Top 10000 Popular Movies Dataset

Including libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Reading csv file as df

In [2]:
```
df = pd.read_csv("/kaggle/input/top-10000-popular-movies-tmdb-05-2023/popular_10000
#First five rows of DataFrame
df.head()
```

Out[2]:

	id	title	release_date	genres	original_language	vote_average	vote_count	p
0	758323	The Pope's Exorcist	2023-04-05	['Horror', 'Mystery', 'Thriller']	English	7.4	619	
1	640146	Ant-Man and the Wasp: Quantumania	2023-02-15	['Action', 'Adventure', 'Science Fiction']	English	6.6	2294	
2	502356	The Super Mario Bros. Movie	2023-04-05	['Animation', 'Adventure', 'Family', 'Fantasy'...	English	7.5	1861	
3	868759	Ghosted	2023-04-18	['Action', 'Comedy', 'Romance']	English	7.2	652	
4	594767	Shazam! Fury of the Gods	2023-03-15	['Action', 'Comedy', 'Fantasy', 'Adventure']	English	6.8	1510	

Shape of DataFrame

In [3]:
```
df.shape
```

Out[3]:
```
(10000, 14)
```

List of columns in DataFrame

In [4]:
```
df.columns
```

Out[4]:
```
Index(['id', 'title', 'release_date', 'genres', 'original_language',
       'vote_average', 'vote_count', 'popularity', 'overview', 'budget',
       'production_companies', 'revenue', 'runtime', 'tagline'],
      dtype='object')
```

In [5]:
```
df['tagline'][0]
```

Out[5]:
```
'Inspired by the actual files of Father Gabriele Amorth, Chief Exorcist of the Vat
ican.'
```

Dropping useless columns from DataFrame

In [6]:
```
df.drop(['id','overview','tagline'], axis=1, inplace=True)
```

List of final columns

In [7]:
```
df.columns
```

Out[7]:
```
Index(['title', 'release_date', 'genres', 'original_language', 'vote_average',
       'vote_count', 'popularity', 'budget', 'production_companies', 'revenue',
       'runtime'],
      dtype='object')
```

Checking for null values in DataFrame

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
title                    0
release_date            21
genres                   0
original_language        0
vote_average             0
vote_count               0
popularity               0
budget                   0
production_companies     0
revenue                  0
runtime                  0
dtype: int64
```

Dropping rows with null values from DataFrame

In [9]:
```
df.dropna(axis=0, inplace=True)
```

Unique languages present in "original_language" column

In [10]:
```
df['original_language'].unique()
```

Out[10]:
```
array(['English', 'French', 'Dutch', 'Spanish', 'Korean', 'Japanese',
       'Finnish', 'Ukrainian', 'Norwegian', 'Estonian', 'cn', 'Polish',
       'Russian', 'German', 'Chinese', 'Italian', 'Basque', 'Thai',
       'Turkish', 'Swedish', 'Icelandic', 'Tagalog', 'Bengali', 'Arabic',
       'Tamil', 'Telugu', 'Romanian', 'Indonesian', 'Galician', 'Danish',
       'Macedonian', 'Portuguese', 'Vietnamese', 'Catalan', 'Hindi',
       'Persian', 'Hebrew', 'Serbian', 'Malayalam', 'Greek', 'Hungarian',
       'Czech', 'Norwegian Bokmal', 'xx', 'Kannada', 'Irish', 'Khmer',
       'sh', 'Dzongkha', 'Panjabi', 'Sundanese'], dtype=object)
```

Rows with "xx" as original_language

In [11]:
```
df[df['original_language'] == 'xx']
```

Out[11]:

	title	release_date	genres	original_language	vote_average	vote_count	popularity	bu
6340	Barbie	1977-01-01	[]	xx	2.0	1	9.890	
7837	Vertigo	2016-08-10	['Drama']	xx	2.0	1	10.093	

Dropping rows with "xx" original_language

In [12]:
```
df = df[~(df['original_language'] == 'xx')]
```

Rows with "sh" as original_language

In [13]:
```
df[df['original_language'] == 'sh']
```

Out[13]:

	title	release_date	genres	original_language	vote_average	vote_count	popularity	bu
9096	I Miss Sonia Henie	1971-01-20	['Comedy', 'Drama']	sh	5.5	15	13.323	

Dropping rows with "sh" original_language

In [14]:
```
df = df[~(df['original_language'] == 'sh')]
```

No. of duplicate entries in the DataFrame

In [15]:
```
df.duplicated().sum()
```

Out[15]:
```
0
```

No. of movies with unique titles

In [16]:
```
df['title'].nunique()
```

Out[16]:
```
9629
```

No. of movies with same title

In [17]:
```
df['title'].duplicated().sum()
```

Out[17]:
```
347
```

Rows with duplicate movie titles

In [18]:
```
df[df['title'].isin(df['title'][df['title'].duplicated()])].sort_values('title')
```

Out[18]:

	title	release_date	genres	original_language	vote_average	vote_count	popularit
9364	3:10 to Yuma	1957-08-07	['Western', 'Drama', 'Thriller']	English	7.2	291	10.99
5116	3:10 to Yuma	2007-09-06	['Western']	English	7.2	3131	18.96
2684	A Nightmare on Elm Street	2010-04-30	['Horror', 'Mystery', 'Thriller']	English	5.5	2415	25.67
1637	A Nightmare on Elm Street	1984-11-09	['Horror']	English	7.3	4430	36.60
7330	A Tale of Two Sisters	2023-05-06	[]	English	0.0	0	10.53
...	...	...	...	...	...	...	
902	Wonder Woman	2017-05-30	['Action', 'Adventure', 'Fantasy']	English	7.2	18595	54.48
1133	Wrong Turn	2003-05-30	['Horror', 'Thriller']	English	6.3	2307	50.41
774	Wrong Turn	2021-01-26	['Horror', 'Thriller', 'Drama']	English	6.0	902	59.58
202	X	2022-03-17	['Horror', 'Mystery', 'Thriller']	English	6.8	1918	129.62
5515	X	2011-11-23	['Action', 'Thriller', 'Romance']	English	6.5	221	20.59

661 rows × 11 columns

Rows with empty genres column

In [19]:
```
df[df['genres'].str.len() == 2]
```

Out[19]:

	title	release_date	genres	original_language	vote_average	vote_count	popularity
610	Snake Beauty	1994-03-26	[]	Chinese	0.0	0	61.292
684	Za gyakutai: Nyotai ikedori-hen	1987-07-18	[]	Japanese	6.5	1	62.096
816	Gabriel's Inferno: Part IV	2022-03-30	[]	English	5.0	2	37.269
943	Yu Pui Tsuen	1986-12-12	[]	cn	4.0	4	78.229
1178	Oppressive Torture	1978-01-14	[]	Japanese	4.7	3	45.046
...	...	...	...	...	...	...	...
8809	The Legend of Zhao Yun	2021-01-04	[]	Chinese	2.0	1	9.461
8982	Russian Nymphet: Temptation	2004-11-28	[]	Russian	5.3	3	16.728
9484	La Boheme: Breathe Umphefumlo	2015-02-05	[]	English	6.7	3	11.171
9775	野浪花	1987-05-26	[]	Chinese	0.0	0	8.040
9991	The Witcher Season One Recap: From the Beginning	2021-12-17	[]	English	5.6	8	9.045

61 rows × 11 columns

Dropping rows with empty genres column

In [20]:
```
df = df[~(df['genres'].str.len() == 2)]
```

Rows with empty production_companies column

In [21]:
```
df[df['production_companies'].str.len() == 2]
```

Out[21]:

	title	release_date	genres	original_language	vote_average	vote_count	populari
21	Adrenaline	2022-12-15	['Action']	English	5.9	39	717.8
27	The Elderly	2023-04-21	['Horror', 'Thriller', 'Fantasy']	Spanish	5.8	6	521.2
28	Ripper's Revenge	2023-04-03	['Horror']	English	4.9	10	523.1
36	Gangs of Lagos	2023-04-07	['Crime']	English	6.1	35	462.8
46	Prizefighter: The Life of Jem Belcher	2022-06-30	['Drama', 'History']	English	6.2	122	365.0
...	...	...	...	...	...	...	
9905	My Mother's Lovers	2020-05-29	['Drama']	Spanish	2.8	5	12.4
9924	My Brother's Wife 2	2016-08-31	['Romance', 'Family', 'Drama']	Korean	3.5	2	10.3
9930	Zombie Fight Club	2014-10-23	['Action', 'Horror']	Chinese	4.7	54	12.8
9975	Heart Shot	2022-02-17	['Romance', 'Crime']	English	5.6	69	9.1
9998	My Sister-in-law's Job	2017-08-31	['Drama', 'Romance']	Korean	5.0	5	10.4

519 rows × 11 columns

Dropping rows with empty production_companies column

In [22]:
```
df = df[~(df['production_companies'].str.len() == 2)]
```

Zeroes Count in Columns

In [23]:
```
(df == 0).sum()
```

Out[23]:
```
title                   0
release_date            0
genres                  0
original_language       0
vote_average          130
vote_count            129
popularity              0
budget               4286
production_companies    0
revenue              3986
runtime               106
dtype: int64
```

Rows with zero vote_average or vote_count or runtime values

In [24]:
```
df[(df['vote_average'] == 0) | (df['vote_count'] == 0) | (df['runtime'] == 0)]
```

Out[24]:

	title	release_date	genres	original_language	vote_average	vote_count	popul
23	Fast X	2023-05-17	['Action', 'Crime', 'Thriller']	English	0.0	0	732
100	Kiss, Kiss!	2023-04-26	['Romance', 'Comedy']	Polish	6.9	15	361
101	The Little Mermaid	2023-05-18	['Adventure', 'Family', 'Fantasy', 'Romance']	English	0.0	0	220
128	Transformers: Rise of the Beasts	2023-06-07	['Action', 'Adventure', 'Science Fiction']	English	0.0	0	222
184	The Flash	2023-06-14	['Science Fiction', 'Action', 'Adventure']	English	0.0	0	123
...	...	...	...	...	...	...	
9847	Divaldo: O Mensageiro da Paz	2019-09-12	['Drama']	Portuguese	8.3	59	16
9889	Influencer	2023-05-18	['Thriller', 'Horror', 'Mystery']	English	0.0	0	8
9922	Mental Finger	2023-05-06	['Comedy', 'Action']	Sundanese	0.0	0	9
9926	Patricia, A Hidden Passion	2020-01-21	['Drama', 'Comedy']	Spanish	6.0	33	12
9932	Godzilla x Kong: The New Empire	2024-03-13	['Action', 'Science Fiction', 'Adventure']	English	0.0	0	11

173 rows × 11 columns

Dropping rows with zero vote_average or vote_count or runtime values

In [25]:
```
df = df[~((df['vote_average'] == 0) | (df['vote_count'] == 0) | (df['runtime'] == 
```

Descriptive Statistics of DataFrame

In [26]:
```
df.describe()
```

Out[26]:

	vote_average	vote_count	popularity	budget	revenue	runtime
count	9223.000000	9223.000000	9223.000000	9.223000e+03	9.223000e+03	9223.000000
mean	6.542546	1652.788247	31.635078	2.081719e+07	6.479669e+07	103.287867
std	0.907423	2952.562540	114.867339	3.935068e+07	1.596469e+08	24.410714
min	1.000000	1.000000	7.411000	0.000000e+00	0.000000e+00	2.000000
25%	6.000000	170.000000	13.591000	0.000000e+00	0.000000e+00	91.000000
50%	6.600000	580.000000	17.619000	2.500000e+06	3.769990e+06	101.000000
75%	7.200000	1726.000000	27.182500	2.500000e+07	5.806524e+07	115.000000
max	10.000000	33633.000000	5089.969000	5.793304e+08	2.923706e+09	449.000000

Adding profit column in DataFrame

In [27]:
```
df['profit'] = df['revenue'] - df['budget']
```

In [28]:
```
df.head()
```

Out[28]:

	title	release_date	genres	original_language	vote_average	vote_count	popularity
0	The Pope's Exorcist	2023-04-05	['Horror', 'Mystery', 'Thriller']	English	7.4	619	5089.96
1	Ant-Man and the Wasp: Quantumania	2023-02-15	['Action', 'Adventure', 'Science Fiction']	English	6.6	2294	4665.43
2	The Super Mario Bros. Movie	2023-04-05	['Animation', 'Adventure', 'Family', 'Fantasy'...	English	7.5	1861	3935.55
3	Ghosted	2023-04-18	['Action', 'Comedy', 'Romance']	English	7.2	652	2791.53
4	Shazam! Fury of the Gods	2023-03-15	['Action', 'Comedy', 'Fantasy', 'Adventure']	English	6.8	1510	2702.59

Saving our DataFrame as csv file

In [29]:
```
df.to_csv("movies.csv", index=False)
```

Top 10 Movies of Every Category

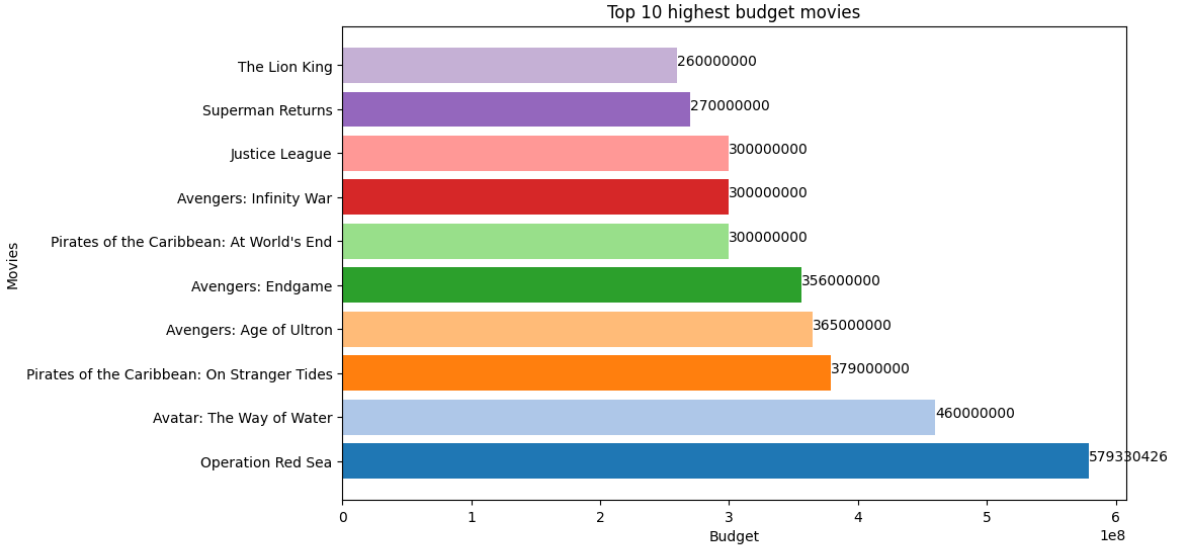
In [30]:
```
max_budget = df.sort_values('budget', ascending=False).head(10)
#max_budget
```

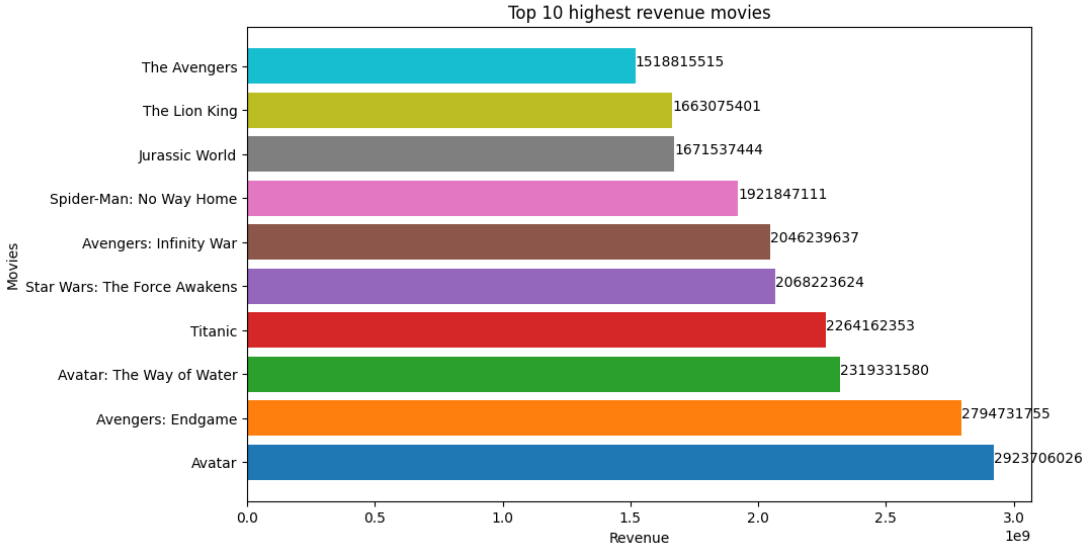
```
max_revenue = df.sort_values('revenue', ascending=False).head(10)
#max_revenue
max_profit = df.sort_values('profit', ascending=False).head(10)
#max_profit
max_loss = df.sort_values('profit', ascending=True).head(10)
max_loss['loss'] = -max_loss['profit']
#max_loss
max_popularity = df.sort_values('popularity', ascending=False).head(10)
#max_popularity
max_vc = df.sort_values('vote_count', ascending=False).head(10)
#max_vc
max_va = df.sort_values('vote_average', ascending=False).head(10)
#max_va
```

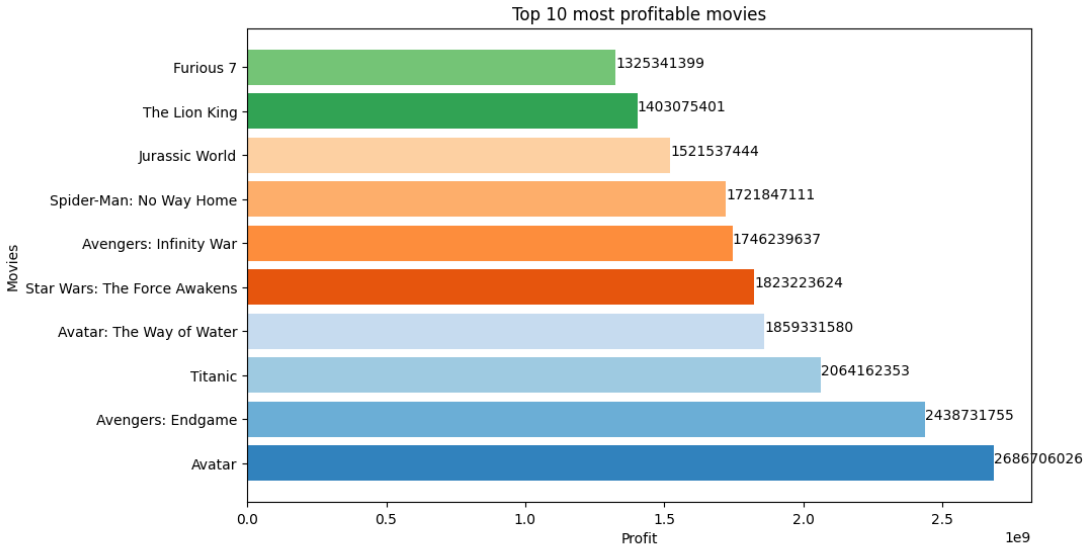
Visualization of Top 10 Movies of Every Category

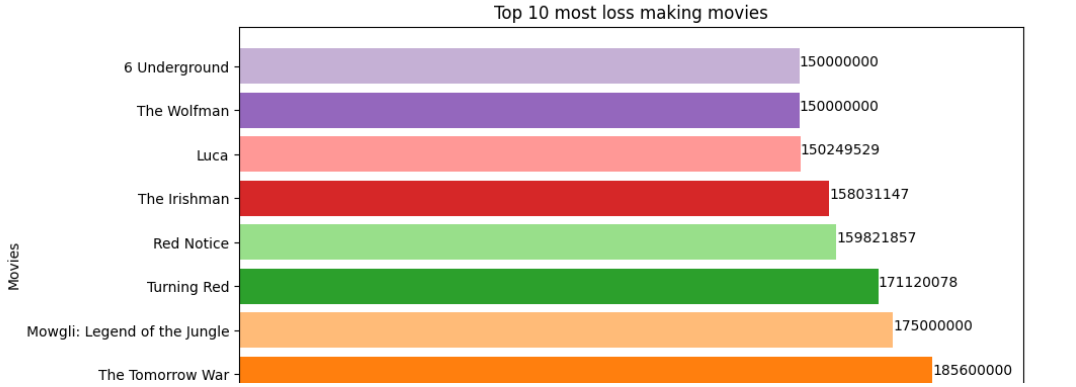
In [31]:

```
fig, axs = plt.subplots(4,1,figsize=(10,28))

col_map = plt.get_cmap('tab20')
axs[0].barh(max_budget['title'], max_budget['budget'], color=col_map.colors)
axs[0].set_title('Top 10 highest budget movies')
axs[0].set_xlabel('Budget')
axs[0].set_ylabel('Movies')
for i in range(len(max_budget)):
    axs[0].text(max_budget['budget'].iloc[i], max_budget['title'].iloc[i], max_bud

col_map = plt.get_cmap('tab10')
axs[1].barh(max_revenue['title'], max_revenue['revenue'], color=col_map.colors)
axs[1].set_title('Top 10 highest revenue movies')
axs[1].set_xlabel('Revenue')
axs[1].set_ylabel('Movies')
for i in range(len(max_revenue)):
    axs[1].text(max_revenue['revenue'].iloc[i], max_revenue['title'].iloc[i], max_

col_map = plt.get_cmap('tab20c')
axs[2].barh(max_profit['title'], max_profit['profit'], color=col_map.colors)
axs[2].set_title('Top 10 most profitable movies')
axs[2].set_xlabel('Profit')
axs[2].set_ylabel('Movies')
for i in range(len(max_profit)):
    axs[2].text(max_profit['profit'].iloc[i], max_profit['title'].iloc[i], max_pro

col_map = plt.get_cmap('tab20')
axs[3].barh(max_loss['title'], max_loss['loss'], color=col_map.colors)
axs[3].set_title('Top 10 most loss making movies')
axs[3].set_xlabel('Loss')
axs[3].set_ylabel('Movies')
for i in range(len(max_loss)):
    axs[3].text(max_loss['loss'].iloc[i], max_loss['title'].iloc[i], max_loss['los

plt.show()
```

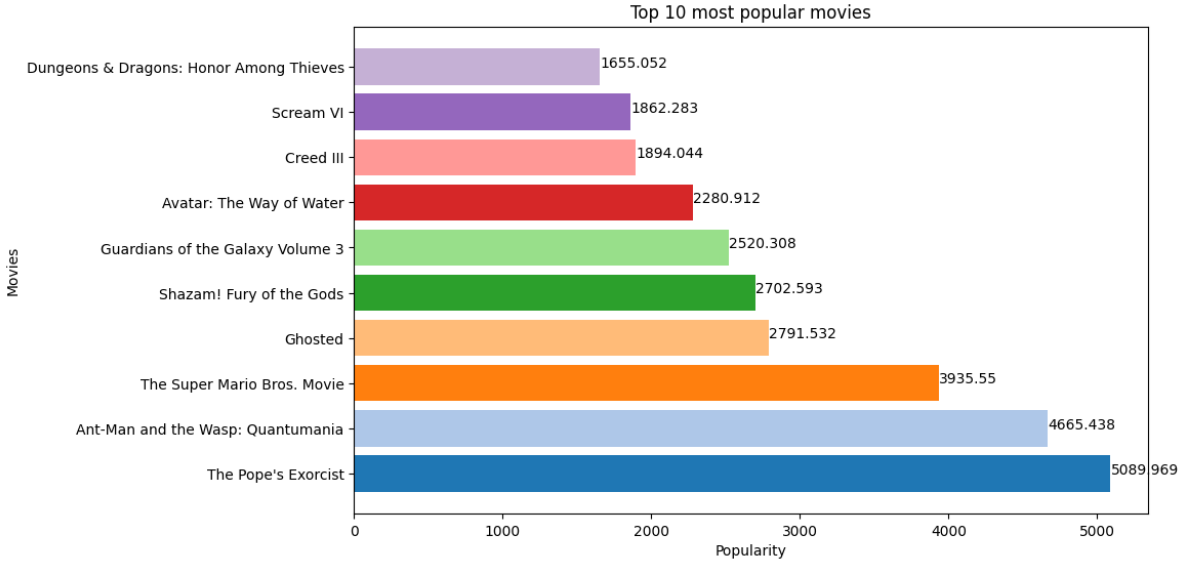

Top 10 highest budget movies
The Lion King
Superman Returns
Justice League
Avengers: Infinity War
Pirates of the Caribbean: At World's End
Avengers: Endgame
Avengers: Age of Ultron
Pirates of the Caribbean: On Stranger Tides
Avatar: The Way of Water
Operation Red Sea
260000000
270000000
300000000
300000000
300000000
356000000
365000000
379000000
460000000
579330426
Movies
Budget
0
1
2
3
4
5
6
1e8

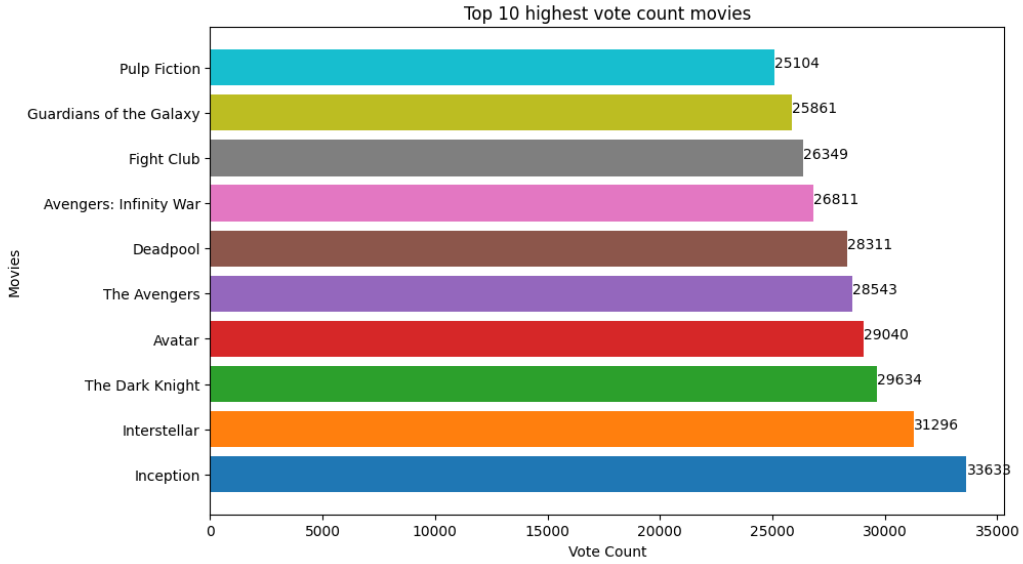

Top 10 highest revenue movies
The Avengers
The Lion King
Jurassic World
Spider-Man: No Way Home
Avengers: Infinity War
Star Wars: The Force Awakens
Titanic
Avatar: The Way of Water
Avengers: Endgame
Avatar
1518815515
1663075401
1671537444
1921847111
2046239637
2068223624
2264162353
2319331580
2794731755
2923706026
Movies
Revenue
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1e9

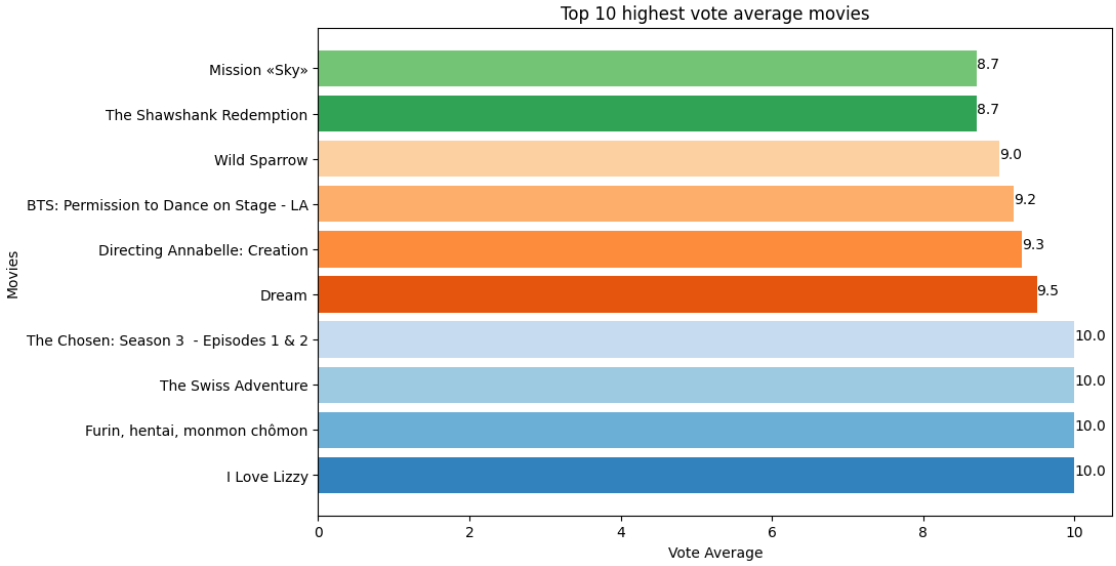

Top 10 most profitable movies
Furious 7
The Lion King
Jurassic World
Spider-Man: No Way Home
Avengers: Infinity War
Star Wars: The Force Awakens
Avatar: The Way of Water
Titanic
Avengers: Endgame
Avatar
1325341399
1403075401
1521537444
1721847111
1746239637
1823223624
1859331580
2064162353
2438731755
2686706026
Movies
Profit
0.0
0.5
1.0
1.5
2.0
2.5
1e9


Top 10 most loss making movies
6 Underground
The Wolfman
Luca
The Irishman
Red Notice
Turning Red
Mowgli: Legend of the Jungle
The Tomorrow War
150000000
150000000
150249529
158031147
159821857
171120078
175000000
185600000
Movies

In [32]:
```
fig, axs = plt.subplots(3,1,figsize=(10,21))

col_map = plt.get_cmap('tab20')
axs[0].barh(max_popularity['title'], max_popularity['popularity'], color=col_map.c
axs[0].set_title('Top 10 most popular movies')
axs[0].set_xlabel('Popularity')
axs[0].set_ylabel('Movies')
for i in range(len(max_popularity)):
    axs[0].text(max_popularity['popularity'].iloc[i], max_popularity['title'].iloc

col_map = plt.get_cmap('tab10')
axs[1].barh(max_vc['title'], max_vc['vote_count'], color=col_map.colors)
axs[1].set_title('Top 10 highest vote count movies')
axs[1].set_xlabel('Vote Count')
axs[1].set_ylabel('Movies')
for i in range(len(max_vc)):
    axs[1].text(max_vc['vote_count'].iloc[i], max_vc['title'].iloc[i], max_vc['vot

col_map = plt.get_cmap('tab20c')
axs[2].barh(max_va['title'], max_va['vote_average'], color=col_map.colors)
axs[2].set_title('Top 10 highest vote average movies')
axs[2].set_xlabel('Vote Average')
axs[2].set_ylabel('Movies')
for i in range(len(max_va)):
    axs[2].text(max_va['vote_average'].iloc[i], max_va['title'].iloc[i], max_va['v

plt.show()
```







Count of Movies wrt Language in Dataset

In [33]:
```
df['original_language'].value_counts()
```

Out[33]:
```
English             6962
Japanese             648
Spanish              289
French               283
Korean               182
Italian              143
cn                   119
Chinese              108
German                78
Russian               65
Tagalog               35
Hindi                 33
Norwegian             31
Danish                30
Polish                26
Portuguese            24
Thai                  23
Swedish               23
Dutch                 22
Indonesian            16
Turkish               14
Tamil                  8
Telugu                 7
Finnish                6
Greek                  4
Romanian               4
Ukrainian              4
Arabic                 3
Basque                 3
Hungarian              3
Persian                3
Galician               2
Khmer                  2
Kannada                2
Czech                  2
Malayalam              2
Serbian                2
Hebrew                 2
Catalan                2
Icelandic              2
Estonian               1
Vietnamese             1
Norwegian Bokmal       1
Irish                  1
Macedonian             1
Dzongkha               1
Name: original_language, dtype: int64
```

No. of all unique genres in Dataset

In [34]:
```
genres_col = df['genres']
# Empty set to store unique genres
unique_genres = set()
# Iterate over each row in the genres column
for genres_list in genres_col:
    genres = eval(genres_list)  # Convert the string representation of list to a l
    unique_genres.update(genres)  # Add the genres to the set

len(unique_genres)
```

Out[34]:
```
19
```

5 Most Popular Movies of Every Genre

In [35]:
```
for genre in unique_genres:
    movies = df[df['genres'].str.contains(genre)].sort_values('popularity', ascend
    movie_title = movies['title']
    print(genre)
    for i ,title in enumerate(movie_title):
        print(i+1, title)
    print("\n")
```

```
Fantasy
1 The Super Mario Bros. Movie
2 Shazam! Fury of the Gods
3 Dungeons & Dragons: Honor Among Thieves
4 Peter Pan & Wendy
5 Puss in Boots: The Last Wish


Documentary
1 Money Shot: The Pornhub Story
2 Orgasm Inc: The Story of OneTaste
3 Cocaine Bear: The True Story
4 Harry Potter 20th Anniversary: Return to Hogwarts
5 Melody Makers


Science Fiction
1 Ant-Man and the Wasp: Quantumania
2 Guardians of the Galaxy Volume 3
3 Avatar: The Way of Water
4 65
5 Black Panther: Wakanda Forever


Thriller
1 The Pope's Exorcist
2 Scream VI
3 AKA
4 John Wick: Chapter 4
5 Cocaine Bear


History
1 The Last Kingdom: Seven Kings Must Die
2 The Woman King
3 3-D Sex and Zen: Extreme Ecstasy
4 Medieval
5 Devotion


Western
1 Ghosts of the Ozarks
2 Django Unchained
3 Spirit: Stallion of the Cimarron
4 Paws of Fury: The Legend of Hank
5 Tom and Jerry Cowboy Up!


Comedy
1 The Super Mario Bros. Movie
2 Ghosted
3 Shazam! Fury of the Gods
4 Dungeons & Dragons: Honor Among Thieves
5 Puss in Boots: The Last Wish


Music
1 Encanto at the Hollywood Bowl
2 Guillermo del Toro's Pinocchio
3 Lyle, Lyle, Crocodile
4 Blue's Big City Adventure
5 Coco
```

```
TV Movie
1 Girl in the Basement
2 Teen Wolf: The Movie
3 Dungeons & Dragons: Wrath of the Dragon God
4 Monster High: The Movie
5 Miraculous World: New York, United HeroeZ


Animation
1 The Super Mario Bros. Movie
2 Puss in Boots: The Last Wish
3 Justice League x RWBY: Super Heroes & Huntsmen, Part One
4 Mummies
5 That Time I Got Reincarnated as a Slime the Movie: Scarlet Bond


Family
1 The Super Mario Bros. Movie
2 Peter Pan & Wendy
3 Puss in Boots: The Last Wish
4 Pirates Down the Street II: The Ninjas from Across
5 Mummies


Adventure
1 Ant-Man and the Wasp: Quantumania
2 The Super Mario Bros. Movie
3 Shazam! Fury of the Gods
4 Guardians of the Galaxy Volume 3
5 Avatar: The Way of Water


Crime
1 AKA
2 John Wick: Chapter 4
3 Murder Mystery 2
4 Cocaine Bear
5 Kill Boksoon


Mystery
1 The Pope's Exorcist
2 Scream VI
3 Clock
4 Invitation to a Murder
5 Batman: The Doom That Came to Gotham


Horror
1 The Pope's Exorcist
2 Scream VI
3 Evil Dead Rise
4 The Communion Girl
5 Winnie the Pooh: Blood and Honey


Drama
1 Creed III
2 The Last Kingdom: Seven Kings Must Die
3 Puss in Boots: The Last Wish
4 The Park
5 Marcel the Shell with Shoes On
```

```
Romance
1 Ghosted
2 Shotgun Wedding
3 The Quintessential Quintuplets Movie
4 Unhappily Ever After
5 The Forbidden Legend: Sex & Chopsticks 2


Action
1 Ant-Man and the Wasp: Quantumania
2 Ghosted
3 Shazam! Fury of the Gods
4 Guardians of the Galaxy Volume 3
5 Avatar: The Way of Water


War
1 The Last Kingdom: Seven Kings Must Die
2 Sisu
3 Sniper: The White Raven
4 Gold Run
5 Devotion
```

5 Most Profitable Movies of Every Genre

In [36]:
```
for genre in unique_genres:
    movies = df[df['genres'].str.contains(genre)].sort_values('profit', ascending=
    movie_title = movies['title']
    print(genre)
    for i ,title in enumerate(movie_title):
        print(i+1, title)
    print("\n")
```

```
Fantasy
1 Avatar
2 Star Wars: The Force Awakens
3 Frozen II
4 Harry Potter and the Deathly Hallows: Part 2
5 Frozen


Documentary
1 Fahrenheit 9/11
2 This Is It
3 Jackass 3D
4 Jackass Forever
5 Jackass Number Two


Science Fiction
1 Avatar
2 Avengers: Endgame
3 Avatar: The Way of Water
4 Star Wars: The Force Awakens
5 Avengers: Infinity War


Thriller
1 Jurassic World
2 Furious 7
3 Jurassic World: Fallen Kingdom
4 Joker
5 The Fate of the Furious


History
1 Bohemian Rhapsody
2 Full River Red
3 The Battle at Lake Changjin: Water Gate Bridge
4 Saving Private Ryan
5 The King's Speech


Western
1 Dances with Wolves
2 The Revenant
3 Django Unchained
4 True Grit
5 City Slickers


Comedy
1 Frozen II
2 Minions
3 The Super Mario Bros. Movie
4 Despicable Me 3
5 Jumanji: Welcome to the Jungle


Music
1 Bohemian Rhapsody
2 Coco
3 Sing
4 La La Land
5 A Star Is Born
```

```
TV Movie
1 High School Musical 2
2 A Year-End Medley
3 Stargate: The Ark of Truth
4 Teen Wolf: The Movie
5 Under the Sea: A Descendants Story


Animation
1 The Lion King
2 Frozen II
3 Frozen
4 Minions
5 Incredibles 2


Family
1 The Lion King
2 Frozen II
3 Frozen
4 Beauty and the Beast
5 Minions


Adventure
1 Avatar
2 Avengers: Endgame
3 Avatar: The Way of Water
4 Star Wars: The Force Awakens
5 Avengers: Infinity War


Crime
1 Furious 7
2 Joker
3 The Fate of the Furious
4 The Dark Knight Rises
5 The Dark Knight


Mystery
1 Harry Potter and the Order of the Phoenix
2 Full River Red
3 The Da Vinci Code
4 The Sixth Sense
5 The Batman


Horror
1 It
2 Jaws
3 The Exorcist
4 It Chapter Two
5 The Meg


Drama
1 Titanic
2 The Lion King
3 Top Gun: Maverick
4 Joker
5 Bohemian Rhapsody
```

```
Romance
1 Titanic
2 Beauty and the Beast
3 Aladdin
4 Shrek 2
5 The Twilight Saga: Breaking Dawn - Part 2


Action
1 Avatar
2 Avengers: Endgame
3 Avatar: The Way of Water
4 Star Wars: The Force Awakens
5 Avengers: Infinity War


War
1 Wolf Warrior 2
2 Dunkirk
3 American Sniper
4 The Battle at Lake Changjin: Water Gate Bridge
5 Saving Private Ryan
```

Top 5 production companies with maximum movie count

In [37]:
```
companies_column = df['production_companies']

# Create an empty dictionary to store company names and their movie counts
company_counts = {}

# Iterate over each row in the companies column
for companies_list in companies_column:
    companies = eval(companies_list)  # Convert the string representation of list t
    for company in companies:
        if company in company_counts:
            company_counts[company] += 1  # Increment the movie count
        else:
            company_counts[company] = 1  # Add the company with initial movie count

sorted_companies = sorted(company_counts.items(), key=lambda x: x[1], reverse=True

top_5_companies = sorted_companies[:5]

for company, count in top_5_companies:
    print(company, ": ", count)
```

```
Warner Bros. Pictures :  488
Universal Pictures :  470
Paramount :  374
Columbia Pictures :  360
20th Century Fox :  348
```

Most popular movie in every 5 years with their popularity

In [38]:
```
# Convert the release_date column to datetime
df['release_date'] = pd.to_datetime(df['release_date'])

# Create a new column for the release half_decade
df['release_hd'] = (df['release_date'].dt.year // 5) * 5
```

```
popular_movies = df.groupby('release_hd').apply(lambda x: x.nlargest(1, 'popularity

for i, r in popular_movies.iterrows():
    print(f"Year: {r['release_hd']}-{r['release_hd']+4}")
    print("Movie:", r['title'])
    print("Popularity:", r['popularity'])
    print("\n")
```

```
Year: 1900-1904
Movie: A Trip to the Moon
Popularity: 14.584


Year: 1920-1924
Movie: Nosferatu
Popularity: 18.37


Year: 1925-1929
Movie: Metropolis
Popularity: 17.496


Year: 1930-1934
Movie: Baby Face
Popularity: 35.37


Year: 1935-1939
Movie: Snow White and the Seven Dwarfs
Popularity: 61.616


Year: 1940-1944
Movie: Bambi
Popularity: 43.576


Year: 1945-1949
Movie: Samson and Delilah
Popularity: 27.865


Year: 1950-1954
Movie: Cinderella
Popularity: 73.639


Year: 1955-1959
Movie: Sleeping Beauty
Popularity: 37.958


Year: 1960-1964
Movie: One Hundred and One Dalmatians
Popularity: 47.549


Year: 1965-1969
Movie: The Jungle Book
Popularity: 52.291


Year: 1970-1974
Movie: The Godfather
Popularity: 113.216


Year: 1975-1979
Movie: Star Wars
Popularity: 90.988
```

```
Year: 1980-1984
Movie: Oscenità
Popularity: 129.957


Year: 1985-1989
Movie: The Little Mermaid
Popularity: 104.76


Year: 1990-1994
Movie: Super Mario Bros.
Popularity: 124.391


Year: 1995-1999
Movie: Titanic
Popularity: 108.782


Year: 2000-2004
Movie: Shrek
Popularity: 152.468


Year: 2005-2009
Movie: The Forbidden Legend: Sex & Chopsticks 2
Popularity: 255.11


Year: 2010-2014
Movie: Guardians of the Galaxy
Popularity: 255.418


Year: 2015-2019
Movie: Guardians of the Galaxy Vol. 2
Popularity: 492.95


Year: 2020-2024
Movie: The Pope's Exorcist
Popularity: 5089.969
```

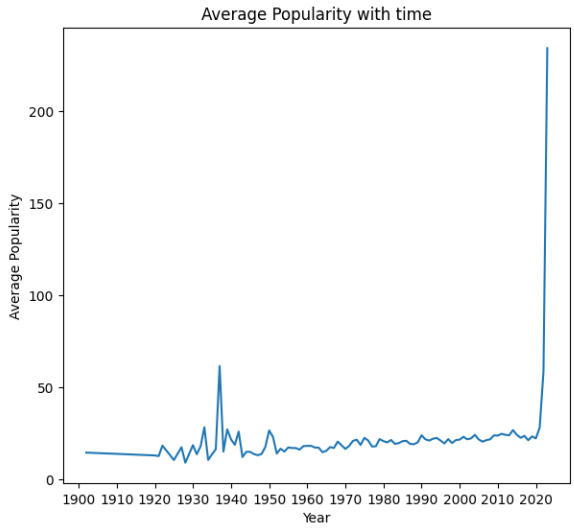
Average Popularity by Year

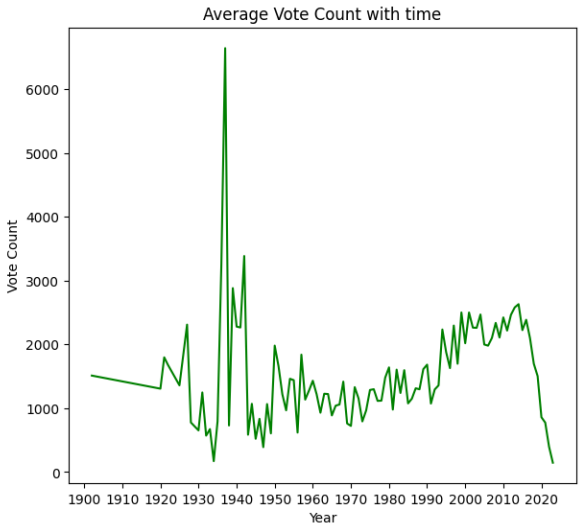
In [39]:
```
df['year'] = df['release_date'].dt.year
avg_pop = df.groupby('year')['popularity'].mean()
```

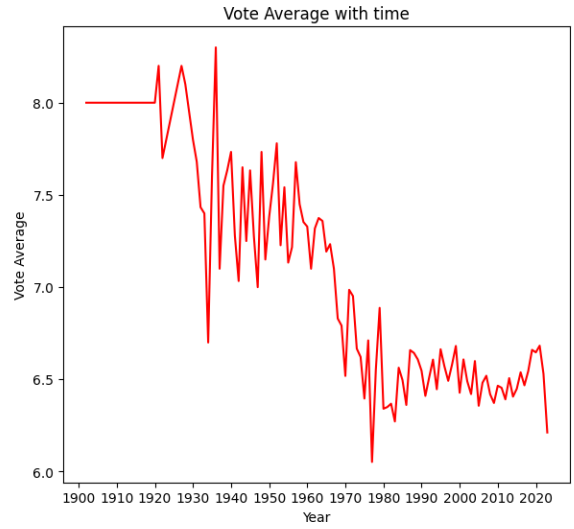
Visualization of Categories by Year

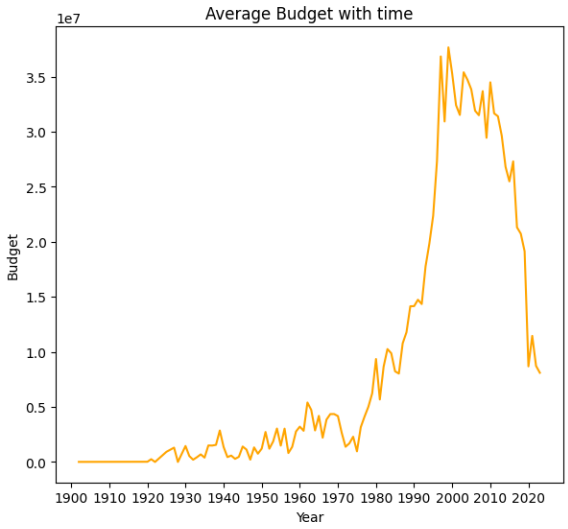
In [40]:
```
fig, axs = plt.subplots(3,2,figsize=(15,21))

axs[0,0].plot(avg_pop.index, avg_pop.values)
axs[0,0].set_title('Average Popularity with time')
axs[0,0].set_xlabel('Year')
axs[0,0].set_ylabel('Average Popularity')
axs[0,0].set_xticks(np.arange(1900, 2030, step=10),rotation=45)
```

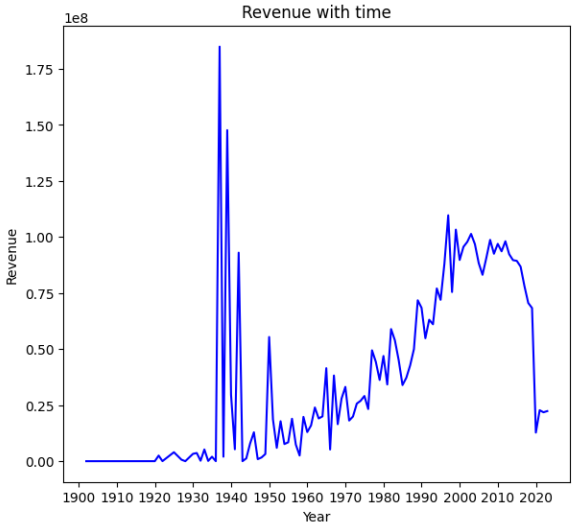
```
axs[0,1].plot(df.groupby('year')['vote_count'].mean(),color='green')
axs[0,1].set_title('Average Vote Count with time')
axs[0,1].set_xlabel('Year')
axs[0,1].set_ylabel('Vote Count')
axs[0,1].set_xticks(np.arange(1900, 2030, step=10),rotation=45)

axs[1,0].plot(df.groupby('year')['vote_average'].mean(),color='red')
axs[1,0].set_title('Vote Average with time')
axs[1,0].set_xlabel('Year')
axs[1,0].set_ylabel('Vote Average')
axs[1,0].set_xticks(np.arange(1900, 2030, step=10),rotation=45)

axs[1,1].plot(df.groupby('year')['budget'].mean(), color='orange')
axs[1,1].set_title('Average Budget with time')
axs[1,1].set_xlabel('Year')
axs[1,1].set_ylabel('Budget')
axs[1,1].set_xticks(np.arange(1900, 2030, step=10),rotation=45)

axs[2,0].plot(df.groupby('year')['revenue'].mean(),color='blue')
axs[2,0].set_title('Revenue with time')
axs[2,0].set_xlabel('Year')
axs[2,0].set_ylabel('Revenue')
axs[2,0].set_xticks(np.arange(1900, 2030, step=10),rotation=45)

axs[2,1].plot(df.groupby('year')['profit'].mean(), color='violet')
axs[2,1].set_title('Profit with time')
axs[2,1].set_xlabel('Year')
axs[2,1].set_ylabel('Profit')
axs[2,1].set_xticks(np.arange(1900, 2030, step=10),rotation=45)

plt.show()
```

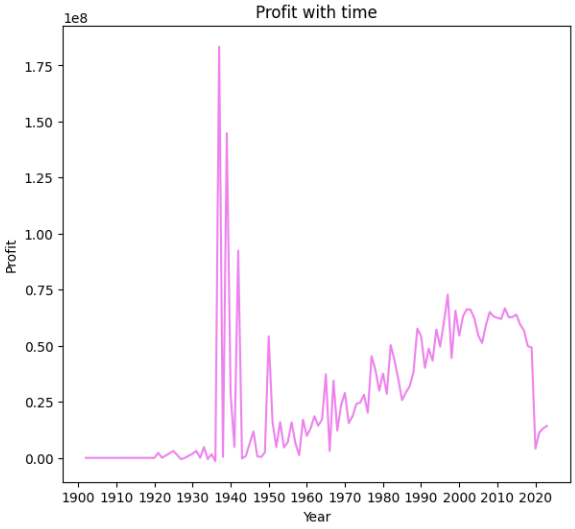








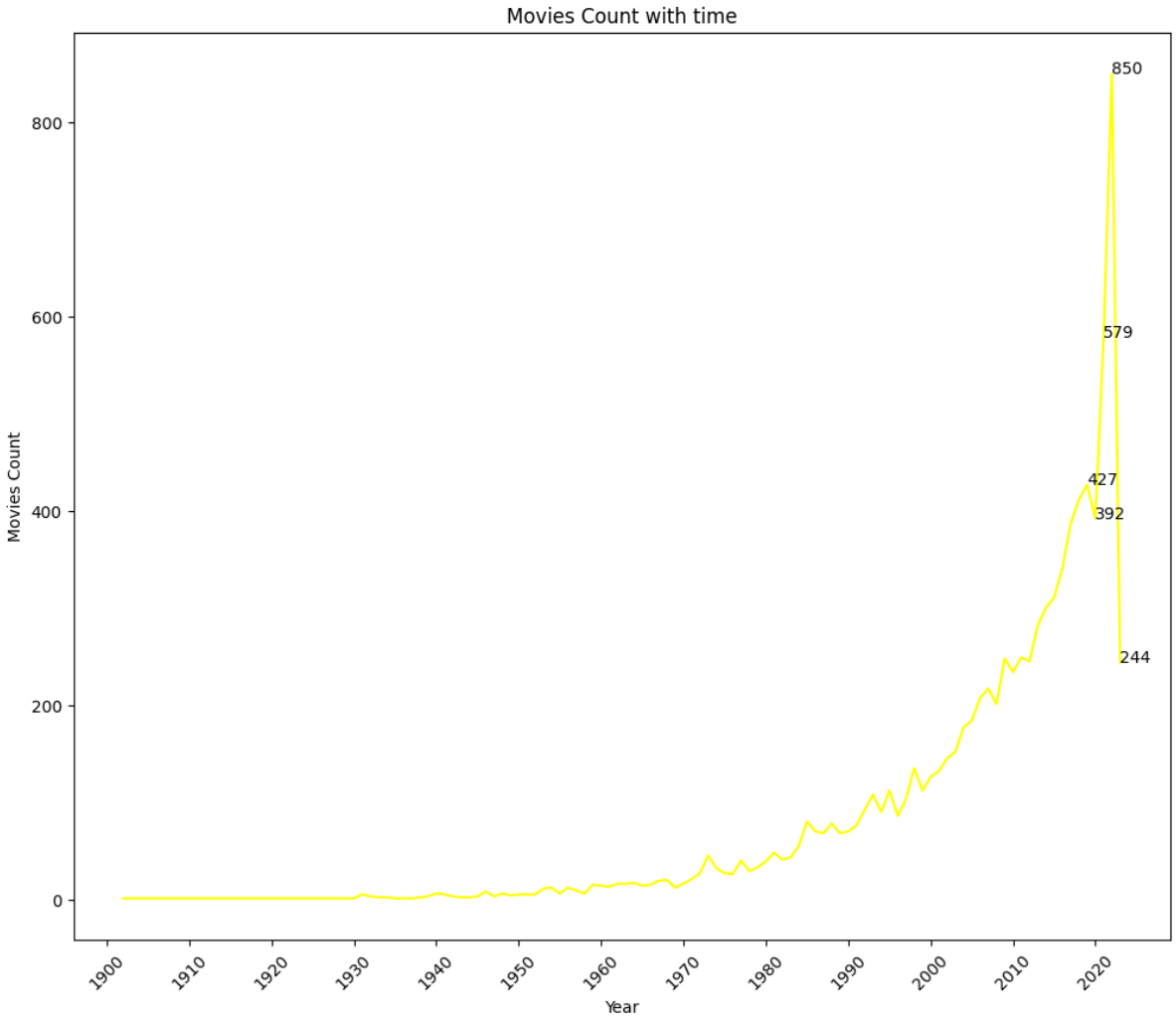




Movie Count with time

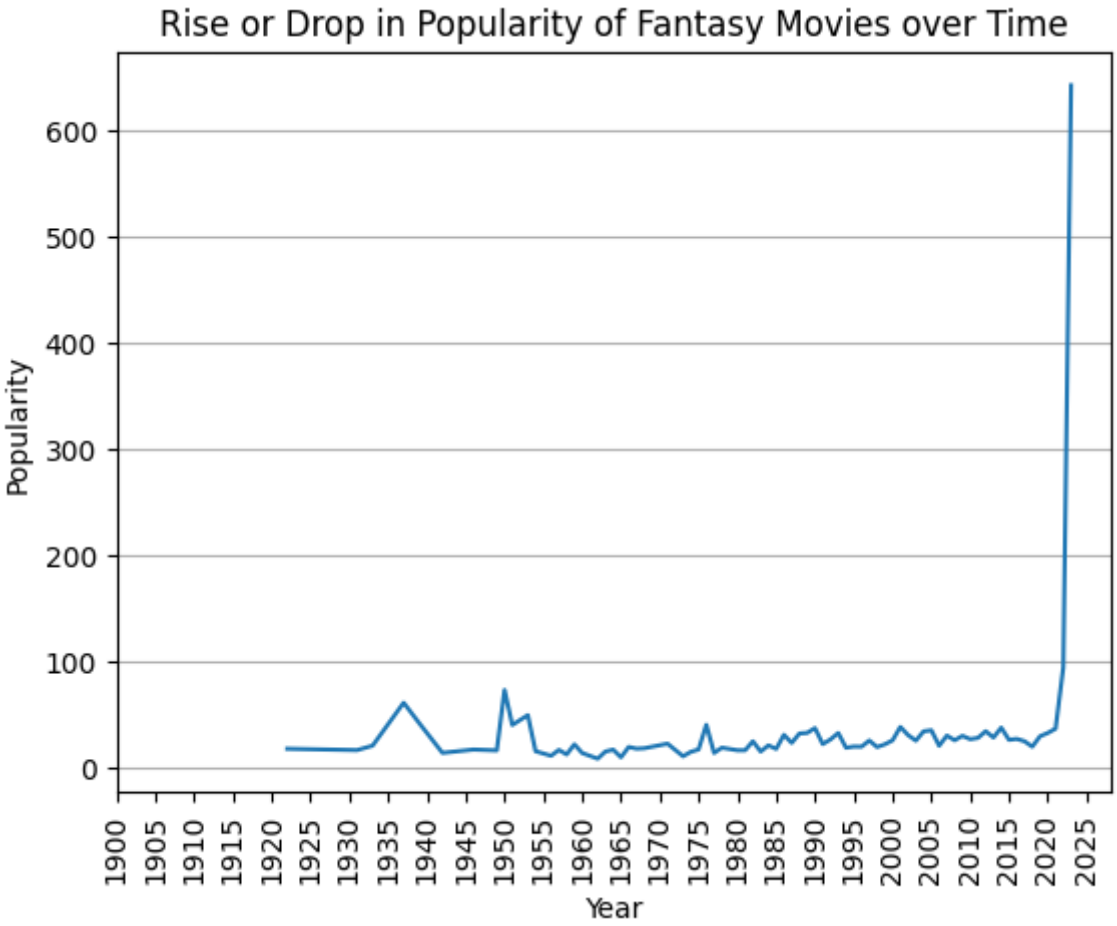
In [41]:
```
movie_count = df['year'].value_counts().sort_index()
fig, ax = plt.subplots(figsize=(12,10))
plt.plot(movie_count, color='yellow')
plt.title('Movies Count with time')
plt.xlabel('Year')
plt.ylabel('Movies Count')
plt.xticks(np.arange(1900, 2030, step=10),rotation=45)
for i in range(5):
```

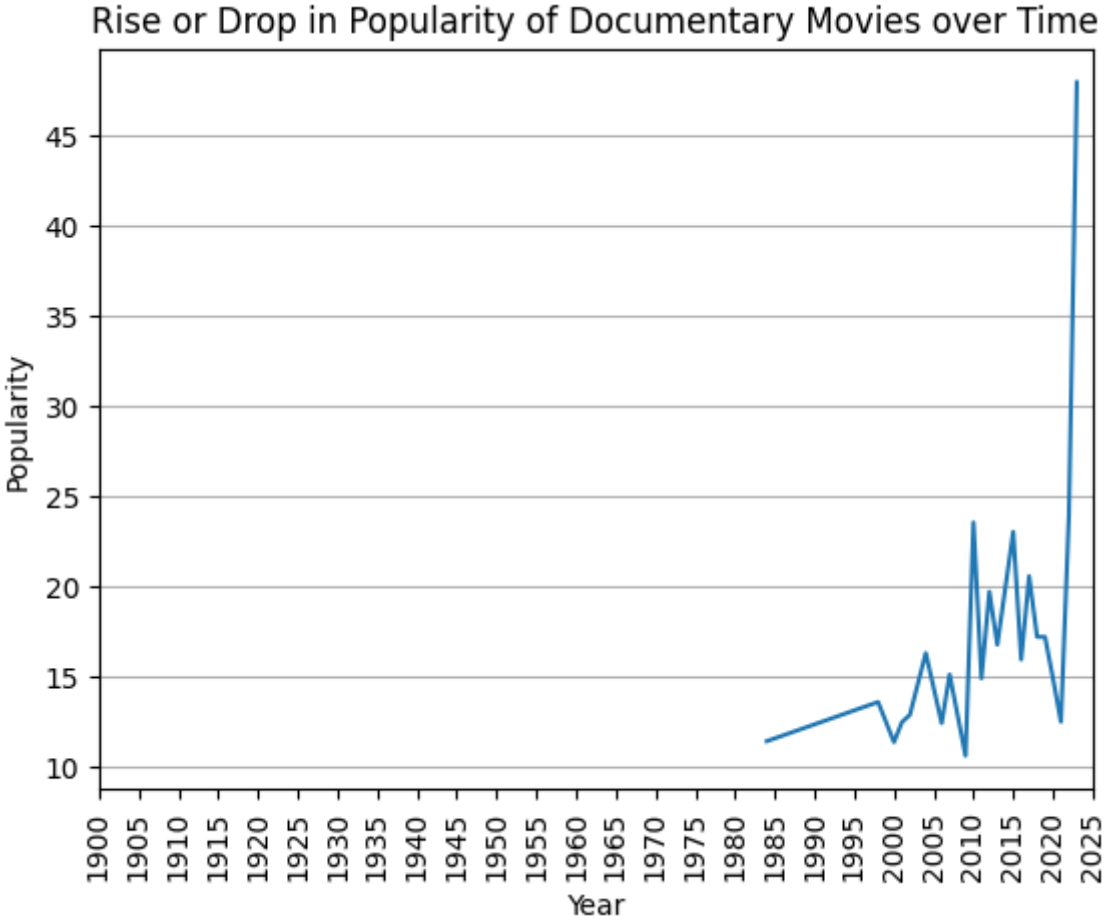
```
    ax.text(movie_count.index[-5+i], movie_count.values[-5+i], movie_count.values[
plt.show()
```

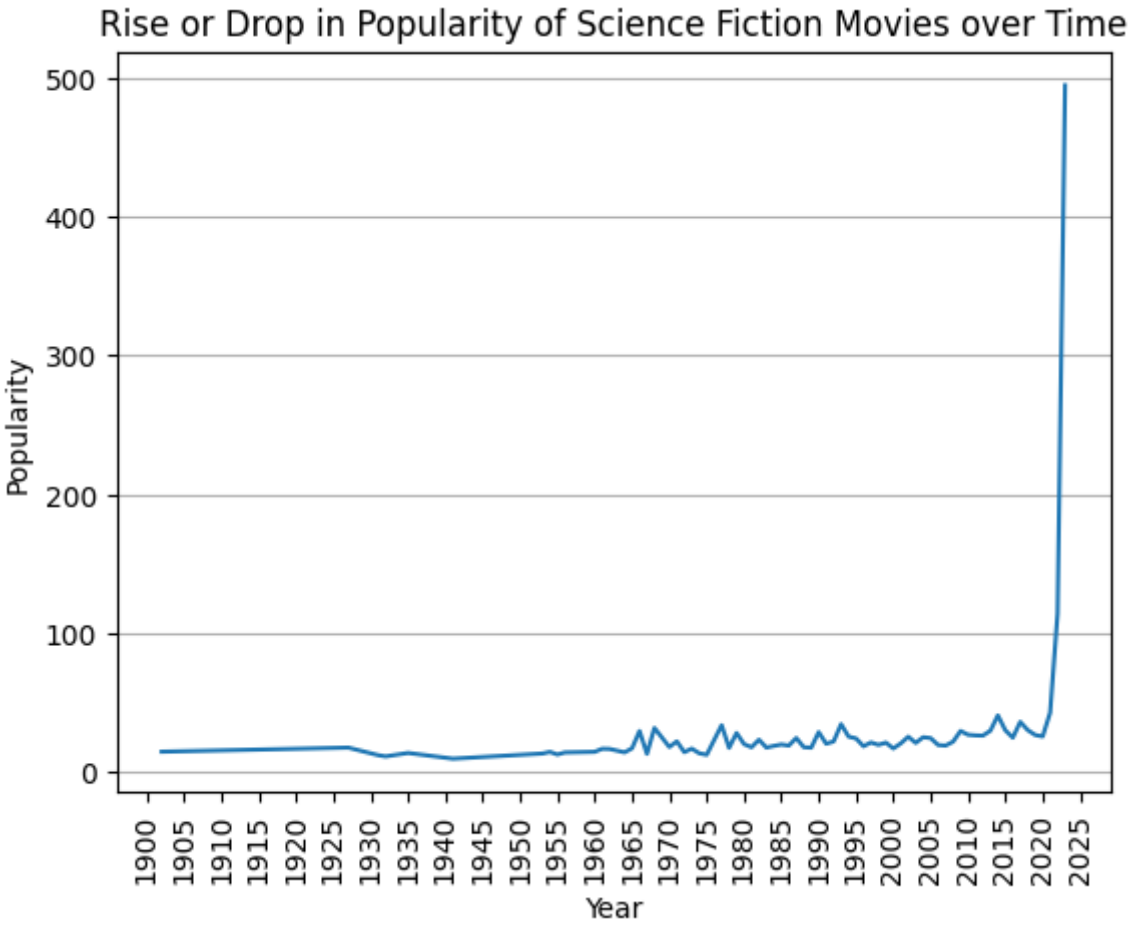


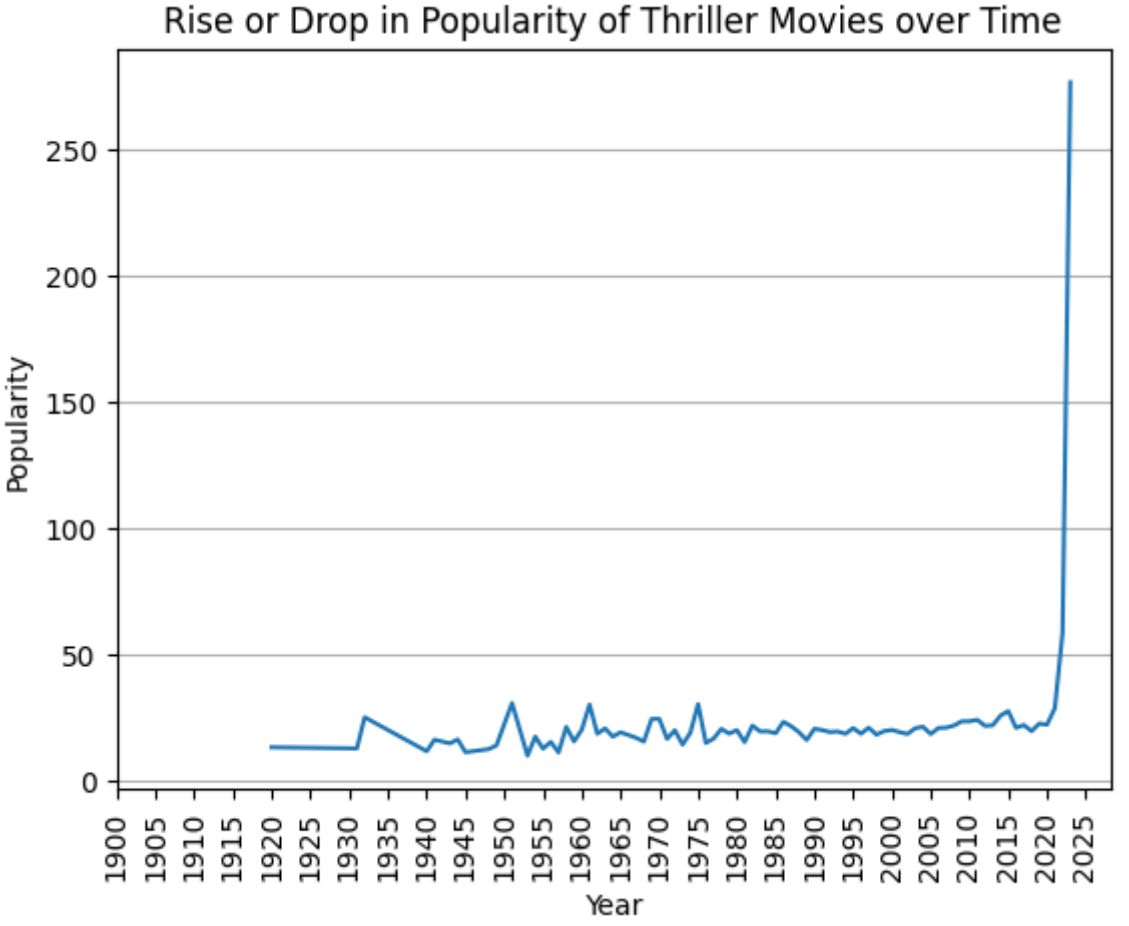
Visualization of Rise and Drop of Popularity with different genres over the time

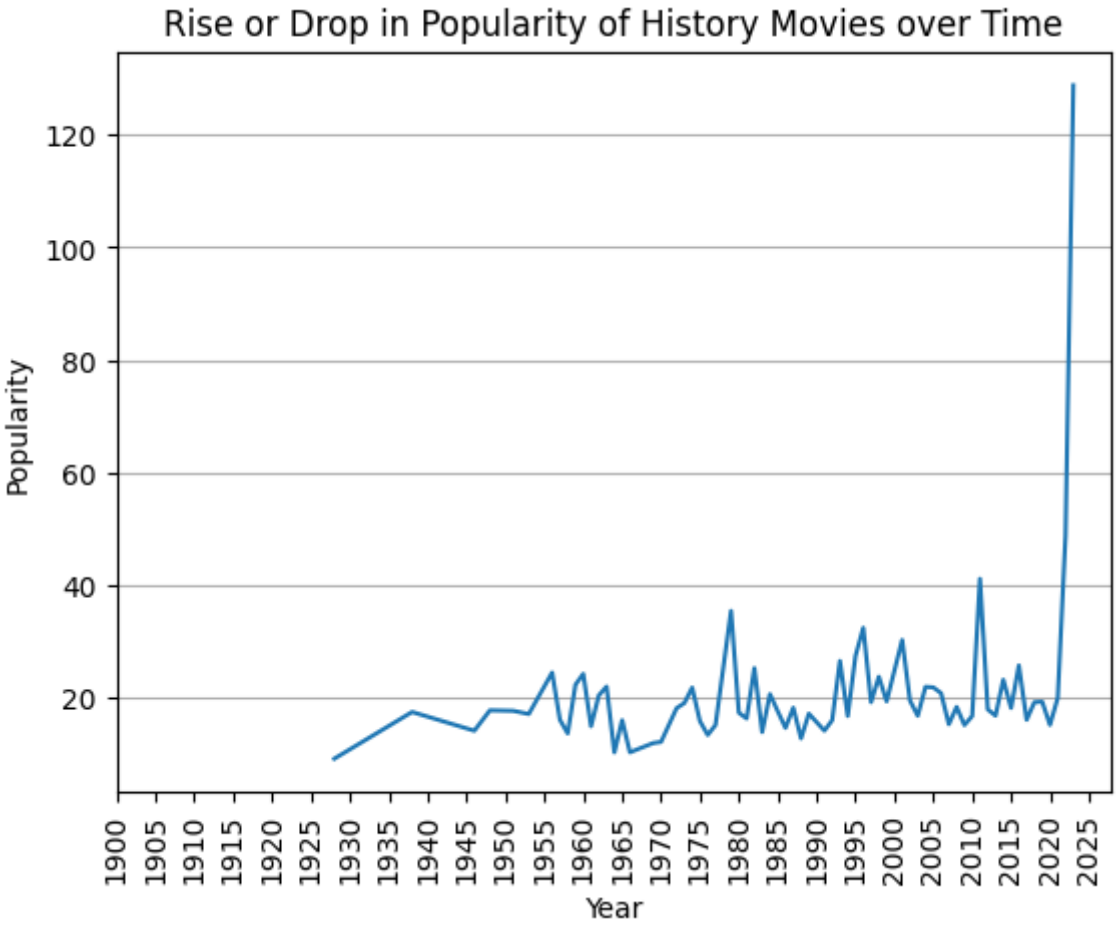
In [42]:
```
for genre in unique_genres:
    movies = df[df['genres'].str.contains(genre)]
    avg_popularity = movies.groupby('year')['popularity'].mean()
    plt.plot(avg_popularity.index, avg_popularity.values)
    plt.title('Rise or Drop in Popularity of '+str(genre)+' Movies over Time')
    plt.xlabel('Year')
    plt.ylabel('Popularity')
    plt.xticks(np.arange(1900, 2030, step=5),rotation=90)
    plt.grid(axis='y')
    plt.show()
```

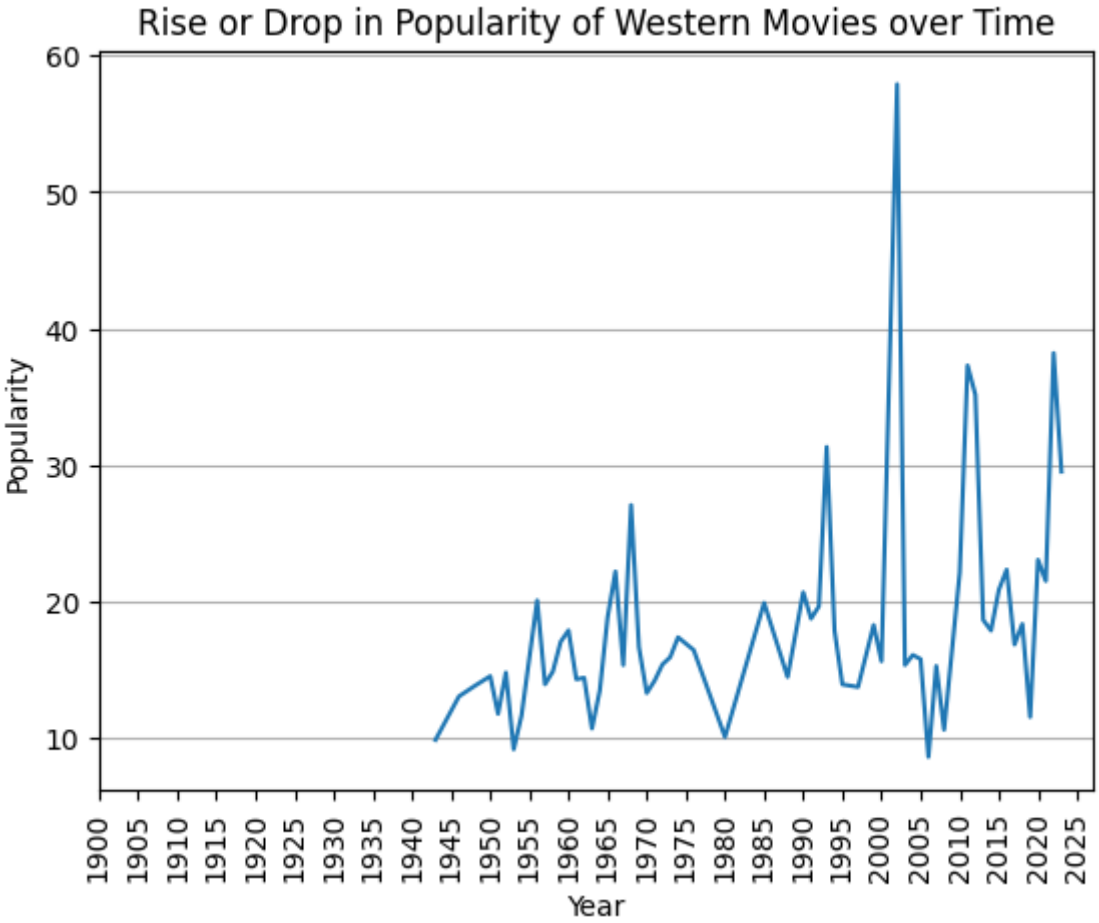

Rise or Drop in Popularity of Fantasy Movies over Time
Popularity
Year
0
100
200
300
400
500
600
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

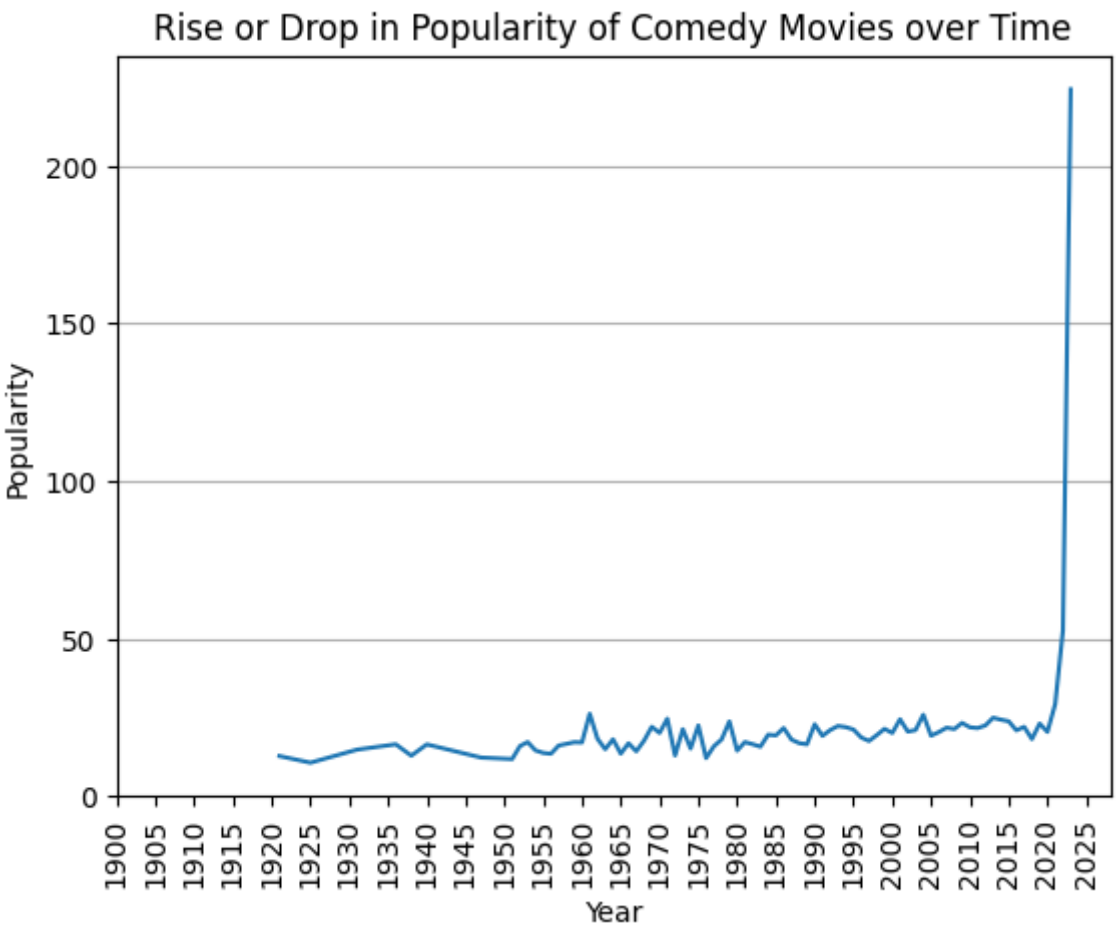

Rise or Drop in Popularity of Documentary Movies over Time
Popularity
Year
10
15
20
25
30
35
40
45
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

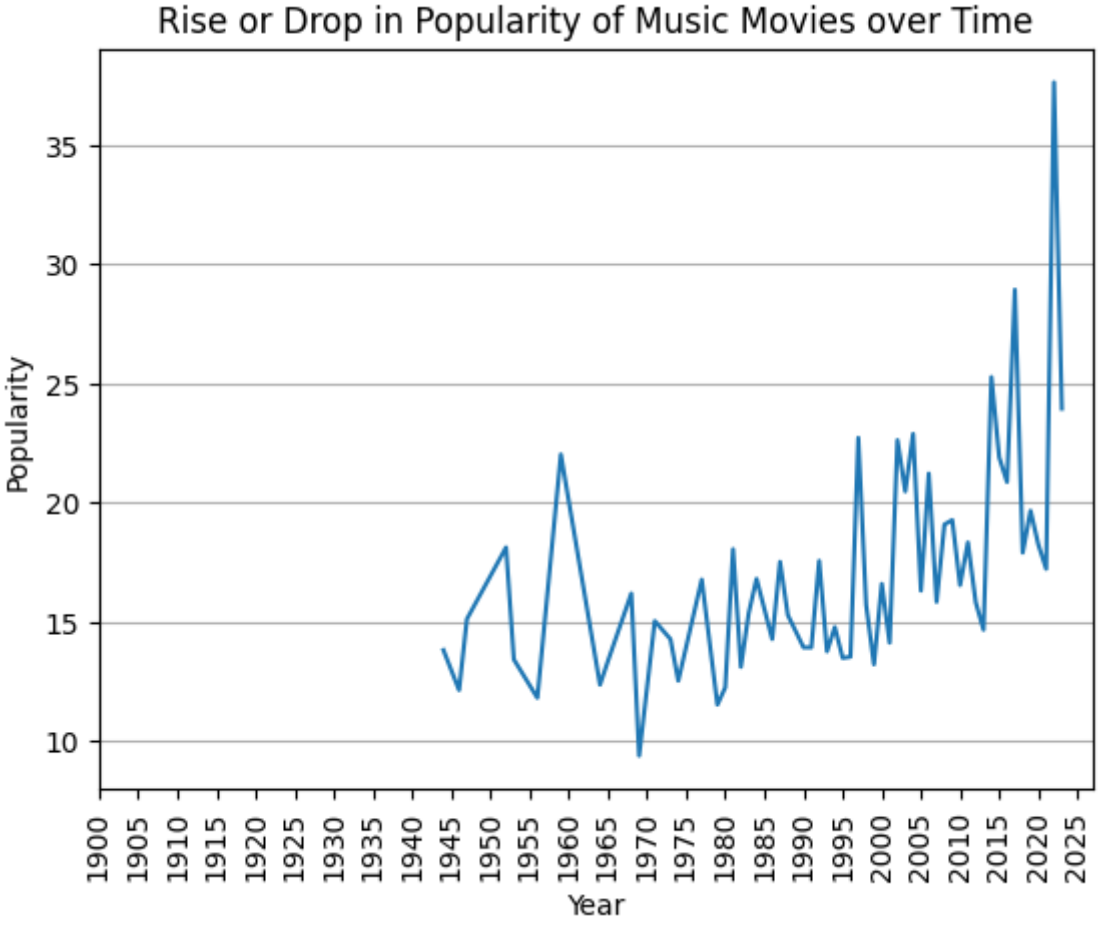

Rise or Drop in Popularity of Science Fiction Movies over Time
Popularity
Year
0
100
200
300
400
500
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

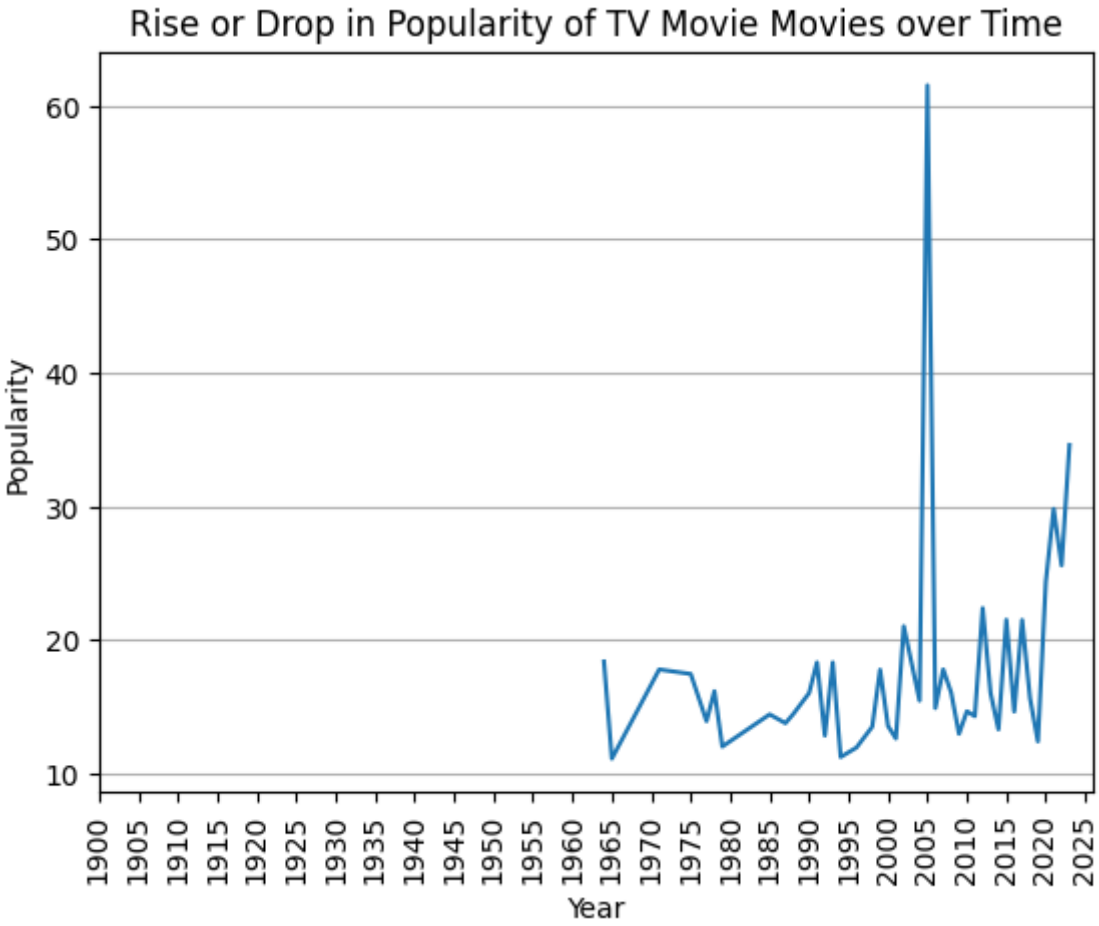

Rise or Drop in Popularity of Thriller Movies over Time
Popularity
Year
0
50
100
150
200
250
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

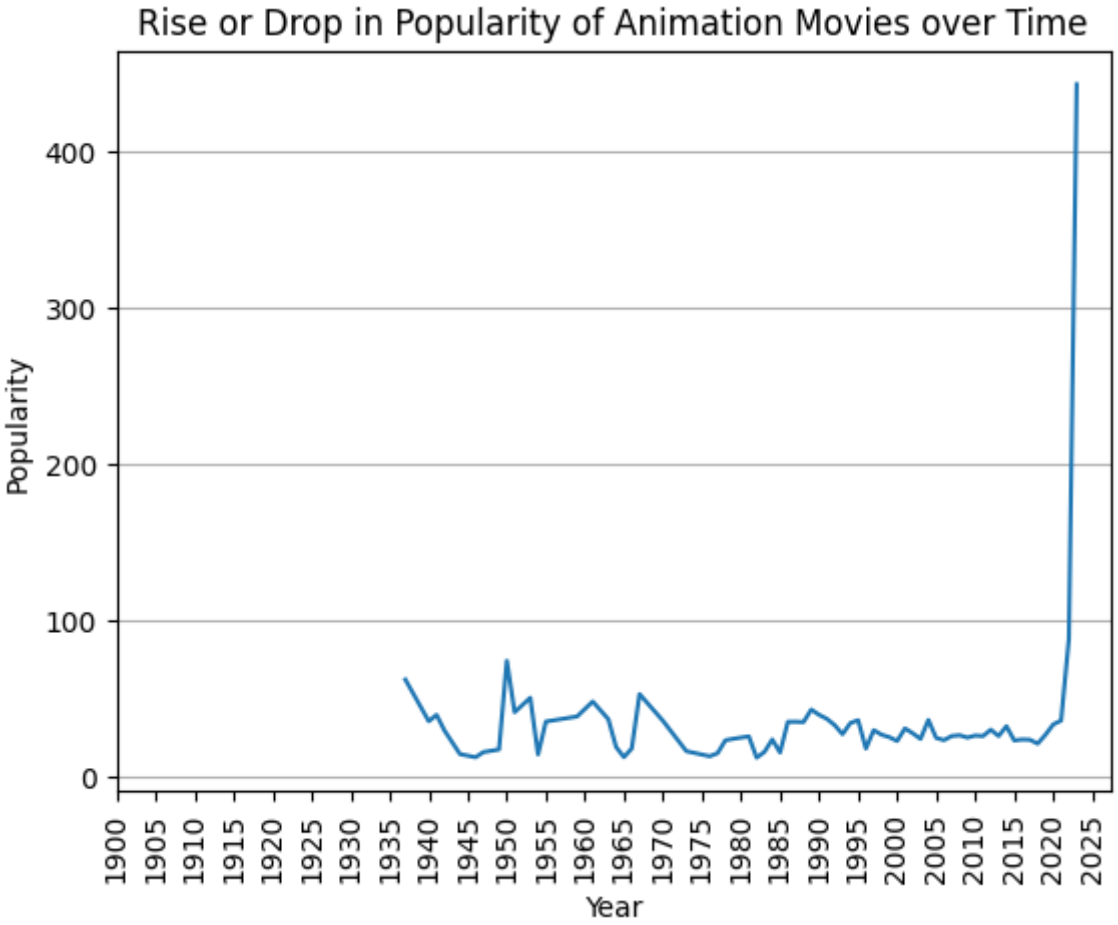

Rise or Drop in Popularity of History Movies over Time
Popularity
Year
120
100
80
60
40
20
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

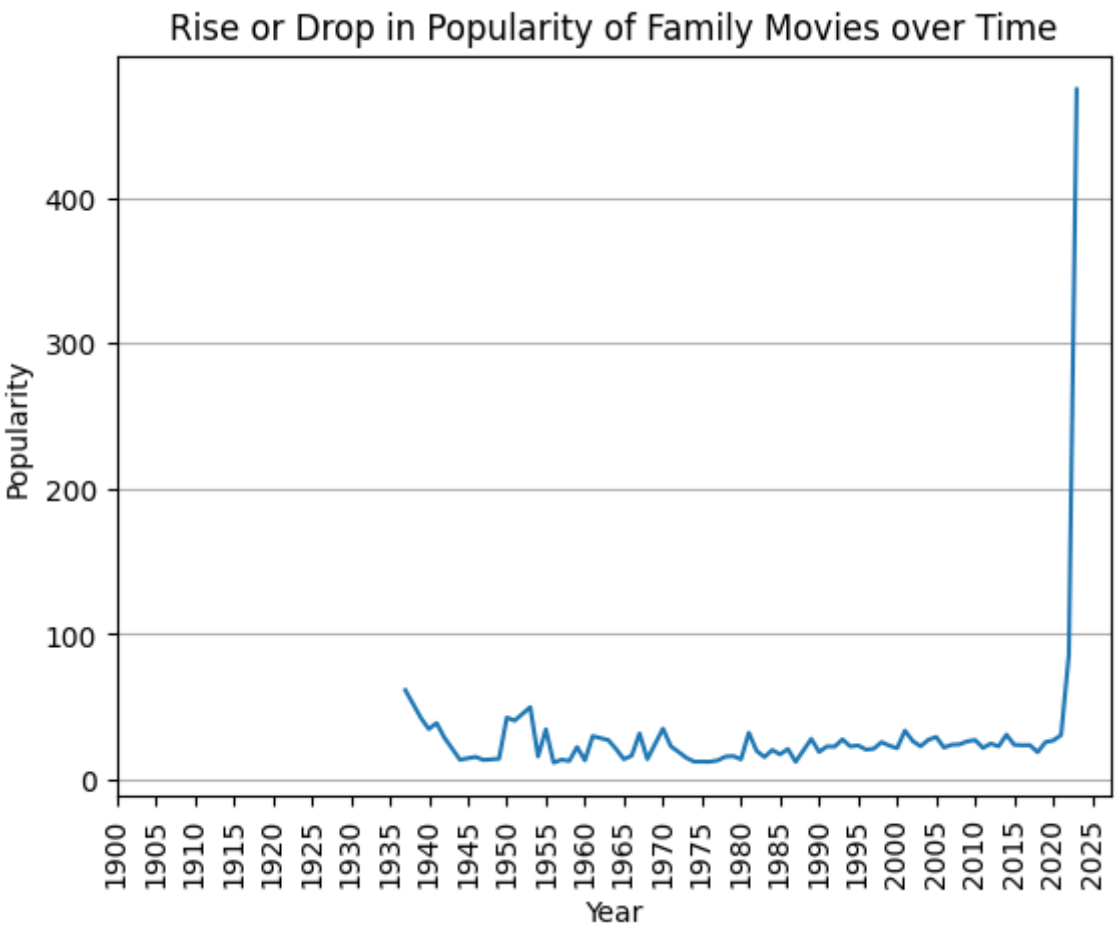

Rise or Drop in Popularity of Western Movies over Time
Popularity
Year
60
50
40
30
20
10
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

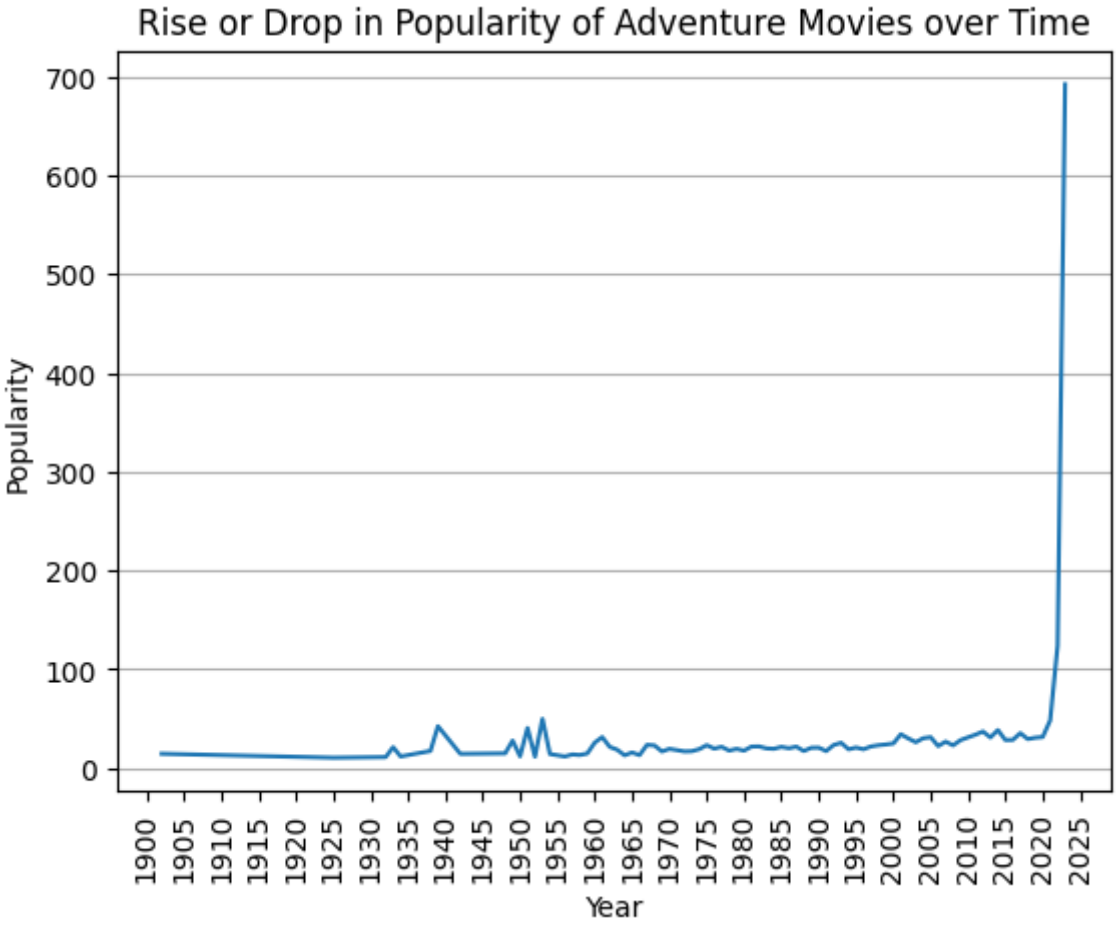

Rise or Drop in Popularity of Comedy Movies over Time
Popularity
Year
0
50
100
150
200
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

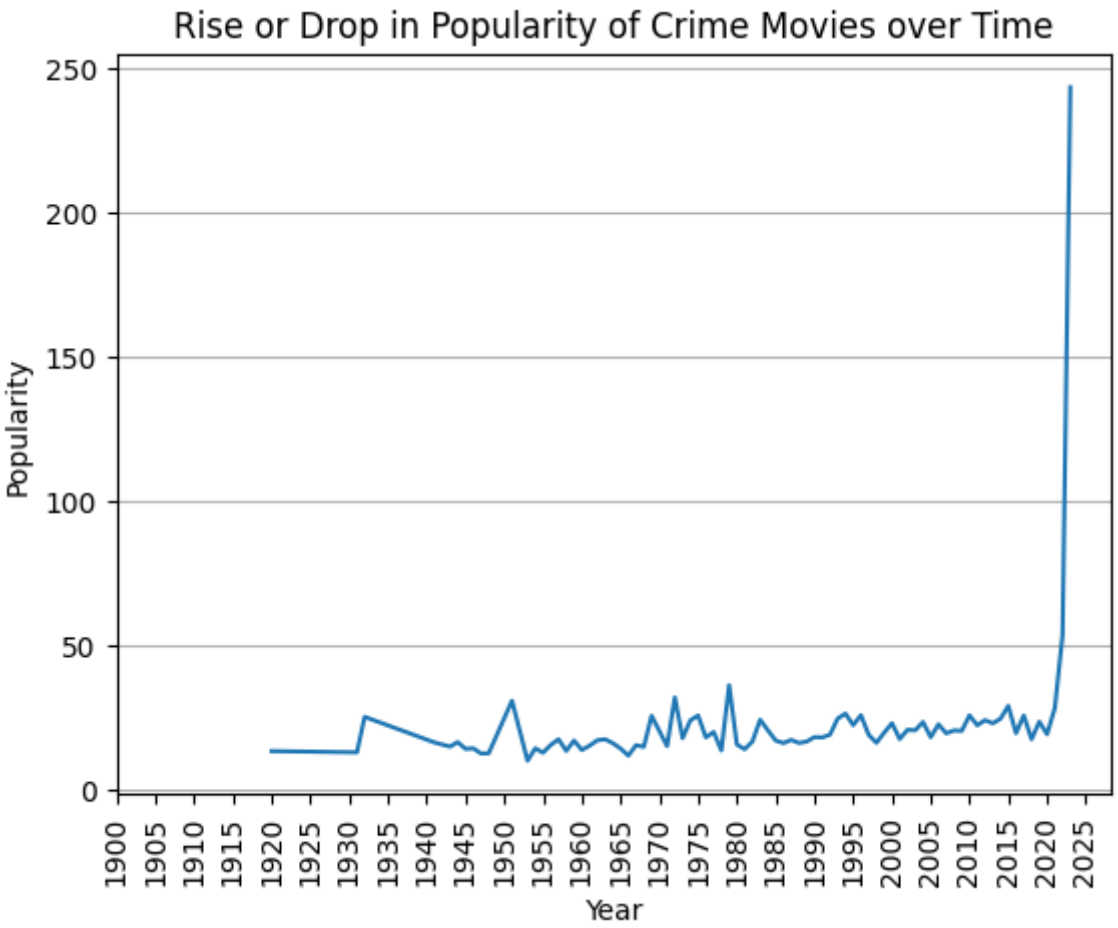

Rise or Drop in Popularity of Music Movies over Time
Popularity
Year
10
15
20
25
30
35
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

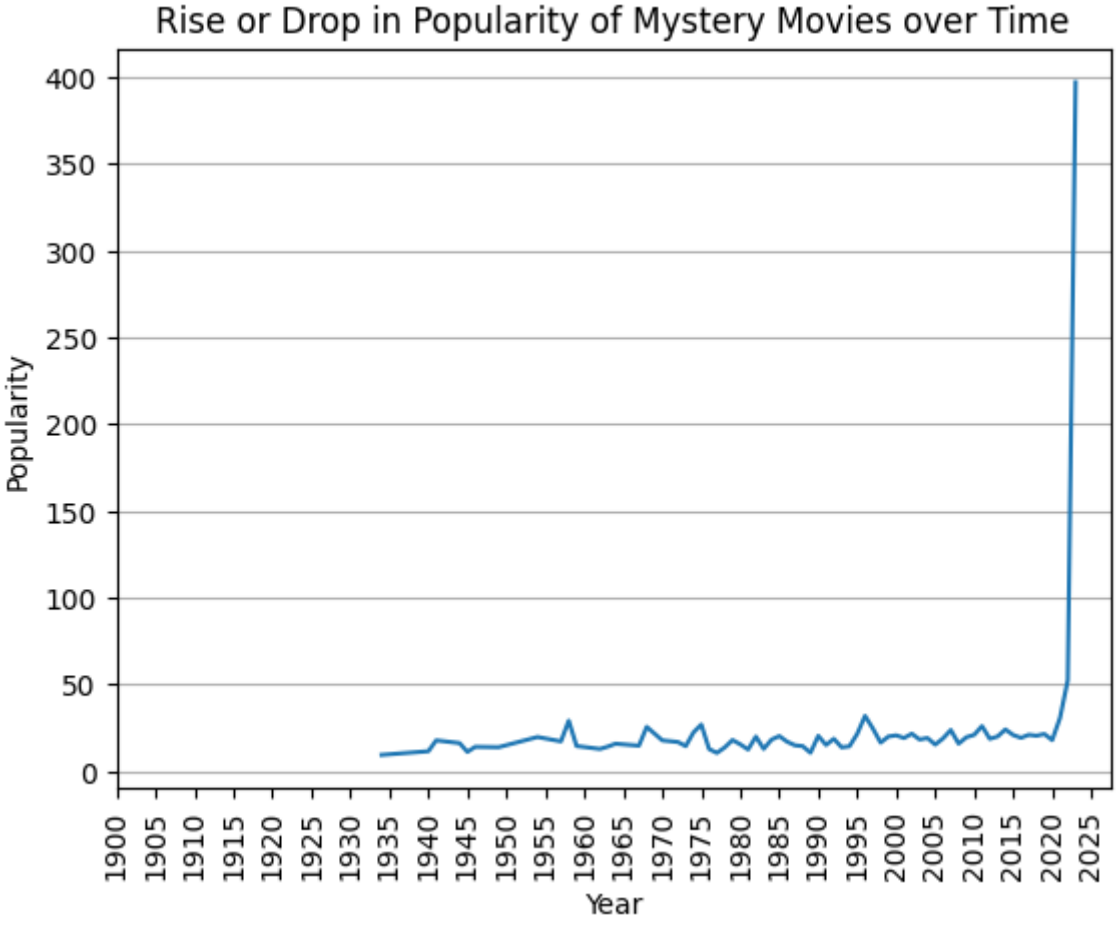

Rise or Drop in Popularity of TV Movie Movies over Time
Popularity
Year
10
20
30
40
50
60
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

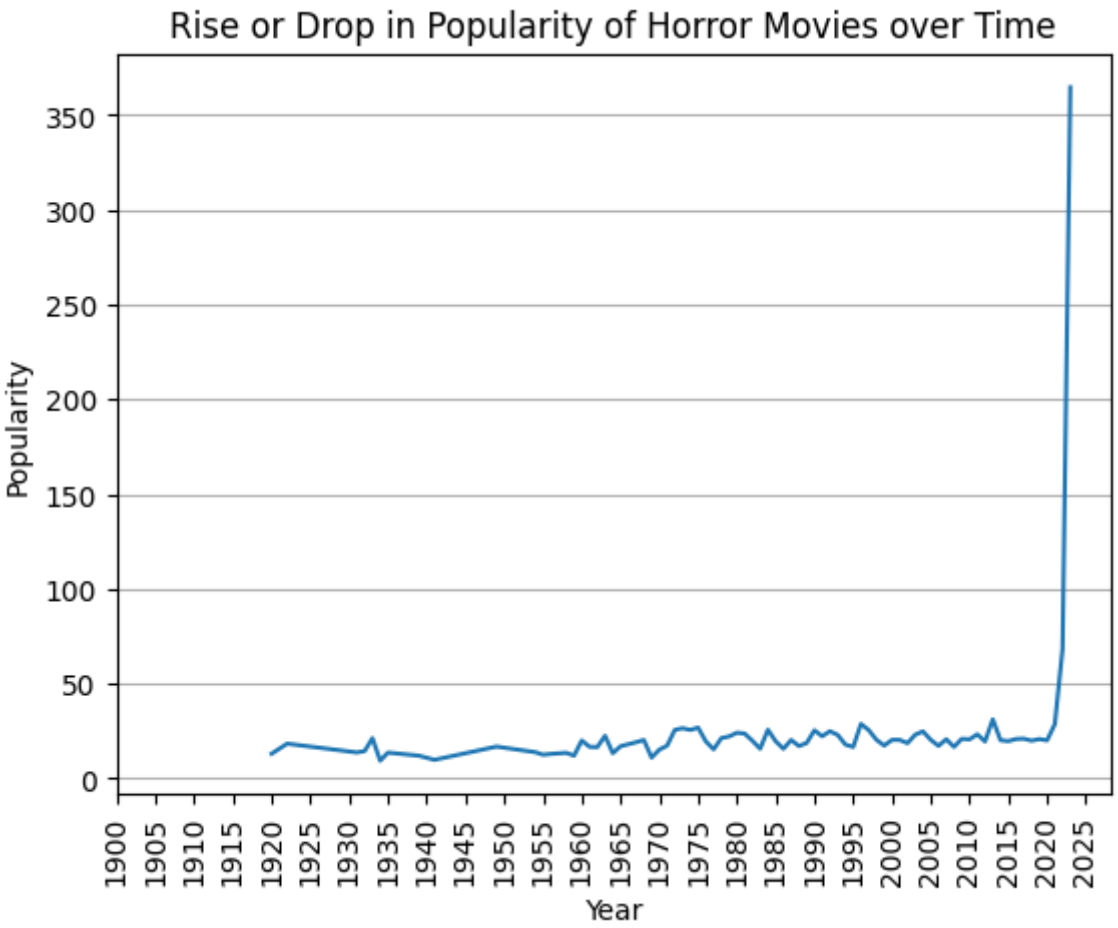

Rise or Drop in Popularity of Animation Movies over Time
Popularity
Year
0
100
200
300
400
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

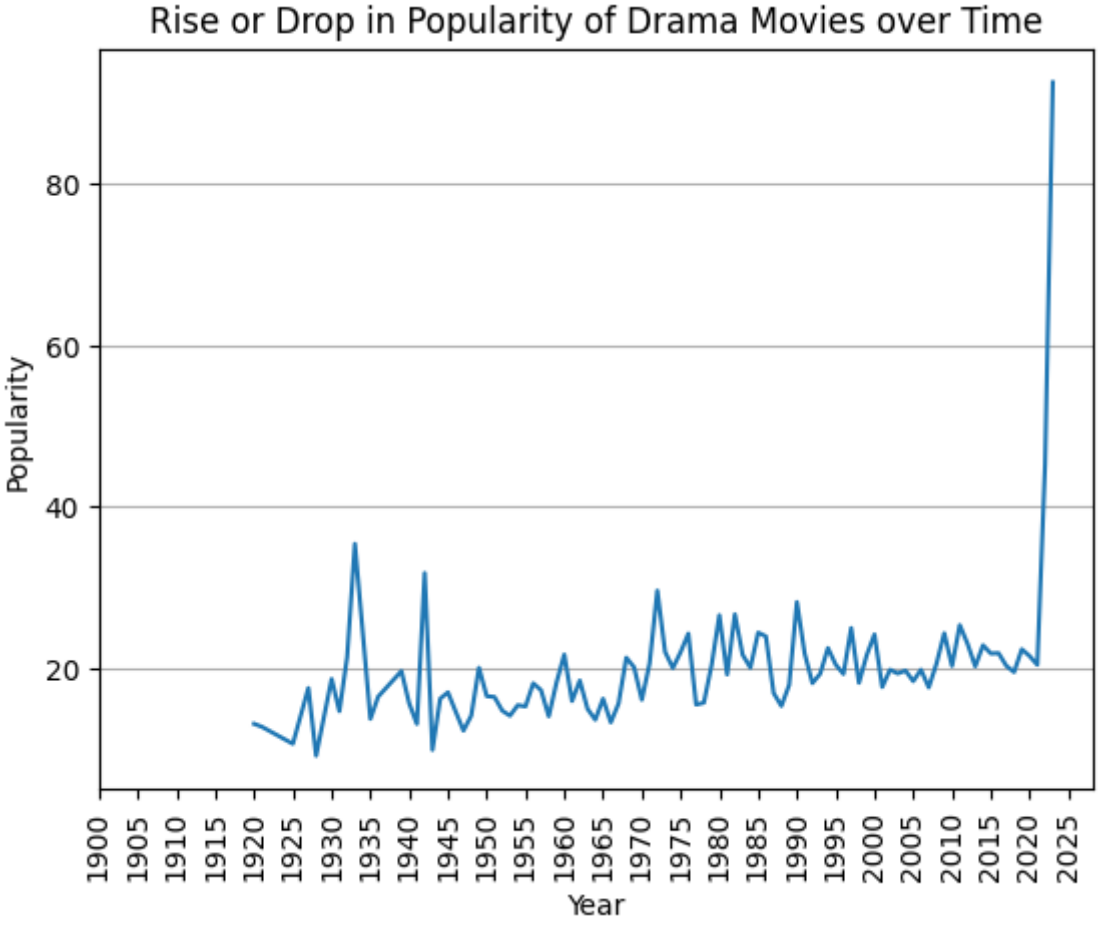

Rise or Drop in Popularity of Family Movies over Time
Popularity
0
100
200
300
400
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025
Year

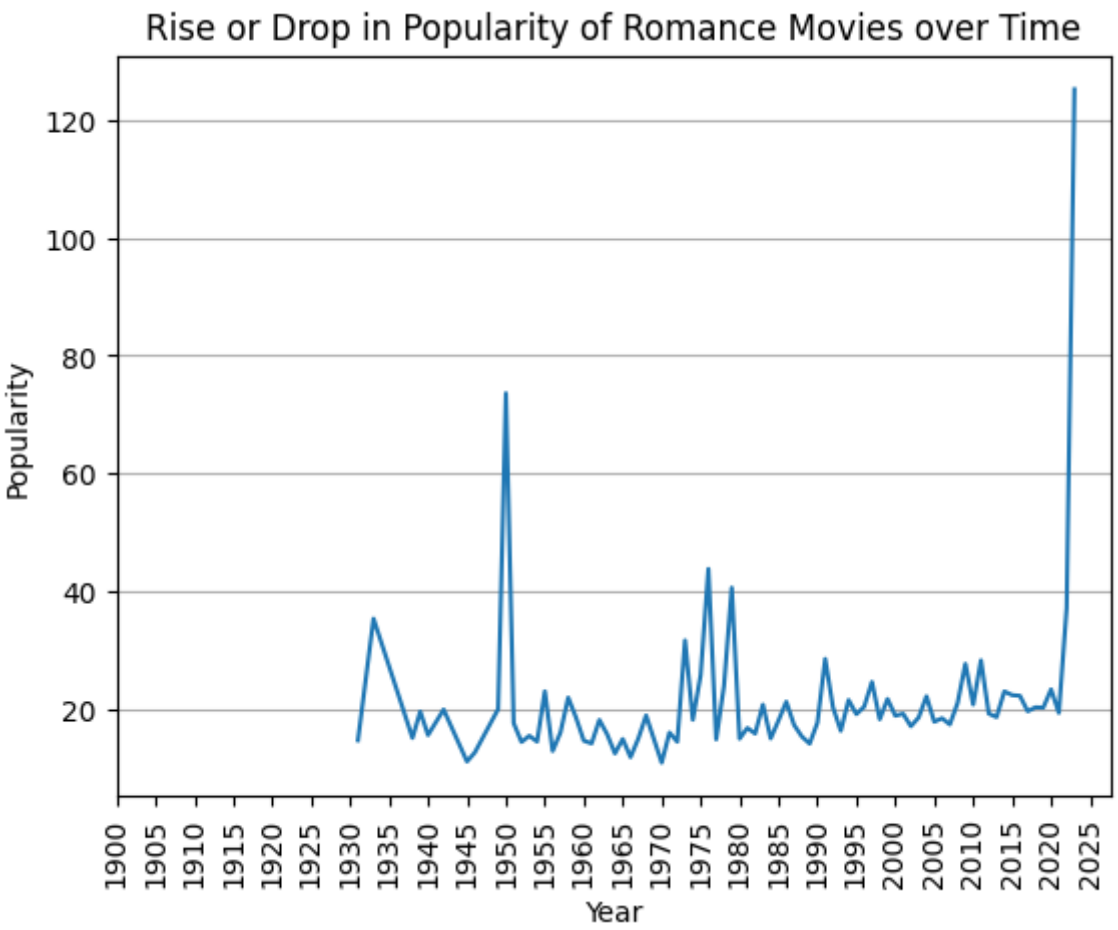

Rise or Drop in Popularity of Adventure Movies over Time
Popularity
0
100
200
300
400
500
600
700
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025
Year

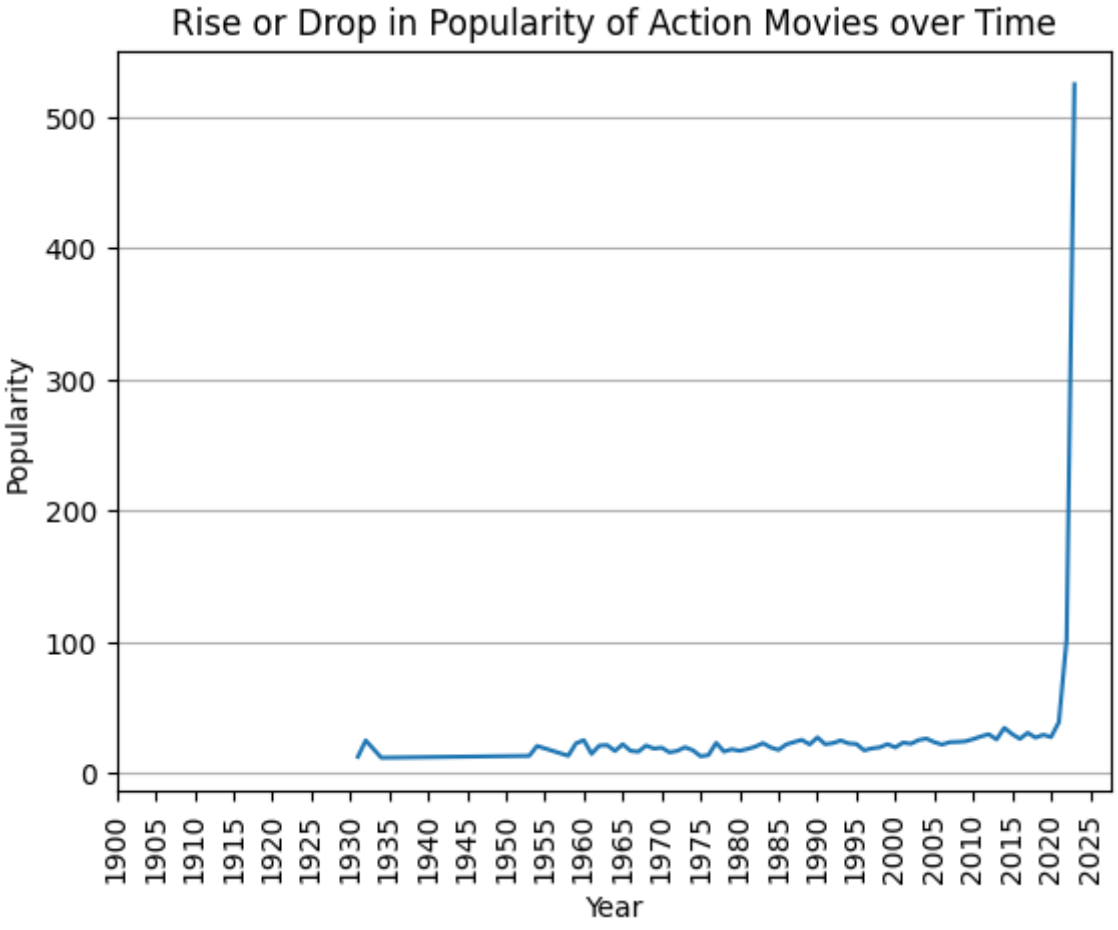

Rise or Drop in Popularity of Crime Movies over Time
Popularity
Year
0
50
100
150
200
250
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

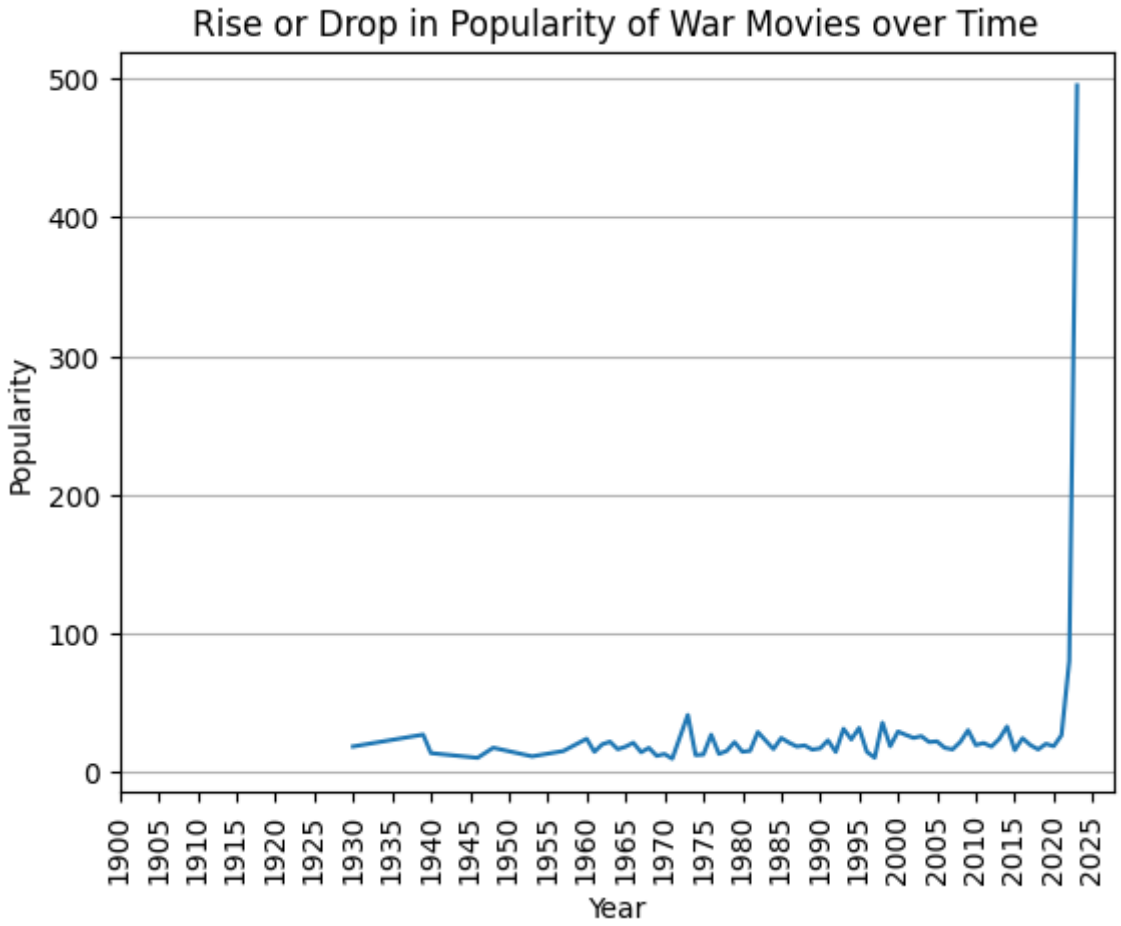

Rise or Drop in Popularity of Mystery Movies over Time
Popularity
Year
0
50
100
150
200
250
300
350
400
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025


Rise or Drop in Popularity of Horror Movies over Time
Popularity
Year
0
50
100
150
200
250
300
350
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025


Rise or Drop in Popularity of Drama Movies over Time
Popularity
Year
20
40
60
80
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

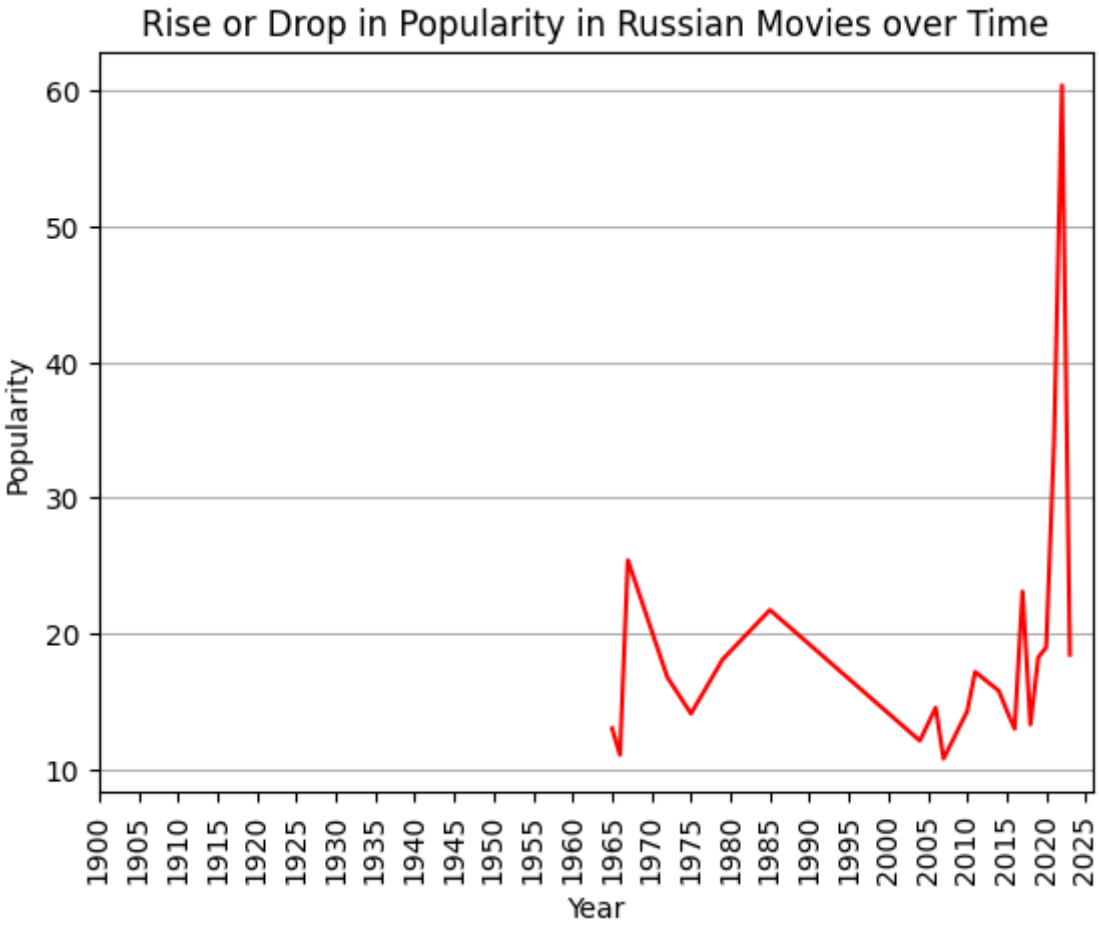

Rise or Drop in Popularity of Romance Movies over Time
Popularity
Year
120
100
80
60
40
20
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

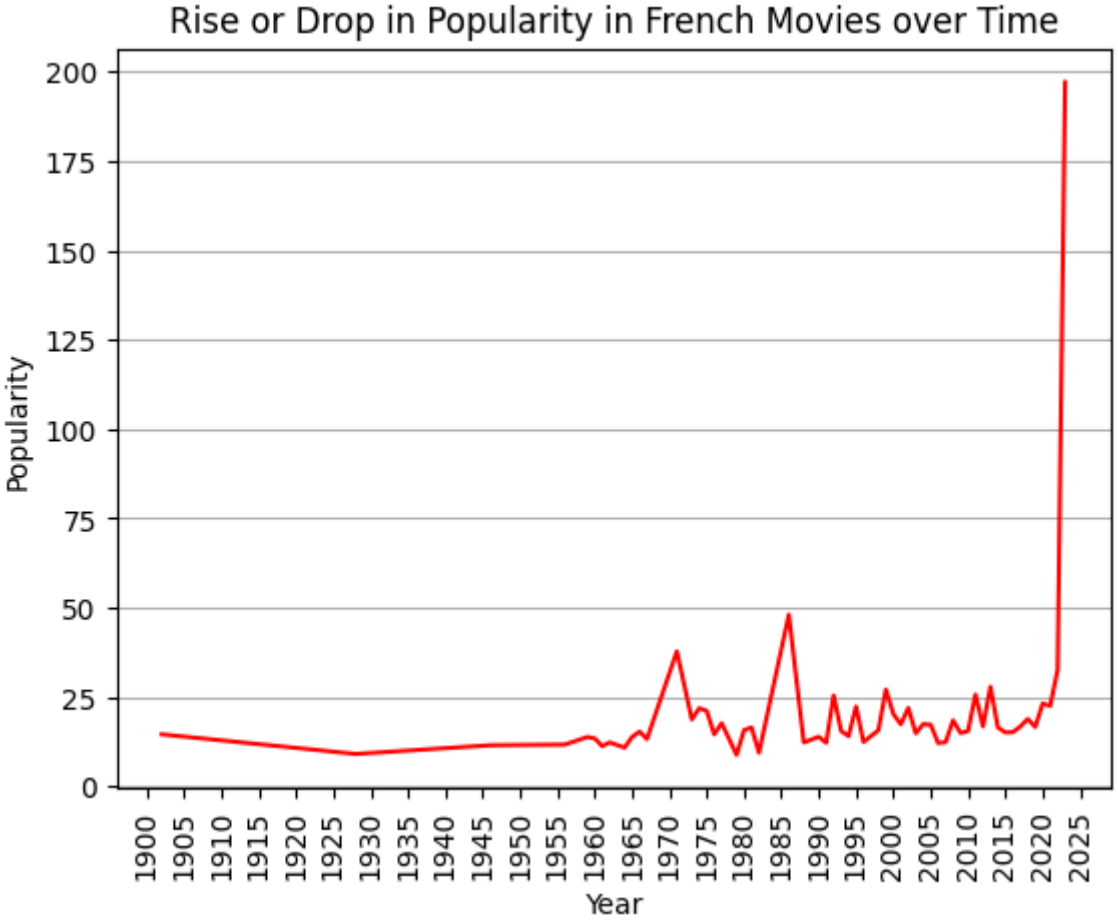

Rise or Drop in Popularity of Action Movies over Time
Popularity
Year
500
400
300
200
100
0
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

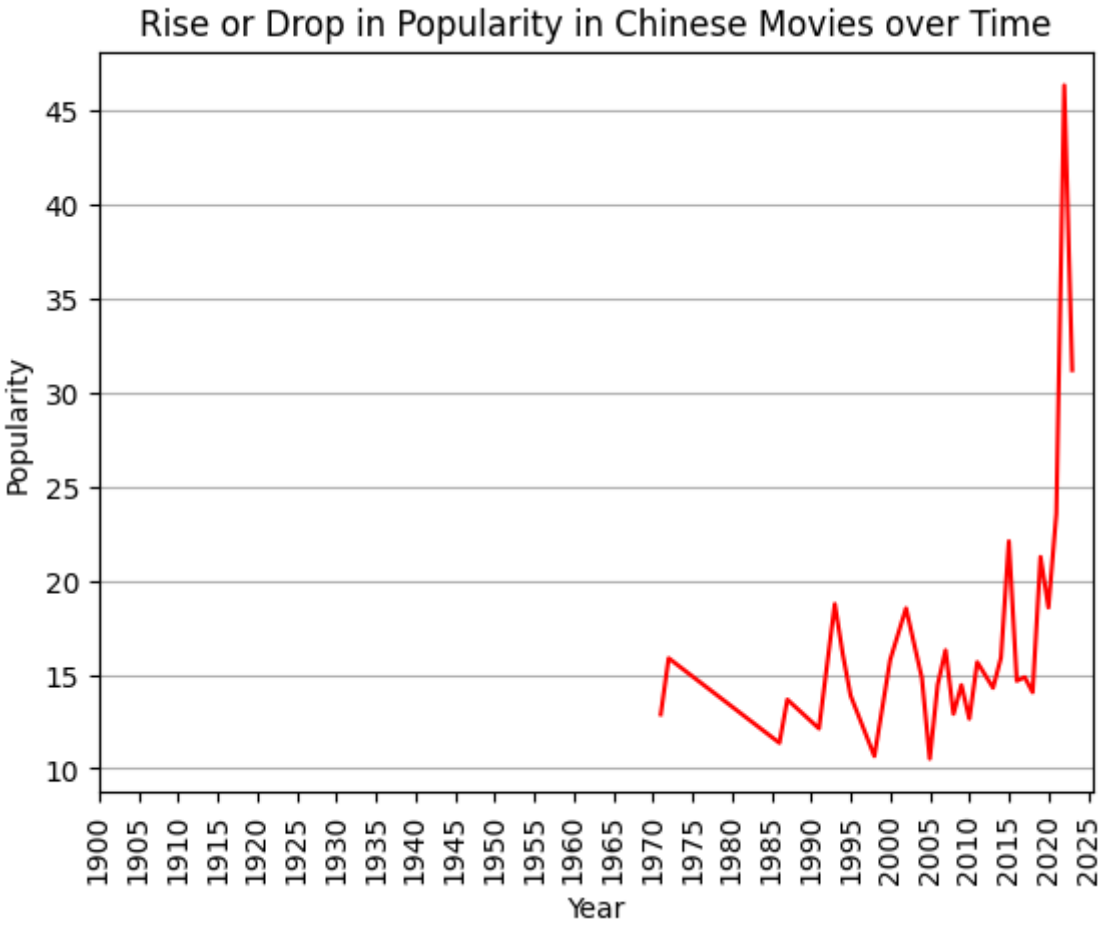


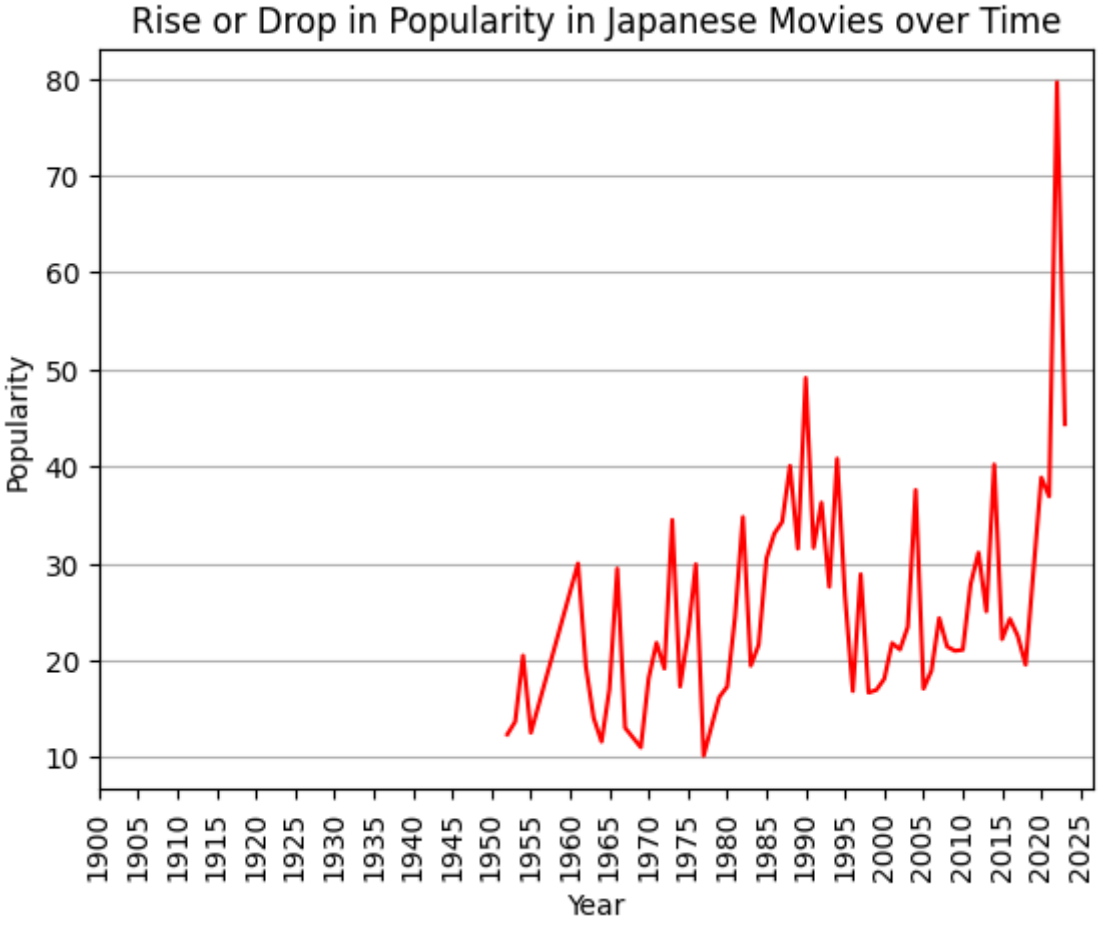
Visualization of Rise and Drop of Popularity with different languages over the time

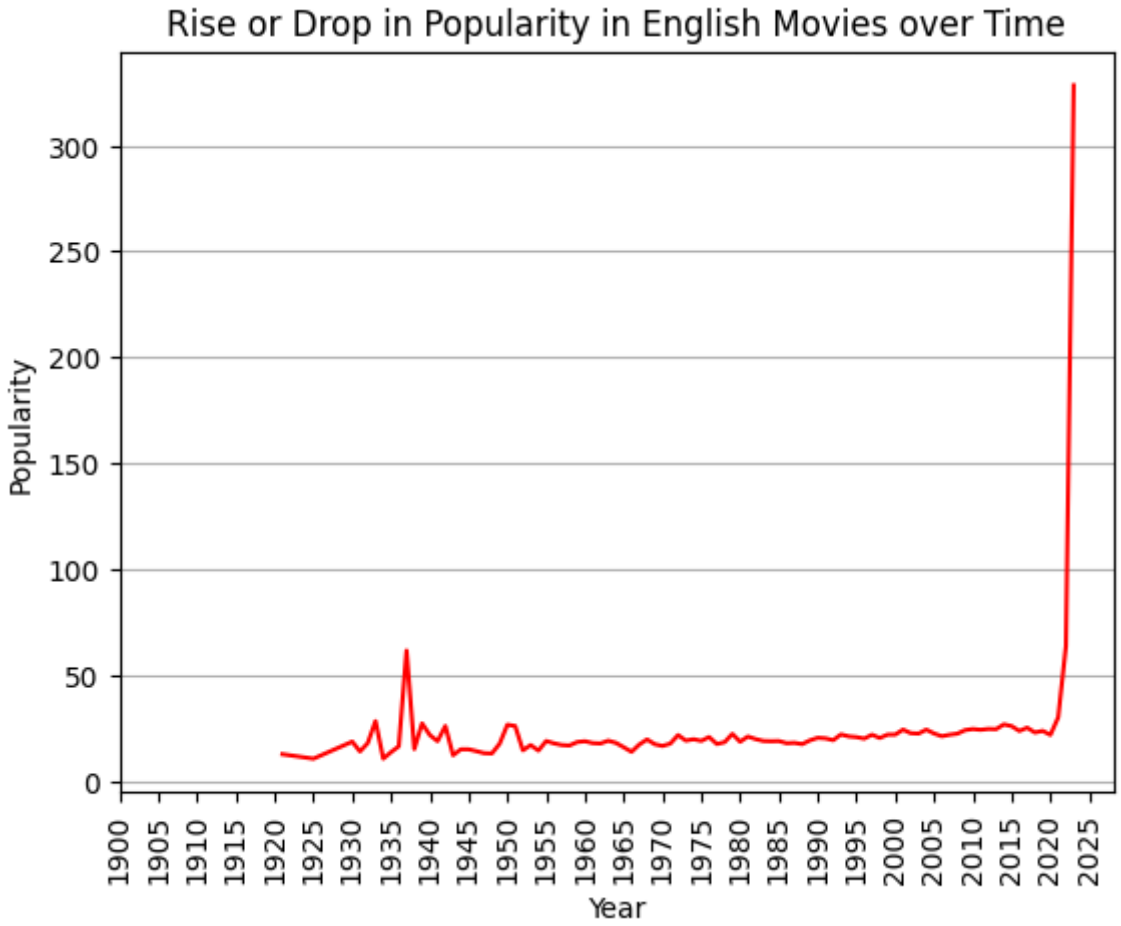
In [43]:
```
unique_lang = {"English","Japanese","French","Chinese","German","Russian"}

for language in unique_lang:
    movies = df[df['original_language'].str.contains(language)]
    avg_popularity = movies.groupby('year')['popularity'].mean()
    plt.plot(avg_popularity.index, avg_popularity.values, color='red')
    plt.title('Rise or Drop in Popularity in '+str(language)+' Movies over Time')
    plt.xlabel('Year')
    plt.ylabel('Popularity')
    plt.xticks(np.arange(1900, 2030, step=5),rotation=90)
    plt.grid(axis='y')
    plt.show()
```

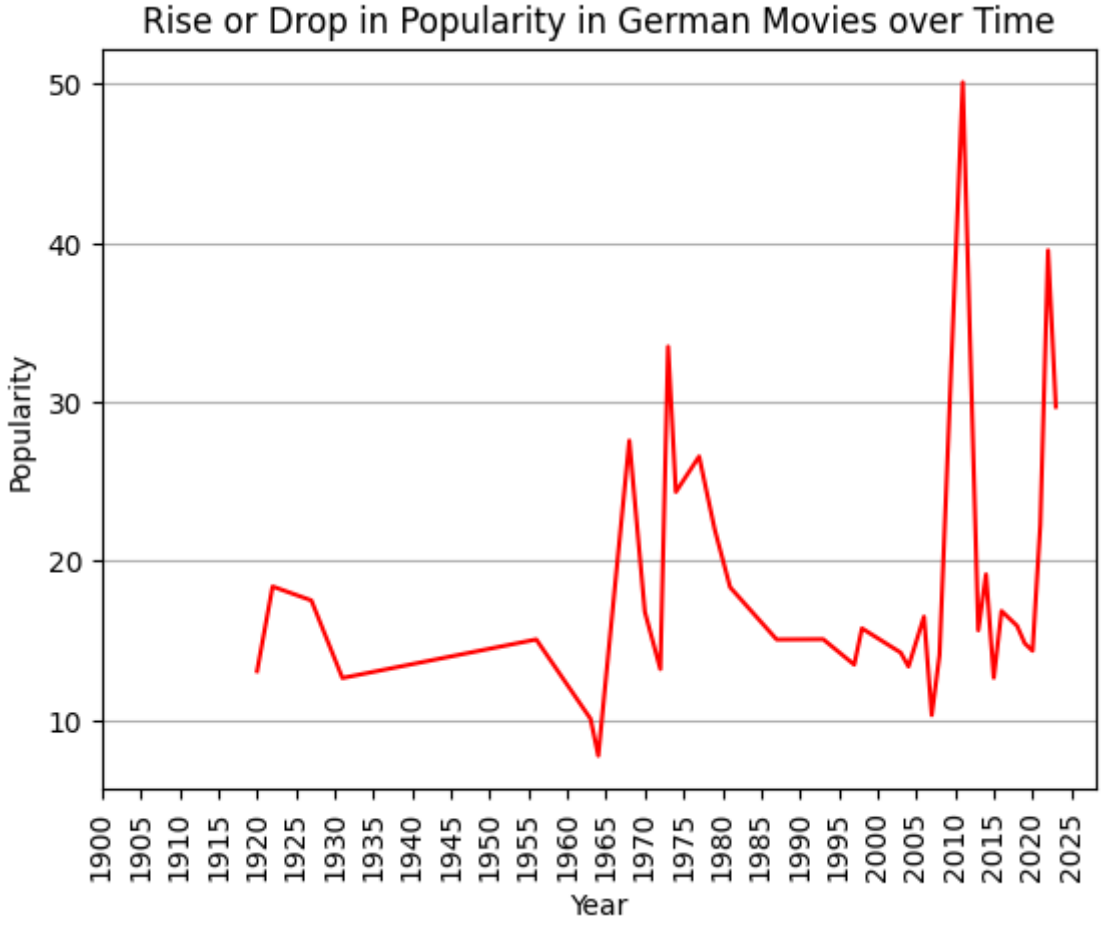

Rise or Drop in Popularity in Russian Movies over Time
Popularity
Year
60
50
40
30
20
10
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025


Rise or Drop in Popularity in French Movies over Time
Popularity
Year
200
175
150
125
100
75
50
25
0
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025


Rise or Drop in Popularity in Chinese Movies over Time
Popularity
Year
10
15
20
25
30
35
40
45
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025


Rise or Drop in Popularity in Japanese Movies over Time
Popularity
Year
10
20
30
40
50
60
70
80
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025


Rise or Drop in Popularity in English Movies over Time
Popularity
Year
0
50
100
150
200
250
300
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025


Rise or Drop in Popularity in German Movies over Time
Popularity
Year
10
20
30
40
50
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025

In [ ]: