Final Project Submission

Please fill out:

- Student name:SONIA AKINYI OJAY
- Student pace: part time
- Scheduled project review date/time:
- Instructor name: WiLLIAM OKOMBA AND NOAH KANDIE
- Blog post URL:https://github.com/soniaojay/dsc-phase-1-project.git (https://github.com/soniaojay/dsc-phase-1-project.git)

In [468]:
```
# Your code here - remember to use markdown cells for comments as well!
```

1. Import data: The Numbers: Cleaning up the data.

In [469]:
```
import pandas as pd
the_numbers_df=pd.read_csv(r"C:\Users\user\Desktop\Moringa\dsc-phase-1-project\zippedDa
the_numbers_df
```

Out[469]:

id	release_date	movie	production_budget	domestic_gross	worldwide_gross
1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279
2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875
3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963
5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747
...	...	...	...	...	...
78	Dec 31, 2018	Red 11	$7,000	$0	$0
79	Apr 2, 1999	Following	$6,000	$48,482	$240,495
80	Jul 13, 2005	Return to the Land of Wonders	$5,000	$1,338	$1,338
81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0
82	Aug 5, 2005	My Date With Drew	$1,100	$181,041	$181,041

5782 rows × 5 columns

In [470]:
```
#Checking for duplicates
the_numbers_df.duplicated().value_counts()
```

Out[470]:
```
False    5782
Name: count, dtype: int64
```

Remove missing values

In [471]:
```
the_numbers_df.dropna()
```

Out[471]:

id	release_date	movie	production_budget	domestic_gross	worldwide_gross
1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279
2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875
3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963
5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747
...	...	...	...	...	...
78	Dec 31, 2018	Red 11	$7,000	$0	$0
79	Apr 2, 1999	Following	$6,000	$48,482	$240,495
80	Jul 13, 2005	Return to the Land of Wonders	$5,000	$1,338	$1,338
81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0
82	Aug 5, 2005	My Date With Drew	$1,100	$181,041	$181,041

5782 rows × 5 columns

Removing Duplicates

In [472]:
```
the_numbers_df.drop_duplicates()
```

Out[472]:

id	release_date	movie	production_budget	domestic_gross	worldwide_gross
1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279
2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875
3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963
5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747
...	...	...	...	...	...
78	Dec 31, 2018	Red 11	$7,000	$0	$0
79	Apr 2, 1999	Following	$6,000	$48,482	$240,495
80	Jul 13, 2005	Return to the Land of Wonders	$5,000	$1,338	$1,338
81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0
82	Aug 5, 2005	My Date With Drew	$1,100	$181,041	$181,041

5782 rows × 5 columns

1. Import data: Box Office MojoLinks Cleaning up the data.

In [473]:
```
box_office_mojo_df=pd.read_csv(r"C:\Users\user\Desktop\Moringa\dsc-phase-1-project\zipp
box_office_mojo_df
```

Out[473]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.0	NaN	2018
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

3387 rows × 5 columns

Remove missing values

In [474]:
```
box_office_mojo_df = box_office_mojo_df.dropna()
box_office_mojo_df
```

Out[474]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
...	...	...	...	...	...
3275	I Still See You	LGF	1400.0	1500000	2018
3286	The Catcher Was a Spy	IFC	725000.0	229000	2018
3309	Time Freak	Grindstone	10000.0	256000	2018
3342	Reign of Judges: Title of Liberty - Concept Short	Darin Southa	93200.0	5200	2018
3353	Antonio Lopez 1970: Sex Fashion & Disco	FM	43200.0	30000	2018

2007 rows × 5 columns

In [475]:
```
#Check number of Duplicates
box_office_mojo_df.duplicated().value_counts()
```

Out[475]:
```
False    2007
Name: count, dtype: int64
```

Top 10 Movies with highest domestic gross in Box Office and The numbers.

In [476]:
```
top_10_box_office_movies = box_office_mojo_df.nlargest(10, 'domestic_gross')
top_10_box_office_movies
```

Out[476]:

	title	studio	domestic_gross	foreign_gross	year
1872	Star Wars: The Force Awakens	BV	936700000.0	1,131.6	2015
3080	Black Panther	BV	700100000.0	646900000	2018
3079	Avengers: Infinity War	BV	678800000.0	1,369.5	2018
1873	Jurassic World	Uni.	652300000.0	1,019.4	2015
727	Marvel's The Avengers	BV	623400000.0	895500000	2012
2758	Star Wars: The Last Jedi	BV	620200000.0	712400000	2017
3082	Incredibles 2	BV	608600000.0	634200000	2018
2323	Rogue One: A Star Wars Story	BV	532200000.0	523900000	2016
2759	Beauty and the Beast (2017)	BV	504000000.0	759500000	2017
2324	Finding Dory	BV	486300000.0	542300000	2016

In [477]:
```
top_10_box_office_movie_titles = top_10_box_office_movies['title'].tolist()
```

In [478]:
```
# Convert 'domestic_gross' column to string type
the_numbers_df['domestic_gross'] = the_numbers_df['domestic_gross'].astype(str)
# Remove currency symbol ($) and any other non-numeric characters
the_numbers_df['domestic_gross'] = the_numbers_df['domestic_gross'].str.replace('[^\d.

# Convert the column to numeric type
the_numbers_df['domestic_gross'] = the_numbers_df['domestic_gross'].astype(float)
```

In [479]:
```python
top_10_the_numbers_movies = the_numbers_df.nlargest(10, 'domestic_gross')
top_10_the_numbers_movies
```

Out[479]:

id	release_date	movie	production_budget	domestic_gross	worldwide_gross
6	Dec 18, 2015	Star Wars Ep. VII: The Force Awakens	$306,000,000	936662225.0	$2,053,311,220
1	Dec 18, 2009	Avatar	$425,000,000	760507625.0	$2,776,345,279
42	Feb 16, 2018	Black Panther	$200,000,000	700059566.0	$1,348,258,224
7	Apr 27, 2018	Avengers: Infinity War	$300,000,000	678815482.0	$2,048,134,200
43	Dec 19, 1997	Titanic	$200,000,000	659363944.0	$2,208,208,395
34	Jun 12, 2015	Jurassic World	$215,000,000	652270625.0	$1,648,854,864
27	May 4, 2012	The Avengers	$225,000,000	623279547.0	$1,517,935,897
5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	620181382.0	$1,316,721,747
44	Jun 15, 2018	Incredibles 2	$200,000,000	608581744.0	$1,242,520,711
75	Jul 18, 2008	The Dark Knight	$185,000,000	533720947.0	$1,001,996,207

Top Ten Box Office Movies with highest domestic gross.

In [480]:
```python
for index, row in top_10_box_office_movies.iterrows():
    print(f"{row['title']}: ${row['domestic_gross']:,.2f}")
```

```
Star Wars: The Force Awakens: $936,700,000.00
Black Panther: $700,100,000.00
Avengers: Infinity War: $678,800,000.00
Jurassic World: $652,300,000.00
Marvel's The Avengers: $623,400,000.00
Star Wars: The Last Jedi: $620,200,000.00
Incredibles 2: $608,600,000.00
Rogue One: A Star Wars Story: $532,200,000.00
Beauty and the Beast (2017): $504,000,000.00
Finding Dory: $486,300,000.00
```

Top Ten The Numbers Movies with highest domestic gross.

In [481]:
```python
for index, row in top_10_the_numbers_movies.iterrows():
    print(f"{row['movie']}: ${row['domestic_gross']:,.2f}")
```

```
Star Wars Ep. VII: The Force Awakens: $936,662,225.00
Avatar: $760,507,625.00
Black Panther: $700,059,566.00
Avengers: Infinity War: $678,815,482.00
Titanic: $659,363,944.00
Jurassic World: $652,270,625.00
The Avengers: $623,279,547.00
Star Wars Ep. VIII: The Last Jedi: $620,181,382.00
Incredibles 2: $608,581,744.00
The Dark Knight: $533,720,947.00
```

Top 10 Movies with highest foreign_gross/ worldwide_gross in Box Office and The numbers.

Assessing to identify the difference to movies.

In [482]:
```
for index, row in largest_foreign_gross.iterrows():
    print(f"{row['title']}: ${row['foreign_gross']:,.2f}")
```

```
Harry Potter and the Deathly Hallows Part 2: $960,500,000.00
Avengers: Age of Ultron: $946,400,000.00
Marvel's The Avengers: $895,500,000.00
Jurassic World: Fallen Kingdom: $891,800,000.00
Frozen: $875,700,000.00
Wolf Warrior 2: $867,600,000.00
Transformers: Age of Extinction: $858,600,000.00
Minions: $823,400,000.00
Aquaman: $812,700,000.00
Iron Man 3: $805,800,000.00
```

In [483]:
```
# Convert 'domestic_gross' column to string type
the_numbers_df['worldwide_gross'] = the_numbers_df['worldwide_gross'].astype(str)
# Remove currency symbol ($) and any other non-numeric characters
the_numbers_df['worldwide_gross'] = the_numbers_df['worldwide_gross'].str.replace('[^\(

# Convert the column to numeric type
the_numbers_df['worldwide_gross'] = the_numbers_df['worldwide_gross'].astype(float)
```

In [484]:
```
largest_worldwide_gross= the_numbers_df.nlargest(10, 'worldwide_gross')
```

In [485]:
```
for index, row in largest_worldwide_gross.iterrows():
    print(f"{row['movie']}: ${row['worldwide_gross']:,.2f}")
```

```
Avatar: $2,776,345,279.00
Titanic: $2,208,208,395.00
Star Wars Ep. VII: The Force Awakens: $2,053,311,220.00
Avengers: Infinity War: $2,048,134,200.00
Jurassic World: $1,648,854,864.00
Furious 7: $1,518,722,794.00
The Avengers: $1,517,935,897.00
Avengers: Age of Ultron: $1,403,013,963.00
Black Panther: $1,348,258,224.00
Harry Potter and the Deathly Hallows: Part II: $1,341,693,157.00
```

In [486]:
```
for index, row in largest_foreign_gross.iterrows():
    print(f"{row['title']}: ${row['foreign_gross']:,.2f}")
```

```
Harry Potter and the Deathly Hallows Part 2: $960,500,000.00
Avengers: Age of Ultron: $946,400,000.00
Marvel's The Avengers: $895,500,000.00
Jurassic World: Fallen Kingdom: $891,800,000.00
Frozen: $875,700,000.00
Wolf Warrior 2: $867,600,000.00
Transformers: Age of Extinction: $858,600,000.00
Minions: $823,400,000.00
Aquaman: $812,700,000.00
Iron Man 3: $805,800,000.00
```

Assessing percentage profit in The Numbers.

In [487]:
```
the_numbers_df['total_gross'] = the_numbers_df['domestic_gross'] + the_numbers_df['worl
```

In [488]:
```
the_numbers_df['production_budget'] = the_numbers_df['production_budget'].astype(str).:
the_numbers_df['production_budget']
```

Out[488]:
```
id
1     425000000.0
2     410600000.0
3     350000000.0
4     330600000.0
5     317000000.0
         ...
78         7000.0
79         6000.0
80         5000.0
81         1400.0
82         1100.0
Name: production_budget, Length: 5782, dtype: float64
```

In [489]:
```
# Calculate the 'percentage_profit'
the_numbers_df['percentage_profit'] = ((the_numbers_df['total_gross'] - the_numbers_df
the_numbers_df
```

Out[489]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	total_gross	percentag
id							
1	Dec 18, 2009	Avatar	425000000.0	760507625.0	2.776345e+09	3.536853e+09	73:
2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09	1.286728e+09	21:
3	Jun 7, 2019	Dark Phoenix	350000000.0	42762350.0	1.497624e+08	1.925247e+08	-4
4	May 1, 2015	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09	1.862020e+09	46:
5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09	1.936903e+09	51
...	...	...	...	...	...	...	
78	Dec 31, 2018	Red 11	7000.0	0.0	0.000000e+00	0.000000e+00	-10(
79	Apr 2, 1999	Following	6000.0	48482.0	2.404950e+05	2.889770e+05	471(
80	Jul 13, 2005	Return to the Land of Wonders	5000.0	1338.0	1.338000e+03	2.676000e+03	-4(
81	Sep 29, 2015	A Plague So Pleasant	1400.0	0.0	0.000000e+00	0.000000e+00	-10(
82	Aug 5, 2005	My Date With Drew	1100.0	181041.0	1.810410e+05	3.620820e+05	3281(

5782 rows × 7 columns

Top 10 Movies with highest gross profit in The numbers.

In [490]:
```python
Top_10_Highest_Profit = the_numbers_df.nlargest(10, 'percentage_profit')
Top_10_Highest_Profit
```

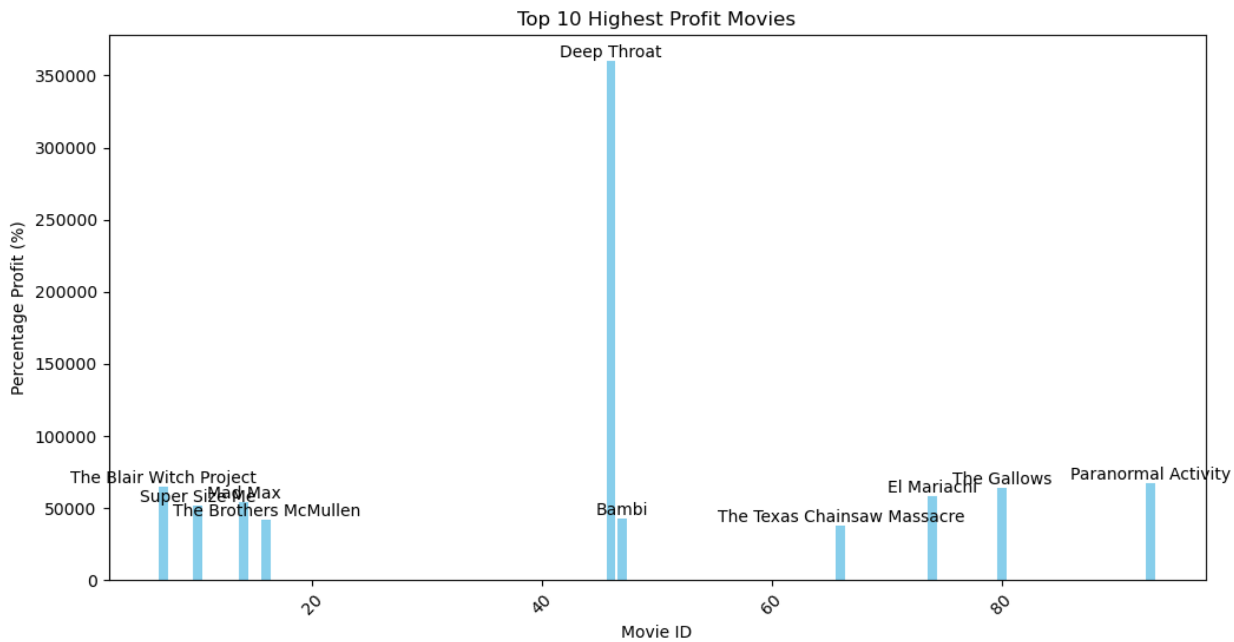
Out[490]:

id	release_date	movie	production_budget	domestic_gross	worldwide_gross	total_gross	percentag
46	Jun 30, 1972	Deep Throat	25000.0	45000000.0	45000000.0	90000000.0	359900
93	Sep 25, 2009	Paranormal Activity	450000.0	107918810.0	194183034.0	302101844.0	67033
7	Jul 14, 1999	The Blair Witch Project	600000.0	140539099.0	248300000.0	388839099.0	64706
80	Jul 10, 2015	The Gallows	100000.0	22764410.0	41656474.0	64420884.0	64320
74	Feb 26, 1993	El Mariachi	7000.0	2040920.0	2041928.0	4082848.0	58226
14	Mar 21, 1980	Mad Max	200000.0	8750000.0	99750000.0	108500000.0	54150
10	May 7, 2004	Super Size Me	65000.0	11529368.0	22233808.0	33763176.0	51843
47	Aug 13, 1942	Bambi	858000.0	102797000.0	268000000.0	370797000.0	43116
16	Aug 9, 1995	The Brothers McMullen	50000.0	10426506.0	10426506.0	20853012.0	41606
66	Oct 18, 1974	The Texas Chainsaw Massacre	140000.0	26572439.0	26572439.0	53144878.0	37860

Graphical presentation of the highest profits for the numbers.

In [491]:
```python
import matplotlib.pyplot as plt
```

In [492]:
```
plt.figure(figsize=(12, 6))
bars = plt.bar(Top_10_Highest_Profit.index, Top_10_Highest_Profit['percentage_profit'],
plt.xlabel('Movie ID')
plt.ylabel('Percentage Profit (%)')
plt.title('Top 10 Highest Profit Movies')
plt.xticks(rotation=45)

for bar, movie_name in zip(bars, Top_10_Highest_Profit['movie']):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), movie_name, ha='cent

# Show the plot
plt.show()
```



Load and Explore the Data - TheMovieDB

In [493]:
```
rotten_tomatoes_df = pd.read_csv(r"C:\Users\user\Desktop\Moringa\dsc-phase-1-project\z:

print(rotten_tomatoes_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         26517 non-null  int64  
 1   genre_ids          26517 non-null  object 
 2   id                 26517 non-null  int64  
 3   original_language  26517 non-null  object 
 4   original_title     26517 non-null  object 
 5   popularity         26517 non-null  float64
 6   release_date       26517 non-null  object 
 7   title              26517 non-null  object 
 8   vote_average       26517 non-null  float64
 9   vote_count         26517 non-null  int64  
dtypes: float64(2), int64(3), object(5)
memory usage: 2.0+ MB
None
```

In [494]:
```
# Display basic statistics of the DataFrame
selected_columns = ['popularity', 'vote_average', 'vote_count']
statistics_selected_columns = the_movie_db_df[selected_columns].describe()

# Displaying descriptive statistics
print(statistics_selected_columns)
```

```
         popularity  vote_average    vote_count
count  26517.000000  26517.000000  26517.000000
mean       3.130912      5.991281    194.224837
std        4.355229      1.852946    960.961095
min        0.600000      0.000000      1.000000
25%        0.600000      5.000000      2.000000
50%        1.374000      6.000000      5.000000
75%        3.694000      7.000000     28.000000
max       80.773000     10.000000  22186.000000
```

Data Cleaning

1. Handling Missing Values

In [495]:
```
the_movie_db_df.fillna(value=0, inplace=True)
```

2. Converting Data types

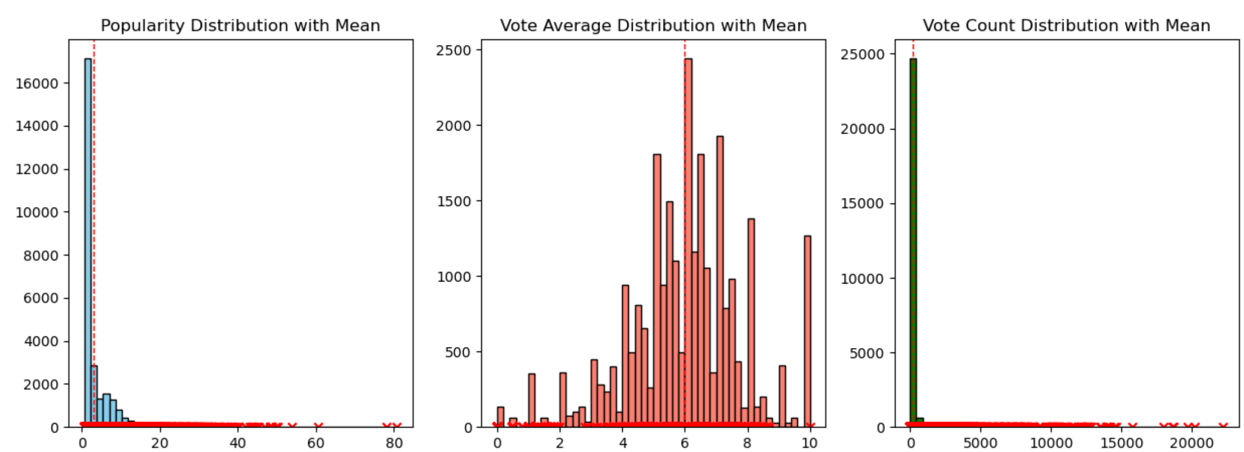
In [496]:

```
rotten_tomatoes_df['popularity'] = pd.to_numeric(rotten_tomatoes_df['popularity'])
rotten_tomatoes_df['vote_average'] = pd.to_numeric(rotten_tomatoes_df['vote_average'])
rotten_tomatoes_df['vote_count'] = pd.to_numeric(rotten_tomatoes_df['vote_count'])
# Extracting year from release_date
rotten_tomatoes_df['release_date'] = pd.to_datetime(rotten_tomatoes_df['release_date']]
rotten_tomatoes_df['release_year'] = rotten_tomatoes_df['release_date'].dt.year
rotten_tomatoes_df['release_month'] = rotten_tomatoes_df['release_date'].dt.month
print(rotten_tomatoes_df.head())
```

```
                           original_title  popularity release_date  \
0  Harry Potter and the Deathly Hallows: Part 1      33.533   2010-11-19
1                    How to Train Your Dragon      28.734   2010-03-26
2                                  Iron Man 2      28.515   2010-05-07
3                                   Toy Story      28.005   1995-11-22
4                                   Inception      27.920   2010-07-16

                                        title  vote_average  vote_count  \
0  Harry Potter and the Deathly Hallows: Part 1           7.7       10788
1                    How to Train Your Dragon           7.7        7610
2                                  Iron Man 2           6.8       12368
3                                   Toy Story           7.9       10174
4                                   Inception           8.3       22186

   release_year  release_month
0          2010             11
1          2010              3
2          2010              5
3          1995             11
4          2010              7
```

3.Identifying outliers in popularity, vote_average, and vote_count

In [497]:
```
import numpy as np

# Calculating IQR for each column
Q1 = the_movie_db_df[['popularity', 'vote_average', 'vote_count']].quantile(0.25)
Q3 = the_movie_db_df[['popularity', 'vote_average', 'vote_count']].quantile(0.75)
IQR = Q3 - Q1

# Determining outliers based on IQR method
outliers_IQR = the_movie_db_df[((the_movie_db_df[['popularity', 'vote_average', 'vote_c

# Plotting mean lines and outliers
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.hist(the_movie_db_df['popularity'], bins=50, color='skyblue', edgecolor='black')
plt.axvline(the_movie_db_df['popularity'].mean(), color='red', linestyle='dashed', lin
plt.title('Popularity Distribution with Mean')
plt.scatter(outliers_IQR['popularity'], np.zeros_like(outliers_IQR['popularity']), col

plt.subplot(1, 3, 2)
plt.hist(the_movie_db_df['vote_average'], bins=50, color='salmon', edgecolor='black')
plt.axvline(the_movie_db_df['vote_average'].mean(), color='red', linestyle='dashed', l
plt.title('Vote Average Distribution with Mean')
plt.scatter(outliers_IQR['vote_average'], np.zeros_like(outliers_IQR['vote_average']),

plt.subplot(1, 3, 3)
plt.hist(the_movie_db_df['vote_count'], bins=50, color='green', edgecolor='black')
plt.axvline(the_movie_db_df['vote_count'].mean(), color='red', linestyle='dashed', lin
plt.title('Vote Count Distribution with Mean')
plt.scatter(outliers_IQR['vote_count'], np.zeros_like(outliers_IQR['vote_count']), col

plt.show()
```



Note: There are outliers in Popularity, Vote average and vote count distribution.

In [498]:
```
# Remove outliers identified using the IQR method
cleaned_the_movie_db_df = the_movie_db_df[~the_movie_db_df.index.isin(outliers_IQR.ind
cleaned_the_movie_db_df
```

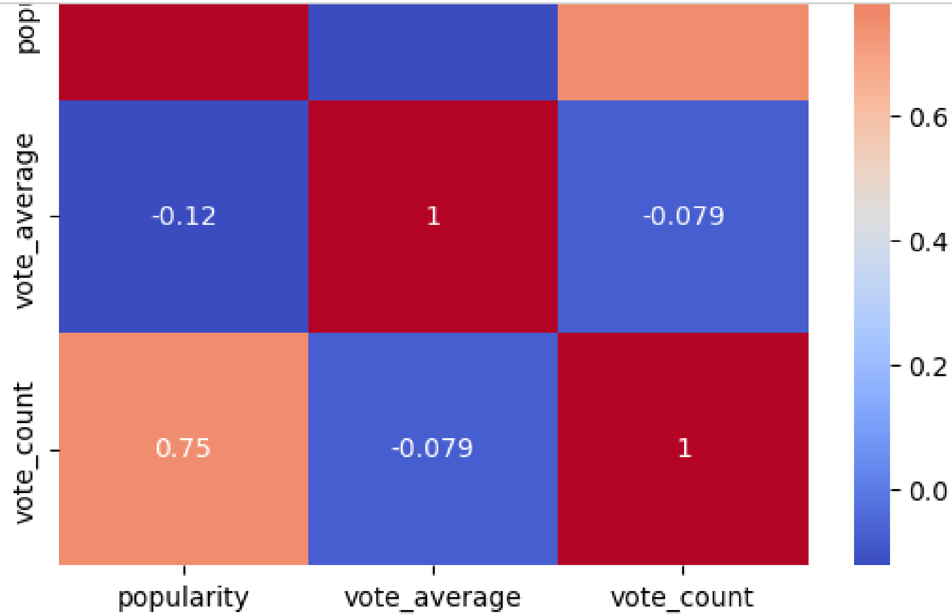
Out[498]:

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_date	title
227	227	[18, 10749]	51736	en	Bloomington	8.176	2010-06-23	Bloomington
229	229	[12, 16, 10751, 35]	43956	en	Tom and Jerry Meet Sherlock Holmes	8.142	2010-08-24	Tom and Jerry Meet Sherlock Holmes
231	231	[28, 27, 878]	52454	en	Mega Shark vs. Crocosaurus	8.129	2010-12-21	Mega Shark vs. Crocosaurus
253	253	[18, 28, 53]	12645	en	Inhale	7.826	2010-10-01	Inhale
270	270	[27, 53, 9648]	19237	en	Kill Theory	7.663	2010-01-29	Kill Theory
...	...	...	...	...	...	...	...	...
26494	26494	[18]	567020	es	La última virgen	0.600	2018-05-26	The Last Virgin
26495	26495	[]	556601	en	Recursion	0.600	2018-08-28	Recursion
26496	26496	[99]	524548	en	The Case of: Caylee Anthony	0.600	2018-05-19	The Case of: Caylee Anthony
26497	26497	[]	514045	en	The Portuguese Kid	0.600	2018-02-14	The Portuguese Kid
26498	26498	[]	497839	en	The 23rd Annual Critics' Choice Awards	0.600	2018-01-11	The 23rd Annual Critics' Choice Awards

21332 rows × 12 columns

Exploratory Data Analysis (EDA): data analysis to identify trends and patterns in the data. Exploring the relationships between different variables such as genre, original language, popularity, release date, average vote, and vote count.

In [499]:
```
# Correlation analysis
correlation_matrix = cleaned_the_movie_db_df [['popularity', 'vote_average', 'vote_cou
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



A corr of 0.75 indicates a strong positive correlation between vote_count and popularity, Corr of -0.12 indicates a weak negative correlation between vote_average and popularity , A corr of -0.079 indicates a very weak negative correlation between vote_average and vote_count.

Diverse Audience Preferences: Different segments of the audience might have divergent tastes. A movie could have a high popularity and vote count due to its broad appeal or marketing efforts but receive lower "vote average" ratings from a subset of viewers who have different preferences or expectations.

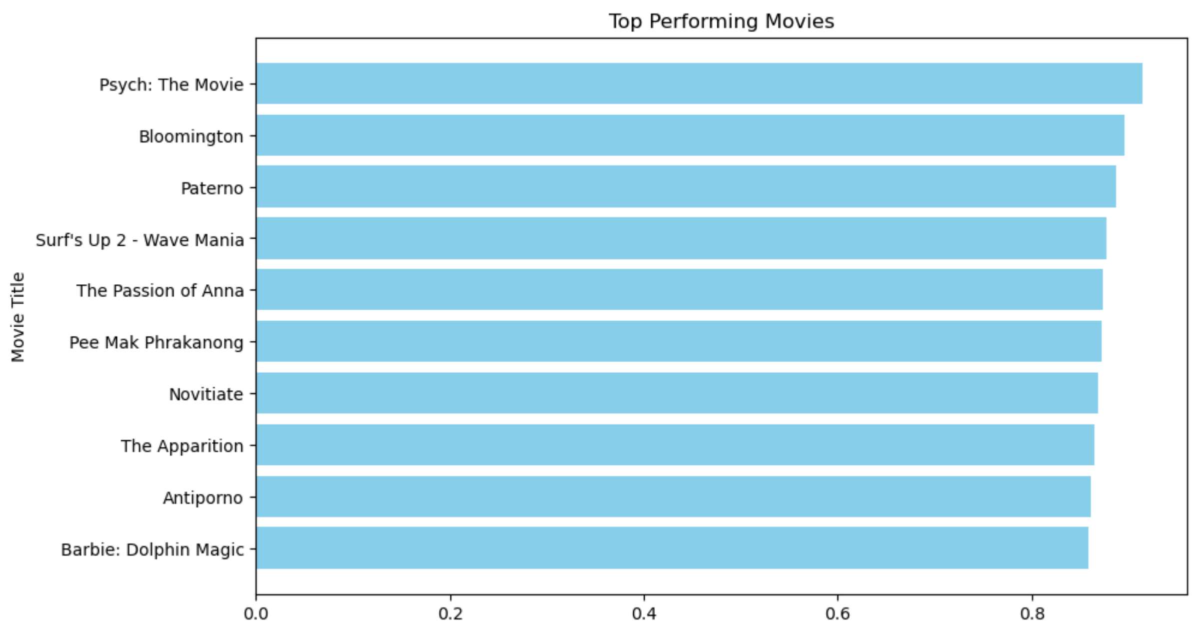
Identify Top Performing Films based on the composite score which is weight between Popularity, vote_count and vote_average.

```
In [500]: # Normalize the data
cleaned_the_movie_db_df.loc[:, 'normalized_popularity'] = (cleaned_the_movie_db_df['po
cleaned_the_movie_db_df.loc[:, 'normalized_vote_count'] = (cleaned_the_movie_db_df['vo
cleaned_the_movie_db_df.loc[:, 'normalized_average_vote'] = (cleaned_the_movie_db_df['

# Assign weights to each metric
weight_popularity = 0.5
weight_vote_count = 0.3
weight_average_vote = 0.2

# Calculate the composite score
cleaned_the_movie_db_df.loc[:, 'composite_score'] = (weight_popularity * cleaned_the_mo

# Rank the films based on the composite score
top_movies = cleaned_the_movie_db_df.sort_values(by='composite_score', ascending=False

# Select top performing films (e.g., top 10)
top_performing_movies = top_movies.head(10)

# Display the top performing films
print("Top 10 Performing Movies based on Composite Score:")
print(top_performing_movies[['title', 'popularity', 'vote_count', 'vote_average', 'com
```

```
20996  Surf's Up 2 - Wave Mania      8.206         65          5.8
24349       The Passion of Anna      7.824         60          7.5
8100          Pee Mak Phrakanong     8.310         54          7.3
21023                  Novitiate     7.967         62          6.6
24331             The Apparition     7.913         63          6.4
21080                  Antiporno     7.456         67          6.7
21064     Barbie: Dolphin Magic      7.631         65          6.5

       composite_score  release_month  release_year
21013         0.914975             12          2017
227           0.895821              6          2010
24302         0.887009              4          2018
20996         0.877761              1          2017
24349         0.872831             11          2018
8100          0.871986              3          2013
21023         0.868670             10          2017
24331         0.864723              9          2018
21080         0.860852             12          2017
21064         0.858078              9          2017
```

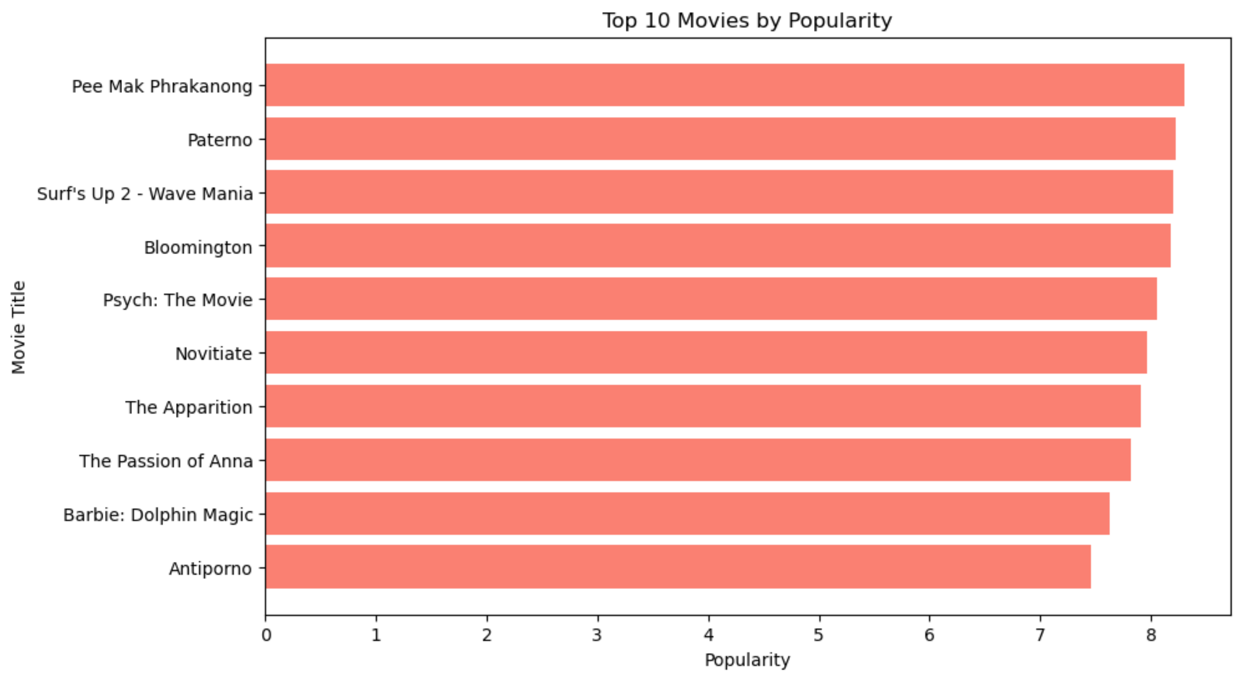
Ploting a Bar Graph for the top 10 movies based on the composite score

In [501]:
```
# Sort the DataFrame by composite score in descending order (just in case it's not alre
top_performing_movies = top_performing_movies.sort_values(by='composite_score', ascendi

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.barh(top_performing_movies['title'], top_performing_movies['composite_score'], colo
plt.xlabel('Composite Score')
plt.ylabel('Movie Title')
plt.title('Top Performing Movies')
plt.gca().invert_yaxis()  # Invert y-axis to display the highest score at the top
plt.show()
```



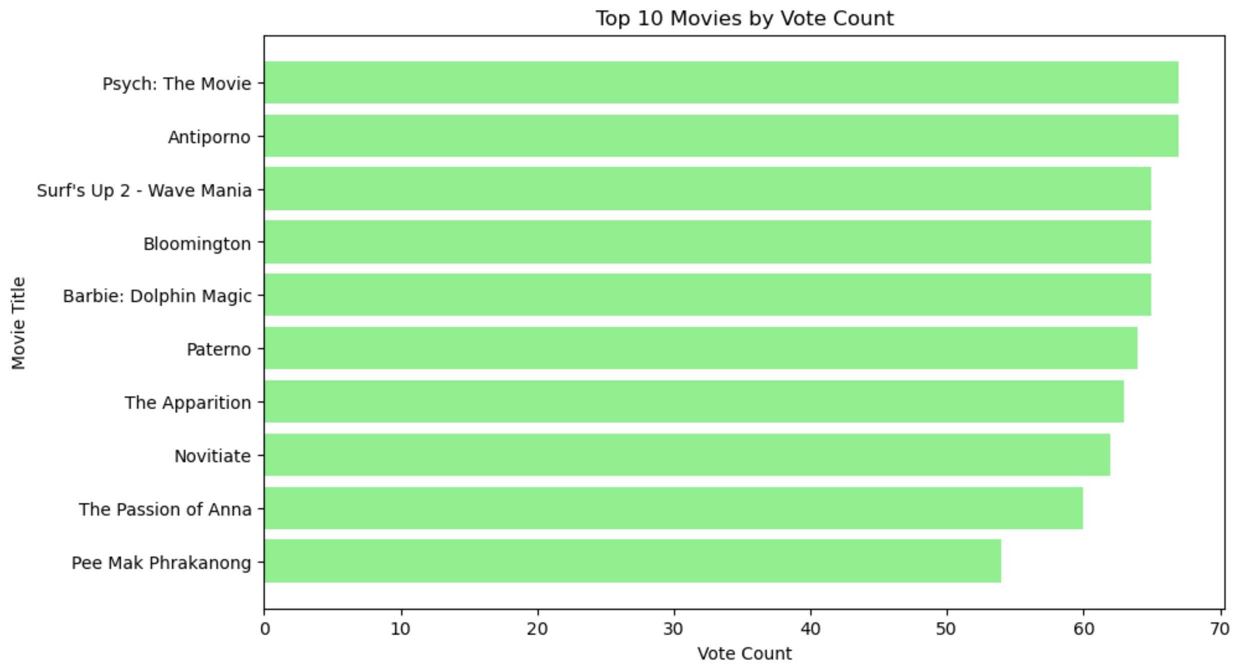
Ploting a Bar Graph for the top 10 movies based on the Popularity

In [502]:
```
# Sort the DataFrame by popularity in descending order (just in case it's not already s
top_performing_movies = top_performing_movies.sort_values(by='popularity', ascending=Fa

# Select top 10 movies
top_10_movies = top_performing_movies.head(10)

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.barh(top_10_movies['title'], top_10_movies['popularity'], color='salmon')
plt.xlabel('Popularity')
plt.ylabel('Movie Title')
plt.title('Top 10 Movies by Popularity')
plt.gca().invert_yaxis()
plt.show()
```



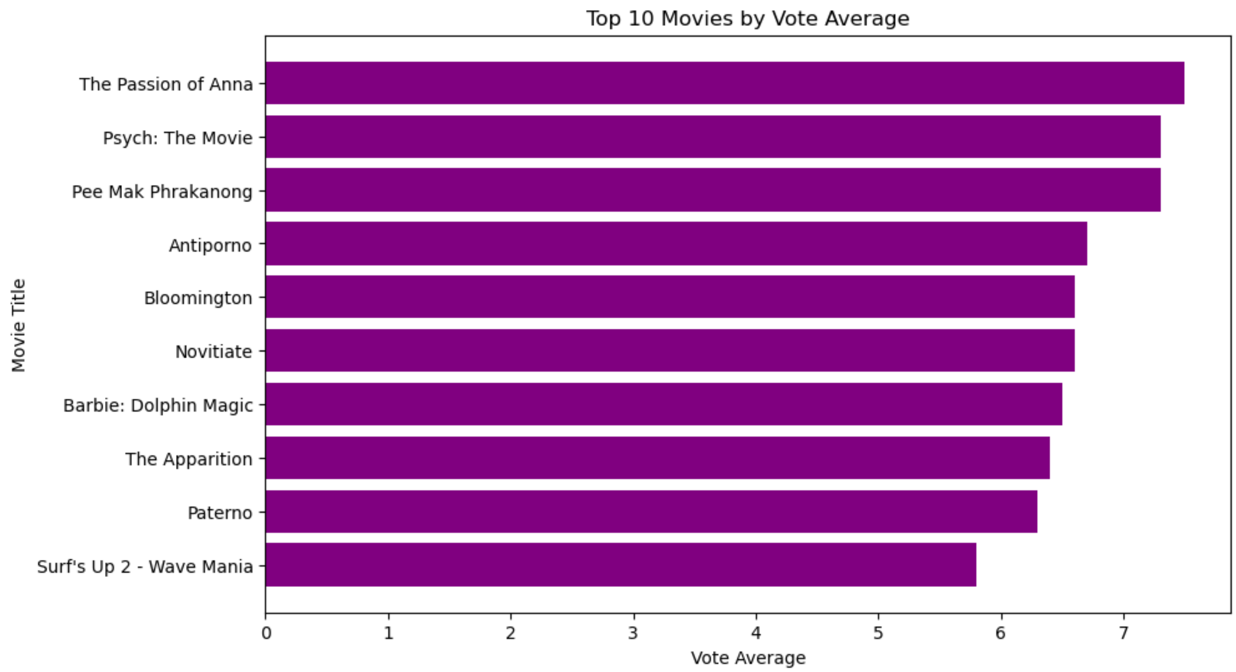
Ploting a Bar Graph for the top 10 movies based on the vote counts

In [503]:
```
top_performing_movies = top_performing_movies.sort_values(by='vote_count', ascending=Fa

# Select top 10 movies
top_10_movies = top_performing_movies.head(10)

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.barh(top_10_movies['title'], top_10_movies['vote_count'], color='lightgreen')
plt.xlabel('Vote Count')
plt.ylabel('Movie Title')
plt.title('Top 10 Movies by Vote Count')
plt.gca().invert_yaxis()
plt.show()
```



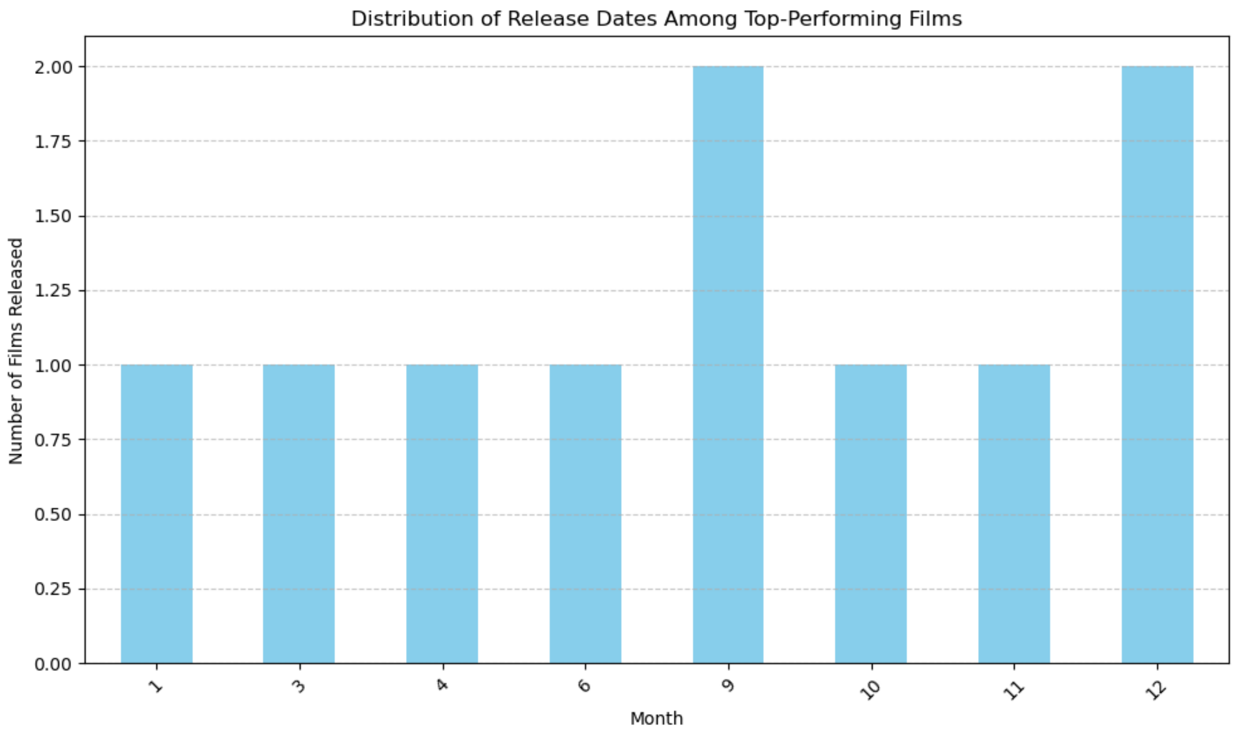
Ploting a Bar Graph for the top 10 movies based on the vote_average

In [504]:
```
top_performing_movies = top_performing_movies.sort_values(by='vote_average', ascending

# Select top 10 movies
top_10_movies = top_performing_movies.head(10)

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.barh(top_10_movies['title'], top_10_movies['vote_average'], color='purple')
plt.xlabel('Vote Average')
plt.ylabel('Movie Title')
plt.title('Top 10 Movies by Vote Average')
plt.gca().invert_yaxis()
plt.show()
```



Release Date Analysis

In [505]:
```
# Group films by release month and count the number of films in each month
release_date_distribution = top_performing_movies.groupby('release_month').size()

# Create a bar plot to visualize the distribution of release dates
plt.figure(figsize=(10, 6))
release_date_distribution.plot(kind='bar', color='skyblue')
plt.title('Distribution of Release Dates Among Top-Performing Films')
plt.xlabel('Month')
plt.ylabel('Number of Films Released')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Release Date Analysis: To find the month and year with the highest top ten popularity, vote count, and vote average.

In [506]:
```
# Create a DataFrame for each metric containing only the top ten movies
top_ten_popularity = cleaned_the_movie_db_df.nlargest(10, 'popularity')
top_ten_vote_count = cleaned_the_movie_db_df.nlargest(10, 'vote_count')
top_ten_vote_average = cleaned_the_movie_db_df.nlargest(10, 'vote_average')

# Count occurrences of each combination of month and year for each metric
popularity_counts = top_ten_popularity.groupby(['release_year', 'release_month']).size
vote_count_counts = top_ten_vote_count.groupby(['release_year', 'release_month']).size
vote_average_counts = top_ten_vote_average.groupby(['release_year', 'release_month']).:

# Identify the month and year with the highest count for each metric
highest_popularity_month_year = popularity_counts.idxmax()
highest_vote_count_month_year = vote_count_counts.idxmax()
highest_vote_average_month_year = vote_average_counts.idxmax()

print("Month and Year with the Highest Top Ten Popularity:", highest_popularity_month_y
print("Month and Year with the Highest Top Ten Vote Count:", highest_vote_count_month_y
print("Month and Year with the Highest Top Ten Vote Average:", highest_vote_average_mor
```

```
Month and Year with the Highest Top Ten Popularity: (2018, 4)
Month and Year with the Highest Top Ten Vote Count: (2012, 12)
Month and Year with the Highest Top Ten Vote Average: (2010, 6)
```

Original_language Analysis: Analyze the distribution of original languages among the top-performing films to understand audience preferences regarding language.

In [507]:
```
# Create a DataFrame for each metric containing only the top ten movies
top_ten_popularity = cleaned_the_movie_db_df.nlargest(10, 'popularity')
top_ten_vote_count = cleaned_the_movie_db_df.nlargest(10, 'vote_count')
top_ten_vote_average = cleaned_the_movie_db_df.nlargest(10, 'vote_average')

# Count the occurrences of each original_language in the top ten for each metric
popularity_by_language = top_ten_popularity['original_language'].value_counts()
vote_count_by_language = top_ten_vote_count['original_language'].value_counts()
vote_average_by_language = top_ten_vote_average['original_language'].value_counts()

# Find the language with the highest count for each metric
highest_popularity_language = popularity_by_language.idxmax()
highest_vote_count_language = vote_count_by_language.idxmax()
highest_vote_average_language = vote_average_by_language.idxmax()

print("Language with the Highest Top Ten Popularity:", highest_popularity_language)
print("Language with the Highest Top Ten Vote Count:", highest_vote_count_language)
print("Language with the Highest Top Ten Vote Average:", highest_vote_average_language]
```

```
Language with the Highest Top Ten Popularity: en
Language with the Highest Top Ten Vote Count: en
Language with the Highest Top Ten Vote Average: en
```

In [ ]:
```

```