In [615]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sqlite3
import numpy as np
import csv
import json
```

```
Importing the Box Office Mojo csv
```

In [616]:
```
BOM_data = pd.read_csv('bom.movie_gross.csv')
```

In [617]:
```
BOM_data.head()
```

Out[617]:

	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

In [ ]:
```
Importing the Rotten Tomato movie infromation csv
```

In [618]:
```
RT_info = pd.read_csv('rt.movie_info.tsv', sep='\t')
```

In [619]:
```
RT_info.head()
```

Out[619]:

	id	synopsis	rating	genre	director	writer	theater_date	dvd_date	currency	box_office	runtime	studio
0	1	This gritty, fast-paced, and innovative police...	R	Action and Adventure|Classics|Drama	William Friedkin	Ernest Tidyman	Oct 9, 1971	Sep 25, 2001	NaN	NaN	104 minutes	NaN
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	David Cronenberg|Don DeLillo	Aug 17, 2012	Jan 1, 2013	$	600,000	108 minutes	Entertainment One
2	5	Illeana Douglas delivers a superb performance ...	R	Drama|Musical and Performing Arts	Allison Anders	Allison Anders	Sep 13, 1996	Apr 18, 2000	NaN	NaN	116 minutes	NaN
3	6	Michael Douglas runs afoul of a treacherous su...	R	Drama|Mystery and Suspense	Barry Levinson	Paul Attanasio|Michael Crichton	Dec 9, 1994	Aug 27, 1997	NaN	NaN	128 minutes	NaN
4	7	NaN	NR	Drama|Romance	Rodney Bennett	Giles Cooper	NaN	NaN	NaN	NaN	200 minutes	NaN

In [620]:
```
RT_info
```

Out[620]:

	id	synopsis	rating	genre	director	writer	theater_date	dvd_date	currency	box_office	runtime	
0	1	This gritty, fast-paced, and innovative police...	R	Action and Adventure|Classics|Drama	William Friedkin	Ernest Tidyman	Oct 9, 1971	Sep 25, 2001	NaN	NaN	104 minutes	
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	David Cronenberg|Don DeLillo	Aug 17, 2012	Jan 1, 2013	$	600,000	108 minutes	Ente
2	5	Illeana Douglas delivers a superb performance ...	R	Drama|Musical and Performing Arts	Allison Anders	Allison Anders	Sep 13, 1996	Apr 18, 2000	NaN	NaN	116 minutes	
3	6	Michael Douglas runs afoul of a treacherous su...	R	Drama|Mystery and Suspense	Barry Levinson	Paul Attanasio|Michael Crichton	Dec 9, 1994	Aug 27, 1997	NaN	NaN	128 minutes	
4	7	NaN	NR	Drama|Romance	Rodney Bennett	Giles Cooper	NaN	NaN	NaN	NaN	200 minutes	
...	...	...	...	...	...	...	...	...	...	...	...	
1555	1996	Forget terrorists or hijackers -- there's a ha...	R	Action and Adventure|Horror|Mystery and Suspense	NaN	NaN	Aug 18, 2006	Jan 2, 2007	$	33,886,034	106 minutes	
1556	1997	The popular Saturday Night Live sketch was exp...	PG	Comedy|Science Fiction and Fantasy	Steve Barron	Terry Turner|Tom Davis|Dan Aykroyd|Bonnie Turner	Jul 23, 1993	Apr 17, 2001	NaN	NaN	88 minutes	F
1557	1998	Based on a novel by Richard Powell, when the l...	G	Classics|Comedy|Drama|Musical and Performing Arts	Gordon Douglas	NaN	Jan 1, 1962	May 11, 2004	NaN	NaN	111 minutes	
1558	1999	The Sandlot is a coming-of-age story about a g...	PG	Comedy|Drama|Kids and Family|Sports and Fitness	David Mickey Evans	David Mickey Evans|Robert Gunter	Apr 1, 1993	Jan 29, 2002	NaN	NaN	101 minutes	
1559	2000	Suspended from the force, Paris cop Hubert is ...	R	Action and Adventure|Art House and Internation...	NaN	Luc Besson	Sep 27, 2001	Feb 11, 2003	NaN	NaN	94 minutes	

1560 rows × 12 columns

Importing the Rotten Tomato movie reviews csv

In [622]:
```
RT_reviews = pd.read_csv('rt.reviews.tsv', sep='\t', encoding='unicode_escape')
```

In [623]:
```
RT_reviews.head()
```

Out[623]:

	id	review	rating	fresh	critic	top_critic	publisher	date
0	3	A distinctly gallows take on contemporary fina...	3/5	fresh	PJ Nabarro	0	Patrick Nabarro	November 10, 2018
1	3	It's an allegory in search of a meaning that n...	NaN	rotten	Annalee Newitz	0	io9.com	May 23, 2018
2	3	... life lived in a bubble in financial dealin...	NaN	fresh	Sean Axmaker	0	Stream on Demand	January 4, 2018
3	3	Continuing along a line introduced in last yea...	NaN	fresh	Daniel Kasman	0	MUBI	November 16, 2017
4	3	... a perverse twist on neorealism...	NaN	fresh	NaN	0	Cinema Scope	October 12, 2017

Importing The Movie Database csv

In [625]:
```
TMDB_data = pd.read_csv('tmdb.movies.csv')
```

In [626]:
```
TMDB_data
```

Out[626]:

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count
0	0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	10788
1	1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610
2	2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368
3	3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174
4	4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186
...	...	...	...	...	...	...	...	...	...	...
26512	26512	[27, 18]	488143	en	Laboratory Conditions	0.600	2018-10-13	Laboratory Conditions	0.0	1
26513	26513	[18, 53]	485975	en	_EXHIBIT_84xxx_	0.600	2018-05-01	_EXHIBIT_84xxx_	0.0	1
26514	26514	[14, 28, 12]	381231	en	The Last One	0.600	2018-10-01	The Last One	0.0	1
26515	26515	[10751, 12, 28]	366854	en	Trailer Made	0.600	2018-06-22	Trailer Made	0.0	1
26516	26516	[53, 27]	309885	en	The Church	0.600	2018-10-05	The Church	0.0	1

26517 rows × 10 columns

```
Importing The Numbers movie budgets csv
```

In [627]:
```
MB_data = pd.read_csv('tn.movie_budgets.csv')
```

In [628]:
```
MB_data
```

0	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875
2	3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
3	4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747
...	...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	$7,000	$0	$0
5778	79	Apr 2, 1999	Following	$6,000	$48,482	$240,495
5779	80	Jul 13, 2005	Return to the Land of Wonders	$5,000	$1,338	$1,338
5780	81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0
5781	82	Aug 5, 2005	My Date With Drew	$1,100	$181,041	$181,041

5782 rows × 6 columns

In [630]:
```
conn = sqlite3.connect('im.db')
cursor = conn.cursor()
```

```
Checking all the available tables, including the csv's that were converted to SQL tables
```

In [631]:
```
query = ("""

SELECT name
FROM sqlite_master
WHERE type='table'



""")

tables = pd.read_sql(query , conn)
print(tables)
```

```
                 name
0        movie_basics
1           directors
2           known_for
3          movie_akas
4       movie_ratings
5             persons
6          principals
7             writers
8           BOM_table
9          TMDB_table
10           MB_table
11        TMDB_table_R
12      combined_table
13           FC_table
14              IM_FC
15           PG_table
16           WS_table
17            g_table
18          PID_table
19          ROI_table
20          FIN_table
21        FINAL_table
22        ROI_table_2
23           FINAL_tb
24          FLT_table
25         PRIN_table
26          DIR_table
27       TEMP_2_table
28    Final_data_table
29     Final_PRN_table
30       FINAL_table_1
```

In [632]:
```
query = ("""

SELECT *
FROM movie_basics






""")

df = pd.read_sql(query , conn)
print(df)
```

```
        movie_id                                primary_title  \
0       tt0063540                                    Sunghursh   
1       tt0066787              One Day Before the Rainy Season   
2       tt0069049                   The Other Side of the Wind   
3       tt0069204                              Sabse Bada Sukh   
4       tt0100275                     The Wandering Soap Opera   
...           ...                                          ...   
146139  tt9916538                          Kuambil Lagi Hatiku   
146140  tt9916622  Rodolpho Teóphilo - O Legado de um Pioneiro   
146141  tt9916706                              Dankyavar Danka   
146142  tt9916730                                       6 Gunn   
146143  tt9916754               Chico Albuquerque - Revelações   

                                     original_title  start_year  \
0                                         Sunghursh        2013   
1                                   Ashad Ka Ek Din        2019   
2                        The Other Side of the Wind        2018   
3                                   Sabse Bada Sukh        2018   
4                             La Telenovela Errante        2017   
...                                             ...         ...   
146139                          Kuambil Lagi Hatiku        2019   
146140  Rodolpho Teóphilo - O Legado de um Pioneiro        2015   
146141                              Dankyavar Danka        2013   
146142                                       6 Gunn        2017   
146143               Chico Albuquerque - Revelações        2013   

        runtime_minutes                genres  
0                 175.0    Action,Crime,Drama  
1                 114.0       Biography,Drama  
2                 122.0                 Drama  
3                   NaN          Comedy,Drama  
4                  80.0  Comedy,Drama,Fantasy  
...                 ...                   ...  
146139            123.0                 Drama  
146140              NaN           Documentary  
146141              NaN                Comedy  
146142            116.0                  None  
146143              NaN           Documentary  

[146144 rows x 6 columns]
```

In [674]:
```
query = ("""

SELECT *
FROM directors






""")

df = pd.read_sql(query , conn)
print(df)
```

```
         movie_id   person_id
0       tt0285252   nm0899854
1       tt0462036   nm1940585
2       tt0835418   nm0151540
3       tt0835418   nm0151540
4       tt0878654   nm0089502
...           ...         ...
291169  tt8999974  nm10122357
291170  tt9001390   nm6711477
291171  tt9001494  nm10123242
291172  tt9001494  nm10123248
291173  tt9004986   nm4993825

[291174 rows x 2 columns]
```

In [675]:
```
query = ("""
SELECT name
FROM sqlite_master
WHERE type='table' AND name='TMDB_table'

""")


table_exists = pd.read_sql(query, conn)


if table_exists.empty:
    TMDB_data.to_sql("TMDB_table", conn, index=False)
else:
    print("Table 'TMDB_table' already exists, skipping creation.")
```

```
Table 'TMDB_table' already exists, skipping creation.
```

In [673]:
```
query = ("""

SELECT *
FROM TMDB_table

""")

df = pd.read_sql(query, conn)
print(df)
```

```
      Unnamed: 0           genre_ids      id original_language  \
0              0      [12, 14, 10751]   12444                en
1              1  [14, 12, 16, 10751]   10191                en
2              2        [12, 28, 878]   10138                en
3              3      [16, 35, 10751]     862                en
4              4        [28, 878, 12]   27205                en
...          ...                  ...     ...               ...
26512      26512             [27, 18]  488143                en
26513      26513             [18, 53]  485975                en
26514      26514         [14, 28, 12]  381231                en
26515      26515      [10751, 12, 28]  366854                en
26516      26516             [53, 27]  309885                en

                                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
...                                          ...         ...          ...
26512                      Laboratory Conditions       0.600   2018-10-13
26513                            _EXHIBIT_84xxx_       0.600   2018-05-01
26514                               The Last One       0.600   2018-10-01
26515                               Trailer Made       0.600   2018-06-22
26516                                 The Church       0.600   2018-10-05

                                         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
...                                          ...           ...         ...
26512                      Laboratory Conditions           0.0           1
26513                            _EXHIBIT_84xxx_           0.0           1
26514                               The Last One           0.0           1
26515                               Trailer Made           0.0           1
26516                                 The Church           0.0           1

[26517 rows x 10 columns]
```

In [638]:
```
TMDB_data.drop('Unnamed: 0', axis = 1, inplace = True)
```

In [639]:
```
TMDB_data
```

Out[639]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count
0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	10788
1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610
2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368
3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174
4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186
...	...	...	...	...	...	...	...	...	...
26512	[27, 18]	488143	en	Laboratory Conditions	0.600	2018-10-13	Laboratory Conditions	0.0	1
26513	[18, 53]	485975	en	_EXHIBIT_84xxx_	0.600	2018-05-01	_EXHIBIT_84xxx_	0.0	1
26514	[14, 28, 12]	381231	en	The Last One	0.600	2018-10-01	The Last One	0.0	1
26515	[10751, 12, 28]	366854	en	Trailer Made	0.600	2018-06-22	Trailer Made	0.0	1
26516	[53, 27]	309885	en	The Church	0.600	2018-10-05	The Church	0.0	1

26517 rows × 9 columns

In [640]:
```
query = ("""
SELECT name
FROM sqlite_master
WHERE type='table' AND name='TMDB_table_R'

""")


table_exists = pd.read_sql(query, conn)


if table_exists.empty:
    TMDB_data.to_sql("TMDB_table_R", conn, index=False)
else:
    print("Table 'TMDB_table_R' already exists, skipping creation.")
```

```
Table 'TMDB_table_R' already exists, skipping creation.
```

In [641]:
```
query = ("""

SELECT *
FROM TMDB_table_R
JOIN BOM_table
USING (title)

""")

combined_df = pd.read_sql(query, conn)
print(combined_df)
```

```
2699       0.600   2018-04-13              The Judge           7.5
2700       0.600   2018-08-08                Flowers           6.0
2701       0.600   2018-11-09            Last Letter           6.0
2702       0.600   2018-11-25                   Eden           0.0

      vote_count studio  domestic_gross foreign_gross  year
0           7610   P/DW     217600000.0     277300000  2010
1          12368   Par.     312400000.0     311500000  2010
2          22186     WB     292600000.0     535700000  2010
3           8340     BV     415000000.0     652000000  2010
4          10057   Uni.     251500000.0     291600000  2010
...          ...    ...             ...           ...   ...
2698           1    ORF      45100000.0      53200000  2015
2699           2     WB      47100000.0      37300000  2014
2700           1   MBox         61600.0          None  2015
2701           1     CL        181000.0          None  2018
2702           1     BG         65500.0          None  2015

[2703 rows x 13 columns]
```

In [642]:
```
combined_df
```

Out[642]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count	studio	domestic_gross	foreigr
0	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610	P/DW	217600000.0	277
1	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368	Par.	312400000.0	311
2	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186	WB	292600000.0	535
3	[16, 10751, 35]	10193	en	Toy Story 3	24.445	2010-06-17	Toy Story 3	7.7	8340	BV	415000000.0	652
4	[16, 10751, 35]	20352	en	Despicable Me	23.673	2010-07-09	Despicable Me	7.2	10057	Uni.	251500000.0	291
...	...	...	...	...	...	...	...	...	...	...	...	
2698	[]	501956	en	Spotlight	0.600	2018-01-28	Spotlight	10.0	1	ORF	45100000.0	53
2699	[99, 99]	474464	en	The Judge	0.600	2018-04-13	The Judge	7.5	2	WB	47100000.0	37
2700	[18, 10751]	574534	fr	Des fleurs	0.600	2018-08-08	Flowers	6.0	1	MBox	61600.0	
2701	[10749, 18]	551634	zh	你好，之华	0.600	2018-11-09	Last Letter	6.0	1	CL	181000.0	
2702	[]	561861	en	Eden	0.600	2018-11-25	Eden	0.0	1	BG	65500.0	

2703 rows × 13 columns

In [644]:
```
query = ("""
SELECT name
FROM sqlite_master
WHERE type='table' AND name='combined_table'

""")


table_exists = pd.read_sql(query, conn)


if table_exists.empty:
    TMDB_data.to_sql("combined_table", conn, index=False)
else:
    print("Table 'combined_table' already exists, skipping creation.")
```

```
Table 'combined_table' already exists, skipping creation.
```

In [645]:
```
query = ("""

SELECT *
FROM combined_table AS c
JOIN MB_table AS m
ON c.title = m.movie

""")

combined_df_final = pd.read_sql(query, conn)
print(combined_df_final)
```

```
               genre_ids      id original_language  \
0     [14, 12, 16, 10751]   10191                en
1          [12, 28, 878]   10138                en
2          [28, 878, 12]   27205                en
3        [16, 10751, 35]   10193                en
4        [16, 10751, 35]   20352                en
...                  ...     ...               ...
1390     [18, 35, 10749]  499722                fr
1391        [28, 12, 16]  332718                en
1392                  []  501956                en
1393            [99, 99]  474464                en
1394                  []  561861                en

                original_title  popularity release_date  \
0       How to Train Your Dragon      28.734   2010-03-26
1                     Iron Man 2      28.515   2010-05-07
2                      Inception      27.920   2010-07-16
3                    Toy Story 3      24.445   2010-06-17
4                  Despicable Me      23.673   2010-07-09
...                        ...         ...          ...
1390         Amoureux de ma femme       6.359   2018-04-30
1391  Bilal: A New Breed of Hero       2.707   2018-02-02
1392                   Spotlight       0.600   2018-01-28
1393                   The Judge       0.600   2018-04-13
1394                        Eden       0.600   2018-11-25

                           title  vote_average  vote_count studio  \
0       How to Train Your Dragon           7.7        7610   P/DW
1                     Iron Man 2           6.8       12368   Par.
2                      Inception           8.3       22186     WB
3                    Toy Story 3           7.7        8340     BV
4                  Despicable Me           7.2       10057   Uni.
...                        ...           ...         ...    ...
1390             The Other Woman           4.9          60    IFC
1391  Bilal: A New Breed of Hero           6.8          54     VE
1392                   Spotlight          10.0           1    ORF
1393                   The Judge           7.5           2     WB
1394                        Eden           0.0           1     BG

     domestic_gross foreign_gross  year   id  release_date  \
0       217600000.0     277300000  2010   30  Mar 26, 2010
1       312400000.0     311500000  2010   15   May 7, 2010
2       292600000.0     535700000  2010   38  Jul 16, 2010
3       415000000.0     652000000  2010   47  Jun 18, 2010
4       251500000.0     291600000  2010   50   Jul 9, 2010
...             ...           ...   ...  ...           ...
1390        25400.0        427000  2011    2  Apr 25, 2014
1391       491000.0       1700000  2018  100   Feb 2, 2018
1392     45100000.0      53200000  2015   34   Nov 6, 2015
1393     47100000.0      37300000  2014   41  Oct 10, 2014
1394        65500.0          None  2015   66  Jan 19, 2016

                           movie production_budget domestic_gross  \
0       How to Train Your Dragon      $165,000,000   $217,581,232
1                     Iron Man 2      $170,000,000   $312,433,331
2                      Inception      $160,000,000   $292,576,195
3                    Toy Story 3      $200,000,000   $415,004,880
4                  Despicable Me       $69,000,000   $251,513,985
...                        ...               ...            ...
1390             The Other Woman       $40,000,000    $83,911,193
1391  Bilal: A New Breed of Hero       $30,000,000       $490,973
1392                   Spotlight       $20,000,000    $45,055,776
1393                   The Judge       $50,000,000    $47,119,388
1394                        Eden        $2,300,000             $0

     worldwide_gross
0       $494,870,992
1       $621,156,389
2       $835,524,642
3     $1,068,879,522
4       $543,464,573
...              ...
1390    $195,111,193
1391        $648,599
1392     $92,088,460
1393     $76,119,388
1394              $0

[1395 rows x 19 columns]
```

In [646]:
```
temp_combined_df = combined_df_final.drop("id", axis=1)
```

In [647]:
```
temp_combined_df
```

Out[647]:

	genre_ids	original_language	original_title	popularity	release_date	title	vote_average	vote_count	studio	domestic_gross	foreign_gross
0	[14, 12, 16, 10751]	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610	P/DW	217600000.0	277300000
1	[12, 28, 878]	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368	Par.	312400000.0	311500000
2	[28, 878, 12]	en	Inception	27.920	2010-07-16	Inception	8.3	22186	WB	292600000.0	535700000
3	[16, 10751, 35]	en	Toy Story 3	24.445	2010-06-17	Toy Story 3	7.7	8340	BV	415000000.0	652000000
4	[16, 10751, 35]	en	Despicable Me	23.673	2010-07-09	Despicable Me	7.2	10057	Uni.	251500000.0	291600000
...	...	...	...	...	...	...	...	...	...	...	...
1390	[18, 35, 10749]	fr	Amoureux de ma femme	6.359	2018-04-30	The Other Woman	4.9	60	IFC	25400.0	427000
1391	[28, 12, 16]	en	Bilal: A New Breed of Hero	2.707	2018-02-02	Bilal: A New Breed of Hero	6.8	54	VE	491000.0	1700000
1392	[]	en	Spotlight	0.600	2018-01-28	Spotlight	10.0	1	ORF	45100000.0	53200000
1393	[99, 99]	en	The Judge	0.600	2018-04-13	The Judge	7.5	2	WB	47100000.0	37300000
1394	[]	en	Eden	0.600	2018-11-25	Eden	0.0	1	BG	65500.0	None

1395 rows × 17 columns

Decided to drop genre_ids, original_title and title. I wanted to have doemstic,foreign and worldwide gross. I then decdied to drop the first domestic_gross and foreign gross column, include the remaining domestic and worldwide gross and calculate the forieng gross using these remaining columns.

In [648]:
```
temp_combined_df.drop("genre_ids", axis=1, inplace=True )
```

```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/pandas/core/frame.py:4163: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  return super().drop(
```

In [649]:
```
temp_combined_df
```

Out[649]:

	original_language	original_title	popularity	release_date	title	vote_average	vote_count	studio	domestic_gross	foreign_gross	year	relea
0	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610	P/DW	217600000.0	277300000	2010	Mar :
1	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368	Par.	312400000.0	311500000	2010	May
2	en	Inception	27.920	2010-07-16	Inception	8.3	22186	WB	292600000.0	535700000	2010	Jul
3	en	Toy Story 3	24.445	2010-06-17	Toy Story 3	7.7	8340	BV	415000000.0	652000000	2010	Jun
4	en	Despicable Me	23.673	2010-07-09	Despicable Me	7.2	10057	Uni.	251500000.0	291600000	2010	Ju
...	...	...	...	...	...	...	...	...	...	...	...	
1390	fr	Amoureux de ma femme	6.359	2018-04-30	The Other Woman	4.9	60	IFC	25400.0	427000	2011	Apr :
1391	en	Bilal: A New Breed of Hero	2.707	2018-02-02	Bilal: A New Breed of Hero	6.8	54	VE	491000.0	1700000	2018	Feb
1392	en	Spotlight	0.600	2018-01-28	Spotlight	10.0	1	ORF	45100000.0	53200000	2015	Nov
1393	en	The Judge	0.600	2018-04-13	The Judge	7.5	2	WB	47100000.0	37300000	2014	Oct
1394	en	Eden	0.600	2018-11-25	Eden	0.0	1	BG	65500.0	None	2015	Jan

1395 rows × 16 columns

In [650]:
```
print(temp_combined_df.columns)
```

```
Index(['original_language', 'original_title', 'popularity', 'release_date',
       'title', 'vote_average', 'vote_count', 'studio', 'domestic_gross',
       'foreign_gross', 'year', 'release_date', 'movie', 'production_budget',
       'domestic_gross', 'worldwide_gross'],
      dtype='object')
```

```
This line removes the column at index 8
```

In [651]:
```
temp_combined_df = temp_combined_df.iloc[:, [i for i in range(temp_combined_df.shape[1]) if i != 8]]
```

In [652]:
```
temp_combined_df
```

Out[652]:

	original_language	original_title	popularity	release_date	title	vote_average	vote_count	studio	foreign_gross	year	release_date	mo
0	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610	P/DW	277300000	2010	Mar 26, 2010	How Train Yc Drag
1	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368	Par.	311500000	2010	May 7, 2010	Iron Mar
2	en	Inception	27.920	2010-07-16	Inception	8.3	22186	WB	535700000	2010	Jul 16, 2010	Incepti
3	en	Toy Story 3	24.445	2010-06-17	Toy Story 3	7.7	8340	BV	652000000	2010	Jun 18, 2010	Toy Stc
4	en	Despicable Me	23.673	2010-07-09	Despicable Me	7.2	10057	Uni.	291600000	2010	Jul 9, 2010	Despical I
...	...	...	...	...	...	...	...	...	...	...	...	
1390	fr	Amoureux de ma femme	6.359	2018-04-30	The Other Woman	4.9	60	IFC	427000	2011	Apr 25, 2014	The Oth Wom
1391	en	Bilal: A New Breed of Hero	2.707	2018-02-02	Bilal: A New Breed of Hero	6.8	54	VE	1700000	2018	Feb 2, 2018	Bilal New Bre of He
1392	en	Spotlight	0.600	2018-01-28	Spotlight	10.0	1	ORF	53200000	2015	Nov 6, 2015	Spotlig
1393	en	The Judge	0.600	2018-04-13	The Judge	7.5	2	WB	37300000	2014	Oct 10, 2014	The Jud
1394	en	Eden	0.600	2018-11-25	Eden	0.0	1	BG	None	2015	Jan 19, 2016	Ed

1395 rows × 15 columns

temp_combined_df_2 = temp_combined_df.iloc[:, [i for i in range(temp_combined_df_2.shape[1]) if i != 9]] temp_combined_df_2

In [653]:
```
temp_combined_df.drop("original_title", axis =1, inplace = True)
```

```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/pandas/core/frame.py:4163: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#re
turning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
a-view-versus-a-copy)
  return super().drop(
```

In [654]:
```
temp_combined_df.drop("title", axis =1, inplace = True)
```

In [655]:
```
temp_combined_df
```

Out[655]:

	original_language	popularity	release_date	vote_average	vote_count	studio	foreign_gross	year	release_date	movie	production_budget	d
0	en	28.734	2010-03-26	7.7	7610	P/DW	277300000	2010	Mar 26, 2010	How to Train Your Dragon	$165,000,000	
1	en	28.515	2010-05-07	6.8	12368	Par.	311500000	2010	May 7, 2010	Iron Man 2	$170,000,000	
2	en	27.920	2010-07-16	8.3	22186	WB	535700000	2010	Jul 16, 2010	Inception	$160,000,000	
3	en	24.445	2010-06-17	7.7	8340	BV	652000000	2010	Jun 18, 2010	Toy Story 3	$200,000,000	
4	en	23.673	2010-07-09	7.2	10057	Uni.	291600000	2010	Jul 9, 2010	Despicable Me	$69,000,000	
...	...	...	...	...	...	...	...	...	...	...	...	
1390	fr	6.359	2018-04-30	4.9	60	IFC	427000	2011	Apr 25, 2014	The Other Woman	$40,000,000	
1391	en	2.707	2018-02-02	6.8	54	VE	1700000	2018	Feb 2, 2018	Bilal: A New Breed of Hero	$30,000,000	
1392	en	0.600	2018-01-28	10.0	1	ORF	53200000	2015	Nov 6, 2015	Spotlight	$20,000,000	
1393	en	0.600	2018-04-13	7.5	2	WB	37300000	2014	Oct 10, 2014	The Judge	$50,000,000	
1394	en	0.600	2018-11-25	0.0	1	BG	None	2015	Jan 19, 2016	Eden	$2,300,000	

1395 rows × 13 columns

In [656]:
```
query = ("""
SELECT name
FROM sqlite_master
WHERE type='table' AND name='FC_table'

""")


table_exists = pd.read_sql(query, conn)


if table_exists.empty:
    TMDB_data.to_sql("FC_table", conn, index=False)
else:
    print("Table 'FC_table' already exists, skipping creation.")
```

```
Table 'FC_table' already exists, skipping creation.
```

```
Sorting movies based on populairty
```

In [657]:
```
query = ("""

SELECT movie, popularity
FROM FC_table
ORDER BY popularity DESC

""")
pd.read_sql(query, conn)
```

Out[657]:

	movie	popularity
0	Avengers: Infinity War	80.773
1	John Wick	78.123
2	The Hobbit: The Battle of the Five Armies	53.783
3	Guardians of the Galaxy	49.606
4	Blade Runner 2049	48.571
...	...	...
1390	Mandy	0.600
1391	Paranoia	0.600
1392	Spotlight	0.600
1393	The Judge	0.600
1394	Eden	0.600

1395 rows × 2 columns

In [46]:
```
temp_combined_df.to_csv("CC_table.csv", index = False)
```

In [659]:
```
query = ("""
SELECT COUNT(original_language), original_language
FROM FC_table
GROUP BY original_language

""")
pd.read_sql(query, conn)
```

Out[659]:

	COUNT(original_language)	original_language
0	2	ar
1	1	da
2	1	de
3	1	el
4	1344	en
5	4	es
6	1	fa
7	11	fr
8	1	he
9	7	hi
10	1	hu

In [661]:
```
query = ("""
SELECT *
FROM movie_akas

""")
pd.read_sql(query, conn)
```

Out[661]:

	movie_id	ordering	title	region	language	types	attributes	is_original_title
0	tt0369610	10	Джурасик свят	BG	bg	None	None	0.0
1	tt0369610	11	Jurashikku warudo	JP	None	imdbDisplay	None	0.0
2	tt0369610	12	Jurassic World: O Mundo dos Dinossauros	BR	None	imdbDisplay	None	0.0
3	tt0369610	13	O Mundo dos Dinossauros	BR	None	None	short title	0.0
4	tt0369610	14	Jurassic World	FR	None	imdbDisplay	None	0.0
...	...	...	...	...	...	...	...	...
331698	tt9827784	2	Sayonara kuchibiru	None	None	original	None	1.0
331699	tt9827784	3	Farewell Song	XWW	en	imdbDisplay	None	0.0
331700	tt9880178	1	La atención	None	None	original	None	1.0
331701	tt9880178	2	La atención	ES	None	None	None	0.0
331702	tt9880178	3	The Attention	XWW	en	imdbDisplay	None	0.0

331703 rows × 8 columns

In [662]:
```
query = ("""

CREATE TABLE IM_FC AS
SELECT m.*, f.*
FROM movie_akas as m
JOIN FC_table as f
ON m.title = f.movie

""")
cursor.execute(query)
conn.commit()
cursor.close()
```

```
---------------------------------------------------------------------------
OperationalError                          Traceback (most recent call last)
<ipython-input-662-d65ade0c9236> in <module>
      8 
      9 """)
---> 10 cursor.execute(query)
     11 conn.commit()
     12 cursor.close()

OperationalError: table IM_FC already exists
```

In [676]:
```
query = ("""

SELECT *
FROM IM_FC

""")

pd.read_sql(query, conn)
```

Out[676]:

	movie_id	ordering	title	region	language	types	attributes	is_original_title	original_language	popularity	release_date	vote_avera
0	tt0369610	14	Jurassic World	FR	None	imdbDisplay	None	0.0	en	20.709	2015-06-12	
1	tt0369610	15	Jurassic World	GR	None	imdbDisplay	None	0.0	en	20.709	2015-06-12	
2	tt0369610	16	Jurassic World	IT	None	imdbDisplay	None	0.0	en	20.709	2015-06-12	
3	tt0369610	20	Jurassic World	SE	None	imdbDisplay	None	0.0	en	20.709	2015-06-12	
4	tt0369610	29	Jurassic World	US	None	None	None	0.0	en	20.709	2015-06-12	
...	...	...	...	...	...	...	...	...	...	...	...	
7725	tt5462602	17	The Big Sick	US	None	imdbDisplay	None	0.0	en	12.322	2017-06-23	
7726	tt5462602	3	The Big Sick	None	None	original	None	1.0	en	12.322	2017-06-23	
7727	tt5462602	8	The Big Sick	FR	None	imdbDisplay	None	0.0	en	12.322	2017-06-23	
7728	tt7098772	1	Unstoppable	HR	None	None	None	0.0	en	0.600	2013-09-24	
7729	tt7098772	1	Unstoppable	HR	None	None	None	0.0	en	14.010	2010-11-12	

7730 rows × 18 columns

In [664]:
```
query = ("""

SELECT *
FROM IM_FC
WHERE region = 'US'


""")

movie_df  = pd.read_sql(query, conn)
```

In [665]:
```
movie_df
```

Out[665]:

	movie_id	ordering	title	region	language	types	attributes	is_original_title	original_language	popularity	release_date	vote_average
0	tt0369610	29	Jurassic World	US	None	None	None	0.0	en	20.709	2015-06-12	6.6
1	tt0401729	2	John Carter	US	None	None	None	0.0	en	18.549	2012-03-09	6.1
2	tt1194173	9	The Bourne Legacy	US	None	None	None	0.0	en	18.050	2012-08-10	6.1
3	tt1219289	9	Limitless	US	None	None	None	0.0	en	19.453	2011-03-08	7.1
4	tt1235522	11	Broken City	US	None	None	None	0.0	en	13.646	2013-01-18	5.9
...	...	...	...	...	...	...	...	...	...	...	...	...
1441	tt5902440	1	Project X	US	None	None	None	0.0	en	9.715	2012-03-02	6.4
1442	tt3348730	23	Jigsaw	US	None	imdbDisplay	None	0.0	en	17.398	2017-10-27	6.1
1443	tt3862762	3	Lockout	US	None	None	DVD box title	0.0	en	11.273	2012-04-13	5.9
1444	tt4651520	25	Bad Moms	US	None	imdbDisplay	None	0.0	en	14.332	2016-07-29	6.5
1445	tt5462602	17	The Big Sick	US	None	imdbDisplay	None	0.0	en	12.322	2017-06-23	7.4

1446 rows × 18 columns

The populairity index on IMDB should be view as important becasue it shows how well liked and the lonegevity of a movie outside of just its pure gross numbers. This is important because recently it's all about creating a 'univserse' around a movie, which means that people are trying to creating alternative froms of renvenue that are losely conncected to a movie, such as tv shows, merchandise, spin-off movies, and books.

In [677]:
```
query = ("""

SELECT DISTINCT(title), popularity, production_budget, domestic_gross, worldwide_gross
FROM IM_FC
WHERE region = 'US'
ORDER BY popularity DESC
LIMIT 20

""")

pd.read_sql(query, conn)
```

Out[677]:

	title	popularity	production_budget	domestic_gross	worldwide_gross
0	Avengers: Infinity War	80.773	$300,000,000	$678,815,482	$2,048,134,200
1	John Wick	78.123	$30,000,000	$43,037,835	$76,235,001
2	The Hobbit: The Battle of the Five Armies	53.783	$250,000,000	$255,119,788	$945,577,621
3	Guardians of the Galaxy	49.606	$170,000,000	$333,172,112	$770,867,516
4	Blade Runner 2049	48.571	$185,000,000	$92,054,159	$259,357,408
5	Fantastic Beasts: The Crimes of Grindelwald	48.508	$200,000,000	$159,555,901	$652,220,086
6	Spider-Man: Homecoming	46.775	$175,000,000	$334,201,140	$880,166,350
7	Ant-Man and the Wasp	44.729	$130,000,000	$216,648,740	$623,144,660
8	Avengers: Age of Ultron	44.383	$330,600,000	$459,005,868	$1,403,013,963
9	Black Panther	44.140	$200,000,000	$700,059,566	$1,348,258,224
10	Thor: Ragnarok	43.450	$180,000,000	$315,058,289	$846,980,024
11	Bumblebee	43.078	$102,000,000	$127,195,589	$465,195,589
12	X-Men: Days of Future Past	41.867	$200,000,000	$233,921,534	$747,862,775
13	Mortal Engines	40.095	$100,000,000	$15,951,040	$85,287,417
14	Robin Hood	39.975	$210,000,000	$105,487,148	$322,459,006
15	Robin Hood	39.975	$99,000,000	$30,824,628	$84,747,441
16	X-Men: Apocalypse	39.293	$178,000,000	$155,442,489	$542,537,546
17	Captain America: Civil War	39.137	$250,000,000	$408,084,349	$1,140,069,413
18	Deadpool 2	38.894	$110,000,000	$324,591,735	$786,680,557
19	Thor	38.068	$150,000,000	$181,030,624	$449,326,618

The vast majoirty of the top 20 movies, based off TMDB's popularity index, are all part of a universe. These universe incldue MCU, the Hobbit univserse and X-men unverse.

In [678]:
```
query = ("""

SELECT *
FROM IM_FC
WHERE region = 'US' AND vote_count > 50
ORDER BY vote_average DESC
LIMIT 20

""")

pd.read_sql(query, conn)
```

bDisplay	None	0.0	en	20.101	2017-11-17	8.2	3959	LGF	Wonder	$20,000,000	$132,422,809	$304,6
bDisplay	None	0.0	en	20.101	2017-11-17	8.2	3959	LGF	Wonder	$20,000,000	$132,422,809	$304,6
bDisplay	None	0.0	en	17.808	2017-11-10	8.2	5432	FoxS	Three Billboards Outside Ebbing, Missouri	$12,000,000	$54,513,740	$160,1
bDisplay	None	0.0	en	17.808	2017-11-10	8.2	5432	FoxS	Three Billboards Outside Ebbing, Missouri	$12,000,000	$54,513,740	$160,1
bDisplay	None	0.0	en	19.135	2018-10-19	8.2	636	Fox	The Hate U Give	$23,000,000	$29,719,483	$35,0
None	None	0.0	en	15.608	2018-03-16	8.2	3165	Fox	Love, Simon	$10,000,000	$40,826,341	$65,5
None	None	0.0	en	18.060	2010-02-18	8.1	12625	Par.	Shutter Island	$80,000,000	$128,012,934	$299,4

In [668]:
```
query = ("""

SELECT *
FROM movie_basics

""")

pd.read_sql(query, conn)
```

Out[668]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy
...	...	...	...	...	...	...
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Drama
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Documentary
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Comedy
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	None
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary

146144 rows × 6 columns

In [60]:
```
query = ("""

CREATE TABLE PG_table AS
SELECT m.*, f.*
FROM movie_basics as m
JOIN FC_table as f
ON m.primary_title = f.movie

""")
cursor.execute(query)
conn.commit()
cursor.close()
```

```
---------------------------------------------------------------------------
OperationalError                          Traceback (most recent call last)
<ipython-input-60-0eba2bd25795> in <module>
      8 
      9 """)
---> 10 cursor.execute(query)
     11 conn.commit()
     12 cursor.close()

OperationalError: table PG_table already exists
```

In [669]:
```
query = ("""

SELECT *
FROM PG_table

""")

pd.read_sql(query, conn)
```

114.0	Action,Crime,Drama	en	19.373	2014-09-19	6.3	1685	Uni.	Among the Tombstones	$28,000,000	$26,017,685	$62,1
124.0	Action,Adventure,Sci-Fi	en	20.709	2015-06-12	6.6	14056	Uni.	Jurassic World	$215,000,000	$652,270,625	$1,648,8
119.0	Comedy,Drama	en	12.011	2011-10-27	5.7	652	FD	The Rum Diary	$45,000,000	$13,109,815	$21,5
...	...	...	...	...	...	...	...	...	...	...	
NaN	Horror,Mystery,Thriller	en	11.927	2016-10-07	6.3	3479	Strand	The Girl on the Train	$45,000,000	$75,395,035	$174,2
90.0	Drama	en	11.250	2012-10-19	7.0	1426	Gold.	The First Time	$2,000,000	$17,061	$1
NaN	Action,Drama	en	10.993	2015-12-18	5.9	922	Uni.	Sisters	$30,000,000	$87,044,645	$106,0
84.0	Documentary	en	0.600	2013-09-24	1.6	4	Fox	Unstoppable	$95,000,000	$81,562,942	$165,7
84.0	Documentary	en	14.010	2010-11-12	6.4	1913	Fox	Unstoppable	$95,000,000	$81,562,942	$165,7

In [670]:
```
query = ("""

SELECT *
FROM movie_basics

""")

pd.read_sql(query, conn)
```

0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy
...	...	...	...	...	...	...
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Drama
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Documentary
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Comedy
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	None
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary

Created a genre based table, which has been commented out about becasue the table creation can only be done once

In [117]:
```
#query = ("""
#CREATE TABLE g_table AS
#SELECT m.genres, p.*
#FROM movie_basics AS m
#JOIN PG_table AS p
#USING(movie_id)

#""")
#b
```

```
---------------------------------------------------------------------------
OperationalError                          Traceback (most recent call last)
<ipython-input-117-0dad8afb3648> in <module>
      8 """)
      9
---> 10 conn.execute(query)
     11 conn.commit()

OperationalError: table g_table already exists
```

In [326]:
```
query = ("""

SELECT *
FROM g_table

""")

gen_df =pd.read_sql(query, conn)
```

In [ ]: Filtered out unnecessary columns

In [166]:
```
query = ("""

SELECT DISTINCT(movie),
movie_id, start_year, runtime_minutes, genres, popularity, release_date, vote_average, vote_count, studio, prod
FROM PG_table

""")

PG_df = pd.read_sql(query, conn)
```

In [168]:
```
PG_df
```

Out[168]:

	movie	movie_id	start_year	runtime_minutes	genres	popularity	release_date	vote_average	vote_count	studio	product
0	On the Road	tt0337692	2012	124.0	Adventure,Drama,Romance	8.919	2012-12-21	5.6	518	IFC	
1	The Secret Life of Walter Mitty	tt0359950	2013	114.0	Adventure,Comedy,Drama	10.743	2013-12-25	7.1	4859	Fox	
2	A Walk Among the Tombstones	tt0365907	2014	114.0	Action,Crime,Drama	19.373	2014-09-19	6.3	1685	Uni.	
3	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	$
4	The Rum Diary	tt0376136	2011	119.0	Comedy,Drama	12.011	2011-10-27	5.7	652	FD	
...	...	...	...	...	...	...	...	...	...	...	
1896	The Girl on the Train	tt9799088	2018	NaN	Horror,Mystery,Thriller	11.927	2016-10-07	6.3	3479	Strand	
1897	The First Time	tt9827712	2018	90.0	Drama	11.250	2012-10-19	7.0	1426	Gold.	
1898	Sisters	tt9851050	2019	NaN	Action,Drama	10.993	2015-12-18	5.9	922	Uni.	
1899	Unstoppable	tt9906218	2019	84.0	Documentary	0.600	2013-09-24	1.6	4	Fox	
1900	Unstoppable	tt9906218	2019	84.0	Documentary	14.010	2010-11-12	6.4	1913	Fox	

1901 rows × 13 columns

In [169]:
```
len(PG_df['production_budget'])
```

Out[169]: 1901

In [170]:
```
PG_df.iloc[0][-3]
```

Out[170]: '$25,000,000'

Got rid of all non-numeric characters and converted the gross and budget columns to floats

In [171]:
```
PG_df["production_budget"] = (
    PG_df["production_budget"]
    .str.replace("$", "", regex=False)
    .str.replace(",", "", regex=False)
    .astype(float)
)
```

In [172]:
```
PG_df["domestic_gross"] = (
    PG_df["domestic_gross"]
    .str.replace("$", "", regex=False)
    .str.replace(",", "", regex=False)
    .astype(float)
)
```

In [173]:
```
PG_df["worldwide_gross"] = (
    PG_df["worldwide_gross"]
    .str.replace("$", "", regex=False)
    .str.replace(",", "", regex=False)
    .astype(float)
)
```

In [174]:
```
PG_df
```

Out[174]:

ar	runtime_minutes	genres	popularity	release_date	vote_average	vote_count	studio	production_budget	domestic_gross	worldwide_
12	124.0	Adventure,Drama,Romance	8.919	2012-12-21	5.6	518	IFC	25000000.0	720828.0	9.31330
13	114.0	Adventure,Comedy,Drama	10.743	2013-12-25	7.1	4859	Fox	91000000.0	58236838.0	1.87861
14	114.0	Action,Crime,Drama	19.373	2014-09-19	6.3	1685	Uni.	28000000.0	26017685.0	6.21085
15	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	215000000.0	652270625.0	1.64885
11	119.0	Comedy,Drama	12.011	2011-10-27	5.7	652	FD	45000000.0	13109815.0	2.15447
...	...	...	...	...	...	...	...	...	...	
18	NaN	Horror,Mystery,Thriller	11.927	2016-10-07	6.3	3479	Strand	45000000.0	75395035.0	1.74278
18	90.0	Drama	11.250	2012-10-19	7.0	1426	Gold.	2000000.0	17061.0	1.70610
19	NaN	Action,Drama	10.993	2015-12-18	5.9	922	Uni.	30000000.0	87044645.0	1.06030
19	84.0	Documentary	0.600	2013-09-24	1.6	4	Fox	95000000.0	81562942.0	1.65720
19	84.0	Documentary	14.010	2010-11-12	6.4	1913	Fox	95000000.0	81562942.0	1.65720

In [207]:
```
PG_df['ROI'] = (PG_df['worldwide_gross']/PG_df['production_budget'])*100
```

In [209]:
```
PG_df
```

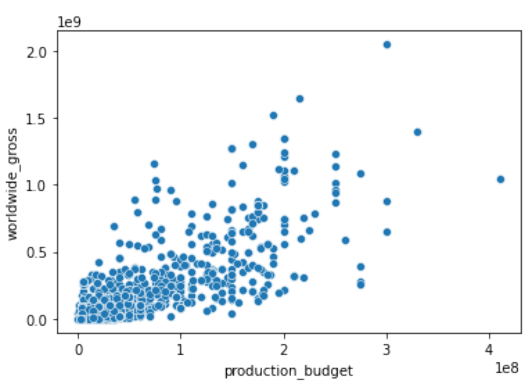
2	Among the Tombstones	tt0365907	2014	114.0	Action,Crime,Drama	19.373	2014-09-19	6.3	1685	Uni.	
3	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	2
4	The Rum Diary	tt0376136	2011	119.0	Comedy,Drama	12.011	2011-10-27	5.7	652	FD	
...	...	...	...	...	...	...	...	...	...	...	
1896	The Girl on the Train	tt9799088	2018	NaN	Horror,Mystery,Thriller	11.927	2016-10-07	6.3	3479	Strand	
1897	The First Time	tt9827712	2018	90.0	Drama	11.250	2012-10-19	7.0	1426	Gold.	
1898	Sisters	tt9851050	2019	NaN	Action,Drama	10.993	2015-12-18	5.9	922	Uni.	
1899	Unstoppable	tt9906218	2019	84.0	Documentary	0.600	2013-09-24	1.6	4	Fox	
1900	Unstoppable	tt9906218	2019	84.0	Documentary	14.010	2010-11-12	6.4	1913	Fox	

1901 rows × 14 columns

In [467]:
```
PG_df.to_csv("L_data_project_2.csv")
```

In [210]:
```
sns.scatterplot(data=PG_df, x='production_budget', y = 'worldwide_gross')
```

Out[210]: <AxesSubplot:xlabel='production_budget', ylabel='worldwide_gross'>



In [212]:
```
top_budget_threshold = PG_df['production_budget'].quantile(0.8)
top_gross_threshold = PG_df['worldwide_gross'].quantile(0.8)
print(top_budget_threshold)
print(top_gross_threshold)
```

```
65000000.0
203127894.0
```

In [213]:
```
query = ("""

SELECT  *
FROM WS_table

""")

pd.read_sql(query, conn)
```

Out[213]:

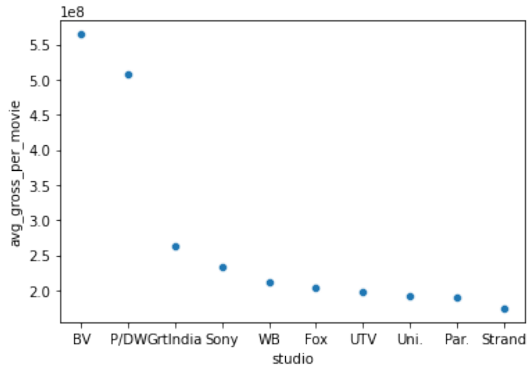
ar	runtime_minutes	genres	popularity	release_date	vote_average	vote_count	studio	production_budget	domestic_gross	worldwide_
12	124.0	Adventure,Drama,Romance	8.919	2012-12-21	5.6	518	IFC	25000000.0	720828.0	9.31330
13	114.0	Adventure,Comedy,Drama	10.743	2013-12-25	7.1	4859	Fox	91000000.0	58236838.0	1.87861
14	114.0	Action,Crime,Drama	19.373	2014-09-19	6.3	1685	Uni.	28000000.0	26017685.0	6.21085
15	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	215000000.0	652270625.0	1.64885
11	119.0	Comedy,Drama	12.011	2011-10-27	5.7	652	FD	45000000.0	13109815.0	2.15447
...	...	...	...	...	...	...	...	...	...	
18	NaN	Horror,Mystery,Thriller	11.927	2016-10-07	6.3	3479	Strand	45000000.0	75395035.0	1.74278
18	90.0	Drama	11.250	2012-10-19	7.0	1426	Gold.	2000000.0	17061.0	1.70610
19	NaN	Action,Drama	10.993	2015-12-18	5.9	922	Uni.	30000000.0	87044645.0	1.06030
19	84.0	Documentary	0.600	2013-09-24	1.6	4	Fox	95000000.0	81562942.0	1.65720
19	84.0	Documentary	14.010	2010-11-12	6.4	1913	Fox	95000000.0	81562942.0	1.65720

In [214]:
```
query = ("""

SELECT  studio,
AVG(worldwide_gross) AS avg_gross_per_movie, popularity,runtime_minutes, movie
FROM WS_table
GROUP BY studio
ORDER BY avg_gross_per_movie DESC
LIMIT 10

""")

WS_df = pd.read_sql(query, conn)
```

```
Showing top average gross per movie by studio
```

In [215]:
```
sns.scatterplot(data=WS_df, x='studio', y = 'avg_gross_per_movie') ###hue='studio')
```

Out[215]: <AxesSubplot:xlabel='studio', ylabel='avg_gross_per_movie'>



In [216]:
```
query = ("""

SELECT DISTINCT(genres), popularity
FROM g_table
GROUP BY genres
HAVING popularity > 20
ORDER BY popularity DESC
LIMIT 25



""")

gen_df =pd.read_sql(query, conn)
```

In [671]:
```
query = ("""

SELECT *
FROM g_table
WHERE movie = 'Split'


""")

pd.read_sql(query, conn)
```

Out[671]:

	genres	movie_id	primary_title	original_title	start_year	runtime_minutes	genres:1	original_language	popularity	relea
0	Comedy,Romance,Sport	tt2660118	Split	Split	2016	90.0	Comedy,Romance,Sport	en	0.906	20
1	Comedy,Romance,Sport	tt2660118	Split	Split	2016	90.0	Comedy,Romance,Sport	en	2.029	20
2	Comedy,Romance,Sport	tt2660118	Split	Split	2016	90.0	Comedy,Romance,Sport	en	2.454	20
3	Comedy,Romance,Sport	tt2660118	Split	Split	2016	90.0	Comedy,Romance,Sport	en	25.783	20
4	Drama,Fantasy	tt3315656	Split	Split	2016	127.0	Drama,Fantasy	en	0.906	20
5	Drama,Fantasy	tt3315656	Split	Split	2016	127.0	Drama,Fantasy	en	2.029	20
6	Drama,Fantasy	tt3315656	Split	Split	2016	127.0	Drama,Fantasy	en	2.454	20
7	Drama,Fantasy	tt3315656	Split	Split	2016	127.0	Drama,Fantasy	en	25.783	20
8	Crime	tt3604256	Split	Split	2016	NaN	Crime	en	0.906	20
9	Crime	tt3604256	Split	Split	2016	NaN	Crime	en	2.029	20
10	Crime	tt3604256	Split	Split	2016	NaN	Crime	en	2.454	20
11	Crime	tt3604256	Split	Split	2016	NaN	Crime	en	25.783	20
12	Horror,Thriller	tt4972582	Split	Split	2016	117.0	Horror,Thriller	en	0.906	20
13	Horror,Thriller	tt4972582	Split	Split	2016	117.0	Horror,Thriller	en	2.029	20
14	Horror,Thriller	tt4972582	Split	Split	2016	117.0	Horror,Thriller	en	2.454	20
15	Horror,Thriller	tt4972582	Split	Split	2016	117.0	Horror,Thriller	en	25.783	20
16	Drama	tt5495666	Split	Split	2016	80.0	Drama	en	0.906	20
17	Drama	tt5495666	Split	Split	2016	80.0	Drama	en	2.029	20
18	Drama	tt5495666	Split	Split	2016	80.0	Drama	en	2.454	20
19	Drama	tt5495666	Split	Split	2016	80.0	Drama	en	25.783	20
20	Action,Drama,Sport	tt6147768	Split	Split	2016	123.0	Action,Drama,Sport	en	0.906	20
21	Action,Drama,Sport	tt6147768	Split	Split	2016	123.0	Action,Drama,Sport	en	2.029	20
22	Action,Drama,Sport	tt6147768	Split	Split	2016	123.0	Action,Drama,Sport	en	2.454	20
23	Action,Drama,Sport	tt6147768	Split	Split	2016	123.0	Action,Drama,Sport	en	25.783	20

In [219]:
```
query = ("""

SELECT *
FROM principals



""")

pd.read_sql(query, conn)
```

Out[219]:

	movie_id	ordering	person_id	category	job	characters
0	tt0111414	1	nm0246005	actor	None	["The Man"]
1	tt0111414	2	nm0398271	director	None	None
2	tt0111414	3	nm3739909	producer	producer	None
3	tt0323808	10	nm0059247	editor	None	None
4	tt0323808	1	nm3579312	actress	None	["Beth Boothby"]
...	...	...	...	...	...	...
1028181	tt9692684	1	nm0186469	actor	None	["Ebenezer Scrooge"]
1028182	tt9692684	2	nm4929530	self	None	["Herself","Regan"]
1028183	tt9692684	3	nm10441594	director	None	None
1028184	tt9692684	4	nm6009913	writer	writer	None
1028185	tt9692684	5	nm10441595	producer	producer	None

1028186 rows × 6 columns

In [220]:
```
query = ("""

SELECT p.movie_id, p.person_id, p.category, p.characters, g.movie, g.popularity, g.studio, g.vote_average, g.ge
FROM principals as p
JOIN g_table as g
USING(movie_id)
""")

PID_df = pd.read_sql(query, conn)
```

In [221]:
```
PID_df
```

0	tt0475290	nm0005683	cinematographer	None	Hail, Caesar!	12.312	Uni.	5.9	Comedy,Drama,Music
1	tt0475290	nm0000982	actor	["Eddie Mannix"]	Hail, Caesar!	12.312	Uni.	5.9	Comedy,Drama,Music
2	tt0475290	nm0000123	actor	["Baird Whitlock"]	Hail, Caesar!	12.312	Uni.	5.9	Comedy,Drama,Music
3	tt0475290	nm2403277	actor	["Hobie Doyle"]	Hail, Caesar!	12.312	Uni.	5.9	Comedy,Drama,Music
4	tt0475290	nm0000146	actor	["Laurence Laurentz"]	Hail, Caesar!	12.312	Uni.	5.9	Comedy,Drama,Music
...	...	...	...	...	...	...	...	...	...
18947	tt9151364	nm0000636	writer	None	The Tempest	6.300	Mira.	5.8	Drama
18948	tt9151364	nm1034863	producer	None	The Tempest	6.300	Mira.	5.8	Drama
18949	tt9151364	nm0745729	editor	None	The Tempest	6.300	Mira.	5.8	Drama
18950	tt9151364	nm0564805	actor	["Stephano"]	The Tempest	6.300	Mira.	5.8	Drama
18951	tt8991250	nm10118085	director	None	The Tree of Life	11.569	FoxS	6.6	Documentary

18952 rows × 9 columns

In [230]:
```
PG_df.to_sql("ROI_table_2", conn, index=False)
```

In [231]:
```
query = ("""

SELECT *
FROM ROI_table_2



""")

pd.read_sql(query,conn)
```

Out[231]:

ninutes	genres	popularity	release_date	vote_average	vote_count	studio	production_budget	domestic_gross	worldwide_gross	
124.0	Adventure,Drama,Romance	8.919	2012-12-21	5.6	518	IFC	25000000.0	720828.0	9.313302e+06	37.2
114.0	Adventure,Comedy,Drama	10.743	2013-12-25	7.1	4859	Fox	91000000.0	58236838.0	1.878612e+08	206.4
114.0	Action,Crime,Drama	19.373	2014-09-19	6.3	1685	Uni.	28000000.0	26017685.0	6.210859e+07	221.8
124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	215000000.0	652270625.0	1.648855e+09	766.9
119.0	Comedy,Drama	12.011	2011-10-27	5.7	652	FD	45000000.0	13109815.0	2.154473e+07	47.8
...	...	...	...	...	...	...	...	...	...	
NaN	Horror,Mystery,Thriller	11.927	2016-10-07	6.3	3479	Strand	45000000.0	75395035.0	1.742782e+08	387.2
90.0	Drama	11.250	2012-10-19	7.0	1426	Gold.	2000000.0	17061.0	1.706100e+04	0.8
NaN	Action,Drama	10.993	2015-12-18	5.9	922	Uni.	30000000.0	87044645.0	1.060307e+08	353.4
84.0	Documentary	0.600	2013-09-24	1.6	4	Fox	95000000.0	81562942.0	1.657209e+08	174.4
84.0	Documentary	14.010	2010-11-12	6.4	1913	Fox	95000000.0	81562942.0	1.657209e+08	174.4

In [232]:
```
query = ("""

SELECT p.person_id, p.category, p.characters, r.*
FROM PID_table as p
JOIN ROI_table_2 as r
USING(movie_id)


""")

pd.read_sql(query,conn)
```

Out[232]:

ime_minutes	genres	popularity	release_date	vote_average	vote_count	studio	production_budget	domestic_gross	worldwide_gross	
106.0	Comedy,Drama,Music	12.312	2016-02-05	5.9	2328	Uni.	22000000.0	30080225.0	64160680.0	291.6
106.0	Comedy,Drama,Music	12.312	2016-02-05	5.9	2328	Uni.	22000000.0	30080225.0	64160680.0	291.6
106.0	Comedy,Drama,Music	12.312	2016-02-05	5.9	2328	Uni.	22000000.0	30080225.0	64160680.0	291.6
106.0	Comedy,Drama,Music	12.312	2016-02-05	5.9	2328	Uni.	22000000.0	30080225.0	64160680.0	291.6
106.0	Comedy,Drama,Music	12.312	2016-02-05	5.9	2328	Uni.	22000000.0	30080225.0	64160680.0	291.6
...	...	...	...	...	...	...	...	...	...	
NaN	Drama	6.300	2010-12-10	5.8	52	Mira.	20000000.0	277943.0	277943.0	1.3
NaN	Drama	6.300	2010-12-10	5.8	52	Mira.	20000000.0	277943.0	277943.0	1.3
NaN	Drama	6.300	2010-12-10	5.8	52	Mira.	20000000.0	277943.0	277943.0	1.3
NaN	Drama	6.300	2010-12-10	5.8	52	Mira.	20000000.0	277943.0	277943.0	1.3
60.0	Documentary	11.569	2011-05-27	6.6	1730	FoxS	35000000.0	13305665.0	61721826.0	176.3

In [234]:
```
query = ("""

SELECT p.person_id, p.category, p.characters, r.*
FROM PID_table as p
JOIN ROI_table_2 as r
USING(movie_id)


""")

FINAL_df = pd.read_sql(query,conn)
```

In [235]:
```
FINAL_df.to_sql("FINAL_tb", conn, index=False)
```

In [236]:
```
query = ("""

SELECT *
FROM FINAL_tb



""")

pd.read_sql(query,conn)
```

Out[236]:

ime_minutes	genres	popularity	release_date	vote_average	vote_count	studio	production_budget	domestic_gross	worldwide_gross	
106.0	Comedy,Drama,Music	12.312	2016-02-05	5.9	2328	Uni.	22000000.0	30080225.0	64160680.0	291.6
106.0	Comedy,Drama,Music	12.312	2016-02-05	5.9	2328	Uni.	22000000.0	30080225.0	64160680.0	291.6
106.0	Comedy,Drama,Music	12.312	2016-02-05	5.9	2328	Uni.	22000000.0	30080225.0	64160680.0	291.6
106.0	Comedy,Drama,Music	12.312	2016-02-05	5.9	2328	Uni.	22000000.0	30080225.0	64160680.0	291.6
106.0	Comedy,Drama,Music	12.312	2016-02-05	5.9	2328	Uni.	22000000.0	30080225.0	64160680.0	291.6
...	...	...	...	...	...	...	...	...	...	
NaN	Drama	6.300	2010-12-10	5.8	52	Mira.	20000000.0	277943.0	277943.0	1.3
NaN	Drama	6.300	2010-12-10	5.8	52	Mira.	20000000.0	277943.0	277943.0	1.3
NaN	Drama	6.300	2010-12-10	5.8	52	Mira.	20000000.0	277943.0	277943.0	1.3
NaN	Drama	6.300	2010-12-10	5.8	52	Mira.	20000000.0	277943.0	277943.0	1.3
60.0	Documentary	11.569	2011-05-27	6.6	1730	FoxS	35000000.0	13305665.0	61721826.0	176.3

In [306]:
```
query = ("""

SELECT DISTINCT(f.movie), f.ROI, f.popularity, f.production_budget, f.worldwide_gross
FROM FINAL_tb f
JOIN (
    SELECT movie, MAX(popularity) AS max_popularity
    FROM FINAL_tb
    GROUP BY movie
    HAVING popularity < 70 AND ROI < 30000
) max_p ON f.movie = max_p.movie AND f.popularity = max_p.max_popularity

;



""")

GEN_df = pd.read_sql(query,conn)
```

```
Retrieves the top 100 movies with the highest ROI
```

In [288]:
```
query = ("""

SELECT DISTINCT(f.movie), f.ROI, f.popularity
FROM FINAL_tb f
JOIN (
    SELECT movie, MAX(popularity) AS max_popularity
    FROM FINAL_tb
    GROUP BY movie
) max_p ON f.movie = max_p.movie AND f.popularity = max_p.max_popularity
ORDER BY f.ROI DESC
LIMIT 100;



""")


ROI_P_df = pd.read_sql(query,conn)
```

In [290]:
```
query = ("""

SELECT DISTINCT(f.movie), f.ROI, f.popularity
FROM FINAL_tb f
JOIN (
    SELECT movie, MAX(popularity) AS max_popularity
    FROM FINAL_tb
    GROUP BY movie
    HAVING movie != 'The Gallows'
) max_p ON f.movie = max_p.movie AND f.popularity = max_p.max_popularity
ORDER BY f.popularity DESC
LIMIT 100;



""")


pd.read_sql(query,conn)
```

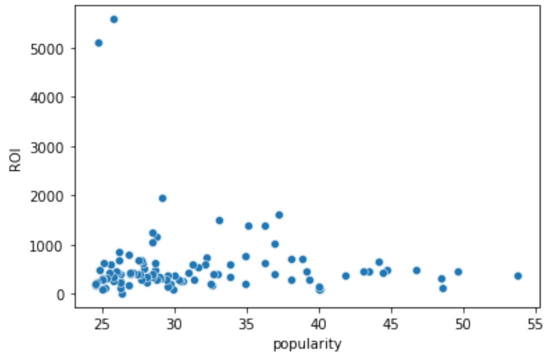
0	Avengers: Infinity War	682.711400	80.773
1	John Wick	254.116670	78.123
2	The Hobbit: The Battle of the Five Armies	378.231048	53.783
3	Guardians of the Galaxy	453.451480	49.606
4	Blade Runner 2049	140.193194	48.571
...	...	...	...
95	Prometheus	321.958612	24.980
96	Tomb Raider	303.863890	24.968
97	Inside Out	488.134853	24.797
98	Get Out	5107.359020	24.739
99	Alien: Covenant	245.898193	24.651

100 rows × 3 columns

In [292]:
```
query = ("""

SELECT DISTINCT(f.movie), f.ROI, f.popularity
FROM FINAL_tb f
JOIN (
    SELECT movie, MAX(popularity) AS max_popularity
    FROM FINAL_tb
    GROUP BY movie
    HAVING popularity < 70
) max_p ON f.movie = max_p.movie AND f.popularity = max_p.max_popularity
ORDER BY f.popularity DESC
LIMIT 100;



""")


POP_R_df = pd.read_sql(query,conn)
```

In [293]:
```
sns.scatterplot(data=POP_R_df, x = 'popularity', y = 'ROI')
```

Out[293]: <AxesSubplot:xlabel='popularity', ylabel='ROI'>



In [294]:
```
query = ("""

SELECT DISTINCT(f.movie), f.ROI, f.popularity
FROM FINAL_tb f
JOIN (
    SELECT movie, MAX(popularity) AS max_popularity
    FROM FINAL_tb
    GROUP BY movie
    HAVING popularity > 10 AND popularity < 30
) max_p ON f.movie = max_p.movie AND f.popularity = max_p.max_popularity
ORDER BY f.ROI DESC
LIMIT 100;



""")


pd.read_sql(query,conn)
```

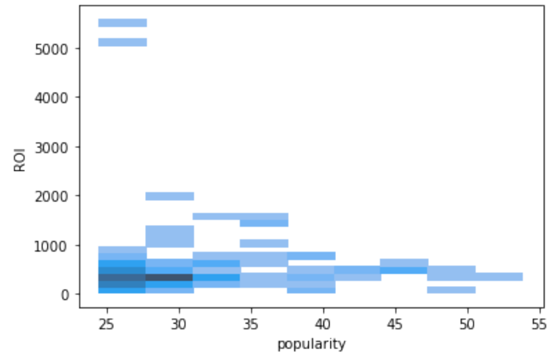
Out[294]:

	movie	ROI	popularity
0	Insidious	6658.059067	16.197
1	Split	5579.296120	25.783
2	Get Out	5107.359020	24.739
3	Chernobyl Diaries	4241.172100	14.658
4	Annabelle	3951.737231	13.989
...	...	...	...
95	Before Midnight	775.064333	11.765
96	The Lazarus Effect	767.186200	10.157
97	Jurassic World	766.909239	20.709
98	Don Jon	750.337800	12.780
99	About Time	744.243150	14.133

100 rows × 3 columns

In [295]:
```
sns.histplot(data=POP_R_df, x = 'popularity', y = 'ROI')
```

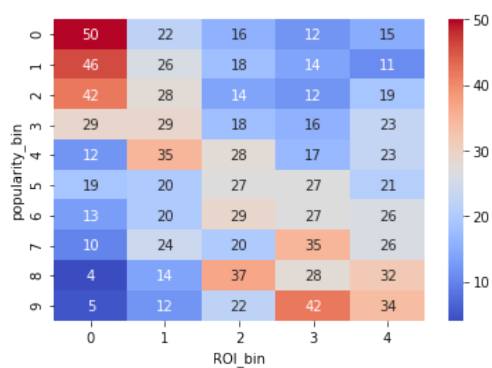
Out[295]: <AxesSubplot:xlabel='popularity', ylabel='ROI'>




```
Created a heatmap broken down into 5 ROI bins and 10 popualirty bins
```

In [318]:
```
GEN_df['popularity_bin'] = pd.qcut(GEN_df['popularity'], q=10, labels=False, duplicates='drop')
GEN_df['ROI_bin'] = pd.qcut(GEN_df['ROI'], q=5, labels=False, duplicates='drop')

heatmap_data = GEN_df.groupby(['popularity_bin', 'ROI_bin']).size().unstack()

sns.heatmap(heatmap_data, cmap="coolwarm", annot=True)
```

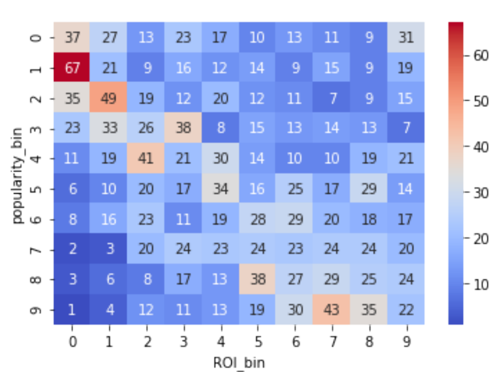
Out[318]: <AxesSubplot:xlabel='ROI_bin', ylabel='popularity_bin'>



In [ ]:
```
Created a heatmap broken down into tenths
```

In [328]:
```
PG_df['popularity_bin'] = pd.qcut(PG_df['popularity'], q=10, labels=False, duplicates='drop')
PG_df['ROI_bin'] = pd.qcut(PG_df['ROI'], q=10, labels=False, duplicates='drop')

heatmap_data_2 = PG_df.groupby(['popularity_bin', 'ROI_bin']).size().unstack()

sns.heatmap(heatmap_data_2, cmap="coolwarm", annot=True)
print(GEN_df['production_budget'].nunique())
```

5



In [516]:
```
PG_df
```

Out[516]:

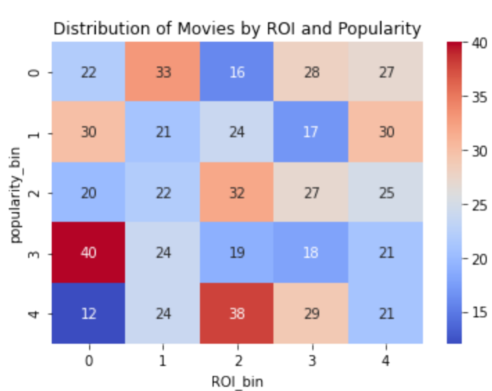
	movie	movie_id	start_year	runtime_minutes	genres	popularity	release_date	vote_average	vote_count	studio	produc
0	On the Road	tt0337692	2012	124.0	Adventure,Drama,Romance	8.919	2012-12-21	5.6	518	IFC	
1	The Secret Life of Walter Mitty	tt0359950	2013	114.0	Adventure,Comedy,Drama	10.743	2013-12-25	7.1	4859	Fox	
2	A Walk Among the Tombstones	tt0365907	2014	114.0	Action,Crime,Drama	19.373	2014-09-19	6.3	1685	Uni.	
3	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	2
4	The Rum Diary	tt0376136	2011	119.0	Comedy,Drama	12.011	2011-10-27	5.7	652	FD	
...	...	...	...	...	...	...	...	...	...	...	
1896	The Girl on the Train	tt9799088	2018	NaN	Horror,Mystery,Thriller	11.927	2016-10-07	6.3	3479	Strand	
1897	The First Time	tt9827712	2018	90.0	Drama	11.250	2012-10-19	7.0	1426	Gold.	
1898	Sisters	tt9851050	2019	NaN	Action,Drama	10.993	2015-12-18	5.9	922	Uni.	
1899	Unstoppable	tt9906218	2019	84.0	Documentary	0.600	2013-09-24	1.6	4	Fox	
1900	Unstoppable	tt9906218	2019	84.0	Documentary	14.010	2010-11-12	6.4	1913	Fox	

1901 rows × 19 columns

```
Created a heatmap broken down of 5 bins for each catageory only looking at data the top half of popularity ROI
```

In [347]:
```
roi_median = PG_df['ROI'].median()
popularity_median = PG_df['popularity'].median()

filtered_df = PG_df.loc[(PG_df['ROI'] > roi_median) & (PG_df['popularity'] > popularity_median)].copy()

filtered_df['popularity_bin'] = pd.qcut(filtered_df['popularity'], q=5, labels=False, duplicates='drop')
filtered_df['ROI_bin'] = pd.qcut(filtered_df['ROI'], q=5, labels=False, duplicates='drop')


heatmap_data_2 = filtered_df.groupby(['popularity_bin', 'ROI_bin']).size().unstack()

plt.xlabel("ROI Bins)")  # Change X-axis label
plt.ylabel("Popularity Bins)")  # Change Y-axis label
plt.title("Distribution of Movies by ROI and Popularity")  # Change Title


sns.heatmap(heatmap_data_2, cmap="coolwarm", annot=True, )
```

Out[347]:
```
<AxesSubplot:title={'center':'Distribution of Movies by ROI and Popularity'}, xlabel='ROI_bin', ylabel='popula
rity_bin'>
```



In [517]:
```
filtered_df
```

Out[517]:

	movie	movie_id	start_year	runtime_minutes	genres	popularity	release_date	vote_average	vote_count	studio	product
3	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	2
6	Tangled	tt0398286	2010	100.0	Adventure,Animation,Comedy	21.511	2010-11-24	7.5	6407	BV	2
11	Real Steel	tt0433035	2011	127.0	Action,Drama,Family	14.811	2011-10-06	6.8	4566	BV	1
13	Toy Story 3	tt0435761	2010	103.0	Adventure,Animation,Comedy	24.445	2010-06-17	7.7	8340	BV	2
14	Lincoln	tt0443272	2012	150.0	Biography,Drama,History	12.693	2012-11-16	6.8	2261	BV	
...	...	...	...	...	...	...	...	...	...	...	
1887	Neighbors	tt9392532	2018	90.0	Comedy,Drama	14.979	2014-05-09	6.2	4536	Uni.	
1892	Neighbors	tt9702034	2012	NaN	Drama	14.979	2014-05-09	6.2	4536	Uni.	
1893	Into the Woods	tt9703646	2018	NaN	Fantasy,Horror	13.726	2014-12-25	5.7	2880	BV	
1896	The Girl on the Train	tt9799088	2018	NaN	Horror,Mystery,Thriller	11.927	2016-10-07	6.3	3479	Strand	
1898	Sisters	tt9851050	2019	NaN	Action,Drama	10.993	2015-12-18	5.9	922	Uni.	

620 rows × 18 columns

```
Created a data frame looking at the top 70% of ROI and 80% of popularity
```

In [336]:
```
roi_40th = filtered_df['ROI'].quantile(0.4)
popularity_60th = filtered_df['popularity'].quantile(0.6)

filtered_df_2 = filtered_df.loc[(filtered_df['ROI'] >= roi_40th) & (filtered_df['popularity'] >= popularity_60t
filtered_df_2
```

Out[336]:

	movie	movie_id	start_year	runtime_minutes	genres	popularity	release_date	vote_average	vote_count	studio	p
3	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	
13	Toy Story 3	tt0435761	2010	103.0	Adventure,Animation,Comedy	24.445	2010-06-17	7.7	8340	BV	
18	Wonder Woman	tt0451279	2017	141.0	Action,Adventure,Fantasy	31.618	2017-06-02	7.3	12566	WB	
33	Ant-Man	tt0478970	2015	117.0	Action,Adventure,Comedy	32.715	2015-07-17	7.1	11949	BV	
65	Interstellar	tt0816692	2014	169.0	Adventure,Drama,Sci-Fi	28.440	2014-11-05	8.2	18597	Par.	
...	...	...	...	...	...	...	...	...	...	...	
1781	A Simple Favor	tt7040874	2018	117.0	Comedy,Crime,Drama	21.121	2018-09-14	6.6	1756	LGF	
1792	Arrival	tt7325124	2012	NaN	Documentary	25.442	2016-11-11	7.4	10387	Par.	
1795	BlacKkKlansman	tt7349662	2018	135.0	Biography,Crime,Drama	25.101	2018-07-30	7.6	3138	Focus	
1821	Hereditary	tt7784604	2018	127.0	Drama,Horror,Mystery	26.185	2018-06-08	7.0	2491	A24	
1840	Inside Out	tt8269544	2018	NaN	None	24.797	2015-06-19	8.0	12691	BV	

149 rows × 18 columns

```
Created a data frame looking at the top 60% of ROI and 70% of popularity
```

In [514]:
```
roi_60th = filtered_df['ROI'].quantile(0.1)
popularity_70th = filtered_df['popularity'].quantile(0.2)

filtered_df_exp = filtered_df.loc[(GEN_df['ROI'] >= roi_60th) & (filtered_df['popularity'] >= popularity_70th)]
filtered_df_exp
```

Out[514]:

	movie	movie_id	start_year	runtime_minutes	genres	popularity	release_date	vote_average	vote_count	studio	producti
6	Tangled	tt0398286	2010	100.0	Adventure,Animation,Comedy	21.511	2010-11-24	7.5	6407	BV	26
11	Real Steel	tt0433035	2011	127.0	Action,Drama,Family	14.811	2011-10-06	6.8	4566	BV	11
18	Wonder Woman	tt0451279	2017	141.0	Action,Adventure,Fantasy	31.618	2017-06-02	7.3	12566	WB	15
21	The Equalizer	tt0455944	2014	132.0	Action,Crime,Thriller	28.942	2014-09-26	7.2	4989	Sony	5
25	Ex Machina	tt0470752	2014	108.0	Drama,Mystery,Sci-Fi	18.485	2015-04-10	7.6	8026	A24	1
...	...	...	...	...	...	...	...	...	...	...	
1128	Whiplash	tt2582802	2014	106.0	Drama,Music	28.784	2014-10-10	8.4	7908	SPC	
1133	Lights Out	tt2611518	2013	NaN	Drama	12.408	2016-07-22	6.3	2220	WB (NL)	
1135	Rio	tt2614250	2012	90.0	Documentary	14.695	2011-04-15	6.6	3730	Fox	9
1138	Ted 2	tt2637276	2015	115.0	Comedy	17.684	2015-06-26	6.1	4227	Uni.	6
1144	Split	tt2660118	2016	90.0	Comedy,Romance,Sport	25.783	2016-09-26	7.2	10375	Uni.	

155 rows × 18 columns

```
Revenue from streaming services is not included in domestic and global gross revenue.
```

In [348]:
```
filtered_df_2.to_sql('FLT_table', conn, index=False)
```

In [469]:
```
filtered_df_2.to_csv('70X80_percentile.csv')
```

In [358]:
```
query = ("""

SELECT COUNT(studio), studio
FROM FLT_table
GROUP BY studio
ORDER BY COUNT(studio) DESC

""")
pd.read_sql(query, conn)
```

Out[358]:

	COUNT(studio)	studio
0	33	BV
1	25	Uni.
2	21	Fox
3	13	WB
4	11	Par.
5	7	Wein.
6	7	LGF
7	6	WB (NL)
8	5	Sony
9	4	FoxS
10	3	Rela.
11	3	BH Tilt
12	2	P/DW
13	1	TriS
14	1	Sum.
15	1	SPC
16	1	SGem
17	1	RTWC
18	1	ORF
19	1	MGM
20	1	Focus
21	1	A24

In [470]:
```
query = ("""

SELECT *
FROM FLT_table


""")
pd.read_sql(query, conn)
```

Out[470]:

	movie	movie_id	start_year	runtime_minutes	genres	popularity	release_date	vote_average	vote_count	studio	pr
0	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	
1	Toy Story 3	tt0435761	2010	103.0	Adventure,Animation,Comedy	24.445	2010-06-17	7.7	8340	BV	
2	Wonder Woman	tt0451279	2017	141.0	Action,Adventure,Fantasy	31.618	2017-06-02	7.3	12566	WB	
3	Ant-Man	tt0478970	2015	117.0	Action,Adventure,Comedy	32.715	2015-07-17	7.1	11949	BV	
4	Interstellar	tt0816692	2014	169.0	Adventure,Drama,Sci-Fi	28.440	2014-11-05	8.2	18597	Par.	
...	...	...	...	...	...	...	...	...	...	...	
144	A Simple Favor	tt7040874	2018	117.0	Comedy,Crime,Drama	21.121	2018-09-14	6.6	1756	LGF	
145	Arrival	tt7325124	2012	NaN	Documentary	25.442	2016-11-11	7.4	10387	Par.	
146	BlacKkKlansman	tt7349662	2018	135.0	Biography,Crime,Drama	25.101	2018-07-30	7.6	3138	Focus	
147	Hereditary	tt7784604	2018	127.0	Drama,Horror,Mystery	26.185	2018-06-08	7.0	2491	A24	
148	Inside Out	tt8269544	2018	NaN	None	24.797	2015-06-19	8.0	12691	BV	

149 rows × 18 columns

In [360]:
```
query = ("""

SELECT DISTINCT(movie)
FROM FLT_table
WHERE studio = 'BV'

""")
pd.read_sql(query, conn)
```

Out[360]:

	movie
0	Toy Story 3
1	Ant-Man
2	Coco
3	Doctor Strange
4	Iron Man 3
5	Frozen
6	Monsters University
7	Maleficent
8	Inside Out
9	Black Panther
10	Captain America: The Winter Soldier
11	Thor: The Dark World
12	Guardians of the Galaxy
13	Big Hero 6
14	Avengers: Age of Ultron
15	Zootopia
16	Captain America: Civil War
17	Thor: Ragnarok
18	Incredibles 2
19	Rogue One: A Star Wars Story
20	Avengers: Infinity War
21	Ant-Man and the Wasp

In [361]:
```
query = ("""

SELECT DISTINCT(movie)
FROM FLT_table
WHERE studio = 'WB'

""")
pd.read_sql(query, conn)
```

Out[361]:

	movie
0	Wonder Woman
1	The Dark Knight Rises
2	Inception
3	The Hangover Part II
4	Gravity
5	Aquaman
6	Sherlock Holmes: A Game of Shadows
7	Ready Player One
8	American Sniper
9	Crazy Rich Asians
10	Sully

In [364]:
```
query = ("""

SELECT f.* , p.person_id, p.category, p.job, p.characters
FROM FLT_table as f
JOIN principals as p
USING(movie_id)

""")
PRIN_df =pd.read_sql(query, conn)
```

In [365]:
```
PRIN_df.to_sql('PRIN_table', conn, index=False)
```

In [367]:
```
query = ("""
SELECT *
FROM PRIN_table

""")
pd.read_sql(query, conn)
```

Out[367]:

	movie	movie_id	start_year	runtime_minutes	genres	popularity	release_date	vote_average	vote_count	studio	...	worldwide_gross
0	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	...	1.648855e+09
1	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	...	1.648855e+09
2	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	...	1.648855e+09
3	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	...	1.648855e+09
4	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	...	1.648855e+09
...	...	...	...	...	...	...	...	...	...	...	...	...
97	Inside Out	tt8269544	2018	NaN	None	24.797	2015-06-19	8.0	12691	BV	...	8.542360e+08
98	Inside Out	tt8269544	2018	NaN	None	24.797	2015-06-19	8.0	12691	BV	...	8.542360e+08
99	Inside Out	tt8269544	2018	NaN	None	24.797	2015-06-19	8.0	12691	BV	...	8.542360e+08
00	Inside Out	tt8269544	2018	NaN	None	24.797	2015-06-19	8.0	12691	BV	...	8.542360e+08
01	Inside Out	tt8269544	2018	NaN	None	24.797	2015-06-19	8.0	12691	BV	...	8.542360e+08

2 rows × 22 columns

In [373]:
```
query = ("""
SELECT p.*, per.primary_name, per.primary_profession
FROM PRIN_table as p
JOIN persons as per
USING(person_id)

""")
DIR_df = pd.read_sql(query, conn)
```

In [374]:
```
DIR_df.to_sql('DIR_table', conn, index=False)
```

In [375]:
```
query = ("""
SELECT *
FROM DIR_table

""")
pd.read_sql(query, conn)
```

Out[375]:

	movie	movie_id	start_year	runtime_minutes	genres	popularity	release_date	vote_average	vote_count	studio	...	worldwide_gr
0	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	...	
1	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	...	
2	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	...	
3	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	...	
4	Jurassic World	tt0369610	2015	124.0	Action,Adventure,Sci-Fi	20.709	2015-06-12	6.6	14056	Uni.	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1397	Inside Out	tt8269544	2018	NaN	None	24.797	2015-06-19	8.0	12691	BV	...	
1398	Inside Out	tt8269544	2018	NaN	None	24.797	2015-06-19	8.0	12691	BV	...	
1399	Inside Out	tt8269544	2018	NaN	None	24.797	2015-06-19	8.0	12691	BV	...	
1400	Inside Out	tt8269544	2018	NaN	None	24.797	2015-06-19	8.0	12691	BV	...	
1401	Inside Out	tt8269544	2018	NaN	None	24.797	2015-06-19	8.0	12691	BV	...	

1402 rows × 24 columns

In [445]:
```
query = ("""
SELECT COUNT(primary_name), primary_name, popularity_bin
FROM DIR_table
GROUP BY primary_name
HAVING popularity_bin = 4
ORDER BY COUNT(primary_name) DESC

""")
pd.read_sql(query, conn)
```

Out[445]:

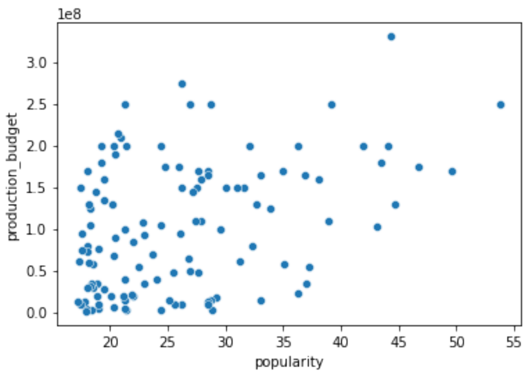
	COUNT(primary_name)	primary_name	popularity_bin
0	11	Jason Blum	4
1	10	Stan Lee	4
2	8	Jack Kirby	4
3	7	Robert Downey Jr.	4
4	5	Leonardo DiCaprio	4
...	...	...	...
627	1	Adam Sandler	4
628	1	Adam McKay	4
629	1	Adam Green	4
630	1	Adam Driver	4
631	1	Aarif Rahman	4

632 rows × 3 columns

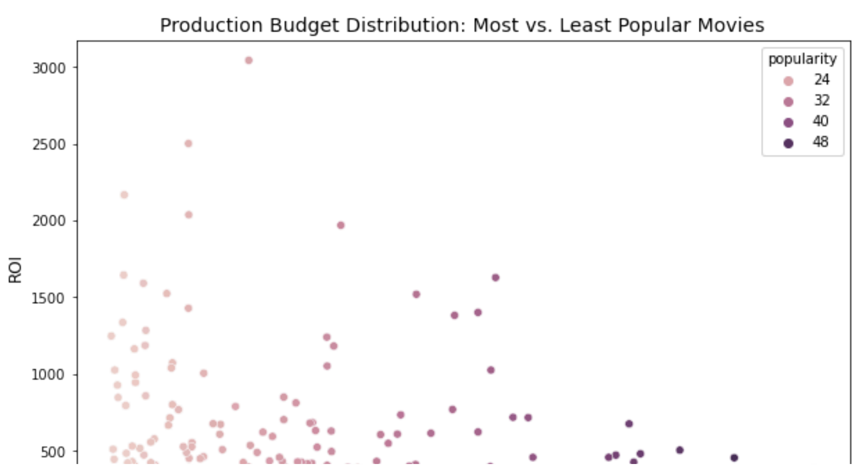
In [422]:
```
query = ("""
SELECT DISTINCT(movie), popularity, production_budget, ROI
FROM FLT_table
WHERE popularity < 70 AND ROI < 4000
ORDER BY popularity DESC


""")
slides_df=pd.read_sql(query, conn)
```

In [423]:
```
sns.scatterplot(data=slides_df, x = 'popularity', y = 'production_budget')
```

Out[423]: <AxesSubplot:xlabel='popularity', ylabel='production_budget'>



In [425]:
```


plt.figure(figsize=(10,6))
sns.scatterplot(x="popularity", y="ROI", hue = 'popularity', color = "blue",data=slides_df, )

plt.xlabel("Popularity Category", fontsize=12)
plt.ylabel("ROI", fontsize=12)
plt.title("Production Budget Distribution: Most vs. Least Popular Movies", fontsize=14)

plt.show()
```



In [434]:
```
print(PG_df.columns)  # Verify column names
```

```
Index(['movie', 'movie_id', 'start_year', 'runtime_minutes', 'genres',
       'popularity', 'release_date', 'vote_average', 'vote_count', 'studio',
       'production_budget', 'domestic_gross', 'worldwide_gross', 'ROI',
       'worldwide_gross_bin', 'production_budget_bin', 'popularity_bin',
       'ROI_bin'],
      dtype='object')
```

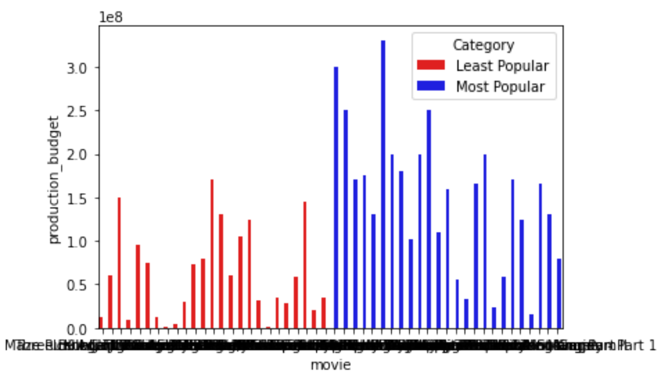
In [460]:
```
query = ("""
SELECT DISTINCT(movie), popularity, production_budget, ROI
FROM FLT_table
ORDER BY popularity ASC
LIMIT 25



""")
least_df=pd.read_sql(query, conn)
least_df["Category"] = "Least Popular"
```

In [461]:
```
query = ("""
SELECT DISTINCT(movie), popularity, production_budget, ROI
FROM FLT_table
ORDER BY popularity DESC
LIMIT 25


""")
most_df=pd.read_sql(query, conn)
most_df["Category"] = "Most Popular"
```

In [462]:
```
df_combined = pd.concat([least_df, most_df])
```

In [463]:
```
sns.barplot(x="movie", y="production_budget", hue="Category", data=df_combined, palette=["red", "blue"])
```

Out[463]:
```
<AxesSubplot:xlabel='movie', ylabel='production_budget'>
```



In [465]:
```
df_combined.to_csv("Project_2_data.csv")
```

In [475]:
```
query = ("""

SELECT f.*, d.person_id, d.category, d.job, d.characters, d.primary_name, d.primary_profession
FROM FLT_table as f
JOIN DIR_table as d
USING (movie_id)


""")
TEMP_2_df=pd.read_sql(query, conn)
```

In [476]:
```
TEMP_2_df.to_sql("TEMP_2_table", conn, index=False)
```

In [480]:
```
query = ("""

SELECT *
FROM TEMP_2_table


""")
pd.read_sql(query, conn)
```

Out[480]:

e_gross_bin	production_budget_bin	popularity_bin	ROI_bin	person_id	category	job	characters	primary_name	primary_
9	9	3	3	nm0000341	writer	based on the characters created by	None	Michael Crichton	writer,prod
9	9	3	3	nm0189777	producer	producer	None	Patrick Crowley	producer,assistant_director,producti
9	9	3	3	nm0339460	actress	None	["Karen"]	Judy Greer	actress,produce
9	9	3	3	nm0397171	actress	None	["Claire"]	Bryce Dallas Howard	actress,d
9	9	3	3	nm0415425	writer	screenplay by	None	Rick Jaffa	writer,pr
...	...	...	...	...	...	...	...	...	
9	9	4	2	nm4944310	producer	producer	None	David Neilson	miscellaneous,location_manageme
9	9	4	2	nm4976347	actress	None	["Annette"]	Keira Lucchesi	
9	9	4	2	nm6745863	actor	None	["Jimmy"]	Gerard Miller	
9	9	4	2	nm7069953	editor	None	None	Alexander Peacock	cinematographer,e
9	9	4	2	nm7604997	actor	None	["Colette"]	Erin Watson	

In [ ]:
```
Finding which people have the highest contributions to horror movies
```

In [495]:
```
query = ("""

SELECT COUNT(primary_name), primary_name, movie category, person_id
FROM TEMP_2_table
WHERE genres LIKE '%Horror%'
GROUP BY primary_name
ORDER BY COUNT(primary_name) DESC
LIMIT 25


""")
pd.read_sql(query, conn)
```

Out[495]:

	COUNT(primary_name)	primary_name	category	person_id
0	8	Jason Blum	The Purge	nm0089658
1	5	Michael Bay	The Purge	nm0000881
2	5	Brad Fuller	The Purge	nm0298181
3	5	Andrew Form	The Purge	nm0286320
4	4	James DeMonaco	The Purge	nm0218621
5	3	Sébastien K. Lemercier	The Purge	nm1085924
6	2	Toby Oliver	Get Out	nm0002947
7	2	Peter Safran	The Conjuring	nm0755911
8	2	James Wan	The Conjuring	nm1490123
9	2	Frank Grillo	The Purge: Anarchy	nm0342029
10	1	Zach Gilford	The Purge: Anarchy	nm1472917
11	1	Zach Fitzpatrick	The Revenant	nm6053523
12	1	Y'lan Noel	The First Purge	nm5002057
13	1	West Dylan Thordson	Split	nm3602603
14	1	Vera Farmiga	The Conjuring	nm0267812
15	1	Tony DeRosa-Grund	The Conjuring	nm0220533
16	1	Toni Collette	Hereditary	nm0001057
17	1	Todd Garner	The Possession of Hannah Grace	nm0307776
18	1	Tobin Bell	Jigsaw	nm0068551
19	1	Timur Bekmambetov	Unfriended: Dark Web	nm0067457
20	1	Steve Riven	Coco	nm9061888
21	1	Steve Harris	The First Purge	nm0004996
22	1	Stephen Susco	Unfriended: Dark Web	nm0839812
23	1	Stephanie Nogueras	Unfriended: Dark Web	nm5453200
24	1	Shay Mitchell	The Possession of Hannah Grace	nm3762213

In [508]:
```
query = ("""

SELECT *
FROM TEMP_2_table
WHERE genres LIKE '%Horror%'


""")
HRR_df=pd.read_sql(query, conn)
```

In [509]:
```
HRR_df.to_csv("HRR_table.csv")
```

In [499]:
```
query = ("""

SELECT primary_name, category, movie
FROM TEMP_2_table
WHERE genres LIKE '%Horror%' AND primary_name = 'Jason Blum'


""")
pd.read_sql(query, conn)
```

Out[499]:

	primary_name	category	movie
0	Jason Blum	producer	The Purge
1	Jason Blum	producer	The Purge: Anarchy
2	Jason Blum	producer	The Purge: Election Year
3	Jason Blum	producer	Unfriended: Dark Web
4	Jason Blum	producer	Split
5	Jason Blum	producer	Get Out
6	Jason Blum	producer	Happy Death Day
7	Jason Blum	producer	The First Purge

In [500]:
```
query = ("""

SELECT primary_name, category, movie
FROM TEMP_2_table
WHERE genres LIKE '%Horror%' AND primary_name = 'Michael Bay'


""")
pd.read_sql(query, conn)
```

Out[500]:

	primary_name	category	movie
0	Michael Bay	producer	The Purge
1	Michael Bay	producer	The Purge: Anarchy
2	Michael Bay	producer	The Purge: Election Year
3	Michael Bay	producer	The First Purge
4	Michael Bay	producer	A Quiet Place

In [504]:
```
query = ("""

SELECT DISTINCT t1.primary_name AS collaborator, t1.category, t1.movie,  COUNT(*) AS collaborations
FROM TEMP_2_table t1
JOIN TEMP_2_table t2
ON t1.movie = t2.movie
WHERE t2.primary_name = 'Michael Bay'
AND t1.primary_name <> 'Michael Bay'
GROUP BY t1.primary_name, t1.category
ORDER BY collaborations DESC
""")
pd.read_sql(query, conn)
```

Out[504]:

	collaborator	category	movie	collaborations
0	Andrew Form	producer	The Purge	5
1	Brad Fuller	producer	The Purge	5
2	Jason Blum	producer	The Purge	4
3	James DeMonaco	director	The Purge	3
4	Sébastien K. Lemercier	producer	The Purge	3
5	Don Murphy	producer	Transformers: Age of Extinction	2
6	Frank Grillo	actor	The Purge: Anarchy	2
7	Lorenzo di Bonaventura	producer	Transformers: Age of Extinction	2
8	Tom DeSanto	producer	Transformers: Age of Extinction	2
9	Adelaide Kane	actress	The Purge	1
10	Bryan Woods	writer	A Quiet Place	1
11	Carmen Ejogo	actress	The Purge: Anarchy	1
12	Christina Hodson	writer	Bumblebee	1
13	Ehren Kruger	writer	Transformers: Age of Extinction	1
14	Elizabeth Mitchell	actress	The Purge: Election Year	1
15	Emily Blunt	actress	A Quiet Place	1
16	Ethan Hawke	actor	The Purge	1
17	Gerard McMurray	director	The First Purge	1
18	Gijs Determeijer	producer	A Quiet Place	1
19	Gijs Kerbosch	producer	A Quiet Place	1
20	Hailee Steinfeld	actress	Bumblebee	1
21	Ian Bryce	producer	Transformers: Age of Extinction	1
22	Jack Reynor	actor	Transformers: Age of Extinction	1
23	James DeMonaco	writer	The First Purge	1
24	Jason Drucker	actor	Bumblebee	1
25	John Cena	actor	Bumblebee	1
26	John Krasinski	actor	A Quiet Place	1
27	Joivan Wade	actor	The First Purge	1
28	Jorge Lendeborg Jr.	actor	Bumblebee	1
29	Joseph Julian Soria	actor	The Purge: Election Year	1
30	Kiele Sanchez	actress	The Purge: Anarchy	1
31	Lena Headey	actress	The Purge	1
32	Lex Scott Davis	actress	The First Purge	1
33	Marco Beltrami	composer	A Quiet Place	1
34	Mark Wahlberg	actor	Transformers: Age of Extinction	1
35	Max Burkholder	actor	The Purge	1
36	Millicent Simmonds	actress	A Quiet Place	1
37	Mykelti Williamson	actor	The Purge: Election Year	1
38	Nicola Peltz	actress	Transformers: Age of Extinction	1
39	Noah Jupe	actor	A Quiet Place	1
40	Olivia van Leeuwen	producer	A Quiet Place	1
41	Roel Oude Nijhuis	producer	A Quiet Place	1
42	Saskia Kievits	editor	A Quiet Place	1
43	Scott Beck	writer	A Quiet Place	1
44	Sjoerd Oostrik	director	A Quiet Place	1
45	Stanley Tucci	actor	Transformers: Age of Extinction	1
46	Steve Harris	actor	The First Purge	1
47	Tim Kerbosch	cinematographer	A Quiet Place	1
48	Travis Knight	director	Bumblebee	1
49	Y'lan Noel	actor	The First Purge	1

	collaborator	category	movie	collaborations
50	Zach Gilford	actor	The Purge: Anarchy	1

In [505]:
```
query = ("""

SELECT DISTINCT t1.primary_name AS collaborator, t1.category, t1.movie,  COUNT(*) AS collaborations
FROM TEMP_2_table t1
JOIN TEMP_2_table t2
ON t1.movie = t2.movie
WHERE t2.primary_name = 'Jason Blum'
AND t1.primary_name <> 'Jason Blum'
GROUP BY t1.primary_name, t1.category
ORDER BY collaborations DESC
""")
pd.read_sql(query, conn)
```

Out[505]:

	collaborator	category	movie	collaborations
0	Andrew Form	producer	The Purge	4
1	Brad Fuller	producer	The Purge	4
2	Michael Bay	producer	The Purge	4
3	James DeMonaco	director	The Purge	3
4	Sébastien K. Lemercier	producer	The Purge	3
...	...	...	...	...
122	Virginie Dubois	producer	Split	1
123	West Dylan Thordson	composer	Split	1
124	Whitney Lafleur	actor	Split	1
125	Y'lan Noel	actor	The First Purge	1
126	Zach Gilford	actor	The Purge: Anarchy	1

127 rows × 4 columns

In [506]:
```
query = ("""

SELECT DISTINCT t1.primary_name AS collaborator, t1.category, t1.movie,  COUNT(*) AS collaborations
FROM TEMP_2_table t1
JOIN TEMP_2_table t2
ON t1.movie = t2.movie
WHERE t2.primary_name = 'Sébastien K. Lemercier'
AND t1.primary_name <> 'Sébastien K. Lemercier'
GROUP BY t1.primary_name, t1.category
ORDER BY collaborations DESC
""")
pd.read_sql(query, conn)
```

Out[506]:

	collaborator	category	movie	collaborations
0	Andrew Form	producer	The Purge	3
1	Brad Fuller	producer	The Purge	3
2	James DeMonaco	director	The Purge	3
3	Jason Blum	producer	The Purge	3
4	Michael Bay	producer	The Purge	3
5	Frank Grillo	actor	The Purge: Anarchy	2
6	Adelaide Kane	actress	The Purge	1
7	Carmen Ejogo	actress	The Purge: Anarchy	1
8	Elizabeth Mitchell	actress	The Purge: Election Year	1
9	Ethan Hawke	actor	The Purge	1
10	Joseph Julian Soria	actor	The Purge: Election Year	1
11	Kiele Sanchez	actress	The Purge: Anarchy	1
12	Lena Headey	actress	The Purge	1
13	Max Burkholder	actor	The Purge	1
14	Mykelti Williamson	actor	The Purge: Election Year	1
15	Zach Gilford	actor	The Purge: Anarchy	1

In [518]:
```
filtered_df.to_sql('Final_data_table', conn , index=False)
```

In [523]:
```
query = ("""

SELECT *
FROM Final_data_table
WHERE runtime_minutes > 100 and  genres LIKE '%Horror%'
ORDER BY production_budget


""")

pd.read_sql(query,conn)
```

Out[523]:

genres	popularity	release_date	vote_average	vote_count	studio	production_budget	domestic_gross	worldwide_gross	ROI	wo
Horror,Mystery,Thriller	16.197	2011-04-01	6.9	3582	FD	1500000.0	54009150.0	99870886.0	6658.059067	
Horror,Mystery,Thriller	13.117	2012-10-12	6.8	2935	LG/S	3000000.0	48086903.0	87727807.0	2924.260233	
Drama,Horror,Mystery	17.892	2016-03-11	6.9	4629	Par.	5000000.0	72082999.0	108286422.0	2165.728440	
Horror,Mystery	10.841	2014-04-03	6.4	1747	Rela.	5000000.0	27695246.0	44115496.0	882.309920	
Horror,Thriller	25.783	2016-09-26	7.2	10375	Uni.	5000000.0	138141585.0	278964806.0	5579.296120	
Horror,Mystery,Thriller	24.739	2017-02-24	7.5	8760	Uni.	5000000.0	176040665.0	255367951.0	5107.359020	
Action,Horror,Sci-Fi	28.424	2014-07-18	6.6	3754	Uni.	9000000.0	71562550.0	111534881.0	1239.276456	
Action,Horror,Sci-Fi	18.975	2016-07-01	6.3	2900	Uni.	10000000.0	79042440.0	118514727.0	1185.147270	
Horror,Mystery,Thriller	16.017	2018-01-05	6.1	1306	Uni.	10000000.0	67745330.0	167885588.0	1678.855880	
Drama,Horror,Mystery	26.185	2018-06-08	7.0	2491	A24	10000000.0	44069456.0	70133905.0	701.339050	
Horror,Mystery,Thriller	21.245	2017-08-11	6.5	3141	WB (NL)	15000000.0	102092201.0	305384865.0	2035.899100	
Horror,Thriller	12.246	2010-02-26	6.2	1036	Over.	19000000.0	39123589.0	56445534.0	297.081758	
Horror,Mystery,Thriller	18.886	2013-07-19	7.5	5912	WB (NL)	20000000.0	137400141.0	318000141.0	1590.000705	
Drama,Horror,Mystery	13.530	2017-02-03	4.9	1785	Par.	25000000.0	27793018.0	82917283.0	331.669132	
Horror,Thriller	13.966	2017-09-08	7.2	10931	WB (NL)	35000000.0	327481748.0	697457969.0	1992.737054	
Horror	13.476	2010-09-17	7.1	2386	WB	37000000.0	92186262.0	152566881.0	412.342922	
Horror,Mystery	16.082	2011-04-11	6.2	1610	W/Dim.	40000000.0	38180928.0	95989590.0	239.973975	
Horror,Sci-Fi,Thriller	24.651	2017-05-19	5.9	4971	Fox	97000000.0	74262031.0	238521247.0	245.898193	
Action,Horror,Sci-Fi	31.397	2018-08-10	5.9	2896	WB	178000000.0	145443742.0	529530715.0	297.489166	
ction,Adventure,Horror	14.582	2013-06-21	6.7	9132	Par.	190000000.0	202359711.0	531514650.0	279.744553	

In [526]:
```
query = ("""

SELECT *
FROM Final_data_table
JOIN principals as p
USING(movie_id)
WHERE runtime_minutes > 100 and  genres LIKE '%Horror%'



""")

Final_PRN_df = pd.read_sql(query,conn)
```

In [527]:
```
Final_PRN_df.to_sql("Final_PRN_table", conn, index=False)
```

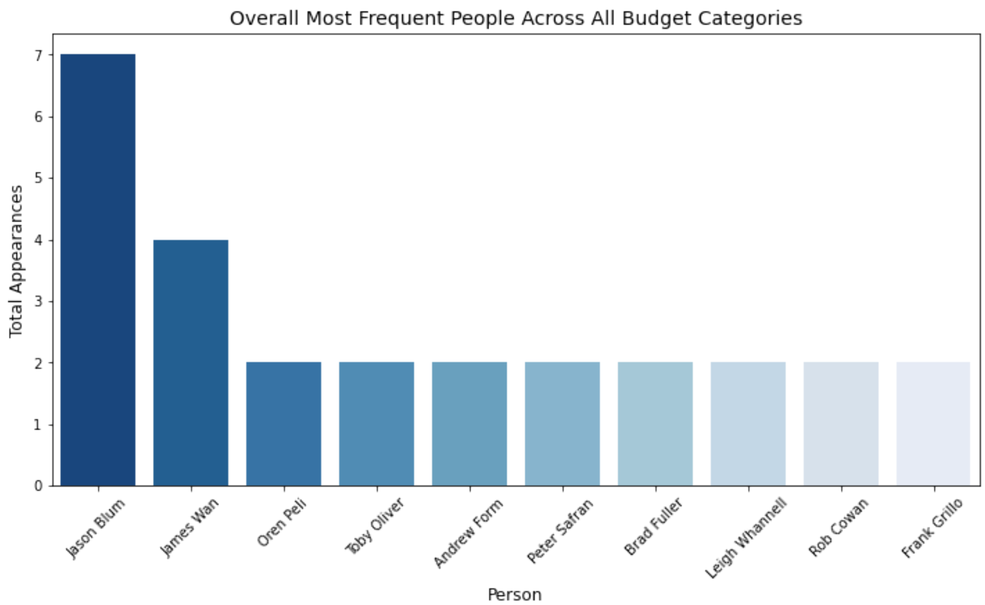
In [538]:
```
query = ("""

SELECT *
FROM Final_PRN_table
JOIN persons
USING (person_id)



""")

FINAL_df_1=pd.read_sql(query,conn)
```

In [539]:
```
print(FINAL_df_1.columns)
```

```
Index(['movie', 'movie_id', 'start_year', 'runtime_minutes', 'genres',
       'popularity', 'release_date', 'vote_average', 'vote_count', 'studio',
       'production_budget', 'domestic_gross', 'worldwide_gross', 'ROI',
       'worldwide_gross_bin', 'production_budget_bin', 'popularity_bin',
       'ROI_bin', 'ordering', 'person_id', 'category', 'job', 'characters',
       'primary_name', 'birth_year', 'death_year', 'primary_profession'],
      dtype='object')
```

In [679]:
```

low_threshold = FINAL_df_1["production_budget"].quantile(0.33)
high_threshold = FINAL_df_1["production_budget"].quantile(0.66)

FINAL_df_1["budget_category"] = pd.cut(
    FINAL_df_1["production_budget"],
    bins=[0, low_threshold, high_threshold, FINAL_df_1["production_budget"].max()],
    labels=["Low Budget", "Medium Budget", "High Budget"],
    include_lowest=True
)

top_people_per_category = FINAL_df_1.groupby(["budget_category", "primary_name"]).size().reset_index(name="coun


top_people_per_category = top_people_per_category.sort_values(["budget_category", "count"], ascending=[True, Fa
```

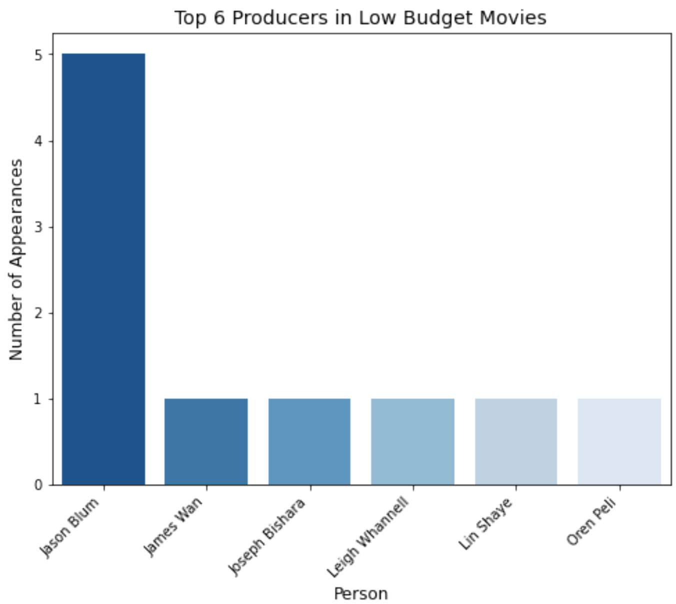
In [680]:
```
top_overall_people = FINAL_df_1["primary_name"].value_counts().reset_index()
top_overall_people.columns = ["primary_name", "count"]


plt.figure(figsize=(12,6))
sns.barplot(
    data=top_overall_people.head(10),  # Show only the top 10
    x="primary_name",
    y="count",
    palette="Blues_r"
)

plt.xlabel("Person", fontsize=12)
plt.ylabel("Total Appearances", fontsize=12)
plt.title("Overall Most Frequent People Across All Budget Categories", fontsize=14)
plt.xticks(rotation=45)
plt.show()
```

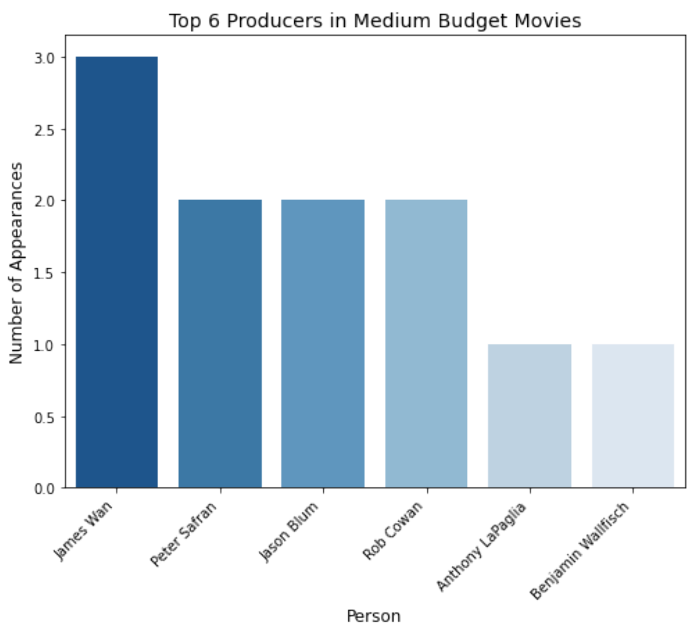


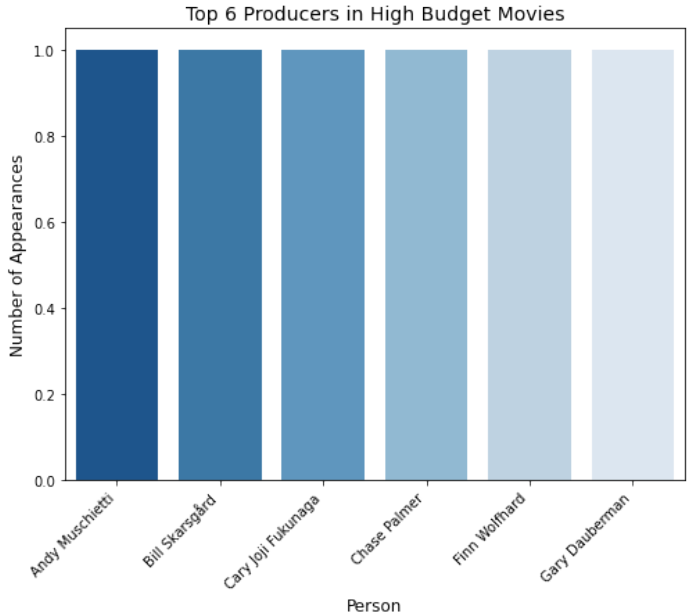
In [545]:
```
top_people_per_category = FINAL_df_1.groupby(["budget_category", "primary_name"]).size().reset_index(name="coun

top_people_per_category = top_people_per_category.sort_values(["budget_category", "count"], ascending=[True, Fa
```

In [682]:
```
top_people_per_category = (
    FINAL_df_1.groupby(["budget_category", "primary_name"])
    .agg(count=("primary_name", "size"), avg_ROI=("ROI", "mean"))  # Count appearances and get avg ROI
    .reset_index()
)

top_people_per_category = (
    top_people_per_category.sort_values(["budget_category", "count", "avg_ROI"], ascending=[True, False, False]
    .groupby("budget_category")
    .head(6)
)
```

In [683]:
```
for category in top_people_per_category["budget_category"].unique():
    plt.figure(figsize=(8,6))

    category_data = top_people_per_category[top_people_per_category["budget_category"] == category]

    sns.barplot(
        data=category_data,
        x="primary_name",
        y="count",
        palette="Blues_r"
    )

    plt.xlabel("Person", fontsize=12)
    plt.ylabel("Number of Appearances", fontsize=12)
    plt.title(f"Top 6 Producers in {category} Movies", fontsize=14)

    plt.xticks(rotation=45, ha='right')  # Rotate x labels for better readability

    plt.show()
```







In [558]:

```
top_people_per_category = (
    FINAL_df_1.groupby(["budget_category", "primary_name"])
    .agg(count=("primary_name", "size"), avg_ROI=("ROI", "mean"))
    .reset_index()
)


top_people_per_category = (
    top_people_per_category.sort_values(["budget_category", "avg_ROI"], ascending=[True, False])
    .groupby("budget_category")
    .head(10)
)
```

In [567]:

```
FINAL_df_1.to_csv("FINAL_data_1.csv")
```

In [568]:

```
FINAL_df_1.to_sql("FINAL_table_1", conn, index=False)
```

In [591]:

```
query = ("""

SELECT DISTINCT(movie), category, primary_name, ROI
FROM FINAL_table_1
WHERE budget_category = 'High Budget'AND category = 'producer'
ORDER BY ROI DESC


""")

FIN_df_4=pd.read_sql(query,conn)
```

In [592]:

```
FIN_df_4.to_csv("HBP.csv")
```

In [593]:
```
query = ("""

SELECT DISTINCT(movie), category, primary_name, ROI
FROM FINAL_table_1
WHERE budget_category = 'High Budget'AND category = 'producer'
ORDER BY ROI DESC


""")

pd.read_sql(query,conn)
```

Out[593]:

	movie	category	primary_name	ROI
0	It	producer	Seth Grahame-Smith	1992.737054
1	The Town	producer	Sean van Hastings	412.342922
2	Rings	producer	Laurie MacDonald	331.669132
3	The Meg	producer	Belle Avery	297.489166
4	Scream 4	producer	Iya Labunka	239.973975

In [612]:
```
query = ("""

SELECT DISTINCT(movie), category, primary_name, ROI, popularity
FROM FINAL_table_1
WHERE budget_category = 'High Budget'AND primary_name = 'Seth Grahame-Smith'
ORDER BY ROI DESC


""")

pd.read_sql(query,conn)
```

Out[612]:

	movie	category	primary_name	ROI	popularity
0	It	producer	Seth Grahame-Smith	1992.737054	13.966

In [614]:
```
query = ("""

SELECT AVG(ROI) AS avg_popularity
FROM FINAL_table_1
WHERE budget_category = 'High Budget';



""")

pd.read_sql(query,conn)
```

Out[614]:

	avg_popularity
0	548.679421

In [613]:
```
query = ("""

SELECT AVG(popularity) AS avg_popularity
FROM FINAL_table_1
WHERE budget_category = 'High Budget';



""")

pd.read_sql(query,conn)
```

Out[613]:

	avg_popularity
0	18.45391

In [ ]: