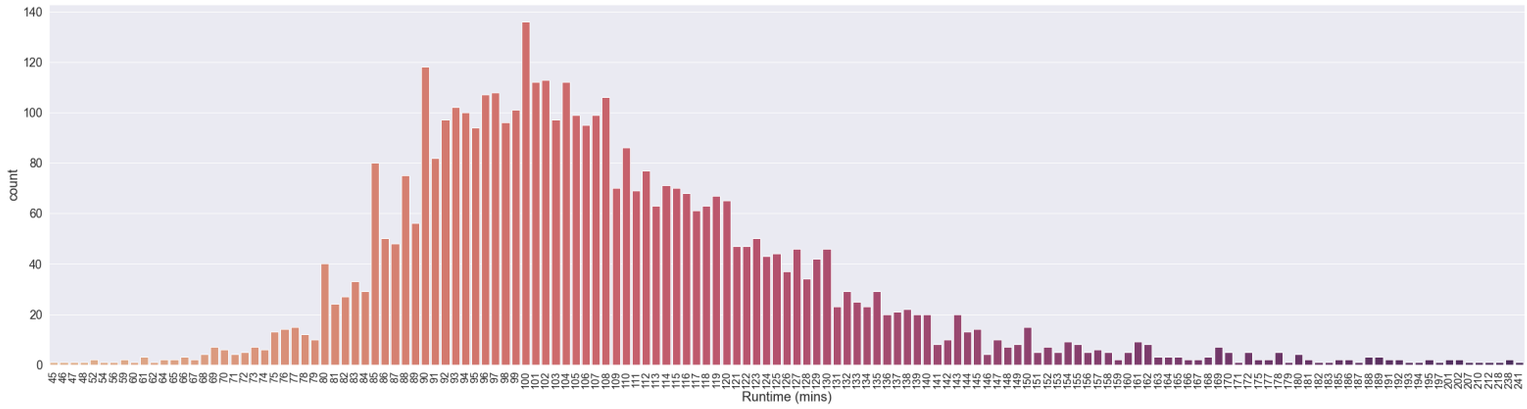
In [ ]:
```
rfdb = sqlite3.connect('rackfocus_out.db')
titles = pandas.read_sql_query('select tconst, runtimeMinutes from title_basics where titleType="movie"', rfdb)
rfdb.close()
```

Movie count by runtime

In [ ]:
```
seaborn.set(rc = {'figure.figsize': (40,10), 'axes.labelsize': 20, 'xtick.labelsize': 16, 'ytick.labelsize': 18})
plot = seaborn.countplot(my_ratings['runtimeMinutes'], palette='flare')
plot.set_xticklabels([int(float(label.get_text())) for label in plot.get_xticklabels()], rotation=90)
plot.set_xlabel('Runtime (mins)')
```

```
/Users/antriksh/Programming/PersonalMovieAnalysis/venv/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

Out[ ]:
```
Text(0.5, 0, 'Runtime (mins)')
```



Plotting all movie runtimes on IMDb

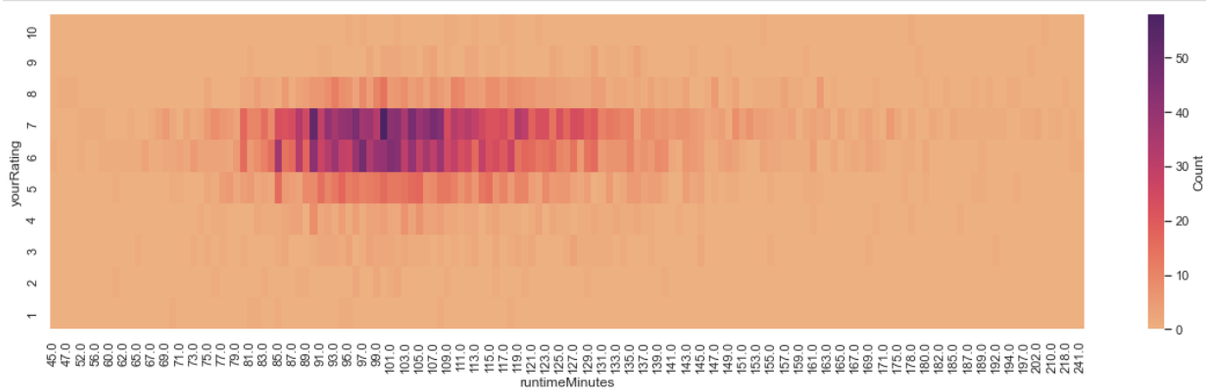
I have no explanations for the patterns nor the extremes. Data could be unclean.

In [ ]:
```
seaborn.set(rc = {'figure.figsize': (90,5)})
plot = seaborn.countplot(titles['runtimeMinutes'], palette='flare')
_ = plot.set_xticklabels(plot.get_xticklabels(), rotation=90)
_ = plot.set_xlabel('Runtime (mins)')
```

```
/Users/antriksh/Programming/PersonalMovieAnalysis/venv/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

Runtime / rating heatmap

In [ ]:
```
my_r_runtime = pandas.crosstab(my_ratings['yourRating'], my_ratings['runtimeMinutes'])
seaborn.set(rc = {'figure.figsize': (20,5)})
plot = seaborn.heatmap(my_r_runtime, cbar_kws={'label': 'Count'}, cmap='flare')
plot.invert_yaxis()
```



Clearing dataframes

In [ ]:
```
titles.iloc[0:0]
my_r_runtime.iloc[0:0]
```

Out[ ]:

runtimeMinutes	45.0	46.0	47.0	48.0	52.0	54.0	56.0	59.0	60.0	61.0	...	195.0	197.0	201.0	202.0	207.0	210.0	212.0	218.0	238.0	241.0
yourRating																					

0 rows × 147 columns

Analysis 5 - Biggest ratings delta

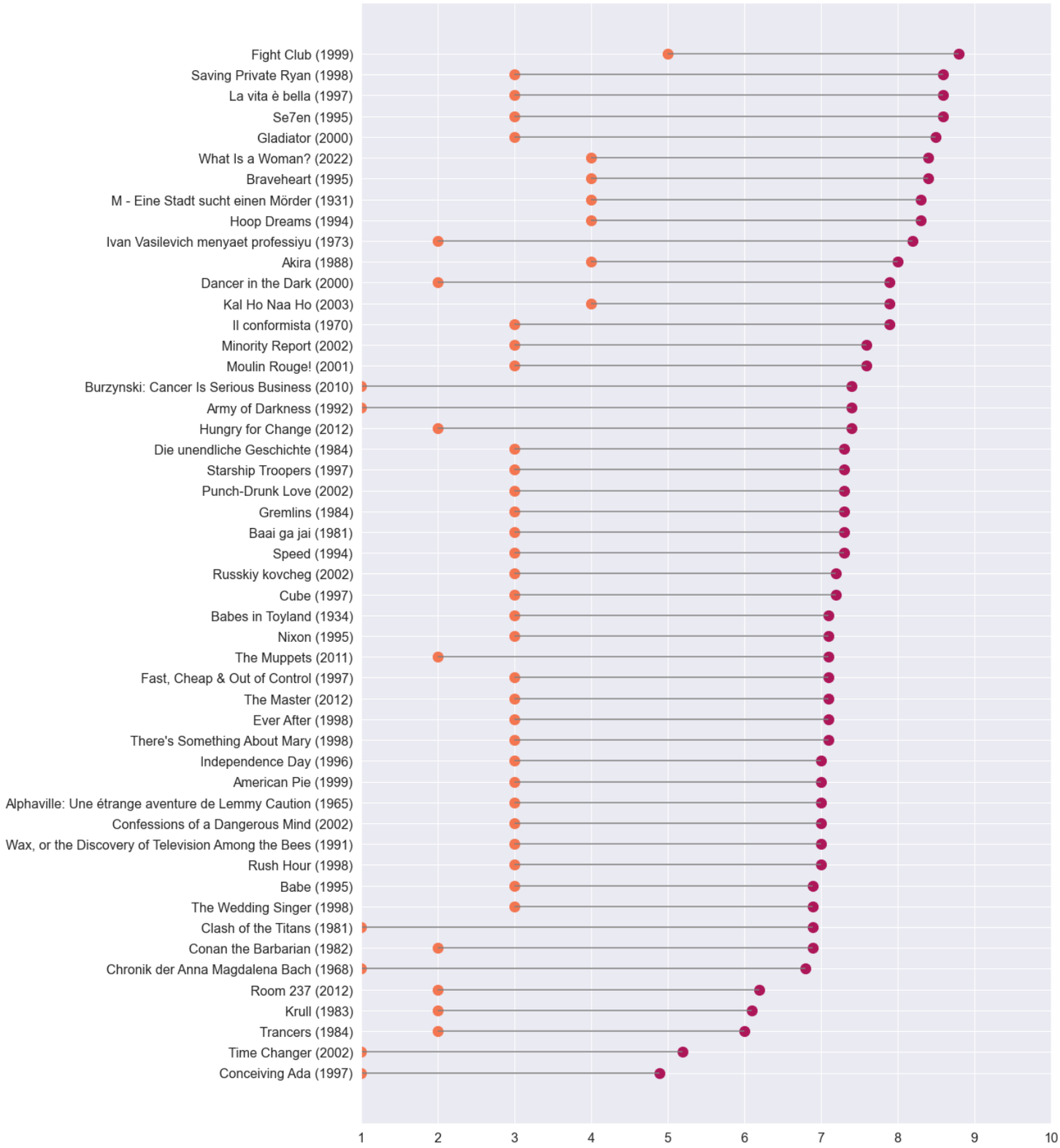
In [ ]:
```
rfdb = sqlite3.connect('rackfocus_out.db')
titles = pandas.read_sql_query('select tconst, startYear as releaseYear from title_basics where titleType="movie"', rfdb)
rfdb.close()
```

"Overrated" movies

Movies that you rated much lower than the IMDb rating populace.

In [ ]:
```
my_r_imdb_r_with_delta = my_ratings[['tconst', 'primaryTitle', 'yourRating', 'averageRating']].merge(titles, on='tconst')
my_r_imdb_r_with_delta['titleAndYear'] = my_r_imdb_r_with_delta.apply(lambda row: f"{row['primaryTitle']} ({int(row['releaseYear'])})", axis=1)
my_r_imdb_r_with_delta['ratingDelta'] = my_r_imdb_r_with_delta.apply(lambda r: r['averageRating'] - r['yourRating'], axis=1)
my_r_imdb_r_with_delta = my_r_imdb_r_with_delta.sort_values(by='ratingDelta', ascending=False).head(50).sort_values(by='averageRating', ascending=False)
```

In [ ]:
```
plt.figure(figsize=(15,25))
plt.hlines(y=my_r_imdb_r_with_delta['titleAndYear'], xmin=my_r_imdb_r_with_delta['yourRating'], xmax=my_r_imdb_r_with_delta['averageRating'], color='grey')
plt.scatter(my_r_imdb_r_with_delta['yourRating'], my_r_imdb_r_with_delta['titleAndYear'], color='#f37651', s=150)
plt.scatter(my_r_imdb_r_with_delta['averageRating'], my_r_imdb_r_with_delta['titleAndYear'], color='#ad1759', s=150)
plt.xlim(1, 10)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
ax = plt.gca()
ax.invert_yaxis()
```

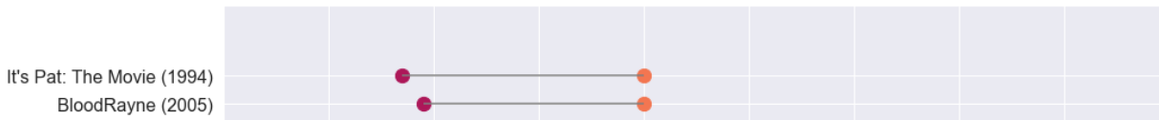


"Underrated" movies

Movies that you rated much higher than the IMDb rating populace.

In [ ]:
```
my_r_imdb_r_with_delta = my_ratings[['tconst', 'primaryTitle', 'yourRating', 'averageRating']].merge(titles, on='tconst')
my_r_imdb_r_with_delta['titleAndYear'] = my_r_imdb_r_with_delta.apply(lambda row: f"{row['primaryTitle']} ({int(row['releaseYear'])})", axis=1)
my_r_imdb_r_with_delta['ratingDelta'] = my_r_imdb_r_with_delta.apply(lambda r: r['yourRating'] - r['averageRating'], axis=1)
my_r_imdb_r_with_delta = my_r_imdb_r_with_delta.sort_values(by='ratingDelta', ascending=False).head(50).sort_values(by='averageRating', ascending=False)
```

In [ ]:
```
plt.figure(figsize=(15,25))
plt.hlines(y=my_r_imdb_r_with_delta['titleAndYear'], xmin=my_r_imdb_r_with_delta['yourRating'], xmax=my_r_imdb_r_with_delta['averageRating'], color='grey')
plt.scatter(my_r_imdb_r_with_delta['yourRating'], my_r_imdb_r_with_delta['titleAndYear'], color='#f37651', s=150)
plt.scatter(my_r_imdb_r_with_delta['averageRating'], my_r_imdb_r_with_delta['titleAndYear'], color='#ad1759', s=150)
plt.xlim(1, 10)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
ax = plt.gca()
```



In [ ]:
```
titles.iloc[0:0]
my_r_imdb_r_with_delta.iloc[0:0]
```

Out[ ]:

tconst	primaryTitle	yourRating	averageRating	releaseYear	titleAndYear	ratingDelta